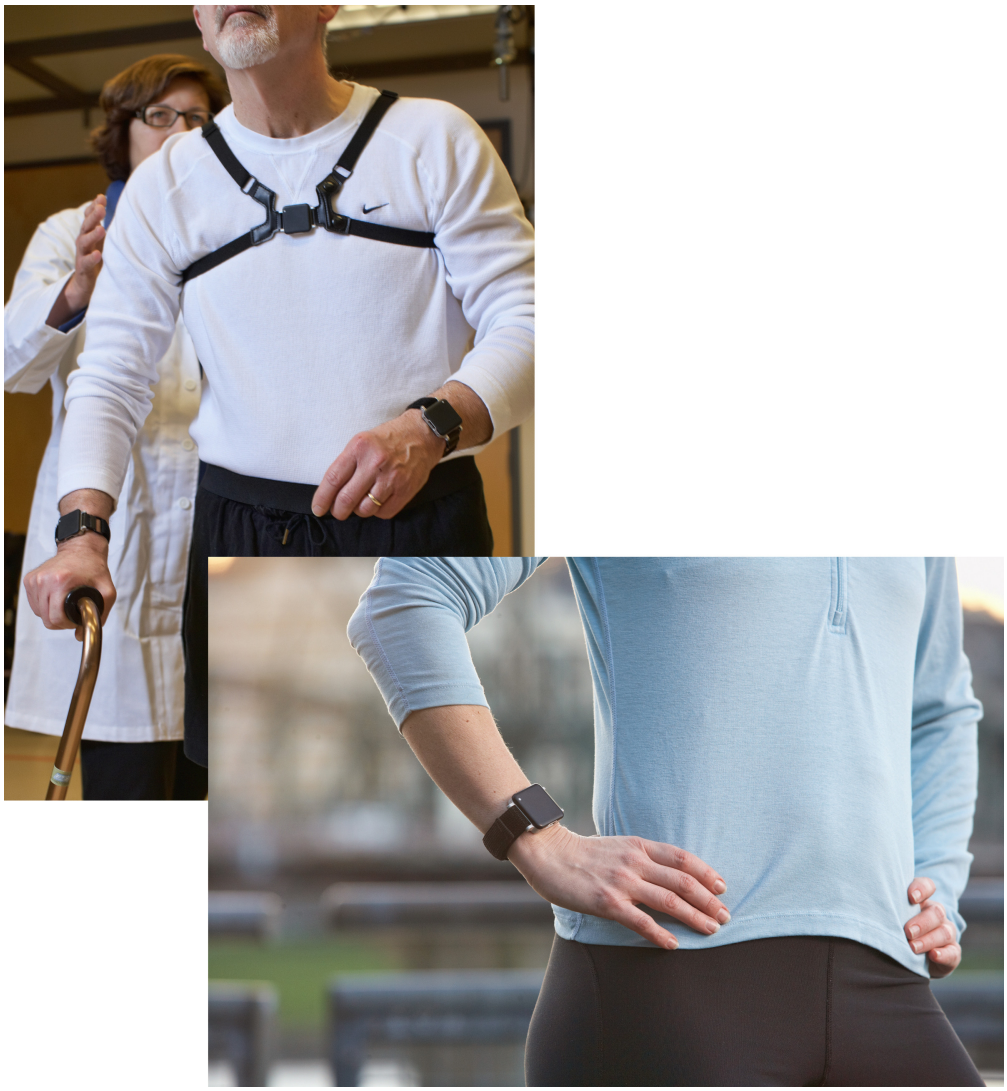
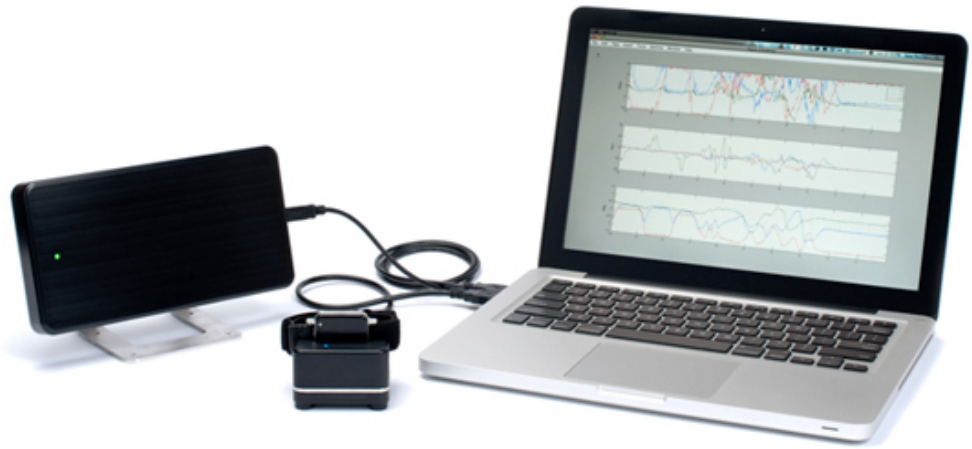


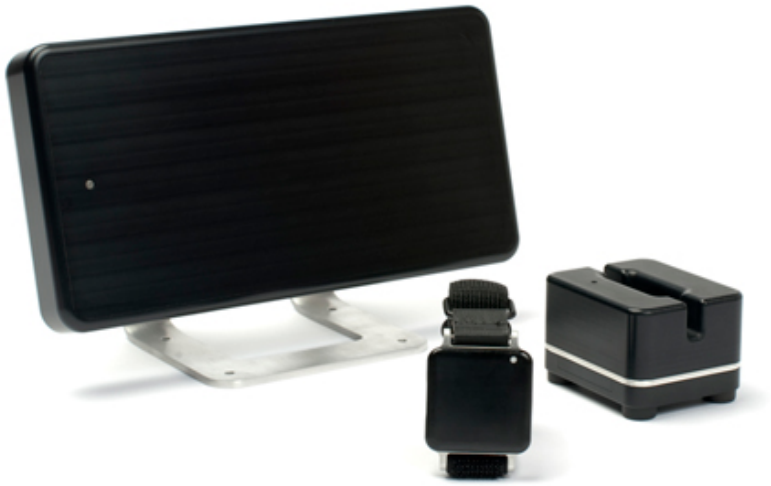
APDM SDK Developer Guide

Contents



1 Welcome

Welcome to the APDM movement monitoring system. The following documentation will guide you through understanding the architecture of the system and how to use it as a developer. The core SDK documentation will focus on the C language implementation utilizing a dynamically linked library. Other language bindings such as Java will be documented as well. For the C language or any language binding not provided it is assumed that the end user should know how to properly load the dynamically linked library and call its functions using the correct calling convention. Most of the documentation will use a cross platform approach with specific platform specific notes if needed. Example code will be provided for all included language bindings to assist in getting the user up and running in the shortest period of time possible.

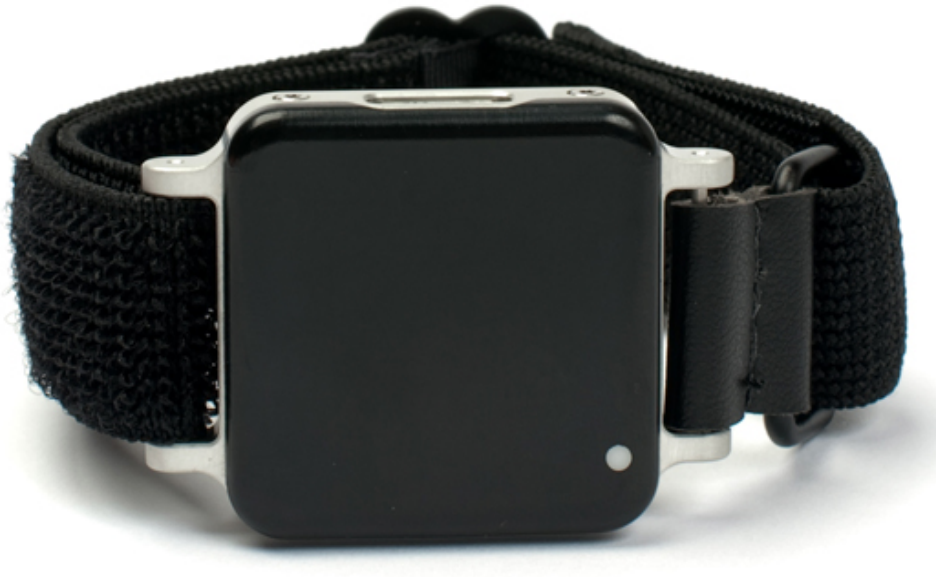


2 System Overview

The APDM movement monitoring system allows the user to record data from multiple monitors; each integrating a suite of sensors. The system can be configured in 3 recording modes allowing for a wide rage of applications. Some movement monitors are limited to a subset of these modes allowing for a lower cost solution. The modes of operation are synchronized streaming, synchronized logging, and low power logging. Regardless of the mode the movement monitor always will record data to its local memory card which can be downloaded for offline analysis.

2.1 Movement Monitors

Movement monitors are the key element of the system and combine a complement of sensors within a single package. Sensors include a 3 axis accelerometer, a 3 axis gyro, a 3 axis magnetometer, and a temperature sensor. The accelerometers can be configured in a high 6G mode, or a low 2G mode depending on the target application. There are a number of options for securing the monitors on subjects using a selection of straps.



The Opal movement monitor

2.1.1 The Opal

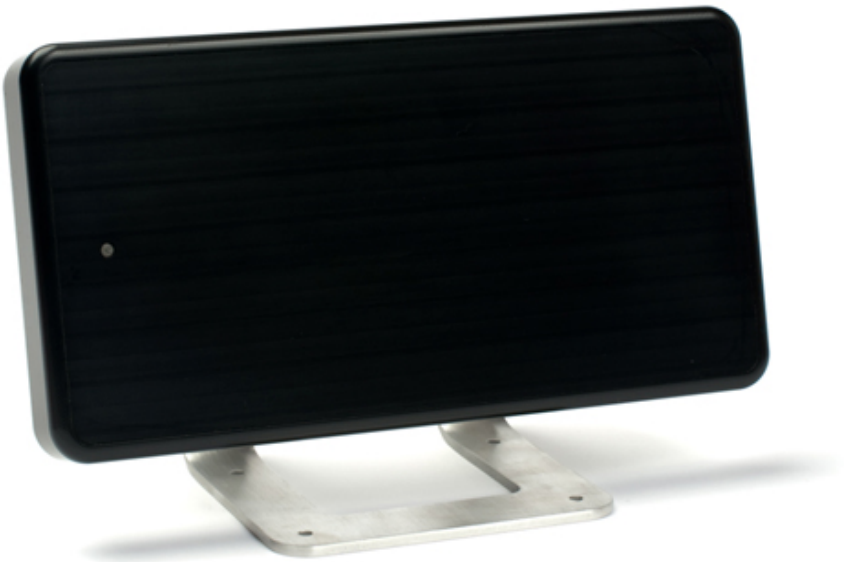
The Opal is APDM's full featured movement monitor allowing for use of all 3 modes of operation.

2.1.2 The Emerald

The Emerald is an option that allows for logging only without the ability to stream data in real time. This version allows for synchronized and low power logging giving the user the ability to do long term studies with subjects at home or in a clinical environment. It is recommended for users that require multiple movement monitors to be recording on a subject at one time.

2.1.3 The Sapphire

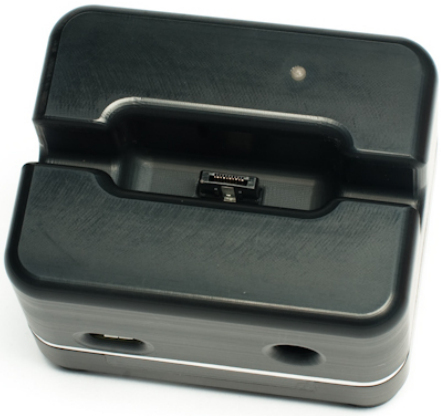
The Sapphire allows only for the low power logging mode. This version of the movement monitor has no wireless capabilities and may be the optimal choice for RF sensitive environments or where a single movement monitor is needed without synchronization.



The access point, for communicating wirelessly with your movement monitors

2.2 Access Point

The wireless access control point (access point for short) allows for wireless communication between the host computer and Opal movement monitors. A single access point can support up to 6 Opals.



The docking station, for charging, configuring, and downloading data from your movement monitors

2.3 Docking Station

The docking station is used to configure, charge, and download data from the movement monitors.

2.4 Recording Modes

Depending on the application of the movement monitor system one of the three available configuration options can be selected. Each will have different requirements so not all configurations may be available.

2.4.1 Synchronized Streaming

In the synchronized streaming mode, access points attached to a PC collect data transmitted wirelessly from the movement monitors. Each movement monitor is also saving all data to its on board memory for download later as a backup collection mechanism. This mode allows for near real time collection and processing of data from multiple synchronized movement monitors. Only the Opals can be used in this mode.

2.4.2 Synchronized Logging

Synchronized logging gives the user the ability to collect data with multiple movement monitors and tightly correlate the individual recordings during offline analysis. The synchronization is achieved through wireless communication between the monitors. In this mode, up to 32 monitors can be synchronized within a single “mesh”. Only Emeralds and Opals are able to use this mode.

2.4.3 Low Power Logging

All movement monitor products (Opals, Emeralds, and Sapphires) are able to operate in the low power logging mode. In this mode, the monitors’ wireless radios are disabled, decreasing the power required for operation. This enables the system to run for longer periods of time. Since the mode does not use any wireless synchronization, each movement monitor will collect data independently and potentially at slightly different rates due to clock drift. This mode is recommended when tight correlation of data between multiple movement monitors is not

needed.

2.5 Motion Studio

Motion Studio is the default software suite bundled with the APDM movement monitor system. It provides an easy way to get up and running collecting data with your movement monitors.

2.6 APDM Software Development Kit

The APDM Software Development Kit (SKD) provides a programming interface to configure and stream data from the movement monitors. In addition, it also provides functions for converting the raw data files found on the monitor's memory card into either a HDF5 (recommended) format or CSV. The SDK provides the same low level interface to the hardware that Motion Studio is built upon.

3 Hardware Driver Installation

3.1 Macintosh OSX (x32/x64)

- Copy the file found at your installation's "MotionStudio/drivers/libftd2xx.0.1.6.dylib" to your "/usr/local/lib" directory. You will need administration privileges to do so.

3.2 Windows XP (x32)

3.2.1 Update Registry

1. Double click on the "MotionStudio/drivers/apdm_usb_serial_number.reg" file and click through the resulting dialogs.
2. If double clicking on the file does not automatically import the registry entry:
   a) Click on the "Start" button.
   b) Click on "Run"
   c) Type "regedit" and press return.
   d) Select "File→Import..." and select the file at "MotionStudio/drivers/apdm_usb_serial_number.reg"

3.2.2 Access Point

1. Plug the access point into your computer.
2. The "Welcome to the Found New Hardware Wizard" dialog will popup
3. Select "No, not this time"
4. Click "Next"
5. Check "Install from a list or specific location"
6. Click "Next"
7. Uncheck "Search removable media"
8. Check "Include this location in the search"
9. Click the "Browse" button and navigate to your installation's "MotionStudio/drivers/apdm_accesspoint_drivers" folder
10. Click "Ok"
11. Click "Next"
12. If Windows prompts you about the driver not being signed, click 'Continue Anyway'
13. Click "Finish"

3.2.3 Docking Station

1. Copy the file “MotionStudio/drivers/ftd2xx.dll” to the folder located at “C:/Windows/system32”. You will need administrator priviledges to do so.
2. Attach a USB cable to the docking station. If more than one docking stations are chained together into a single unit, then external power has to be connected as well.
3. Plug the docking station into your computer.
4. The “Welcome to the Found New Hardware Wizard” dialog will popup
5. Select “No, not this time”
6. Click “Next”
7. Check “Install from a list or specific location” and click “Next”
8. Uncheck “Search removable media”
9. Check “Include this location in the search”
10. Click the “Browse” button and navigate to the “MotionStudio/drivers/apdm_docking_station_drivers” folder
11. Click “Ok” and Click “Next”
12. If Windows prompts you about the driver not being signed, click ’Continue Anyway’
13. Click “Finish”
14. This action may need to be repeated for each docking station installed. Docking stations that are plugged in but are not yet installed will blink blue.

3.3 Windows Vista (x32/x64)

3.3.1 Update Registry

1. Double click on the “MotionStudio/drivers/apdm_usb_serial_number.reg” file and click through the resulting dialogs.
2. If double clicking on the file does not automatically import the registry entry:
   a) Click on the “Start” button.
   b) Click on “Run”
   c) Type “regedit” and press return.
   d) Select “File→Import...” and select the file at “MotionStudio/drivers/apdm_usb_serial_number.reg”

3.3.2 Access Point

1. Plug the access point into your computer.
2. A “Found New Hardware” dialog will pop-up.
3. Select the “Locate and install driver software” option
4. Select “I don’t have the disc. Show me other options”.
5. Select “Browse my computer for driver software”.
6. Click on the “Browse” button and navigate to the “MotionStudio/drivers/apdm_accesspoint_drivers” folder.

7. Check “Include Subfolders”.
8. Click “Next”.
9. A warning message will be presented indicating that “Windows cannot verify the publisher of this driver software”.
10. Click “Install this driver software anyway”.
11. Close the confirmation dialog.

3.3.3 Docking Station

1. Copy the file “MotionStudio/drivers/ftd2xx.dll” to the folder located at “C:/Windows/system32”. You will need administrator priviledges to do so.
2. Attach a USB cable to the docking station. If more than one docking stations are chained together into a single unit, then external power has to be connected as well.
3. Plug the docking station into your computer.
4. A “Found New Hardware” dialog will pop-up.
5. Select the “Locate and install driver software” option
6. Select “I don’t have the disc. Show me other options”.
7. Select “Browse my computer for driver software”.
8. Click on the “Browse” button and navigate to the “MotionStudio/drivers/apdm_docking_station_drivers” folder.
9. Check “Include Subfolders”.
10. Click “Next”.
11. A warning message will be presented indicating that “Windows cannot verify the publisher of this driver software”.
12. Click “Install this driver software anyway”.
13. Close the confirmation dialog.
14. This action may need to be repeated for each docking station installed. Docking stations that are plugged in but are not yet installed will blink blue.

3.4 Windows 7 (x32/x64)

3.4.1 Access Point

1. Plug the access point into your computer.
2. There may be an notification that the device driver could not be installed.
3. Click on the Windows “Start” button
4. Right-click on the “Computer” button and select “Manage”
5. Select the “Device Manager”
6. Under “Other Devices” there should be an entry for “AccessPoint” with a yellow exclamation point next to it.
7. Right-click on the “AccessPoint” entry and select “Update Driver Software...”

8. Select “Browse my computer for driver software”
9. Click on the “Browse” button and navigate to the “MotionStudio/drivers/apdm_accesspoint_drivers” folder.
10. Check “Include Subfolders”
11. Click “Next”
12. A warning message will be presented indicating that “Windows cannot verify the publisher of this driver software”.
13. Click “Install this driver software anyway”.
14. Close the confirmation dialog.

3.4.2 Docking Station

1. Copy the file “MotionStudio/drivers/ftd2xx.dll” to the folder located at “C:/Windows/system32”. You will need administrator priviledges to do so.
2. Attach a USB cable to the docking station. If more than one docking stations are chained together into a single unit, then external power has to be connected as well.
3. Plug the docking station into your computer.
4. There may be an notification that the device driver could not be installed.
5. Click on the Windows “Start” button
6. Right-click on the “Computer” button and select “Manage”
7. Select the “Device Manager”
8. Under “Other Devices” there should be an entry for “DockingStation” with a yellow exclamation point next to it.
9. Right-click on the “DockingStation” entry and select “Update Driver Software...”
10. Select “Browse my computer for driver software”
11. Click on the “Browse” button and navigate to the “MotionStudio/drivers/apdm_docking_station_drivers” folder.
12. Check “Include Subfolders”
13. Click “Next”
14. A warning message will be presented indicating that “Windows cannot verify the publisher of this driver software”.
15. Click “Install this driver software anyway”.
16. Close the confirmation dialog.
17. This action may need to be repeated for each docking station installed. Docking stations that are plugged in but are not yet installed will blink blue.

3.5 Linux (x32/x64)

The user running the APDMsoftware libraries will need to have appropriate permissions to interface with particular USB devices. This can be configured via the udev system. The user will need access to devices with the following vendor ID (VID) and product ID (PID):

Access Point:	**VID:** 0x224F	**PID:** 0x0001
Docking Station:	**VID:** 0x224F	**PID:** 0x0002

An example set of udev rules for the access point and docking station are as follows:

```
ACTION=="add", ATTRS{idVendor}=="224f", ATTRS{idProduct}=="0001", MODE:="0666"
ACTION=="add", ATTRS{idVendor}=="224f", ATTRS{idProduct}=="0002", MODE:="0666"
```

4 Software Tools and Libraries

4.1 Programming Libraries

APDM provides programming libraries to allow integration on a variety of operating systems and platforms. The following operating systems and versions are supported as of Feb, 2010.

Language	Supported Operating Systems	Comments
C/C++	Linux, 32bit Mac OSX 10.6, Snow Leopard Mac OSX 10.5, Leopard Windows XP, 32bit, SP2/SP3 Windows 7, 64bit	
Java	Linux, 32bit Mac OSX 10.6, Snow Leopard	JNI Bindings
Matlab	Mac OSX 10.6, Snow Leopard	DYLIB Loading with C calls

4.1.1 Development Environments

- Windows / Visual Studio 2005 / CYGWIN / MinGW / GCC
- Mac OSX / GCC
- Linux / GCC

4.2 C API

4.2.1 Documentation

Included in the APDM software distribution is function API documentation, including descriptions of functions purpose, parameters and return values. This can be found under “docs/doxygen” in the software distribution.

4.2.2 Using the Host Libraries

The host libraries allow you to create handles to any given access point or docking station attached to the system. With an AP handle or docking station handle, you can query the given device for information, and send configuration commands to the given device. If there is an movement monitor attached to a docking station, then you can also send commands to the movement monitor thru the docking station handle.

4.2.3 Headers

Two headers will be necessary to include in your project, `apdm.h` and `apdm_types.h`.

4.2.4 System Context

The host libraries also provide the notion of a system context. A context is a logical collection of access points and docking stations (movement monitors attached therein) that can be configured as a group and work in concert with each other. The context allows you to correctly configure wireless channels and redundant wireless streaming AP's, as well as provide correlation of the samples sent out by all the sensors (correlation in time by sync value).

The data type used for a context is:

`apdm_ctx_t`

and can be allocated with the `apdm_ctx_allocate_new_context()` function, and free'ed with the `apdm_ctx_free_context()` function .

4.2.5 Docking Station Handle

The data type used for a docking station handle is:

`apdm_device_handle_t`

The easiest way to create a handle is to use the `apdm_sensor_allocate_and_open()` func-

tion, passing in the index of the given docking station number that you want a handle on. Similarly, calling the `apdm_sensor_close_and_free_handle()` function to cleanly close the handle and free it's respective memory.

4.2.6 Configuration of Movement Monitors on a Docking Station

The host libraries contain a number of functions, starting with `apdm_sensor_cmd_XXXX()` that are used to configure movement monitors. Settings such as sampling rates, enabling and disabling different sensors, configuration of wireless parameters etc can be done using thees function calls. See low level API documentation for details on these commands.

4.2.7 Access Point Handle

The data type used for an access point handle is:

`apdm_ap_handle_t`

An AP handle can be allocated with the

`apdm_allocate_ap_handle()`

function, free'ed with the

`apdm_free_ap_handle()`

function. Once a handle is allocated, you can open a given access point by index using the

`apdm_ap_connect()`

function. Once you've connected, you can then send commands to the AP and query the AP for information using AP specific functions.

Access Point specific functions are of the form with `apdm_ap_XXXX()`.

4.2.8 Configuration of Synchronized Wireless Streaming & Logging Mode

The host libraries provide a function

`apdm_autoconfigure_devices_and_accesspoint4()`

that can be used to configure a group of AP's and movement monitors for streaming mode. After a context has been allocated and initialized, and the

`apdm_open_all_access_points()`

function has been called with the respective context, you can call auto configure to configure the system. Once the system is configured, you can disconnect the movement monitors from the docking station to allow them to stream data, and begin to use the

`apdm_ctx_get_next_access_point_record_list()`

and

`apdm_ctx_extract_data_by_device_id()`

functions to stream data.

The maximum number of movement monitors in a single configuration is 24

The maximum number of access points in a single configuration is 6

4.2.9 Wireless Channel Selection

Movement monitors transmit data in the 2.4ghz wireless spectrum range. Channel zero corresponds to roughly 2.4000ghz, and channel 90 corresponds to roughly 2.4900ghz. The 2.4ghz spectrum has many other consumer electronic devices, such as WiFi routers, cordless phones and blue-tooth devices, that also operate in this area of the spectrum. As such, it's important to choose a channel that is not already in use by another device.

The most common source of interference is from wireless network access points. You can determine the channel that the WiFi router is running on and determine its corresponding frequency from the following URL: http://en.wikipedia.org/wiki/IEEE_802.11

4.2.10 Configuration of Synchronized Logging Mode

The host libraries provide a function,

`apdm_autoconfigure_mesh_sync()`

that will allow you to configure all movement monitors attached to the host in mesh time synchronization and data logging mode.

In synchronized logging mode, the movement monitors will transmit and receive their current time values, to and from each other such that their internal clocks all maintain the same notion of time.

There can be a maximum of 32 devices used in synchronized logging mode.

4.2.11 Configuration of Low Power Logging Mode

Low power logging mode consists of enabling/disabling the sensors of interest on the movement monitor, and disabling wireless. Wireless can be disabled with a call to

`apdm_sensor_cmd_config_set()`

and passing in

`CONFIG_ENABLE_WIRELESS`

and a value of 0.

4.2.12 Converting .APDM files to HDF5 or CSV

Data stored on the movement monitor is in a proprietary .apdm binary format. To make this data useful, it must first be passed through a set of filters that generate correct, calibrated SI unit output data.

`apdm_process_raw()`

is used to convert .apdm files to HDF5 or CSV. Raw data can also be optionally saved. The basic process is as follows:

1. Create a file with `apdm_create_file_hdf()` or `apdm_create_file_csv()`.
2. Get the device info structure for each device that's streaming using the `apdm_sensor_populate_device` function.
3. Pass an array of records and device info structures to the `apdm_write_record_hdf()` or `apdm_write_record_csv()` function for each new sample.
4. Close the data file with `apdm_close_file_hdf()` or `apdm_close_file_csv()` when done.

4.2.13 Return Codes

Most library functions return a value of type

`enum APDM_Status`

(defined in `apdm_types.h`), which indicates success or failure code of the given function that was called. A convenience function,

`apdm_strerror()`

is provided for converting these error codes to strings if necessary. Refer to function specific documentation for the details of each function.

4.2.14 Logging

The APDM libraries have logging information that is generated at various points of it's internal processing. Each log event that occurs has a specified severity, all logging funnels thru a single piece of common infrastructure. By default, log messages are sent to `STDOUT`, but by calling

```
apdm_set_log_file()
```

you can re-direct logging output to a file.

4.2.15 Threading

The host libraries are not thread safe. Thread safety, synchronization and enforcement of mutual exclusion are left up to the application in which the libraries are to be used.

4.3 Wireless Buffering and Data Correlation

In wireless streaming mode, the system utilizes numerous levels of buffering, including on-device buffering, in access point buffering, and buffering in the host libraries. There are many reasons that this buffing is necessary, including temporary wireless issues, scenarios where the host application does not retrieve data from the access point and times when the application wants to wait a short amount of time for a movement monitor to retransmit data after the wireless issues pass.

Due to the hardware level properties of the system, it becomes necessary to process data from sensors and access points knowing about potential transient problems at the hardware level. Some of the issues include the following:

- Duplicate data transmission by a sensor to one or more access points in the event that the sensor does not receive the ACK from the access point
- Variable delay in the relative streams of data from the movement sensors. e.g. one sensor may be transmitting data that is older then then the other sensors while it is catching up from a transient wireless problem.
- Missing data from a sensor, in the event that the sensor is turned off, or goes out of range

for an extended period of time.

By in large, when the system context is used for streaming data, it will resolve these issues prior to emitting data from the libraries. There are some configuration parameters that will affect the behavior of the libraries with regard to timing and potentially missing data.

`apdm_ctx_sync_record_list_head()`

Before the application begins to received data, it should call the

`apdm_ctx_sync_record_list_head()`

function. This will cause the host libraries and access point to clear out all it's buffers, stream in a few samples such that a subsequent call to

`apdm_ctx_get_next_access_point_record_list()`

will return a full sample set, with data from all sensors in the system.

If this function is not called, you may get old data, or partial sets of data from a call to

`apdm_ctx_get_next_access_point_record_list()`

4.3.1 Max Delay / Max Latency

During auto configuration, and via library calls to

`apdm_ctx_set_max_sample_delay_seconds()`

you can specify the maximum amount of time to wait for sample(s) to be re-transmitted from an movement sensor.

This setting has some important implications with regards to data reliability and the latency of

data by the time it's received by the user application.

- If a movement monitor is unable to transmit samples to an access point, the host libraries will stall their data output, waiting until max-latency seconds elapse, before giving up and emitting a partial sample set. E.G. If there are 6 sensors configured, and one of them is unable to transmit, the libraries will emit 5 samples, and indicate that they have missed the 6th sample.
- For as long as the given sensor is having problems transmitting, the host libraries will continue to delay outputting of data until the max-latency threshold for data age has elapses. So, if you have max-latency set to 15 seconds, and a sensor goes out of range, you'll find an initial pause of 15 seconds while the max-latency period elapses, then you will continue to receive data from the libraries, but as long as the sensor cannot transmit, the data will be 15 seconds old.
- The default max-latency setting is 15 seconds

4.4 DLL's, DYLIB's and SO's

Depending on platform, a DLL, DYLIB or SO will be linked in with your application at run time. These library files provide access to all the functions necessary to configure and communicate with movement monitors, docking stations and access points.

These libraries are written in C and provide standard C-symbols so as to facilitate linking with as many other languages, systems and platforms as possible.

Common ways of getting the dynamic library to load include, but are not limited to the following:

- compile time flags in your build system and making available the dynamic library for the system in one of the standard library search paths
- a call to the LoadLibrary() function on Microsoft platforms

4.4.1 Java

Java language bindings are also provided with the library distribution. These provide an object oriented interface to access points, docking stations, movement monitors and contexts. When using the java bindings, you'll need to make sure the DLL/DYLIB/SO library file is in one of the library search paths. Environmental variables can be set to achieve this or command line

parameters can be passed into the JVM to indicate where it should search for these libraries.

4.4.2 Other Systems

Many other systems, such as MatLab and LabView provide the ability to load 3rd party DLL's and call functions provided in those DLL's. Please refer to the documentation provided by your application or system on how to load and call functions from external libraries.

4.5 Example Code

The host library distribution provides sample code under dist/samples. Samples include source code and pre-compiled binaries for the respective applications. The sample applications of most interest are as follows:

- `autoconfigure_system.c`: This is used to configure a set of attached movement monitors and access points into wireless streaming mode.
- `stream_data.c`: After a system has been autoconfigured and is streaming data to its respective access points, this sample will stream data off the access points and print the data, correctly grouped, to the console.
- `autoconfigure_mesh.c`: This program is used to configure a set of movement monitors into mesh time synchronization and logging mode.
- `convert_raw.c`: This program is used to convert raw ".apdm" files from a movement monitor into a CSV or HDF output file.
- `configure_low_power_mode.c`: This program is used to configure any attached movement monitors into low power, non-streaming mode.

4.6 Programming

4.6.1 Configuring a System

Physical Configuration During Setup

C Programming

1. Allocate a handle:

`apdm_ctx_allocate_new_context()`

2. Using the handle, open access points attached to the system:
   `apdm_open_all_access_points()`

3. Autoconfigure the access point(s) and attached movement monitors.
   `apdm_autoconfigure_devices_and_accesspoint2()`

4. Disconnect from the attached access points and movement monitors
   `apdm_ctx_disconnect()`

5. Free the allocated context
   `apdm_ctx_free_context()`

Java Programming An example program for configuring and streaming data from a java application is provided below. Functions available in the java libraries are usually mappings of the corresponding c-functions, and more detailed documentation can be found in the dOxygen documentation.

```
import java.util.List;
import com.apdm.APDMNoMoreDataException;
import com.apdm.Context;
import com.apdm.IContext;
import com.apdm.RecordRaw;

public class StreamDataSample {

    public static void main(String args[]) throws Exception {
        apAutoConfig();
        System.out.println("Please disconnect devices from their cables and press enter...");
        System.in.read();
        streamData();
    }

    public static void apAutoConfig() throws Exception {

        IContext ap = Context.getInstance();
        ap.open();
```

```
        ap.autoConfigureDevicesAndAccessPoint3(90, true, false);
        // physically disconnect sensors from cables
        ap.close();

    }

    public static void streamData() throws Exception {
        IContext ap = Context.getInstance();
        ap.open();
        // Call this many times to stream data
        for (int i = 0; i < 100; i++) {
            List<RecordRaw> records = null;
            try {
                records = ap.getNextRecordList();
            } catch (APDMNoMoreDataException ex) {
                Thread.sleep(100);
            }
            if (records != null) {
                System.out.println("====================================");
                for (RecordRaw rec : records) {
                    System.out.println(rec.toString());
                }
            }
        }
        ap.close();
    }
}
```

4.7 Streaming Data from a Configured System

4.7.1 System Configuration During Data Streaming

Programming Steps

1. Allocate a handle:
   `apdm_ctx_allocate_new_context()`

2. Using the handle, open access points attached to the system:
   `apdm_open_all_access_points()`

3. Set the max latency value in the libraries.
   `apdm_ctx_set_max_sample_delay_seconds()`

4. Get the attached movement monitor ID list, if useful:
   `apdm_ctx_get_device_id_list()`

5. Synchronize the record head list in the libraries.
   `apdm_ctx_sync_record_list_head()`

6. Collect a list of sensor readings, from all movement monitors, for the same sample point in time. This is usually used within loop or as a regular event.
   `apdm_ctx_get_next_access_point_record_list()`

7. Extract data readings on a per-movement monitor basis, by movement monitor ID number.
   `apdm_ctx_extract_data_by_device_id()`

8. Disconnect from the attached access points and movement monitors
   `apdm_ctx_disconnect()`

9. Free the allocated context
   `apdm_ctx_free_context()`

5 Working with HDF5 Files

HDF5 is the preferred format for storing APDM movement monitor data. It is a standard format for scientific data that is efficient and widely supported. It uses less space than CSV, is faster to load, and supports more structured data. This section will cover the organization of the APDM movement monitor data and the basics of reading HDF5 files in MATLAB.

5.1 HDFView

A free program called HDFView (http://www.hdfgroup.org/hdf-java-html/hdfview/) can be used to explore, plot, and export this data into other formats. A variety of free open source tools for working with HDF files are also available at http://www.hdfgroup.org/HDF5/release/obtain5.html.

5.2 Data Organization

HDF5 files are organized like a file structure. The root of the file contains two attributes. One is a list of monitor IDs that have data stored in this file. The other is a version number for the organization of the HDF 5 file.

5.3 File Structure

5.3.1 Version 2

- **MonitorLabelList** Attribute containing an array of monitor labels in the same order as the CaseIdList
- **CaseIdList Attribute** containing an array of monitor case IDs in the same order as the MonitorLabelList
- **FileFormatVersion** Attribute containing the file format version (2)
- **Annotations** Table containing annotations
  - **Time** Annotation time in epoch microseconds
  - **Case ID** A movement monitor case ID associated with the annotation
  - **Annotation** The annotation string
- **AA-XXXXXX** A group is included in the file for each monitor in the CaseIdList, with the name equal to the case ID
  - **SampleRate** Attribute containing the output data rate for the monitor
  - **DecimationFactor** Decimation factor for the monitor's internal processing

- **ModuleID** The module ID for the monitor
- **TimeGood** Flag indicating whether the time has been set on the monitor since it powered on
- **RecordingMode** One of: ”Wireless streaming”, ”Synchronized logging”, or ”Unsynchronized logging”
- **DataMode** Indicates whether the data was retrieved wirelessly or copied from the monitor’s internal storage while docked. One of: ”Streamed wirelessly” or ”Logged to monitor”
- **AccelerometersEnabled** 1 for enabled, 0 for disabled
- **GyroscopesEnabled** 1 for enabled, 0 for disabled
- **MagnetometersEnabled** 1 for enabled, 0 for disabled
- **DecimationBypass** Internal use, deprecated
- **CalibrationVersion** Version of the calibration data used to convert from raw samples to calibrated SI units
- **VersionString1** Firmware version string 1
- **VersionString2** Firmware version string 2
- **VersionString3** Firmware version string 3
- **CalibratedDataPopulated** 1 for populated, 0 for unpopulated
- **LocalTimeOffset** Time in milliseconds to add to UTC to convert to local time
- **SyncValue** Dataset containing the internal sync value for each sample
  - **Units** Attribute string containing the timestamp units (1/2560th of a second since 0:00 Jan 1, 1970 UTC)
- **Time** Dataset containing a timestamp for each sample
  - **Units** Attribute string containing the units (microseconds since 0:00 Jan 1, 1970 UTC)
- **Calibrated** Group containing calibrated data
  - **Accelerometers** Dataset containing accelerometer data (Nx3)
    - **Units** Attribute string containing the accelerometer units (m/s^2)
    - **Range** Attribute containing the range setting for the accelerometer (2g or 6g)
  - **Gyroscopes** Dataset containing gyroscope data (Nx3)
    - **Units** Attribute string containing the gyroscope units (rad/s)
  - **Magnetometers** Dataset containing magnetometer data (Nx3)
    - **Units** Attribute string containing the magnetometer units (μT)
  - **Temperature** Dataset containing the temperature (Nx1)
    - **Units** Attribute string containing the temperature units (°C)
  - **TemperatureDerivative** Dataset containing the temperature derivative (Nx1)
    - **Units** Attribute string containing the temperature derivative units (°C/s)
- **Raw** Group containing raw data if selected during import
  - **Accelerometers**
  - **Gyroscopes**
  - **Magnetometers**
  - **DataFlags**
  - **OptData**
  - **Temperature**
  - **TemperatureDerivative**

5.3.2 Version 1

This version is deprecated. All new files created will use the most recent version.

- **Device_List** Attribute containing a list of monitors present in the file
- **File_Format_Version** Attribute containing the file version
- **Annotations** Table containing annotations
  - **Time** Annotation time in epoch microseconds
  - **Device ID** A movement monitor ID associated with the annotation
  - **Annotation** The annotation string
- **Opal_xxx**/ Group containing information about and data from monitor ID xxx
  - **Sample_Rate** Attribute containing the output data rate for the monitor
  - **Decimation_Factor** Decimation factor for the monitor's internal processing
  - **Time_Good** Flag indicating whether the monitor has had its time set since turning on
  - **Decimation_Bypass** Internal use, deprecated
  - **Calibration_Version** Version of the calibration data used to convert from raw samples to calibrated SI units
  - **Version_String1** Firmware version string 1
  - **Version_String2** Firmware version string 2
  - **Version_String3** Firmware version string 3
  - **Acceleration** Dataset containing data from the accelerometers (Nx3)
    - **Units** Attribute string containing the acceleration units (m/s^2)
  - **Angular_Velocity** Dataset containing data from the gyroscopes (Nx3)
    - **Units** Attribute string containing the angular velocity units (rad/s)
  - **Magnetic_Field** Dataset containing data from the magnetometers (Nx3)
    - **Units** Attribute string containing the magnetic field units (a.u.)
  - **Temperature** Dataset containing the temperature of the monitor (Nx1)
    - **Units** Attribute string containing the temperature units (°C)
  - **Temperature_Derivative** Dataset containing the rate of change of temperature
    - **Units** Attribute string containing the temperature derivative units (°C/s)
  - **Sync_Value** Dataset containing the internal timestamp of each sample
    - **Units** Attribute string containing the timestamp units (1/2560th of a second since 0:00 Jan 1, 1970 UTC)
    - **Time** Dataset containing the time for each sample in microseconds since 0:00 Jan 1, 1970 UTC

Additional fields present when raw data is also stored:

- **Opal_XX**/
  - **Calibration_Data** Attribute containing binary block of calibration data

- **Raw_File_Version** Attribute containing the version string of the raw file (if this was converted from a .apdm file instead of streamed)
- **Accelerometers_Raw** Dataset containing raw accelerometer data ($Nx3$)
- **Gyroscopes_Raw** Dataset containing raw gyroscope data ($Nx3$)
- **Magnetometers_Raw** Dataset containing raw magnetometer data ($Nx3$)
- **Data_Flags** Dataset containing flags used for processing the raw data
- **Opt_Data** Dataset containing several measurements taken at a low data rate
- **Temperature_Raw** Dataset containing lowpass filtered, but uncalibrated temperature data ($Nx1$)

5.4 Working with HDF 5 in MATLAB

MATLAB contains two high level functions for working with HDF5 files. Additional help and examples are included in the built in help documentation for these functions.

`hdf5info` reads the structure of the file and all of the attribute values and returns them in an easy to browse MATLAB structure.

`hdf5read` reads a complete dataset or attribute from the HDF5 file.

Additionally, one more high level helper function is included with the APDM movement monitor software. This function also contains built in help documentation and examples.

`hdf5readslab` reads a portion of a dataset from the HDF5 file.

5.5 Examples

Below is simple example of loading acceleration data from an APDM movement monitor HDF5 file (version 2) in MATLAB.

```
filename = 'example.h5';
try
   vers = hdf5read(filename, '/FileFormatVersion');
catch
   try
      vers = hdf5read(filename, '/File_Format_Version');
   catch
      error('Couldn''t determine file format');
   end
```

```
end
if vers ~= 2
   error('This example only works with version 2 of the data file')
end
caseIdList = hdf5read(filename, '/CaseIdList');
groupName = caseIdList(1).data;
accPath = [groupName '/Calibrated/Accelerometers'];
fs = hdf5read(filename, [groupName '/SampleRate']);
fs = double(fs);
acc = hdf5read(filename, accPath)'; %Transposed to make Nx3 in MATLAB
t = (1:size(acc,1))/fs;
figure;
plot(t,acc);
```

A more complicated example using the flexibility of HDF5 to load and process only part of a data set. This can be useful when the data set is too large to fit into memory. Care is taken not to attempt to read beyond the end of the file.

```
filename = 'example.h5';
try
   vers = hdf5read(filename, '/FileFormatVersion');
catch
   try
      vers = hdf5read(filename, '/File_Format_Version');
   catch
      error('Couldn''t determine file format');
   end
end
if vers ~= 2
   error('This example only works with version 2 of the data file')
end
idList = hdf5read(filename, '/CaseIdList');
groupName = idList(1).data;
accPath = [groupName '/Calibrated/Accelerometers'];
fs = hdf5read(filename, [groupName '/SampleRate']);
fs = double(fs);
fhandle = H5F.open(filename, 'H5F_ACC_RDONLY', 'H5P_DEFAULT');
dset = H5D.open(fhandle, [groupName '/Calibrated/Accelerometers'], 'H5P_DEFAULT');
dspace = H5D.get_space(dset);
[ndims, dims] = H5S.get_simple_extent_dims(dspace);
nSamples = dims(1);
```

```
nSamplesRead = min(nSamples, 60*fs); %read at most one minute of data
accSegment = hdf5readslab(filename, accPath, [0,0], [nSamplesRead, 3])';
t = (1:nSamplesRead)/fs;
figure;
plot(t,accSegment);
```

5.6 Notes

- Arrays in MATLAB use the FORTRAN convention of storing them in memory by column then row, instead of the C convention (used by HDF 5) of row then column. This has the effect of making the returned arrays transposed from how this document (and many other interfaces to HDF5) claim they are laid out.
- Older versions of MATLAB (before 2009a) did not support the compression used in Motion Studio's HDF 5 files. If you are using one of these older versions, the free h5repack utility available from the HDF Group can remove the compression. This utility is available at:
  http://www.hdfgroup.org/HDF5/release/obtain5.html
  The command to repack the file is:
  `h5repack -f NONE example.h5 example_no_compression.h5`

6 Monitor Reference

6.1 Charging

A movement monitor charges its internal battery any time it is connected to a docking station. At the optimal charge rate the movement monitors internal battery will complete its bulk charge (80%-90%) within an hour for a fully discharged battery. It is recommended that the movement monitor be charged for up to 3 hours to provide a peak charge to the battery ensuring it has the longest run time and improves battery life. It is recommended for the health of the battery to to have at least a bulk charge for storage of the movement monitor.

6.2 Powering Down

If you wish to power down your monitors for storage or travel, dock or plug in the monitors you wish to power down and select the “Tools→Halt All Monitors” option in Motion Studio. After this is selected, all monitors will power down when they are undocked or unplugged.

6.3 Data Storage

The movement monitor utilizes a flash card to store data while logging. This data can be downloaded by using a docking station to dock the movement monitor. When the movement monitor is docked it finishes up writing to the internal flash card and then releases it to the docking station. At this time the docking station indicates to the PC that there is a new read only removable drive to be mounted. Using your file browser you can navigate to the removable drive and copy the files off of it. The files are in a proprietary raw format and need to be converted to either a HDF5 or CSV format that will provide data in calibrated SI units. This conversion happens automatically if Motion Studiois used to import the data. Alternately, there are functions in the SDK to do this conversion programmatically.

6.4 Cleaning and Storage

Cleaning the movement monitors case should be done by wiping the bottom of the case where it contacts the skin with Rubbing alcohol or other cleaning wipe. If the entire case needs to be cleaned use only an ethyl alcohol or isopropyl alcohol based wipe. Methyl alcohol should be

avoided for cleaning the top since it will cause degradation of the plastic over time. The movement monitor should not be submerged in any liquids or subjected to any high temperatures for cleaning. The straps on the monitor can be cleaned by wiping them down with Rubbing alcohol. Alternatively the straps can be removed and washed separately using mild soap and water. Storage of the movement monitor should be in a dry static fee location. An anti-static bag or in the supplied case is recommended. The movement monitor should also not be subjected to any large G forces to prevent damage or changes to the calibration of the sensors in the monitor. The movement monitor should also have an adequate charge to ensure a good battery lifetime.

6.5 Drivers

Drivers are provided as part of the library distribution and Motion Studio. Instructions for installing drivers are provide in the “Hardware Driver Installation” section of this document.

6.6 Firmware Updates

Updating the movement monitor firmware should be done using the Motion Studio software.

6.7 Technical Specifications

- The accelerometer range is $\pm 58.8\,\mathrm{m/s^2}$ (6 g) (optionally $\pm 19.6\,\mathrm{m/s^2}$ (2 g)).
- Accelerometers have a typical noise density of $1.3\,\mathrm{mm/s^2/\sqrt{Hz}}$.
- The X and Y axis gyros have a range of ± 34.9 rad/s (2000 dps)
- The Z axis gyro has a range of ± 26.8 rad/s (1500 dps)
- The X and Y axis gyros have a typical noise density of 0.81 mrad/s/$\sqrt{\mathrm{Hz}}$
- The Z axis gyro have a typical noise density of 2.2 mrad/s/$\sqrt{\mathrm{Hz}}$
- Magnetometers have a range of ± 6 Gauss
- The magnetometers have a typical noise density is 160 nT/$\sqrt{\mathrm{Hz}}$
- Positive X is pointing from the monitor toward the connector. Positive Y is pointing left of X looking top down at the monitor. Z is pointing up out of the top of the case. Angular velocity sign is defined according to a right hand rule. A counterclockwise rotation about the Z axis looking from the +Z direction is positive.

6.8 LED Reference

6.8.1 Status Codes and LED Colors/Patterns

The LEDs on the access points and movement monitors provide important information about the operating state of the hardware, including error statuses. The tables below list the LED patterns associated with these states and can be useful in troubleshooting issues encountered with the hardware.

6.8.2 Movement Monitor LED Reference

Movement monitors contain a RGB LED capable of outputting a wide array of colors to the user to indicate its current state. The following colors are used: white (○), red (●), yellow (●), green (●), cyan (●), blue (●), magenta (●), and led off (_). In the off state the LED will appear as a non illuminated white dot in the corner of the monitor opposite the docking connector. All LED patterns are output on a repeating cycle which may vary in period depending on the pattern. In all cases the last color listed will stay constant until the pattern repeats. For example "●_●_" will blink yellow twice and then stay off until the pattern repeats.

State	LED Pattern
Startup (boot loader)	
Startup wait (5 sec)	●
Failed to load firmware	●
Boot loader mode	○
Firmware	
Reset mode	○_
Docked mode (transition)	●_
Docked mode (bulk charging)	●●(fast)
Docked mode (trickle charging)	●●(slow)
Docked mode (full charge)	●
Docked mode (bad cable connection)	●_●
Error mode: default	●_●_
Error mode: configuration	●_●_●_
Error mode: system	●_●_●_●_
Error mode: data buffer	●_●_●_●_●_
Error mode: SD buffer	●_●_●_●_●_●_
Error mode: SD I/O	●_●_●_●_●_●_●_
Card is full	●_
Run mode (transition)	●
Run mode (battery level indication off)	●_
Run mode (battery level 4/4)	●_●_●_●_
Run mode (battery level 3/4)	●_●_●_
Run mode (battery level 2/4)	●_●_
Run mode (battery level 1/4)	●_
Powering off	●_
Wireless Streaming Debug LED Modes	
Normal	●_
CPU out of cycles	●_●_
Missed an access point time packet	●_●_

7 Access Point Reference

7.1 Drivers

Drivers are provided as part of the library distribution and Motion Studio. Instructions for installing drivers are provide in the “Hardware Driver Installation” section of this document.

7.2 Firmware Updates

Updating the movement monitor firmware should be done using the Motion Studio software.

7.3 Mounting and Placement

The antennas of the access point are located directly behind the black plastic face of the access point. The access point(s) should be aimed such that this face is in the approximate direction of the area where the movement monitors will be used.

7.4 Single vs. Dual

With the wide range of wireless environments it is not always possible to provide a reliable channel of communication between any two nodes. Using two access points per set of up to six movement monitors is one method of improving reliability by utilizing spacial diversity. In this mode a movement monitor can hop between access points depending on which one has a better signal path. For large spaces or with spaces that have obstacles that may block wireless signals this mode of operation is recommended. Configuration of a system in this mode is transparent to the user and is automatically selected when there is enough access points available for the given number of movement monitors. Using multiple access points requires them to all have a synchronization cable connected between them.

7.5 LED Reference

Access points contain a RGB LED capable of outputting a wide array of colors to the user to indicate its current state. The following colors are used: white (○), red (●), yellow (●), green (●), cyan (●), blue (●), magenta (●), and led off (_). All LED patterns are output on a repeating

cycle which may vary in period depending on the pattern. In all cases the last color listed will stay constant until the pattern repeats. For example "●_●_" will blink yellow twice and then stay off until the pattern repeats.

State	LED Pattern
Access point is powered on and is not receiving data from any monitors	●
Access point is receiving data from all monitors and there is no excessive latency for any of the monitors	●_
Access point is receiving data from all monitors but there is excessive latency (¿3s) in one or more monitors. The latency is, however, decreasing (improving). This usually indicates that one or more monitors was temporarily obstructed and is now catching up.	●●
Access point is receiving data from all monitors but there is excessive latency (¿3s) in one or more monitors which is increasing (getting worse). This usually indicates that one or more monitors is obstructed and is having trouble transmitting its data.	●●
Access point is receiving data from one or more, but not all, of the movement monitors	●_
Access point is receiving data from one or more monitors that it is not expecting to receive data (e.g. there is a monitor configured on another computer system streaming data)	●●(fast)

8 Docking Station Reference

8.1 Drivers

Drivers are provided as part of the library distribution and Motion Studio. Instructions for installing drivers are provide in the “Hardware Driver Installation” section of this document.

8.2 LED Reference

Docking stations contain a RGB LED capable of outputting a wide array of colors to the user to indicate its current state. The following colors are used: white (○), red (●), yellow (●), green (●), cyan (●), blue (●), magenta (●), and led off (_). All LED patterns are output on a repeating cycle which may vary in period depending on the pattern. In all cases the last color listed will stay constant until the pattern repeats. For example “●_●_” will blink yellow twice and then stay off until the pattern repeats.

State	LED Pattern
OK	●
Powered off, USB suspended, or booloader pause	●
OK, but USB not enumerated	●
Power problem. Need to plug in external power or USB power.	●_
Docked, SD unavailable on host	●_●
Docked, SD card mounted on host	●
SD card read-access in progress	●_
USB error	●
Error	●_
Error: SD card mounting error	●_●_
Error: in-dock USB hub problem	●_●_●_
Bootloader mode	●
Updating firmware	○
Hardware Error -DA	●_○_●_○_●_○_
Hardware Error - GA	●_●_●_●_●_●_
Hardware Error - PA	●_●_●_●_●_●_
Hardware Error - UA	●_●_●_●_●_●_

8.3 Power

- If running a single docking station, it can be powered from:
  - a USB cable plugged into a dedicated USB port on your computer
  - a USB cable plugged into a a powered USB hub
  - a USB cable plugged into a wall adapter (charging only)
  - the external AC adapter (charging only)
- If running a chain of 2 or more docking stations:
  - For data transfer, both USB and external AC power are required. If a power-related error occurs, then the docking station will blink yellow until external or power is plugged in.
  - if only charging is required, the external AC power must be used

9 Technical Support

APDM is pleased to assist you with any questions you may have about our software or about the use of the technology for your application.

Please contact us at:
email: info@apdm.com
telephone: 888-988-APDM (2736)