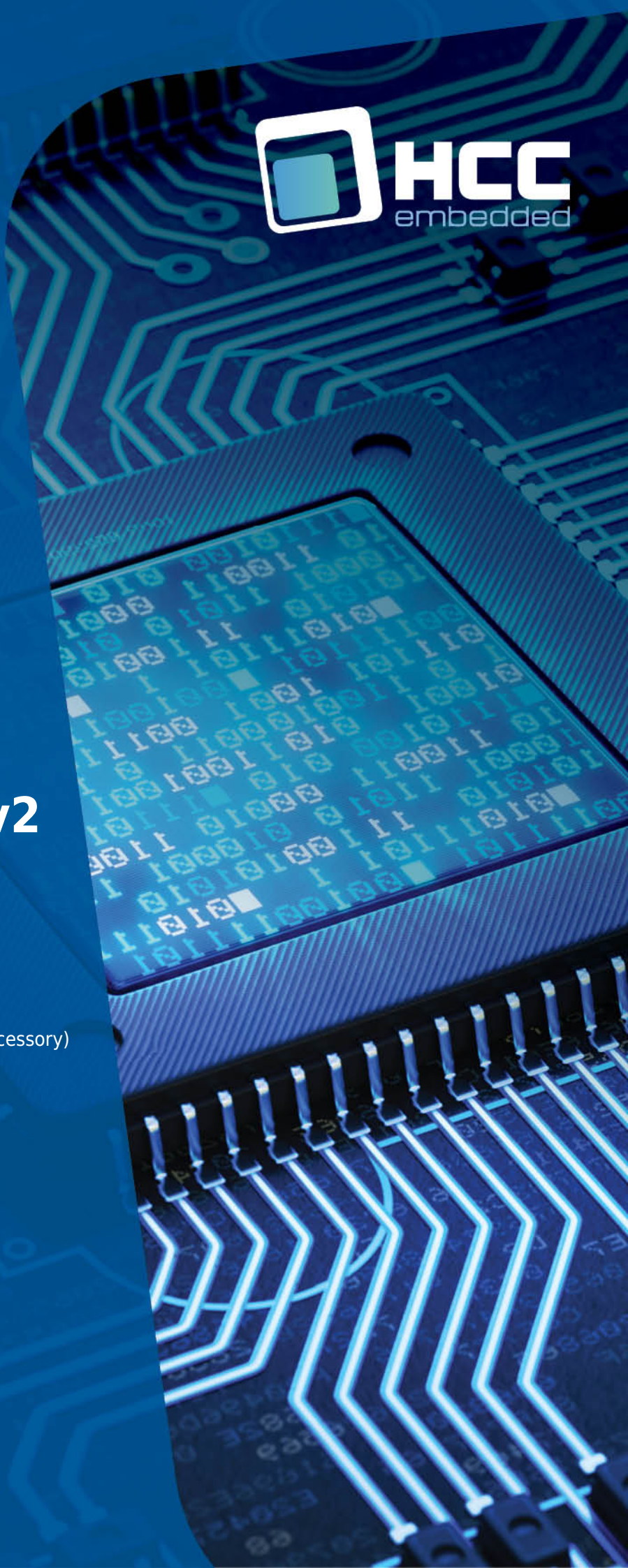



# USB Host AOAv2 Class Driver

Version 1.00

For use with USBH AOAv2 (Android Open Accessory) Class Driver versions 1.01 and above

# Table of Contents

# 1. System Overview

This chapter contains the fundamental information for this module.

The component sections are as follows:

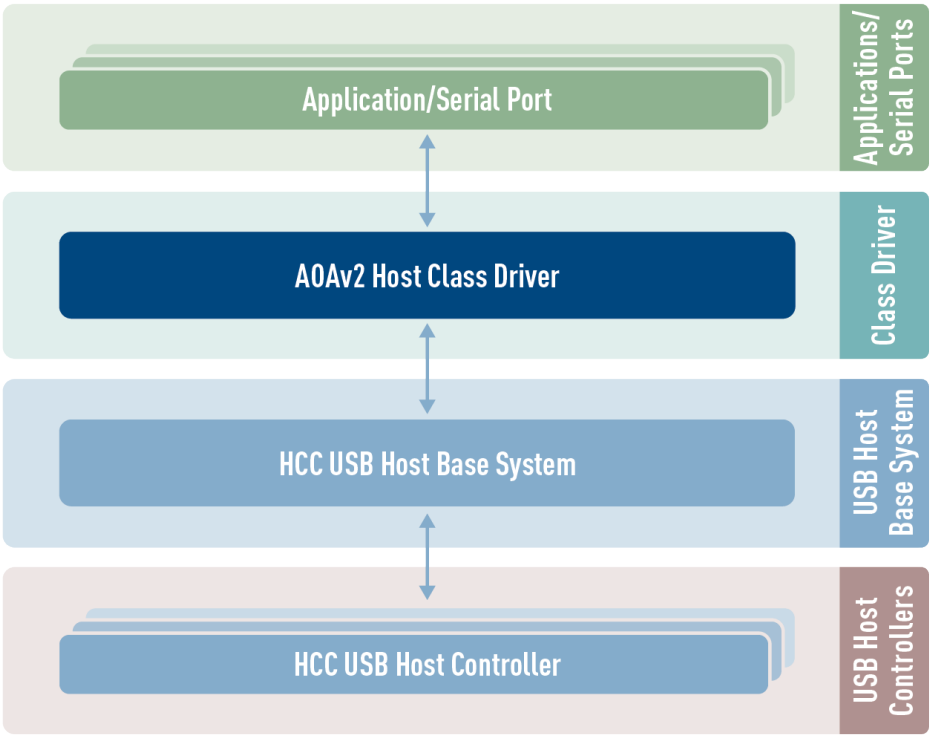
- Introduction – describes the main elements of the module.
- Feature Check – summarizes the main features of the module as bullet points.
- Packages and Documents – the *Packages* section lists the packages that you need in order to use this module. The *Documents* section lists the relevant user guides.
- Change History – lists the earlier versions of this manual, giving the software version that each manual describes.

## 1.1. Introduction

This guide is for those who want to implement an Embedded USB Host class driver to control Android Open Accessory (AOAv2) USB devices. The USB AOAv2 class is designed for connecting Android devices that support accessory mode.

AOA support allows external USB hardware (Android USB accessories) to interact with Android-powered devices in *accessory mode*. When an Android-powered powered device is in accessory mode, the connected accessory acts as the USB host (powers the bus and enumerates devices) and the Android-powered device acts as the USB accessory.

The **usbh_cd_aoa2** package provides a host class driver for a USB stack. The system structure is shown in the diagram below:




**Note:** For detailed information about the lower layer, see the *HCC USB Host Base System User Guide* that is shipped with the base system.

The package provides a set of API functions for controlling access to a device.

## 1.2. Feature Check

The main features of the class driver are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Supports Android Open Accessory (AOAv2) devices.
- Compatible with all HCC USB host controllers.
- A callback can be registered for RX/TX transfer complete notification.

## 1.3. Packages and Documents

## Packages

The table below lists the packages that you need in order to use this module:

| Package | Description |
|---|---|
| **hcc_base_doc** | This contains the two guides that will help you get started. |
| **usbh_base** | The USB host base package. This is the framework used by USB class drivers to communicate over USB using a specific USB host controller package. |
| **usbh_cd_aoa2** | The USB device AOA2 host class driver package described in this document. |

## Documents

For an overview of HCC's embedded USB stacks, see Product Information on the main HCC website.

Readers should note the points in the HCC Documentation Guidelines on the HCC documentation website.

#### HCC Firmware Quick Start Guide

This document describes how to install packages provided by HCC in the target development environment. Also follow the *Quick Start Guide* when HCC provides package updates.

#### HCC Source Tree Guide

This document describes the HCC source tree. It gives an overview of the system to make clear the logic behind its organization.

#### HCC USB Host Base System User Guide

This document defines the USB host base system upon which the complete USB stack is built.

#### HCC Embedded USB Host AOA2 Class Driver User Guide

This is this document.

## 1.4. Change History

This section describes past changes to this manual.

- To download this manual or a PDF describing an earlier software version, see USB Host PDFs.
- For the history of changes made to the package code itself, see History: usbh_cd_aoa2.

The current version of this manual is 1.00.

| Manual version | Date | Software version | Reason for change |
|---|---|---|---|
| 1.00 | 2020-06-23 | 1.01 | First release. |

# 2. Source File List

The following sections describe all the source code files included in the system. These files follow the HCC Embedded standard source tree system, described in the *HCC Source Tree Guide*. All references to file pathnames refer to locations within this standard source tree, not within the package you initially receive.

**Note:** Do not modify any files except the configuration file.

## API Header File

The file **src/api/api_usbh_aoa2.h** is the only file that should be included by an application using this module. **This file should only be modified by HCC**. For details of the API functions, see Application Programming Interface.

## Configuration File

The file **src/config/config_usbh_aoa2.h** contains all the configurable parameters. Configure these as required. For details of these options, see Configuration Options.

## System File

The file **src/usb-host/class-drivers/aoa2/usbh_aoa2.c** holds the source code. **This file should only be modified by HCC**.

## Version File

The file **src/version/ver_usbh_aoa2.h** contains the version number of this module. This version number is checked by all modules that use this module to ensure system consistency over upgrades.

## Demo files

The files **usbh_aoa2_demo.c** and **.h** are in **src/psp_template/board/demo**. For details see Sample Code.

# 3. Configuration Options

Set the system configuration options in the file **src/config/config_usbh_aoa2.h**. This section lists the available options and their default values. These provide identification string information that the host can send to the device.

**USBH_AOA2_MANUFACTURER_NAME**

The name of the host system manufacturer. The default is "".

**USBH_AOA2_MODEL_NAME**

The name of the host system model. The default is "".

**USBH_AOA2_DESCRIPTION**

The host system description. The default is "".

**USBH_AOA2_VERSION**

The AOA2 protocol version. The default is "".

**USBH_AOA2_URI**

The Uniform Resource Identifier (URI). The default is "".

**USBH_AOA2_SERIAL_NUMBER**

The host system serial number. The default is "".

# 4. Application Programming Interface

This section documents the Application Programming Interface (API). It includes all the functions that are available to an application program.

## 4.1. Module Management

These functions manage the class driver.

| Function | Description |
|---|---|
| **usbh_aoa2_init()** | Initializes the class driver and allocates the required resources. |
| **usbh_aoa2_start()** | Starts the class driver. |
| **usbh_aoa2_stop()** | Stops the class driver. |
| **usbh_aoa2_delete()** | Deletes the class driver and releases the associated resources. |

## usbh_aoa2_init

Use this function to initialize the class driver and allocate the required resources.

For an example of this function in use, see the Initialization Example.

### Format

```
int usbh_aoa2_init ( void )
```

### Arguments

None.

### Return Values

| Return value | Description |
|---|---|
| USBH_SUCCESS | Successful execution. |
| USBH_ERR_RESOURCE | OS resource error. |

## usbh_aoa2_start

Use this function to start the class driver.

For an example of this function in use, see the Initialization Example.

**Note:** Call **usbh_aoa2_init()** before this to initialize the class driver.

### Format

```
int usbh_aoa2_start ( void )
```

### Arguments

None.

### Return Values

| Return value | Description |
|---|---|
| USBH_SUCCESS | Successful execution. |
| USBH_ERR_RESOURCE | OS resource error. |

## usbh_aoa2_stop

Use this function to stop the class driver.

### Format

```
int usbh_aoa2_stop ( void )
```

### Arguments

None.

### Return Values

| Return value | Description |
|---|---|
| USBH_SUCCESS | Successful execution. |
| USBH_ERR_RESOURCE | OS resource error. |

## usbh_aoa2_delete

Use this function to delete the class driver and release the associated resources.

### Format

```
int usbh_aoa2_delete ( void )
```

### Arguments

None.

### Return Values

| Return value | Description |
|---|---|
| USBH_SUCCESS | Successful execution. |
| USBH_ERR_RESOURCE | OS resource error. |

## 4.2. AOAv2 Class Driver Management

These functions manage the class driver.

| Function | Description |
|---|---|
| **usbh_aoa2_read()** | Starts a read transfer. |
| **usbh_aoa2_read_state()** | Checks the status of a transfer started with **usbh_aoa2_read()**. |
| **usbh_aoa2_write()** | Starts a write transfer. |
| **usbh_aoa2_write_state()** | Checks the status of a transfer started with **usbh_aoa2_write()**. |
| **usbh_aoa2_get_port_hdl()** | Gets the handle of the USB port the device is connected to. |
| **usbh_aoa2_get_protocol_version()** | Gets the protocol version. |
| **usbh_aoa2_is_present()** | Checks whether a device is connected. |
| **usbh_aoa2_register_ntf()** | Registers a notification function for a specified event type. |

## usbh_aoa2_read

Use this function to start a read transfer.

To check for completion of the transfer, call the **usbh_aoa2_read_state()** function.

### Format

```
int usbh_aoa2_read (
    uint8_t     channel,
    uint8_t *   pc_dst,
    uint32_t    blen,
    uint8_t     is_skipping_zlp )
```

### Arguments

| Parameter | Description | Type |
|---|---|---|
| channel | The channel number. The range is 0..1. | uint8_t |
| pc_dst | A pointer to the destination buffer. | uint8_t * |
| blen | The length of the buffer. | uint32_t |
| is_skipping_zlp | If not zero, do not expect a zero length packet. | uint8_t |

### Return Values

| Return value | Description |
|---|---|
| USBH_AOA2_SUCCESS | Successful execution. |
| USBH_AOA2_BUSY | The channel is in use. |
| USBH_AOA2_NOT_PRESENT | The channel is not present. |
| USBH_AOA2_ERROR_PARAMETER | A parameter is invalid. |
| USBH_AOA2_ERROR_TRANSFER | The transfer failed. |

## usbh_aoa2_read_state

Use this function to check the status of a transfer started with **usbh_aoa2_read()**.

### Format

```
int usbh_aoa2_read_state (
    uint8_t      channel,
    uint8_t      b_block,
    uint32_t *   p_rlen )
```

### Arguments

| Parameter | Description | Type |
|---|---|---|
| channel | The channel number. The range is 0..1. | uint8_t |
| b_block | Set this to 1 to make this function wait until the IN transfer finishes. | uint8_t |
| p_rlen | A pointer to the number of bytes read. | uint32_t * |

### Return Values

| Return value | Description |
|---|---|
| USBH_AOA2_SUCCESS | Successful execution. |
| USBH_AOA2_BUSY | The channel is in use by another transfer. |
| USBH_AOA2_NOT_PRESENT | The channel is not present. |
| USBH_AOA2_ERROR_PARAMETER | A parameter is invalid. |
| USBH_AOA2_ERROR_TRANSFER | The transfer failed. |

## usbh_aoa2_write

Use this function to start a write transfer.

To check for completion of the transfer, call the **usbh_aoa2_write_state()** function.

### Format

```
int usbh_aoa2_write (
    uint8_t     channel,
    uint8_t *   pc_buf,
    uint32_t    blen,
    uint8_t     is_skipping_zlp )
```

### Arguments

| Parameter | Description | Type |
|---|---|---|
| channel | The channel number. The range is 0..1. | uint8_t |
| pc_buf | A pointer to the buffer which holds the data to write. This buffer preserves data until the end of the transfer. | uint8_t * |
| blen | The length of the buffer. | uint32_t |
| is_skipping_zlp | If this is not zero, do not send zero length packets. | uint8_t |

### Return Values

| Return value | Description |
|---|---|
| USBH_AOA2_SUCCESS | Successful execution. |
| USBH_AOA2_BUSY | The channel is in use. |
| USBH_AOA2_NOT_PRESENT | The channel is not present. |
| USBH_AOA2_ERROR_PARAMETER | A parameter is invalid. |
| USBH_AOA2_ERROR_TRANSFER | The transfer failed. |

## usbh_aoa2_write_state

Use this function to check the status of a transfer started with **usbh_aoa2_write()**.

### Format

```
int usbh_aoa2_write_state (
    uint8_t     channel,
    uint8_t     b_block )
```

### Arguments

| Parameter | Description | Type |
|---|---|---|
| channel | The channel number. The range is 0..1. | uint8_t |
| b_block | Set this to 1 to make this function wait until the OUT transfer finishes. | uint8_t |

### Return Values

| Return value | Description |
|---|---|
| USBH_AOA2_SUCCESS | Successful execution. |
| USBH_AOA2_BUSY | The channel is in use by another transfer. |
| USBH_AOA2_NOT_PRESENT | The channel is not present. |
| USBH_AOA2_ERROR_PARAMETER | A parameter is invalid. |
| USBH_AOA2_ERROR_TRANSFER | The transfer failed. |

## usbh_aoa2_get_port_hdl

Use this function to get the handle of the USB port the device is connected to.

### Format

```
t_usbh_port_hdl usbh_aoa2_get_port_hdl ( void )
```

### Arguments

None.

### Return Values

| Return value | Description |
|---|---|
| The port handle. | Successful execution. |
| USBH_PORT_HDL_INVALID | Operation failed. |

## usbh_aoa2_get_protocol_version

Use this function to get the AOA2 protocol version.

### Format

```
uint8_t usbh_aoa2_get_protocol_version ( void )
```

### Arguments

None.

### Return Values

| Return value | Description |
|---|---|
| 0 | The connected Android device does not support AOA. |
| 1 | Protocol version v1. |
| 2 | Protocol version v2 |

## usbh_aoa2_is_present

Use this function to check whether a device is connected or not.

### Format

```
int usbh_aoa2_is_present ( void )
```

### Arguments

None.

### Return Values

| Return value | Description |
|---|---|
| 0 | No device is present. |
| 1 | A device is present. |

## usbh_aoa2_register_ntf

Use this function to register a notification function for a specified event type.

When a read or write completes, the notification function is called.

**Note:** It is the user's responsibility to provide any notification functions required by the application. Providing such functions is optional.

### Format

```
int usbh_aoa2_register_ntf (
    t_usbh_ntf       ntf,
    t_usbh_ntf_fn    p_ntf_fn )
```

### Arguments

| Parameter | Description | Type |
|---|---|---|
| ntf | The notification ID. | t_usbh_ntf |
| p_ntf_fn | The notification function to call when an event occurs. | t_usbh_ntf_fn |

### Return Values

| Return value | Description |
|---|---|
| USBH_SUCCESS | Successful execution. |
| USBH_ERR_INVALID | A parameter is invalid. |

## 4.3. Error Codes

If a function executes successfully it returns with a USBH_AOA2_SUCCESS code, a value of 0. The following table shows the meaning of these error codes:

| Return Code | Value | Description |
|---|---|---|
| USBH_AOA2_SUCCESS | 0 | Successful execution. |
| USBH_AOA2_BUSY | 1 | Another transfer is already in progress. |
| USBH_AOA2_NOT_PRESENT | 2 | No device is connected/no channel is available. |
| USBH_AOA2_ERROR_PARAMETER | 3 | A parameter is invalid. |
| USBH_AOA2_ERROR_TRANSFER | 4 | Cannot process the transfer. |

# 4.4. Types and Definitions

This section describes the *t_usbh_ntf_fn* and the codes and other elements that are defined in API Header files.

## t_usbh_ntf_fn

The **t_usbh_ntf_fn** definition specifies the format of the notification function. It is defined in the USB host base system in the file **api_usb_host.h**.

### Format

```
int ( * t_usbh_ntf_fn )(
    t_usbh_unit_id   uid,
    t_usbh_ntf       ntf )
```

### Arguments

| Parameter | Description | Type |
|---|---|---|
| uid | The unit ID. | t_usbh_unit_id |
| ntf | The notification code. | t_usbh_ntf |

## Notification Codes

The notification codes are shown below:

| Notification | Value | Description |
|---|---|---|
| USBH_AOA2_MAX_CHANNELS | 2 | The maximum number of channels. |
| USBH_AOA2_NTF_RX( ch ) | ( USBH_NTF_CD_BASE + ch ) | Read notification code. |
| USBH_AOA2_NTF_TX( ch ) | ( USBH_NTF_CD_BASE + USBH_AOA2_MAX_CHANNELS + ch ) | Transfer notification code. |

# 5. Integration

This section specifies the elements of this package that need porting, depending on the target environment.

## 5.1. OS Abstraction Layer

All HCC modules use the OS Abstraction Layer (OAL) that allows the module to run seamlessly with a wide variety of RTOSes, or without an RTOS.

The module uses the following OAL components:

| OAL Resource | Number Required |
|---|---|
| Tasks | 0 |
| Mutexes | 1 |
| Events | 4 |

## 5.2. PSP Porting

The Platform Support Package (PSP) is designed to hold all platform-specific functionality, either because it relies on specific features of a target system, or because this provides the most efficient or flexible solution for the developer. For full details of its functions, see the *HCC Base Platform Support Package User Guide*.

The module makes use of the following standard PSP functions:

| Function | Package | Component | Description |
|---|---|---|---|
| **psp_memset()** | psp_base | psp_string | Sets the specified area of memory to the defined value. |
| **psp_strnlen()** | psp_base | psp_string | Returns the length of a string. |

# 6. Sample Code

This section shows example code for the class driver. You are provided with two demo files:

- **usbh_aoa2_demo.h** - defines the functions **usbh_aoa2_demo_init()** and **usbh_aoa2_demo_run()**.
- **usbh_aoa2_demo.c** - this includes the functions **usbh_aoa2_demo_init()** and **usbh_aoa2_demo_task_run()** shown below. It also contains a number of static functions - refer to the file for details of these.

## 6.1. Initialization Example

This example code initializes the AOAv2 class driver and all the required products.

```
void aoa2_demo_init ( void )
{
  int  rc = 0;

  rc = hcc_mem_init();

  if ( rc == 0 )
  {
    rc = usbh_init();
  }

  if ( rc == 0 )
  {
    rc = usbh_hc_init( 0, usbh_ehci_hc, 0 );
  }

  if ( rc == 0 )
  {
    rc = usbh_aoa2_init();
  }

  if ( rc == 0 )
  {
    rc = usbh_start();
  }

  if ( rc == 0 )
  {
    rc = usbh_hc_start( 0 );
  }

  if ( rc == 0 )
  {
    rc = usbh_aoa2_start();
  }

  if ( rc == 0 )
  {
    rc = usbh_aoa2_demo_init();
  }
} /* aoa2_demo_init */
```

## 6.2. usbh_aoa2_demo.c

This is the code from **usbh_aoa2_demo.c** that runs the demo task.

```
void usbh_aoa2_demo_task_run ( void )
{
  if ( usbh_aoa2_is_present() == TRUE )
  {
    switch ( s_demo_state )
    {
      case USBH_AOA2_DEMO_STATE_PENDING:
        break;

      case USBH_AOA2_DEMO_STATE_INIT:
        psp_printf( "Starting demo.\r\n" );
        s_demo_state = USBH_AOA2_DEMO_STATE_WRITE;
        break;

      case USBH_AOA2_DEMO_STATE_WRITE:
        demo_state_write( &s_demo_state );
        break;

      case USBH_AOA2_DEMO_STATE_WRITE_WAIT:
        demo_state_write_wait( &s_demo_state );
        break;

      case USBH_AOA2_DEMO_STATE_READ:
        demo_state_read( &s_demo_state );
        break;

      case USBH_AOA2_DEMO_STATE_READ_WAIT:
        demo_state_read_wait( &s_demo_state );
        break;

      case USBH_AOA2_DEMO_STATE_ERROR:
        break;
    } /* switch */
  }
  else
  {
    s_demo_state = USBH_AOA2_DEMO_STATE_INIT;
  }

#if ( OAL_TASK_POLL_MODE == 0 )
  oal_task_sleep( TIME_TO_SLEEP_MS );
#endif
} /* usbh_aoa2_demo_task_run */
```

This is the code from **usbh_aoa2_demo.c** that registers the callback functions for the connect and disconnect events:

```
int usbh_aoa2_demo_init ( void )
{
  int  result;

  result = usbh_aoa2_register_ntf( USBH_NTF_CONNECT, aoa2_device_connected );

  if ( result == USBH_SUCCESS )
  {
    result = usbh_aoa2_register_ntf( USBH_NTF_DISCONNECT, aoa2_device_disconnected );
  }

  return result;
}
```

# 7. Version

Version 1.00

For use with USBH  AOAv2 (Android Open Accessory) Class Driver versions 1.01 and above