APPLICATION NOTE

Atmel AVR4960: USB Host Android Accessory

32-bit Atmel Microcontrollers

Features

- Control an accessory from an Android device
- Send data to and from an Android device to an accessory
- Supported development kits: Atmel EVK1104 and EVK1105
- Code examples for IAR and GCC Compiler

Introduction

Android smartphones represent nowadays (as of early 2012) more than 50% of a 650 million units per year market. Connecting an electronic device to an Android phone is becoming more and more popular. Examples range from sports equipment to medical devices, toys, audio systems, etc.

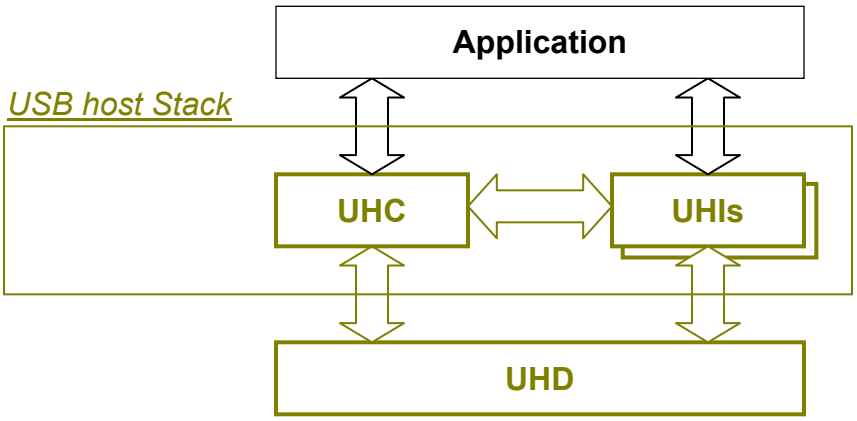
This document introduces the Android Open Accessory Protocol for Atmel® microcontrollers. The library and examples are included in the Atmel AVR® Software Framework (ASF) to provide the customer with a quick and easy way to get started with developing an Android accessory. The existing demos are described in the Atmel AVR32848: Android Accessory Demo application note.

Table of Contents

1. Organization

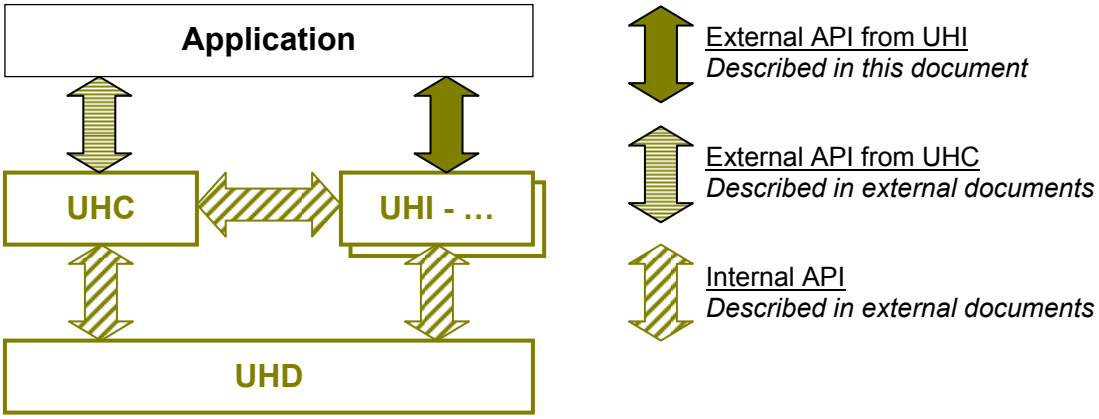
1.1 Overview

The library integrates into the USB Host Stack V2 described in Atmel AVR4950: ASF – USB Host Stack application note. It covers the UHI layer and keeps the interaction between the application and the UHC to a minimum. To fully understand the concept of the USB Host stack it is recommended to read the Atmel AVR4950 application note, which explains the structure.



2. Application Programming Interface (API)

This chapter describes the API between the application and the UHI.



The full API from the application to the UHC is described in the AVR4950 application note.

2.1 External API form UHI

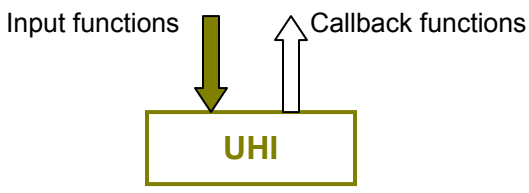


Table 2-1. API description.

Declaration	Description
uhi_aoa_read (uint8_t*payload, uint_ payload_size, uhd_callback_trans_t callback_end)	Read data from the Andriod device
uhi_aoa_write (uint8_t*payload, uint16_t payload_size, uhd_callback_trans_t callback_end)	Send data to the Android device

3. Using the Android Open Accessory service

This chapter covers what is necessary to use the Android Open Accessory service integrated in ASF. Like any other service it can be imported using the ASF wizard. Search for USB to find the “USB Host Android Accessory Protocol” service. To use the service to connect to an Android device some configuration needs to be done.

3.1 Setting up the clock

The Clock settings must provide proper clocks to the USB modules of the chip to function correctly. The clock is specified in the datasheet of the corresponding microcontroller. The configuration of the clocks for the UC3A0 and the UC3A3 are described briefly in the next chapters.

3.1.1 Clock Settings for the UC3A0 on the Atmel EVK1105

Extract from the UC3A0 datasheet: “The USB controller requires a 48MHz ±0.25% reference clock […].” This means we have to provide the proper clock internally to the USB Module. This will take care of generating the clock for full-speed and low-speed connections.

Here is one way how to use the conf_clock.h file to use with ASF to provide the proper clock to the USB Module.

conf_clock.h

```
/* ===== System Clock Source Options */
/* #define CONFIG_SYSCLK_SOURCE       SYSCLK_SRC_RCSYS */
#define CONFIG_SYSCLK_SOURCE         SYSCLK_SRC_OSC0
/* #define CONFIG_SYSCLK_SOURCE       SYSCLK_SRC_PLL0 */

/* ===== PLL0 Options */
/* #define CONFIG_PLL0_SOURCE         PLL_SRC_OSC0 */
/* #define CONFIG_PLL0_SOURCE         PLL_SRC_OSC1 */
/* #define CONFIG_PLL0_MUL            4 */
/* #define CONFIG_PLL0_DIV            1 */

/* ===== PLL1 Options */
#define CONFIG_PLL1_SOURCE           PLL_SRC_OSC0
/* #define CONFIG_PLL1_SOURCE         PLL_SRC_OSC1 */
#define CONFIG_PLL1_MUL              8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL1_DIV              2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

/* ===== System Clock Bus Division Options */
/* #define CONFIG_SYSCLK_CPU_DIV      0 */ /* Fcpu = Fsys/(2 ^ CPU_div) */
/* #define CONFIG_SYSCLK_PBA_DIV      0 */ /* Fpba = Fsys/(2 ^ PBA_div) */
/* #define CONFIG_SYSCLK_PBB_DIV      0 */ /* Fpbb = Fsys/(2 ^ PBB_div) */

/* ===== Peripheral Clock Management Options */
/* #define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | \
   (1 << SYSCLK_OCD)) */
/* #define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0) */
/* #define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX) */
/* #define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB) */
```

```
/* ===== USB Clock Source Options */
/* #define CONFIG_USBCLK_SOURCE      USBCLK_SRC_OSC0 */
/* #define CONFIG_USBCLK_SOURCE      USBCLK_SRC_PLL0 */
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV           1 /* Fusb = Fsys/(2 ^ USB_div) */
```

The first highlighted line specifies that OSC0 is used which is a 12MHz clock in case of the Atmel EVK1105. This means we have to use the PLL to convert the 12MHz clock into a 48MHz clock to feed it into the USB module. By using a multiplier of 8 and a divider of 2 we get a frequency of 48MHz. The last two lines make sure that this clock generated by the PLL1 is used by the USB module.

3.1.2 Clock Settings for the Atmel AVR UC3A3 on the Atmel EVK1104

Extract from the UC3A3 datasheet: “The UTMI transceiver requires an external 12MHz clock as a reference to its internal 480MHz PLL”. This means we have to provide the proper clock internally to the USB Module. This will take care of generating the clock for high-speed, full-speed and low-speed connections.

Here is one way how to use the conf_clock.h file to use with ASF to provide the proper clock to the USB Module.

conf_clock.h

```
/* ===== System Clock Source Options */
/*#define CONFIG_SYSCLK_SOURCE      SYSCLK_SRC_RCSYS */
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC0
/*#define CONFIG_SYSCLK_SOURCE      SYSCLK_SRC_PLL0 */

/* ===== PLL0 Options */
#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
/*#define CONFIG_PLL0_SOURCE        PLL_SRC_OSC1 */
#define CONFIG_PLL0_MUL             11 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL0_DIV             2  /* Fpll = (Fclk * PLL_mul) / PLL_div */

/* ===== PLL1 Options */
/*#define CONFIG_PLL1_SOURCE        PLL_SRC_OSC0 */
/*#define CONFIG_PLL1_SOURCE        PLL_SRC_OSC1 */
/*#define CONFIG_PLL1_MUL           8 */
/*#define CONFIG_PLL1_DIV           2 */

/* ===== System Clock Bus Division Options */
#define CONFIG_SYSCLK_CPU_DIV       0 /* Fcpu = Fsys/(2 ^ CPU_div) */
#define CONFIG_SYSCLK_PBA_DIV       0 /* Fpba = Fsys/(2 ^ PBA_div) */
/*#define CONFIG_SYSCLK_PBB_DIV     0 */ /* Fpbb = Fsys/(2 ^ PBB_div) */

/* ===== Peripheral Clock Management Options */
/*#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | \
      (1 << SYSCLK_OCD)) */
/*#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0) */
/*#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX) */
/*#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB) */

/* ===== USB Clock Source Options */
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
/*#define CONFIG_USBCLK_SOURCE      USBCLK_SRC_PLL0 */
/*#define CONFIG_USBCLK_SOURCE      USBCLK_SRC_PLL1 */

#define CONFIG_USBCLK_DIV           1 /* Fusb = Fsys/(2 ^ USB_div) */
```

Since we only need to provide a 12MHz clock and the Atmel EVK1104 has a 12MHz clock as OSC0 we can feed this clock directly into the USB Module with a divisor of 1.

3.2 Connection process

3.2.1 Identifying string information

After setting up the clock the programmed device should be able to connect to the Android device. By default, the identifying information of an Atmel EVK1105 will be used. If another combination of board and Android application should be used it is necessary to provide the proper strings as defines.

Identifying string information defines

```
#define AOA_STRING_MANUFACTURER  "Atmel"
#define AOA_STRING_MODEL         "EVK1105"
#define AOA_STRING_DESCRIPTION   "EVK1105 Application Board"
#define AOA_STRING_VERSION       "1.0"
#define AOA_STRING_URL           "http://www.Atmel.com/Images/AVR32848"
#define AOA_STRING_SERIAL        "0000000012345678"
```

Important for the launch of an Android App are the strings manufacturer, model and version as they are used to determine if an existing app on the Android side supports the board. The description is used to give the user some more information about the accessory and the URL can be provided to point the user to a place to download a suitable app or get more information on the accessory. Please keep in mind that the description and the URL are only evaluated if no suited app is installed on the Android device.

3.2.2 Callbacks

The library does not provide callbacks to the application level. All necessary callbacks are handled by the UHC. Please refer to the Atmel AVR4950: ASF – USB Host Stack application note on how to use these callbacks.

Appendix A. Revision history

Doc. Rev.	Date	Comments
42003C	8/2012	New template and layout. One link is corrected
42003B	6/2012	Corrected to 32-bit, added some short-cuts and made some minor corrections in the text
42003A	5/2012	Initial document release

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370