PROGRAMMING GUIDE

VAB-600 SmartETK_SDK

1.01-08032015-110100

Revision History

Version	Date	Remarks
1.00	2015/07/22	Initial external release.
1.01	2015/07/31	Removed Smart ETK version

Table of Contents

1. Introduction

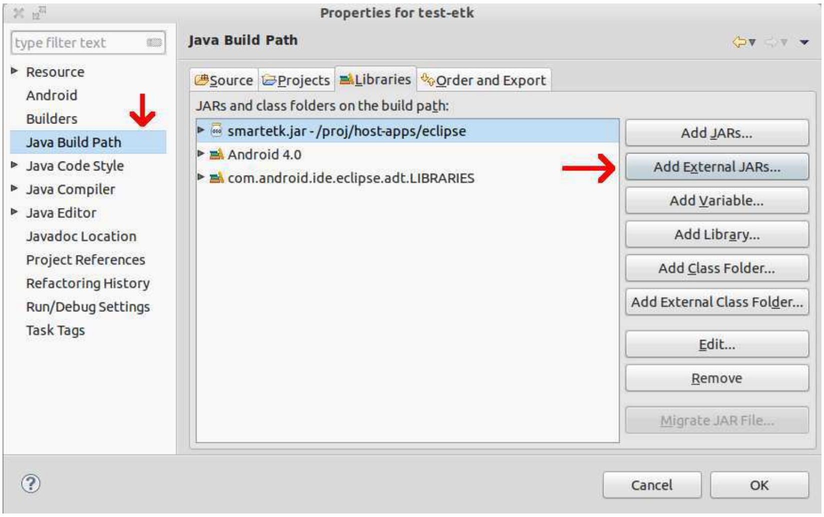
Smart ETK is programmed with the socket IO as a communication between JAVA and C language to control the hardware modules. We implemented the board support service such as bss_vab600 to meet the request from Smart ETK API.

VAB-600 is designed with enhanced features including Watch Dog, RTC Power On, Programmable GPIO, and Serial Ports.

2. Develop the environment

Eclipse is an officially recommended and integrated development environment (IDE) for the Android application. It requires importing the specific reference Java Archive file in order to develop VAB-600 enhance features.

Open Eclipse IDE and create an Android project. In project properties, import the smartetk.jar into your project by clicking "**Add External JARs**...".


Properties for test-etk

Java Build Path

JARs and class folders on the build path:

- smartetk.jar - /proj/host-apps/eclipse
- Android 4.0
- com.android.ide.eclipse.adt.LIBRARIES

2.1. SDK Class Definitions

All functions and values are placed in class named **com.via.vepd.SmartETK**. Import this package into Java code to use them.

GPIO PIN definitions:

ID	0	1	2	3	4	5	6	7
GPIO	20	21	22	23	24	25	26	27

2.2. Function Return Values

Other than the Boolean TRUE and FALSE return values, there are eight other types of return values found throughout the Smart ETK SDK API.

SmartETK.S_OK

Description:	The S_OK return value has the constant value 0. When a function returns the S_OK value, it indicates that the function has successfully completed

SmartETK.E_FAIL

Description:	The E_FAIL return value has the constant value -1000. When a function returns the E_FAIL value, it indicates that the function has failed to complete

SmartETK.E_BOARD_NOT_SUPPORT

Description:	The E_BOARD_NOT_SUPPORT return value has the constant value -1001. When a function returns the E_BOARD_NOT_SUPPORT value, it indicates that the mainboard is not supported

SmartETK.E_ALREADY_INITIALIZED

Description:	The E_ALREADY_INITIALIZED return value has the constant value -1002. When a function returns the E_ALREADY_INITIALIZED value, it indicates that libbssjni.so has already been loaded

SmartETK.E_INVALID_ARG

Description:	The E_INVALID_ARG return value has the constant value -1003. When a function returns the E_INVALID_ARG value, it indicates that the arguments are invalid

SmartETK.E_OUT_OF_MEMORY

Description:	The E_OUT_OF_MEMORY return value has the constant value -1004. When a function returns the E_OUT_OF_MEMORY value, it indicates that there is insufficient memory

SmartETK.E_FUNC_NOT_SUPPORT

Description:	The E_FUNC_NOT_SUPPORT return value has the constant value -1005. When a function returns the E_FUNC_NOT_SUPPORT value, it indicates that the function is not supported on this board

SmartETK.E_HARDWARE_ERROR

Description:	The E_HARDWARE_ERROR return value has the constant value -1006. When a function returns the E_HARDWARE_ERROR value, it indicates that there is a hardware error

2.3. API List

SmartETK.Init

Syntax:	static int Init(void);
Description:	This function initializes the Smart ETK and loads the JNI.
Parameters:	None
Return:	S_OK – if the function succeeded E_FAIL – if the function fails

SmartETK. SupportGpio

Syntax:	static int SupportGpio( );
Description:	Check if current platform supports GPIO.
Parameters:	None.
Return:	S_OK – if it supports GPIO. E_FUNC_NOT_SUPPORT – if not support GPIO.

SmartETK.SupportWdt

Syntax:	static int SupportWdt();
Description:	Check if current platform supports watch dog timer
Parameters:	None
Return:	S_OK – if it supports watch dog timer E_FUNC_NOT_SUPPORT – if it does not support watch dog timer

SmartETK. SupportRtcWake

Syntax:	static int SupportRtcWake( );
Description:	Check if current platform supports RTC wake-up.
Parameters:	None.
Return:	S_OK – if it support RTC wake-up. E_FUNC_NOT_SUPPORT – if it does not support RTC wake-up.

SmartETK. Gpio_GetInfo

Syntax:	GpioInfo Gpio_GetInfo( );
Description:	This function gets the information of GPIO.
Parameters:	None.
Return:	A SmartETK.GpioInfo object that containing an array representing GPIO.

SmartETK. Gpio_Enable

Syntax:	static int Gpio_Enable(int pinId, boolean enable);
Description:	Enable specific GPIO pin.
Parameters:	pinId – ID of GPIO pin. Refer 5.2 pin definition enable – true for enable, false for disable.
Return:	S_OK – if the function succeeded. E_FAIL – if the function fails.

SmartETK. Gpio_Set

Syntax:	public static int Gpio_Set(int pinId, int inOut, int upDown);
Description:	Setup I/O configuration for specific GPIO pin.
Parameters:	pinId: ID of GPIO pin. Refer 5.2 pin definition inOut: SmartETK.GM_GPI for input, SmartETK.GM_GPO for output. upDown: SmartETK.GM_NO_PULL for disable internal pull, SmartETK.GM_PULL_UP for pull up, SmartETK.GM_PULL_DOWN for pull down.
Return:	S_OK – if the function succeeded. E_FAIL – if the function fails.

SmartETK. Gpio_Write

Syntax:	static int Gpio_Write(int pinId, int pinVal);
Description:	Set GPIO output signal.
Parameters:	pinId – ID of GPIO pin. Refer 5.2 pin definition pinVal – GPIO signal, 0 for logic low, 1 for logic high.
Return:	S_OK – if the function succeeded. E_FAIL – if the function fails.

SmartETK. Gpio_Read

Syntax:	static int Gpio_Read(int pinId, int pinVal[]);
Description:	Get GPIO input signal.
Parameters:	pinId – ID of GPIO pin. Refer 5.2 pin definition pinVal – A single-element array returns GPIO signal. 0 for logic low, 1 for logic high. Caller should allocate array space manually.
Return:	S_OK – if the function succeeded. E_FAIL – if the function fails.

SmartETK.Wdt_SetTimer

Syntax:	static int Wdt_SetTimer(int sec);
Description:	Set watch dog timer to specific time in second. The time should not exceed the number defined by a constant SmartETK.WDT_MAX_SEC.
Parameters:	sec – Time in second to reset.
Return:	S_OK – if the function succeeded. E_FAIL – if the function fails.

SmartETK.Wdt_Refresh

Syntax:	static int Wdt_Refresh();
Description:	Refresh the timer to the time set in Wdt_SetTimer.
Parameters:	None
Return:	S_OK – if the function succeeded. E_FAIL – if the function fails.

SmartETK.Wdt_Start

Syntax:	static int Wdt_Start();
Description:	Start watch dog timer. If application does not call Wdt_Refresh in time set in Wdt_SetTimer , system will be reset.
Parameters:	None
Return:	S_OK – if the function succeeded. E_FAIL – if the function fails.

SmartETK.Wdt_Stop

Syntax:	static int Wdt_Stop();
Description:	Stop watch dog timer.
Parameters:	None
Return:	S_OK – if the function succeeded. E_FAIL – if the function fails.

SmartETK. Rtc_WakeupEnable

Syntax:	Rtc_WakeupEnable( );
Description:	Enable RTC wake-up.
Parameters:	None
Return:	S_OK – if the function succeeded. E_FAIL – if the function fails.

SmartETK. Rtc_WakeupDisable

Syntax:	Rtc_WakeupDisable( );
Description:	Disable RTC wake-up.
Parameters:	None.
Return:	S_OK – if the function succeeded. E_FAIL – if the function fails.

SmartETK. Rtc_SetWakeupTime

Syntax:	static int Rtc_SetWakeupTime(int mode, int yr, int mm, int dd, int hr, int min, int sec);
Description:	Set time to wake up by RTC.
Parameters:	mode – SmartETK.RWM_HHMM : Wake up when hour and minute match. SmartETK.RWM_MONTH : Wake up when month day, hour and minute match. SmartETK.RWM_WEEK : Wake up when week day, hour and minute match. yr – Year to wake up. mm – Month to wake up. dd – Day to wake up. hr – Hour to wake up. min – Minute to wake up. sec – Second to wake up.
Return:	S_OK – if the function succeeded. E_FAIL – if the function fails.

SmartETK. Uart_GetConfig

Syntax:	native public static UartConfig Uart_GetConfig(FileDescriptor fd);
Description:	Return terminal configuration of specific FileDescriptor.
Parameters:	fd – FileDescriptor of opened file, must be a device file (under /dev).
Return:	UartConfig – if the function succeeded, see section 4.4 for detail. null – if the function fails.

SmartETK. Uart_SetConfig

Syntax:	native public static int Uart_SetConfig(FileDescriptor fd, UartConfig cfg);
Description:	Set terminal configuration of specific FileDescriptor.
Parameters:	fd – FileDescriptor of opened file, must be a device file (under /dev). cfg – Terminal configuration, see section 4.4 for detail.
Return:	0 – if the function succeeded. 1 – if the function fails.

2.4. UART Parameters

SmartETK defines UART (Universal Asynchronous Receiver/Transmitter) related values into another class com.via.UartConfig for configuration.

Instead of one-time setup for other devices, UART is transferring data continually. Developers may blocked read, selected read, or buffered read from UART. In order to provide developers all types of transmissions, the APIs should be based on Java class java.io.FileDescriptor.

To get FileDescriptor, the developers have to open the device file manually (e.g. /dev/ttyS1 or /dev/ttyUSB0) with Java class java.io.File. Then select a suitable Java transmission helper class (e.g. java.io.FileOutputStream).

To setup UART parameters with FileDescriptor and class UartConfig which are provided by VIA. Please check SDK sample code for details. Parameters in UartConfig are the same as the values in Linux Kernel. Developers who are familiar with terminal programming in C may apply the same macro to port existing applications to Java with UartConfig. See Linux Programmer's Manual for details. To read the manual, enter "man tcsetattr" in ubuntu terminal.

Currently, four connection options (in struct termios) and two speed options for Linux are available.

The four connection options are: c_cflag, c_iflag, c_oflag, c_lflag

The two speed options are: ispeed, ospeed

Four options will be applied with tcsetattr in Linux C library. c_cc[VMIN] is set to 1 and c_cc[VTIME] is set to 0. ispeed and ospeed in UartConfig will be applied to struct termios with cfsetispeed and cfsetospeed in Linux C library.

All options have to be set before applying them. It is recommended to get current configurations of FileDescriptor into UartConfig via SmartETK.Uart_GetConfig instead of allocating one manually.

All types of file transmissions in Java are available since we shared existing FileDescriptor class. After FileDescriptor is configured, developers can access UART as a normal file.

Taiwan Headquarters
1F, 531 Zhong-Zheng Road
Xindian, Taipei, 23148
Taiwan

TEL: 886.2.2218.5452
FAX: 886.2.2218.5453
Email: embedded@via.com.tw

USA
940 Mission Court
Fremont, CA 94539
USA

TEL: 1.510.683.3300
FAX: 1.510.687.4654
Email: embedded@viatech.com

Europe
In den Dauen 6
53117 Bonn
Germany

TEL: 49.228.688565.0
FAX: 49.228.688565.19
Email: embedded@via-tech.eu

China
Tsinghua Science Park Bldg. 7
No. 1 Zongguancun East Road
Haiden District, Beijing, 100084
China

TEL: 86.10.59852288
FAX: 86.10.59852299
Email: embedded@viatech.com.cn

Japan
3-15-7 Ebisu MT Bldg. 6F
Higashi, Shibuya-ku
Tokyo 150-0011
Japan

TEL: 81.3.5466.1637
FAX: 81.3.5466.1638
Email: embedded@viatech.co.jp