



PROGRAMMING GUIDE

VAB-630

Smart ETK SDK

v1.0.3



Copyright

Copyright © 2017 VIA Technologies Incorporated. All rights reserved.

No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise without the prior written permission of VIA Technologies, Incorporated.

Trademarks

All brands, product names, company names, trademarks and service marks are the property of their respective holders.

Disclaimer

VIA Technologies makes no warranties, implied or otherwise, in regard to this document and to the products described in this document. The information provided in this document is believed to be accurate and reliable as of the publication date of this document. However, VIA Technologies assumes no responsibility for the use or misuse of the information (including use or connection of extra device/equipment/add-on card) in this document and for any patent infringements that may arise from the use of this document. The information and product specifications within this document are subject to change at any time, without notice and without obligation to notify any person of such change.

VIA Technologies, Inc. reserves the right to make changes to the products described in this manual at any time without prior notice.



Revision History

Version	Date	Remarks
1.00	08/14/2017	Initial release



Table of Contents

1. Introduction	1
2. Install Development Environment	1
3. Smart ETK SDK API.....	5
3.1. Function Return Values.....	5
3.2. Initializes.....	7
3.2.1. SmartETK Initializes	7
3.3. GPIO	8
3.3.1. Support GPIO.....	8
3.3.2. Enable GPIO Pin	8
3.3.3. Set GPIO Pin Direction.....	9
3.3.4. Set GPO Value.....	9
3.3.5. Get GPI Value	10
3.4. UART	11
3.4.1. Get Configurations of UART Device.....	11
3.4.2. Set Configurations of UART Device.....	11
3.4.3. UartConfig Parameters.....	12
3.5. Watchdog Timer	13
3.5.1. Support Watchdog Timer.....	13
3.5.2. Set Timeout	13
3.5.3. Refresh Watchdog Timer	13
3.5.4. Start Watchdog Timer	14
3.5.5. Stop Watchdog Timer	14



1. Introduction

This section provides an overview of the Smart ETK program for the VAB-630 board which has enhance features for the board including, APIs for GPIO and UART (RS-232) function and the Watchdog Timer.

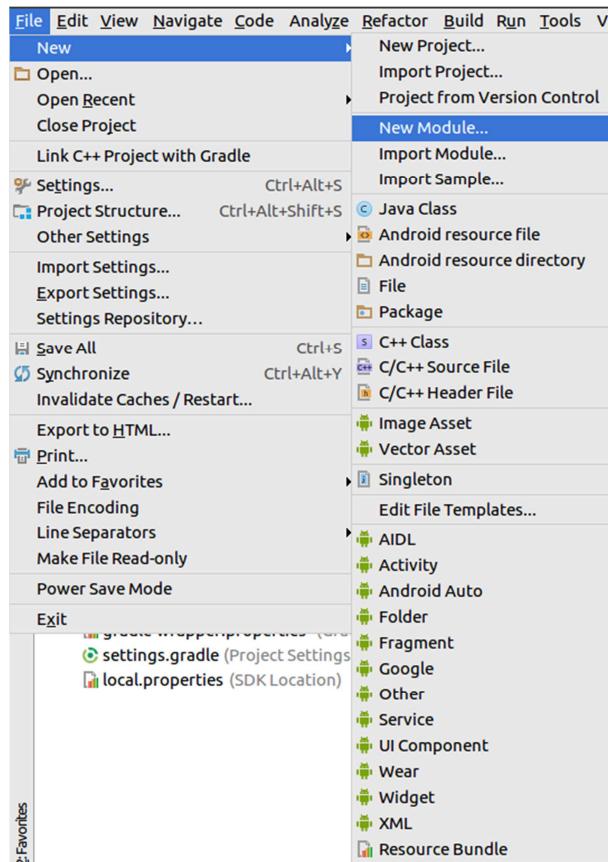
The Smart ETK is programmed with a binder as a communication between Java and C language to control the hardware modules. The bbservice (Board Support Service) is implemented in order to perform a requested function from a Smart ETK API.

2. Install Development Environment

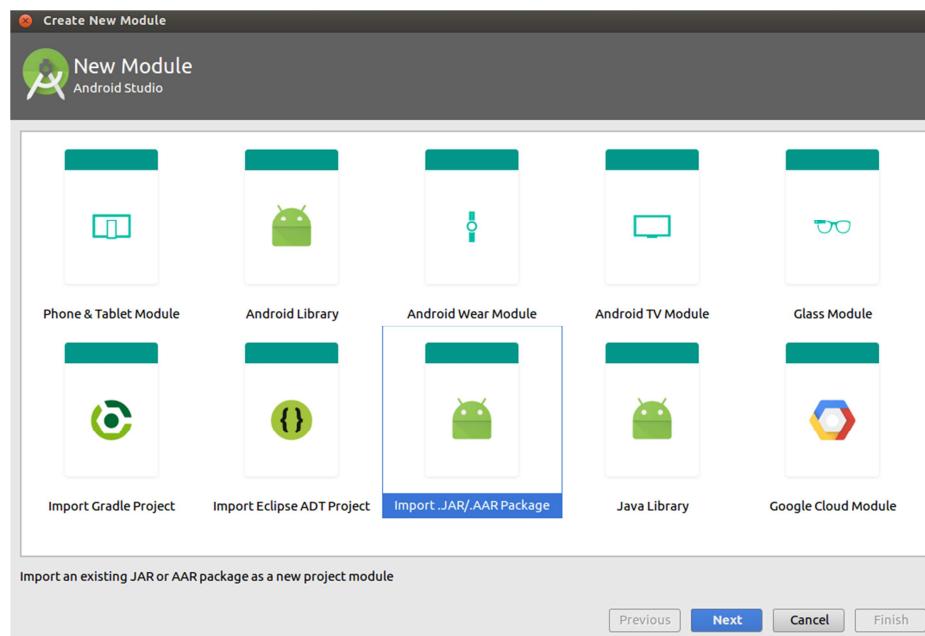
The Android Studio is an officially recommended and integrated development environment (IDE) in place of the old Eclipse for the Android application. It requires importing the specific JAR library, which can be extracted from the Smart_ETK_v1.0.3_SourceCode.zip, in order to integrate the Smart ETK enhanced features in your development.

After creating a new project in the Android Studio, import the JAR library from the Smart ETK to the project. Please follow the steps below to import the JAR library.

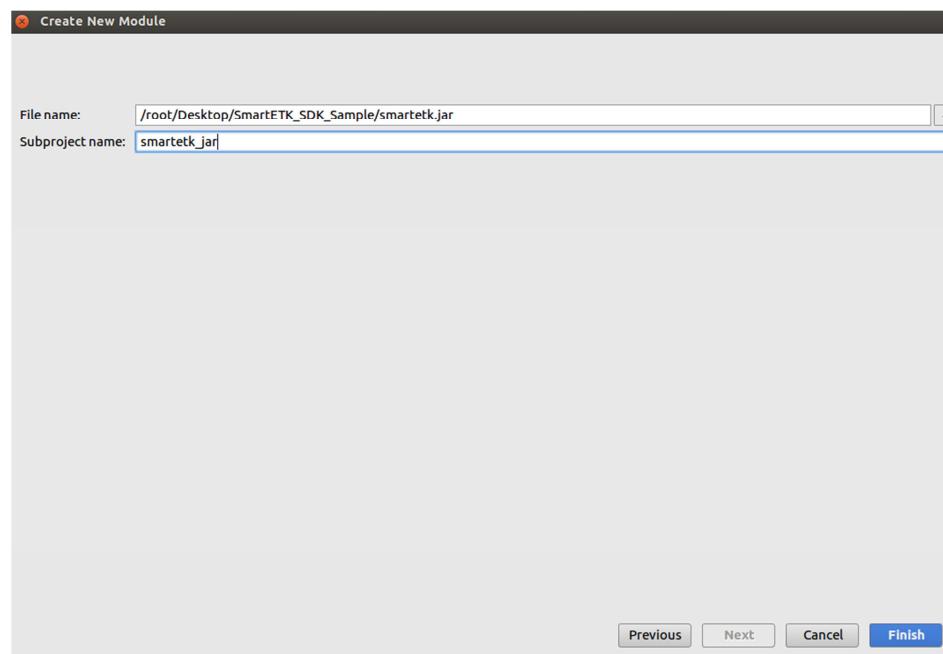
1. Start Android Studio and create a new module in the project.



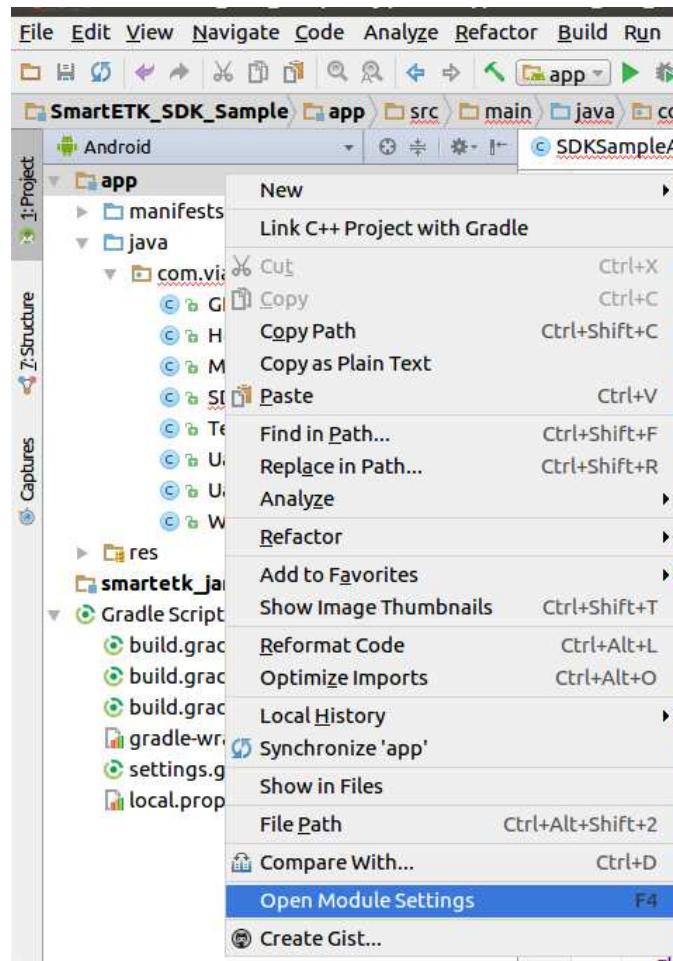
2. Choose **Import.JAR/.AAR Package** in the pop-up window and then click Next.



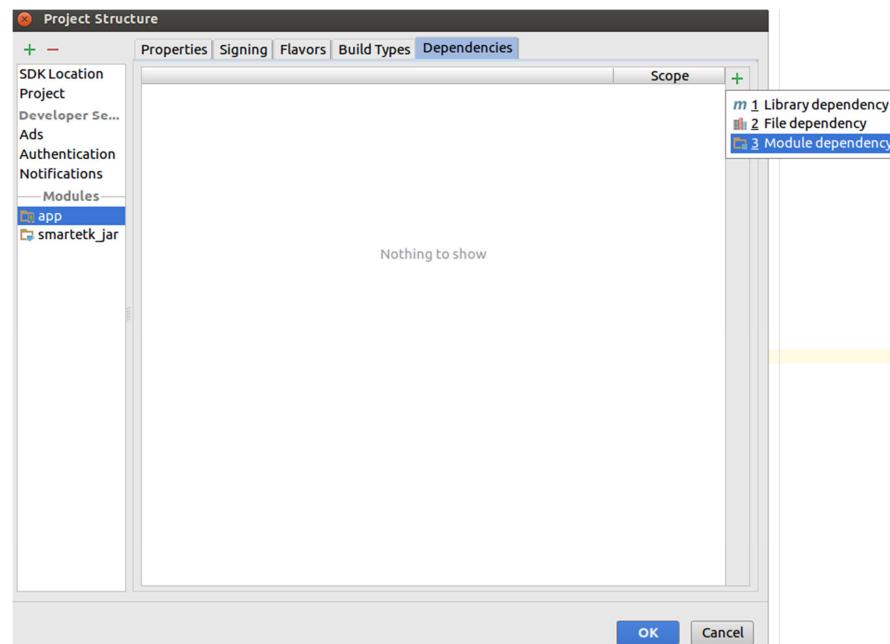
3. Specify the file name (SmartETK_SDK_Sample) and path of the JAR library, and then click Finish.



4. Open the module settings.

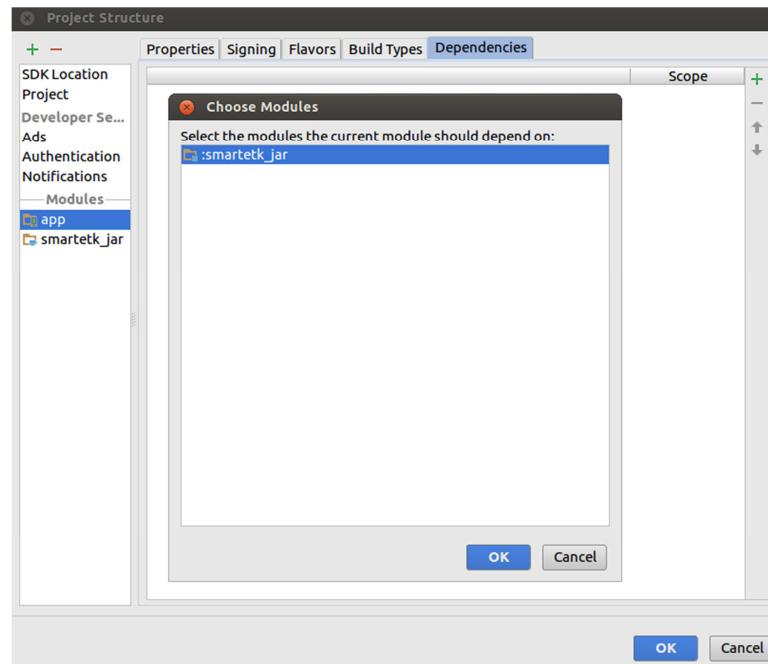


5. Select number 3. Module dependency in the pop-up window.

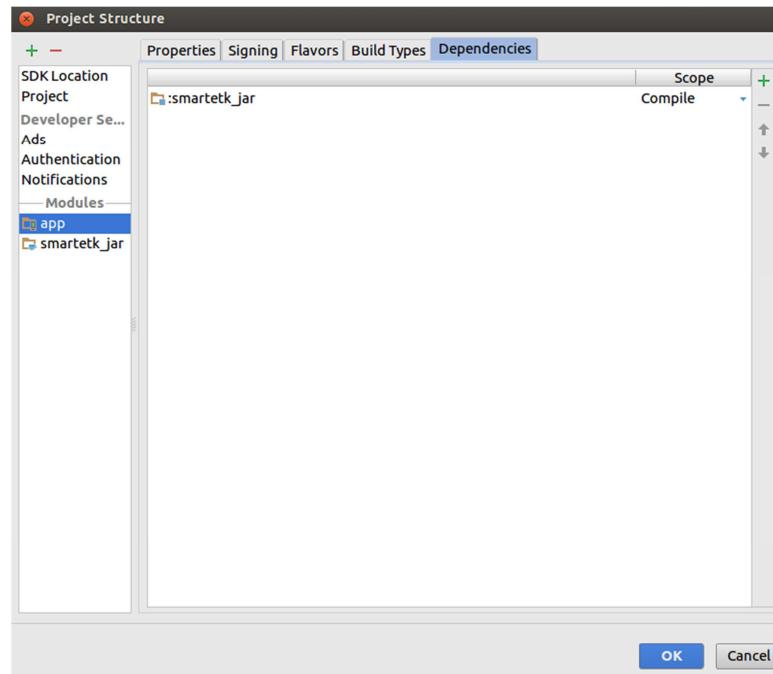




6. Choose the module to import the :smartetk_jar.



7. After importing the module (:smartetk_jar), it will be shown in the window, then click OK to complete this setting.





3. Smart ETK SDK API

This section explains three types of function that are found in the smartetk.jar. The smartetk.jar should be imported in the Smart ETK demo APP source code.

The GPIO, Watchdog timer and UART functions and variables are placed in Java package named “com.via.SmartETK”. Import this package into Java code in order to use them.

3.1. Function Return Values

These are some types of return values throughout the Smart ETK SDK API.

Static Int Variable	Return Value	Description
SmartETK.S_OK	0	When a function returns the S_OK, it indicates that the function has been completed successfully.
SmartETK.E_FAIL	-1000	When a function returns the E_FAIL, it indicates that the function has failed to complete.
SmartETK.E_BOARD_NOT_SUPPORT	-1001	When a function returns the E_BOARD_NOT_SUPPORT, it indicates that the mainboard is not supported.
SmartETK.E_ALREADY_INITIALIZED	-1002	When a function returns the E_ALREADY_INITIALIZED, it indicates that libbssjni.so has already been loaded.
SmartETK.E_INVALID_ARG	-1003	When a function returns the E_INVALID_ARG, it indicates that the arguments are invalid.
SmartETK.E_OUT_OF_MEMORY	-1004	When a function returns the E_OUT_OF_MEMORY, it indicates that there is insufficient memory.



SmartETK.E_FUNC_NOT_SUPPORT	-1005	When a function returns the E_FUNC_NOT_SUPPORT, it indicates that the function is not supported on this board.
SmartETK.E_HARDWARE_ERROR	-1006	When a function returns the E_HARDWARE_ERROR, it indicates that there is a hardware error.



3.2. Initializes

This section explains how to use the Initializes object and its functions.

3.2.1. SmartETK Initializes

Description:

This function initializes the Smart ETK and loads the JNI.

Syntax:

```
static int Init(void);
```

Parameters:

None

Return:

[S_OK](#) – if the function succeeded

[E_FAIL](#) – if the function fails



3.3. GPIO

This section explains how to use the GPIO object and its functions. The GPIO pin table is defined as follows.

ID	1	2	3	4	5	6	7	8	9	10
Physical GPIO	1	2	3	5	6	7	8	12	13	14

Definition of GPIO pins

***Note:** If the specified ID is out of range, the GPIO APIs return a negative error number.

3.3.1. Support GPIO

Description:

Check if current platform supports GPIO.

Syntax:

```
static int SupportGpio( );
```

Parameters:

None

Return:

`S_OK` – if it supports GPIO.

`E_FUNC_NOT_SUPPORT` – if not support GPIO.

3.3.2. Enable GPIO Pin

Description:

Enable specific GPIO pin.

Syntax:

```
static int Gpio_Enable(int pinId, boolean enable);
```

Parameters:

`pinId` – ID of GPIO pin. Please refer to Definition of GPIO pins.

`enable` – true for enable, false for disable.

Return:

`S_OK` – if the function succeeded.

`E_FAIL` – if the function fails.



3.3.3. Set GPIO Pin Direction

Description:

Set input/output direction for specified GPIO pin.

Syntax:

```
public static int Gpio_Set(int pinId, int inOut, int upDown);
```

Parameters:

`pinId` – ID of GPIO pin. Please refer to Definition of GPIO pins.

`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.

3.3.4. Set GPO Value

Description:

Set output signal for specified GPIO pin.

Syntax:

```
static int Gpio_Write(int pinId, int pinVal);
```

Parameters:

`pinId` – ID of GPIO pin. Please refer to Definition of GPIO pins.

`pinVal` – GPIO signal, 0 for logic low, 1 for logic high.

Return:

`S_OK` – if the function succeeded.

`E_FAIL` – if the function fails.



3.3.5. Get GPI Value

Description:

Get input signal for specified GPIO pin.

Syntax:

```
static int Gpio_Read(int pinId, int pinVal[]);
```

Parameters:

`pinId` – ID of GPIO pin. Please refer to Definition of GPIO pins.

`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.



3.4. UART

This section explains how to use the UART object and its functions.

3.4.1. Get Configurations of UART Device

Description:

Return terminal configuration of specific FileDescriptor.

Syntax:

```
native public static UartConfig Uart_GetConfig(FileDescriptor fd);
```

Parameters:

fd – FileDescriptor of opened file, must be a device file (under /dev).

Return:

UartConfig – if the function succeeded, see section 3.6.3 for detail.

null – if the function fails.

3.4.2. Set Configurations of UART Device

Description:

Set terminal configuration of specific FileDescriptor.

Syntax:

```
native public static int Uart_SetConfig(FileDescriptor fd, UartConfig cfg);
```

Parameters:

fd – FileDescriptor of opened file, must be a device file (under /dev).

cfg – Terminal configuration, see section 3.6.3 for detail.

Return:

0 – if the function succeeded.

1 – if the function fails.



3.4.3. UartConfig Parameters

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

UART is transferring data continually; please use blocking-read, selected-read, or buffered-read to access UART. In order to provide all types of transmissions, the APIs should be designed base on Java class `java.io.FileDescriptor`.

To get `FileDescriptor`, 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 the SDK sample code for details. Parameters in `UartConfig` are the same as the values in Linux Kernel. If using programming C, apply the same parameters to map the existing application to Java with `UartConfig`. Please refer to the 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 the 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, the UART can be accessed as a normal file.



3.5. Watchdog Timer

This section explains how to use the Watchdog timer object and its functions.

3.5.1. Support Watchdog Timer

Description:

Check if current platform supports watchdog timer.

Syntax:

```
static int SupportWdt();
```

Parameters:

None

Return:

`S_OK` – if it supports watchdog timer.

`E_FUNC_NOT_SUPPORT` – if it does not support watchdog timer.

3.5.2. Set Timeout

Description:

Set watchdog timer to specific time in second. The time should not exceed the number defined by a constant `SmartETK.WDT_MAX_SEC`.

Syntax:

```
static int Wdt_SetTimer(int sec);
```

Parameters:

`sec` – Time in second to reset.

Return:

`S_OK` – if the function succeeded.

`E_FAIL` – if the function fails.

3.5.3. Refresh Watchdog Timer

Description:

Refresh the timer to the time set in `Wdt_SetTimer`.

Syntax:

```
static int Wdt_Refresh();
```



Parameters:

None

Return:

S_OK – if the function succeeded.

E_FAIL – if the function fails.

3.5.4. Start Watchdog Timer

Description:

Start watchdog timer. If application does not call Wdt_Refresh in time set in Wdt_SetTimer , system will be reset.

Syntax:

```
static int Wdt_Start();
```

Parameters:

None

Return:

S_OK – if the function succeeded.

E_FAIL – if the function fails.

3.5.5. Stop Watchdog Timer

Description:

Stop watchdog timer.

Syntax:

```
static int Wdt_Stop();
```

Parameters:

None

Return:

S_OK – if the function succeeded.

E_FAIL – if the function fails.



 Taiwan Headquarters
1F, 531 Zhong-zheng Road,
Xindian Dist., New Taipei City 231
Taiwan

Tel: 886-2-2218-5452
Fax: 886-2-2218-9860
Email: embedded@via.com.tw

 USA
940 Mission Court
Fremont, CA 94539,
USA

Tel: 1-510-687-4688
Fax: 1-510-687-4654
Email: embedded@viatech.com

 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

 China
Tsinghua Science Park Bldg. 7
No. 1 Zongguancun East Road,
Haidian Dist., Beijing, 100084
China

Tel: 86-10-59852288
Fax: 86-10-59852299
Email: embedded@viatech.com.cn

 Europe
Email: embedded@via-tech.eu