Programming Guide

VIA Smart ETK SDK
Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.1</td>
<td>2014/04/03</td>
<td>Initial draft – SmartETK v0.0.11</td>
</tr>
<tr>
<td>0.0.2</td>
<td>2014/08/01</td>
<td>Add UART module and remove network Ethernet functions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modify watch dog scenario – SmartETK v0.0.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- add timeout and keep alive functions</td>
</tr>
</tbody>
</table>

1. Introduction ........................................................................................................1
2. System Requirements ..............................................................................................1
   2.1 Hardware Requirements ...................................................................................1
   2.2 Software Requirements ...................................................................................1
3. Install and Usage ....................................................................................................2
   3.1 Installation on development computer .........................................................2
   3.2 Installation on target board ............................................................................3
4. Smart ETK SDK API ..................................................................................................4
   4.1 Class definitions ...............................................................................................4
   4.2 Function return values.......................................................................................4
      SmartETK.S_OK .....................................................................................................4
      SmartETK.E_FAIL .................................................................................................4
      SmartETK.E_VERSION_NOT_SUPPORT .....................................................................4
      SmartETK.E_INVALID_ARG ...............................................................................5
      SmartETK.E_FUNC_NOT_SUPPORT .......................................................................5
      SmartETK.E_CONNECTION_FAIL ..........................................................................5
      SmartETK.E_NOT_RESPOND_YET .........................................................................5
      SmartETK.E_TIMEOUT .........................................................................................5
      SmartETK.E_UART_OPENFAIL .............................................................................5
      SmartETK.E_UART_NOT_OPEN .............................................................................5
      SmartETK.E_UART_ALREADY_OPENED ...................................................................6
      SmartETK.E_UART_TTY_BEEN_USED .....................................................................6
      SmartETK.E_UART_BAUDRATE_NOT_SUPPORT .....................................................6
4.3 Network Class ........................................................................................................7
   Network Class ........................................................................................................7
   Network.setWakeOnLan() ......................................................................................7
   Network.getWakeOnLan() ......................................................................................7
4.4 WatchDog Class ......................................................................................................9
   WatchDog Class ......................................................................................................9
   WatchDog.setEnable() .........................................................................................9
   WatchDog.getEnable() .........................................................................................9
   WatchDog.setTimeout() .........................................................................................10
   WatchDog.getTimeout() .......................................................................................10
   WatchDog.keepAlive() ........................................................................................11
4.5 RTC Class ..............................................................................................................12
   RTC Class .............................................................................................................12
   RTC.setWakeUpTime() ..........................................................................................13

8. Annex D: UART

7. Annex C: RTC

6. Annex B: Watch Dog

5. Annex A: Network

4.7 SystemETK Class

4.6 Uart Class

3. SystemETK.suspend()
1. Introduction

VIA Smart ETK SDK supports the hardware controlling API for Network, Watch Dog, RTC, and UART modules.

Smart ETK is programmed with the socket IO as the communication between JAVA and C language to control hardware modules. We implement the board support service like bss_vt6080 to listen the request from Smart ETK API. We bind 127.0.0.1 as the internal listening IP, to keep it from establishing the connection with the external network.

2. System Requirements

2.1 Hardware Requirements

VIA Smart ETK SDK is compatible with the following main board:

DS2 (VT6080) with Android BSP

2.2 Software Requirements

Microsoft Windows or Ubuntu Linux
Eclipse IDE with Android Development Tools (ADT) installed
3. Install and Usage

3.1 Installation on development computer

Open Eclipse IDE and create an Android project. In project properties, import SmartETK.jar by “Add External JARs...” into project.

Select “Order and Export” tab, move SmartETK.jar to the top and choose it.
3.2 Installation on target board

Install VIA released firmware and done.
4. Smart ETK SDK API

4.1 Class definitions

Network, Watch Dog, RTC, and UART modules are placed in class named `com.viaembedded.smartetk`, and returned values is placed in class named `com.viaembedded.smartetk`. Import this package `com.viaembedded.smartetk.*` into Java code to use them.

4.2 Function return values

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

**SmartETK.S_OK**

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

**SmartETK.E_FAIL**

When a function returns the E_FAIL value, that indicates that the function has failed to complete.

**SmartETK.E_VERSION_NOT_SUPPORT**

When a function returns the E_VERSION_NOT_SUPPORT value, that indicates that the versions of SmartETK.jar and bsservice are not compatibility.
**SmartETK.E_INVALID_ARG**
When a function returns the `E_INVALID_ARG` value, that indicates that the arguments are invalid.

**SmartETK.E_FUNC_NOT_SUPPORT**
When a function returns the `E_FUNC_NOT_SUPPORT` value, that indicates that the function is not supported on this board.

**SmartETK.E_CONNECTION_FAIL**
When a function returns the `E_CONNECTION_FAIL` value, that indicates that the bsservice doesn't response the request. Please make sure bsservice is running successfully.

**SmartETK.E_NOT_RESPOND_YET**
When a function returns the `E_NOT_RESPOND_YET` value, that indicates that the bsservice function is still running and doesn't be finished yet.

**SmartETK.E_TIMEOUT**
When a function returns the `E_TIMEOUT` value, that indicates that there is no corresponding data had been received within the period.

**SmartETK.E_UART_OPENFAIL**
When `Uart.open()` returns the `E_UART_OPENFAIL` value, that indicates that the UART device can't be opened successfully. Please make sure the name of the tty device is existent.

**SmartETK.E_UART_NOT_OPEN**
When a function returns the `E_UART_NOT_OPEN` value, that indicates that uart object cannot be operated normally. It might represent the application doesn't
open uart device before calling other operating function; or it was reset by the other uart object.

**SmartETK.E_UART_ALREADY_OPENED**

When `Uart.open()` returns the `E_UART_ALREADY_OPENED` value, that indicates that the uart object had been opened. If you need to open other uart device, please calling close function to close the current device, then open the other uart again.

**SmartETK.E_UART_TTY_BEEN_USED**

When `Uart.open()` returns the `E_UART_TTY_BEEN_USED` value, that indicates that the tty device had been used by other uart object. If you want to use it, you can call reset function to release the resource and open it again.

**SmartETK.E_UART_BAUDRATE_NOT_SUPPORT**

When `Uart.setConfig()` returns the `E_UART_BAUDRATE_NOT_SUPPORT` value, that indicates that baud rate is not support.
4.3 Network Class

**Network Class**

**Syntax:**

```java
Network();
```

**Description:**
Create a new Network object.

**Example:**
Create a new Network object.

```java
Network m_network = new Network();
```

**Network.setWakeOnLan()**

**Syntax:**

```java
int setWakeOnLan(boolean bEnable);
```

**Description:**
Enable or disable Network wake on LAN function from suspend mode.

**Parameters:**
- `bEnable` – enable or disable Network wake on LAN function from suspend mode.

**Return:**
- `S_OK` – if the function succeeds.
- `E_*` – if the function fails.

**Network.getWakeOnLan()**

**Syntax:**

```java
int getWakeOnLan(boolean[] bEnable);
```

**Description:**
Get the status if Network wake on LAN function from suspend mode is enabled or disabled.
Parameters:
bEnable – return true for enable, return false for disable.

Return:
S_OK – if the function succeeds.
E_* – if the function fails.
4.4 WatchDog Class

WatchDog Class

Syntax:
WatchDog();

Description:
Create a new WatchDog object.

Example:
Create an WatchDog object.

WatchDog m_watchdog = new WatchDog();

WatchDog.setEnable()

Syntax:
int setEnable(boolean bEnable);

Description:
Enable or disable watch dog function. If watch dog function was enabled, it should be fed within a period, otherwise the system will reboot.

Parameters:
bEnable – enable or disable watch dog function.

Return:
S_OK – if the function succeeds.
E_* – if the function fails.

WatchDog.getEnable()

Syntax:
int getEnable(boolean[] bEnable);
**Description:**
Get the status if watch dog function is enabled or disabled.

**Parameters:**
bEnable – return true for enable, return false for disable.

**Return:**
S_OK – if the function succeeds.
E_* – if the function fails.

**WatchDog.setTimeout()**

**Syntax:**
```c
int setTimeout(int iTimeout);
```

**Description:**
Set watch dog timeout value. The argument is an integer representing the timeout in seconds.

**Parameters:**
iTimeout – timeout value. (only support timeout in 2, 4, 8, 16, 32, and 64 seconds).

**Return:**
S_OK – if the function succeeds.
E_* – if the function fails.

**WatchDog.getTimeout()**

**Syntax:**
```c
int getTimeout(int[] iTimeout);
```

**Description:**
Get watch dog timeout value.

**Parameters:**
iTimeout – return timeout value.
Return:
S_OK – if the function succeeds.
E_* – if the function fails.

WatchDog.keepAlive()

Syntax:
int keepAlive();

Description:
Keep watch dog alive to avoid rebooting the system.

Return:
S_OK – if the function succeeds.
E_* – if the function fails.
4.5 RTC Class

RTC Class

Syntax:
RTC();

Description:
Create a new RTC object.

Example:
Create an RTC object.

RTC m_rtc = new RTC();
RTC.setWakeUpTime()

Syntax:
```java
int setWakeUpTime(byte byMode, int iYear, byte byMonth, byte byDay, byte
byHour, byte byMin, byte bySec);
```

Description:
Set the wake up time and mode in RTC. The behavior of wake up from suspend mode will start at the wake up time, and it must loop according to the wake up mode.

Parameters:
byMode –
- RTC.ARG_RTC_MODE_DAY for wake up every day.
- RTC.ARG_RTC_MODE_MONTH for wake up every month.
- RTC.ARG_RTC_MODE_WEEK for wake up every week.
iYear – Year since 1900 ~ 2155 for wake up time
byMonth – Month (1 ~ 12) for wake up time
byDay – Day of the month (1 ~ 31) for wake up time
byHour – Hours (0 ~ 23) for wake up time
byMin – Minutes (0 ~ 59) for wake up time
bySec – Seconds (0 ~ 59) for wake up time

Return:
S_OK – if the function succeeds.
E_* – if the function fails.
RTC.getWakeUpTime()

Syntax:
int getWakeUpTime(RTCStatus RS);

Description:
Get the wake up time and mode set in RTC.

Parameters:
RTCStatus – Wake up time and mode set in RTC
class RTCStatus
{
    byMode –
    RTC.ARG_RTC_MODE_DAY for wake up every day.
    RTC.ARG_RTC_MODE_MONTH for wake up every month.
    RTC.ARG_RTC_MODE_WEEK for wake up every week.
    iYear – Year since 1900 ~ 2155 for wake up time
    byMonth – Month (1 ~ 12) for wake up time
    byDay – Day of the month (1 ~ 31) for wake up time
    byHour – Hours (0 ~ 23) for wake up time
    byMin – Minutes (0 ~ 59) for wake up time
    bySec – Seconds (0 ~ 59) for wake up time
}

Return:
S_OK – if the function succeeds.
E_* – if the function fails.
**RTC.setEnable()**

**Syntax:**

```java
int setEnable(boolean bEnable);
```

**Description:**
Enable or disable RTC wake up function from suspend mode.

**Parameters:**

`bEnable` – enable or disable RTC wake up function from suspend mode.

**Return:**

- S_OK – if the function succeeds.
- E_* – if the function fails.

**RTC.getEnable()**

**Syntax:**

```java
int getEnable(boolean[] bEnable);
```

**Description:**
Get the status if wake up function from suspend mode is enabled or disabled.

**Parameters:**

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

**Return:**

- S_OK – if the function succeeds.
- E_* – if the function fails.
4.6 Uart Class

Uart Class
Syntax:

Uart();

Description:
Create a new Uart object.

Example:
Create an Uart object.

Uart m_uart = new Uart();

Uart.open()
Syntax:

int open(String sDev);

Description:
Open the specified Uart device.

Parameters:
sDev – Uart device name. (Ex. ttyS0, ttyS2)

Return:
S_OK – if the function succeeds.
E_UART_OPENFAIL – open device is failed.
E_UART_ALREADY_OPENED – object already had been opened.
E_UART_TTY_BEEN_USED – device had been used by other object.
E_* – if the function fails.

Uart.close()
Syntax:

int close();
**Description:**
Close the current opened Uart device.

**Return:**
S_OK – if the function succeeds.
E_* – if the function fails.

**Uart.setConfig()**

**Syntax:**
```cpp
int setConfig(int iBaudRate, byte byDataBIts, byte byStopBits, byte byParity, byte byFlowCtrl);
```

**Description:**
Set the configurations of the opened Uart device.

**Parameters:**
iBaudRate – baud rate (Ex. 115200)
byDataBits – data bits. 7: 7-bit data bits; 8: 8-bit data bits
byStopBits – stop bits. 1: 1-stop bits; 2: 2-stop bits
byParity – parity. 0: none; 1: odd; 2: even
byFlowCtrl – flow control. 0: none; 1: CTS/RTS

**Return:**
S_OK – if the function succeeds.
E_* – if the function fails.

**Uart.getConfig()**

**Syntax:**
```cpp
int getConfig(UartConfig UC);
```

**Description:**
Get the configurations of the opened Uart device and stored them in passed UartConfig Class.

**Parameters:**
UartConfig – Uart Configuration
class UartConfiguration
{
    int iBaudRate – baud rate (Ex. 115200)
    byte byDataBits – data bits. 7: 7-bit data bits; 8: 8-bit data bits
    byte byStopBits – stop bits. 1: 1-stop bits; 2: 2-stop bits
    byte byParity – parity. 0: none; 1: odd; 2: even
    byte byFlowCtrl – flow control. 0: none; 1: CTS/RTS
}

Return:
S_OK – if the function succeeds.
E_* – if the function fails.

Example:
UartConfig UC = m_uart.new UartConfig();
if(SmartETK.S_OK != m_uart.getConfig(UC))
{
    cleanStatus();
    return;
}

Uart.setTimeout()

Syntax:
int setTimeout(boolean bEnable, int iTimeout);

Description:
Set the timeout of the opened Uart device.
bEnable = true, iTimeout = 0 (polling read)
bEnable = true, iTimeout > 0 (read with timeout)
bEnable = false (blocking read)

Parameters:
bEnable – enable or disable timeout function.
iTimeout – timeout value. Range 0 – 255 (0 ~ 25.5 seconds), unit is 0.1 second.

Return:
S_OK – if the function succeeds.
E_* – if the function fails.
**Uart.getTimeout()**

**Syntax:**

```java
int getTimeout(Timeout T);
```

**Description:**
Get the timeout configuration of the opened Uart device and stored them in passed Timeout Class.

**Parameters:**

- `Timeout` – timeout configuration class
  ```java
  class Timeout {
    boolean bEnable – enable or disable timeout function
    int iTimeout – timeout value. Range 0 – 255 (0 ~ 25.5 seconds), unit is 0.1 second.
  }
  ```

**Return:**
- `S_OK` – if the function succeeds.
- `E_*` – if the function fails.

**Example:**

```java
Timeout T = m_uart.new Timeout();
if(SmartETK.S_OK != m_uart.getTimeout(T)) {
    cleanStatus();
    return;
}
```

---

**Uart.setReturnChar()**

**Syntax:**

```java
int setReturnChar(boolean bEnable, byte byReturnChar);
```

**Description:**


Set the termination character of the opened Uart device.
bEnable = true (blocking until byReturnChar is received, or read buffer is full.)
bEnable = false (ignore byReturnChar checking when reading data)

**Parameters:**
- bEnable – enable or disable the termination character function.
- byReturnChar – the termination character

**Return:**
- S_OK – if the function succeeds.
- E_* – if the function fails.
Uart.getReturnChar()

Syntax:
```java
int getReturnChar(ReturnChar RC);
```

Description:
Get the termination character configuration of the opened Uart device and stored them in passed ReturnChar Class.

Parameters:
ReturnChar – termination character configuration class ReturnChar
```java
{
    boolean bEnable – enable or disable the termination character function
    byte byReturnChar – the termination character
}
```

Return:
S_OK – if the function succeeds.
E_ – if the function fails.

Example:
```java
ReturnChar RC = m_uart.new ReturnChar();
if(SmartETK_.OK != m_uart.getReturnChar(RC))
{
    cleanStatus();
    return;
}
```
**Uart.readData()**

**Syntax:**

```c
int readData(int iReadLen, byte[] byRead, int[] iActualLen);
```

**Description:**
receive data from the opened Uart device.

**Parameters:**
- `iReadLen` – number of bytes to read, maximum 1024 bytes per transfer.
- `byRead` – pointer to the buffer pointer.
- `iActualLen` – the actual number of bytes received.

**Return:**
- S_OK – if the function succeeds.
- E_* – if the function fails.

**Uart.writeData()**

**Syntax:**

```c
int writeData(int iWriteLen, byte[] byWrite);
```

**Description:**
receive data from the opened Uart device.

**Parameters:**
- `iWriteLen` – number of bytes to write, maximum 1024 bytes per transfer.
- `byWrite` – pointer to the buffer pointer.

**Return:**
- >=0 – if the function succeeds, return the actual number of bytes received.
- <0(E_*) – if the function fails.
Description:
send the data to the opened Uart device.

Parameters:
iWriteLen – number of bytes to transmit, maximum 1024 bytes per transfer.
byWrite – pointer to data buffer.

Return:
S_OK – if the function succeeds.
E_* – if the function fails.

Uart.reset()

Syntax:
```c
int reset();
```

Description:
reset the opened or open failed Uart device.
If the uart device had been used by other object, open function will return
E_UART_ALREADY_OPENED fails. The object could call reset function to
release the uart resource and call open uart device again.

Return:
S_OK – if the function succeeds.
E_* – if the function fails.
4.7 SystemETK Class

SystemETK Class
Syntax:
SystemETK();

Description:
Create a new SystemETK object.

Example:
Create a SystemETK object.
SystemETK m_system = new SystemETK();

SystemETK.reboot()
Syntax:
int reboot();

Description:
Reboot the machine.

Return:
S_OK – if the function succeeds.
E_* – if the function fails.

SystemETK.suspend()
Syntax:
int suspend();
Description:
Suspend the machine.

Return:
S_OK – if the function succeeds.
E_* – if the function fails.
5. Annex A: Network

5.1 Set Wake On LAN From Suspend mode
The following is the sample code:

```java
boolean bSetEnable = true;

if(null == m_network)
    m_network = new Network();

if(SmartETK.S_OK != m_network.setWakeOnLan(bSetEnable))
    return false;
```

5.2 Get Wake On LAN From Suspend mode Status
The following is the sample code:

```java
if(null == m_network)
    m_network = new Network();

boolean[] bGetEnable = new boolean[1];

if(SmartETK.S_OK != m_network.getWakeOnLan(bGetEnable))
    return false;

return bGetEnable[0];
```
6. Annex B: Watch Dog

6.1 Enable Watch Dog
The following is the sample code:

```java
if(null == m_watchdog)
    m_watchdog = new WatchDog();
if(SmartETK.S_OK != m_watchdog.enable(true))
    return false;
```

6.2 Disable Watch Dog
The following is the sample code:

```java
if(null == m_watchdog)
    m_watchdog = new WatchDog();
if(SmartETK.S_OK != m_watchdog.enable(false))
    return false;
```

6.3 Set Watch Dog Timeout Value
The following is the sample code:

```java
if(null == m_watchdog)
    m_watchdog = new WatchDog();
if(SmartETK.S_OK != m_watchdog.setTimeout(32))
    return false;
```
6.4 Get Watch Dog Status
The following is the sample code:

```java
if (null == m_watchdog)
    m_watchdog = new WatchDog();

boolean[] bGetEnable = new boolean[1];
int[] iTimeout = new int[1];

if (SmartETK.S_OK == m_watchdog.getEnable(bGetEnable))
{ // Do something ...
}
if (SmartETK.S_OK == m_watchdog.getTimeout(iTimeout))
{ // Do something ...
}
```

6.5 Keep Watch Dog Alive
The following is the sample code:

```java
if (null == m_watchdog)
    m_watchdog = new WatchDog();

if (SmartETK.S_OK != m_watchdog.keepAlive())
    return false;
```
7. Annex C: RTC

7.1 Set RTC Wake Up From Suspend mode
The following is the sample code:

```java
boolean bSetEnable = true;
if(null == m_rtc)
    m_rtc = new RTC();
if(SmartETK.S_OK != m_rtc.setEnable(bSetEnable))
    return false;
```

7.2 Get RTC Wake Up Status
The following is the sample code:

```java
if(null == m_rtc)
    m_rtc = new RTC();
boolean[] bGetEnable = new boolean[1];
if(SmartETK.S_OK != m_rtc.getEnable(bGetEnable))
    return false;
```
7.3 Set RTC Wake Up Time

Wake up from suspend since 2014/5/1, every day at 12:00.

The following is the sample code:

```c
byte byMode = RTC.ARG_RTC_MODE_DAY;
int iYear = 2014;
byte byMonth = IntToByte(5);
byte byDay = IntToByte(1);
byte byHour = IntToByte(12);
byte byMin = IntToByte(0);
byte bySec = IntToByte(0);

if(null == m_rtc)
    m_rtc = new RTC();

if(SmartETK.S_OK != m_rtc.setWakeUpTime(byMode, iYear, byMonth, byDay, byHour, byMin, bySec))
    [ return false;
]
```

7.4 Get RTC Wake Up Time

The following is the sample code:

```c
if(null == m_rtc)
    {
        m_rtc = new RTC();
        m_RS = m_rtc.new RTCStatus();
    }

if(SmartETK.S_OK != m_rtc.getWakeUpTime(m_RS))
    [ return false;
]```
8. Annex D: UART

8.1 UART Initialize Communication

The following is the sample code:

```java
private Uart m_uart = null;

m_uart = new Uart();
if (null == m_uart)
    { cleanStatus();
      return;
    }

if (SmartETK.S_OK != m_uart.open((m_sDev = mETDev.getText().toString())))
    { cleanStatus();
      return;
    }

if (SmartETK.S_OK != m_uart.setConfig((m_iBaudRate = Integer.valueOf(mETBaudRate.getText().toString())), (byte)8,
    (byte)1, (byte)0, (byte)0))
    { cleanStatus();
      return;
    }
```

8.2 UART Write Data

The following is the sample code:

```java
Notice that “mETWrite” is the EditView to store writing texts.

if (SmartETK.S_OK != m_uart.writeData(mETWrite.getText().toString().getBytes().length, mETWrite.getText().toString().getBytes()))
    { return;
    }
```

8.3 UART Read Data

The following is the sample code:
int iReadLen = \texttt{LENGTH};
byte[] byRead = \texttt{new byte[LENGTH]};
int[] iActualLen = \texttt{new int[1]};

while(SmartETK._S_OK == m_mainThreadUart.readData(iReadLen, byRead, iActualLen))
{
    if(0 == iActualLen[0])
        continue;

    /* Process received byRead byte array ... */

    for(int i = 0; i < byRead.length; i++)
        byRead[i] = 0;

    iActualLen[0] = 0;
}

9.1 Reboot the Machine
The following is the sample code:

```csharp
private SystemETK m_system = null;

if (null == m_system)
    m_system = new SystemETK();

if (SmartETK.S_OK != m_system.reboot())
    return;
```

9.2 Suspend the Machine
The following is the sample code:

```csharp
private SystemETK m_system = null;

if (null == m_system)
    m_system = new SystemETK();

if (SmartETK.S_OK != m_system.suspend())
    return;
```