Atop Technologies, Inc.

SDK Porting Guide

User Manual

V0.4

23 August 2019

This PDF Document contains internal hyperlinks for ease of navigation.
For example, click on any item listed in the **Table of Contents** to go to that page.

Published by:

Atop Technologies, Inc.

2F, No. 146, Sec. 1, Tung-Hsing Rd,
30261 Chupei City,
Hsinchu County
Taiwan, R.O.C.

Tel: +886-3-550-8137
Fax: +886-3-550-8131
sales@atop.com.tw
www.atoponline.com
www.atop.com.tw

Important Announcement

The information contained in this document is the property of Atop technologies, Inc., and is supplied for the sole purpose of operation and maintenance of Atop Technologies, Inc., products.

No part of this publication is to be used for any other purposes, and it is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system, or translated into any human or computer language, in any form, by any means, in whole or in part, without the prior explicit written consent of Atop Technologies, Inc.

Offenders will be held liable for damages and prosecution. All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

Disclaimer

We have checked the contents of this manual for agreement with the hardware and the software described. Since deviations cannot be precluded entirely, we cannot guarantee full agreement. However, the data in this manual is reviewed regularly and any necessary corrections will be included in subsequent editions.

Suggestions for improvement are welcome. All other product's names referenced herein are registered trademarks of their respective companies.

Documentation Control

Author:	Stanley Chung
Revision:	0.4
Revision History:	Add explain of Firewall script and NAT
Creation Date:	April 2019
Last Revision Date:	July 2019
Product Reference:	SDK Porting Guide
Document Status:	Update

Table of Contents

Table of Figures

List of Tables

1 Preface

1.1 *Purpose of the Manual*

This manual supports the user with effective steps for SDK porting. As such, it contains some advanced network management knowledge, instructions, examples, guidelines and general theories designed to help users manage this device and its corresponding software. A background in general theory is necessary when reading it. Please refer to the Glossary for technical terms and abbreviations (if any).

1.2 *Who Should Use This User Manual*

This manual is to be used by qualified network personnel or support technicians who are familiar with embedded Linux or C-programming skill. It might be useful for system programmers or network planners as well. This manual also provides helpful and handy information for first time users. For any related problems, please contact your local distributor. If they are unable to assist you, please redirect your inquiries to www.atop.com.tw or www.atoponline.com.

2 Introduction

ATOP SDK (software development kit) is a software package which helps you to easily implement applications on ATOP platforms. This document provides you with a quick and easy guide to help you implement functions with ATOP SDK.

3 Software Block Diagram

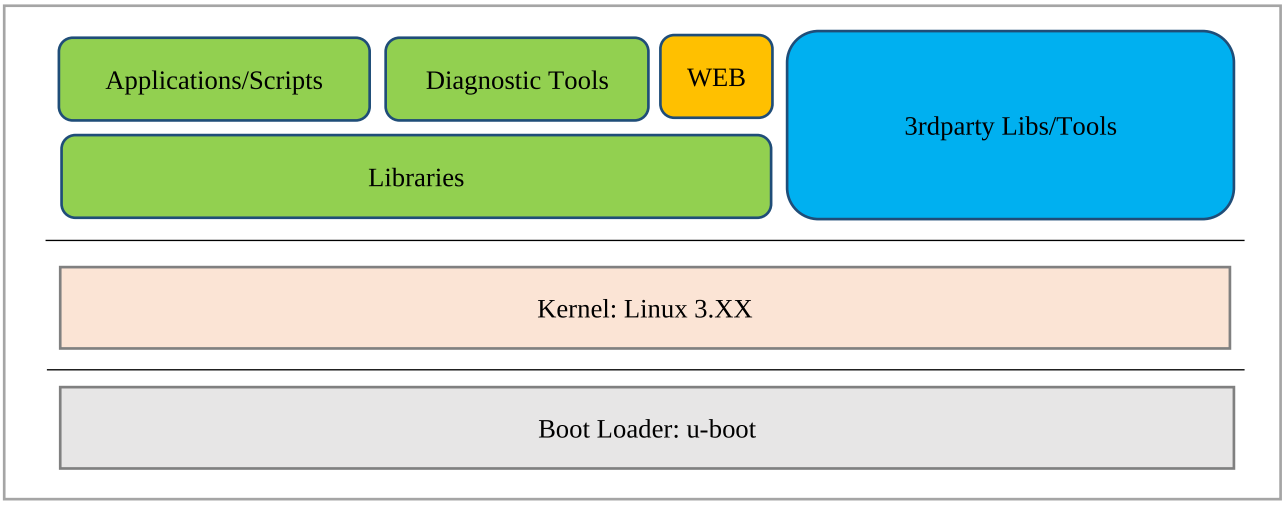


Figure 1: Software block diagram

Folder	Description
Bootloader	ATOP SDK support u-boot as the bootloader
Kernel	The OS used by ATOP SDK is the Linux
Libraries	The libraries provide some ATOP proprietary APIs for users to easy access system or peripheral components.
Applications/Scripts	SDK provides some basic applications and scripts to bring up network and some basic services.
Diagnostic tools	The “Diagnostic tools” are available for users to test and verify peripheral components.
WEB	SDK package uses the lighttpd as the WEB server. The simple WEB server helps users to manage system settings.
3rd Party Tools/Library	3rd party tools and libraries used in SDK

Table 1: SDK folders and description

4 Source Architecture

Below figure illustrates the source architecture of SDK:

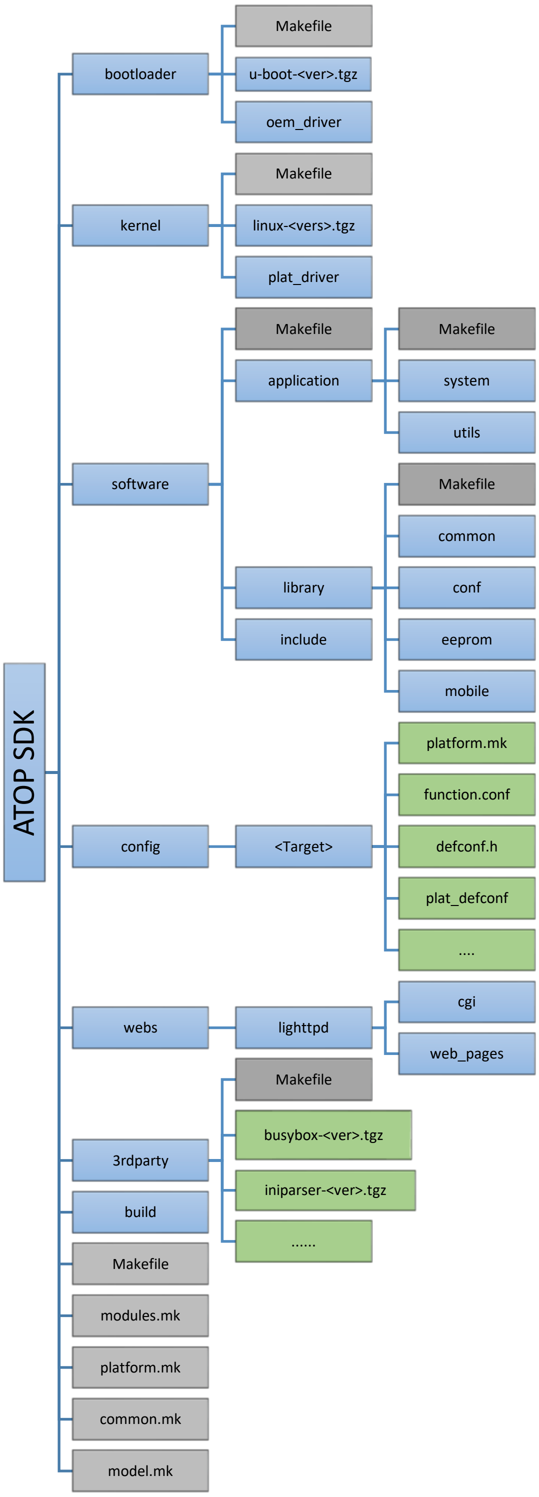


Figure 2: Source architecture of SDK

Folders	Descriptions
3rd Party	All 3rdparty tools and libraries are put under this folder.
Bootloader	This folder collects the boot source and related object codes.
Build	After source code is compiled successfully, generated images are put in this folder.
Config	This folder collects the platform/target configurations.
File System	This folder collects default scripts files and contents of image file system.
Kernel	This folder collects the kernel source and related object codes.
Software	This folder collects ATOP proprietary applications, libraries, and diagnostic tools.
Webs	This folder collects the WEB CGI files, pages and java scripts files.

Table 2: Source architecture's folders and description

5 Build Enviornment Setup

Supported operating system:

1. Ubuntu-16.04 (i386)
2. Ubuntu-18.04 (x64)

- For TI platform, install the toolchain to “/opt/ti-am335x-linux-devkit-08.00.00.00”
- For Nuvoton platform, install the toolchain to “/usr/local/arm_linux_4.8”
- Following below steps to setup build environment

5.1 *Ubuntu 16.04 (i386)*

These are the following steps you need to undertake to setup the build environment in Ububtu 16.04 (i386):

1. Copy and decompress the tool chain to build host (ti-am335x-linux-devkit-08.00.00.00.tar.bz2)

   TI Platform

   ```
   # sudo cp ti-am335x-linux-devkit-08.00.00.00.tar.bz2 /opt/
   # cd /opt/; sudo tar jxf ti-am335x-linux-devkit-08.00.00.00.tar.bz2
   ```

   Nuvoton sPlatform

   ```
   # sudo cp arm_linux_4.8_nuvoton.tgz /usr/local/
   # cd /usr/local/; sudo tar zxf arm_linux_4.8_nuvoton.tgz
   ```

2. Edit the bashrc file

   ```
   # vi ~/.bashrc
   ```

   Add the line mentioned below at the end of the file to set environment while system start-up

   TI Platform

   ```
   export PATH=/opt/ti-am335x-linux-devkit-08.00.00.00/bin:$PATH
   ```

   Nuvoton Platform

   ```
   export PATH=/usr/local/arm_linux_4.8:$PATH
   ```

3. Install essential components

   ```
   $ sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils kernel-package openssl libssl-dev autotools-dev autoconf libtool
   ```

4. Install image generating tools

   ```
   sudo apt-get install genext2fs u-boot-tools
   ```

5. Build libraries for ip tables (required only in case of ip table full support)

   ```
    $ sudo apt-get install flex bison libnfnetlink-dev libnetfilter-conntrack-dev libnetfilter-log-dev
   ```

6. Build libraries for glib (Required only in case of glib support)

   ```
   $ sudo apt-get install pkg-config libmount-dev libpcre3-dev
   ```

5.2 *Ubuntu 18.04 (x64)*

These are the following steps you need to undertake to setup the build environment in Ububtu 18.04 (x64):

1. Copy and decompress the tool chain to build host (ti-am335x-linux-devkit-08.00.00.00.tar.bz2)

   ```
   #TI:

   $sudo cp ti-am335x-linux-devkit-08.00.00.00.tar.bz2 /opt
   $sudo tar jxf ti-am335x-linux-devkit-08.00.00.00.tar.bz2 /opt

   # Nuvoton:

   $sudo cp arm_linux_4.8_nuvoton /usr/local
   $ cd /usr/local/; sudo tar zxfarm_linux_4.8_nuvoton.tgz
   ```

2. Edit the file of ".bashrc"

   ```
   $ vi ~/.bashrc
   # Add below line

   ...

   export PATH=/opt/ti-am335x-linux-devkit-08.00.00.00/bin:$PATH

   #or

   #export PATH=/usr/local/arm_linux_4.8/bin:$PATH
   ```

3. Install essential components

   ```
   $ sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils kernel-package openssl
   libssl-dev autotools-dev autoconf libtool
   ```

4. Instal image generating tools

   ```
   $ sudo apt-get install genext2fs u-boot-tools
   ```

5. For Linux 18.04, enable i386 architecture first

   ```
   $ sudo dpkg --add-architecture i386
   $ sudo apt-get update
   ```

6. Install 32-bit libraries

   ```
   $ sudo apt-get install lib32ncurses5 lib32z1
   ```

```
$ sudo apt-get install libstdc++6:i386 libncurses5:i386 libz1:i386 libc6:i386 libc6-dev-i386 g++-multilib
```

7. Switch shell from dash to bash

```
$ sudo dpkg-reconfigure dash
#Select no when prompted
```

8. Build libraries for glib (Required only when glib support)

```
$ sudo apt-get install pkg-config libmount-dev libpcre3-dev
```

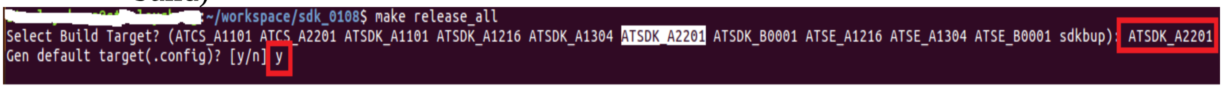
6 Source Code Compilation

Most of the compiling methods are supported in *<sdk>/modules.mk*. Here are the basic commands used to compile the sources and generate the image files:

1. Build the whole system image (filesystem and image) - *$ make release_all*
2. Build the bootloader image (boot) only - *$ make uboot*
3. Build the system image (build with kernel without bootloader) - *$ make release_img*
4. You may also try to use this command to generate the system image without building the kernel source (Make sure kernel built successfully at first time) - *$ make image*
5. Compile the folders of software and webs, then generate the image - *$ make fwimg*
6. Compile the software folder - *$ make swbuild*
7. Compile the web folder - *$ make websvr*
8. Compile the 3rd party folder - *$ make opensrc*

Here is an example how to generate the bootloader and system images:

1. Switch to SDK repository - $ *cd <your working spaces>*
2. Type this command to compile sources, and generate the bootloader and firmware image – *$ make release_all*
3. For compiling the sources for the first time, system may ask for the “Select Build Target” and “Gen default target”. (If not, you can ignode this step)
   - Select Build Target: <Your target platform to compile> (Ex: ATSDKC_A2201 is SDK for SE5901b platform)
   - Gen default target: y (Please type “y”, thus there is no need to specify build target in next build)

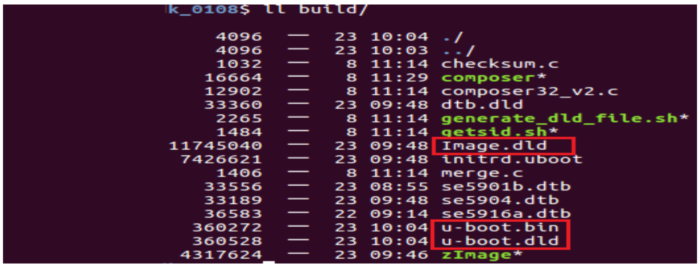


```
[redacted]:~/workspace/sdk_0108$ make release_all
Select Build Target? (ATCS_A1101 ATCS_A2201 ATSDK_A1101 ATSDK_A1216 ATSDK_A1304 ATSDK_A2201 ATSDK_B0001 ATSE_A1216 ATSE_A1304 ATSE_B0001 sdkbup): ATSDK_A2201
Gen default target(.config)? [y/n] y
```

Figure 3: Selecting build target

4. After build is successful, the boot and system images would be generated in the folder - “<SDK>/build”

   $ <TARGET>.dld // for system
   $ uboot.dld // for bootloader



```
k_0108$ ll build/

    4096  —  23 10:04 ./
    4096  —  23 10:03 ../
    1032  —   8 11:14 checksum.c
   16664  —   8 11:29 composer*
   12902  —   8 11:14 composer32_v2.c
   33360  —  23 09:48 dtb.dld
    2265  —   8 11:14 generate_dld_file.sh*
    1484  —   8 11:14 getsid.sh*
11745040  —  23 09:48 Image.dld
 7426621  —  23 09:48 initrd.uboot
    1406  —   8 11:14 merge.c
   33556  —  23 08:55 se5901b.dtb
   33189  —  23 09:48 se5904.dtb
   36583  —  22 09:14 se5916a.dtb
  360272  —  23 10:04 u-boot.bin
  360528  —  23 10:04 u-boot.dld
 4317624  —  23 09:46 zImage*
```

Figure 4: Generating boot and system images

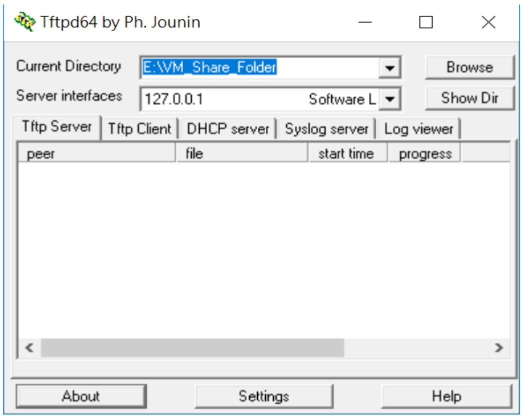
7 Upgrade system/firmware image to hardware platform

In total there are three ways to upgrade the system/firmware images to hardware platforms:

7.1 *Upgrade system image or bootloader from bootloader (with TFTP protocol)*

Here are the following steps:

1. Copy generated firmware of "Image.dld" (or <Target>.dld or ***u-boot.dld***) to tftp server folder (tftpd32/64)

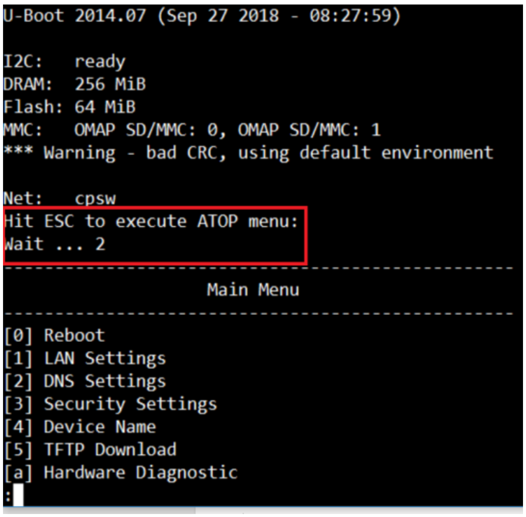


Current Directory: E:\VM_Share_Folder

Server interfaces: 127.0.0.1 Software L

Figure 5: Copying generated firmware to tftp server folder

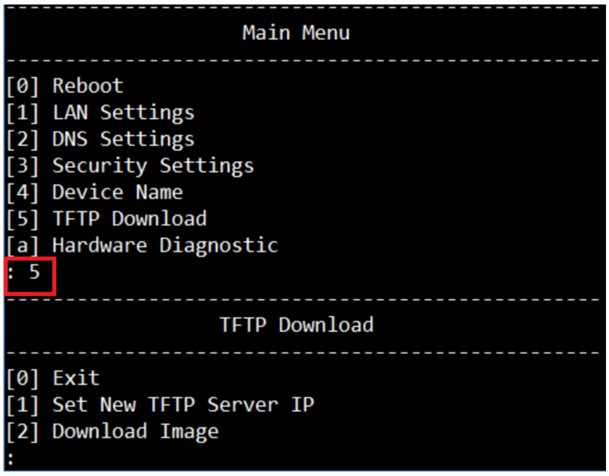
2. Reset the target device and press the "ESC" button to enter boot shell command menu



```
U-Boot 2014.07 (Sep 27 2018 - 08:27:59)

I2C:   ready
DRAM:  256 MiB
Flash: 64 MiB
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
*** Warning - bad CRC, using default environment

Net:   cpsw
Hit ESC to execute ATOP menu:
Wait ... 2
--------------------------------------------------
                    Main Menu
--------------------------------------------------
[0] Reboot
[1] LAN Settings
[2] DNS Settings
[3] Security Settings
[4] Device Name
[5] TFTP Download
[a] Hardware Diagnostic
:
```

Figure 6: Reseting target device

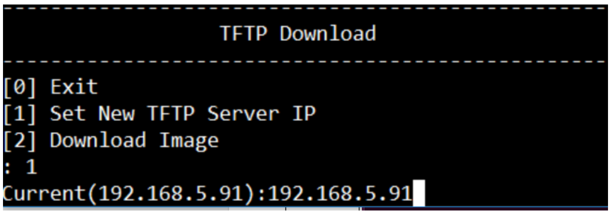
3. Type "5" to enter "TFTP Download" mode



```
-------------------------------------------------
                    Main Menu
-------------------------------------------------
[0] Reboot
[1] LAN Settings
[2] DNS Settings
[3] Security Settings
[4] Device Name
[5] TFTP Download
[a] Hardware Diagnostic
: 5
-------------------------------------------------
                  TFTP Download
-------------------------------------------------
[0] Exit
[1] Set New TFTP Server IP
[2] Download Image
:
```

Figure 7: Entering TFPT Download mode

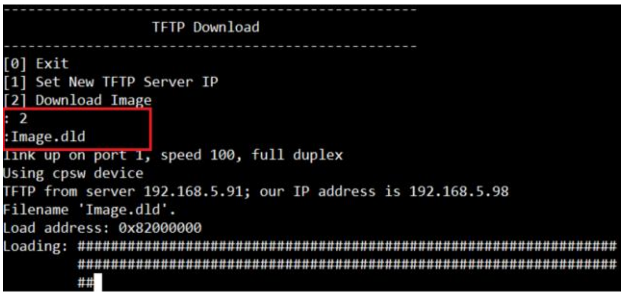
4. Type “1” to input correct TFTP server address



```
-------------------------------------------------
                  TFTP Download
-------------------------------------------------
[0] Exit
[1] Set New TFTP Server IP
[2] Download Image
: 1
Current(192.168.5.91):192.168.5.91
```

Figure 8: Input TFPT server address

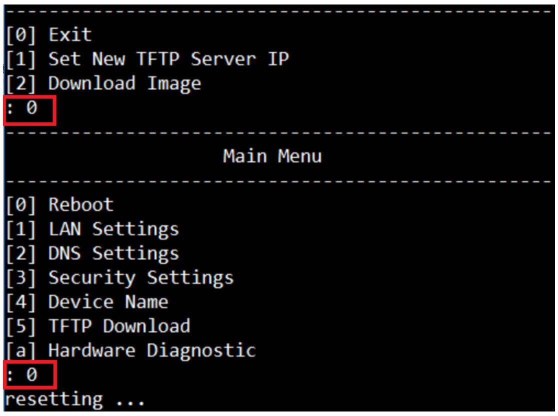
5. Type “2” and input the file name of “Image.dld” (or <Target>.dld). Then type “Enter” to activate the firmware upgrade progress



```
-------------------------------------------------
                  TFTP Download
-------------------------------------------------
[0] Exit
[1] Set New TFTP Server IP
[2] Download Image
: 2
:Image.dld
link up on port 1, speed 100, full duplex
Using cpsw device
TFTP from server 192.168.5.91; our IP address is 192.168.5.98
Filename 'Image.dld'.
Load address: 0x82000000
Loading: #################################################################
         #################################################################
         ##
```

Figure 9: Input file name of “Image.dld”

6. After firmware is upgraded successfully, reset the target device and make sure it can start-up properly. (Manual press “0”-> “0” to reset device)



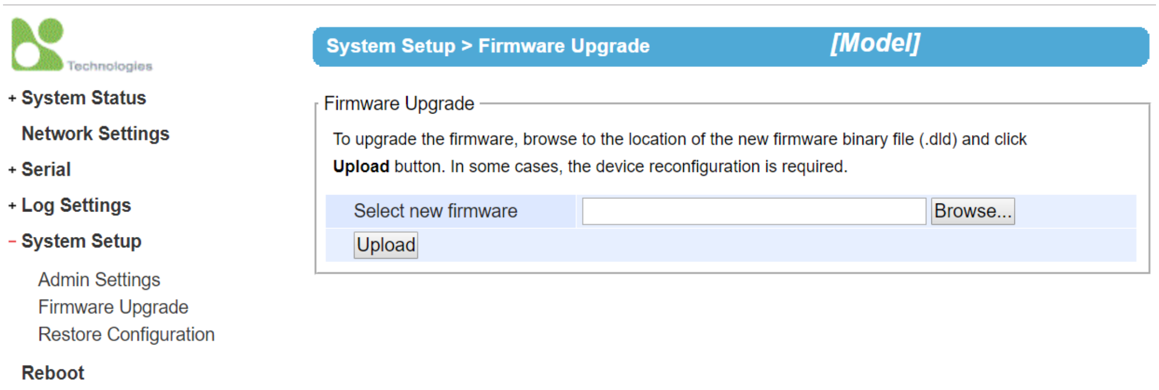
```
-------------------------------------------------
[0] Exit
[1] Set New TFTP Server IP
[2] Download Image
: 0
-------------------------------------------------
                    Main Menu
-------------------------------------------------
[0] Reboot
[1] LAN Settings
[2] DNS Settings
[3] Security Settings
[4] Device Name
[5] TFTP Download
[a] Hardware Diagnostic
: 0
resetting ...
```

Figure 10: Resetting target device

7.2 *Upgrade system image or bootloader through WEB page*

Here are the steps you need to follow:

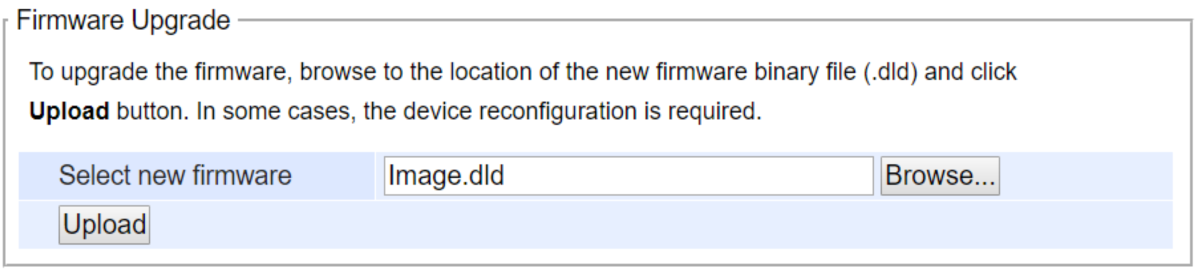
1. Login to WEB page (default account/password: admin/default)
2. Switch to Firmware Upgrade



To upgrade the firmware, browse to the location of the new firmware binary file (.dld) and click **Upload** button. In some cases, the device reconfiguration is required.

Figure 11: Login to ATOP SDK webpage

3. Click “Browser...” button to select firmware (Image.dld) from local host

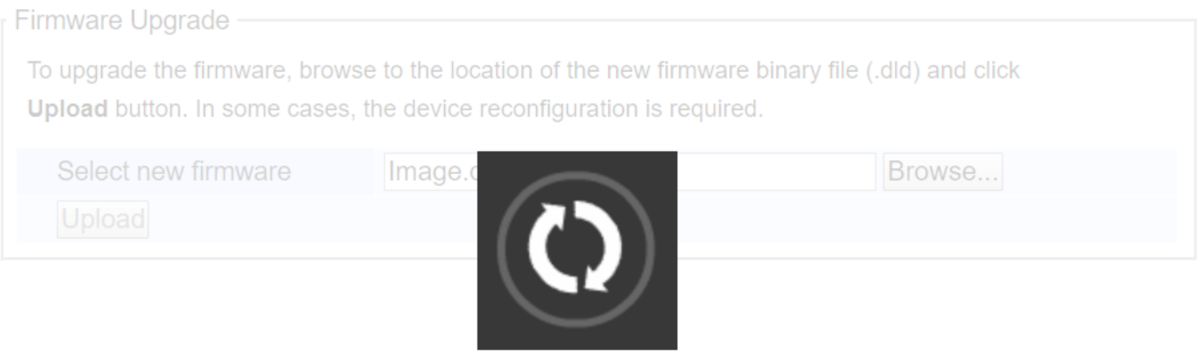


To upgrade the firmware, browse to the location of the new firmware binary file (.dld) and click **Upload** button. In some cases, the device reconfiguration is required.

Select new firmware: Image.dld

Figure 12: Select firmware from local host

4. Click “Upload” button to upload image to device

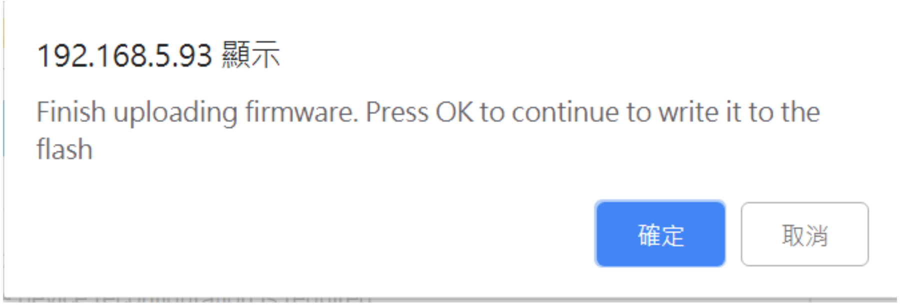


To upgrade the firmware, browse to the location of the new firmware binary file (.dld) and click **Upload** button. In some cases, the device reconfiguration is required.

Select new firmware: Image.d

Figure 13: Upload image to device

5. Click “Ok” to start the firmware upgrade process

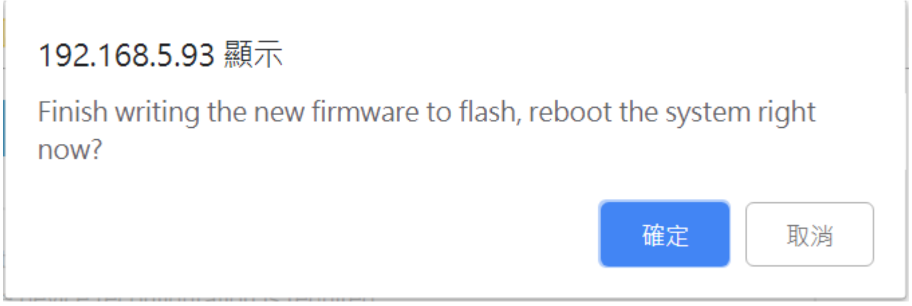


192.168.5.93 顯示

Finish uploading firmware. Press OK to continue to write it to the flash

Figure 14: Starting firmware process

6. Click “Ok” to finish the firmware upgrade process and reset device



192.168.5.93 顯示

Finish writing the new firmware to flash, reboot the system right now?

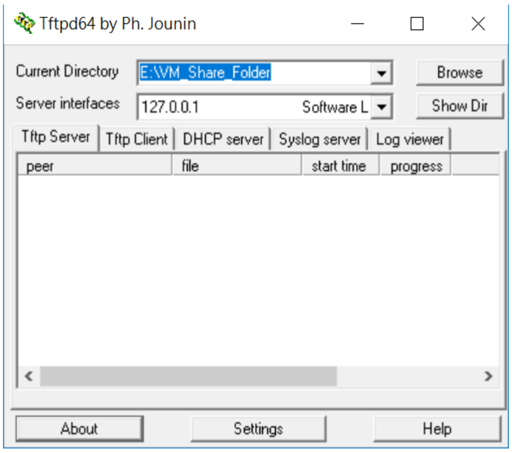
Figure 15: Finishing firmware process

7. Check if firmware has upgraded successfully after the device is rebooted

7.3 *Manually upgrade system image or bootloader from debug console*

Here are the steps to manually upgrade system image or bootloader from debug console:

1. Start the TFTP server and copy generated firmware to tftp server folder (tftpd32/64)



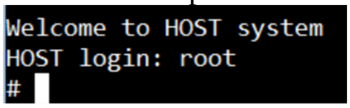
Tftpd64 by Ph. Jounin

Current Directory: E:\VM_Share_Folder

Server interfaces: 127.0.0.1 Software L

Figure 16: Copying generated firmware to tftp server folder

2. Login to debug console window (default account/password: root/none; baudrate: 115200)

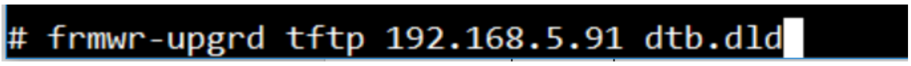


```
Welcome to HOST system
HOST login: root
#
```

Figure 17: Login to debug console window

3. Execute below command to activate the FW upgrade progress
   # frmwr-upgrd tftp <ftp svr. Addr> <fw image>

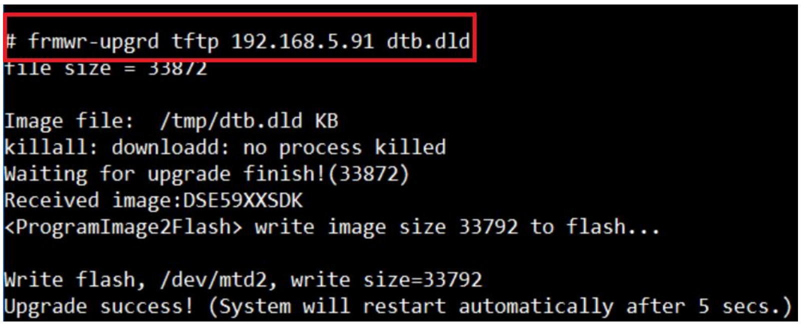
   Note: fw image can be system image (xxx.dld), boot image (u-boot.dld), or device
   Tree (dtb.dld)



```
# frmwr-upgrd tftp 192.168.5.91 dtb.dld
```

Figure 18: Activating FW upgrade process

4. Check if system resets automatically after FW upgrade



```
# frmwr-upgrd tftp 192.168.5.91 dtb.dld
file size = 33872

Image file:  /tmp/dtb.dld KB
killall: downloadd: no process killed
Waiting for upgrade finish!(33872)
Received image:DSE59XXSDK
<ProgramImage2Flash> write image size 33792 to flash...

Write flash, /dev/mtd2, write size=33792
Upgrade success! (System will restart automatically after 5 secs.)
```

Figure 19: Checking auto-system reset

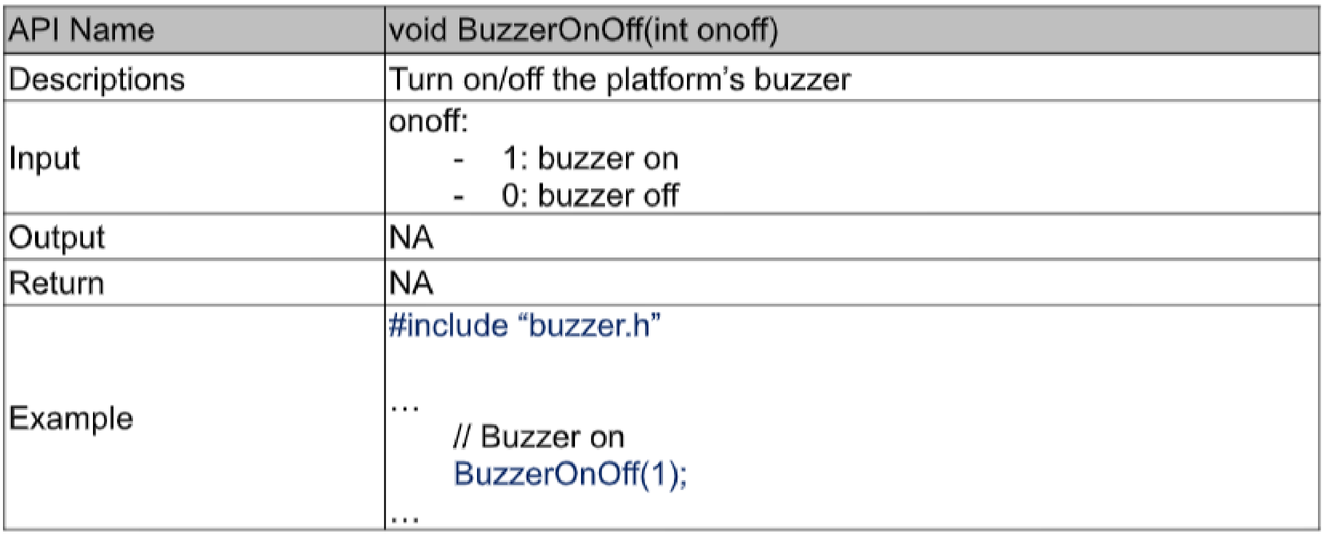
5. Check if system starts-up properly

8 Platform APIs

This section introduces APIs that are available in the SDK package. With these APIs, users can easily access/control peripheral components.

Note: API support varies on different platforms

8.1 *Buzzer*



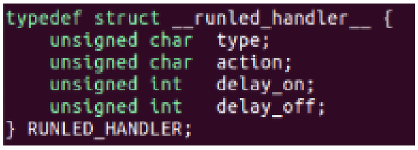
API Name	void BuzzerOnOff(int onoff)
Descriptions	Turn on/off the platform’s buzzer
Input	onoff: - 1: buzzer on - 0: buzzer off
Output	NA
Return	NA

Example	#include “buzzer.h”
	…
	// Buzzer on
	BuzzerOnOff(1);
	…

Table 3: API for buzzer

8.2 *Run Led*

API Name	void setRunLed(RUNLED_HANDLER *pHandler)
Descriptions	Handle the behaviors of “Run LED”

Input	pHandler: the pointer of RUNLED_HANDLER
	
	`typedef struct __runled_handler__ {`
	`    unsigned char  type;`
	`    unsigned char  action;`
	`    unsigned int   delay_on;`
	`    unsigned int   delay_off;`
	`} RUNLED_HANDLER;`
	
	- type:
	0: none
	1: solid on/off
	2: blink
	3: blink as heart beat
	
	- action:
	0: LED off
	1: LED on
	
	- delay_on:
	interval of LED on
	
	- delay_off:
	interval of LED off

Output	NA
Return	NA

Example	#include “runled.h”
	
	…
	RUNLED_HANDLER handler;
	
	handler.type= 2; // blink
	handler.action= 1; // on
	handler.delay_on= 1000; // on interval, 1000 ms
	handler.delay_on= 500; // off interval; 50ms
	
	SetRunLed(&handler);
	…

Table 4: API for Run LED

8.3 *Alarm Led*

API Name	void setAlarmLed(unsigned int onoff)
Descriptions	Turn of/off the alarm led
Input	onoff: 0: off 1: on
Output	NA
Return	NA

Example	#include “alarmled.h”
	…
	// Turn on alarm LED
	SetAlarmLed(1);
	…

Table 5: API for Alarm LED

8.4 *DI, DO*

API Name	void sysGetDI(in index)
Descriptions	Get DI pin status
Input	index: Index of pin
Output	NA
Return	NA

Example	#include “di.h”
	…
	//Get DI0 pin status
	SysGetDI(0);
	…

Table 6: API for DI

API Name	void sysSetDO(int index, in value)
Descriptions	Set DO state
Input	index: Index of pin value: 0: off 1: on
Output	NA
Return	NA

Example	#include “alarmled.h”
	
	…
	// Set DO0 on
	SysSetDO(0, 1);
	// Set DO1 off
	SysSetDO(1, 0);
	…

Table 7: API for DO

8.5 *HW watchdog (TI plarform only)*

API Name	void hwd_enable(void)
Descriptions	Enable HW watchdog
Input	NA
Output	NA
Return	NA

Example	#include “sys_hwd.h”
	
	…
	// enable HW watchdog
	hwd_enable();
	…

Table 8: API for HW watchdog

API Name	void hwd_disable(void)
Descriptions	Disable HW watchdog

Input	NA
Output	NA
Return	NA

Example	#include “sys_hwd.h”
	
	…
	// enable HW watchdog
	hwd_disable();
	…

Table 9: API for HW watchdog

API Name	void hwd_clear(void)
Descriptions	Clear HW watchdog timer count
Input	NA
Output	NA
Return	NA

Example	#include “sys_hwd.h”
	
	…
	// clear HW watchdog
	hwd_clear();
	…

Table 10: API for HW watchdog

API Name	void hwd_time(int interval)
Descriptions	Set HW watchdog timer interval
Input	Interval: interval of timeout (sec)
Output	NA
Return	NA

Example	#include “sys_hwd.h”
	
	…
	// Set HW watchdog timer interval to 10 secs
	hwd_timeout(10);
	…

Table 11: API for HW watchdog

8.6 *COM Management*

API Name	Int SysUARTNumber(void)

Descriptions	Query supported number of COM ports
Input	NA
Output	NA
Return	Number of supported COM ports
Example	#include “sys_uart.h” … // Get COM port number Int num = SysUARTNumber(); …

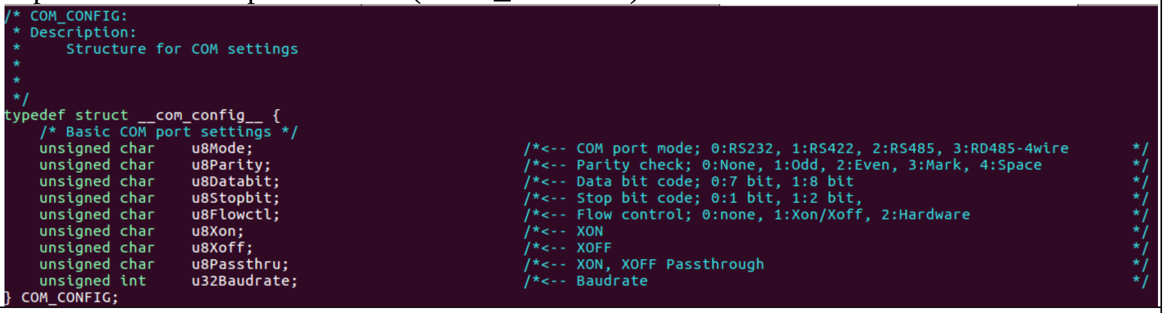
Table 12: API for COM Management

API Name	void comport_init()
Descriptions	Initialize COM ports depending on COM configurations
Input	NA
Output	NA
Return	NA

Example	#include “comport.h”
	
	…
	// Depending on COM settings to Initialize physical port settings
	comport_init();
	…

Table 13: API for COM Management

API Name	int comport_set(unsigned char index, void *pConf)
Descriptions	Set COM port configurations
Input	Index: Index of physical COM port

	pConf: pointer of COM port handler (COM_CONFIG) 

```
/* COM_CONFIG:
 * Description:
 *     Structure for COM settings
 *
 *
 */
typedef struct __com_config__ {
    /* Basic COM port settings */
    unsigned char    u8Mode;                             /*<-- COM port mode; 0:RS232, 1:RS422, 2:RS485, 3:RD485-4wire      */
    unsigned char    u8Parity;                           /*<-- Parity check; 0:None, 1:Odd, 2:Even, 3:Mark, 4:Space         */
    unsigned char    u8Databit;                          /*<-- Data bit code; 0:7 bit, 1:8 bit                              */
    unsigned char    u8Stopbit;                          /*<-- Stop bit code; 0:1 bit, 1:2 bit,                             */
    unsigned char    u8Flowctl;                          /*<-- Flow control; 0:none, 1:Xon/Xoff, 2:Hardware                 */
    unsigned char    u8Xon;                              /*<-- XON                                                          */
    unsigned char    u8Xoff;                             /*<-- XOFF                                                         */
    unsigned char    u8Passthru;                         /*<-- XON, XOFF Passthrough                                        */
    unsigned int     u32Baudrate;                        /*<-- Baudrate                                                     */
} COM_CONFIG;
```

Output	NA
Return	NA

Example	#include “com_conf.h”
	#include “comport.h”
	
	…
	COM_CONF conf;
	
	memset(&conf, 0, sizeof(COM_CONF));
	
	conf.u8Mode = 0; // RS-232
	conf.u8Parity = 0; // none
	conf.u8Databit = 1; // 8 bit
	conf.u8Stopbit = 0; // 1 bit
	conf.u8Xon = 0xff; // 0xff
	conf.u8Xoff = 0xff; // 0xff
	conf.u8Mode = 0; // RS-232
	conf.u8Passthru = 0; // none
	conf.u32Baudrate = 115200; // 115200
	
	// Set COM0 settings
	comport_set(0, &conf);
	…

Table 14: API for COM Management

8.7 *Relay*

API Name	void SetRelayOnOff (unsigned int onoff)
Descriptions	Switch relay state: on or off
Input	onoff: 0: off 1: on

Output	NA
Return	NA

Example	#include “relay.h”
	
	…
	// Set Relay state to on
	SetRelayOnOff(1);
	…

Table 15: API for Relay

8.8 *Log*

API Name	void SendSysLog (severity_e serv, char *prefix, char *msg)
Descriptions	Send messages to Log file
Input	serv: EVT_INFO EVT_WARN EVT_ERR prefix: Prefix information LOG_SYS: “Sys” LOG_NET: “Net” msg: Message contents
Output	NA
Return	NA

Example	#include “loginfo.h”
	
	…
	// Set Relay state to on
	SendSysLog(EVT_INFO, LOG_SYS, “This is a test!”);
	…

Table 16: API for Log

API Name	void SendSysLog (severity_e serv, char *prefix, char *msg)
Descriptions	Send messages to Log file
Input	serv: EVT_INFO EVT_WARN EVT_ERR prefix: Prefix information LOG_SYS: “Sys” LOG_NET: “Net” msg:

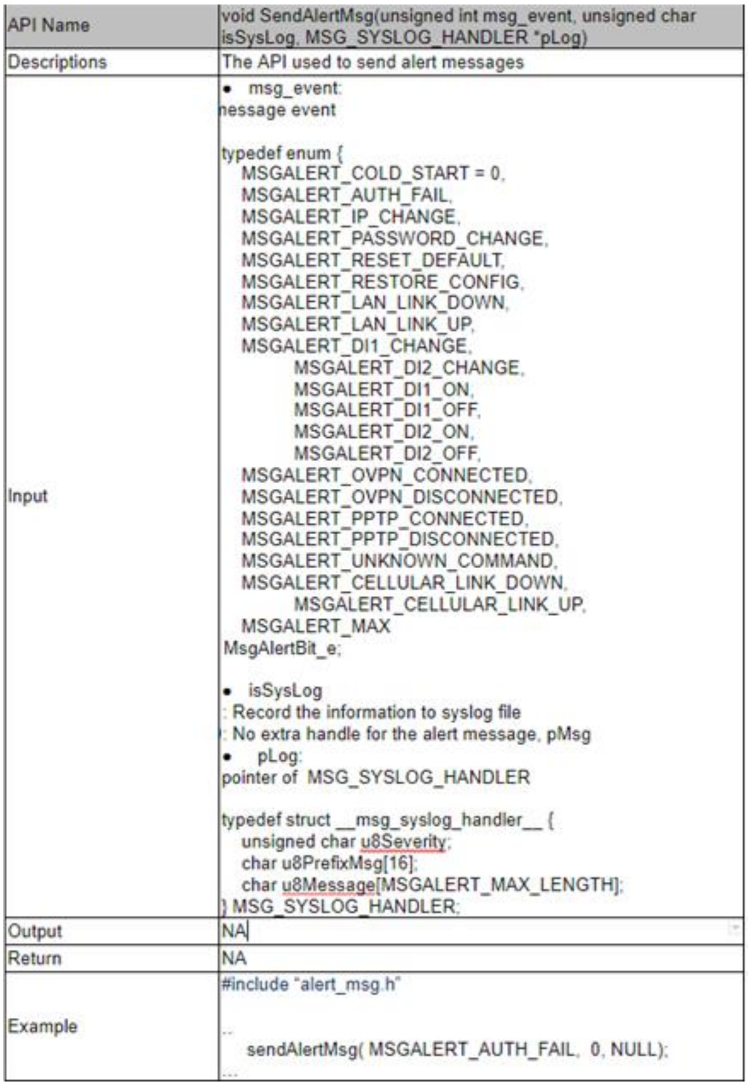
	Message contents
Output	NA
Return	NA

Example	#include “loginfo.h”
	
	…
	    // Set Relay state to on
	    SendSysLog(EVT_INFO, LOG_SYS, “This is a test!”);
	…

Table 17: API for Log

8.9 *Alert Message Management*

Depending on platforms, SDK supports some alert API to help users send alert information while receiving specified events.



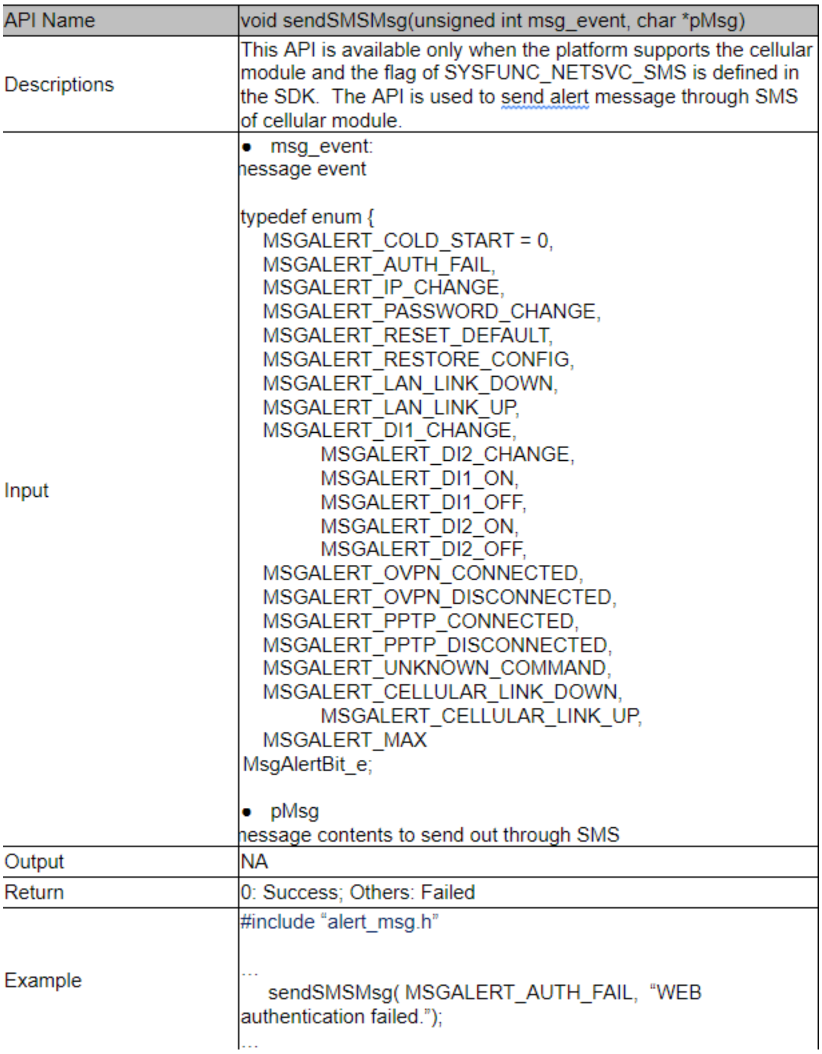
API Name	void SendAlertMsg(unsigned int msg_event, unsigned char isSysLog, MSG_SYSLOG_HANDLER *pLog)
Descriptions	The API used to send alert messages

Input	• msg_event:
	message event
	
	typedef enum {
	MSGALERT_COLD_START = 0,
	MSGALERT_AUTH_FAIL,
	MSGALERT_IP_CHANGE,
	MSGALERT_PASSWORD_CHANGE,
	MSGALERT_RESET_DEFAULT,
	MSGALERT_RESTORE_CONFIG,
	MSGALERT_LAN_LINK_DOWN,
	MSGALERT_LAN_LINK_UP,
	MSGALERT_DI1_CHANGE,
	MSGALERT_DI2_CHANGE,
	MSGALERT_DI1_ON,
	MSGALERT_DI1_OFF,
	MSGALERT_DI2_ON,
	MSGALERT_DI2_OFF,
	MSGALERT_OVPN_CONNECTED,
	MSGALERT_OVPN_DISCONNECTED,
	MSGALERT_PPTP_CONNECTED,
	MSGALERT_PPTP_DISCONNECTED,
	MSGALERT_UNKNOWN_COMMAND,
	MSGALERT_CELLULAR_LINK_DOWN,
	MSGALERT_CELLULAR_LINK_UP,
	MSGALERT_MAX
	MsgAlertBit_e;
	
	• isSysLog
	: Record the information to syslog file
	: No extra handle for the alert message, pMsg
	• pLog:
	pointer of MSG_SYSLOG_HANDLER
	
	typedef struct __msg_syslog_handler__ {
	unsigned char u8Severity;
	char u8PrefixMsg[16];
	char u8Message[MSGALERT_MAX_LENGTH];
	} MSG_SYSLOG_HANDLER;

Output	NA
Return	NA

Example	#include "alert_msg.h"
	
	..
	sendAlertMsg(MSGALERT_AUTH_FAIL, 0, NULL);
	...

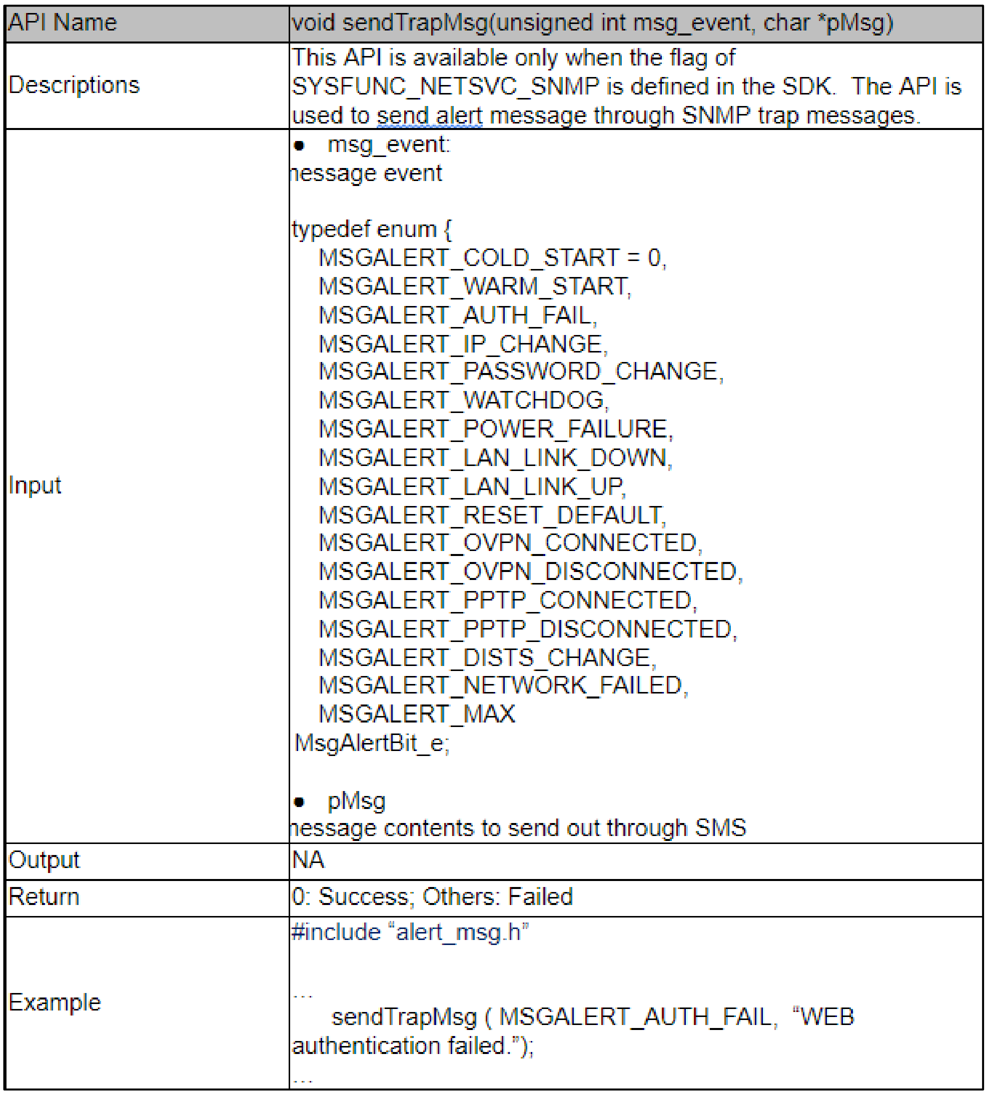
Table 18: API for alert message management



API Name	void sendSMSMsg(unsigned int msg_event, char *pMsg)
Descriptions	This API is available only when the platform supports the cellular module and the flag of SYSFUNC_NETSVC_SMS is defined in the SDK. The API is used to send alert message through SMS of cellular module.
Input	● msg_event: hessage event typedef enum { MSGALERT_COLD_START = 0, MSGALERT_AUTH_FAIL, MSGALERT_IP_CHANGE, MSGALERT_PASSWORD_CHANGE, MSGALERT_RESET_DEFAULT, MSGALERT_RESTORE_CONFIG, MSGALERT_LAN_LINK_DOWN, MSGALERT_LAN_LINK_UP, MSGALERT_DI1_CHANGE, MSGALERT_DI2_CHANGE, MSGALERT_DI1_ON, MSGALERT_DI1_OFF, MSGALERT_DI2_ON, MSGALERT_DI2_OFF, MSGALERT_OVPN_CONNECTED, MSGALERT_OVPN_DISCONNECTED, MSGALERT_PPTP_CONNECTED, MSGALERT_PPTP_DISCONNECTED, MSGALERT_UNKNOWN_COMMAND, MSGALERT_CELLULAR_LINK_DOWN, MSGALERT_CELLULAR_LINK_UP, MSGALERT_MAX MsgAlertBit_e; ● pMsg hessage contents to send out through SMS
Output	NA
Return	0: Success; Others: Failed

Example	#include "alert_msg.h"
	
	…
	sendSMSMsg(MSGALERT_AUTH_FAIL, "WEB authentication failed.");
	…

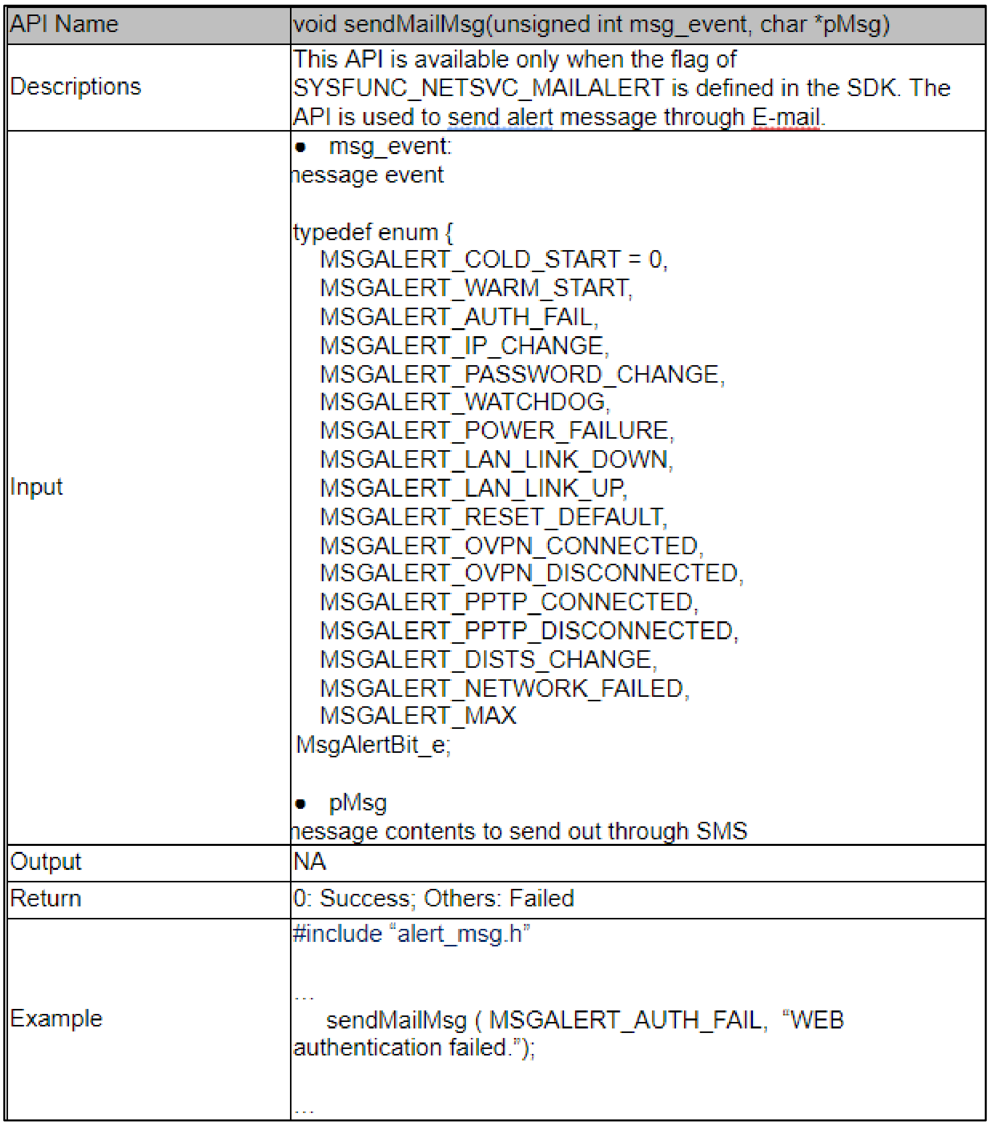
Table 19: API for SMS message management



API Name	void sendTrapMsg(unsigned int msg_event, char *pMsg)
Descriptions	This API is available only when the flag of SYSFUNC_NETSVC_SNMP is defined in the SDK. The API is used to send alert message through SNMP trap messages.
Input	● msg_event: hessage event typedef enum { MSGALERT_COLD_START = 0, MSGALERT_WARM_START, MSGALERT_AUTH_FAIL, MSGALERT_IP_CHANGE, MSGALERT_PASSWORD_CHANGE, MSGALERT_WATCHDOG, MSGALERT_POWER_FAILURE, MSGALERT_LAN_LINK_DOWN, MSGALERT_LAN_LINK_UP, MSGALERT_RESET_DEFAULT, MSGALERT_OVPN_CONNECTED, MSGALERT_OVPN_DISCONNECTED, MSGALERT_PPTP_CONNECTED, MSGALERT_PPTP_DISCONNECTED, MSGALERT_DISTS_CHANGE, MSGALERT_NETWORK_FAILED, MSGALERT_MAX MsgAlertBit_e; ● pMsg hessage contents to send out through SMS
Output	NA
Return	0: Success; Others: Failed

Example	#include "alert_msg.h"
	
	…
	sendTrapMsg (MSGALERT_AUTH_FAIL, "WEB authentication failed.");
	…

Table 20: API for Trap message management



API Name	void sendMailMsg(unsigned int msg_event, char *pMsg)
Descriptions	This API is available only when the flag of SYSFUNC_NETSVC_MAILALERT is defined in the SDK. The API is used to send alert message through E-mail.
Input	● msg_event: nessage event typedef enum { MSGALERT_COLD_START = 0, MSGALERT_WARM_START, MSGALERT_AUTH_FAIL, MSGALERT_IP_CHANGE, MSGALERT_PASSWORD_CHANGE, MSGALERT_WATCHDOG, MSGALERT_POWER_FAILURE, MSGALERT_LAN_LINK_DOWN, MSGALERT_LAN_LINK_UP, MSGALERT_RESET_DEFAULT, MSGALERT_OVPN_CONNECTED, MSGALERT_OVPN_DISCONNECTED, MSGALERT_PPTP_CONNECTED, MSGALERT_PPTP_DISCONNECTED, MSGALERT_DISTS_CHANGE, MSGALERT_NETWORK_FAILED, MSGALERT_MAX MsgAlertBit_e; ● pMsg nessage contents to send out through SMS
Output	NA
Return	0: Success; Others: Failed

Example	#include "alert_msg.h"
	
	…
	sendMailMsg (MSGALERT_AUTH_FAIL, "WEB authentication failed.");
	
	…

Table 21: API for Mail message management

8.10 *Firmware Upgrade*

API Name	Int fw_upgade(char *fw_addr, int length)
Descriptions	Programming firmware to flash
Input	fw_addr: buffer address of firmware image length: length of firmware buffer
Output	NA
Return	0: Success; Others: Failed

Example	#include “sys_upgapi.h”
	#include “sys_reboot.h”
	
	…
	/* Calling firmware upgrade lib-api */
	if (fw_upgrade(buff, file_size) == EXECUTE_SUCCESS)
	{
	printf("Upgrade success! (System will restart automatically after 5 secs.)\n");
	sleep(5);
	SysRebootSystem();
	}
	else
	{
	printf("Upgrade failed!!!!\nPlease making sure your fw image is correct and try again!\n\n");
	}
	…

Table 22: API for firmware upgrade

API Name	Int fwupg_alloc_shmbuf(unsigned int length)
Descriptions	Allocate shared memory buffer for fw imge
Input	Buffer length(or image length) of the shared memory
Output	NA
Return	Pointer of shared memory buffer address
Example	See frmwr-upgrd.c

Table 22: API for firmware upgrade

API Name	Char *fwupg_get_shmbuf(unsigned int length)
Descriptions	Get shared memory buffer
Input	Length of firmware image

Output	NA
Return	Pointer of shared memory buffer address
Example	See <sdk>/web/cgi/firmwareUpgrade.c

Table 23: API for firmware upgrade

API Name	Char *fwupg_unlink_shmbuf()
Descriptions	Unlink shared memory buffer of firmware image
Input	NA
Output	NA
Return	NA
Example	See <sdk>/web/cgi/firmwareUpgrade.c

Table 24: API for firmware upgrade

8.11 *System Reboot*

API Name	Int SysRebootSystem(void)
Descriptions	Reboot system with the signal of “SIGTERM”
Input	NA
Output	NA
Return	NA

Example	#include “sys_reboot.h”
	…
	SysRebootSystem();
	…

Table 25: API for system reboot

API Name	Int RebootSystem2(void)
Descriptions	Reboot system without the signal of “SIGTERM”
Input	NA
Output	NA
Return	NA

Example	#include “sys_reboot.h”
	…
	RebootSystem2();
	…

Table 26: API for system reboot

8.12 *System Management*

API Name	Int SysBootloaderVersion(SysVersion_t *pVer) Int SysKernelVersion(SysVersion_t *pVer) Int SysAPVersion(SysVersion_t *pVer) Int SysCPLDVersion(SysVersion_t *pVer)
Descriptions	Get boot loader version Get kernel version Get AP version Get CPLD version
Input	Structure pointer of SysVersion_t
Output	Bootloader version information Kernel version information AP version information CPLD version information
Return	NA

Example	#include “ver_info.h”
	
	SysVersion_t ver;
	…
	SysBootloaderVersion(&ver);
	Printf(“blVer: %u.%02u”, ver.VerMajor, ver.VerMinor);
	
	SysKernelVersion(&ver);
	Printf(“kernelVer: %u.%02u”, ver.VerMajor, ver.VerMinor);
	
	SysAPVersion(&ver);
	Printf(“apVer: %u.%02u”, ver.VerMajor, ver.VerMinor);
	
	SysCPLDVersion(&ver);
	Printf(“cpldVer: %u.%02u”, ver.VerMajor, ver.VerMinor);
	
	…

Table 27: API for system management

API Name	Int SysStrKernelVersion(void *) Int SysStrAPVersion(void *)
Descriptions	Get kernel version Get AP version
Input	NA
Output	Kernel version information AP version information
Return	NA
Example	#include “ver_info.h” …

	printf("kernelVer: %s", SysStrKernelVersion(NULL));
	printf("apVer: %s", SysStrAPVersion(NULL));
	…

Table 28: API for system management

API Name	Int ExecuteSysCommand(char *cmd, int limit)
Descriptions	Execute system command (popen, pipe stream)
Input	cmd: string buffer of system command limit: limitation of reading length after command execution -1 or 0 to indicate to ignore limitation check
Output	NA
Return	0: failed; 1: success

Example	#include "sys_cmd.h"
	…
	ExecuteSysCommand("/sbin/reboot", -1);
	…

Table 29: API for system management

8.13 *Cellular Control (Cellular 3G/4G platform only)*

- Establishing cellular connection:

API Name	void Dial_connect(void)
Descriptions	Establish the 3G/4G connection
Input	NA
Output	NA
Return	NA

Example	#include "lib_dial.h"
	…
	Dial_connect();
	…

Table 30: Estabishing cellular connection

- Terminating cellular connection:

API Name	void Dial_disconnect(void)

Descriptions	Disconnect the 3G/4G connection
Input	NA
Output	NA
Return	NA

Example	#include “lib_dial.h”
	
	…
	Dial_disconnect();
	
	…

Table 31: Terminating cellular connection

- Get status information of cellular connection:

API Name	int Get_dial_info(DIAL_INFO *pInfo)
Descriptions	Get dialing status
Input	pInfo: Pointer of DIAL_INFO
Output	Dialing information
Return	Success: CMD_GET_SUCCESS Error: Others

Example	#include “lib_dial.h”
	
	DIAL_INFO dial_info;
	…
	Get_dial_info(&dial_info);

Table 32: Get status information of cellular connection

- Get GPS information (for GPS model only):

API Name	int Get_gps_info(GPS_INFO *pInfo)
Descriptions	Get GPS status
Input	pInfo: Pointer of GPS_INFO
Output	GPS information
Return	Success: CMD_GET_SUCCESS Error: Others

Example	#include “lib_dial.h”
	
	GPS_INFO gps_info;
	…
	Get_gps_info(&gps_info);
	
	…

Table 33: Get GPS information

- Check 4G support on platform:

API Name	int Sys4GSupport(void)
Descriptions	Check 4G support
Input	NA
Output	NA
Return	1: 4G function is supported on this platform 0: 4G function is not supported on this device

Example	#include “cellular_api.h”
	
	…
	If(Sys4GSupport()) {
	printf(“4G module is supported on this platform”);
	}
	
	…

Table 34: Get 4G support

- Check if 4G module is detected by system:

API Name	int Sys4GSupport(void)
Descriptions	Check 4G support
Input	NA
Output	NA
Return	1: 4G function is supported on this platform 0: 4G function is not supported on this device

Example	#include “cellular_api.h”
	
	…
	If(Sys4GSupport()) {
	printf(“4G module is supported on this platform”);
	}
	…

Table 35: Check 4G detection

- Get used 4G interface name:

API Name	int Sys4GInterface(char *pIf)
Descriptions	Get used 4G interface name
Input	pIf: pointer of buffer
Output	Interface name of 4G interface

Return	-1: 4G interface is not named with “eth” interface. In such case, you can read “pIf” to get 4G interface name >= 0: if 4G interface is named as “eth”, index is returned

Example	#include “cellular_api.h”
	Int index = -1;
	Char interface[16] = {0};
	…
	If((index = Sys4GInterface(&interface)) < 0) {
	printf(“4G interface: %s\n”, interface);
	} else {
	printf(“4G interface: eth%d\n”, index);
	}
	…

Table 36: Get used 4G interface name

9 INI Configs Read/Write (Settings Management)

Most of the device configurations in SDK are stored in INI files. System allows some APIs to help users easily access INI files. With these APIs, users can easily access the settings by using specified feature IDs. Below are feature IDs currently supported in the SDK package:

Feature	ID
SYSCONF_FEATURE_ALL	0x00
SYSCONF_FEATURE_BOARD	0x01
SYSCONF_FEATURE_COM	0x02
SYSCONF_FEATURE_SYSTEM	0x03
SYSCONF_FEATURE_SYSLOG	0x04
SYSCONF_FEATURE_NET	0x05
SYSCONF_FEATURE_NETDNS	0x06
SYSCONF_FEATURE_PORTFORWARD	0x07
SYSCONF_FEATURE_NETWORK_3G	0x08
SYSCONF_FEATURE_NAT	0x09
SYSCONF_FEATURE_SMS	0x0A
SYSCONF_FEATURE_SNMP	0x0B
SYSCONF_FEATURE_VIP	0x0C
SYSCONF_FEATURE_OVPN	0x0D
SYSCONF_FEATURE_PPTP	0x0E
SYSCONF_FEATURE_IPSEC	0x0F
SYSCONF_FEATURE_RSTP	0x10
SYSCONF_FEATURE_URLINK	0x11
SYSCONF_FEATURE_SMTP	0x12
SYSCONF_FEATURE_OEM4	0xFA
SYSCONF_FEATURE_OEM3	0xFB
SYSCONF_FEATURE_OEM2	0xFC
SYSCONF_FEATURE_OEM1	0xFD
SYSCONF_FEATURE_OEM	0xFE

Table 37: IDs supported in SDK package

Note: Feature settings are available only when the specified functions are available in the SDK support list.

9.1 *Read configurations from shared memory*

API Name	Int SysConf_Get_Shmcfg(unsigned char u8Id, void *pConf)
Descriptions	Based on feature ID to read configurations from shared memory to pConf
Input	u8Id: Feature ID
Output	pConf: pointer of buffer
Return	0: Success; -1: Failed

Example	#include “shmapi.h”
	#include “net_conf.h”
	
	…
	NET_CONFIG conf[MAX_NIC_UMBER];
	
	// Read NET configuration from shared memory
	SysConf_Get_Shmcfg(SYSCONF_FEATURE_NET, &conf[0]);
	…

Table 38: Read config from shared memory

API Name	Int SysConf_Shm_GetKey(unsigned char u8Id, void *pConf, char *key, char *val)
Descriptions	Based on feature ID and configuration pointer (pConf) to get value of specified key
Input	u8Id: Feature ID pConf: pointer of feature configurations key: Key string in INI file
Output	val: key value
Return	0: Success; -1: Failed

Example	#include “shmapi.h”
	#include “net_conf.h”
	NET_CONFIG conf[MAX_NIC_NUMBER];
	char ip[16] = {0};
	
	// Read NET configuration from shared memory
	SysConf_Get_Shmcfg(SYSCONF_FEATURE_NET, &conf[0]);
	…
	// Read IPv4 Address
	SysConf_Shm_GetKey(SYSCONF_FEATURE_NET, &conf[0], NET_KEY_IP4_ADDR, ip);

Table 39: Read config from shared memory

API Name	Int SysConf_Get_ShmKey(unsigned char u8Id, unsigned char section, char *key, char *val)
Descriptions	Based on feature ID and section index to read value of specified key from shared memory
Input	u8Id: Feature ID section: Section index in INI file key: Key string in INI file
Output	val: key value
Return	0: Success; -1: Failed

Example	#include “shmapi.h”
	#include “system_conf.h”
	
	…
	char user[32] = {0};
	char pass[32] = {0};
	…
	// Read user name and password
	SysConf_Get_ShmKey(SYSCONF_FEATURE_SYSTEM, 0, SYSTEM_KEY_USERNAME, user);
	SysConf_Get_ShmKey(SYSCONF_FEATURE_SYSTEM, 0, SYSTEM_KEY_PASSWORD, pass);
	…

Table 40: Read config from shared memory

9.2 *Set configuration to shared memory*

API Name	Int SysConf_Shm_SetKey(unsigned char u8Id, void *pConf, char *key, char *val)
Descriptions	Based on feature ID to set value of specified key to pConf
Input	u8Id: Feature ID pConf: pointer of feature configurations key: Key string in INI file val: key value
Output	NA
Return	0: Success; -1: Failed

Example	#include “shmapi.h”
	#include “net_conf.h”
	
	…
	NET_CONFIG conf[MAX_NIC_ NUMBER];
	char ip[16] = “192.168.5.123”;
	
	// Read original NET configuration from shared memory
	SysConf_Get_Shmcfg(SYSCONF_FEATURE_NET, &conf[0]);
	…
	// Update IPv4 Address
	SysConf_Shm_SetKey(SYSCONF_FEATURE_NET, &conf[0], NET_KEY_IP4_ADDR, ip);
	…

Table 41: Set config to shared memory

API Name	Int SysConf_Set_Shmcfg(unsigned char u8Id, void *pConf)
Descriptions	Based on feature ID to write configurations to shared memory
Input	u8Id: Feature ID pConf: pointer of buffer
Output	NA
Return	0: Success; -1: Failed

Example	#include “shmapi.h”
	#include “system_conf.h”
	
	…
	NET_CONFIG conf[MAX_NIC_ NUMBER];
	char ip[16] = “192.168.5.123”;
	
	// Read original NET configuration from shared memory
	SysConf_Get_Shmcfg(SYSCONF_FEATURE_NET, &conf[0]);
	…
	// Update IPv4 Address
	SysConf_Shm_SetKey(SYSCONF_FEATURE_NET, &conf[0], NET_KEY_IP4_ADDR, ip);
	…
	// Update Configurations to shared memory
	SysConf_Set_Shmcfg(SYSCONF_FEATURE_NET, &conf[0]);

Table 42: Set config to shared memory

9.3 *Update Configurations to INI files*

API Name	Int SysConf_Update_Shmcfg(unsigned char u8Id, void *pConf)
Descriptions	Based on feature ID to update feature configurations to shared memory and INI file
Input	u8Id: Feature ID pConf: pointer of buffer
Output	NA
Return	0: Success; -1: Failed

Example	#include “shmapi.h”
	#include “com_conf.h”
	
	…
	
	// Change COM port mode to RS-232
	SysConf_Shm_SetKey(SYSCONF_FEATURE_COM, &conf[0], COM_KEY_MODE, “0”);
	…
	// Update IPv4 Address
	SysConf_Shm_SetKey(SYSCONF_FEATURE_NET, &conf[0], NET_KEY_IP4_ADDR, ip);
	…
	// Update Configurations to shared memory and INI file simutaneously
	SysConf_Update_Shmcfg(SYSCONF_FEATURE_NET, &conf[0]);

Table 43: Update config to INI files

API Name	Int SysConf_Update_ShmKey(unsigned char u8Id, unsigned char section, char *key, char *value)
Descriptions	Based on feature ID to update key value to shared memory and INI file
Input	u8Id: Feature ID pConf: pointer of buffer
Output	NA
Return	0: Success; -1: Failed

Example	#include “shmapi.h”
	#include “com_conf.h”
	// Update Password:”12345678” to shared memory and INI file
	SysConf_Update_ShmKey(SYSCONF_FEATURE_SYSTEM, 0, SYSTEM_KEY_PASSWORD, “12345678”);

Table 44: Update config to INI files

9.4 *Add new configurations*

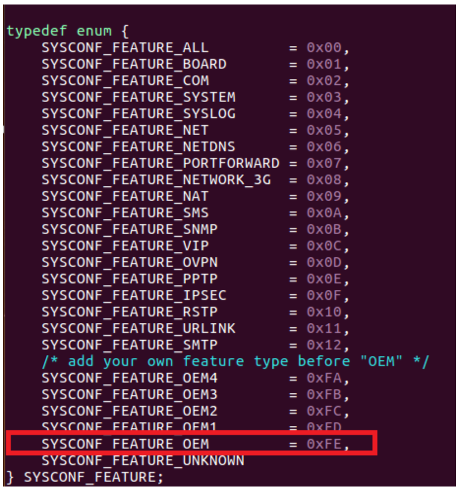
The SDK package already provides an example for users to easily implement the INI feature settings. Here are the following steps which need to be undertaken:

1. Example of feature configurations:

   <sdk>/software/include/sysconf.h
   <sdk>/software/library/conf/oem_conf.h
   <sdk>/software/library/conf/conf_handler.c (required while adding/modifying the feature name)

2. Define the feature ID in "<sdk>/software/include/sysconf.h"

   # vi <sdk>/software/include/sysconf.h

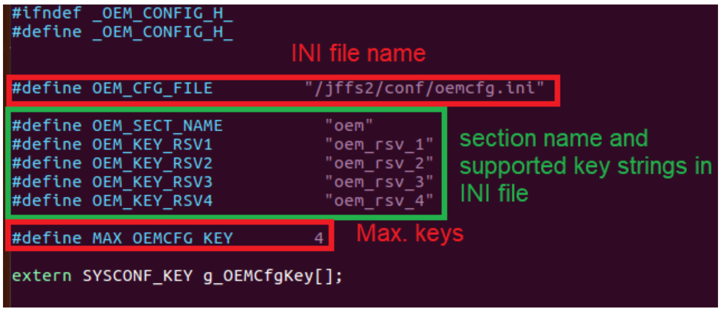


```
typedef enum {
    SYSCONF_FEATURE_ALL          = 0x00,
    SYSCONF_FEATURE_BOARD        = 0x01,
    SYSCONF_FEATURE_COM          = 0x02,
    SYSCONF_FEATURE_SYSTEM       = 0x03,
    SYSCONF_FEATURE_SYSLOG       = 0x04,
    SYSCONF_FEATURE_NET          = 0x05,
    SYSCONF_FEATURE_NETDNS       = 0x06,
    SYSCONF_FEATURE_PORTFORWARD  = 0x07,
    SYSCONF_FEATURE_NETWORK_3G   = 0x08,
    SYSCONF_FEATURE_NAT          = 0x09,
    SYSCONF_FEATURE_SMS          = 0x0A,
    SYSCONF_FEATURE_SNMP         = 0x0B,
    SYSCONF_FEATURE_VIP          = 0x0C,
    SYSCONF_FEATURE_OVPN         = 0x0D,
    SYSCONF_FEATURE_PPTP         = 0x0E,
    SYSCONF_FEATURE_IPSEC        = 0x0F,
    SYSCONF_FEATURE_RSTP         = 0x10,
    SYSCONF_FEATURE_URLINK       = 0x11,
    SYSCONF_FEATURE_SMTP         = 0x12,
    /* add your own feature type before "OEM" */
    SYSCONF_FEATURE_OEM4         = 0xFA,
    SYSCONF_FEATURE_OEM3         = 0xFB,
    SYSCONF_FEATURE_OEM2         = 0xFC,
    SYSCONF_FEATURE_OEM1         = 0xFD,
    SYSCONF_FEATURE_OEM          = 0xFE,
    SYSCONF_FEATURE_UNKNOWN
} SYSCONF_FEATURE;
```

Figure 20: Defining feature ID

3. Define the feature section name and supported keys in "<sdk>/software/include/oem_conf.h"

   # vi <sdk>/software/include/oem_conf.h

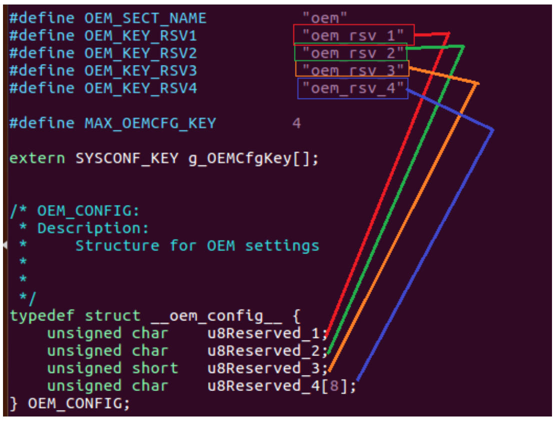


```
#ifndef _OEM_CONFIG_H_
#define _OEM_CONFIG_H_

#define OEM_CFG_FILE              "/jffs2/conf/oemcfg.ini"

#define OEM_SECT_NAME               "oem"
#define OEM_KEY_RSV1                "oem_rsv_1"
#define OEM_KEY_RSV2                "oem_rsv_2"
#define OEM_KEY_RSV3                "oem_rsv_3"
#define OEM_KEY_RSV4                "oem_rsv_4"

#define MAX_OEMCFG_KEY              4

extern SYSCONF_KEY g_OEMCfgKey[];
```

Figure 21: Defining feature section name

4. Define a structure to handle settings in "<sdk>/software/include/oem_conf.h"

# vi <sdk>/software/include/oem_conf.h

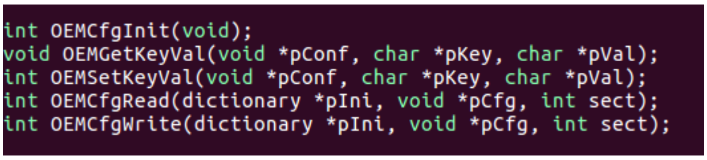


```
#define OEM_SECT_NAME             "oem"
#define OEM_KEY_RSV1              "oem_rsv_1"
#define OEM_KEY_RSV2              "oem_rsv_2"
#define OEM_KEY_RSV3              "oem_rsv_3"
#define OEM_KEY_RSV4              "oem_rsv_4"

#define MAX_OEMCFG_KEY          4

extern SYSCONF_KEY g_OEMCfgKey[];


/* OEM_CONFIG:
 * Description:
 *      Structure for OEM settings
 *
 *
 */
typedef struct __oem_config__ {
    unsigned char     u8Reserved_1;
    unsigned char     u8Reserved_2;
    unsigned short    u8Reserved_3;
    unsigned char     u8Reserved_4[8];
} OEM_CONFIG;
```

Figure 22: Defining structure to handle SDK settings

5. Define function names that are used to init/read/write feature settings

   # vi <sdk>/software/include/oem_conf.h

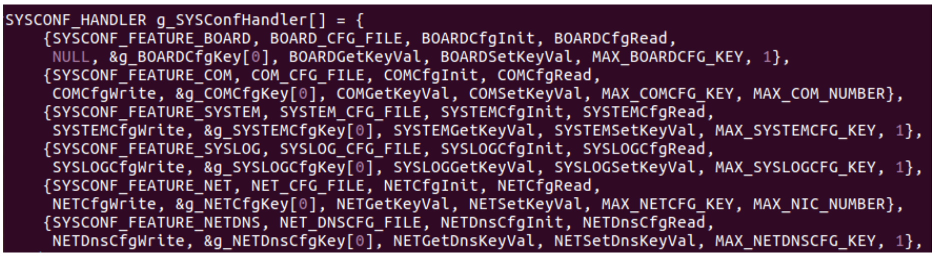


```
int OEMCfgInit(void);
void OEMGetKeyVal(void *pConf, char *pKey, char *pVal);
int OEMSetKeyVal(void *pConf, char *pKey, char *pVal);
int OEMCfgRead(dictionary *pIni, void *pCfg, int sect);
int OEMCfgWrite(dictionary *pIni, void *pCfg, int sect);
```

Figure 23: Defining function names for feature settings

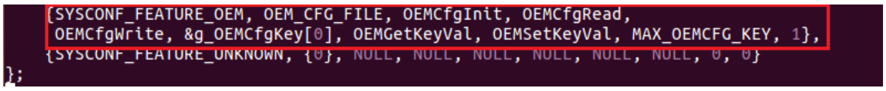
6. Add the ID of "SYSCONF_FEATURE_OEM" in gSYSConfHadnler[] in

   "<sdk>/software/library/conf/conf_handler.c"



```
SYSCONF_HANDLER g_SYSConfHandler[] = {
    {SYSCONF_FEATURE_BOARD, BOARD_CFG_FILE, BOARDCfgInit, BOARDCfgRead,
     NULL, &g_BOARDCfgKey[0], BOARDGetKeyVal, BOARDSetKeyVal, MAX_BOARDCFG_KEY, 1},
    {SYSCONF_FEATURE_COM, COM_CFG_FILE, COMCfgInit, COMCfgRead,
     COMCfgWrite, &g_COMCfgKey[0], COMGetKeyVal, COMSetKeyVal, MAX_COMCFG_KEY, MAX_COM_NUMBER},
    {SYSCONF_FEATURE_SYSTEM, SYSTEM_CFG_FILE, SYSTEMCfgInit, SYSTEMCfgRead,
     SYSTEMCfgWrite, &g_SYSTEMCfgKey[0], SYSTEMGetKeyVal, SYSTEMSetKeyVal, MAX_SYSTEMCFG_KEY, 1},
    {SYSCONF_FEATURE_SYSLOG, SYSLOG_CFG_FILE, SYSLOGCfgInit, SYSLOGCfgRead,
     SYSLOGCfgWrite, &g_SYSLOGCfgKey[0], SYSLOGGetKeyVal, SYSLOGSetKeyVal, MAX_SYSLOGCFG_KEY, 1},
    {SYSCONF_FEATURE_NET, NET_CFG_FILE, NETCfgInit, NETCfgRead,
     NETCfgWrite, &g_NETCfgKey[0], NETGetKeyVal, NETSetKeyVal, MAX_NETCFG_KEY, MAX_NIC_NUMBER},
    {SYSCONF_FEATURE_NETDNS, NET_DNSCFG_FILE, NETDnsCfgInit, NETDnsCfgRead,
     NETDnsCfgWrite, &g_NETDnsCfgKey[0], NETGetDnsKeyVal, NETSetDnsKeyVal, MAX_NETDNSCFG_KEY, 1},
```

Figure 24: Add sysconfig ID

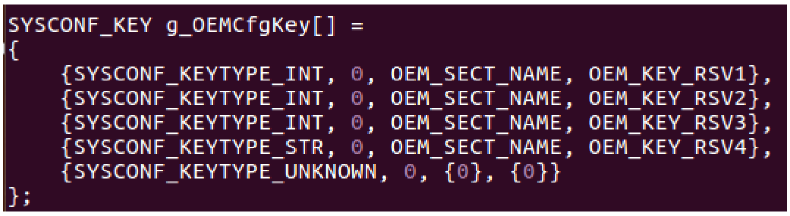


```
    {SYSCONF_FEATURE_OEM, OEM_CFG_FILE, OEMCfgInit, OEMCfgRead,
     OEMCfgWrite, &g_OEMCfgKey[0], OEMGetKeyVal, OEMSetKeyVal, MAX_OEMCFG_KEY, 1},
    {SYSCONF_FEATURE_UNKNOWN, {0}, NULL, NULL, NULL, NULL, NULL, NULL, 0, 0}
};
```

Figure 25: Add sysconfig ID

7. Define the key mapping table in "<sdk>/software/library/conf/oemconf.c"

# vi <sdk>/software/library/conf/oemconf.c

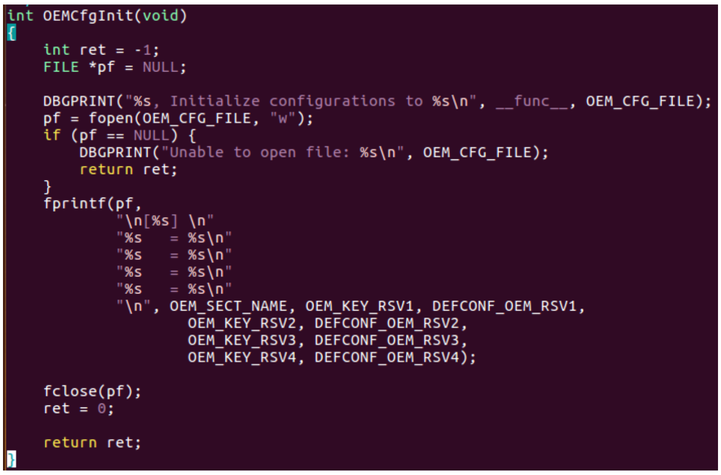


```
SYSCONF_KEY g_OEMCfgKey[] =
{
    {SYSCONF_KEYTYPE_INT, 0, OEM_SECT_NAME, OEM_KEY_RSV1},
    {SYSCONF_KEYTYPE_INT, 0, OEM_SECT_NAME, OEM_KEY_RSV2},
    {SYSCONF_KEYTYPE_INT, 0, OEM_SECT_NAME, OEM_KEY_RSV3},
    {SYSCONF_KEYTYPE_STR, 0, OEM_SECT_NAME, OEM_KEY_RSV4},
    {SYSCONF_KEYTYPE_UNKNOWN, 0, {0}, {0}}
};
```

Figure 26: Defining key mapping table

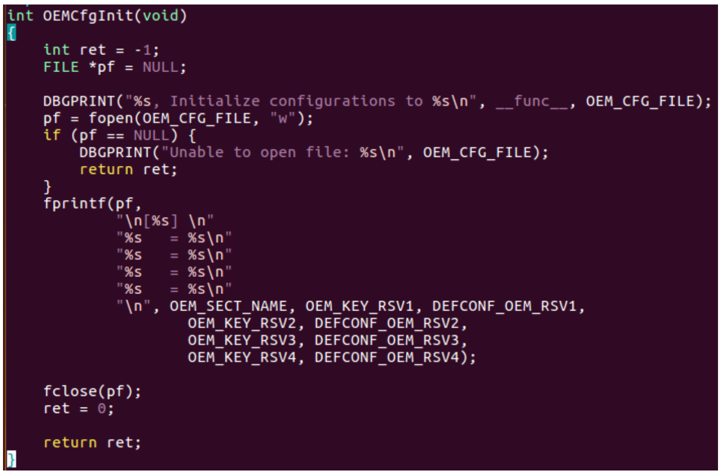
8. Implement init function in“<sdk>/software/library/conf/oemconf.c”

# vi <sdk>/software/library/conf/oemconf.c



```
int OEMCfgInit(void)
{
    int ret = -1;
    FILE *pf = NULL;

    DBGPRINT("%s, Initialize configurations to %s\n", __func__, OEM_CFG_FILE);
    pf = fopen(OEM_CFG_FILE, "w");
    if (pf == NULL) {
        DBGPRINT("Unable to open file: %s\n", OEM_CFG_FILE);
        return ret;
    }
    fprintf(pf,
            "\n[%s] \n"
            "%s   = %s\n"
            "%s   = %s\n"
            "%s   = %s\n"
            "%s   = %s\n"
            "\n", OEM_SECT_NAME, OEM_KEY_RSV1, DEFCONF_OEM_RSV1,
                OEM_KEY_RSV2, DEFCONF_OEM_RSV2,
                OEM_KEY_RSV3, DEFCONF_OEM_RSV3,
                OEM_KEY_RSV4, DEFCONF_OEM_RSV4);

    fclose(pf);
    ret = 0;

    return ret;
}
```

Figure 27: Implementing init function

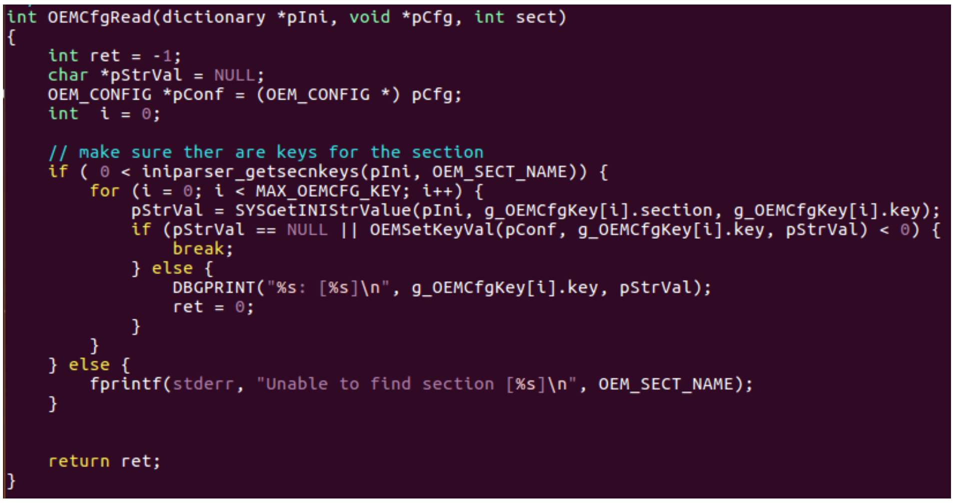


```
int OEMCfgInit(void)
{
    int ret = -1;
    FILE *pf = NULL;

    DBGPRINT("%s, Initialize configurations to %s\n", __func__, OEM_CFG_FILE);
    pf = fopen(OEM_CFG_FILE, "w");
    if (pf == NULL) {
        DBGPRINT("Unable to open file: %s\n", OEM_CFG_FILE);
        return ret;
    }
    fprintf(pf,
            "\n[%s] \n"
            "%s   = %s\n"
            "%s   = %s\n"
            "%s   = %s\n"
            "%s   = %s\n"
            "\n", OEM_SECT_NAME, OEM_KEY_RSV1, DEFCONF_OEM_RSV1,
                OEM_KEY_RSV2, DEFCONF_OEM_RSV2,
                OEM_KEY_RSV3, DEFCONF_OEM_RSV3,
                OEM_KEY_RSV4, DEFCONF_OEM_RSV4);

    fclose(pf);
    ret = 0;

    return ret;
}
```

Figure 28: Implementing init function

9. Implement read function in“<sdk>/software/library/conf/oemconf.c”

# vi <sdk>/software/library/conf/oemconf.c

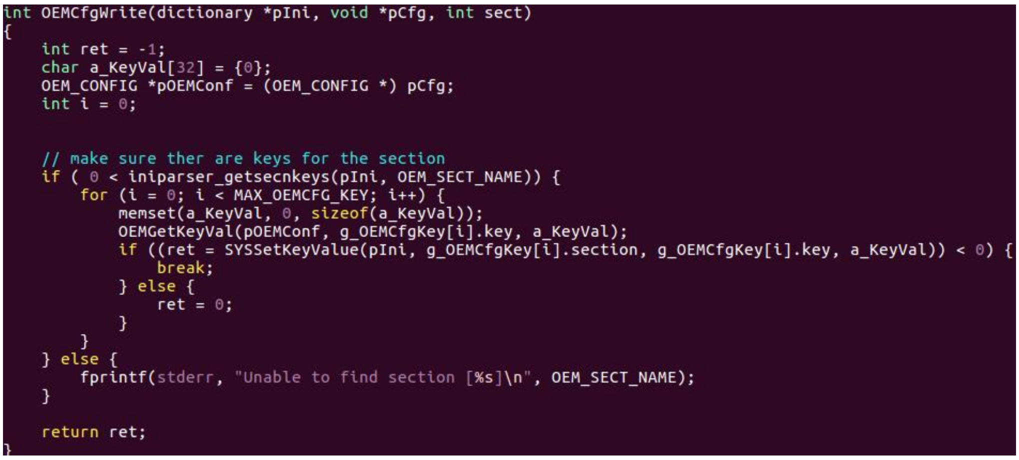


```
int OEMCfgRead(dictionary *pIni, void *pCfg, int sect)
{
    int ret = -1;
    char *pStrVal = NULL;
    OEM_CONFIG *pConf = (OEM_CONFIG *) pCfg;
    int  i = 0;

    // make sure ther are keys for the section
    if ( 0 < iniparser_getsecnkeys(pIni, OEM_SECT_NAME)) {
        for (i = 0; i < MAX_OEMCFG_KEY; i++) {
            pStrVal = SYSGetINIStrValue(pIni, g_OEMCfgKey[i].section, g_OEMCfgKey[i].key);
            if (pStrVal == NULL || OEMSetKeyVal(pConf, g_OEMCfgKey[i].key, pStrVal) < 0) {
                break;
            } else {
                DBGPRINT("%s: [%s]\n", g_OEMCfgKey[i].key, pStrVal);
                ret = 0;
            }
        }
    } else {
        fprintf(stderr, "Unable to find section [%s]\n", OEM_SECT_NAME);
    }


    return ret;
}
```

Figure 29: Implementing read function

10. Implement write function in“<sdk>/software/library/conf/oemconf.c”

# vi <sdk>/software/library/conf/oemconf.c

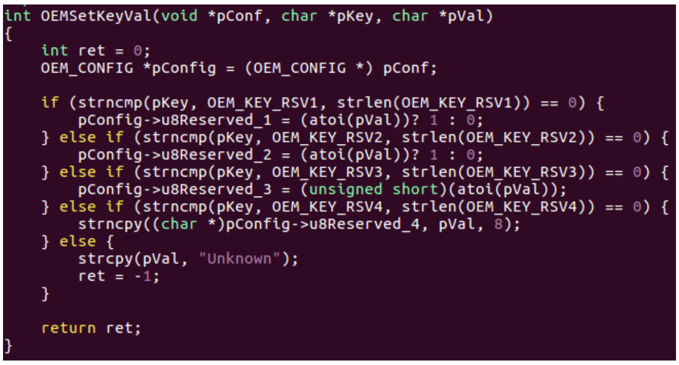


```
int OEMCfgWrite(dictionary *pIni, void *pCfg, int sect)
{
    int ret = -1;
    char a_KeyVal[32] = {0};
    OEM_CONFIG *pOEMConf = (OEM_CONFIG *) pCfg;
    int i = 0;


    // make sure ther are keys for the section
    if ( 0 < iniparser_getsecnkeys(pIni, OEM_SECT_NAME)) {
        for (i = 0; i < MAX_OEMCFG_KEY; i++) {
            memset(a_KeyVal, 0, sizeof(a_KeyVal));
            OEMGetKeyVal(pOEMConf, g_OEMCfgKey[i].key, a_KeyVal);
            if ((ret = SYSSetKeyValue(pIni, g_OEMCfgKey[i].section, g_OEMCfgKey[i].key, a_KeyVal)) < 0) {
                break;
            } else {
                ret = 0;
            }
        }
    } else {
        fprintf(stderr, "Unable to find section [%s]\n", OEM_SECT_NAME);
    }

    return ret;
}
```

Figure 30: Implementing write function

11. Implement key get function in“<sdk>/software/library/conf/oemconf.c”

# vi <sdk>/software/library/conf/oemconf.c

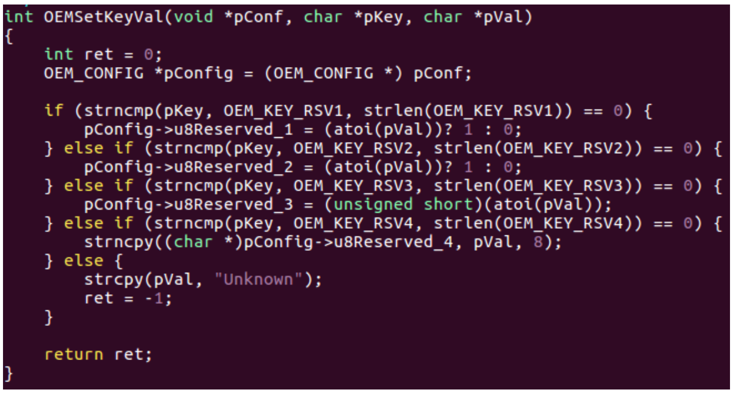


```
int OEMSetKeyVal(void *pConf, char *pKey, char *pVal)
{
    int ret = 0;
    OEM_CONFIG *pConfig = (OEM_CONFIG *) pConf;

    if (strncmp(pKey, OEM_KEY_RSV1, strlen(OEM_KEY_RSV1)) == 0) {
        pConfig->u8Reserved_1 = (atoi(pVal))? 1 : 0;
    } else if (strncmp(pKey, OEM_KEY_RSV2, strlen(OEM_KEY_RSV2)) == 0) {
        pConfig->u8Reserved_2 = (atoi(pVal))? 1 : 0;
    } else if (strncmp(pKey, OEM_KEY_RSV3, strlen(OEM_KEY_RSV3)) == 0) {
        pConfig->u8Reserved_3 = (unsigned short)(atoi(pVal));
    } else if (strncmp(pKey, OEM_KEY_RSV4, strlen(OEM_KEY_RSV4)) == 0) {
        strncpy((char *)pConfig->u8Reserved_4, pVal, 8);
    } else {
        strcpy(pVal, "Unknown");
        ret = -1;
    }

    return ret;
}
```

Figure 31: Implementing key get function

12. Implement key set function in“<sdk>/software/library/conf/oemconf.c”

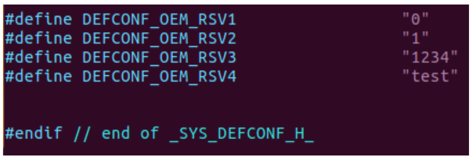
# vi <sdk>/software/library/conf/oemconf.c



```
int OEMSetKeyVal(void *pConf, char *pKey, char *pVal)
{
    int ret = 0;
    OEM_CONFIG *pConfig = (OEM_CONFIG *) pConf;

    if (strncmp(pKey, OEM_KEY_RSV1, strlen(OEM_KEY_RSV1)) == 0) {
        pConfig->u8Reserved_1 = (atoi(pVal))? 1 : 0;
    } else if (strncmp(pKey, OEM_KEY_RSV2, strlen(OEM_KEY_RSV2)) == 0) {
        pConfig->u8Reserved_2 = (atoi(pVal))? 1 : 0;
    } else if (strncmp(pKey, OEM_KEY_RSV3, strlen(OEM_KEY_RSV3)) == 0) {
        pConfig->u8Reserved_3 = (unsigned short)(atoi(pVal));
    } else if (strncmp(pKey, OEM_KEY_RSV4, strlen(OEM_KEY_RSV4)) == 0) {
        strncpy((char *)pConfig->u8Reserved_4, pVal, 8);
    } else {
        strcpy(pVal, "Unknown");
        ret = -1;
    }

    return ret;
}
```

Figure 32: Implementing key set function

13. Edit default configurations in <sdk>/config/<targte>/defconf.h



```
#define DEFCONF_OEM_RSV1                        "0"
#define DEFCONF_OEM_RSV2                        "1"
#define DEFCONF_OEM_RSV3                        "1234"
#define DEFCONF_OEM_RSV4                        "test"


#endif // end of _SYS_DEFCONF_H_
```

Figure 33: Edit default configurations

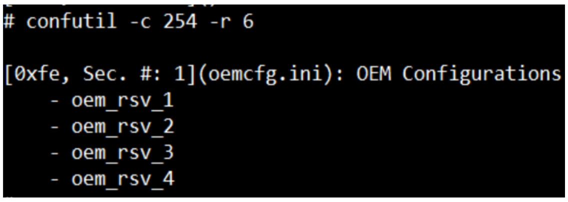
14. Compile the software to make sure no error happened

# make swbuild

15. Build the system and burn to target device

16. Open debug console, and run this command to check if feature settings are working

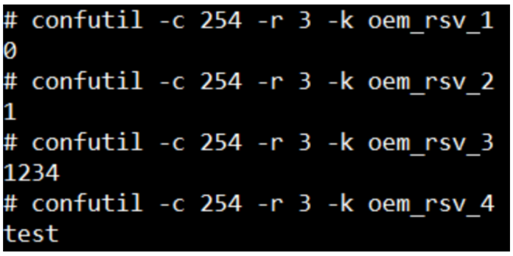
# confutil -c 254 -r 6



```
# confutil -c 254 -r 6

[0xfe, Sec. #: 1](oemcfg.ini): OEM Configurations
    - oem_rsv_1
    - oem_rsv_2
    - oem_rsv_3
    - oem_rsv_4
```

Figure 34: Running commands in open debug console

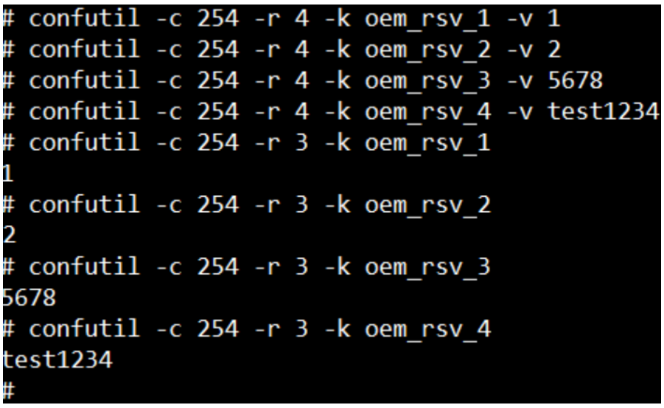
17. Run these commands to check each key values from debug console:



```
# confutil -c 254 -r 3 -k oem_rsv_1
0
# confutil -c 254 -r 3 -k oem_rsv_2
1
# confutil -c 254 -r 3 -k oem_rsv_3
1234
# confutil -c 254 -r 3 -k oem_rsv_4
test
```

Figure 35: Running commands in open debug console

18. Running these commands to change key values and check if the function works:

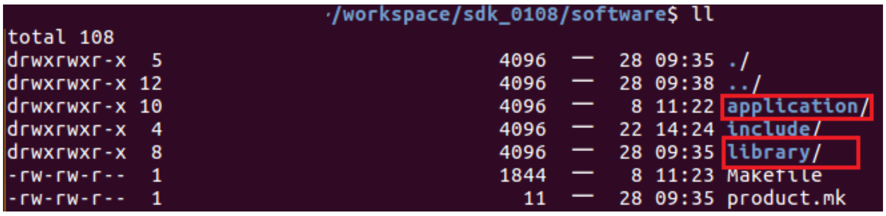


```
# confutil -c 254 -r 4 -k oem_rsv_1 -v 1
# confutil -c 254 -r 4 -k oem_rsv_2 -v 2
# confutil -c 254 -r 4 -k oem_rsv_3 -v 5678
# confutil -c 254 -r 4 -k oem_rsv_4 -v test1234
# confutil -c 254 -r 3 -k oem_rsv_1
1
# confutil -c 254 -r 3 -k oem_rsv_2
2
# confutil -c 254 -r 3 -k oem_rsv_3
5678
# confutil -c 254 -r 3 -k oem_rsv_4
test1234
#
```

Figure 36: Running commands in open debug console

10 Software

The software folder in SDK collects common libraries and applications.



```
                                    ·/workspace/sdk_0108/software$ ll
total 108
drwxrwxr-x  5                                    4096  —  28 09:35 ./
drwxrwxr-x 12                                    4096  —  28 09:38 ../
drwxrwxr-x 10                                    4096  —   8 11:22 application/
drwxrwxr-x  4                                    4096  —  22 14:24 include/
drwxrwxr-x  8                                    4096  —  28 09:35 library/
-rw-rw-r--  1                                    1844  —   8 11:23 Makefile
-rw-rw-r--  1                                      11  —  28 09:35 product.mk
```

Figure 37: Software folder in SDK

10.1 *Application*

Folder	Descriptions
system	This folder collects common applications and scripts
utils	This folder collects diagnostic tools

Table 45: Software application folder

10.2 *Library*

Folder	Descriptions
common	This folder collects common libraries, such as platform IO access
conf	This folder collects the libraries related to the INI files access
eeprom	This folder collects the libraries related to the EEPROM access
firewall	This folder collects the libraries related to the Firewall operations
mobile	This folder collects the libraries related to the 3G/4G module control

Table 46: Software library folder

11 Web

SDK package supports the simple WEB server with “lighttpd”.(Defaul Address: http://10.0.50.100 or https://192.168.1.100) The CGI files and WEB pages are placed in these directories:

- CGI Files: <SDK>/webs/lighttpd/cgi/
- WEB Pages: <SDK>/webs/lighttpd/web_pages/

11.1 *Web Account/Password*

Default WEB account and password:

- User Name: admin
- Password: default

11.2 *Change Web Logo*

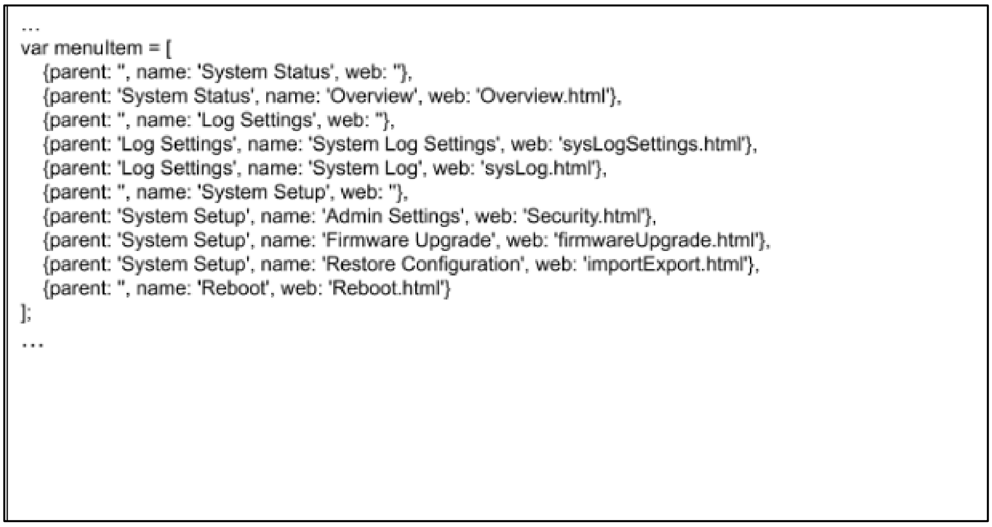
Please replacing the default image file with your own logo:

<sdk>/webs/lighttpd/web_pages/images/logo.gif

- Width: 200px
- Height: 48px

11.3 *Add a new page in selection menu*

Ref File: <sd>/webs/lighttpd/web_pages/javascript/quickmenu.js

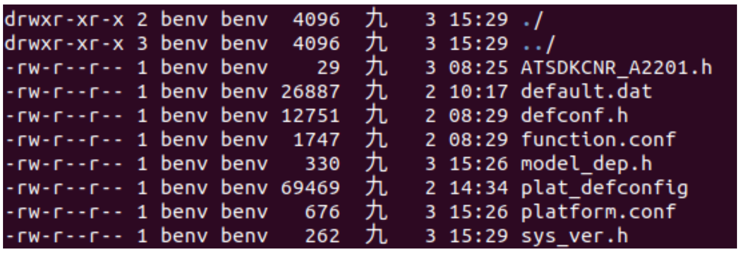


```
...
var menuItem = [
    {parent: '', name: 'System Status', web: ''},
    {parent: 'System Status', name: 'Overview', web: 'Overview.html'},
    {parent: '', name: 'Log Settings', web: ''},
    {parent: 'Log Settings', name: 'System Log Settings', web: 'sysLogSettings.html'},
    {parent: 'Log Settings', name: 'System Log', web: 'sysLog.html'},
    {parent: '', name: 'System Setup', web: ''},
    {parent: 'System Setup', name: 'Admin Settings', web: 'Security.html'},
    {parent: 'System Setup', name: 'Firmware Upgrade', web: 'firmwareUpgrade.html'},
    {parent: 'System Setup', name: 'Restore Configuration', web: 'importExport.html'},
    {parent: '', name: 'Reboot', web: 'Reboot.html'}
];
...
```

Figure 38: Add new page in selection menu

12 System

In SDK repository, most configurations are placed under “***<SDK>/config/<Target>/***” folder. The figure below illustrates contents of target configurations:

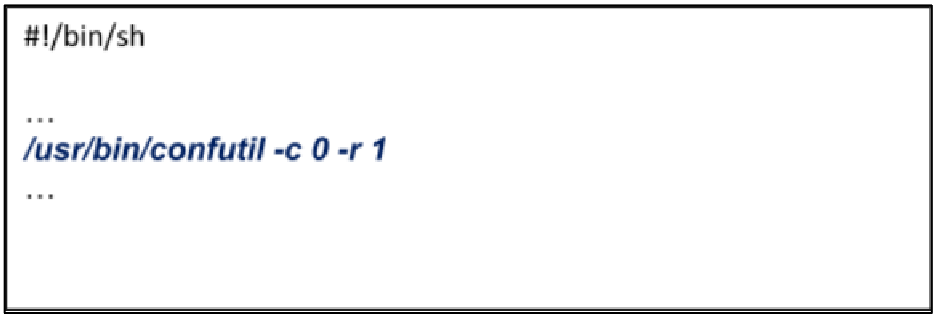


```
drwxr-xr-x 2 benv benv  4096 九  3 15:29 ./
drwxr-xr-x 3 benv benv  4096 九  3 15:29 ../
-rw-r--r-- 1 benv benv    29 九  3 08:25 ATSDKCNR_A2201.h
-rw-r--r-- 1 benv benv 26887 九  2 10:17 default.dat
-rw-r--r-- 1 benv benv 12751 九  2 08:29 defconf.h
-rw-r--r-- 1 benv benv  1747 九  2 08:29 function.conf
-rw-r--r-- 1 benv benv   330 九  3 15:26 model_dep.h
-rw-r--r-- 1 benv benv 69469 九  2 14:34 plat_defconfig
-rw-r--r-- 1 benv benv   676 九  3 15:26 platform.conf
-rw-r--r-- 1 benv benv   262 九  3 15:29 sys_ver.h
```

Figure 39: System target configurations

12.1 *System start script files*

- In user space, system start-up scripts are put under “<sdk>/filesystem/etc/init.d/”
- System initializes all features’ settings in ***S01logging***



```
#!/bin/sh

...
/usr/bin/confutil -c 0 -r 1
...
```

Figure 40: System start script files

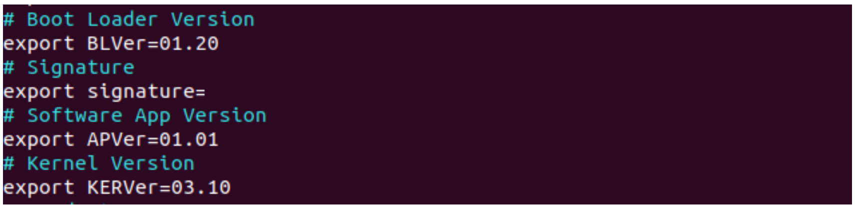
- ATOP’s main initial flow is implemented in ***S21SysInit***

12.2 *Account and password of Debug Console*

- User Name: root
- Password: NULL

12.3 *Change System Version Information*

Ref: <SDK>/config/<target>/platform.conf



```
# Boot Loader Version
export BLVer=01.20
# Signature
export signature=
# Software App Version
export APVer=01.01
# Kernel Version
export KERVer=03.10
```

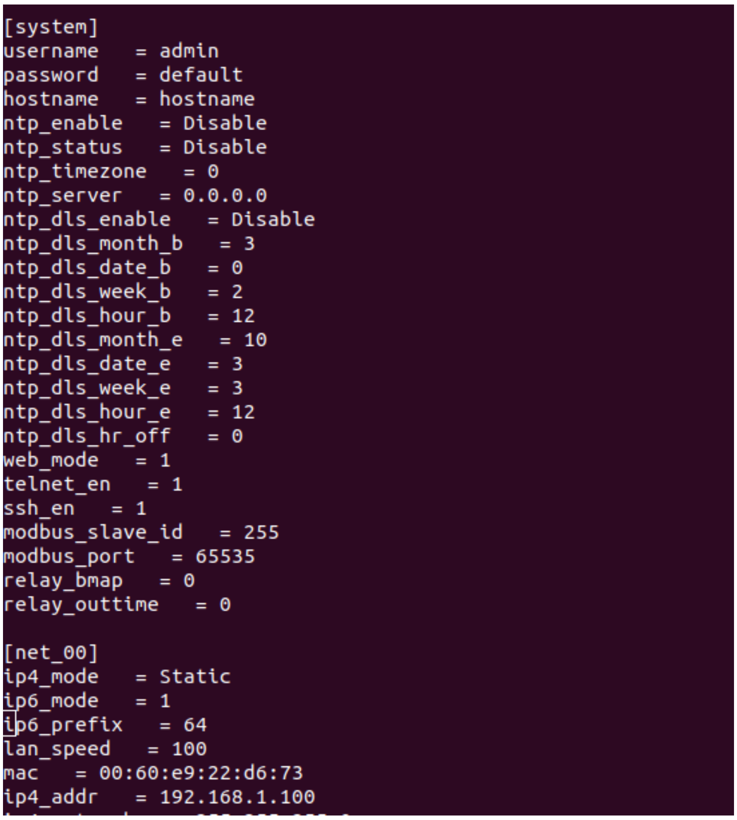
Figure 41: Changing system version information

Run this command to rebuild the library, and update version information:

# make swbuild; make image;
# make swbuild; make fwimg;

12.4 *Platform Default Configurations*

The initial system settings of the project are configured in “<SDK>/config/<Target>/default.dat”. Users can easily change each project’s default settings in this file.



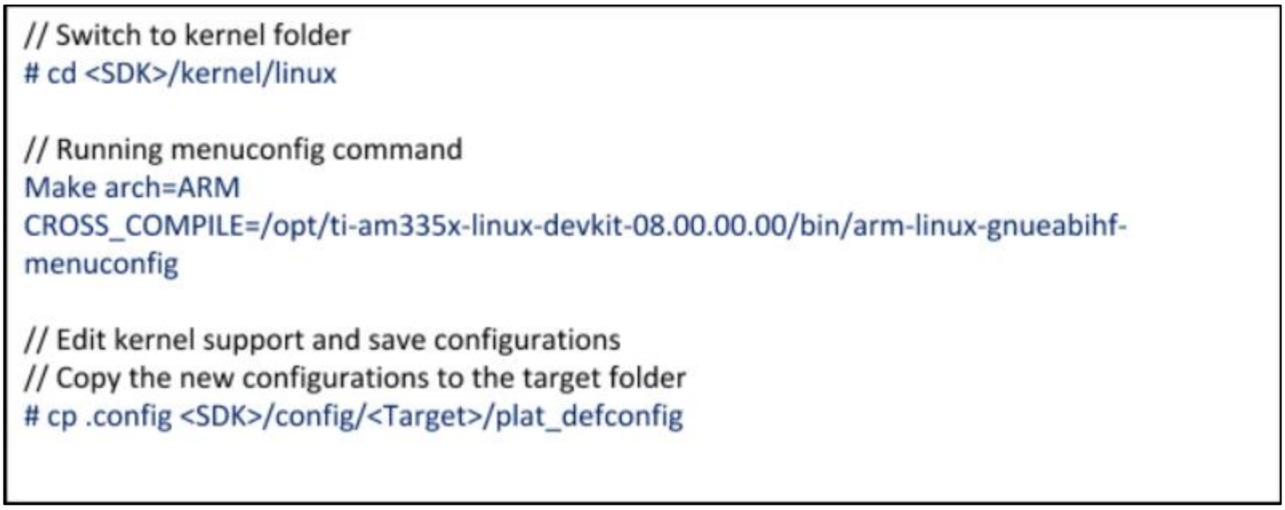
```
[system]
username    = admin
password    = default
hostname    = hostname
ntp_enable    = Disable
ntp_status    = Disable
ntp_timezone    = 0
ntp_server    = 0.0.0.0
ntp_dls_enable    = Disable
ntp_dls_month_b    = 3
ntp_dls_date_b    = 0
ntp_dls_week_b    = 2
ntp_dls_hour_b    = 12
ntp_dls_month_e    = 10
ntp_dls_date_e    = 3
ntp_dls_week_e    = 3
ntp_dls_hour_e    = 12
ntp_dls_hr_off    = 0
web_mode    = 1
telnet_en    = 1
ssh_en    = 1
modbus_slave_id    = 255
modbus_port    = 65535
relay_bmap    = 0
relay_outtime    = 0

[net_00]
ip4_mode    = Static
ip6_mode    = 1
ip6_prefix    = 64
lan_speed    = 100
mac    = 00:60:e9:22:d6:73
ip4_addr    = 192.168.1.100
```

Figure 42: Platform default configurations

12.5 *Kernel Configurations*

The project’s kernel configuration file is “***<SDK>/config/<TARGET>/plat_defconfig***”.

To change it, you may execute below commands to enable/disable configurations:



```
// Switch to kernel folder
# cd <SDK>/kernel/linux

// Running menuconfig command
Make arch=ARM
CROSS_COMPILE=/opt/ti-am335x-linux-devkit-08.00.00.00/bin/arm-linux-gnueabihf-
menuconfig

// Edit kernel support and save configurations
// Copy the new configurations to the target folder
# cp .config <SDK>/config/<Target>/plat_defconfig
```

Figure 43: Kernel Configurations

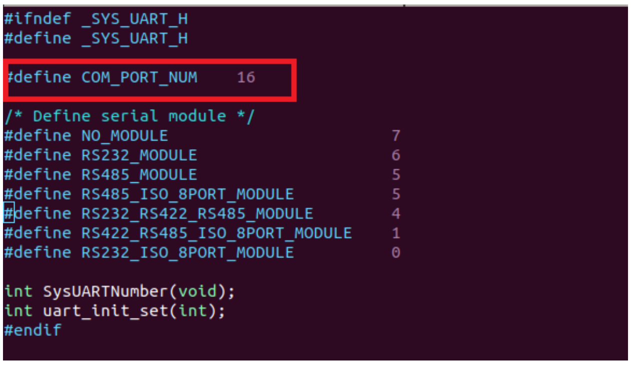
12.6 *Flash Layout*

ATOP SDK limits the modification of flash partition. We do not recommend that users update the flash layout, as the flash partitions are pre-defined in HW configurations. ATOP SDK provides the HW configurations in binary format. If users indeed needed to modify the flash layout, customers can request a layout update before product shipment.

12.7 *Change COM Number*

The physical COM port support varies with platforms. If the physical COM number is not matched with your platform, you can modify it using the below file:

<sdk>/software/include/sys_uart.h



```
#ifndef _SYS_UART_H
#define _SYS_UART_H

#define COM_PORT_NUM    16

/* Define serial module */
#define NO_MODULE                           7
#define RS232_MODULE                        6
#define RS485_MODULE                        5
#define RS485_ISO_8PORT_MODULE              5
#define RS232_RS422_RS485_MODULE            4
#define RS422_RS485_ISO_8PORT_MODULE        1
#define RS232_ISO_8PORT_MODULE              0

int SysUARTNumber(void);
int uart_init_set(int);
#endif
```

Figure 44: Change COM number

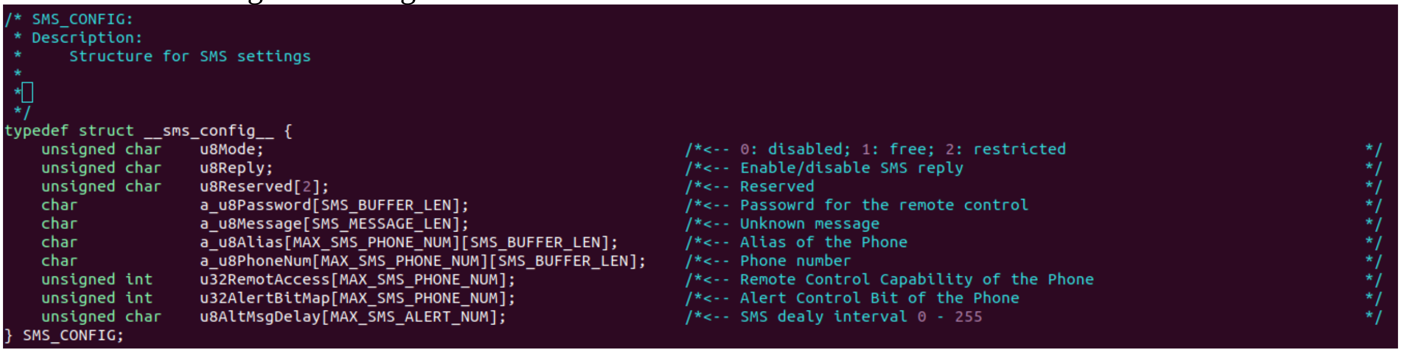
13 SMS Managemet (3G/4G Cellular Only)

ATOP SDK provides a simple mechanism for users to easily manage SMS with sms tools.

Note: Before using it, please make sure the SIM is ready on your device.

13.1 *SMS Settings*

In SDK SMS settings are managed in this structured container:



```
/* SMS_CONFIG:
 * Description:
 *     Structure for SMS settings
 *
 *□
 */
typedef struct __sms_config__ {
    unsigned char    u8Mode;                                             /*<-- 0: disabled; 1: free; 2: restricted                                */
    unsigned char    u8Reply;                                            /*<-- Enable/disable SMS reply                                           */
    unsigned char    u8Reserved[2];                                      /*<-- Reserved                                                           */
    char             a_u8Password[SMS_BUFFER_LEN];                       /*<-- Passowrd for the remote control                                    */
    char             a_u8Message[SMS_MESSAGE_LEN];                       /*<-- Unknown message                                                    */
    char             a_u8Alias[MAX_SMS_PHONE_NUM][SMS_BUFFER_LEN];       /*<-- Alias of the Phone                                                 */
    char             a_u8PhoneNum[MAX_SMS_PHONE_NUM][SMS_BUFFER_LEN];    /*<-- Phone number                                                       */
    unsigned int     u32RemotAccess[MAX_SMS_PHONE_NUM];                  /*<-- Remote Control Capability of the Phone                             */
    unsigned int     u32AlertBitMap[MAX_SMS_PHONE_NUM];                  /*<-- Alert Control Bit of the Phone                                     */
    unsigned char    u8AltMsgDelay[MAX_SMS_ALERT_NUM];                   /*<-- SMS dealy interval 0 - 255                                         */
} SMS_CONFIG;
```

Figure 45: SMS Settings

Field	Description
u8Mode	SMS management mode • 0 : Disable • 1: Free, no limitation • 2: Restricted, only configured phone number is available
u8Reply	Enable/Disable SMS reply when receiving SMS remote control command
a_u8Password	Password of SMS remote control (max. 16 characters)
a_u8Message	Messages to reply when receiving a unknown remote control command (max 64 characters)
a_u8Alias	Alias of phone number (max. 5 phone numbers,)
a_u8PhoneNum	Phone number Default MAX_SMS_PHONE_NUM is 5
a_u8RemoteAccess	Enable/Disable remote control of each phone number
a_u8AlertBitMap	Bit map of alert event for each phone number
a_u8AltMsgDelay	Alert messages delay interval of each alert event.

Table 47: SMS Settings

To access the SMS settings, please refer to the “Ch8. INI Configurations Read/Write (Settings Management)”.

13.2 *SMS Remote Control*

- Control Message Format

#<Password of SMS control>#<SMS control messages>

Example:

- Password of SMS control: “12345678”
- SMS control message: “echo_test”
- Users send remote control message:
  #12345678#echo_test

- SMS Event Handler
  The script file used to handle the SMS event is

  <sdk>/3rdparty/patch/smstools3-3.1.21/scripts/smsevent

- SMS Remote Control Command List:
  The file used to define supported remote SMS control message are stored at below file

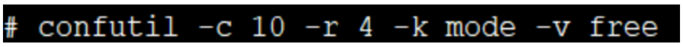
  <sdk>/3rdparty/patch/smstools3-3.1.21/smscmd.lst

13.3 *SMS Alert Messages*

See ***7-9. Alert Message Control*** to get idea how to access SMS settings through APIs.

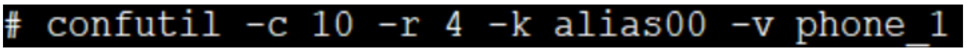
Here are the steps to enable alert messages:

1. Set SMS management mode to “free”



```
# confutil -c 10 -r 4 -k mode -v free
```

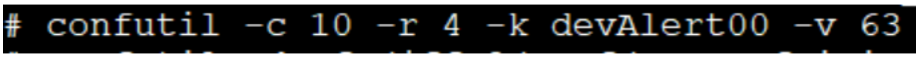
2. Set the alias for the phone 1 as “phone_1”



```
# confutil -c 10 -r 4 -k alias00 -v phone_1
```

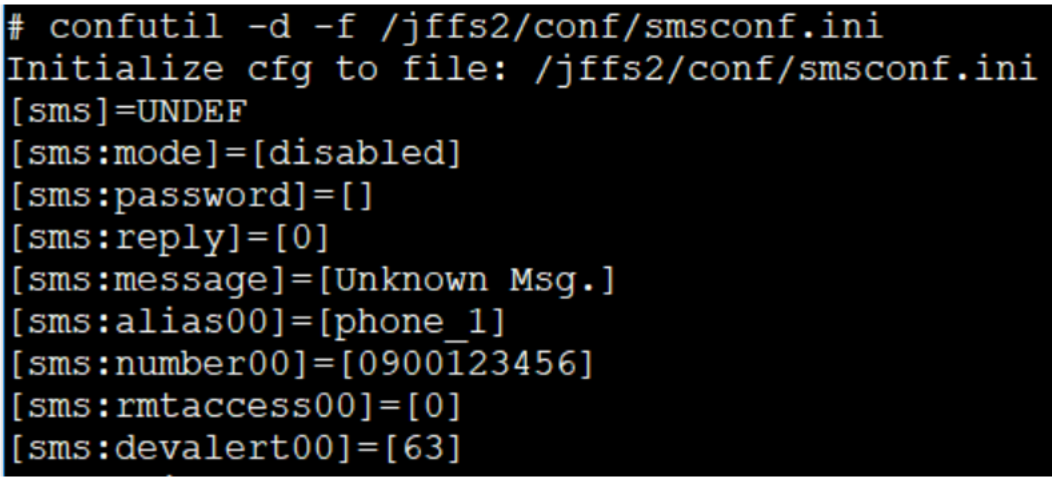
3. Set the phone number for phone 1

4. Set the Alert control to 63 (Bit 0 - 5)



```
# confutil -c 10 -r 4 -k devAlert00 -v 63
```

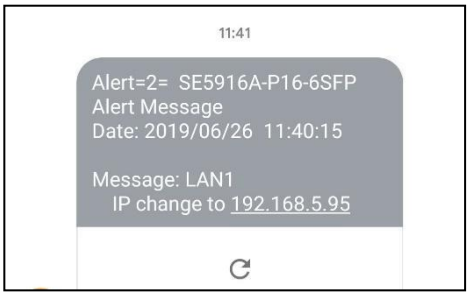
5. Check the configurations:



```
# confutil -d -f /jffs2/conf/smsconf.ini
Initialize cfg to file: /jffs2/conf/smsconf.ini
[sms]=UNDEF
[sms:mode]=[disabled]
[sms:password]=[]
[sms:reply]=[0]
[sms:message]=[Unknown Msg.]
[sms:alias00]=[phone_1]
[sms:number00]=[0900123456]
[sms:rmtaccess00]=[0]
[sms:devalert00]=[63]
```

Figure 46: SMS Configuration

6. Change IP address from your WEB and check if you can receive the alert message:



11:41

Alert=2= SE5916A-P16-6SFP
Alert Message
Date: 2019/06/26 11:40:15

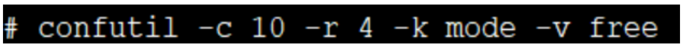
Message: LAN1
IP change to 192.168.5.95

Figure 47: SMS Alert message

13.4 *SMS Remote Control*

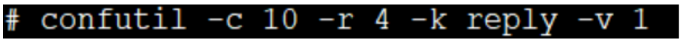
Please follow the following steps:

1. Set SMS management mode to “free”



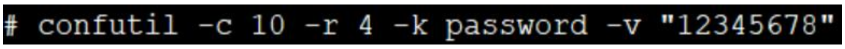
```
# confutil -c 10 -r 4 -k mode -v free
```

2. Enable SMS reply



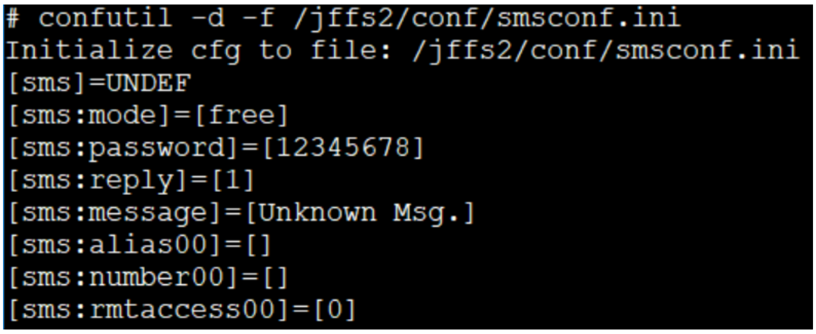
```
# confutil -c 10 -r 4 -k reply -v 1
```

3. Set the remote control password



```
# confutil -c 10 -r 4 -k password -v "12345678"
```

4. Check the configurations:



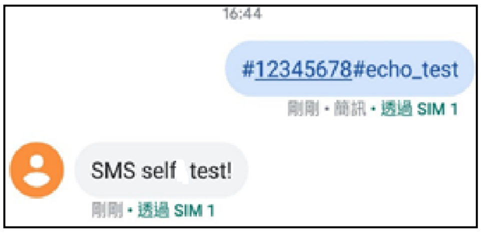
```
# confutil -d -f /jffs2/conf/smsconf.ini
Initialize cfg to file: /jffs2/conf/smsconf.ini
[sms]=UNDEF
[sms:mode]=[free]
[sms:password]=[12345678]
[sms:reply]=[1]
[sms:message]=[Unknown Msg.]
[sms:alias00]=[]
[sms:number00]=[]
[sms:rmtaccess00]=[0]
```

Figure 48: SMS remote control configuration

5. Send a message to the device (Suppose the phone number is “0901123456”)

   Note: SMS remote control message format is: ***“#<password>#<command>”***
   # sendsms 0901123456 “#12345678#echo_test”

6. Check remote control response.
   The number will receive a SMS messages: “SMS self test!”



#12345678#echo_test

SMS self test!

Figure 49: SMS Self Test

14 Firewall Support (Gateway Platform Only)

For ATOP's gateway platform, SDK provides the basic firewall rules with "iptables". In SDK, the firewall is activated when the NAT function is enabled. Users can reference the start-up script file to implement proprietary firewall mechanism.

- Script file to set the firewall

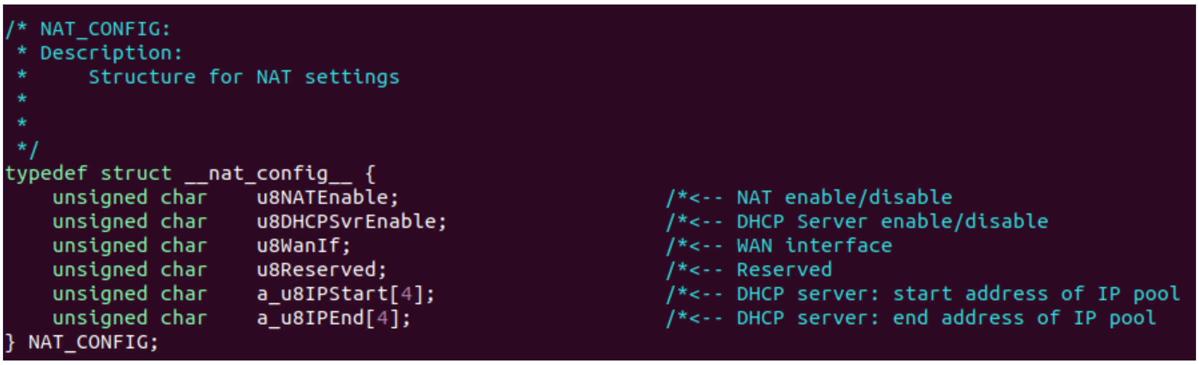
  <sdk>/software/application/system/firewall.sh

- Firewall script activation
  When the WAN interface is up, system will activate the firewall script file

  <sdk>/application/system/if-up.sh

14.1 *NAT*

The NAT settings are managed in below container:



```
/* NAT_CONFIG:
 * Description:
 *     Structure for NAT settings
 *
 *
 */
typedef struct __nat_config__ {
    unsigned char    u8NATEnable;                      /*<-- NAT enable/disable
    unsigned char    u8DHCPSvrEnable;                  /*<-- DHCP Server enable/disable
    unsigned char    u8WanIf;                          /*<-- WAN interface
    unsigned char    u8Reserved;                       /*<-- Reserved
    unsigned char    a_u8IPStart[4];                   /*<-- DHCP server: start address of IP pool
    unsigned char    a_u8IPEnd[4];                     /*<-- DHCP server: end address of IP pool
} NAT_CONFIG;
```

Figure 50: Firewall NAT

Field	Description
u8NATEnable	NAT enable/disable • 0 : Disable • 1: Enable
u8DHCPSvrEnable	When NAT is enabled, users can determine to enable/disable DHCP server function on local LAN DHCP server enable/disable • 0 : Disable • 1: Enable
a_u8WanIF	Index of WAN interface. The filed would be useful only when there are two LAN interfaces supported on the platform
a_u8IPStart	Start IP addresses that DHCP server to assign
a_u8IPEnd	End IP addresses that DHCP server to assign

Table 48: NAT Settings

14.2 *Firewall Scripts: Deny/Allow/Forward*

SDK provide the simple mechanism to allow users to activate firewall on device. Users can easily establish their own firewall on their gateway device by adding/creating their rules in these shell script files.

- /etc/iptables/iptables.deny
- /etc/iptables/iptables.allow
- /etc/iptables/port_forward

When firewall.sh runs and upper script files exist, the related script would be activated.

15 Examples

Add a new daemon in SDK:

SDK provides the example code of com_tcp_server in software folder.
The com_tcp_server ("<SDK>/software/application/utils/com_tcp_server") is an example used to exchange data between COM port and network. Users can reference the example code (tcp_server.c and Makefile) to get an idea how to create a daemon in the system.

16 Warranty

Limited Warranty Conditions

Products supplied by Atop Technologies Inc. are covered in this warranty for undesired performance or defects resulting from shipping, or any other event deemed to be the result of Atop Technologies Inc. mishandling. The warranty doesnot cover; however, equipment which has been damaged due to accident, misuse, abuse, such as:

- Use of incorrect power supply, connectors, or maintenance procedures
- Use of accessories not sanctioned by us
- Improper or insufficient ventilation
- Improper or unauthorized repair
- Replacement with unauthorized parts
- Failure to follow our operating Instructions
- Fire, flood, "Act of God", or any other contingencies beyond our control.

RMA and Shipping Reimbursement

- Customers must always obtain an authorized **"RMA" number** from us before shipping the goods to be repaired.
- When in normal use, a sold product shall be replaced with a new one within 3 months upon purchase. The shipping cost from the customer to us will be reimbursed.
- After 3 months and still within the warranty period, it is up to us whether to replace the unit with a new one; normally, as long as a product is under warranty, all parts and labour are free-of-charge to the customers.
- After the warranty period, the customer shall cover the cost for parts and labour.
- Three months after purchase, the shipping cost from the customer to us will not be reimbursed, but the shipping costs from us to the customer will be paid by us.

Limited Liability

Atop Technologies Inc. shall not be held responsible for any consequential losses from using our products.

Warranty

Atop Technologies Inc. provides a 5-year maximum warranty for Industrial Serial Device Server products.

Atop Technologies, Inc.

www.atoponline.com
www.atop.com.tw

TAIWAN HEADQUARTER:

2F, No. 146, Sec. 1, Tung-Hsing Rd,
30261 Chupei City, Hsinchu County
Taiwan, R.O.C.
Tel: +886-3-550-8137
Fax: +886-3-550-8131

ATOP CHINA BRANCH:

3F, 75th, No. 1066 Building,
Qingzhou North Road,
Shanghai, China
Tel: +86-21-64956231

ATOP INDIA OFFICE:

Abhishek Srivastava
Head of India Sales
Atop Communication Solution(P) Ltd.
No. 311, 6th Main Rd, Indiranagar,
Bangalore, 560038, India
Tel: +91-80-4920-6363
E-mail: Abhishek.S@atop.in

ATOP INDONESIA BRANCH:

Jopson Li
Branch Director
Wisma Lampung Jl.
No. 40, Tomang Raya
Jakarta, Barat, 11430, Indonesia
Tel: +62-857-10595775
E-mail: jopsonli@atop.com.tw

ATOP EMEA OFFICE:

Prashant Mishra
Business Development (EMEA)
Atop Communication Solution(P) Ltd.
No. 311, 6th Main Rd, Indiranagar,
Bangalore, 560038, India
Tel: +91-738-702-0003
E-mail: prashant.m@atop.in

ATOP AMERICAs OFFICE:

Venke Char
Sr. Vice President & Head of Business
11811 North Tatum Blvd, Suite 3031
Phoenix, AZ 85028,
United States
Tel: +1-602-953-7669
E-mail: venke@atop.in