Atop Technologies, Inc.

SDK Porting Guide

User Manual

V0.5

26 March 2021

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:	**0.5**
Revision History:	
Creation Date:	**April 2019**
Last Revision Date:	**26 March 2021**
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 porting your application or script to ATOP SDK. 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 https://atoponline.com/.

1.3 *Getting the Source Code*

The ATOP SDK source package could be downloaded from ATOP FTP site. Please contact your sales representative or local distributor to get your account information. If they are unable to assist you, please redirect your inquiries to www.atop.com.tw or https://atoponline.com/.

2 Introduction

ATOP has over 20 years of experience in designing and manufacturing high-tech equipment. SE59XX platform is one of the major products developed by ATOP. It has been developed to meet the growing need of robust platform for developing embedded applications to meet business needs. To facilitate faster custom application development for ATOP's clients, ATOP provides a standard SE59XX software development kit (SDK). SE59XX is equipped with various commonly used hardware capabilities. This document describes in detail the high-level software architecture of SE59XX platform, SDK to build application, and software interfaces.

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

3 Software Block Diagram

The software architecture of SE59XX SDK is designed to be a generic platform with a wide range of most commonly used features required to develop embedded applications. Figure 3.1 illustrates the software block diagram of SE590X SDK, which is categorized into 3 layers: Boot Loader, Kernel, and Libraries & Applications.

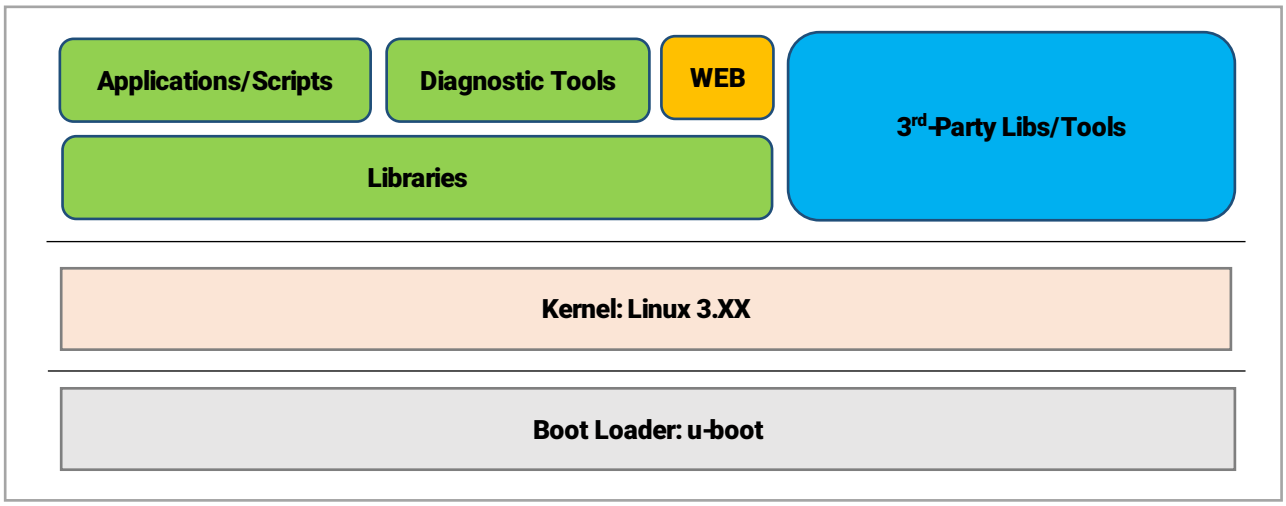

Figure 3.1 Software Block Diagram

The software layers in Figure 3.1 are placed on top of SE59XX hardware platform which is not shown in the figure. The boot loader (bootstrap loader) layer is the first software that is run or loaded on the hardware of SE59XX. The boot loader provides a way to load the operating system (OS) and applications into the memory of the hardware. SE59XX employs u-boot or universal boot loader which is an open-source boot loader for embedded device. Currently, SE59XX utilizes Linux operating sytem in which there is a Linux Kernel running at the second layer as depicted in the figure. Note that as of this writing the SE59XX is based on the Linux kernel version 3.XX as its embedded operating system. Table 3.1 summarizes each block in the software block diagram of Figure 3.1.

In the upper layer (layer three) of the software, ATOP SDK provides proprietary libraries for ATOP's users to access SE59XX system and peripheral components. Additionally, diagnostic tools are also available in the SDK to help test and verify peripheral components of SE59XX. Note that ATOP SDK package employs an open-source small memory footprint web-server called lighttpd. This web-server enables the users to easily manage the system setting via web browser-based user interface (Web-UI). The users can develop their own or customized applications or scripts to run on top of these ATOP SDK architecture. There are also 3-rd party libraries or tools that are compatible with Linux kernel used in the ATOP SDK.

Table 3.1 Description of Component in ATOP Softare Block Diagram

Folder	Description
Bootloader	ATOP SDK supports 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 access system or peripheral components easily.
Applications/Scripts	ATOP SDK provides some basic applications and scripts to bring up network and some basic network services.
Diagnostic tools	The "Diagnostic tools" are available for users to test and verify peripheral components.
WEB	ATOP SDK package uses the lighttpd as the WEB server. The simple WEB server helps users to manage system settings via WEB UI easily.
3rd Party Tools/Library	3rd-Party tools and libraries used in ATOP SDK

4 Source Architecture

Figure 4.1 illustrates the source architecture of ATOP SDK. The figure outlines the directories or folders along with their contents. Brief description of each folder is summarized in Table 4.1.

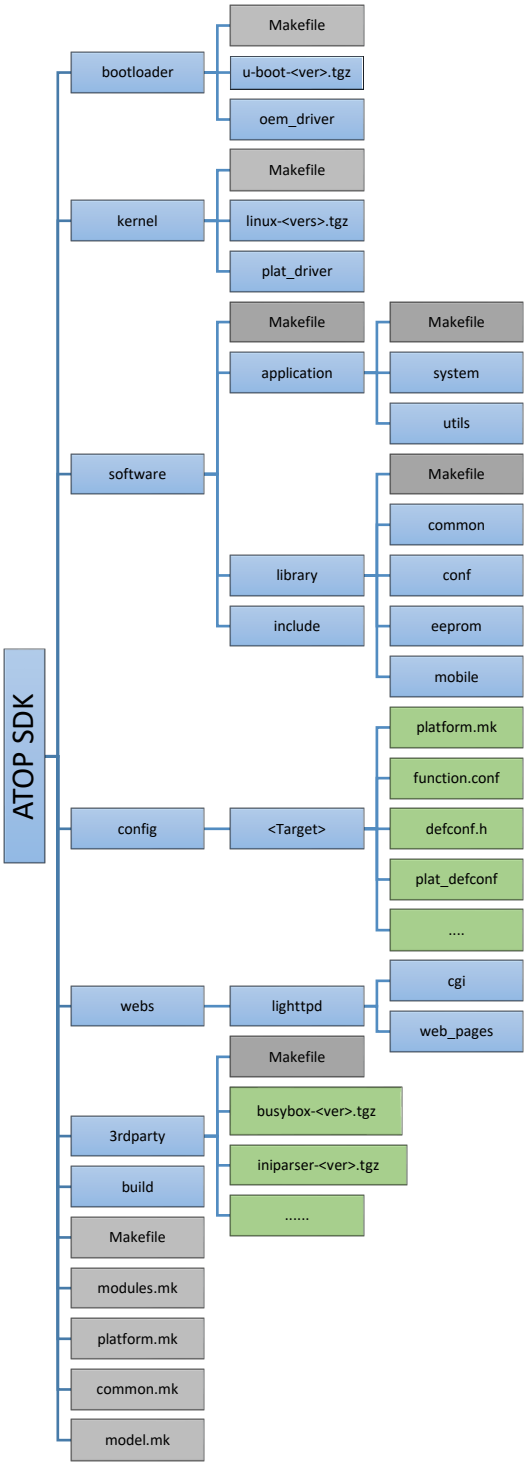

Figure 4.1 Source Architecture of ATOP SDK

Table 4.1 Source Architecture's Folders and Their Descriptions

Folders	Descriptions
3rdparty	All 3rd-Party tools and libraries are collected in this folder.
Bootloader	This folder contains the boot source and related object codes.
Build	After source codes are compiled successfully, firmware images are generated in this folder.
Config	This folder collects the platform/target configurations.
File System	This folder collects default scripts and content files of platform/target file system.
Kernel	This folder collects the Linux kernel source and related ATOP proprietary object codes.
Software	This folder collects ATOP proprietary applications, libraries, and diagnostic tools.
Webs	This folder collects the WEB CGI files, HTML webpages and Java Script files.

To develop user's applications or scripts, the users will require a host Personal Computer (PC) to install the above source code. For example, the host PC may be based on an x86 processor architecture. Then, the users can perform a cross-compiling of firmware for SE59XX on the host PC and then upgrade the firmware of SE59XX to install your applications or scripts. Note that the SE59XX is based on an embedded microcontroller architecture such as ARM cortex.

5 Build Enviornment Setup

To build a firmware for your custom applications or scripts, you will need a host PC with suitable operating system (OS) and build environment for ATOP SDK. Currently, ATOP SDK for SE59XX supports two hardware architectures from Texas Instrument (TI) and Nuvoton platforms. Both are based on ARM microcontroller architecture. Below is the list of supported operating systems for host PC that can install ATOP SDK:

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

Once you obtained the SDK source package from ATOP. You will also need cross compiler toolchain because it contains a set of programming tools used to develop your applications and scripts for corresponding hardware platform. You can install it into the recommended folders:

- **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"**

The following sections will describe the required steps with command lines to setup build environment for TI platform toolchain and Nuvoton platform toolchain on Ubuntu-16.04 (i386) and Ubuntu-18.04 (x64), respectively. Note that you will require the root priviledge of your operating system on host PC to perform those commands.

5.1 *Ubuntu 16.04 (i386)*

For Ubuntu 16.04 (i386), please follow the steps below to setup the build environment. Note that the command line behind the `$' should be entered in the shell prompt of your operating system.

1. **Copy and decompress the toolchain to host PC (*ti-am335x-linux-devkit-08.00.00.00.tar.bz2 or arm_linux-4.8_nuvoton.tgz*)**

   For 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

   For Nuvoton Platform:
   $ 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**
   $ sudo vi ~/.bashrc

   Add the line mentioned below at the end of the file (.bashrc) to set environment while system start-up. Note that you can use any editor such as *vi*, *vim*, or *nano*.

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

   For 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 iptables (required only in case of iptables 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)*

For Ubuntu 18.04 (x64), please follow the steps below to setup the build environment. Note that the command line behind the `$' should be entered in the shell prompt of your operating system.

1. **Copy and decompress the toolchain to host PC (*ti-am335x-linux-devkit-08.00.00.00.tar.bz2 or arm_linux_4.8_nuvoton.tgz*)**

   For TI Platform:

   $ 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

   For Nuvoton Platform:

   $ 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 file of ".bashrc"**

   $ sudo vi ~/.bashrc
   # Add below line

   ...

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

   For Nuvoton Platform:
   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 flex bison

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

5.3 *Source Code Extraction*

This section lists the commands required to extract the source code of ATOP SDK to your host PC. Starting by changing the current directory (using *cd* command) to the folder where ATOP SDK package was downloaded. Then, there are two steps to extract the SDK source.

1. **Decompress the zipped source package.**
   $ unzip ATSDK_<Model>_<Version>_<Date>.zip

   Enter the password that described with prefix Pwd: in release_ATSDK_<MODEL>_<VERSION>_<DATE>.txt

2. **Extract the source code**
   $ tar –zxf ATSDK_<Model>_<Version>.tgz

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 images - *$ make release_all***
2. **Build the system images (build with kernel) - *$ make release_img***
3. **You may also try to use this command to generate the system image without building the kernel source (Make sure that kernel was built successfully before using this command) - *$ make image***
4. **Compile the folders of software and webs, then generate the image - *$ make fwimg***
5. **Compile the software folder - *$ make swbuild***
6. **Compile the web folder - *$ make websvr***
7. **Compile the 3rd party folder - *$ make opensrc***

Here is an example how to generate the system images:

1. **Switch to ATOP SDK repository - $ *cd <your working spaces>***
2. **Type this command to compile sources, and generate the bootloader and system images –**
   $ 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)**

Figure 6.1 Selection of Build Target

4. **After the build is successful, the bootloader and system images would be generated in the folder - "<SDK>/build"**

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

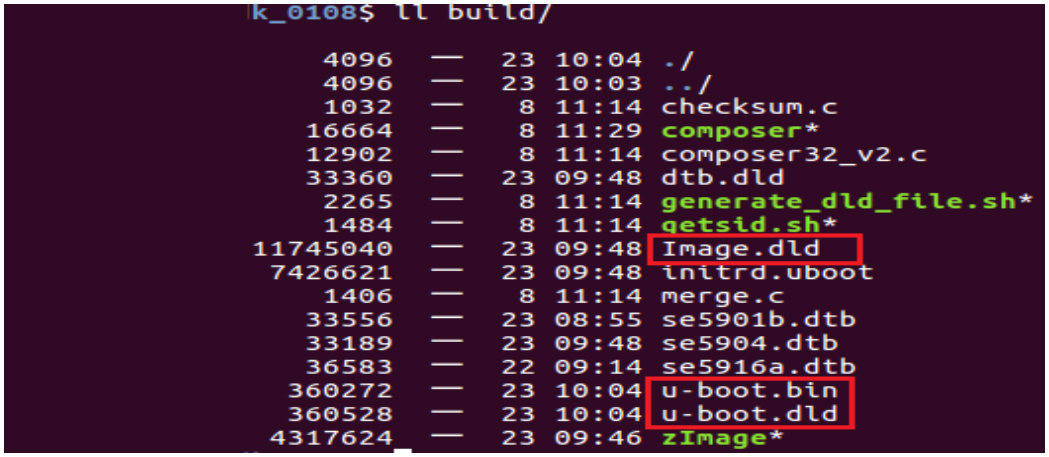

Figure 6.2 Generating system images

7 Upgrade System/Firmware Image to Hardware Platform

When you successfully built the new system/firmware image from the source code in the previous chapter, you can deploy your newly develop applications or scripts on the actual device such as SE59XX. There are three methods to upgrade the system/firmware images to hardware platforms. These methods are described in the following sections:

7.1 *Upgrade System Image or Bootloader from Bootloader (with TFTP protocol)*

The first method described in this section requires that you have access to the console port of the device via a debug line and the device is also connected to a host PC that runs TFTP server via an RJ-45 cable and Ethernet port. You will need a terminal application such as tera term or

HyperTerminal to access the debug console of the device. Additionally, you will need to install the TFTP on the host PC. Please follow the prerequisite steps as followings.

1. **Using a debug line (serial cable) to connect a COM (USB) port of the host PC to the debug console port (USB2) of the device (hardware platfrom SE59XX) and a RJ-45 line (Ethernet cable) between the LAN port of the device (SE59XX) to the LAN port of the host PC.**
2. **Launch a terminal application such as tera term on the host PC and setup the COM port parameters such as baud rate, data bits, parity bit, stop bits, and flow control to the values shown in Figure 7.1.**

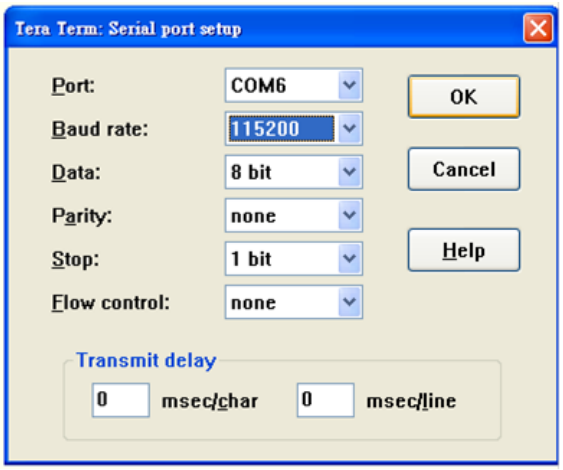

Figure 7.1 Serial Port Setup for Debug Console

3. **Run the TFTP server on the host PC using application program such as tftpd64.exe or tftpd32.exe and setup the Server Interfaces to the IP address of LAN port of the host PC.**

To upgrade the new firmware to your hardware platform, please follow these steps:

1. **Copy generated firmware of "Image.dld" (or <Target>.dld or *u-boot.dld*) to tftp server folder (tftpd32/64) on the host PC. Make sure that you run the TFTP server and set the Current Directory to the folder that the firmware is placed as shown in Figure 7.2.**

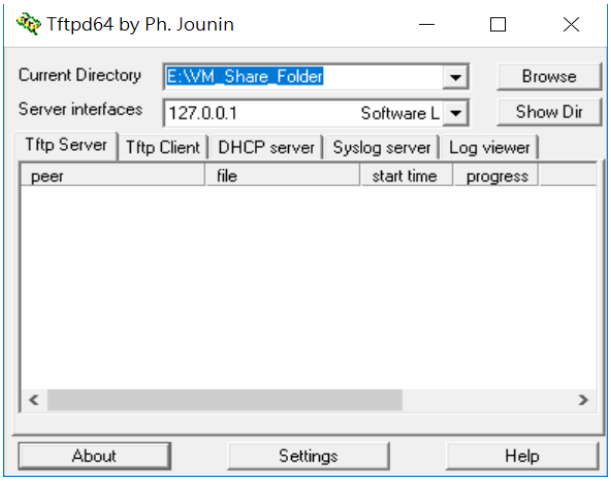

Figure 7.2 Copying generated firmware to tftp server folder

2. **On the debug console terminal, reset the target device and press the "ESC" button (via the console terminal such as tera term) to enter bootloader shell command menu as shown in Figure 7.3.**

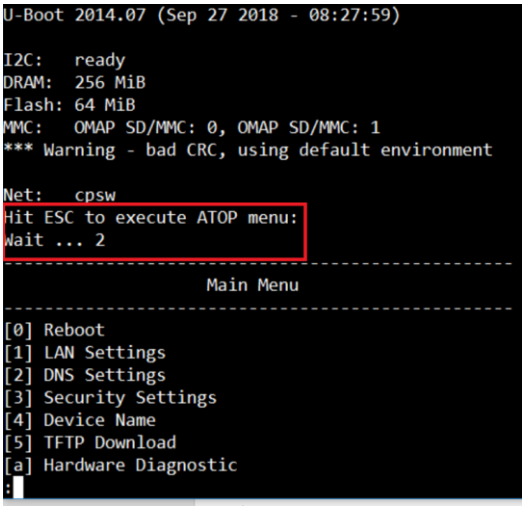

Figure 7.3 Resetting target device

3. **On the debug console terminal, type "5" to enter "TFTP Download" mode as shown in Figure 7.4.**

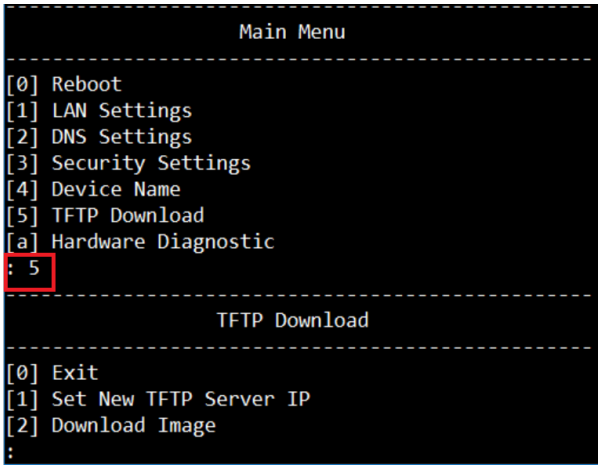

Figure 7.4 Entering TFTP Download Mode

4. **On the debug console terminal, type "1" to input correct TFTP server address (the same address as the LAN port of the host PC running TFTP server) as shown in Figure 7.5.**

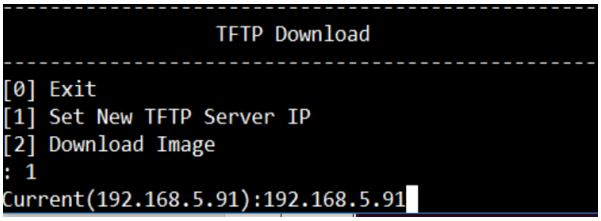

Figure 7.5 Input TFTP Server Address

5. **On the debug console terminal, type "2" and input the file name of "Image.dld" (or <Target>.dld) which is already placed on the host PC. Then type "Enter" to activate the firmware upgrade progress as shown in Figure 7.6.**

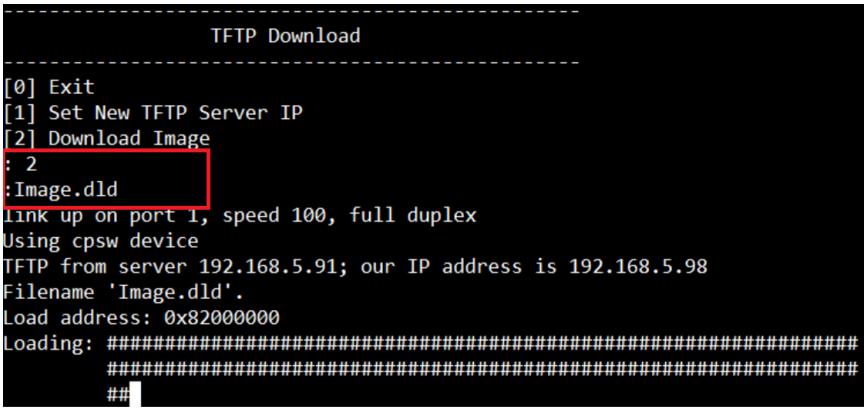

Figure 7.6 Input File Name of "Image.dld"

6. **After the firmware was upgraded successfully, reset the target device (SE59XX), and make sure that it can start-up properly. Note that on the debug console device, manually press "0" to Exit then press "0" to Reboot or reset the device as shown in Figure 7.7.**

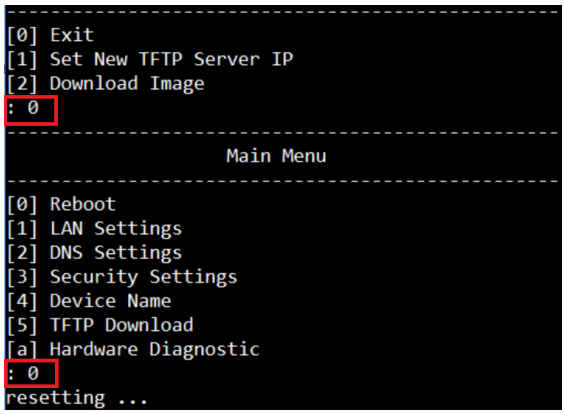

Figure 7.7 Resetting Target Device

7.2 *Upgrade System Image or Bootloader through Webpage*

The second method to upgrade the system image or boot loader is through the web browser that accesses the webpage of the device. Here are the steps that you need to follow:

1. **Login to the device's webpage using default username account/password: admin/default.**
2. **Select to System Setup menu and then select sub-menu Firmware Upgrade as shown in Figure 7.8.**

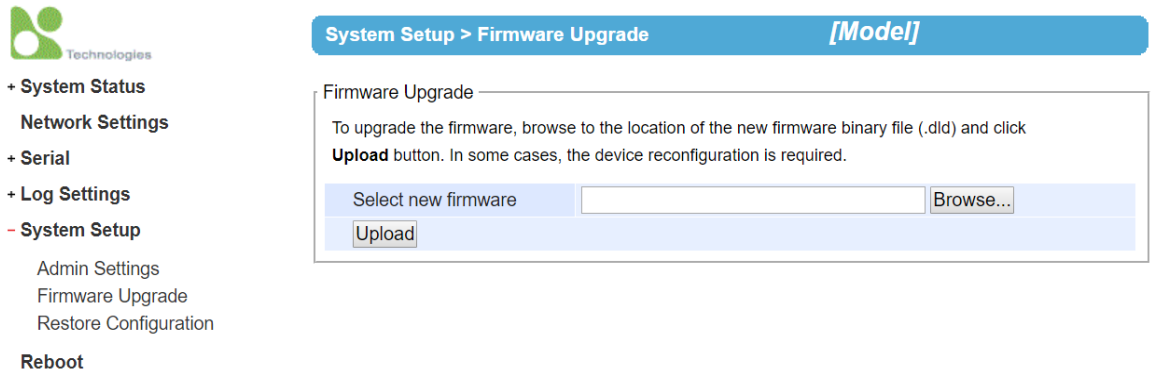

Figure 7.8 Using Firmware Upgrade Web Page for ATOP SDK

3. **On the web page, click "Browse..." button to select firmware (Image.dld) from your local host as shown in Figure 7.9.**

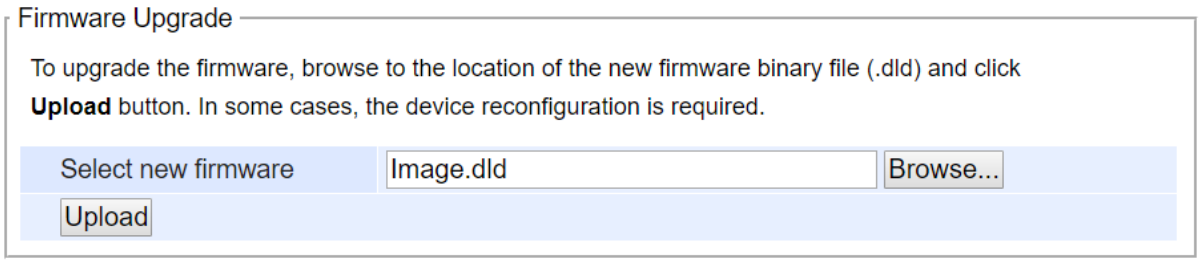

Figure 7.9 Select new firmware from local host folder

4. **On the web page, click "Upload" button to start upload new image to the device (SE95XX) and the progress of uploading the new image is shown in Figure 7.10.**

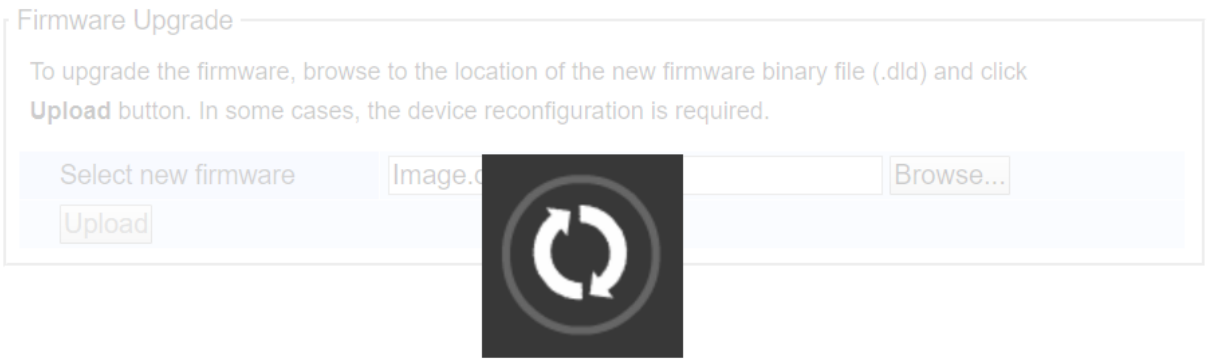

Figure 7.10 Progress of uploading new image to device

5. **On the web page, click "OK" to start the firmware upgrade process as shown in Figure 7.11.**

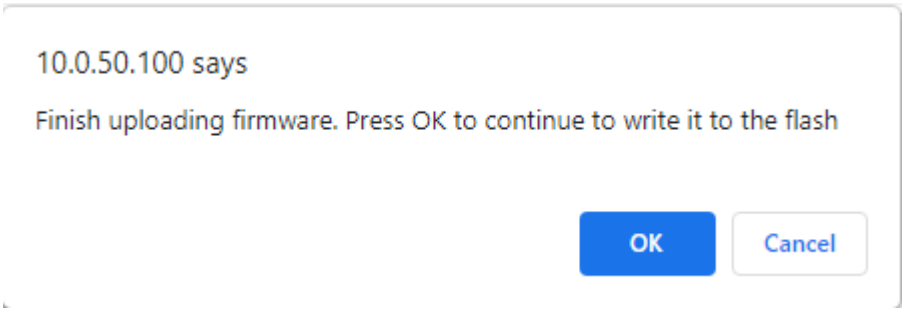

Figure 7.11 Starting of firmware upgrade process

6. **Click "OK" to finish the firmware upgrade process and reset device as shown in Figure 7.12.**

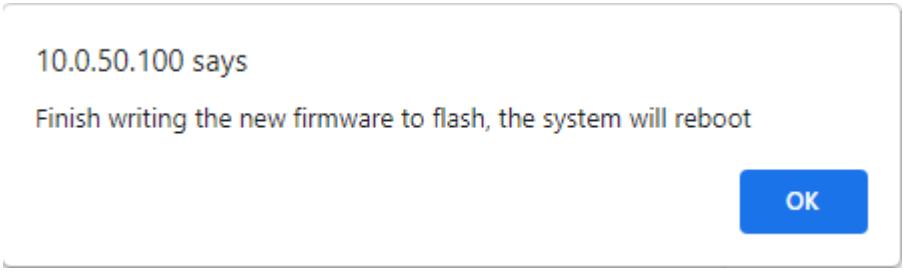

Figure 7.12 Finishing firmware upgrade and reboot

7. **Check if firmware has upgraded successfully after the device is rebooted by login to the device again via web browser.**

7.3 *Manually Upgrade System Image or Bootloader from Debug Console*

The third method for upgrading the firmware is also possible using only command line via the debug console. Once again, you will need access to the debug port of the hardware platform and also require to have a TFTP server installed and run on a host PC. Here are the steps to manually upgrade system image or bootloader from the debug console:

1. **Start the TFTP server on your host PC and copy generated firmware to tftp server folder using tftpd32/64 application as shown in Figure 7.13.**

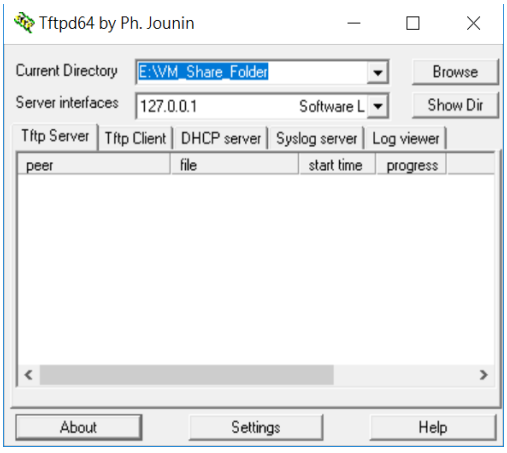

Figure 7.13 Copying generated firmware to host PC's tftp server folder

2. **Login to debug console window using terminal application such as tera term as shown in Figure 7.14. Enter the default account = 'root' and no password or null as password. Note**

that the serial configuration should set the baudrate = 115200 bps and other parameters as shown in Figure 7.1.

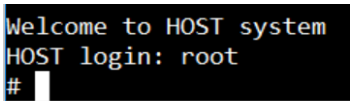

Figure 7.14 Login to debug console window

3. **Execute the following command on the prompt to activate the FW upgrade process as show in Figure 7.15.**
   # frmwr-upgrd tftp <ftp svr. Addr> <fw image>

   Note: *fw image* can be system image (xxx.dld), bootloader image (u-boot.dld), or Linux device tree image (dtb.dld)

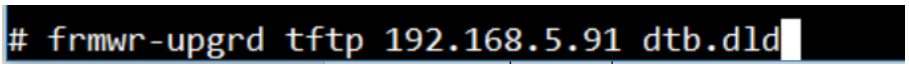

Figure 7.15 Activating firmware upgrade process

4. **Check if system resets automatically after the firmware was upgraded as indicated in Figure 7.16.**

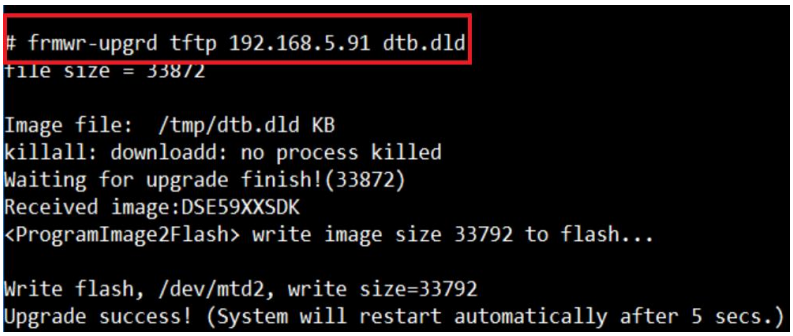

Figure 7.16 Checking auto-system restart

5. **Check if system starts up properly by logging in the system via debug console or through web browser.**

8 Platform APIs

This chapter introduces APIs that are available in the ATOP SDK package. With these APIs, users can easily access and control peripheral components on the hardware platform such as SE59XX. Note that the supported APIs may vary on different hardware platforms.

8.1 *Buzzer*

Table 8.1 API for buzzer

API Name	void BuzzerOnOff(int onoff)
Descriptions	Turn of/off the platform's buzzer
Header	buzzer.h
Input	onoff: 1: buzzer on 0: buzzer off
Output	N/A
Return	N/A
Example	#include "buzzer.h" ... // Buzzer on BuzzerOnOff(1); ...

8.2 *Run LED*

Table 8.2 API for Run LED

API Name	void SetRunLed(RUNLED_HANDLER *pHandler);
Descriptions	Handle the behaviors of "Run Led"
Header	runled.h
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_HANDER handler; handler.type=2;//blink handler.action=1;//on handler.delay_on-1000;// on interval, 1000ms handler.delay_off=500;//of interval; 500ms SetRunLed(&handler);

8.3 *Alarm LED (Support Platform: 5904D, 5908A, 5916A)*

Table 8.3 API for Alarm LED

API Name	void SetAlarmLed(unsigned char onoff)
Descriptions	Turn of/off the alarm led
Header	alarmled.h
Input	onoff: 0: off 1: on
Output	NA
Return	NA
Example	#include "alarmled.h" ... // Turn on alarm LED SetAlarmLed(1); ...

8.4 *DI, DO*

Table 8.4 API for DI (Digital Input)

API Name	int SysGetDI(int index)
Descriptions	Get DI pin status
Header	di.h
Input	index: Index of pin
Output	NA
Return	0 for low level signal, 1 for high level signal
Example	#include "di.h" ... int ret_status = 0; //Get DI0 pin status ret_status = SysGetDI(0); printf("DI0 pin status:%d\n", ret_status); ...

Table 8.5 API for DO (Digital Output)

API Name	int SysSetDO(int index, int value)
Descriptions	Set DO state
Header	do.h
Input	index: Index of pin value: 0: output low level signal 1: output high level signal
Output	NA
Return	-1 for failure, otherwise for success
Example	#include "do.h" ... // Set DO0 on SysSetDO(0, 1); // Set DO1 off SysSetDO(1, 0); ...

8.5 *Hardware Watchdog (TI plarform only)*

Table 8.6 API for Enabling Hardware Watchdog (TI platform only)

API Name	void hwd_enable(void)
Descriptions	Enable HW watchdog
Header	sys_hwd.h
Input	NA
Output	NA
Return	NA
Example	#include "sys_hwd.h" ... // enable HW watchdog hwd_enable(); ...

Table 8.7 API for Disabling Hardware Watchdog (TI platform only)

API Name	void hwd_disable(void)
Descriptions	Disable HW watchdog
Header	sys_hwd.h
Input	NA
Output	NA
Return	NA
Example	#include "sys_hwd.h" ... // enable HW watchdog hwd_disable(); ...

Table 8.8 API for Clearing Hardware Watchdog (TI platform only)

API Name	void hwd_clear(void)
Descriptions	Clear HW watchdog timer count
Header	sys_hwd.h
Input	NA
Output	NA
Return	NA
Example	#include "sys_hwd.h" ... // clear HW watchdog hwd_clear() ; ...

Table 8.9 APT for Setting Timeout Interval for Hardware Watchdog (TI platform only)

API Name	void hwd_timeout(unsigned int timeout)
Descriptions	Set HW watchdog timer's timeout interval
Header	sys_hwd.h
Input	interval of timeout (sec)
Output	NA
Return	NA

Example	**#include "sys_hwd.h"** **...** **// Set HW watchdog timer timeout interval to 10 secs** **hwd_timeout(10);** **...**

8.6 *COM Management*

Table 8.10 API for COM Port (UART) Management

API Name	**int SysUARTNumber(void)**
Descriptions	**Query supported number of COM ports**
Header	**sys_uart.h**
Input	**NA**
Output	**NA**
Return	**Number of supported COM ports**
Example	**#include "sys_uart.h"** **...** **// Get COM port number** **int num = SysUARTNumber();** **printf("COM port number:%d\n", num);** **...**

Table 8.11 API for COM Port Initialization

API Name	**void comport_init(void)**
Descriptions	**Initialize COM ports depending on COM configurations**
Header	**comport.h**
Input	**NA**
Output	**NA**
Return	**NA**
Example	**#include "comport.h"** **...** **// Depending on COM settings to Initialize physical port settings** **comport_init();** **...**

Table 8.12 API for Setting COM Port Configuration

API Name	int comport_set(unsigned char index, void *pConf)
Descriptions	Set COM port configurations
Header	comport.h
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_CONFIG 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.u8Flowctl = 0; // none 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); …

8.7 *Relay (Support Platform: 5901B, 5904D, 5908/16, 5908A/16A)*

Table 8.13 API for Setting Relay State

API Name	void SetRelayOnOff (unsigned char onoff)
Descriptions	Switch relay state: on or off
Header	relay.h
Input	onoff: 0: off 1: on
Output	NA
Return	NA
Example	#include "relay.h" ... // Set Relay state to on SetRelayOnOff(1); ...

8.8 *Log*

Table 8.14 API for Sending Message to System Log File

API Name	void SendSysLog (severity_e serv, char *prefix, char *msg)
Descriptions	Send messages to system Log file
Header	loginfo.h
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 log test!"); ...

8.9 *Alert Message Management*

Depending on hardware platforms, ATOP SDK supports a number of alert APIs to help users sending alert information while receiving specified events.

Table 8.15 API for Alert Message Management

API Name	void SendAlertMsg(unsigned int msg_event, unsigned char isSysLog, MSG_SYSLOG_HANDLER *pLog);
Descriptions	Send messages to Log file
Header	alert_msg.h
Input	msg_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_IPSEC_CONNECTED, MSGALERT_IPSEC_DISCONNECTED, 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 pLog: 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 (EVT_INFO, LOG_SYS, 0,NULL); ...

Table 8.16 API for SMS Message Management

API Name	void sendSMSMsg(unsigned int msg_event, char *pMsg); (Support Platform: 5901B)
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.
Header	alert_msg.h
Input	msg_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_IPSEC_CONNECTED, MSGALERT_IPSEC_DISCONNECTED, 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: Meseage contents to send out through SMS
Output	NA
Return	NA
Example	#include "alert_msg.h" ... sendSMSMsg (MSGALERT_AUTH_FAIL,"WEB authentication failed"); ...

8.10 *Firmware Upgrade*

Table 8.17 API for Firmware Upgrade to Flash

API Name	int fw_upgrade(char *fw_addr, int length)
Descriptions	Programming firmware to flash
Header	sys_upgapi.h
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" ... char *buff = NULL; int file_size = 0; ... /* query filesize */ file_size = fwupg_get_filesize(); /* allocate buffer to receive buffer */ buff = fwupg_alloc_shmbuf(file_size); /* 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 8.18 API for Allocating Share Memory Buffer for Firmware Image

API Name	char fwupg_alloc_shmbuf(unsigned int length)
Descriptions	Allocate shared memory buffer for firmware image
Header	sys_upgapi.h
Input	Buffer length(or image length) of the shared memory
Output	NA
Return	Pointer of shared memory buffer address
Example	See <sdk>/software/application/utils/diag_tool/frmwr-upgrd.c

Table 8.19 API for Getting Shared Memory Buffer

API Name	Char *fwupg_get_shmbuf(unsigned int length)
Descriptions	Get shared memory buffer
Header	sys_upgapi.h
Input	Length of firmware image
Output	NA
Return	Pointer of shared memory buffer address
Example	See <sdk>/webs/lighttpd/cgi/firmwareUpgrade.c

Table 8.20 API for Unlinking Shared Memory Buffer of Firmware Image

API Name	Char *fwupg_unlink_shmbuf()
Descriptions	Unlink shared memory buffer of firmware image
Header	sys_upgapi.h
Input	NA
Output	NA

Return	**NA**
Example	**See <sdk>/webs/lighttpd/cgi/firmwareUpgrade.c**

8.11 *System Reboot*

Table 8.21 API for Rebooting System with SIGTERM

API Name	**void SysRebootSystem(void)**
Descriptions	**Reboot system with the signal of "SIGTERM"**
Header	**sys_reboot.h**
Input	**NA**
Output	**NA**
Return	**NA**
Example	**#include "sys_reboot.h"** **...** **SysRebootSystem();** **...**

Table 8.22 API for Rebooting System without SIGTERM

API Name	**void RebootSystem2(void)**
Descriptions	**Reboot system without the signal of "SIGTERM"**
Header	**sys_reboot.h**
Input	**NA**
Output	**NA**
Return	**NA**
Example	**#include "sys_reboot.h"** **...** **RebootSystem2();** **...**

8.12 *System Management*

Table 8.23 API for Getting System Information (Version of Firmware)

API Name	int SysBootloaderVersion(SysVersion_t *ver) int SysKernelVersion(SysVersion_t *ver) void SysAPVersion(SysVersion_t *ver) int SysCPLDVersion(SysVersion_t * ver)
Descriptions	Get boot loader version Get kernel version Get AP version Get CPLD version
Header	ver_info.h
Input	Structure pointer of SysVersion_t
Output	Bootloader version information Kernel version information AP version information CPLD version information
Return	0: Success; Others: Failed
Example	#include “ver_info.h” SysVersion_t ver; ... SysBootloaderVersion(&ver); printf(“blVer: %u.%02u\n”, ver.VerMajor, ver.VerMinor); SysKernelVersion(&ver); printf(“kernelVer: %u.%02u\n”, ver.VerMajor, ver.VerMinor); SysAPVersion(&ver); printf(“apVer: %u.%02u\n”, ver.VerMajor, ver.VerMinor); SysCPLDVersion(&ver); printf(“cpldVer: %u.%02u\n”, ver.VerMajor, ver.VerMinor); ...

Table 8.24 API for Getting Firmware Version (Only Kernel and AP)

API Name	char *SysStrKernelVersion(void *info) char *SysStrAPVersion(void *info)
Descriptions	Get kernel version Get AP version
Header	ver_info.h
Input	NA
Output	NA
Return	Kernel version information AP version information
Example	#include “ver_info.h” ... printf(“kernel Ver: %s\n”, SysStrKernelVersion(NULL)); printf(“apVer: %s\n”, SysStrAPVersion(NULL)); ...

Table 8.25 API for Executing System Command

API Name	int ExecuteSysCommand(char *cmd, int limit)
Descriptions	Execute system command (popen, pipe stream)
Header	sys_cmd.h
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); ...

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

Table 8.26 API for Establishing Cellular Connection

API Name	void Dial_connect(void)
Descriptions	Establish the 3G/4G connection
Header	lib_dial.h
Input	NA
Output	NA
Return	NA
Example	#include "lib_dial.h" //setting INI *NETWORK_3G_CONFIG Conf;* *SysConf_Shm_SetKey(SYSCONF_FEATURE_NETWORK_3G ,&Conf, NET3G_KEY_APNCODE,"Public");* *SysConf_Shm_SetKey(SYSCONF_FEATURE_NETWORK_3G ,&Conf, NET3G_KEY_PINCODE,"0000");* *SysConf_Update_Shmcfg (SYSCONF_FEATURE_NETWORK_3G, &Conf);* Dial_connect(); ...

Table 8.27 API for Terminating Cellular Connection

API Name	void Dial_disconnect(void)
Descriptions	Disconnect the 3G/4G connection
Header	lib_dial.h
Input	NA
Output	NA
Return	NA
Example	#include "lib_dial.h" ... Dial_disconnect(); ...

Table 8.28 API for Getting Status Information of Cellular Connection

API Name	int Get_dial_info(DIAL_INFO *pInfo)
Descriptions	Get dialing information
Header	lib_dial.h
Input	pInfo: Pointer of DIAL_INFO
Output	Dialing information
Return	Type:DIAL_STATE_E Success: DIAL_STATUS_CONNCETING/DIAL_STATUS_CONNCETED Error: DIAL_STATUS_DISCONNCET
Example	#include "lib_dial.h" DIAL_INFO dial_info; ... Get_dial_info(&dial_info); printf("connect state: %s\n", dial_info.connState); printf("dial state: %s\n", dial_info.dialState); printf("pin state: %s\n", dial_info.pinState); printf("ip: %s\n", dial_info.ip); ...

Table 8.29 API for Getting GPS Information (for GPS supported Model only)

API Name	int Get_gps_info(GPS_INFO *pInfo)
Descriptions	Get GPS information
Header	lib_dial.h
Input	pInfo: Pointer of GPS_INFO
Output	GPS information
Return	Type: cmd_return_E Success: CMD_GET_SUCCESS Error: CMD_GET_ERROR/CMD_SEND_FAIL/CMD_NOT_SEARCH/ CMD_LEN_SHORT
Example	#include "lib_dial.h" GPS_INFO gps_info; ... Get_gps_info(&gps_info); printf("latitude: %s\n", gps_info.latitude); printf("longitude: %s\n", gps_info.longitude); ...

Table 8.30 API for Checking Supporting of 4G on Hardware Platform

API Name	int Sys4GSupport(void)
Descriptions	Check if 4G is supported on current platform
Header	cellular_api.h
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 8.31 API for Detecting 4G Module on Hardware Platform

API Name	int Sys4GModule(void)
Descriptions	Check if 4G module is detected on current platform
Header	cellular_api.h
Input	NA
Output	NA
Return	1: 4G module is supported on this platform 0: 4G module is not supported on this device
Example	#include “cellular_api.h” … If(Sys4GSupport()) { printf(“4G module is supported on this platform”); } …

Table 8.32 API for Getting the 4G Interface Name

API Name	int Sys4GInterface(char *pIfname)
Descriptions	Get used 4G interface name
Header	cellular_api.h
Input	pIfname: 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 “pIfname” 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); } ...

9 INI Configs Read/Write (Settings Management)

Most of the device's configurations in SDK are stored in INI files. The system allows a number of APIs to help users easily access these INI files. With these APIs, the users can easily access particular settings by using specified feature IDs. Table 9.1 lists feature IDs currently supported in the SDK package:

Table 9.1 Feature IDs Supported in SDK Package

Feature	ID	Key Heahdr
SYSCONF_FEATURE_ALL	0x00	NA
SYSCONF_FEATURE_BOARD	0x01	board_conf.h
SYSCONF_FEATURE_COM	0x02	com_conf.h
SYSCONF_FEATURE_SYSTEM	0x03	system_conf.h
SYSCONF_FEATURE_SYSLOG	0x04	syslog_conf.h
SYSCONF_FEATURE_NET	0x05	net_conf.h
SYSCONF_FEATURE_NETDNS	0x06	net_conf.h
SYSCONF_FEATURE_PORTFORWARD	0x07	port_forward.h
SYSCONF_FEATURE_NETWORK_3G	0x08	network_3g.h
SYSCONF_FEATURE_NAT	0x09	nat_conf.h
SYSCONF_FEATURE_SMS	0x0A	sms_conf.h
SYSCONF_FEATURE_SNMP	0x0B	Reserved
SYSCONF_FEATURE_VIP	0x0C	Reserved
SYSCONF_FEATURE_OVPN	0x0D	Reserved
SYSCONF_FEATURE_PPTP	0x0E	Reserved
SYSCONF_FEATURE_IPSEC	0x0F	Reserved
SYSCONF_FEATURE_RSTP	0x10	Reserved
SYSCONF_FEATURE_URLINK	0x11	Reserved
SYSCONF_FEATURE_SMTP	0x12	Reserved
SYSCONF_FEATURE_PING_REBOOT	0x13	Reserved
SYSCONF_FEATURE_FIREHOL	0x14	Reserved
SYSCONF_FEATURE_DDNS	0x15	Reserved
SYSCONF_FEATURE_PAMAUTH	0x16	Reserved
SYSCONF_FEATURE_NETOPT	0x17	Reserved
SYSCONF_FEATURE_USER	0x18	Reserved

Feature	ID	Key Heahdr
SYSCONF_FEATURE_OEM4	0xFA	Reserved
SYSCONF_FEATURE_OEM3	0xFB	Reserved
SYSCONF_FEATURE_OEM2	0xFC	Reserved
SYSCONF_FEATURE_OEM1	0xFD	Reserved
SYSCONF_FEATURE_OEM	0xFE	oem_conf.h

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

9.1 *Read Configurations from Shared Memory*

Table 9.2 API: Read Configurations from Shared Memory

API Name	int SysConf_Get_Shmcfg(unsigned char u8Id, void *pConf)
Descriptions	Read configurations from shared memory to pConf based on feature ID
Header	shmapi.h
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 9.3 API: Get Specified Key of System Configuration based on Feature ID from Shared Memory

API Name	int SysConf_Shm_GetKey(unsigned char u8Id, void *pConf, char *key, char *value)
Descriptions	Get value of specified key based on feature ID and configuration pointer (pConf)
Header	shmapi.h
Input	u8Id: feature ID pConf: pointer of feature configurations key: key string in INI file
Output	value: 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 9.4 API: Read Value of Specified Key based on Feature ID and Section Index from Shared Memory

API Name	int SysConf_Get_ShmKey(unsigned char u8Id, unsigned char section, char *key, char *value)
Descriptions	Read value of specified key from shared memory based on feature ID and section index
Header	shmapi.h
Input	u8Id: feature ID section: section index in INI file key: key string in INI file
Output	value: 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);* ...

9.2 *Set Configuration to Shared Memory*

Table 9.5 API: Set Value of Specified Key Based on Feature ID

API Name	int SysConf_Shm_SetKey(unsigned char u8Id, void *pConf, char *key, char *value)
Descriptions	Set value of specified key to pConf based on feature ID
Header	shmapi.h
Input	u8Id: feature ID pConf: pointer of feature configurations key: key string in INI file value: 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 9.6 API: Write Configurations based on Feature ID

API Name	int SysConf_Set_Shmcfg(unsigned char u8Id, void *pConf)
Descriptions	Write configurations to shared memory based on feature ID
Header	shmapi.h
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]);*

9.3 *Update Configurations to INI Files*

Table 9.7 API: Update Configurations Based on Feature ID to Shared Memory and INI file

API Name	int SysConf_Update_Shmcfg(unsigned char u8Id, void *pConf)
Descriptions	Update feature configurations to shared memory and INI file based on feature ID
Header	shmapi.h
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 simultaneously *SysConf_Update_Shmcfg(SYSCONF_FEATURE_NET, &conf[0]);*

Table 9.8 API: Update Key Value based on Feature ID to Shared Memory and INI File

API Name	int SysConf_Update_ShmKey(unsigned char u8Id, unsigned char section, char *key, char *value)
Descriptions	Update key value to shared memory and INI file based on feature ID
Header	shmapi.h
Input	u8Id: feature ID pConf: pointer of buffer key: key string in INI file value: key value
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”);*

9.4 *Add New Configurations*

The SDK package already provides an example for users to easily implement the INI feature settings. Here are the steps required to follow:

1. **Example of feature configurations can be found in the following files and their corresponding directories:**

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

2. **You can define any new feature IDs in "<sdk>/software/include/sysconf.h" as shown in Figure 9.1 below the comment "/* add your own feature type before "OEM" */".**

   # 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 9.1 Type Definition of Feature IDs

3. **Then you can define the feature section name and supported keys in "<sdk>/software/include/oem_conf.h" as shown in Figure 9.2.**

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

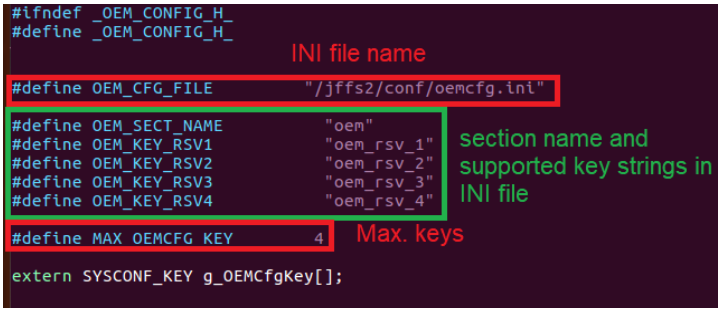

Figure 9.2 Defining of Feature Section Name

4. **Next, you can define a structure to handle settings in "<sdk>/software/include/oem_conf.h" as shown in Figure 9.3.**

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

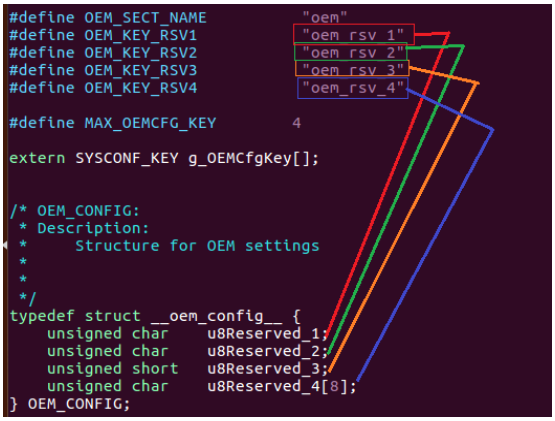

Figure 9.3 Defining stucture to handle SDK settings

5. **Next, you can define function names that are used to init/read/write feature settings the *oem_conf.h* file as shown in Figure 9.4.**

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

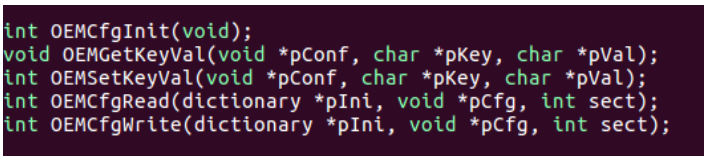

Figure 9.4 Defining function names for feature settings

6. **Next, add the ID of "SYSCONF_FEATURE_OEM" in gSYSConfHandler[] in the file *"<sdk>/software/library/conf/conf_handler.c"* as shown in Figure 9.5 and Figure 9.6.**

```
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 9.5 Adding sysconfig ID in conf_handler.c by locating the g_SYSConfHandler[]

```
    {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, 0, 0}
};
```

Figure 9.6 Adding sysconfig ID

7. **Then, define the key mapping table in "<sdk>/software/library/conf/oemconf.c" file as shown in Figure 9.7.**
   # 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 9.7 Defining Key Mapping Table

8. **Next, implement init function in "<sdk>/software/library/conf/oemconf.c" file as shown in Figure 9.8.**
   # 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 9.8 Implementing init function

9. **Then, implement read function in "<sdk>/software/library/conf/oemconf.c" file as shown in Figure 9.9.**

   # 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 9.9 Implementing Read Function

10. **Next, implement write function in "<sdk>/software/library/conf/oemconf.c" file as shown in Figure 9.10.**

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 9.10 Implementing Write Function

11. **Then, implement key get function in "<sdk>/software/library/conf/oemconf.c" file as shown in Figure 9.11.**

    # 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 9.11 Implementing Key Get Function

12. **Next, implement key set function in "<sdk>/software/library/conf/oemconf.c" file as shown in Figure 9.12.**

    # 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 9.12 Implementing Key Set Function

13. **Then, edit default configurations in "<sdk>/config/<target>/defconf.h" file as shown in Figure 9.13.**

```
#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 9.13 Editing Default Configurations

14. **Next, compile the software, and make sure that there is no error occurred after compiling using the following command.**

    # make swbuild

15. **Build the system and burn the new firmware to your target device. Please refer the Chapter 6: Source Code Compilation and Chapter 7: Upgrade System/Firmware Image to Hardware platform for more details.**

16. **Open debug console and connect to the target device. Run the command below to check whether the feature settings are working as shown in Figure 9.14:**

    # 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 9.14 Running Command in Open Debug Console

17. **Run the list of commands in Figure 9.15 to check each key value 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 9.15 Running Commands to Check Key Values

18. **Running another list of commands in Figure 9.16 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 9.16 Running Another Set of Commandsin Open Debug Console

10 Software

This chapter provides overview of the software directories or folders under the ATOP SDK. These software folders in ATOP SDK collect common libraries and applications. Figure 10.1 shows an example of directory list on the console.

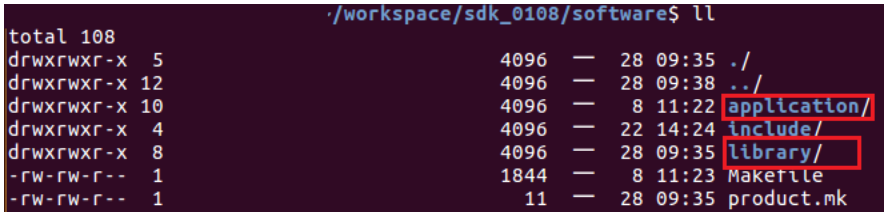

Figure 10.1 Software Folders in SDK

10.1 *Application*

Table 10.1 briefly summarizes the descriptions of software folder under the *"application"* directory.

Table 10.1 Descriptions of Application Folders

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

10.2 *Library*

Table 10.2 briefly summarizes the descriptions of software folder under the *"library"* directory.

Table 10.2 Descriptios of Library Folder

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.

11 Web

The ATOP SDK package supports a lightweight WEB server called "lighttpd". This chapter provides information related to the web server on the platform. Note that the default WEB URL is either http://10.0.50.100 or http://192.168.1.100. The CGI (Common Gateway Interface) files and webpages are placed in these directories:

- **CGI Files: <SDK>/webs/lighttpd/cgi/**
- **Webpages: <SDK>/webs/lighttpd/web_pages/**

11.1 *Web Account/Password*

By default, the WEB login account and password are set as followings:

- **User Name: admin**
- **Password: default**

11.2 *Change Web Logo*

The users can replace the default image file of the web logo with your own logo at this file location:
<sdk>/webs/lighttpd/web_pages/images/logo.gif,
with the following image size:

- **Width: 200px**
- **Height: 48px.**

11.3 *Add a New Webpage in Selection Menu*

The user can modify the menu items or new webpage by editing the following reference 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 11.1 Adding New Webpage in Selection Menu

12 System

In ATOP 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 12.1 System Target Configurations

12.1 *System Start-up Script Files*

- **On platform device, system start-up scripts are placed under "<sdk>/filesystem/etc/init.d/" folder.**
- **When system initializes, all features' settings are starting in *S01logging* file.**

```
#!/bin/sh

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

Figure 12.2 System Start-up Script File

- **ATOP's main initial flow is implemented in *S21SysInit* file.**

12.2 *Account and Password of Debug Console*

The username and password for accessing debug console has the following default values:

- **User Name: root**
- **Password: NULL**

12.3 *Change System Version Information*

You can update or change system version information of you firmware, which includes boot loader version, signature, software app version and kernel version, by modifying the following file: *<sdk>/config/<target>/platform.conf* as shown in Figure 12.3.

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

Figure 12.3 Changing of System Version Information

Then, run the following commands to rebuild the library, and update version information:
make swbuild; make image;
make swbuild; make fwimg;

12.4 *Platform Default Configurations*

The initial default system settings of the customizable firmware project are configured in "<sdk>/config/<target>/default.dat". Users can easily change each project's default settings in this file as shown in Figure 12.4.

```
[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 12.4 Platform Default Configurations

12.5 *Kernel Configurations*

The project's kernel configuration file is located in "<sdk>/config/<target>/plat_defconfig" file. To change it, you may execute the commands in Figure 12.5 to enable or 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 12.5 Kernel Configurations

12.6 *Flash Partition Layout*

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

12.7 *Change COM Number*

The physical COM port(s) support(s) varies with hardware platforms. If the physical COM port number does not match with your hardware platform, you can modify it using the following file:

<sdk>/software/include/sys_uart.h

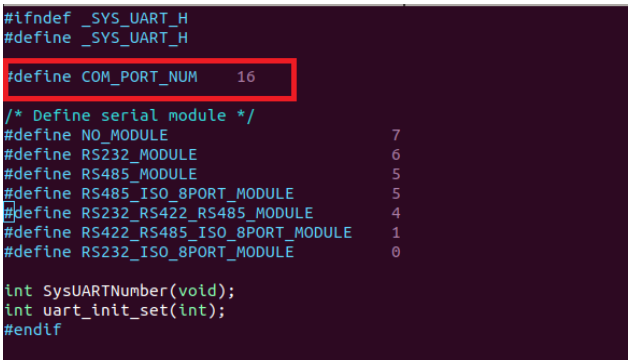

Figure 12.6 Changing of COM Port Number

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

ATOP SDK provides a simple mechanism for users to easily manage SMS with sms tools. This chapter elaborates on how to manage your SMS configuration and operations.

Note: Before using it, please make sure that the SIM card is already equipped or inserted in your hardware device.

13.1 *SMS Settings*

In ATOP SDK, SMS settings are managed by SMS_CONFIG data structure inside the "<sdk>/software/include/sms_conf.h" file as shown in Figure 13.1. Table 13.1 provides description of SMS setting fields.

```
/* 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 13.1 SMS Settings

Table 13.1 Description of 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 an 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	Bitmap of alert event for each phone number
a_u8AltMsgDelay	Alert messages delay interval of each alert event.

To access the SMS settings, please refer to Chapter 9:INI Configurations Read/Write (Settings Management).

13.2 *SMS Remote Control*

This section describes how to set up SMS remote control operation. Using the following format for Control Message, which will be sent via SMS. An example of control message is given below.

- **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

To write a script containing response or event handler for the SMS control message, you can modify the following file.

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

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

A file that lists supported SMS remote control messages is stored in the following location.

- **SMS Remote Control Command List:**

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

13.3 *SMS Alert Messages*

This section provides the steps on how to setup SMS alert messages. This will allow your hardware platform to send SMS alert messages to any mobile phone. Note that you can check Section 8.9: Alert Message Management in Chapter 8 to get the idea on how to acces SMS settings through ATOP SDK APIs. Here are the procedures to enable the SMS 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.**

confutil -c 10 -r 4 -k number00 -v 0900123456

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

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

5. **Check the configurations using the command in Figure 13.2:**

```
# 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 13.2 SMC Configuration

6. **Change IP address from your WEB and check if you can receive the alert message as shown in Figure 13.3:**

Figure 13.3 SMS Alert Message

13.4 *Testing of SMS Remote Control*

To verify your SMS Remote Control operation, please follow steps for testing SMS remote control:

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 remote control password**

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

4. **Check the configurations as shown in Figure 13.4:**

```
# 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 13.4 SMS Remote Control Configuration

5. **Send a message to the device (Suppose that 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 phone will receive a SMS messages: "SMS self test!" as shown in Figure 13.5.

Figure 13.5 SMS Self Test

14 Firewall Support (Gateway Platform Only)

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

- **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>/software/application/system/if-up.sh

14.1 *NAT*

The NAT (Network Address Translation) settings are managed in the data structure called NAT_CONFIG inside <sdk>/software/include/nat_conf.h file as shown Figure 14.1. Table 14.1 summarizes description of the fields inside this data structure.

```
/* 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 14.1 Firewall NAT

Table 14.1 Description of Fields in NAT Setting

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 interface** **DHCP server enable/disable** • **0: Disable** • **1: Enable**

Field	Description
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

14.2 *Firewall Scripts: Deny/Allow/Forward*

ATOP SDK provides the simple mechanism to allow users to activate firewall on a gateway device. Users can easily establish their own firewall on their gateway device by adding or creating their rules within the following shell script files.

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

When firewall.sh script runs and the above script files exist, the related scripts will be activated.

15 Examples

This chapter provides an example of applications or scripts for ATOP SDK.

Example 1: Adding a new daemon in ATOP SDK:

ATOP SDK provides the example codes 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(s) and TCP network. Users can reference the example codes (tcp_server.c and Makefile) to gain an idea on how to create a daemon on the system of their device.

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