



Linux SDK

Integration Guide



SIERRA
WIRELESS

2131103
Rev 3

Important Notice

Due to the nature of wireless communications, transmission and reception of data can never be guaranteed. Data may be delayed, corrupted (i.e., have errors) or be totally lost. Although significant delays or losses of data are rare when wireless devices such as the Sierra Wireless modem are used in a normal manner with a well-constructed network, the Sierra Wireless modem should not be used in situations where failure to transmit or receive data could result in damage of any kind to the user or any other party, including but not limited to personal injury, death, or loss of property. Sierra Wireless accepts no responsibility for damages of any kind resulting from delays or errors in data transmitted or received using the Sierra Wireless modem, or for failure of the Sierra Wireless modem to transmit or receive such data.

Safety and Hazards

Do not operate the Sierra Wireless modem in areas where blasting is in progress, where explosive atmospheres may be present, near medical equipment, near life support equipment, or any equipment which may be susceptible to any form of radio interference. In such areas, the Sierra Wireless modem **MUST BE POWERED OFF**. The Sierra Wireless modem can transmit signals that could interfere with this equipment.

Do not operate the Sierra Wireless modem in any aircraft, whether the aircraft is on the ground or in flight. In aircraft, the Sierra Wireless modem **MUST BE POWERED OFF**. When operating, the Sierra Wireless modem can transmit signals that could interfere with various onboard systems.

Note: Some airlines may permit the use of cellular phones while the aircraft is on the ground and the door is open. Sierra Wireless modems may be used at this time.

The driver or operator of any vehicle should not operate the Sierra Wireless modem while in control of a vehicle. Doing so will detract from the driver or operator's control and operation of that vehicle. In some states and provinces, operating such communications devices while in control of a vehicle is an offence.

Limitation of Liability

The information in this manual is subject to change without notice and does not represent a commitment on the part of Sierra Wireless. SIERRA WIRELESS AND ITS AFFILIATES SPECIFICALLY DISCLAIM LIABILITY FOR ANY AND ALL DIRECT, INDIRECT, SPECIAL, GENERAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES INCLUDING, BUT NOT LIMITED TO, LOSS OF PROFITS OR REVENUE OR ANTICIPATED PROFITS OR REVENUE ARISING OUT OF THE USE OR INABILITY TO USE ANY SIERRA WIRELESS PRODUCT, EVEN IF SIERRA WIRELESS AND/OR ITS AFFILIATES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR THEY ARE FORESEEABLE OR FOR CLAIMS BY ANY THIRD PARTY.

Notwithstanding the foregoing, in no event shall Sierra Wireless and/or its affiliates aggregate liability arising under or in connection with the Sierra Wireless product, regardless of the number of events, occurrences, or claims giving rise to liability, be in excess of the price paid by the purchaser for the Sierra Wireless product.

Patents

This product includes technology licensed from QUALCOMM® 3G.

Manufactured or sold by Sierra Wireless or its licensees under one or more patents licensed from InterDigital Group.

Copyright

©2010 Sierra Wireless. All rights reserved.

Trademarks

AirCard® and Watcher® are registered trademarks of Sierra Wireless. Sierra Wireless™, AirPrime™, AirLink™, AirVantage™ and the Sierra Wireless logo are trademarks of Sierra Wireless.

Windows® and Windows Vista® are registered trademarks of Microsoft Corporation.

Macintosh® and Mac OS® are registered trademarks of Apple Inc., registered in the U.S. and other countries.

QUALCOMM® is a registered trademark of QUALCOMM Incorporated. Used under license.

Other trademarks are the property of their respective owners.

Contact Information

Sales Desk:	Phone:	1-604-232-1488
	Hours:	8:00 AM to 5:00 PM Pacific Time
	E-mail:	sales@sierrawireless.com
Post:	Sierra Wireless 13811 Wireless Way Richmond, BC Canada V6V 3A4	
Fax:	1-604-231-1109	
Web:	www.sierrawireless.com	

Consult our website for up-to-date product descriptions, documentation, application notes, firmware upgrades, troubleshooting tips, and press releases:

www.sierrawireless.com

Revision History

Revision number	Release date	Changes
1.0	Oct. 2008	Initial release
1.1	Nov. 2008	Updated Crosstool chains section (page 34)
1.2	Apr. 2009	General updates to procedures, removed obsolete references, etc.

Revision number	Release date	Changes
1.3	Oct. 2009	<ul style="list-style-type: none">• Updated sdk executable name• Added WWAN connection (Direct IP) details in Data Connection appendix• Updated PPP connection procedure in Data Connection appendix• Reworked Minicom appendix• Updated 'make' commands throughout document• Updated API-side design considerations section• Updated Ubuntu installation procedure• Updated aptest command format throughout document
2.0	Jul 2010	<p>New branding / template, including general review/update of content. Notable changes include:</p> <ul style="list-style-type: none">• Updated Ubuntu release support term to two years (from four)• Updated Unpacking the distributed files on page 27• Updated OABI (Old Application Binary Interface) crosstool on page 35• Updated Downloading the kernel on page 37• Updated Checking the current USB driver version on page 37• Updated Building images using cross tools on Ubuntu 8.04 on page 40• Added Debian package management on page 44• Updated Restoring the TS-7800 SBC on-board flash on page 45
3	Oct 2010	Clarified several procedures in Connecting to the TS-7800 ARM Platform

>> Contents

Introduction	11
Scope	11
Supported platforms	11
Supported Sierra Wireless modems	12
Document overview	12
 Machine Setup	 13
Installing Ubuntu 8.04	13
Opening a terminal window	14
Setting up the development environment	15
Checking system resources	15
Ubuntu passwords	15
Installing build-essential	15
Installing Minicom	15
Installing/updating Sierra Wireless Linux drivers	16
 Modem Setup	 19
CDMA modems	19
Pre-installation notes	19
AT commands	19
UMTS modems	20
Pre-installation notes	20
AT commands	20
 Introduction to the SDK	 23
SDK Quick Start Guide	23
Building the executables	23
Starting the Linux SDK executable	24
Compatibility	26
Hardware compatibility	26

Software compatibility	26
C programming language	26
Libraries	27
Object files	27
Installation	27
Unpacking the distributed files	27
Verifying the contents	27
Executables	28
API-Side design considerations.	29
WARNINGS	29
Documentation	30
Connecting to the TS-7800 ARM Platform	31
TS-7800 SBC.	31
Connecting a Linux Ubuntu 8.04 PC to TS-7800 SBC	31
Preparing the connection	32
Testing the connection	33
Restoring the PC's original connection	33
Accounts/passwords	34
Crosstool chains	34
Technologic Linux ARM kernel	36
Downloading the kernel	37
Setting up the USB driver for the TS-7800 SBC	37
Checking the current USB driver version	37
Downloading and building the USB driver	38
Installing the USB driver	38
Building images using cross tools on Ubuntu 8.04	40
Downloading images on the TS-7800 SBC	40
TS-7800 SBC with onboard flash	40
TS-7800 SBC with SD card	42
Building images directly on the TS-7800 SBC.	43
Debian package management	44

Miscellaneous TS-7800 SBC setup	44
Enabling logging	44
Enabling coredump feature	45
Restoring the TS-7800 SBC on-board flash	45
Establishing a Data Connection	47
Wireless WAN connection	47
WWAN connection procedure	47
PPP connection	49
Using Minicom	51
Identifying the AT commands tty device	51
Downloading and installing minicom	52
Index	53

>> 1: Introduction

1

The Linux SDK provides a series of Application Programmer Interfaces (APIs) that facilitate the configuration and control of Sierra Wireless modems by customer-supplied applications. These applications can range from elaborate connection managers to simple applications for monitoring some aspect of the modem's operation.

Scope

The audience for this document is anyone who needs to understand the steps required to set up a computer for developing applications atop the Sierra Wireless Linux SDK software package for either of the supported architectures. While there is a vast selection of software packages for Linux systems, this document's purpose is to identify the minimum set of packages required for compiling and building the SDK components themselves. Instructions for more complicated goals are beyond the scope of this document and it is up to individual customers to determine these requirements.

Supported platforms

The Linux SDK supports PC and embedded platforms as described below.

Sierra Wireless provides technical support to customers using these platforms. For information on other Linux kernel / architecture combinations, please contact Sierra Wireless Professional Services.

PC platforms

For PC application development, the SDK supports Ubuntu Linux 8.04 on x86/32 architecture.

The minimum recommended configuration (from the Ubuntu website, www.ubuntu.com) for the Intel platform is:

- 700 MHz x86 processor
- 384 MB of system memory (RAM)
- 8 GB of disk space
- Graphics card capable of 1024x768 resolution
- A network or Internet connection

Embedded platforms

For embedded application development, the SDK supports ARM9 with the Debian Sarge 3.1 distribution.

Support for the ARM9 processor is based upon Technologic Systems' TS-7800 platform running Linux Kernel version 2.6.21 with the Debian distribution installed. The SDK has been built and tested using the TS-7800 ARM9 SBC (Single Board Computer) with 128 MB DDR-RAM and 512 MB Flash (part number TS-7800-128-512F) and the Technologic Systems Development Kit (part number KIT-7800.)

If you are developing products for ARM9 architecture, consider purchasing the Technologic ARM9 SBC. For more information and to order, visit:

www.embeddedarm.com/products/board-detail.php?product=TS-7800

Supported Sierra Wireless modems

The Linux SDK consists of software that runs in user-mode on Linux systems. The SDK source uses the modem's USB Vendor and Product ID to identify the specific model of modem. For a current list of supported Sierra Wireless modems, see [Supported Sierra Wireless modems](#) on page 26.

Document overview

This document explains how to set up a new computer to begin developing command-line interface applications. For embedded systems this level of interaction is usually sufficient.

>> 2: Machine Setup

2

This chapter provides instructions on how to set up your computer by installing Linux and additional packages needed to enable you to build the SDK and drivers.

Installing Ubuntu 8.04

If you already have a computer with a Live CD installation of the 8.04 version of Ubuntu, skip to [Opening a terminal window](#) on page 14. The 8.04 release of Ubuntu is designated LTS (Long Term Support)—Ubuntu will continue to support this release for two years from its release date (April of 2008¹).

For this step you need:

- A computer with the minimum requirements described in [Supported platforms](#) on page 11
- A Ubuntu Live Installation CD containing Ubuntu 8.04 LTS Desktop Edition
- A working Ethernet data connection and cable

If you don't have access to a Live CD for Ubuntu 8.04, you can obtain one directly from the Ubuntu website at:

www.ubuntu.com/getubuntu

If you understand how to install Linux from a Live CD, then proceed directly to [Step 2](#) of the following installation procedure. The procedure is straightforward and, depending upon the computer, can take from just a few minutes to over an hour to complete.

Note: Before you start this procedure, make sure your computer has a working network connection.

To install Ubuntu 8.04:

1. Prepare the computer. If you are installing on a new machine, you have the following options:
 - Replace the existing operating system with Linux.
 - Buy a new hard drive and use it for the new installation.
2. When you have decided what hard drive to use, insert the Live CD into the CD-ROM drive and boot the computer from it. After a few minutes, the Live CD finishes booting and a Ubuntu desktop appears on the computer. Note the icon labelled "install" on the upper left corner of the desktop; click this icon to start the installation process.

1. In Ubuntu Linux, the release version numbers are composed of the year and the month. Therefore, Ubuntu 8.04 was released in month 04 (April) of 2008. Ubuntu aims to provide a release every six months.

3. At a point early in the installation process you are prompted to partition the hard drive. The Linux SDK has no specific requirements for how the hard drive is partitioned, so you are free to set up your disk partition any way you like. The simplest choice is to select the default option “Use the Entire Disk”.

Warning: *The computer also formats the drive during this step, so **be aware** that all information on all of the drive partitions will be gone once you begin this process.*

4. During the installation process you are prompted to configure a user account. Be especially careful to enter a username and password *that you will remember*; otherwise you may need to redo the installation from scratch. Make note of the information somewhere if you don’t think you can remember it. Each created user has `sudo`¹ privileges.

Once you have responded to the prompts, the computer starts the installation. This step takes from about 10 minutes to over an hour depending on the type of computer.

5. When the installation is complete, the computer prompts you to remove the CD and reboot (or continue using the Live CD). Reboot the machine and use the *username* and *password* noted in [Step 4](#) to log into the computer.
6. Once the installation is complete, the computer’s update manager starts automatically and determines whether there are updates to be installed. You may choose whether or not to proceed with any identified updates. (The SDK will work with Ubuntu 8.04 as installed or with any updates that are applied.)

Congratulations, you have completed the installation of Linux onto your computer!

Opening a terminal window

The procedures in this document require you to enter shell commands from a terminal window.

To start a bash shell in a terminal window:

1. Click the Applications menu, located in the upper left corner of the screen. A drop down menu appears.
2. Select Accessories->Terminal.

A window appears with the title “Bash Shell”.

You can open as many terminal sessions as you like in this manner.

1. Sudo is a Linux utility that allows you to access parts of your computer that are normally restricted to the computer administrator.

Setting up the development environment

Checking system resources

The following Linux commands are useful for checking system resources.

- Check available RAM: `$free -m`
- Check CPU information: `$cat /proc/cpuinfo`
- Check available disk space: `$df -m`
- Check kernel version: `$uname -a`

Sample output:

```
Linux 2.6.24-16-generic #1 SMP through April 10 13:23:42 UTC 2008  
i686 GNU/Linux
```

Ubuntu passwords

Some procedures in this document require your Ubuntu user account password.

Installing build-essential

The Ubuntu 8.04 distribution comes with the required compiler (i.e. GCC), however other essential header files and libraries are not included. If you intend to use your Ubuntu 8.04 machine as a development platform for the Linux SDK, you need to install the *build-essential* package.

You may need the Ubuntu 8.04 installation CD for this step.

To install the application:

1. Open a terminal window and run the command:

```
$sudo apt-get install build-essential
```
2. You will be prompted to enter a password. Use your account login password.
3. You are prompted with how much disk space this package consumes and asked whether you want to continue.
If you have the available space on disk, enter “Y”; the installation completes automatically.

Installing Minicom

Minicom is a Linux-based terminal emulation program similar to HyperTerminal used on Windows®-based computers. It is useful as an alternate means of testing that the modem is properly connected and functioning correctly.

You may need the Ubuntu 8.04 installation CD for this step.

1. Open a terminal window and run the command:

```
$sudo apt-get install minicom
```
2. If you are prompted to enter a password, use your account login password.
3. Accept the defaults for the prompts you may receive during the installation.

When the command prompt returns, minicom is ready to run. See [Using Minicom](#) on page 51 for details on setting up and using Minicom.

Installing/updating Sierra Wireless Linux drivers

Determining current driver versions

Depending upon the kernel version you are running, you may need to download and install the most appropriate, up-to-date Sierra Wireless drivers. There are two drivers available:

- Serial driver (“sierra.c”)—Typically bundled with your Linux operating system, this driver exposes several modem USB interfaces at ttys.
- Network driver (“sierra_net.c”)—Used with Sierra Wireless modems configured for Direct IP operation.

To determine which versions of these drivers are installed on your Linux system:

1. From a terminal window, enter the following command:

```
$modinfo <driver> (where <driver> is “sierra” or “sierra_net”)
```

The response will be “modinfo: could not find module <driver>” (the driver is not installed), or a comprehensive listing of driver information with the version number at the beginning.

Identifying available drivers

To identify available drivers:

1. For sierra.c, check the Driver Downloads table at http://sierrawireless.custhelp.com/app/answers/detail/a_id/500, which shows the latest driver version for each kernel.
2. For sierra_net.c, check the Driver Downloads table at http://sierrawireless.custhelp.com/app/answers/detail/a_id/641, which shows the latest driver version for each kernel.

Obtaining and installing drivers

The serial driver can be installed by itself (for Linux kernels not implementing Direct IP), or both the serial and network drivers can be installed together (for kernels implementing Direct IP). (The network driver is always installed together with the serial driver.)

To obtain and install only the serial driver:

1. Download the Sierra Wireless driver source code from http://sierrawireless.custhelp.com/app/answers/detail/a_id/500.
2. Navigate to the directory that contains the sierra drivers and extract the zip file by typing the following commands:


```
# tar -xvf v.x.y.z_Kernel2.x.y.zip
# cd v.x.y.z_Kernel2.x.y
```
3. Compile and install the new driver using the following commands:


```
# make
# sudo make install
```

Note: The Driver Downloads table lists the appropriate drivers for several Linux kernels—make sure to download the correct one.

4. If you are prompted to enter a password, use your Linux account login password.
5. To verify the driver installed correctly, repeat the procedure in [Determining current driver versions](#) on page 16.

To obtain and install both the serial and network drivers:

1. Download the Sierra Wireless serial and Direct IP driver source code from http://sierrawireless.custhelp.com/app/answers/detail/a_id/641.
2. Navigate to the directory that contains the sierra drivers and extract the zip file by typing the following commands:

```
# unzip vx.y.z_Kernel2.xy.y.zip  
# cd v.x.y.z_Kernel2.x.y
```
3. Compile and install the new drivers using the following commands:

```
# make  
# sudo make install
```
4. If you are prompted to enter a password, use your Linux account login password.
5. To verify the drivers installed correctly, repeat the procedure in [Determining current driver versions](#) on page 16.

Note: Sierra Wireless Linux drivers are free software; you can redistribute downloaded driver images and/or modify them under the terms of the GNU General Public License as published by the Free Software Foundation.

>> 3: Modem Setup

The tests in this chapter enable you to quickly determine whether your modem is connected to your development machine, and whether the machine detects it and allows users and programs to interact with it. (This chapter is not intended to be exhaustive.)

The port your Sierra Wireless modem uses for AT commands varies depending upon the model of modem you are using. You will be using a terminal program such as minicom to send AT commands and see their responses. [Appendix B](#) on page 51 provides a detailed explanation of how to determine which Linux file handles map to specific services on the modem. Unless you are sure of which tty to use, review [Using Minicom](#) on page 51 before attempting to use AT commands on your modem.

CDMA modems

Pre-installation notes

If you have a CDMA modem and it has not been activated, you must first activate it on a Windows system with the supplied software. Some service providers pre-activate their modems, but Sierra Wireless recommends that you first try the modem on a Windows machine.

AT commands

Usually (depending on your modem's configuration), AT commands should be issued to interface 0 on CDMA modems and cannot be used during a data connection (the Heatherington escape method is not supported). See [Identifying the AT commands tty device](#) on page 51 to determine the device handle for AT commands.

- Enter the following AT command in any terminal application (e.g. minicom) to obtain the signal quality (RSSI):

`at!rssi? or at!status`

A range from -60 dbm to -90 dBm is considered adequate.

- Enter the following AT command to see whether your modem is online:

`at!pcinfo`

- Enter the following AT command to turn the radio on (if the modem is in low power mode (LPM) the radio is off):

`at!pcstate=1`

For modems, the power LED on the card lights up.

UMTS modems

Pre-installation notes

You must ensure that the modem is properly configured before you can make a data connection to the wireless network. You need the following items:

Note: Through the rest of this document, SIM should be read as “SIM or USIM”.

Note: If you are using the MC8700 module you can connect to the network using Sierra Wireless' Direct IP technology. In this case the username and password information are optional and only the APN is required.

- A SIM or USIM with an activated account (Contact your wireless provider to obtain this.)
- The APN, username, and password. (Contact your wireless service provider for this information.) For example, for Rogers Wireless in Canada, this information is:
 - APN—internet.com
 - username—wapuser1
 - password—wap
- An appropriate means of connecting your modem to your host computer (for example, a USB cable)
- An appropriate power supply for your modem (for example, power adapter, batteries, etc., depending on your modem type)
- Access to a terminal emulation program such as minicom for interacting with the modem via its AT command interface

Install the SIM in the slot in the modem, connect the power to the modem, and connect the modem to the host computer.

AT commands

This section explains how to configure the modem so you can establish data connections with it and verify that it is properly connected to your host computer.

1. Determine which /dev/ttyUSBn file handle to use for AT Commands (see [Identifying the AT commands tty device](#) on page 51).
2. Start minicom (see [Downloading and installing minicom](#) on page 52) and enter the following AT command to verify that the modem is receiving and responding to AT commands:

```
ati <CR>

Manufacturer: Sierra Wireless, Inc.
Model: MC8780
Revision: F1_2_3_15AP C:/WS/FW/F1_2_3_15AP/MSM7200R3/SRC/
AMSS 2008/07/09 13:02:1
1
IMEI: 354219011234562
IMEI SV: 13
FSN: D330188396810
3GPP Release 6
+GCAP: +CGSM,+FCLASS,+DS
OK
```

If you do not get a response, then there is a problem—most likely one of the following:

- Modem is not properly connected to the computer
- Modem is not connected to power
- Sierra Wireless Linux driver is not installed

3. Check the state of the SIM connection to the modem using the following AT command (Note: The correct response is displayed in this example):

```
at+cpin? <CR>

+CPIN: READY
OK
```

This indicates that the modem detects the SIM and that the SIM is functioning correctly. If there is a problem, you will see one of the following errors instead:

+CME ERROR: —Indicates the SIM is not inserted.

+CPIN: SIM PIN—Indicates the SIM is locked and requires a PIN to be entered.

4. To display basic modem status details, enter the following command:

```
at!gstatus? <CR>

!GSTATUS:
Current Time:415Temperature:28
Bootup Time:349Mode:LOW POWER MODE
System mode:GSMPS state:Not attached
WCDMA band:WCDMA800GSM band:GSM900
WCDMA channel:4449GSM channel:1
GMM (PS) state:NULL---
MM (CS) state:NULL---
WCDMA L1 State:L1M_DEEP_SLEEP RRC State:DISCONNECTED
RX level (dBm):-91 (Not updated)
OK
```

The output shown is an example of what you can expect to see in response to the request.

5. Verify that the modem is set up correctly—Quickly test this by configuring a profile and establishing a data connection. (You could also use available APIs for managing modem profiles instead of this method.)

Using the APN for your modem (from [Pre-installation notes](#) on page 20), enter the following command:

```
at+cgdcont=1,"IP","<apn>" <CR>

OK
```

Your modem should now be ready for use.

If you arrived at this step without noticing any major deviation from the examples above, then your modem is properly connected to the host computer, the Sierra Wireless driver is installed and recognizes the modem, and the modem is functioning properly.

>> 4: Introduction to the SDK

This chapter contains a comprehensive introduction to the SDK. The first section is the quick start guide, which explains how to unpack and build the SDK executables and how to unpack and start executing the SDK. The remaining sections go into more detail on various aspects of the SDK.

SDK Quick Start Guide

This section describes the steps needed to quickly unpack and start using a release of the SDK. After reading this section, you should be able to compile applications and/or execute them on a supported Sierra Wireless Linux SDK development platform.

Building the executables

Although the as-delivered release already contains executables, libraries and object files, you may wish to become familiar with the procedure for building them. The steps in this section guide you through the procedure for compiling and linking the source files provided with the Linux SDK.

Note: If you simply want to run the executables and verify that they can function on your machine, skip this section and go to [Starting the Linux SDK executable](#) on page 24.

The steps in this procedure are all performed from a Bash shell. If you do not know how to open one, review the information in [Opening a terminal window](#) on page 14.

This procedure shows you how to build images that can run directly on 80x86/32 Linux platforms, or on ARM9 platforms.

1. Go to the desired directory on your computer and uncompress the contents of the release into it. The top level directory is named after the release. For example, when you have completed this step, the form of the directory name is:

LinuxSDK_a_b_c_d

where “a_b_c_d” represents the release version number. As an example, if the release were named LinuxSDK_V1_3_0_0, use the following command to uncompress it:

\$tar -xvzf LinuxSDK_V1_3_0_0.tar.gz

2. Go to the “pkgs” directory, below the directory just created:

\$cd LinuxSDK_V1_3_0_0/pkgs

3. Enter the following command:

\$make -f pkgs.mak clean

Note: For the ARM9 build, you must spell “arm9” with lower-case letters; “ARM9” will not work.

This removes all the objects for all the packages that make up the SDK so that the next step forces the objects to be built.

4. Run the make command without arguments to build all the core files required by the SDK:
 - (80x86/32 Linux platforms) `$make -f pkgs.mak`
 - (ARM9 platforms) `$make -f pkgs.mak CPU=arm9`
5. Inspect the AP and SDK folders and confirm that the executables are present—“sdi386”, and “aptesti386” or “aptestarm9” depending on the platform you compiled for. This confirms that you are able to build the SDK on this machine.
6. To verify that the executables are runnable:
 - (80x86/32 Linux platforms) Proceed to the next section—[“Starting the Linux SDK executable”](#) (below).
 - (ARM9 platforms) See [Downloading images on the TS-7800 SBC](#) on page 40.

Note: If you intend to use the GDB debugger for testing, the SDK makefiles support an option (SYMBOLS) to compile for this purpose. The SYMBOLS variable should be enabled as follows:

`make -f pkgs.mak SYMBOLS=ON`

This variable can be set for any make, not just for “pkgs.mak”.

Starting the Linux SDK executable

The information in this section is duplicated elsewhere in this document. However, if all you want to do is unpack the SDK and test that it works, then this section describes the steps you need to follow. It assumes you are running Ubuntu 8.04 on an 80x86/32 architecture.

By following the steps below, you can quickly unpack the contents of this release and verify that it functions correctly.

This procedure assumes your computer already has the Sierra Wireless Linux driver installed. If you have not already done so, see [Installing/updating Sierra Wireless Linux drivers](#) on page 16 for details.

Note: You can follow this procedure with or without a Sierra Wireless modem attached to a USB port on your computer. The end results will differ, but the test application and the SDK daemon will still start and run.

1. If you have not already done so, perform steps 1 and 2 in [Building the executables](#) on page 23 to unpack your release.
2. Change to the “ap” directory under the “pkgs” directory:
`$cd ap`

Note: [Step 3](#) is suggested as a convenient way to verify the SDK process is running. The process runs as a daemon and therefore cannot generate `printf()` messages to a shell window. Instead, it sends logs to the identified file. You can use the log to verify the operation of the SDK and, in future, to troubleshoot problems you may encounter during development.

3. Optionally, open another bash shell to monitor the SDK daemon's logging traffic—Enter the following command:
`$tailf /var/log/user.log`
4. Execute the Sierra Wireless engineering test executable using the following command:
`$./aptesti386 --help`
 This command prints a short summary of how to use the `aptesti386` executable.
5. The test program gives you access to most of the APIs through a series of built-in test procedures. If you want to see a list of these procedures, enter the following command:
`$./aptesti386 -p ../../build/bin/i386/swisdk -n x`
 You will see a two-column list of test cases that you can try. The left-hand column identifies the test case to use, and the right-hand column identifies the API that will be called. Use the test case identifier in the left-hand column to specify a test when running the executable.
 You will also see activity in the logging window you opened in [Step 3](#), above. This indicates the SDK daemon was started by the test program. Various logging messages appear periodically until the daemon is killed.

The following steps demonstrate how to use one of the test cases you discovered in the previous procedure.

Note: For this step, a Sierra Wireless modem must be connected to a USB port on your computer and the computer must have a suitable version of the Sierra Wireless Linux driver installed.

1. To obtain the firmware version number of the modem that is connected to your computer, run the following command:
`$./aptesti386 -p ../../build/bin/i386/swisdk -n t1`
 If you connect the modem to the computer and then run this command in quick sequence, you might notice the application's repeated printing of the message:
`Air Server not available - SLEEPING!`
 However, after a few seconds the SDK detects the modem and the test proceeds. When the test is complete, you should see a message similar to the following examples (dependent upon the type of modem you have connected):
 (UMTS)
`H1_1_9_27MCAP G:/WS/FW/H1_1_9_27MCAP/MSM6280/SRC 2008/04/15 09:20:32`
 (CDMA)
`Modem Firmware Version:
p2415801`
2. The program should end normally. If you have made it this far, then the executables delivered with the SDK have been demonstrated to run successfully on your machine.

Note: The SDK process continues to run after the test has finished.

Compatibility

This section outlines the hardware and software compatibility of the SDK.

Hardware compatibility

Two Sierra Wireless software components have hardware dependencies built into them—the Sierra Wireless USB driver and the SDK itself. This dependency is predominantly restricted to the USB Vendor ID and Product ID that the driver and SDK detect, although there are additional dependencies in the driver, not discussed in this document.

Supported architectures

This build of the Linux SDK contains object files for 32-bit Intel 80x86 and ARM9 architectures; the ARM9 build only runs on the Technologic Systems TS-7800 SBC. If you want to build for ARM9 targets on your computer, you need to install additional compilation tools (see [Crosstool chains](#) on page 34 for details).

Sierra Wireless recommends the use of the [Technologic Systems](#) TS-7800 SBC for your early-stage development of ARM9-based applications.

Supported Sierra Wireless modems

[Table 4-1](#) lists the Sierra Wireless modems that are compatible with this version of the SDK. If you wish to use the Linux SDK with a Sierra Wireless product not listed in this table, please contact your Sierra Wireless Account Manager.

Table 4-1: AirPrime Modems Supported by Linux SDK

CDMA	UMTS
MC5725, MC5725V MC5727, MC5727V MC5728V	MC8700 MC8775, MC8775V MC8780, MC8781, MC8785V MC8790, MC8790V, MC8791V, MC8792V

Software compatibility

This version of the Linux SDK has been developed and tested using Ubuntu 8.04 LTS as installed from the Live CD. The as-delivered image contains source code, objects and executables for the i386/32 bit architecture.

C programming language

The Linux SDK is coded in the C programming language and all external entry points (API functions) are callable by any language that can call C functions.

Libraries

Libraries are created using the Linux archive tool, “ar”.

Object files

Objects compiled for Intel x86 architecture are output from the Linux GCC compiler. The Linux “file” utility identifies these object files as follows:

```
filename.o: ELF 32-bit LSB relocatable, Intel 80386, version 1 (SYSV), not
stripped
```

Installation

SDK releases are distributed as tarred, zipped images. The steps for unpacking the software are described in detail in the following subsection.

Unpacking the distributed files

To unpack your Linux SDK release distribution file (for example, LinuxSDK_a_b_c_d.tar.gz):

1. Copy the distribution file into a directory where the Linux SDK directory tree will be extracted (for example, /exampleonly).
2. Use the ‘tar’ command to uncompress the distributed file, storing the contents in a subdirectory named for the Linux SDK release:

```
$tar -xvzf LinuxSDK_a_b_c_d.tar.gz
```

The contents are saved to distribution file are stored in /exampleonly/
LinuxSDK_a_b_c_d.

Distribution root directory—<SDK_ROOT>

The distribution root directory (in this example, /exampleonly/LinuxSDK_a_b_c_d) is referred to as <SDK_ROOT> throughout this document.

Verifying the contents

Executables for the desktop environment are included with distributions and should work out-of-the-box. To ensure the executables are able to run on your machine:

1. Open two shell windows and position them so they can be seen simultaneously. In one of the windows enter the following Linux command:

```
$tailf /var/log/user.log
```

This causes the contents of the user.log file to be printed to the console whenever any changes are made to that file.

2. In the other shell window, use the following command to ensure that a previous version of the Linux SDK process is not already running:

```
$killall swisdk
```

These cause the SDK to be terminated immediately if there is one running. ([Executables](#) on page 28 explains why this step is required.)

3. Change to the Linux SDK application source code directory ('ap') that was created in the directory where when you unpacked the distribution. (For example, /exampleonly/LinuxSDK_a_b_c_d/pkg/ap.)
4. Start your version of the SDK by entering the following command into the same shell window you used to kill the currently-executing SDK (if any) in [Step 2](#):

```
$/aptesti386 -p ../../build/bin/i386/swisdk -v -n t1
```

If there is no Sierra Wireless modem installed on your computer, the test application will start and display the following message approximately once every ten seconds:

```
Air Server not available - SLEEPING
```

You should see the logfile (displayed in the other shell window) come alive with the display of several messages, indicating the SDK process has started.

If you have a compatible Sierra Wireless modem installed, after several seconds the aptesti386 command you entered above terminates and the firmware version string from the attached modem is displayed (similar to the following examples, dependent upon the type of modem you have connected):

```
(UMTS)
H1_1_9_27MCAP G:/WS/FW/H1_1_9_27MCAP/MSM6280/SRC 2008/
04/15 09:20:32
```

```
(CDMA)
Modem Firmware Version:
p2415801
```

Executables

A functioning application consists of two executable images—one provided by the customer embodying their application and containing all the Linux SDK APIs, the other designed to run as a daemon under Linux. For simplicity, this document refers to these executables as the *API-side* and the *SDK-side*. Each one is a separate process under Linux, and the API-side process has certain constraints placed upon it that developers are required to adhere to.

When you start your application by starting the API-side process, it is the API-side's job to start up the SDK by calling the entry point *SwiApiStartup()*. The first time this is done after rebooting the computer, the call to *SwiApiStartup()* actually starts the SDK process. Once the SDK is running it is designed to never terminate – until the hosting computer is subsequently switched off or rebooted. Therefore, if your application is started, then terminated, then restarted again some time later, subsequent calls to *SwiApiStartup()* will detect that the SDK-side is already running and take no further action. It is important to keep this in mind when making changes to code that runs in the SDK side. If you don't kill the existing SDK process, the new one won't start and your changes won't take effect.

Note: Even if you restart a test application or any of the utilities or sample applications subsequent to this, you must always specify the path to the SDK executable on the command line using the "-p" switch and path.

To terminate the SDK-side for any reason, use the following Linux command:

```
$killall swisdk
```

API-Side design considerations

As a developer, you are in charge of the architecture of the API process, but here are some suggestions for how to organize it. Since there are two processes (SDK-side and API-side), a method of inter-process communication (IPC) is required for them to exchange requests and responses. The SDK uses UNIX Socket Datagram protocol, which is available in most Linux systems. The protocol over these IPC channels has the following properties:

- Packets are guaranteed to be delivered, so the protocol is as reliable as TCP.
- Packets are sent as discrete datagrams and are similar in this regard to UDP.

The vast majority of APIs are implemented as blocking calls—in other words, the calling thread is blocked until the modem's response is received. However, it would be impractical for applications to be forced to poll every single value of interest on the modem and so the modem is designed to support notifications on selected data elements. If an element changes significantly, the modem originates a notification packet. Applications must specifically enable notifications on all of the modem's data elements of interest. Once done, notifications remain enabled until the modem is subsequently disconnected or rebooted.

The Linux SDK uses two separate IPC channels per application—one to handle stop-and-wait calls and the other to handle notifications from the modem. Developers must allocate a separate thread to manage traffic over each of these IPC channels in addition to whatever other mechanisms their applications may require.

WARNINGS

- Calls made via API functions are stop-and-wait. Therefore, if you call a second API function while you are waiting for the response from a previous call, the SDK will abort, reporting a stop-and-wait violation error. If this happens, check your application to make sure you are not calling a second API function before the previous response is received.
- If your application creates just one single thread for handling Notifications and stop-and-wait calls, reception of the notifications will be delayed until the API function call returns. This would not be good practice. For an example of a better way to handle notifications, refer to `aptest.c` and `aptestnotify.c` in the `pkgs/ap` directory.

Documentation

The SDK uses Doxygen to generate an API function and data reference. Doxygen outputs this information in HTML format and any HTML-compatible browser can be used to view the information.

Note: Only the source material in the `pkgs/ap` directory is documented in this way.

>> 5: Connecting to the TS-7800 ARM Platform

For embedded application development, the SDK supports ARM9 with the Debian Sarge 3.1 distribution.

If you are developing products for ARM9 architecture, consider purchasing the Technologic ARM9 SBC (Single Board Computer). For more information and to order, visit:

www.embeddedarm.com/products/board-detail.php?product=TS-7800

TS-7800 SBC

The TS-7800 SBC has no video controller or keyboard interface. COM1 is typically used as a console port to interface the TS-7800 SBC to a standard terminal emulation program on a host PC.

The TS-7800 SBC has three jumpers (JP1, JP2, JP3):

- JP1: If JP1 is installed, the board attempts to boot (load kernel and initial RAM disk) from the SD card, totally independent of the on-board flash. If JP1 is not installed, the board boots from on-board flash and executes Linux from the SD card. Both boot sequences are identical. (Booting exclusively from the SD Card (JP1 installed) allows onboard flash recovery, if needed.)
- JP2: turns ON or OFF console output to COM1
- JP3: If JP3 is installed, the CPU runs at 333MHz instead of 500MHz, saving power and increasing reliability in extreme thermal conditions.

Use a terminal emulation program such as HyperTerminal on Windows systems or minicom on Linux PCs.

Technologic Systems provides a full manual for the TS-7800 SBC at www.embeddedarm.com/about/resource.php?item=393.

Connecting a Linux Ubuntu 8.04 PC to TS-7800 SBC

A host PC can communicate with the TS-7800 SBC using:

- A serial (RS-232) cable
- An Ethernet connection (after establishing a serial connection and booting a full Linux distribution on the TS-7800 SBC, described below)

Preparing the connection

For both methods, you must prepare the Linux Ubuntu 8.04 PC and TS-7800 SBC for communication as follows:

1. Configure a serial connection:
 - a. On the TS-7800 SBC, make sure jumper JP2 is installed.
 - b. Connect a serial cable between the serial ports of the PC and TS-7800 SBC.
 - c. On the Ubuntu 8.04 Linux PC, use the minicom terminal emulation program to set the serial port to the following parameters:
115200, 8N1, No hardware flow control
(See [Using Minicom](#) on page 51 for basic help using minicom.)
2. Prepare the TS-7800 SBC for communication. (Note: The TS-7800 SBC must boot with a full Linux distribution before communication can occur with a host PC.)
 - a. Boot the TS-7800 SBC. A Linux prompt appears on the terminal emulator within two to four seconds depending on whether it boots from on-board flash or from an SD card.
(Note: This is not a full Linux bootup, so communication is not yet possible.)
 - b. On the terminal emulator, enter the following shell command to boot a full Debian Linux distribution from the SD card or on-board flash (this results in a writeable file system):
`#exit`
Communication between the host and the TS-7800 SBC is now possible.
3. If you want to connect via Ethernet:
 - a. Disconnect the PC's existing network connection (Ethernet cable, wireless device, etc.).
 - b. Connect the TS-7800 SBC board to the PC using an Ethernet *crossover* cable.
 - c. On the PC, select System>Administration>Network.
 - d. In the Network Settings window, click the Unlock button to enable access to network settings.
 - e. If the Authenticate window appears asking for a password to proceed, enter your user account password (not the root password) and click the Authenticate button.
 - f. In the Network Settings window, select Wired Connection and click the Properties button.
 - g. In the Properties window, uncheck Enable roaming mode.
This reveals the Connection Settings fields. Complete them as follows:
 - Configuration: Static IP address
 - IP address: 192.168.0.1
 - Subnet mask: 255.255.255.0
 - Gateway address: 192.168.0.1

Note: You can use a different terminal emulator if you prefer.

- h. Click OK.
- i. Close the Network Settings window

Testing the connection

To check the connection between the Ubuntu 8.04 Linux PC and the TS-7800 SBC, use any of the following methods:

Note: The TS-7800 SBC has a default IP address of 192.168.0.50.

- To test the connection
 - a. On the PC, enter the following command:
`$ping 192.168.0.50`
- To log into the TS-7800 SBC:
 - a. On the Ubuntu 8.04 Linux PC, use the following command (or other preferred access method)
`$ssh root@192.168.0.50` ¹

Restoring the PC's original connection

After finishing with the TS-7800 SBC, you can restore the Ubuntu 8.04 Linux PC to dynamically assigned IP address so that it can access the Internet.

To restore the Ubuntu 8.04 Linux PC to dynamically assigned IP addresses:

- a. In the Network Settings window, click the Unlock button to enable access to network settings.
- b. In the Authenticate window, enter the password as requested and click the Authenticate button.
- c. In the Network Settings window, select Wired Connection and click the Properties button.
- d. In the Properties window, select Enable roaming mode.
- e. Click OK.
- f. Close the Network Settings window.
- g. Restore the PC's previous network connection (Ethernet cable, wireless device, etc.).

1. ssh and scp commands work only if the specified account has a non-empty password. If the 'root' password is empty, you can use the 'eclipse' account if it has root permissions, or set a password as described in [Assigning a root password](#) on page 34.

Accounts/passwords

The TS-7800 SBC ships with the following accounts and passwords:

- root—Empty password
- eclipse—Password: 'eclipse'

Some of the procedures in this chapter use the 'scp' and 'ssh' commands. These commands can be run from root only if a non-empty password has been set, otherwise they must be run from the eclipse account (after assigning root permission to the account).

Assigning a root password

To assign a non-empty root password, you can use the following method:

1. Use a serial terminal connected to the TS-7800 SBC to log in as:
Username: eclipse
Password: eclipse
2. Enter the following command:

```
$ su
```
3. At the shell prompt, enter the following command:

```
# passwd
```
4. When prompted, enter your new password.
5. When prompted, enter the new password again to confirm the change.

Crosstool chains

Crosstool chains are available from Technologic Systems for cross-compiling images (using the Linux SDK) on a Ubuntu 8.04 Linux PC for download to the TS-7800 SBC, and for building the TS-7800 SBC kernel and Sierra USB driver.

The build environment as delivered is structured for use with the gcc compiler. To use a different compiler, please contact Sierra Wireless Technical Support for assistance.

Cross-compiling to other platforms

If you want to cross-compile images for other hardware platforms, you must update the gen.mak makefile (located in the <SDKROOT>/pkgs directory) to include the appropriate crosstool chain as follows (see the excerpt following the procedure for an example):

1. In the file gen.mak, make a copy of the "ifeq (\$CPU),arm9)" statement.
2. In the new statement, replace "arm9" with an identifier of your choice for the new platform (for example, the "xyz" platform).
3. Set the CROSS_COMPILE variable to the path of the correct crosstool for the xyz platform. (In this example, the crosstool is "xyz-linux-".)

4. Make sure the crosstool chain includes the gcc, ar, and ranlib tools.

Excerpt from gen.mak showing support for 'xyz' platform

```

1  ifeq ($(CPU),arm9)
2      CROSS_COMPILE=/usr/local/opt/crosstool/arm-linux/
   gcc-3.3.4-glibc-2.3.2/bin/arm-linux-
3  endif
4
5  ifeq ($(CPU),xyz)
6      CROSS_COMPILE=/usr/local/opt/crosstool/xyz-linux/
   gcc-3.3.4-glibc-2.3.2/bin/xyz-linux-
7  endif
8
9  #$(CROSS_COMPILE) is empty if not defined
10 CC:=$(CROSS_COMPILE)gcc
11 AR:=$(CROSS_COMPILE)ar
12 RANLIB:=$(CROSS_COMPILE)ranlib

```

OABI (Old Application Binary Interface) crosstool

The OABI crosstool chain is required if you want to develop applications using the Linux SDK (Debian Sarge uses OABI executables so these tools are necessary).

This tool chain runs with the onboard flash.

Download and install the OABI crosstool chain

1. On the Ubuntu 8.04 Linux PC, open a browser and go to:
<ftp.embeddedarm.com/ts-arm-sbc/ts-7800-linux/cross-toolchains>.
2. Select the link for the crosstool chain—ts7800-crosstool-linux-oldabi-0.28rc39.tar.bz2
3. When prompted, save the image to disk.
4. When the download finishes, move the file to your root ('/') directory using the following commands:


```

$ sudo mv ts7800-crosstool-linux-oldabi-0.28rc39.tar.bz2 /
<enter your login password if prompted>
$ cd /

```
5. Untar the image (extract the files) using the following command:


```

$ sudo tar -xvf ts7800-crosstool-linux-oldabi-0.28rc39.tar.bz2

```
6. If desired, either move the original tar file to a permanent storage location, or delete it. (You must be logged in as 'root' to perform this operation.)

Note: This image may take several minutes to download.

Note: The “a_b_c_d” in the directory name represent the SDK release number.

Make SDK executables for the TS-7800 SBC

After installing the OABI crosstool chain, you can then make the SDK executables for the TS-7800 SBC:

1. Change directory to <SDK_ROOT>/pkgs (where <SDK_ROOT> is the directory where you installed the SDK. See [Unpacking the distributed files](#) on page 27.)
2. Enter the following command (make sure ‘arm9’ is in lower-case; ‘ARM9’ will not work):

```
$make -f pkgs.mak CPU=arm9 clean; make -f pkgs.mak CPU=arm9 complete
```
3. You can now download the following executables to the TS-7800 SBC (See [Downloading images on the TS-7800 SBC](#) on page 40):
 - <SDK_ROOT>/build/bin/arm9/swisdk
 - <SDK_ROOT>/pkgs/ap/aptestarm9

EABI (Embedded Application Binary Interface) crosstool

The EABI crosstool chain is required to build the TS-7800 SBC kernel and the Sierra USB driver.

To download and install the EABI crosstool chain

1. On the Ubuntu 8.04 Linux PC, open a browser and go to: <ftp.embeddedarm.com/ts-arm-sbc/ts-7800-linux/cross-toolchains>.
2. Select the link for the crosstool chain—ts7800-crosstool-linux-gnueabi-2500q3-2.tar.bz2.
3. When prompted, save the image to disk.
4. When the download finishes, untar the image into a temporary directory (it untars into the subdirectory ‘arm-none-linux-gnueabi’).
5. From the temporary directory, move the extracted files to /usr/local using the following command (makefiles expect the files to be at /opt/sw):

```
$sudo mv arm-none-linux-gnueabi /opt/sw
```

Note: This image may take several minutes to download.

Technologic Linux ARM kernel

To build the Sierra USB driver, you must download the Linux ARM kernel source code. (The source includes a header file that is used when building the USB driver.)

Note: You do not need to compile the kernel before building the USB driver.

Downloading the kernel

To download the TS-7800 SBC kernel source code:

1. On the Ubuntu 8.04 Linux PC, open a browser and go to:
<ftp.embeddedarm.com/ts-arm-sbc/ts-7800-linux/sources>
2. Double-click the tar file—linux-2.6.21-ts-src-latest.tar.gz.
3. When prompted, save the image to disk.
4. Untar the image into the /opt/sw directory.
5. Enter the following commands:

```
$make distclean  
$make ts7800_defconfig  
$make oldconfig  
$make vmlinux  
$make modules
```

Setting up the USB driver for the TS-7800 SBC

Make sure that you have the correct USB driver for the kernel version you are using. (The TS-7800 SBC comes with pre-installed kernel 2.6.21-ts.) If you need a new driver, download the source code from the Sierra Wireless knowledge base, and then build and install the driver using the instructions below.

Checking the current USB driver version

To determine if a newer Sierra Wireless driver must be installed:

1. On the TS-7800 SBC, enter the following command to check which version is currently installed:

```
#!/sbin/modinfo sierra
```
2. On the Ubuntu 8.04 Linux PC, open a browser and go to <http://sierrawireless.com>.
3. Select support+downloads > Knowledge Base.
4. Search for “Linux operating system”.
5. From the results, select the article “Can I use my Sierra Wireless modem on a Linux operating system? (v.x.y.z)” (where x.y.z is a version number like “1.17.32”)
6. Review the Driver Downloads list to determine if a newer driver exists for your kernel version (2.6.21). If a newer driver exists, then continue to the next section—["Downloading and building the USB driver"](#).

Downloading and building the USB driver

1. If not already done, follow the [Checking the current USB driver version](#) procedure above to display a list of available driver downloads.
2. In the Driver Downloads list, click the link for the driver that matches your kernel (2.6.21).
3. When prompted, save the image to disk in a temporary directory.
4. Uncompress the downloaded image.
5. Modify the makefile to reflect the same cross tools used to configure the kernel 2.6.21-ts as shown below:

```
BUILDDIR := /opt/sw/linux-2.6.21-ts-src-latest/
PWD      := $(shell pwd)
USBDIR   := /opt/sw/linux-2.6.21-ts-src-latest/drivers/usb/serial
GCC      := /opt/sw/arm-none-linux-gnueabi/bin/arm-none-linux-gnueabi-gcc
CC       := /opt/sw/arm-none-linux-gnueabi/bin/arm-none-linux-gnueabi-gcc
```

6. Build the driver—Enter the following command:
`$make`
7. Install the USB driver—See ["Installing the USB driver"](#) below.

Installing the USB driver

After you build the driver, you must move it on to the TS-7800 SBC.

1. On a serial terminal connected to the TS-7800 SBC, enter the following command to boot a full Debian Linux distribution from the SD card or on-board flash (this results in a writeable file system):
`#exit`
2. Remove the old sierra driver:
 - a. On the Ubuntu 8.04 Linux PC, use the following command (or other preferred access method) to log in to the TS-7800 SBC:
`$ssh root@192.168.0.50` ¹
 - b. On the TS-7800 SBC (with root permissions), enter:
`#cd /lib/modules/2.6.21-ts/kernel/drivers/usb/serial/`
 - c. Make a backup copy of the old driver under a different name (this example uses sierra.old):
`#cp sierra.ko sierra.old`

1. ssh and scp commands work only if the specified account has a non-empty password. If the 'root' password is empty, you can use the 'eclipse' account if it has root permissions, or set a password as described in [Assigning a root password](#) on page 34.

Note: The message "ERROR: Module sierra is in use" appears if you try to remove the driver while a modem is still connected.

Note: The message "ERROR: Module usbserial is in use" appears if you try to uninstall/reinstall the driver while a modem is still connected.

Note: You could also use 'lsmod', but modinfo is preferred.

- d. Copy the new sierra driver (sierra.ko) from the Linux PC to the TS-7800 SBC. (You must have root access to perform this operation.)
From the directory containing the new driver on the Linux PC, enter the following command:
`$scp sierra.ko root@192.168.0.50:/lib/modules/2.6.21ts/kernel/drivers/usb/serial/`¹
- e. Make sure there is no modem connected to the TS-7800 SBC, and then (on the TS-7800 SBC) delete the old driver (sierra):
`#rmmod sierra`
If no error message appears, you can continue to the next step.
3. Replace the usbserial driver before installing a new sierra driver—this clears any dependencies between the usbserial and old sierra drivers:
 - a. Uninstall the usbserial driver—Enter:
`#rmmod usbserial`
 - b. Reinstall the usbserial driver—Enter:
`#insmod usbserial.ko`
4. Install the new sierra driver—Enter the following command to insert the sierra.ko module into the kernel on the TS-7800 SBC:
`#insmod sierra.ko`
5. Confirm the driver loaded correctly—Enter the following command:
`#!/sbin/modinfo sierra` (This should display the version number of the new driver.)

Reinstalling drivers after rebooting

Whenever the TS-7800 SBC is rebooted, you need to reinstall both drivers:

1. On a serial terminal connected to the TS-7800 SBC, enter the following command to boot a full Debian Linux distribution from the SD card or on-board flash (this results in a writeable file system):
`#exit`
2. On the Ubuntu 8.04 Linux PC, use the following command (or other preferred access method) to log in to the TS-7800 SBC:
`$ssh root@192.168.0.50`¹
3. On the TS-7800 SBC, enter:
`#insmod usbserial.ko`
`#insmod sierra.ko`

1. ssh and scp commands work only if the specified account has a non-empty password. If the 'root' password is empty, you can use the 'eclipse' account if it has root permissions, or set a password as described in [Assigning a root password](#) on page 34.

Building images using cross tools on Ubuntu 8.04

To build the API and SDK images for use on the TS-7800 SBC, the OABI crosstool chain must have been downloaded earlier—see [OABI \(Old Application Binary Interface\) crosstool](#) on page 35. (The cross tools should be in the Linux PC's /usr/local/opt directory.)

1. On the Ubuntu 8.04 Linux PC, enter the following command to build the SDK and API images:

```
$make -f pkgs.mak CPU=arm9 clean; make -f pkgs CPU=arm9 complete
```

 This produces the <SDK_ROOT>/build/bin/arm9/swisdk and <SDK_ROOT>/pkgs/ap/aptestarm9 executables which can now be downloaded to the TS-7800 SBC.
2. Download the SDK and API images (executables) to the TS-7800 SBC—see ["Downloading images on the TS-7800 SBC"](#) below.

Downloading images on the TS-7800 SBC

TS-7800 SBC with onboard flash

The TS-7800 SBC's root login is used for the on-board flash distribution.

Downloading the SDK image

To download the SDK image to the TS-7800 SBC:

1. On the TS-7800 SBC (booted from on-board flash), create a temporary directory for downloading images—this example uses /home/testapi, but you can name the directory as you like.
2. Enter the following command from the LinuxSDK_a_b_c_d/pkgs/sdk directory to copy the SDK executable to the test directory where it will be used to run the test API (See [Running the test API](#) on page 41.):

```
$scp swisdk root@192.168.0.50:/home/testapi/ 1
```

If this is the first time the scp command is run, Ubuntu 8.04 issues the following message:

```
The authenticity of host '192.168.0.50
(192.168.0.50)' can't be established.
RSA key fingerprint is 7f:...
Are you sure you want to continue connecting (yes/no)?"
```

1. ssh and scp commands work only if the specified account has a non-empty password. If the 'root' password is empty, you can use the 'eclipse' account if it has root permissions, or set a password as described in [Assigning a root password](#) on page 34.

Note: The root account's home directory is '/.

- a. Enter Y to establish the connection.
Ubuntu 8.04 replies with:
“Warning: Permanently added ‘192.168.0.50’ (RSA) to the list of known hosts.”
3. When prompted, enter the TS-7800 SBC root password. When you enter the correct password, the image is copied onto the TS-7800 SBC.

Downloading the application image

To download the Application image to the TS-7800 SBC:

1. On the Ubuntu 8.04 Linux PC, enter the following command from the LinuxSDK_a_b_c_d/pkgs/ap directory:

```
$scp aptestarm9 root@192.168.0.50:/home/testapi/ 1
```
2. When prompted, enter the TS-7800 SBC root password. When you enter the correct password, the image is copied onto the TS-7800 SBC.

Accessing the TS-7800 SBC

To access the TS-7800 SBC:

1. On the Ubuntu 8.04 Linux PC, enter the following command and enter the root password when prompted:

```
$ssh root@192.168.0.50 1
```
2. Change directory to the test directory where you previously downloaded the images:

```
#cd /home/testapi
```

Running the test API

To run the test API on the TS-7800 SBC:

1. Enter the following command (the -p switch specifies the path to the SDK executable—in the previous procedures, swisdk and aptestarm9 were placed in the same directory):

```
#./aptestarm9 -p ./swisdk -n t1
```

Note: To display available options, enter #./aptestarm9 -h.

Checking the SDK state

To check whether the SDK is executing:

1. On the TS-7800 SBC, enter the following command:

```
#ps aux | grep sdk
```

Terminating SDK processes

To terminate SDK processes:

1. On the TS-7800 SBC, enter the following command:

```
#killall swisdk
```

TS-7800 SBC with SD card

Note: The TS-7800 SBC boots from the SD card if JP1 is installed; otherwise, it boots from onboard flash. It then executes Linux from the SD card. The boot sequences from the SD card and from onboard flash are identical.

Note: The eclipse account is factory-preset on the SD card. Its home directory is '/home/eclipse'.

Note: The factory default eclipse account userid and password are 'eclipse'.

To download the SDK image to the TS-7800 SBC:

1. On the TS-7800 SBC (booted from external SD card), create a temporary directory for downloading images—this example uses /home/eclipse/testapi, but you can name the directory as you like. With the 2GB SD card Linux distribution, you can create a user account and download and execute images there.
2. On the Ubuntu 8.04 Linux PC, enter the following command from the LinuxSDK_a_b_c_d/pkgs/sdk directory:

```
$scp swisdk eclipse@192.168.0.50:testapi/
```


(This command copies into /home/eclipse/testapi.)
3. When prompted, enter the TS-7800 SBC eclipse password. When you enter the correct password, the image is copied onto the TS-7800 SBC.

To download the Application image to the TS-7800 SBC:

1. On the Ubuntu 8.04 Linux PC, enter the following command from the LinuxSDK_a_b_c_d/pkgs/ap directory:

```
$scp aptestarm9 eclipse@192.168.0.50:testapi/
```
2. When prompted, enter the TS-7800 SBC root password. When you enter the correct password, the image is copied onto the TS-7800 SBC.

To access the TS-7800 SBC:

1. On the Ubuntu 8.04 Linux PC, enter the following command:

```
$ssh eclipse@192.168.0.50
```
2. When prompted for a password, enter the eclipse account password ('eclipse').

Building images directly on the TS-7800 SBC

The TS-7800 Debian Sarge Linux platform comes equipped with vi (text editor), gdb (debugger), and gcc (compiler). With these tools, you can build SDK and API images (with source level debugging) on the TS-7800 SBC.

However, due to memory space limitations on embedded platforms, we recommend building sample code and utilities using cross-tools on a PC and then downloading only the resulting executables onto the TS-7800 SBC. (See “SDK Directory Tree Contents” in the *UMTS SDK Developer's Guide and CDMA SDK Developer's Guide* for details on available sample code and utilities.) If you don't need source level debugging for the SDK and API, we recommend you also build those on a PC using cross-tools.

If you want to use source level debugging on the TS-7800 SBC, you need to build the SDK and API images using the minimum required source code (the pkgs directory) on that platform as follows:

1. On the TS-7800 SBC, create a temporary directory for downloading images—this example uses /home/testapi, but you can name the directory as you like.
2. Copy the pkgs directory from the host to the TS-7800 SBC.

On the Linux PC:

- a. To avoid copying i386 executables to the TS-7800 SB, clear any executables by running `$make clean` from the pkgs, SampleCode, and Utilities directories.
- b. Enter the following command to copy the pkgs directory to the TS-7800 SBC (the ‘-r’ option indicates this is a recursive copy of the pkgs directory):

```
$scp -r pkgs/* root@192.168.0.50:/home/testapi/ 1
```

3. On the TS-7800 SBC, update the makefile (gen.mak) to use the local gcc compiler and remove references to the crosstool, as shown below:

Excerpt from gen.mak before changes

1	ifeq (\$(CPU),arm9)
2	CROSS_COMPILE=/usr/local/opt/crosstool/arm-
	linux/gcc-3.3.4-glibc-2.3.2/bin/arm-linux-
3	endif
4	
5	##\$(CROSS_COMPILE) is empty if not defined
	CC:=\$(CROSS_COMPILE)gcc
6	AR:=\$(CROSS_COMPILE)ar
7	RANLIB:=\$(CROSS_COMPILE)ranlib

Note: You could also update the gen.mak file on the Linux PC before copying the pkgs directory.

1. ssh and scp commands work only if the specified account has a non-empty password. If the ‘root’ password is empty, you can use the ‘eclipse’ account if it has root permissions, or set a password as described in [Assigning a root password](#) on page 34.

Excerpt from gen.mak after changes

```

1  ifeq ($(CPU),arm9)
2      CROSS_COMPILE =
3  endif
4
5  #$(CROSS_COMPILE) is empty if not defined
6  CC:=$(CROSS_COMPILE)gcc
7  AR:=$(CROSS_COMPILE)ar
8  RANLIB :=$(CROSS_COMPILE)ranlib

```

4. Build the source tree with the options “CPU=arm9 SYMBOLS=ON”:

```
#make -f pkgs.mak CPU=arm9 clean; make -f pkgs.mak CPU=arm9
SYMBOLS=ON; make -f pkgs.mak build
```

This produces the <SDK_ROOT>/build/bin/arm9/swisdk and <SDK_ROOT>/pkgs/ap/aptestarm9 executables, which are now ready to be executed on the TS-7800 SBC.

To start either executable with gdb:

1. Enter one of the following commands:

- #gdb ./swisdk
- #gdb ./aptestarm9

2. If you are running aptestarm9, you must start the swisdk executable from within the debugger. Enter the following command inside the gdb debugger:

```
set args -p ./swisdk -n t1
```

3. The application is now ready to run in the debugger. Use appropriate gdb commands to run the application.

Note: You must use the -p option to specify the path to the SDK executable. In this example, swisdk is in the same directory as aptestarm9.

Debian package management

When using the Debian Linux file system, new packages are added and obsolete packages are removed using Debian package management commands. For details on available commands, visit www.debian.org.

Miscellaneous TS-7800 SBC setup

This section describes procedures that facilitate troubleshooting/debugging.

Enabling logging

Log files are stored in /var/log/user.log for the Debian Sarge distribution on TS-7800 SBC. Logging has to be explicitly enabled on TS-7800 SBC (by default, logging is disabled).

To enable logging for troubleshooting during development:

1. On the TS-7800 SBC, go to the /etc directory.
2. Copy syslog.conf-debian to syslog.conf.
3. Inspect the /var/log directory for log files. If any appear, then logging is working.

Enabling coredump feature

The TS-7800 SBC comes equipped with gdb. The core dump feature, which is useful for troubleshooting (including analyzing crashes), is disabled by default.

To enable coredump generation:

1. On the TS-7800 SBC, enter the following command:

```
#ulimit -c unlimited
```

Restoring the TS-7800 SBC on-board flash

To restore the on-board flash on the TS-7800 SBC to factory configuration, use a bootable SD card, either supplied by Technologic Systems, or created as described below.

How to create a bootable SD card

Note: The SD card must be 512 MB or larger.

Note: This may be a very large file that could take several minutes to download.

To configure a new or existing external SD card's flash memory on the TS-7800 SBC to factory configuration, write a bootable dd image on the SD card:

1. On the Ubuntu 8.04 Linux PC, download the new SD card image:
 - a. Open a browser and go to ftp.embeddedarm.com/ts-arm-sbc/ts-7800-linux/binaries/ts-images.
 - b. Click the link labeled "512mbsd-latest.dd.bz2".
 - c. When prompted, save the file to disk.
 - d. Unzip the file to a temporary directory.

Note: This image does not include the Eclipse IDE that is distributed with 2Gb cards from Technologic Systems.

2. Insert the SD card to be configured into an SD card reader, and then connect the reader to your Linux PC.
3. If the SD card does not have a functioning filesystem, enter the following command to create a new filesystem:

```
$sudo mkfs.ext3 /dev/sdb
```

4. Copy the new image to the SD card—On Ubuntu Linux systems, enter the following command:

```
$sudo dd if=512mbsd-latest.dd of=/dev/sdc
```

where /dev/sdc is the device handle for the SD card, *not* just a partition on the card. When the command finishes, the SD card is ready for use.

How to restore the TS-7800 SBC's on-board flash

To restore the on-board flash using a bootable SD card:

1. Place the JP1 jumper on the TS-7800 SBC and reset the board. The board fast boots from the SD card.

2. On the TS-7800 SBC, enter the following commands to restore the TS-7800 512MB NAND flash to factory settings:
 # createmtdroot
 # createmtdboot



A: Establishing a Data Connection

A

On Linux systems, Sierra Wireless modems can connect to your network service provider's wireless network (dependent on the modem type and device drivers installed on the host computer) via:

- Wireless WAN connection—UMTS modems only
- PPP connection (Point-to-Point Protocol)—UMTS and CDMA modems

Note: In both cases, you must configure your modem as required by your network service provider to access their wireless network.

Wireless WAN connection

(UMTS modems only)

Note: Your host computer must have the following drivers installed: `sierra.c` and `sierra_net.c`. For instructions, see [Installing/updating Sierra Wireless Linux drivers](#) on page 16.

You can use either an API method or an AT command to establish a WAN connection between your host computer and a wireless network.

Once the modem establishes the connection, details (connection's IP address, DNS addresses, etc.) are passed to the host computer through the Direct IP driver (`sierra_net`). The host can then start using the connection to communicate with the wireless network.

WWAN connection procedure

Note: Both of these connection methods require you to use profile 1.

Configure profile 1 for use with your wireless network service provider's APN, and then use one of the following methods to start a WWAN connection:

- Connect using AT command
- Connect using API method

Connect/disconnect using AT command

Use the AT command `!SCACT` to establish a data connection (using profile 1) as follows:

1. In a terminal window, start `minicom` (or an equivalent program). For instructions on installing `minicom`, see [Using Minicom](#) on page 51

2. Type the following command to establish a data connection:

```
AT!SCACT=1,1
```

When the command prompt returns with a return code of 0 or OK, your host computer will detect the data connection and make it available for use by applications (browsers, email clients, etc.).

Note: To deactivate (tear down) the connection, use the command `AT!SCACT=0,1`.

Connect/disconnect using API method

The API method `SwiApiActivateProfile()` is used to establish (or tear down) a data connection.

The AP package includes sample code (`aptestgsmnet.c`) that demonstrates how to activate and deactivate a profile (connection) using the methods `doSwiActivateProfile()` and `doSwiDeActivateProfile()`.

These methods are also used by `aptesti386`, which you can use as follows to start a data connection using profile 1:

1. On the host computer, go to `<SDK_ROOT>/pkgs/ap` (where `<SDK_ROOT>` is the directory where you installed the SDK. See [Unpacking the distributed files](#) on page 27.)
2. To establish a WWAN connection, type:

```
./aptesti386 -p ../../build/bin/i386/swisdk -n t13
```

A message appears, indicating if the connection attempt is successful.

Note: To deactivate (tear down) the WWAN connection, type `./aptesti386 -p ../../build/bin/i386/swisdk -n t14`

3. To verify the connection, from a terminal window type the following to display available interfaces:

```
$ ifconfig
```

Output similar to the following appears:

```
usb0 Link encap:Ethernet HWaddr 00:00:00:00:00:00
inet addr:10.1.32.114 Bcast:10.1.32.255
Mask:255.255.255.0
inet6 addr: fe80::21e:37ff:fe21:9c80/64 Scope:Link
UP BROADCAST RUNNING MULTICAST
MTU:1500 Metric:1
RX packets:15736 errors:0 dropped:0 overruns:0
frame:0
TX packets:2708 errors:0 dropped:0 overruns:0
carrier:0
collisions:0 txqueuelen:1000
RX bytes:2322552 (2.2 MB) TX bytes:547571
(534.7 KB)
Interrupt:21
```

PPP connection

Note: Your host computer must have the sierra.c driver installed. For instructions, see [Installing/updating Sierra Wireless Linux drivers](#) on page 16.

On Linux computers, you connect to networks via Point-to-point protocol (PPP) using the programs `pppd` and `chat`. If either program is not installed, refer to your Linux distribution's documentation for installation instructions. (Both programs are included with Ubuntu Linux 8.04.)

Sierra Wireless provides scripts for establishing data connections using these programs.

To download these scripts:

1. Open a browser and go to http://sierrawireless.custhelp.com/app/answers/detail/a_id/500.
2. In the Downloads section, find and click the link to download `pppd` dialing scripts.
3. When prompted, save the scripts archive file to a temporary directory on your computer.
4. In a terminal window, go to the directory where you saved the file.
5. Extract the scripts to the default location by typing the following command:

```
$tar -zxf pppd-scripts.tar.gz
```
6. Switch to the root user and copy the files to the `ppp/peers` directory by typing the following commands:

```
$su  
$cp -r ./ppp /etc/  
$cd /etc/ppp  
$chmod a+x ip-up.local ip-down.local
```

To establish a data connection:

1. If you are using a GSM/UMTS modem, enter the following commands to set the authentication settings:

```
$cd /etc/ppp/peers  
$vi ./gsm_chat
```

 (You may use other editing programs such as `emacs` or `gedit` to edit the script.)
 - a. Go to the APN section and replace the listed APN with that of your service provider (for example, if your service provider is Cingular, you would type `isp.cingular`).
(Note: There are a few sample APN lines listed in the script that you can try.)
 - b. Save and exit.

Note: This example uses vi, however you can use any editing program.

2. Test the connection by entering the following command (you may need to use the root account to run pppd):

CDMA modems:

`$pppd call cdma`

GSM/UMTS modems:

`$pppd call gsm`

3. If the connection test fails

- a. Enter one of the following commands to edit the script file (further authentication may be required):

CDMA modems: `$vi ./cdma`

GSM/UMTS modems: `$vi ./gsm`

- b. In the editor:

- i. Put a '#' next to the "noauth" line (this disables the line).
- ii. Remove the '#' next to the user and password lines.
- iii. Type in the appropriate user name and password (you will need to contact your service provider if you do not know what these are).
- iv. Save and exit.

>> B: Using Minicom

B

Minicom is a Linux-based terminal emulation program similar to HyperTerminal (used on Windows-based computers) that is included on some Linux distributions.

This chapter describes how to identify the device handle you will use to send AT commands to the modem, and how to download, configure, and begin using minicom.

Identifying the AT commands tty device

Your Sierra Wireless modem exposes several ttys. To determine which tty is used for AT commands:

1. Open a terminal window and change to the <SDK_ROOT>/pkgs/ap directory (where <SDK_ROOT> is the directory where you installed the SDK. See [Unpacking the distributed files](#) on page 27.)
2. Run the “aptest” program using the following command:

```
./aptesti386 -p ../../build/bin/i386/swisdk -n t73
```

This command displays output similar to the following:

```
API Opened Successfully
Using SDK version <SDK Version Number>
Air Server not available - SLEEPING!
Register callback notification successful
Callback has been added to callback array
Invoking test: t73, SwiGetUsbPortName
Usb Port Name Response:
HIP: /dev/ttyUSB0
DIAG: /dev/ttyUSB1
NMEA: /dev/ttyUSB2
AT: /dev/ttyUSB3
DATA1: /dev/ttyUSB4
DATA2: /dev/ttyUSB5
DATA3: /dev/ttyUSB6
Direct IP:
QMI:
Mass Storage:
Note: Depending on modem and configuration, not all names
may be available.
Test Completed -----
```

3. Locate the output line labeled “AT:”. This is the tty you will use to enter AT commands.
In this example, the correct tty device is /dev/ttyUSB3.

Downloading and installing minicom

The following instructions describe how to download minicom (if it is not included with your distribution) and configure it for use with Sierra Wireless modems running on Linux systems.

To complete this procedure, your user account must be in the sudoers list or have root access privileges, and your system must have an active Internet connection:

1. Determine which device handle your modem uses for entering AT commands—see [Identifying the AT commands tty device](#) on page 51.
2. Determine whether minicom is already installed—Open a terminal window and enter the following command:

```
$minicom -s
```

If the program is installed, a menu appears entitled “configuration”. If you see this, skip to [Step 6](#).

3. If the program is not installed, download and install it. On Ubuntu Linux systems, enter the following command:

```
$sudo apt-get install minicom
```

4. When prompted, enter your own user password to begin the download and installation. When minicom is installed, the shell prompt appears.
5. Configure minicom to communicate with your modem:
 - a. Start minicom with the following command:

```
$minicom -s
```
 - b. Use the down arrow key to select the “Serial port setup” option.
 - c. Indicate the file handle to use for AT commands (from [Step 1](#))—Enter A and then replace the serial device string with the AT file handle (for example, /dev/ttyUSB3).
 - d. Press Enter twice, then use the down-arrow key to select Save setup as dfl.
 - e. Select exit.
6. If your modem is connected and running, then minicom proceeds to terminal mode and you should be able to enter AT commands to interact with it. If the modem is not connected, then minicom will exit.

On subsequent starts of minicom do not use the “-s” option—simply type minicom on the command line to start up terminal mode directly.

Note: On other Linux distributions, use whichever command is appropriate.

>> Index

Symbols

!ati, [20](#), [21](#)
!gstatus
 show modem details, [21](#)
<SDK_ROOT> defined, [27](#)

A

API functions
 calling, limitations, [29](#)
 reference, Doxygen, [30](#)
API-side
 design considerations, [29](#)
architectures, supported, [26](#)
ARM
 kernel, downloading, [36](#)
AT commands
 !ati, [20](#), [21](#)
 !gstatus, modem status details, [21](#)
 !pcinfo, [19](#)
 !pcstate, [19](#)
 !rssi, [19](#)
 CDMA modem, pre-installation, [19](#)
 UMTS modem, pre-installation, [20](#)
 UMTS, configure modem for data connections, [20](#)

B

bash shell, opening in terminal window, [14](#)
build
 SDK executables, [23](#)
build-essential
 installing, [15](#)

C

C programming language
 SDK, [26](#)
CDMA modem
 AT commands, pre-installation, [19](#)
 pre-installation requirements, [19](#)
 See also *modem*
compatibility with SDK
 hardware SDK, [26](#)
 software, [26](#)
connections, data
 establishing, [49](#)
core dump
 enabling, TS-7800 SBC, [45](#)
crosstool chains
 EABI crosstool, installing, [36](#)
 images, building, [40](#)
 OABI crosstool, installing, [35](#)
 overview, [34](#)

D

distribution, SDK
 unpacking, [27](#)
 verifying contents, [27](#)
Doxygen, API function/data reference, [30](#)
driver, sierra Linux
 definition, [16](#)
 installing, [16](#)
 version, displaying, [16](#)

E

EABI crosstool
 installing, [36](#)
executables
 SDK, building, [23](#)
 SDK, building for TS-7800 SBC, [36](#)
 SDK, overview, [28](#)

F

flash, onboard
 TS-7800 SBC, downloading images, [40](#)

H

hardware
 compatibility with SDK, [26](#)

I

images
 building on TS-7800 SBC, [43](#)
 building with crosstools, [40](#)
 downloading, TS-7800 SBC, [40](#)
install
 build-essential, [15](#)
 minicom, [15](#)
 sierra Linux driver, [16](#)
 Ubuntu Linux, [13](#)
IP address
 TS-7800 SBC, default, [33](#)

J

jumpers, TS-7800 SBC
 boot method, [31](#)
 COM1 console output, [31](#)
 CPU speed control, [31](#)
 descriptions, [31](#)

K

kernel
ARM, downloading, [36](#)

L

libraries
SDK, archive format, [27](#)
Linux
SDK executable, starting the, [24](#)
Live CD, Ubuntu Linux, [13](#)
Live Installation, Ubuntu Linux CD, [13](#)
logging
enabling, TS-7800 SBC, [44](#)

M

minicom
description, [51](#)
downloading and installing, [52](#)
installing, [15](#)
modem
firmware version, displaying, [25](#)
types supported, SDK, [26](#)
modinfo
display driver version, [16](#)

N

notifications
single thread, limitations, [29](#)

O

OABI crosstool
installing, [35](#)
object file format, SDK, [27](#)
on-board flash
restoring, TS-7800 SBC, [45](#)

P

password
Ubuntu user account, [15](#)
platforms
ARM9, configuration, [11](#)
supported, [11](#)
Ubuntu Linux PC, minimum configuration, [11](#)
PPP
connecting to network, [49](#)
pppd scripts, downloading, [49](#)

R

restore
on-board flash, TS-7800 SBC, [45](#)

S

scope of document, [11](#)
SD card
TS-7800 SBC, downloading images, [42](#)
SDK
API-side design considerations, [29](#)
distribution, unpacking, [27](#)
distribution, verifying contents, [27](#)
executable images, overview, [28](#)
executables, building, [23](#)
functions, C programming language, [26](#)
libraries, archive format, [27](#)
Linux executable, starting the, [24](#)
modems, types supported, [26](#)
object files, format, [27](#)
<SDK_ROOT> defined, [27](#)
sierra Linux driver
installing, [16](#)
single board computer, See *TS-7800 SBC*
software
compatibility with SDK, [26](#)
sudo
build-essential, installing, [15](#)
definition, [14](#)
minicom, installing, [15](#)
system resources, Linux commands to display, [15](#)

T

terminal window
process for opening, [14](#)
TS-7800 SBC
boot method, JP1 control, [31](#)
building images directly, [43](#)
COM1 console output, JP2 control, [31](#)
core dump, enabling, [45](#)
CPU speed, JP3 control, [31](#)
description, functional, [31](#)
host PC communication
Ethernet, [33](#)
preparing, [32](#)
images, downloading, [40](#)
IP address, default, [33](#)
logging, enabling, [44](#)
on-board flash, restoring, [45](#)
SDK executables, building, [36](#)
USB driver version, [37](#)
USB driver version, download, [38](#)
USB driver version, install, [38](#)

U

Ubuntu Linux

installation requirements, [13](#)installing, [13–14](#)password, user account, [15](#)See also *Linux*

UMTS modem

AT commands, pre-installation, [20](#)pre-installation requirements, [20](#)See also *modem*

USB driver, TS-7800 SBC

download, [38](#)install, [38](#)version, [37](#)

