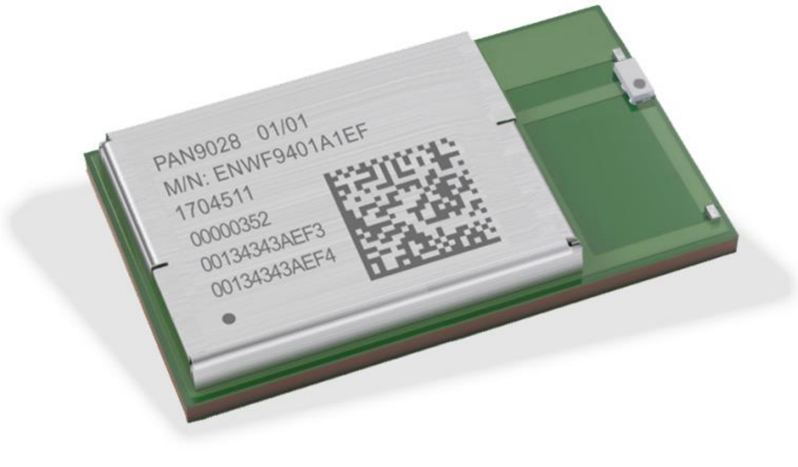
Panasonic
INDUSTRY

PAN9028

Wi-Fi Dual Band 2.4 GHz/5 GHz and Bluetooth® Module

Software Guide

Rev. 1.0



Wireless Connectivity

Table of Contents

1 About This Document

1.1 Purpose and Audience

This Software Guide describes the basic steps required for compiling and installing the software drivers, bringing up and configuring the PAN9028 on a standard Linux® distribution. It further describes the basic usage of the Wireless Utilities and gives an introduction to the command structure.

The document is intended for software engineers.

The product is referred to as “the PAN9028” or “the module” within this document.

1.2 Revision History

Revision	Date	Modifications/Remarks
1.0	2021-07-27	First version

1.3 Use of Symbols

Symbol	Description
ⓘ	**Note** Indicates important information for the proper use of the product. Non-observance can lead to errors.
⚠	**Attention** Indicates important notes that, if not observed, can put the product’s functionality at risk.
✎	**Tip** Indicates useful information designed to facilitate working with the software.
➩ [chapter number] [chapter title]	**Cross reference** Indicates cross references within the document. **Example:** Description of the symbols used in this document ➩ 1.3 Use of Symbols.
✓	**Requirement** Indicates a requirement that must be met before the corresponding tasks can be completed.
➔	**Result** Indicates the result of a task or the result of a series of tasks.
This font	**GUI text** Indicates fixed terms and text of the graphical user interface. **Example:** Click **Save**.

Symbol	Description
Menu > **Menu item**	**Path** Indicates a path, e.g. to access a dialog. **Example:** In the menu, select **File** > **Setup page**.
`This font`	**File names, userinput** Indicates file names or messages and information displayed on the screen or to be selected or entered by the user. **Examples:** `pan1760.c` contains the actual module initialization. Enter the value `Product 123`.
`This font`	**Systemoutput** Indicates messages and information displayed on the screen. **Example:** The message `Failed to save your data` is displayed.
Key	**Key** Indicates a key on the keyboard. **Example:** Press **F10**.

1.4 Related Documents

For related documents please refer to the Panasonic website ⇨ 11.2 Product Information.

2 Overview

The PAN9028 is a dual band 2.4 GHz/5 GHz 802.11 a/b/g/n/ac Wi-Fi radio module with integrated Bluetooth BR/EDR/Low Energy, specifically designed for highly integrated and cost-effective applications. The simultaneous and independent operation of the two standards enables very high data rates (802.11ac, 80 MHz) and low-power operation (Bluetooth Low Energy).

A Linux-based installation is chosen as the software platform. The Linux kernel provides an established environment for running the Wi-Fi and Bluetooth drivers and the available applications make it possible to use the PAN9028 module to its full extend.

Please refer to the Panasonic website for related documents ⇨ 11.2 Product Information.

3 Software

The PAN9028 module is based on the NXP® chipset 88W8987 supporting 802.11a/b/g/n/ac simultaneous station, access point, and Wi-Fi Direct operations. The module is connected through the SDIO device interface to the host processor platform running a Linux operating system. NXP usually provides software package releases with the driver sources and firmware binaries for a reference platform and operating system version (e.g. x86 platforms and Ubuntu® 16). The driver sources can be easily ported to comparable platforms (e.g. ARM® Cortex®-A) using different Linux distributions (e.g. Yocto Linux).

3.1 Linux Kernel and User Space Architecture

The Linux system architecture is divided into the kernel space and user space.

Kernel Space

The kernel runs in the dedicated part of memory. The kernel space manages applications and processes running in user space. The kernel space can access all the memory. If a process performs a system call, a software interrupt is sent to kernel which dispatches an appropriate interrupt handler.

User Space

The user space is set of locations where normal user processes run. These processes cannot access kernel space directly. Some part of kernel space can be accessed via system calls. These system calls act as software interrupts in kernel space.

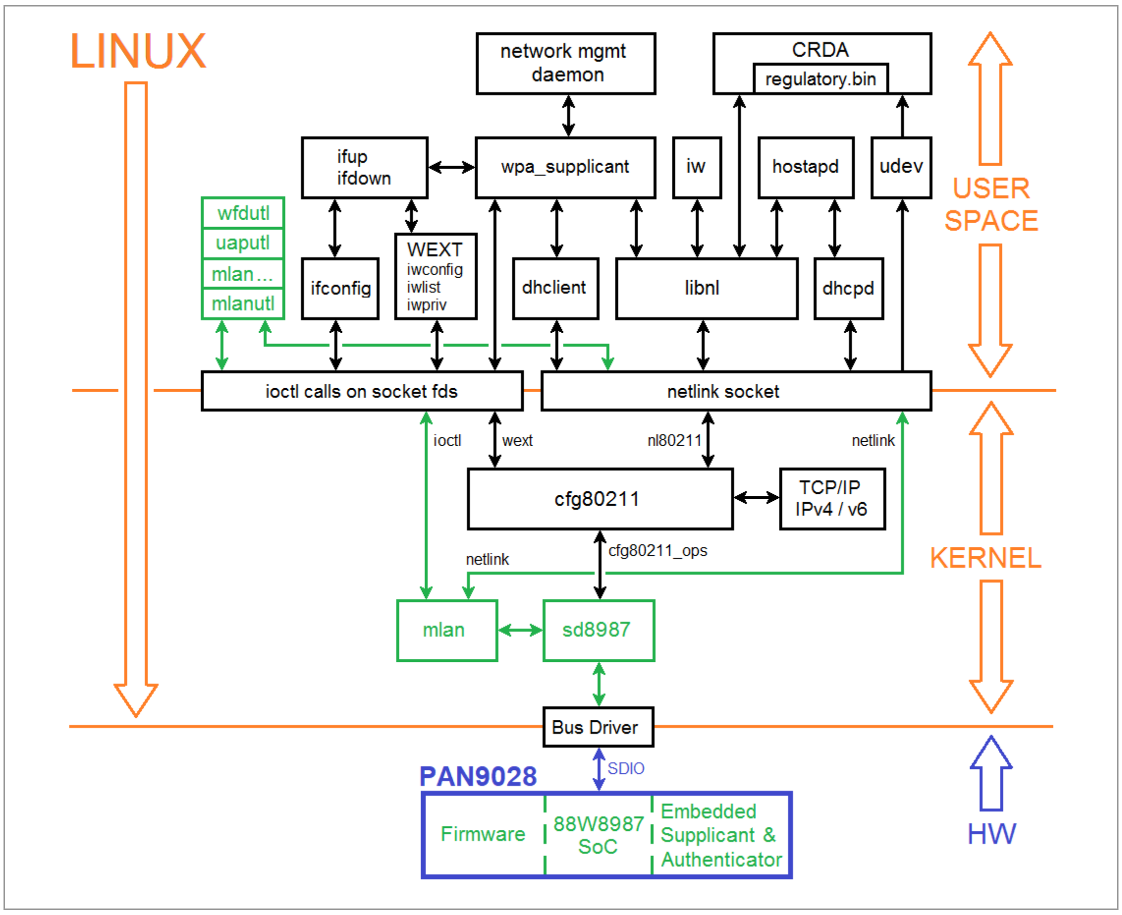
3.2 Linux Wireless Subsystem

The Linux Wireless (IEEE802.11) subsystem consists of the 802.11 drivers, the core mac80211, and cfg80211 components along with the user space and in-kernel nl80211 configuration interface.

The cfg80211 kernel module is the Linux 802.11 configuration API. It replaces the Wireless Extensions (WExt). The nl80211 is used to configure a cfg80211 device and is used for kernel to user space communication. When implementing a cfg80211 driver, wireless extensions support can still be provided through cfg80211 and by enabling WExt support. The cfg80211 also provides full regulatory support; this is done through “wireless-regdb” and the usage of Central Regulatory Domain Agent (CRDA). It is a user space agent which uploads regulatory domains into the kernel, and it acts as an “udev” helper. All Linux wireless drivers shall be written targeting either cfg80211 for FullMAC devices or mac80211 for SoftMAC devices.

3.3 NXP Drivers and Tools in the Wireless Subsystem

The PAN9028 module requires a kernel driver located on the host system and a firmware running on the module's 88W8987 System-on-Chip. The firmware image is downloaded to the module if the bus driver detects the module's SDIO interface. The NXP kernel driver (sd8987) is placed between the bus driver and the attached network stack from the cfg80211 kernel module which is part of the Linux Wireless (IEEE802.11) subsystem. Additionally, NXP provides various control and configuration tools which are located in the Linux user space. These tools are using either ioctl (input-output control) calls or the netlink socket.

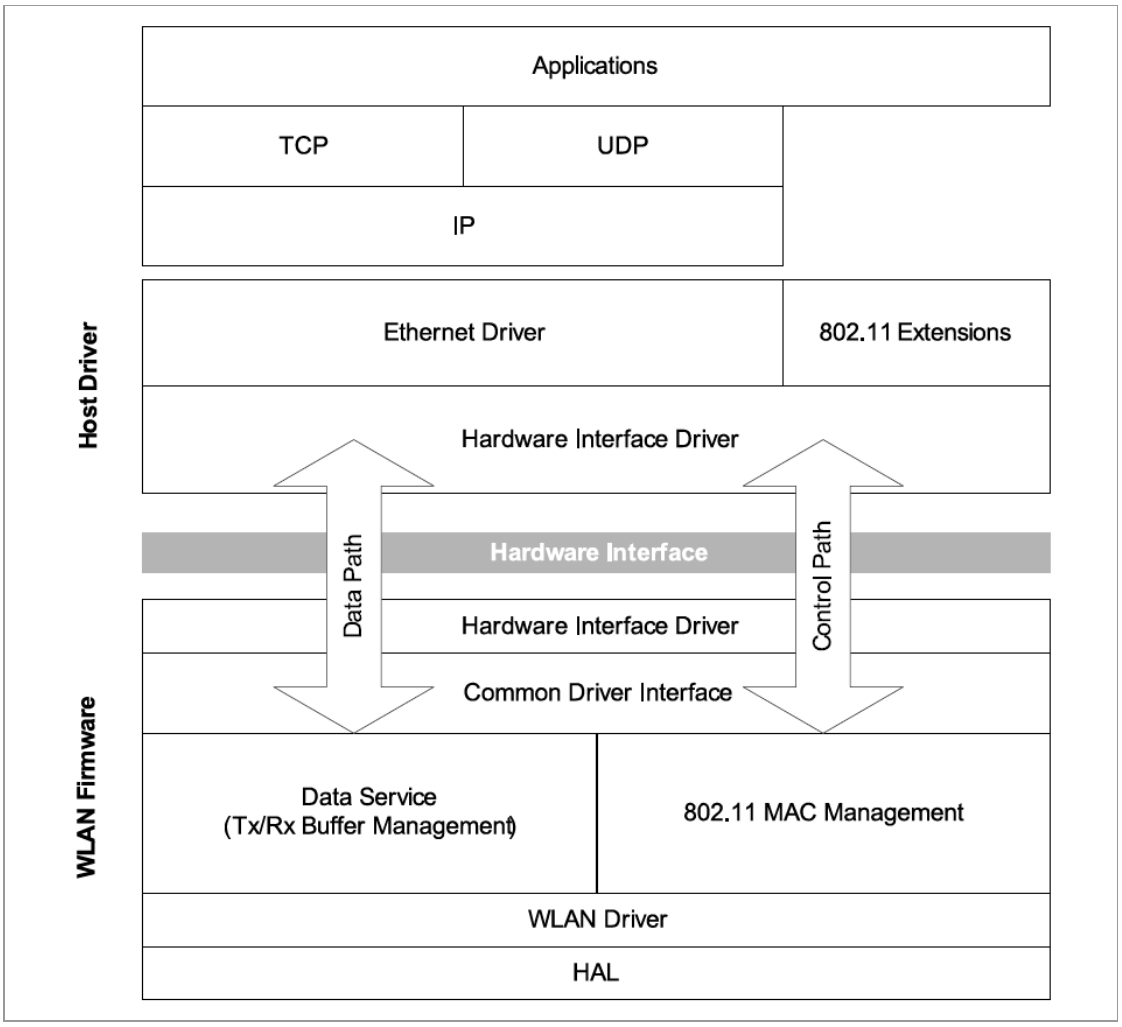


3.4 Driver and Firmware Architecture

The NXP drivers are inserted into the kernel of the Linux host operating system at runtime in the form of kernel modules. The WLAN device firmware image is stored on the Linux host operating system as well, but it is downloaded through the bus driver interface to the module during loading of host driver. After hardware initialization the firmware is running on the module and network devices are registered on the Linux host operating system.

Architecture of the Host Driver and WLAN Device Firmware

The firmware is typical of a thick firmware architecture, where it handles all 802.11 MAC management tasks. The host driver downloads standard 802.3 frames to the firmware, to transmit over the wireless link as 802.11 frames. Furthermore, the WLAN firmware processes the received 802.11 frames and converts them into 802.3 frames before forwarding to the host driver.



Installation Steps by Host Driver

1. The host driver registers the kernel module with the SDIO Bus driver (mmc device).
2. After successful registration: the bus driver detects the module hardware and calls the driver’s probe handler.
3. The probe handler allocating and initializing internal structures registers the interrupt service and starts the driver threads.
4. The firmware image is downloaded to the module.
5. After successful firmware download: the module hardware is initialized.
6. Ethernet network devices (mlan0, uap0, wfd0, nan0) are created based on driver mode.

4 Preparing the Software Drivers

4.1 Host Communication Interfaces

The PAN9028 has two host communication interfaces: SDIO device interface and universal asynchronous receiver/transmitter (UART) interface.

SDIO Device Interface

This interface conforms to the industry standard SDIO full-speed card specification and allows a host controller, using the SDIO bus protocol to access the device. The device supports SDIO 3.0 Standard with 1-bit SDIO or 4-bit SDIO transfer modes with clock range up to 208 MHz in SDR104 mode.

The SDIO interface is used for the communication between WLAN kernel driver (sd8987) and firmware. It is also used for uploading the firmware during the WLAN kernel driver loading. This interface can also be used for the Bluetooth HCI communication between Bluetooth kernel driver (bt8987, mbt8987) and firmware. In this case it is a time-shared interface for WLAN and Bluetooth communication.

Universal Asynchronous Receiver/Transmitter (UART) Interface

This high-speed interface compliant with the “industry standard 16550 specification”. The UART interface can be separately used for the Bluetooth HCI communication between Bluetooth kernel driver (bt8987, mbt8987) and firmware. It is offloading the SDIO and internal bus traffic for WLAN processing.

4.2 Downloading the Software Package

The NXP chipset 88W8987 used in PAN9028, is not supported in Linux mainline kernel. It is necessary to download the software package from the NXP website. Please contact your local Panasonic Sales office for creating a customer account to get access to the NXP download area ⇨ 11.1 Contact Us.

The following requirement must be met:

✓ NXP customer account

Download the software package (with the NXP driver sources and firmware binaries) from the NXP website https://www.nxp.com/.

Software Package Versions

For chipset 88W8987 the following software package versions are available at the NXP website, which are belonging to the host communication interface and software license type:

Software Package Version	WLAN Interface	Bluetooth Interface	Software License
`SD-WLAN-SD-BT-8987-…MGPL.zip`	SDIO	SDIO	MGPL (full)
`SD-WLAN-UART-BT-8987-…-MGPL.zip`	SDIO	UART	MGPL (full)
`SD-WLAN-SD-BT-8987-…-GPL.zip`	SDIO	SDIO	GPL
`SD-WLAN-UART-BT-8987-…-GPL.zip`	SDIO	UART	GPL

Software License Type

- **GPL Release**: The module “moal” is licensed under the GPL, the module “mlan” and “mapps” sample applications are NXP Proprietary Code.
- **MGPL Release**: The modules “moal” and “mlan”, and “mapps” sample applications are all licensed under the GPL.

Software Package Release Version Information (Example)

Example:
`SD-WLAN-SD-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-C4X16651-MGPL`

Type	Convention	Description
WLAN Interface Prefix	SD-WLAN	SDIO interface for WLAN device communication
Bluetooth Interface Prefix	SD-BT (UART-BT)	SDIO or UART interface for Bluetooth HCI communication
NXP SoC Version	8987	Chipset 88W8987 (PAN9028)
Linux Distribution	U16	Ubuntu 16 test environment
Interface Type	MMC	Multimedia card
WLAN Firmware Version	W16.68.10.p33	WLAN firmware: `sd8987_wlan.bin`
	W	Indicates the release includes the WPA/WPA2 VU, WPA Replay Protection Fixes required for WFA Security Detection/Certification testing purpose and memory copy vulnerability fix.
	16	Major revision (first number from the left): Tracks the main firmware version
	68	Minor revision (second number from the left): Tracks the chip family, firmware branch, custom projects, etc.
	10	Release number (third number from the left): Tracks the incremental changes in the consequent firmware releases given to Quality Assurance (QA) or customers.
	p33	Patch number (fourth number from the left): The patch number starts at zero (no patch) and increments with release of subsequent builds with bug fixes.

Type	Convention	Description
Bluetooth Firmware Version	16.26.10.p33	Bluetooth firmware: `sd8987_bt.bin`, `uart8987_bt.bin`
	16	Major revision (first number from the left)
	26	Minor revision (second number from the left)
	10	Release number (third number from the left)
	p33	Patch number (fourth number from the left)
Driver Package Version	C4X16651	WLAN driver: `mlan.ko`, `sd8987.ko` SD-BT driver: `bt8987.ko` (BlueZ), `mbt8987.ko` (HCI) UART-BT driver: `hci_uart.ko`
	C	Indicates NXP WLAN/Bluetooth combo driver: `sdsd8987_combo.bin`, `sduart8987_combo.bin`
	4X	Indicates the version of Linux kernel used to release the NXP driver.
	16	Release number: Tracks the incremental changes in the consequent driver releases given to QA.
	651	Patch Number: The patch number starts at zero (no patch) and increments with release of subsequent builds with more bug fixes.
Software License Type	MGPL	NXP full GPL code (“moal”,” mlan”, and “mapps”)

Supported Linux Kernel Version

Linux Kernel Version	From	To	FW Version
Software Package Release	2.6.32	5.5.2	W16.87.10.p134-16.136.10.p134

4.3 Unpacking the Software Package

The following requirement must be met:

✓ The software package is downloaded ⇨ 4.2 Downloading the Software Package.

1. Create a working directory for the software package ZIP archive:

```
$> mkdir -p /tmp/pan9028
$> cd /tmp/pan9028
$> TOP=$(pwd)
$> cd ${TOP}
```

➔ In the directory `tmp` a working directory `pan9028` is created.

➔ After having changed to the subdirectory `pan9028`, a variable `TOP` is set as actual working directory.

2. Copy the software package ZIP archive to working directory:

```
$> cp [path-to-ZIP]/ SD-WLAN-SD-BT-8987-U16-MMC-W16.68.10.p33-
16.26.10.p33-C4X16651-MGPL.zip ${TOP}
```

3. Unzip the software package ZIP archive inside working directory:

```
$> cd ${TOP}
$> unzip SD-WLAN-SD-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-
C4X16651-MGPL.zip -d p33
Archive:  SD-WLAN-SD-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-
C4X16651-MGPL.zip
  inflating: p33/SD-WLAN-SD-BT-8987-U16-MMC-W16.68.10.p33-
16.26.10.p33-C4X16651-MGPL.tar
  inflating: p33/SD-WLAN-SD-BT-8987-U16-MMC-W16.68.10.p33-
16.26.10.p33-C4X16651-MGPL-Release Notes.pdf
```

➔ A subdirectory with the name of the software patch is created. The release notes and a TAR file is inflated inside the subdirectory.

4. Change the directory to the software patch subdirectory and uncompressing the TAR files:

```
$> cd ${TOP}/p33
$> ls
SD-WLAN-SD-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-C4X16651-MGPL-
Release Notes.pdf
SD-WLAN-SD-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-C4X16651-
MGPL.tar
$> tar xvf SD-WLAN-SD-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-
C4X16651-MGPL.tar
SD-BT-8987-U16-MMC-16.26.10.p33-C4X14113-GPL-src.tgz
SD-BT-CHAR-8987-U16-MMC-16.26.10.p33-C4X14113-GPL-src.tgz
SD-UAPSTA-8987-U16-MMC-W16.68.10.p33-C4X16651-MGPL-src.tgz
SD-UAPSTA-8987-U16-MMC-W16.68.10.p33-C4X16651-app-src.tgz
SD-UAPSTA-8987-U16-MMC-W16.68.10.p33-C4X16651-mlan-src.tgz
FwImage/sd8987_ble.bin
FwImage/sd8987_bt.bin
FwImage/sd8987_wlan.bin
FwImage/sdsd8987_combo.bin
```

➔ Five TGZ files are inflated in the directory.

➔ An additional subdirectory with the name `FwImage` is created.

5. Change the directory to the software patch subdirectory and uncompressing the TGZ files:

```
$> cd ${TOP}/p33
$> tar xvfz SD-BT-8987-U16-MMC-16.26.10.p33-C4X14113-GPL-src.tgz
$> tar xvfz SD-BT-CHAR-8987-U16-MMC-16.26.10.p33-C4X14113-GPL-src.tgz
$> tar xvfz SD-UAPSTA-8987-U16-MMC-16.68.10.p33-C4X16651-app-src.tgz
$> tar xvfz SD-UAPSTA-8987-U16-MMC-16.68.10.p33-C4X16651-MGPL-src.tgz
$> tar xvfz SD-UAPSTA-8987-U16-MMC-16.68.10.p33-C4X16625-mlan-src.tgz
```

➔ An additional subdirectory with the name of the software driver package is created.

4.4 Copying the Firmware

The following requirement must be met:

✓ The software package is unpacked ⇨ 4.3 Unpacking the Software Package.

Linux Firmware Library

The PAN9028 requires a System-on-Chip firmware to be uploaded to the NXP chipset “88W8987”. The kernel modules will be configured to load the NXP firmware files from the firmware path `/lib/firmware/nxp`.

4.4.1 Copying the Firmware (SD-WLAN-SD-BT)

Change the directory to the subdirectory `FwImage` and copy firmware binaries to firmware library:

```
$> cd ${TOP}/p33/FwImage
$> ls
sd8987_ble.bin
sd8987_bt.bin
sd8987_wlan.bin
sd8987_combo.bin
$> sudo cp sd8987_ble.bin /lib/firmware/nxp
$> sudo cp sd8987_bt.bin /lib/firmware/nxp
$> sudo cp sd8987_wlan.bin /lib/firmware/nxp
$> sudo cp sd8987_combo.bin /lib/firmware/nxp
```

➔ The System-on-Chip firmware is copied to the NXP vendor specific firmware library.

4.4.2 Copying the Firmware (SD-WLAN-UART-BT)

Change the directory to the subdirectory `FwImage` and copy firmware binaries to firmware library:

```
$> cd ${TOP}/p33/FwImage
$> ls
sd8987_wlan.bin
sduart8987_combo.bin
uart8987_bt.bin
$> sudo cp sd8987_wlan.bin /lib/firmware/nxp
$> sudo cp sduart8987_combo.bin /lib/firmware/nxp
$> sudo cp uart8987_bt.bin /lib/firmware/nxp
```

➔ The System-on-Chip firmware is copied to the NXP vendor specific firmware library.

5 Compiling the Drivers

Please note that the NXP drivers cannot be compiled statically into the kernel.

The following requirement must be met:

✓ The target system contains all required functionality to compile out-of-tree kernel modules.

5.1 Cross-Compilation

For cross-compiling the driver sources, it is necessary to specify the target architecture, the cross-toolchain, and the directory.

Use the driver sources to build the kernel modules for the target system:

```
$> make CROSS_COMPILE=${CROSS_COMPILE} KERNELDIR=${KERNEL_DIR}
ARCH=${ARCH} build
```

➔ Now the driver sources can be used for cross-compilation.

Definition of Variables

`ARCH` = arm
`CROSS_COMPILE` = <path to cross-compiler>
`KERNEL_DIR` = <path to kernel sources>

5.2 Compiling the Bluetooth Driver Sources

5.2.1 Building Result of Driver Sources

The following kernel modules and utilities (configuration tools) will be generated or copied by compiling the driver sources:

File Name	Description
bt8987.ko	Bluetooth HCI kernel driver (hci0 device)
mbt8987.ko	Bluetooth character kernel driver (mbtchar0 device)
fmapp	Utility to configure Bluetooth connections, send HCI, or vendor specific commands.
load	Bluetooth driver load shell script
unload	Bluetooth driver unload shell script
README	Readme files as user manual for the utilities with instructions and examples
config/*	Subdirectory for various sample configuration files used by utilities

5.2.2 Compiling the Bluetooth Character Driver (SD-BT)

Change the directory to the Bluetooth character driver source subdirectory:

```
$> cd ${TOP}/p33/SD-UAPSTA-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-
C4X16651-MGPL
$> ls
mbtc_src  mbt_src  wlan_src
$> cd mbtc_src
$> ls
app  config  README  bt  Makefile  script
$> make clean && make build
find . -name "*.o" -exec rm {} \;
find . -name "*.ko" -exec rm {} \;
find . -name ".*.cmd" -exec rm {} \;
find . -name "*.mod.c" -exec rm {} \;
find . -name "*.symvers" -exec rm {} \;
find . -name "modules.order" -exec rm {} \;
find . -name ".*.dwo" -exec rm {} \;
find . -name "*dwo" -exec rm {} \;
rm -rf .tmp_versions
make -C app/fm_app clean
…
cp -f mbt8xxx.ko ../bin_sd8987_btchar/mbt8987.ko
cp -r config ../bin_sd8987_btchar
cp -f script/sdio_mmc/* ../bin_sd8987_btchar/
cp -f README ../bin_sd8987_btchar
make -C app/fm_app build INSTALLDIR=../bin_sd8987_btchar;
…
cp -f app/fm_app/fmapp ../bin_sd8987_btchar;
$> ls
app  config    mbt8xxx.ko     mbt8xxx.mod.o  modules.order   README
bt   Makefile  mbt8xxx.mod.c  mbt8xxx.o      Module.symvers  script
$> cd ${TOP}/p33/SD-UAPSTA-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-
C4X16651-MGPL
$> ls
bin_sd8987_btchar  mbtc_src  mbt_src  wlan_src
```

➔ An additional subdirectory is created. The driver and utility binaries can be found in the directory `../bin_sd8xxx_btchar`.

5.2.3 Compiling the Bluetooth HCI Driver (SD-BT)

Change the directory to the Bluetooth driver source subdirectory:

```
$> cd ${TOP}/p33/SD-UAPSTA-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-
C4X16651-MGPL
$> ls
bin_sd8987_btchar  mbtc_src  mbt_src  wlan_src
$> cd mbt_src
$> ls
app  bt  config  Makefile  README  script
$> make clean && make build
find . -name "*.o" -exec rm {} \;
find . -name "*.ko" -exec rm {} \;
find . -name ".*.cmd" -exec rm {} \;
find . -name "*.mod.c" -exec rm {} \;
find . -name "*.symvers" -exec rm {} \;
find . -name "modules.order" -exec rm {} \;
find . -name ".*.dwo" -exec rm {} \;
find . -name "*dwo" -exec rm {} \;
rm -rf .tmp_versions
make -C app/fm_app clean
make[1]: Entering directory …
…
cp -f bt8xxx.ko ../bin_sd8987_bt/bt8987.ko
cp -r config ../bin_sd8987_bt
cp -f script/sdio_mmc/* ../bin_sd8987_bt/
cp -f README ../bin_sd8987_bt
make -C app/fm_app build INSTALLDIR=../bin_sd8987_bt;
…
cp -f app/fm_app/fmapp ../bin_sd8987_bt;
$> ls
app  bt8xxx.ko     bt8xxx.mod.o   config     modules.order    README
bt   bt8xxx.mod.c  mbt8xxx.o      Makefile   Module.symvers   script
$> cd ${TOP}/p33/SD-UAPSTA-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-
C4X16651-MGPL
$> ls
bin_sd8987_bt  bin_sd8987_btchar  mbtc_src  mbt_src  wlan_src
```

➔ An additional subdirectory is created. The driver and utility binaries can be found in the directory `../bin_sd8xxx_bt`.

5.2.4 Compiling the Bluetooth HCI Driver (UART-BT)

> ⚠ The software package Version “SD-WLAN-UART-BT-8987-…” is released separately. The source code and firmware binaries are different ⇨ Software Package Versions.

Change the directory to the Bluetooth driver source subdirectory:

```
$> cd ${TOP}/p134/SD-UAPSTA-BT-UART-8987-U16-MMC-W16.87.10.p134-
16.87.10.p134-C4X16687-MGPL
$> ls
muart_src
$> cd muart_src
$> ls
bt_drv.h     hci_ldisc.c  hci_uart.h     include     mbt_char.h
hci_bcsp.c   hci_ll.c     hci_wrapper.c  Makefile    modules.order
hci_h4.c     hci_ps.c     hci_wrapper.h  mbt_char.c  README
$> make clean && make build
find . -name "*.o" -exec rm {} \;
find . -name "*.*~" -exec rm {} \;
find . -name "*.d" -exec rm {} \;
find . -name "*.mod.c" -exec rm {} \;
find . -name "*.ko" -exec rm {} \;
find . -name "*.symvers" -exec rm {} \;
find . -name ".*.cmd" -exec rm {} \;
rm -rf .tmp_versions
make -C …
make[1]: Entering directory …
…
$> ls
bt_drv.h     hci_ldisc.c  hci_ps.o        hci_uart.o     Makefile
hci_bcsp.c   hci_ldisc.o  hci_uart.h      hci_wrapper.c  mbt_char.c
README       hci_ll.c     hci_uart.ko     hci_wrapper.h  mbt_char.h
hci_h4.c     mbt_char.o   hci_uart.mod.c  hci_wrapper.o Module.sysmvers
hci_h4.o     hci_ps.c     hci_uart.mod.o  include        modules.order
$> cd ${TOP}/p134/SD-UAPSTA-BT-UART-8987-U16-MMC-W16.87.10.p134-
16.87.10.p134-C4X16687-MGPL
$> ls
bin_muart  muart_src
```

➔ An additional subdirectory is created. The driver and utility binaries can be found in the directory `../bin_muart`.

5.3 Compiling the WLAN Driver Sources

5.3.1 Building Result of Driver Sources

The following kernel modules and utilities (configuration tools) will be generated or copied by compiling the WLAN driver sources:

File Name	Description
mlan.ko	Kernel driver for event handling through the netlink layer and ioctl calls
sd8987.ko	WLAN device kernel driver (mlan0, uap0, wfd0, nan0 device)
mlanutl	Utility for configuration additional parameters available for mdriver
mlanevent.exe	Utility to listen for and obtain events from the driver through the netlink layer
mlan2040coex	Application handles the 802.11n 20/40 coexistence operation for mdriver
uaputl.exe	Utility to get and set uAP's settings
load	WLAN driver load shell script
unload	WLAN driver unload shell script
README*	Readme files as user manual for the utilities with instructions and examples
config/*	Subdirectory for various sample configuration files used by utilities

5.3.2 Compiling the WLAN Driver

Change the directory to the WLAN driver source subdirectory:

```
$> cd ${TOP}/p33/SD-UAPSTA-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-
C4X16651-MGPL
$> ls
bin_sd8987_bt  bin_sd8987_btchar  mbtc_src  mbt_src  wlan_src
$> cd wlan_src
$> ls
gpl-2.0.txt  mapp  mlinux  README_MLAN     README_UAP          script
Makefile     mlan  README  README_OPENWRT  README_WIFIDIRECT
$> make clean && make build
find . -name "*.o" -exec rm {} \;
find . -name "*.ko" -exec rm {} \;
find . -name ".*.cmd" -exec rm {} \;
find . -name "*.mod.c" -exec rm {} \;
find . -name "Module.symvers" -exec rm {} \;
find . -name "Module.markers" -exec rm {} \;
find . -name "modules.order" -exec rm {} \;
find . -name ".*.dwo" -exec rm {} \;
rm -rf .tmp_versions
make -C mapp/mlanconfig clean
…
```

```
cp -rpf script/wifidirect ../bin_sd8987
cp -rpf script/wifidisplay ../bin_sd8987
make -C mapp/wifidirectutl build INSTALLDIR=../bin_sd8987
…
make -C mapp/mlanevent build INSTALLDIR=../bin_sd8987
…
$> ls
gpl-2.0.txt  mlan   mlan.mod.o  modules.order   README_MLAN
README_WIFIDIRECT  sd8xxx.mod.c Makefile  mlan.ko  mlan.o
Module.symvers  README_OPENWRT  script  sd8xxx.mod.o  mapp
mlan.mod.c  mlinux  README  README_UAP  sd8xxx.ko  sd8xxx.o
$> cd ${TOP}/p33/SD-UAPSTA-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-
C4X16651-MGPL
$> ls
bin_sd8987  bin_sd8987_bt  bin_sd8987_btchar  mbtc_src  mbt_src
wlan_src
```

➔ An additional subdirectory is created. The driver and utility binaries can be found in the directory `../bin_sd8xxx`.

6 WLAN Device Driver

6.1 Installing the WLAN Device Driver Kernel and Firmware

All available WLAN driver options of NXP device kernel module are listed in the file `README` which is included in the software package.

Installing the WLAN Device Driver Kernel

Calibration Data Configuration Parameter (cal_data_cfg)

To get use of the calibration parameter which are pre-stored on the one-time-programmable memory of the PAN9028, the following driver option must be set:

```
cal_data_cfg=none
```

Dual-Rapid-Channel-Selection Parameter (**cfg80211_drcs)**

If the simultaneous and independent WLAN and Bluetooth operation is configured, it is necessary to disable the Dual-Rapid-Channel-Selection (DRCS) mechanism:

```
cfg80211_drcs=0
```

By enabling the Dual-Rapid-Channel-Selection mechanism it is not possible to use the Bluetooth interface, but it is possible to generate a second micro-AP by setting the max_uap_bss parameter:

```
cfg80211_drcs=1 max_uap_bss=2
```

1. Change the directory to the compiled WLAN driver binary directory.
2. Use the Linux command **`insmod`** or **`modprobe`** to install the kernel driver:

```
$> cd ${TOP}/p33/SD-UAPSTA-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-
C4X16651-MGPL
$> ls
config          mlan.ko      README_UAP          unload
load            mlanutl      README_WIFIDIRECT   wifidirect
mlan2040coex    README       sd8987.ko           wifidirectutl
mlanevent.exe   README_MLAN  uaputl.exe          wifidisplay
$> insmod mlan.ko
$> insmod sd8987.ko drv_mode=3 fw_name=nxp/sdsd8987_combo.bin
   cal_data_cfg=none cfg80211_wext=0xf cfg80211_drcs=0
```

Kernel Driver Configuration

WLAN Driver Mode Option (drv_mode)

The parameter `drv_mode` shall be configured by following bit settings:

- Bit 0: STA
- Bit 1: uAP
- Bit 2: WIFIDIRECT
- Bit 4: NAN

To load driver in STA only mode, use: **`drv_mode=1`**

To load driver in UAP only mode, use: **`drv_mode=2`**

To load driver in STA+UAP mode, use: **`drv_mode=3`**

To load driver in STA+UAP+WFD+NAN mode, use: **`drv_mode=23 (0x17)`**

The default mode is STA+UAP+WFD: **`drv_mode=7`**

System-on-Chip Firmware Definition (fw_name)

In case of using the WLAN firmware with parallel download option, the Bluetooth firmware can be downloaded by installing the Bluetooth kernel module.

During loading of the WLAN device kernel module, the SoC firmware is downloaded to the PAN9028 module. The SoC firmware binaries are stored at the Linux vendor specific library path `/lib/firmware/nxp`.

To define the SoC firmware binary file, use:

```
fw_name
```

To specify the WLAN/Bluetooth combo firmware, use:

```
fw_name=nxp/sdsd8987_combo.bin
```

To specify the WLAN firmware (without Bluetooth firmware part) for parallel download, use:

```
fw_name=nxp/sd8987_wlan.bin fw_serial=0
```

WLAN Network Link Configuration (cfg80211_wext)

The WLAN device kernel module is linked through the Linux cfg80211 kernel module and the nl80211 network link layer to the Linux user space. In parallel to the “nl80211” based user space applications such as `wpa_supplicant` or `hostapd`, it is possible to use the Wireless Extensions (WExt) toolchain (e.g. iwconfig, iwlist, etc.).

To configure the driver parameter, use:

```
cfg80211_wext=0xf
```

WLAN Enhanced Power Management Features (auto_ds, ps_mode)

The power management features of WLAN driver are set by default to MLAN firmware default (**`auto_ds=0`**, **`ps_mode=0`**). For particular WLAN use cases it is recommended and sometimes necessary to disable power management features (such as WLAN deep sleep or 802.11 PSM mode) beacuse time critical events from host driver need to wake up the WLAN device in time.

To configure the power management options of driver (which can be disabled with driver loading), use:

```
auto_ds=2 ps_mode=2
```

6.2 Loading the WLAN Device Driver

The firmware image is downloaded to the module if the bus driver detects the module’s SDIO interface.

➔ After the hardware interface has been detected, the following kernel messages will be shown:

```
kernel: mmc0: new high speed SDIO card at address 0001
kernel: vendor=0x02DF device=0x9149 class=0 function=1
kernel: SDIO: max_segs=128 max_seg_size=65535
kernel: rx_work=1 cpu_num=2
kernel: wlan: Enable TX SG mode
kernel: wlan: Enable RX SG mode
kernel: Request firmware: nxp/sdsd8987_combo.bin
kernel: Wlan: FW download over, firmwarelen=635912 downloaded 635912
kernel: WLAN FW is active
kernel: fw_cap_info=0x187ccf03, dev_cap_mask=0xffffffff
kernel: max_p2p_conn = 8, max_sta_conn = 8
```

➔ After hardware initialization the firmware is running on the module and network, devices are registered on the Linux host system:

Interface	Description
mlan0	WLAN Station/client mode
uap0	WLAN micro access point mode (uAP)
wfd0	Wi-Fi Direct™ mode (WFD Client or Group Owner)
nan0	Neighbor Awareness Networking mode (Wi-Fi Aware™)

➔ After complete loading of the WLAN kernel driver and after the hardware is initialized, the following kernel modules are listed at the system:

```
$> lsmod
Module              Size      Used by
sd8xxx            450560      0
mlan              438272      1 sd8xxx
```

6.3 Removing the WLAN Device Driver

Before the WLAN kernel driver can be removed, it is necessary to shut down the Ethernet devices, which have been configured by WLAN driver mode option `drv_mode` and brought-up for network-manager service.

Driver mode STA:	`drv_mode=[1,3,5,7]` → **`ifconfig mlan0 down`**
Driver Mode UAP:	`drv_mode=[2,3,6,7]` → **`ifconfig uap0 down`**
Driver Mode WFD:	`drv_mode=[4,5,6,7]` → **`ifconfig wfd0 down`**
Driver Mode NAN:	`drv_mode=[16 … 23]` → **`ifconfig nan0 down`**

The following requirement must be met:

✓ Ethernet devices are shut down.

To remove the loaded WLAN device driver, use:

```
$> ifconfig mlan0 down
$> ifconfig uap0 down
$> ifconfig wfd0 down
$> ifconfig nan0 down
$> rmmod sd8xxx
$> rmmod mlan
```

7 Bluetooth HCI Device Driver

7.1 Installing the Bluetooth HCI Device Driver (SD-BT)

Important Kernel Driver Options

During loading of the WLAN device kernel module usually the WLAN/Bluetooth combo firmware is uploaded to the PAN9028 module. The firmware binaries are stored at the Linux vendor specific library path `/lib/firmware/nxp`. By using the driver option **`fw_name`**, it is possible to define only the Bluetooth firmware binary file.

To specify only the Bluetooth firmware for parallel download, use:

`fw_name=nxp/sd8987_bt.bin  bt_fw_serial=0`

All available Bluetooth driver options of NXP device kernel module are listed in the file `README` which is included in the software package.

In case of using the WLAN firmware with parallel download option, the Bluetooth firmware can be downloaded by installing the Bluetooth kernel module.

1. Change the directory to the compiled Bluetooth driver binary directory.
2. Use the Linux command **`insmod`** to install the NXP kernel driver:

```
$> cd ${TOP}/p33/SD-UAPSTA-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-
C4X16651-MGPL/bin_sd8987_bt
$> ls
bt8987.ko  config  fmapp  load  README  unload
$> insmod bt8987.ko
```

➔ Now the Bluetooth HCI device driver is installed.

7.2 Installing the Bluetooth HCI Device Driver (UART-BT)

All available Bluetooth driver options of NXP device kernel module are listed in the file `README` which is included in the software package.

1. Change the directory to the compiled Bluetooth driver binary directory.

2. Use the Linux command `insmod` to install the NXP kernel driver.

```
$> cd ${TOP}/p134/SD-UAPSTA-UART-BT-8987-U16-MMC-W16.87.10.p134-
16.136.10.p134-C4X16687-MGPL/bin_muart
$> ls
hci_uart.ko  README
$> insmod hci_uart.ko reset=1 wakeupmode=1
```

➔ Now the Bluetooth HCI device driver is installed.

7.3 Loading the Bluetooth Device Driver

In case of using the WLAN firmware with parallel download option, the Bluetooth firmware can be downloaded by installing the Bluetooth kernel module.

Usually the Bluetooth SoC firmware is downloaded combined with the WLAN SoC firmware.

By loading of the Bluetooth device kernel, the driver activates the Bluetooth firmware, and initializes the module's hardware.

➔ The following kernel messages will be shown:

```
kernel: BT: Loading driver
kernel: BT Request firmware: nxp/sd8987_bt.bin
kernel: BT: FW download over, size 210952 bytes
kernel: BT FW is active
kernel: BT: Driver loaded successfully
```

➔ The following kernel module will be listed at the system:

```
$> lsmod
Module                  Size      Used by
bt8xxx                 81920      1
```

➔ A new Bluetooth device will be registered on the Linux host system: "hci0" (Bluetooth device interface).

7.4 Bluetooth Stack and Services

The Bluetooth device kernel driver is interconnected to the BlueZ stack.

The Bluetooth "bt8987" kernel module requires the installation of the BlueZ stack on the Linux system, which supports the core Bluetooth layers and protocols. Additionally, applications such as Bluetooth service (bluetoothd), rfkill, and rfcomm can be installed.

To install software packages for enabling the Bluetooth service (using “system” daemon), use:

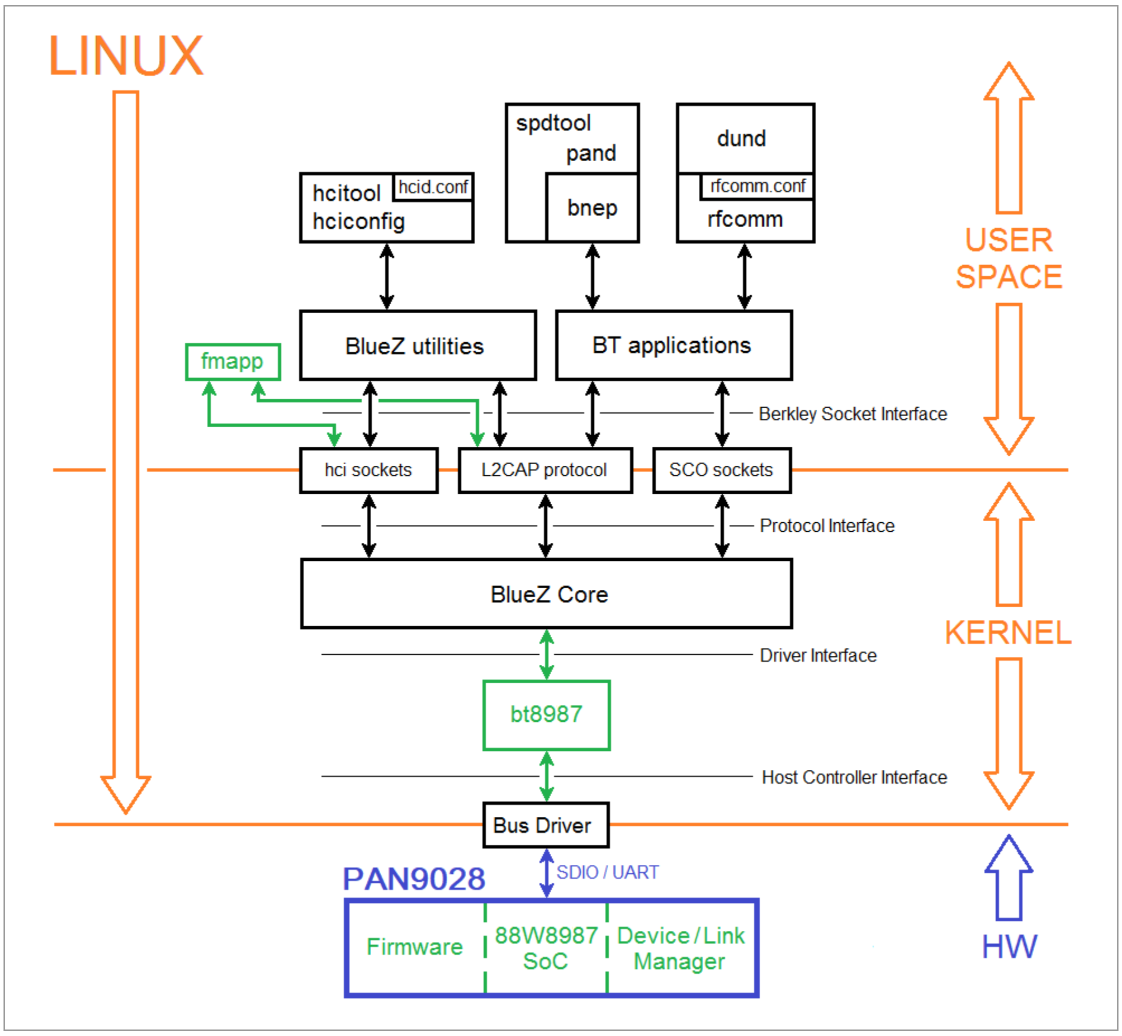
```
$> sudo apt-get install bluetooth bluez bluez-tools rfkill rfcomm
$> sudo systemctl start bluetooth.service
$> sudo systemctl enable bluetooth.service
```

7.5 Bluetooth Protocol Stack

The Linux system provides a standard Bluetooth protocol stack by BlueZ. The Bluetooth device driver runs on top of the SDIO bus driver. The driver forwards data and commands between the Bluetooth SoC firmware and the driver interface of BlueZ core. Additionally, private commands are handled from the Bluetooth driver.

By loading the Bluetooth device driver:

- the hardware is registered by the bus driver,
- the Bluetooth SoC firmware is initialized, and
- a HCI device is registered with the BlueZ stack.



7.6 Bluetooth HCI Device Interface (SD-BT)

The Bluetooth HCI device can be controlled by Linux Bluetooth Services.

After having installed the Bluetooth “bt8987” kernel module, the Bluetooth service is creating an interface with the name “hci0”. The tool “hciconfig” can be used to bring-up the Bluetooth interface.

To bring up the Bluetooth HCI device, use:

```
$> hciconfig hci0 up
```

Blocked Bluetooth Device

If the interface “hci0” cannot be brought-up make sure that the Bluetooth device is turned on and not blocked.

Unblocking the Bluetooth Device

To unblock the Bluetooth HCI device, use:

```
$> sudo rfkill unblock bluetooth
$> hciconfig hci0 up
```

7.7 Bluetooth HCI Device Interface (UART-BT)

The Bluetooth HCI device can be controlled by Linux Bluetooth Services.

After having installed the Bluetooth “bt8987” kernel module, the HCI device (connected via UART interface to the Linux Host system) needs to be attached for interconnecting to BlueZ stack.
The tool “hciconfig” can be used to bring-up the Bluetooth device.

To attach the serial device via UART HCI to BlueZ stack, use:

```
$> hciattach -b /dev/ttyS1 any 3000000 flow
```

To bring up the Bluetooth HCI device, use:

```
$> hciconfig hci0 up
```

7.8 Controlling the Bluetooth HCI Device

The Bluetooth device can be configured and controlled by the “hcitool”. The “hcitool” is used to configure Bluetooth connections and send special commands to Bluetooth devices.

Usage of hcitool

```
$> sudo hcitool -i hci0 scan
Scanning ...
       00:13:43:75:91:80      another_BT_device
$> sudo ping -i hci0 00:13:43:75:91:80
Ping: 00:13:43:75:91:80 from 00:13:43:88:91:38 (data size 44)
4 bytes from 00:13:43:75:91:80 id 0 time 15.38ms
4 bytes from 00:13:43:75:91:80 id 1 time 6.17ms
```

Setting Example of Bluetooth Device by hcitool

```
$> sudo hcitool -i hci0 cmd 0x03 0x0003
$> sudo hcitool -i hci0 cmd 0x03 0x001a 0x03
$> sudo hcitool -i hci0 cmd 0x03 0x0005 0x02 0x00 0x03
$> sudo hcitool -i hci0 cmd 0x06 0x0003
```

Description of Settings

Bluetooth HCI device `hci0` settings:

- Reset
- Write scan enable
- Set event filter to allow all connections with role “switchb”
- Enable device under test mode

7.9 Removing the Bluetooth HCI Device

To remove the Bluetooth HCI Device (SD-BT), use:

```
$> hciconfig hci0 down
$> rmmod bt8xxx
```

To remove the Bluetooth HCI Device (UART-BT), use:

```
$> hciconfig hci0 down
$> sudo killall -9 hciattach
$> rmmod hci_uart
```

8 Bluetooth Character Device Driver

8.1 Installing the Bluetooth Character Device Driver

Important Kernel Driver Options

During loading of the WLAN device kernel module usually the WLAN/Bluetooth combo firmware is uploaded to the PAN9028 module. The firmware binaries are stored at the Linux vendor specific library path `/lib/firmware/nxp`. By using the driver option `fw_name` it is possible to define only the Bluetooth firmware binary file.

To specify only the Bluetooth firmware for parallel download, use:

```
fw_name=nxp/sd8987_bt.bin bt_fw_serial=0
```

All available Bluetooth driver options of the NXP device kernel module are listed in the file `README` which is included in the software package.

In case of using the Bluetooth firmware with parallel download option, the WLAN firmware can be downloaded by installing the WLAN kernel module.

1. To change the directory to the compiled Bluetooth driver binary directory, use:

```
$> cd ${TOP}/p33/SD-UAPSTA-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-C4X16651-MGPL/bin_sd8987_btchar
$> ls
mbt8987.ko  config  fmapp  load  README  unload
```

2. To install the NXP kernel driver use the Linux command `insmod`:

```
$> insmod mbt8987.ko
```

8.2 Controlling the Bluetooth Character Device Interface

The Bluetooth character device can be controlled through the following interface: The Bluetooth mbt8987 device kernel is creating a device interface at the Linux device directory `/dev/mbtchar0`.

8.3 Controlling the Bluetooth Character Device

The Bluetooth character device can be controlled by a NXP utility. After having installed the Bluetooth “mbt8987” device kernel, the kernel driver is creating the interface “mbtchar0”.

To control the Bluetooth character interface, the utility “fmapp” can be used.

Setting Example of Bluetooth Character Device by utility “fmapp”

```
$> ./fmapp mbtchar0 cmd 0x03 0x0003
$> ./fmapp mbtchar0 cmd 0x03 0x001a 0x03
$> ./fmapp mbtchar0 cmd 0x03 0x0005 0x02 0x00 0x03
$> ./fmapp mbtchar0 cmd 0x06 0x0003
```

Description of Settings

Bluetooth character device `mbtchar0` setting:

- Reset
- Write scan enable
- Set event filter to allow all connections with role “switch"
- Enable device under test mode

8.4 Removing the Bluetooth Character Device

To remove the Bluetooth HCI Device, use:

```
$> rmmod mbt8xxx
```

9 WLAN Utilities

NXP provides utilities for the user space to configure, control, and obtain events for driving the WLAN device in various modes.

9.1 MLAN Utility for Station Mode

To execute the MLANUTL utility in the subdirectory of WLAN driver and utility binaries, use:

```
$> cd ${TOP}/p33/SD-UAPSTA-BT-8987-U16-MMC-W16.68.10.p33-16.26.1.p33-
C4X16651-MGPL/bin_sd8987
```

9.1.1 Enabling the Clear Channel Assessment (CCA)

Adaptivity Capability

The CCA is required to perform the adaptivity test in compliance with ETSI EN 300 328 V2.1.1 and ETSI EN 301 893 V2.1.1.

To configure the CCA for adaptivity capability of station, use:

```
$> ./mlanutl mlan0 hostcmd config/ed_mac_ctrl_V3_8987.conf
ed_mac_ctrl_v3
```

9.1.2 Configuring the Network Interface of Station Mode

To configure the network interface by Linux Network tools, use:

```
$> kill -9 wpa_supplicant
$> wpa_supplicant -i mlan0 -c ${CONFIG_DIRECTORY}/${SSID}.conf -f
/tmp/wpa_supplicant.log
$> dhclient -d -v mlan0 &> /tmp/dhclient.log
```

Definition of wpa_supplicant Configuration File

`SSID` = Service Set Identifier of Access Point (AP)

Content of SSID.conf:

```
network={
      ssid="SSID"
      psk="pre-shared-key"
}
```

9.1.3 Configuring the Stations Tx Data Rate (txratecfg)

The data rate can be set only after association.

To configure the Tx data rate of station, use:

```
$> ./mlanutl mlan0 txratecfg ${HTCFG} ${TXRATECFG}
```

Definition of Variables

`TXRATECFG` = [TXRATE]

`HTCFG` = [0, 1, 2]

With `HTCFG`=0 (Legacy Mode 802.11a/b/g):

• TXRATE=0	for 1 Mbps (802.11b)
• TXRATE=1	for 2 Mbps (802.11b)
• TXRATE=2	for 5.5 Mbps (802.11b)
• TXRATE=3	for 11 Mbps (802.11b)
• TXRATE=4	for 6 Mbps (802.11a/g)
• TXRATE=5	for 9 Mbps (802.11a/g)
• TXRATE=6	for 12 Mbps (802.11a/g)
• TXRATE=7	for 18 Mbps (802.11a/g)
• TXRATE=8	for 24 Mbps (802.11a/g)
• TXRATE=9	for 36 Mbps (802.11a/g)
• TXRATE=10	for 48 Mbps (802.11a/g)
• TXRATE=11	for 54 Mbps (802.11a/g)

With `HTCFG`=1 (HT Mode 802.11n):

• TXRATE=0	for MCS0 (802.11n)
• TXRATE=1	for MCS1 (802.11n)
• TXRATE=2	for MCS2 (802.11n)
• TXRATE=3	for MCS3 (802.11n)
• TXRATE=4	for MCS4 (802.11n)
• TXRATE=5	for MCS5 (802.11n)
• TXRATE=6	for MCS6 (802.11n)
• TXRATE=7	for MCS7 (802.11n)
• TXRATE=32	for MCS32 (802.11n)

With `HTCFG`=2 (VHT Mode 802.11ac):

TXRATE=0	for MCS0 (802.11ac)
TXRATE=1	for MCS1 (802.11ac)
TXRATE=2	for MCS2 (802.11ac)
TXRATE=3	for MCS3 (802.11ac)
TXRATE=4	for MCS4 (802.11ac)
TXRATE=5	for MCS5 (802.11ac)
TXRATE=6	for MCS6 (802.11ac)
TXRATE=7	for MCS7 (802.11ac)
TXRATE=8	for MCS8 (802.11ac)
TXRATE=9	for MCS9 (802.11ac, only for 40 MHz or 80 MHz channels)
TXRATE=32	for MCS32 (802.11ac)

For auto rate (default setting), use:

```
$> ./mlanutl mlan0 txratecfg 0xff
```

9.1.4 Setting the Station

9.1.4.1 For 2.4 GHz Band in Legacy Mode (802.11b/g)

To configure the station device, use:

```
$> ./mlanutl mlan0 httxcfg 0x2C 1
$> ./mlanutl mlan0 htcapinfo 0x4800000 1
```

9.1.4.2 For 5 GHz Band in Legacy Mode (802.11a)

To configure the station device, use:

```
$> ./mlanutl mlan0 httxcfg 0x6E 2
$> ./mlanutl mlan0 htcapinfo 0x5820000 2
```

9.1.4.3 For HT20 Mode (802.11n)

To configure the station device, use:

```
$> ./mlanutl mlan0 httxcfg 0x3C 0
$> ./mlanutl mlan0 htcapinfo 0x20800000 0
```

9.1.4.4 For HT40 Mode (802.11n)

To configure the station device, use:

```
$> ./mlanutl mlan0 httxcfg 0x5E 0
$> ./mlanutl mlan0 htcapinfo 0x21020000 0
```

9.1.4.5 For VHT20 Mode (802.11ac)

To configure the station device, use:

```
$> ./mlanutl mlan0 vhtcfg 2 2 0 0x33c179b0 0xfffa 0xfffa
$> ./mlanutl mlan0 vhtcfg 2 1 0 0x33c179b0 0xfffa 0xfffa
$> ./mlanutl mlan0 httxcfg 0x20 0
$> ./mlanutl mlan0 htcapinfo 0x4800000 0
```

9.1.4.6 For HT40 Mode (802.11ac)

To configure the station device:

```
$> ./mlanutl mlan0 vhtcfg 2 2 0 0x33c179b0 0xfffa 0xfffa
$> ./mlanutl mlan0 vhtcfg 2 1 0 0x33c179b0 0xfffa 0xfffa
$> ./mlanutl mlan0 httxcfg 0x62 0
$> ./mlanutl mlan0 htcapinfo 0x5820000 0
```

9.1.4.7 For HT80 Mode (802.11ac)

To configure the station device:

```
$> ./mlanutl mlan0 vhtcfg 2 2 0 0x33c179b0 0xfffa 0xfffa
$> ./mlanutl mlan0 vhtcfg 2 1 1 0x33c179b0 0xfffa 0xfffa
$> ./mlanutl mlan0 httxcfg 0x62 0
$> ./mlanutl mlan0 htcapinfo 0x5820000 0
```

9.1.4.8 With HT20/40 Coexistence Mechanism

The HT20/40 coexistence configuration shall be used in country domains, where 802.11n HT20 WLAN channel and HT40 WLAN channel coexists and can overlap. By this command the firmware sends 802.11n 20/40 coexistence management frames to the external access point.

The configuration parameters are set in the file `11n_2040coex.conf`, available in the driver directory `config`.

To configure the station device, use:

```
$> ./mlanutl mlan0 hostcmd config/11n_2040coex.conf 2040coex
```

9.2 UAP Utility for Micro-AP Mode

To execute the UAPUTL utility in the subdirectory of WLAN driver and utility, use:

```
$> cd ${TOP}/p33/SD-UAPSTA-BT-8987-U16-MMC-W16.68.10.p33-16.26.10.p33-
C4X16651-MGPL/bin_sd8987
```

9.2.1 Enabling the Clear-Channel-Assessment (CCA)

Adaptivity Capability

The CCA is required to perform the adaptivity test in compliance with ETSI EN 300 328 V2.1.1 and ETSI EN 301 893 V2.1.1.

To configure the CCA for adaptivity capability of BSS (Basic Service Set), use:

```
$> ./uaputl.exe -i uap0 hostcmd config/ed_mac_ctrl_V3_8987.conf
ed_mac_ctrl_v3
```

9.2.2 Stopping the Basic Service Set (bss_stop)

To stop the BSS, use:

```
$> ./uaputl.exe -i uap0 bss_stop
```

9.2.3 Resetting the System Basic Service Set (sys_reset)

To reset the system of BSS, use:

```
$> ./uaputl.exe -i uap0 sys_reset
```

9.2.4 Starting the Basic Service Set

9.2.4.1 At 2.4 GHz Band in LT Mode (802.11b)

Definition of Variables

`CHANNEL` = [1, 2, 3, ..., 11, 12, 13]

Some channels might be blocked in pre-stored OTP data due to regulatory issues. To see which channels are supported, please consider the Product Specification of your product.

To configure and start the BSS, use:

```
$> ./uaputl.exe -i uap0 bss_stop
$> ./uaputl.exe -i uap0 sys_reset
$> ./uaputl.exe -i uap0 sys_cfg_2040_coex 1
$> ./uaputl.exe -i uap0 sys_cfg_11n 0
$> ./uaputl.exe -i uap0 sys_cfg_rates 0x82 0x84 0x8B 0x96
$> ./uaputl.exe -i uap0 sys_cfg_channel_ext ${CHANNEL} 0
$> ./uaputl.exe -i uap0 sys_cfg_ssid "PAN9028_uAP0"
$> ./uaputl.exe -i uap0 bss_start
```

9.2.4.2 At 2.4 GHz Band in LT Mode (802.11bg)

Definition of Variables

`CHANNEL` = [1, 2, 3, ..., 11, 12, 13]

Some channels might be blocked in pre-stored OTP data due to regulatory issues. To see which channels are supported, please consider the Product Specification of your product.

To configure and start the BSS, use:

```
$> ./uaputl.exe -i uap0 bss_stop
$> ./uaputl.exe -i uap0 sys_reset
$> ./uaputl.exe -i uap0 sys_cfg_2040_coex 1
$> ./uaputl.exe -i uap0 sys_cfg_11n 0
$> ./uaputl.exe -i uap0 sys_cfg_rates 0x82 0x84 0x8B 0x96 0x0C 0x12
0x18 0x24 0x30 0x48 0x60 0x6C
$> ./uaputl.exe -i uap0 sys_cfg_channel_ext ${CHANNEL} 0
$> ./uaputl.exe -i uap0 sys_cfg_ssid "PAN9028_uAP0"
$> ./uaputl.exe -i uap0 bss_start
```

9.2.4.3 At 2.4 GHz Band in HT20 Mode (802.11n)

Definition of Variables

`CHANNEL` = [1, 2, 3, ..., 11, 12, 13]

Some channels might be blocked in pre-stored OTP data due to regulatory issues. To see which channels are supported, please consider the Product Specification of your product.

To configure and start the BSS, use:

```
$> ./uaputl.exe -i uap0 bss_stop
$> ./uaputl.exe -i uap0 sys_reset
$> ./uaputl.exe -i uap0 sys_cfg_2040_coex 1
$> ./uaputl.exe -i uap0 httxcfg 0x20
$> ./uaputl.exe -i uap0 sys_cfg_11n 1 0x012C 3
$> ./uaputl.exe -i uap0 sys_cfg_rates 0x82 0x84 0x8B 0x96 0x0C 0x12
0x18 0x24 0x30 0x48 0x60 0x6C
$> ./uaputl.exe -i uap0 sys_cfg_channel_ext ${CHANNEL} 0
$> ./uaputl.exe -i uap0 sys_cfg_ssid "PAN9028_uAP0"
$> ./uaputl.exe -i uap0 bss_start
```

9.2.4.4 At 2.4 GHz Band in HT40 Mode (802.11n)

Definition of Variables

`CENTER_CHANNEL` = [3, 6, 7, 9, 11]

`SECONDARY_CHANNEL` = [2, 4]

- with `SECONDARY_CHANNEL=2` (above)
  `CHANNEL=${CENTER_CHANNEL - 2}`
- with `SECONDARY_CHANNEL=4` (below)
  `CHANNEL=${CENTER_CHANNEL + 2}`

Some channels might be blocked in pre-stored OTP data due to regulatory issues. To see which channels are supported, please consider the Product Specification of your product.

To configure and start the BSS, use:

```
$> ./uaputl.exe -i uap0 bss_stop
$> ./uaputl.exe -i uap0 sys_reset
$> ./uaputl.exe -i uap0 sys_cfg_2040_coex 0
$> ./uaputl.exe -i uap0 httxcfg 0x62
$> ./uaputl.exe -i uap0 sys_cfg_11n 1 0x016E 3
$> ./uaputl.exe -i uap0 sys_cfg_rates 0x82 0x84 0x8B 0x96 0x0C 0x12
0x18 0x24 0x30 0x48 0x60 0x6C
$> ./uaputl.exe -i uap0 sys_cfg_channel_ext ${CHANNEL} 0
${SECONDARY_CHANNEL}
$> ./uaputl.exe -i uap0 sys_cfg_ssid "PAN9028_uAP0"
$> ./uaputl.exe -i uap0 bss_start
```

9.2.4.5 At 5 GHz Band in LT Mode (802.11a)

Definition of Variables

`CHANNEL` = [36, 40, 44, 48, 52, 56, 60, 64, 100, 104, …, 165]

Some channels might be blocked in pre-stored OTP data due to regulatory issues. To see which channels are supported, please consider the Product Specification of your product.

To configure and start the BSS, use:

```
$> ./uaputl.exe -i uap0 bss_stop
$> ./uaputl.exe -i uap0 sys_reset
$> ./uaputl.exe -i uap0 sys_cfg_2040_coex 1
$> ./uaputl.exe -i uap0 sys_cfg_11n 0
$> ./uaputl.exe -i uap0 sys_cfg_rates 0x8C 0x12 0x98 0x24 0xB0 0x48
0x60 0x6C
$> ./uaputl.exe -i uap0 sys_cfg_channel_ext ${CHANNEL} 1
$> ./uaputl.exe -i uap0 sys_cfg_ssid "PAN9028_uAP0"
$> ./uaputl.exe -i uap0 bss_start
```

9.2.4.6 At 5 GHz Band in HT20 Mode (802.11n)

Definition of Variables

`CHANNEL` = [36, 40, 44, 48, 52, 56, 60, 64, 100, 104, …, 165]

Some channels might be blocked in pre-stored OTP data due to regulatory issues. To see which channels are supported, please consider the Product Specification of your product.

To configure and start the BSS, use:

```
$> ./uaputl.exe -i uap0 bss_stop
$> ./uaputl.exe -i uap0 sys_reset
$> ./uaputl.exe -i uap0 sys_cfg_2040_coex 1
$> ./uaputl.exe -i uap0 httxcfg 0x20
$> ./uaputl.exe -i uap0 sys_cfg_11n 1 0x012C 3
$> ./uaputl.exe -i uap0 sys_cfg_rates 0x8C 0x12 0x98 0x24 0xB0 0x48
0x60 0x6C
$> ./uaputl.exe -i uap0 sys_cfg_channel_ext ${CHANNEL} 1
$> ./uaputl.exe -i uap0 sys_cfg_ssid "PAN9028_uAP0"
$> ./uaputl.exe -i uap0 bss_start
```

9.2.4.7 At 5 GHz band in HT40 Mode (802.11n)

Definition of Variables

`CENTER_CHANNEL` = [38, 46, 54, 62, 102, 110, 118, …, 159]

`SECONDARY_CHANNEL` = [2 , 4]

- with `SECONDARY_CHANNEL=2` (above)
  `CHANNEL=${CENTER_CHANNEL - 2}`
- with `SECONDARY_CHANNEL=4` (below)
  `CHANNEL=${CENTER_CHANNEL + 2}`

Some channels might be blocked in pre-stored OTP data due to regulatory issues. To see which channels are supported, please consider the Product Specification of your product.

To configure and start the BSS, use:

```
$> ./uaputl.exe -i uap0 bss_stop
$> ./uaputl.exe -i uap0 sys_reset
$> ./uaputl.exe -i uap0 sys_cfg_2040_coex 0
$> ./uaputl.exe -i uap0 httxcfg 0x62
$> ./uaputl.exe -i uap0 sys_cfg_11n 1 0x016E 3
$> ./uaputl.exe -i uap0 sys_cfg_rates 0x8C 0x12 0x98 0x24 0xB0 0x48
0x60 0x6C
$> ./uaputl.exe -i uap0 sys_cfg_channel_ext ${CHANNEL} 1
${SECONDARY_CHANNEL}
$> ./uaputl.exe -i uap0 sys_cfg_ssid "PAN9028_uAP0"
$> ./uaputl.exe -i uap0 bss_start
```

9.2.4.8 At 5 GHz Band in VHT20 Mode (802.11ac)

Definition of Variables

`CHANNEL` = [36, 40, 44, 48, 52, 56, 60, 64, 100, 104, …, 165]

Some channels might be blocked in pre-stored OTP data due to regulatory issues. To see which channels are supported, please consider the Product Specification of your product.

To configure and start the BSS, use:

```
$> ./uaputl.exe -i uap0 bss_stop
$> ./uaputl.exe -i uap0 sys_reset
$> ./uaputl.exe -i uap0 vhtcfg 2 3 0 0x33c179b0 0xfffa 0xfffa
$> ./uaputl.exe -i uap0 httxcfg 0x20
$> ./uaputl.exe -i uap0 sys_cfg_11n 1 0x016E 3
$> ./uaputl.exe -i uap0 sys_cfg_rates 0x8C 0x12 0x98 0x24 0xB0 0x48
0x60 0x6C
$> ./uaputl.exe -i uap0 sys_cfg_channel_ext ${CHANNEL} 1
$> ./uaputl.exe -i uap0 sys_cfg_ssid "PAN9028_uAP0"
$> ./uaputl.exe -i uap0 bss_start
```

9.2.4.9 At 5 GHz Band in VHT40 Mode (802.11ac)

Definition of Variables

`CENTER_CHANNEL` = [38, 46, 54, 62, 102, 110, 118, …, 159]

`SECONDARY_CHANNEL` = [2 , 4]

- with `SECONDARY_CHANNEL=2` (above)
  `CHANNEL=${CENTER_CHANNEL - 2}`
- with `SECONDARY_CHANNEL=4` (below)
  `CHANNEL=${CENTER_CHANNEL + 2}`

Some channels might be blocked in pre-stored OTP data due to regulatory issues. To see which channels are supported, please consider the Product Specification of your product.

To configure and start the BSS, use:

```
$> ./uaputl.exe -i uap0 bss_stop
$> ./uaputl.exe -i uap0 sys_reset
$> ./uaputl.exe -i uap0 vhtcfg 2 3 0 0x33c179b0 0xfffa 0xfffa
$> ./uaputl.exe -i uap0 httxcfg 0x62
$> ./uaputl.exe -i uap0 sys_cfg_11n 1 0x016E 3
$> ./uaputl.exe -i uap0 sys_cfg_rates 0x8C 0x12 0x98 0x24 0xB0 0x48
0x60 0x6C
$> ./uaputl.exe -i uap0 sys_cfg_channel_ext ${CHANNEL} 1
${SECONDARY_CHANNEL}
$> ./uaputl.exe -i uap0 sys_cfg_ssid "PAN9028_uAP0"
$> ./uaputl.exe -i uap0 bss_start
```

9.2.4.10 At 5 GHz Band in VHT80 Mode (802.11ac)

Definition of Variables

`CENTER_CHANNEL` = [42, 58, 106, 122, 138, 155]

`SECONDARY_CHANNEL` = [2 , 4]

- with `SECONDARY_CHANNEL=2` (above)
  `CHANNEL=${CENTER_CHANNEL - 2}`
- with `SECONDARY_CHANNEL=4` (below)
  `CHANNEL=${CENTER_CHANNEL + 2}`

Some channels might be blocked in pre-stored OTP data due to regulatory issues. To see which channels are supported, please consider the Product Specification of your product.

To configure and start the BSS, use:

```
$> ./uaputl.exe -i uap0 bss_stop
$> ./uaputl.exe -i uap0 sys_reset
$> ./uaputl.exe -i uap0 vhtcfg 2 3 1 0x33c179b0 0xfffa 0xfffa
$> ./uaputl.exe -i uap0 httxcfg 0x62
$> ./uaputl.exe -i uap0 sys_cfg_11n 1 0x016E 3
$> ./uaputl.exe -i uap0 sys_cfg_rates 0x8C 0x12 0x98 0x24 0xB0 0x48
0x60 0x6C
$> ./uaputl.exe -i uap0 sys_cfg_channel_ext ${CHANNEL} 1
${SECONDARY_CHANNEL}
$> ./uaputl.exe -i uap0 sys_cfg_ssid "PAN9028_uAP0"
$> ./uaputl.exe -i uap0 bss_start
```

9.2.5 Configuring the Tx Data Rate for BSS (txratecfg)

Changing of Tx data rate is only possible after the BSS has been started.

Definition of Variables

`TXRATECFG` = [TXRATE]

`HTCFG` = [0, 1, 2]

To configure the Tx data rate of BSS, use:

```
$> ./uaputl.exe -i uap0 txratecfg ${HTCFG} ${TXRATECFG}
```

With `HTCFG`=0 (Legacy Mode 802.11a/b/g)

TXRATE=0	for 1 Mbps (802.11b)
TXRATE=1	for 2 Mbps (802.11b)
TXRATE=2	for 5.5 Mbps (802.11b)
TXRATE=3	for 11 Mbps (802.11b)
TXRATE=4	for 6 Mbps (802.11a/g)
TXRATE=5	for 9 Mbps (802.11a/g)
TXRATE=6	for 12 Mbps (802.11a/g)
TXRATE=7	for 18 Mbps (802.11a/g)
TXRATE=8	for 24 Mbps (802.11a/g)
TXRATE=9	for 36 Mbps (802.11a/g)
TXRATE=10	for 48 Mbps (802.11a/g)
TXRATE=11	for 54 Mbps (802.11a/g)

With `HTCFG`=1 (HT Mode 802.11n)

TXRATE=0	for MCS0 (802.11n)
TXRATE=1	for MCS1 (802.11n)
TXRATE=2	for MCS2 (802.11n)
TXRATE=3	for MCS3 (802.11n)
TXRATE=4	for MCS4 (802.11n)
TXRATE=5	for MCS5 (802.11n)
TXRATE=6	for MCS6 (802.11n)
TXRATE=7	for MCS7 (802.11n)
TXRATE=32	for MCS32 (802.11n)

With `HTCFG`=2 (VHT Mode 802.11ac)

TXRATE=0	for MCS0 (802.11ac)
TXRATE=1	for MCS1 (802.11ac)
TXRATE=2	for MCS2 (802.11ac)
TXRATE=3	for MCS3 (802.11ac)
TXRATE=4	for MCS4 (802.11ac)
TXRATE=5	for MCS5 (802.11ac)
TXRATE=6	for MCS6 (802.11ac)
TXRATE=7	for MCS7 (802.11ac)
TXRATE=8	for MCS8 (802.11ac)
TXRATE=9	for MCS9 (802.11ac, only for 40 MHz or 80 MHz channels)
TXRATE=32	for MCS32 (802.11ac)

With `HTCFG`="" (empty)

TXRATE=0xff	for auto rating (default)

9.2.6 Configuring Network Interface of Basic Service Set

dhcpd.conf file content

```
subnet 192.168.33.0 netmask 255.255.255.0 {
      range 192.168.33.10 192.168.33.200;
      option routers 192.168.33.1;
      option domain-name-servers 8.8.8.8;
}
```

To configure the network interface and DHCP-server for BSS:

```
$> ifconfig uap0 192.168.33.1 netmask 255.255.255.0
$> kill -9 dhcpd
$> dhcpd -4 -f -d -cf ${CONFIG_DIRECTORY}/dhcpd.conf uap0
```

9.3 UAP Utility Configuration File

The UAP Utility Configuration File can only be used for 802.11a/b/g/n

9.3.1 Reading the System Configuration of Basic Service Set (sys_config)

To read the system configuration of BSS, use:

```
$> ./uaputl.exe -i uap0 sys_config
```

9.3.2 Loading the Basic Service Set Settings from UAPUTL Configuration File

UAPUTL Configuration File

The file `uaputl.conf` is available in the subdirectory `config` of the WLAN driver directory `bin_sd8xxx`.

To configure and start the BSS, use:

```
$> ./uaputl.exe -i uap0 bss_stop
$> ./uaputl.exe -i uap0 sys_reset
$> ./uaputl.exe -i uap0 sys_config config/uaputl.conf
$> ./uaputl.exe -i uap0 bss_config config/uaputl.conf
$> ./uaputl.exe -i uap0 bss_start
```

9.3.3 Default UAPUTL Configuration File

⚠ The embedded authenticator of UAPUTL utility does not support WPA3 SAE authentication mode. If WPA3 SAE authentication is required, please use the WPA3 SAE implementation of `hostapd` version 2.6.

Default BSS Configuration Parameter

Configuration Parameter ap_config={ … }	**Parameterize**	**Remark**
SSID	Marvell Micro AP	SSID of Micro AP
BeaconPeriod	100	Beacon period in TU
Channel	6	Radio channel 6 (Band=0)
ChanList	1, 6, 11	Scan channel list (Band=0)
Band	0	0 = 2.4 GHz band 1 = 5 GHz band
RxAntenna	0	0 = antenna A
TxAntenna	0	1 = antenna B
Rate	0x82,0x84,0x8b,0x96, 0x0C,0x12,0x18,0x24, 0x30,0x48,0x60,0x6c	Set of data rate that a station in the BSS use
TxPowerLevel	13	Transmit power level in dBm
BroadcastSSID	1	Broadcast SSID feature 1 = enable 0 = disable
RTSThreshold	2 347	RTS threshold value
FragThreshold	2 346	Fragmentation threshold value
DTIMPeriod	1	DTIM period in beacon periods
MCBCdataRate	0	MCBC rate to use for packet transmission (0 = auto)
TxBeaconRate	0	Beacon rate to use for Beacon transmission (0 = auto)
PktFwdCtl	1	Packet forwarding control

Configuration Parameter ap_config={ … }	**Parameterize**	**Remark**
StaAgeoutTimer	1 800	Inactive client station age out timer value in units of 100 ms
PSStaAgeoutTimer	400	Inactive client PS station age out timer value in units of 100 ms
MaxStaNum	10	Maximum number of stations allowed to connect. Need to change to MaxStaNum = 8 for “8987”.
Retrylimit	7	Retry limit to use for packet transmissions
AuthMode	0	0 = open authentication 1 = shared key authentication 3 = WPA3 SAE
Protocol	1	Protocol to use 1 = no security 2 = static WEP 8 = WPA 32 = WPA2 40 = WPA2 mixed mode 64 = WPA3 SAE
RSNReplayProtection	0	RSN replay protection 0 = disabled 1 = enabled
PairwiseUpdateTimeout	100	Pairwise handshake update timeout: 100 ms
PairwiseHandshakeRetries	3	Pairwise handshake retries: 3
GroupwiseUpdateTimeout	100	Groupwise handshake update timeout: 100 ms
GroupwiseHandshakeRetries	3	Groupwise handshake retries: 3
GroupRekeyTime	86400	Group key re-key interval, in second, 0 mean never re-key
Enable11n	1	1 = enable 0 = disable
HTCapInfo	0x111c	HTCapInfo
AMPDU	0x03	AMPDU
HT_MCS_MAP	0x000000ff	Bit 7-0 = MCS_SET_0 Bit 15-8 = MCS_SET_1
Enable2040Coex	1	Enable 20/40 coex feature
11d_enable	0	0 = disable 1 = enable
country	US	Country information

9.3.4 Parameterizing the Maximum Station Number

⚠ If using the file `uaputl.conf` for BSS configuration of PAN9028 micro-AP it is necessary to parameterize the parameter `MaxStaNum` as following:

```
MaxStaNum=8
```

Parameterize BSS configuration at file `uaputl.conf` as following:

Configuration Parameter ap_config={ … }	Parameterize	Remark
MaxStaNum	8	Maximum number of stations for chipset “8987”

9.3.5 Parameterizing the SSID

Parameterize BSS configuration at file `uaputl.conf` as following:

Configuration Parameter ap_config={ … }	Parameterize	Remark
SSID	PAN9028_uAP0	SSID of Micro AP

9.3.6 Parameterizing the Authentication and Protocol for WPA2 Mixed Mode

Parameterize BSS configuration at file `uaputl.conf` as following:

Configuration Parameter ap_config={ … }	Parameterize	Remark
AuthMode	0	0 = open authentication 1 = shared key authentication 255 = auto (open and shared key) authentication
Protocol	40	1 = no security 2 = WEP static 8 = WPA 32 = WPA2 40 = WPA2 mixed mode
PwkCipherWPA	8	Pairwise cipher type 4 = TKIP 8 = AES CCMP 12 = AES CCMP+TKIP
PwkCipherWPA2	8	
GwkCipher	8	Group cipher type 4 = TKIP 8 = AES CCMP
PSK	1234567890	WPA/WPA2 passphrase
GroupRekeyTime	86400	Group key re-key interval (in second) 0 = never re-key

9.3.7 Parameterizing the Authentication and Protocol for WPA2 Mode

Parameterize BSS configuration at file `uaputl.conf` as following:

Configuration Parameter ap_config={ … }	**Parameterize**	**Remark**
AuthMode	0	0 = open authentication 1 = shared key authentication 255 = auto (open and shared key) authentication
Protocol	32	1 = no Security 2 = WEP Static 8 = WPA 32 = WPA2 40 = WPA2 Mixed Mode
PwkCipherWPA	8	Pairwise cipher type 4 = TKIP 8 = AES CCMP 12 = AES CCMP+TKIP
PwkCipherWPA2	8	
GwkCipher	8	Group cipher type 4 = TKIP 8 = AES CCMP
PSK	1234567890	WPA/WPA2 passphrase
GroupRekeyTime	86400	Group key re-key interval (in second) 0 = never re-key

9.3.8 Enabling the RSN Replay Protection for WPA2

Parameterize BSS configuration at file `uaputl.conf` as following:

Configuration Parameter ap_config={ … }	**Parameterize**	**Remark**
RSNReplayProtection	1	RSN replay protection 1 = enabled 0 = disabled
PairwiseUpdateTimeout	100	Pairwise handshake update timeout: 100 ms
PairwiseHandshakeRetries	3	Pairwise handshake retries: 3
GroupwiseUpdateTimeout	100	Groupwise handshake update timeout: 100 ms
GroupwiseHandshakeRetries	3	Groupwise handshake retries: 3

9.3.9 Parameterizing the Channel

9.3.9.1 In 2.4 GHz and with HT Mixed Mode (802.11bgn)

Parameterize BSS configuration at file `uaputl.conf` as following:

Configuration Parameter ap_config={ … }	**Parameterize**	**Remark**
Channel	1	Radio Channel 1 (Band=0)
Band	0	0 = 2.4 GHz band 1 = 5 GHz band
Rate	0x82,0x84,0x8b,0x96, 0x0c,0x12,0x18,0x24, 0x30,0x48,0x60,0x6c	Set of data rate that a station in the BSS use
Enable11n	1	1 = enable 0 = disable
HTCapInfo	0x111c	HTCapInfo
AMPDU	0x03	AMPDU
HT_MCS_MAP	0x000000ff	Bit 7-0 = MCS_SET_0 Bit 15-8 = MCS_SET_1
Enable2040Coex	1	Enable 20/40 coex feature

9.3.9.2 In 2.4 GHz with LG Mode (802.11bg)

Parameterize BSS configuration at file `uaputl.conf` as following:

Configuration Parameter ap_config={ … }	**Parameterize**	**Remark**
Channel	1	Radio Channel 1 (Band=0)
Band	0	0 = 2.4 GHz band 1 = 5 GHz band
Rate	0x82,0x84,0x8b,0x96, 0x0c,0x12,0x18,0x24, 0x30,0x48,0x60,0x6c	Set of data rate that a station in the BSS use
Enable11n	0	1 = enable 0 = disable
HTCapInfo	0x111c	HTCapInfo
AMPDU	0x03	AMPDU
HT_MCS_MAP	0x000000ff	Bit 7-0 = MCS_SET_0 Bit 15-8 = MCS_SET_1
Enable2040Coex	1	Enable 20/40 coex feature

9.3.9.3 In 2.4 GHz with HT20 Mode (802.11n)

Parameterize BSS configuration at file `uaputl.conf` as following:

Configuration Parameter ap_config={ … }	**Parameterize**	**Remark**
Channel	1	Radio Channel 1 (Band=0)
Band	0	0 = 2.4 GHz band 1 = 5 GHz band
Rate	0x82,0x84,0x8b,0x96, 0x0c,0x12,0x18,0x24, 0x30,0x48,0x60,0x6c	Set of data rate that a station in the BSS use
Enable11n	1	1 = enable 0 = disable
HTCapInfo	0x012c	HTCapInfo
AMPDU	0x03	AMPDU
HT_MCS_MAP	0x000000ff	Bit 7-0 = MCS_SET_0 Bit 15-8 = MCS_SET_1
Enable2040Coex	1	Enable 20/40 coex feature

9.3.9.4 In 2.4 GHz with HT40 Mode (802.11n)

Parameterize BSS configuration at file `uaputl.conf` as following:

Configuration Parameter ap_config={ … }	**Parameterize**	**Remark**
Channel	5,2	Primary Channel 5, secondary channel above
	(9,4)[1]	Primary Channel 9, secondary channel below
Band	0	0 = 2.4 GHz band 1 = 5 GHz band
Rate	0x82,0x84,0x8b,0x96, 0x0c,0x12,0x18,0x24, 0x30,0x48,0x60,0x6c	Set of data rate that a station in the BSS use
Enable11n	1	1 = enable 0 = disable
HTCapInfo	0x016e	HTCapInfo
AMPDU	0x03	AMPDU
HT_MCS_MAP	0x000000ff	Bit 7-0 = MCS_SET_0 Bit 15-8 = MCS_SET_1
Enable2040Coex	0	Enable 20/40 coex feature

[1] Another configuration example.

9.3.9.5 In 5 GHz with LG Mode (802.11a)

Parameterize BSS configuration at file `uaputl.conf` as following:

Configuration Parameter ap_config={ … }	Parameterize	Remark
Channel	36	Radio Channel 36 (Band=1)
Band	1	1 = 5 GHz band 0 = 2.4 GHz band
Rate	0x8c, 0x12, 0x98, 0x24, 0xb0, 0x48, 0x60, 0x6c	Set of data rate that a station in the BSS use
Enable11n	0	0 = disable 1 = enable
HTCapInfo	0x111c	HTCapInfo
AMPDU	0x03	AMPDU
HT_MCS_MAP	0x000000ff	Bit 7-0: MCS_SET_0 Bit 15-8: MCS_SET_1
Enable2040Coex	1	Enable 20/40 coex feature

9.3.9.6 In 5 GHz with HT20 Mode (802.11n)

Parameterize BSS configuration at file `uaputl.conf` as following:

Configuration Parameter ap_config={ … }	Parameterize	Remark
Channel	36	Radio Channel 36 (Band=1)
Band	1	0 for 2.4 GHz band, 1 for 5 GHz band
Rate	0x8c,0x12,0x98,0x24, 0xb0,0x48,0x60,0x6c	Set of data rate that a station in the BSS use
Enable11n	1	1: enable 0: disable
HTCapInfo	0x012c	HTCapInfo
AMPDU	0x03	AMPDU
HT_MCS_MAP	0x000000ff	Bit 7-0: MCS_SET_0 Bit 15-8: MCS_SET_1
Enable2040Coex	1	Enable 20/40 coex feature

9.3.9.7 In 5 GHz with HT40 Mode (802.11n)

Parameterize BSS configuration at file `uaputl.conf` as following:

Configuration Parameter ap_config={ … }	Parameterize	Remark
Channel	36,2	Primary channel 36, secondary channel above
	(40,4)[2]	Primary channel 40, secondary channel below
Band	1	1 = 5 GHz band 0 = 2.4 GHz band
Rate	0x8c,0x12,0x98,0x24, 0xb0,0x48,0x60,0x6c	Set of data rate that a station in the BSS use
Enable11n	1	1 = enable 0 = disable
HTCapInfo	0x016e	HTCapInfo
AMPDU	0x03	AMPDU
HT_MCS_MAP	0x000000ff	Bit 7-0 = MCS_SET_0 Bit 15-8 = MCS_SET_1
Enable2040Coex	0	Enable 20/40 coex feature

9.4 Hostapd Utility for MicroAP Mode

Hostapd (host access point daemon) is a user space daemon that enables a network interface card to act as an access point and authentication server.

9.4.1 Configuring the 2.4 GHz MicroAP

Definition of hostapd configuration file for 2.4 GHz

```
interface=uap0
hw_mode=g
channel=0
SSID=PAN9028_uAP_24G
ieee80211n=1
```

[2] Another configuration example.

1. To start hostapd deamon with the configuration file, use:

```
$> hostapd -dd /etc/hostapd.conf
```

2. To configure the network interface and DHCP-server for BSS, use:

```
$> ifconfig uap0 192.168.33.1 netmask 255.255.255.0
$> kill -9 dhcpd
$> dhcpd -4 -f -d -cf ${CONFIG_DIRECTORY}/dhcpd.conf uap0
```

dhcpd.conf file content

```
subnet 192.168.33.0 netmask 255.255.255.0 {
      range 192.168.33.10 192.168.33.200;
      option routers 192.168.33.1;
      option domain-name-servers 8.8.8.8;
}
```

9.4.2 Configuring the 5 GHz MicroAP

Definition of hostapd configuration file for 5 GHz

```
interface=uap0
hw_mode=a
channel=0
SSID=PAN9028_uAP_5G
ieee80211n=1
```

1. To start hostapd deamon with the configuration file, use:

```
$> hostapd -dd /etc/hostapd.conf
```

2. To configure the network interface and DHCP-server for BSS, use:

```
$> ifconfig uap0 192.168.33.1 netmask 255.255.255.0
$> kill -9 dhcpd
$> dhcpd -4 -f -d -cf ${CONFIG_DIRECTORY}/dhcpd.conf uap0
```

dhcpd.conf file content

```
subnet 192.168.33.0 netmask 255.255.255.0 {
      range 192.168.33.10 192.168.33.200;
      option routers 192.168.33.1;
      option domain-name-servers 8.8.8.8;
}
```

9.4.3 Configuring the File for hostapd with WPA, WPA2, and WPA3

Definition of hostapd configuration file for 2.4 GHz with WPA

```
interface=uap0
hw_mode=g
channel=0
SSID=PAN9028_uAP_24G_WPA
ieee80211n=1
auth_algs=1
ieee80211n=1
wpa=1
rsn_pairwise=CCMP
wpa_passphrase=0123456789
```

Definition of hostapd configuration file for 2.4 GHz with WPA2

```
interface=uap0
hw_mode=g
channel=0
SSID=PAN9028_uAP_24G_WPA2
ieee80211n=1
auth_algs=1
ieee80211n=1
wpa=2
rsn_pairwise=CCMP
wpa_passphrase=0123456789
```

Definition of hostapd configuration file for 2.4 GHz with WPA3 SAE

```
interface=uap0
hw_mode=g
channel=0
SSID=PAN9028_uAP_24G_WPA3_SAE
ieee80211n=1
auth_algs=1
ieee80211n=1
wpa=2
rsn_pairwise=CCMP
wpa_passphrase=0123456789
wpa_key_mgmt=SAE
wpa_group_rekey=86400
rsn_pairwise=CCMP
ieee80211w=2
sae_groups=19 20 21
sae_require_mfp=1
sae_anti_clogging_threshold=5
```

Definition of hostapd configuration file for 2.4 GHz with WPA3 SAE transition mode (compatibility with WPA2-PSK)

```
interface=uap0
hw_mode=g
channel=0
SSID=PAN9028_uAP_24G_WPA2_SAE
ieee80211n=1
auth_algs=1
ieee80211n=1
wpa=2
rsn_pairwise=CCMP
wpa_passphrase=0123456789
wpa_key_mgmt=WPA-PSK SAE
wpa_group_rekey=86400
rsn_pairwise=CCMP
ieee80211w=1
sae_groups=19 20 21
sae_require_mfp=1
sae_anti_clogging_threshold=5
```

10 Life Support Policy

This Panasonic Industrial Devices Europe GmbH product is not designed for use in life support appliances, devices, or systems where malfunction can reasonably be expected to result in a significant personal injury to the user, or as a critical component in any life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Panasonic customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Panasonic Industrial Devices Europe GmbH for any damages resulting.

11 Contact Details

11.1 Contact Us

Please contact your local Panasonic Sales office for details on additional product options and services:

For Panasonic Sales assistance in the **EU**, visit
https://eu.industrial.panasonic.com/about-us/contact-us
Email: wireless@eu.panasonic.com

For Panasonic Sales assistance in **North America**, visit the Panasonic website
“Sales & Support” to find assistance near you at
https://na.industrial.panasonic.com/distributors

Please visit the **Panasonic Wireless Technical Forum** to submit a question at
https://forum.na.industrial.panasonic.com

11.2 Product Information

Please refer to the Panasonic Wireless Connectivity website for further information on our products and related documents:

For complete Panasonic product details in the **EU**, visit
http://pideu.panasonic.de/products/wireless-modules.html

For complete Panasonic product details in **North America**, visit
http://www.panasonic.com/rfmodules