

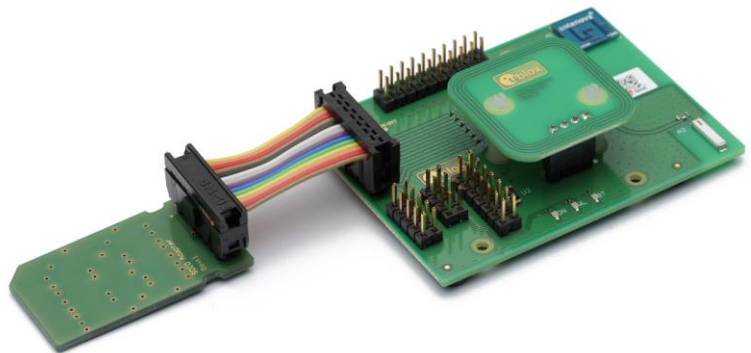
EVK-EMMY-W1

Evaluation kit for EMMY-W1 host-based multiradio modules with Wi-Fi, Bluetooth and NFC

User Guide

Abstract

This document describes how to set up the EVK-EMMY-W163-A and EVK-EMMY-W161-A evaluation kits. It also provides a technical overview of the EVK-EMMY-W1 series and describes how to compile the reference driver.



www.u-blox.com

UBX-15012713 - R04

Document Information	
Title	EVK-EMMY-W1
Subtitle	Evaluation kit for EMMY-W1 host-based multiradio modules with Wi-Fi, Bluetooth and NFC
Document type	User Guide
Document number	UBX-15012713
Revision, date	R04 23-Nov-2016
Document status	Early Production Information

Document status information	
Objective Specification	Document contains target values. Revised and supplementary data will be published later.
Advance Information	Document contains data based on early testing. Revised and supplementary data will be published later.
Early Production Information	Document contains data from product verification. Revised and supplementary data may be published later.
Production Information	Document contains the final product specification.

This document applies to the following products:

Product name	Type number	Firmware version	PCN / IN
EVK-EMMY-W161	EVK-EMMY-W161-A-01	SDIO-UART: Package: SD-WLAN-UART-BT-NFC-8887-U14-MMC-15.68.7.p73-15.28.7.p73-C3X15148_A2-GPL Firmware version: WiFi: 15.68.7.p73; BT 15.28.7.p73 Driver version: C3X15148	N/A
EVK-EMMY-W163	EVK-EMMY-W163-A-01	SDIO-SDIO: Package: SD-WLAN-SD-BT-FM-NFC-8887-KK44_LINUX_3_10_33_M-PXA1908-15.68.7.p71-15.29.7.p71-C3X15146_A2-GPL Firmware version: 15.68.7.p71-15.29.7.p71 Driver version: C3X15146	N/A

u-blox reserves all rights to this document and the information contained herein. Products, names, logos and designs described herein may in whole or in part be subject to intellectual property rights. Reproduction, use, modification or disclosure to third parties of this document or any part thereof without the express permission of u-blox is strictly prohibited.

The information contained herein is provided "as is" and u-blox assumes no liability for the use of the information. No warranty, either express or implied, is given, including but not limited, with respect to the accuracy, correctness, reliability and fitness for a particular purpose of the information. This document may be revised by u-blox at any time. For most recent documents, please visit www.u-blox.com.

Copyright © 2016, u-blox AG

u-blox® is a registered trademark of u-blox Holding AG in the EU and other countries.

Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. PCI, PCI Express, PCIe, and PCI-SIG are trademarks or registered trademarks of PCI-SIG. All other registered trademarks or trademarks mentioned in this document are property of their respective owners.

Contents

Contents	3
1 Introduction	5
1.1 Overview	5
1.2 Schematic.....	7
1.3 Host interface operation modes.....	8
1.4 Jumper settings for the operation mode SDIO-UART.....	8
1.5 Jumper settings for the operation mode SDIO-SDIO.....	10
1.6 Changing the antenna path.....	12
2 Software	13
2.1 Available software packages.....	13
2.2 Marvell version scheme.....	13
2.3 Structure of the SDIO-UART software package	14
2.4 Structure of the SDIO-SDIO software package	15
2.5 Wi-Fi driver architecture.....	16
2.6 Bluetooth driver architecture	17
2.7 Patching the driver.....	18
2.8 Compiling the drivers.....	19
2.9 Deploying the drivers	21
2.10 Blacklist open source drivers.....	21
3 Runtime usage	23
3.1 Firmware boot options	23
3.2 Wi-Fi through SDIO interface.....	23
3.3 Bluetooth through UART interface.....	24
3.4 Bluetooth through SDIO interface.....	25
3.5 NFC through SDIO.....	25
3.6 Verifying the firmware version	26
3.7 Verifying the interfaces.....	27
3.8 Access point with provided Marvell tools	27
3.9 Station mode.....	28
4 Driver debugging	29
4.1 Compile-time debug options	29
4.2 Runtime debug options	29
5 Known issues	30
Appendix	31
A List of acronyms	31
Related documents	32

Revision history.....32

Contact.....33

1 Introduction

This document describes how to set up the EVK-EMMY-W1 evaluation kit (EVK-EMMY-W161 or the EVK-EMMY-W163) to evaluate the EMMY-W1 host-based multiradio modules with Wi-Fi, Bluetooth and NFC. It also provides a technical overview and describes how to compile the reference drivers that are developed and provided by Marvell.



The open source driver mwifiex, which is in the main development line of the Linux kernel is not explained in this document.

1.1 Overview

The EVK-EMMY-W1 evaluation kit includes an evaluation board (EVB-EMMY-W1), which can be used as a reference design for the EMMY-W1 series modules, and an SDIO adapter. The EVB-EMMY-W1 has a separate NFC antenna as well as an onboard Wi-Fi and an onboard Bluetooth antenna. The evaluation kit also offers a standard full sized SDIO connector for host communication. The main features of the EVK-EMMY-W1 are:

- Provides an external connector to all host interfaces (SDIO and UART) including the SDIO adapter
- Provides PCM interface through an on board accessible connector
- Has GPIO pins that are accessible through an on board connector
- Provides a matched antenna solution for NFC connectivity
- Has a dual frequency antenna included as well as a single band antenna (EVK-EMMY-W163 only)
- Allows configuration of different voltage sources

Table 1 lists the different evaluation kit versions:

Evaluation kit	Description	Suitable for evaluation of
EVK-EMMY-W161	Evaluation kit for the module variant with 1 antenna pin and an internal LTE filter; uses the EMMY-W161 and EMMY-W161-A. When no co-located LTE device is present in the evaluation setup, this kit can also be used to evaluate the module variant with 1 antenna pin and without internal LTE filter (EMMY-W165 and EMMY-W165-A)	EMMY-W161, EMMY-W161-A EMMY-W165, EMMY-W165-A
EVK-EMMY-W163	Evaluation kit for versions with 2 antenna pins; uses the EMMY-W163 and EMMY-W163-A modules	EMMY-W163, EMMY-W163-A

Table 1: List of available EVK-EMMY-W1 evaluation kits



The evaluation of the EMMY-W165 and EMMY-W165-A modules with the EVK-EMMY-W161 are valid only when NOT co-located with an LTE device.

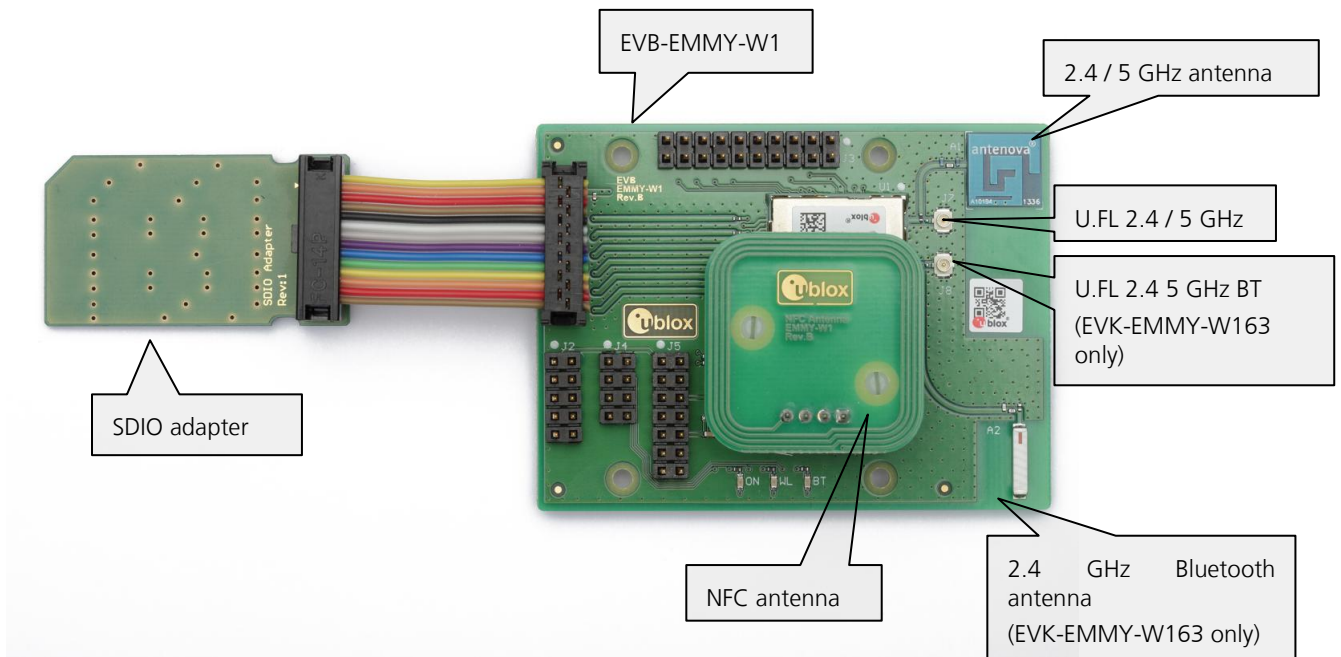


Figure 1: Components of EVK-EMMY-W1 evaluation kit



The 2.4 GHz antenna is used for Bluetooth on the EVK-EMMY-W163 only.

The EVK-EMMY-W1 evaluation kit is based on Marvell Avastar 88W8887 chipset revision A2 [1]. The EVK-EMMY-W1 evaluation kit can be powered through one of the following connectors:

- An SDIO (SD_3V3@J1) interface.
- A 3.3 V (Vin_3V3@J5) from an external power supply

The SDIO interface option is used in this document.

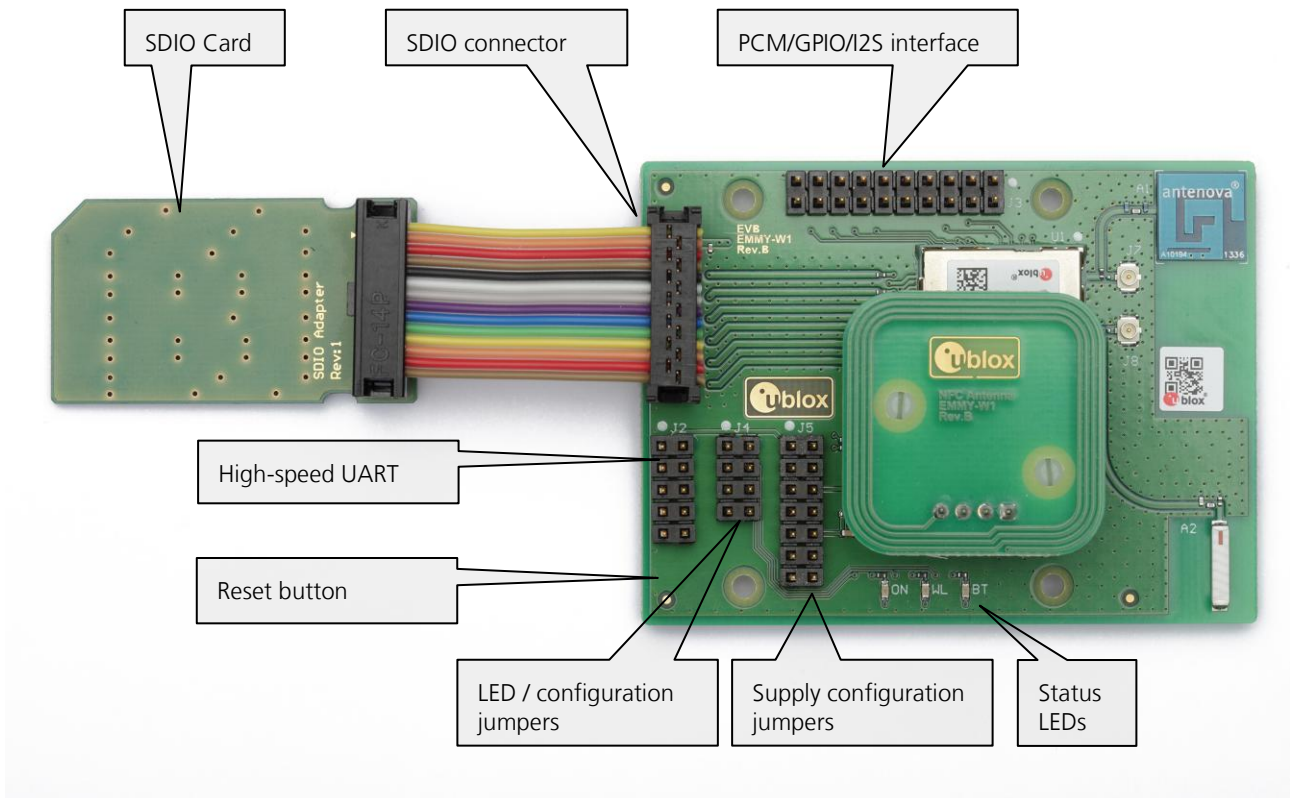


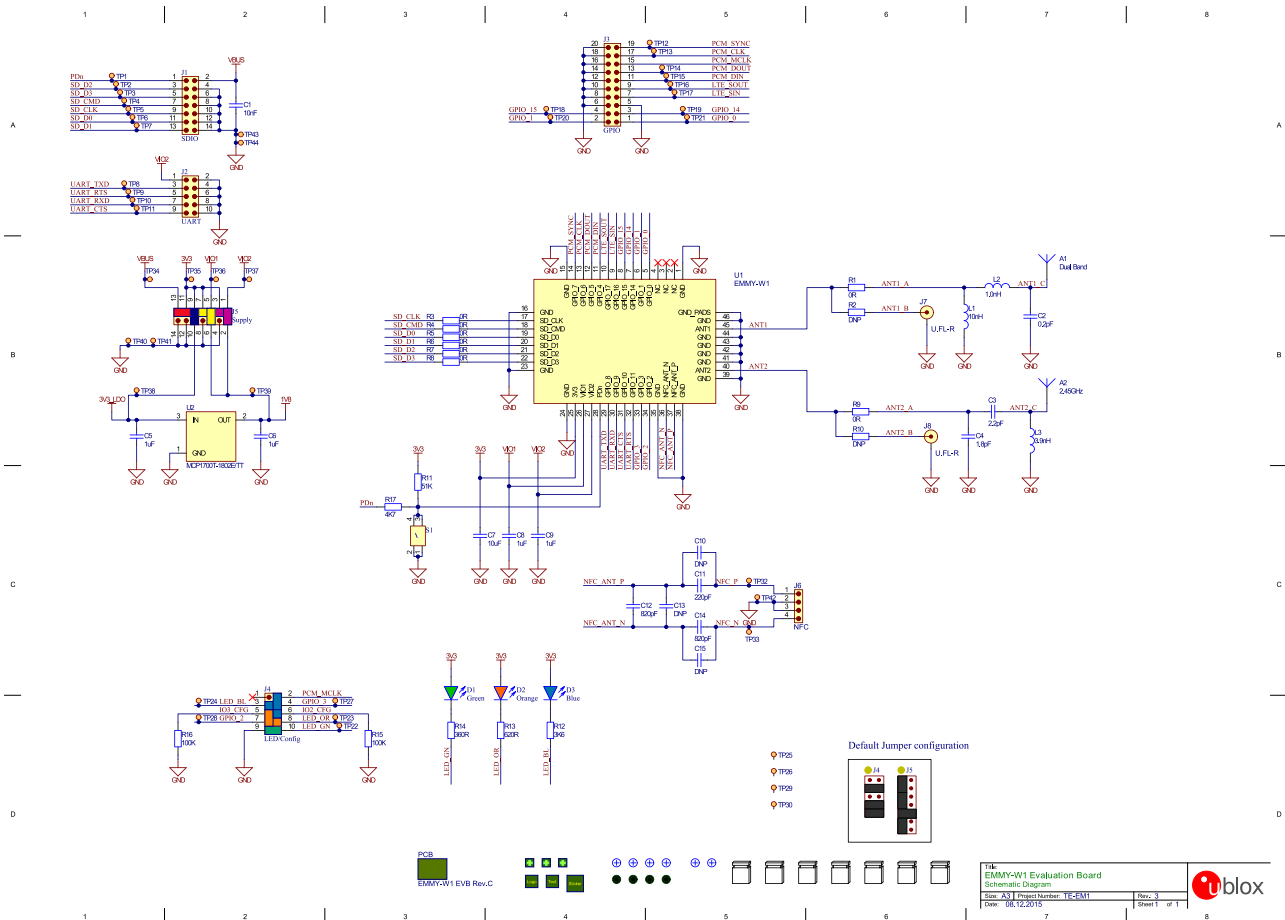
Figure 2: Overview of EVK-EMMY-W1 evaluation kit

- ⚠ The RESET Button and LED / Configuration jumpers might slightly differ from the above-mentioned picture. This will be updated in the future.

1.2 Schematic

The schematic of the EVK-EMMY-W1 is shown in Figure 3.

- ⚠ The second antenna path (ANT2 / PIN 40) will only be used for the EVK-EMMY-W163 variant. For the EVK-EMMY-W161, the antenna path (ANT1 / PIN 45) will be shared for Wi-Fi and Bluetooth traffic.


Figure 3: Schematic of EVK-EMMY-W1

1.3 Host interface operation modes

The EMMY-W1 module series can be operated with the following interfaces to the host:

1. **SDIO-UART** mode: Commands and data regarding to the Wi-Fi traffic will be transferred via the SDIO Bus to the module. For traffic that is in the domain of Bluetooth or NFC, only the high-speed UART interface can be used.
2. **SDIO-SDIO** mode: Wi-Fi traffic as well as traffic in the Bluetooth or NFC domain will be transferred via the SDIO interface.

The operation modes can be configured with the jumper group J4. The important use cases will be shown in the following sections. See section 1.2 for additional information.

1.4 Jumper settings for the operation mode SDIO-UART

All jumper settings are documented in the schematic, which is shown in section 1.2. You can configure the operation mode using the jumpers of the connector J4. See section 3.1 for more detailed description about the operation mode. The configuration group J5 can be used to configure the supply source of the interface signal level (3.3 V or 1.8 V) for the SDIO voltage domain (VIO1) and the IO voltage domain (VIO2).

1.4.1 Signal level 3.3V

In Figure 4, the jumper settings for the operation mode SDIO-UART with a signal level of 3.3 V is shown. The supply voltage will be provided from the SDIO bus (VBUS). The signal level for SDIO (VIO1) is configured to 3.3 V and the signal level for UART (VIO2) has been configured to 3.3 V.

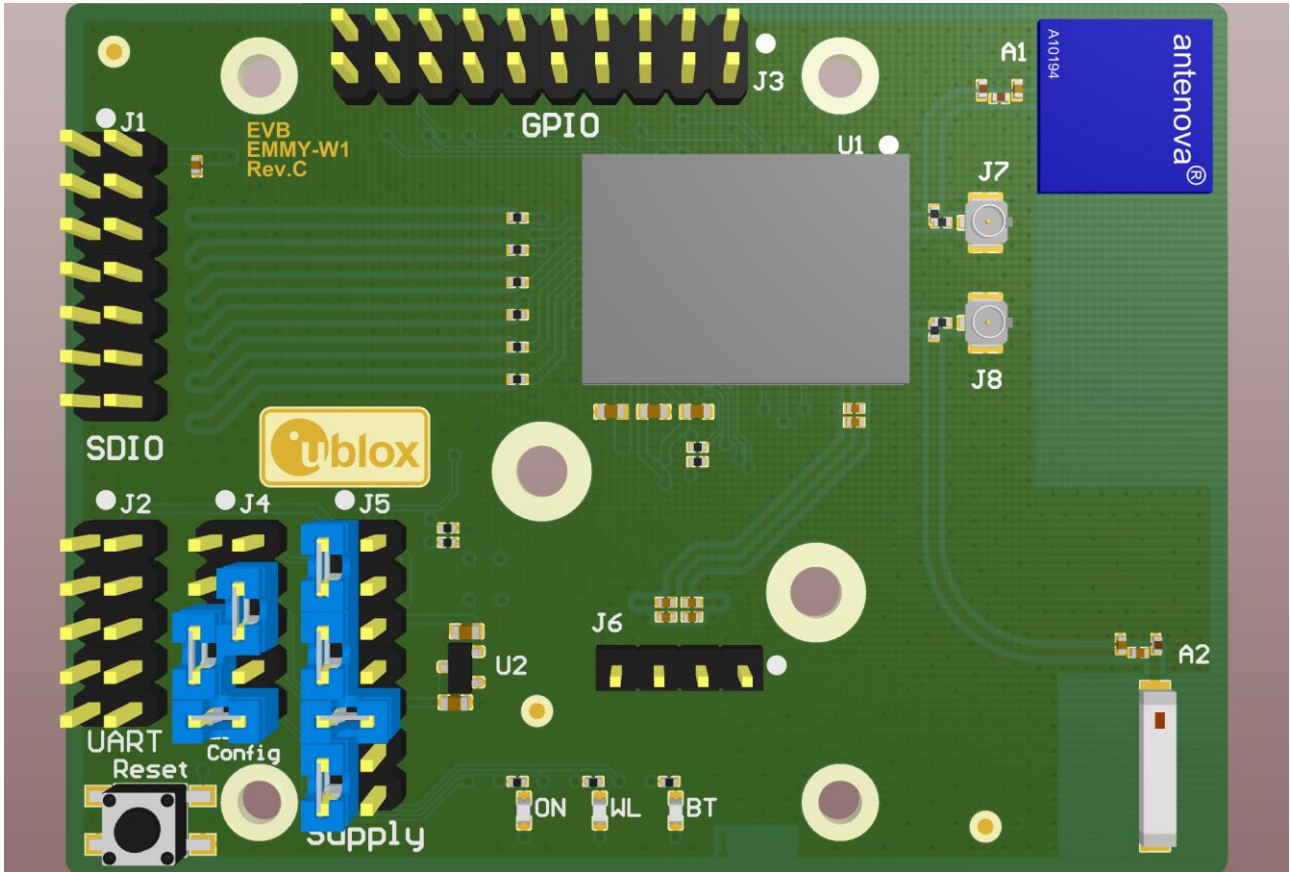


Figure 4: Jumper settings of the EVK-EMMY-W1 for *SDIO-UART* operation mode with 3.3V signal level.

1.4.2 Signal level 1.8V

In Figure 5, the jumper settings for the operation mode *SDIO-UART* with a signal level of 1.8 V is shown. The supply voltage will be provided from the *SDIO* bus (*VBUS*). The signal level for *SDIO* (*VIO1*) is configured to 1.8 V and the signal level for *UART* (*VIO2*) has been configured to 1.8 V. The 1.8 V is provided from the on board LDO (*U2*).

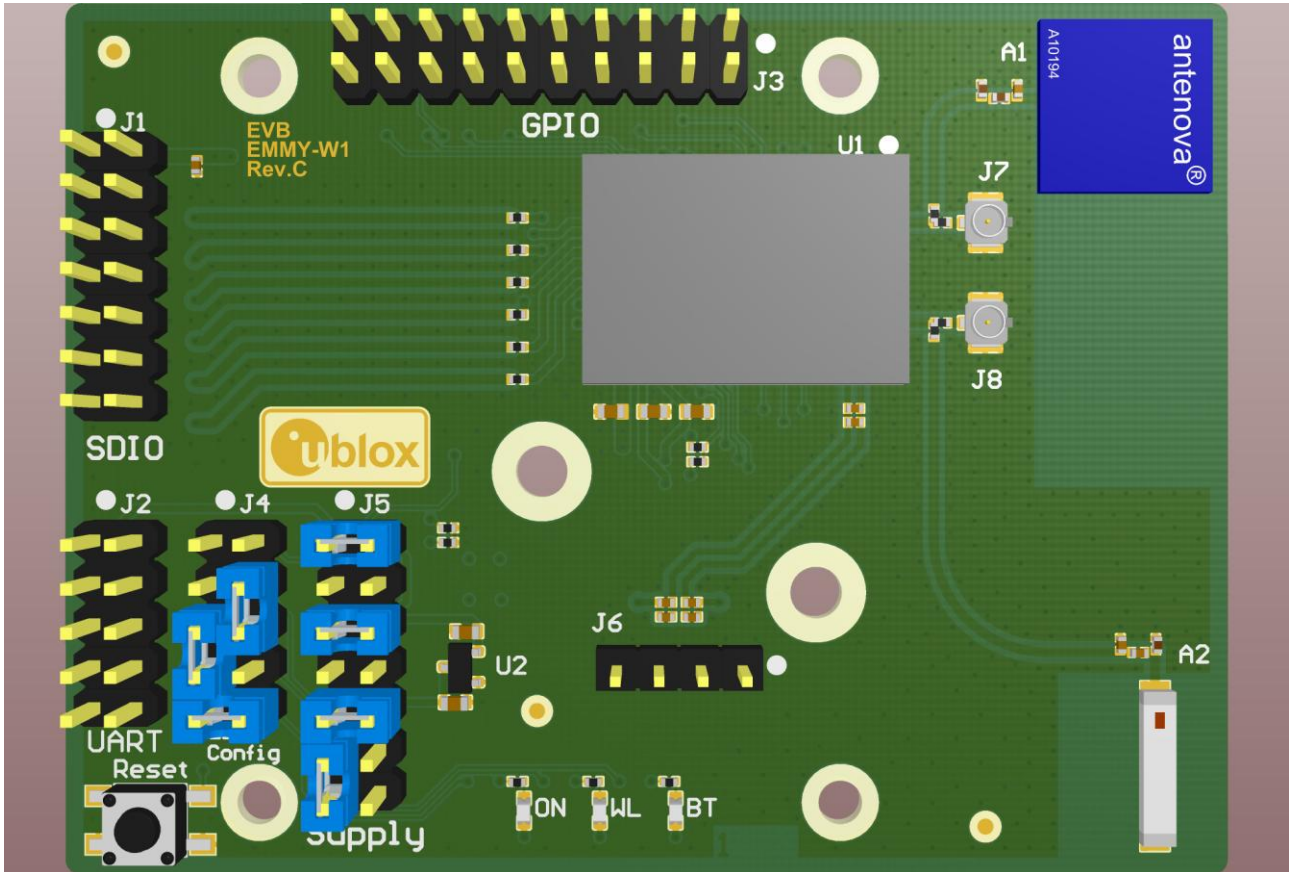


Figure 5 Jumper settings of the EVK-EMMY-W1 for *SDIO-UART* operation mode with 1.8 V signal level

1.5 Jumper settings for the operation mode SDIO-SDIO

All jumper settings are documented in section 1.2 - Schematics. You can configure the operation mode using the jumpers of the connector J4. The configuration group J5 can be used to configure the supply source of the interface signal level (3.3 V or 1.8 V).

1.5.1 Signal level 3.3 V

The jumper settings for the operation mode SDIO-SDIO with a signal level of 3.3 V is shown in Figure 6. The supply voltage will be provided from the SDIO bus (VBUS). The signal level for SDIO (VIO1) is configured to 3.3 V and the signal level for UART (VIO2) has been configured to 3.3 V.

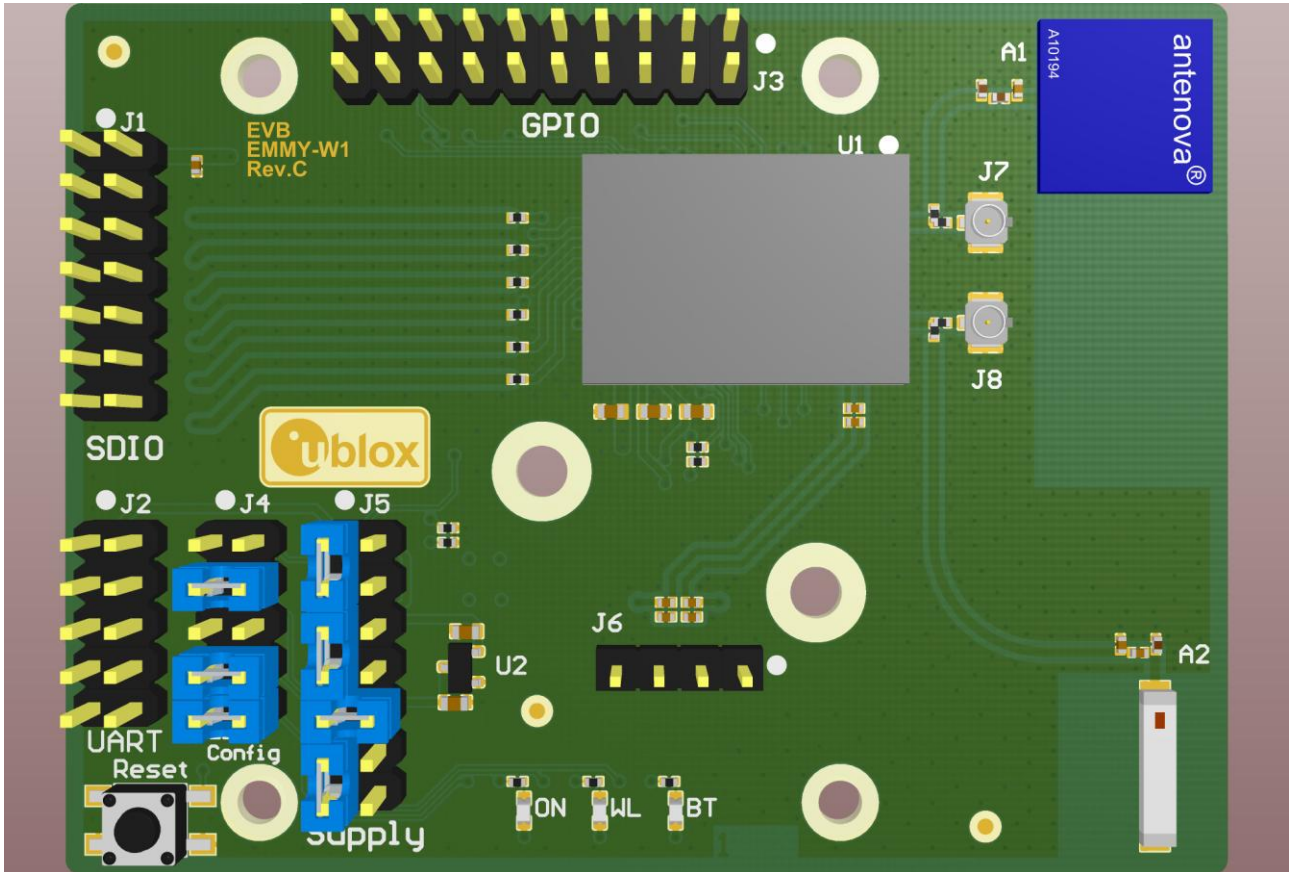


Figure 6: Jumper settings of the EVK-EMMY-W1 for *SDIO-SDIO* operation mode with 3.3V signal level.

1.5.2 Signal level 1.8V

In Figure 7 the jumper settings for the operation mode *SDIO-SDIO* with a signal level of 1.8V is shown. The supply voltage will be provided from the *SDIO* bus (VBUS). The signal level for *SDIO* (VIO1) is configured to 1.8V and the signal level for *UART* (VIO2) has been configured to 1.8V. The 1.8V is provided from the on board LDO (U2).

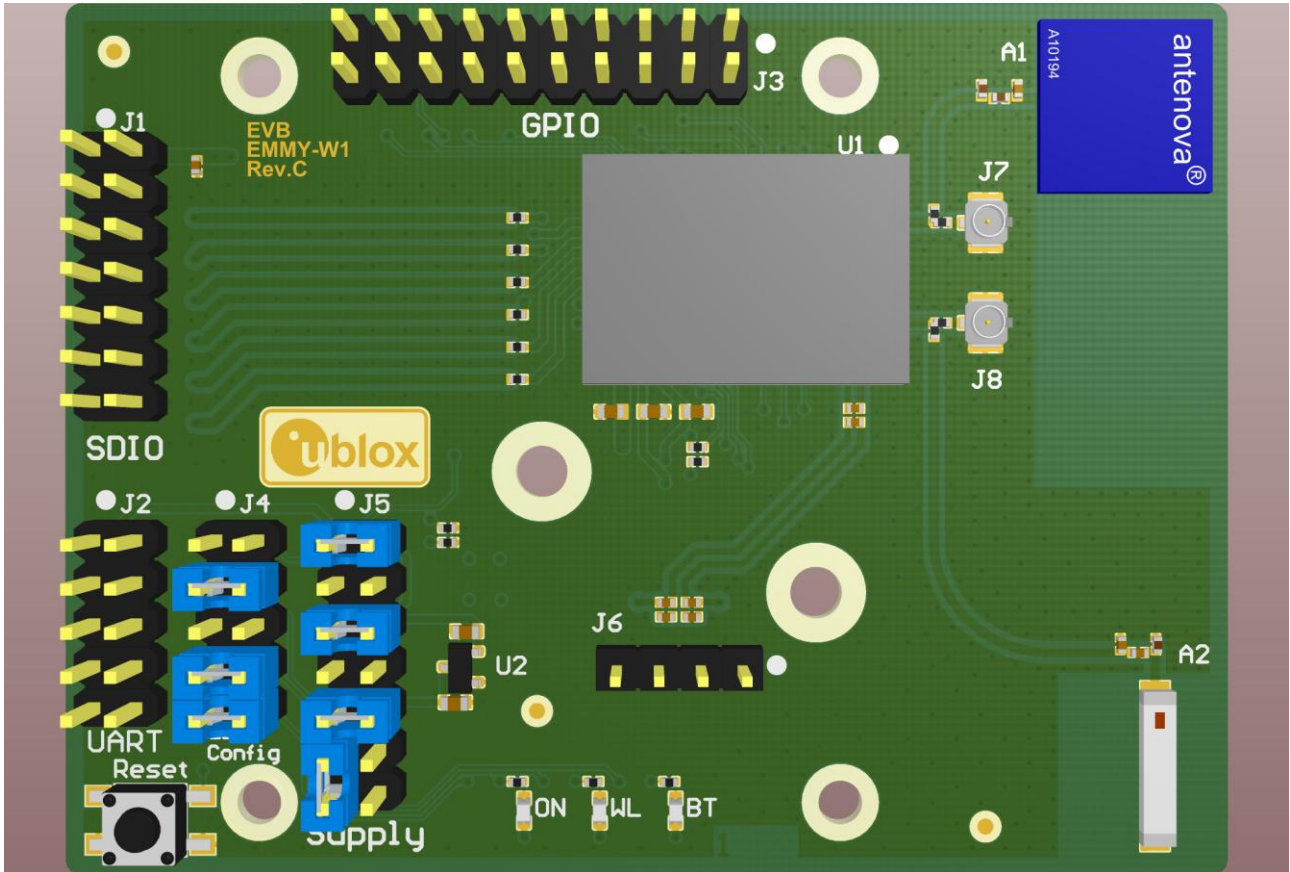


Figure 7 Jumper settings of the EVK-EMMY-W1 for SDIO-SDIO operation mode with 1.8 V signal level

1.6 Changing the antenna path

The EVK-EMMY-W1 evaluation kit includes a dual band (Wi-Fi only on EVK-EMMY-W163 and Wi-Fi/BT on EVK-EMMY-W161 variant) and a single band chip antenna (used only for EVK-EMMY-W163). The Wi-Fi and the Bluetooth antenna path can be changed for conducted measurements and to connect external antennas via the U.FL connectors J7 and J8 (see Figure 1).

To enable the antenna path to the J7 and J8 connectors, the resistors R1 and R9 must be moved. To enable the J7 connector, the R1 resistor has to be moved to R2. The resistor R9 has to be moved to R10 to enable the J8 connector (valid for EVK-EMMY-W163 only).

See section 1.2 - Schematic for detailed information.

2 Software

The reference driver developed by Marvell is distributed by u-blox to those customers who have signed the limited use license agreement (LULA) [2] with u-blox. A valid non-disclosure agreement (NDA) is mandatory if you have obtained the driver package from Marvell directly.


In this section, the different driver packages that are provided are described. This section also provides information on how these driver packages are structured and how they can be compiled.

As previously mentioned, the EMMY-W1 series modules can be operated with the operation modes SDIO-UART and SDIO-SDIO (see Section 1.3). Each operation mode needs to use a dedicated host driver. Two variants of the software packages will be provided.

2.1 Available software packages

The software packages have been developed by Marvell. We distinguish between two variants of host drivers, the variant SDIO-UART and the variant SDIO-SDIO. All types of modules, either EMMY-W161 or EMMY-W163, can be operated with both variants of host drivers. Both variants of host drivers are explained below:

- The SDIO-UART reference driver is packaged and provided with the name “SD-WLAN-UART-BT-NFC-8887-U14-MMC-15.68.7.p73-15.28.7.p73-C3X15148_A2-GPL”. The SDIO-UART reference driver provides a reference implementation of a Wi-Fi and Bluetooth driver with NFC capabilities and some example applications, which show how to configure the EMMY-W1 series module. These drivers and applications are released for Linux only though they can be ported to android based systems as well.

 **The *SDIO-UART* software package does not include a driver to access Bluetooth through the SDIO interface. It includes a driver for accessing Bluetooth (and NFC) through the high-speed UART interface only. For accessing the Bluetooth part of the EMMY-W1 module through the SDIO interface, the Bluetooth kernel module provided by the SDIO-SDIO software package can be used for automotive applications.**

- Use the software package “SD-WLAN-SD-BT-FM-NFC-8887-KK44_LINUX_3_10_33_M-PXA1908-15.68.7.p71-15.29.7.p71-C3X15146_A2-GPL” for SDIO-SDIO software package usage. This reference implementation is released for an Android based system. It might be possible to use these drivers in Linux environments as well. But changes to the source code might be necessary.

 The open source driver (mwifiex), which is distributed with the Linux kernel sources, is not reviewed by u-blox for the EMMY-W1 series module.

2.2 Marvell version scheme

Each released driver package from Marvell follows a specific version scheme. For instance, the release with the string SD-WLAN-UART-BT-NFC-8887-U14-MMC-15.68.7.p73-15.28.7.p73-C3X15148_A2-GPL has been released for the SOC Version 88W8887 [3]. This includes:

- A binary firmware in the version WiFi:15.68.7.p73; BT 15.28.7.p73.
- A driver package in the version C3X15146.

This version scheme applies for both SDIO_UART and SDIO-SDIO software packages.

2.2.1 Driver version

- C : Indicates Marvell OS independent driver
- 3.X : Indicates support for kernel version 3.x
- Release Number: This number tracks the incremental changes in the consequent driver releases given to QA or customers.
- Patch Number: Customers may want to receive a driver build based on a previous release plus specific bug fixes, or patches. Such requests are made normally before production. The patch number starts at zero (no patch), and increments on release of subsequent builds with more bug fixes.

2.2.2 Firmware version

Following is an explanation of each digit in the versioning scheme designed for the firmware:

- Major Revision (first number from the left): Tracks the main firmware version.
- Minor Revision (second number from the left): Tracks the chip family, firmware branch, custom projects and so on.
- Release Number (third number from the left): Tracks the incremental changes in the consequent firmware releases given to QA or customers.
- Patch Number (fourth number from the left): Customers may want to receive a firmware build based on a previous release plus specific bug fixes, or patches. Such requests are made normally before production. The patch number starts at zero (no patch), and increments on release of subsequent builds with more bug fixes.

2.3 Structure of the SDIO-UART software package

The compressed file can be unpacked using the following command:

```
unzip "SD-WLAN-UART-BT-NFC-8887-U14-MMC-15.68.7.p73-15.28.7.p73-C3X15148_A2-GPL.zip" \
&& tar *.tar && for i in $(ls *.tgz) ; do tar xpf $i; done
```

After extracting multiple recursive compressed files, the file system structure should be as shown below:

```
.
├── app_uart8887d                ← firmware uploader for mfg
├── Cal-Data_Conf_files
│   ├── wlanCalData_ext_AG2.conf
│   ├── wlanCalData_ext_AG2_DUAL_ANT.conf
│   ├── wlanCalData_ext_AQB.conf
│   ├── wlanCalData_ext_AQB_DUAL_ANT.conf
│   ├── wlanCalData_ext_CAC_AG1.conf
│   ├── wlanCalData_ext_CSP_TB.conf
│   └── wlanCalData_ext_QFN_TB.conf
├── FwImage
│   ├── release_FM
│   ├── release_NFC
│   ├── sd8887_wlan_a2.bin
│   ├── sduart8887_uapsta_a2.bin
│   └── uart8887_bt_a2.bin
├── SD-WLAN-UART-BT-NFC-8887-U14-MMC-15.68.7.p73-15.28.7.p73-C3X15148_A2-GPL
│   ├── muart_src                ← hci_uart kernel driver(part of linux)
│   │   └── include
│   └── wlan_src                 ← kernel driver for WiFi (sd8887 mlan)
│       ├── mapp
│       ├── mlan
│       ├── mlinux
│       └── script
```

2.3.1 Kernel modules

The software package provides source for the following three kernel modules:

- mlan (mlan.ko)
- sd8xxx (sd8xxx.ko)
- hci_uart (hci_uart.ko)

The hci_uart is used for accessing the Bluetooth or NFC functions of the EVK-EMMY-W1 and has no dependencies. Marvell's Wi-Fi implementation is spread between the modules sd8887 and mlan. The latter

implements the chip specific functions and is intended to be independent from the operating system (OS). The kernel module sd8887 implements the OS specific bindings. The sd8887 handles the standard interfaces from the OS such as the network interface and also manages to load the firmware to the EMMY-W1 during the initialization phase.

⚠ The Bluetooth driver (hci-uart) is also part of the Linux kernel distribution. Marvell adds additional functionality to this kernel module such as NFC support. It is recommended to use the provided hci_uart module instead of the module that is distributed with the Linux kernel.

2.3.2 Firmware

The host controller must load the firmware to the EMMY-W1 module during initialization. Every driver package includes a binary form of the firmware. The firmware image includes a part for Wi-Fi and another for Bluetooth. The kernel module for Wi-Fi or Bluetooth can load the firmware. These modules can detect an EMMY-W1 module, which runs on a firmware. In this case, the driver will skip loading the firmware and moves on. As the EMMY-W1 is built with the chipset of the revision a2, the image sduart8887_uapsta_a2.bin should be used.

2.3.3 Example applications

Marvell provides the following example applications:

- mlanutl - Configures the additional parameters available for Marvell mdriver
- uaputl.exe - This tool can be used to set/get micro-AP settings. To change the access point (AP) settings, use "bss_stop" command to stop the AP before making change and "bss_start" command to restart the AP after making the change.

2.4 Structure of the SDIO-SDIO software package

The compressed file can be unpacked using the following command:

```
unzip "SD-WLAN-SD-BT-FM-NFC-8887-KK44_LINUX_3_10_33_M-PXA1908-15.68.7.p71-15.29.7.p71-C3X15146_A2-GPL.zip"
\
&& tar *.ta
```

After extracting multiple recursive compressed files, the file system structure should be as shown below:

```
.
|-- FwImage
|   |-- release_FM
|   |-- release_NFC
|   |-- sd8887_bt_a2.bin
|   |-- sd8887_uapsta_a2.bin
|   `-- sd8887_wlan_a2.bin
`-- SD-WLAN-SD-BT-FM-NFC-8887-KK44_LINUX_3_10_33_M-PXA1908-15.68.7.p71-15.29.7.p71-C3X15146_A2-GPL
    |-- mbt_src
    |-- mbtc_src
    `-- wlan_src
```

2.4.1 Kernel modules

The software package provides source for the following kernel modules:

- mlan (mlan.ko) – OS independent module for Wi-Fi
- sd8xxx (sd8xxx.ko) – OS dependent module for Wi-Fi (SDIO)
- bt8xxx (bt8xxx.ko) – Bluetooth HCI interface via SDIO
- mbt8xxx (mbt8xxx.ko) – Alternative Bluetooth character device

The SDIO-SDIO software package provides the same Wi-Fi reference driver as in the SDIO-UART software package. Marvell's Wi-Fi implementation is spread between the modules sd8xxx and mlan. The latter implements the chip specific functions and is intended to be independent from the operating system (OS). The kernel module

sd8xxx implements the OS specific bindings. The sd8xxx handles the standard interfaces from the OS such as the network interface and also manages to load the firmware to the module during initialization.

Another driver for Bluetooth is also available in the SDIO-SDIO software package reference driver (mbtc_src), which is not bound to bluez, instead, it exports a character device that can be used by third-party user space Bluetooth stacks.

2.4.2 Firmware

The host controller must load the firmware to the EMMY-W1 series module during initialization. Every driver package includes a binary form of the firmware. The firmware image includes a part for Wi-Fi and another part for Bluetooth. The kernel module for Wi-Fi or Bluetooth can load the firmware. These modules can detect an EMMY-W1 module, which is loaded with a valid firmware. In this case, the driver will skip loading the firmware again. As the EMMY-W1 is built with the chipset of the revision a2, the image sd8xxx_uapsta_a2.bin should be used.

2.4.3 Example applications

Marvell provides the following example applications:

- mlanutl - Configures the additional parameters available for Marvell mdriver
- uaputl.exe - This tool can be used to set/get micro-AP settings. To change the access point (AP) settings, use "bss_stop" command to stop the AP before making change and "bss_start" command to restart the AP after making the change.

2.5 Wi-Fi driver architecture

The provided package (SDIO-UART and SDIO-SDIO software packages) includes a dedicated Wi-Fi driver.

The figure below describes the basic architecture of the reference driver. The example applications mentioned in subsection 2.3.3 enables communication amongst the driver interfaces. The driver provides a wireless extension (Wext) and a Netlink-based interface for cfg80211.

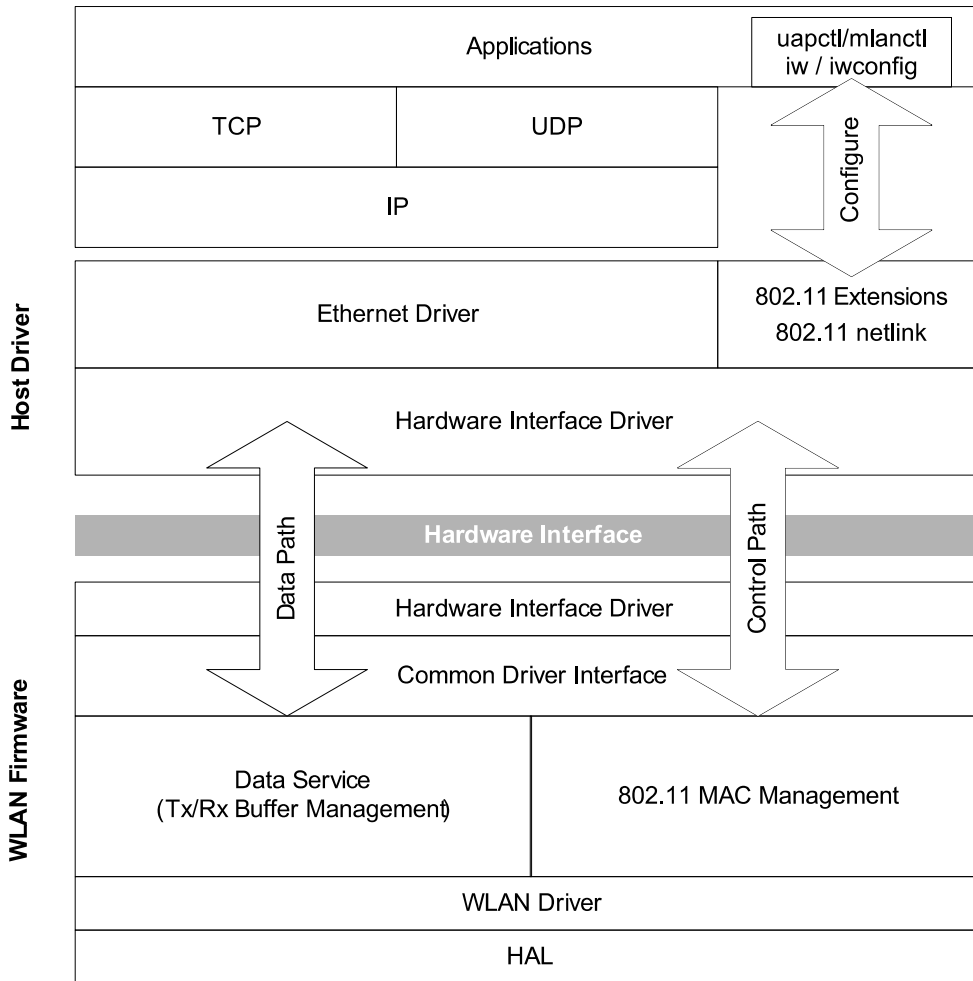


Figure 8 Basic Wi-Fi host driver and firmware architecture

2.6 Bluetooth driver architecture

The standard Bluetooth protocol stack in Linux is provided by bluez. The SDIO-SDIO software package reference driver provides a Bluetooth driver for the EMMY-W1 series module. This Bluetooth driver is a client driver that runs on top of the MMC/SDIO bus driver and performs the following:

- Forwards the data and commands between upper protocol stack layers and the firmware
- Handles some private commands that are used to handshake between the driver and firmware only.

The architecture of the Bluetooth driver is shown in the following figure. The host system can access the EMMY-W1 module either through the SDIO or UART interface depending on the chosen operation mode (see section 2.1).

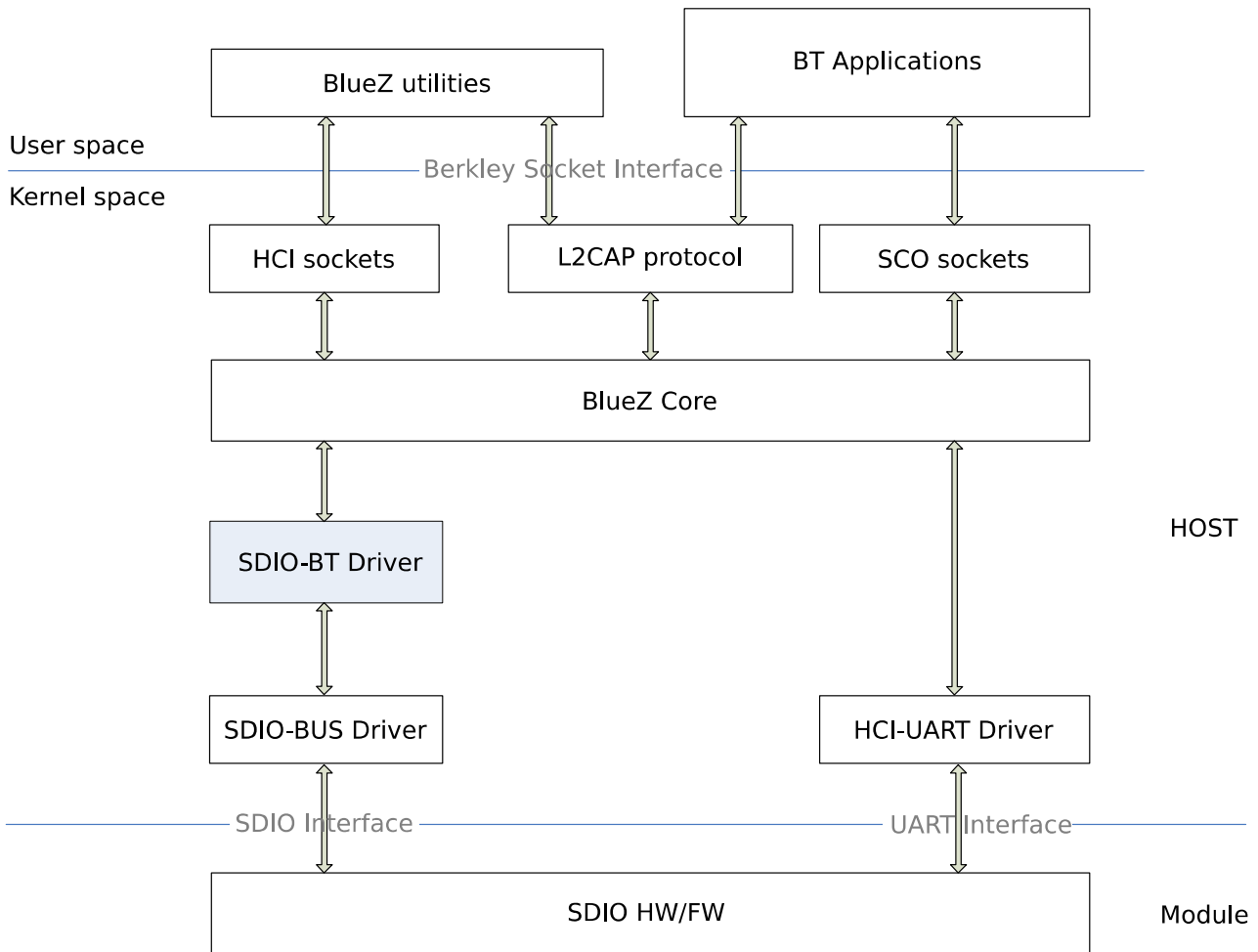


Figure 9 Bluetooth driver and protocol stack

Another driver for Bluetooth is also available in the reference driver package, which is not bound to bluez; instead it exports a character device that can be used by third-party user space Bluetooth stacks.

2.7 Patching the driver

The delivered driver package requires calibration data to be specified as a module parameter. Since EMMY-W1 modules are calibrated during production and all necessary calibration data is stored to OTP on the module, it is not necessary to download calibration data from the host via the driver. To prevent the driver package from stopping during the initialization phase, a patch must be applied to the kernel module sources (see Listing 1). Do not specify the calibration data while loading the driver.

The EMMY-W1 series module is calibrated during the production test. The calibration is done for each module and is unique for each module. All the necessary calibration data is stored persistent into the one time programmable memory (OTP).

```

From bc9848a16f5c3b99f33ec0d145b262911a46de6b Mon Sep 17 00:00:00 2001
From: ublox
Date: Thu, 12 Nov 2015 10:27:37 +0100
Subject: [PATCH] enable 8887 fw dpc

Omit downloading external calibration data, read data stored on the module
(EEPROM or OTP) instead.
---
 wlan_src/mlinux/moal_main.c | 4 ----
 1 file changed, 4 deletions(-)

diff --git a/wlan_src/mlinux/moal_main.c b/wlan_src/mlinux/moal_main.c
index e7fdbc1..3b667d4 100755
--- a/wlan_src/mlinux/moal_main.c
+++ b/wlan_src/mlinux/moal_main.c
@@ -2241,10 +2241,6 @@ woal_init_fw_dpc(moal_handle *handle)
                                     goto done;
                                     }
                                     }
-    } else if (!cal_data_cfg) {
-        PRINTM(MERROR, "Please add cal_data_cfg for 8887\n");
-        ret = MLAN_STATUS_FAILURE;
-        goto done;
-    }
    if (handle->user_data) {
        param.pcal_data_buf = (t_u8 *)handle->user_data->data;
--
2.1.4

```

Listing 1 Patch for disabling the forced loading of calibration data via the driver, 0001-enable-8887-fw-dpc.patch

The patch can be applied from the root directory of the driver package using the following command:

```
$ patch -p1 < 0001-enable-8887-fw-dpc.patch
```

2.8 Compiling the drivers

The following description applies to both Wi-Fi and Bluetooth drivers. The Wi-Fi driver and tools are located in the `wlan_src` directory.

The compilation process can be invoked with the `make` utility as described in the excerpt below. The variables `KERNELDIR` and `CROSS_COMPILE` have to be modified according to the used environment.

```
$ make -e MAKEFLAGS= KERNELDIR=/usr/src/kernel CROSS_COMPILE=${CROSS_COMPILE} CROSS=${CROSS_COMPILE}
build
```

The results are the kernel modules and binaries of the example application. They will be stored in the directory `bin_sd8xxx` (for the Wi-Fi driver) or `bin_muart` (for the `hci_uart` driver) for the SDIO-SDIO software package reference driver as summarized in the below-mentioned excerpt.

```

SD-WLAN-UART-BT-NFC-8887-U14-MMC-15.68.7.p73-15.28.7.p73-C3X15148_A2-GPL
├─ bin_muart
│   ├── hci_uart.ko
│   └─ README
├─ bin_sd8xxx
│   ├── config
│   │   ├── 11n_2040coex.conf
│   │   ├── 80211d_domain.conf
│   │   └─ ...
│   ├── load
│   ├── mlan2040coex
│   ├── mlanevent.exe
│   ├── mlan.ko
│   ├── mlanutl
│   ├── README
│   ├── README_MLAN
│   ├── README_RBC
│   ├── README_UAP
│   ├── README_WIFIDIRECT
│   ├── sd8xxx.ko
│   ├── uaputl.exe
│   ├── unload
│   ├── wifidirect
│   │   ├── start_auto_go.sh
│   │   ├── start_find_phase.sh
│   │   ├── start_listen_state.sh
│   │   ├── stop_auto_go.sh
│   │   ├── stop_wifidirect_client.sh
│   │   └─ update_mac.sh
│   ├── wifidirectutl
│   └─ wifidisplay
│       ├── start_auto_go.sh
│       ├── start_find_phase.sh
│       └─ update_mac.sh
├─ muart_src
└─ wlan_src
    
```

Listing 2 Build results of the SDIO-UART software package reference driver



For a proper usage of the `hci_uart` kernel module, it is recommended to enable the following features in the kernel configuration:

```

CONFIG_BT_HCIUART=m
CONFIG_BT_HCIUART_H4=y
CONFIG_BT_HCIUART_BCSP=y
CONFIG_BT_HCIUART_ATH3K=y
CONFIG_BT_HCIUART_LL=y
CONFIG_BT_HCIUART_3WIRE=y
CONFIG_BT_HCIBCM203X=y
CONFIG_BT_HCIBPA10X=y
CONFIG_BT_HCIBFUSB=y
    
```

2.9 Deploying the drivers

The drivers can be deployed on reasonable locations in the target root file system. The following excerpt is shown as an example:

```

/
├── etc
│   ├── modprobe.d
│   │   └── emmy-w1-driver-sd8887.conf
├── lib
│   ├── firmware
│   │   └── mrvl
│   │       ├── sd8887_uapsta_a0.bin
│   │       ├── sd8887_uapsta_a2.bin
│   │       ├── sd8887_wlan_a2.bin
│   │       └── uart8887_bt_a2.bin
│   └── modules
│       ├── 3.17.4
│       └── updates
├── opt
│   └── emmy-w1
│       ├── bin
│       │   ├── config -> ../etc
│       │   ├── mlan2040coex
│       │   ├── mlanevent
│       │   ├── mlanutl
│       │   ├── uaputl
│       │   ├── wifidirect -> ../lib/wifidirect
│       │   ├── wifidirectutl
│       │   └── wifidisplay -> ../lib/wifidisplay
│       ├── etc
│       │   └── *.conf
│       ├── lib
│       │   ├── wifidirect
│       │   └── wifidisplay
├── LICENCE.Marvell
├── usr
├── share
├── doc
├── README
├── README_MLAN
├── README_RBC
├── README_UAP
└── README_WIFIDIRECT
    
```

2.10 Blacklist open source drivers

Since the Linux kernel distributes an open source implementation of the reference driver implementation provided by Marvell, both implementation are in conflict with each other. It cannot be decided which driver will be loaded from the kernel because both register for the same device.

To prevent loading, the open source kernel modules `mwifiex`, `mwifiex_sdio`, `btmrv1` and `btmrv1_sdio` can be blacklisted. The following excerpt could be added to the `/etc/modprobe.conf` respective `/etc/modprobe.d/` configuration files:

```
blacklist mwifiex  
blacklist mwifiex_sdio  
blacklist btmv1  
blacklist btmv1_sdio
```

3 Runtime usage

This section describes how to load the specific drivers and establish an access point or connect to an access point with the provided tools.

3.1 Firmware boot options

The EMMY-W1 module can be used with different interfaces. The specific interfaces that should be used for different parts of the EMMY-W1 can be decided through the firmware boot options. The boot options can be defined with the jumpers on J4 (GPIO_2 and GPIO_3). These options have no hardware impact. The firmware reads the configuration and boots accordingly.

GPIO [3:2]	Wi-Fi	BT/BLE/NFC	Firmware Download	Firmware Download Mode	SDIO Functions
00	SDIO	UART	SDIO	Serial	Wi-Fi
01	SDIO	SDIO	SDIO	Parallel	Wi-Fi, Bluetooth and NFC
10	SDIO	UART	SDIO+UART	Parallel	Wi-Fi
11(default)	SDIO	SDIO	SDIO	Serial	Wi-Fi Bluetooth and NFC

Table 2: EVK-EMMY-W1 boot options

A serial firmware download mode refers to the operation where the firmware image for Wi-Fi and Bluetooth is downloaded at once. For the parallel firmware download mode, both dedicated host drivers are responsible to download its own part of the firmware. For example, for the configuration 00 (see Table 2), the firmware can only be downloaded to the EMMY-W1 module through the Wi-Fi driver via the SDIO interface. This configuration is used in this document for the description of the SDIO-UART software package.


3.2 Wi-Fi through SDIO interface

By default, the kernel module sd8xxx loads the firmware from `mrsl/sd8xxx_uapsta.bin`. As the EMMY-W1 series module is equipped with the chip revision a2, it is mandatory to select a different firmware while loading the modules. The driver expects the firmware relative to the path `/lib/firmware`.

```
# insmod mlan.ko && insmod sd8887.ko fw_name=mrsl/sd8887_uapsta_a2.bin cfg80211_wext=0xf
[410849.009430] wlan: Loading MWLAN driver
[410849.014397] wlan: Driver loaded successfully
root@:/# lsmod
Module                Size    Used by
sd8xxx                397563  0
mlan                  312099  1 sd8xxx
$ make -e MAKEFLAGS= KERNELDIR=/usr/src/kernel CROSS_COMPILE=${CROSS_COMPILE} CROSS=${CROSS_COMPILE} build
```

The EVK-EMMY-W1 is attached by inserting the SDIO adapter. The kernel module sd8xxx will perform an automatic firmware download to the module.

```
[415132.619458] mmc1: new high speed SDIO card at address 0001
[415132.637015] vendor=0x02DF device=0x9135 class="0" function=1
[415132.648238] SDIO: max_segs=1024 max_seg_size=33553920
[415132.653724] rx_work=0 cpu_num=1
[415133.449321] wlan: FW download over, firmwarelen=631900 downloaded 631900
[415133.955808] WLAN FW is active
[415133.974657] fw_cap_info=0xff03, dev_cap_mask=0xffffffff
```

 **It is not possible to load more than one reference driver of the same kind from Marvell simultaneously. For instance, the Wi-Fi reference driver of the SDIO-SDIO software package will collide with the reference driver of the SDIO-UART software package.**

The kernel modules can be unloaded using the following commands:

```
# rmmmod sd8xxx wlan
[415240.024054] wlan: Unloading MWLAN driver
[415240.032803] wlan: Driver unloaded
```

3.3 Bluetooth through UART interface



This functionality is provided by the SDIO-UART software package (see section 1) only.



This section expects that the EVK-EMMY-W1 board has already been connected to the host and the Wi-Fi drivers have been loaded according to section 3.2, which has initiated the download of the firmware. The EVK-EMMY-W1 board is also expected to be configured as described in section 1.4.

The module for Bluetooth communication through the UART interface provided with the SDIO-UART software package can be loaded using the commands mentioned below:

```
# insmod ./hci_uart.ko
[ 103.508917] Bluetooth: HCI UART driver ver 2.2
[ 103.513730] Bluetooth: HCI H4 protocol initialized
[ 103.518814] Bluetooth: HCI BCSP protocol initialized
[ 103.524056] Bluetooth: HCILL protocol initialized
[ 103.529022] Bluetooth: HCIATH3K protocol initialized
[ 103.534257] Bluetooth: HCI Three-wire UART (H5) protocol initialized
```



The `hci_uart.ko` kernel module is originated and distributed with the kernel sources. This means it might be possible, that the feature has been already included in the kernel. In that case, it is not possible to load the module provided by Marvell. Either the feature has to be configured as a module and must not be loaded or the feature has to be disabled. The configuration of the running kernel shall include `CONFIG_BT_HCIUART=m` and must not include `CONFIG_BT_HCIUART=y` (see section 2.7).

It is necessary to bind a serial interface to the HCI driver. For this, the `hciattach` tool from the bluez package can be used. The following excerpt shows an example of how to attach to the EMMY-W1 through the `/dev/tty02` serial device. The option `-s` specifies an initial speed instead of the default.

```
# hciattach /dev/tty02 any 3000000 flow
Device setup complete
```

Then, an hci interface (here `hci0`) is available:

```
# hciconfig -a
hci0:          Type: BR/EDR  Bus: UART
              BD Address: 00:06:C6:FF:AD:75  ACL MTU: 1021:7  SCO MTU: 120:6
              UP RUNNING PSCAN
              RX bytes:694 acl:0 sco:0 events:40 errors:0
              TX bytes:1000 acl:0 sco:0 commands:40 errors:0
              Features: 0xff 0xfe 0x8f 0xfe 0xdb 0xff 0x7b 0x87
              Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
              Link policy: RSWITCH HOLD SNIFF
              Link mode: SLAVE ACCEPT
              Name: 'elin-w160-evk-0'
              Class: 0x400100
              Service Classes: Telephony
              Device Class: Computer, Uncategorized
              HCI Version: 4.0 (0x6)  Revision: 0x8300
              LMP Version: (0x7)  Subversion: 0x530a
              Manufacturer: Marvell Technology Group Ltd. (72)
```


3.4 Bluetooth through SDIO interface



This functionality is provided by the SDIO-SDIO software package only (see section 1.6).

The standard Bluetooth protocol stack in Linux is provided by bluez. The SDIO-SDIO software package provides a Bluetooth driver for the EMMY-W1 series module. This Bluetooth driver is a client driver that runs on top of the MMC/SDIO bus driver and performs the following:

- Forwards the data and commands between upper protocol stack layers and the firmware
- Handles some private commands that are used as handshake between the driver and firmware only.

On loading, it registers with the bus driver, downloads the firmware if not already loaded by the Wi-Fi driver, and registers a new HCI device with the bluez stack.

To load the Bluetooth driver:

```
$ modprobe bt8887
$ dmesg
[ 1171.576539] BT: Loading driver
[ 1171.576611] BT: Driver loaded successfully
[ 1197.139206] mmc0: new SDIO card at address 0001
[ 1197.253645] sdio_bt mmc0:0001:2: firmware: direct-loading firmware mrvl/sd8887_uapsta_a2.bin
[ 1198.048389] BT: FW download over, size 642664 bytes
[ 1198.545260] BT FW is active(5)
[ 1198.558402] BT: Create /dev/mbtchar0
[ 1198.612113] BT: Create /dev/mfmchar0
[ 1198.640096] BT: Create /dev/mnfcchar0
[ 1198.668103] BT: Create /dev/mdebugchar0
```

This will register a new Bluetooth device (hci0 in this case). The firmware download will be skipped if the Wi-Fi driver has already been loaded.

```
$ hciconfig
hci0:          Type: BR/EDR  Bus: SDIO
              BD Address: 00:06:C6:46:DF:7B  ACL MTU: 1021:6  SCO MTU: 120:6
              UP RUNNING PSCAN
              RX bytes:656 acl:0 sco:0 events:28 errors:0
              TX bytes:986 acl:0 sco:0 commands:28 errors:0
```

3.5 NFC through SDIO



This functionality is provided by the SDIO-SDIO software package only (see section 1.6).

The Bluetooth driver provides access to the Bluetooth feature of the module as described in section 3.4. Additionally, that driver also manages the communication with the NFC part of the firmware. The driver provides an NCI interface (character device /dev/mnfcchar0). See the following excerpt:

```
$ modprobe bt8887
$ dmesg
...
[ 1198.545260] BT FW is active(5)
[ 1198.558402] BT: Create /dev/mbtchar0
[ 1198.612113] BT: Create /dev/mfmchar0
[ 1198.640096] BT: Create /dev/mnfcchar0
[ 1198.668103] BT: Create /dev/mdebugchar0
```

The `mnfcchar0` character device acts as an NFC controller interface (nci) [4] endpoint. NCI commands will be encapsulated via the HCI protocol. Marvell enhances the NCI protocol with proprietary commands [5].

Additionally, Marvell provides a shared library MRVL NCI. MRVL NCI is a shared library that implements the NCI specification [6]. It handles the connection between the DH (running MRVL NCI) and NFCC.

MRVL NCI is designed to be OS independent and to work on all available host interfaces (transport layer). Currently, the OS abstraction layer is implemented only for Linux.

The source code [6] provides the demonstration applications `demo_reader(1)`, `demo_text(2)` and `demo_bt_oob(3)`. The necessary modifications to use these applications with the `mnfcchar0` character device are shown in the following excerpt:

```
diff --git a/demo_reader.c b/demo_reader.c
index f9cc265..19ca00d 100644
--- a/demo_reader.c
+++ b/demo_reader.c
@@ -69,9 +69,9 @@ int main(void)
     .hci_mux      = true,                // NCI packets in HCI BT packets
     .u            = {
         .tty = {
-         .dev      = "/dev/ttyUSB1",      // TTY
+         .dev      = "/dev/mnfcchar0",    // TTY
         .baudrate  = 115200,             // Baudrate
-         .mode     = MRVL_NCI_TTY_MODE_CHAR,
+         .mode     = MRVL_NCI_TTY_MODE_PACKET,
         }
     }
 };
```

The `demo_reader` application activates the polling mode of the module and provides the raw data on the stdout stream.

3.6 Verifying the firmware version

The version of the loaded driver can be verified using the following commands:

```
# iwpriv wlan0 version
wlan0    version:SD8887-15.68.7.p72-C3X15C147-GPL-(FP68)
# /opt/emmy-w1/bin/mlanutl wlan0 version
Version string received: SD8887-15.68.7.p72-C3X15147-GPL-(FP68)
```

3.7 Verifying the interfaces

You can verify whether all the interfaces are available using the following commands:

```
# ifconfig -a
mlan0      Link encap:Ethernet  HWaddr 00:06:C6:FF:AD:0D
           BROADCAST MULTICAST  MTU:1500  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0          collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
uap0      Link encap:Ethernet  HWaddr 00:06:C6:FF:AD:0D
           BROADCAST MULTICAST  MTU:1500  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0          collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
wfd0      Link encap:Ethernet  HWaddr 02:06:C6:FF:AD:0D
           BROADCAST MULTICAST  MTU:1500  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0          collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

3.8 Access point with provided Marvell tools



This section expects that the EVK-EMMY-W1 module has been connected to the host and the Wi-Fi drivers have been loaded according to section 3.1.

In the following scenario, an access point is created using the example configuration tool provided by Marvell. The access point is configured with an SSID value - "EMMY" for 40 MHz bandwidth and with the passphrase "12345678" for WPA2 based encryption. The description of the commands is provided in the README_uap and README_Mlan.

To set up the SSID and the encryption mechanisms:

```
./uaputl.exe sys_cfg_ssid EMMY
./uaputl.exe sys_cfg_auth 0
./uaputl.exe sys_cfg_protocol 32 #sets WPA2 protocol
./uaputl.exe sys_cfg_wpa_passphrase 12345678
./uaputl.exe sys_cfg_cipher 8 8 #PAIRWISE_CIPHER:AES CCMP GROUP_CIPHER:AES CCMP
```

To set up the RF parameters:

```
./uaputl.exe sys_cfg_channel 48 4      #Set AP primary radio channel to 48, and
# secondary channel is below.
./uaputl.exe sys_cfg_2040_coex 1      #enable 20/40 BSS coexistence Configuration
./uaputl.exe sys_cfg_11n 1 0x116e 3 0 0xff
./uaputl.exe sys_cfg_rates 0xc 0x12 0x18 0x24 0x30 0x48 0x60 0x6c
```

To set up the regulation domain:

```
./uaputl.exe sys_cfg_80211d state 1 country DE
```

To start the BSS:

```
./uaputl.exe bss_start
```

To assign an IP to the interface:

```
ifconfig uap0 192.168.1.1
```

Additionally, the usage of a DHCP server on the interface is recommended.

3.9 Station mode

3.9.1 Through wpa_supplicant

```
# cat > /etc/wpa_supplicant.conf << EOF
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1
EOF
```

To set wireless network settings such as SSID "EMMY" and the passphrase "12345678":

```
# wpa_passphrase <ssid> <passphrase> >> /etc/wpa_supplicant.conf
```

To run wpa_supplicant daemon:

```
# wpa_supplicant -B -D nl80211 -i wlan0 -c /etc/wpa_supplicant.conf
```

To acquire an IP address:

```
# udhcpc -i wlan0
```

3.9.2 Through the provided Marvell example tools

You can also connect to an AP using Marvell's example configuration tools:

```
# ./mланut1 wlan0 countrycode DE
# ./mланut1 wlan0 bandcfg 84 # set for 802.11a/an/ac
# ./mланut1 wlan0 passphrase "1;ssid=<ssid>;passphrase=<sharedkey>"
# iwconfig wlan0 essid <ssid>
# udhcpc -i wlan0
```

4 Driver debugging

Driver debugging is provided through the kernel print function `printk` and the `proc` file system. The driver states are recorded and can be retrieved through the `proc` file system during runtime. The following files containing the debug information are provided (the actual location is dependent on the Linux kernel version):

- `/proc/mwlan/config` or `/proc/net/mwlan/config`
- `/proc/mwlan/mlanX/info` or `/proc/net/mwlan/mlanX/info`
- `/proc/mwlan/mlanX/debug` or `/proc/net/mwlan/mlanX/debug`



`mlanX` is the name of the device node created at runtime. Other possibilities are `uapX` and `wfdX` for the access point and Wi-Fi Direct interfaces respectively.

The debug messages are also printed to the kernel ring buffer through `printk` calls. These messages can be accessed using the `/proc/kmsg` interface or by the `dmesg` command. Alternatively, this can also be handled by advanced logging facilities.

4.1 Compile-time debug options

The extent to which the debug messages are available to be printed at runtime is controlled by the `CONFIG_DEBUG` variable in the driver's Makefile. The `CONFIG_DEBUG` variable can have the following values:

- `n`: debug messages are disabled and not compiled into the driver module
- `1`: all kinds of debug messages can be configured except for `MENTRY`, `MWARN` and `MINFO`. By default, `MMSG`, `MFATAL` and `MERROR` are enabled.
- `2`: all kinds of debug messages can be configured

4.2 Runtime debug options

Once debugging is enabled in the Makefile, the debug messages can be selectively enabled or disabled at runtime by setting or clearing the corresponding bits of the `drvdbg` parameter:

```

bit 0: MMSG          PRINTM(MMSG,...)
bit 1: MFATAL       PRINTM(MFATAL,...)
bit 2: MERROR       PRINTM(MERROR,...)
bit 3: MDATA        PRINTM(MDATA,...)
bit 4: MCMND        PRINTM(MCMND,...)
bit 5: MEVENT       PRINTM(MEVENT,...)
bit 6: MINTR        PRINTM(MINTR,...)
bit 7: MIOCTL       PRINTM(MIOCTL,...)
...
bit 16: MDAT_D      PRINTM(MDAT_D,...), DBG_HEXDUMP(MDAT_D,...)
bit 17: MCMD_D      PRINTM(MCMD_D,...), DBG_HEXDUMP(MCMD_D,...)
bit 18: MEVT_D      PRINTM(MEVT_D,...), DBG_HEXDUMP(MEVT_D,...)
bit 19: MFW_D       PRINTM(MFW_D,...),  DBG_HEXDUMP(MFW_D,...)
bit 20: MIF_D       PRINTM(MIF_D,...),  DBG_HEXDUMP(MIF_D,...)
...
bit 28: MENTRY      PRINTM(MENTRY,...), ENTER(), LEAVE()
bit 29: MWARN       PRINTM(MWARN,...)
bit 30: MINFO       PRINTM(MINFO,...)
    
```

The value of `drvdbg` can be given as a module parameter when the driver is loaded, by writing to the `proc` file system's `debug` file or by setting it through the `iwpriv` or `mlanutl` tool.

```

iwpriv wlan0 drvdbg          # Get the current driver debug mask
iwpriv wlan0 drvdbg 0       # Disable all debug messages
echo "drvdbg=0x7" > /proc/mwlan/mlan0/debug # enable MMSG, MFATAL and MERROR
mlanutl wlan0 drvdbg -1    # Enable all debug messages
    
```

5 Known issues

- No known issues

Appendix

A List of acronyms

Abbreviation / Term	Explanation / Definition
BOM	Bill of Materials
BSS	Basic Service Set
DH	Device Host
DHCP	Dynamic Host Configuration Protocol
FM	Frequency Modulation
IO	Input-Output
LDO	Low-Dropout
MFG	Manufacturing support, a special firmware with test support
NCI	NFC controller interface
NFC	Near Field Communication
NFCC	NFC Controller
OS	Operating System
PCM	Pulse Code Modulation
QA	Quality Assurance
SDIO	Secure Digital Input Output
SSID	Service Set Identifier
uAP	Micro Access Point
UART	Universal Asynchronous Receiver-Transmitter
Wext	Wireless Extensions
WLAN	Wireless Local Area Network
WPA2	Wi-Fi Protected Access II

Related documents

- [1] EMMY-W1 series Data sheet, Document Number UBX-15011785
- [2] Marvell software Limited Use License Agreement
- [3] Avastar 88W8887 website - <http://www.marvell.com/wireless/88W8887/>
- [4] NFC Controller Interface (NCI) Specification; NFC Forum™; NFCForum-TS-NCI-1.0; 2012-11-06
- [5] NFC Controller Interface – Software Specification; Doc. No. MV-S800898-00; May 16, 2014, 2.00; Marvell.
- [6] mrvl_nci-5f5d74db07d96dce467410a38d6e131abd270ccc.tar.bz2; Marvell

Revision history

Revision	Date	Name	Status / Comments
R01	26-May-2015	shoe	Initial release
R02	5-Apr-2016	shoe, lalb	Document status changed to Advance Information. Updated to new revision of the evaluation kit (type numbers EVK-EMMY-W161-A-01 and EVK-EMMY-W163-A-01) with new design of the evaluation board. Updated the firmware version that this document applies to in page 2. Added section 1.3 - Host interface operation modes, section 2.10 - Blacklist open source drivers and section 3.5 - NFC through SDIO. Removed a few resolved issues from chapter 5 - Known issues.
R03	09-May-2016	shoe	Updated the baud rate to 3M baud in the example in section 3.3. Updated the firmware versions used in the examples.
R04	23-Nov-2016	shoe, kgom	Changed the document status to Early Production Information. Added patch description to disable forcing of calibration data from the driver (section 2.7). Updated driver versions 15.68.7.p73-15.28.7.p73-C3X15148; 15.68.7.p71-15.29.7.p71-C3X15146. Added information for evaluation of the new module version EMMY-W165 and EMMY-W165-A in section 1.1. Added information about antenna path configuration for J7 and J8 U.FL connectors (section 1.6).

Contact

For complete contact information visit us at www.u-blox.com.

u-blox Offices

North, Central and South America

u-blox America, Inc.

Phone: +1 703 483 3180
E-mail: info_us@u-blox.com

Regional Office West Coast:

Phone: +1 408 573 3640
E-mail: info_us@u-blox.com

Technical Support:

Phone: +1 703 483 3185
E-mail: support_us@u-blox.com

Headquarters

Europe, Middle East, Africa

u-blox AG

Phone: +41 44 722 74 44
E-mail: info@u-blox.com
Support: support@u-blox.com

Asia, Australia, Pacific

u-blox Singapore Pte. Ltd.

Phone: +65 6734 3811
E-mail: info_ap@u-blox.com
Support: support_ap@u-blox.com

Regional Office Australia:

Phone: +61 2 8448 2016
E-mail: info_anz@u-blox.com
Support: support_ap@u-blox.com

Regional Office China (Beijing):

Phone: +86 10 68 133 545
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Chongqing):

Phone: +86 23 6815 1588
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Shanghai):

Phone: +86 21 6090 4832
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Shenzhen):

Phone: +86 755 8627 1083
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office India:

Phone: +91 80 4050 9200
E-mail: info_in@u-blox.com
Support: support_in@u-blox.com

Regional Office Japan (Osaka):

Phone: +81 6 6941 3660
E-mail: info_jp@u-blox.com
Support: support_jp@u-blox.com

Regional Office Japan (Tokyo):

Phone: +81 3 5775 3850
E-mail: info_jp@u-blox.com
Support: support_jp@u-blox.com

Regional Office Korea:

Phone: +82 2 542 0861
E-mail: info_kr@u-blox.com
Support: support_kr@u-blox.com

Regional Office Taiwan:

Phone: +886 2 2657 1090
E-mail: info_tw@u-blox.com
Support: support_tw@u-blox.com

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for [Interface Development Tools](#) category:

Click to view products by [U-Blox](#) manufacturer:

Other Similar products are found below :

[CY4607M](#) [PEX 8748-CA RDK](#) [DP130DSEVM](#) [DP130SSEVM](#) [ISO3086TEVM-436](#) [SP338EER1-0A-EB](#) [ADM00276](#) [ADM3054WBRWZ-RL7](#)
[ADP5585CP-EVALZ](#) [PEX8724-CA RDK](#) [PEX 8732-CA RDK](#) [PEX8747-CA RDK](#) [PS081-EVA-KIT](#) [CHA2066-99F](#) [AS8650-DB](#) [MLX80104](#)
[TESTINTERFACE I2C-CPEV/NOPB](#) [ISO35TEVM-434](#) [KIT33978EKEVB](#) [416100120-3](#) [XR17D158CV-0A-EVB](#) [XR17V358/SP339-E4-EB](#)
[XR17V358/SP339-E8-EB](#) [XR18910ILEVB](#) [XR22804IL56-0A-EB](#) [ZSC31150KIT V1.2](#) [SCRUBBER-EVM](#) [SI838XISO-KIT](#) [73931-3022](#)
[XIO2200AEVM](#) [XIB-E](#) [XBIB-U-SP](#) [TW-DONGLE-USB](#) [EVAL-ADM2483EBZ](#) [EVAL-ADM2491EEBZ](#) [EVB-USB83340](#) [MAX9921EVKIT](#)
[MAXREFDES23DB#](#) [MAX9291COAXEVKIT#](#) [MAX9286COAXEVKIT#](#) [MAX3535EEVKIT+](#) [MAX3223EEVKIT+](#) [MAX3100EVKIT](#)
[MAX13235EEVKIT](#) [MAX14970EVKIT#](#) [MAX148X1EVKIT#](#) [MAX14826EVKIT#](#) [3298](#) [XR21B1424IV64-0A-EVB](#) [XR21B1421IL24-0A-EVB](#)