WILC Linux Driver Porting Guide for TI Sitara AM437x EVM Board

INTRODUCTION

Microchip's SmartConnect ATWILC1000 is an IEEE 802.11 b/g/n IoT link controller module. The ATWILC1000 connects to any processors with minimal resource requirements with a simple SPI/SDIO-to-Wi-Fi interface. This will indeed reduce the developer's work easy and more focus can be shifted to other areas of development rather than spending time in board bring up. The wi-fi driver support is widely present in the Board Support Package (BSP) releases that comes along with the specific hosts system. Which make the device interfacing with any host platform easier without any dependent factor in between. This user guide explains in detail about interfacing Microchip’s wi-fi device with TI AM437x platform and achieves wireless connectivity to the system.

The AM437x GP EVM is a standalone test, development, and evaluation module system that encompasses an ARM Cortex-A9 32-Bit RISC Microprocessor with Processing Speed up to 1000 MHz. Sitara AM437x processors offer secure boot and cryptographic acceleration features for industrial applications.

ATWILC1000 features a fully integrated Power Amplifier, LNA, Switch and Power Management. The only external clock source needed is a high-speed crystal or oscillator. Which indeed make this as the efficient device to bring Wi-Fi connectivity to system. Texas Instruments’ all Processor SDK Linux releases include the support for the WILC1000 device driver.

System Introduction

This section will describe in detail about the Linux host system platform and the Wi-Fi device that bring the connectivity to the host to run IoT based applications. Before starting any development activity, it is important to gather enough information about the host system and connectivity device. As we discussed before, in this development activity, we will be interfacing Microchip's WILC1000 Wi-Fi device with Texas Instruments AM437x General Purpose (GP) Evaluation module by getting the host up and running with Processor SDK.

AM437x Evaluation Module

AM437x General Purpose (GP) Evaluation module contain an AM4378 processor which comes under Sitara Processor family. The TI AM437x high-performance processors are based on the ARM Cortex-A9 core. The devices support high-level operating systems (HLOS). Linux® is available free of charge from TI.

System View

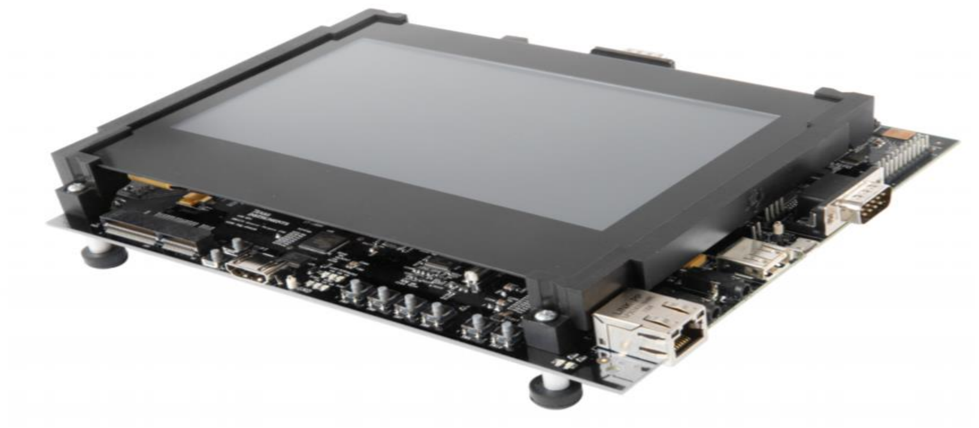


Figure 1. AM437x GP EVM Top View

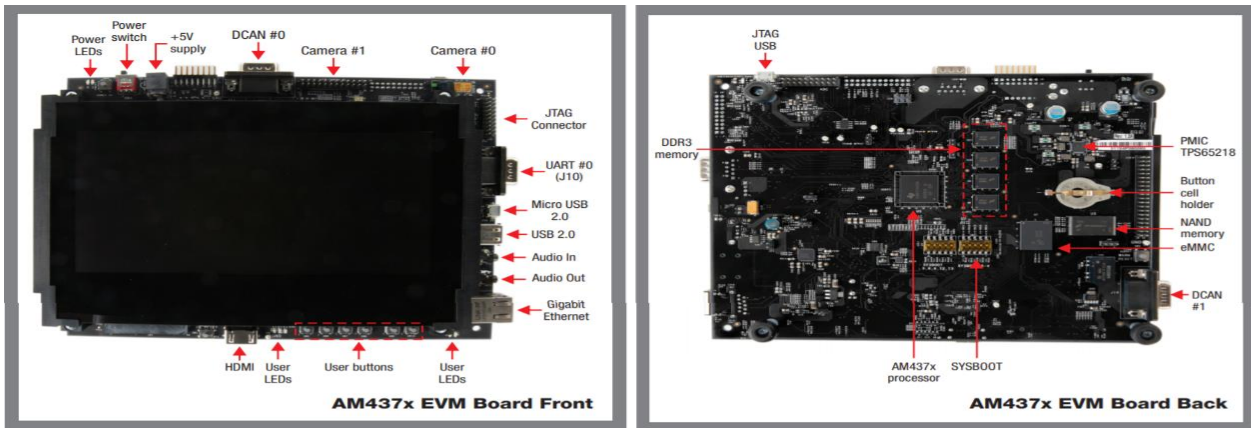


Figure 2. AM437x GP EVM Front and Back view

Features

Hardware	Connectivity
Sitara™ ARM® Cortex®-A9 32-Bit RISC Processor with Processing Speed up to 1000 MHz	Up to Two USB 2.0 High-Speed Dual-Role (Host or Device) Ports With Integrated PHY Up to Two Industrial Gigabit Ethernet MACs (10, 100, and 1000 Mbps)
256KB of On-Chip Boot ROM	Six UARTs, Two McASPs, Five McSPIs, Three I^2C Ports, One QSPI, and One HDQ or 1-Wire
64KB of On-Chip RAM	Micro SD/MMC (1)
7” capacitive touch screen LCD	USB 2.0 OTG/HOST (1/1)
2 Camera Modules	Audio in/out
General-Purpose Memory Support (NAND, NOR, SRAM) Supporting up to 16-Bit ECC	Supports Protocols such as EtherCAT®, PROFIBUS®, PROFINET®, and EtherNet/IP™, EnDat 2.2, and More

Wireless Connectivity Device

ATWILC1000-MR110PB Module

Microchip's SmartConnect ATWILC1000 is an IEEE 802.11 b/g/n IoT link controller module. The ATWILC1000 features a fully integrated Power Amplifier, LNA, Switch and Power Management. The only external clock source needed is a high-speed crystal or oscillator. The ATWILC1000 can be used in Linux, RTOS or Bare metal Environments

Module View

Figure 3. ATWILC1000-MR110PB

Features	IEEE 802.11 b/g/n (1x1)
	Supports Personal & Enterprise IEEE 802.11 WEP, WPA, WPA2 Security
	SPI and SDIO host interfaces
	Wi-Fi Direct, station mode and Soft-AP support
	On-chip memory management engine to reduce host load
	Integrated PA and T/R Switch
	Superior Sensitivity and Range via advanced PHY signal processing
	Operating temperature range of -40C to +85C

ATWILC1000 evaluation kit

ATWILC1000-SD Card

Microchip's ATWILC1000-SD evaluation kit is a hardware platform to evaluate the Microchip's ATWILC1000-MR110PB module. This is a ready to plug in device into the SD Card interface of the host system. This device can be interfaced with the AM437x GP Evaluation board through SDIO interface.

Device View

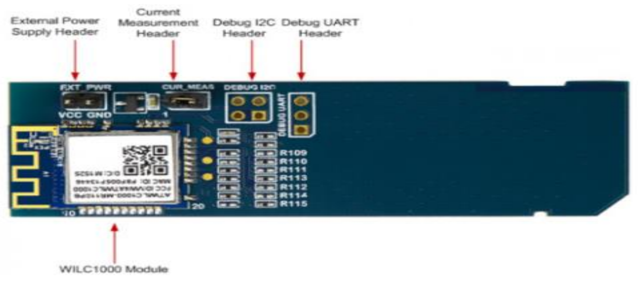


Figure 4. ATWILC1000 SD Card

ATWINC1500-XPRO

The ATWINC1500-XPRO is an extension board to the Xplained Pro evaluation platform. The ATWINC1500-XPRO extension board allows you to evaluate the ATWINC1500 low cost, low power 802.11 b/g/n Wi-Fi network controller module. This device can work as the link controller by itself. The other option is to unmount and replace the WINC1500 module with WILC1000 module. We will be using this device to evaluate the SPI interface of ATWILC device with AM437x GP Evaluation board.

Device View

Figure 5. ATWINC1500 XPRO

References

This section discuses about the necessary documents that are required to add the WILC driver support for the AM437x GP EVM. Before we kick start with the development activity, download these reference documents and keep it handy.

1) Sitara Processor Official web page: http://www.ti.com/product/AM4372
2) Evaluation Module Quick Start Guide: http://www.ti.com/lit/ml/sprw256/sprw256.pdf
3) Datasheet: http://www.ti.com/product/AM4378
4) AM437x Technical Reference Manual: http://www.ti.com/product/AM4378
5) AM437x General Purpose Evaluation Module (EVM) Schematic: http://www.ti.com/general/docs/lit/getliterature.tsp?baseLiteratureNumber=sprr396&fileType=zip
6) AM437x EVM Hardware User Guide: http://www.ti.com/lit/ug/spruhw7/spruhw7.pdf
7) WINC1500 Xplained Pro User guide: http://ww1.microchip.com/downloads/en/DeviceDoc/50002616A.pdf

Interfacing ATWILC with AM437x GP EVM

In this section, we will kick start our development activity by interfacing ATWILC1000 device with AM437x GP EVM board. As we discussed before, ATWILC1000 module offer 2 interfacing option for the module to interface with host system. We will discuss both the interfacing methods in this document. First start with interfacing WILC1000 device to evaluation board using SPI interface.

For this purpose, we will use, the WINC1500 XPRO board. The official web page link to the device: https://www.microchip.com/DevelopmentTools/ProductDetails/ATWINC1500-XPRO

It is recommended to download and keep the WINC1500 Xplained Pro User's Guide handy. The user guide is available at: http://ww1.microchip.com/downloads/en/DeviceDoc/50002616A.pdf

NOTE: The WINC1500 XPRO board contain the WINC1500 module. This device can work as the link controller as it is. The other option is to unmount and replace the WINC1500 module with WILC1000 module.

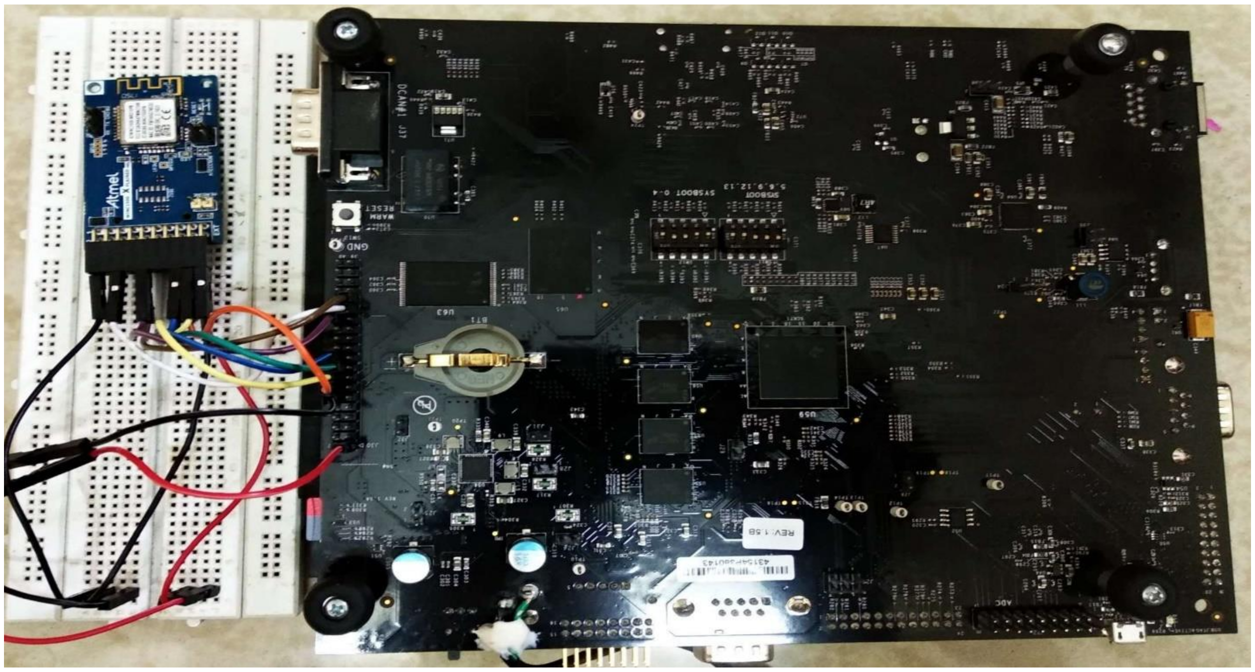


Figure 6. WILC interfaced with AM437x GP EVM

Connecting the WILC1500 Xplained Pro board to AM437x EVM Board

WILC1000 is connected to the AM437x GP EVM using SPI interface. AM437x GP EVM have SPI pins routed to the J30 connector. Here, our first interest is to find the pins allocated for SPI from the Schematics and then identify the pins on the connector.

Schematic for the AM437x GP EVM is available at Design file section: http://www.ti.com/tool/TMDSEVM437X#0

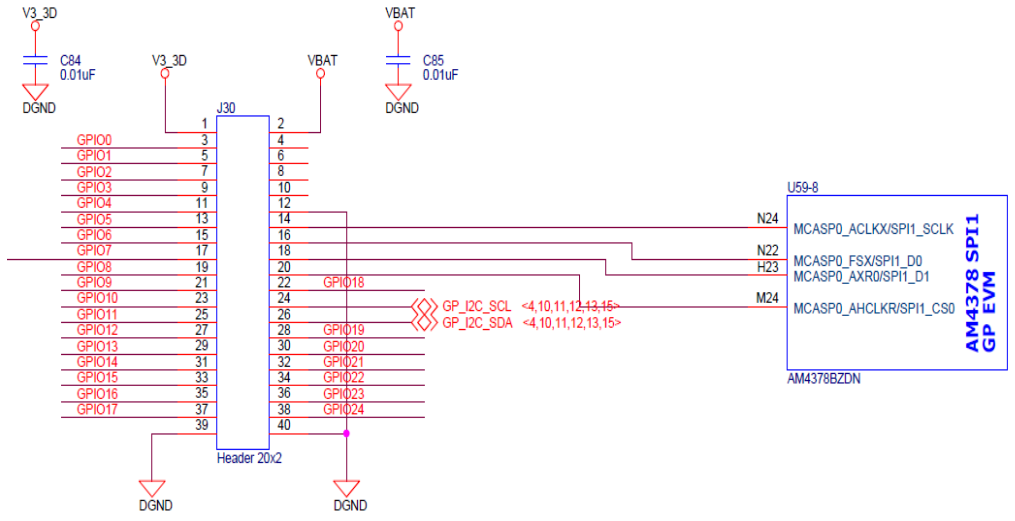


Figure 7. Schematic

Check for the SPI1 pins connected to the J30 header from the AM4378 SPI1. In addition to the SPI pins, we need 3 GPIO pins for CHIP_EN, RESET, IRQ for interfacing WILC1000.

Connect the WINC1500 Xplained Pro board to the AM437x EVM board J30 connector located at the back side of the EVM boar. Refer the WINC1500 Xplained Pro User guide to see the SPI pin mapping.

1) WINC1500 Xplained Pro User guide: http://ww1.microchip.com/downloads/en/DeviceDoc/50002616A.pdf

   Section 3 shows the Xplained Pro Standard Extension Header

2) AM437x EVM Hardware User Guide: http://www.ti.com/lit/ug/spruhw7/spruhw7.pdf

   Section 7.15 show the pins mapped to GPIO Header - J30

Connection Table

AM437x EVM Board		WINC1500 Xplained Pro
PIN Number	Signal	PIN
12	DGND	2
12	DGND	19
1	V3_3D	20
14	SPI1_SCLK	18 (SPI_SCK)
16	SPI1_D0	17(SPI_MOSI)
18	SPI1_D1	16 (SPI_MISO)
20	SPI1_CS0	15 (SPI_SS_A)
33 (GPIO5_10, F23, GPIO15, 33, chip_en)	GPIO	10 (CHIP_EN)
28 (GPIO5_5, R24, GPIO 19, 28, irq)	GPIO	9 (IRQ/GPIO)
31 (GPIO10, G20, GPIO14, 31, reset)	GPIO	5 (RESET_N)

PROCESSOR SDK

Processor SDK (Software Development Kit) is a unified software platform for TI embedded processors providing easy setup and fast out-of-the-box access to benchmarks and demos. Processor SDK which is considered as a starting point to develop an embedded system on a TI Processor running Linux is more than just a typical Board Support Package (BSP) containing a bootloader, Linux kernel, and filesystem. SDK also contains tools for developing on TI Processors (a validated cross-compiling toolchain, for example), pre-built libraries that you can use without having to rebuild them yourself, and some documentation to help explain how all these pieces work together. It wraps all of this up into one simple installer that helps get everything you need in the right place to do development.

Processor SDK v.02.xx includes support for both Linux and TI-RTOS operating systems.

Linux Highlights:

- Long-Term Stable (LTS) Mainline Linux kernel support
- U-Boot bootloader support
- Linaro GNU compiler collection (GCC) tool chains
- Yocto Project™ OE Core compatible file systems

Evaluating the SDK Embedded Linux System

Install the SDK - Within your Linux host machine

Processor SDK Installer is 64-bit and installs only on 64-bit host machine. Support for 32-bit host is dropped as the cross-compiler toolchain is available only for 64-bit machine. At least 20 GB of free space is required on the host machine for installing Processor SDK Linux.

As the first step download and install the Processor SDK in the Linux host machine. Select the Processor SDK link (PROCESSOR-SDK-LINUX-AM437X) which is available under the "order now" section in the mentioned link: http://www.ti.com/tool/TMDSEVM437X

Download the .bin file (ti-processor-sdk-linux-am437x-evm-06.00.00.07-Linux-x86-Install.bin)from the product download section into the Linux host machine.

Install the Linux processor SDK Installer in the Linux host machine

Installing the processor SDK installer in the Linux host machine will extract all the necessary components of the processor SDK into the specified location in the Linux machine.

The Processor SDK Installer (ti-processor-sdk-linux-[platformName]-evm-xx.xx.xx.xx-Linux-x86-Install.bin) will install the necessary components to start your development on the TI microprocessor. The SDK consists of source for the Matrix App launcher starting point application, a development filesystem, a target filesystem, example applications, toolchain and board support package, ease of use scripts and documentation. The Processor SDK also includes the ARM GCC toolchain.

1) Download the installer. It will be a .bin file(ti-processor-sdk-linux-am437x-evm-06.00.00.07-Linux-x86-Install.bin)

2) First change the permission of the file

```
chmod -c 777 ti-processor-sdk-linux-am437x-evm-06.00.00.07-Linux-x86-Install.bin
```

3) Enter into admin mode

```
sudo su
```

4) Run

```
./ti-processor-sdk-linux-am437x-evm-06.00.00.07-Linux-x86-Install.bin
```

This will open an installer window and proceed further by selecting next until finish (like how we install software in windows). This will extract the .bin file and generate a directory in the specified location which contain the SDK package. The SD Card creation script is available inside this directory.

NOTE: The installing will take longer time to complete.

Flashing into SD Card

The SD card you wish to create is inserted into the host Linux system and should be large (16GB or larger) to hold at least the bootloaders, kernel, and root file system.

Before the Linux® OS kernel can boot on an TI AM437x GP EVM board, the Linux® image needs to be copied to a boot device and the boot switches need to be set accordingly.

To bring up the board and run Linux® , four elements are needed:

- Boot loader (U-Boot)
- Linux® kernel image (zImage)
- A device tree file (.dtb) for the board being used
- A root file system (rootfs)

Open the Linux terminal and move to the location where the SDK is located. Type "sudo su" command to be a super user mode.

Invoking the Script

The **create-sdcard.sh** script is present in the bin directory of the SDK. It can be run from any location but must be run with **root** permissions. This usually means using the **sudo** command to start execution of the script. For example:

sudo <SDK INSTALL DIR>/bin/create-sdcard.sh If you fail to execute the script without root permissions you will receive a message that root permissions are required and the script will exit.

Select the SD Card Device

The first step of the script will ask you to select the drive representing the SD card that you want to format. In most cases your host root file system drive has been masked off to prevent damage to the host system. When prompted enter the device number corresponding to the SD card. For example if the output looks like:

```
Availible Drives to write images to:

#  major   minor    size   name
1:   8       16    7761920 sdb

Enter Device Number:
```

You would enter **1** to select the **sdb** device

Re-Partitioning the SD Card

Any partitions of the device that are already mounted will be un-mounted so that the device is ready for partitioning.

If the SD Card already has partition you will see a prompt like the following asking you if you would like to repartition the card. If the card was not already partitioned then this step will be skipped and you can move on to the next step.

```
Would you like to re-partition the drive anyways [y/n] :
```

Options:

- **y** - This will allow you to change the partitioning of the SD card. For example if you have a 3 partition card and want to create a 2 partition card to give additional storage space to the root file system you would select **y** here. **NOTE:** This operation **WILL ERASE** the contents of your SD card
- **n** - If the SD card already has the desired number of partitions then this will leave the partitioning alone.

Select Number of Partitions

You should now see a prompt like the following which will ask you how many partitions you want to create for the SD card.

```
Number of partitions needed [2/3] :
```

- Options:
  - **2** - This is the most common use case and will give the most space to the root file system.
  - **3** - This case should only be used by board manufacturers making SD cards to go in the box with the EVM. This requires access to the partition tarballs used for Out-Of-Box SD cards.

Installing SD Card Content

After the SD card is partitioned, you will be prompted whether you want to continue installing the file system or safely exit the script.

  - **y** - Selecting yes here will begin the process of installing the SD card contents. This operation **WILL ERASE** any existing data on the SD card.

SD Card Using Default Images

The purpose of this section is to cover how to use the **create-sdcard.sh** script to populate an SD card that can be used to boot the device using the default images that ship with the Processor SDK for Linux.

Choose Install Pre-built Images

You should now see a prompt like:

```
################################################################################

    Choose file path to install from

    1 ) Install pre-built images from SDK
    2 ) Enter in custom boot and rootfs file paths

################################################################################

Choose now [1/2] :
```

You should choose option **1** to create an SD card using the pre-built images from the SDK

If you executed this script from within the SDK then the script can determine the SDK path automatically and will start copying the contents to the SD card. Once the files are copied the script will exit.

If you executed the script from outside of the SDK (i.e. you copied it to some other directory and executed it there) please see the next section.

NOTE: option 1 will only work with the format of the default SDK directory name, which makes the Hands on with the SDK training easiest. If you have to change the directory name, use option 2 to enter the custom file paths.

Choose rootfs tarball for K2G

You should now see a prompt like:

```
################################################################################

   Multiple rootfs Tarballs found

################################################################################

        1:tisdk-server-extra-rootfs-image-k2g-evm.tar.gz
        2:tisdk-server-rootfs-image-k2g-evm.tar.gz

Enter Number of rootfs Tarball:
```

Choose option **1** to create an SD card using the complete filesystem image from the SDK. Option **2** provides the base filesystem image of smaller size, and it can be used when the SD card does not have enough space.

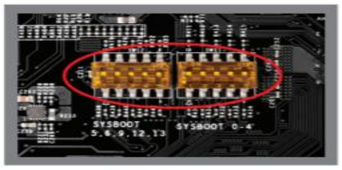
By completing this step, the SD Card will be flashed with the image, which contain all the necessary modules to boot up the board.

CONFIGURATION AND SETUP

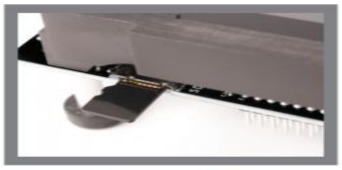
Boot Configuration

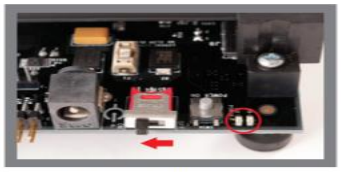
The AM437x has sysboot pins that can be configured using two 5-bit DIP switches on the EVM. These sysboot switches will configure the AM437x to different boot settings. The SW12 switch can be used to set sysboot bits 0 to 4, and the SW11 switch can set sysboot bits 5, 6, 9, 12, and 13. Other sysboot pin settings are done through resistors either pulled high or low. See the AM437x TRM and data sheet for the definitions of each of the sysboot signals. See the GP EVM schematic for more details.

By Default, all the DIP switches are at ON state.

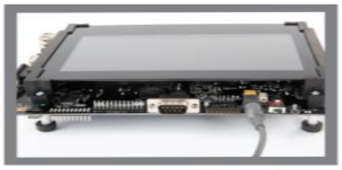


Verify all the DIP switches are set as shown (all ON). These switches are located on the back of the AM437x EVM.


Insert the Linux µSD card into the AM437x EVM as shown. The black surface should be facing up.


To turn on, slide the power switch (SW4) on the AM437x EVM to the left as shown. Both power LEDs (shown inside the red circle) will turn on. To turn off, slide the power button to the right.

Connect the supplied serial null modem cable to the UART DB-9 (J10) connector. *Note:* You must use the supplied cable or another serial null modem cable.


Connect the power cable to the power jack as shown and plug in to an AC power source.

Note that this EVM comes with a coin battery holder. A coin battery is not necessary for normal operation of the EVM. A coin battery (not included) can be inserted for testing ultra-low-power RTC-only mode.

Here the RS232 cable provision is provided to see the serial logs. We should have a RS232 to Serial convertor cable to connect the other end of the cable to the Linux host machine to see the console logs. Connect the serial null modem cable to the UART DB-9 (J10) connector.

In the serial terminal the booting logs will be displayed, and kernel will be loaded and system boots up successfully.

Adding the WILC Driver Support

By now, we could see that the system booted up with the default image present in the processor SDK. In the existing kernel image WILC driver support is not added. For that, we need to build the zImage by adding the WILC driver support.

The processor SDK have the kernel present at the board-support directory. The current SDK package have the Linux version 4.19.38.

Open the terminal and move into the kernel directory to build the zImage and dtb.

Building the Kernel and device tree file

- Export the toolchain path into the newly opened terminal.
- Configure the kernel with the defconfig file is located at arch/arm/configs

```
sudo  make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
tisdk_am437x-evm_defconfig
```

- Replace the existing driver with the latest driver at driver/staging/wilc1000 folder
  - Download the latest ATWILC driver from the repository https://github.com/linux4wilc/driver
  - In the kernel, replace the existing wilc1000 driver directory in `drivers/staging/wilc1000` with the downloaded ATWILC driver.
- In driver/staging/Makefile file, change the CONFIG_WILC1000 to CONFIG_WILC
- Open menuconfig using the command

```
sudo make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
```

and Modularize('M') the WILC_SPI and WILC_SDIO under Device Drivers/Staging drivers/ and save the settings.

NOTE: In the next section we will be interfacing WILC SD Device. For that, we need the wilc_sdio.ko file. It is advisable to build the wilc_sdio.ko module now.

- Build the kernel zImage:
  ```
  sudo make ARCH=arm CROSS_COMPILE=arm-linux-
  gnueabihf-  zImage
  ```
  kernel image will be present at arch/arm/boot directory
- To built the dtb file:
  am437x-gp-evm.dts file needs to be edited to add the SPI node

  Add the below code to the am437x-gp-evm.dts file

```
spi1_pins_default: spi1_pins_default {
        pinctrl-single,pins = <
            AM4372_IOPAD(0x990, PIN_INPUT | MUX_MODE3) /* (N24)
            mcasp0_aclkx.spi1_sclk */
            AM4372_IOPAD(0x994, PIN_OUTPUT_PULLUP | MUX_MODE3) /*
            (N22) mcasp0_fsx.spi1_d0 */
            AM4372_IOPAD(0x998, PIN_INPUT_PULLUP | MUX_MODE3) /* (H23)
            mcasp0_axr0.spi1_d1 */
            AM4372_IOPAD(0x99c, PIN_OUTPUT | MUX_MODE3) /* (M24)
            mcasp0_ahclkr.spi1_cs0 */
            AM4372_IOPAD(0xa54, PIN__PULLUP | MUX_MODE7) /* (R24)
            spi4_d0.gpio5[5] */

            AM4372_IOPAD(0xa40, PIN_OUTPUT_PULLUP | MUX_MODE7) /*
            (G20) gpio5_10.gpio5[10] */
            AM4372_IOPAD(0xa44, PIN_OUTPUT_PULLUP | MUX_MODE7) /*
            (F23) gpio5_11.gpio5[11] */
        >;
    };
```

```
&spi1 {
        compatible = "ti,am4372-mcspi","ti,omap4-mcspi";
        #address-cells = <1>;
        #size-cells = <0>;
        status = "okay";
        pinctrl-names = "default";
        pinctrl-0 = <&spi1_pins_default>;
        ti,pindir-d0-out-d1-in = <1>;
        wilc_spi@0{
            #address-cells = <1>;
            #size-cells = <0>;
            status = "okay";
            compatible = "microchip,wilc1000";
            reg = <0>;
            spi-max-frequency = <25000000>;
            interrupt-parent = <&gpio5>;
            irq-gpios = <&gpio5 5 IRQ_TYPE_EDGE_FALLING>;
            reset-gpios = <&gpio5 10 GPIO_ACTIVE_HIGH>;
            chip_en-gpios = <&gpio5 11 GPIO_ACTIVE_HIGH>;
        };
 };
```

- Build the dtb file after adding the spi node to dts file

```
sudo make ARCH=arm CROSS_COMPILE=arm-linux-
gnueabihf-  am437x-gp-evm.dtb
```

after the build is successful, dtb file will be present at arch/arm/boot/dts directoy

Building the Modules

Initialization of the WILC module requires the module file (wilc-spi.ko).

Build the modules, using the following command:

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- module
```

Replacing the zImage and dtb

- Replace the existing zImage and dtb file in the FAT partition of the SD Card with the latest build zImage and dtb file.

Copy the WILC module file(wilc-spi.ko)

- Copy the module file wilc-sdio.ko in the EXT4 partition(root) of the SD Card.

Loading Firmware

- The default rootfs doesn't contain the wilc firmware. In order to download the firmware into the wilc device during the wilc initialization, firmware file needs to be placed into specific directory in rootfs. Rootfs is present in the EXT4 partition of SD Card.
- Create a directory named "mchp" under lib/firmware directory in rootfs.
- Download the latest firmware from https://github.com/linux4wilc/firmware and place inside mchp directory

Initialization of WILC Module

- Move to root

```
cd /
```

ls command will list the available files in rootfs. Previously placed wilc-spi.ko file will be present here.

- Load the module using the command:

```
insmod wilc-spi.ko
```

WILC driver will be loaded successfully.

Configuring WILC Module as Station and Connecting to AP

By default, the wilc wpa supplicant file won't be available in the rootfs. In order to initialize the supplicant, the supplicant file need to be created and content needs to be added.

- Create the file inside the /etc using the vi command vi /etc/wilc_wpa_supplicant.conf
- Add the below code into the file

```
ctrl_interface=/var/run/wpa_supplicant  update_config=1
ap_scan=1
```

- Initializing the supplicant

```
wpa_supplicant -iwlan0 -Dnl80211 -c /etc/wilc_wpa_supplicant.conf -B
```

Connecting to a AP using following command

1) wpa_cli -p /var/run/wpa_supplicant ap_scan 1
2) wpa_cli -p /var/run/wpa_supplicant add network
3) wpa_cli -p /var/run/wpa_supplicant set_network 0 ssid ‘”DEMO_LINUX”’
4) wpa_cli -p /var/run/wpa_supplicant set_network 0 key_mgmt WPA-PSK
5) wpa_cli -p /var/run/wpa_supplicant set_network 0 psk ‘”12345678’”
6) wpa_cli -p /var/run/wpa_supplicant select_network 0
7) udhcpc -iwlan0

Interfacing ATWILC1000 SD with AM437x GP EVM

In this section we will discuss in detail about interfacing WILC1000 SD Card device with the TI AM437x GP EVM board. WILC1000 with SD Card interface have better performance in terms of data exchange than SPI interface. For interfacing WILC through SPI interface, we had our image flashed into SD Card and made the board boot form this micro SD Card.

AM437x GP EVM have only one micro SD Card slot. This limit the opportunity to directly plug in the WILC SD Device to host board. For this, we must use an extra micro to SD convertor adaptor device.

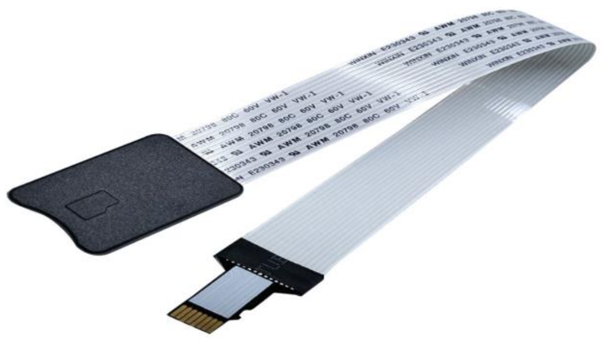


Figure 8. micro SD to SD Card convertor Adapter

Till now we saw how to download and install the processor SDK installer and create the SD Card image using script. Having the SD Card image flashed into the micro SD Card we will start with our next step to make the AM437x GP EVM board boot from the NAND flash memory by copying the image from the Micro SD card inserted in the sd card slot into the NAND flash memory. After copying the image into NAND Flash, we remove the micro SD Card and insert the ATWILC1000 SD Card into the micro sd slot using a convertor Adapter.

Booting from NAND Flash

In this section we will discuss about how to make the AM437x GP EVM board to boot from NAND Flash memory.

Steps to Copy the image from SD Card to NAND Flash

- Insert the micro SD Card with the preloaded image into the micro sd card slot
- Interrupt the board booting process at the u-boot timeout session
- Next, we will copy the image from SD card to NAND flash using below commands

```
-  mmc rescan

- nand erase.chip

- load mmc 0 0x81000000 MLO

- nand write 0x81000000 NAND.SPL

- nand write 0x81000000 NAND.SPL.backup1

- nand write 0x81000000 NAND.SPL.backup2
    –
– nand write 0x81000000 NAND.SPL.backup3

- load mmc0 0x81000000 u-boot.img

- nnand write 0x81000000 NAND.u-boot
-  load mmc 0:2 0x81000000 /boot/zImage

-  nand write 0x81000000 NAND.kernel

-  load mmc 0:2 0x81000000 /boot/am437x-gp-evm.dtb

-  nand write 0x81000000 NAND.u-boot-spl-os

-  load mmc 0:2 0x81000000 tisdk-docker-rootfs-
image-am437x-evm.ubi

-  nand write 0x81000000 NAND.file-system

-  saveenv

-  run nandboot
```

Connecting ATWILC1000 SD Card

In the previous section we saw how to copy the boot image from the micro SD Card slot to the NAND Flash memory and make the AM437x GP EVM Board to boot from the NAND flash memory.

After the board successfully booting form the NAND Flash memory, remove the micro SD Card from the slot and insert the micro SD to SD Card convertor adapter in the slot and connect the ATWILC1000 SD card into the adaptor and press reset button.

By Default, all the DIP switches are at ON state. This way the board boot process follows a sequence for booting. Initially board look for the boot image at the micro SD card slot. Since we have connected the ATWILC1000 SD Card device at micro sd card slot using an adapter it fails the voltage criteria and look for the image in the NAND Flash memory and boot the image form NAND Flash memory.

Copy the WILC module file(wilc-sdio.ko)

- Copy the previously build wilc-sdio.ko file in to the root using a USB device. There is a USB port available on the AM437x GP EVM Board.

Loading Firmware

- The default rootfs doesn’t contain the wilc firmware. In order to download the firmware into the wilc device during the wilc initialization, firmware file needs to be placed into specific directory in rootfs.
- Create a directory named “mchp” under lib/firmware directory in rootfs.
- Download the latest firmware from https://github.com/linux4wilc/firmware and place inside mchp directory

Initialization of WILC Module

- Move to root

```
cd /
```

ls command will list the available files in rootfs. Previously placed wilc-spi.ko file will be present here.

- Load the module using the command:

```
insmod wilc-sdio.ko
```

WILC driver will be loaded successfully.