SX-PCEAC2
User's Guide:
NXP i.MX8MQuad EVK

silex technology

Table of Contents

Introduction

This User's Guide provides an example of SX-PCEAC2 evaluation with NXP i.MX8MQuad EVK. The demonstration guides the users through setting up throughput test firmware with iperf.

Pre-requisites

This User's Guide expects the user to be familiar with the following topics and concepts:

- Bash (Linux command line interface)
- Yocto Linux development environment (https://www.yoctoproject.org/)
- IEEE 802.11 Network
- hostapd and wpa_supplicant (https://w1.fi)
- iperf (https://iperf.fr)

> The user should have a running version of NXP BSP in their Ubuntu Linux development host PC environment.

Requirements

- Ubuntu 16.04 LTE x86_64 Development Host PC
- NXP i.MX8M Linux BSP Version L4.9.51_8MQ_GA
- NXP i.MX8M Linux BSP update file from Silex.
- IEEE 802.11ac Access Point
- NXP i.MX8M Development Board (MCIMX8M-EVK)
- Silex SX-PCEAC2 mPCIe 802.11ac wireless module
- SX-PCEAC2 Board Data file (Silex_OG_BE_FIX.bin)
- mPCIe to M.2 Key A/E adapter
- USB 2.0 cable (Micro-B to Type A)
- 8GB Micro SD card

Hardware Setup

The SX-PCEAC2-EVK comes with the following:

- SX-PCEAC2-HMC-SP
- Two Unictron Antennas
- mPCIe to M.2 adapter
  - The parts must be assembled as show in figure 1
- Insert for "Documentation"
- 8.0 GB SD Card (blank)

The following steps are required for the hardware setup.

1. Attach SX-PCEAC2 to i.MX8M Development Board with mPCIe to M.2 adapter (See Figure 1).
2. Connect i.MX8M Development Board and Development PC with USB2.0 cable (Micro-B to Type A). It is used for UART console access.
3. Connect power supply cable to i.MX8M Development Board.

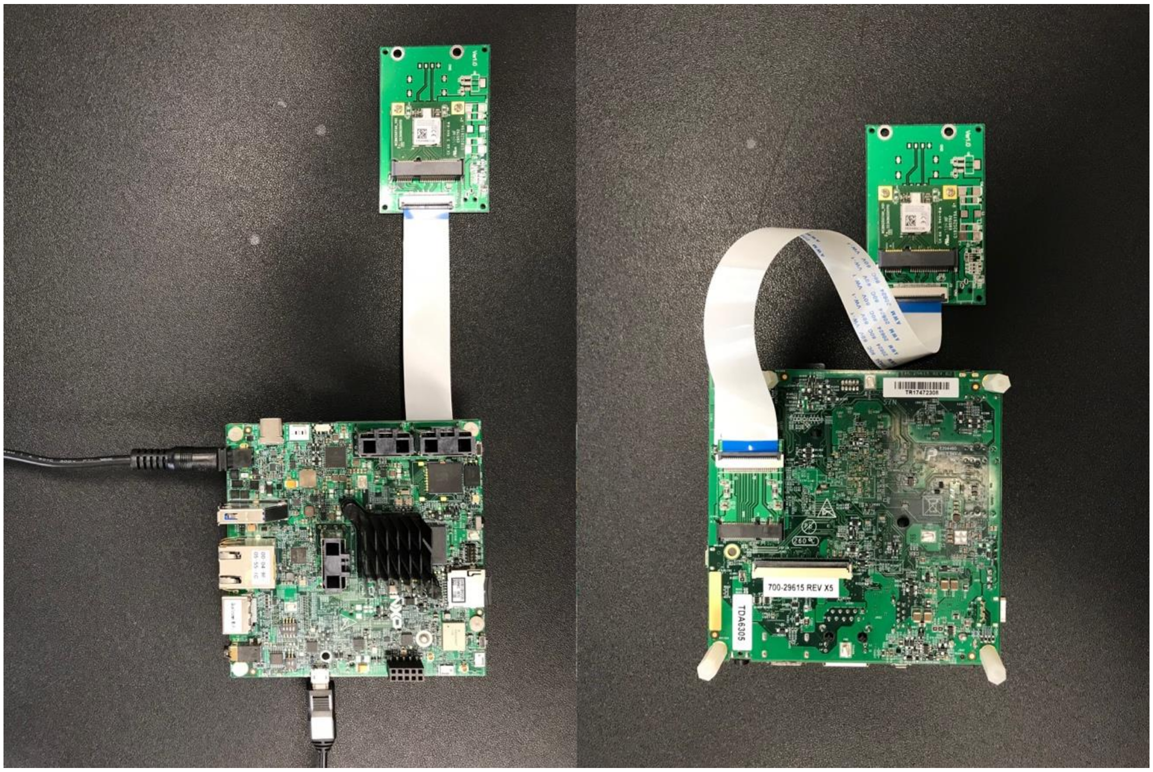


Figure 1 PCEAC2 and i.MX8M Development Board (left: top, right: bottom)

Evaluation Software

There are 3 options available to begin your evaluation.

i) Use the shipping NXP BSP:

The NXP i.MX8M ships with the Linux BSP that uses the WLAN ath10k driver. The board data file used is not optimized for best performance for both the onboard radio and the Silex's SX-PCEAC2. But it does work and is the quickest way to evaluation (Not recommended for evaluation for performance and mass production purposes).

ii) Use the Silex Evaluation Image (RECOMMENDED):

Silex has put together an evaluation image that uses the ath10k WLAN driver. It also includes the security supplicant and some basic tools to test the basic wireless functionality, wireless security and data throughput. We have included the fully optimized board data file for the Wi-Fi radio. We have taken this board data file through radio certification. *Customers can leverage this for both evaluation and begin mass production by leveraging Silex's modular certification for the QCA6174 based radio module. Also we have enabled "AP mode" functionality in our image.*

Downloading the EVK Image

To obtain the image on the web site use the following instructions:

1) Go to https://www.silextechnology.com/sx-pceac2-evaluation/
2) Under the "What you will need?" section, click on "Evaluation Software".
3) You will be asked to Accept an evaluation agreement.
4) Complete the information form, click Submit.
5) A confirmation email will be sent to the email provided. Upon verification you will be provided a link to download the evaluation image.
6) Decompress the EVK image file to a location where it can be accessed for the SD card writing process.

Creating Bootable SD Card – Linux Host Platform

To create a bootable SD card we will use the **dd** command, from the Linux command console. The format of the **dd** command is as follows:

```
dd if=[input file/device] of=[output file/device] [options]
```

To write the bootable card we will use the Silex EVK card image as the input file and the SD card device as the output file. We will use a specific option to help the writing of the SD card.

Firstly, you must have completed section 4.1 and downloaded the EVK SD card image to a known location. You will need to know the path and full name of the image (.img) file. Secondly you will need to establish the device name for the SD card you are going to write.

To do this you will need to be root:

```
sudo su
```

Next we need to find the name of the device in the /dev directory. To do this you should insert your blank SD card into the SD slot on your machine then enter the following:

```
dmesg | tail
```

This will pipe the device messages into the tail command. You should see one of the lines, probably the last one in the list, it will look something like:

```
[1345842.064126] sd 3:0:0:0: [sdb] Attached  SCSI removable disk
```

You will use the sdb or similar device name assigned to the SD card. To write the image to the card issue the following command:

```
dd if=~/<image file name> of=/dev/sdb bs=512
```

This command writes the Silex image file, stored in the home directory to the SD card, using a block size of 512 bytes. The process may take a while when complete the Linux command prompt will be displayed.

Note: sdb is an example of the SD card device name assigned by Linux. Different Linux machine may have different device name assignment. Specifying an incorrect SD card device name may results in data loss on unintended data storage devices.

Creating Bootable SD Card – Windows Host Platform

To create a bootable SD card using a windows machine you will need a compatible SD slot. An external USB SD card writer/Reader is recommended.

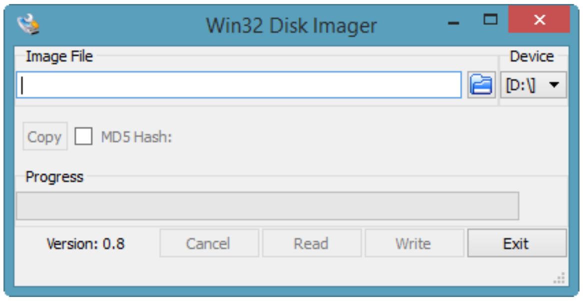
The following section is an example of how an SD card can be written using an application running on the Microsoft Windows OS. Silex has no affiliation with the developer of the mentioned application nor is responsible for issues using the application or bugs found. Use at your own risk.

An example of an application that can write to an SD card using the windows OS is Win32 Disk Imager. The application can be found for free on the sourceforge.net website. The following section will step through how to write an SD card using Win32 Disk Imager and the Silex EVK demo image.

To complete this section, you must have first downloaded the EVK image and installed the Win32 Disk Imager application.

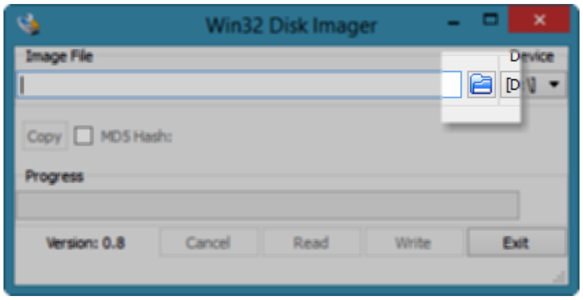
Firstly, run the Win32 Disk Imager application (as an administrator). You will see the following screen:

Figure 2- Win32 Disk Image Initial Screen



Next select the disk image file from the location it was stored. This is done by pressing the folder icon next to the **Image File** field:

Figure 3- Win32 Disk Image Find Image File



When the image file is selected it should look like the following:

Figure 4- Win32 Disk Image Select Image File

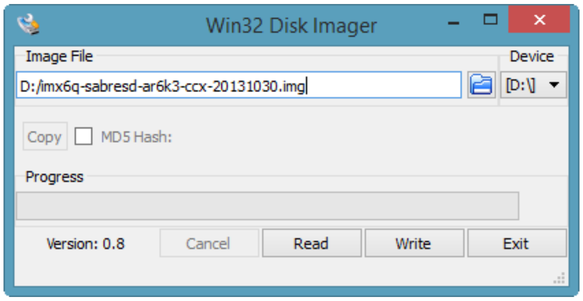
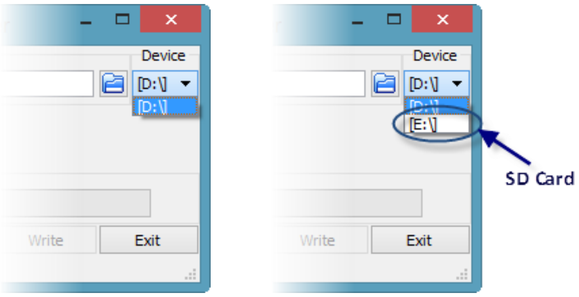


Image File: D:/imx6q-sabresd-ar6k3-ccx-20131030.img

Device: [D:\]

Next you must select the SD card to write to. This is done using the **Device** selection. It is important to select the correct device as the writing process will destroy all data on the device selected. The best way to find out which device holds the SD card is to press the down arrow next to the device letter, note the displayed devices. Insert your SD card that you want to write to and then repeat the step above, the new device letter in the list indicates the assigned device for the SD card:

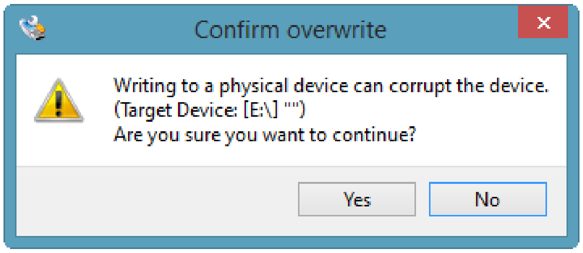
Figure 5- Win32 Disk Device Selection



Once the image and SD card are selected you are ready to write the image to the card.

Press the **Write** button. You will see the following warning:

Figure 6- Win32 Disk Image Confirm Writing



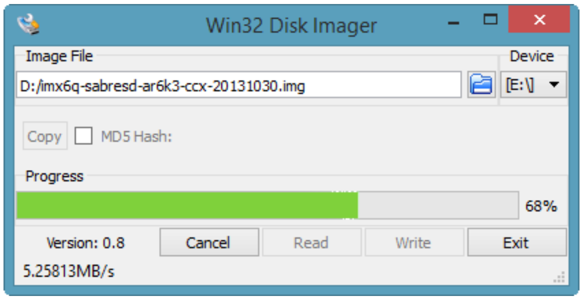
Confirm overwrite

Writing to a physical device can corrupt the device.
(Target Device: [E:\] "")
Are you sure you want to continue?

Press **Yes**.

The Windows will display the progress bar and indicate the percentage complete:

Figure 7- Win32 Disk Image Writing Image



Win32 Disk Imager

Image File: D:/imx6q-sabresd-ar6k3-ccx-20131030.img

Device: [E:\]

□ MD5 Hash:

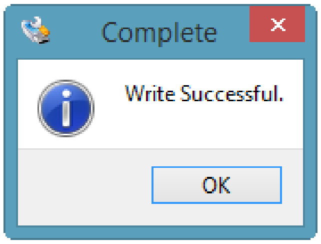
Progress: 68%

Version: 0.8

5.25813MB/s

When complete, the following message will be displayed:

Figure 8- Win32 Disk Image Writing Complete



Complete

Write Successful.

Press OK.

The SD card is now ready for use with the Silex EVK.

iii) Build i.MX8M Development Board Firmware

(Skip this section if using option ii)

This option enables you to build your own image using NXP NSP that Silex is done DVT on. This is only required if you do not want to use the image built and provided by Silex in option (ii) above. This takes the most time and maybe useful for software engineers trying to build their own EVI.

You may want to contact Silex to acquire NXP i.MX8M Linux BSP update file from Silex that enables you to leverage the bdata file for radio certification, changes to linux kernel configuration to enable hostapd mode etc.

Initial Host Firmware Build

1. Download NXP i.MX8M Linux BSP L4.9.51_8MQ_GA (search L4.9.51_8MQ_GA_LINUX_DOCS in https://www.nxp.com) to Development PC.
2. Follow Section 3 through 5 of i.MX Yocto Project User's Guide (IMXLXYOCTOUG, Rev. L4.9.51_imx8mq-ga, 03/2018) for full instruction. Commands below shows summary of firmware build procedure.

```
$ mkdir imx-yocto-bsp
$ cd imx-yocto-bsp
$ repo init \
> -u https://source.codeaurora.org/external/imx/imx-manifest \
> -b imx-linux-morty -m imx-4.9.51-8mq ga.xml
```

```
$ repo sync

$ MACHINE=imx8mqevk DISTRO=fsl-imx-xwayland \
> source fsl-setup-release.sh -b build-xwayland
$ bitbake fsl-image-validation-imx
```

> This initial host firmware build will take hours depending on Development PC performance.

Add Hostapd and Iperf to Host Firmware

Use following commands to install hostapd and iperf. Because the distribution fsl-imx-xwayland don't build these by default.

```
$ echo “IMAGE_INSTALL_append = \” iperf hostapd\”” >>
conf/local.conf
```

Update Ath10k Driver, Hostapd and Linux Kernel Configuration

```
$ cd tmp/work-shared/imx8mqevk/kernel-source
$ patch -p1 < ~/ath10k_regdomain.patch
$ cd -

$ cp ~/hostapd.defconfig ../source/meta-openembedded/meta-oe/recipes-
connectivity/hostapd/hostapd/defconfig
(It must be single line of command. No CR or LF in between the lines.)

$ cp ~/linux.config tmp/work/imx8mqevk-poky-linux/linux-imx/4.9.51-
r0/build/.config
(It must be single line of command. No CR or LF in between the lines.)
```

> The commands above assume that NXP i.MX8M Linux BSP update files from Silex are stored in user's home directory of Development PC.

Rebuild Host Firmware

```
$ bitbake fsl-image-validation-imx
```

Program Micro SD Card

```
$ bzip2 -dc tmp/deploy/images/imx8mqevk/fsl-image-validation-imx-
imx8mqevk.sdcard.bz2 | sudo dd of=/dev/sdb bs=512
(It must be single line of command. No CR or LF in between.)
```

	Device file name of SD card reader (/dev/sdb above) must be replaced with the device file name of your Development PC.
	When you run dd command, all partition (slice) of SD card must be unmounted.

Update Board Data File

```
$ mkdir /tmp/sdcard
$ sudo mount /dev/sdb2 /tmp/sdcard
$ sudo cp ~/Silex_OG_BE_FIX.bin \
> /tmp/sdcard/lib/firmware/ath10k/QCA6174/hw3.0/board.bin
$ sudo rm /tmp/sdcard/lib/firmware/ath10k/QCA6174/hw3.0/board-2.bin
$ sudo umount /dev/sdb2
```

	Device file name of SD card reader (/dev/sdb above) must be replaced with the device file name of your Development PC.
	The commands above assume that NXP i.MX8M Linux BSP update files from Silex are stored in user's home directory of Development PC.

Run Throughput Test

Boot i.MX8M Development Board

1. Insert Micro SD card to Micro SD card slot of i.MX8M Development Board.
2. Open UART port of Development PC with minicom (or equivalent console software).
3. Power on i.MX8M Development Board.
4. Login to Linux command user interface (username "root", no password.)

```
NXP i.MX Release Distro 4.9.51-mx8-ga imx8mqevk ttymxc0

Imx8mqevk login: root
Last login: Tue May 16 21:17:09 UTC 2018 on ttymxc0
root@imx8mqevk:~#
```

Stop Wpa_supplicant

Yocto Linux start wpa_supplicant listen to D-Bus by default. This guide will use configuration file in place of D-Bus command.

```
root@imx8mqevk:~# systemctl stop wpa_supplicant.service
```

Disable Murata Onboard mPCIe Module

```
root@imx8mqevk:~# echo 0000:01:00.0 > \
> /sys/bus/pci/drivers/ath10k_pci/unbind
```

Now your board is ready. You can setup i.MX8M Development Board either as station (Scenario 1: Station Mode) or as AP (Scenario 2: AP Mode).

Scenario 1: Station Mode

1. Create configuration file (wpas.conf) as shown below.

```
root@imx8mqevk:~# vi wpas.conf
ctrl_interface=/var/run/wpa_supplicant
network={
   ssid=“SSID of your AP”
   proto=RSN
   key_mgmt=WPA-PSK
   psk=”passphrase of your AP”
   pairwise=CCMP
   group=CCMP
}
```

2. Start wpa_supplicant

```
root@imx8mqevk:~# wpa_supplicant -I wlP1p1s0 -D nl80211 \
> -c wpas.conf -B
```

3. Assign IP address to network interface

```
root@imx8mqevk:~# udhcpc -i wlP1p1s0
udhcpc (v1.24.1) started
Sending discover...
Sending select for 192.168.77.148...
Lease of 192.168.77.148 obtained, lease time 600
/etc/udhcpc.d/50default: Adding DNS 192.168.77.254
```

Scenario 2: AP Mode

1. Create configuration file (hapd.conf) as shown below.

```
root@imx8mqevk:~# vi hapd.conf
ctrl_interface=/var/run/hostapd
interface=wlP1p1s0
driver=nl80211
ssid=YourAP
hw_mode=a
ieee80211n=1
channel=44
ht_capab=[HT40+][SHORT-GI-20][SHORT-GI-40]
ieee80211ac=1
```

```
vht_oper_chwidth=1
vht_oper_centr_freq_seg0_idx=42
vht_capab=[MAX-PDU-11454][SHORT-GI-80]
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=securepassphrase
rsn_pairwise=CCMP
```

2. Assign static IP address to network address

```
root@imx8mqevk:~# ifconfig wlP1p1s0 192.168.0.1
```

3. Start hostapd

```
root@imx8mqevk:~# hostapd hapd.conf &
```

4. Connect Development PC to i.MX8M Development Board

> Assign static IP address (192.168.0.x/255.255.255.0) to wireless network interface of Development PC.

Run Throughput Test

1. Start iperf server on i.MX8M Development Board

```
$ iperf -s
```

2. Start iperf client on Development PC

```
$ iperf -c <IP address of i.MX8M Development Board>
```

Known Issues

- Linux ath10k_pci improperly handle internal regulatory domain code 0x6c. Currently, the driver default to use US country code. For DE country code, please see ath10k_regdomain.patch (c.f. Section **Update Ath10k Driver, Hostapd and Linux Kernel Configuration**.)
- Tx data throughput is limited to ~30Mbps due to the issue of Linux ath10k_pci.
- Tx data link rate reported by iw command does not reflect actual link rate.

About Silex Technology America, Inc.

Silex Technology builds on more than 40 years of hardware and software connectivity know-how and IP, custom design development experience, and in-house manufacturing capabilities, bringing value to customers with a foundation of technical expertise. With relentless attention to quality, exclusive access to Qualcomm Atheros expertise, and strategic partnerships with leading semiconductor providers, Silex Technology is the global leader in reliable Wi-Fi connectivity for products ranging from a medical device to a document imaging product to a video or digital display. With Silex Technology, customers get a single vendor that provides hardware and software support from design through manufacturing for successful product after successful product. For more information, please visit www.silexamerica.com.

Silex Technology America has made reasonable efforts to ensure the accuracy of the information contained herein as of the date of this publication, but does not warrant that the information is accurate or complete. Silex Technology America undertakes no obligation to update the information in this publication. Silex Technology America specifically disclaims any and all liability for loss or damages of any kind resulting from decisions made or actions taken by any party based on this information.

Revision History

Revision	Date	By	Notes
A	6/21/2018	T. Nakase	Initial release.

silex
technology