NXP Matter User Guide

NXP Linux OTBR solutions [matter v1.1]

Revision History

Date	Authors	Revision	Description
09/07/22	Gnar Fang Patrick Pan	0.1	Matter TE9
11/21/22	Gnar Fang Patrick Pan Linda Wang	0.9	1. Update Matter to v1.0 2. Added IMX6ULL OTBR solution 3. Added IW612 OTBR solution 4. Added Android controller demo
6/26/23	Gnar Fang	1.0	1. Update Matter to v1.1 2. Added i.MX93 EVK + IW612 M.2 Module solution

contents

1. Background 4
2. NXP Linux OTBR Solutions 5
    2.1 Hardware Requirements 5
    2.2 Software Requirements 5
    2.3 Hardware Connection Guide 6
3. Build OpenThread Border Router components 8
    3.1 Build the yocto image with integrated Opentheread Border Router 8
    3.2 Build the Matter OpenThread Border Router with yocto SDK 9
        3.2.1 Generated SDK 9
        3.2.2 Install YOCTO SDK 9
        3.2.3 Build OpenThread Border Router binaries. 10
    3.3 Build the Matter Application with yocto SDK 11
4. Setup K32W RCP and K32W lighting-app 13
    4.1 Flash K32W RCP binary to K32W061DK6 or USB dongle 13
        4.1.1 Download K32W SDK and Install DK6programmer 13
        4.1.2 Flash K32W Thread RCP binary 13
    4.2 Setup K32W lighting-app, K32W0-Lighting-App-Build-Guide 13
        4.2.1 Build K32W lighting app 13
        4.2.2 Flash K32W Lighting-app 13
5. Setup Matter 13
    5.1 Setup OTBR on IMX8MMEVK + 88W8987+K32W or IMX93EVK + IW612 13
        5.1.1 Connect the OTBR to the target Wi-Fi AP network 13
        5.1.2 Setup Openthread 14
        5.1.3 Config OpenThread network 15
        5.1.4 Get thread network credential information 16
        5.1.5 OTBR quick start 16

5.2 Setup OTBR on IMX6ULLEVK + 88W8987+K32W 17
5.2.1 change fdt_file to setup WiFi and BT. 17
5.3 Setup OTBR on IMX8MMEVK + IW612 RD EVK 17
6. Matter Demo 17
6.1 Add a K32W end device in Matter Network 18
6.1.1 Pairing commissionee (Lighting node) to commissioner (OTBR) 18
6.1.2 Control end device 19
6.2 Android CHIP-TOOL 19
6.2.1 Confirm OTBR has been assigned an ipv6 address 19
6.2.2 Pairing commissionee (Lighting node) to commissioner (Android) 20
Appendix 21
Revision History 21
Legal 21

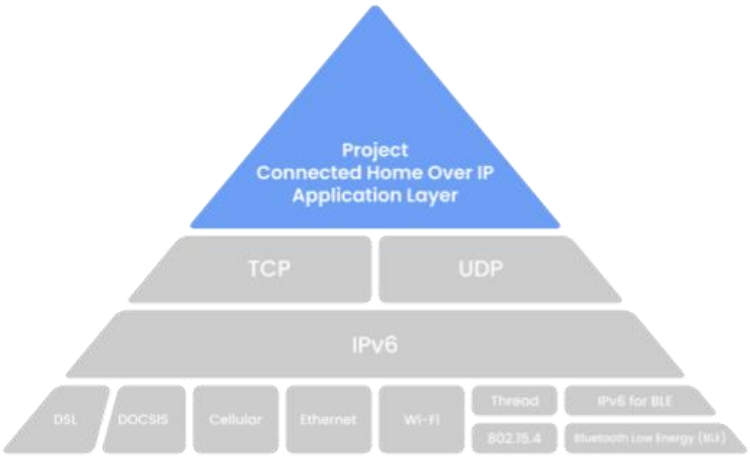
1. Background

Matter (formerly Project Connected Home over IP, or Project CHIP) is a new Working Group within the Connectivity Standards Alliance (CSA, formerly Zigbee Alliance). This Working Group plans to develop and promote the adoption of a new, royalty-free connectivity standard to increase compatibility among smart home products, with security as a fundamental design tenet.

The goal of the Matter project is to simplify development for manufacturers and increase compatibility for consumers. The project is built around a shared belief that smart home devices should be secure, reliable, and seamless to use. By building upon Internet Protocol (IP), the project aims to enable communication across smart home devices, mobile apps, and cloud services and to define a specific set of IP-based networking technologies for device certification.

The CSA officially opened the Matter Working Group on January 17, 2020 and is in the process of drafting the specification.

Visit buildwithmatter.com to learn more and read the latest news and updates about the project.



NXP have many productions support Matter, Details can be accessed nxp official website https://www.nxp.com/products/wireless/matter:MATTER-PROTOCOL.

2. NXP Linux OTBR Solutions

This article presentation three OTBR solutions based on NXP i.MX application processor. For more Matter solutions, please consult marketing or Sales.

- **i.MX6ULL** + **88W8987**(WiFi-BT combo Module) + **K32W**(OpenThread RCP module)
- **i.MX8MM** + **88W8987**(WiFi-BT combo Module) + **K32W**(OpenThread RCP module)
- **i.MX8MM** + **IW612-RD-EVK** (WiFi-BT-Thread tri-radio single-chip module)
- **i.MX93** + **IW612** (WiFi-BT-Thread tri-radio single-chip module)

2.1 Hardware Requirements

- i.MX 93 Evaluation Kit
  - USB Type C power supply
  - USB A to USB C cable
- i.MX 8M Mini Evaluation Kit (which contains the required materials below)
  - i.MX 8M Mini EVK
  - USB Type C power supply
  - USB A to micro USB cable
  - USB A to USB C cable
- K32W061 DK006
  - K32W061 DK006 + OM15082
  - K32W USB Dongle or K32W061 DK006
  - USB A to MINI USB
  - USB type C to A
- Murata 2EL (IW612) Module
- Linux host computer
- Wi-Fi Access Point which supports IPv6 and IPv6 DHCP server

2.2 Software Requirements

- Universal Update Utility (UUU)
- A serial COM terminal for the host computer
  - For instructions on how to set up a serial COM terminal software, please refer to page 10 in the i.MX 8M Mini LPDDR4 EVKB Quick Start Guide.

2.3 Hardware Connection Guide

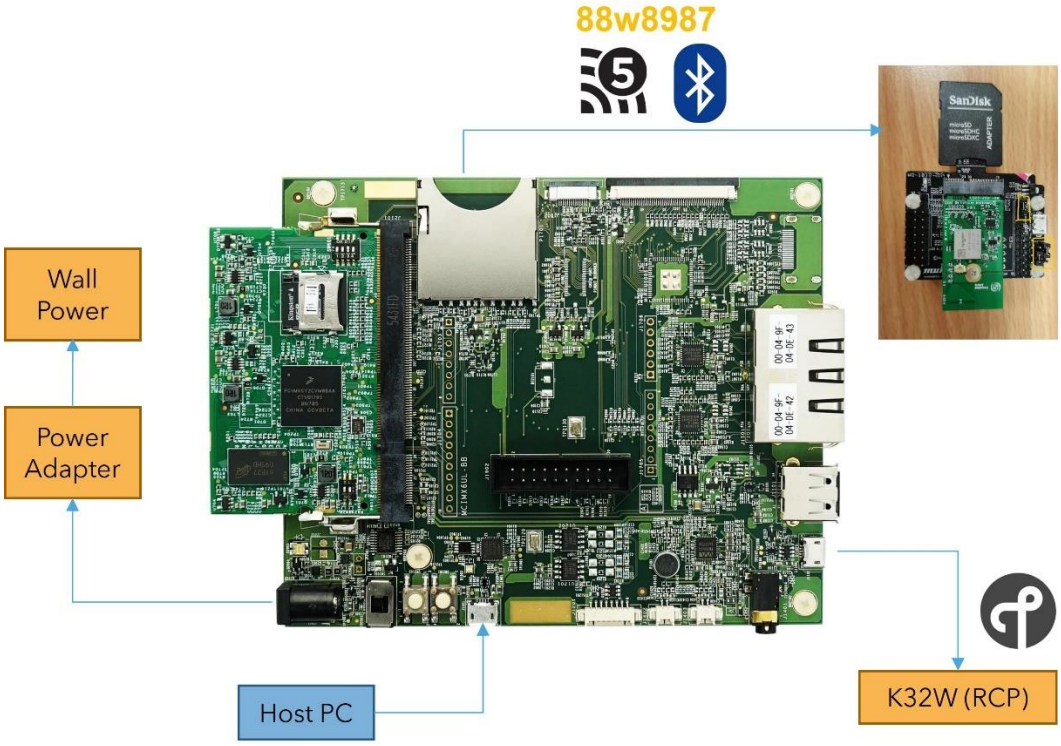


Figure 2-1 OTBR i.MX6ULL + 88W8987 + K32W

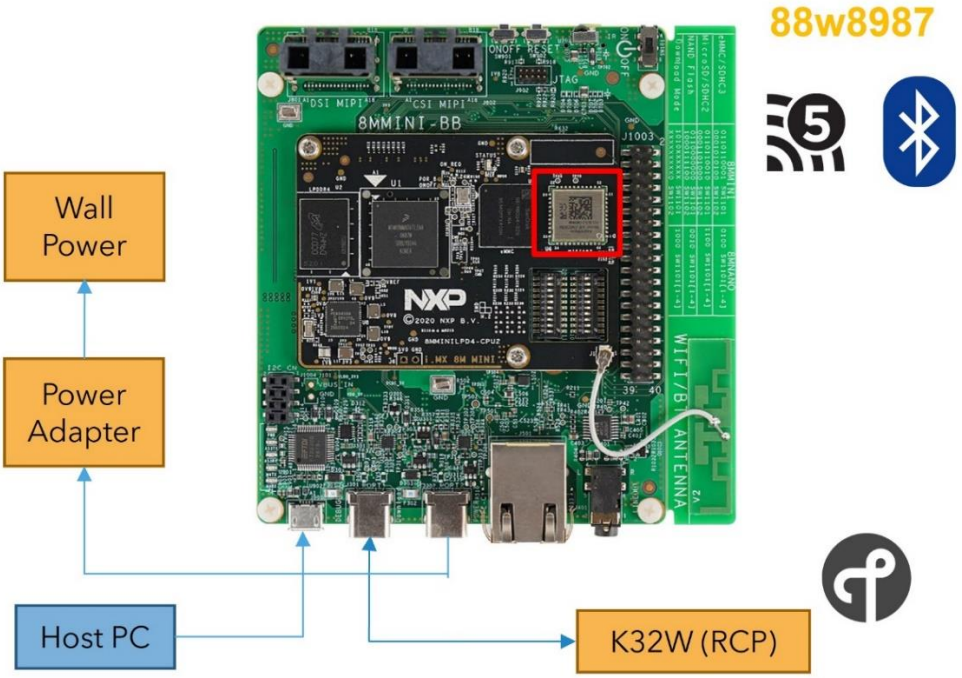


Figure 2-2 OTBR i.MX8MM + 88W8987 + K32W

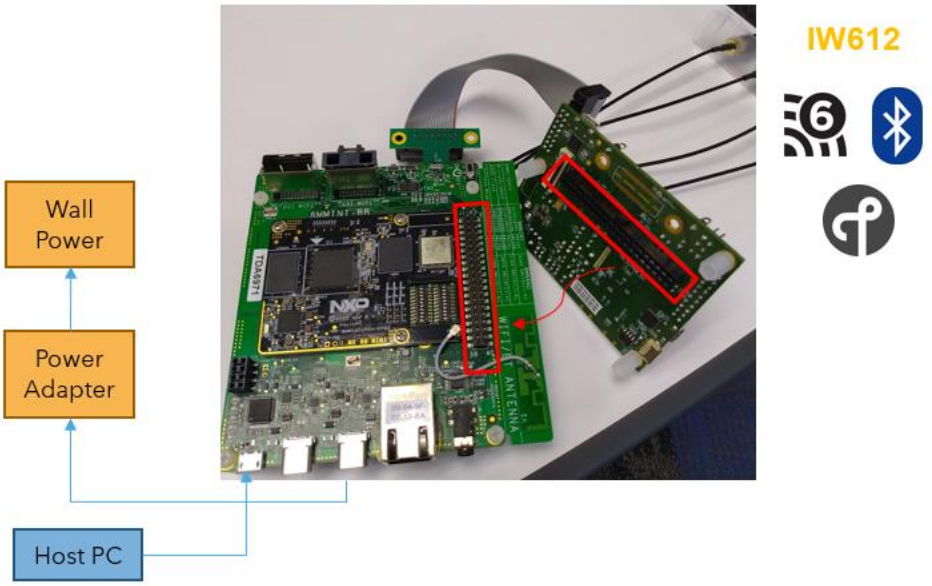


Figure 2-3 OTBR i.MX8MM + IW612-RD-EVK

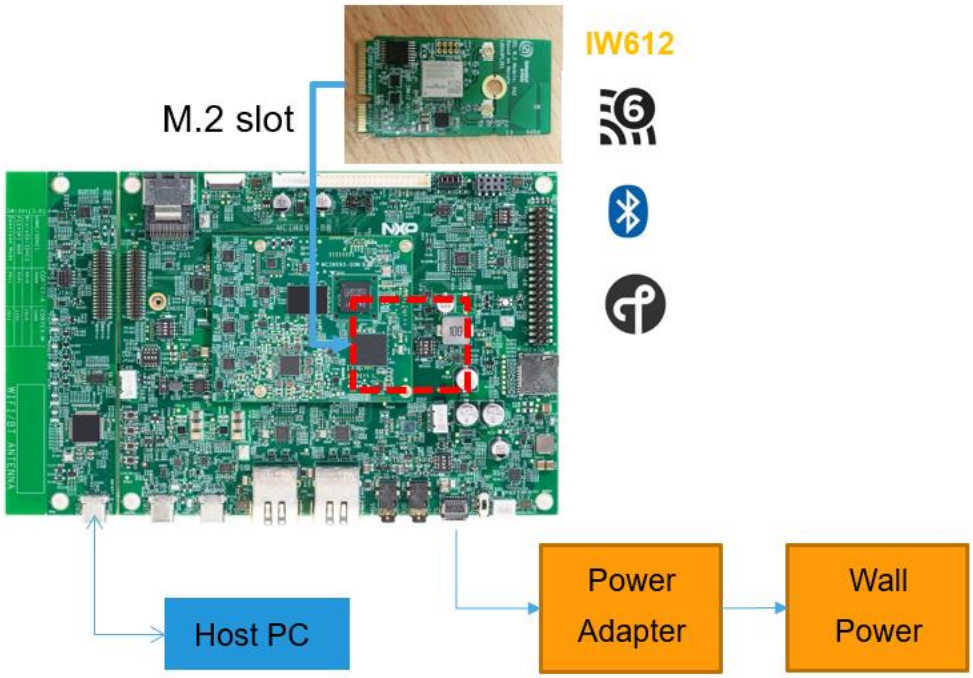


Figure 2-4 OTBR i.MX93 + IW612 M.2 Module

3. Build OpenThread Border Router components

There are two methods to integrated OpenThread Border Router. Build the yocto image with integrated Opentheread Border Router or Build the Matter OpenThread Border Router with yocto SDK.

Reference: [imx_matter_2023_q2]

3.1 Build the yocto image with integrated Opentheread Border Router

On the Linux host machine, run the command below to install the packages that are needed to build the image:

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
build-essential chrpath socat cpio python3 python3-pip python3-pexpect \
xz-utils debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa
libsdl1.2-dev \
pylint3 xterm npm zstd build-essential libpython3-dev libdbus-1-dev python3.8-
venv
If the repo tool is not installed, install it:
$ mkdir ~/bin
$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ export PATH=~/bin:$PATH
```

To build the yocto project, some python dependency packages need to be installed.

```
$ wget https://raw.githubusercontent.com/project-
chip/connectedhomeip/master/scripts/constraints.txt
$ pip3 install -r constraints.txt
$ pip3 install build mypy==0.910 types-setuptools pylint==2.9.3
$ pip install dbus-python
```

If Git is not configured, configure Git:

```
  $ git config --global user.name "Your Name"
  $ git config --global user.email "Your Email"
  $ git config --list
```

Create the Yocto build environment:

```
$ mkdir ~/bin
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo >
~/bin/repo
$ chmod a+x ~/bin/repo
$ export PATH=${PATH}:~/bin
$ mkdir ${MY_YOCTO}
$ cd ${MY_YOCTO}
```

```
$ repo init -u https://github.com/nxp-imx/imx-manifest -b imx-linux-langdale -m
imx-6.1.1-1.0.0.xml
$ repo sync
```

Then integrate the meta-matter into the yocto code base:

```
$ cd ${MY_YOCTO}/sources/
$ git clone https://github.com/nxp-imx/meta-matter.git
$ git checkout imx_matter_2023_q2
```

Build the Yocto image which integrated Matter.

```
$ cd ${MY_YOCTO}
# For i.MX93 EVK
$ MACHINE=imx93evk DISTRO=fsl-imx-xwayland source sources/meta-
matter/tools/imx-matter-setup.sh bld-xwayland-imx93
# for i.MX8MM EVK
$ MACHINE=imx8mmevk-matter DISTRO=fsl-imx-xwayland source sources/meta-
matter/tools/imx-matter-setup.sh bld-xwayland-imx8mm
# for i.MX6ULL EVK
MACHINE=imx6ullevk DISTRO=fsl-imx-xwayland source sources/meta-
matter/tools/imx-matter-setup.sh bld-xwayland-imx6ull
# build yocto
$ Bitbake imx-image-multimedia
```

Used the UUU tool to flash the yocto image to IMX6ULLEVK,IMX8MMEVK or IMX93EVK. Refer IMX_LINUX_USERS_GUIDE.pdf to using UUU tool.

```
$ cd ${MY_YOCTO}/bld-xwayland-imx8mm/tmp/deploy/images/imx93evk/imx-image-
multimedia-imx93evk.wic.zst for i.MX93 EVK
$ uuu -b emmc_all imx-image-multimedia-imx93evk.wic
```

3.2 Build the Matter OpenThread Border Router with yocto SDK

3.2.1 Generated SDK

Reference: IMX_YOCTO_PROJECT_USERS_GUIDE.pdf

```
$ cd ${MY_YOCTO}
$ source setup-environment <build-dir>
$ Bitbake imx-image-multimedia -c populate_sdk
```

3.2.2 Install YOCTO SDK

```
# For i.MX93EVK
$ sudo tmp/deploy/sdk/fsl-imx-xwayland-glibc-x86_64-imx-image-multimedia-
armv8a-imx93evk-toolchain-6.1-langdale.sh

# For i.MX8MMEVK
$ sudo tmp/deploy/sdk/fsl-imx-xwayland-glibc-x86_64-imx-image-multimedia-
armv8a-imx8mmevk-toolchain-6.1-langdale.sh
```

```
# For i.MX6ULLEVK
$ sudo tmp/deploy/sdk/fsl-imx-xwayland-glibc-x86_64-imx-image-multimedia-
cortexa7t2hf-neon-imx6ullevk-toolchain-6.1-langdale.sh
```

3.2.3 Build OpenThread Border Router binaries.

```
# For i.MX93EVK or i.MX8MMEVK
$ ./opt/fsl-imx-wayland/6.1-langdale-imx93/environment-setup-armv8a-poky-linux
# for imx6ullevk
$ ./opt/fsl-imx-wayland/6.1-langdale-imx93/environment-setup-armv8a-poky-linux

$ git clone https://github.com/openthread/ot-br-posix
$ cd ot-br-posix
$ git checkout -t origin/main
$ git submodule update --init

# For i.MX93 EVK
$ ./script/cmake-build -DOTBR_BORDER_ROUTING=ON -DOTBR_REST=ON -DOTBR_WEB=ON \
-DBUILD_TESTING=OFF -DOTBR_DBUS=ON -DOTBR_DNSSD_DISCOVERY_PROXY=ON \
-DOTBR_SRP_ADVERTISING_PROXY=ON -DOT_THREAD_VERSION=1.3 \
-DOTBR_INFRA_IF_NAME=mlan0 -DOTBR_BACKBONE_ROUTER=ON \
-DOT_BACKBONE_ROUTER_MULTICAST_ROUTING=ON -DOTBR_MDNS=mDNSResponder \
-DOT_POSIX_CONFIG_RCP_BUS=SPI \
-DCMAKE_TOOLCHAIN_FILE=./examples/platforms/nxp/linux-imx/aarch64.cmake

# For i.MX8MMEVK
$ ./script/cmake-build -DOTBR_BORDER_ROUTING=ON -DOTBR_REST=ON -DOTBR_WEB=ON \
-DBUILD_TESTING=OFF -DOTBR_DBUS=ON   -DOTBR_DNSSD_DISCOVERY_PROXY=ON \
-DOTBR_SRP_ADVERTISING_PROXY=ON -DOT_THREAD_VERSION=1.3  \
-DOTBR_INFRA_IF_NAME=mlan0  -DOTBR_BACKBONE_ROUTER=ON  \
-DOT_BACKBONE_ROUTER_MULTICAST_ROUTING=ON -DOTBR_MDNS=mDNSResponder \
-DCMAKE_TOOLCHAIN_FILE=./examples/platforms/nxp/linux-imx/aarch64.cmake

# For i.MX6ULLEVK
$ ./script/cmake-build -DOTBR_BORDER_ROUTING=ON -DOTBR_REST=ON -DOTBR_WEB=ON \
-DBUILD_TESTING=OFF -DOTBR_DBUS=ON   -DOTBR_DNSSD_DISCOVERY_PROXY=ON \
-DOTBR_SRP_ADVERTISING_PROXY=ON -DOT_THREAD_VERSION=1.3 \
-DOTBR_INFRA_IF_NAME=mlan0 -DOTBR_BACKBONE_ROUTER=ON \
-DOT_BACKBONE_ROUTER_MULTICAST_ROUTING=ON -DOTBR_MDNS=mDNSResponder \
-DCMAKE_TOOLCHAIN_FILE=./examples/platforms/nxp/linux-imx/arm.cmake

# chip-tool
$ ./scripts/build/build_examples.py  --target imx-chip-tool build

# nxp-themostat-app
$ ./scripts/build/imxlinux_examples.sh examples/nxp-thermostat/linux/ out/nxp-
thermosta debug
```

The otbr-agent is built in ${MY_OTBR}/build/otbr/src/agent/otbr-agent.

The otbr-web is built in ${MY_OTBR}/build/otbr/src/web/otbr-web.
The ot-ctl is built in ${MY_OTBR}/build/otbr/third_party/openthread/repo/src/posix/ot-ctl.
Please copy them into EVK /usr/sbin/.

The OTBR does not support incremental compilation. If an error occurs during compilation, or if you need to recompile, please delete ${MY_OTBR}/build before recompiling.

```
$ cd ${MY_OTBR}
$ rm -rf build/
```

3.3 Build the Matter Application with yocto SDK

The Matter application has been installed into the Yocto image by default. If you want build it separately, run the below commands to download the Matter application source code and switch to v1.1 branch.

```
# For i.MX93 EVK
$ export IMX_SDK_ROOT=/opt/fsl-imx-xwayland/6.1-langdale-imx93

# For i.MX8M Mini EVK
$ export IMX_SDK_ROOT=/opt/fsl-imx-xwayland/6.1-langdale-imx8mm

# For i.MX6ULL EVK
$ export IMX_SDK_ROOT=/opt/fsl-imx-xwayland/6.1-langdale-imx6ull

# download matter project
$ mkdir ${MY_Matter_Apps}
# this is top level directory of this project
$ cd ${MY_Matter_Apps}
$ git clone https://github.com/NXP/matter.git
$ cd matter
$ git checkout origin/v1.1-branch-nxp_imx_2023_q2
$ git submodule update --init

# start build
$ source scripts/activate.sh

# Build the all-clusters example with below command
$ ./scripts/build/build_examples.py --target imx-all-clusters-app build

# Build the lighting example with below command
$ ./scripts/build/build_examples.py --target imx-lighting-app  build

# Build the thermostat example with below command
$ ./scripts/build/build_examples.py  --target imx-thermostat build

# Build the chip-tool example with below command
$ ./scripts/build/build_examples.py  --target imx-chip-tool build
```

```
# Build the ota-provider example with below command
$ ./scripts/build/build_examples.py  --target imx-ota-provider-app build

# Build the nxp-thermostat-app for certification device reference
$ ./scripts/examples/imxlinux_example.sh examples/nxp-thermostat/linux/
out/nxp-thermostat debug

# Build the chip-bridge-app example with below command
$ ./scripts/examples/imxlinux_example.sh examples/bridge-app/linux/ out/bridge-
app debug

# Build the security enhanced with Trusty OS application using
build_examples.py, by adding "-trusty" to te target. For example:
$ ./scripts/build/build_examples.py  --target imx-chip-tool-trusty build

# Build the security enhanced with Trusty OS application using
imxlinux_example.sh, by adding "trusty" to the command. For example:
$ ./scripts/examples/imxlinux_example.sh examples/nxp-thermostat/linux out/nxp-
thermostat-trusty trusty
```

The applications are built in out/ subdirectories; the subdirectory name is specified with --target option, when building the examples. For example, the imx-all-clusters-app executable files can found in ${MY_Matter_Apps}/connectedhomeip/out/imx-all-clusters-app/.

Make sure the subdirectories do not exist before building an application with the same name. If an application needs to be built for several boards (both i.MX8M Mini EVK, i.MX6ULL EVK and i.MX93 EVK), user can specify a board dedicated directory with --out-prefix option; for example:

```
$./scripts/build/build_examples.py  --target imx-chip-tool --out-prefix ./out/imx8mm
build
```

An official Matter document explaining how to use chip-tool as a Matter controller can be found here.

A document explaining how to use Matter applications on the i.MX MPU platform can be found in the NXP Matter binaries guide.

4. Setup K32W RCP and K32W lighting-app

4.1 Flash K32W RCP binary to K32W061DK6 or USB dongle

4.1.1 Download K32W SDK and Install DK6programmer

Download K32W0 SDK 2.6.11 for Project CHIP , The DK6programmer.exe installer under the *SDK_2_6_8_K32W061DK6/tools/JN-SW-4407-DK6-Flash-Programmer* folder.

4.1.2 Flash K32W Thread RCP binary

The RCP binary is support i.MX series MPU andin K32W0 SDK,
SDK_2_6_11_K32W061DK6/tools/wireless/ot-rcp/ot-rcp-uart-no-fc-usb.bin

```
$ DK6Programmer.exe -s <COM_PORT> -e FLASH -p ot-rcp-uart-no-fc-usb.bin
```

4.2 Setup K32W lighting-app, K32W0-Lighting-App-Build-Guide

4.2.1 Build K32W lighting app

```
$ export NXP_K32W0_SDK_ROOT=/home/user/Desktop/SDK_2_6_11_K32W061DK6/
$ source ./scripts/activate.sh
$ cd examples/lighting-app/nxp/k32w/k32w0
$ gn gen out/debug --args="k32w0_sdk_root=\"${NXP_K32W0_SDK_ROOT}\"
chip_with_OM15082=1 chip_with_ot_cli=0 is_debug=false chip_crypto=\"platform\"
chip_with_se05x=0 chip_pw_tokenizer_logging=true"
$ ninja -C out/debug
```

4.2.2 Flash K32W Lighting-app

ssbl.bin path *SDK_2_6_11_K32W061DK6/boards/k32w061dk6/wireless_examples/framework/ssbl/binary/ssbl.bin*

```
$ DK6Programmer.exe -s <COM_PORT> -e FLASH -p k32w061dk6_ssbl.bin
$ DK6Programmer.exe -V5 -s COM39 -P 1000000 -w image_dir_0=0000000010000000
$ DK6Programmer.exe -V5 -s COM39 -P 1000000 -w image_dir_1=00400000CD040101
$ DK6Programmer.exe -V2 -s COM39 -P 1000000 -Y -p FLASH@0x4000=chip-k32w0x-light-
example.bin
```

5. Setup Matter

5.1 Setup OTBR on IMX8MMEVK + 88W8987+K32W or IMX93EVK + IW612

5.1.1 Connect the OTBR to the target Wi-Fi AP network

```
$ modprobe moal mod_para=nxp/wifi_mod_para.conf
$ wpa_passphrase ${SSID} ${PASSWORD} > imxrouter.conf
$ wpa_supplicant -d -B -i mlan0 -c ./imxrouter.conf
$ udhcpc -i mlan0

$ systemctl restart radvd
```

```
$ sleep 4

$ echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
$ echo 1 > /proc/sys/net/ipv4/ip_forward
$ echo 2 > /proc/sys/net/ipv6/conf/all/accept_ra
$ ln -sf /usr/sbin/xtables-nft-multi /usr/sbin/ip6tables
$ ipset create -exist otbr-ingress-deny-src hash:net family inet6
$ ipset create -exist otbr-ingress-deny-src-swap hash:net family inet6
$ ipset create -exist otbr-ingress-allow-dst hash:net family inet6
$ ipset create -exist otbr-ingress-allow-dst-swap hash:net family inet6

# For IMX8MM + 8987 + k32W0(RCP)
bt_devif=ttymxc0

# For IMX93 + iw612
bt_devif=ttyLP4

# BT
$ /usr/libexec/bluetooth/bluetoothd &
$ hciattach /dev/ttymxc0 any 115200 flow
$ sleep 1
$ hciconfig hci0 up
$ hcitool -i hci0 cmd 0x3f 0x0009 0xc0 0xc6 0x2d 0x00
$ killall hciattach     hciattach ${bt_devif} any -s 3000000 3000000 flow
$ sleep 1
$ hciconfig hci0 up
```

5.1.2 Setup Openthread

```
# For IMX93 + iw612
$ gpioset gpiochip6 0=1
$ otbr-agent -I wpan0 -B mlan0 'spinel+spi:///dev/spidev0.0?gpio-reset-
device=/dev/gpiochip6&gpio-int-device=/dev/gpiochip4&gpio-int-line=10&gpio-reset-
line=1&spi-mode=0&spi-speed=1000000&spi-reset-delay=0&' &

# For IMX8MM + 8987 + k32W0(RCP)
$ ./otbr-agent -I wpan0 -B mlan0 'spinel+hdlc+uart:///dev/ttyUSB0?uart-
baudrate=1000000' -v -d 5 &


$ iptables -A FORWARD -i mlan0 -o wpan0 -j ACCEPT
$ iptables -A FORWARD -i wpan0 -o mlan0 -j ACCEPT
$ ip6tables -A FORWARD -i mlan0 -o wpan0 -j ACCEPT
$ ip6tables -A FORWARD -i wpan0 -o mlan0 -j ACCEPT
$ hciconfig hci0 up
$ otbr-web &
```

5.1.3 Config OpenThread network

In Web browser input IP address of the IMX8MMEVK then click “Form” on left side of the web page where any of the parameter for Openthread network can be changed, including PAN ID/ExtPAN ID/Channel/NwkKey,etc.

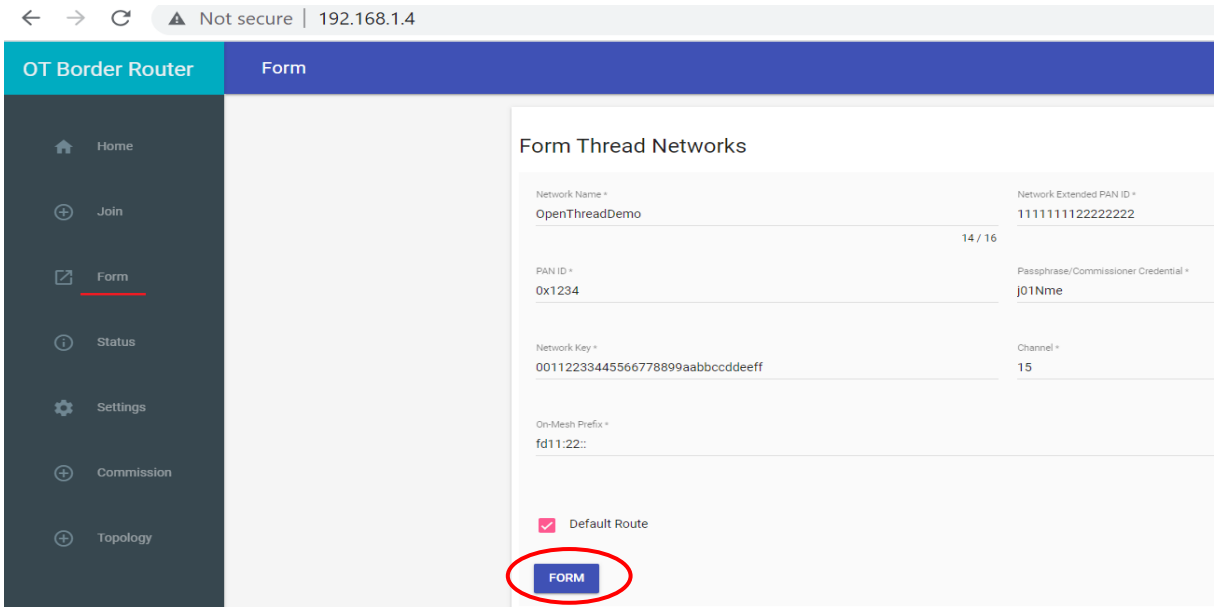


Figure 5-1 otbr-web config

After prompt as following to indicate thread network FORM is successful

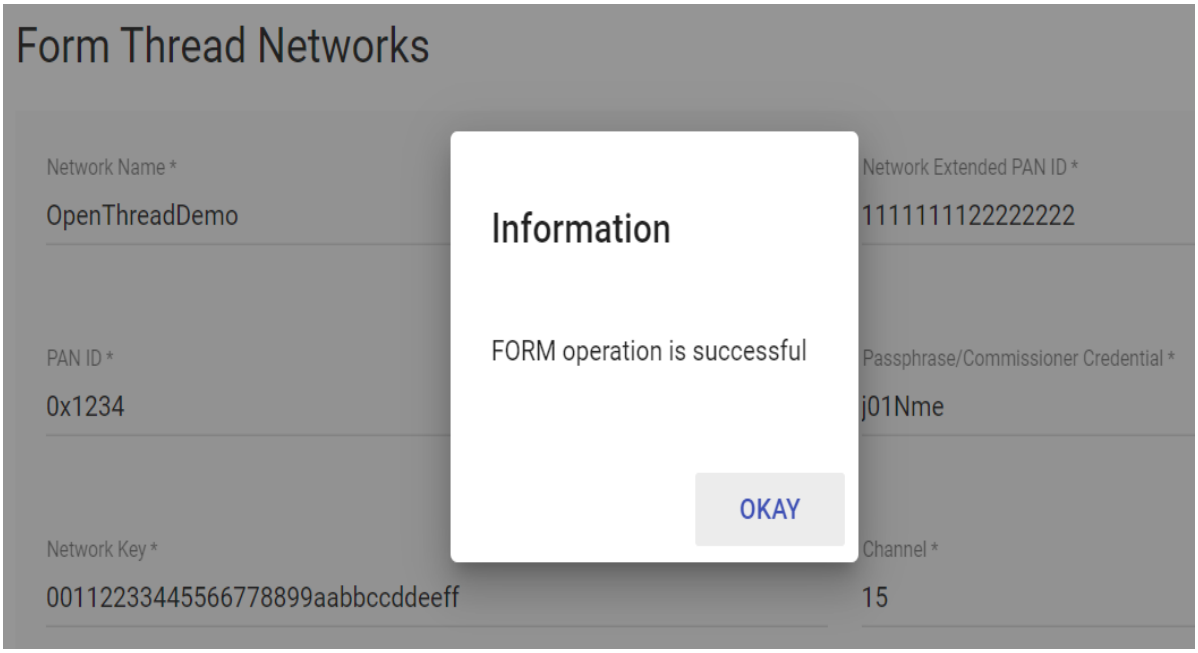


Figure 5-2 OTBR config success

Click “Topology” on left side of the web page and one Leader in Blue circle will displayed

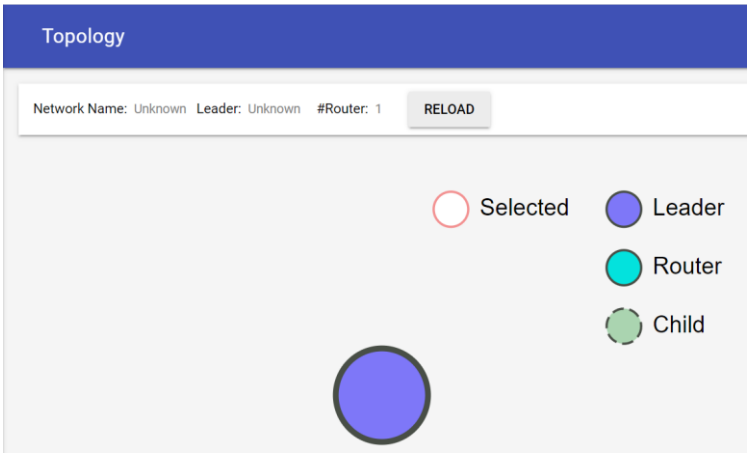


Figure 5-3 otbr topology Map

Or manually form the Openthread network by following steps:

```
$ ./ot-ctl dataset init new
$ ./ot-ctl dataset networkkey 00112233445566778899aabbccddeeff
$ ./ot-ctl dataset channel 15
$ ./ot-ctl dataset panid 0x1234
$ ./ot-ctl dataset extpanid 1111111122222222
$ ./ot-ctl dataset networkname Matter-NXP-1
$ ./ot-ctl prefix add fd08:b89:78:f372::/64 paos med
$ ./ot-ctl ifconfig up
$ ./ot-ctl thread start
$ ./ot-ctl state
```

ot-ctl state will return with “leader” indicating that thread network is created successful

5.1.4 Get thread network credential information

```
$ ot-ctl dataset active -x
```

Get operational_dataset.

```
0e080000000000000000000300000b35060004001fffe00208dead00beef00cafe0708fddead00bee
f000005106fb5e8851e6985cc4f77b981eba5b94a030a4f70656e5468726561640102079f04108fa1
fc598d79f6c0d8fd77f3b44404b80c0402a0fff8
```

this information will feed from commissioner to the joining matter device to config it pairing to open thread network properly.

5.1.5 OTBR quick start

```
# IMX8MM + 88w8987 + K32W0
$ ./imx8mm_k32w_matter.sh init
# i.MX93 + IW612
$ ./imx93_matter.sh  init
```

5.2 Setup OTBR on IMX6ULLEVK + 88W8987+K32W

Connect RCP module (K32W061 USB Dongle flashed with *ot-rcp-uart-no-fc-usb.bin*) and 88W8987 module(Figure 2-1) to IMX6ULLEVK.

5.2.1 change fdt_file to setup WiFi and BT.

entery uboot pattern

```
=> print fdt_file
fdt_file=undefined
=> fatls mmc 1
   36171   imx6ull-14x14-evk-btwifi-sdio3_0.dtb
   36035   imx6ull-14x14-evk-btwifi.dtb
   35839   imx6ull-14x14-evk-emmc.dtb
   36249   imx6ull-14x14-evk-gpmi-weim.dtb
   35747   imx6ull-14x14-evk.dtb
  354120   tee.bin
  354184   uTee-6ullevk
 9656568   zImage

8 file(s), 0 dir(s)

=> setenv fdt_file imx6ull-14x14-evk-btwifi-sdio3_0.dtb
=> saveenv
Saving Environment to MMC... Writing to MMC(1)... OK

#make sure fdt_file have been changed
=> print fdt_file
fdt_file=imx6ull-14x14-evk-btwifi-sdio3_0.dtb
```

MX6ULL can play the same role as i.MX8M Mini. i.MX6ULL OTBR setup scripts are similar to the i.MX8M Mini (Chapter5.1), the only difference is hciattanch command.

```
#for i.MX8M Min
$ hciattach /dev/ttymxc0 any 115200 flow
#for i.MX6ULL
$ hciattach /dev/ttymxc1 any 115200 flow
```

5.3 Setup OTBR on IMX8MMEVK + IW612 RD EVK

Refer Doc. Matter Reference Kit Application Note

6. Matter Demo

The chapter will demonstrate the features of Matter through demos.

6.1 Add a K32W end device in Matter Network

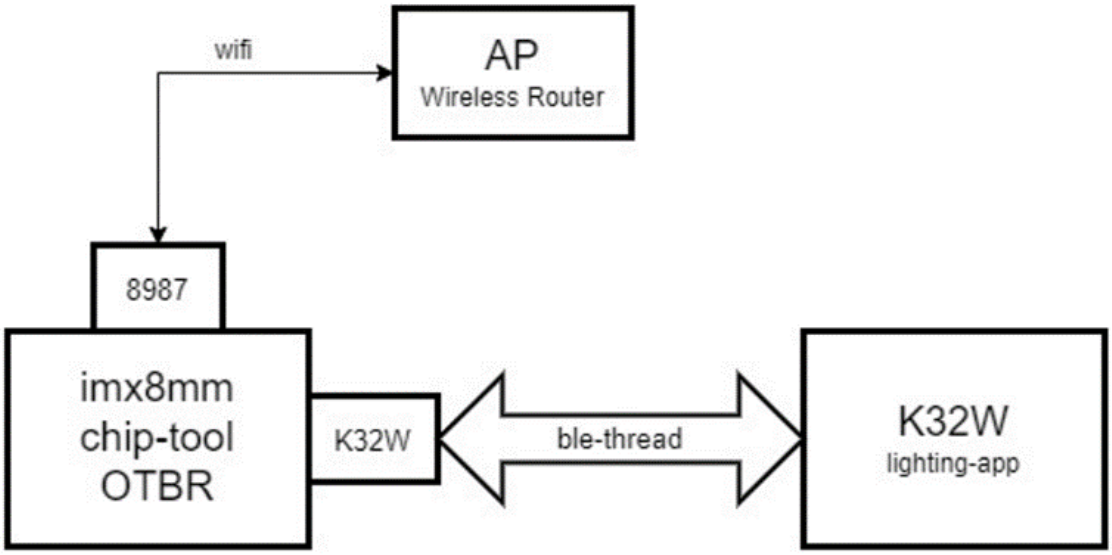


Figure 6-1 base demo for matter

6.1.1 Pairing commissionee (Lighting node) to commissioner (OTBR)

- Factory K32W061 Lighting-app
  Step1. Toggle reset
  Step2. Toggle Button SW2 on OM15802, After 6 seconds will cause the device to reset its persistent configuration and initiate a reboot.
  Step3. Press **USERINTERFACE** button to start BLE Advertising which is mandatory for device commissioning

- Pair (on controller, In the demo, the OTB as controller)
  Parameter after "**ble-thread**" is node-id (here is 1,can be any positive integer starting from 1) that should be set uniquely for each Matter device that joined the network, the last but one parameter is **pin code** (here fixed to 20202021), the last parameter is **discriminator** (here fixed to 3840)

```
$ chip-tool pairing ble-thread 1
hex:0e080000000000000000000300000b35060004001fffe00208dead00beef00cafe0708fddea
d00beef000005106fb5e8851e6985cc4f77b981eba5b94a030a4f70656e5468726561640102079f
04108fa1fc598d79f6c0d8fd77f3b44404b80c0402a0fff8 20202021 3840
```

  Normally the whole commissioning process will cost approximately 20s before completed. If there is "Device commissioning completed with success" message prompt on 8M Mini chip-tool log, which means the Matter device successfully joined the network, otherwise no further operation can be undertaken.

6.1.2 Control end device

```
$ chip-tool onoff toggle 1 1
```

Then can see the LED3 ON/OFF.

The last one parameter (here is 1) is node id that must be exactly the same as that used in previous step.

6.2 Android CHIP-TOOL

In the demo the android smartphone as controller, add the K32W end device in Matter Network. Android chip-tool build guide.

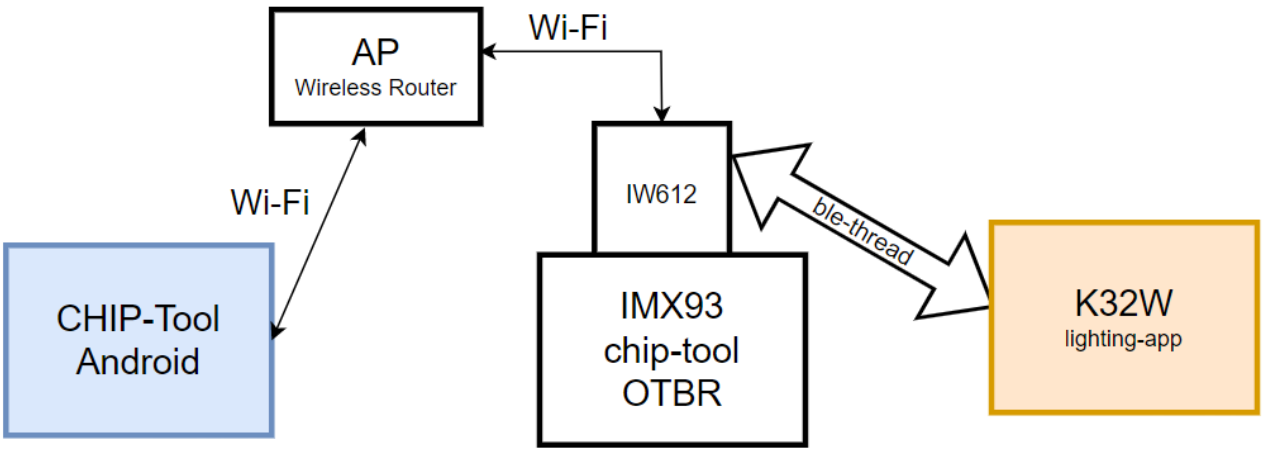


Figure 6-2 Android controller in Matter network

6.2.1 Confirm OTBR has been assigned an ipv6 address

- Confirm AP enable IPv6 DHCP server
- Confirm WAN supports ipv6(no WAN can ignore it)
- Confirm OTBR has ipv6 address, if only have one **inet6 fe80::xxx:xxx:...** mean that OTBR was not successfully assigned an IPv6 address.

```
$ ifconfig
mlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.3.169  netmask 255.255.254.0  broadcast 192.168.3.255
        inet6 2001:470:11e:f:c295:daff:fe00:e31f  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::c295:daff:fe00:e31f  prefixlen 64  scopeid 0x20<link>
        ether c0:95:da:00:e3:1f  txqueuelen 1000  (Ethernet)
        RX packets 23  bytes 2888 (2.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 41  bytes 7853 (7.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

6.2.2 Pairing commissionee (Lighting node) to commissioner (Android)

- Install android chip-tool apk

```
adb.exe install -r app-debug-nxp-v1.apk
```

- Factory K32W Lighting-app
- Chose **PROVISION CHIP DEVICE WITH THREAD**, then scan or input QR code. Any matter end device will carry a QR code which can be found in the boot log.

QR link :*https://project-chip.github.io/connectedhomeip/qrcode.html?data=MT%3A6FCJ142C00KA0648G00*

Figure 6-3 Matter Light QR code

Or Input "MT:**6FCJ142C00KA0648G00**" into smartphone like this.

Figure 6-4 input QR code into android chip-tool

- Click submit, then input OTBR network information. Should same with 5.1.3.
- If pair success, choice **LIGHT ON/OFF&LEVEL CLUSTER** to control the device.

Appendix

Revision History

Revision	Date	Description	Author
1.0	06/26/2022	Internal Release	Gnar Fang/Patrick Pa

Legal