Application Note

Enabling Matter on Sitara MPU

Krunal Bhargav, Randolph Sapp, Divyansh Mittal

ABSTRACT

This application note explores the implementation and usage of the Matter connectivity protocol on Sitara processor devices. The following sections outline the enablement and demonstration of Matter, including example data collected from SK-AM6X devices.

Table of Contents

List of Figures

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

Matter is an open-source application-layer connectivity protocol that specializes in creating a uniform method of interacting with IoT devices. It's built on top of IP allowing it to work natively over multiple network standards, such as WiFi (802.11), Ethernet (802.3), and Thread (802.15.4).

2 Current Implementation

The most common implementation of this protocol is the reference implementation present in the chip-tool in the connectedhomeip project at: https://github.com/project-chip/connectedhomeip. This repository contains:

- An implementation of the Matter server
- A definition of the messaging interface
- All the required networking utils for broadcasting and listening for broadcast events, including:
  - A mDNS server
  - A DNS resolver
- Tools for enabling bluetooth provisioning
- A definition of every possible endpoint cluster type
- An example for every endpoint cluster
- An example of a Controller / Administrator application

There are only two things that are importatant for a simple demo: an Administrator and an Endpoint. As such, the focus will be on the chip-tool and lock-app examples. Starting with chip-tool, this example application has a Command Line Interface (CLI) that acts as an Administrator capable of linking to endpoints and issuing commands or fetching status based on the clusters enabled by that endpoint. The lock-app is an example of an endpoint that would normally be controlling an electronic latch. This application registers a handful of commands like:

- Lock
- Unlock
- Unbolt
- GetUser
- SetUser
- GetDoorState
- SetDoorState
- SetCredential
- GetCredential

Where each of these commands are registered with chiptool and have accompanying log and state change messages that are broadcast when called.

3 Enablement

For our demo, we have used AM62P and AM62L. Any SoC working with Linux and able to connect to a network can be used in general. For more information, refer: AM62P and AM62L. With regards to software, the following steps may be used for cross-compilaton on a host PC:

1. On your Ubuntu host machine, download and install an SDK corresponding tho the kernel version you want to use, like: AM62L 11.00 SDK Installer.
2. Untar the rootfs inside the SDK using:

   ```
   cd <SDK Install Path>/filesystem/<device>
   tar -xf tisdk-default-image-am62lxx-evm.rootfs.tar.xz -C temp/
   ```

   Keep the the path of this extracted directory handy.
3. Clone and update the Matter repo using the following:

   ```
   git clone --recurse-submodules git@github.com:project-chip/connectedhomeip.git
   cd connectedhomeip
   git pull
   git submodule update --init
   ```

The repo size is large and will take some time to clone.

4. Download the build_matter_example.sh script and place inside the Matter's root directory:

```
#!/bin/bash
set -e

# =============================================================================
# Matter aarch64 Cross-Compilation Build Script (Unified)
#
# This script handles all necessary fixes and configurations:
# - Bluezoo dependency fix for Python 3.10
# - TI SDK toolchain wrapper creation
# - Complete Matter build process
# =============================================================================

if [[ $# -ne 1 ]]; then
  echo "Error: Please enter exactly one example-name as argument"
  echo "Usage: $0 <your_argument>"
  exit 1
fi

EXAMPLE_NAME="$1"

# =============================================================================
# CONFIGURATION - MODIFY THESE VARIABLES FOR YOUR SETUP
# =============================================================================

# SDK Path
SDK_PATH="/home/<user>/ti-processor-sdk-linux-am62lxx-evm-11.00.15.05"

# Path to your aarch64 sysroot
SYSROOT_AARCH64="$SDK_PATH/filesystem/am62lxx-evm/temp"     # TI SDK sysroot

# Toolchain binary prefix (TI SDK uses aarch64-oe-linux)
TOOLCHAIN_TARGET="aarch64-oe-linux"       # TI SDK compatible target

# Path to your aarch64 cross-compilation toolchain (using TI SDK native toolchain)
TOOLCHAIN_PREFIX="$SDK_PATH/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/
$TOOLCHAIN_TARGET"  # TI SDK toolchain

# Path to connectedhomeip repository (relative to script location)
REPO_PATH="."                               # CHANGE THIS if different

echo "=== Matter aarch64 Cross-Compilation Build Script (Unified) ==="
echo "Toolchain: $TOOLCHAIN_TARGET"
echo "Sysroot: $SYSROOT_AARCH64"
echo "Example: $EXAMPLE_NAME"
echo ""

echo "============================================================================="
echo "STEP 1: FIX BLUEZOO DEPENDENCY ISSUE"
echo "============================================================================="

REQUIREMENTS_FILE="$REPO_PATH/scripts/tests/requirements.txt"
if [ -f "$REQUIREMENTS_FILE" ]; then
    # Check if bluezoo is already commented out
    if grep -q "^bluezoo" "$REQUIREMENTS_FILE"; then
        echo "Commenting out bluezoo dependency (requires Python 3.11+)..."
        sed -i 's/^bluezoo/#bluezoo/' "$REQUIREMENTS_FILE"
        echo "✓ Bluezoo dependency commented out"
    else
        echo "✓ Bluezoo dependency already fixed"
    fi
else
    echo "Warning: Requirements file not found at $REQUIREMENTS_FILE"
fi

echo "============================================================================="
echo "STEP 2: VERIFY PATHS AND TOOLCHAIN"
echo "============================================================================="

# Construct toolchain binary paths
export CC_AARCH64="$TOOLCHAIN_PREFIX/${TOOLCHAIN_TARGET}-gcc"
export CXX_AARCH64="$TOOLCHAIN_PREFIX/${TOOLCHAIN_TARGET}-g++"
export AR_AARCH64="$TOOLCHAIN_PREFIX/${TOOLCHAIN_TARGET}-ar"
export STRIP_AARCH64="$TOOLCHAIN_PREFIX/${TOOLCHAIN_TARGET}-strip"
export LD_AARCH64="$TOOLCHAIN_PREFIX/${TOOLCHAIN_TARGET}-ld"
```

```
# Add toolchain to PATH for any remaining usage
export PATH="$TOOLCHAIN_PREFIX:$PATH"

if [ ! -f "$CC_AARCH64" ]; then
    echo "Error: Compiler not found at $CC_AARCH64"
    echo "Please check your TOOLCHAIN_PREFIX and TOOLCHAIN_TARGET variables"
    exit 1
fi

if [ ! -d "$SYSROOT_AARCH64" ]; then
    echo "Error: Sysroot not found at $SYSROOT_AARCH64"
    echo "Please check your SYSROOT_AARCH64 path"
    exit 1
fi

if [ ! -d "$REPO_PATH" ]; then
    echo "Error: Repository not found at $REPO_PATH"
    echo "Please check your REPO_PATH variable"
    exit 1
fi

echo "✓ Toolchain: $($CC_AARCH64 --version | head -1)"
echo "✓ Sysroot: $SYSROOT_AARCH64"
echo "✓ Repository: $REPO_PATH"

echo "============================================================================="
echo "STEP 3: CREATE TOOLCHAIN WRAPPER"
echo "============================================================================="

cd "$REPO_PATH"
#
# # Create toolchain wrapper directory
WRAPPER_DIR="$PWD/toolchain-wrapper-bin"
mkdir -p "$WRAPPER_DIR"

# Create symbolic links with the names GN expects
echo "Creating symbolic links for GN compatibility..."
ln -sf "$TOOLCHAIN_PREFIX/${TOOLCHAIN_TARGET}-gcc" "$WRAPPER_DIR/aarch64-linux-gnu-gcc"
ln -sf "$TOOLCHAIN_PREFIX/${TOOLCHAIN_TARGET}-g++" "$WRAPPER_DIR/aarch64-linux-gnu-g++"
ln -sf "$TOOLCHAIN_PREFIX/${TOOLCHAIN_TARGET}-ar" "$WRAPPER_DIR/aarch64-linux-gnu-ar"
ln -sf "$TOOLCHAIN_PREFIX/${TOOLCHAIN_TARGET}-strip" "$WRAPPER_DIR/aarch64-linux-gnu-strip"
ln -sf "$TOOLCHAIN_PREFIX/${TOOLCHAIN_TARGET}-ld" "$WRAPPER_DIR/aarch64-linux-gnu-ld"
ln -sf "$TOOLCHAIN_PREFIX/${TOOLCHAIN_TARGET}-objdump" "$WRAPPER_DIR/aarch64-linux-gnu-objdump"
ln -sf "$TOOLCHAIN_PREFIX/${TOOLCHAIN_TARGET}-nm" "$WRAPPER_DIR/aarch64-linux-gnu-nm"

echo "✓ Created toolchain wrapper directory: $WRAPPER_DIR"
echo "Contents:"
ls -la "$WRAPPER_DIR/"

# Add wrapper to PATH
export PATH="$PWD/toolchain-wrapper-bin:$PATH"

echo "============================================================================="
echo "STEP 4: SETUP BUILD ENVIRONMENT"
echo "============================================================================="

source scripts/activate.sh

echo "Testing cross-compilation..."
echo 'int main(){return 0;}' > test.c
$CC_AARCH64 --sysroot="$SYSROOT_AARCH64" -o test test.c
file test
rm test test.c
echo "✓ Cross-compilation test passed"

echo "============================================================================="
echo "STEP 5: CONFIGURE AND BUILD"
echo "============================================================================="

#Check if the example exists
if [ ! -d "examples/${EXAMPLE_NAME}" ]; then
    echo -e "No such '$EXAMPLE_NAME' exists in examples!! \nExiting !!"
    exit 1
#Check if example does not need specific platform like linux to build
elif [[ $(find ./examples/ -maxdepth 2 -type f -name
args.gni | grep -c "$EXAMPLE_NAME") -gt 0 ]]; then
    ROOT_PATH="examples/$EXAMPLE_NAME"
#Check if example needs specific platform to build and linux platform is available
```

```
elif [[ $(find ./examples/ -type f -name "args.gni"
-path "*/linux/*" | grep -c "$EXAMPLE_NAME") -gt 0 ]]; then
    ROOT_PATH="examples/$EXAMPLE_NAME/linux"
#Check if example needs specific platform to build but linux platform is NOT available
else
    echo -e "'$EXAMPLE_NAME' is not supported on Linux!! \nExiting !!"
    exit 1
fi

gn gen "out/${EXAMPLE_NAME}-arm64" --root="$ROOT_PATH" --args="
  target_cpu=\"arm64\"
  target_os=\"linux\"
  sysroot=\"$SYSROOT_AARCH64\"
  is_clang=false
  treat_warnings_as_errors=false
  target_cflags = [
    \"-D_GNU_SOURCE\",
    \"-D__USE_GNU\",
    \"-pthread\",
    \"-DCHIP_DEVICE_CONFIG_WIFI_STATION_IF_NAME=\\\"wlan0\\\"\",
    \"-DCHIP_DEVICE_CONFIG_LINUX_DHCPC_CMD=\\\"udhcpc -b -i %s \\\"\",
  ]
  target_ldflags=[\"-pthread\"]
"

echo "Building $EXAMPLE_NAME..."
ninja -C "out/${EXAMPLE_NAME}-arm64"

echo "============================================================================="
echo "STEP 6: VERIFY BUILD RESULTS"
echo "============================================================================="
EXECUTABLE_NAME=$(awk -F'"' '/executable\("/ {print $2}' $ROOT_PATH/BUILD.gn)
echo "Expected executable name: $EXECUTABLE_NAME"
BINARY_PATH="out/${EXAMPLE_NAME}-arm64/${EXECUTABLE_NAME}"
if [ -f "$BINARY_PATH" ]; then
    file "$BINARY_PATH"
    echo "✓ Build complete! Binary located at: $BINARY_PATH"
else
    echo "Error: Build failed, binary not found at $BINARY_PATH"
    exit 1
fi

echo "============================================================================="
echo "BUILD SUMMARY"
echo "============================================================================="
echo "Target: aarch64 (ARM64)"
echo "Toolchain: $TOOLCHAIN_TARGET"
echo "Example: $EXAMPLE_NAME"
echo "Output: out/${EXAMPLE_NAME}-arm64/${EXECUTABLE_NAME}"
echo ""
echo "All fixes applied:"
echo "✓ Bluezoo dependency commented out for Python 3.10 compatibility"
echo "✓ TI SDK native toolchain configured for compatibility"
echo "✓ Toolchain wrapper created for GN naming conventions"
echo "✓ Matter build completed successfully"
echo ""
echo "To modify configuration, edit the variables at the top of this script:"
echo "- TOOLCHAIN_PREFIX: $TOOLCHAIN_PREFIX"
echo "- SYSROOT_AARCH64: $SYSROOT_AARCH64"
echo "- TOOLCHAIN_TARGET: $TOOLCHAIN_TARGET"
echo "- EXAMPLE_NAME: $EXAMPLE_NAME"
echo ""
echo "To build a different example, run the script with other example's name as argument."
```

5. Modify the build_matter_example.sh to update the following variables:

   a. SDK_PATH -> Path of installed Processor SDK.
   b. SYSROOT_AARCH64 -> Path of extracted filesystem in Processor SDK.

6. Run the script using:

```
# sudo ./build_matter_example.sh <example-name>
# For example:
sudo ./build_matter_example.sh chip-tool
sudo ./build_matter_example.sh lock-app
```

7. **Note**: On running the above script, if you are stuck on "STEP 4: SETUP BUILD ENVIRONMENT" or your environment is out-of-date, exit the script and run :

   ```
   sudo -E bash scripts/bootstrap.sh
   ```

   This will re-create the environment from scratch and will take some time. Re-run the build_matter_example.sh again.
8. Upon successful build, script will mention the path of output binary: <Matter root>/out/<example_directory>/<example_executable>.
9. Copy the executable to your target. In our example we have copied 'chip-lock-app' on AM62L and 'chip-tool' on AM62P with both devices connected on the same network.

The experiments in this application note has been tested with the following version of SDK/repositories in the last revision:

Processor SDK	11.00.15.05
Matter Repo	Commit: e156205783 (master branch)

4 Demonstration

Figure 4-1 shows AM62L using the lock-app and AM62P using the chip-tool interfacing with each other over Ethernet.

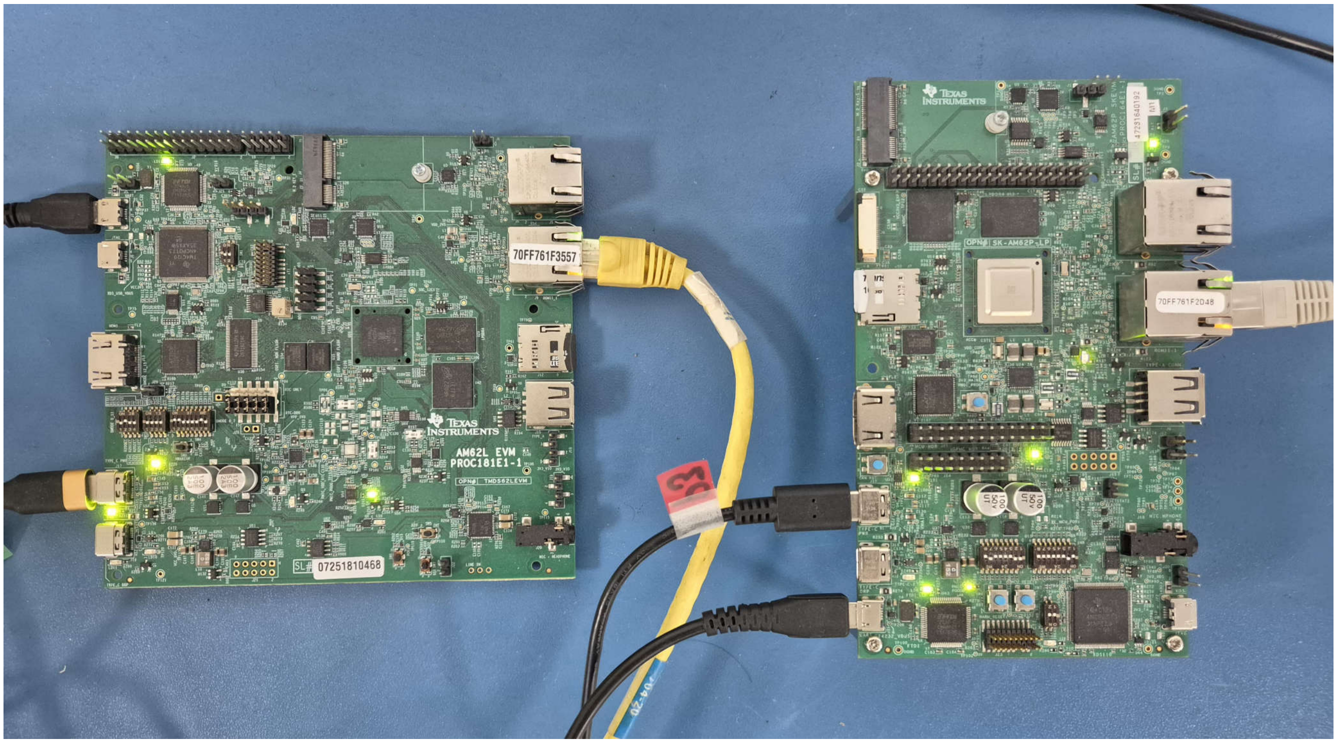
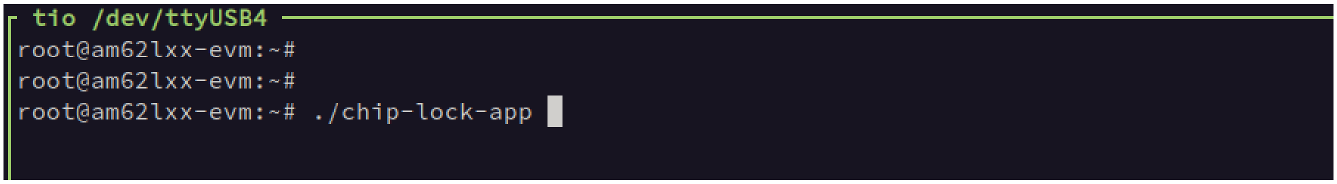


Figure 4-1. Hardware Setup

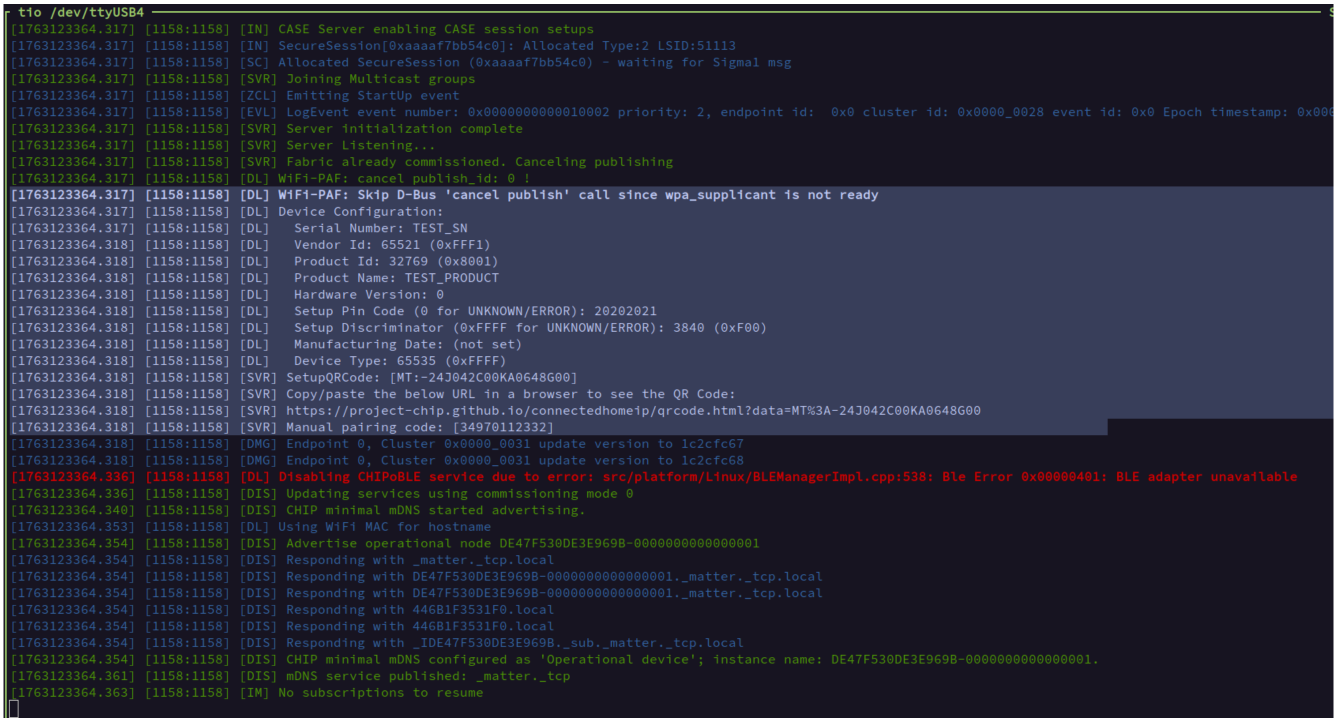
Figure 4-2 shows how to setup the AM62L device as an endpoint using the lock app.



```
tio /dev/ttyUSB4
root@am62lxx-evm:~#
root@am62lxx-evm:~#
root@am62lxx-evm:~# ./chip-lock-app
```

Figure 4-2. Creating an Endpoint

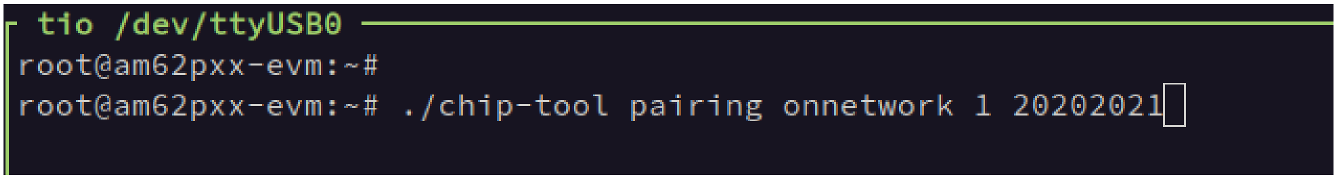
Figure 4-3 shows what an expected endpoint log should look like. Note the device configuration information.



```
tio /dev/ttyUSB4
[1763123364.317] [1158:1158] [IN] CASE Server enabling CASE session setups
[1763123364.317] [1158:1158] [IN] SecureSession[0xaaaaf7bb54c0]: Allocated Type:2 LSID:51113
[1763123364.317] [1158:1158] [SC] Allocated SecureSession (0xaaaaf7bb54c0) - waiting for Sigma1 msg
[1763123364.317] [1158:1158] [SVR] Joining Multicast groups
[1763123364.317] [1158:1158] [ZCL] Emitting StartUp event
[1763123364.317] [1158:1158] [EVL] LogEvent event number: 0x0000000000010002 priority: 2, endpoint id:  0x0 cluster id: 0x0000_0028 event id: 0x0 Epoch timestamp: 0x000
[1763123364.317] [1158:1158] [SVR] Server initialization complete
[1763123364.317] [1158:1158] [SVR] Server Listening...
[1763123364.317] [1158:1158] [SVR] Fabric already commissioned. Canceling publishing
[1763123364.317] [1158:1158] [DL] WiFi-PAF: cancel publish_id: 0 !
[1763123364.317] [1158:1158] [DL] WiFi-PAF: Skip D-Bus 'cancel publish' call since wpa_supplicant is not ready
[1763123364.317] [1158:1158] [DL] Device Configuration:
[1763123364.317] [1158:1158] [DL]   Serial Number: TEST_SN
[1763123364.318] [1158:1158] [DL]   Vendor Id: 65521 (0xFFF1)
[1763123364.318] [1158:1158] [DL]   Product Id: 32769 (0x8001)
[1763123364.318] [1158:1158] [DL]   Product Name: TEST_PRODUCT
[1763123364.318] [1158:1158] [DL]   Hardware Version: 0
[1763123364.318] [1158:1158] [DL]   Setup Pin Code (0 for UNKNOWN/ERROR): 20202021
[1763123364.318] [1158:1158] [DL]   Setup Discriminator (0xFFFF for UNKNOWN/ERROR): 3840 (0xF00)
[1763123364.318] [1158:1158] [DL]   Manufacturing Date: (not set)
[1763123364.318] [1158:1158] [DL]   Device Type: 65535 (0xFFFF)
[1763123364.318] [1158:1158] [SVR] SetupQRCode: [MT:-24J042C00KA0648G00]
[1763123364.318] [1158:1158] [SVR] Copy/paste the below URL in a browser to see the QR Code:
[1763123364.318] [1158:1158] [SVR] https://project-chip.github.io/connectedhomeip/qrcode.html?data=MT%3A-24J042C00KA0648G00
[1763123364.318] [1158:1158] [SVR] Manual pairing code: [34970112332]
[1763123364.318] [1158:1158] [DMG] Endpoint 0, Cluster 0x0000_0031 update version to 1c2cfc67
[1763123364.318] [1158:1158] [DMG] Endpoint 0, Cluster 0x0000_0031 update version to 1c2cfc68
[1763123364.336] [1158:1158] [DL] Disabling CHIPoBLE service due to error: src/platform/Linux/BLEManagerImpl.cpp:538: Ble Error 0x00000401: BLE adapter unavailable
[1763123364.336] [1158:1158] [DIS] Updating services using commissioning mode 0
[1763123364.340] [1158:1158] [DIS] CHIP minimal mDNS started advertising.
[1763123364.353] [1158:1158] [DL] Using WiFi MAC for hostname
[1763123364.354] [1158:1158] [DIS] Advertise operational node DE47F530DE3E969B-0000000000000001
[1763123364.354] [1158:1158] [DIS] Responding with _matter._tcp.local
[1763123364.354] [1158:1158] [DIS] Responding with DE47F530DE3E969B-0000000000000001._matter._tcp.local
[1763123364.354] [1158:1158] [DIS] Responding with DE47F530DE3E969B-0000000000000001._matter._tcp.local
[1763123364.354] [1158:1158] [DIS] Responding with 446B1F3531F0.local
[1763123364.354] [1158:1158] [DIS] Responding with 446B1F3531F0.local
[1763123364.354] [1158:1158] [DIS] Responding with _IDE47F530DE3E969B._sub._matter._tcp.local
[1763123364.354] [1158:1158] [DIS] CHIP minimal mDNS configured as 'Operational device'; instance name: DE47F530DE3E969B-0000000000000001.
[1763123364.361] [1158:1158] [DIS] mDNS service published: _matter._tcp
[1763123364.363] [1158:1158] [IM] No subscriptions to resume
```

Figure 4-3. Expected Endpoint Log

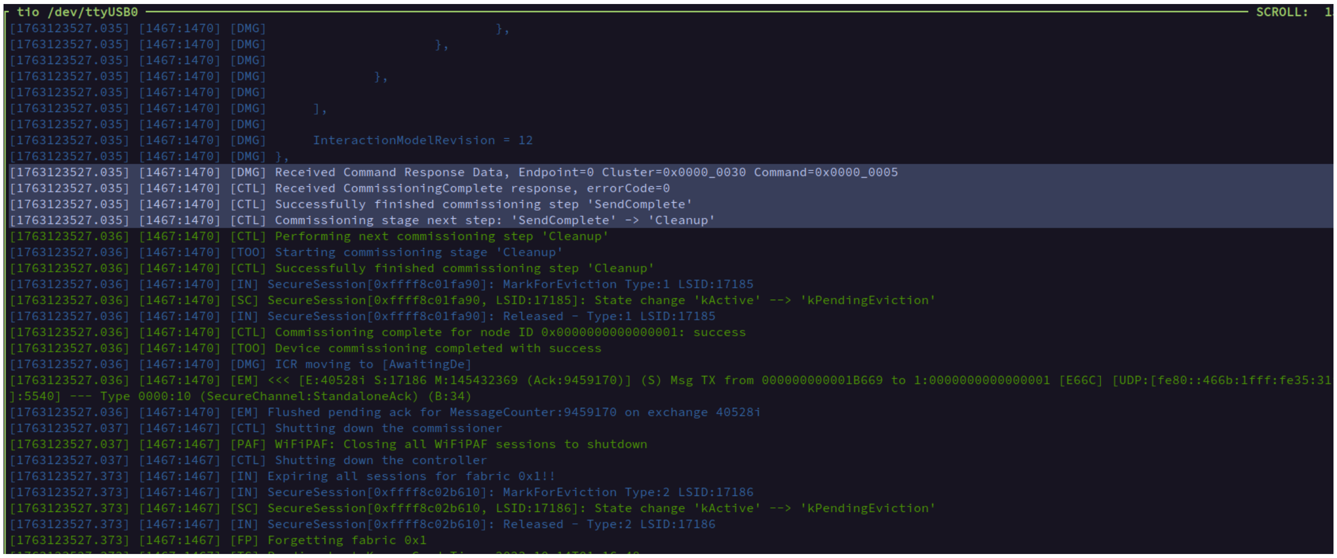
Figure 4-4 shows how an administrator would pair with the endpoint using the chip-tool.



```
tio /dev/ttyUSB0
root@am62pxx-evm:~#
root@am62pxx-evm:~# ./chip-tool pairing onnetwork 1 20202021
```

Figure 4-4. Pairing With Endpoint Device

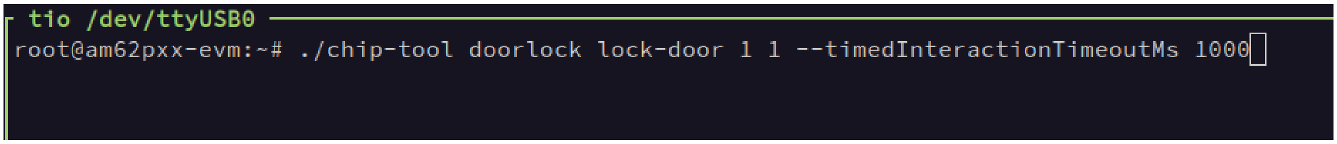
Figure 4-5 shows the expect log of a successful pairing attempt. Note the CommissioningComplete response in the log.



```
tio /dev/ttyUSB0                                                                                                  SCROLL:  1
[1763123527.035] [1467:1470] [DMG]                              },
[1763123527.035] [1467:1470] [DMG]                      },
[1763123527.035] [1467:1470] [DMG]
[1763123527.035] [1467:1470] [DMG]              },
[1763123527.035] [1467:1470] [DMG]
[1763123527.035] [1467:1470] [DMG]      ],
[1763123527.035] [1467:1470] [DMG]
[1763123527.035] [1467:1470] [DMG]      InteractionModelRevision = 12
[1763123527.035] [1467:1470] [DMG] },
[1763123527.035] [1467:1470] [DMG] Received Command Response Data, Endpoint=0 Cluster=0x0000_0030 Command=0x0000_0005
[1763123527.035] [1467:1470] [CTL] Received CommissioningComplete response, errorCode=0
[1763123527.035] [1467:1470] [CTL] Successfully finished commissioning step 'SendComplete'
[1763123527.035] [1467:1470] [CTL] Commissioning stage next step: 'SendComplete' -> 'Cleanup'
[1763123527.036] [1467:1470] [CTL] Performing next commissioning step 'Cleanup'
[1763123527.036] [1467:1470] [TOO] Starting commissioning stage 'Cleanup'
[1763123527.036] [1467:1470] [CTL] Successfully finished commissioning step 'Cleanup'
[1763123527.036] [1467:1470] [IN] SecureSession[0xffff8c01fa90]: MarkForEviction Type:1 LSID:17185
[1763123527.036] [1467:1470] [SC] SecureSession[0xffff8c01fa90, LSID:17185]: State change 'kActive' --> 'kPendingEviction'
[1763123527.036] [1467:1470] [IN] SecureSession[0xffff8c01fa90]: Released - Type:1 LSID:17185
[1763123527.036] [1467:1470] [CTL] Commissioning complete for node ID 0x0000000000000001: success
[1763123527.036] [1467:1470] [TOO] Device commissioning completed with success
[1763123527.036] [1467:1470] [DMG] ICR moving to [AwaitingDe]
[1763123527.036] [1467:1470] [EM] <<< [E:40528i S:17186 M:145432369 (Ack:9459170)] (S) Msg TX from 000000000001B669 to 1:0000000000000001 [E66C] [UDP:[fe80::466b:1fff:fe35:31
]:5540] --- Type 0000:10 (SecureChannel:StandaloneAck) (B:34)
[1763123527.036] [1467:1470] [EM] Flushed pending ack for MessageCounter:9459170 on exchange 40528i
[1763123527.037] [1467:1467] [CTL] Shutting down the commissioner
[1763123527.037] [1467:1467] [PAF] WiFiPAF: Closing all WiFiPAF sessions to shutdown
[1763123527.037] [1467:1467] [CTL] Shutting down the controller
[1763123527.373] [1467:1467] [IN] Expiring all sessions for fabric 0x1!!
[1763123527.373] [1467:1467] [IN] SecureSession[0xffff8c02b610]: MarkForEviction Type:2 LSID:17186
[1763123527.373] [1467:1467] [SC] SecureSession[0xffff8c02b610, LSID:17186]: State change 'kActive' --> 'kPendingEviction'
[1763123527.373] [1467:1467] [IN] SecureSession[0xffff8c02b610]: Released - Type:2 LSID:17186
[1763123527.373] [1467:1467] [FP] Forgetting fabric 0x1
```

Figure 4-5. Successful Pairing

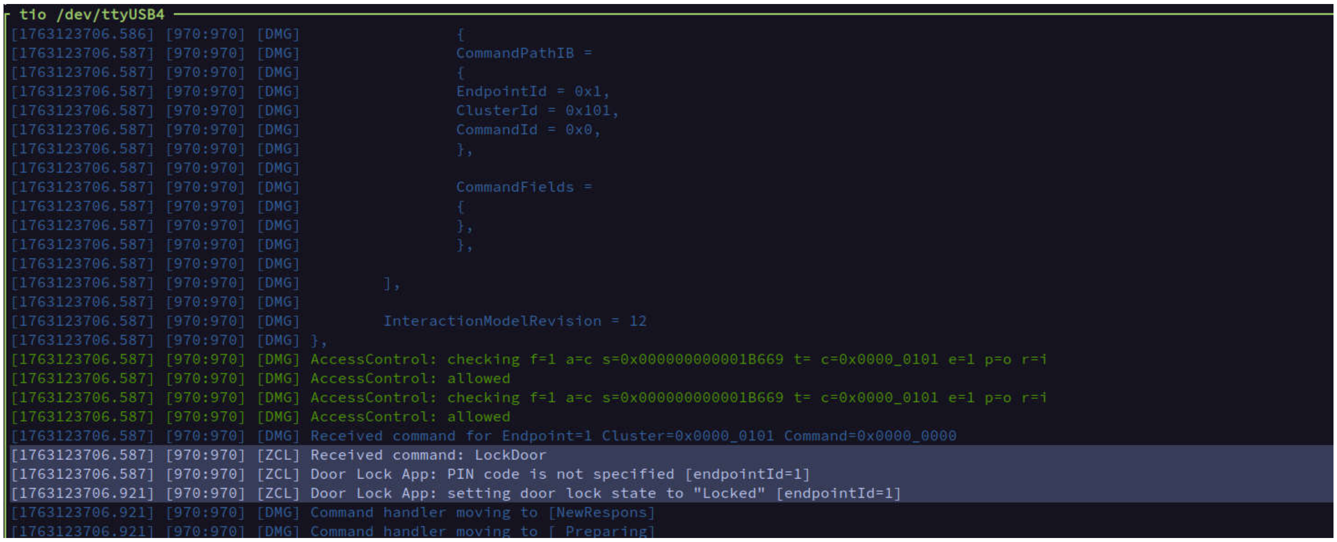
Figure 4-6 shows that the status of the endpoint is set to be locked.



```
tio /dev/ttyUSB0
root@am62pxx-evm:~# ./chip-tool doorlock lock-door 1 1 --timedInteractionTimeoutMs 1000
```

Figure 4-6. Setting Lock Status to Locked

Figure 4-7 shows the status reported on the endpoint following the lock-door request.



```
tio /dev/ttyUSB4
[1763123706.586] [970:970] [DMG]                 {
[1763123706.587] [970:970] [DMG]                 CommandPathIB =
[1763123706.587] [970:970] [DMG]                 {
[1763123706.587] [970:970] [DMG]                 EndpointId = 0x1,
[1763123706.587] [970:970] [DMG]                 ClusterId = 0x101,
[1763123706.587] [970:970] [DMG]                 CommandId = 0x0,
[1763123706.587] [970:970] [DMG]                 },
[1763123706.587] [970:970] [DMG]
[1763123706.587] [970:970] [DMG]                 CommandFields =
[1763123706.587] [970:970] [DMG]                 {
[1763123706.587] [970:970] [DMG]                 },
[1763123706.587] [970:970] [DMG]                 },
[1763123706.587] [970:970] [DMG]
[1763123706.587] [970:970] [DMG]         ],
[1763123706.587] [970:970] [DMG]
[1763123706.587] [970:970] [DMG]         InteractionModelRevision = 12
[1763123706.587] [970:970] [DMG] },
[1763123706.587] [970:970] [DMG] AccessControl: checking f=1 a=c s=0x000000000001B669 t= c=0x0000_0101 e=1 p=o r=i
[1763123706.587] [970:970] [DMG] AccessControl: allowed
[1763123706.587] [970:970] [DMG] AccessControl: checking f=1 a=c s=0x000000000001B669 t= c=0x0000_0101 e=1 p=o r=i
[1763123706.587] [970:970] [DMG] AccessControl: allowed
[1763123706.587] [970:970] [DMG] Received command for Endpoint=1 Cluster=0x0000_0101 Command=0x0000_0000
[1763123706.587] [970:970] [ZCL] Received command: LockDoor
[1763123706.587] [970:970] [ZCL] Door Lock App: PIN code is not specified [endpointId=1]
[1763123706.921] [970:970] [ZCL] Door Lock App: setting door lock state to "Locked" [endpointId=1]
[1763123706.921] [970:970] [DMG] Command handler moving to [NewRespons]
[1763123706.921] [970:970] [DMG] Command handler moving to [ Preparing]
```

Figure 4-7. Lock Status in Endpoint Log

To see a recorded demonstration of the above with full endpoint and administrator logs being updated in sync, see the following: https://asciinema.org/a/755835.

5 Summary

The main goal of this application note is to demonstrate how to compile a reference implementation of matter from the connectedhomeip project and run a simple lock/unlock demo. Even though a AM62L and AM62P devices is used, the above instructions are applicable to any ARM32 bit and ARM64-bit TI processors.

6 References

- Texas Instruments, *AM62P*, product folder.
- Texas Instruments, *AM62L*, product folder.

7 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from January 30, 2024 to November 21, 2025 (from Revision * (January 2024) to Revision A (November 2025)) Page

- Updated Matter enablement from Yocto based to Cross-Compilation based approach....................................2
- Modified the images and console output to reflect results with latest Matter revision.......................................6

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale, TI's General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.