

Application Note

在 Sitara MPU 上启用 Matter



Krunal Bhargav, Randolph Sapp, Divyansh Mittal

摘要

本应用手册探讨了如何在 Sitara 处理器器件上实现和使用 Matter 连接协议。以下各节概述了 Matter 的实现和演示，提供了从 SK-AM6X 器件收集的示例数据。

内容

1 简介.....	2
2 当前设计.....	2
3 支持.....	2
4 演示.....	6
5 总结.....	8
6 参考资料.....	9
7 修订历史记录.....	9

插图清单

图 4-1. 硬件设置.....	6
图 4-2. 创建端点.....	6
图 4-3. 预期端点日志.....	7
图 4-4. 与端点器件配对.....	7
图 4-5. 成功配对.....	8
图 4-6. 将锁定状态设置为已锁定.....	8
图 4-7. 端点日志中的锁定状态.....	8

商标

所有商标均为其各自所有者的财产。

1 简介

Matter 是一种开源应用层连接协议，专门用于创建与物联网设备交互的统一方法。它构建在 IP 基础之上，这使其原生适用于多种网络标准，例如 WiFi (802.11)、以太网 (802.3) 和 Thread (802.15.4)。

2 当前设计

此协议常见的实现方式是 **connectedhomeip** 项目的 **chip-tool** 中的参考设计，网址为：<https://github.com/project-chip/connectedhomeip>。该存储库包含：

- **Matter** 服务器的实现方式
- 消息传递接口的定义
- 广播和侦听广播事件所需的所有网络实用程序，包括：
 - mDNS 服务器
 - DNS 解析器
- 用于启用蓝牙配置的工具
- 每种可能的端点群集类型的定义
- 每个端点群集的示例
- 控制器/管理员应用的示例

对于一个简单的演示来说，只有两个方面很重要：管理员和端点。因此，我们将重点介绍 **chip-tool** 和锁定应用示例。从 **chip-tool** 开始，此示例应用有一个命令行界面 (CLI)，它充当管理员，能够链接到端点并根据该端点启用的集群发出命令或获取状态。**lock-app** 是端点的一个例子，通常用于控制电子锁存器。此应用会注册一些命令，例如：

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

其中每个命令都在 **chiptool** 中注册，并附带日志和状态更改消息，这些消息在调用时会广播。

3 支持

在演示中，我们使用了 **AM62P** 和 **AM62L**。一般情况下，可以使用任何支持 Linux 并可连接网络的 SoC。如需更多信息，请参阅：[AM62P](#) 和 [AM62L](#)。关于软件，可按照以下步骤在主机 PC 上进行交叉编译：

1. 在 Ubuntu 主机上，下载并安装与您要使用的内核版本相对应的 SDK，例如：[AM62L 11.00 SDK 安装程序](#)。
2. 使用以下命令，解压缩 SDK 中的 rootfs：

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

妥善留存此解压缩目录的路径。

3. 使用以下命令，克隆并更新 **Matter** 存储库：

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

存储库体积较大，克隆过程需要耗费一些时间。

4. 下载 [build_matter_example.sh](#) 脚本，并将其放在 **Matter** 的根目录中：

```
#!/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_DHCP_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. 修改 build_matter_example.sh , 以更新以下变量 :

- a. SDK_PATH -> 安装的处理器的 SDK 的路径。
- b. SYSROOT_AARCH64 -> 处理器 SDK 中提取的文件系统的路径。

6. 使用以下命令运行脚本 :

```

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

```

7. **注意：**在运行上述脚本时，如果您卡在“第 4 步：设置构建环境”，或您的环境已过时，请退出脚本并运行以下命令：

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

这会从头开始重新创建环境，并将耗费一些时间。再次运行 `build_matter_example.sh`。

8. 成功构建后，脚本将提及输出二进制文件的路径：`<Matter root>/out/<example_directory>/<example_executable>`。
9. 将可执行文件复制到目标。在示例中，我们复制了 'chip-lock-app' (在 AM62L 上) 和 'chip-tool' (在 AM62P 上)，两个器件连接了相同的网络。

本应用手册中的实验，在最新版本修订时已通过以下 SDK/存储库版本的测试验证：

处理器 SDK 11.00.15.05

Matter 存储库 提交：e156205783 (主分支)

4 演示

图 4-1 展示了使用锁应用的 AM62L 和使用芯片工具的 AM62P，两个器件通过以太网相互连接。

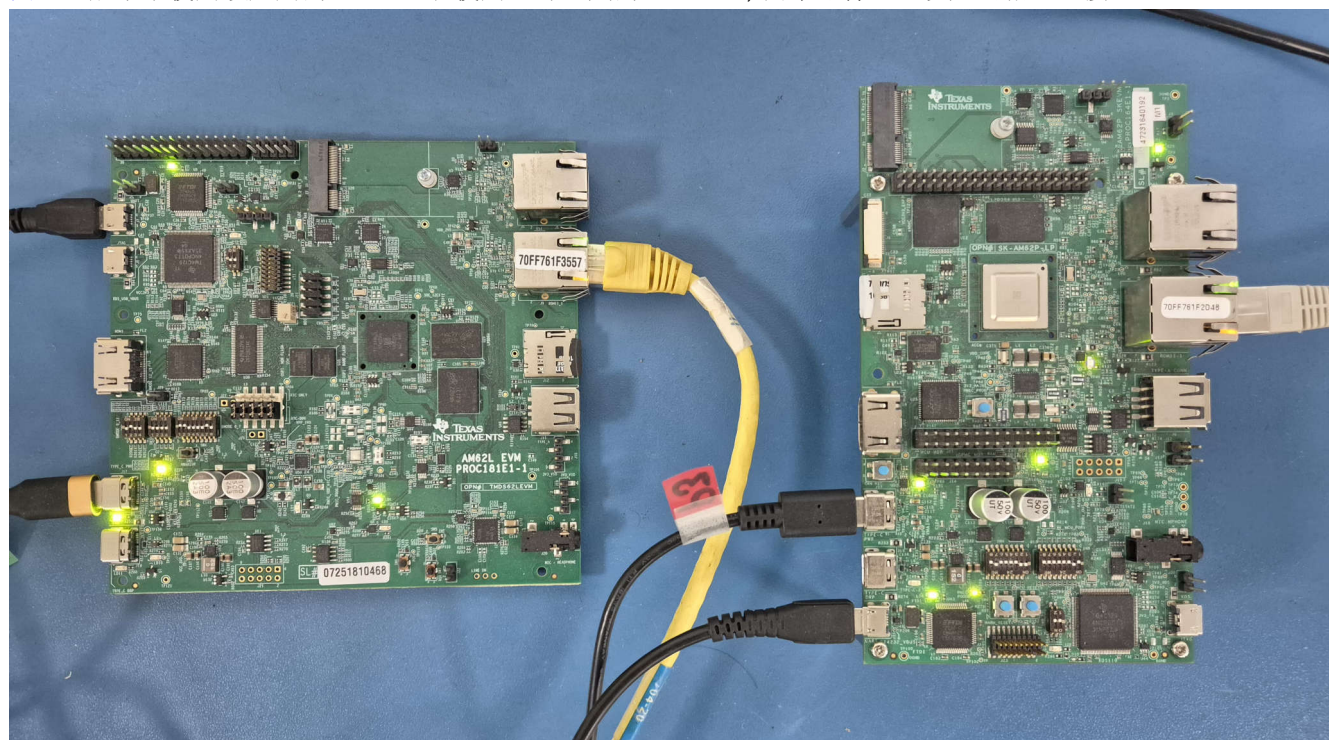


图 4-1. 硬件设置

图 4-2 显示了如何使用锁定应用将 AM62L 器件设置为端点。



图 4-2. 创建端点

图 4-3 显示了预期的端点日志。请注意器件配置信息。

```

t1o /dev/ttyUSB4
[1763123364.317] [1158:1158] [IN] CASE Server enabling CASE session setups
[1763123364.317] [1158:1158] [IN] SecureSession[0xaaaaaf7bb54c0]: Allocated Type:2 LSID:51113
[1763123364.317] [1158:1158] [SC] Allocated SecureSession (0xaaaaaf7bb54c0) - waiting for Signal 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: 0x0000000000001002 priority: 2, endpoint id: 0x0 cluster id: 0x0000_0028 event id: 0x0 Epoch timestamp: 0x00
[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 (0xFFFF1)
[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

```

图 4-3. 预期端点日志

图 4-4 显示了管理员如何使用芯片工具与端点配对。

```

t1o /dev/ttyUSB0
root@am62pxx-evm:~#
root@am62pxx-evm:~# ./chip-tool pairing onnetwork 1 20202021

```

图 4-4. 与端点器件配对

图 4-5 显示预期的成功配对尝试日志。注意日志中的 CommissioningComplete 响应。

```
tio /dev/ttyUSB0
[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:495281 S:17186 M:145432369 (Ack:9459170)] (S) Msg TX from 0000000000001B669 to 1:0000000000000001 [E66C] [UDP:{fe80::466b:1fff:fe35:31:5540} --- Type 0900:19 (SecureChannel:StandaloneAck) (B:34)
[1763123527.036] [1467:1470] [EM] Flushed pending ack for MessageCounter:9459170 on exchange 405281
[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
[1763123527.037] [1467:1467] [IN] Expiring all sessions for fabric 0x1!
[1763123527.037] [1467:1467] [IN] SecureSession[0xffff8c02b610]: MarkForEviction Type:2 LSID:17186
[1763123527.037] [1467:1467] [SC] SecureSession[0xffff8c02b610, LSID:17186]: State change 'kActive' --> 'kPendingEviction'
[1763123527.037] [1467:1467] [IN] SecureSession[0xffff8c02b610]: Released - Type:2 LSID:17186
[1763123527.037] [1467:1467] [FP] Forgetting fabric 0x1
```

图 4-5. 成功配对

图 4-6 显示端点的状态设置为锁定。

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

图 4-6. 将锁定状态设置为已锁定

图 4-7 显示了门锁请求之后在端点上报告的状态。

```
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] 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] InteractionModelRevision = 12
[1763123706.587] [970:970] [DMG] },
[1763123706.587] [970:970] [DMG] AccessControl: checking f=1 a=c s=0x0000000000001B669 t= c=0x0000_0101 e=1 p=o r=1
[1763123706.587] [970:970] [DMG] AccessControl: allowed
[1763123706.587] [970:970] [DMG] AccessControl: checking f=1 a=c s=0x0000000000001B669 t= c=0x0000_0101 e=1 p=o r=1
[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 [NewResponses]
[1763123706.921] [970:970] [DMG] Command handler moving to [Preparing]
```

图 4-7. 端点日志中的锁定状态

要查看上述内容的录制演示以及同步更新的完整端点和管理员日志，请参阅以下内容：<https://asciinema.org/a/755835>。

5 总结

本应用手册的主要目标是演示如何从 `connectedhomeip` 工程编译 `matter` 的参考设计，并运行简单的锁定/解锁演示。尽管使用的是 AM62L 和 AM62P 器件，上述说明适用于任何 ARM 32 位和 ARM 64 位 TI 处理器。

6 参考资料

- 德州仪器 (TI) , [AM62P](#) 产品文件夹。
- 德州仪器 (TI) , [AM62L](#) 产品文件夹。

7 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

Changes from JANUARY 30, 2024 to NOVEMBER 21, 2025 (from Revision * (January 2024) to Revision A (November 2025))		Page
• 将 Matter 实现方法从基于 Yocto 更新为基于交叉编译。.....		2
• 修改了图像和控制台输出，以反映具有最新 Matter 版本的结果。.....		6

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 [TI 销售条款](#)、[TI 通用质量指南](#) 或 [ti.com](#) 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

版权所有 © 2025，德州仪器 (TI) 公司

最后更新日期：2025 年 10 月