MPC-3000 Series Linux Software User Manual

Version 1.0, October 2024

www.moxa.com/products

MPC-3000 Series Linux Software User Manual

The software described in this manual is furnished under a license agreement and may be used only in accordance with the terms of that agreement.

Copyright Notice

Trademarks

Disclaimer

- Information in this document is subject to change without notice and does not represent a commitment on the part of Moxa.
- Moxa provides this document as is, without warranty of any kind, either expressed or implied, including, but not limited to, its particular purpose. Moxa reserves the right to make improvements and/or changes to this manual, or to the products and/or the programs described in this manual, at any time.
- Information provided in this manual is intended to be accurate and reliable. However, Moxa assumes no responsibility for its use, or for any infringements on the rights of third parties that may result from its use.
- This product might include unintentional technical or typographical errors. Changes are periodically made to the information herein to correct such errors, and these changes are incorporated into new editions of the publication.

Technical Support Contact Information

www.moxa.com/support

Table of Contents

1. Introduction

This manual will help Linux users on MPC-3000 computers understand and navigate Linux utilities and the standard Linux operating system.

The following sections contain comprehensive information on getting started, x86 Linux SDK wizard, peripheral interface Operations, basic Linux concepts, and troubleshooting.

2. Getting Started

The Getting Started section will introduce the Linux OS distribution installation instructions.

Linux OS Installation Instructions

Preparing a bootable USB Drive

At first, prepare a **USB storage drive**, download the Rufus to create bootable USB drive. Download the ISO image file and restore ISO image into USB storage drive.

Linux Distributions Supported

- **Debian**
  - Debian 11 (bullseye), Linux kernel 5.10
  - Debian 12 (bookworm), Linux kernel 6.1
  - Official Debian installation guide
- **Ubuntu**
  - Ubuntu 20.04 LTS (Focal Fossa), Linux kernel 5.4 (20.04.1), Linux kernel 5.15 (20.04.5), HWE kernel 5.15 or later version
  - Ubuntu 22.04 LTS (Jammy Jellyfish), Linux kernel 5.15 (22.04.3), Linux kernel 6.5 (22.04.4), HWE kernel 6.5 or later version
  - Official Ubuntu installation guide
- **RedHat**
  - RedHat 9, Linux kernel 5.14
    - Official RedHat 9 download link
    - Official RedHat 9 installation guide

Entering the BIOS Menu

Boot up device and press **F2** key from keyboard to enter BIOS menu, and select **boot from USB** from **UEFI mode**.

Then follow the distribution's official installation guide to finish OS installation procedure.

3. x86 Linux SDK Wizard

Basic Information

The **Moxa x86 Linux SDK** enables the easy deployment on the Moxa x86 IPC platform. The SDK contains components for peripheral drivers, peripheral control tools and configuration files.

It also provides deployment features, such as build & installation log, dry-run, and self-test on target model. Users can download the Moxa x86 Linux SDK zip file from official product's website.

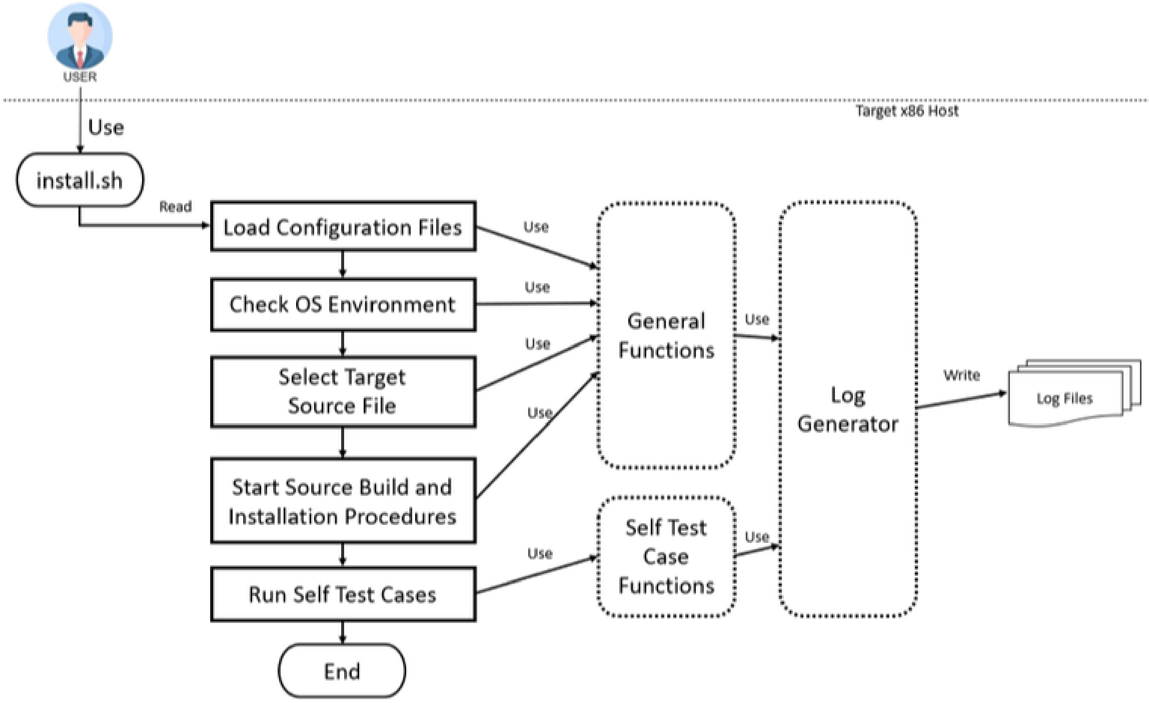
Below is the list of files:

- ***.tgz -** the tarball file of x86 Linux SDK Install Wizard
- **README.docx/README.md -** the user manual of x86 Linux SDK Install Wizard
- **sources_list -** the list of source code
- **build_info -** build information

NOTE

Please extract the **tgz** tarball file under Linux OS environment to avoid file permission issue.

Software Flow Diagram



User Interface

User Interface	Main Command	Sub Command	Option	Description
`install.sh`				Start to install all procedures (default)
			-y, --yes	Automatic yes to prompts
	-h, --help			Display the help menu
	-v, --version			Display the version information
	-s, --selftest			Run the self-test cases
	--uninstall			Uninstall driver and tool
	--dry-run			It won't perform the installation, list available driver and tool only
			--force	Install driver and tool even if the version is the same or older (default is to install newer version)

Before Starting Installation

- Please configure your **network settings** before installation
- To extract the tgz tarball file under Linux environment (e.g. **tar xvf *.tar.gz**)
- Run **--dry-run** option before installation, to check the target host device and environment are available
- Run **--selftest** option after installation, to check the status of drivers and tools

4. Peripheral Interface Operations

This guide is introduced the usage of **Moxa peripheral interface control utility**. These utilities should be installed after the x86 Linux SDK Wizard installation procedure.

Users can check the status of utilities via running **./install.sh --selftest** command.

Utilities

Serial Port Utility

The Moxa serial port mode control utility **mx-uart-ctl** is for getting and setting the serial port's UART mode.

- Drivers dependency
  - ➢ moxa-it87-gpio-driver
  - ➢ moxa-it87-serial-driver
- Libraries dependency
  - ➢ None

Usage of UART mode control

```
Usage:
                mx-uart-ctl -p <port_number> [-m <uart_mode>]

OPTIONS:
                -p <port_number>
                                Set target port.
                -m <uart_mode>
                                Set target port to uart_mode
                                0 --> set to RS-232 mode
                                1 --> set to RS-485-2W mode
                                2 --> set to RS-422 mode
                                3 --> set to RS-485-4W mode

Example:
                Get mode from port 0
                # mx-uart-ctl -p 0

                Set port 1 to RS232 mode
                # mx-uart-ctl -p 1 -m 0
```

Digital IO (DIO) Port Utility

Moxa DIO port control tool **mx-dio-ctl** is for getting DI/DO and setting DO ports statuses (low/high).

- Drivers dependency
  - moxa-it87-gpio-driver
- Libraries dependency
  - None

Usage of DIO state control

```
Usage:
        mx-dio-ctl <-i|-o <#port number> [-s <#state>]>

OPTIONS:
        -i <#DIN port number>
        -o <#DOUT port number>
        -s <#state>
                Set state for target DOUT port
                0 --> LOW
                1 --> HIGH

Example:
        Get value from DIN port 0
        # mx-dio-ctl -i 0
        Get value from DOUT port 0
        # mx-dio-ctl -o 0

        Set DOUT port 0 value to LOW
        # mx-dio-ctl -o 0 -s 0
        Set DOUT port 0 value to HIGH
        # mx-dio-ctl -o 0 -s 1
```

Scaler Utility

Moxa scaler utility is designed to configure basic settings of display devices, such as brightness, touch panel status, and OSD settings

Usage

```
Usage:
        mx-scaler-util [Options]...
Options:
        -v, --version
                Get scaler firmware version
        -s, --status
                Get system status
        -m, --model
                Get model name
        -b, --brightness [0~10]
                Set brightness, query without arg
        -p, --touch-panel [0~1]
                Set touch panel on(1)/off(0), query without arg
        -o, --osd-config [0~1]
                Set OSD on(1)/off(0), query without arg
```

MCU Upgrade Utility

The mx-lpc-mcu-upgrade-tool is a command-line utility designed for upgrading the firmware on the MCU.

WARNING

Before using the MCU firmware upgrade tool, ensure that all Moxa MCU related services are stopped to avoid communication conflicts.

Usage

```
Usage:
        mx-lpc-mcu-upgrade-tool [Options]...
Options:
        -f, --file
                Start MCU upgrade from file
        -v, --version
                Get current MCU version
Example:
        mx-lpc-mcu-upgrade-tool -f FB_MCU_V3000_V1.00S03_22060219.bin
        mx-lpc-mcu-upgrade-tool -v
```

MCIM Wrapper Utility

Moxa Computer Interface Manager (MCIM) is a shell script based wrapper that provides commands similar to MCIM commands for peripherals.

Usage (general)

```
 The Moxa Computer Interface Manager (MCIM) is a tool designed to simplify
 user control of peripherals. The design of MCIM aims to enhance
 operational efficiency, enabling users to conveniently handle tasks
 related to peripheral devices.
Usage:
  mx-interface-mgmt [command]

Available Commands:
  cellular     Manages the cellular modem
  dio          Manages digital inputs and outputs for external devices
  led          Manages LED indicators
  relay        Manages the relay mode
  serialport   Manages the serial port
  input_power  Manages the power input state
  usb_power    Manages the usb power state

Flags:
  -h, --help      help for mx-interface-mgmt

Use "mx-interface-mgmt [command] --help" for more information about a command.
```

Usage (serialport wrapper)

```
Usage:
mx-interface-mgmt serialport <NAME> <COMMAND> [ARG]

Available Commands:
Get the interface of a serial port
$ mx-interface-mgmt serialport <serialport_name> get_interface
Set the interface of a serial port
$ mx-interface-mgmt serialport <serialport_name> set_interface
<serial_interface>

Arguments:
serialport_name: The number of serial port (e.g. 0, 1, 2, ......)
serial_interface:
0 --> set to RS-232 mode
1 --> set to RS-485-2W mode
2 --> set to RS-422 mode
3 --> set to RS-485-4W mode
```

Usage (cellular wrapper)

```
Usage:
  mx-interface-mgmt cellular <NAME> <COMMAND> [ARG]

Available Commands:
  Get the power state of a cellular
    $ mx-interface-mgmt cellular <cellular_name> get_power
  Set the power state of a cellular
    $ mx-interface-mgmt cellular <cellular_name> set_power <power_state>
  Get the SIM slot of a cellular
    $ mx-interface-mgmt cellular <cellular_name> get_sim_slot
  Set the SIM slot of a cellular
    $ mx-interface-mgmt cellular <cellular_name> set_sim_slot <sim_slot>

Arguments:
  cellular_name: The slot number of cellular (e.g. 1|2)
  power_state: on|off
  sim_slot: 1|2
```

Usage (dio wrapper)

```
Usage:
  mx-interface-mgmt dio <NAME> <COMMAND> [ARG]

Available Commands:
  Get the state of a dio
    $ mx-interface-mgmt dio <dio_name> get_state
  Set the state of a dio
    $ mx-interface-mgmt dio <dio_name> set_state <dio_state>

Arguments:
  dio_name: The name of dio (e.g. DI0DO0)
  dio_state: 0(low)|1(high)
```

Usage (led wrapper)

```
Usage:
  mx-interface-mgmt led <NAME> <COMMAND> [ARG]

Available Commands:
  Get the state of a LED
    $ mx-interface-mgmt led <led_name> get_state
  Set the state of a LED
    $ mx-interface-mgmt led <led_name> set_state <led_state>

Arguments:
  led_name: The number of LED (e.g. 0, 1, 2, ......)
  led_state: on|off
```

Usage (relay wrapper)

```
Usage:
  mx-interface-mgmt relay <NAME> <COMMAND> [ARG]

Available Commands:
  Get the mode of a relay
    $ mx-interface-mgmt relay <relay_name> get_mode
  Set the mode of a relay
    $ mx-interface-mgmt relay <relay_name> set_mode <relay_mode>

Arguments:
  relay_name: The number of relay (e.g. 0, 1, 2, ......)
  relay_mode: 0|1
              0 --> set to NC (Normal Closed) mode
              1 --> set to NO (Normal Open) mode
```

Usage (input_power wrapper)

```
Usage:
  mx-interface-mgmt input_power <NAME> <COMMAND> [ARG]

Available Commands:
  Get the state of a input_power
    $ mx-interface-mgmt input_power <input_power_name> get_state

Arguments:
  input_power_name: The number of input_power (e.g. 0, 1, 2, ......)
```

Usage (usb_power wrapper)

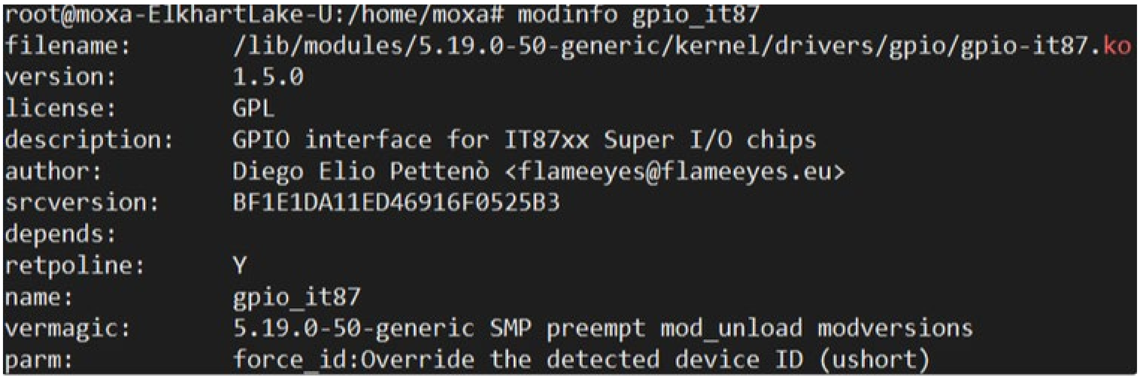
```
Usage:
  mx-interface-mgmt usb_power <NAME> <COMMAND> [ARG]

Available Commands:
  Get the usb power state of a port
    $ mx-interface-mgmt usb_power <usb_port> get_state
  Set the usb power state of a port
    $ mx-interface-mgmt usb_power <usb_port> set_state <state>

Arguments:
  usb_port: Get USB port power state
                      0: front
                      1: rear
                      2: internal
  state: Set USB port power state
                      0: off
                      1: on
```

Drivers

moxa-it87-gpio-driver

The purpose of **moxa-it87-gpio-driver** is to control the GPIO interface for **IT87xx Super I/O** chips, based on the Linux kernel, remove label for compatibility of Moxa utilities, and fix some issues. For details, see drivers/gpio/gpio-it87.c.

Kernel module information



```
root@moxa-ElkhartLake-U:/home/moxa# modinfo gpio_it87
filename:       /lib/modules/5.19.0-50-generic/kernel/drivers/gpio/gpio-it87.ko
version:        1.5.0
license:        GPL
description:    GPIO interface for IT87xx Super I/O chips
author:         Diego Elio Pettenò <flameeyes@flameeyes.eu>
srcversion:     BF1E1DA11ED46916F0525B3
depends:
retpoline:      Y
name:           gpio_it87
vermagic:       5.19.0-50-generic SMP preempt mod_unload modversions
parm:           force_id:Override the detected device ID (ushort)
```

Once the **gpio_it87** driver has been probed, the gpiochip interfaces **/sys/class/gpio/gpiochip*** and **/sys/class/gpio/gpio*** are created.

E.g.

```
# cat /sys/class/gpio/gpiochip698/label
gpio_it87
# cat /sys/class/gpio/gpio699/value
0
```

Thus, by read/write the gpio value, user can get/set the super IO gpio value.

NOTE

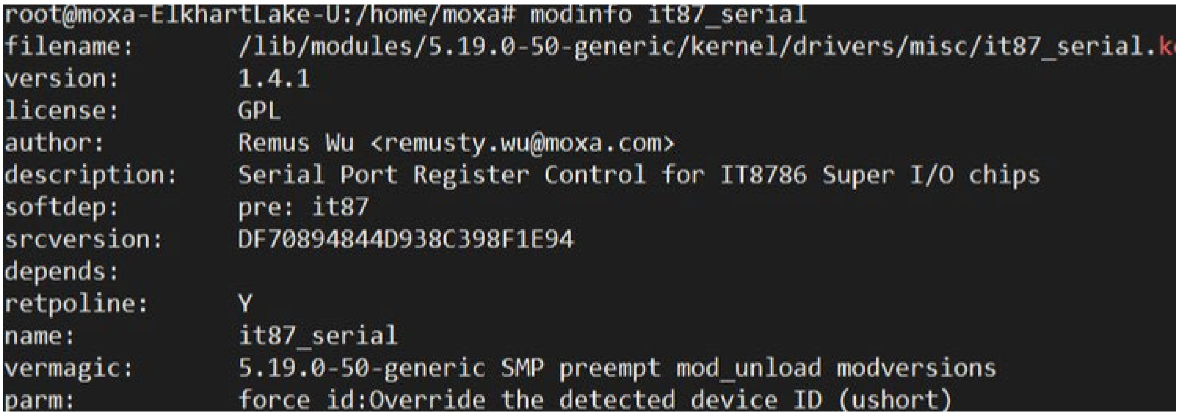
If the Linux kernel version ≥ 5.x, default uses the **libgpiod** to set/get set/get gpio value.

Alternatively, for Linux kernel version ≤ 3.x, default uses the **sys class gpio** to set/get gpio.

moxa-it87-serial-driver

IT87xx Super I/O chips support six standard serial ports and **RS485 automatic direction control (ADDC)**. This driver provides an interface under **misc** device for controlling the serial register.

Kernel module information



```
root@moxa-ElkhartLake-U:/home/moxa# modinfo it87_serial
filename:       /lib/modules/5.19.0-50-generic/kernel/drivers/misc/it87_serial.k
version:        1.4.1
license:        GPL
author:         Remus Wu <remusty.wu@moxa.com>
description:    Serial Port Register Control for IT8786 Super I/O chips
softdep:        pre: it87
srcversion:     DF70894844D938C398F1E94
depends:
retpoline:      Y
name:           it87_serial
vermagic:       5.19.0-50-generic SMP preempt mod_unload modversions
parm:           force_id:Override the detected device ID (ushort)
```

Once the **it87_serial** driver has been probed, the **/sys/class/misc./it87_serial/serial[p]** interface are created by driver.

E.g.

```
# cat /sys/class/misc/it87_serial/serial1/serial1_rs485
0
```

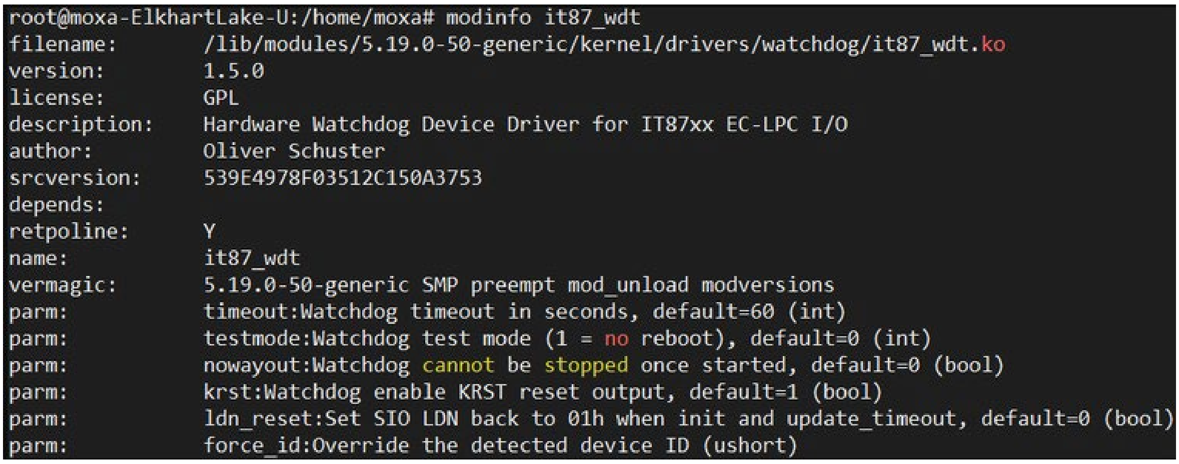
If returns 0, the RS485 automatic direction control (ADDC) is disabled. Alternatively, if returns 1, the ADDC is enabled.

The **UART RS485 ADDC state** selection has been imported into **mx-uart-ctl** utility.

moxa-it87-wdt-driver

Watchdog timer driver for ITE IT87xx environment control. The moxa-it87-wdt-driver is based on Linux kernel drivers/watchdog/it87_wdt.c driver, and add kernel parameters to support Moxa platform's hardware design.

Kernel module information



```
root@moxa-ElkhartLake-U:/home/moxa# modinfo it87_wdt
filename:       /lib/modules/5.19.0-50-generic/kernel/drivers/watchdog/it87_wdt.ko
version:        1.5.0
license:        GPL
description:    Hardware Watchdog Device Driver for IT87xx EC-LPC I/O
author:         Oliver Schuster
srcversion:     539E4978F03512C150A3753
depends:
retpoline:      Y
name:           it87_wdt
vermagic:       5.19.0-50-generic SMP preempt mod_unload modversions
parm:           timeout:Watchdog timeout in seconds, default=60 (int)
parm:           testmode:Watchdog test mode (1 = no reboot), default=0 (int)
parm:           nowayout:Watchdog cannot be stopped once started, default=0 (bool)
parm:           krst:Watchdog enable KRST reset output, default=1 (bool)
parm:           ldn_reset:Set SIO LDN back to 01h when init and update_timeout, default=0 (bool)
parm:           force_id:Override the detected device ID (ushort)
```

The watchdog device node **/dev/watchdog0** is created by the **it87_wdt** driver.

The x86 Linux SDK Wizard will by default set up the watchdog daemon configuration file **/etc/watchdog.conf** and enable services for specific Linux distributions.

The default timeout of the watchdog device is 60 seconds (maximum is 65535 seconds). If you want to change the timeout value, edit the watchdog daemon config file at **/etc/watchdog.conf**

E.g., to change the watchdog timeout to 300 seconds, run the command:

```
watchdog-timeout = 300
```

moxa-sdhci-pci-driver

NOTE

Available only on Debian 11 to resolve the SD card detection issue.

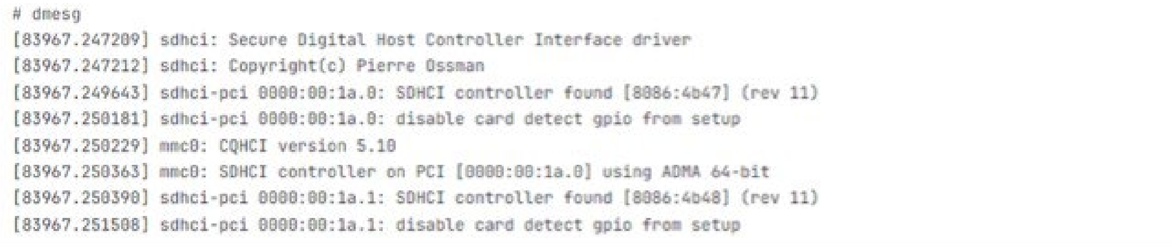
The purpose of **moxa-sdhci-pci-driver** is SDHCI on PCI bus interface driver.

Due to the SD host controller communicates with the CPU via SDIO, it would not initialize successfully on **Debian 11**. To resolve the issue, we need this driver to add module parameter (**enable_probe_cd_gpio**) to determine if the probe card can detect gpio or not.

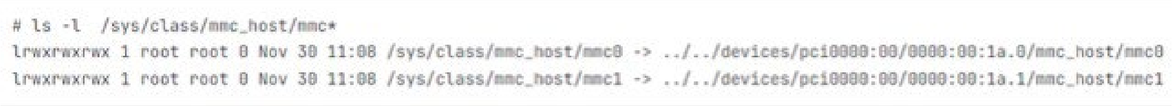
```
modprobe sdhci_pci enable_probe_cd_gpio=0
```

Or add modprobe configuration file: **`/lib/modprobe.d/sdhci-pci-option.conf`**

Kernel message and SD card interface:



```
# dmesg
[83967.247209] sdhci: Secure Digital Host Controller Interface driver
[83967.247212] sdhci: Copyright(c) Pierre Ossman
[83967.249643] sdhci-pci 0000:00:1a.0: SDHCI controller found [8086:4b47] (rev 11)
[83967.250181] sdhci-pci 0000:00:1a.0: disable card detect gpio from setup
[83967.250229] mmc0: CQHCI version 5.10
[83967.250363] mmc0: SDHCI controller on PCI [0000:00:1a.0] using ADMA 64-bit
[83967.250390] sdhci-pci 0000:00:1a.1: SDHCI controller found [8086:4b48] (rev 11)
[83967.251508] sdhci-pci 0000:00:1a.1: disable card detect gpio from setup
```



```
# ls -l  /sys/class/mmc_host/mmc*
lrwxrwxrwx 1 root root 0 Nov 30 11:08 /sys/class/mmc_host/mmc0 -> ../../devices/pci0000:00/0000:00:1a.0/mmc_host/mmc0
lrwxrwxrwx 1 root root 0 Nov 30 11:08 /sys/class/mmc_host/mmc1 -> ../../devices/pci0000:00/0000:00:1a.1/mmc_host/mmc1
```

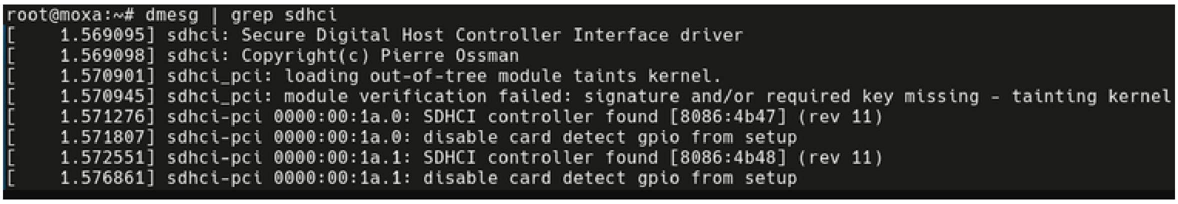
5. Linux Functions

The section introduces basic Linux concepts, like x86 secure boot, IO interfaces, TPM2 module, SD card slot mounting, and Linux PTP (IEEE 1588).

To provide skills and basic information for newcomers to learn more about Linux.

Mounting the SD Card Slot

The **MPC 3000** family supports one SD card slot (SD 3.0 interface (SDHC/SDXC)). Make sure your SD card is inserted into the SD card slot after which the following kernel message should be shown:

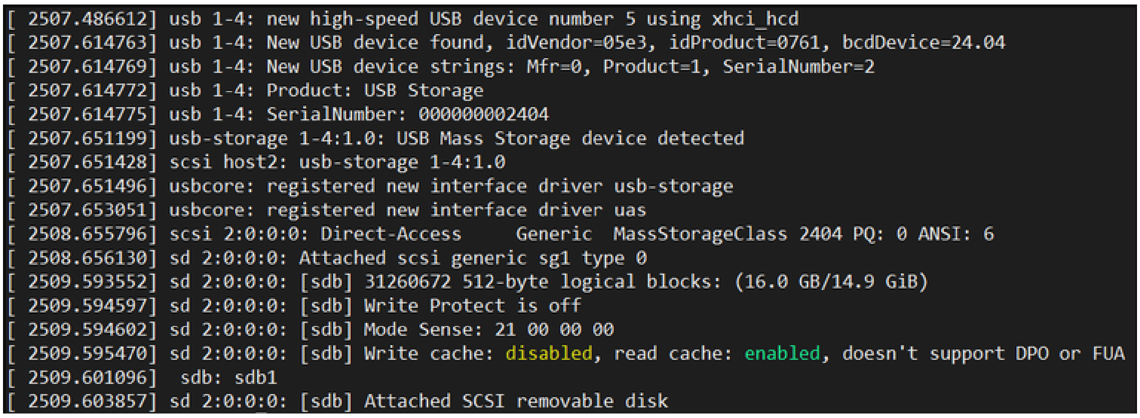


```
root@moxa:~# dmesg | grep sdhci
[    1.569095] sdhci: Secure Digital Host Controller Interface driver
[    1.569098] sdhci: Copyright(c) Pierre Ossman
[    1.570901] sdhci_pci: loading out-of-tree module taints kernel.
[    1.570945] sdhci_pci: module verification failed: signature and/or required key missing - tainting kernel
[    1.571276] sdhci-pci 0000:00:1a.0: SDHCI controller found [8086:4b47] (rev 11)
[    1.571807] sdhci-pci 0000:00:1a.0: disable card detect gpio from setup
[    1.572551] sdhci-pci 0000:00:1a.1: SDHCI controller found [8086:4b48] (rev 11)
[    1.576861] sdhci-pci 0000:00:1a.1: disable card detect gpio from setup
```

To **mount** the SD Card:

The block devices **/dev/mmcblk1**, the block device is created from sdhci driver.

Then, user can create a mount point on directory (e.g. **/mnt**): **sudo mount /dev/mmcblk1p1 /mnt**



```
[ 2507.486612] usb 1-4: new high-speed USB device number 5 using xhci_hcd
[ 2507.614763] usb 1-4: New USB device found, idVendor=05e3, idProduct=0761, bcdDevice=24.04
[ 2507.614769] usb 1-4: New USB device strings: Mfr=0, Product=1, SerialNumber=2
[ 2507.614772] usb 1-4: Product: USB Storage
[ 2507.614775] usb 1-4: SerialNumber: 000000002404
[ 2507.651199] usb-storage 1-4:1.0: USB Mass Storage device detected
[ 2507.651428] scsi host2: usb-storage 1-4:1.0
[ 2507.651496] usbcore: registered new interface driver usb-storage
[ 2507.653051] usbcore: registered new interface driver uas
[ 2508.655796] scsi 2:0:0:0: Direct-Access     Generic  MassStorageClass 2404 PQ: 0 ANSI: 6
[ 2508.656130] sd 2:0:0:0: Attached scsi generic sg1 type 0
[ 2509.593552] sd 2:0:0:0: [sdb] 31260672 512-byte logical blocks: (16.0 GB/14.9 GiB)
[ 2509.594597] sd 2:0:0:0: [sdb] Write Protect is off
[ 2509.594602] sd 2:0:0:0: [sdb] Mode Sense: 21 00 00 00
[ 2509.595470] sd 2:0:0:0: [sdb] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[ 2509.601096]  sdb: sdb1
[ 2509.603857] sd 2:0:0:0: [sdb] Attached SCSI removable disk
```

Secure Boot

The **UEFI Secure Boot** is a security feature that has been widely adopted in modern computer systems, especially those running Windows and some Linux distributions.

Its primary purpose is to ensure the integrity and authenticity of the operating system and bootloader during the system boot process, protecting the system against boot-time malware and other unauthorized software.

Secure Boot Purpose

Secure Boot is designed to prevent the loading of malicious software, such as rootkits and bootkits, during the boot process.

It does this by ensuring that only trusted and digitally **signed** bootloaders and OS kernels are executed.

Thus, if user loads **unsigned** bootloaders and OS kernels on target Linux distributions when UEFI secure boot has been enabled on BIOS menu, the boot process or kernel modules should be failed due to unauthorized policy.

Operating System Support

Check the following links for information on UEFI Secure Boot:

- Debian Secure Boot
- Ubuntu Secure Boot
- RedHat Secure Boot

Linux PTP (IEEE 1588)

The **Precision Time Protocol (PTP)** is a protocol used to synchronize clocks throughout a computer network. PTP provides higher precision and faster synchronization than NTP even without hardware support. With hardware support, sub-microsecond accuracy can be expected.

Whereas NTP is intended for WAN use, PTP is designed for LAN environments and makes use of UDP multicast.

Available LAN chip

- Intel I210 (driver: ibg)
- Intel I219 (driver: e1000e)

Debian Linuxptp package

Linuxptp package is an implementation of the Precision Time Protocol (PTP) according to IEEE standard 1588 for Debian Linux. Features include:

1. support for hardware and software time stamping via the Linux **SO_TIMESTAMPING** socket option.
2. support for the **Linux PTP Hardware Clock (PHC)** subsystem by using the **lock_gettime** family of calls, including the new clock_adjtimex system call
3. implementation of **Boundary Clock (BC)** and **Ordinary Clock (OC)**
4. transport over UDP/IPv4, UDP/IPv6, and raw Ethernet (Layer 2)
5. support for IEEE 802.1AS-2011 in the role of end station

Debian phc2sys program

phc2sys is a program which synchronizes two or more clocks in the system. Typically, it is used to synchronize the system clock to a PTP hardware clock (PHC), which itself is synchronized by the ptp4l(8) program. See the manpage for more information.

- Prerequisites
  - ➢ Install **Debian 11** or later version
  - ➢ Install **Linuxptp** package: **apt update && apt install linuxptp**
  - ➢ Stop and disable systemd time sync daemon service to avoid some unexpected operations: **systemctl stop systemdtimesyncd && systemctl disable systemd-timesyncd**

Example for Linux PTP setting up

[Ordinary Clock (OC) mode]

Set as **OC master** mode: Layer 2, P2P mode, peer delay mechanism

```
# Assume A side interface device is 'enp4s0'
ip link set dev enp4s0 up
ptp4l -m -2 -P -i enp4s0
```

Set as **OC slave** mode: Layer 2, P2P mode, peer delay mechanism

```
# Assume B side interface device is 'enp5s0'
ip link set dev enp5s0 up
ptp4l -m -2 -P -s -i enp5s0
# or with log: ptp4l -m -2 -s -P -i enp5s0 2>&1 | tee $(date +%Y%m%d%H%M%S.log)
# use phc2sys to sync sys clock for 10Hz
phc2sys -a -m -r -R 10
```

[Boundary Clock (BC) mode]

Set as **BC mode** host

- clock_type Specifies the kind of PTP clock. Valid values are "OC" for ordinary clock, "BC" for boundary clock, "P2P_TC" for peer to peer transparent clock, and "E2E_TC" for end to end transparent clock. A multi-port ordinary clock will automatically be configured as a boundary clock. The default is "OC".
- boundary_clock_jbod When running as a **boundary clock** (that is, when more than one network interface is configured), ptp4l performs a sanity check to make sure that all of the ports share the same hardware clock device. This option allows ptp4l to work as a boundary clock using "just a bunch of devices" that are not synchronized to each other. For this mode, the collection of clocks must be synchronized by an external program, for example phc2sys(8) in "automatic" mode. The default is 0 (disabled).

Example for BC mode

```
# For example, edit config file 'bc.cfg'
# and assume 'enp12s0' and 'enp4s0' are connected network interface
[global]
sanity_freq_limit 0
step_threshold 0.000002
tx_timestamp_timeout 10
logMinPdelayReqInterval 0
logSyncInterval 0
logAnnounceInterval 0
announceReceiptTimeout 3
syncReceiptTimeout 2
twoStepFlag 1
summary_interval 0
clock_type BC
priority1 128
priority2 127
delay_mechanism P2P
[enp12s0]
boundary_clock_jbod 1
network_transport L2
fault_reset_interval 0
[enp4s0]
boundary_clock_jbod 1
network_transport L2
fault_reset_interval 0
# run the ptp4l procedure
ip link set dev enp12s0 up
ip link set dev enp4s0 up
ptp4l -m -f bc.cfg
# use phc2sys to sync sys clock for 10Hz
```

```
phc2sys -a -m -r -R 10
```

On OC **Grandmaster**

```
# assume interface is enp5s0
ip link set dev enp5s0 up
ptp4l -2 -m -P -i enp5s0
```

On OC **Slave**

```
# assume interface is enp4s0
ip link set dev enp4s0 up
ptp4l -2 -m -s -P -i enp4s0
# with log: ptp4l -2 -m -s -P -i enp4s0 2>&1 | tee $(date +%Y%m%d%H%M%S.log)
```

[Transparent Clock (TC) mode]

Set **TC mode** host

```
# For example, edit config file 'tc.cfg'
# and assume 'enp12s0' and 'enp4s0' are connected network interface
[global]
priority1 254
priority2 253
free_running 1
freq_est_interval 3
tc_spanning_tree 1
clock_type P2P_TC
network_transport L2
delay_mechanism P2P
[enp12s0]
egressLatency 0
ingressLatency 0
delay_mechanism P2P
network_transport L2
[enp4s0]
egressLatency 0
ingressLatency 0
delay_mechanism P2P
network_transport L2
# run the ptp4l procedure
ip link set dev enp12s0 up
ip link set dev enp4s0 up
ptp4l -m -f tc.cfg
# use phc2sys to sync sys clock between master & slave for 10Hz
# -c Specify the slave clock by device (e.g. /dev/ptp1) or interface (e.g.
eth1)
# -s Specify the master clock by device (e.g. /dev/ptp0) or interface (e.g.
eth0)
phc2sys -s enp12s0 -c enp4s0 -O 0 -R 10 -m
```

As OC **Grandmaster**

```
# assume interface is enp5s0
ip link set dev enp5s0 up
ptp4l -2 -m -P -i enp5s0
```

As OC **Slave**

```
# assume interface is enp4s0
ip link set dev enp4s0 up
ptp4l -2 -m -s -P -i enp4s0
# use phc2sys to sync sys clock for 10Hz on slve
phc2sys -a -m -r -R 10
```

6. Troubleshooting

This section provides basic information on system logging, debugging, Moxa x86 SDK Wizard, and issue tracing.

Printing a Kernel Message

The **dmesg** command is used to display the kernel ring buffer, which contains messages related to the kernel and hardware events.

It's a useful tool for troubleshooting hardware-related issues, monitoring system-level events and diagnosing hardware issues.

To simply view the kernel ring buffer, run the following command: **dmesg**

You can save the output of **dmesg** to a file for further analysis. For instance, to save the log to a file named **kernel.log**, use the following command:

```
# save kernel message to log
dmesg >kernel.log

# or simply to save the error and warninglevel log:
dmesg --level=err,warn > kernel_err_warn.log
```

Collecting System Logs

The following procedure describes the collecting of log files. Log files in the **/var/log** directory.

Archive and compress all log files and put them in **/tmp**

```
tar czvf /tmp/varlog.tar.gz /var/log/*.log.*
```

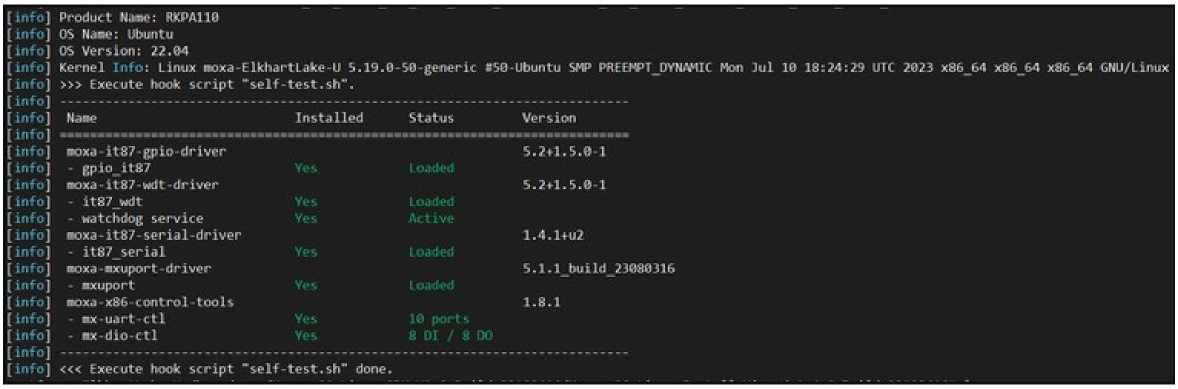
The output file **/tmp/varlog.tar.gz** can be transferred for debugging usage.

Getting Installation Logs From Moxa x86 Linux SDK Installation Wizard

Moxa x86 Linux SDK provides the **self-test** command for diagnosing the status of drivers and tools after installation. To see the log, run the command as follows:

```
./install.sh --selftest
```

The status of the drivers and tools are shown on a terminal. See the following example:



```
[info] Product Name: RKPA110
[info] OS Name: Ubuntu
[info] OS Version: 22.04
[info] Kernel Info: Linux moxa-ElkhartLake-U 5.19.0-50-generic #50-Ubuntu SMP PREEMPT_DYNAMIC Mon Jul 10 18:24:29 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
[info] >>> Execute hook script "self-test.sh".
[info] ---------------------------------------------------------------------------
[info]  Name                          Installed      Status         Version
[info] ===========================================================================
[info]  moxa-it87-gpio-driver                                       5.2+1.5.0-1
[info]  - gpio_it87                   Yes            Loaded
[info]  moxa-it87-wdt-driver                                        5.2+1.5.0-1
[info]  - it87_wdt                    Yes            Loaded
[info]  - watchdog service            Yes            Active
[info]  moxa-it87-serial-driver                                     1.4.1+u2
[info]  - it87_serial                 Yes            Loaded
[info]  moxa-mxuport-driver                                         5.1.1_build_23080316
[info]  - mxuport                     Yes            Loaded
[info]  moxa-x86-control-tools                                      1.8.1
[info]  - mx-uart-ctl                 Yes            10 ports
[info]  - mx-dio-ctl                  Yes            8 DI / 8 DO
[info] ---------------------------------------------------------------------------
[info] <<< Execute hook script "self-test.sh" done.
```

An installation log is also created under
Moxa_x86_Linux_Install_Wizard_<version>_Build_<build_date>/install.log

View the **install.log** file to further check for issues.

Getting the Hardware Information of the Host

IOS exports the hardware information on **DMI** (Desktop Management Interface) table.

Linux **dmidecode** is a tool for dumping a computer DMI (some say **SMBIOS**) table contents in a human-readable format. This table contains a description of the system's hardware components, as well as other useful pieces of information such as serial numbers and BIOS revision.

Install dmidecode Package

- Ubuntu/Debian: **sudo apt-get install dmidecode**
- RHEL: **sudo yum install dmidecode**

Example

[Get model name and hardware version]

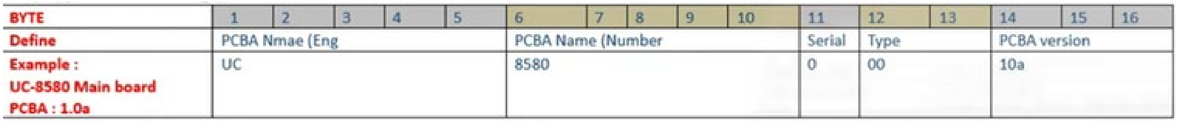
The Option 1 (or Option 2) displays the 16 bytes information, for example: RKP A110000091

RKP A110000091 means

- PCBA name = RKP
- PCBA number = A110
- PCBA serial = 0
- PCBA type = 00
- PCBA hw version = 091 (v0.91)

How to get information from dmitable

```
# dmidecode -t 12
Handle 0x0021, DMI type 12, 5 bytes
System Configuration Options
        Option 1:   RKP A110000091
        Option 2:
        Option 3:
...
```



BYTE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Define	PCBA Nmae (Eng					PCBA Name (Number					Serial	Type		PCBA version		
Example : UC-8580 Main board PCBA : 1.0a	UC					8580					0	00		10a		

[Get current BIOS version]

```
# dmidecode -t bios
BIOS Information
        Vendor: INSYDE Corp.
        Version: V1.0.0S04
        Release Date: 05/15/2023
        Address: 0xE0000
        Runtime Size: 128 kB
        ROM Size: 10 MB
...
```

[Get memory and processor hardware information]

```
# dmidecode -t memory
Physical Memory Array
        Location: System Board Or Motherboard
        Use: System Memory
        Error Correction Type: None
        Maximum Capacity: 16 GB
        Error Information Handle: Not Provided
        Number Of Devices: 2
...
# sudo dmidecode -t processor
Processor Information
        Socket Designation: U3E1
        Type: Central Processor
        Family: Other
        Manufacturer: Intel(R) Corporation
        ID: 61 06 09 00 FF FB EB BF
        Version: Intel Atom(R) x6425E Processor @ 2.00GHz
        Voltage: 1.1 V
        External Clock: 100 MHz
...
```

7. Appendix

The Licensing /Commercial Use of Linux Distributions

A Linux distribution is a version of the Linux operating system that includes the Linux kernel, system utilities, libraries, and additional software and applications. Linux distributions are created by various organizations, communities, and individuals, each tailoring the operating system to meet specific needs and preferences.

Linux distribution include:

Debian

Debian is a free and open-source operating system, and its intellectual property rights policy is based on a commitment to free software principles. Debian adheres to a set of guidelines and policies outlined in the Debian Free Software Guidelines (DFSG). The DFSG defines the criteria that software must meet to be considered "free" in the context of Debian.

Commercial use:

Free redistribution.

The license of a Debian component **may not restrict** any party from selling or giving away the software as a component of an aggregate software distribution containing programs from many diverse sources. The license may not require a royalty or other fee for such sale.

https://wiki.debian.org/DebianFreeSoftwareGuidelines

https://www.debian.org/trademark

https://wiki.debian.org/ProposedTrademarkPolicy

Ubuntu

Ubuntu is built by Canonical and the Ubuntu community. We share access rights owned by Canonical with the Ubuntu community for the purposes of discussion, development and advocacy. We recognise that most of the open source discussion and development areas are for non-commercial purposes and we therefore allow the use of Canonical IP in this context, as long as there is no commercial use and that the Canonical IP is used in accordance with this IPRights Policy.

You can modify Ubuntu for **personal** or **internal commercial** use.

You can **redistribute** Ubuntu, but only where there has been **no modification** to it.

For more Canonical’s intellectual property rights policy:

https://ubuntu.com/legal/intellectual-property-policy

Red Hat Enterprise Linux (RHEL)

Red Hat Enterprise Linux (RHEL) is a **commercial** Linux distribution provided by Red Hat, Inc. It is designed for enterprise environments and comes with a subscription-based pricing model.

https://www.redhat.com/en/store/linux-platforms

https://www.redhat.com/en/about/trademark-guidelines-and-policies

https://www.redhat.com/en/about/terms-use