# phyFLEX-i.MX 8M Plus FPSC BSP Manual

PHYTEC Messtechnik GmbH

2026 年 07 月 22 日

# Contents

下表显示了与本手册兼容的 BSP：

| 适用 BSP | BSP 发布类型 | Yocto Version | BSP 发布日期 | BSP 状态 |
|---|---|---|---|---|
| BSP-Yocto-NXP-i.MX8MP-PD26.1.0 | 大版本 | Walnascar | 2026/06/08 | 已发布 |

本手册指导您完成 BSP 包的安装、编译和烧写，并描述如何使用 **phyFLEX-i.MX 8M Plus Libra Rapid Development Kit** 的硬件接口。本手册还包括如何从源码编译内核、u-boot 镜像。本手册包含需要在 PC(Linux 操作系统) 上执行的指令。

**备注**

This document contains code examples that describe the communication with the board over the serial shell. The code examples lines begin with `host:~$`, `target:~$` or `u-boot=>`. This describes where the commands are to be executed. Only after these keywords must the actual command be copied.

PHYTEC provides a variety of hardware and software documentation for all of its products. This includes any or all of the following:

**QS Guide**
A short guide on how to set up and boot a phyCORE based board.

**Hardware Manual**
A detailed description of the System-on-Module and accompanying carrierboard.

**Yocto Guide**
A comprehensive guide for the Yocto version the phyCORE uses. This guide contains an overview of Yocto; introducing, installing, and customizing the PHYTEC BSP; how to work with programs like Poky and Bitbake; and much more.

**BSP 手册**
A manual specific to the BSP version of the phyCORE. Information such as how to build the BSP, booting, updating software, device tree, and accessing peripherals can be found here.

**Development Environment Guide**
This guide shows how to work with the Virtual Machine (VM) Host PHYTEC has developed and prepared to run various Development Environments. There are detailed step-by-step instructions for Eclipse and Qt Creator, which are included in the VM. There are instructions for running demo projects for these programs on a phyCORE product as well. Information on how to build a Linux host PC yourself is also a part of this guide.

**Pin Muxing Table**
phyCORE SOMs have an accompanying pin table (in Excel format). This table will show the complete default signal path, from the processor to the carrier board. The default device tree muxing option will also be included. This gives a developer all the information needed in one location to make muxing changes and design options when developing a specialized carrier board or adapting a PHYTEC phyCORE SOM to an application.

除了这些标准手册和指南之外，PHYTEC 还将提供产品变更通知、应用说明和技术说明。这些文档将根据具体案例进行针对性提供。大部分文档都可以在我们产品的 https://www.phytec.de/produkte/system-on-modules/phyflex-imx-8m-plus-fpsc/#downloads 中找到。

CHAPTER 1

# 支持的硬件

在我们的网页上，您可以查看适用于 BSP 版本 BSP-Yocto-NXP-i.MX8MP-PD26.1.0 的所有 Machine 及其对应的 Article Numbers(产品型号)：网页.

如果您在“Supported Machines”一栏选择了特定的 **Machine Name** ，您可以查看该 machine 下可用的 **Article Numbers** 以及硬件信息的简短描述。如果您只有硬件的 **Article Numbers** ，您可以将 **Machine Name** 下拉菜单留空，仅选择您的 **Article Numbers** 。现在，它应该会显示您特定硬件所需的 **Machine Name**

## 1.1 phyFLEX Libra 器件

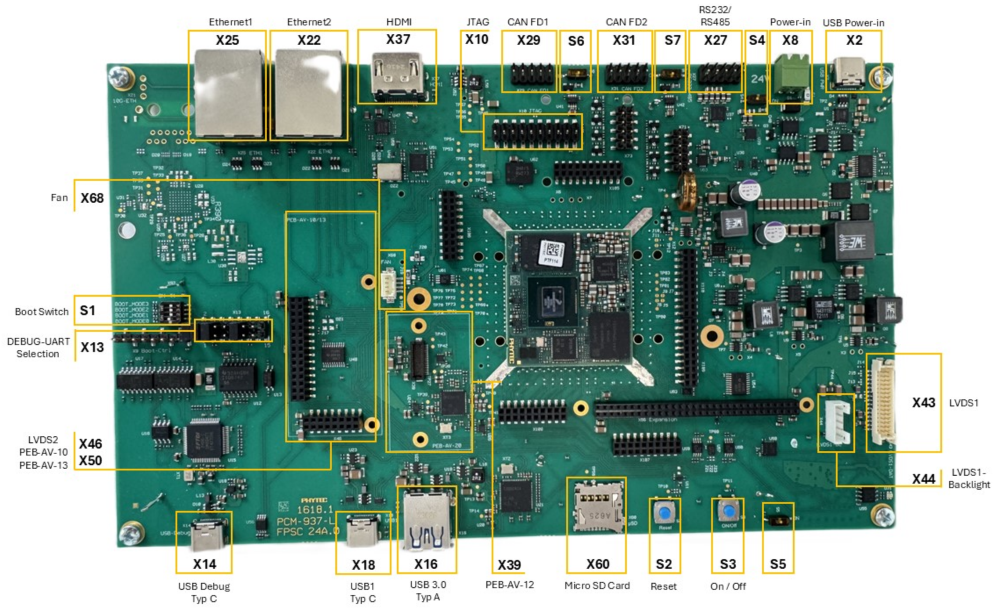



图 1: Libra 器件（顶部）

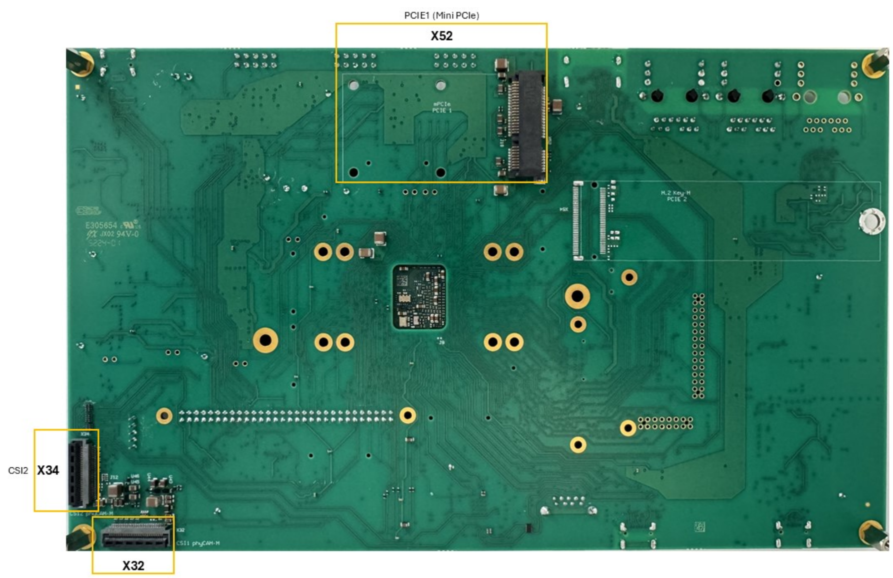



图 2: Libra 器件（底部）

CHAPTER 2

# 开始使用

The **phyFLEX-i.MX 8M Plus Libra Rapid Development Kit** is shipped with a pre-flashed SD card. It contains the phytec-qt6demo-image and can be used directly as a boot source. The e.MMC is programmed with only a U-Boot by default. You can get all sources from the BSP downloads page. This chapter explains how to flash a BSP image to SD card and how to start the board.

There are several ways to flash an image to SD card or even e.MMC. Most notably using simple, sequential writing with the Linux command line tool `dd`. An alternative way is to use PHYTEC's system initialization program called partup, which makes it especially easy to format more complex systems. You can get prebuilt Linux binaries of partup from its release page. Also read partup's README for installation instructions.

## 2.1 下载镜像

The image contains all necessary files and makes sure partitions and any raw data are correctly written. Both the partup package and the WIC image, which can be flashed using `dd`, can be downloaded from our BSP downloads page.

Note that you can find different image versions and variants on our download server. The images are located on the server by folders per "BSP-Version", "Distro-Name" and "Machine-Name".

Example to download a partup package and a WIC image from the download server:

```
host:~$ wget https://download.phytec.de/Software/Linux/BSP-Yocto-i.MX8MP/BSP-Yocto-NXP-i.MX8MP-PD26.1.0/
↪images/ampliphy-vendor/imx8mp-phyflex-libra-rdk-2/phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.
↪partup
host:~$ wget https://download.phytec.de/Software/Linux/BSP-Yocto-i.MX8MP/BSP-Yocto-NXP-i.MX8MP-PD26.1.0/
↪images/ampliphy-vendor/imx8mp-phyflex-libra-rdk-2/phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.
↪wic.xz
```

**备注**

For e.MMC, more complex partitioning schemes or even just large images, we recommend using the partup package, as it is faster in writing than `dd` and allows for a more flexible configuration of the target flash device.

## 2.2 将镜像写入 SD 卡

**警告**

要创建 SD 卡启动盘，必须要拥有 Linux PC 上的 root 权限。在选择烧写设备时请务必小心！所选设备上的所有文件将在命令执行后立即被擦除，而且擦除前不会有任何进一步的确认！

选择错误的设备可能会导致 **数据丢失**，例如，可能会擦除您当前所在 PC 上的系统！

### 2.2.1 寻找正确的设备

要创建 SD 卡启动盘，首先要找到 PC 上您 SD 卡对应的正确设备名称。在开始将镜像复制到 SD 卡之前，请卸载任何已挂载的分区。

1. 为了获取正确的设备名称，请移除您的 SD 卡并执行：

```
host:~$ lsblk
```

2. 现在插入你的 SD 卡，然后再次执行命令：

```
host:~$ lsblk
```

3. 比较两个输出，以获取第二个输出中的新设备名称。这些是 SD 卡的设备名称（如果 SD 卡已格式化，则包括设备名称和对应的分区）。
4. 为了验证找到的设备名称的最终正确性，请执行命令 `sudo dmesg`。在其输出的最后几行中，您应该也能找到设备名称，例如 `/dev/sde` 或 `/dev/mmcblk0` （具体取决于您的系统）。

或者，您可以使用图形化的程序，例如 GNOME Disks 或 KDE Partition Manager 来找到正确的设备。

现在您已经得到了正确的设备名称，例如 `/dev/sde`，如果 SD 卡曾格式化过，需要确认已取消其分区的挂载，您可以在输出中看到带有附加了数字的设备名称（例如 `/dev/sde1`），它们是 SD 卡的分区。一些 Linux 发行版系统在设备插入时会自动挂载分区。在写入之前，必须卸载这些分区，以避免数据损坏。

卸载所有这些分区，例如：

```
host:~$ sudo umount /dev/sde1
host:~$ sudo umount /dev/sde2
```

Now, the SD card is ready to be flashed with an image, using either `partup`, `dd` or `bmaptool`.

### 2.2.2 Using bmaptool

One way to prepare an SD card is using bmaptool. Yocto automatically creates a block map file (`<IMAGENAME>-<MACHINE>.wic.bmap`) for the WIC image that describes the image content and includes checksums for data integrity. *bmaptool* is packaged by various Linux distributions. For Debian-based systems install it by issuing:

```
host:~$ sudo apt install bmap-tools
```

通过以下命令将 WIC 镜像烧写到 SD 卡：

```
host:~$ bmaptool copy phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2?(.rootfs).wic?(.xz) /dev/<your_device>
```

将 <your_device> 替换为您之前找到的 SD 卡设备名称，并确保将文件 `<IMAGENAME>-<MACHINE>.wic.bmap` 与 WIC 镜像文件放在一起，以便 bmaptool 知道哪些块需要写入，哪些块需要跳过。

**警告**

*bmaptool* 仅擦写 SD 卡上镜像数据所在的区域。这意味着在写入新的镜像后，之前写入的旧 U-Boot 环境变量可能仍然可用。

### 2.2.3 使用 partup

使用 partup 烧写 SD 卡只需一个命令：

```
host:~$ sudo partup install phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2?(.rootfs).partup /dev/<your_
↪device>
```

确保将 <your_device> 替换为您之前找到的设备名称。

关于 partup 的进一步使用说明，请参阅其 官方文档 。

**警告**

使用 resize2fs 版本 1.46.6 及更早版本的 PC 系统（例如 Ubuntu 22.04）无法烧写在 Mickledore 以及更新的 yocto 版本上创建的 partup 软件包。这个是因为 resize2fs 新增了默认选项而导致的兼容性问题。有关详细信息，请参阅 release notes 。

**备注**

*partup* 具有清除 eMMC user 区域中特定区域的功能，我们提供的 partup 程序中用该功能擦除 U-Boot 环境变量。这是 *bmaptool* 工具所无法完成的一点，如前一部分所提到的。

partup 相较于其他烧写工具的一个主要优势是，它可以配置 MMC 的特定部分，比如他可以直接写入 eMMCboot 分区，无需调用其他命令。

### 2.2.4 使用 dd

在卸载所有 SD 卡的挂载分区后，您可以烧写 SD 卡。

一些 PHYTEC BSP 会生成未压缩的镜像（文件名扩展名为 *.wic），而另一些则生成压缩的镜像（文件名扩展名为 *.wic.xz）。

要写入未压缩的镜像（*.wic），请使用以下命令：

```
host:~$ sudo dd if=phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2?(.rootfs).wic of=/dev/<your_device>␣
↪bs=1M conv=fsync status=progress
```

或者要写入压缩后的镜像（*.wic.xz），请使用以下命令：

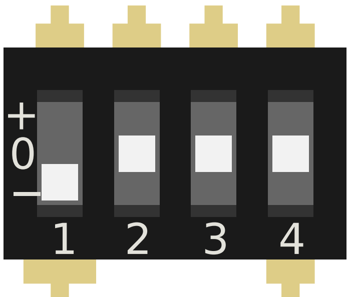
```
host:~$ xzcat phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2?(.rootfs).wic.xz | sudo dd of=/dev/<your_
↪device> bs=1M conv=fsync status=progress
```

再次确保将 <your_device> 替换为之前找到的设备名称。

参数 `conv=fsync` 强制在 `dd` 返回之前对设备进行 sync 操作。这确保所有数据块都已写入 SD 卡，而没有任何数据缓存在内存中。参数 `status=progress` 将打印出进度信息。

## 2.3 首次启动

- 要从 SD 卡启动，*bootmode switch (S1)* 需要设置为以下位置：




- 插入 SD 卡
- Connect the target and the host with **USB-C** on *(X14)* debug USB
- 给开发板通电

CHAPTER 3

# 编译 BSP

本节将指导您使用 Yocto 和 phyLinux 脚本进行 i.MX 8M Plus BSP 的编译。更多有关 phytec meta-layer 和 Yocto 的信息，请访问：Yocto Reference Manual (walnascar) 。

## 3.1 基本设置

如果您从未在您的主机上使用 Yocto 编译过 Phytec BSP，您应查看 Yocto Reference Manual (walnascar) 中的 BSP Workspace 安装一节。

## 3.2 下载 BSP

There are two ways to get the BSP sources. You can download the complete BSP sources from our BSP downloads page; or you can fetch and build it yourself with Yocto. This is particularly useful if you want to make customizations.

The phyLinux script is a basic management tool for PHYTEC Yocto BSP releases written in Python. It is mainly a helper to get started with the BSP sources structure.

- 创建一个新的项目文件夹，获取 phyLinux 脚本，并赋予脚本具备可执行权限：

```
host:~$ mkdir ~/yocto
host:~$ cd yocto/
host:~/yocto$ wget https://download.phytec.de/Software/Linux/Yocto/Tools/phyLinux
host:~/yocto$ chmod +x phyLinux
```

> **警告**
>
> 我们需要一个空的项目文件夹，phyLinux 首先会清理当前所在的工作目录。从一个不为空的目录下调用 phyLinux 将会产生告警。

- 运行 phyLinux：

```
host:~/yocto$ ./phyLinux init
```

**备注**

在首次初始化时，phyLinux 脚本会要求您在 `/usr/local/bin` 目录中安装 Repo 工具。

- During the execution of the init command, you need to choose your processor platform (SoC), PHYTEC's BSP release number, and the hardware (MACHINE) you are working on.

**备注**

If you cannot identify your board with the information given in the selector, have a look at the invoice for the product. And have a look at the webpage of our BSP.

- 也可以通过命令行参数直接传递这些信息：

```
host:~/yocto$ DISTRO=ampliphy-vendor MACHINE=imx8mp-phyflex-libra-rdk-2 ./phyLinux init -p imx8mp -r BSP-Yocto-NXP-i.MX8MP-PD26.1.0
```

After the execution of the init command, phyLinux will print a few important notes. For example, it will print your git identity, SOC and BSP release which was selected as well as information for the next steps in the build process.

### 3.2.1 开始构建

- 设置 Shell 环境变量：

```
host:~/yocto$ source sources/poky/oe-init-build-env
```

**备注**

在每次打开新的用于编译的 shell 时，都需要先执行这一步骤。

- 当前的工作目录会变更为 build/。
- 打开主配置文件，同意并接受 GPU 和 VPU 二进制文件的许可证协议。通过取消注释相应的行来完成此操作，如下所示。

```
host:~/yocto/build$ vim conf/local.conf
# Uncomment to accept NXP EULA
# EULA can be found under ../sources/meta-freescale/EULA
ACCEPT_FSL_EULA = "1"
```

- 编译您的镜像：

```
host:~/yocto/build$ bitbake phytec-qt6demo-image
```

**备注**

对于第一次编译，我们建议从我们的较小的非图形化镜像 phytec-headless-image 开始，以查看一切是否正常工作。

```
host:~/yocto/build$ bitbake phytec-headless-image
```

第一次构建过程在现代的 Intel Core i7 处理器上大约需要 40 分钟。后续的构建将使用本次编译产生的缓存，大约需要 3 分钟。

### 3.2.2 BSP 镜像

所有由 Bitbake 生成的镜像都放在 `~/yocto/build/deploy*/images/<machine>` 。例如以下列表是 imx8mp-phyflex-libra-rdk-2 machine 生成的所有文件：

- **u-boot.bin**: 编译后的 U-boot bootloader 二进制文件。不是最终镜像中的 bootloader！
- **oftree**: 默认内核设备树
- **u-boot-spl.bin**: 二级程序加载器 (SPL)
- **bl31-imx8mp.bin**: ARM 可信固件二进制文件
- **lpddr4_pmu_train_2d_dmem_202006.bin, lpddr4_pmu_train_2d_imem_202006.bin**: DDR PHY 固件镜像
- **imx-boot**：由 imx-mkimage 编译的 bootloader 镜像，包括 SPL、U-Boot、ARM 可信固件和 DDR 固件。这是最终的可引导 bootloader 镜像。
- **fitImage**: Linux 内核 FIT 镜像
- **fitImage-its*.its**
- **Image**: Linux 内核镜像
- **Image.config**: 内核 config 文件
- **imx8mp-phyflex-libra-rdk*.dtb**: Kernel device tree file
- **imx8mp-phyflex-libra*.dtbo**: Kernel device tree overlay files
- **phytec-qt6demo-image*.tar.gz**: 根文件系统
- **phytec-qt6demo-image*.rootfs.wic.xz**: 压缩的 SD 卡镜像
- ***.scr.uimg**: compiled bootscripts

CHAPTER 4

# 安装操作系统

## 4.1 启动模式开关 (S1)

**小技巧**

Hardware revision baseboard: 1618.2

The phyFLEX Libra features a boot switch with four individually switchable ports to select the phyFLEX-i.MX 8M Plus FPSC Gamma default bootsource.

**Internal fuses**

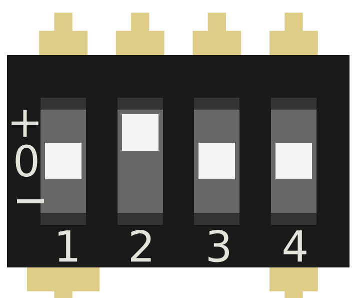



图 1: Boot from internal fuses

**USB**

**eMMC**

**SD card**

**SPI NOR flash**

**JTAG**

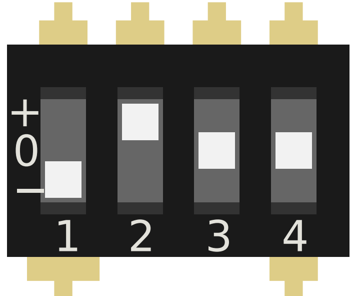



图 2: USB serial download

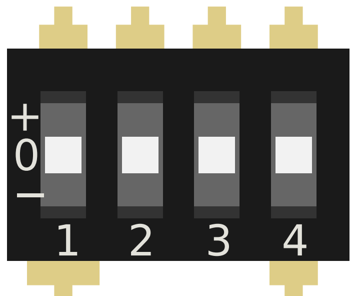



图 3: Boot from onboard e.MMC

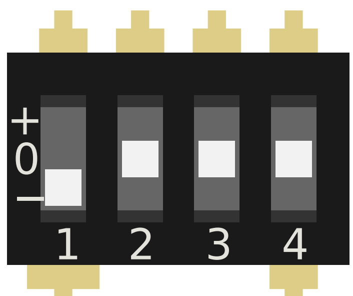



图 4: Boot from external SD card

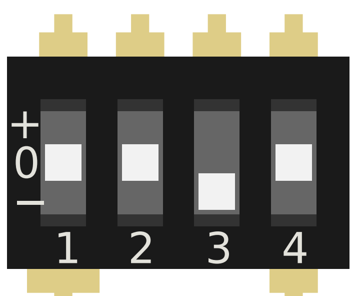



图 5: Boot from onboard SPI NOR flash

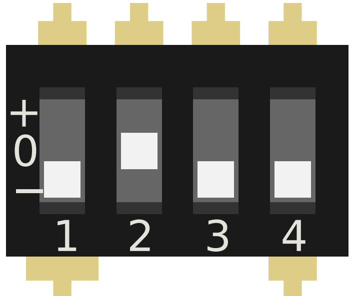



图 6: JTAG mode

To boot from e.MMC, make sure that the BSP image is flashed correctly to the e.MMC and the *bootmode switch (S1)* is set to **e.MMC**.

## 4.2 Flash e.MMC

为了保持文档的一致性和简洁性，假设已经配置好了 TFTP 服务器；所有生成的镜像（如上所列）都被复制到默认的/srv/tftp 目录。如果您没有进行设置，您需要修改路径到包含镜像的目录。有关如何设置 TFTP 服务器和目录的说明，请参见*Setup Network Host* 。

### 4.2.1 Flash e.MMC from Network

i.MX 8M Plus boards have an Ethernet connector and can be updated over a network. Be sure to set up the development host correctly. The IP needs to be set to 192.168.3.10, the netmask to 255.255.255.0, and a TFTP server needs to be available. From a high-level point of view, an e.MMC device is like an SD card. Therefore, it is possible to flash the **WIC image** (`<name>.wic`) from the Yocto build system directly to the e.MMC. The image contains the bootloader, kernel, device tree, device tree overlays, and root file system.

#### Flash e.MMC via Network in Linux on Host

It is also possible to install the OS at e.MMC from your Linux host. As before, you need a complete image on your host.

> **小技巧**
>
> 需要保证设备和存储镜像的主机之间的网络正常！ *Setup Network Host*

查看主机上可用的镜像文件：

```
host:~$ ls /srv/tftp
phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic.xz
phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic.bmap
```

Send the image with the `bmaptool` command combined with ssh through the network to the e.MMC of your device:

```
host:~$ scp /srv/tftp/phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic.* root@192.168.3.11:/tmp &&
↪ ssh root@192.168.3.11 "bmaptool copy /tmp/phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic.xz /
↪dev/mmcblk2"
```

### Flash e.MMC via Network in Linux on Target

You can update the e.MMC from your target.

**小技巧**

需要保证设备和存储镜像的主机之间的网络正常！ *Setup Network Host*

Take a compressed or decompressed image with the accompanying block map file **.bmap* on the host and send it with *ssh* through the network to the e.MMC of the target with a one-line command:

```
target:~$ scp <USER>@192.168.3.10:/srv/tftp/phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic.* /
↪tmp && bmaptool copy /tmp/phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic.xz /dev/mmcblk2
```

### Flash e.MMC from Network in U-Boot on Target

These steps will show how to update the e.MMC via a network.

**小技巧**

This step only works if the size of the image file is less than 1GB due to limited usage of RAM size in the Bootloader after enabling OP-TEE.

**小技巧**

需要保证设备和存储镜像的主机之间的网络正常！ *Setup Network Host*

解压缩您的镜像

```
host:~$ unxz /srv/tftp/phytec-headless-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic.xz
```

通过网络将您的镜像加载到内存中：

- 使用 DHCP

```
u-boot=> dhcp phytec-headless-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic
BOOTP broadcast 1
DHCP client bound to address 192.168.3.1 (1 ms)
Using ethernet@30be0000 device
TFTP from server 192.168.3.10; our IP address is 192.168.3.1
Filename 'phytec-headless-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic'.
Load address: 0x40480000
Loading: ######################################
         ######################################
         ######################################
         ...
         ...
         ...
         ######################################
         #############
         11.2 MiB/s
done
Bytes transferred = 911842304 (36599c00 hex)
```

- 使用静态 IP 地址（必须先设置 serverip 和 ipaddr）。

```
u-boot=> tftp ${loadaddr} phytec-headless-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic
Using ethernet@30be0000 device
TFTP from server 192.168.3.10; our IP address is 192.168.3.11
Filename 'phytec-headless-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic'.
Load address: 0x40480000
Loading: ######################################
         ######################################
         ######################################
         ...
         ...
         ...
         ######################################
         #############
         11.2 MiB/s
done
Bytes transferred = 911842304 (36599c00 hex)
```

Write the image to the e.MMC:

```
u-boot=> mmc dev 2
switch to partitions #0, OK
mmc2(part 0) is current device
u-boot=> setexpr nblk ${filesize} / 0x200
u-boot=> mmc write ${loadaddr} 0x0 ${nblk}

MMC write: dev # 2, block # 0, count 1780942 ... 1780942 blocks written: OK
```

### 4.2.2 Flash e.MMC U-Boot image via Network from running U-Boot

Update the standalone U-Boot image imx-boot is also possible from U-Boot. This can be used if the bootloader on e.MMC is located in the e.MMC user area.

**小技巧**

需要保证设备和存储镜像的主机之间的网络正常！ *Setup Network Host*

Load image over tftp into RAM and then write it to e.MMC:

```
u-boot=> tftp ${loadaddr} imx-boot
u-boot=> setexpr nblk ${filesize} / 0x200
u-boot=> mmc dev 2
u-boot=> mmc write ${loadaddr} 0x40 ${nblk}
```

**提示**

十六进制值表示偏移量，单位为 512 字节块的倍数。请参阅偏移表 以获取相应 SoC 的正确值。

### 4.2.3 Flash e.MMC from USB stick

#### Flash e.MMC from USB in Linux

These steps will show how to flash the e.MMC on Linux with a USB stick. You only need a complete image saved on the USB stick and a bootable WIC image. (e.g. phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.|yocto-imageext|). Set the *bootmode switch (S1)* to SD card.

- 插入并挂载 U 盘：

```
[   60.458908] usb-storage 1-1.1:1.0: USB Mass Storage device detected
[   60.467286] scsi host0: usb-storage 1-1.1:1.0
[   61.504607] scsi 0:0:0:0: Direct-Access                              8.07 PQ: 0 ANSI: 2
[   61.515283] sd 0:0:0:0: [sda] 3782656 512-byte logical blocks: (1.94 GB/1.80 GiB)
[   61.523285] sd 0:0:0:0: [sda] Write Protect is off
[   61.528509] sd 0:0:0:0: [sda] No Caching mode page found
[   61.533889] sd 0:0:0:0: [sda] Assuming drive cache: write through
[   61.665969]  sda: sda1
[   61.672284] sd 0:0:0:0: [sda] Attached SCSI removable disk
target:~$ mount /dev/sda1 /mnt
```

- 现在查看您在 USB 优盘上保存的镜像文件：

```
target:~$ ls /mnt
phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic.xz
phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic.bmap
```

- Write the image to the phyFLEX-i.MX 8M Plus FPSC Gamma e.MMC (MMC device 2 without partition):

```
target:~$ bmaptool copy /mnt/phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic.xz /dev/
↪mmcblk2
```

- After a complete write, your board can boot from e.MMC.

> **小技巧**
>
> 在此之前，您需要将*bootmode switch (S1)* 配置为 **eMMC**。

#### Flash e.MMC from USB stick in U-Boot on Target

> **小技巧**
>
> 此步骤仅在镜像文件小于 1GB 的情况下会被执行成功，因为在启用 OPTEE 后，Bootloader 中可用的 RAM 大小有限，不足以加载超过 1GB 的镜像

These steps will show how to update the e.MMC via a USB device. Configure the *bootmode switch (S1)* to SD card and insert an SD card. Power on the board and stop in U-Boot prompt. Insert a USB device with the copied uncompressed WIC image to the USB slot.

将镜像从 USB 设备加载到 RAM 中：

```
u-boot=> usb start
starting USB...
USB0:   USB EHCI 1.00
```

(续下页)

(接上页)

```
scanning bus 0 for devices... 2 USB Device(s) found
       scanning usb for storage devices... 1 Storage Device(s) found
u-boot=> fatload usb 0:1 0x58000000 phytec-headless-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic
497444864 bytes read in 31577 ms (15 MiB/s)
```

Write the image to the e.MMC:

```
u-boot=> mmc dev 2
switch to partitions #0, OK
mmc2(part 0) is current device
u-boot=> setexpr nblk ${filesize} / 0x200
u-boot=> mmc write 0x58000000 0x0 ${nblk}

MMC write: dev # 2, block # 0, count 1024000 ... 1024000 blocks written: OK
u-boot=> boot
```

### 4.2.4 Flash e.MMC from SD card

Even if there is no network available, you can update the e.MMC. For that, you only need a ready-to-use image file (`*.wic`) located on the SD card. Because the image file is quite large, you need to allocate more SD card space. To create a new partition or enlarge your SD card, see *Resizing ext4 Root Filesystem*.

或者，使用 partup 包烧写 SD 卡，如*Getting Started* 中所述。这样就可使用 SD 卡的全部容量。

#### Flash e.MMC from SD card in Linux on Target

You can also flash the e.MMC on Linux. You only need a partup package or WIC image saved on the SD card.

- 检查在 SD 卡上保存的 partup 包或 WIC 镜像文件：

  ```
  target:~$ ls
  phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.partup
  phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic.xz
  phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic.bmap
  ```

- Write the image to the phyFLEX-i.MX 8M Plus FPSC Gamma e.MMC (MMC device 2 **without** partition) using partup:

  ```
  target:~$ partup install phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.partup /dev/mmcblk2
  ```

  Flashing the partup package has the advantage of using the full capacity of the e.MMC device, adjusting partitions accordingly.

  > **备注**
  >
  > 另外，也可以使用 `bmaptool` 工具：
  >
  > ```
  > target:~$ bmaptool copy phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic.xz /dev/mmcblk2
  > ```
  >
  > 请注意，在使用 `bmaptool` 烧写时，根文件系统分区并不会使用 eMMC 的最大容量。

- After a complete write, your board can boot from e.MMC.

> **警告**
>
> Before this will work, you need to configure the *bootmode switch (S1)* to e.MMC.

### Flash e.MMC from SD card in U-Boot on Target

> **小技巧**
>
> This step only works if the size of the image file is less than 1GB due to limited usage of RAM size in the Bootloader after enabling OPTEE. If the image file is too large use the *Updating e.MMC from SD card in Linux on Target* subsection.

- 将一个可用的镜像烧写到 SD 卡，并创建一个 EXT4 格式的第三分区。将 WIC 镜像（例如 phytec-qt6demo-image.rootfs.wic）复制到该分区。
- Configure the *bootmode switch (S1)* to SD card and insert the SD card.
- 打开电源并进入 U-Boot。
- 加载镜像：

```
u-boot=> ext4load mmc 1:3 ${loadaddr} phytec-headless-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic
reading
911842304 bytes read in 39253 ms (22.2 MiB/s)
```

- Switch the mmc dev to e.MMC:

```
u-boot=> mmc list
FSL_SDHC: 1 (SD)
FSL_SDHC: 2 (eMMC)
u-boot=> mmc dev 2
switch to partitions #0, OK
mmc2(part 0) is current device
```

- Flash your WIC image (for example phytec-qt6demo-image.rootfs.wic) from the SD card to e.MMC. This will partition the card and copy imx-boot, Image, dtb, dtbo, and root file system to e.MMC.

```
u-boot=> setexpr nblk ${filesize} / 0x200
u-boot=> mmc write ${loadaddr} 0x0 ${nblk}

MMC write: dev # 2, block # 0, count 1780942 ... 1780942 blocks written: OK
```

- Power off the board and change the *bootmode switch (S1)* to e.MMC.

CHAPTER 5

# 开发

## 5.1 独立编译准备

In this section, we describe how to build the U-Boot and the Linux kernel without using the Yocto Project. This procedure makes the most sense for development. The U-Boot source code, the Linux kernel, and all other git repositories are available on GitHub.

### 5.1.1 Git 仓库

- 使用的 U-Boot 仓库：

```
https://github.com/phytec/u-boot-phytec-imx
```

- 我们的 U-Boot 基于 u-boot-phytec-imx 并添加了一些硬件相关的补丁。
- 使用的 Linux 内核仓库：

```
https://github.com/phytec/linux-phytec-imx
```

- 我们的 i.MX 8M Plus 内核是基于 linux-phytec-imx 内核。

要找出核心板应使用的 u-boot 和 kernel 版本对应的 git 仓库 tag 标签，请查看您的 BSP 源文件夹：

```
meta-phytec/recipes-kernel/linux/linux-phytec-imx_*.bb
meta-phytec/recipes-bsp/u-boot/u-boot-phytec-imx_*.bb
```

### 5.1.2 获取 SDK

You can download the SDK from the SDK downloads page, or build it yourself with Yocto:

- 移动到 Yocto 的 build 目录：

```
host:~$ source sources/poky/oe-init-build-env
host:~$ bitbake -c populate_sdk phytec-qt6demo-image # or another image
```

在成功编译后，SDK 安装包保存在 `build/deploy*/sdk`。

### 5.1.3 安装 SDK

- 设置正确的权限并安装 SDK：

```
host:~$ chmod +x phytec-ampliphy-vendor-glibc-x86_64-phytec-qt6demo-image-cortexa53-crypto-toolchain-
↪5.2.4.sh
host:~$ ./phytec-ampliphy-vendor-glibc-x86_64-phytec-qt6demo-image-cortexa53-crypto-toolchain-5.2.4.sh
===============================================================================================
Enter target directory for SDK (default: /opt/ampliphy-vendor/5.2.4):
You are about to install the SDK to "/opt/ampliphy-vendor/5.2.4". Proceed [Y/n]? Y
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
```

### 5.1.4 使用 SDK

通过在工具链目录中 source *environment-setup* 文件来初始化您的 shell 交叉编译环境：

```
host:~$ source /opt/ampliphy-vendor/5.2.4/environment-setup-cortexa53-crypto-phytec-linux
```

### 5.1.5 安装所需工具

独立编译 Linux kernel 和 U-Boot 需要主机安装一些额外的工具。对于 Ubuntu，您可以使用以下命令安装它们：

```
host:~$ sudo apt install bison flex libssl-dev
```

**警告**

在较旧的主机发行版（例如，Ubuntu 20.04 LTS）上使用 Scarthgap NXP 基础的 BSP 时，构建用于主机的 U-Boot 或 Linux 内核工具可能会出现问题。如果遇到"undefined reference" 错误，一种解决方法是在 PATH 环境变量中添加主机的 binutils。

```
host$ export PATH=/usr/bin:$PATH
```

在加载 SDK *environment-setup* 后，运行此命令。

请注意，在较新版本的发行版（如 Ubuntu 22.04）中未观察到 SDK 相关问题，这些发行版似乎无需任何修改即可正常工作。

## 5.2 单独编译 U-Boot

### 5.2.1 获取源代码

- 获取 U-Boot 源代码：

```
host:~$ git clone https://github.com/phytec/u-boot-phytec-imx
```

- 要获取正确的 *U-Boot* **tag**，您需要查看我们的 release notes，可以在这里找到：release notes
- 此版本中使用的 **tag** 称为 v2025.04-2.2.0-phy9
- 查看所需的 *U-Boot* **tag**：

```
host:~$ cd ~/u-boot-phytec-imx/
host:~/u-boot-phytec-imx$ git fetch --all --tags
host:~/u-boot-phytec-imx$ git checkout tags/v2025.04-2.2.0-phy9
```

- 设置编译环境：

```
host:~/u-boot-phytec-imx$ source /opt/ampliphy-vendor/5.2.4/environment-setup-cortexa53-crypto-phytec-
↪linux
```

### 5.2.2 获取所需的二进制文件

要编译 bootloader，您需要将这些文件复制到您的 u-boot-phytec-imx 编译目录，并将其重命名以适应 *mkimage* 脚本：

- **ARM Trusted firmware 二进制文件**（*mkimage* 工具兼容格式 **bl31.bin** ）：bl31-imx8mp.bin
- **OPTEE 镜像** (可选的)：tee.bin
- **DDR firmware files** ( *mkimage* 工具兼容格式 **lpddr4_[i,d]mem_*d_*.bin** )：lpddr4_dmem_1d_*.bin, lpddr4_dmem_2d_*.bin, lpddr4_imem_1d_*.bin, lpddr4_imem_2d_*.bin

如果您已经使用 Yocto 编译了我们的 BSP，您可以在 yocto 工程目录中获取 bl31-imx8mp.bin、tee.bin 和 lpddr4_*.bin： *BSP Images*

或者你可以在这里下载文件：https://download.phytec.de/Software/Linux/BSP-Yocto-i.MX8MP/BSP-Yocto-NXP-i.MX8MP-PD26.1.0/images/ampliphy-vendor/imx8mp-phyflex-libra-rdk-2/imx-boot-tools/

> **警告**
>
> 确保您重命名所需的文件，以和 *mkimage tool* 兼容。

### 5.2.3 编译 bootloader

- 编译 flash.bin (imx-boot):

```
host:~/u-boot-phytec-imx$ make imx8mp-phyflex-libra-rdk_defconfig
host:~/u-boot-phytec-imx$ make flash.bin
```

### 5.2.4 将 bootloader 烧写到块设备上

flash.bin 文件可以在 u-boot-phytec-imx/ 目录下找到，现在可以进行烧写。需要指定芯片特定的偏移量：

| SoC | User 分区偏移量 | Boot 分区偏移量 | e.MMC Device |
|---|---|---|---|
| i.MX 8M Plus | 32 kiB | 0 kiB | /dev/mmcblk2 |

例如，烧写 SD 卡：

```
host:~/u-boot-phytec-imx$ sudo dd if=flash.bin of=/dev/sd[x] bs=1024 seek=32 conv=fsync
```

**提示**

如果您有我们的 BSP Yocto 工程代码，具体的偏移值也会在 Yocto 变量”BOOTLOADER_SEEK”和”BOOTLOADER_SEEK_EMMC” 中声明。

## 5.3 单独编译内核

内核与设备树一起打包在 FIT 镜像中。U-Boot 已被配置为能够加载 FIT 镜像并引导其中包含的内核。因此，内核镜像必须打包在 FIT 镜像中。

### 5.3.1 配置源代码

- 使用的 linux-phytec-imx 分支可以在 release notes 中找到
- 此版本本所需的标签称为 v6.12.49-2.2.0-phy11
- Check out 所需的 linux-phytec-imx 标签：

```
host:~$ git clone https://github.com/phytec/linux-phytec-imx
host:~$ cd ~/linux-phytec-imx/
host:~/linux-phytec-imx$ git fetch --all --tags
host:~/linux-phytec-imx$ git checkout tags/v6.12.49-2.2.0-phy11
```

- 为了提交更改，强烈建议切换到一个新分支：

```
host:~/linux-phytec-imx$ git switch --create <new-branch>
```

- 设置编译环境：

```
host:~/linux-phytec-imx$ source /opt/ampliphy-vendor/5.2.4/environment-setup-cortexa53-crypto-phytec-
↪linux
```

### 5.3.2 编译内核

- 编译 Linux 内核：

```
host:~/linux-phytec-imx$ make imx8_phytec_defconfig
host:~/linux-phytec-imx$ make -j$(nproc)
```

- 安装内核模块，比如安装到 NFS 目录：

```
host:~/linux-phytec-imx$ make INSTALL_MOD_PATH=/home/<user>/<rootfspath> modules_install
```

- 镜像可以在 ~/linux-phytec-imx/arch/arm64/boot/Image.gz 找到
- dtb 文件可以在 ~/linux-phytec-imx/arch/arm64/boot/dts/freescale/imx8mp-phyflex-libra-rdk.dtb 找到
- 要（重新）编译设备树和 -overlay 文件，只需运行

```
host:~/linux-phytec-imx$ make dtbs
```

or, to build a specific dtb (e.g. imx8mp-phyflex-libra-rdk.dtb):

```
host:~/linux-phytec-imx$ make freescale/imx8mp-phyflex-libra-rdk.dtb
```

**备注**

如果您遇到以下编译问题:

```
scripts/dtc/yamltree.c:9:10: fatal error: yaml.h: No such file or directory
```

确保您在主机系统上安装了 *"libyaml-dev"* 包:

```
host:~$ sudo apt install libyaml-dev
```

### 5.3.3 将内核打包成 FIT 镜像

要简单地替换内核,您需要一个 `image tree source` (.its) 文件。如果您已经使用 Yocto 编译了我们的 BSP,可以从此处提到的目录获取 its 文件: *BSP Images* 或者您可以在这里下载该文件:https://download.phytec.de/Software/Linux/BSP-Yocto-i.MX8MP/BSP-Yocto-NXP-i.MX8MP-PD26.1.0/images/ampliphy-vendor/imx8mp-phyflex-libra-rdk-2/

将.its 文件复制到当前工作目录,创建一个指向内核镜像的链接,并使用 mkimage 创建最终的 fitImage。

```
host:~/linux-phytec-imx$ cp /path/to/yocto/deploydir/fitimage-its*.its .
                && ln -s arch/arm64/boot/Image.gz linux.bin
                && mkimage -f fitImage-its*.its fitImage
```

### 5.3.4 Copy FIT image and kernel modules to SD card

FIT 镜像以及内核 module 可以用以下方式复制到已挂载的 SD 卡上。

```
host:~/linux-phytec-imx$ cp fitImage /path/to/sdcard/boot/
host:~/linux-phytec-imx$ make INSTALL_MOD_PATH=/path/to/sdcard/root/ modules_install
```

## 5.4 Working with UUU

The Universal Update Utility (UUU) by NXP is software to execute on the host for loading and running the bootloader on the board through SDP (Serial Download Protocol). For detailed information visit https://github.com/nxp-imx/mfgtools or download the Official UUU-tool documentation.

### 5.4.1 Host preparations for UUU Usage

- 请按照 https://github.com/nxp-imx/mfgtools#linux 上的说明进行操作。
- 如果您要从源代码编译 UUU,请将其添加到 `PATH` 中:

  This BASH command adds uuu only temporarily to `PATH`. To add it permanently, add this line to `~/.bashrc`.

  ```
  export PATH=~/mfgtools/uuu/:"$PATH"
  ```

- 设置 udev 规则(在 `uuu -udev` 中有详细说明):

  ```
  host:~$ sudo sh -c "uuu -udev >> /etc/udev/rules.d/70-uuu.rules"
  host:~$ sudo udevadm control --reload
  ```

### 5.4.2 获取镜像

Download imx-boot from our server or get it from your Yocto build directory at build/deploy-ampliphy-vendor/images/imx8mp-phyflex-libra-rdk-2/. For flashing a wic image to e.MMC, you will also need phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic.

### 5.4.3 开发板准备

将*bootmode switch (S1)* 设置为 **USB 串行下载**。同时，将 USB 端口*X18* 连接到主机。

### 5.4.4 通过 UUU 工具启动 bootloader

执行并给开发板上电：

```
host:~$ sudo uuu -b spl imx-boot
```

您可以像往常一样通过*(X14)* 在终端上查看启动日志。

> **备注**
>
> The default boot command when booting with UUU is set to fastboot. If you want to change this, please adjust the environment variable bootcmd_mfg in U-Boot prompt with setenv bootcmd_mfg. Please note, when booting with UUU the default environment is loaded. `saveenv` has no effect. If you want to change the boot command permanently for uuu-boot, you need to change this in U-Boot code.

### 5.4.5 Flashing U-boot Image to e.MMC via UUU

> **警告**
>
> UUU flashes U-boot into e.MMC BOOT (hardware) boot partitions, and it sets the BOOT_PARTITION_ENABLE in the e.MMC! This is a problem since we want the bootloader to reside in the e.MMC USER partition. Flashing next U-Boot version .wic image and not disabling BOOT_PARTITION_ENABLE bit will result in device always using U-boot saved in BOOT partitions. To fix this in U-Boot:
>
> ```
> u-boot=> mmc partconf 2 0 0 0
> u-boot=> mmc partconf 2
> EXT_CSD[179], PARTITION_CONFIG:
> BOOT_ACK: 0x0
> BOOT_PARTITION_ENABLE: 0x0
> PARTITION_ACCESS: 0x0
> ```
>
> or check Disable booting from e.MMC boot partitions from Linux.
>
> This way the bootloader is still flashed to e.MMC BOOT partitions but it is not used!
>
> When using **partup** tool and `.partup` package for e.MMC flashing this is done by default, which makes partup again superior flash option.

执行并给开发板上电：

```
host:~$ sudo uuu -b emmc imx-boot
```

### 5.4.6 Flashing wic Image to e.MMC via UUU

执行并给开发板上电：

```
host:~$ sudo uuu -b emmc_all imx-boot phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic
```

### 5.4.7 Flashing U-Boot to SD Card via UUU

执行并给开发板上电：

```
host:~$ sudo uuu -b sd imx-boot
```

### 5.4.8 Flashing wic Image to SD Card via UUU

执行并给开发板上电：

```
host:~$ sudo uuu -b sd_all imx-boot phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.wic
```

## 5.5 主机网络准备

为了在 bootloader 中执行涉及网络的各种任务，需要配置一些主机服务。在开发主机上，必须安装和配置 TFTP、NFS 和 DHCP 服务。启动以太网所需的工具如下：

```
host:~$ sudo apt install tftpd-hpa nfs-kernel-server kea
```

### 5.5.1 TFTP 服务设置

- 首先，创建一个目录来存储 TFTP 文件：

  ```
  host:~$ sudo mkdir /srv/tftp
  ```

- 然后将您的 BSP 镜像文件复制到此目录，并确保 other 用户也对 tftp 目录中的所有文件具有读取权限，否则将无法从开发板访问这些文件。

  ```
  host:~$ sudo chmod -R o+r /srv/tftp
  ```

- 您还需要为相应的接口配置一个静态 IP 地址。PHYTEC 开发板的默认 IP 地址是 192.168.3.11。可以将主机地址设置为 192.168.3.10，子网掩码为 255.255.255.0

  ```
  host:~$ ip addr show <network-interface>
  ```

  将 <network-interface> 替换为连接到开发板的网络接口。您可以通过不指定网络接口来显示所有可选网络接口。

- 返回的结果应包含以下内容：

  ```
  inet 192.168.3.10/24 brd 192.168.3.255
  ```

- 创建或编辑 `/etc/default/tftpd-hpa` 文件：

  ```
  # /etc/default/tftpd-hpa

  TFTP_USERNAME="tftp"
  TFTP_DIRECTORY="/srv/tftp"
  TFTP_ADDRESS=":69"
  TFTP_OPTIONS="-s -c"
  ```

- 将 TFTP_DIRECTORY 设置为您的 TFTP 服务器根目录
- 将 TFTP_ADDRESS 设置为 TFTP 服务监听的主机地址（设置为 0.0.0.0:69 以监听 69 端口上所有 IP）。
- 设置 TFTP_OPTIONS，以下命令显示可配置的选项：

```
host:~$ man tftpd
```

- 重新启动服务以应用配置更改：

```
host:~$ sudo service tftpd-hpa restart
```

现在将开发板的以太网端口连接到您的主机。我们还需要在开发板和运行 TFTP 服务的主机之间建立网络连接。TFTP 服务器的 IP 地址应设置为 192.168.3.10，子网掩码为 255.255.255.0。

### NFS 服务器设置

- 创建一个 NFS 目录：

```
host:~$ sudo mkdir /srv/nfs
```

- Temporarily export the nfs directory: The NFS server is not restricted to a certain file system location, so all we have to do is to export our root file system to the embedded network. In this example, the whole directory is exported and the "lab network" address of the development host is 192.168.3.10. The IP address has to be adapted to the local needs:

  ```
  host:~$ sudo exportfs -i -o rw,no_root_squash,sync,no_subtree_check 192.168.3.0/255.255.255.
  ↪0:/srv/nfs
  ```

- unexport the rootfs when finished:

  ```
  host:~$ sudo exportfs -u 192.168.3.0/255.255.255.0:/srv/nfs
  ```

### Permanent export

- To make the export persistent across reboots on most distributions, modify the `/etc/exports` file and export it:

```
/srv/nfs 192.168.3.0/255.255.255.0(rw,no_root_squash,sync,no_subtree_check)
```

- 现在 NFS 服务器需要再次读取 `/etc/exportfs` 文件：

```
host:~$ sudo exportfs -ra
```

### DHCP 服务器设置

- 创建或编辑 `/etc/kea/kea-dhcp4.conf` 文件；以内部子网为例，将 <network-interface> 替换为物理网络接口的名称：

```
{
  "Dhcp4": {
    "interfaces-config": {
      "interfaces": [ "<network-interface>/192.168.3.10" ]
    },
    "lease-database": {
```

（续下页）

(接上页)

```
      "type": "memfile",
      "persist": true,
      "name": "/tmp/dhcp4.leases"
    },
    "valid-lifetime": 28800,
    "subnet4": [{
        "id": 1,
        "next-server": "192.168.3.10",
        "subnet": "192.168.3.0/24",
        "pools": [
          { "pool": "192.168.3.1 - 192.168.3.255" }
        ]
    }]
  }
}
```

**警告**

在创建子网时请小心，因为这可能会扰乱公司网络政策。为了安全起见，请使用不同的子网，并通过 `interfaces` 配置选项指定该网络。

- 现在 DHCP 服务需要重新读取 `/etc/kea/kea-dhcp4.conf` 文件：

```
host:~$ sudo systemctl restart kea-dhcp4-server
```

当您启动/重启主机时，如果 kea-dhcp4 配置中指定的网络接口未处于活动状态，kea-dhcp4-server 将无法启动。因此请确保在连接接口后启动或者重启该 systemd 服务。

**备注**

DHCP server setup is only needed when using dynamic IP addresses. For our vendor BSPs, static IP addresses are used by default.

```
u-boot=> env print ip_dyn
ip_dyn=no
```

To use dynamic IP addresses for netboot, `ip_dyn` needs to be set to `yes`.

## 5.6 从网络启动内核

从网络启动意味着通过 TFTP 加载内核和设备树，并通过 NFS 加载根文件系统。但 bootloader 需要从另外的的启动设备加载。

### 5.6.1 在主机上放置网络启动的镜像

- 将内核 fitimage 复制到您的 tftp 目录中：

```
host:~$ cp fitImage /srv/tftp
```

- 将启动脚本复制到您的 tftp 目录中：

```
host:~$ cp net_boot_fit.scr.uimg /srv/tftp/
```

- 确保 other 用户对 tftp 目录中的所有文件具有读取权限，否则将无法从开发板访问它们：

```
host:~$ sudo chmod -R o+r /srv/tftp
```

- 将根文件系统解压到您的 NFS 目录中：

```
host:~$ sudo tar -xvzf phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.rootfs.tar.gz -C /srv/nfs
```

**备注**

请确保使用 sudo 执行命令，以保留根文件系统中文件的所属权限。

### 5.6.2 开发板上的网络设置

如果要自定义开发板上的以太网配置，请按照此处的说明进行操作：*Network Environment Customization*

### 5.6.3 从开发板启动

将开发板启动到 U-boot，按任意键暂停。

- 要从网络启动，请运行：

```
u-boot=> setenv boot_targets ethernet
u-boot=> bootflow scan -lb
```

To persistently boot from network, save the environment after setting boot_targets to ethernet.

```
u-boot=> env set boot_targets "ethernet"
u-boot=> env save
```

To use DHCP for booting from a network:

```
u-boot=> env set ip_dyn true
u-boot=> env save
```

When not using DHCP (ip_dyn set to false), U-Boot needs ipaddr and serverip to be set. If addresses different from the PHYTEC provided defaults are desired, they can be set in the U-Boot environment:

```
u-boot=> env set ipaddr <xxx.xxx.xxx.xxx>
u-boot=> env set serverip <xxx.xxx.xxx.xxx>
u-boot=> env save
```

## 5.7 获取 BSP 开发中版本

### 5.7.1 当前 release 的开发中版本

这些 release manifest 文件是为了让您访问 *Yocto* BSP 的开发版本。它们不会在 phyLinux 选择菜单中显示，需要手动选择。可以使用以下命令行来完成此操作：

```
host:~$ ./phyLinux init -p imx8mp -r BSP-Yocto-NXP-i.MX8MP-PD26.1.y
```

这将初始化一个 BSP，用于跟踪当前版本（BSP-Yocto-NXP-i.MX8MP-PD26.1.0 ）的最新开发版本。从现在开始，在此文件夹中执行 *repo sync* 将从我们的 Git 仓库中拉取所有最新的更改：

```
host:~$ repo sync
```

### 5.7.2 即将发布版本的开发中版本

即将发布版本的开发中版本可以通过这种方式访问。请执行以下命令，并查找一个比最新版本（BSP-Yocto-NXP-i.MX8MP-PD26.1.0 ）的 PDXX.Y 数字更高的版本，并且以 `.y` 结尾：

```
host:~$ ./phyLinux init -p imx8mp
```

## 5.8 获取最新的 Upstream 支持

我们有一个使用 Yocto 主分支（不是 NXP 发布的）的 manifest，他使用 upstream 的 Linux 和 U-Boot。这可以用来测试最新的 upstream kernel/U-Boot。

**备注**

master 分支的 manifest 反映了最新的开发状态。有时会出现一些 bug。我们会定期修复 master 分支。

```
host:~$ ./phyLinux init -p imx8mp -r BSP-Yocto-Ampliphy-i.MX8MP-master
```

## 5.9 Format SD card

使用单一的 SD 卡启动盘对存储介质进行烧写是开发过程中的常见任务。本章节针对此场景提供基础说明。大多数镜像的大小超过了默认的 root 分区剩余容量。要使用 SD 卡进行烧写，根文件系统需要扩展或创建一个单独的分区。有几种不同的方法可以格式化 SD 卡。最简单的方法是使用 Gparted。

### 5.9.1 Gparted

- 获取 GParted：

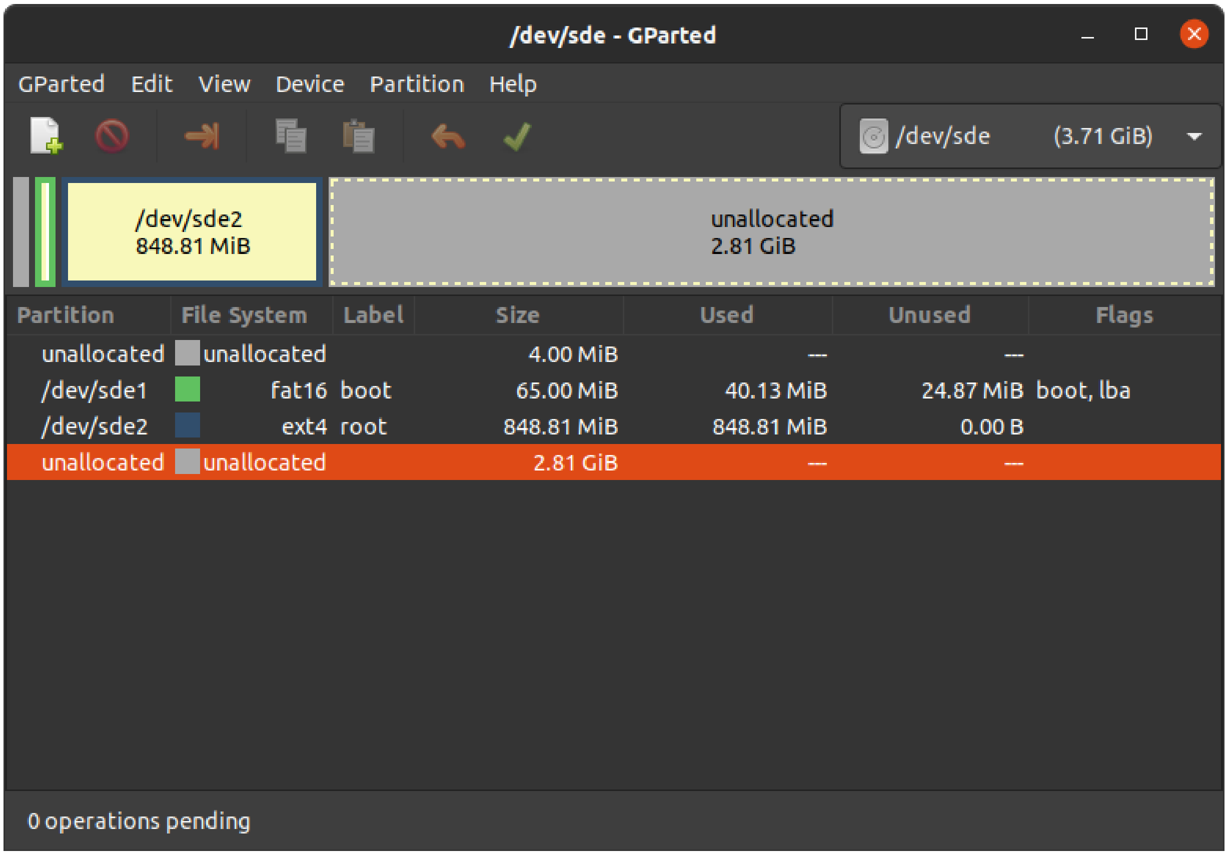
  ```
  host:~$ sudo apt install gparted
  ```

- Insert the SD card into your host and get the device name:

  ```
  host:~$ dmesg | tail
  ...
  [30436.175412] sd 4:0:0:0: [sdb] 62453760 512-byte logical blocks: (32.0 GB/29.8 GiB)
  [30436.179846]  sdb: sdb1 sdb2
  ...
  ```

- Unmount all SD card partitions.
- 启动 GParted：

  ```
  host:~$ sudo gparted
  ```

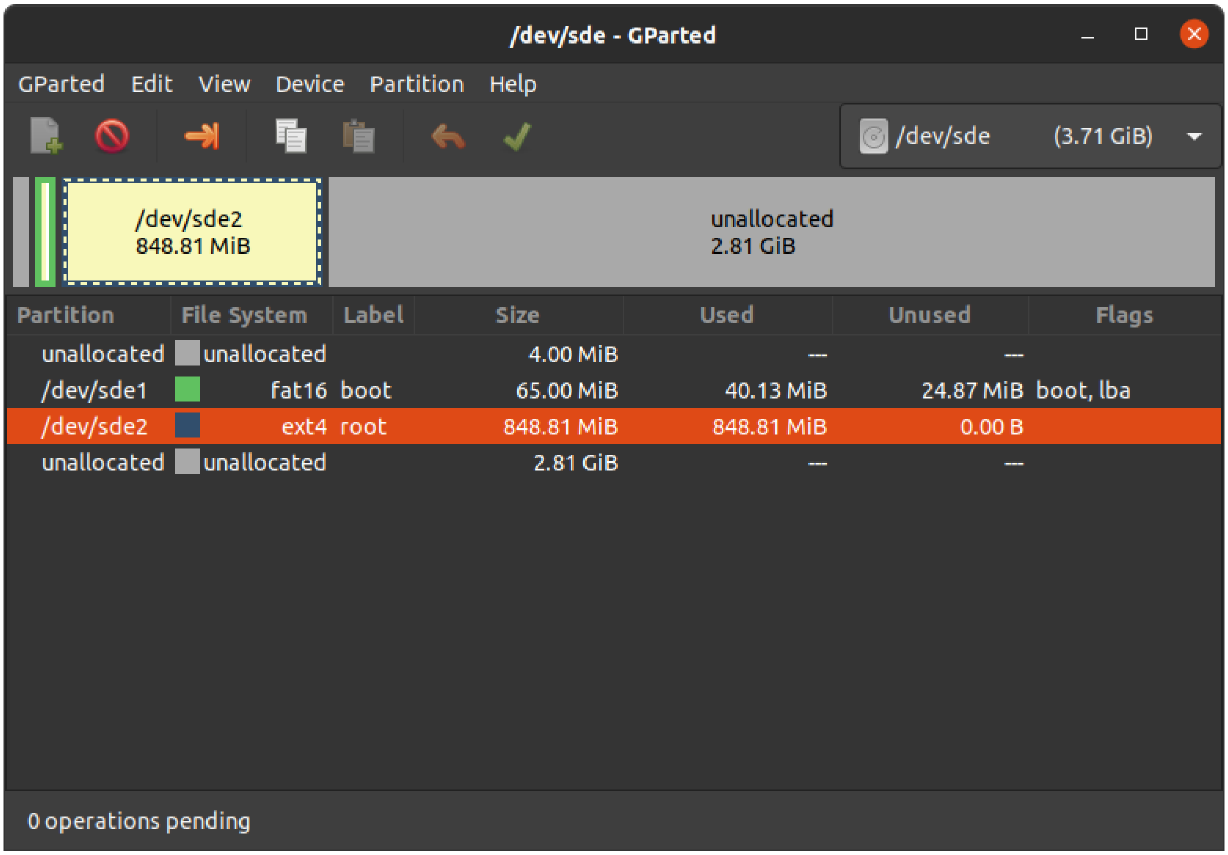


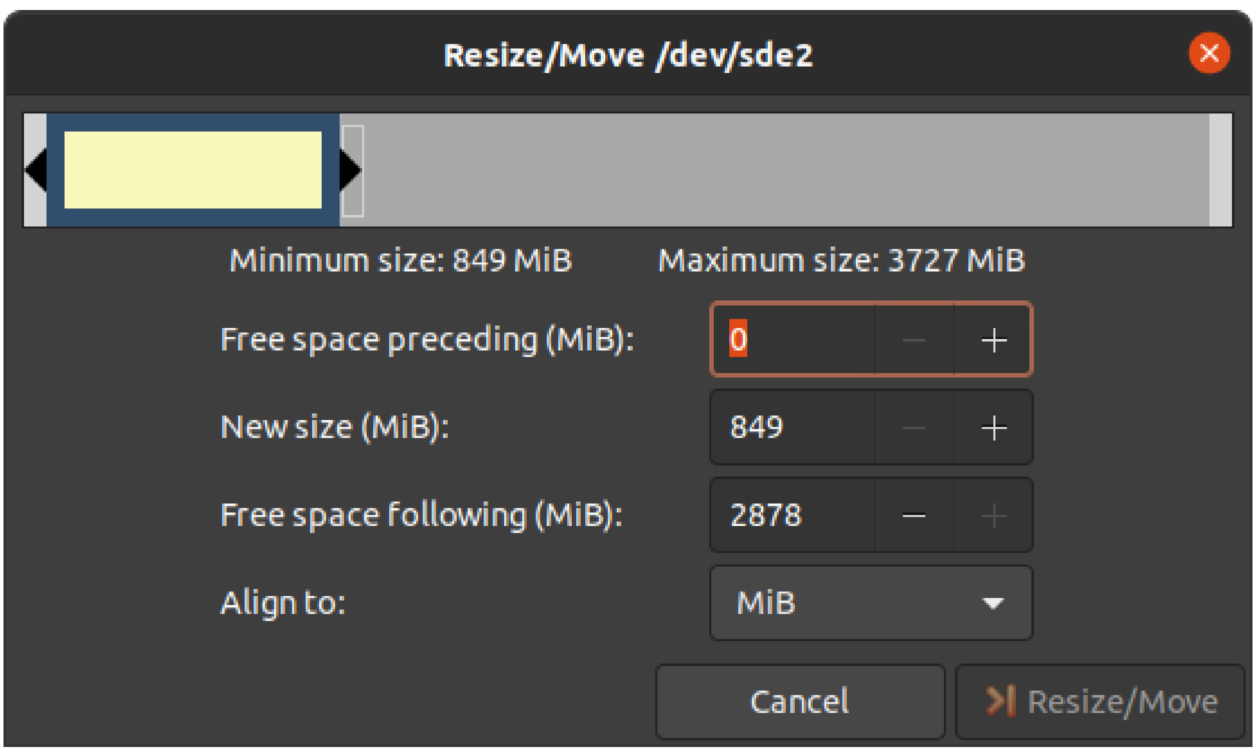

### 扩展根文件系统

**警告**

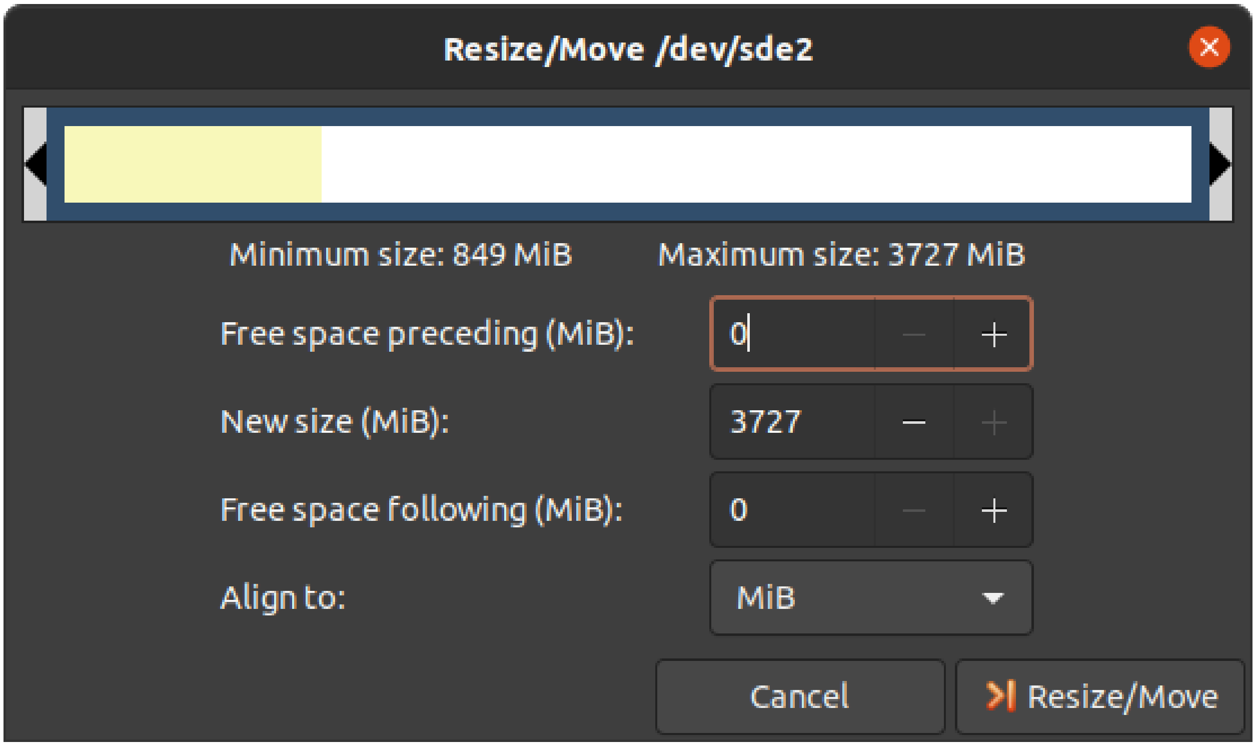
使用 resize2fs 版本 1.46.6 及更早版本的 PC 系统（例如 Ubuntu 22.04）无法烧写在 Mickledore 以及更新的 yocto 版本上创建的 partup 软件包。这个是因为 resize2fs 新增了默认选项而导致的兼容性问题。有关详细信息，请参阅 发布说明 。

- Choose your SD card device at the drop-down menu on the top right
- 选择 ext4 根分区并点击调整大小：

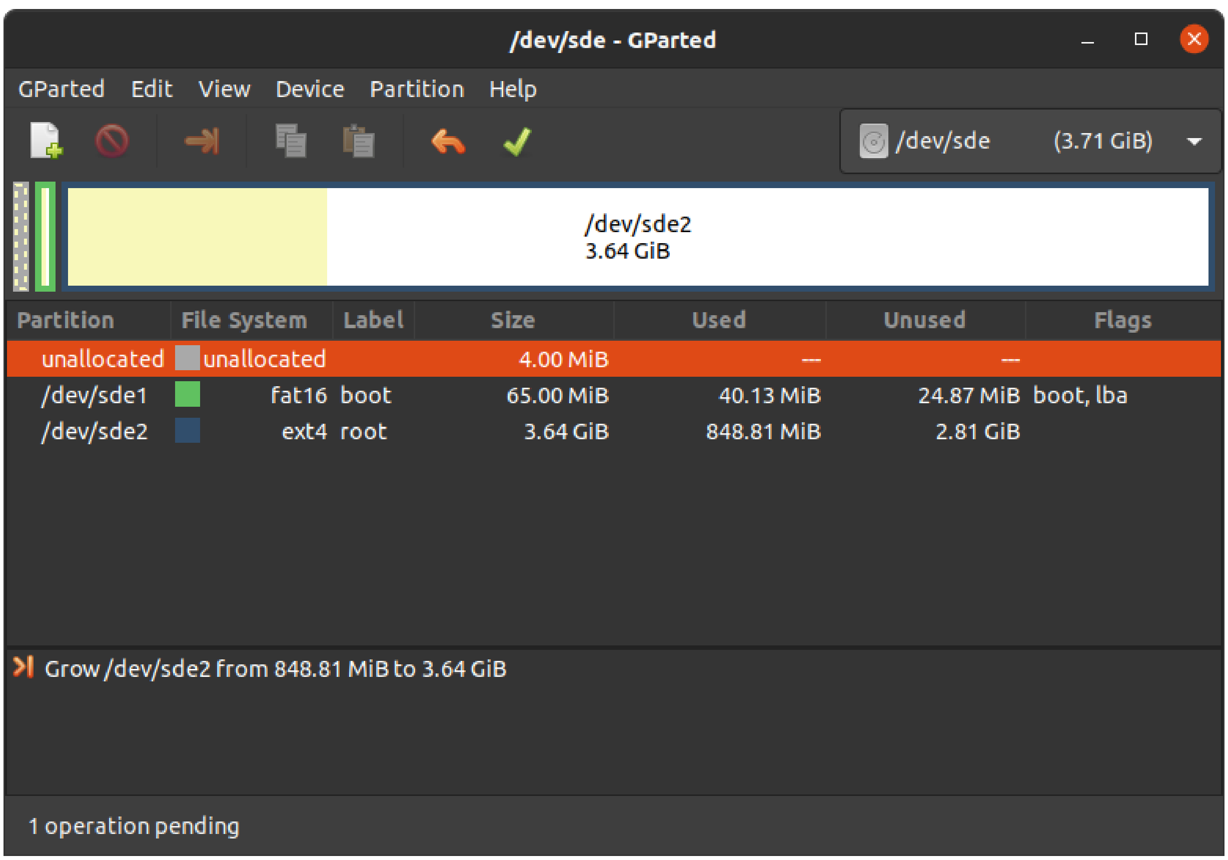



- 您可以根据需要拖动滑块或手动输入大小。




- 通过点击“Change Size”按钮确认您的输入。



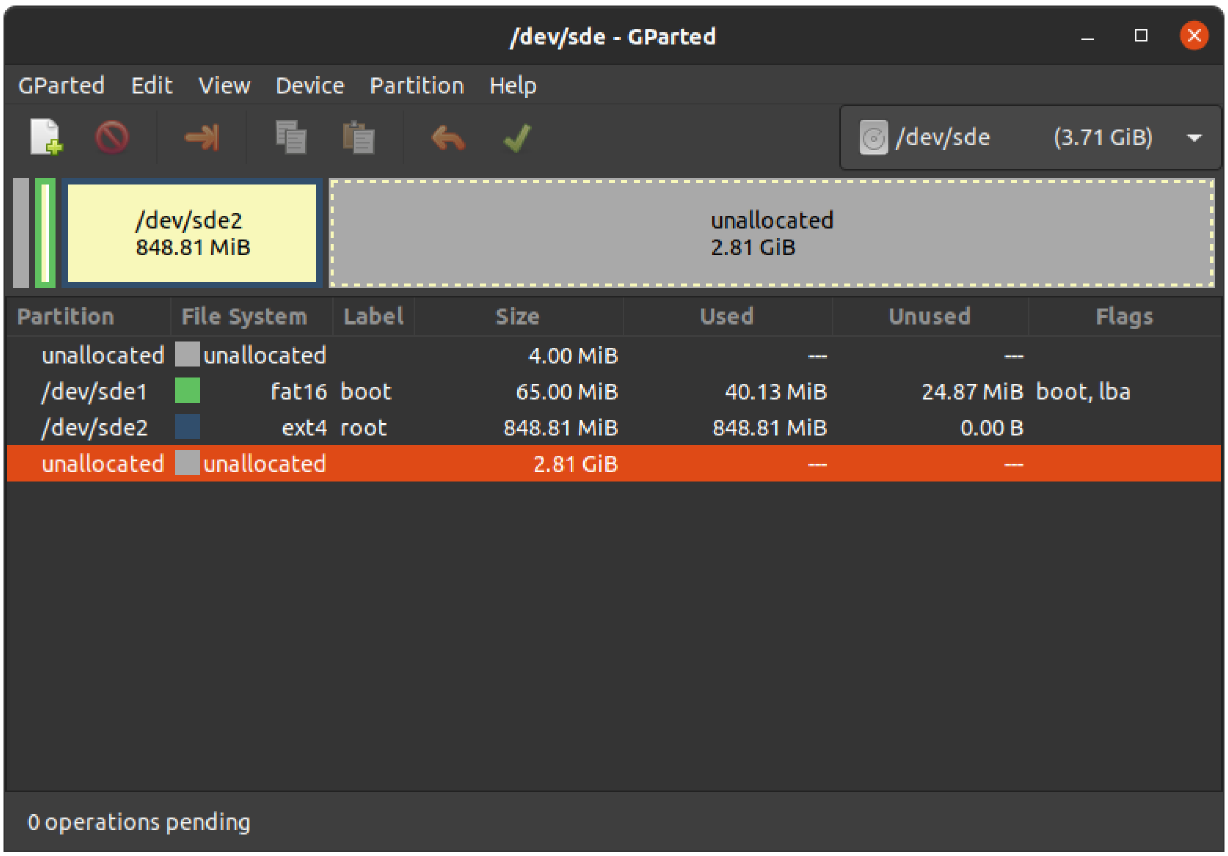

- 要应用您的更改，请按绿色勾号。

- 现在您可以挂载根分区并将 phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.wic 镜像复制到其中。然后再卸载它：

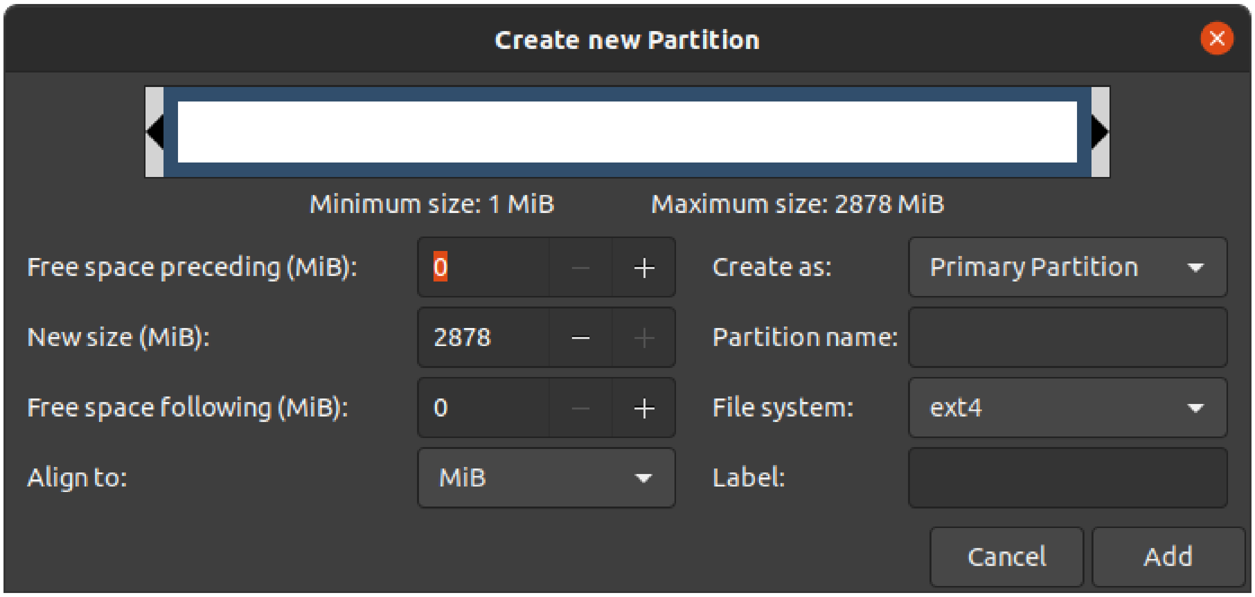
```
host:~$ sudo cp phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.wic /mnt/ ; sync
host:~$ umount /mnt
```

### 创建第三个分区

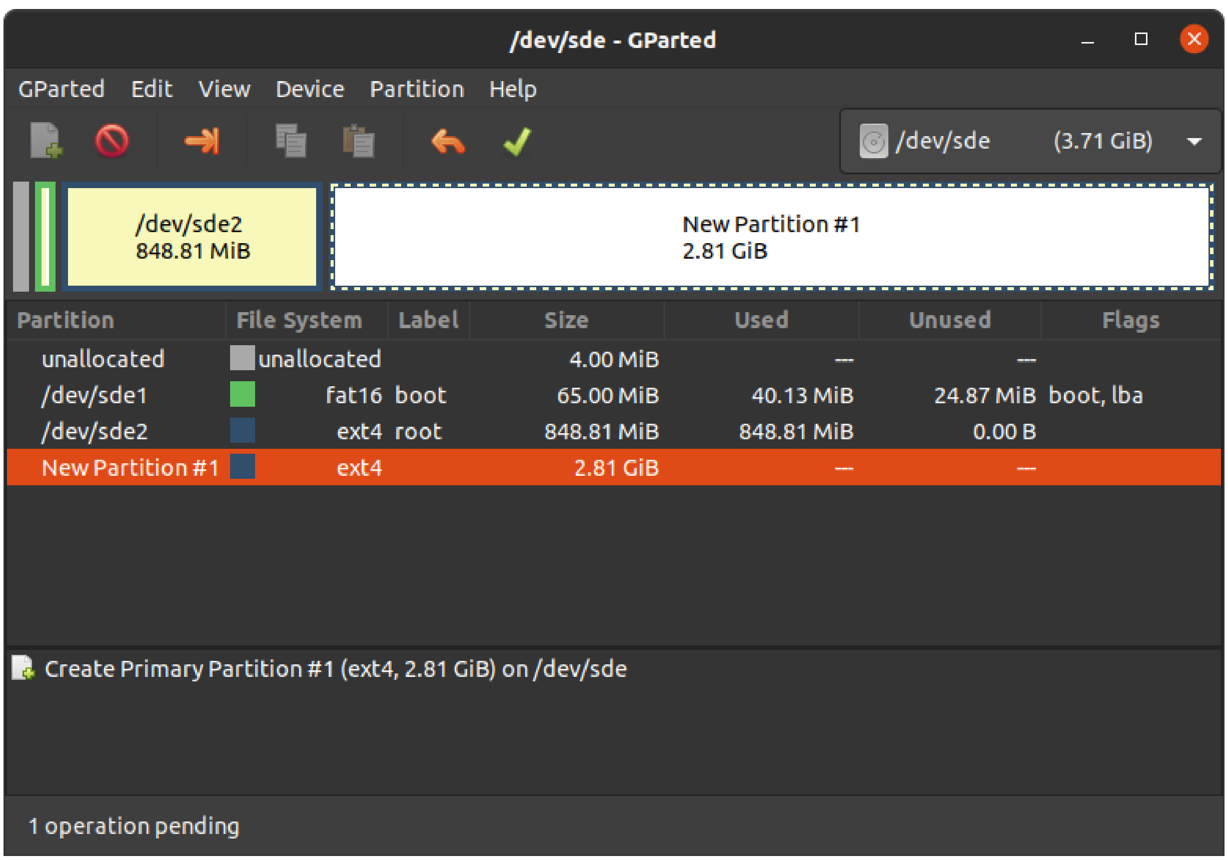
- Choose your SD card device at the drop-down menu on the top right



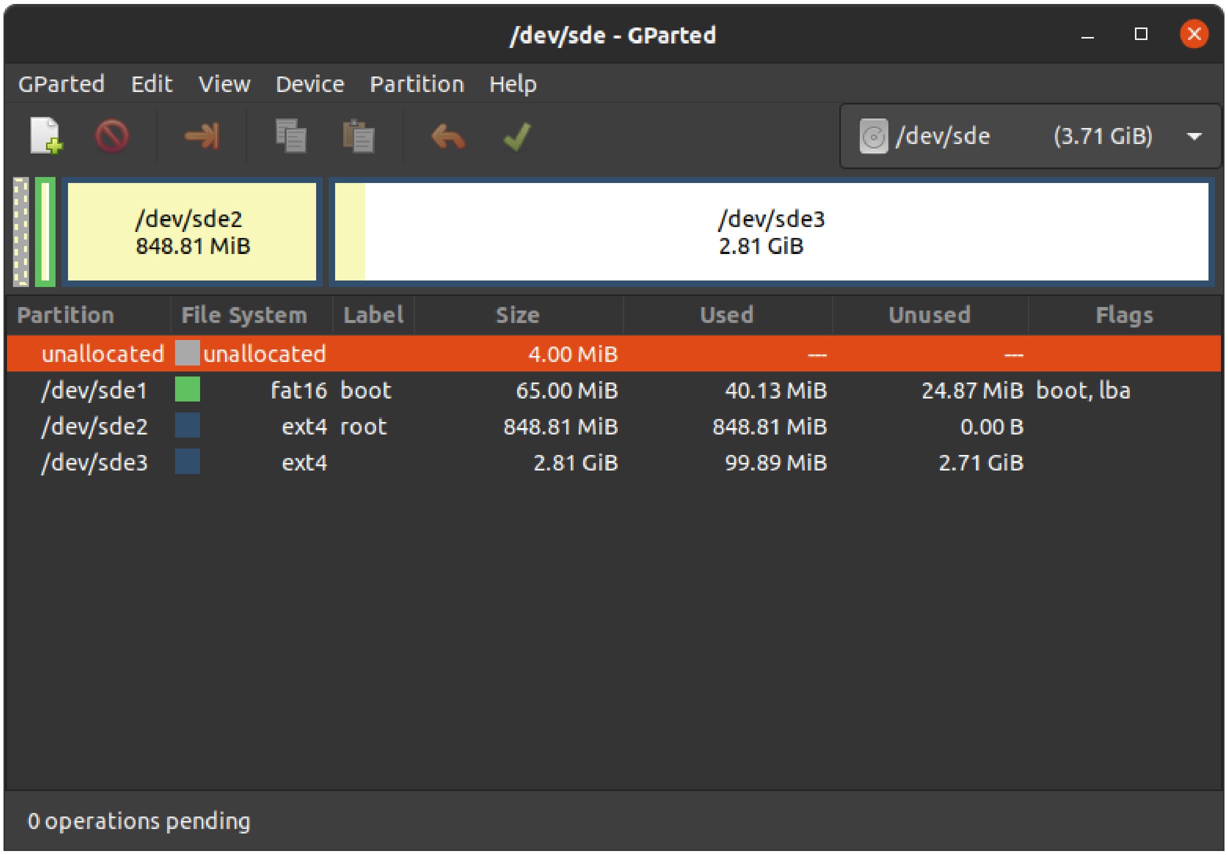

- 选择更大的未分配区域，然后点击”New”：




- 点击”Add”




- 按绿色勾确认更改。




- 现在您可以挂载新的分区并将 phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.wic 镜像复制到其中。然后卸载它：

```
host:~$ sudo mount /dev/sde3 /mnt
host:~$ sudo cp phytec-qt6demo-image-imx8mp-phyflex-libra-rdk-2.wic /mnt/ ; sync
host:~$ umount /mnt
```

## 5.10 Ampliphy-boot

Ampliphy-boot is a collection of bootscripts for booting the Ampliphy distro, which work uniformly across different PHYTEC platforms. The bootscripts are developed to work with the standard boot U-Boot feature. More details about ampliphy-boot, the bootscripts and their Yocto implementation can be found in the Yocto Reference Manual (walnascar).

Ampliphy-boot currently provides these bootscripts:

```
mmc_boot
mmc_boot_fit
net_boot_fit
spi_boot_fit
```

Not all platforms support all scripts. The **phyFLEX-i.MX 8M Plus Libra Rapid Development Kit** supports the following bootscripts:

```
mmc_boot_fit
net_boot_fit
```

### 5.10.1 U-Boot standard boot

U-Boot standard boot is a built-in way to boot different operating systems. It handles scanning devices for the supported boot methods and starts the bootflow.

For ampliphy-boot the `script` bootmethod is used. This means a bootscript is placed inside the boot partition on the bootdevice that should be booted. This bootscript contains the logic for booting the ampliphy distro that previously was stored inside the U-Boot environment. This leads to the environment being a lot less cluttered and better readable for the user.

U-Boot standard boot uses and sets some environment variables, which are documented here. The most interesting variables for the BSP user are:

**ip_dyn**
: determines if the bootscript is loaded with dhcp (dynamic ip) or tftp (static ip). Values can be [yes/no], [true/false], [1/0]. If the variable is not set, dhcp will be used as default.

**boot_script_dhcp**
: Sets the name of the bootscript loaded from network.

**boot_targets**
: Determines the scanned bootsources and their order.

**bootmeths**
: Determines the scanned bootmethods and their order.

Default values for `boot_targets` and `bootmeths` for standard boot can be set in the devicetree or in the environment. The environment variables will overwrite the devicetree values if set.

For the **phyFLEX-i.MX 8M Plus Libra Rapid Development Kit**, the default values are defined in the U-Boot devicetree (e.g. arch/arm/dts/imx8mp-phyflex-libra-rdk-u-boot.dtsi):

```
bootstd {
        bootph-verify;
        compatible = "u-boot,boot-std";

        filename-prefixes = "/", "/boot/";
        bootdev-order = "mmc2", "mmc1", "ethernet";

        rauc {
                compatible = "u-boot,distro-rauc";
        };

        script {
                compatible = "u-boot,script";
        };
};
```

The filename-prefixes property describes the paths that will be searched for the bootscripts. In this case this is the root of the partition as well as the boot folder. The bootdev-order property sets the default value for the boot_targets variable. The supported bootmeths will also be named. In this case the efi, rauc and script bootmethods are supported.

## 5.11 Working with FIT images

Flattened Image Tree (FIT) images can be used to pack binaries into a devicetree blob. They are used in our BSPs to pack the kernel, devicetree and devicetree overlays.

### 5.11.1 Show FIT image content

Printing the FIT image content on the host can be done with the following commands:

```
host:~$ dumpimage -l fitImage
```

or

```
host:~$ mkimage -l fitImage
```

Alternatively the content can also be printed on target in U-Boot:

```
u-boot=> load mmc ${mmcdev}:1 ${loadaddr} fitImage
u-boot=> iminfo ${loadaddr}
```

### 5.11.2 Building FIT images standalone

For building a FIT, an image tree source file (`*.its`) and binaries to be packed are needed. The `fitImage.its` file and kernel and devicetree binaries can be fetched from the Yocto deploy folder, the BSP downloads page or can be built standalone. The paths of the binaries in the `data` properties may need adjustment. For the kernel, the default data property will mention `linux.bin`, but the binary from the Yocto build is named `Image`.

Default:

```
images {
            kernel-1 {
                    description = "Linux kernel";
                    data = /incbin/("linux.bin");
                    type = "kernel";
...
```

Customized:

```
images {
            kernel-1 {
                    description = "Linux kernel";
                    data = /incbin/("Image");
                    type = "kernel";
...
```

Data properties for devicetrees also may need to be adjusted. After that the FIT image can be built using the `mkimage` command.

```
mkimage -f fitImage.its fitImage
```

### 5.11.3 Adding overlays to a FIT image

Adding overlays to the FIT image can be done by editing the `fitImage.its` file. For that the overlays need to be added to the `images` node and configurations need to be created for them. Already existing overlays can be used for reference. Only the names need to be adjusted.

CHAPTER 6

# 设备树 (DT)

## 6.1 介绍

以下文本简要描述了设备树，关于设备树的相关文档可以在 Linux kernel 文档中找到（https://docs.kernel.org/devicetree/usage-model.html）。

"Open Firmware Device Tree"或简称设备树（DT）是一种用于描述硬件的数据结构和语言。更具体地说，它是一个可由操作系统读取的硬件描述，以便操作系统不需要对 machine 的细节进行硬编码

内核文档是学习设备树的一个非常好的资源。关于设备树数据格式的概述可以在 devicetree.org 的设备树使用页面找到。

## 6.2 PHYTEC i.MX 8M Plus BSP 设备树概念

以下部分说明了 PHYTEC 配置基于 i.MX 8M Plus 的核心板设备树的一些规则。

### 6.2.1 设备树结构

- *Module.dtsi* - 文件包括所有安装在核心板上的设备，例如 PMIC 和 RAM。
- *Board.dts* - 包含核心板 dtsi 文件。从 SoC i.MX 8M Plus 引出并在底板使用的设备也包含在此 dts 中。
- *Overlay.dtso* - 根据核心板或底板上可选硬件（例如 SPI 闪存或 PEB-AV-10）的情况来启用/禁用一些功能。

在 Linux 内核的根目录下，我们的 i.MX 8 平台的设备树文件可以在 `arch/arm64/boot/dts/freescale/` 找到。

### 6.2.2 设备树 Overlay

设备树 Overlay 是可以在启动时合并到设备树中的设备树片段。下面是扩展板的硬件描述。对比源码中的 include，overlay 通过覆盖的方式来生效。overlay 也可以根据实际开发板的硬件配置来设置设备树节点状态。设备树 Overlay 与我们 Linux 内核仓库中的其他设备树文件一起放在子文件夹 `arch/arm64/boot/dts/freescale/` 中。

imx8mp-phyflex-libra-rdk-2.conf 可用的 overlay 文件有：

```
imx8mp-phyflex-libra-rdk-lvds-etml1010g3dra.dtbo
imx8mp-phyflex-libra-rdk-lvds-peb-av-10.dtbo
imx8mp-phyflex-libra-rdk-lvds-ph128800t006-zhc01.dtbo
imx8mp-phyflex-libra-rdk-vm016-csi1.dtbo
imx8mp-phyflex-libra-rdk-vm016-csi1-fpdlink-port0.dtbo
imx8mp-phyflex-libra-rdk-vm016-csi1-fpdlink-port1.dtbo
imx8mp-phyflex-libra-rdk-vm016-csi2.dtbo
imx8mp-phyflex-libra-rdk-vm016-csi2-fpdlink-port0.dtbo
imx8mp-phyflex-libra-rdk-vm016-csi2-fpdlink-port1.dtbo
imx8mp-phyflex-libra-rdk-vm017-csi1.dtbo
imx8mp-phyflex-libra-rdk-vm017-csi1-fpdlink-port0.dtbo
imx8mp-phyflex-libra-rdk-vm017-csi1-fpdlink-port1.dtbo
imx8mp-phyflex-libra-rdk-vm017-csi2.dtbo
imx8mp-phyflex-libra-rdk-vm017-csi2-fpdlink-port0.dtbo
imx8mp-phyflex-libra-rdk-vm017-csi2-fpdlink-port1.dtbo
imx8mp-phyflex-libra-rdk-vm020-csi1.dtbo
imx8mp-phyflex-libra-rdk-vm020-csi1-fpdlink-port0.dtbo
imx8mp-phyflex-libra-rdk-vm020-csi1-fpdlink-port1.dtbo
imx8mp-phyflex-libra-rdk-vm020-csi2.dtbo
imx8mp-phyflex-libra-rdk-vm020-csi2-fpdlink-port0.dtbo
imx8mp-phyflex-libra-rdk-vm020-csi2-fpdlink-port1.dtbo
imx8mp-phyflex-libra-rdk-vm024-csi1.dtbo
imx8mp-phyflex-libra-rdk-vm024-csi2.dtbo
```

可以在 linux 中使用以下命令来列出 FIT 镜像中所有的 overlay 配置

```
host:~$ dumpimage -l fitImage
```

或者在 u-boot：

```
u-boot=> load mmc ${mmcdev}:1 ${loadaddr} fitImage
u-boot=> iminfo
```

可以在 Linux 或 U-Boot 环境下配置 overlay。overlay 是在引导命令调用后、内核加载之前生效。接下来的部分将更详细地解释配置方法。

### Loading overlays

#### FIT image bootscripts

In ampliphy-boot FIT image bootscripts, the `fit_overlay_conf` U-Boot environment variable contains a number-sign (#) separated list of overlay configurations that will be applied during boot. This variable is used for overlays describing expansion boards and cameras that can not be detected during run time. The overlays listed in the fit_overlay_conf variable must be included in the FIT image. Overlays set in the $KERNEL_DEVICETREE Yocto machine variable will automatically be added to the FIT image.

The `fit_overlay_conf` variable can either be set directly in the U-Boot environment or can be part of the external `overlays.txt` environment file. By default, the fit_overlay_conf variable comes from the external overlays.txt environment file which is located in the boot partition. Otherwise if the fit_overlay_conf variable is set in U-Boot environment, the overlays.txt file value will not be loaded. The content from the file is defined in the Yocto recipe bootenv found in meta-phytec: https://github.com/phytec/meta-phytec/tree/walnascar/recipes-bsp/bootenv You can read and write the file on booted target from linux:

```
target:~$ cat /boot/overlays.txt
fit_overlay_conf=conf-imx8mp-phyflex-libra-rdk-overlay1.dtbo#conf-imx8mp-phyflex-libra-rdk-overlay2.dtbo
```

Changes will take effect after the next reboot. If no `overlays.txt` file is available the `fit_overlay_conf` variable can be set directly in the U-Boot environment.

```
u-boot=> setenv fit_overlay_conf conf-imx8mp-phyflex-libra-rdk-overlay1.dtbo
u-boot=> printenv fit_overlay_conf
fit_overlay_conf=conf-imx8mp-phyflex-libra-rdk-overlay1.dtbo
u-boot=> boot
```

#### Non FIT image bootscripts

In ampliphy-boot non FIT image bootscripts, the `overlays` U-Boot environment variable contains a space ( ) separated list of overlays that will be applied during boot. The overlays variable can also be set directly in the U-Boot environment or can be part of the external `overlays.txt` environment file.

By default, the overlays variable comes from the external overlays.txt environment file which is located in the boot partition. Otherwise if the overlays variable is set in U-Boot environment, the overlays.txt file value will not be loaded. The content from the file is defined in the Yocto recipe bootenv found in meta-phytec: https://github.com/phytec/meta-phytec/tree/walnascar/recipes-bsp/bootenv You can read and write the file on booted target from linux:

```
target:~$ cat /boot/overlays.txt
overlays=imx8mp-phyflex-libra-rdk-overlay1.dtbo imx8mp-phyflex-libra-rdk-overlay2.dtbo
```

Changes will take effect after the next reboot. If no `overlays.txt` file is available the `overlays` variable can be set directly in the U-Boot environment.

```
u-boot=> setenv overlays imx8mp-phyflex-libra-rdk-overlay1.dtbo
u-boot=> printenv overlays
overlays=imx8mp-phyflex-libra-rdk-overlay1.dtbo
u-boot=> boot
```

### 6.2.3 在 Linux 环境下更改开发板上的 U-boot 环境变量

Libubootenv 是我们镜像中包含的一个工具，用于在开发板 linux 上修改 U-Boot 环境。

使用以下命令打印 U-Boot 环境：

```
target:~$ fw_printenv
```

使用以下命令修改 U-Boot 环境：

```
target:~$ fw_setenv <variable> <value>
```

**小心**

Libubootenv 会读取配置文件中配置的环境变量。要修改的环境变量会被插入到该文件中，默认情况下使用 eMMC 中存储环境变量。

如果 eMMC 没有被烧写过或者 eMMC 环境被擦除，libubootenv 将无法工作。您应该修改 `/etc/fw_env.config` 文件，以匹配您想要使用的环境源。

### 6.2.4 PHYTEC i.MX 8M Plus FPSC Device Tree Differences

Due to the standardization of signals and interfaces that PHYTEC FPSC SoMs must supply, there is potential for confusion of the names as defined by the SoC manufacturer, in Software by the device tree and in the FPSC specification.

Example of i.MX 8M Plus SoC with a Libra i.MX 8M Plus FPSC board:

| SoC vendor | device tree | FPSC Spec. |
|---|---|---|
| uSDHC1 | usdhc1 | SDIO |
| uSDHC2 | usdhc2 | SD-Card |
| uSDHC3 | usdhc3 | |

To reduce confusion, interfaces in the device tree are annotated with the FPSC specification name.

Example (taken from imx8mp-phycore-fpsc.dtsi)

```
&usdhc2 { /* FPSC SDCARD */
        pinctrl-0 = <&pinctrl_usdhc2>;
        pinctrl-1 = <&pinctrl_usdhc2_100mhz>;
        pinctrl-2 = <&pinctrl_usdhc2_200mhz>;
        pinctrl-names = "default", "state_100mhz", "state_200mhz";
        sd-uhs-sdr104;
        vmmc-supply = <&reg_usdhc2_vmmc>;
        vqmmc-supply = <&ldo5>;
};

pinctrl_usdhc2: usdhc2grp {
        fsl,pins = <
                MX8MP_IOMUXC_SD2_CD_B__USDHC2_CD_B   0x40    /* SDCARD_CD */
                MX8MP_IOMUXC_SD2_WP__USDHC2_WP       0x40    /* SDCARD_WP */
                MX8MP_IOMUXC_SD2_CLK__USDHC2_CLK     0x190   /* SDCARD_CLK */
                MX8MP_IOMUXC_SD2_CMD__USDHC2_CMD     0x1d0   /* SDCARD_CMD */
                MX8MP_IOMUXC_SD2_DATA0__USDHC2_DATA0 0x1d0   /* SDCARD_DATA0 */
                MX8MP_IOMUXC_SD2_DATA1__USDHC2_DATA1 0x1d0   /* SDCARD_DATA1 */
                MX8MP_IOMUXC_SD2_DATA2__USDHC2_DATA2 0x1d0   /* SDCARD_DATA2 */
                MX8MP_IOMUXC_SD2_DATA3__USDHC2_DATA3 0x1d0   /* SDCARD_DATA3 */
                MX8MP_IOMUXC_GPIO1_IO04__USDHC2_VSELECT      0xc0
        >;
};
```

In the first node, the interface as defined per FPSC is annotated: `FPSC SDCARD` In the second node, the signal names for this (SDCARD) interface are annotated: `SDCARD_WP`, etc. .

### FPSC Device Tree Structure

In addition to the usual PHYTEC device tree structure, all FPSC defined signals are also predefined in the SoM device tree as can be seen in the example above. There is nothing connected to the SDCARD controller (on the SoM), yet the FPSC SoM must preconfigure SDCARD signals per the FPSC specification. The result is that on the baseboard, no care needs to be taken in regards to signal line configuration (muxing); this is done already.

Example cont. (imx8mp-libra-rdk-fpsc.dts)

```
/* SD-Card */
&usdhc2 {
        assigned-clocks = <&clk IMX8MP_CLK_USDHC2>;
        assigned-clock-rates = <200000000>;
        bus-width = <4>;
        disable-wp;
        status = "okay";
};
```

### Using the phyFLEX-i.MX 8M Plus FPSC Gamma for non-FPSC-conformant carrier boards

Even though signals for pins are predefined within the FPSC world, the phyFLEX-i.MX 8M Plus FPSC Gamma may also be used in custom, non-conforming (with regards to FPSC), i.e. ”phyCORE”, design context.

Example:

SDCARD interface is not needed on custom board, but an additional i2c interface is required.

The following could be done

```
&i2c4 {
     pinctrl-0 = <&pinctrl_i2c4>;
     pinctrl-names = "default";
     status = "okay";
};

pinctrl_i2c4: i2c4grp {
       fsl,pins = <
               MX8MP_IOMUXC_SD2_DATA0__I2C4_SDA 0x400001c2
               MX8MP_IOMUXC_SD2_DATA1__I2C4_SCL 0x400001c2
       >;
};
```

Note that SD2_DATA0 pad, even though muxed in the phyFLEX-i.MX 8M Plus FPSC Gamma dtsi file, is not used since the interface controller line it is muxed to is not activated with a `status = "okay";` property.

CHAPTER 7

# 访问外设

To find out which boards and modules are supported by the release of PHYTEC's phyFLEX-i.MX 8M Plus FPSC Gamma BSP described herein, visit our BSP web page and click the corresponding BSP release in the download section. Here you can find all hardware supported in the columns "Hardware Article Number" and the correct machine name in the corresponding cell under "Machine Name".

为了最大化软件的可复用性，Linux 内核提供了一个巧妙的软件架构，软件会根据不同硬件组件来分层。BSP（板级支持包）尽可能地对套件的功能进行模块化。当定制开发板或自定义核心板时，大部分软件配置可以简单的复制粘贴。与具体的开发板相关的内核代码可以在内核代码仓库中的设备树（DT）中找到，路径为 `arch/arm64/boot/dts/freescale/*.dts` 。

实际上，软件复用是 Linux 内核最重要的特性之一，尤其是在 ARM 架构中，它必须应对大量复杂且不同的系统级芯片（SoC）。整个开发板的硬件在设备树（DT）中描述，独立于内核镜像。硬件描述在一个单独的二进制文件中，称为设备树二进制文件（Device Tree Blob，DTB）（参见*device tree*）。

请阅读 PHYTEC i.MX 8M Plus BSP 设备树概念部分，以了解我们的 i.MX 8 BSP 设备树模型。

以下部分概述了 i.MX 8 平台上支持的硬件组件及其对应操作系统驱动程序。客户可以根据自身的需求进行更改。

## 7.1 i.MX 8M Plus 引脚复用

该 i.MX 8M Plus Soc 包含许多外设接口。为了在保持最大功能性的同时减少封装尺寸和降低整体系统成本，许多 i.MX 8M Plus 引脚可以多路复用为多达八种信号功能。尽管存在许多可能的引脚多路复用组合，但由于时序限制，只有一定数量的组合被称为有效的 IO 集合。这些有效的 IO 集合经过精心挑选，以为用户提供尽可能多的应用场景。

请参考我们的硬件手册或 NXP i.MX 8M Plus 参考手册，以获取有关特定引脚和复用能力的更多信息。

IO 集合的配置，也称为复用（muxing），是在设备树中完成的。驱动程序 pinctrl-single 读取设备树的节点 fsl,pins，并进行引脚复用配置。

The following is an example of the pin muxing of the UART3 device in imx8mp-phyflex-fpsc-g-som.dtsi:

```
pinctrl_uart4: uart4grp {
        fsl,pins = <
                MX8MP_IOMUXC_UART4_RXD__UART4_DCE_RX 0x140   /* UART3_RXD */
                MX8MP_IOMUXC_UART4_TXD__UART4_DCE_TX 0x140   /* UART3_TXD */
        >;
};
```

The first part of the string MX8MP_IOMUXC_UART4_RXD___UART4_DCE_RX names the pad (in this example UART4_RX). The second part of the string (UART3_DCE_RX) is the desired muxing option for this pad. The pad setting value (hex value on the right) defines different modes of the pad, for example, if internal pull resistors are activated or not. In this case, the internal resistors are disabled.

The device tree representation for UART3 pinmuxing: https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-fpsc-g-som.dtsi#L612

## 7.2 RS232

The FPSC Standard supports 3 UART units. On the phyFLEX Libra, TTL level signals of UART3 (the standard console) and UART2 are routed to a FT4232H UART to USB converter expansion. This USB is brought out at USB-C connector X14. UART1 is connected to a multi-protocol transceiver for RS-232 and RS-485, available at pin header connector *X27* at the RS-232 level, or at the RS-485 level. The muxing of the used transceivers is done by switch *S5* on the baseboard. Presently, RS485 is not working and will be fixed in 1618.4 SoM revision. For more information about the correct setup please refer to the phyFLEX-i.MX 8M Plus FPSC Gamma/phyFLEX Libra Hardware Manual section UARTs. The switch *S5* need to be set correctly.

- 以人类可读的格式显示终端的当前设置：

```
target:~$ stty -a
```

- By default crtscts is enabled, as hardware flow control is not functioning, need to configure UART interface with stty. This will be fixed in 1618.4 SoM revision. For example:

```
target:~$ stty -F /dev/ttymxc1 115200 -crtscts raw -echo
```

- 通过简单的 echo 和 cat，可以测试基本的通信。示例：

```
target:~$ echo 123 > /dev/ttymxc1
```

```
host:~$ cat /dev/ttyUSB2
```

The device tree representation for RS232: https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-libra-rdk.dts#L299

## 7.3 Ethernet

phyFLEX Libra-i.MX 8M Plus 提供两个以太网接口。我们的核心板和底板各提供一个千兆以太网接口。

所有接口都提供一个标准的 Linux 网络端口，可以使用 BSD 套接字接口进行编程。整个网络配置由 systemd-networkd 守护进程管理。相关的配置文件可以在开发板的 `/lib/systemd/network/` 目录中找到，以及在 BSP 中的 `meta-ampliphy/recipes-core/systemd/systemd-conf` 目录中。

IP addresses can be configured within *.network files. The interfaces are configured to static IP as default. The default IP address and netmask for end1 is:

```
end1: 192.168.3.11/24
```

To configure end1 to dynamic IP over DHCP, go to `/lib/systemd/network/\*-end1.network` and delete the line:

```
Address=192.168.3.11/24
```

The DT Ethernet setup might be split into two files depending on your hardware configuration: the module DT and the board-specific DT. The device tree set up for the ethernet where the PHY is populated on the SoM can be found here: https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-fpsc-g-som.dtsi#L94.

The device tree set up for EQOS Ethernet IP core where the PHY is populated on the phyFLEX Libra can be found here: https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-libra-rdk.dts#L94.

### 7.3.1 网络配置

#### U-boot 网络环境

- 要在 bootloader 中查找以太网设置：

  ```
  u-boot=> printenv ipaddr serverip netmask
  ```

- 在将主机设置为 IP 192.168.3.10 和子网掩码 255.255.255.0 的情况下，开发板应该返回：

  ```
  u-boot=> printenv ipaddr serverip netmask
  ipaddr=192.168.3.11
  serverip=192.168.3.10
  netmask=255.225.255.0
  ```

- 如果您需要进行任何更改：

  ```
  u-boot=> setenv <parameter> <value>
  ```

  <parameter> 应该是 ipaddr、netmask、gatewayip 或 serverip 中的一个。<value> 将是所选参数的设定值。

- 您所做的更改目前是临时的。要保存这些更改：

  ```
  u-boot=> saveenv
  ```

在这里，您也可以将 IP 地址更改为 DHCP，而不是使用静态 IP 地址。

- 配置：

  ```
  u-boot=> setenv ip dhcp
  ```

- 设置 TFTP 和 NFS 的路径。修改可以如下所示：

  ```
  u-boot=> setenv nfsroot /home/user/nfssrc
  ```

请注意，这些修改只会影响 bootloader 的设置。

**小技巧**

像 nfsroot 和 netargs 这样的变量可以被 U-boot 外部环境重新赋值。对于网络启动，外部环境将通过 tftp 加载。例如，要在 `bootenv.txt` 文件中设置 nfsroot 变量，请在 tftproot 目录修改：

```
nfsroot=/home/user/nfssrc
```

无需在开发板上存储这些信息。请注意，U-boot 外部环境对于像 ipaddr 或 serveraddr 这样的变量不起作用，因为它们在加载外部环境之前已经被设置完成。

#### Multiple network interfaces in U-Boot

Some boards may have support for multiple network interfaces in U-Boot. U-Boot lists all supported network interfaces during startup. This generic example illustrates support for multiple network interfaces:

```
Net:
eth0: ethernet@10000000, eth1: ethernet@10010000
```

There are three environment variables that will contribute to the selection of the ethernet interface for example when loading a file over tftp.

**`ethrotate`**
: Determines if other interfaces will be probed if the first one fails. Valid values are yes/no. Default is yes if unset.

**`ethact`**
: Sets the current active ethernet interface. This interface will be used (first). It is usually unset at boot, but will be set automatically by U-Boot to the last used interface as soon as ethernet is used for the first time. U-Boot will overwrite existing values. Only one ethernet interface can be set, lists are not allowed.

**`ethprime`**
: This will determine the preferred ethernet interface if ethact is unset. Otherwise this will be ignored. Only one ethernet interface can be set, lists are not allowed.

**备注**

The variables `ipaddr`, `serverip` and `netmask` are global and not specific to a certain ethernet interface, so they might need to be adapted to the ethernet interface in use.

#### Secondary Ethernet Interface Configuration in U-Boot

By default, U-Boot utilizes the Ethernet PHY located on the module. To use the network connection provided by the PHY on the carrier board, configuration changes are required.

To enable the secondary Ethernet interface in U-Boot, the active Ethernet connection must be adjusted. The IP address configuration in U-Boot may also need modification.

Configure the development host with IP address 192.168.4.10 and netmask 255.255.255.0. The target device must then be configured as follows:

```
u-boot=> setenv ethact eth2
u-boot=> setenv ipaddr 192.168.4.11
```

### 内核网络环境

- Find the ethernet settings for end1 in the target kernel:

```
target:~$ ip -statistics address show end1
2: end1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 50:2d:f4:19:d6:33 brd ff:ff:ff:ff:ff:ff
    RX:  bytes packets errors dropped  missed   mcast
             0       0      0       0       0       0
    TX:  bytes packets errors dropped carrier collsns
             0       0      0       0       0       0
```

- Temporary adaption of the end1 configuration:

```
target:~$ ip address add 192.168.3.11/24 dev end1
```

## 7.4 SD card

The i.MX 8M Plus supports a slot for Secure Digital cards to be used as general-purpose block devices. These devices can be used in the same way as any other block device.

**警告**

这些设备是热插拔的。然而，您必须确保在设备仍然挂载时不要拔掉它。这可能会导致数据丢失！

After inserting an SD card, the kernel will generate new device nodes in /dev. The full device can be reached via its /dev/mmcblk1 device node. SD card partitions will show up as:

```
/dev/mmcblk1p<Y>
```

<Y> 作为分区编号，从 1 开始计数，直到该设备的最大分区数量。分区可以使用任何类型的文件系统进行格式化，并且可以以标准方式进行处理，例如，可以使用 mount 和 umount 命令进行分区挂载和卸载。

**小技巧**

这些分区设备节点要求 SD 卡包含有效的分区表（类似于“硬盘”）。如果没有分区表，则整个设备作为一个文件系统使用（类似于“软盘”）。在这种情况下，必须使用 /dev/mmcblk1 进行格式化和挂载。卡始终以可写方式挂载。

DT configuration for the MMC (SD card slot) interface can be found here: https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-fpsc-g-som.dtsi#L821 and https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-libra-rdk.dts#L345

DT configuration for the e.MMC interface can be found here: https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-fpsc-g-som.dtsi#L831

## 7.5 e.MMC Devices

PHYTEC modules like phyFLEX-i.MX 8M Plus FPSC Gamma are populated with an e.MMC memory chip as the main storage. e.MMC devices contain raw Multi-Level Cells (MLC) or Triple-Level Cells (TLC) combined with a memory controller that handles ECC and wear leveling. They are connected via an SD/MMC interface to the i.MX 8M Plus and are represented as block devices in the Linux kernel like SD cards, flash drives, or hard disks.

The electric and protocol specifications are provided by JEDEC (https://www.jedec.org/standards-documents/technology-focus-areas/flash-memory-ssds-ufs-emmc/e-mmc). The e.MMC manufacturer's datasheet is relatively short and meant to be read together with the supported version of the JEDEC e.MMC standard.

PHYTEC currently utilizes the e.MMC chips with JEDEC Version 5.0 and 5.1

### 7.5.1 扩展 CSD 寄存器

e.MMC devices have an extensive amount of extra information and settings that are available via the Extended CSD registers. For a detailed list of the registers, see manufacturer datasheets and the JEDEC standard.

在 Linux 用户空间中，您可以查询寄存器：

```
target:~$ mmc extcsd read /dev/mmcblk2
```

你将会看到：

```
=============================================
  Extended CSD rev 1.7 (MMC 5.0)
=============================================

Card Supported Command sets [S_CMD_SET: 0x01]
[...]
```

### 7.5.2 使能后台操作 (BKOPS)

In contrast to raw NAND Flash, an e.MMC device contains a Flash Transfer Layer (FTL) that handles the wear leveling, block management, and ECC of the raw MLC or TLC. This requires some maintenance tasks (for example erasing unused blocks) that are performed regularly. These tasks are called **Background Operations (BKOPS)**.

默认情况下（取决于芯片），后台操作可能会定期执行，也可能不会，他影响读写的最大延迟时间。

The JEDEC Standard has specified a method since version v4.41 that the host can issue BKOPS manually. See the JEDEC Standard chapter Background Operations and the description of registers BKOPS_EN (Reg: 163) and BKOPS_START (Reg: 164) in the e.MMC datasheet for more details.

寄存器 BKOPS_EN（寄存器：163）的位 MANUAL_EN（位 0）的含义：

- 值 0：主机不支持手动触发 BKOPS。设备写入性能会受到影响。
- 值 1：主机支持手动触发 BKOPS。当主机不进行设备读写时，它会不时触发 BKOPS。

The mechanism to issue background operations has been implemented in the Linux kernel since v3.7. You only have to enable BKOPS_EN on the e.MMC device (see below for details).

JEDEC 标准 v5.1 引入了一种新的自动 BKOPS 功能。它使主机能够定期触发后台操作，因为设备在空闲时会自动启动 BKOPS（请参见寄存器 BKOPS_EN（寄存器：163）中位 AUTO_EN 的描述）。

- 要检查 *BKOPS_EN* 是否已设置，请执行：

  ```
  target:~$ mmc extcsd read /dev/mmcblk2 | grep BKOPS_EN
  ```

  输出将会是，例如：

```
Enable background operations handshake [BKOPS_EN]: 0x01
#OR
Enable background operations handshake [BKOPS_EN]: 0x00
```

值 0x00 表示 BKOPS_EN 被禁用，设备的写入性能受到影响。值 0x01 表示 BKOPS_EN 被启用，主机将不时发起后台操作。

- 通过以下命令使能 BKOPS_EN：

```
target:~$ target:~$ mmc --help

[...]
mmc bkops_en <auto|manual> <device>
    Enable the eMMC BKOPS feature on <device>.
    The auto (AUTO_EN) setting is only supported on eMMC 5.0 or newer.
    Setting auto won't have any effect if manual is set.
    NOTE!  Setting manual (MANUAL_EN) is one-time programmable (unreversible) change.
```

- 要设置 BKOPS_EN 位，请执行：

```
target:~$ mmc bkops_en manual /dev/mmcblk2
```

- 为了确保新设置生效并且内核能够自动触发 BKOPS，请先关闭系统：

```
target:~$ poweroff
```

**小技巧**

BKOPS_EN 位是一次性可编程的，无法恢复。

### 7.5.3 可靠写入

有两种不同的可靠写入选项：

1. Reliable Write option for a whole e.MMC device/partition.
2. 单次写的可靠写入方式。

**小技巧**

Do not confuse e.MMC partitions with partitions of a DOS, MBR, or GPT partition table (see the previous section).

The first Reliable Write option is mostly already enabled on the e.MMCs mounted on the phyFLEX-i.MX 8M Plus FPSC Gamma SoMs. To check this on the running target:

```
target:~$ mmc extcsd read /dev/mmcblk2 | grep -A 5 WR_REL_SET
Write reliability setting register [WR_REL_SET]: 0x1f
 user area: the device protects existing data if a power failure occurs during a write o
peration
 partition 1: the device protects existing data if a power failure occurs during a write
 operation
 partition 2: the device protects existing data if a power failure occurs during a write
 operation
```

(续下页)

(接上页)

```
 partition 3: the device protects existing data if a power failure occurs during a write
 operation
 partition 4: the device protects existing data if a power failure occurs during a write
 operation
--
 Device supports writing EXT_CSD_WR_REL_SET
 Device supports the enhanced def. of reliable write
```

如果默认没有启用，可以使用 mmc 工具启用它：

```
target:~$ mmc --help

[...]
mmc write_reliability set <-y|-n|-c> <partition> <device>
    Enable write reliability per partition for the <device>.
    Dry-run only unless -y or -c is passed.
    Use -c if more partitioning settings are still to come.
    NOTE!  This is a one-time programmable (unreversible) change.
```

第二个可靠写入方式是命令 CMD23 中的配置位 Reliable Write Request parameter（可靠写入请求参数）（位 31）。自内核版本 v3.0 起，文件系统（例如 ext4 的日志）和用户空间应用程序（如 fdisk 的分区表）会通过内核使用该可靠写功能。在 Linux 内核源代码中，它通过标志 REQ_META 进行处理。

**结论**：使用挂载选项 data=journal 的 ext4 文件系统在断电情况下是安全的。文件系统检查可以在断电后恢复文件系统，但在断电前刚写入的数据可能会丢失。在各种情况下，都可以恢复文件系统的正常状态。为了确保应用程序文件的正常保存，应用程序中应使用系统函数 fdatasync 或 fsync。

### 7.5.4 调整 ext4 根文件系统的大小

When flashing the SD card image to e.MMC the ext4 root partition is not extended to the end of the e.MMC. parted can be used to expand the root partition. The example works for any block device such as e.MMC, SD card, or hard disk.

- 获取当前设备大小：

```
target:~$ parted /dev/mmcblk2 print
```

- 输出如下：

```
Model: MMC Q2J55L (sd/mmc)
Disk /dev/mmcblk2: 7617MB
Sect[ 1799.850385]  mmcblk2: p1 p2
or size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      4194kB  72.4MB  68.2MB  primary  fat16        boot, lba
 2      72.4MB  537MB   465MB   primary  ext4
```

- 使用 parted 将文件系统分区调整为设备的最大大小：

```
target:~$ parted /dev/mmcblk2 resizepart 2 100%
Information: You may need to update /etc/fstab.

target:~$ parted /dev/mmcblk2 print
```

(续下页)

(接上页)

```
Model: MMC Q2J55L (sd/mmc)
Disk /dev/mmcblk2: 7617MB
Sector size (logical/physical): 512[ 1974.191657]  mmcblk2: p1 p2
B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      4194kB  72.4MB  68.2MB  primary  fat16        boot, lba
 2      72.4MB  7617MB  7545MB  primary  ext4
```

- 将文件系统调整为新的分区大小：

```
target:~$ resize2fs /dev/mmcblk2p2
resize2fs 1.46.1 (9-Feb-2021)
Filesystem at /dev/mmcblk2p2 is mounted on /; on-line resizing required
[ 131.609512] EXT4-fs (mmcblk2p2): resizing filesystem
from 454136 to 7367680 blocks
old_desc_blocks = 4, new_desc_blocks = 57
[ 131.970278] EXT4-fs (mmcblk2p2): resized filesystem to 7367680
The filesystem on /dev/mmcblk2p2 is now 7367680 (1k) blocks long
```

Increasing the filesystem size can be done while it is mounted. But you can also boot the board from an SD card and then resize the file system on the e.MMC partition while it is not mounted.

### 7.5.5 启用伪 SLC 模式

e.MMC devices use MLC or TLC (https://en.wikipedia.org/wiki/Multi-level_cell) to store the data. Compared with SLC used in NAND Flash, MLC or TLC have lower reliability and a higher error rate at lower costs.

如果您更喜欢可靠性而不是存储容量，可以启用伪 SLC 模式或 SLC 模式。这个方法采用了增强属性，该属性在 JEDEC 标准中有所描述，可以对设备的一个连续区域设置。JEDEC 标准并未规定增强属性的实现细节和保证，这由芯片制造商自行决定。对于美光（Micron）芯片，增强属性提高了可靠性，但也将容量减半。

> **警告**
>
> 在设备上启用增强属性时，所有数据将被丢失。

以下步骤展示了如何启用增强属性。

- First obtain the current size of the e.MMC device with:

```
target:~$ parted -m /dev/mmcblk2 unit B print
```

  你将收到：

```
BYT;
/dev/mmcblk2:63652757504B:sd/mmc:512:512:unknown:MMC S0J58X:;
```

  如您所见，该设备的容量为 63652757504 字节 = 60704 MiB。

- 要获取启用伪 SLC 模式后的设备的大小，请使用：

```
target:~$ mmc extcsd read /dev/mmcblk2 | grep ENH_SIZE_MULT -A 1
```

例如:

```
Max Enhanced Area Size [MAX_ENH_SIZE_MULT]: 0x000764
i.e. 3719168 KiB
--
Enhanced User Data Area Size [ENH_SIZE_MULT]: 0x000000
i.e. 0 KiB
```

这里的最大大小是 3719168 KiB = 3632 MiB。

- 现在，您可以通过输入以下命令为整个设备设置增强属性，例如 3719168 KiB:

```
target:~$ mmc enh_area set -y 0 3719168 /dev/mmcblk2
```

你将获得:

```
Done setting ENH_USR area on /dev/mmcblk2
setting OTP PARTITION_SETTING_COMPLETED!
Setting OTP PARTITION_SETTING_COMPLETED on /dev/mmcblk2 SUCCESS
Device power cycle needed for settings to take effect.
Confirm that PARTITION_SETTING_COMPLETED bit is set using 'extcsd read' after power cycle
```

- 为了确保新设置已生效，请关闭系统:

```
target:~$ poweroff
```

并进行上下电。建议您现在确认设置是否正确。

- 首先，检查 ENH_SIZE_MULT 的值，它必须是 3719168 KiB:

```
target:~$ mmc extcsd read /dev/mmcblk2 | grep ENH_SIZE_MULT  -A 1
```

您应该看到:

```
Max Enhanced Area Size [MAX_ENH_SIZE_MULT]: 0x000764
i.e. 3719168 KiB
--
Enhanced User Data Area Size [ENH_SIZE_MULT]: 0x000764
i.e. 3719168 KiB
```

- 最后，检查设备的大小:

```
target:~$ parted -m /dev/mmcblk2 unit B print
BYT;
/dev/mmcblk2:31742492672B:sd/mmc:512:512:unknown:MMC S0J58X:;
```

### 7.5.6 擦除设备

It is possible to erase the e.MMC device directly rather than overwriting it with zeros. The e.MMC block management algorithm will erase the underlying MLC or TLC or mark these blocks as discard. The data on the device is lost and will be read back as zeros.

- After booting from SD card execute:

```
target:~$ blkdiscard -f --secure /dev/mmcblk2
```

选项 --secure 确保命令在 eMMC 设备擦除所有块之前会等待。-f (强制) 选项强制擦写，当 eMMC 设备包含有效数据分区时需要使用-f 选项。

**小技巧**

```
target:~$ dd if=/dev/zero of=/dev/mmcblk2 conv=fsync
```

该命令也会擦除设备上的所有信息，但这个命令不利于设备的磨损均衡，并且需要花费更长的时间！

### 7.5.7 e.MMC Boot Partitions

An e.MMC device contains four different hardware partitions: user, boot1, boot2, and rpmb.

The user partition is called the User Data Area in the JEDEC standard and is the main storage partition. The partitions boot1 and boot2 can be used to host the bootloader and are more reliable. Which partition the i.MX 8M Plus uses to load the bootloader is controlled by the boot configuration of the e.MMC device. The partition rpmb is a small partition and can only be accessed via a trusted mechanism.

此外，User 分区可以分为四个自定义的一般用途分区。对此功能的解释不在本文件涵盖的范围。有关更多信息，请参阅 JEDEC 标准第 7.2 章分区管理。

**小技巧**

Do not confuse e.MMC partitions with partitions of a DOS, MBR, or GPT partition table.

The current PHYTEC BSP does not use the extra partitioning feature of e.MMC devices. The U-Boot is flashed at the beginning of the user partition. The U-Boot environment is placed at a fixed location after the U-Boot. An MBR partition table is used to create two partitions, a FAT32 boot, and ext4 rootfs partition. They are located right after the U-Boot and the U-Boot environment. The FAT32 boot partition contains the kernel and device tree.

With e.MMC flash storage it is possible to use the dedicated boot partitions for redundantly storing the bootloader. The Bootloader environment still resides in the user area before the first partition. The user area also still contains the bootloader which the image first shipped during its initialization process. Below is an example, to flash the bootloader to one of the two boot partitions and switch the boot device via userspace commands.

#### 通过用户空间命令

在主机上运行：

```
host:~$ scp <bootloader> root@192.168.3.11:/tmp/
```

The partitions boot1 and boot2 are read-only by default. To write to them from user space, you have to disable `force_ro` in the sysfs.

To manually write the bootloader to the e.MMC boot partitions, first disable the write protection:

```
target:~$ echo 0 > /sys/block/mmcblk2boot0/force_ro
target:~$ echo 0 > /sys/block/mmcblk2boot1/force_ro
```

Write the bootloader to the e.MMC boot partitions:

```
target:~$ dd if=/tmp/<bootloader> of=/dev/mmcblk2boot0
target:~$ dd if=/tmp/<bootloader> of=/dev/mmcblk2boot1
```

下表是 i.MX 8M Plus SoC 的偏移量：

| SoC | User 分区偏移量 | Boot 分区偏移量 | e.MMC Device |
|---|---|---|---|
| i.MX 8M Plus | 32 kiB | 0 kiB | /dev/mmcblk2 |

After that set the boot partition from user space using the `mmc` tool:

(对于'boot0')：

```
target:~$ mmc bootpart enable 1 0 /dev/mmcblk2
```

(对于'boot1')：

```
target:~$ mmc bootpart enable 2 0 /dev/mmcblk2
```

To disable booting from the e.MMC boot partitions simply enter the following command:

```
target:~$ mmc bootpart enable 0 0 /dev/mmcblk2
```

To explicitly enable booting from the e.MMC user area, run:

```
target:~$ mmc bootpart enable 7 0 /dev/mmcblk2
```

#### Automatic failover

The ROM loader implements an automatic failover mechanism for e.MMC boot partitions. If booting from the primary partition fails, the system automatically attempts to boot from the secondary partition. This failover is indicated by a change in the boot message from `Boot Stage: Primary boot` to `Boot Stage: Secondary boot`. This functionality is limited to boot0 and boot1 partitions and does not apply to the user area.

## 7.6 SPI 主设备

i.MX 8M Plus 控制器包含一个 FlexSPI 和一个 ECSPI IP 核。FlexSPI 主控制器支持两个 SPI 通道，最多可连接 4 个设备。每个通道支持单通道/双通道/四通道/八通道模式的数据传输（1/2/4/8 条数据线）。ECSPI 控制器支持 3 个 SPI 接口，每个接口都有一个专用的 CS（chip select）引脚。由于 CS 也可通过 GPIO 实现，因此每个通道上可以连接多个设备。

### 7.6.1 SPI NOR 烧写

phyFLEX Libra is equipped with a QSPI NOR Flash which connects to the i.MX 8M Plus's FlexSPI interface.

From Linux userspace, the NOR Flash partitions are accessible via /dev/mtd<N> devices where <N> is the MTD device number associated with the NOR flash partition to access. To find the correct MTD device number for a partition, run on the target:

```
root@imx8mp-phyflex-libra-rdk-2:~$ mtdinfo --all
Count of MTD devices:           1
Present MTD devices:            mtd0
Sysfs interface supported:      yes

mtd0
Name:                           30bb0000.spi
Type:                           nor
Eraseblock size:                131072 bytes, 128.0 KiB
```

(续下页)

(接上页)

```
Amount of eraseblocks:          512 (67108864 bytes, 64.0 MiB)
Minimum input/output unit size: 1 byte
Sub-page size:                  1 byte
Character device major/minor:   90:0
Bad blocks are allowed:         false
Device is writable:             true
```

它列出了所有 MTD 设备及其对应的分区名称。闪存节点在模块 DTS 中的 SPI 主节点内定义。SPI 节点包含连接到此 SPI 总线的所有设备，在这种情况下只有 SPI NOR Flash。

设备树中 SPI 主节点的定义可以在这里找到：

https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-libra-rdk.dts#L136

## 7.7 GPIOs

phyFLEX Libra 具有一组专门用于 GPIO 的引脚。这些引脚直接连接到 i.MX 8M Plus 引脚，并被复用为 GPIO。它们可以直接在 Linux 用户空间中使用。处理器将其 GPIO 组织为 5 个 GPIO 组（GPIO1 – GPIO5)，每个组包含 32 个 GPIO。gpiochip0、gpiochip32、gpiochip64、gpiochip96 和 gpiochip128 是这些内部 i.MX 8M Plus GPIO 组 GPIO1 –GPIO5 的 sysfs 表示。

GPIO 被标识为 GPIO<X>_<Y>（例如：GPIO5_07)。<X> 表示 GPIO Bank，从 1 计数到 5，而 <Y> 表示该 Bank 内的 GPIO。<Y> 从 0 计数到 31（每个 bank 有 32 个 GPIO)。

相比之下，Linux 内核使用一个单一的整数来枚举系统中所有可用的 GPIO。计算正确数字的公式是：

```
Linux GPIO number: <N> = (<X> - 1) * 32 + <Y>
```

从用户空间访问 GPIO 将使用 libgpiod。它提供了一个库和工具，用于与 Linux GPIO 字符设备进行交互。以下是一些工具的用法示例：

- 检测芯片上的 gpiochips：

  ```
  target:~$ gpiodetect
  gpiochip0 [30200000.gpio] (32 lines)
  gpiochip1 [30210000.gpio] (32 lines)
  gpiochip2 [30220000.gpio] (32 lines)
  gpiochip3 [30230000.gpio] (32 lines)
  gpiochip4 [30240000.gpio] (32 lines)
  ```

- GPIO expander

  Beside the GPIOs of the i.MX 8M Plus SoC, the phyFLEX Libra has two GPIO expanders, which add more GPIOs to the system.

  ```
  gpiochip5 [2-0021] (16 lines)
  gpiochip6 [3-0021] (16 lines)
  ```

  DT configuration for the GPIO expanders can be found here:

  https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-libra-rdk.dts#L178

  https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-libra-rdk.dts#L202

- 显示关于 gpiochips 的详细信息，包括它们的名称、consumer、方向、活动状态和附加 flag：

```
target:~$ gpioinfo -c gpiochip0
```

- 读取 GPIO 的值（例如从 gpiochip0 的 GPIO 20）：

```
target:~$ gpioget -c gpiochip0 20
```

- 将 gpiochip0 上的 GPIO 20 的值设置为 0 并退出工具：

```
target:~$ gpioset -z -c gpiochip0 20=0
```

- gpioset 的帮助文本显示了可能的选项：

```
target:~$ gpioset --help
Usage: gpioset [OPTIONS] <line=value>...

Set values of GPIO lines.

Lines are specified by name, or optionally by offset if the chip option
is provided.
Values may be '1' or '0', or equivalently 'active'/'inactive' or 'on'/'off'.

The line output state is maintained until the process exits, but after that
is not guaranteed.

Options:
      --banner            display a banner on successful startup
  -b, --bias <bias>       specify the line bias
                  Possible values: 'pull-down', 'pull-up', 'disabled'.
                  (default is to leave bias unchanged)
      --by-name           treat lines as names even if they would parse as an offset
  -c, --chip <chip>       restrict scope to a particular chip
  -C, --consumer <name>   consumer name applied to requested lines (default is 'gpioset')
  -d, --drive <drive>     specify the line drive mode
                  Possible values: 'push-pull', 'open-drain', 'open-source'.
                  (default is 'push-pull')
  -h, --help              display this help and exit
  -l, --active-low        treat the line as active low
  -p, --hold-period <period>
                  the minimum time period to hold lines at the requested values
  -s, --strict            abort if requested line names are not unique
  -t, --toggle <period>[,period]...
                  toggle the line(s) after the specified period(s)
                  If the last period is non-zero then the sequence repeats.
      --unquoted  don't quote line names
  -v, --version           output version information and exit
  -z, --daemonize set values then detach from the controlling terminal

Chips:
    A GPIO chip may be identified by number, name, or path.
    e.g. '0', 'gpiochip0', and '/dev/gpiochip0' all refer to the same chip.

Periods:
    Periods are taken as milliseconds unless units are specified. e.g. 10us.
    Supported units are 's', 'ms', and 'us'.

*Note*
    The state of a GPIO line controlled over the character device reverts to default
```

(续下页)

(接上页)

```
when the last process referencing the file descriptor representing the device file exits.
This means that it's wrong to run gpioset, have it exit and expect the line to continue
being driven high or low. It may happen if given pin is floating but it must be interpreted
as undefined behavior.
```

**警告**

某些 GPIO 用于特殊功能。在使用某个 GPIO 之前，请参考您板子的原理图或硬件手册，以确保该 IO 未被其他功能占用。

### 7.7.1 通过 sysfs 访问 GPIO

**警告**

通过 sysfs 访问 GPIO 已经过时了，我们建议使用 libgpiod。

默认情况下不再支持通过 sysfs 访问 GPIO。只有手动在内核配置中启用 `CONFIG_GPIO_SYSFS` 后才能支持。要在 menuconfig 中使 `CONFIG_GPIO_SYSFS` 可见，必须首先启用选项 `CONFIG_EXPERT` 。

You can also add this option for example to the defconfig you use in `arch/arm64/configs/` in the linux kernel sources. For our NXP based releases, this could be for example imx8_phytec_defconfig:

```
..
CONFIG_EXPERT=y
CONFIG_GPIO_SYSFS=y
..
```

您也可以创建一个新的 config 片段。有关详细信息，请参阅我们的 Yocto Reference Manual。

## 7.8 LED 灯

如果有任何 LED 灯连接到 GPIO 管脚，您可以通过特定的 LED 驱动程序接口访问它们，而不是使用通用的 GPIO 接口（请参见 GPIO 部分）。您将通过 `/sys/class/leds/` 而不是 `/sys/class/gpio/` 来访问它们。LED 的最大亮度可以从 `max_brightness` 文件中读取。brightness 文件将设置 LED 的亮度（取值范围从 0 到 max_brightness）。大多数 LED 硬件上不支持调整亮度，所以在所有非零亮度下都会点亮。

下面是一个简单的例子。

要获取所有可用的 LED，请输入：

```
target:~$ ls /sys/class/leds
led-1@  led-2@  led-3@  mmc1::@  mmc2::@
```

The phyFLEX Libra provides the following LED indicators: red:status, green:status and blue:status.

- 打开 LED 灯：

  ```
  target:~$ echo 255 > /sys/class/leds/red\:status/brightness
  ```

- 关闭 LED：

```
target:~$ echo 0 > /sys/class/leds/red\:status/brightness
```

Device tree configuration for the User I/O configuration can be found here: https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-libra-rdk.dts#L219

## 7.9 I²C 总线

该 i.MX 8M Plus 包含多个多主支持快速模式的 I²C 模块。PHYTEC 板提供了许多不同的 I²C 设备，这些设备连接到 i.MX 8M Plus 的 I²C 模块。本节描述了我们 phyFLEX Libra 中集成的一些 I²C 设备的基本设备使用及其设备树（DT）表示。

i2c 的设备树节点包含一些设置，例如时钟频率，用于设置总线频率，以及引脚控制设置，包括 scl-gpios 和 sda-gpios，这些是用于总线恢复的备用引脚配置。

General I²C bus configuration from SoM (e.g. imx8mp-phyflex-fpsc-g-som.dtsi): https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-fpsc-g-som.dtsi#L205

General I²C bus configuration from carrierboard (e.g. imx8mp-phyflex-libra-rdk.dts) https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-libra-rdk.dts#L158

## 7.10 EEPROM

The system features three I2C EEPROM devices distributed across the SoM and carrier board:

On the phyFLEX-i.MX 8M Plus FPSC Gamma SoM:

- SoM Detection EEPROM (write-protected)
  - Bus: I2C-1
  - Address: 0x51
  - Purpose: Factory configuration for SoM identification
- User EEPROM
  - Bus: I2C-1
  - Address: 0x50
  - Purpose: Available for user applications

Device Tree Reference for SoM EEPROMs: https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-fpsc-g-som.dtsi#L293

And on the phyFLEX Libra carrier board:

- Board Detection EEPROM
  - Bus: I2C-2
  - Address: 0x51
  - Purpose: Reserved for carrier board identification
- User EEPROM
  - Bus: I2C-2

  - Address: 0x52
  - Purpose: Available for user applications

Device Tree Reference for Carrier Board: https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-libra-rdk.dts#L162

### 7.10.1 EEPROM reading and writing in U-Boot

In U-Boot the `i2c` command can be used for EEPROM read and write operations.

First set the correct i2c bus, where the eeprom is connected. *<bus-no>* is the i2c bus number of the EEPROM.

```
u-boot=> i2c dev <bus-no>
```

To read and print the first 32 bytes from EEPROM execute. *<addr>* is the i2c address of the EEPROM.

```
u-boot=> i2c md <addr> 0x00 0x20
```

To read the first 32 bytes from EEPROM into memory (loadaddr) execute:

```
u-boot=> i2c read <addr> 0x00 0x20 $loadaddr
```

To write 0xff to the first 32 bytes execute:

```
u-boot=> i2c mw <addr> 0x00 0xff 0x20
```

To write 32 bytes to EEPROM from memory (loadaddr) execute:

```
u-boot=> i2c write $loadaddr <addr> 0x00 0x20
```

### 7.10.2 EEPROM reading and writing in Linux

Reading and writing can also be done via the sysfs in Linux. For this, find the correct path in sysfs first. It follows this logic: /sys/class/i2c-dev/i2c-<bus-no>/device/<bus-no>-<addr>/eeprom *<bus-no>* is the bus number of the EEPROM *<addr>* is the address of the EEPROM in 4 digits hex.

To read and print the EEPROM content as a hex number, execute:

```
target:~$ hexdump -C <eeprom-sysfs-path>
```

or:

```
target:~$ dd if=<eeprom-sysfs-path> | od -x
```

To fill the EEPROM with zeros use:

```
target:~$ dd if=/dev/zero of=<eeprom-sysfs-path>
```

## 7.11 RTC

RTC 可以通过 `/dev/rtc*` 访问。由于 PHYTEC 板通常有多个 RTC，因此可能会有多个 RTC 设备文件。

- 要找到 RTC 设备的名称，可以通过以下方式读取其 sysfs 条目：

```
target:~$ cat /sys/class/rtc/rtc*/name
```

- 例如，你将得到：

```
rtc-rv3028 0-0052
snvs_rtc 30370000.snvs:snvs-rtc-lp
```

**小技巧**

这将列出所有实时时钟（RTC），包括非 I²C 接口的 RTC。如果存在设备树/aliases 条目，Linux 会根据这些条目分配 RTC 设备 ID。

日期和时间可以通过 `hwclock` 工具和 date 命令进行操作。要显示目标上设置的当前日期和时间：

```
target:~$ date
Thu Jan  1 00:01:26 UTC 1970
```

使用日期命令更改日期和时间。日期命令以以下语法设置时间："YYYY-MM-DD hh:mm:ss (+|-)hh:mm"：

```
target:~$ date -s "2022-03-02 11:15:00 +0100"
Wed Mar  2 10:15:00 UTC 2022
```

**备注**

您的时区（在此示例中为 +0100）可能会有所不同。

使用 date 命令并不会改变实时时钟（RTC）的时间和日期，因此如果我们重启开发板，这些更改将会被丢弃。要写入 RTC，我们需要使用 `hwclock` 命令。使用 `hwclock` 工具将当前的日期和时间（通过 date 命令设置）写入 RTC，然后重启开发板以检查更改是否已应用到 RTC 上：

```
target:~$ hwclock -w
target:~$ reboot
    .
    .
    .
target:~$ date
Wed Mar  2 10:34:06 UTC 2022
```

要从实时时钟（RTC）设置系统时间和日期，请使用：

```
target:~$ date
Thu Jan  1 01:00:02 UTC 1970
target:~$ hwclock -s
target:~$ date
Wed Mar  2 10:45:01 UTC 2022
```

### 7.11.1 RTC 唤醒 alarm

可以从实时时钟（RTC）发出中断以唤醒系统。该格式使用 Unix 纪元时间，即自 1970 年 1 月 1 日 UTC 午夜以来的秒数。要在从挂起到 RAM 状态后的 4 分钟唤醒系统，请输入：

```
target:~$ echo "+240" > /sys/class/rtc/rtc0/wakealarm
target:~$ echo mem > /sys/power/state
```

**备注**

内部唤醒 alarm 时间将被向上舍入到下一个分钟，因为 alarm 功能不支持秒。

### 7.11.2 RTC 参数

实时时钟（RTC）具有一些功能，可以通过 `hwclock` 工具进行读取和设置。

- 我们可以通过以下方式检查 RTC 支持的功能：

```
target:~$ hwclock --param-get features
The RTC parameter 0x0 is set to 0x71.
```

这个值的含义在内核中进行了编码，每个位的定义为：

```
#define RTC_FEATURE_ALARM               0
#define RTC_FEATURE_ALARM_RES_MINUTE    1
#define RTC_FEATURE_NEED_WEEK_DAY       2
#define RTC_FEATURE_ALARM_RES_2S        3
#define RTC_FEATURE_UPDATE_INTERRUPT    4
#define RTC_FEATURE_CORRECTION          5
#define RTC_FEATURE_BACKUP_SWITCH_MODE  6
#define RTC_FEATURE_ALARM_WAKEUP_ONLY   7
#define RTC_FEATURE_CNT                 8
```

- 我们可以通过以下方式检查 RTC BSM（Backup Switchover Mode 备份切换模式）：

```
target:~$ hwclock --param-get bsm
The RTC parameter 0x2 is set to 0x1.
```

- 我们可以通过以下方式设置 RTC BSM：

```
target:~$ hwclock --param-set bsm=0x2
The RTC parameter 0x2 will be set to 0x2.
```

BSM 位的定义为：

```
#define RTC_BSM_DISABLED   0
#define RTC_BSM_DIRECT     1
#define RTC_BSM_LEVEL      2
#define RTC_BSM_STANDBY    3
```

**小技巧**

您应该将 BSM 模式设置为 DSM 或 LSM，以便在初始电源不可用时，RTC 可以切换到备用电源。请查看 **RV-3028** RTC 的 Datasheet，以了解 LSM（电平切换模式）和 DSM（直接切换模式）这两个定义的工作模式。

DT representation for I²C RTCs: https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-fpsc-g-som.dtsi#L318

And the addions on the carrierboard: https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-libra-rdk.dts#L287

## 7.12 USB 主控制器

i.MX 8M Plus SoC 的 USB 控制器为众多消费类便携设备提供了一种低成本的连接解决方案，实现 USB 设备之间的数据传输，传输速度可达 4 Gbit/s（超高速'SS'）。USB 子系统具有两个独立的 USB 控制器。这两个控制器都能够作为 USBDevice 或 USB Host 使用。每个核心都连接到一个 USB 3.0 物理层（PHY）。

BSP 支持大容量存储设备（优盘）和键盘。其他与 USB 相关的设备驱动程序必须根据需要在内核配置中启用。由于 udev，所有连接的存储设备都会获得唯一的 ID，并可以在 `/dev/disk/by-id` 中找到。这些 ID 可以在 `/etc/fstab` 中用于以不同的方式挂载不同的 USB 存储设备。

DT representation for USB Host: https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-libra-rdk.dts#L327

## 7.13 CAN FD

phyFLEX Libra 支持两个 flexCAN 接口，支持 CAN FD。这些接口支持 Linux 标准 CAN 框架，该框架建立在 Linux 网络层之上。使用这个框架，CAN 接口表现得像普通的 Linux 网络设备，同时具备一些 CAN 特有的附加功能。更多信息可以在 Linux 内核文档中找到：https://www.kernel.org/doc/html/latest/networking/can.html

> **备注**
>
> The switches S6 and S7 are switching the 120 Ohm bus termination resistors. For proper functionality of the CAN FD interface, the bus needs to be terminated. If no external bus termination resistors are mounted, the switches S6 (for CAN FD1) and S7 (for CAN FD2) need to be set to ON.

- 使用：

  ```
  target:~$ ip link
  ```

  to see the state of the interfaces.

- To get information on fcan1, such as bit rate and error counters, type:

  ```
  target:~$ ip -d -s link show fcan1
  4: fcan1: <NOARP,UP,LOWER_UP,ECHO> mtu 72 qdisc pfifo_fast state UP mode DEFAULT group default qlen 10
     link/can  promiscuity 0 allmulti 0 minmtu 0 maxmtu 0
     can <FD> state ERROR-ACTIVE (berr-counter tx 0 rx 0) restart-ms 0
           bitrate 500000 sample-point 0.875
           tq 25 prop-seg 37 phase-seg1 32 phase-seg2 10 sjw 5 brp 1
           flexcan: tseg1 2..96 tseg2 2..32 sjw 1..16 brp 1..1024 brp_inc 1
           dbitrate 2000000 dsample-point 0.750
           dtq 25 dprop-seg 7 dphase-seg1 7 dphase-seg2 5 dsjw 2 dbrp 1
           flexcan: dtseg1 2..39 dtseg2 2..8 dsjw 1..4 dbrp 1..1024 dbrp_inc 1
           clock 40000000
           re-started bus-errors arbit-lost error-warn error-pass bus-off
           0          0          0          0          0          0         numtxqueues 1 numrxqueues 1␣
  ↪gso_max_size 65536 gso_max_segs 65535 tso_max_size 65536 tso_max_segs 65535 gro_max_size 65536
  gso_ipv4_max_size 65536 gro_ipv4_max_size 65536 parentbus platform parentdev 443a0000.can
     RX:  bytes packets errors dropped  missed   mcast
              0       0      0       0       0       0
  ```

（续下页）

(接上页)

```
    TX:  bytes packets errors dropped carrier collsns
            0       0      0       0       0       0
```

输出包含一组标准参数，这些参数也适用于以太网接口，因此并非所有参数对于 CAN 都是相关的（例如 MAC 地址）。以下输出参数包含有用的信息：

| fcan1 | 接口名称 |
|---|---|
| NOARP | CAN 无法使用 ARP 协议 |
| MTU | 最大传输单元 |
| RX packets | 接收的数据包数量 |
| TX packets | 发送的数据包数量 |
| RX bytes | 接收字节数 |
| TX bytes | 发送字节数 |
| errors… | 总线错误统计信息 |

The CAN configuration is done in the systemd configuration file `/lib/systemd/network/11-can.network`. For a persistent change of (as an example, the default bitrates), change the configuration in the BSP under `./meta-ampliphy/recipes-core/systemd/systemd-conf/11-can.network` in the root filesystem and rebuild the root filesystem.

**备注**

By default, we enable CAN-FD (flexible datarate) in our BSP. In case CAN Classic is required one needs to remove options `FDMode` and `DataBitRate` from the `/lib/systemd/network/11-can.network` file.

To disable flexible datarate manually, one can use:

```
target:~$ ip link set fcan1 down
target:~$ ip link set fcan1 type can bitrate 500000 dbitrate 0 fd off
target:~$ ip link set fcan1 up
```

The bitrate can also be changed manually, for example:

```
target:~$ ip link set fcan1 down
target:~$ ip link set fcan1 txqueuelen 10 up type can bitrate 500000 sample-point 0.75 dbitrate 4000000 dsample-point 0.8 fd on
target:~$ ip link set fcan1 up
```

您可以使用 cansend 发送消息，或使用 candump 接收消息：

```
target:~$ cansend fcan1 123#45.67
target:~$ candump fcan1
```

要生成用于测试目的的随机 CAN 流量，请使用 cangen：

```
target:~$ cangen
```

`cansend --help` 和 `candump --help` 提供了关于选项和用法的帮助信息。

Device Tree CAN configuration of imx8mp-phyflex-fpsc-g-som.dtsi: https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-fpsc-g-som.dtsi#L121 and of imx8mp-phyflex-libra-rdk.dts: https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy11/arch/arm64/boot/dts/freescale/imx8mp-phyflex-libra-rdk.dts#L126

## 7.14 视频

### 7.14.1 视频与 Gstreamer

默认情况下，BSP 安装了一个示例视频，路径为 */usr/share/qtphy/videos/* 。可以使用以下命令之一开始视频播放：

```
target:~$ gst-launch-1.0 playbin uri=file:///usr/share/qtphy/videos/caminandes_3_llamigos_720p_vp9.webm
```

- 或者：

```
target:~$ gst-launch-1.0 -v filesrc location=/usr/share/qtphy/videos/caminandes_3_llamigos_720p_vp9.webm ! decodebin name=decoder decoder. ! videoconvert ! waylandsink
```

- 或者：

```
target:~$ gst-play-1.0 /usr/share/qtphy/videos/caminandes_3_llamigos_720p_vp9.webm --videosink waylandsink
```

## 7.15 显示

The phyFLEX Libra supports up to 3 different display outputs. The following table shows the required extensions and devicetree overlays for the different interfaces. For the alpha release, we have included overlays for two different LVDS displays. These displays are `edt,etml1010g3dra` or `powertip,ph128800t006-zhc01`. The name can be found on the back of the display.

**备注**

Currently only LVDS0 (onboard LVDS) is supported

| 接口 | 扩展板 | 设备树 overlay | |
|---|---|---|---|
| LVDS0 | phyFLEX Libra | imx8mp-libra-rdk-fpsc-lvds-etml1010g3dra.dtbo | imx8mp-libra-rdk-fpsc-lvds-ph128800t006-zhc01.dtbo |

**备注**

- 在更改 Weston 输出时，请确保音频输出也相匹配。
- LVDS0 (PEB-AV-10) and LVDS1 (onboard) can not be used at the same time.

The default interface is LVDS0 (onboard LVDS).

**备注**

The current display driver limits the pixel clock for a display connected to LVDS to 74.25 MHz (or a divider of it). If this does not fit your display requirements, please contact Support for further help.

### 7.15.1 Weston 配置

In order to get an output from Weston on the correct display, it still needs to be configured correctly. This will be done at `/etc/xdg/weston/weston.ini`.

#### 单一显示器

In our BSP, the default Weston output is set to LVDS-1 (onboard LVDS).

```
[output]
name=LVDS-1
mode=preferred
```

### 7.15.2 Qt Demo

使用 `phytec-qt6demo-image` 时，Weston 会在启动时启动。我们的 Qt6 DEMO 应用程序名为 "qtphy"，可以通过以下方式停止：

```
target:~$ systemctl stop qtphy
```

- 要重新开始 Demo，请运行：

  ```
  target:~$ systemctl start qtphy
  ```

- 要禁用 Demo 的自动启动，请运行：

  ```
  target:~$ systemctl disable qtphy
  ```

- 要启用 Demo 的自动启动，请运行：

  ```
  target:~$ systemctl enable qtphy
  ```

- Weston 可以通过以下方式停止：

  ```
  target:~$ systemctl stop weston
  ```

**备注**

在关闭 Weston 之前，必须先关闭 Qt Demo。

### 7.15.3 背光控制

如果 LCD 连接到 PHYTEC 开发板，可以通过 Linux 内核的 sysfs 接口控制其背光。系统中所有可用的背光设备可以在文件夹/sys/class/backlight 中找到。读取相应的文件并向其写入数据可以控制背光。

**备注**

一些具有多显示的开发板在 /sys/class/backlight 有多个背光控制。比如：backlight0 和 backlight1

- 例如，要获取最大亮度级别（max_brightness），请执行：

  ```
  target:~$ cat /sys/class/backlight/backlight/max_brightness
  ```

  有效的亮度值范围是 0 到 <max_brightness>。

- 要获取当前亮度级别，请输入：

  ```
  target:~$ cat /sys/class/backlight/backlight/brightness
  ```

- 写入文件 brightness 以更改亮度：

  ```
  target:~$ echo 0 > /sys/class/backlight/backlight/brightness
  ```

  例如，关闭背光。

  有关所有文件的文档，请参见 https://www.kernel.org/doc/Documentation/ABI/stable/sysfs-class-backlight。

Device tree description of LVDS-0 can be found here: https://github.com/phytec/linux-phytec-imx/tree/v6.12.49-2.2.0-phy/arch/arm64/boot/dts/freescale/imx8mp-phyflex-libra-rdk.dts#L253

## 7.16 GPU

The i.MX 8M Plus has a Vivante GC7000UL GPU.

The phytec-qt6demo-image has glmark2 as a GPU benchmark built in. To run the benchmark, execute:

```
target:~$ glmark2-es2-wayland
```

> **备注**
>
> For this benchmark to run properly, weston needs to be started successfully. For this display overlays might need to be loaded.

By default the benchmark is executed in 800x600 windowed mode. The benchmark can also be executed in fullscreen mode with the `--fullscreen` flag. Alternatively with the `--off-screen` flag the benchmark will not be shown on display.

## 7.17 电源管理

### 7.17.1 CPU 核心频率调节

i.MX 8M Plus SoC 中的 CPU 能够调整时钟频率和电压。这用于在不需要 CPU 的全部性能时节省电力。调整频率和电压被称为“动态电压和频率调整”（DVFS）。i.MX 8M Plus BSP 支持 DVFS 功能。Linux 内核提供了一个 DVFS 框架，允许每个 CPU 核心设置最小或最大频率和一个管理其运行的 governor。根据使用的 i.MX 8 型号，支持几种不同的频率。

> **小技巧**
>
> 尽管 DVFS 框架为每个 CPU 核心提供了频率设置，但一个 CPU 核心的频率更改会影响其他 CPU 核心。因此，所有 CPU 核心始终共享相同的 DVFS 设置。每个核心的单独 DVFS 设置是不可能的。

- 要获取完整列表，请输入：

  ```
  target:~$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
  ```

  例如 i.MX 8MPlus CPU，最高可达约 1.6 GHz，则结果将是：

```
1200000 1600000
```

- 要查询当前的频率输入：

```
target:~$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
```

governor 会根据它们的目标自动选择这些频率中的一个。

- 列出所有可用的 governor，使用以下命令：

```
target:~$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_governors
```

结果是：

```
conservative ondemand userspace powersave performance schedutil
```

- **conservative** governor 与 ondemand governor 非常相似。只是它的行为有所不同，它会更保守地增减 CPU 速度，而不是在 CPU 有任何负载的瞬间就跳到最大速度。
- **ondemand** （默认）根据当前系统负载在可能的 CPU 核心频率之间切换。当系统负载超过特定值时，它会立即提高 CPU 核心频率。
- **powersave** 始终选择最低的 CPU 核心频率。
- **performance** 始终选择最高的 CPU 核心频率。
- **userspace** 允许以 root 身份运行的用户或用户空间程序设置特定频率（例如，设置为 1600000）。输入：
- 要查询当前的 governor，请输入：

```
target:~$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

您通常会得到：

```
ondemand
```

- 切换到另一个 governor（例如，userspace）可以通过以下方式完成：

```
target:~$ echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

- 现在你可以设置频率：

```
target:~$ echo 1600000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

有关 governor 的更详细信息，请参阅 Linux 内核代码库中的 Linux 内核文档，路径为 `Documentation/admin-guide/pm/cpufreq.rst` 。

### 7.17.2 CPU 核心管理

该 i.MX 8M Plus SoC 芯片上可以有多个处理器核心。例如，该 i.MX 8M Plus 具有 4 个 ARM 核心，可以在运行时单独开启和关闭。

- 要查看系统中所有可用的核心，请执行：

```
target:~$ ls /sys/devices/system/cpu  -1
```

- 这将显示，例如：

```
cpu0    cpu1   cpu2   cpu3   cpufreq
[...]
```

这里系统有四个处理器核心。默认情况下，系统中所有可用的核心都被启用，以获得最佳性能。

- 要关闭某个核，请执行：

```
target:~$ echo 0 > /sys/devices/system/cpu/cpu3/online
```

作为确认，您将看到：

```
[  110.505012] psci: CPU3 killed
```

现在核心已关闭电源，并且该核心上不再安排任何进程。

- 您可以使用 top 命令查看核心和进程的图形概览：

```
target:~$ htop
```

- 要重新启用核心，请执行：

```
target:~$ echo 1 > /sys/devices/system/cpu/cpu3/online
```

## 7.18 热管理

### 7.18.1 U-Boot

之前 U-Boot 中的温度控制不够理想。现在，U-Boot 增加了温度关机功能，以防止在启动过程中板子过热。关机发生的温度与内核中的温度一致。

当前温度的各个温度范围在启动日志中显示：

```
CPU:   Industrial temperature grade (-40C to 105C) at 33C
```

### 7.18.2 内核

Linux 内核集成了热管理功能，能够监测芯片（SoC）温度，降低 CPU 频率，控制风扇，通知其他驱动程序减少功耗，并在最坏的情况下关闭系统（https://www.kernel.org/doc/Documentation/thermal/sysfs-api.txt）。

本节描述了如何在 i.MX 8M Plus SoC 平台上使用热管理内核 API。i.MX 8 具有用于 SoC 的内部温度传感器。

- 当前温度可以以毫摄氏度为单位读取：

```
target:~$ cat /sys/class/thermal/thermal_zone0/temp
```

- 例如，你将得到：

```
49000
```

imx_thermal 内核驱动注册了两个触发点。这些触发点根据 CPU 型号的不同而有所不同。主要区分工业版和商业版。

| | 商业 | 工业 |
|---|---|---|
| 被动（警告） | 85°C | 95°C |
| 严重（关机） | 90°C | 100°C |

（请查看内核 sysfs 文件夹 `/sys/class/thermal/thermal_zone0/`）

内核热管理使用这些触发点来触发事件并实施冷却措施。内核中可用的热政策（也称为 thermal governor）包括：Step Wise、Fair Share、Bang Bang 和 Userspace。BSP 中使用的默认策略是 Step Wise。如果 sysfs 文件 temp 中的 SoC 温度值高于 *trip_point_0*，则 CPU 频率将设置为最低 CPU 频率。当 SoC 温度降到 *trip_point_0* 以下时，限制将被解除。

**备注**

由于我们安装了不同温度等级的 CPU，因此热触发点的实际值可能会有所不同。

## 7.19 看门狗

PHYTEC i.MX 8M Plus 模块包含一个硬件看门狗，当系统挂起时能够重置开发板。看门狗在 U-Boot 中默认启动，超时时间为 60 秒。因此，即使在早期内核启动过程中，看门狗也已经开始运行。Linux 内核驱动程序控制看门狗，并确保它有被踢到。本节将解释如何使用 systemd 在 Linux 中配置看门狗，以避免系统挂起和重启期间的情况。

### 7.19.1 Systemd 中的看门狗支持

Systemd 从版本 183 开始支持硬件看门狗。

- 要启用看门狗支持，需要通过启用选项来配置 `/etc/systemd/` 中的文件 system.conf 文件：

```
RuntimeWatchdogSec=60s
RebootWatchdogSec=10min
```

*RuntimeWatchdogSec* defines the timeout value of the watchdog, while *RebootWatchdogSec* defines the timeout when the system is rebooted. For more detailed information about hardware watchdogs under systemd can be found at http://0pointer.de/blog/projects/watchdog.html. The changes will take effect after a reboot or run:

```
target:~$ systemctl daemon-reload
```

## 7.20 Power Key

The X_ONOFF pin connected to the ON/OFF button can be pressed long (for 5 seconds) to trigger Power OFF without SW intervention or used to wake up the system out of suspend. With the snvs_pwrkey driver, the KEY_POWER event is also reported to userspace when the button is pressed. On default, systemd is configured to ignore such events. The function of Power OFF without SW intervention is not configured. Triggering a power off with systemd when pushing the ON/OFF button can be configured under `/usr/lib/systemd/logind.conf.d/00-systemd-conf-imx.conf` and set using:

```
HandlePowerKey=poweroff
```

To use the changes, reboot or restart the logind service.

```
systemctl restart systemd-logind
```

By default, ON/OFF button is configured to emit the KEY_POWER event code when short-pressed. This behavior can be verified using the `evtest` tool, which is included in our BSP:

```
target:~$ evtest /dev/input/by-path/platform-30370000.snvs:snvs-powerkey-event
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x0 product 0x0 version 0x0
Input device name: "30370000.snvs:snvs-powerkey"
Supported events:
Event type 0 (EV_SYN)
Event type 1 (EV_KEY)
   Event code 116 (KEY_POWER)
Properties:
Testing ... (interrupt to exit)
Event: time 1762862749.414004, type 1 (EV_KEY), code 116 (KEY_POWER), value 1
Event: time 1762862749.414004, -------------- SYN_REPORT ------------
Event: time 1762862749.542006, type 1 (EV_KEY), code 116 (KEY_POWER), value 0
Event: time 1762862749.542006, -------------- SYN_REPORT ------------
```

However, it is possible to reassign the short-press event by modifying the device-tree and changing the default linux,keycode property of the snvs_pwrkey node:

```
&snvs_pwrkey {
   linux,keycode = <KEY_SLEEP>;
};
```

In this example, the ON/OFF button is assigned KEY_SLEEP code, and pressing the button will now put the board into sleep mode by default.

All valid key codes are defined in the `include/uapi/linux/input-event-codes.h` file part of the Linux kernel.

## 7.21 ISP

i.MX 8M Plus SoC 包含一个图像信号处理器（ISP）。有关更多信息，请参阅 |sbc| i.MX 8M Plus 文档中的使用 ISP 部分。

## 7.22 片上一次性可编程控制器 (OCOTP_CTRL) - eFuse

该 i.MX 8M Plus 提供一次性可编程 fuse，用于存储信息，例如 MAC 地址、启动配置和其他永久设置（在 i.MX 8M Plus reference manual 中称为 “片上 OTP 控制器 (OCOTP_CTRL)”）。以下列表摘自 i.MX 8M Plus reference manual，包括 OCOTP_CTRL 中的一些有用寄存器（基地址为 0x30350000）：

| 名称 | Bank | 字 | 内存偏移量为 0x30350000 | 描述 |
|---|---|---|---|---|
| OCOTP_MAC_AI | 9 | 0 | 0x640 | 包含 ENET0 MAC 地址的低 32 位 |
| OCOTP_MAC_AI | 9 | 1 | 0x650 | 包含 ENET0 MAC 地址的高 16 位和 ENET1 MAC 地址的低 16 位 |
| OCOTP_MAC_AI | 9 | 2 | 0x660 | 包含 ENET1 MAC 地址的高 32 位 |

关于 OCOTP_CTRL 中的 fuse 与启动配置之间的完整列表和详细映射，请参阅 i.MX 8M Plus Security Reference Manual 中的”Fuse Map” 部分。

### 7.22.1 在 uBoot 中读取 fuse 的值

您可以使用内存映射的 shadow 寄存器读取 fuse 寄存器。要计算内存地址，请使用以下公式计算：

OCOTP_MAC_ADDR:

```
u-boot=> fuse read 9 0
```

### 7.22.2 在 Linux 中读取 fuse 值

要访问 Linux 中的 fuse 内容，NXP 提供了 NVMEM_IMX_OCOTP 模块。所有内存映射的 shadow 寄存器的 fuse 内容可以通过 sysfs 访问：

```
target:~$ hexdump /sys/devices/platform/soc@0/30000000.bus/30350000.efuse/imx-ocotp0/nvmem
```