Geniatech

Spec V1.0

Geniatech NXP Series

Linux Software Development Guide

Overview

This document serves as a guide for the Geniatech nxp series Linux software development, and is intended to help software development engineers and technical support engineers get started with the Geniatech NXP platform Linux development and debugging faster. The Geniatech NXP series product development guide will be continuously updated on this document.

Product Versions

Product Name	Kernel Version	Yocto Versions
xpi-imx8mmevk	Linux 4.14	Sumo

Readership

This document (this guide) is intended for engineers who are: Software Development Engineers

Revision Record

Date	Versions	Author	Audit	Modification instructions
2022-08-06	V1.00	lh	cf	1. Initializing the Geniatech MXP linux development guide architecture 2. Support xpi-imx8mmevk

Catalog

1. Support List

1.1 Product List

Platform	Product Model	Software Documentation Support	Function Description
NXP	imx8mmevk	Support	
NXP	imx8mqevk	Support	

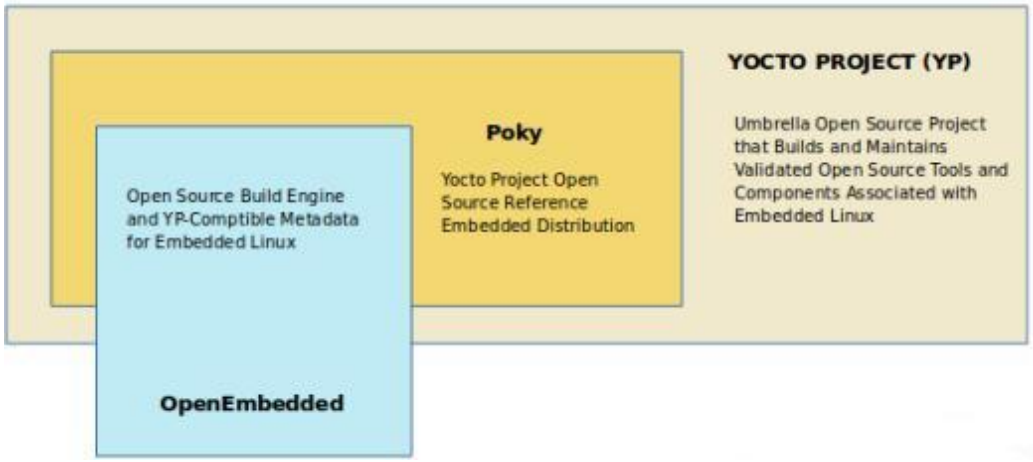
1.2 Documentation Development Support List

Products	Hardware Board Type	Function Description
xpi-imx8mmevk	Development Boards	hdmi out, ent, usbx4, wifi/bt, lte, rtc, pin out 40pin Lineup

2. SDK Software Architecture

2.1 SDK Overview

The NXP Linux SDK is a Linux development BSP based on Yocto, an open source collaborative project that helps developers make custom Linux-based systems for embedded products with different hardware architectures. The Yocto Project provides a flexible toolset and development environment that allows embedded device developers around the world to collaborate by sharing technologies, software stacks, configurations, and best practices for creating these custom Linux images.



Yocto has the following advantages.

1. Widespread adoption: Many semiconductor, operating system, software, and service providers have adopted and support the Yocto Project in their products and services, and there is also a large community of support, so check out the Yocto Project community and the companies involved in the Yocto Project

2. Multiple architectures supported: The Yocto project supports Intel, ARM, MIPS, AMD, PPC and other architectures. Most ODMs, OSVs and chip vendors create and provide BSPs that support their hardware, and if you want to add a custom chip, you can create a BSP that supports that architecture. in addition to the extensive architectural support, the Yocto project also fully supports various device emulations through Quick EMUlator (QEMU).

3. For resource-constrained embedded IoT devices: Unlike full Linux distributions, yocto lets you customize your image, so you can decide which features or modules to put into your image, for example, many devices don't have display screens, so components like X11, GTK+, Qt or SDL can be left out. The final image will be small enough and will have no extra features.

2.2 SDK Software Framework

SDK software framework, from bottom to top, is divided into four layers: bootloader, Linux Kernel, i.MX release layer, Yocto Project community layers.

The content of each level is as follows.

- The bootloader layer provides the underlying system support packages such as
- Bootloader, U-Boot, ATF related support. The Kernel layer provides the standard implementation of the Linux Kernel, which is also an open operating system. The SDK's

The Linux kernel is the standard Linux 4.14 kernel, which provides basic support for security, memory management, process management, network stack, etc. The main purpose of the Linux kernel is to manage device hardware resources such as CPU scheduling, cache, memory, I/O, etc.

- The i.MX release layer includes meta-freescale, poky, meta-openembedded layers and meta- freescale-distros.
- Yocto Project community layers include support for base and i.MX Arm reference boards, support for third party and partner boards, freescale distribution, basic configuration for FSL Community BSP, base components for POKY, multiple browsers and QT5 related applications.

components.

2.3 SDK development process

Geniatech released the NXP platform SDK development kit for a variety of different product forms developed SDK. Based on this SDK, system customization and application porting development can be effectively implemented.

system development environment and compiled code. The following will briefly describe the process.

1) Check the system requirements: Before downloading the code and compiling, make sure the local development equipment can meet the requirements, including the machine's hardware capacity, software system, tool chain, etc. Currently, the SDK supports compilation on Linux operating system and provides toolchain support on Linux environment.

2) Build the compilation environment: Introduce the various packages and tools that need to be installed on the development machine

3) Selecting the device: During the development process, developers are required to select the corresponding hardware board type according to their needs

4) Download source code: After selecting the device type, you need to install the repo tool to download the source code in bulk.

5) System customization : Developers can customize U-Boot, Kernel, Rootfs according to the hardware board and product definition used.

6) Compile and package: After introducing the source code, select the product and initialize the related compile environment, then execute the compile command, including the whole or module compile and compile cleanup, etc.

7) Burn and run.

3. Development Environment Setup

3.1 Overview

This section describes how to build a local build environment to compile the NXP Yocto Linux SDK source code. Currently the SDK only supports compilation in Linux environment and provides a cross-compilation toolchain for Linux.

A typical embedded development environment usually includes a Linux server, a Windows PC, and a target hardware version. imx8mmevk is an example of a typical development environment as shown in Figure 3- 1.

- Build cross-compiling environment on Linux server, provide code update download
- and code cross-compiling service for software development. Windows PC and Linux server share the program, and install SecureCRT or puTTY, and log in remotely via

network to

Linux server for cross-compiling and code development and debugging.

- The Windows PC is connected to the target hardware board via serial port and USB, and the compiled image file can be burned to the target hardware board and debug the system or application.

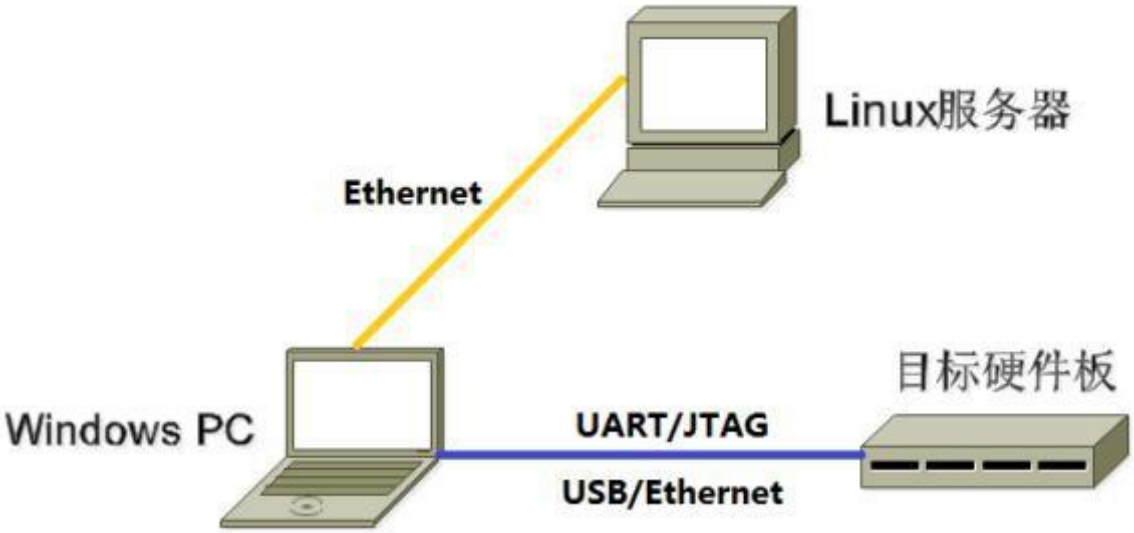


Figure 3-1

Note: Windows PC is used in the development environment, in fact, many tasks can be done on Linux PC, such as using minicom instead of SecureCRT or puTTY, etc., which users can choose by themselves.

3.2 Linux server development environment construction

The NXP Yocto Linux SDK was developed and tested on Ubuntu 16.04. Therefore, we recommend compiling with Ubuntu 16.04. Other versions have not been tested specifically and may require the package to be adjusted accordingly.

In addition to the system requirements, there are other hard and soft requirements.

- Hardware requirements: 64-bit system, more than 50G of hard disk space, we recommend more than 120G. If you do multiple builds, you will need more hard disk space.
- Package dependencies: In addition to python 2.7, make 3.8, and git 1.7, you will need to install some additional packages, which are listed in the package installation section.

3.2.1 The release package uses the Linux server system version

The SDK development environment is installed with the following version of Linux, and the SDK is compiled with this Linux system by default: Ubuntu 16.04 LTS

3.2.2 Dependency Package Installation

After the operating system is installed and the user has configured the network environment, you can continue with the following steps to complete the installation of the relevant software packages.

Platform	Installing dependencies
imx8mmevk	1. Installing dependency packages sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat libsdl1.2-dev 2. Ubuntu 12.04 and 14.04 dependency packages

	sudo apt-get install libsdl1.2-dev xterm sed cvs subversion coreutils texi2html \ docbook-utils python-pysqlite2 help2man make gcc g++ desktop-file-utils \ libgl1-mesa-dev libglu1-mesa-dev mercurial autoconf automake groff curl lzop asciidoc \ 3. Ubuntu 12.04 dependent packages sudo apt-get install uboot-mkimage 4. Ubuntu 14.04 dependent packages sudo apt-get install u-boot-tools If you encounter an error in the compilation, you can install the corresponding package according to the error message.

4. SDK installation preparation

4.1 SDK Get

The SDK is distributed through Geniatech's external server, and customers need to apply for the SDK from our company accordingly.

4.1.1 SDK download link

Platform	Download Links
xpi-imx8mmevk	http://www.geniatech.net/hefei/sd-release/imx8mm_xpi_yocto-kernel-uboot-source-20210909/imx8mm_xpi_yocto-kernel-uboot-source-20210909.zip

4.1.2 SDK code zip

To facilitate quick access to the sdk source code, geniatech usually provides an init zip file for the corresponding hardware. Developers can do this by using the following methods.

Take xpi-imx8mmevk as an example:

unzip imx8mm_xpi_yocto-kernel-uboot-source-20210909.zip

Note: Please get the unzip password from the person in charge.

cd imx8mm_xpi_yocto-kernel-uboot-source-20210909

Product Platform	Software zip package	Versions
xpi-imx8mmevk	imx8mm_xpi_yocto-kernel-uboot-source-20210909.zip	V1.00

4.2 SDK directory structure

After downloading the SDK, you can see the following directory structure in the root directory: xpi-imx8mmevk for example.

```
├── fsl-setup-af.sh
├──── fsl-setup-release.sh
├── geniatech
├── loong
├──── lunch.sh
├──── overlay
├──── README
├──── README-IMXBSP
├──── setup-environment
└──── source
```

- fsl-setup-af.sh adds the environment configuration for AF meta-layer
- fsl-setup-release.sh yocto environment configuration, you need to run this script before compiling
- The geniatech directory holds the uboot, kernel source code
- The loong directory holds the project customization files and package compilation scripts
- lunch.sh soft link, run before compiling to select the project to be compiled
- overlay project overaly
- README for Freescale Yocto BSP readme
- README-IMXBSP for BSP 4.14.98_2.0.0 Release readme
- setup-environment environment configuration script, which is called before compilation
- The source directory holds all application recipes

4.3 SDK update and issue feedback

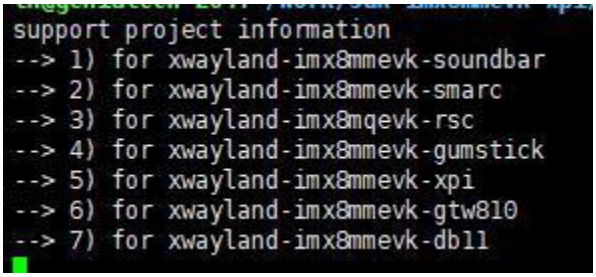
Contact geniatech

5. SDK Compilation

5.1 Select Project

- xpi-imx8mmevk

Executing source lunch.sh will list the currently supported projects as follows.



then select the appropriate item and enter serial number 5 for environment initialization.

5.2 Uboot compilation

- xpi-imx8mmevk

. /build.sh -i uboot

After compiling, the build-xwayland-imx8mmevk-xpi/tmp/deploy/images/imx8mmevk directory will generate an image file such as u-boot.bin.

5.3 Kernel Compilation

- xpi-imx8mmevk

. /build.sh -i bootimg

After compiling, the build-xwayland-imx8mmevk-xpi/tmp/deploy/images/imx8mmevk directory will generate boot.img and dtb image files.

5.4 Rootfs Compilation

- **xpi-imx8mmevk**

. /build.sh -i rootfs

After compiling, the build-xwayland-imx8mmevk-xpi/tmp/deploy/images/imx8mmevk directory will generate the fsl-image- qt5-validation-imx-imx8mmevk.tar.bz2 rootfs archive and the fsl-image-qt5-validation-imx- imx8mmevk.sdcard.bz burned archive

5.5 Firmware packaging

- **xpi-imx8mmevk**

. /build.sh -i pack

After compiling, the firmware package for burning will be generated in the loong/out/nxp-imx8/yocto/nxp-imx8_yocto_xwayland-imx8mmevk-xpi directory

5.6 Application part compilation

The compilation of individual application recipes can be done with the bitbake command. For details on how to use it, please refer to the yocto official Net.

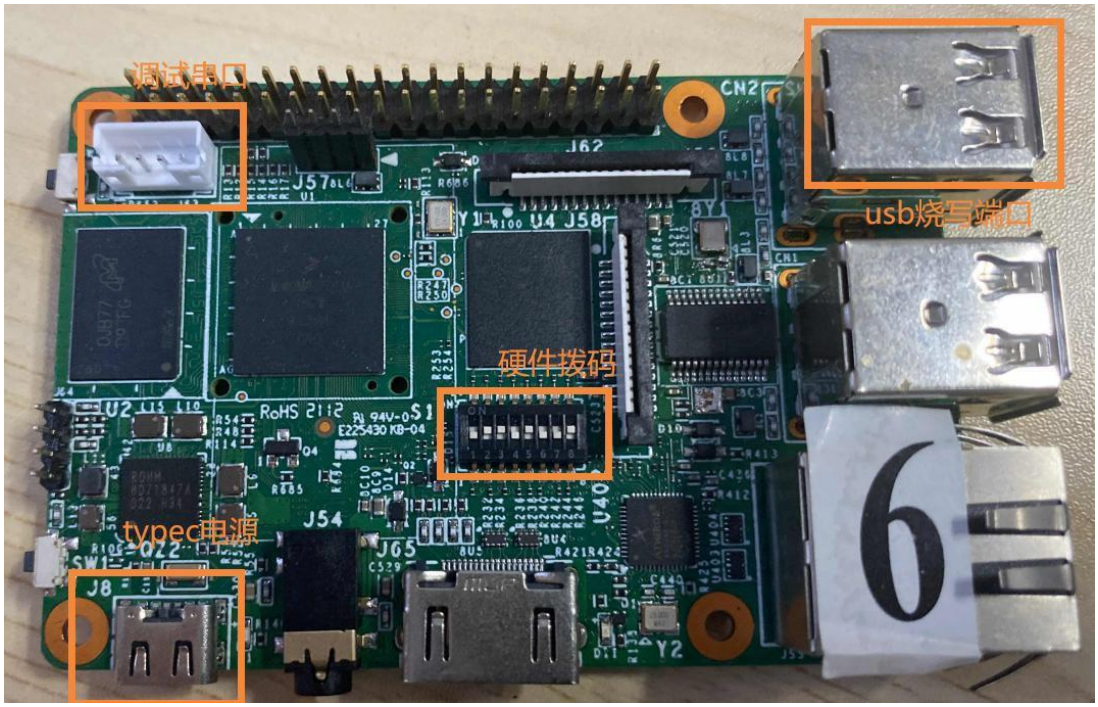
6. SDK image burning

6.1 Overview

This section introduces how to burn and run the finished image file on hardware device. NXP provides uuuu burning tool to finish image burning.

Tool s	Running System	Desc ript ion
uuu	Windows	NXP whole package burning firmware tool

6.2 Development Board Introduction



6.3 Burn mode switching

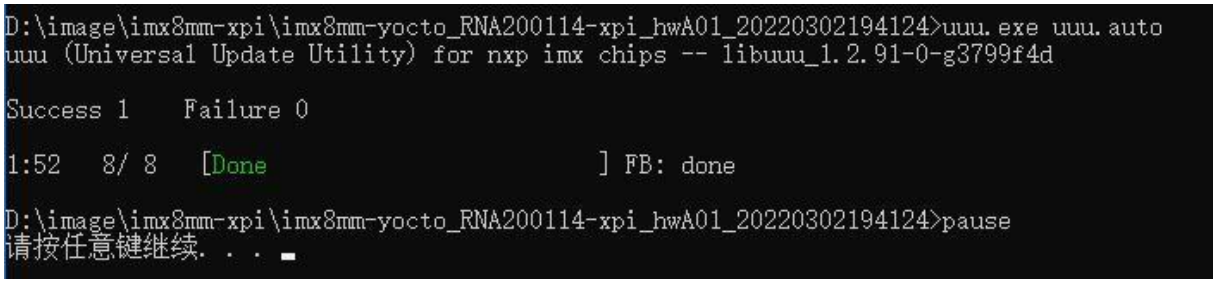
Mode	Dipswitch
Burn mode	10101001
Start-up mode	01101001

6.4 Introduction to burn-in packages

```
├── flash.bat
├── fsl-image-qt5-validation-imx-imx8mmevk.sdcard
├── imx-boot-imx8mmevk-sd.bin-flash_evk
├── uuu.auto
└── uuu.exe
```

- flash.bat burn batch process
- fsl-image-qt5-validation-imx-imx8mmevk.sdcard Burn the whole package
- imx-boot-imx8mmevk-sd.bin-flash_evk uboot image
- uuu.auto uuu burn script
- uuu.exe uuu windows program

Switch the board to burn mode, connect the computer and the USB burn port of the board via the USB cable with dual USB A connectors, double click flash.bat and power on the board for burning, the prompt will be as follows when the burning is completed.



Switch to boot mode to re-power the system after burn-in is complete.

7. Question Feedback

7.1 Feedback channels

Please contact the person in charge and report it to R&D.