AN14092

为i.MX 8M Linux用户空间Cortex-A开发设置VS Code

第1版 — 2023年10月3日 **应用笔记**

文档信息

信息	内容
关键词	Visual Studio Code、调试、Real-Time-Edge、i.MX 8M Cortex-A用户空间
摘要	本文档介绍了如何为在运行Real-Time-Edge软件的i.MX电路板上开发和调试用户空间Cortex-A应用设置Visual Studio Code。

1 介绍

本应用笔记介绍了如何在Windows主机上使用WSL (Windows Subsystem for Linux)，为在运行Real-Time Edge软件的i.MX 8MP上开发和调试用户空间Cortex-A应用设置Visual Studio Code。虽然本文档使用i.MX 8MP，但整个i.MX 8M系列都适用于此流程，只需根据具体型号做出细微的修改即可。

1.1 步骤概述

假设主机已安装了最新版本的 Windows 10/11。主要步骤如下：

- 在Windows主机上设置WSL1。
- 安装开发工具链。
- 安装和配置VS Code。
- 配置电路板，在i.MX 8MP上测试应用调试。

1.2 软件环境

- 下载i.MX 8MP电路板的Real Time Edge软件2.6.0版文件并解压缩。
- 解压缩`Real-time_Edge_v2.6_IMX8MP-LPDDR4-EVK/real-time-edge/nxp-image-real-time-edge-imx8mp-lpddr4-evk.wic.zs`文件，获得可烧录的镜像。
- 使用SD卡烧录软件，如Balena Etcher，将`nxp-image-real-time-edge-imx8mp-lpddr4-evk.wic`镜像写入SD卡。

1.3 硬件设置与设备

- 开发套件：恩智浦i.MX 8MP EVK LPDDR4
- Micro SD卡：当前实验使用了SanDisk Ultra 32-GB Micro SDHC I Class 10
- 用于调试端口的USB-C电缆
- 以太网电缆

2 前提条件

通过DEBUG USB-UART连接器和PC USB连接器之间的USB电缆将i.MX 8MP平台连接到Windows主机。Windows会自动查找串行设备。

查找名为COM*的COM设备，以确定调试端口。在i.MX 8MP上，会出现4个端口。在后两个端口中，一个端口用于Cortex-A53的调试消息，另一个用于Cortex-M7。端口号是随机分配的，因此打开两个端口有利于开发。

设备管理器显示i.MX 8MP的4个COM端口，如图1所示。这里使用的后两个端口以红色突出显示。

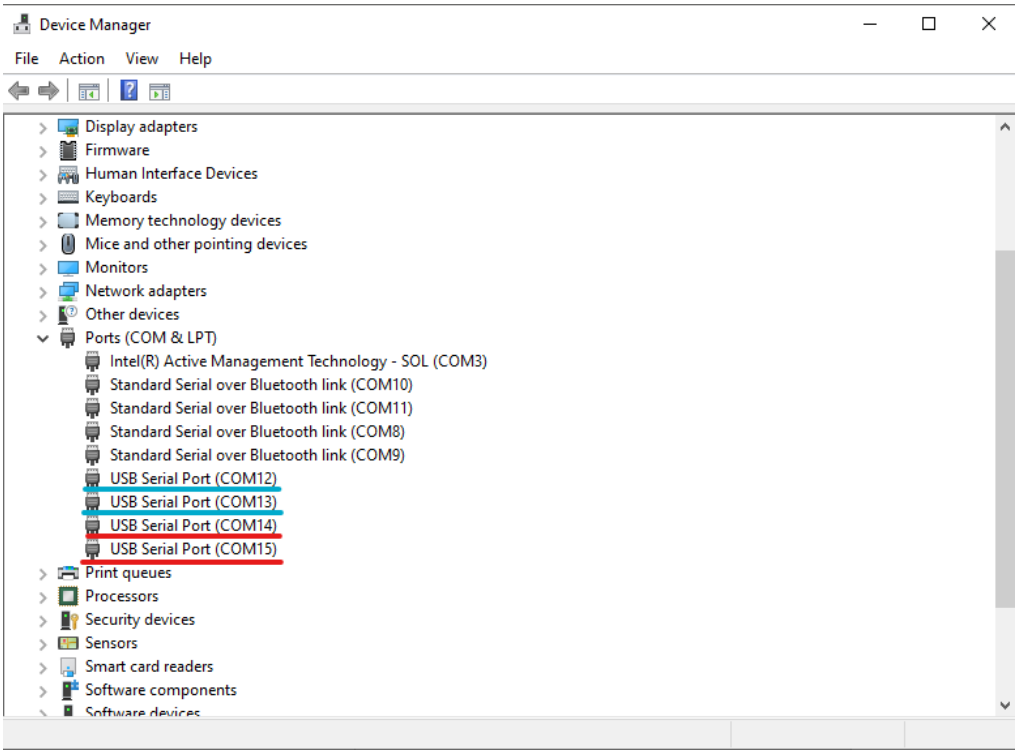

图1. 设备管理器

在首选的串行终端仿真器（例如，PuTTY）中打开串行设备，将速度设置为 115200比特/秒，数据位为8，停止位为1（115200, 8N1），无奇偶校验位。

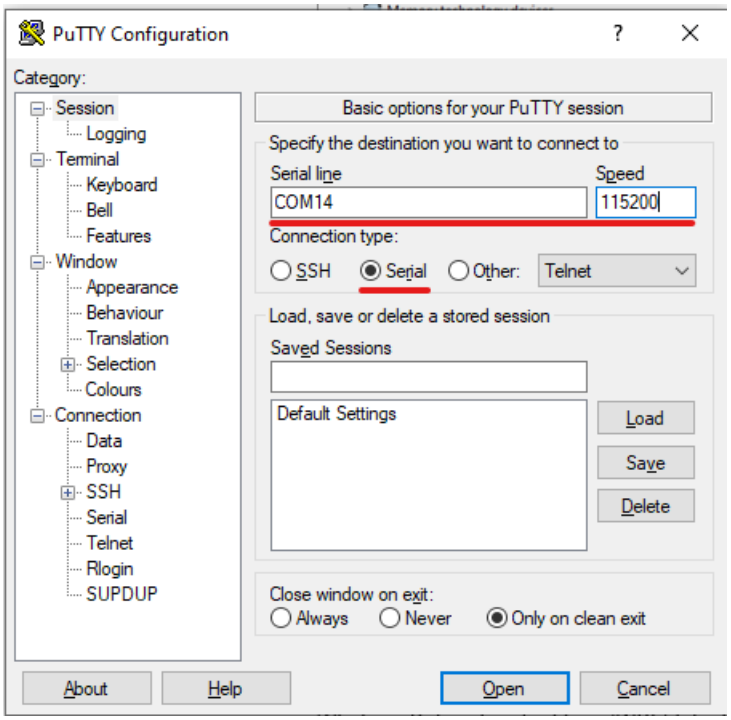

图2. 用于打开电路板上的COM14端口的PuTTY配置

3 设置WSL

Windows Subsystem for Linux (WSL)允许开发人员在Windows下安装和运行Linux发行版。

3.1 安装WSL

1. 在Windows主机上，打开Windows功能，然后打开Windows Subsystem for Linux功能。

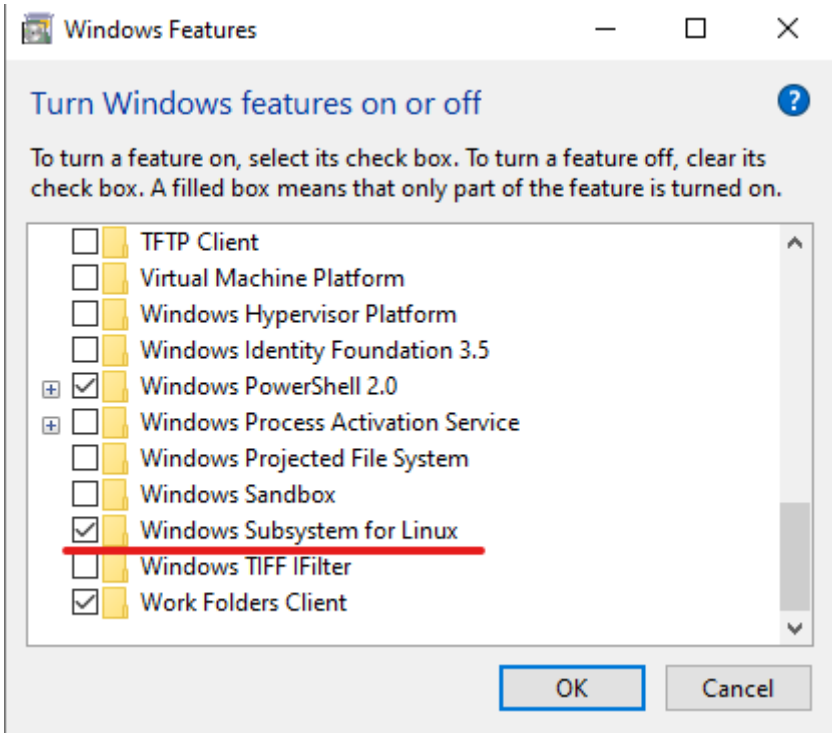

图3. Windows的功能，WSL已启用

2. 打开CMD窗口，将默认WSL版本设置为1：

```
> wsl --set-default-version 1
```

3. 从微软商店搜索并下载Ubuntu发行版。
4. 安装后，打开它。出现一个终端窗口，要求您设置用户名和密码。设置完成后，bash shell启动。

3.2 安装依赖项

在打开的shell中，运行以下命令来安装依赖项：

```
$ sudo apt-get -y update
$ sudo apt-get -y install build-essential gdb gdb-multiarch git
```

3.3 安装开发工具链

要交叉编译应用并访问电路板/软件特定的库，需要一个开发工具链。可以从Real-Time-Edge Yocto工程中获得。该工程可以在WSL或任何其他Ubuntu主机上设置。

- 安装Real Time Edge软件2.6.0环境。

- 为i.MX 8MP设置Real-time Edge Yocto环境。有关如何操作的更多详细信息，请参阅《Real-Time Edge Yocto工程用户指南》（文档REALTIMEEDGEUG）第3节（依赖项）和第5.5节（构建指南），而不运行最终的bitbake命令。运行以下程序来构建SDK，而不是可烧录的镜像：

```
$ bitbake nxp-image-real-time-edge -c populate_sdk
```

该任务生成一个可用于安装SDK的shell脚本。它位于<yocto_build_directory>/tmp/deploy/sdk/目录。

- 在WSL中运行上面生成的脚本，以安装SDK：

```
$ ./nxp-real-time-edge-glibc-x86_64-nxp-image-real-time-edge
-armv8a-imx8mp-lpddr4-evk-toolchain-2.5.sh
```

注：*脚本会询问是否要更改/opt/nxp-real-time-edge/2.5/中的默认安装位置。对于本指南，假定使用默认位置。*

4 设置VS Code

本节介绍如何设置VS Code的详细信息。

4.1 安装VS Code

在Windows上，安装从官方网站下载的VS Code。

4.2 安装所需的扩展

打开VS Code，转到左侧边栏的扩展选项卡，安装以下扩展：

- C/C++是官方C/C++开发扩展。
- WSL用于将VS Code连接到WSL。
- 串行监测器用于直接从VS Code连接到电路板上的串行端口。

注：*此步骤为可选*

4.3 创建/打开工程

要在连接到WSL的VS Code中创建或打开工程，请从WSL shell转到工程目录，然后输入以下命令：

```
$ code .
```

5 配置用于开发和调试的工程

在本例中，使用了一个用于hello-world问题的新工程。然后在电路板上开发和调试所需的配置。本应用笔记中的基本配置可以轻松转移到其他工程。

由于使用了ssh和gdbserver，这种方法要求电路板和主机通过网络连接。

5.1 创建工程

运行以下命令以创建和打开工程：

```
$ mkdir demo-proj
$ cd demo-proj
$ code .
```

5.2 创建源文件

- 在工程中，使用一个简单的程序创建一个`hello-world.c`文件：

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    printf("Hello, World!\n");

    return 0;
}
```

- 创建一个包含以下内容的简单Makefile：

```
CC ?= gcc

CFLAGS ?= -Wall -Wextra

DEBUGFLAGS = -Og -g

TARGET = hello.bin
SOURCE = hello-world

OBJECTS = $(TARGET).o

.PHONY: all clean

all: $(TARGET)

$(TARGET): $(OBJECTS)
 $(CC) $(CFLAGS) $(DEBUGFLAGS) -o $@ $^

$(OBJECTS): $(SOURCE).c
 $(CC) $(CFLAGS) $(DEBUGFLAGS) -c -o $@ $^
```

Makefile使用环境定义编译器（CC）和CFLAGS将`hello-world.c`可执行文件构建为名为`hello.bin`的二进制文件，如果未设置，则返回默认值。

5.3 VS Code工程配置

`demo-proj`工程文件夹中的`.vscode`文件夹包含工程配置文件。如果文件夹并非自动创建，请手动创建：

```
$ mkdir .vscode
```

然后，创建/编辑以下文件：

- settings.json，该文件包含用于配置其他文件中的工作区的变量的值。

  内容：

```
{
 /* Target Device Settings */
 "TARGET_IP":"169.254.158.177",

 /* Project Settings */
 "PROGRAM":"hello.bin",

 /* SDK Configuration */
 "ARCH":"aarch64-poky-linux",
 "OECORE_NATIVE_SYSROOT":"/opt/nxp-real-time-edge/2.5/sysroots/x86_64-pokysdk-
linux",
 "SDKTARGETSYSROOT": "/opt/nxp-real-time-edge/2.5/sysroots/armv8a-poky-linux",

 /* SDK Constants */
 "CC_PREFIX": "${config:OECORE_NATIVE_SYSROOT}/usr/bin/${config:ARCH}/
${config:ARCH}-",
 "CXX": "${config:CC_PREFIX}g++ -march=armv8-a+crc+crypto -fstack-protector-
strong  -O2 -D_FORTIFY_SOURCE=2 -Wformat -Wformat-security -Werror=format-
security --sysroot=${config:SDKTARGETSYSROOT}",
 "CC": "${config:CC_PREFIX}gcc -march=armv8-a+crc+crypto -fstack-protector-
strong  -O2 -D_FORTIFY_SOURCE=2 -Wformat -Wformat-security -Werror=format-
security --sysroot=${config:SDKTARGETSYSROOT}",
 "CPP": "${config:CC_PREFIX}gcc -E -march=armv8-a+crc+crypto -fstack-protector-
strong  -O2 -D_FORTIFY_SOURCE=2 -Wformat -Wformat-security -Werror=format-
security --sysroot=${config:SDKTARGETSYSROOT}",
}
```

  其中：

  - TARGET_IP：要调试的i.MX电路板的IP。
  - PROGRAM：已编译的可执行文件名称。
  - ARCH：要编译的架构。
  - OECORE_NATIVE_SYSROOT：本机系统根的位置。
  - SDKTARGETSYSROOT：目标SDK系统根的位置。
  - CC_PREFIX：交叉编译器二进制文件的路径前缀。
  - CXX/CC/CPP：交叉编译器二进制文件的完整路径，包含SDK环境设置脚本设置的默认标志（/opt/nxp-real-time-edge/2.5/environment-setup-armv8a-poky-linux）。

  它们可以根据需要进行修改或删除，但交叉编译所需的sysroot参数除外。

- c_cpp_properties.json描述了c/c++扩展配置。以下是IntelliSense查找头文件和编译器路径的IncludePath的设置。

内容：

```
{
    "configurations": [
        {
            "name": "Linux",
            "includePath": [
                "${workspaceFolder}/**",
                "${config:SDKTARGETSYSROOT}/usr/include/**"
            ],
            "compilerPath": "${config:CC_PREFIX}gcc",
            "intelliSenseMode": "linux-gcc-arm64",
```

```
            "browse": {
                "path": [
                    "${workspaceFolder}/**",
                    "${config:SDKTARGETSYSROOT}/usr/include/**"
                ],
                "limitSymbolsToIncludedHeaders": true
            }
        }
    ],
    "version": 4
}
```

- `tasks.json`用于覆盖或添加新任务。它在执行VS Code构建命令时运行Makefile，并定义调试任务。

  内容：

```
{
    "version": "2.0.0",
    /* Configure Yocto SDK Constants from settings.json */
    "options": {
        "env": {
            "CXX": "${config:CXX}",
            "CC": "${config:CC}",
            "CPP": "${config:CPP}"
        }
     },
     /* Configure integrated VS Code Terminal */
     "presentation": {
        "echo": false,
        "reveal": "always",
        "focus": true,
        "panel": "dedicated",
        "showReuseMessage": true,
    },
    "tasks": [
        /* Configure launch.json (debug) preLaunchTask Task */
        {
            "label": "imx-deploy-gdb",
            "isBackground": true,
            "problemMatcher":{
                "base": "$gcc",
                "background": {
                    "activeOnStart": true,
                    "beginsPattern":"Deploying to target",
                    "endsPattern":  "Starting GDB Server on Target"
                }
            },
            "type": "shell",
            "command": "sh",
            "args": [
                "imx-deploy-gdb.sh",
                "${config:TARGET_IP}",
                "${config:PROGRAM}"
            ],
            "dependsOn": ["build"],
        },
        /* Configure Build Task */
        {
            "label": "build",
            "type": "shell",
```

```
            "command": "make clean; make -j$(nproc)",
            "problemMatcher": ["$gcc"]
        },
    ]
}
```

其中：

- options.env：将要编译的环境变量设置为在settings.json中设置的环境变量。
- tasks[0]：定义调试任务。首先运行build任务，然后运行第5.4节中描述的imx-deploy-gdb.sh脚本，连接到电路板并启动调试会话。
- tasks[1]：定义构建任务。该示例使用一个简单的make命令，但可以根据需要进行编辑，以适应其他工程。

- launch.json是一个VS Code文件，用于配置调试设置。它运行上面的imx-deploy-gdb任务。

```
{
    "version": "0.2.0",
    "configurations": [{
        "name": "GDB debug",
        "type": "cppdbg",
        "request": "launch",
        "program": "${config:PROGRAM}",
        "args": [],
        "stopAtEntry": true,
        "cwd": "${workspaceFolder}",
        "environment": [],
        "MIMode": "gdb",
        "targetArchitecture": "arm64",
        "preLaunchTask": "imx-deploy-gdb",
        "setupCommands": [{
            "description": "Enable pretty-printing for gdb",
            "text": "-enable-pretty-printing",
            "ignoreFailures": true
        }],
        "miDebuggerPath": "/usr/bin/gdb-multiarch",
        "miDebuggerServerAddress": "${config:TARGET_IP}:3000",
    }]
}
```

其中：

- configurations.program：最终可执行文件的名称。
- configurations.args：在执行程序时传递给程序的参数。
- configurations.preLaunchTask：来自tasks.json的imx-deploy-gdb任务。
- configurations.miDebuggerPath：支持目标架构的gdb二进制文件的路径。
- configurations.miDebuggerServerAddress：电路板上gdb-服务器（由部署脚本启动）的地址和端口。

5.4 调试部署脚本

创建imx-deploy-gdb.sh脚本，放在工程的根目录下，内容如下：

```
#!/bin/bash
readonly TARGET_IP="$1"
readonly PROGRAM="$2"
readonly TARGET_DIR="/home/root"

# Must match startsPattern in tasks.json
```

```
echo "Deploying to target"

# kill gdbserver on target and delete old binary
ssh root@${TARGET_IP} "sh -c '/usr/bin/killall -q gdbserver; rm -rf
 ${TARGET_DIR}/${PROGRAM}  exit 0'"

# send the program to the target
scp ${PROGRAM} root@${TARGET_IP}:${TARGET_DIR}

# Must match endsPattern in tasks.json
echo "Starting GDB Server on Target"

# start gdbserver on target
ssh -t root@${TARGET_IP} "sh -c 'cd ${TARGET_DIR}; gdbserver localhost:3000
 ${PROGRAM}'"
```

工程的最终文件结构如图4所示。

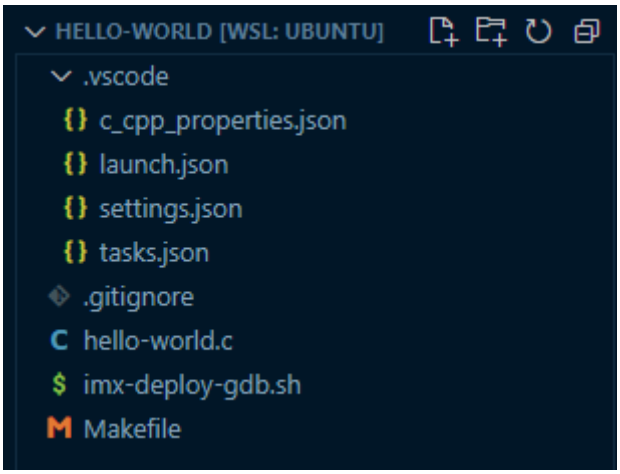

图4. 最终演示工程结构

5.5 设置和目标配置

- 通过以太网连接电路板和PC，将串行端口连接到PC。
- 几分钟后，i.MX电路板在连接的接口上有一个IP地址。可以通过串行终端运行以下命令并查看以太网地址来进行检查：

```
[I.MX Board Serial]
$ ip a s
```

- 电路板的IP地址必须与在`settings.json`中配置的IP地址相匹配。在这个示例中，手动将电路板的IP设置为`settings.json`中的IP，但也可以使用其他方法。要设置电路板的IP，请运行以下命令：

```
[I.MX Board Serial]
$ ifconfig eth0 169.254.158.177
```

- 使用ping测试PC与电路板之间的连接，然后确认`ssh`正在工作。在WSL上运行以下命令：

```
[WSL]
$ ping 169.254.158.177
$ ssh root@169.254.158.177
```

5.6 调试程序

- 在`hello-world.c`文件的主函数中设置一个断点。
- 在顶部栏中，按Run（运行）->Start Debugging（运行->开始调试）。
- VS Code编译可执行文件，将其发送到电路板，并启动一个调试会话，该会话在设置的断点处停止，如图5所示：

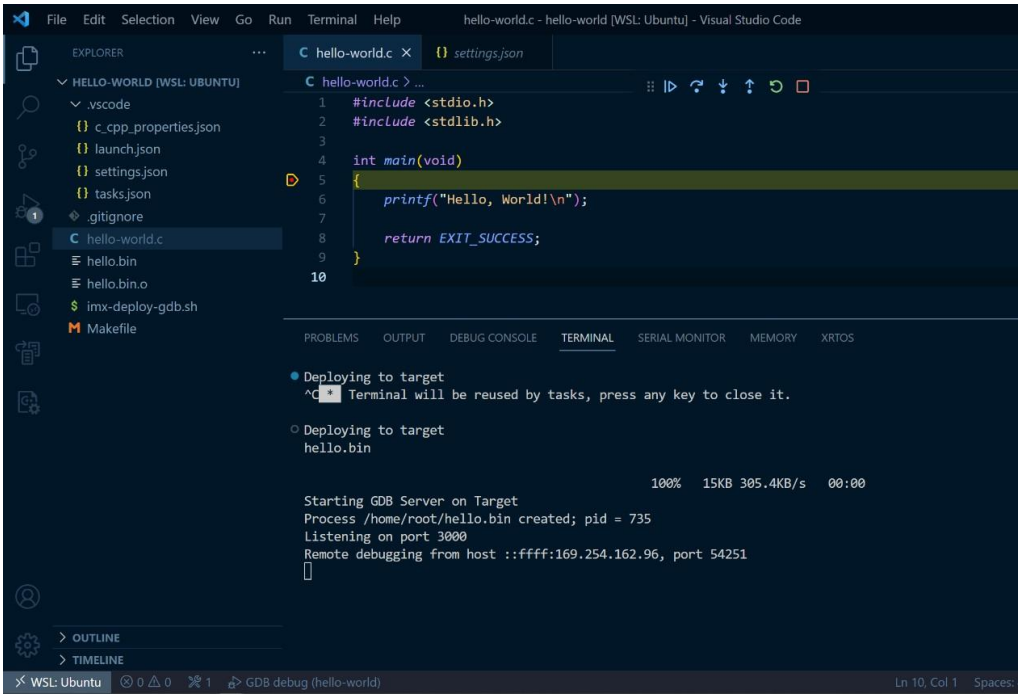

图5. 演示应用远程调试会话

6 关于本文中源代码的说明

本文中所示的示例代码具有以下版权和BSD-3-Clause许可：

2023年恩智浦版权所有。在满足以下条件的情况下，允许以源代码和二进制文件的形式重新分发和使用本源代码（无论是否经过修改）：

1. 重新分发源代码必须保留上述版权声明、这些条件和以下免责声明。
2. 以二进制文件形式重新分发时，必须在文档和/或随分发提供的其他材料中必须复制上述版权声明、这些条件和以下免责声明。
3. 未经事先书面许可，不得使用版权所有者的姓名或参与者的姓名为本软件的衍生产品进行背书或推广。

本软件由版权所有者和参与者“按原样”提供，不承担任何明示或暗示的担保责任，包括但不限于对适销性和特定用途适用性的暗示保证。在任何情况下，无论因何种原因或根据何种法律条例，版权所有者或参与者均不对因使用本软件而导致的任何直接、间接、偶然、特殊、惩戒性或后果性损害（包括但不限于采购替代商品或服务；使用损失、数据损失或利润损失或业务中断）承担责任，无论是因合同、严格责任还是侵权行为（包括疏忽或其他原因）造成的，即使事先被告知有此类损害的可能性也不例外。

7 修订历史

表1汇总了对本文件的修订。

表1. 修订历史

版本号	发布日期	说明
1	2023年10月3日	首次公开发布

8 Legal information

8.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

8.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com.cn/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. - NXP B.V. is not an operating company and it does not distribute or sell products.

8.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

i.MX — is a trademark of NXP B.V.

目录

For more information, please visit: http://www.nxp.com.cn

Date of release: 3 October 2023
Document identifier: AN14092