

应用笔记

N32H76x_H78x系列鸿蒙LiteOS-M移植应用笔记

简介

本文档介绍了在 N32H76x_78x 系列 MCU 中，如何移植基于 LiteOS-M 内核的 OpenHarmony 系统，适用于国民技术的 N32H76x、N32H78x 系列产品。

国民技术 版权所有

目录

目录	I
1. 概述	1
2. 开发环境与工具	1
2.1 GCC 开发环境	1
2.2 软件工具	1
2.3 硬件工具	1
3. LITEOS-M 移植	2
3.1 源码下载	2
3.1.1 LiteOS-M 内核源码下载	2
3.1.2 第三方开源软件下载	2
3.2 文件与目录	2
3.3 功能演示	4
3.3.1 代码修改	4
3.3.2 编译下载	5
3.3.3 补充说明	6
4. 历史版本	7
5. 声明	8

概述

鸿蒙系统采用了多内核结构，支持 Linux、LiteOS-A、LiteOS-M。N32H76x_78x 系列 MCU 支持 LiteOS-M 内核

本文档以 N32H785xXIB7 Cortex-M7 内核为例，介绍了在 GCC 环境下，如何移植基于 LiteOS-M 内核的 OpenHarmony 系统。

开发环境与工具

1.1 GCC开发环境

请参照应用笔记 AN_N32H78x_GCC_Development_V1.0.0 搭建 GCC 开发环境。

1.2 软件工具

示例基于以下软件测试通过：

- 编辑器 Visual Studio Code V1.101.0
- 编译工具链 gcc-arm-none-eabi-gcc 10-2020-q4-major
- GNU Make 4.3
- JLink V6.88b
- 串口助手 SSCOM V5.13.1

1.3 硬件工具

示例中用到了以下设备：

- 开发板 N32H785XIB7_STB
- 调试器 J-Link V9

LiteOS-M移植

1.4 源码下载

1.4.1 LiteOS-M内核源码下载

- 下载地址: https://gitee.com/openharmony/kernel_liteos_m
在 OpenHarmony 官方 Gitee 仓库下载, 下载前需要注册账号。

1.4.2 第三方开源软件下载

- 下载 third_party_bounds_checking_function-master:
https://gitee.com/openharmony/third_party_bounds_checking_function

1.5 文件与目录

- 1) 先将下载的 LiteOS-M 源码 kernel_liteos_m-master.zip 解压, 然后在 kernel_liteos_m-master 目录下新建 2 个目录: targets、thirdparty, 如下图所示。

📁 > D:_ThinkPad (D:) > temp > kernel_liteos_m-master

名称	修改日期	类型
📁 .gitee	2025/5/22 14:42	文件夹
📁 arch	2025/5/22 14:42	文件夹
📁 components	2025/5/22 14:42	文件夹
📁 drivers	2025/5/22 14:42	文件夹
📁 figures	2025/5/22 14:42	文件夹
📁 kal	2025/5/22 14:42	文件夹
📁 kernel	2025/5/22 14:42	文件夹
📁 targets	2025/5/22 18:01	文件夹
📁 testsuites	2025/5/22 14:42	文件夹
📁 third_party	2025/5/22 15:38	文件夹
📁 tools	2025/5/22 14:42	文件夹
📁 utils	2025/5/22 14:42	文件夹

- 2) 将第三方开源软件解压至新建的 thirdparty 目录下, 如下图所示。

📁 > D:_ThinkPad (D:) > temp > kernel_liteos_m-master > third_party

名称	修改日期	类型
📁 third_party_bounds_checking_functi...	2025/5/22 15:32	文件夹

- 3) 将 N32H785 SDK 中的软件代码复制到 targets 目录下, 然后在 examples 目录下新建 OpenHarmony 目录, 并创建测试工程 OpenHarmonyDemo。

D:_ThinkPad (D:) > temp > kernel_liteos_m-master > targets > Nations.N32H78x_Library.1.0.0 > projects > n32h78x_EVAL > examples > OpenHarmony			
名称	修改日期	类型	大小
OpenHarmonyDemo	2025/6/12 19:21	文件夹	

4) 在以下目录中，查看是否有下图所示 GCC 链接配置文件，如果没有则创建。

D:_ThinkPad (D:) > temp > kernel_liteos_m-master > targets > Nations.N32H78x_Library.1.0.0 > firmware > CMSIS > device			
名称	修改日期	类型	大小
startup	2025/6/12 19:20	文件夹	
n32h76x_78x.h	2025/5/20 16:06	C Header 源文件	2,109 KB
n32h78x_flash.ld	2025/6/4 17:40	LD 文件	7 KB
system_n32h76x_78x.c	2025/5/20 17:53	C 源文件	10 KB
system_n32h76x_78x.h	2025/3/18 10:25	C Header 源文件	4 KB

5) 在以下目录中，查看是否有下图所示串口打印重映射文件，如果没有则创建。

D:_ThinkPad (D:) > temp > kernel_liteos_m-master > targets > Nations.N32H78x_Library.1.0.0 > projects > n32h78x_EVAL > bsp > src			
名称	修改日期	类型	大小
delay.c	2025/5/20 16:26	C 源文件	6 KB
log.c	2025/3/6 10:44	C 源文件	6 KB
print_remap.c	2025/5/30 21:33	C 源文件	4 KB

6) 在以下目录中，查看是否有下图所示 GCC 启动文件，如果没有则创建。

D:_ThinkPad (D:) > temp > kernel_liteos_m-master > targets > Nations.N32H78x_Library.1.0.0 > firmware > CMSIS > device > startup			
名称	修改日期	类型	大小
startup_n32h78x_cm4.s	2025/4/14 17:06	S 文件	52 KB
startup_n32h78x_cm7.s	2025/5/20 16:07	S 文件	52 KB
startup_n32h78x_gcc.s	2025/6/9 19:45	S 文件	36 KB

7) 在工程 OpenHarmonyDemo 下，创建 GCC 目录，并创建以下文件：

- GCC 启动文件：startup_n32h78x_gcc.s
从上一步的目录中复制 startup_n32h78x_gcc.s 到 GCC 目录。
- GCC 编译脚本：Makefile
- LiteOS-M 编译脚本：liteos_m.mk
- J-Link 下载脚本：flash.jlink

最终，GCC 目录下共有 4 个文件，如下图所示：

kernel_liteos_m-master > targets > Nations.N32H78x_Library.1.0.0 > projects > n32h78x_EVAL > examples > OpenHarmony > OpenHarmonyDemo > GCC			
名称	修改日期	类型	大小
flash.jlink	2025/6/11 19:52	JLINK 文件	1 KB
liteos_m.mk	2025/6/9 20:17	Makefile 源文件	3 KB
Makefile	2025/6/9 20:31	文件	10 KB
startup_n32h78x_gcc.s	2025/6/9 19:45	S 文件	36 KB

8) 在工程 inc 目录中创建 LiteOS-M 自定义配置文件 target_config.h，如下图所示：

p > kernel_liteos_m-master > targets > Nations.N32H78x_Library.1.0.0 > projects > n32h78x_EVAL > examples > OpenHarmony > OpenHarmonyDemo > inc

名称	修改日期	类型	大小
main.h	2025/6/9 15:25	C Header 源文件	5 KB
n32h76x_78x_it.h	2025/5/29 16:22	C Header 源文件	4 KB
target_config.h	2025/6/11 16:37	C Header 源文件	8 KB

1.6 功能演示

1.6.1 代码修改

- 1) 修改内核头文件 core_cm7.h，屏蔽未定义的__COMPILER_BARRIER 函数。

电脑 > D:_ThinkPad (D:) > temp > kernel_liteos_m-master > targets > Nations.N32H78x_Library.1.0.0 > firmware > CMSIS > core

名称	修改日期	类型	大小
arm_common_tables.h	2025/3/6 10:44	C Header 源文件	6 KB
arm_const_structs.h	2025/3/6 10:44	C Header 源文件	3 KB
arm_math.h	2025/3/6 10:44	C Header 源文件	241 KB
cmsis_armcc.h	2025/3/6 10:44	C Header 源文件	29 KB
cmsis_armclang.h	2025/3/6 10:44	C Header 源文件	54 KB
cmsis_compiler.h	2025/3/6 10:44	C Header 源文件	10 KB
cmsis_gcc.h	2025/3/6 10:44	C Header 源文件	59 KB
cmsis_icarm.h	2025/3/6 10:44	C Header 源文件	27 KB
cmsis_version.h	2025/3/6 10:44	C Header 源文件	2 KB
core_cm3.c	2025/3/6 10:44	C 源文件	18 KB
core_cm3.h	2025/3/6 10:44	C Header 源文件	88 KB
core_cm4.h	2025/3/6 10:44	C Header 源文件	111 KB
core_cm4_simd.h	2025/3/6 10:44	C Header 源文件	25 KB
core_cm7.h	2025/6/12 19:34	C Header 源文件	146 KB
core_cmFunc.h	2025/3/6 10:44	C Header 源文件	18 KB
core_cmInstr.h	2025/3/6 10:44	C Header 源文件	21 KB
mpu_armv7.h	2025/3/6 10:44	C Header 源文件	12 KB

```

1903  __STATIC_INLINE void __NVIC_EnableIRQ(IRQn_Type IRQn)
1904  {
1905      if ((int32_t)(IRQn) >= 0)
1906      {
1907          //__COMPILER_BARRIER();
1908          NVIC->ISER[(((uint32_t)IRQn) >> 5UL)] = (uint32_t)(1UL << (((uint32_t)IRQn) & 0x1FUL));
1909          //__COMPILER_BARRIER();
1910      }
1911  }

```

- 2) 以下需要创建的文件均可在例程中找到，可参照修改使用：
- GCC 启动文件：startup_n32h78x_gcc.s
 - GCC 编译脚本：Makefile
 - GCC 链接配置文件：n32h78x_flash.ld
 - 串口重映射：print_remap.c
 - LiteOS-M 编译脚本：liteos_m.mk
 - LiteOS-M 自定义配置文件：target_config.h
 - J-Link 下载脚本：flash.jlink
- 3) 建议在 GCC 编译脚本 Makefile 中，选择需要用到的外设驱动并屏蔽未使用的外设驱动

```













36 # BSP
37 C_SOURCES += $(wildcard $(PROJSRC)/projects/n32h78x_EVAL/bsp/src/*.c)
38
39 # Driver
40 C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/misc.c
41 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_adc.c
42 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_comp.c
43 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_cordic.c
44 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_crc.c
45 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_dac.c
46 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_dbg.c
47 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_dcmu.c
48 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_dma.c
49 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_dmamux.c
50 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_dsmu.c
51 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_dvp.c
52 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_eccmon.c
53 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_eth.c
54 C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_exti.c
55 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_fdcan.c
56 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_femc.c
57 #C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_fmac.c
58 C_SOURCES += $(PROJSRC)/firmware/n32h76x_78x_std_periph_driver/src/n32h76x_78x_gpio.c

```

4) LiteOS-M 默认配置在 `los_config.h` 文件中，如下图所示。

如果需要修改默认配置，可在 LiteOS-M 自定义配置文件 `target_config.h` 中增加对应宏定义。

电脑 > D:_ThinkPad (D:) > temp > kernel_liteos_m-master > kernel > include

名称	修改日期	类型	大小
 los_config.h	2025/4/9 16:36	C Header 源文件	21 KB
 los_event.h	2025/4/9 16:36	C Header 源文件	13 KB
 los_membox.h	2025/4/9 16:36	C Header 源文件	8 KB
 los_memory.h	2025/4/9 16:36	C Header 源文件	21 KB
 los_mux.h	2025/4/9 16:36	C Header 源文件	13 KB
 los_queue.h	2025/4/9 16:36	C Header 源文件	55 KB
 los_sched.h	2025/4/9 16:36	C Header 源文件	5 KB
 los_sem.h	2025/4/9 16:36	C Header 源文件	13 KB
 los_sortlink.h	2025/4/9 16:36	C Header 源文件	5 KB
 los_swtmr.h	2025/4/9 16:36	C Header 源文件	19 KB
 los_task.h	2025/4/9 16:36	C Header 源文件	56 KB
 los_tick.h	2025/4/9 16:36	C Header 源文件	18 KB

1.6.2 编译下载

OpenHarmonyDemo 例程编译后，下载到开发板，程序自动运行，开发板上 D1 闪烁，同时可通过串口助手查看相关信息，主要实现以下功能：

- 1) 通过串口打印输出运行信息：波特率 115200，TX-PA9, RX-PA10。
- 2) 主任务 TaskLoop 每 5 秒打印一次“TaskLoop running..”，同时 D1 闪烁。
- 3) 轻按 KEY1(PC0),触发任务 TaskSem，打印一次“TaskSem running...”

- 4) 轻按 KEY2(PC1),触发任务 TaskPowerMode, 系统进入 STOP0 模式, 可通过 WKUP(PA0)唤醒
- 5) 通过串口发送数据给 MCU, 返回取反后的数据
 - 如果数据长度大于 8byte, 第 1 个字节必须为数据长度, 触发 TaskUart 任务。
 - 如果数据长度大于 8byte, 数据可任意定义, 触发 TaskUartCopy 任务。

1.6.3 补充说明

- 1) 鸿蒙 LiteOS-M 内核心跳周期通过宏“LOSCFG_BASE_CORE_TICK_PER_SECOND”定义, 默认值为 100 (即 10ms)。例程中修改了心跳周期, 在“target_config.h”中重新定义为 1000 (1ms)。
- 2) SysTick 中断周期与心跳周期无关, 通过宏“LOSCFG_BASE_CORE_TIMESLICE_TIMEOUT”定义, 默认值为 20000 (即 20ms), 并且会动态调整。
- 3) 任务切换通过 SysTick 中断触发, 即单个任务连续执行时间约为 20ms。

历史版本

版本	日期	备注
V1.0.0	2025.06.12	新建文档

声明

国民技术股份有限公司（下称“国民技术”）对此文档拥有专属产权。依据中华人民共和国的法律、条约以及世界其他法域相适用的管辖，此文档及其中描述的国民技术产品（下称“产品”）为公司所有。

国民技术在此并未授予专利权、著作权、商标权或其他任何知识产权许可。所提到或引用的第三方名称或品牌（如有）仅用作区别之目的。

国民技术保留随时变更、订正、增强、修改和改良此文档的权利，恕不另行通知。请使用者在下单购买前联系国民技术获取此文档的最新版本。

国民技术竭力提供准确可信的资讯，但即便如此，并不推定国民技术对此文档准确性和可靠性承担责任。

使用此文档信息以及生成产品时，使用者应当进行合理的设计、编程并测试其功能性和安全性，国民技术不对任何因使用此文档或本产品而产生的任何直接、间接、意外、特殊、惩罚性或衍生性损害结果承担责任。

国民技术对于产品在系统或设备中的应用效果没有任何故意或保证，如有任何应用在其发生操作不当或故障情况下，有可能致使人员伤亡、人身伤害或严重财产损失，则此类应用被视为“不安全使用”。

不安全使用包括但不限于：外科手术设备、原子能控制仪器、飞机或宇宙飞船仪器、所有类型的安全装置以及其他旨在支持或维持生命的应用。

所有不安全使用的风险应由使用人承担，同时使用人应使国民技术免于因为这类不安全使用而导致被诉、支付费用、发生损害或承担责任时的赔偿。

对于此文档和产品的任何明示、默示之保证，包括但不限于适销性、特定用途适用性和不侵权的保证责任，国民技术可在法律允许范围内进行免责。

未经明确许可，任何人不得以任何理由对此文档的全部或部分进行使用、复制、修改、抄录和传播。