

OPL1000

ULTRA-LOW POWER 2.4GHZ WI-FI + BLUETOOTH SMART SOC

SDK Getting Start Guide



OPULINKS

<http://www.opulinks.com/>

Copyright © 2017-2018, Opulinks. All Rights Reserved.

OPL1000-SDK-getting-start-guide-R01 | Version V01

Date	Version	Contents Updated
2018-05-11	0.1	<ul style="list-style-type: none">Initial Release

TABLE OF CONTENTS

1. 介绍 1

1.1. 文档应用范围 1

1.2. 缩略语 1

1.3. 参考文献 1

2. OPL1000 APP 开发流程 2

2.1. APP 与 ROM Code 和 Patch 关系 2

2.2. APP 开发流程 3

3. 使用 Keil 调试应用程序 6

3.1. Keil 工程配置 6

3.2. 应用程序在线调试 7

LIST OF FIGURES

Figure 1: 用户 APP 和 ROM CODE · Patch 之间的关系 2

Figure 2: 用户 APP 和 Patch 编译、载入过程 3

Figure 3: IDE 在线调试开发模式 4

Figure 4: 串口调试模式 5

Figure 6: J-link ICE 仿真器正确识别 7

Figure 7: Keil 编译 hello_world 示例代码 8

Figure 8: 使用 Keil C 在线调试 9

Figure 9: APS 串口打印确认程序执行结果 9

1. 介绍

1.1. 文档应用范围

本文档介绍了基于 OPL1000 DEVKIT 上开发 OPL1000 应用程序的流程和方法。

1.2. 缩略语

Abbr.	Explanation
APP	APPLication 应用程序
APS	Application Sub-system 应用子系统，在本文中亦指 M3 MCU
AT	Attention 终端命令指令集
DevKit	Development Kit 开发工具板
EVB	Evaluation Board 评估板
FW	FirmWare 固件，处理器上运行的嵌入式软件
ICE	In-Circuit Emulator 在线仿真调试工具
RX	Receive 接收
SWD	Serial Wire Debug 串行线调试
TX	Transmit 发送

1.3. 参考文献

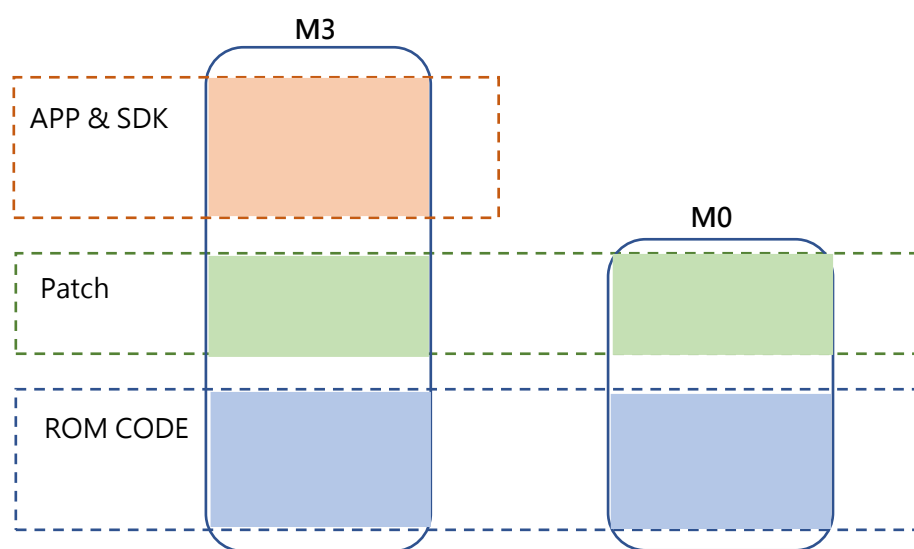
[1] OPL1000-patch-download-tool-user-guide-R01.pdf
[2] OPL1000-DEVKIT-getting-start-guide-R01.pdf

2. OPL1000 APP 开发流程

2.1. APP 与 ROM Code 和 Patch 关系

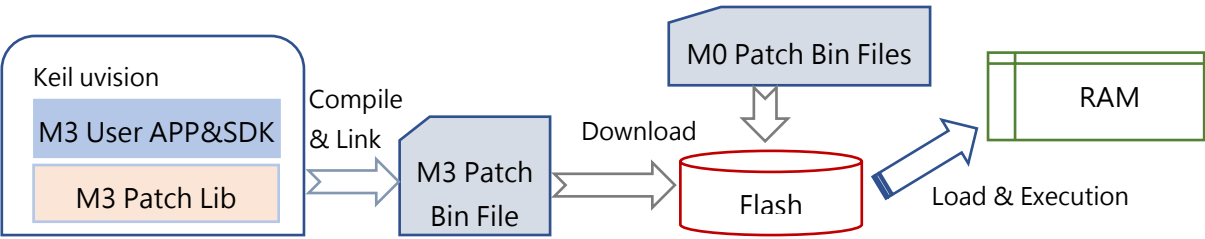
OPL1000 包含两个 MCU，ARM Cortex M3 和 Cortex M0。所谓 OPL1000 APP 开发是指在 OPL1000 的 M3 MCU 上开发用户的应用程序。OPL1000 的原初 M3、M0 固件以 ROM CODE 的方式包含在芯片中。除此之外由于功能扩展和修复 Bug，OPL1000 也提供了 M3 和 M0 的固件补丁。因此用户 App 的开发是基于 ROM Code 和固件补丁基础上完成的。它们之间的关系可以用图 Figure 1 表示。

Figure 1: 用户 APP 和 ROM CODE、Patch 之间的关系



M0 的 Patch 以二进制文件的方式由 Opulinks 给出。M3 Patch 以 lib 库文件的方式提供，用户的 APP 和 Opulinks 提供的 SDK 源码作为一个 Keil C 工程进行编译，因此生成的 M3 bin 文件包含 M3 的固件补丁，SDK 和用户 App 应用程序。最终 M0 的 Patch Bin 文件和 M3 的 Bin 文件合并后下载到片外 Flash 中，OPL1000 芯片上电后，将 Flash 中的 M3/M0 Bin 文件载入到 RAM 中执行。整个过程可以用图 Figure 2 表示。

Figure 2: 用户 APP 和 Patch 编译、载入过程



2.2. APP 开发流程

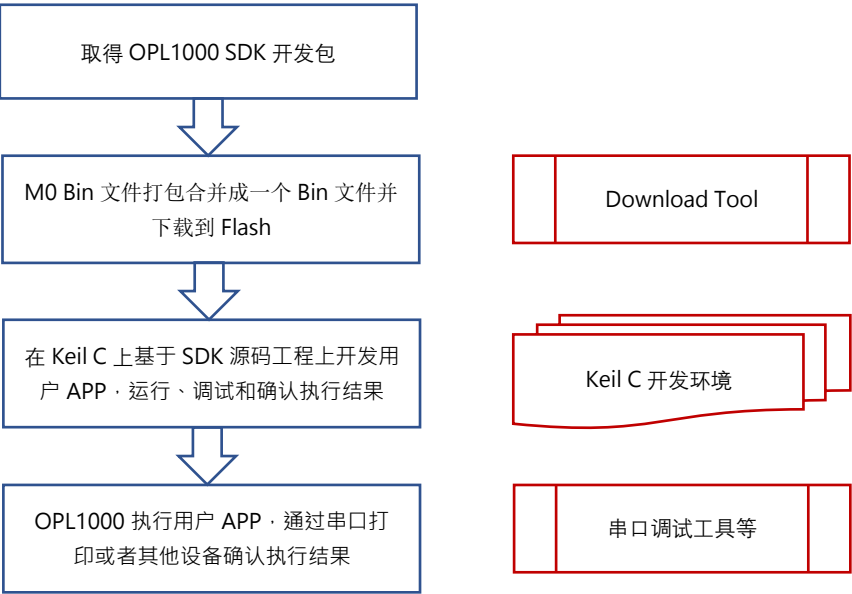
用户可以使用两种模式来开发应用程序。

模式一：IDE 在线调试开发模式

开发包括 4 个步骤，整个过程如图 Figure 3 所示。

- 1. 从原厂拿到 OPL1000 SDK 软件包。
- 2. 将 SDK 软件包的 FW_Binary 目录下 M0 Bin 文件（例如 OPL1000_M0_IRAMx.bin，x 随版本不同范围 1~3）按照参考文献[1]的 download 工具使用指南说明打包成一个 Bin 文件并下载到 Flash 中。注意 M0 bin 文件在 SDK 开发包没有更新的情况下只需要向 SPI flash 下载一次即可。
- 3. 在 Keil uvision 上开发用户 APP（基于 SDK 示例源码工程）。
- 4. 在 Keil C 环境下，在线仿真运行、调试，通过串口确认执行结果，注意此时并没有把代码下载到 flash 中，代码仅仅在 RAM 中执行。

Figure 3: IDE 在线调试开发模式

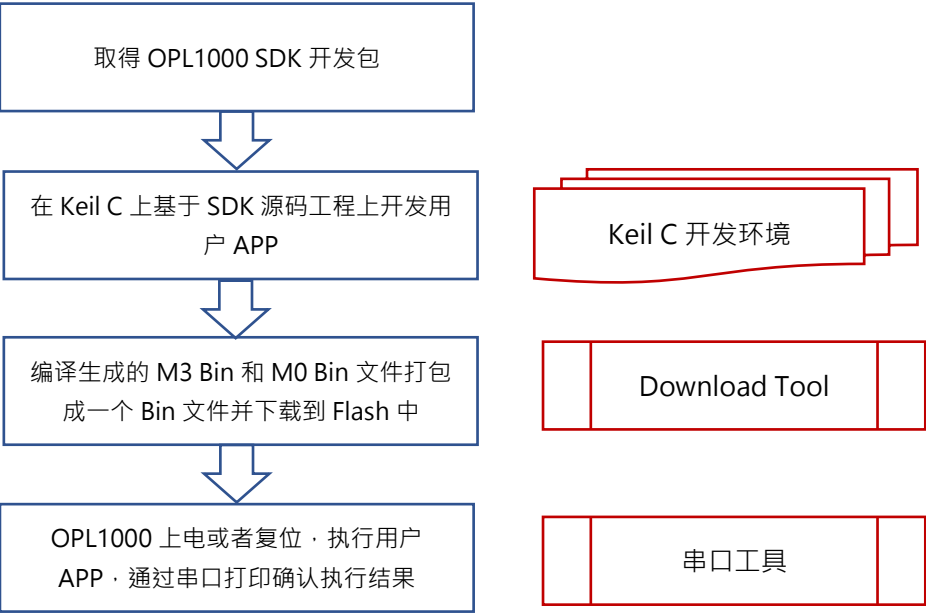


模式二：串口调试开发模式

开发包括 5 个步骤，整个过程如图 Figure 4 所示。

- 1. 从原厂拿到 OPL1000 SDK 软件包。
- 2. 在 Keil uvision 上开发用户 APP（基于 SDK 源码工程）。
- 3. 取得编译完成的 M3 Bin 文件
- 4. 将 SDK 软件包的 FW_Binary 目录下 M0 Bin 文件（例如 OPL1000_M0_IRAMx.bin，x 随版本不同范围 1~3）按照[参考文献\[1\]](#)的 download 工具使用指南说明打包成一个 Bin 文件并下载到 Flash 中。
- 5. OPL1000 上电或者复位，执行用户 APP，通过串口 log 信息确认执行结果。

Figure 4: 串口调试模式



在实际开发过程中用户可以结合自己的开发需求在两种开发模式之间灵活切换。

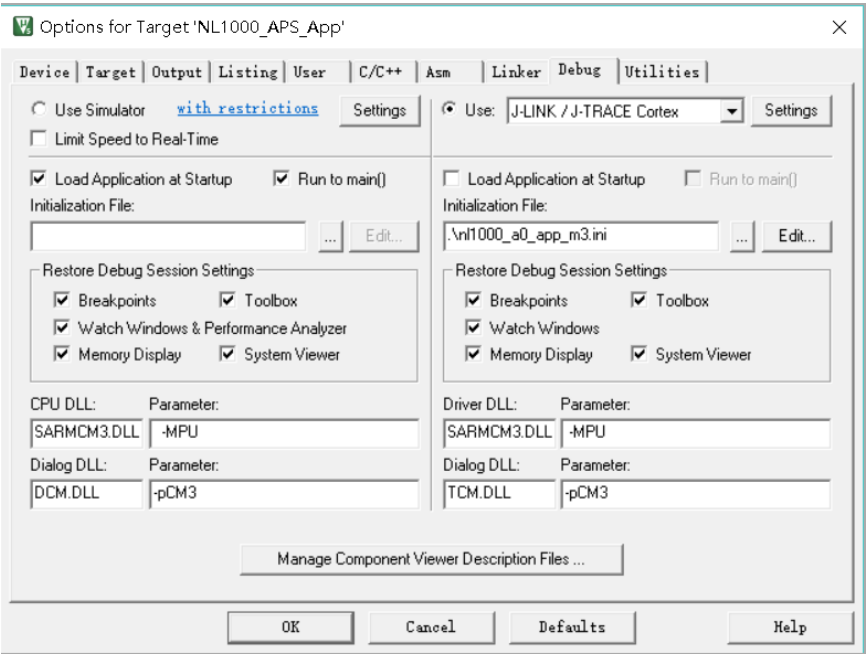
3. 使用 KEIL 调试应用程序

应用程序可以在 DEVKIT 板进行开发和调试。DEVKIT 板的使用可以参考[文献\[2\]](#) DEVKIT 快速上手指南。

3.1. Keil 工程配置

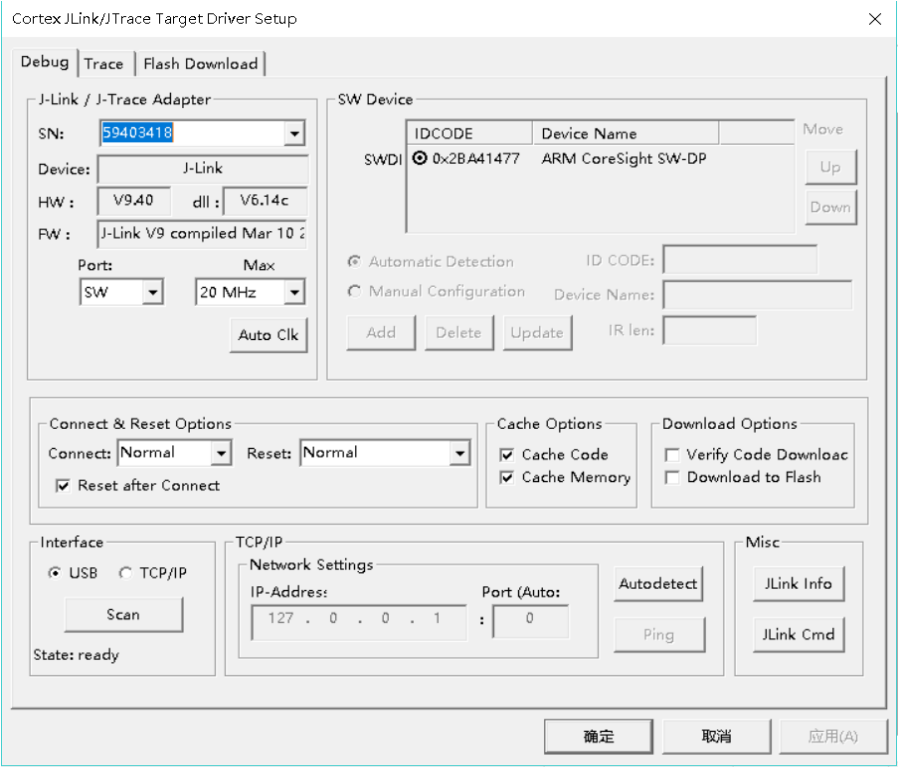
将 J-link 仿真器和 DevKit 板正确连接，并且 DevKit 板使用 USB 供电。打开工程：SDK\APS_PATCH\examples\get_started\hello_world。选择 Options for Target 按钮，弹出设置对话框，选择 Debug 界面，出现下图所示界面。

Figure 5: Options for Target 设置对话框



调试接口选择 JLINK / J-TRACE Cortex 接口，如果用户使用其他仿真器请选择对应的仿真接口类型，例如使用 CMSIS DAP 仿真器则需要选择 CMSIS-DAP Debugger 接口。选择正确后进入 Settings 按钮，出现 Figure 6Figure 6 所示结果。在 SW Device 显示的是设备 ID 编码，表示设备连接和工作正常，如果 SW Device 为空则需要检查设备接线，确保接线正确。

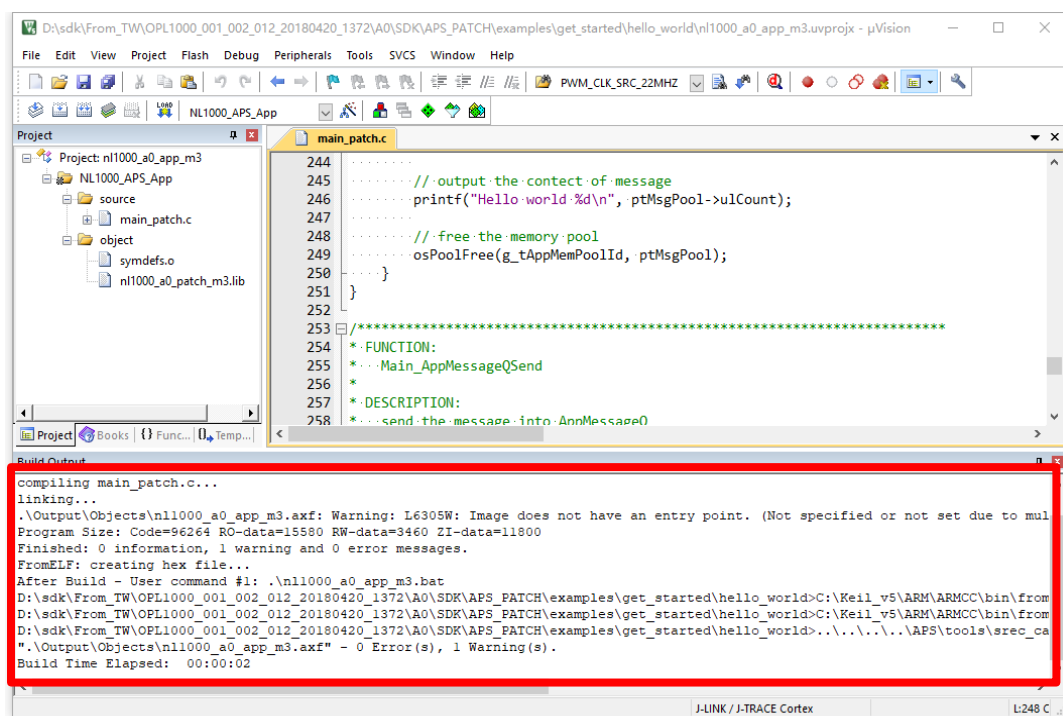
Figure 6: J-link ICE 仿真器正确识别



3.2. 应用程序在线调试

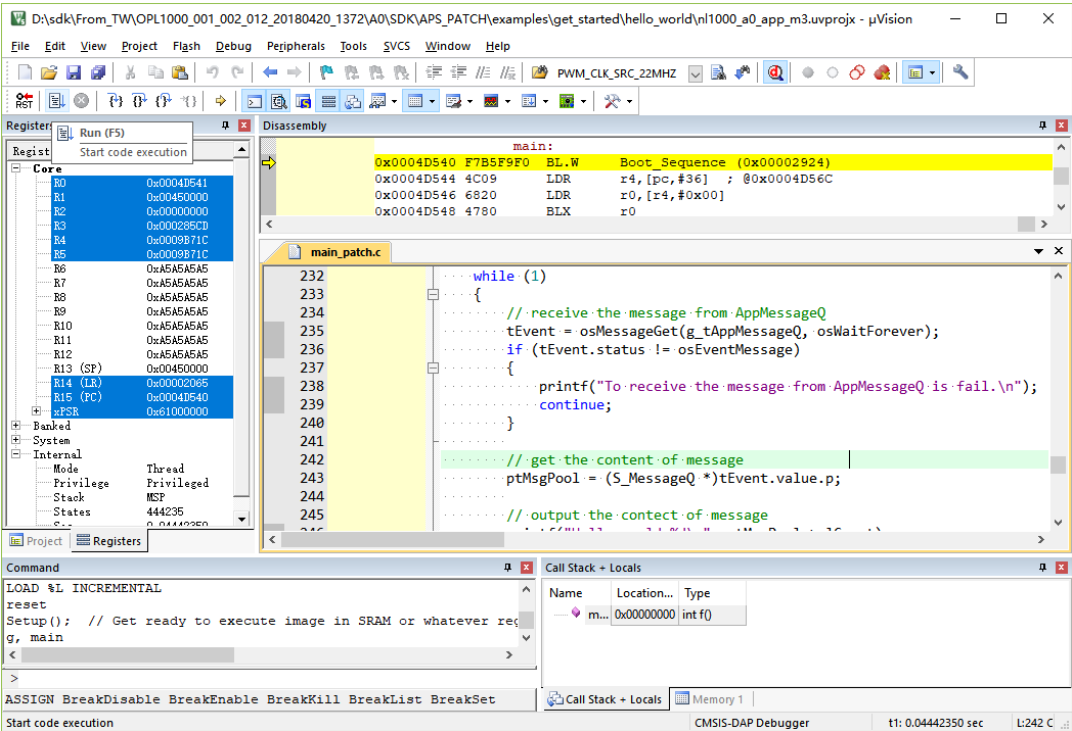
Keil 软件正确识别到 OPL1000 设备后，build 示例工程，得到 Figure 7 画面，表示编译正确完成。

Figure 7: Keil 编译 hello_world 示例代码



编译完成后，点击 Debug 按钮，正确进入 debugger 环境，如图 Figure 8 所示。

Figure 8: 使用 Keil C 在线调试




在确保串口接线正常的情况下，使用串口工具打开 APS 串口，同时在 keil 里面点击全速运行 ，此时 APS 串口打印 Figure 9 所示 log 信息，表明程序在 RAM 中执行正确。

Figure 9: APS 串口打印确认程序执行结果

```
The init of MM_FIM is done.

[Lib] SDK version info: 1369
[Lib] Compile time: 2018/04/20 14:54:58

[SVN REV] SVN_REVISION:809
wifiMac Task create successful
Supplicant task is created successfully!
controller_queue creates successfully!
controller_queue_ble creates successfully!
controller_task_create successful!
LE Task create successful
Sw patch is changed successfully.
wifi_mac_tassupplicant_controller_tle Task Runk_init entrytask_init eask is going

ntry
!Debug Switc
h = f000ea2 flag = 0
Diag task is created successfully!
Hello world 1
Hello world 2
Hello world 3
Hello world 4
Hello world 5
Hello world 6
Hello world 7
Hello world 8
Hello world 9
```

CONTACT

sales@Opulinks.com