

BS2XV100 SDK 开发环境搭建
BS20/BS21/BS21E/BS22/BS26

用户指南

文档版本 01
发布日期 2024-05-15

前言

概述

本文档系统性描述 BS2X 芯片解决方案 SDK 提供的内容，介绍了 BS2X SDK 的开发环境以及开发流程，提供编译下载示例帮助开发者快速上手 BS2X 芯片开发。

本文档以 BS21 为例进行说明示例，后续不再单独说明，请用户知悉。




读者对象



本文档主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示如不可避免则将会导致死亡或严重伤害的具有高等级风险的危害。
 警告	表示如不可避免则可能导致死亡或严重伤害的具有中等级风险的危害。
 注意	表示如不可避免则可能导致轻微或中度伤害的具有低等级风险的危害。

符号	说明
 须知	用于传递设备或环境安全警示信息。如不可避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
 说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
BS2X	V100

修改记录

文档版本	发布日期	修改说明
01	2024-05-15	第一次正式版本发布。
00B03	2024-02-29	更新 “3.1.2 编译配置” 小节内容。
00B02	2023-12-04	更新 “3.1.1 启动编译” 小节内容。
00B01	2023-10-27	第一次临时版本发布。

目 录

前言	i
1 SDK 发布概述	1
1.1 doc 目录概述	1
1.2 software 目录概述	2
1.3 Tools 目录概述	2
2 编译环境介绍	4
2.1 编译构建环境	4
2.1.1 操作系统	4
2.1.2 依赖软件	5
2.1.3 编译工具链	5
2.2 编译系统注意事项	5
3 SDK 开发流程	7
3.1 版本编译	7
3.1.1 启动编译	7
3.1.2 编译配置	8
3.1.3 Bin 文件描述	9
3.2 版本烧录	10
4 开发示例	11
4.1 添加代码	11
5 开发调试工具应用	15
5.1 基础开发工具应用介绍	15
5.2 基础维测工具应用介绍	16

5.3 基础烧录工具应用介绍.....16

1 SDK 发布概述

BS2X 芯片解决方案 SDK 包含 doc 目录、software 目录以及 Tools 目录。

- doc 目录下存放系统软件各子系统开发指南、说明文档以及 API 手册。
- software 目录下分为 bin 目录以及 code 目录，分别存放发布以及系统启动需要的二进制文件以及 SDK 发布代码。
- Tools 目录用于存放系统编译、开发、调试、资源压缩、维测、产测等环节所用到的工具。

1.1 doc 目录概述

1.2 software 目录概述

1.3 Tools 目录概述

1.1 doc 目录概述

doc 目录用于存放系统软件各子系统的开发指南以及 API 手册。

开发指南主要包含如下大类以及内容，通过 pdf 以及 word 格式分别提供以下手册。

- 业务领域开发指南：《BS2XV100 星闪软件开发指导书》、《BS2XV100 蓝牙软件 UUID Server Sample 说明书》、《BS2XV100 蓝牙软件开发指导书》。
- 系统功能领域开发指南：《BS2XV100 FOTA 开发指南》、《BS2XV100 低功耗软件开发指南》、《BS2XV100 NV 存储用户指南》、《BS2XV100 SDK 开发指南》。
- 驱动开发指南：《BS2XV100 设备驱动开发指南》。

API 手册：各业务、驱动、系统、工具等领域的 API 集合，用于 API 功能查询以及参数查询的全量子系统。

1.2 software 目录概述

software 目录主要存放 SDK 软件代码、工程以及系统 A 核、bt 核分别启动所需要的二进制版本文件。其中 bt 核为固定发布，不支持基于发布的 SDK 代码进行增量开发以及修改。用户可以基于 A 核版本进行增量开发。

sdk 目录包含如下子目录：

表1-1 SDK 目录

application	应用目录，支持用户在此目录下增量开发上层系统内容以及应用内容。全系统启动的代码也在此目录下。当前 SDK 会在此目录下提供部分 sample 代码用于客户开发参考。
bootloader	用于存放系统启动过程中二进制加载流程的代码。
build	编译构建工程代码。
drivers	设备驱动代码，提供 BSP 驱动代码。
include	公共头文件。
interim_binary	软件库、二进制文件存放目录。
kernel	操作系统以及操作系统功能封装代码。
middleware	中间件代码，提供系统框架代码。
open_source	开源相关组件代码。
output	编译构建版本生成目录（编译时生成）。
test	部分驱动测试代码。
Tools	编译工具链及打包工具。

1.3 Tools 目录概述

Tools 目录用于存放开发、调试以及版本烧录的多个工具链，当前提供三种工具内容，如表 1-2 所示。

表1-2 Tools 工具

DebugKits	提供日志打印解析、定制命令输入、内存导出解析等功能。
BurnTool	提供版本烧录功能。
DevEco	开发工具，集成一键式编译、烧录、单步调试以及内存查询，变量监控等功能。

更多详细工具使用、安装功能请参考对应工具使用指南文档。

2 编译环境介绍

BS2X SDK 按照不同的交付模式，分为两种编译构建系统：

- 对接型编译系统。

即 BS2X SDK 整体作为芯片解决方案提供底层解决方案能力，交付 SDK 给客户系统，按照用户整体系统编译框架的编译原则进行编译。此种情况下需要支持与用户的系统编译框架进行对接。当前基于此种编译系统，BS2X 支持基于 cmake 和 makefile 两种编译语言系统的编译框架对接。

- 开发者模式编译系统。

开发者模式编译系统主要通过 BS2X SDK 提供全套的编译框架和编译系统，用户通过 IDE 添加文件或通过适配修改自身编译系统的方式，来适配 BS2X SDK 的编译系统。

两种编译构建系统分别对接两类不同模式的用户群体。后续会根据这两种不同的编译系统类型进行编译系统的开发指导介绍。

2.1 编译构建环境

2.2 编译系统注意事项

2.1 编译构建环境

2.1.1 操作系统

支持 linux 环境下以及 windows 环境下进行编译环境部署以及版本编译。

对应操作系统版本支持如表 2-1 所示。

表2-1 编译环境操作系统

操作系统	操作系统版本
linux	Ubuntu 20.04
windows	Windows 10

2.1.2 依赖软件

表2-2 软件依赖列表

软件名称	版本信息
python	3.8.0 以上。
Cmake	3.14.1 以上。
ninja	SDK 发布，仅 windows 下编译需要对应 ninja.exe 文件，需要配置环境变量。

2.1.3 编译工具链

系统提供基于 windows 以及 linux 两套编译工具链，以提供 llvm 编译能力：

cc_riscv32_musl cc_riscv32_musl_fp	linux 编译工具链。
cc_riscv32_musl_fp_win	windows 编译工具链。

linux 编译器路径：

sdk\tools\bin\compiler\riscv\cc_riscv32_musl_b090\cc_riscv32_musl_fp

windows 编译器路径：

sdk\tools\bin\compiler\riscv\cc_riscv32_musl_b090\cc_riscv32_musl_fp_win

2.2 编译系统注意事项

BS21 芯片解决方案 SDK 提供基于 windows 下 IDE 工具的一键编译能力，也提供基于 windows/linux 系统下命令行编译能力。全系统编译方式均为 python+cmake+riscv 编

译器的方式，支持 windows/linux 跨操作系统编译。需要注意的是，windows 下编译需要 ninja 库支持，对应 ninja.exe 也会在 SDK 包中同步提供。

3 SDK 开发流程

3.1 版本编译

3.2 版本烧录

3.1 版本编译

3.1.1 启动编译

BS2X 提供了 Windows 下 IDE 一键编译和 Windows/Linux 下命令行编译两种编译方式。基于 IDE 的编译，请参考《BS2XV100 IDE 工具使用指南》，采取工程式一键编译，本文档中不做赘述。当前描述基于 SDK 使用 cmake 的命令行编译方式。

cmake 编译

linux 下 cmake 命令行编译，在安装好对应编译依赖软件后，在 linux 下运行 build.py 脚本进行编译。整体步骤如下：

步骤 1 在 SDK 根目录下，通过 “./build.py bs21 -c” 指令，即可编译。

步骤 2 运行结束后在 output 目录下获取编译后生成的二进制文件，整体编译完成。

----结束

windows 下 cmake 编译，在安装好对应编译依赖软件后，在 tools\DevEco\DevTools_BS21_v1.3（路径待确认）下运行 env_start.bat 脚本，整体步骤如下：

步骤 1 进入 tools\DevEco\DevTools_BS21_v1.3 目录，双击运行 env_start.bat 脚本。

步骤 2 在命令行指令框内，通过输入 windows 的 “CD: ” 命令，将命令行盘符切换至 software\code\sdk 目录，运行 “python build.py bs21 -c” 指令即可开始编译。

步骤 3 运行结束后在 output 目录下获取编译后生成的二进制文件，整体编译完成。

----结束

3.1.2 编译配置

编译过程中使用到的一些依赖文件以及配置文件主要存放在 sdk/build 目录下，其子目录相关描述如下：

- cmake：存放了编译过程依赖的.cmake 文件。
- config：存放了编译过程用到的配置文件。
- script：存放了编译过程依赖的 python 脚本。
- toolchains：存放编译工具链配置。

其中 cmake、script 以及 toolchains 主要与编译流程相关，用户无需过多关注，只需要关注 config 目录下配置文件即可。目前 sdk 的编译方式以组件的形式进行编译，编译组件通过 sdk\build\config\target_config\bs21\config.py 文件进行控制，在 config.py 文件中定义了各个编译目标，其成员可以配置编译该目标时的使用宏定义、组件以及其他暂不需要关注的功能选项。

以 standard-bs21 编译目标为例，其中 defines 用于追加编译 standard-bs21 目标时使用的宏定义，ram_component 用于编译 standard-bs21 目标时需要编译的单个组件。ram_component_set 用于编译 standard-bs21 目标时需要编译的组件集合（组件集合在 sdk\build\config\target_config\common_config.py 中定义）。下图出现的 “-:CHIP_BS21=1” 表示选择 BS21 芯片。

```
build > config > target_config > bs21 > target_config.py ...
9
10 target_template = {
11     'target_bs21_application_template': {
12         'chip': 'bs21',
13         'core': 'acore',
14         'board': 'evb',
15         'tool_chain': 'riscv32_musl_b090_fp',
16         'build_type': 'COMPILE',
17         'os': 'litedos',
18         'std_libs': [],
19         'CONFIG_TIMER_USING_V150': 'y',
20         'CONFIG_I2C_USING_V151': 'y',
21         'defines': ['-CHIP_BS21-1', 'LIBCPU_UTILS', 'LIBLIB_UTILS', 'PRE_ASIC', 'LIBPANIC', 'LIBAPP_VERSION'],
22         'VERSION_STANDARD', 'LIBBUILD_VERSION', 'CONFIG_UART_FIFO_DEPTH_64',
23         'LIBTEST_COMMON', 'ALL_SOURCE', 'LSCFG_MEM_TASK_STAT', 'SUPPORT_ECK', 'BS21_PRODUCT_EVB',
24         'LIBLOG', 'LIBLOG_READER', 'USE_LITEOS', 'USE_VECTORS', 'BUILD_APPLICATION_STANDARD', 'CHD_ENABLE',
25         'LITEOS_', 'LIBCPU_LOAD', 'LSCFG_DRIVERS_BPMK', 'UART_DRIVER_CONFIG_USE_VECTOS_INSTEAD_OF_TIMERS',
26         'USE_OSSIS_OS', 'LIBCPU', 'LITEOS_OSSIS_ARCH', 'LIBCPU_INTERRUPT', 'LSCFG_65_FAT_CACHE', 'DOWN_MEM_SUPPORT',
27         'CONFIG_NV_SUPPORT_SINGLE_CORE_SYSTEM', 'CONFIG_NV_FEATURE_SUPPORT',
28         'LIBUTIL_COMPAT', 'LITEOS_288F', 'BTH_LOG_BUG_FIX', 'OSALLOG_DISABLE', 'SUPPORT_DIAG_V2_PROTOCOL',
29         'defines_set': ['libsec_defines', 'chip_defines', 'version_defines'],
30         'ram_component': ['main_init_porting', 'arch_boot', 'pmu_porting', 'libboundscheck', 'non_os', 'board_config', 'ulp_aon', 'osal',
31             'clocks_porting', 'error_code', 'hal_mips', 'mcm_config_porting',
32             'dtx_exception', 'lpm', 'chip_porting', 'cmn_header', 'driver_header',
33             'hal_uart', 'uart', 'uart_porting', 'connectivity',
34             'gssll_hmac_sm3', 'systick', 'txco', 'hal_systick', 'hal_txco', 'lib_utils', 'porting_inc',
35             'segger_b090_fp', 'usb_class', 'usb_class_open', 'usb_class_header', 'test_usb_unified',
36             'trng_port', 'trng', 'hal_trng', 'nv', 'nv_porting', 'partition', 'partition_porting',
37             'at', 'at_cmd_port', 'at_plt_cmd', 'at_btc_cmd'],
38         'ram_component_set': ['cpu', 'l2c', 'lpc', 'cpu_trace', 'gpio_v150', 'time_set', 'otp', 'mem',
39             'watchdog', 'pinctrl', 'pm_set', 'usb_unified', 'pmp_set', 'pm_clock_set', 'pm_pmu_set'],
40         'rom_component': [
41             ],
42         'bin_name': 'application',
43         'ccflags': [
44             '-mabi=ilp32', '-march=rv32imc', '-mabi=ilp32f', '-march=rv32imcf', '-madjust-regorder', '-madjust-const-cost', '-freorder-commu-args', '-fimm-compare-expand',
45             '-fmu-str-zero', '-wfp-const-opt', '-mswitch-jump-table', '-ftrtl-sequence-abstract',
46             '-ftrtl-hoist-sink', '-fsafe-alias-multipointer', '-finline-optimize-size', '-short-enums',
```

用户如果需要添加编译组件以及编译时使用的宏定义可以在此处追加。

3.1.3 Bin 文件描述

- bin 文件介绍。

用户编译时默认已经生成了 loadboot.bin、flashboot.bin。application.bin 需要客户自己编译，编译完成后会将上述 bin 合入成一个 fwpkg 文件。loadboot.bin、flashboot.bin 文件位于 sdk\interim_binary\bs21\bin\路径下，用户无法二次开发，bs21_all_nv.bin 当前版本不支持。

loadboot.bin	用于下载镜像。
flashboot.bin	二级 boot 流程。
application.bin	a 核系统版本二进制文件，可通过 code 目录 SDK 版本增量开发后生成。
bs21_all_nv.bin	nv 组件依赖的 bin 文件。

- bin 文件打包。

以 bs21 为例，在配置了 sdk\build\config\target_config\bs21\config.py 中编译目标的 packet 属性为 true 后，每次编译后会自动将编译生成的 bin 文件和 sdk\interim_binary\bs21\bin 目录下的其他 bin 文件打包成一个 fwpkg 文件，位于 “sdk\tools\pkg\fwpkg\bs21\bs21_all.fwpkg” 路径下，该打包文件用于后续的版本烧录流程。

3.2 版本烧录

编译完成、系统版本准备就绪后，SDK 提供两种烧录方式：

- 烧录工具进行烧录。

SDK 包 Tools 目录下提供 BurnTool 工具，专用于版本烧录，同时支持产测阶段一拖多进行烧写。同时支持基于指令运行的烧写方式。详情请参见《BS2X BurnTool 工具使用指导书》。

- IDE 工具集成烧录。

SDK 包 Tools 目录下提供 DevEco IDE 工具包，安装后自动支持烧写功能。支持一键式编译后进行烧写。详情请参见《BS2X IDE 工具使用指南》。

两种方式均可独立完成烧录功能，均支持图形化界面对待烧录的二进制文件进行勾选。烧写均基于串口方式进行。

4 开发示例

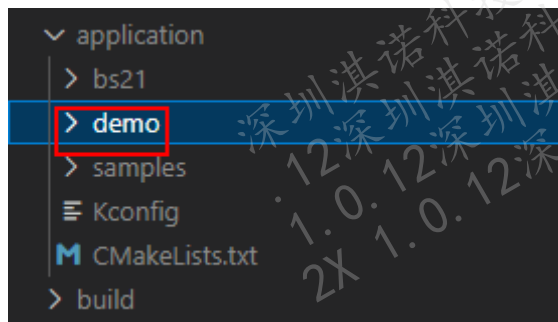
本章节以 IDE 编译下载方式为例，展示在获取到 BS2X SDK 并搭建好开发环境后，用户如何添加自定义代码。

4.1 添加代码

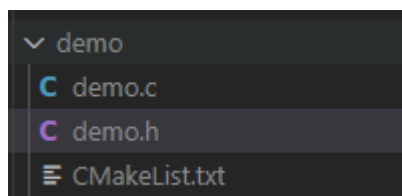
4.1 添加代码

用户添加自定义代码组件示例如下：

- 在 application 中创建一个 demo 代码目录。



- 添加用户自定义代码 demo.c、demo.h，同时创建一个空白 CMakeLists.txt 文件。

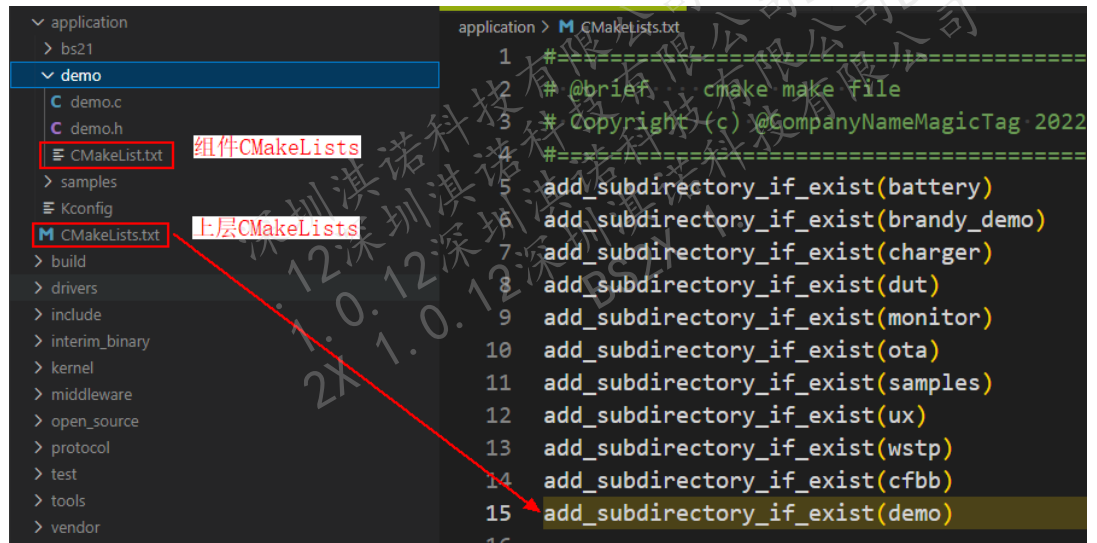


- 在 CMkakeLists.txt 中为组件模板添加源码文件以及源码头文件路径。

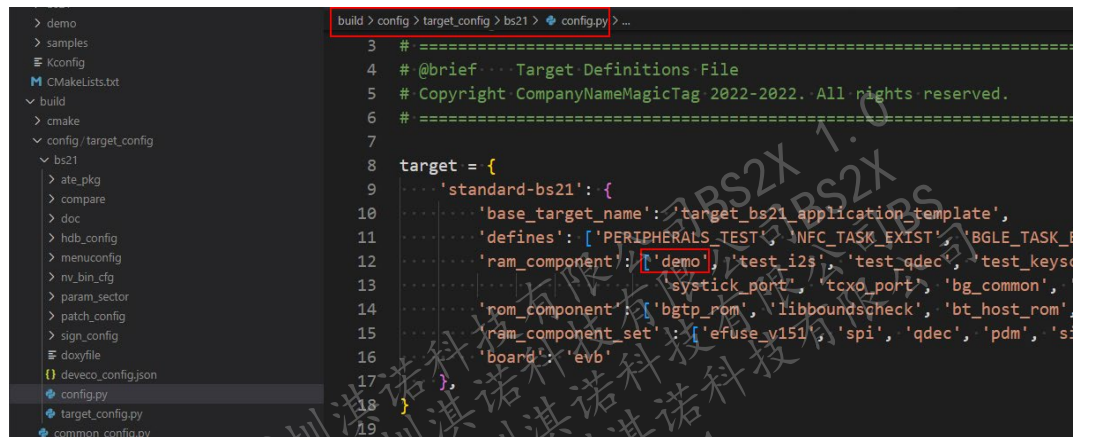
application > demo > CMakeList.txt

```
1  #=====
2  # @brief...cmake file
3  # Copyright (c) @CompanyNameMagicTag 2023-2024. All rights reserved.
4  #=====
5  set(COMPONENT_NAME "demo")
6
7  set(SOURCES
8      "${CMAKE_CURRENT_SOURCE_DIR}/demo.c"
9  )
10
11 set(PUBLIC_HEADER
12     "${CMAKE_CURRENT_SOURCE_DIR}"
13 )
14
15 set(PRIVATE_HEADER
16 )
17
18 set(PRIVATE_DEFINES
19 )
20
21 set(PUBLIC_DEFINES
22 )
23
24 # use this when you want to add ccflags like -include xxx
25 set(COMPONENT_PUBLIC_CCFLAGS
26 )
27
28 set(COMPONENT_CCFLAGS
29 )
30
31 set(WHOLE_LINK
32     true
33 )
34
35 set(MAIN_COMPONENT
36     false
37 )
38
39 build_component()
```

- 将组件的内的顶层 CMakeLists 添加到其上一层 CMakeLists 下，使构建系统能够执行到 Demo 组件的 CMake。



- 以 BS21 为例，将组件添加到 sdk\build\config\target_config\bs21\config.py 的编译目标内。



- 以 BS21 为例，在 sdk\application\bs21\bs21_acore\app_os_init.c 中 application\bs21\bs21_acore\app_os_init.c 的 app_main 线程调用 demo 代码。

```
application > bs21 > standard > C app_os_init.c > app_main
249
250 __attribute__((weak)) void app_main(void *unused)
251 {
252     ... UNUSED(unused);
253     #if (ENABLE_LOW_POWER == YES)
254     ... uapi_pm_add_sleep_veto(PM_VETO_ID_MCU);
255     #endif
256     ... demo_print();
257     ... hal_reboot_clear_history();
258     ... system_boot_reason_print();
259     ... system_boot_reason_process();
260     #if (USE_COMPRESS_LOG_INSTEAD_OF_SDT_LOG == NO)
261     ... log_exception_dump_reg_check();
262     #endif
```

- 单击 IDE build 编译代码，编译完成后烧录版本，观察串口打印。

说明

linux 下 sdk 的自定义模块添加方式与 windows 下一致。

5 开发调试工具应用

本次主要提供三款工具，分别用于系统调试维测、版本烧录、系统工程开发。

- DebugKits：提供日志打印解析、定制命令输入、内存导出解析等功能。
- BurnTool：提供版本烧录功能。
- DevEco：开发工具，集成一键式编译、烧录、单步调试以及内存查询，变量监控等功能。

5.1 基础开发工具应用介绍

5.2 基础维测工具应用介绍

5.3 基础烧录工具应用介绍

5.1 基础开发工具应用介绍

BS2X 系统解决方案 SDK 提供 DevEco IDE 工具用于 SDK 增量功能开发，提供如下功能：

工具链	当前支持功能
DevEco 南向 IDE	<ul style="list-style-type: none">• 支持工程管理。• 支持常用编辑功能。• 集成编译烧写调试。• 支持镜像分析、栈估算。• 支持常用调试功能。• 支持一键安装编译环境。

详情请参见《BS2X IDE 工具使用指南》。

5.2 基础维测工具应用介绍

BS2X 系统解决方案 SDK 提供 DebugKits 工具用于基础系统维测，提供如下功能：

- 支持日志解析/查看。
- 支持自定义参数配置（需适配板端 Diag 模块）。
- 支持自定义维测命令（需适配板端 Diag 模块）。
- 支持多种系统内存数据导入导出。
- 支持板端寄存器读写。
- 支持死机分析。

详情请参见《BS2XDebugKits 工具使用指南》。

5.3 基础烧录工具应用介绍

BS2X 系统解决方案 SDK 提供 BurnTool 工具用于基础版本烧录，提供如下功能：

- 支持多种烧写协议（基于脚本集成）。
- 支持 efuse 读写（当前不支持）。
- 支持 flash 导出。
- 支持一拖多。

详情请参见《BS2XV100 BurnTool 工具使用指南》。