

萤石IoT-SDK芯片适配指导手册

根据近期多款芯片的适配工作，总结出当前版本SDK适配芯片的最佳实践经验，希望能帮助其他同学能够快速、准确的评估适工作量以及快速完成适配验证工作。

1 移植条件评估

1.1 评估编译环境

1. 支持以工具链形式进行交叉编译
2. 支持ubuntu 18.04 TLS x64编译环境
3. 测试build_demo验证可用，见附件 [build_demo.zip](#)

1.2 评估资源

- 支持多任务：预留给SDK最少4个并发运行任务。
- 单品产品：240K ROM 和 40K RAM
- 网关类产品：260K ROM + 400K可擦写存储 和 440K RAM（按100个子设备估算）





1.3 评估接口

1.3.1 是否支持常用的libc接口

printf、memcpy、memset、strcpy、strncpy、sprintf、snprintf、malloc、free、calloc、zalloc、realloc

1.3.2 是否是常用操作系统

如果芯片使用的是Linux、FreeRTOS、RT-Thread其中的一种，不需要适配底层接口。否则芯片/模组厂需要帮适配以下抽象接口：

| 名称 | 修改日期 | 类型 | 大小 |
|---|-----------------|--------------|------|
|  hal_net_tcp.h | 2021/6/29 10:58 | C Header 源文件 | 3 KB |
|  hal_semaphore.h | 2021/6/29 10:58 | C Header 源文件 | 2 KB |
|  hal_thread.h | 2021/6/29 10:58 | C Header 源文件 | 3 KB |
|  hal_time.h | 2021/6/29 10:58 | C Header 源文件 | 2 KB |

1.4 评估时间

| 工作项 | 是 | 否 |
|----------------------------|-----|-----------|
| 支持以工具链形式进行交叉编译 | 0.5 | 5 |
| 支持ubuntu 18.04 TLS x64编译环境 | 0.5 | 2 |
| 测试build_demo验证可用 | 1 | 5 |
| 是否支持常用的libc接口 | 0 | 2 |
| 是否是常用操作系统 | 0 | 2 |
| 萤石验证模组基本功能 | 3 | 5（三方厂商验证） |

2 适配工作

2.1 安装工具链

编译服务器地址：10.1.14.113，没有账号找陈腾飞5开通

工具链安装路径：/opt/toolchain/

芯片SDK安装路径：/opt/

```
xurongjun@CI-Ezviz-Slave-14-113:/opt$ ls
bl_sdk  cmake-3.13.0  esp_sdk  l610  ln882x-sdk  sdk-ameba-v7.1d  toolchain  tr6260
xurongjun@CI-Ezviz-Slave-14-113:/opt$ cd toolchain/
xurongjun@CI-Ezviz-Slave-14-113:/opt/toolchain$ ls
asdk-6.5.0  esp32  esp8266  fulltan  gcc-arm-none-eabi-4.8.2  gcc-arm-none-eabi-6.2-2016q4  ln882x  nds32le-elf-mculib-v3  openwrt-linux-ramips-mt7688  tc32  toolchain.rar
```

2.2 更改shell脚本

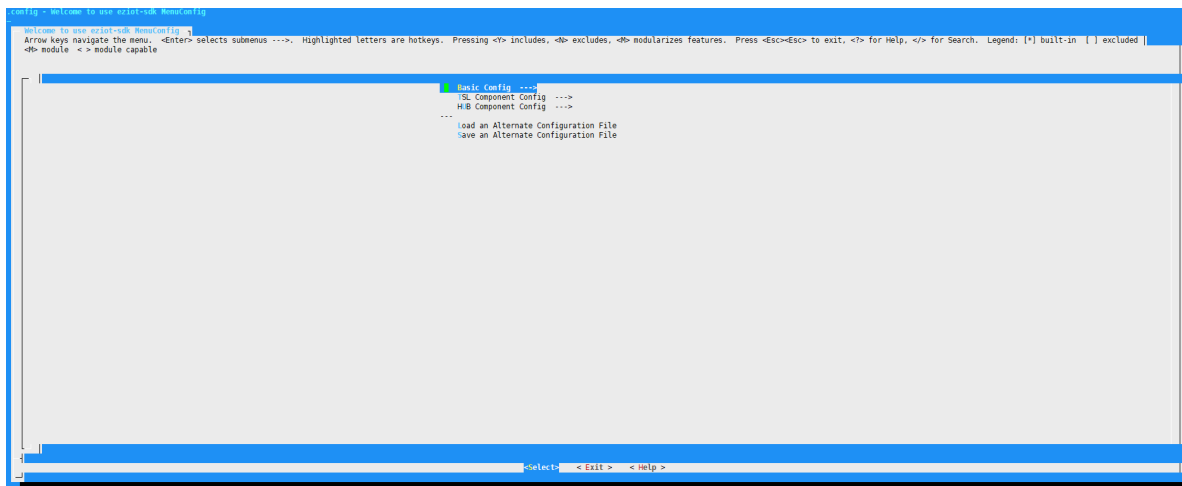
- 打开根目录build.sh文件。
- 在ln_ToolChain_all函数中添加工具链路径导出语句，此处的\$BUILD_PORT变量一般用芯片型号命名，后面的步骤中有几处目录会以此命名。

```
function ln_ToolChain_all(){
+   if [ "esp8266" == $BUILD_PORT ]; then
+       export IDF_PATH=/opt/esp_sdk/ESP8266_RTOS_SDK
+   elif [ "esp32" == $BUILD_PORT ]; then
+       export IDF_PATH=/opt/esp_sdk/esp-idf
+   elif [ "l610" == $BUILD_PORT ]; then
+       export PATH=${PATH}:/opt/l610/prebuilts/linux/gcc-arm-none-eabi/bin
+   elif [ "tr6260" == $BUILD_PORT ]; then
+       export PATH=${PATH}:/opt/toolchain/nds32le-elf-mculib-v3/bin
+   elif [ "rtl8710" == $BUILD_PORT ]; then
+       export PATH=${PATH}:/opt/toolchain/asdk-6.5.0/linux/newlib/bin
+   elif [ "mt7688" == $BUILD_PORT ]; then
+       export PATH=${PATH}:/opt/toolchain/openwrt-linux-ramips-mt7688/staging_dir/toolchain-mipsel_24kec+dsp-gcc-4.8-linaro_glibc-2.19/bin
+   elif [ "Rexense7628" == $BUILD_PORT ]; then
+       export PATH=${PATH}:/home/yangweifeng9/toolchain/1806/toolchain-mipsel_24ke_gcc-7.3.0_musl/bin
+   elif [ "ln882x" == $BUILD_PORT ]; then
+       export PATH=${PATH}:/opt/toolchain/ln882x/gcc-arm-none-eabi-10-2020-q4-major/bin
+   fi
}
```

2.3 新增配置目录

新增一个配置目录：cp -r ./build_conf/template ./build_conf/\$BUILD_PORT，同上\$BUILD_PORT一般为芯片型号。

配置在Kconfig GUI配置SDK功能：



2.4 更改编译配置项

2.4.1 编辑ToolChain.cmake

- 更改CMAKE_SYSTEM_NAME, linux操作系统填“Linux”, 否则填“Generic”。例:

```
SET(CMAKE_SYSTEM_NAME Generic)
```

- 更改工具链名字CMAKE_C_COMPILER, CMAKE_CXX_COMPILER, 这两项改成芯片对应的gcc、g++。例:

```
SET(CMAKE_C_COMPILER "arm-none-eabi-gcc")
SET(CMAKE_CXX_COMPILER "arm-none-eabi-g++")
```

- 设置芯片型号。例:

```
SET(SDK_BUILD_PORT "ln882x")
```

- 设置芯片操作系统, 不属于Linux、FreeRTOS则填写芯片型号。例:

```
SET(SDK_BUILD_OS "ln882x")
```

- 编辑主要编译的模块, 不需要则注释。例:

```
# SET(SDK_COMPONENT_AP_SUPPORT ON)           #AP配网
SET(SDK_COMPONENT_LINK_SUPPORT ON)           #基础连接
SET(SDK_COMPONENT_LOGGER_SUPPORT ON)
SET(SDK_COMPONENT_TSL_SUPPORT ON)            #物模型
# SET(SDK_COMPONENT_CONNECT_SUPPORT ON)       #局域网互联互通
SET(SDK_COMPONENT_HUB_SUPPORT ON)            #子设备管理
SET(SDK_COMPONENT_OTA_SUPPORT ON)            #升级
# SET(SDK_COMPONENT_TIME_SUPPORT ON)          #校时
```

- 设置芯片SDK依赖头文件目录, 不需要则注释。例:

```
SET(SDK_ADD_PRIV_INCLUDEDIRS
    /opt/ln882x-sdk/components/net/lwip-2.0.3/src/include
    /opt/ln882x-sdk/components/net/lwip-2.0.3/src/port
)
```

- 设置预编译宏，不需要则注释。例：

```
# SET(SDK_ADD_PRIV_PREMACRO "-DRT_USING_SAL")
```

- 设置私有编译参数，不需要则注释。例：

```
SET(SDK_ADD_PRIV_CFLAGS "-wno-frame-address -ffunction-sections -fdata-  
sections -fstrict-volatile-bitfields -nostdlib -march=rv32imfc -  
mabi=ilp32f")  
SET(SDK_ADD_PRIV_CXXFLAGS "-wno-frame-address -ffunction-sections -fdata-  
sections -fstrict-volatile-bitfields -nostdlib -march=rv32imfc -  
mabi=ilp32f")
```

2.4.2 编辑mcuconfig.h

对于一些没有操作系统的芯片，有可能并不支持标准libc函数，这种情况需要通过宏来替换成模组适配的接口，最后在编译可执行文件的时候完成链接。

```
#define printf      system_printf  
#define malloc      os_malloc  
#define free        os_free  
#define calloc      os_calloc  
#define zalloc      os_zalloc  
#define realloc      os_realloc  
#define strstr      os_strstr  
#define strchr      os_strchr  
#define strdup      os_strdup  
#define localtime   os_localtime  
#define time        os_time  
#define sprintf      os_sprintf  
#define snprintf     os_snprintf  
#define strdup       os_strdup  
#define vsnprintf    os_vsnprintf
```

2.5 适配跨平台接口

如芯片操作系统是Linux、FreeRTOS，不需要适配，否则需要在src\ez_iot_sdk\src\ez_hal目录下完成抽象接口适配。例：

| › branches › V1.5.0 › src › ez_iot_sdk › src › ez_hal | | | | |
|---|-----------------|-----|----|--|
| 名称 | 修改日期 | 类型 | 大小 | |
| bl602 | 2021/6/19 13:52 | 文件夹 | | |
| dna | 2021/6/19 13:52 | 文件夹 | | |
| freertos | 2021/6/19 13:52 | 文件夹 | | |
| linux | 2021/6/19 13:52 | 文件夹 | | |
| mt7688 | 2021/6/19 13:52 | 文件夹 | | |
| rtl8710 | 2021/6/19 15:58 | 文件夹 | | |

2.6 适配日志库

日志模块是移植的开源组件，当前版本SDK还没有将其配置项纳入Kconfig统一表，需要单独配置。如芯片操作系统是Linux、FreeRTOS，不需要适配，否则需要在src\ez_iot_sdk\src\component\logger\wrappers目录下完成适配。例：

| branches > V1.5.0 > src > ez_iot_sdk > src > component > logger > wrappers | | | | |
|--|-----------------|-----|----|--|
| 名称 | 修改日期 | 类型 | 大小 | |
| bl602 | 2021/6/19 13:52 | 文件夹 | | |
| dna | 2021/6/19 13:52 | 文件夹 | | |
| freertos | 2021/6/19 13:52 | 文件夹 | | |
| l610 | 2021/6/19 13:52 | 文件夹 | | |
| linux | 2021/6/19 13:52 | 文件夹 | | |
| ln882x | 2021/7/6 14:51 | 文件夹 | | |
| mt7688 | 2021/6/19 13:52 | 文件夹 | | |
| tr6260 | 2021/6/19 13:52 | 文件夹 | | |

2.7 编译

在根目录下执行build.sh \$BUILD_PORT，\$BUILD_PORT为芯片型号。

2.8 验证

这里分两种情况，一种是三方接入，验证工作由他们进行。另外一种是我们自研模组，可通过单元测试进行快速验证。

- 适配，在unit_test创建芯片的目录，完成入口函数和文件系统适配：

| < > branches > V1.5.0 > unit_test > src > port | | | | |
|--|-----------------|-----|----|--|
| 名称 | 修改日期 | 类型 | 大小 | |
| bl602_app | 2021/6/19 13:53 | 文件夹 | | |
| esp32 | 2021/6/19 13:53 | 文件夹 | | |
| esp8266 | 2021/6/19 13:53 | 文件夹 | | |
| linux | 2021/7/15 0:26 | 文件夹 | | |

- 运行：

@rongjun-VirtualBox: /media/sf_svn/IOT-SDK-V1.5.0/unit_test/bin/linux

```
[-----] [ testcase ] (ut_kvdb_blob_raw8K) finished
[-----] [ testcase ] (ut_kvdb_blob_raw16k_less) started
[ PASSED ] [ result ] testcase (ut_kvdb_blob_raw16k_less)
[-----] [ testcase ] (ut_kvdb_blob_raw16k_less) finished
[-----] [ testcase ] (ut_kvdb_init10M) started
[FlashDB][kv][data10M] All sector header is incorrect. Set it to default.

[ PASSED ] [ result ] testcase (ut_kvdb_init10M)
[-----] [ testcase ] (ut_kvdb_init10M) finished
[-----] [ testcase ] (ut_online_errcode) started
[FlashDB][kv][ez_kvdb] All sector header is incorrect. Set it to default.

[ PASSED ] [ result ] testcase (ut_online_errcode)
[-----] [ testcase ] (ut_online_errcode) finished
[-----] [ testcase ] (ut_online_access) started
[ PASSED ] [ result ] testcase (ut_online_access)
[-----] [ testcase ] (ut_online_access) finished
[-----] [ testcase ] (ut_online_restart) started
[ PASSED ] [ result ] testcase (ut_online_restart)
[-----] [ testcase ] (ut_online_restart) finished
[-----] [ testcase ] (ut_ota_errcode) started
[ PASSED ] [ result ] testcase (ut_ota_errcode)
[-----] [ testcase ] (ut_ota_errcode) finished
[-----] [ testcase ] (ut_tsl_errcode) started
[ PASSED ] [ result ] testcase (ut_tsl_errcode)
[-----] [ testcase ] (ut_tsl_errcode) finished
[-----] [ testcase ] (ut_tsl_prop_report_bool) started
[ PASSED ] [ result ] testcase (ut_tsl_prop_report_bool)
[-----] [ testcase ] (ut_tsl_prop_report_bool) finished
[-----] [ testcase ] (ut_tsl_prop_report_int) started
[ PASSED ] [ result ] testcase (ut_tsl_prop_report_int)
[-----] [ testcase ] (ut_tsl_prop_report_int) finished
[-----] [ testcase ] (ut_tsl_prop_report_float) started
[ PASSED ] [ result ] testcase (ut_tsl_prop_report_float)
[-----] [ testcase ] (ut_tsl_prop_report_float) finished
[-----] [ testcase ] (ut_tsl_prop_report_str) started
[ PASSED ] [ result ] testcase (ut_tsl_prop_report_str)
[-----] [ testcase ] (ut_tsl_prop_report_str) finished
[-----] [ testcase ] (ut_tsl_prop_report_jobj) started
[ PASSED ] [ result ] testcase (ut_tsl_prop_report_jobj)
[-----] [ testcase ] (ut_tsl_prop_report_jobj) finished
[-----] [ testcase ] (ut_tsl_prop_report_jarr) started
[ PASSED ] [ result ] testcase (ut_tsl_prop_report_jarr)
[-----] [ testcase ] (ut_tsl_prop_report_jarr) finished
[-----] [ testcase ] (ut_tsl_event_report_null) started
[ PASSED ] [ result ] testcase (ut_tsl_event_report_null)
[-----] [ testcase ] (ut_tsl_event_report_null) finished
[-----] [ testcase ] (ut_tsl_event_report_obj) started
[ PASSED ] [ result ] testcase (ut_tsl_event_report_obj)
[-----] [ testcase ] (ut_tsl_event_report_obj) finished
[-----] [ testcase ] (ut_tsl_event_report_obj2) started
```