
# ZS110A-UART-PRODUCT

发布 ***1.0.0***

**2018** 年 **11** 月 **02** 日

# 目录

CHAPTER 1

# 文档介绍

## 1.1 文档目的

通过示例介绍如何生成通过 uart 量产的固件。

## 1.2 术语说明

表 1.1: 术语说明

| 术语 | 说明 |
| --- | --- |
| uart prodcut | uart 量产，指通过 uart 将固件烧写到设备的存储介质上 |
| uart_adfu | dfu over uart for actions，uart 量产中使用的升级协议， |
| att | auto test tool，自动化测试工具 |
| att 固件 | 适用于 att 工具的固件 |

## 1.3 参考文档

- 无

## 1.4 版本历史

表 1.2: 版本历史

| 日期 | 版本 | 注释 | 作者 |
| --- | --- | --- | --- |
| 2018-10-15 | 1.0 | 初始版本 | ZS110A 项目组 |

CHAPTER 2

# uart 量产的限制说明

ATB110X ROM 中固化了 uart 量产的代码。固化代码对量产使用 UART 控制器/PIN 脚都有限制。请重点注意以下限制说明：

**注解:**

- 只支持通过 UART0(GPIO2/GPIO3) 进行量产。
- 对于 ATB1103，只支持将固件烧写到 IC 内部 MCP 的 spinor。
- 对于 ATB1109，只支持将固件烧写到连接到 GPIO14/15/16/17 的 spinor。

CHAPTER 3

# 如何生成 att 固件

uart_prodcut 工程 (\samples\uart_product) 基于 helloworld 示例生成了可以用 uart 量产的固件。

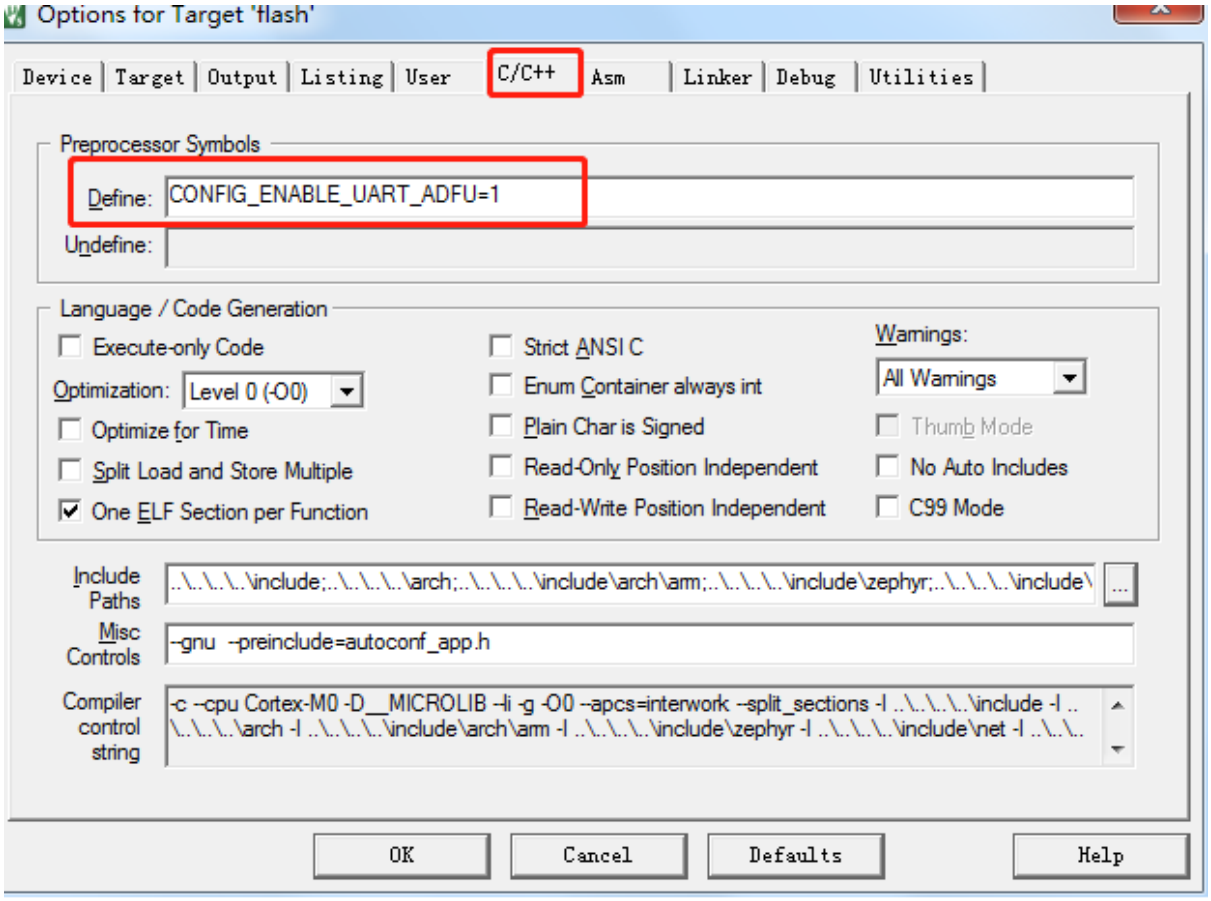
固件生成包括以下几个步骤:

- 修改 helloworld 工程，编译生成二进制文件
- 通过配置 firmware.xml 生成插入校验信息的二进制文件
- 通过配置 fw_maker.cfg、att.ini、upgrade.ini，生成临时固件
- 通过配置 config.txt，生成用于 att 工具的固件

## 3.1 编译生成二进制文件

对比 uart_prodcut 目录下的 helloworld 工程和 peripheral 目录下的 helloworld 工程 (\samples\peripheral\helloworld) 后, 发现两个工程之间只有一点区别：

uart_prodcut 目录下的 helloworld 工程中增加了宏 CONFIG_ENABLE_UART_ADFU 的定义。




添加这个宏定义之后，SystemInit 函数中会使能 check_uart_adfu()。该函数实现了“与 PC uart 握手并完成固件烧写”的功能。

```
__init_once_text void SystemInit(void)
{
        /* clear watchdog */
        sys_write32(0, WD_CTL);

#ifdef CONFIG_ENABLE_UART_ADFU
        check_uart_adfu();
#endif

        debug_uart_init();
}
```

## 3.2 为二进制文件添加校验信息

如果将 keil 工程编译后的 bin 文件烧写到设备的 spinor 上，设备重新上电后会因为校验出错而引导失败。下一步就需要给二进制文件添加校验信息。

运行 build_firmware.py 脚本，会给 firmware.xml 指定的 bin 文件添加上校验信息。

```
python -B ..\%scripts_path%\build_firmware.py -c .\firmware.xml -
↪b zs110a -ota 0
```

firmware.xml 示例如下：

```
<?xml version="1.0" encoding="utf-8"?>
<firmware>
        <descirption>Firmware layout for ZS110A</descirption>
        <disk_size>0x80000</disk_size>

        <partitions>
                <partition>
                        <type>SYSTEM</type>
                        <name>fw0_app</name>
                        <file>hello.bin</file>
                        <address>0x0</address>
                        <enable_crc>false</enable_crc>
                        <enable_randomize>false</enable_
↪randomize>
                        <enable_ota>true</enable_ota>
                        <enable_raw>true</enable_raw>
                        <enable_dfu>true</enable_dfu>
                        <fw_id>0</fw_id>
                </partition>
        </partitions>
</firmware>
```

**注解：** 只需要修改 xml 中的 name 和 address 字段，其他字段不用修改。

## 3.3 生成临时固件

有了带校验信息的 bin 文件，下一步就要将 bin 文件和量产辅助文件打包成 fw 文件。

生成临时固件的命令如下：

```
%maker_path%\Maker.exe -c .\fw_maker.cfg -o .\outdir\zs110a.fw -mf
```

fw_maker.cfg 示例如下：

```
SETPATH = "..\..\scripts\support\actions\prebuilt\basefw";
BASEFILE = "base.fw";

SETPATH=".";
FWIM="att.ini";
FWIM="upgrade.ini";
```

(continues on next page)

(续上页)

```
SETPATH=".\outdir\bin";
FWIM="hello.bin";
```

fw_maker.cfg 中包含的文件说明：

- base.fw: 量产的辅助文件。

**注解：** 不能删除、修改 base.fw。

- att.ini: 握手后 uart 参数配置等

```
//每包 payload 最大长度，与 UART 串口处理能力有关
payload = 2048
baudrate = 2000000
//获取设备端 uart_adfu 协议版本
version
```

- upgrade.ini: 和烧写 spinor 相关的操作，包括初始化 spi 控制器、擦除 spinor、烧写 spinor。

```
//初始化 spi0 控制器，命令格式：sinit 控制器编号 (目前只支持 =0)
sinit 0

//擦除 spinor 指定区域，命令格式：serase start_addr len
serase 0x0 0x10000

//烧写文件到 spinor 指定区域，命令格式：swrite start_addr storage_type(目前只支持 =0) file_name
swrite 0x0 0 hello.bin
```

**注解：** 每烧写一个文件需要一组命令 (serase+swrite)。多个文件就需要多组命令。

- hello.bin: 烧写到 spinor 的文件名。支持多个文件，根据实际情况配置。

## 3.4 生成 att 固件

有了临时固件，下一步就是生成 att 固件。

生成 att 固件的命令如下：

```
%att_maker_path%\att_maker.exe zs110a_atf.fw 4 acttest.ap config.xml␣
↪zs110a.fw config.txt
```

命令说明如下：

命令格式：

att_maker.exe output_filename argument_counts arg1 arg2 arg3 arg4 ···

参数说明：* acttest.ap: 必备项。不能删除

- config.xml:att 测试项定义。不能修改。

  目前支持三个测试项：

  1. 量产固件；
  2. 修改蓝牙地址；
  3. 测试频偏。

- zs110a.fw: 前一个步骤生成的固件
- config.txt：att 测试项的配置，和 config.xml 对应。默认不需要修改。必要时可以使用“炬芯工具集/自动化测试工具 (BLE)/脚本修改工具”进行配置。

# List of Figures

# List of Tables