**本文档基于detools 开源算法，可用于测试对二进制文件的差分升级测试。**

**Delta firmware updates:**

首先用VSCode或者命令行编译出两个不同的binary 升级文件，其中一个为源文件（source.bin）,另一个为目的文件（target.bin）, 并基于这两个文件生成差分包（patch.bin）;然后将差分包烧入指定的flash区（secondary slot）。当MCU重启后会检查此块flash区中差分文件是否有效，验证签名等信息，然后再比较差分文件的头八个字节。新生成的差分包会在文件头部插入八个字节的标记（前四个字节为字符串“NEWP”,后四个字节为patch size）, 后面紧跟着源文件的32 Bytes差分值，只有这些信息全部匹配上才会执行差分升级。差分包apply后会改写前四个字节，这样MCU就不会重复执行apply操作了。执行apply patch.bin的操作时，会将旧的image（source.bin）和patch.bin结合生成新的image并保存在flash slot1区域, 设备重启完成升级，整个过程支持掉电保护。

**Requirements:**

**1 .初次测试时请先从git hub 下载最新的boot代码，并替换自己SDK目录下的boot文件夹（C:\NCS\_SDK\v2.1.0\bootloader\mcuboot\boot）。 Git hub 链接如下：**

[**https://github.com/Noy0908/delta\_mcuboot.git**](https://github.com/Noy0908/delta_mcuboot.git) **branch: single\_slot**

**2. 安装detool工具，在python环境下执行 “pip install –user detools”指令。**

**3. 将zephyr/scripts/patch.exe 文件拷贝到应用工程根目录下，生成差分文件是双击该文件即可。**

**MCU 端测试步骤：**

1. 生成差分文件。差分文件是由原始文件和目标文件的差值比较而来。原始文件是原始工程编译后得到的：app\_signed.hex，将此文件转成：app\_signed.bin（转换命令：arm-none-eabi-objcopy --input-target=ihex --output-target=binary --gap-fill=0xff app\_signed.hex app\_signed.bin），然后改名为source\_xxx.bin,比如source\_1.0.0.bin; 或者直接用J-Flash工具将其另存为source\_xxx.bin文件。修改原始工程，再次编译，得到目标文件：app\_update.bin，将其改名为target\_xxx.bin，比如target\_2.0.0.bin，将source\_xxx.bin和target\_xxx.bin拷贝到binaries\signed\_images。然后执行scripts/patch.exe，将在目录binaries/patches自动生成差分文件，请使用差分文件signed\_patch.bin
2. 烧录bootloader文件，可直接用VSCode烧录编译好的工程，因为已经使能了MCUBoot宏，所以会自动将MCUBoot一起烧录；
3. 在工程目录下双击patch.exe生成差分文件，差分文件名字会明确指明原版本号和目标版本号，目前支持签名验证，因此请选择signed的patch文件，如signed\_patch\_from\_1.2.01\_to\_3.1.0.bin, 请仔细检查source 和target是否正确。
4. 烧写差分文件到指定的flash区域，建议secondary slot区域，支持4G/WiFi/蓝牙/NFC/Zigbee/NB-IoT等OTA方式，也可以通过有线方式进行升级，比如通过UART，USB或者SPI通信接口来烧写patch 文件。烧写完成后就可以正常执行MCUBoot后续步骤实现delta DFU 了。

**PC 端测试：**

**Requirements:**

1. PC 端测试时需要安装make 工具，

链接：https://pan.baidu.com/s/1t93DDxYdGzBD1itOSJd5IA

提取码：dota

1. 初次测试时需要确保所需工具安装完成，可执行 make tools指令自动安装所需工具，主要工具有 detools，pyocd, pynrfjprog 等

工程路径中的Makefile文件已集成各种测试命令，具体步骤如下；

1. VSCode 先后编译生成两个不同的app\_update.bin文件， 并将其拷入工程目录下的binaries\signed\_images文件夹，将其中一个改为source.bin即源文件，另一个改为target.bin即目标文件
2. 生成差分包patch.bin, 输入指令 make create-patch. 注意此时生成的patch,bin文件头部不需要插入八字结的标志位，否则执行apply操作的时候会报错。此时需要注释掉Makefile文件的create-patch指令最后面的一行脚本。

A picture containing graphical user interface

Description automatically generated

1. 执行apply操作， 输入指令 make apply-patch, 此时PC会将souce.bin和patch.bin结合生成新的image保存在binaries\flash\_dumps\target.bin文件中。
2. 烧入新生成的image文件， 输入命令 make flash-apply-target， 此时查看MCU的输出是否和target.bin一致。 也可以直接用软件对比新生成的image和target.bin是否一致。