**7 片内flash的空中升级**

7.1 使用片内flash空中升级的要求和约束条件

当使用CC2640F128内部的flash时，flash的开始的6个page也就是开始的24KB是预留给flash interrupt vectors，BIM和永久保留的OAD Target App。OAD Target App中用到了一部分固化在ROM中的TI-RTOS的代码。BIM和OAD Target App还用到了flash的page 31中的一些存储空间。Page 31的起始地址是0x1F000, CCFG也用到了这一部分空间。BIM和OAD Target App都是被设计为不可升级的。

OAD image默认是放在从0x6000到0xEFFF的9个flash page 中一共36KB的空间内。因为Page 0是不能升级的，所以image中的代码必须在flash中包含它自己的instance。而不是依赖ROM中的TI-RTOS来实现。OAD image也和其他的代码共享CCFG。CCFG是不能通过OAD来升级。

OAD Target和OAD image共享相同的RAM。但是在每次器件复位后只有一个程序可以使用这些共享的RAM。OAD image必须是一个完整的用户代码。它要可以独立于固化的OAD Target运行。

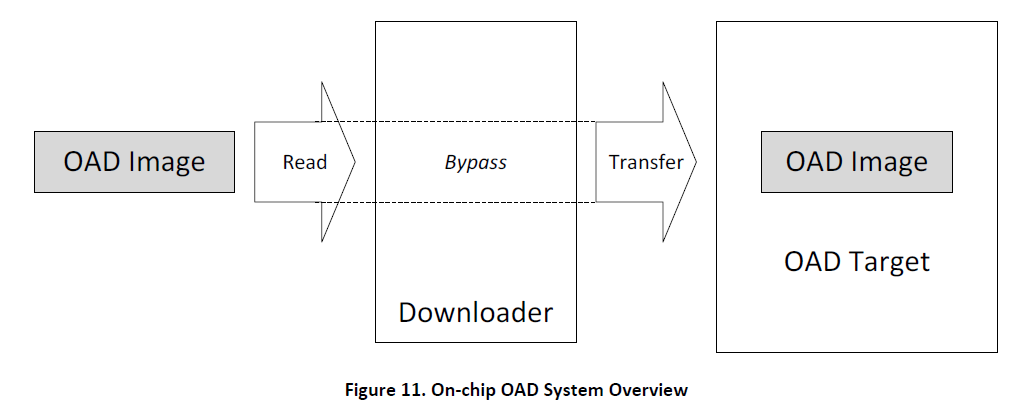
BLE stack默认在从0xF000到0x1DFFF的15个flash的page中共60KB。 SNV在从0x1E000到0x1EFFF的一个flash的page中共4KB。如果OAD image太大放不到它自己的36KB的空间内，可以考虑裁剪BLE stack的部分功能来腾出空间放OAD image。这一部分将在7.5.5章进一步讨论。

OAD Target App即OAD image A与OAD image B共用同一个BLE stack。BLE stack是不能OAD的。

Flash的第一页和最后一页不能Erase的，如果这样做使芯片进入一个不安全的状态，如果这时候发生复位，芯片会“变砖”。一旦芯片“变砖”没有JTAG或是serial boot loader是无法恢复的。

7.2 系统概述

在Figure 11中所示的OAD系统中，系统由OAD image，Downloader和OAD Target组成。



OAD Target就是我们CC2650。OAD image就是你要下载到OAD Target中的用户代码。OAD image格式应该是intel hex格式的.hex文件。Downloader可以由运行在PC机上的“BLE Device Monitor”和运行程序”HostTest application”的CC2540 dongle来组成。Downloader也可以由任何专用的应用程序或者手机上APP来实现，只要它实现了7.3章所描述的OAD Profile客户端。

TI专用的OAD Profile定义了OAD Target与Downloader之间的通讯。这个Profile里面定义了一系列相应的characteristics来支持image identification, image block request/response and image count setup等功能。

7.3 OAD Profile

这个Profile是为了给客户提供一种简单的定制化的OAD profile而设计的。对于片内flash OAD来说这个Profile最基本的功能有：根据image header的要求来接受OAD请求；将OAD image储存到OAD target device片内的flash中；如果下载成功复位芯片，使下载的用户的应用代码可以由BIM运行。Downloader是client的角色，而OAD target device是server的角色。

7.3.1 依赖关系

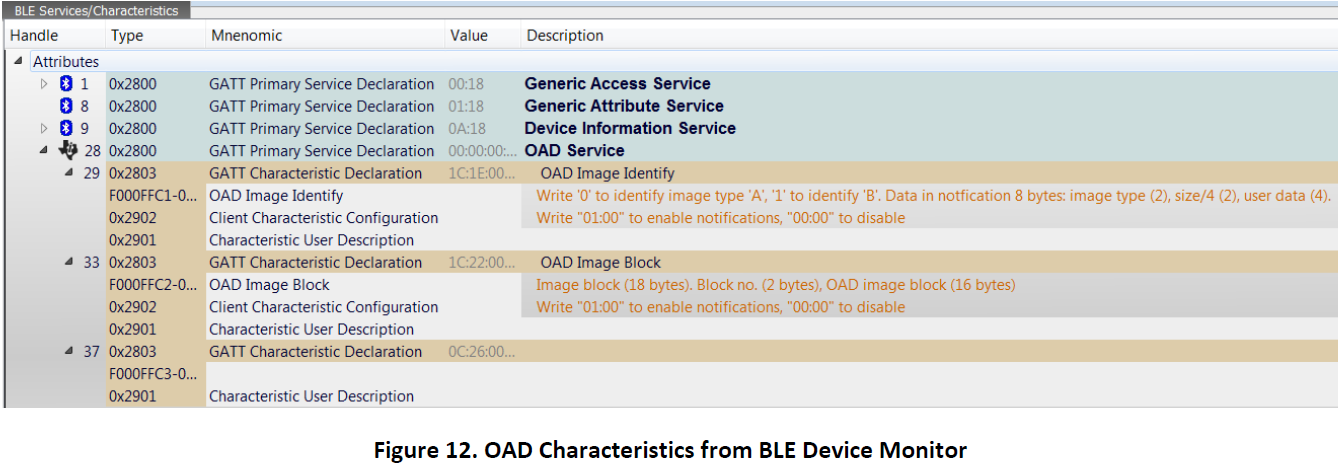
OAD profile根据最终客户的应用来决定连接参数和发起OAD的条件。

7.3.2 Message

这里使用“无回复写入”作为从downloader到OAD target device的默认的消息类型。这种类型的消息可以减少代码体积并尽可能的增加传输速率。之所以这样决定是应为,如果用“有回复写入”类型的消息如GATT Notification需要调用函数GATT\_ClientInit()，这样会明显的增加代码的体积。在噪声或者RF信号很差的环境中，这种“无回复写入”类型的GATT消息可能无法成功的传输整个image。这就需要在软件中加入超时和重传机制。因此downloader将被初始化成client, “有回复写入”的Notification被选为默认的从OAD target device 到downloader的消息类型。

7.3.3 Characteristics

OAD的profile只有三个characteristics. Downloader负责发现OAD target device上的这三个characteristics的handle。这些characteristics的结构如下图所示。

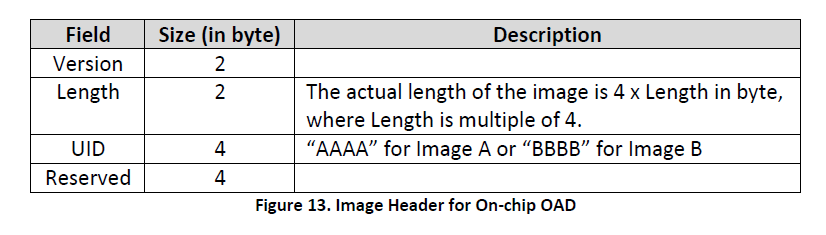


7.3.3.1 OAD image Identify Characteristics

Image识别 characteristics被用来交换片内flash OAD中的嵌入到OAD image中的image header信息。或者被用来交换片外flash OAD中Downloader为OAD image生成的元数据。OAD Target用这个元数据来决定是否要发起OAD。“01:00”应该写到这个characteristics的 Client Characteristic Configuration后文统一简称CCC来使能notification。

7.3.3.1.1 Image Header

对于片内OAD，Image Header位于OAD Image的开始的部分。它开始于第5个字节，或者CRC和CRC Shadow的后面。 CRC和CRC Shadow总共4个字节并且位于OAD Image的开头。Downloader从OAD Image中获取Image Header中的信息，并写入OAD Image Identify characteristic的数据中。Image header的具体描述见Figure 13。



7.3.3.2 OAD image Block Characteristics

OAD image Block Characteristics用来请求和传输OAD image的数据。当这个Characteristics 的CCC被写入“01:00”时使能notification，写入“00:00”关闭notification。

7.3.3.3 OAD image Count Characteristics

OAD image Count Characteristics用来设置需要下载的OAD image的数量。它是专门用在片外OAD的中的，默认值是1。

7.3.4 发起片上OAD的步骤

在建立一个新连接后，首先跟新connect interval来满足快速 OAD， 并且使能OAD image Block Characteristics和OAD image Identify Characteristics的notification。之后Downloader给OAD Target的OAD image Identify Characteristics写入数据。写入的数据来自于OAD Image的Image Header。

一旦收到downloader写入到image Identify Characteristics的数据，OAD target 将把收到的OAD image的信息和自己正在运行的image做比较。默认情况下只有image size和version number用来检查和决定是否接受新image升级。image size和version number用来决定image是image A还是image B。

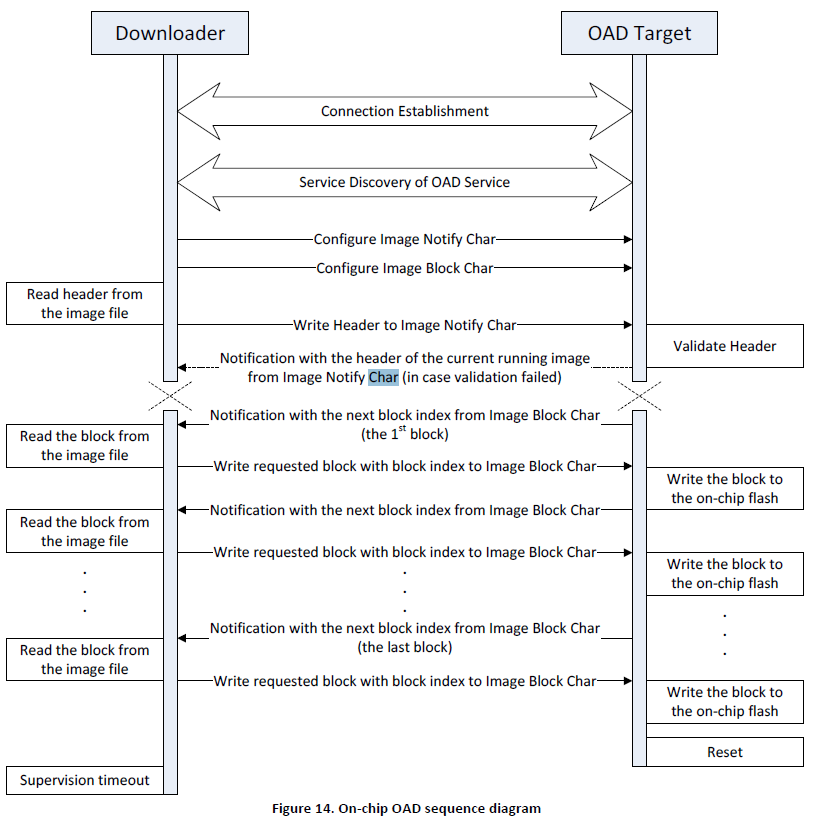
如果OAD Target决定接受OAD image，OAD target将向downloader发送一个OAD image Block Characteristics中的notify来向downloader请求新image的第一个数据块。如果OAD Target发现new image不符合开始OAD的标准。它将发送一个OAD image Identify Characteristics中的notify来告知downloader。OAD Target 发送的notify包含自己的Image Header用来表示拒绝。在这种情况下OAD的进程将在Figure 14中虚线X的地方终止。

7.3.5 Image Block Transfers

Image Block Characteristics可以让两个设备一次请求和相应一个数据块传输。Image 数据块的大小定义为16 byte, 见oad.h中的OAD\_BLOCK\_SIZE。O5AD Target向Image Block Characteristics发送一个包含正确的数据块的索引的notify，以此来向downloader请求一个image的数据块。然后Downloader向Image Block Characteristics中写入image的数据块来相应OAD target的请求。当OAD Target准备好消化下一个image数据块的时候，它将向Image Block Characteristics发送包含需要的数据块的索引的notify。Downloader随后相应这个请求。

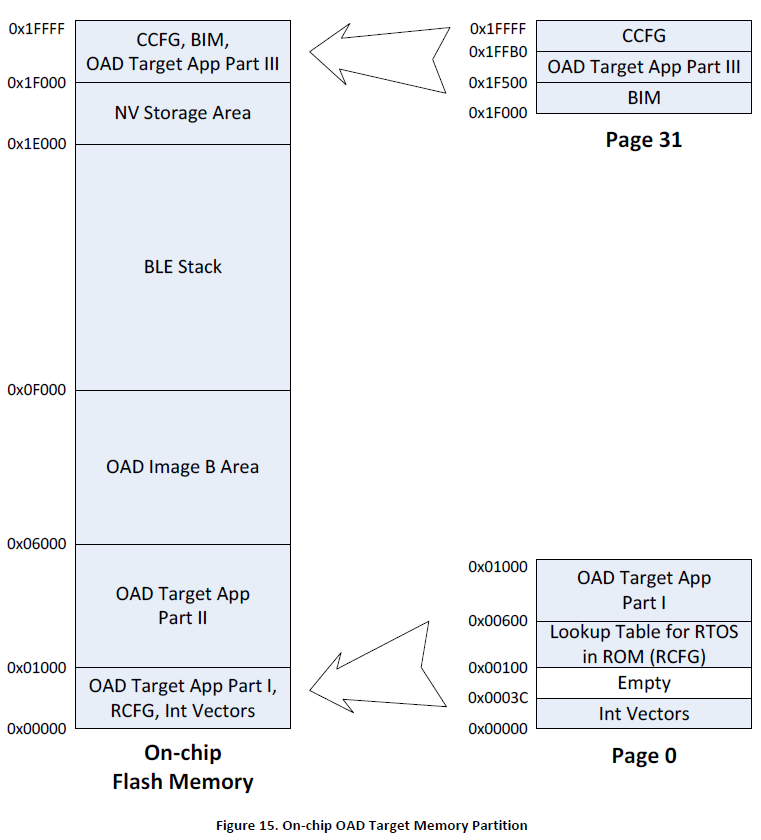
7.3.6 Completion of the On-chip OAD Process

当OAD Target收到最后一个数据块后，它将计算整个储存的OAD image的CRC。OAD Target以此来检验image是否被正常接收并正确存储到flash中。然后OAD Target将废除自己的image并且reset使BIM可以运行新的image替代原来的image。然后烦恼就到了downloader那里。在OAD target检验和实例化的过程中，downloader将会遭遇到很多次的BLE连接丢失。Downloader会重新scanning，重新建立连接并检验新的image是否真的在运行。



7.3 OAD Target

在片内OAD时，OAD Target的flash中有Interrupt Vector, RCFG,永久保留的OAD Target App(Image A),Image B, BLE stack, NV Storage Area, BIM, 和CCFG。如Figure 15所示。

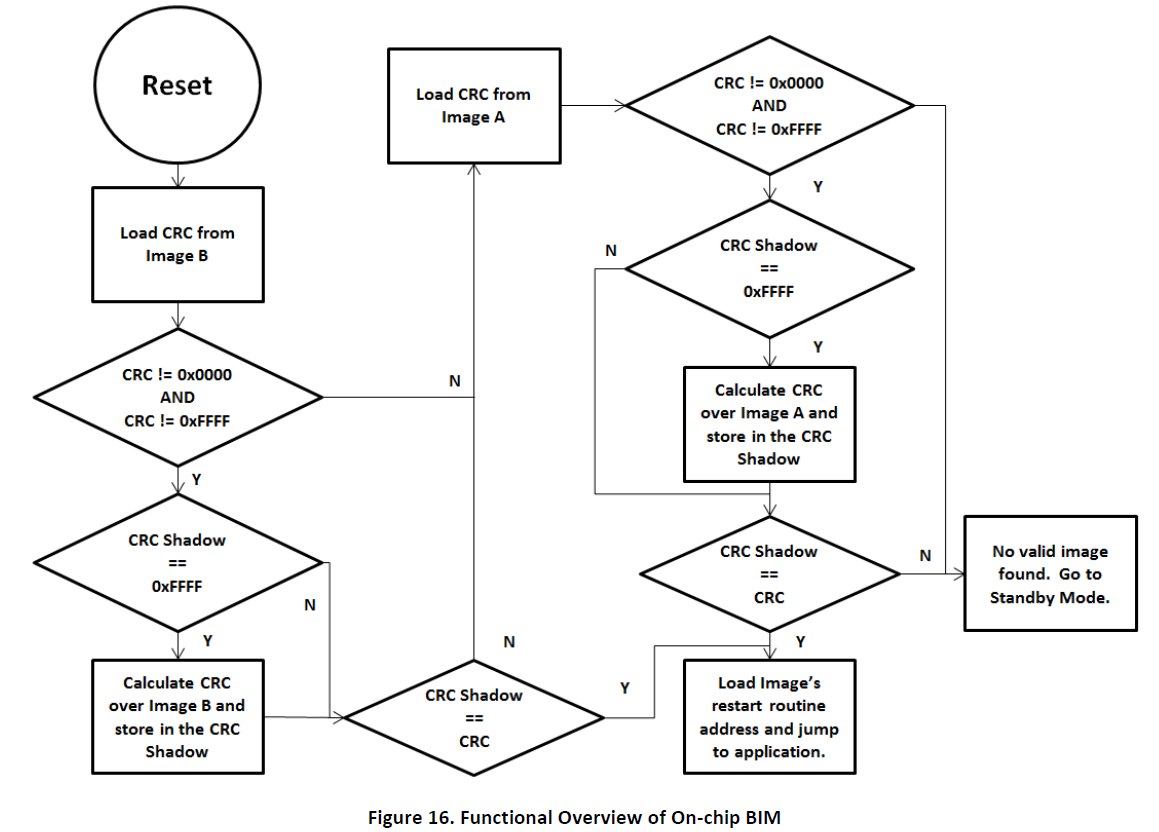


BIM的设计提供了一种灵活性，可以有两个正确的Image随时准备运行。BIM决定运行哪个image。只有image B可供下载。OAD Target APP也就是Image A是永久保留的。它依赖于在flash的第一页和最后一页的代码。如果在掉电的时候image A 被Erase的话，整个设备有可能无法启动。Image A的唯一的作用是实现BLE stack和OAD profile。永久保存image A的好处是，可以增大image B可以使用的flash空间。客户代码image B的开发者不需要考虑OAD profile的实现。Image B中唯一需要涉及OAD功能的只有需要提供一个Figure 13所描述的合法的OAD Header。合法的OAD Header需要用到如7.6.2章所述的OAD Reset Service。Image A和Image B需要使用完全相同的编译的BLE stack，而且BLE stack必须链接到相同的地址。

7.4.2 BIM for On-chip OAD

OAD方案需要永久保留的boot代码，并且需要BIM来提供一种失效安全的机制以决定Image是否可以运行。当BIM找到了一个正确Image, BIM跳转到那里，Image开始运行。Image A和Image B中的一个必须实现独有的TI OAD Profile。默认的情况下是由Image A来实现的。当Image A通过OAD Profile下载到了一个新的Image，执行一个system reset来返回BIM。然后在BIM中检查下载的Image是否正确，并在通过检查后执行新的Image。

BIM和CCFG, OAD Target一起占用flash的最后一个page。BIM使用flash的第一个Page中的interrupt vectors。Flash第一个page中的Reset Interrupt Vector调用BIM来保证复位后BIM对器件的控制权。



作为flash interrupt vector的永久拥有者，reset vector首先指向BIM以提供一种失效安全机制。BIM把硬件置于一种安全的状态，并且在读取Image A和Image B的header后采取适当的行动。

默认BIM会优先运行Image B。只有在需要升级 image B或者没有正确的image B的时候才运行image A。当一个Image无法运行的时候BIM会去检查另外一个。当两个image都不正确的时候，BIM会让器件进入到一种低功耗的待机模式。因为Image A也就是OAD Target App从来不需要被Erase，所以这种情况看似不可能发生。默认Image A是不进行CRC校验的。这是应为OAD Target App被认为是固化的image。BIM只是通过读取IAR生成的checksum来判断image A是否存在，并不去计算CRC。

为了检查一个image是否正确，BIM将去查询一个4个byte的CRC和CRC-shadow码。其中两个Byte是编译时生成的CRC，另外两个byte的CRC-shadow是BIM计算得出的。如果这两个值是一致的，image被立刻执行。如果CRC不是0也不是Erase后的0xFFFF，并且CRC-shadow是Erase后的0xFFFF，BIM会计算不包括这四个Byte的整个image的CRC，计算的结果会去和CRC output比较。根据比较的结果决定是否可以执行这个image。

7.5 Building On-chip OAD

7.5.1 Building BIM

启动代码通过IAR IDE分别编译器编译、调试和烧写。有一个和OAD Target App一起的工程。还有一个独立的工程在下面的路径：

<INSTALL\_DIR>\Projects\ble\util\BIM\CC26xx

在CC26xx中flash在写入代码前需要先Erase。这使得BIM的下载变的复杂，因为BIM和OAD Target Application 的代码共享flash的最后一个page。解决的办法是先编译生成一个BIM的hex，再把这个hex和OAD Target的hex按照7.5.3描述的方法合并。

默认BIM定义符号FEATURE\_FIXED\_IMAGE来跳过对Image A的CRC。我们不推荐去掉这个符号，除非用户需要改变OAD的架构来要求CRC校验以验证Image A的正确性。Image B总是需要进行CRC校验来保证安全和完整性。

7.5.2 Building the BLE Stack

OAD Target App也就是image A和image B共享BLE stack image。Image A不能通过OAD去升级或者修改。需要强调的是开发和测试image B一定要使用和下载到器件中的image相同的BLE stack。和BIM一样，BLE stack编译生成的hex被合并到OAD Target App。所以BIM，BLE stack和OAD Target App三个工程可以合并到一个hex文件中，一次下载到器件中。为了减小BLE stack image的体积，NV memory存储空间只有一个flash page。它有两个page的NV系统的全部功能，只有两点与两个page的NV系统不一样。第一:系统预留了cache memory,它不能被用户禁用或者修改；第二:在NV memory正在压缩的过程中，如果丢失电源，可能会丢掉NV memory中的所有数据。为了给OAD image也就是image B预留空间，两页的NV模块太大放不下。BLE stack被要求使用功能最简化的库文件CC2640\_BLE\_peri\_HL\_CL\_FlashROM.a。默认OAD Target相应的CC2640 stack project已经按上面的要求进行了配置。所以CC2640 stack project输出的image应当作为整个OAD系统共享的stack。

7.5.3 Building the OAD Target Application

OAD Target App是永久保留的image，用来把一个image通过BLE下载到Image B的区域。这个工程位于路径“<INSTALL\_DIR>/Projects/ble/OADTarget/CC26xx/IAR”。为了简单，OAD Target App从flash的第一页开始，在自己的RCFG (RTOS in ROM Configuration Table) 后面。在BIM中OAD Target App是image A。所以当image B正确时，默认运行下载的image B。在这个工程的post build instruction中会执行一个python脚本，将OAD Target Image, BLE stack Image, BIM合并到一个.hex文件中。



**Figure 17. SmartRF Flash Programmer 2**

使用Flash Programmer 2下载hex文件到CC2640。如果没有安装这个工具，可以到这个链接去下载：<http://www.ti.com/tool/flash-programmer>。在主选项卡中点击按钮“Browse”，找到合并后的hex文件并选择。点击Figure 17中左边窗口“Connected devices”下面的“Refresh”按钮。如果你的CC26xx已经连接，它应该显示在窗口“Connected devices”。如图Figure 17点击CC2650，选中要下载的器件，被选中的选项会高亮显示。在下部左侧的窗口中勾选“Erase” 和“All unprotected Pages”。并勾选它右侧窗口中的 “Program” 和“Entire source file”。点击“play”按钮下载程序。

7.5.4 Building OAD Image B

虽然带有OAD功能的image是与BIM分开编译和链接的，但是它必须永远遵循BIM所要求的边界限制，以及BIM相关的外部接口的位置限制（比如CRC和Image Header）。OAD Target App image也依赖于BIM。因为只有BIM会在flash的第一页最开始的地方放一个中断向量表。这个位置的中断向量表保证了器件的正常使用。创建image B的例程包含在“SimpleBLEPeripheral”的工程的“FlashOnly\_OAD\_ImgB”中。我们通过下列的步骤可以把任意已经存在的工程转换成可以通过OAD下载的image B。注意即使提前选择了配置“FlashOnly\_OAD\_ImgB”，step VII仍然需要手动设置，来改变RTOS在flash中的配置。

I. 选择 *Project*→*Options*→*C/C++ Compiler→Preprocessor*→*Defined symbols* 添加下面的定义:

ICALL\_STACK0\_ADDR=0xF000

FEATURE\_OAD\_ONCHIP

IMAGE\_INVALIDATE

HAL\_IMAGE\_B

添加下面的内容到 “Additional include directories”: $PROJ\_DIR$/../../../../../../../Projects/ble/Profiles/OAD/CC26xx

II. This step is for OAD Manager which is out of scope of this document. Select *Project→Options→Build Actions→Post-build command line* and paste the line below to build the binary image, assuming the configuration name is ‘FlashOnly\_OAD\_ImgB’ and the output hex file name is ‘SBP\_OAD\_ImgB.hex’: python "C:\Python27\Scripts\hex2bin.py" -r "6000:EFFF" "$PROJ\_DIR$\FlashOnly\_OAD\_ImgB\Exe\SBP\_OAD\_ImgB.hex" "$PROJ\_DIR$\FlashOnly\_OAD\_ImgB\Exe\OADbin.bin"

III. Select *Project→Options→Linker→Config*. Paste the following line to ‘Linker configuration file’:

$PROJ\_DIR$\..\..\..\..\..\common\cc26xx\IAR\cc26xx\_ble\_app\_oad.icf

And add the following symbol to ‘Configuration file symbol definitions’: FLASH\_ONLY\_BUILD=1

IV. Setup the Linker for an image’s flash and RAM usage. By default the linker guarantees 9 flash pages, or 36KB, to the OAD image starting at 0x6000. It is generally recommended that the values for Image B starting address are not changed from the default settings unless OAD Target App needs to be modified in its size.

V. Select *Project→Options→Linker→Checksum* to configure a CRC16 calculation over the application image. Make sure that the start address does not include the CRC and CRC Shadow locations and that the checksum ends at the last address of the specified image region of the OAD Target. By default, then, the CRC covers the range 0x6004 through 0xEFFF. Make sure that the Algorithm uses CRC16 with 0x1021. If this value is set to any other value than 0x1021, it can be modified by setting the algorithm first to CRC polynomial, and then setting the Algorithm back to CRC16.