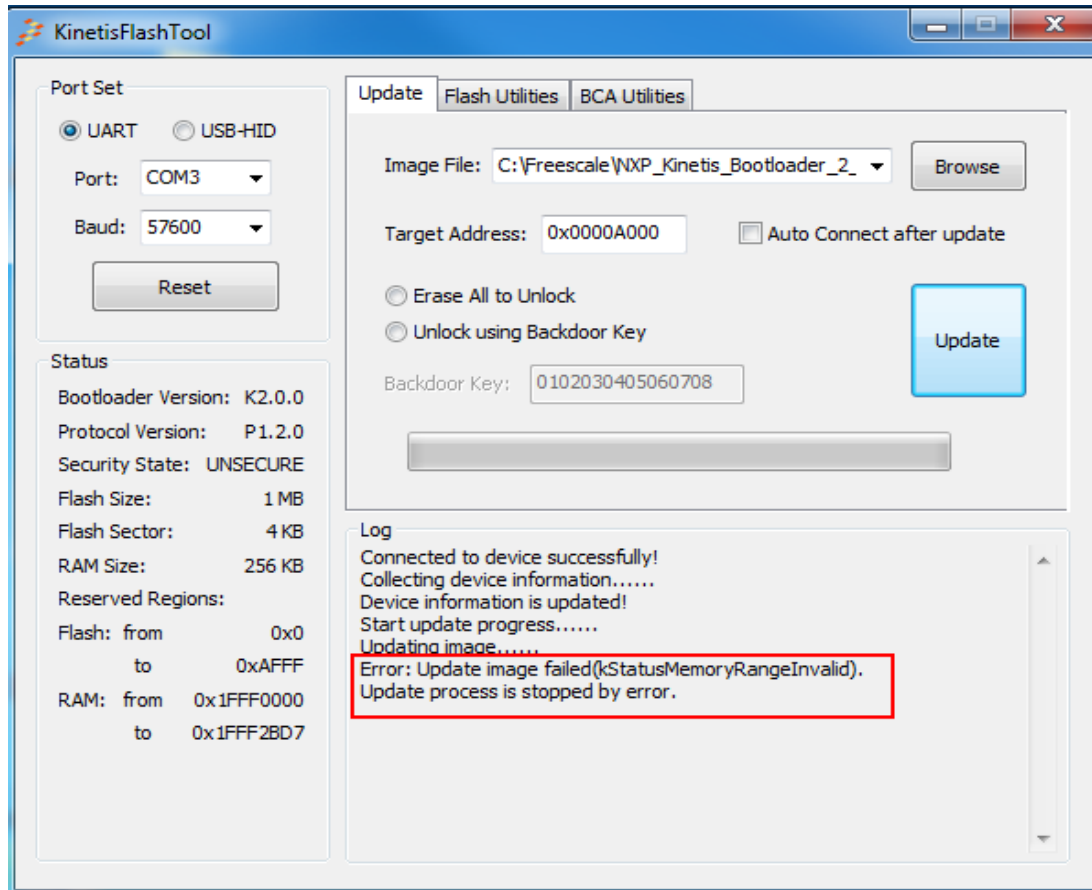


Using Kinetis Bootloader v2 on KDS in Debug Mode

The Kinetis Bootloader(KBOOT) v2 is now available on www.nxp.com/kboot webpage, it supports building bootloader image on KDS v3.2 IDE. While when using KDS build the bootloader image, there is one point need to be paid attention: the image size in debug mode build is over 0xa000(the 0xa000 is user's application start address). That is to say , **if we download the debug build image to mcu, can't use it** . So this doc provides the correct way to use the KBOOT on KDS, the problem shows in FRDM-K64 and FRDM-K22 boards, on other chips , if they also have the out of range problem, the correct method is the same .

1. Error when using debug build image

When download the debug build image , the board can be connected to PC successfully, while when update user's application , it has error:

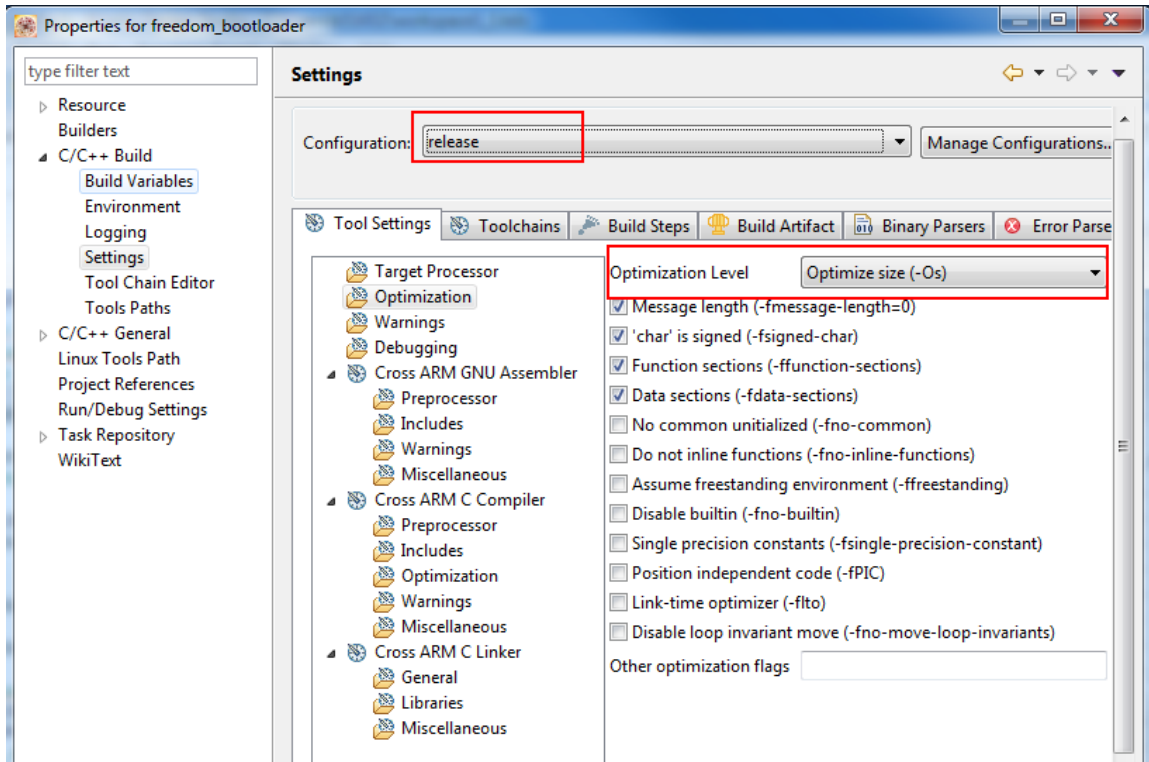


2. Error Root Cause

2.1. For debug version, the image size is beyond 0xa000, so when update the user's app, the KinetisFlashTool GUI shows" kStatus Memory Range Invalid".

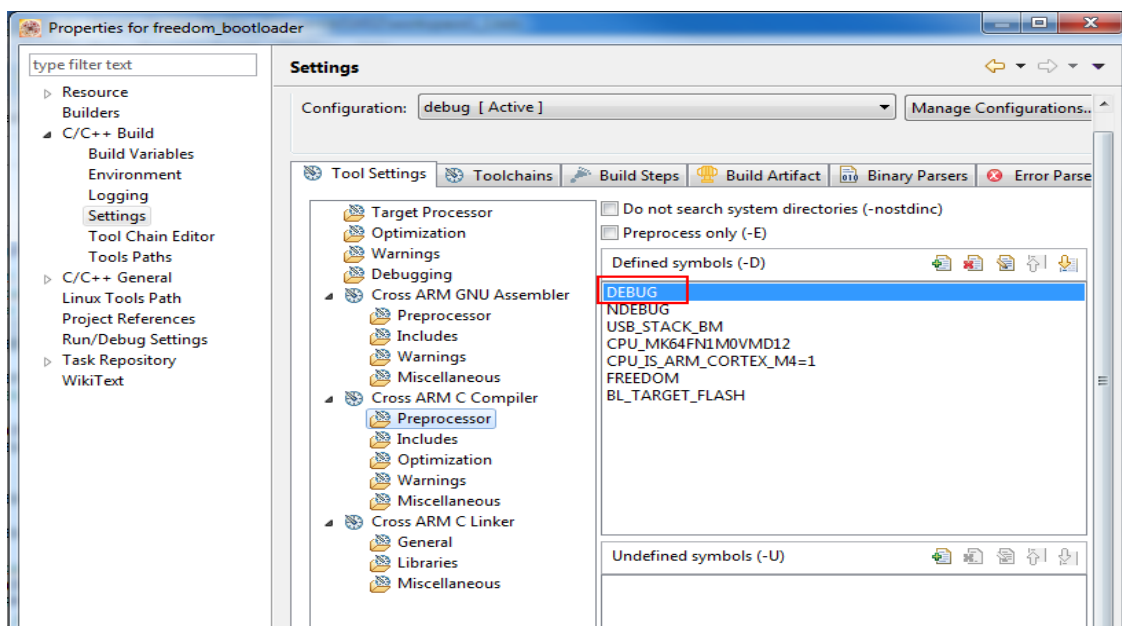
2.2. For release version , the image size is not beyond 0xa000, there are mainly two reasons.

(1) The debugger uses optimize size option , it can reduce the code size.

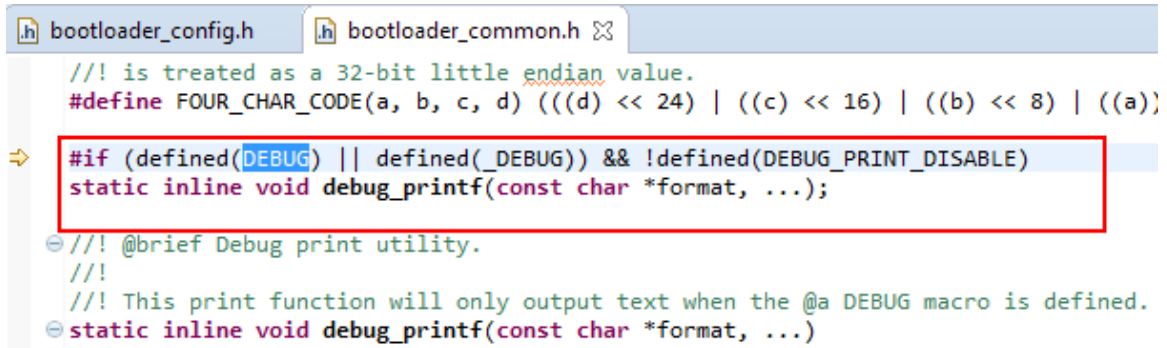


(2) There are some macro definitions to choose functions whether to be used, thus release mode reduces some functions that used in debug mode.

For example on debug mode , we can see it defines the macro of "DEBUG",



from the below file , if defined the DEBUG, it will uses the debug_printf() function , as we know , it occupy large memory .



```
bootloader_config.h | bootloader_common.h
// is treated as a 32-bit little endian value.
#define FOUR_CHAR_CODE(a, b, c, d) (((d) << 24) | ((c) << 16) | ((b) << 8) | ((a) << 0))
// if (defined(DEBUG) || defined(_DEBUG)) && !defined(DEBUG_PRINT_DISABLE)
static inline void debug_printf(const char *format, ...);
// @brief Debug print utility.
//
// This print function will only output text when the @a DEBUG macro is defined.
static inline void debug_printf(const char *format, ...)
```

We can see in the release mode not define it , so the release build image is smaller than the debug mode image.

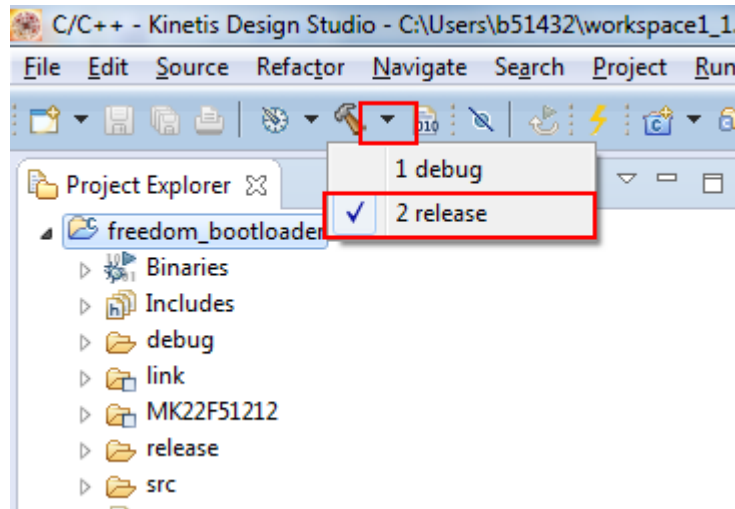
3. Workarounds

The “Kinetis Bootloader v2.0.0 Reference Manual” have reminder that “Load the Release build of the flash-resident bootloader if you plan to place the user application at 0xA000. Loading the Debug build requires you to move the application address beyond the end of the bootloader image. This address can be determined from the bootloader map file.”(At the part of 10.4.2 Bootloader configuration)

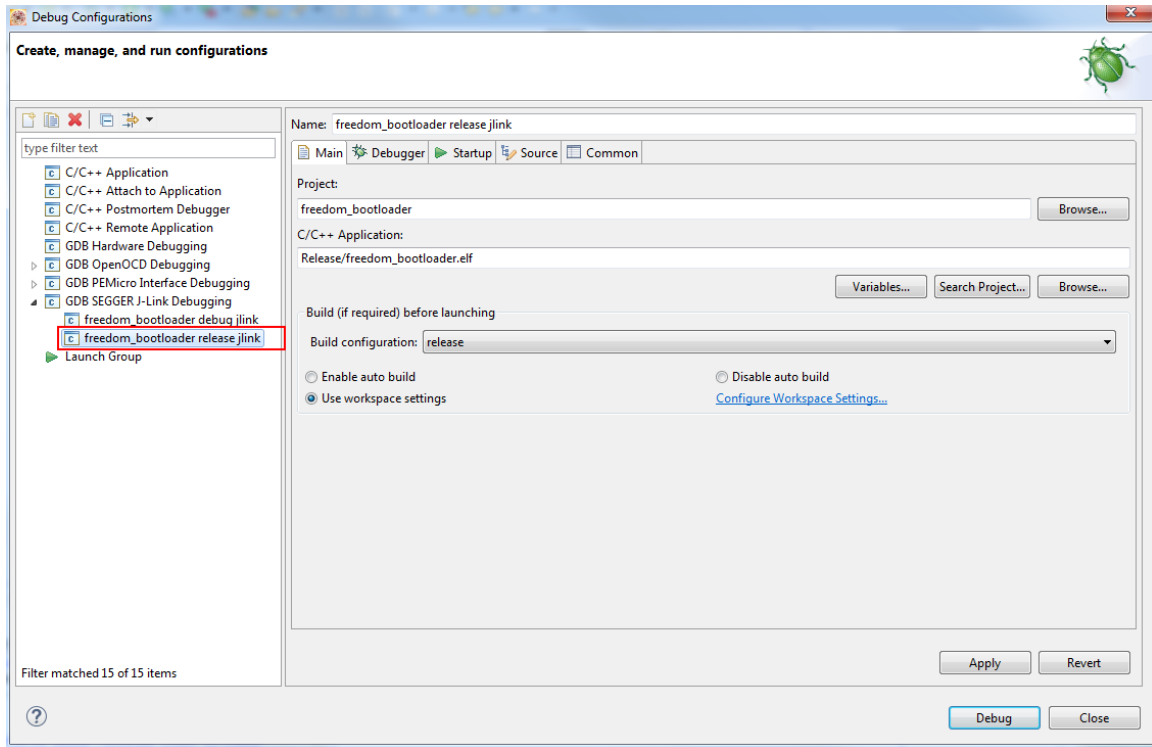
So there are two methods to fix this problem.

3.1. Use the release build image, not use the debug build image

- a. Please select “release” option before build bootloader project:



- b. Select the release image to download and debug :



Now we can refer to KBOOT user's guide to download application successfully.

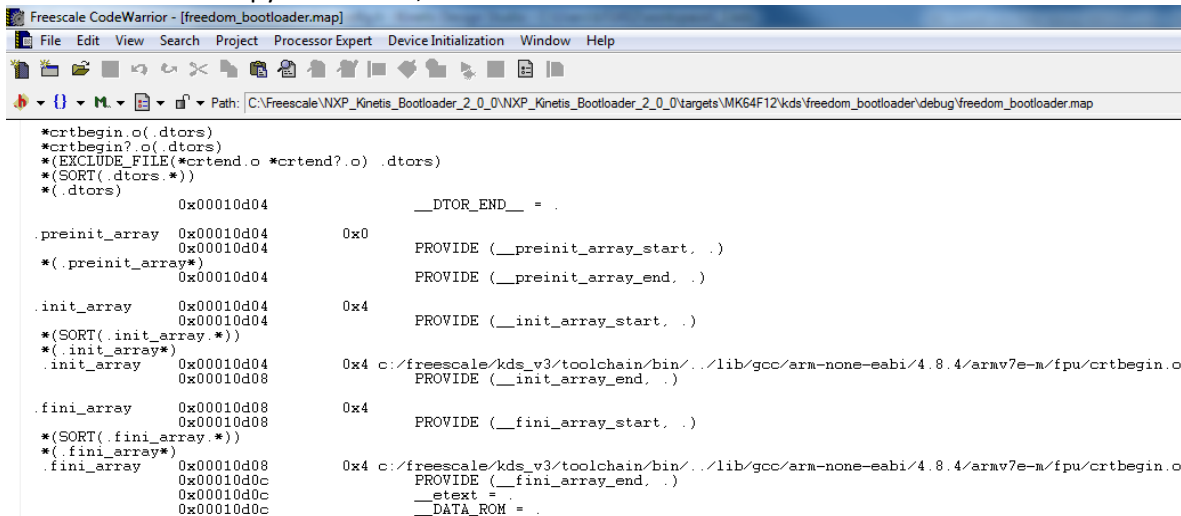
3.2. Revise the application start address outside of the bootloader flash region

If use the debug build , we need change user's application start address outside of the bootloader region.

For example the FRDM-K64 board:

- a. Revise the macro define of "BL_APP_VECTOR_TABLE_ADDRESS" on bootloader_config.h file:

After build debug version bootloader , from freedom_bootloader.map file , we can see bootloader occupy to 0x10d0c, over 0xa000.



So, I define the bootloader region to 0x12000, only need change the define of "BL_APP_VECTOR_TABLE_ADDRESS" to 0x12000.

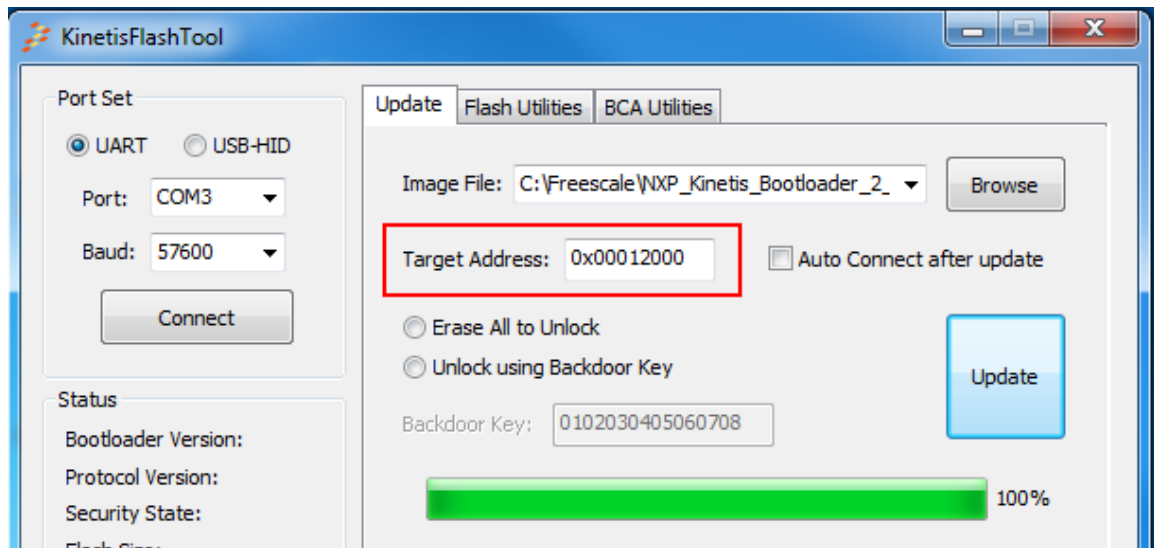
```
bootloader_config.h
// The bootloader will check this address for the application vector table upon startup.
#if !defined(BL_APP_VECTOR_TABLE_ADDRESS)
#define BL_APP_VECTOR_TABLE_ADDRESS 0xa000
#define BL_APP_VECTOR_TABLE_ADDRESS 0x12000
#endif
```

b. Revise the user's application linker file(*.ld file):

```
MK64FN1M0xxx12_application_0xa000.ld - Notepad
File Edit Format View Help
M_VECTOR_RAM_SIZE = DEFINED(__ram_vector_table__) ? 0x0400 : 0x0;

/* specify the memory areas */
MEMORY
{
  m_interrupts      (RX)  : ORIGIN = 0x00012000, LENGTH = 0x00000400
  m_flash_config    (RX)  : ORIGIN = 0x00012400, LENGTH = 0x00000010
  m_text            (RX)  : ORIGIN = 0x00012410, LENGTH = 0x000E08F0
  m_data            (RW)  : ORIGIN = 0x1FFF0000, LENGTH = 0x00040000
}
```

c. Revise the "Target Address" to 0x12000 on KinetisFlashTool GUI:



Pay attention that, the three addresses I revised should be the same, here I only take 0x12000 for instance.

Now we can use the KBOOT successfully.