インテル® MAX® 10 FPGA On-Chip Flash を使用するときの注意点

Ver.18.0

2018 年 12 月 Rev.1

インテル® MAX® 10 FPGA - On-Chip Flash を使用するときの注意点

目次

はじめに 3

1. MAX® 10 FPGA デバイスの On-Chip Flash 3

2. MAX® 10 FPGA で Nios® II を使用する際の On-Chip Flash での注意点 4

2-1. プログラムが UFM より大きいサイズの場合でも Error にならずビルドが通ってしまう 5

2-2. サイズオーバーした elf ファイルは HEX 変換や POF の生成を行っても Error にならない 6

3. 対処方法 7

改版履歴 12

はじめに

インテル® MAX® 10 FPGA にはフラッシュメモリーが内蔵されており、後述で示す通り CFM/UFM という形で提供されます。この時 Nios® II のソフトウェアをフラッシュメモリーに格納する場合 UFM に実装することになりますが、フラッシュメモリーのサイズとしては CFM を含めた全体のサイズで定義されます。このため Nios® II を使用する際にプログラム・サイズが On-Chip Flash の UFM 領域のサイズを超えていても Error として通知されません。

その為、MAX® 10 FPGA の UFM にユーザープログラムを格納する場合には、ユーザー側で常にサイズに注意する必要があります。

この資料では、MAX® 10 FPGA で Nios® II を使用する際の On-Chip Flash での注意点と対処方法について説明します。

1. MAX® 10 FPGA デバイスの On-Chip Flash
2. MAX® 10 FPGA で Nios® II を使用する際の On-Chip Flash での注意点
3. 対処方法

1. MAX® 10 FPGA デバイスの On-Chip Flash

インテル® MAX® 10 FPGA は、低コスト、シングルチップ、スモール・フォーム・ファクターの不揮発性プログラマブル・ロジック・デバイスです。

アナログ・デジタル・コンバーター（ADC）や、2 つのイメージを格納してダイナミックに切り替えることが可能なデュアル・コンフィグレーション・フラッシュメモリーなどの機能をシングルチップ上に搭載しています。

また、Nios® II ソフト・コア・エンベデッド・プロセッサーのサポート、デジタル信号処理（DSP）ブロック、ソフト DDR3 メモリー・コントローラーといった、フル装備の FPGA 機能を備えています。

インテル® MAX® 10 FPGA についての詳細は、以下のページを参照ください。

インテル® MAX® 10 FPGA

https://www.intel.co.jp/content/www/jp/ja/products/programmable/fpga/max-10.html

MAX® 10 FPGA デバイスには、On-Chip Flash が 2 つの部分に分割されています。

- **Configuration Flash Memory（CFM）**
  MAX® 10 FPGA のハードウェア・コンフィギュレーション・データを格納します。
- **User Flash Memory（UFM）**
  ユーザーデータまたはソフトウェア・アプリケーションを格納します。

MAX® 10 FPGA デバイスの UFM についての詳細は、以下の資料を参照ください。

Intel® MAX® 10 User Flash Memory User Guide

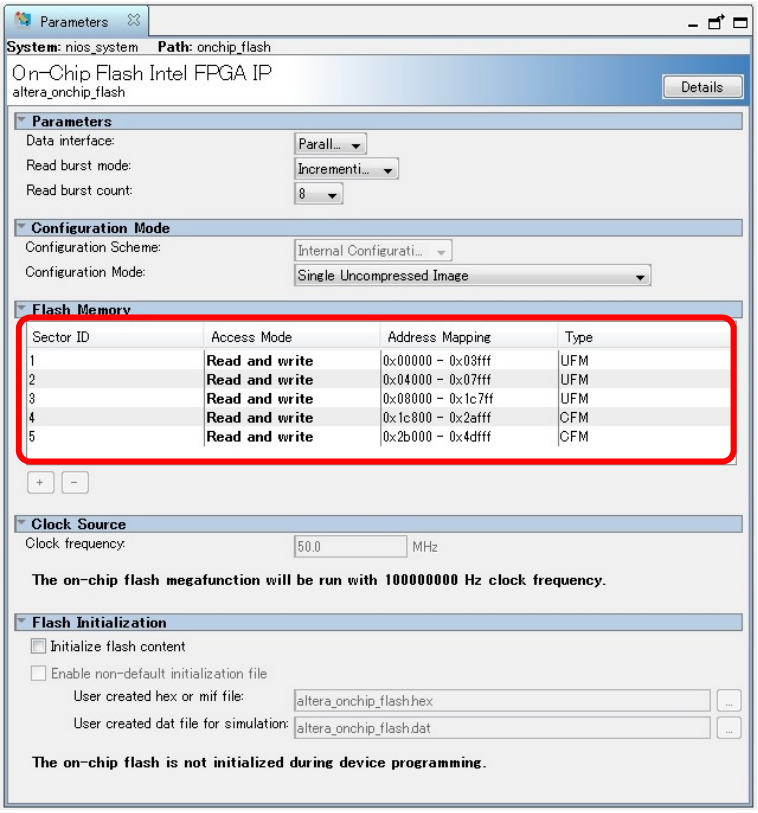
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/max-10/ug_m10_ufm.pdf

MAX® 10 ユーザー・フラッシュメモリー・ユーザーガイド（日本語版）

https://www.intel.co.jp/content/dam/altera-www/global/ja_JP/pdfs/literature/hb/max-10/ug_m10_ufm_j.pdf

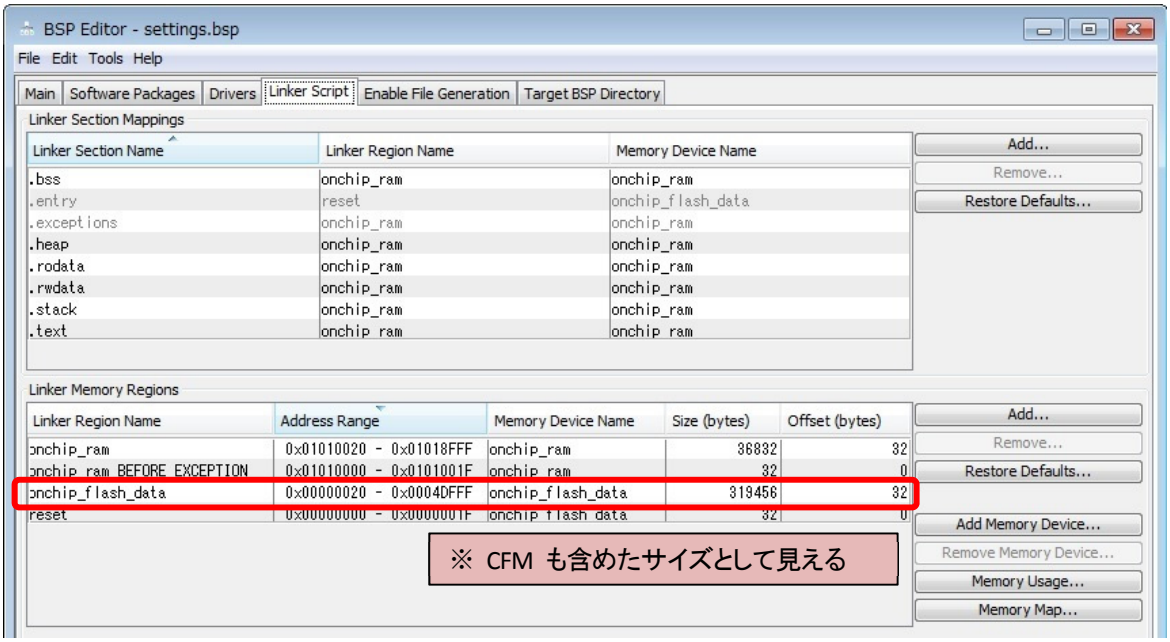
2. MAX® 10 FPGA で Nios® II を使用する際の On-Chip Flash での注意点

Platform Designer にて On-Chip Flash Intel FPGA IP（Altera On-Chip Flash IP）を下図のように設定した場合、Nios® II Software Build Tools (SBT) for Eclipse では UFM 領域のみならず、CFM の領域まで On-Chip Flash として見えています（この資料では、MAX® 10 FPGA デバイス 10M08 を例として説明しています）。



【図 2-1】 On-Chip Flash IP の設定

Nios® II SBT の BSP Editor から見ても、UFM と CFM の区別はしておらず、CFM も含めたサイズ（319456 Bytes）として On-Chip Flash 領域が見えています。



【図 2-2】 Nios® II SBT から見た On-Chip Flash 領域

2-1. プログラムが UFM より大きいサイズの場合でも Error にならずビルドが通ってしまう

【注意点 1】

上記のことから、.text を On-Chip Flash に配置する際は、プログラム（.elf ファイル）が UFM に収まるかではなく、On-Chip Flash に配置できるかでしかビルド時に判定されません。

つまりプログラムが UFM より大きいサイズの場合でも Error の表示無くビルドが通ってしまいます。

そのため UFM のサイズを考慮してプログラムを作成する必要があります。

Table 4: UFM and CFM Sector Size

This table lists the dimensions of the UFM and CFM arrays.

Device	Pages per Sector					Page Size (Kbit)	Maximum User Flash Memory Size (Kbit)(4)	Total Configura-tion Memory Size (Kbit)	OCRAM Size (Kbit)
	UFM1	UFM0	CFM2	CFM1	CFM0				
10M02	3	3	0	0	34	16	96	544	108
10M04	0	8	41	29	70	16	1248	2240	189
10M08	8	8	41	29	70	16	1376	2240	378
10M16	4	4	38	28	66	32	2368	4224	549
10M25	4	4	52	40	92	32	3200	5888	675
10M40	4	4	48	36	84	64	5888	10752	1260
10M50	4	4	48	36	84	64	5888	10752	1638

※ 10M08 の場合 UFM サイズは、((8*16)+(8*16))/8 = ***32 KByte***

- UFM0 のページサイズ ： 8（page）* 16（Kbit）
- UFM1 のページサイズ ： 8（page）* 16（Kbit）
- 上記 2 つの値を合算し、単位を Byte に直すために ÷8 で 32 KByte

【図 2-3】 UFM と CFM のアレイサイズ

```
[BSP build complete]
Info: Compiling hello_world.c to obj/default/hello_world.o
nios2-elf-gcc -xc -MP -MMD -c -I../hello_bsp//HAL/inc -I../hello_bsp/ -I../hello_bsp//drivers/ir
Info: Linking hello.elf
nios2-elf-g++  -T'../hello_bsp//linker.x' -msys-crt0='../hello_bsp//obj/HAL/src/crt0.o' -msys-li
nios2-elf-insert hello.elf --thread_model hal --cpu_name nios2 --qsys true --simulation_enabled
Info: (hello.elf) 33 KBytes program size (code + initialized data).
Info:              2872 Bytes free for stack + heap.
Info: Creating hello.objdump
nios2-elf-objdump --disassemble --syms --all-header --source hello.elf >hello.objdump
[hello build complete]

19:47:07 Build Finished (took 20s.247ms)
```

※ 10M08 では UFM が 32 Kbyte であるが、ビルドした elf ファイルが 33 Kbyte でもエラーは出ない

【図 2-4】 Nios® II SBT でビルドしたプログラムが 10M08 の UFM サイズ 32 KB を超えた例

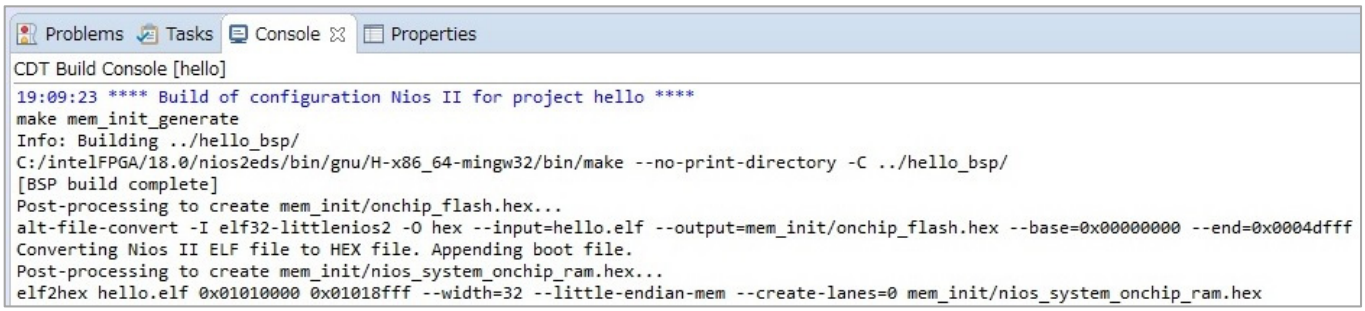
2-2. サイズオーバーした elf ファイルは HEX 変換や POF の生成を行っても Error にならない

【注意点 2】

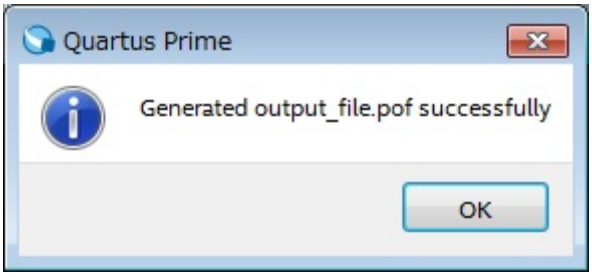
.elf ファイルはその後、

- HEX に変換
- POF を生成
- プログラムする

などに使用されますが、サイズオーバーした elf ファイルは、これらのどの過程でも Error は発生しません。



【図 2-5】 HEX に変更した場合のログ



【図 2-6】 POF を生成した場合のログ

【参考】

Nios® II SBT でプログラムを HEX ファイルに変換する方法や、Quartus® Prime で SOF ファイルと UFM 用 HEX ファイルから POF ファイルを生成する方法については、以下のページが参考になります。

MAX® 10 の UFM で Nios® II をブートさせてみよう［前編］

https://service.macnica.co.jp/library/118989

MAX® 10 の UFM で Nios® II をブートさせてみよう［後編］

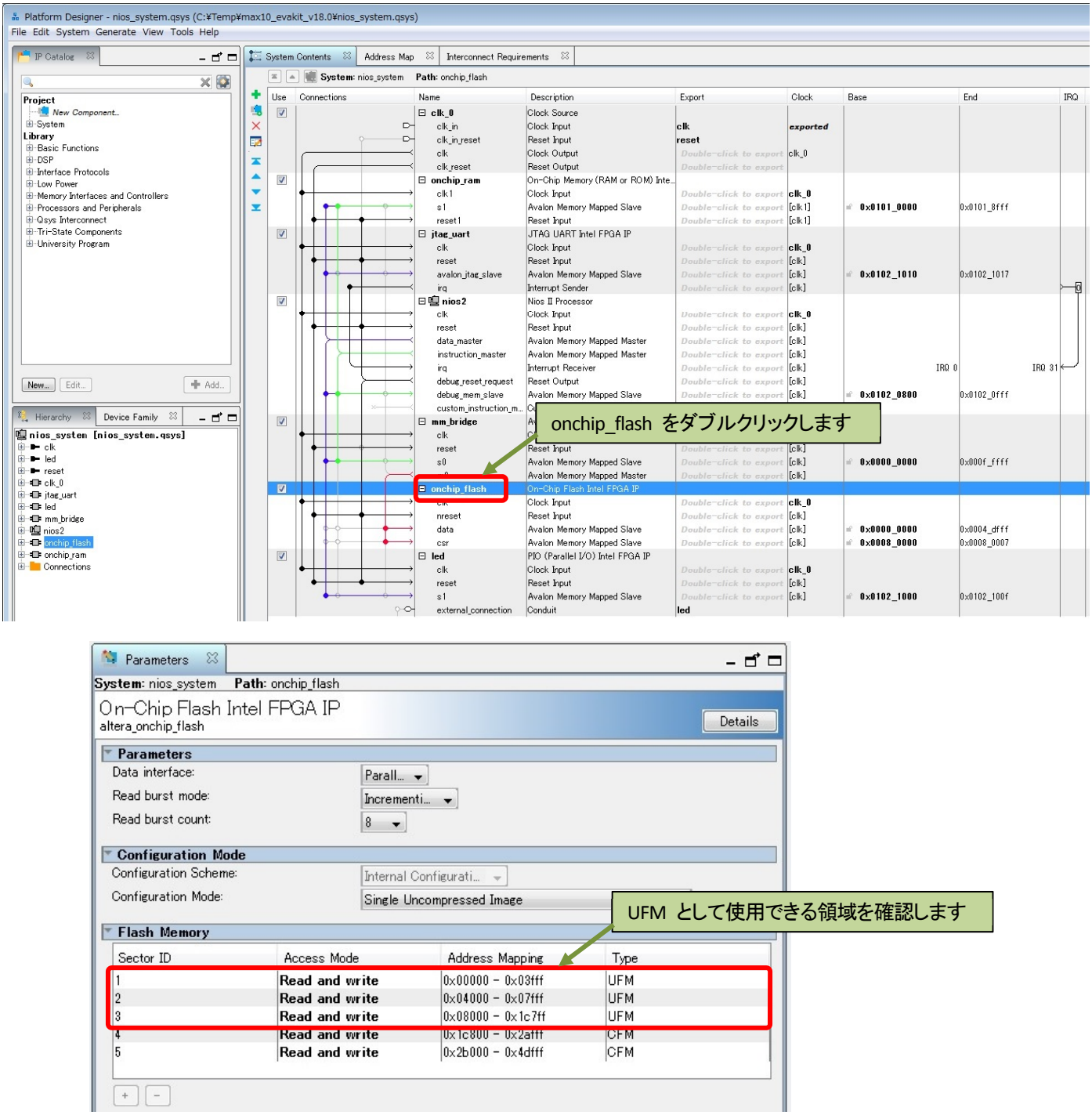
https://service.macnica.co.jp/library/119173

3. 対処方法

Nios® II SBT の BSP Editor 上であらかじめ配置できるセクション・サイズに変更します。
セクション・サイズを変更することでビルド時に Error として出力されるようになります。

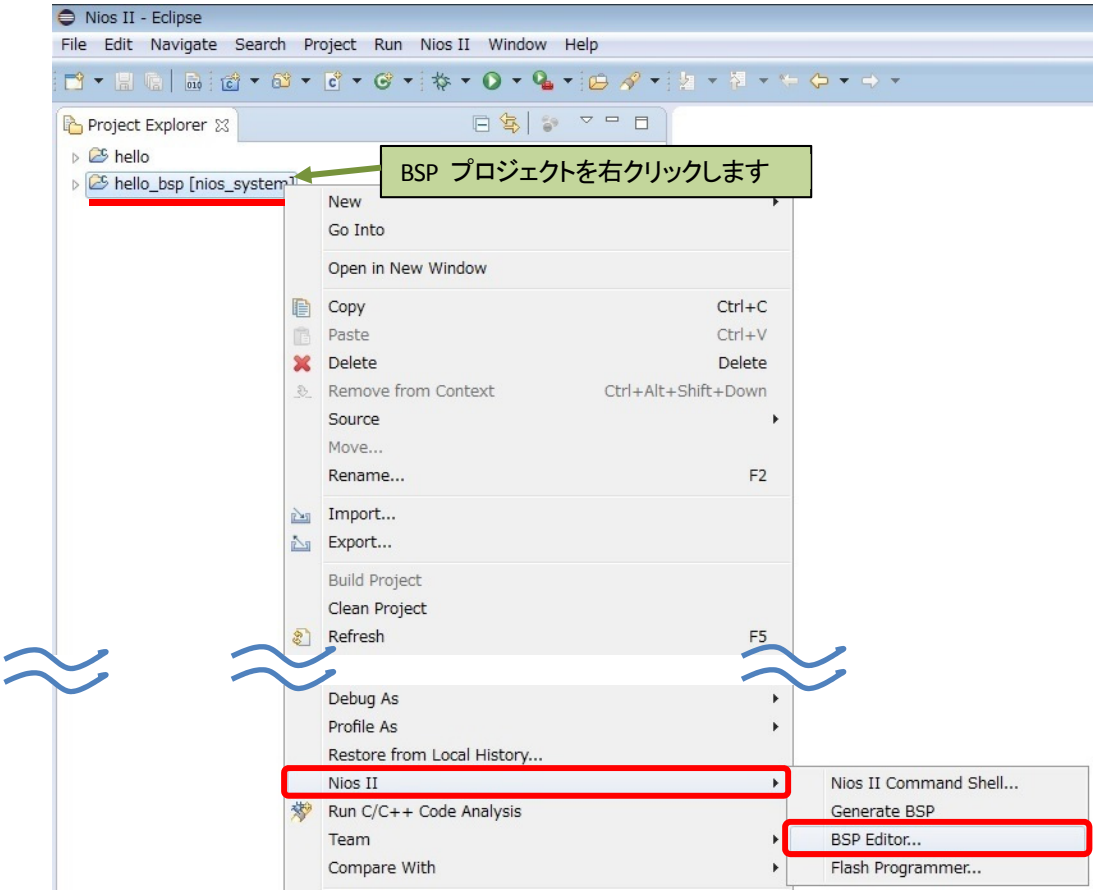
① Platform Designer の On-Chip Flash Intel FPGA IP（Altera On-Chip Flash IP）において UFM として使用できる領域を確認します。

例）この例では、0x0 ～ 0x1c7ff まで使用可能です（※ 0x1c800 → 116736 Byte）



【図 3-1】 On-Chip Flash IP において UFM として使用できる領域を確認

② Nios® II SBT の BSP プロジェクトを右クリックし、「**Nios II**」⇒「**BSP Editor**」を開きます。

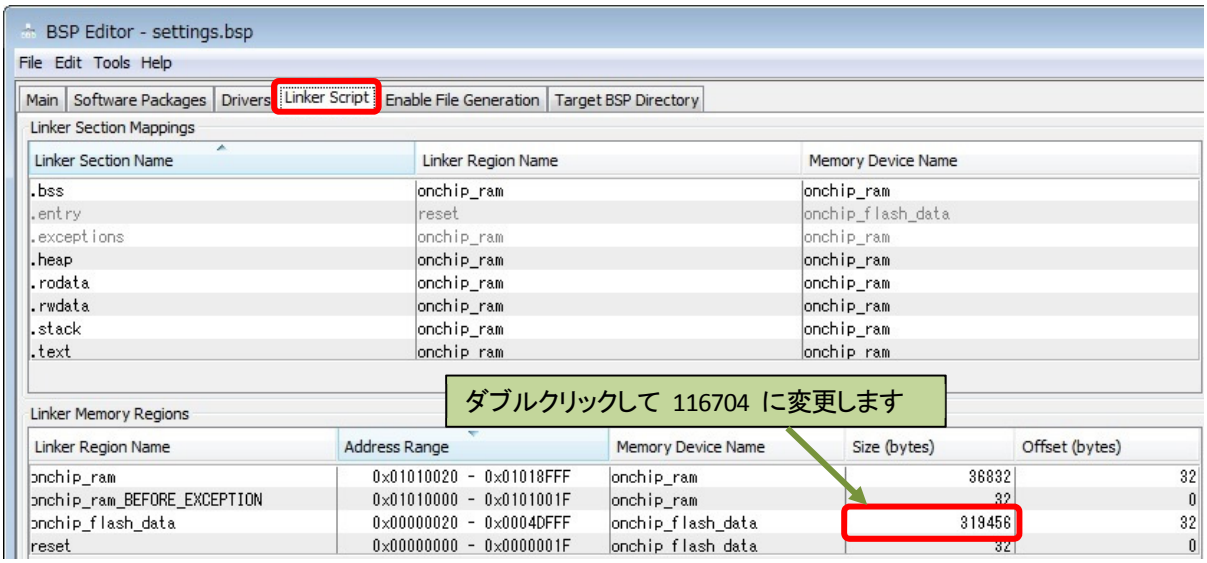


【図 3-2】 Nios II SBT の BSP Editor を開く

③ BSP Editor の **Linker Script** タブにおいて、onchip_flash_data の領域を Platform Designer で確認したサイズに設定します。このとき、先頭の 32 Byte（リセットベクター領域）を引いた値で設定します。

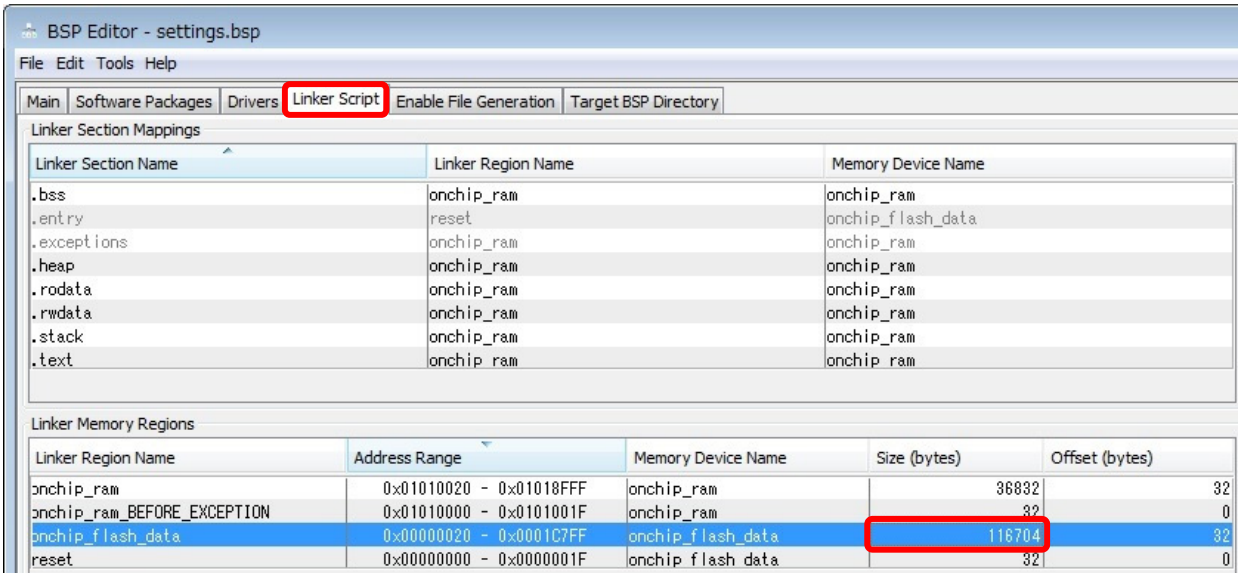
例）**10M08** の場合は、 116736 - 32 = **116704** Byte

設定前 : onchip flash data size = ***319456*** Byte



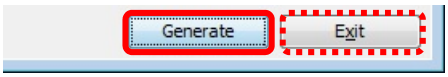
【図 3-3】 変更前の onchip_flash_data のサイズ（10M08 の場合の例）

設定後 : onchip flash data size = ***116704*** Byte



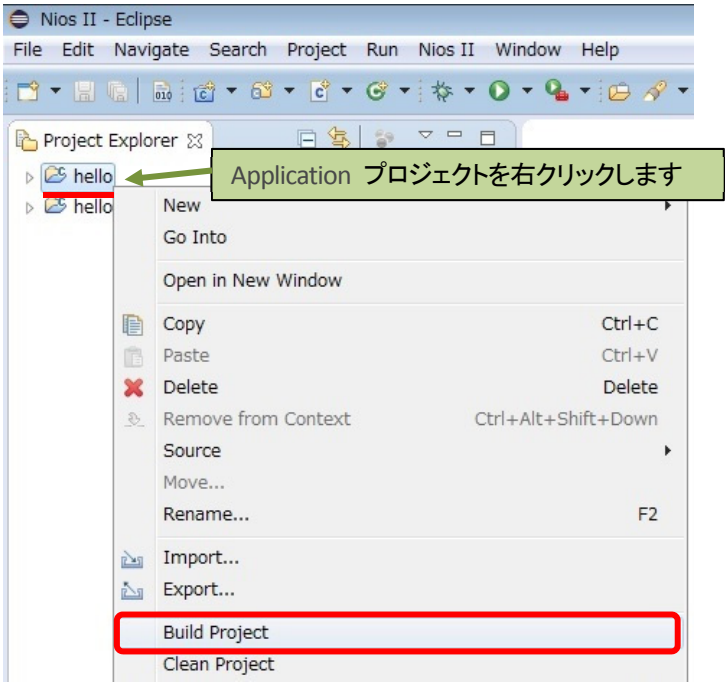
【図 3-4】 変更後の onchip_flash_data のサイズ（10M08 の場合の例）

④ ［**Generate**］ボタンをクリックし、その後［**Exit**］ボタンをクリックします。



【図 3-5】［Generate］ボタンをクリックし、その後［Exit］ボタンをクリック

⑤ Nios® II SBT の Application プロジェクトを右クリックし、「**Build Project**」を実行します。



【図 3-6】 Application プロジェクトのビルド

本問題に対応するために、MAX 10 の UFM 内にソフトウェアが収まるかどうかを簡易的にチェックするソフトウェア **check_size.exe**（check_size.c）を用意しました（別途、ダウンロードしてください）。

※ *ソフトウェア check_size.exe（check_size.c）は、参考として提供するものであり、運用した結果の影響については責任を負いかねますので、あらかじめご了承ください。*

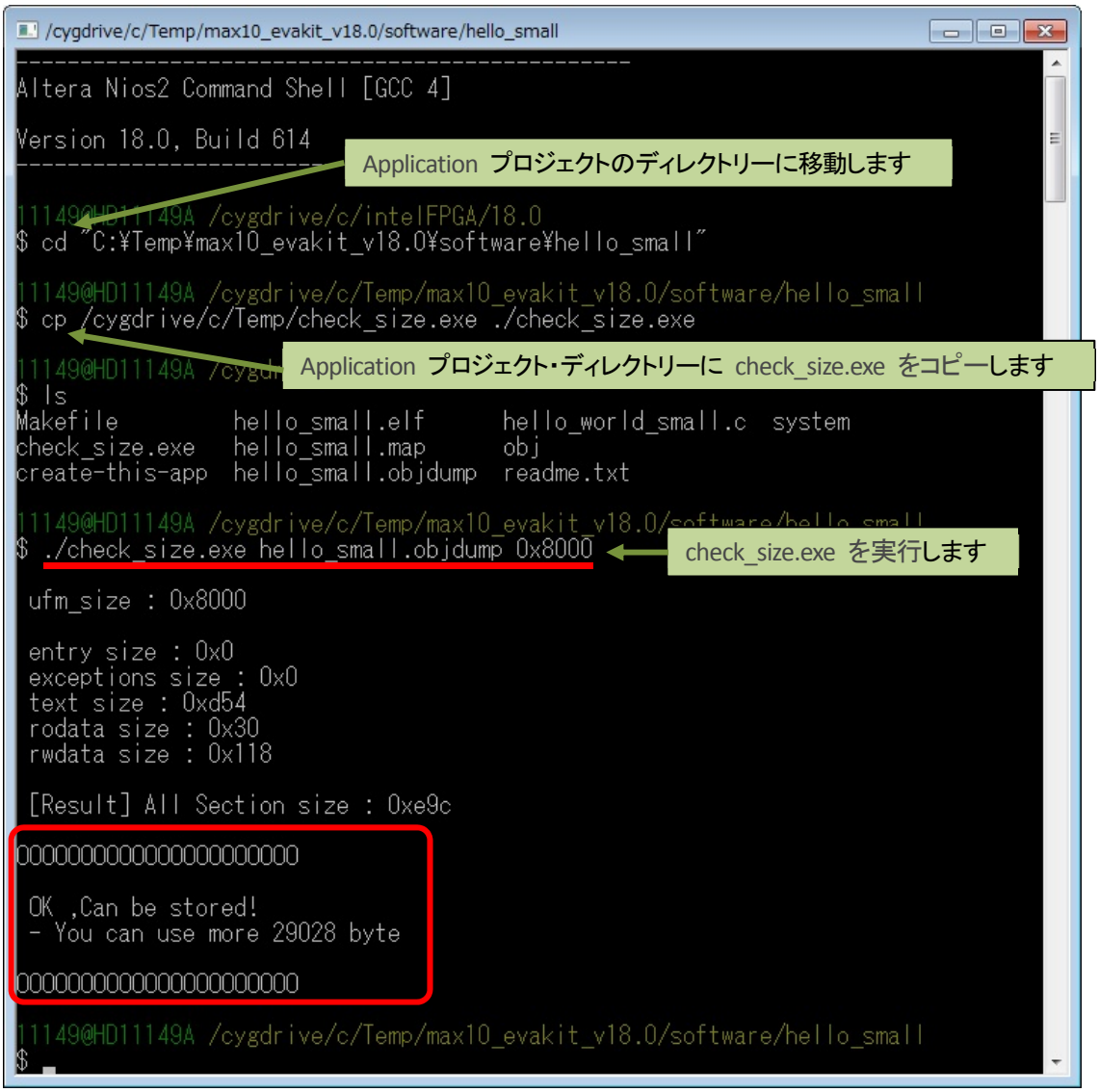
このソフトウェアでは、Nios® II SBT の Application プロジェクト内でビルド時に作成された .objdump ファイル内の記載から対象セクションのサイズを抜き出し足し合わせて、最終的にどれくらいのサイズが必要なのかを出力しています。

⑥ Nios® II SBT によるビルド後、Nios® II Command Shell より下記コマンドを入力し **check_size.exe** を実行して、作成したソフトウェアが指定した UFM に収まるかを確認します。

```
$ ./check_size.exe  <アプリケーション名.objdump>  <UFM サイズ>
```

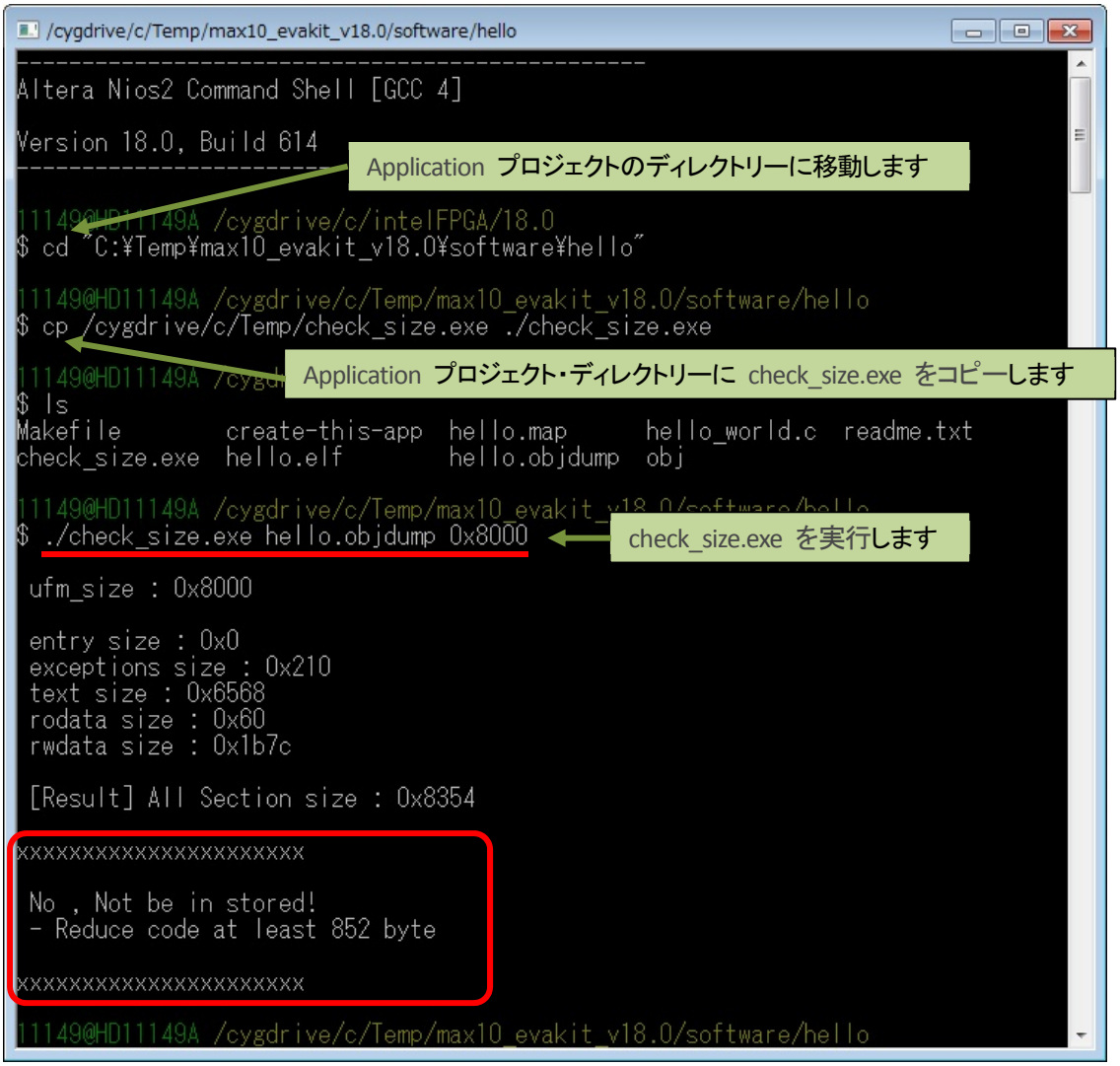
※ *check_size.exe を実行する前に、Application プロジェクト・ディレクトリーに check_size.exe をコピーしておきます。*

UFM に収まる場合



【図 3-7】 check_size.exe ツールによる UFM に収まる場合の表示例

UFM に収まらない場合



【図 3-8】 check_size.exe ツールによる UFM に収まらない場合の表示例

改版履歴

Revision	年月	概要
1	2018 年 12 月	初版