Preloader Generator の使用方法

Ver.14

2020 年 1 月 Rev.3

Preloader Generator の使用方法

目次

1. はじめに

この資料は、Altera SoC Embedded Design Suite（以後、SoC EDS）に付属の Preloader Support Package Generator（別名、bsp-editor）の使用方法について解説します。

1-1. 必要条件

本資料で解説される各種手順を実施するためには次の開発環境があらかじめホスト PC にインストールされている必要があります。

- Quartus® II v14.0
- SoC EDS v14.0

本資料の解説には v14.0 を使用しており、デフォルトのインストール・パスである以下のロケーションにツールをインストールしていることを前提として解説を進めます。 本資料内で使用されるスクリーン・キャプチャなどで確認できるツールのインストール・パスなどについては必要に応じて適宜読み替えてご参照ください。

Quartus II のインストール・パス： `C:¥altera¥14.0¥quartus`

SoC EDS のインストール・パス： `C:¥altera¥14.0¥embedded`

1-2. 関連文書

本資料は以下の資料および Web サイトの内容をベースに記述されております。これらの情報も合わせてご確認ください。

- Altera SoC Embedded Design Suite User Guide
- Cyclone V Device Handbook
- RocketBoards.org（ALTERA SoC コミュニティ・ポータルサイト）

1-3. Preloader とは?

Preloader は U-boot second program loader（以後、u-boot spl）をベースに Altera SoC 向けにカスタマイズが加えられたブートローダです。 Preloader の役割は以下の通りです。

- HPS ピン・マルチプレクスの設定
- HPS IOCSR の設定
- HPS PLL とクロックの設定
- HPS ペリフェラルのリセット解除
- SDRAM の初期化（キャリブレーションなど）
- SDRAM へ次ステージブート・イメージの展開

上記の通り、Preloader は HPS ブロックの初期化と、u-boot や OS を SDRAM にロードする機能を提供します。本資料で後述する通り、Preloader は Quartus II / Qsys の設計時に自動生成されるハンドオフ・ファイルを用いることで自動生成されます。このため、ユーザ側で初期化用ソフトウェアの構築をすることなく Quartus II / Qsys で設定した内容を HPS ブロックに反映することができます。

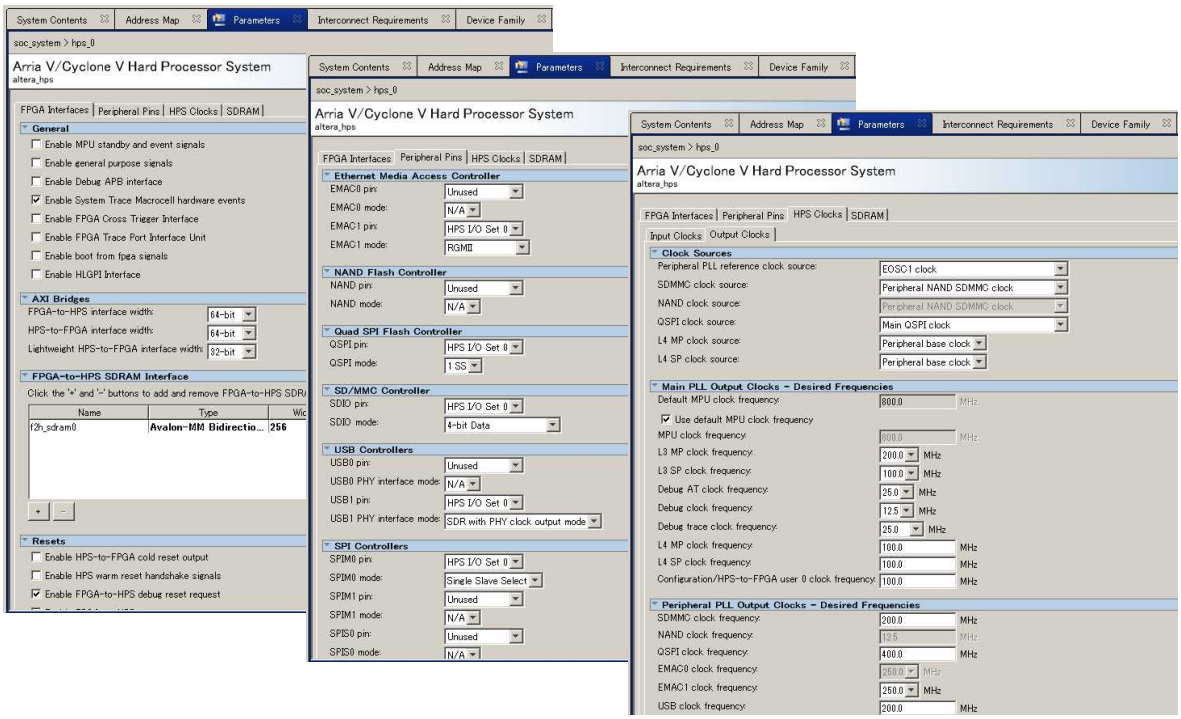

図 1-1 Qsys HPS の設定例

1-4. ブート・シーケンス

以下に SoC デバイスにおける一般的なブート・シーケンスを示します。HPS のブートは HPS 内部の BootROM を先頭に複数のステージに分かれて実施されます。それぞれのステージは次のステージのブート・イメージをロードする役割を担っており、これにより順々にブート・イメージがロードされ実行されます。

以下で紹介するブート・シーケンスはあくまでも一例であり、その他の構成で実現することも可能であることに注意してください。

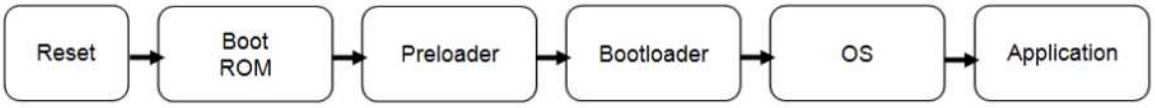

図 1-2 SoC の一般的なブート・シーケンス

以下に一般的なブート・シーケンスの各ステージにおける役割を示します。

1. BootROM
   HPS 内部の ARM プロセッサはリセットが解除されると BootROM にジャンプしソフトウェアの実行を開始します。BootROM では、BSEL ピン、CSEL ピンの設定から Preloader のロードに必要な最低限の初期化のみを実施し、BSEL ピンで指定された ブート・ソースから Preloader をロードします。BootROM のソフトウェアの書き換えは不可です。

2. Preloader
   前述の通り、Preloader は HPS 部の初期化を実行するためのブートローダです。Preloader を実行することで HPS 部の初期化および SDRAM の初期化を実行し、次ステージの Bootloader を SDRAM に展開し処理を渡します。

3. Bootloader
   このステージのブートローダは一般に Operating System（以後 OS）固有のローダとして配置されます。ALTERA Linux の場合には、オープンソースで提供される u-boot を Linux のブートローダとして利用しています。

4. OS / Application
   OS は起動時に必要な処理を実行しアプリケーションの実行を開始します。

ブート・シーケンスに関する詳細は以下の資料をご参照ください。

Booting and Configuration Introduction

2. Preloader Generator の使用方法

このセクションでは、Preloader Generator の使用手順および各種オプションに関して解説します。

2-1. Embedded Command Shell の起動

SoC EDS に付属の Embedded Command Shell を起動します。Windows のスタートメニュー、もしくは、Windows エクスプローラにて SoC EDS のインストール・フォルダ以下に格納される起動用スクリプトを実行します。

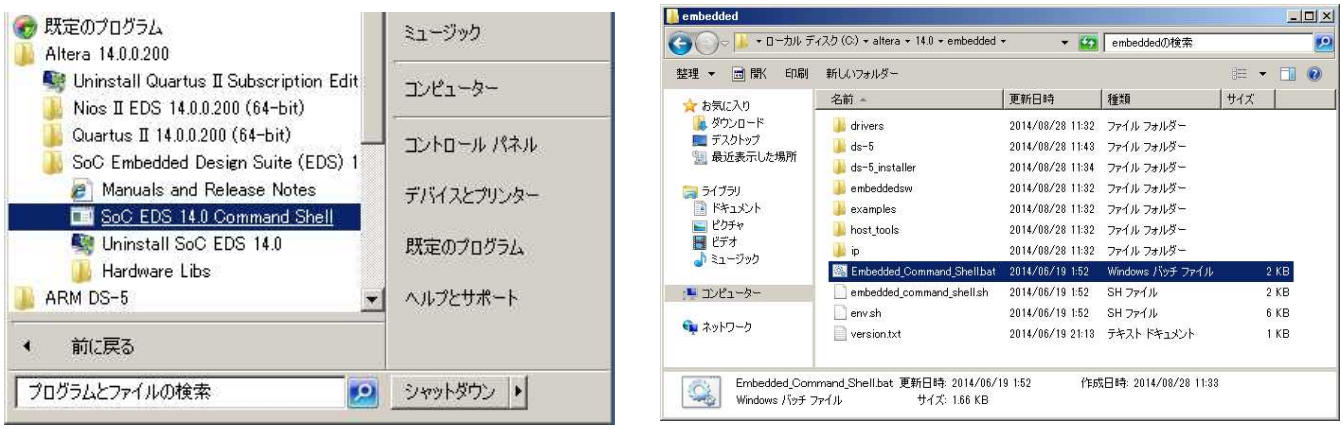

図 2-1 Embedded Command Shell の起動

2-2. bsp-editor（Preloader Generator）の起動

Embedded Command Shell に ”bsp-editor” とタイプし、bsp-editor の GUI を起動します。

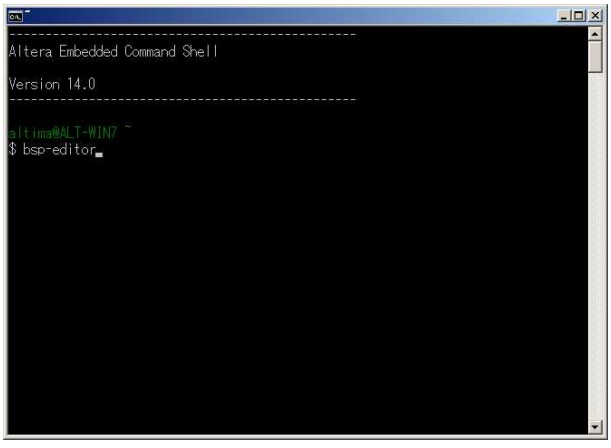

図 2-2 bsp-editor の起動

2-3. 新規プロジェクトの作成

File ⇒ New BSP... を選択し、プロジェクトを新規作成します。

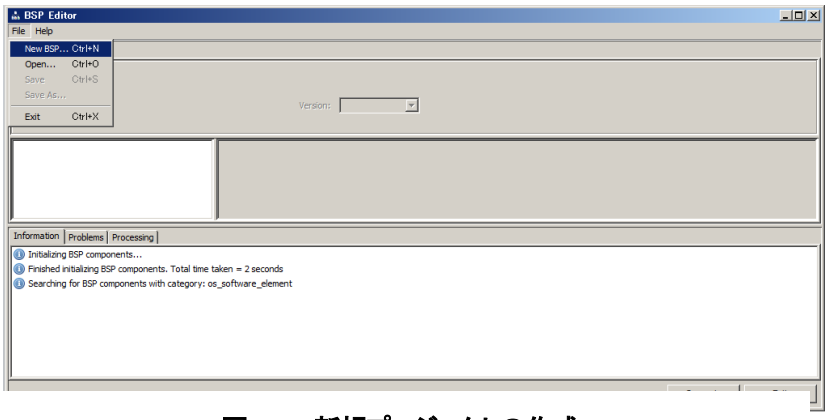

図 2-3 新規プロジェクトの作成

2-4. ハンドオフ・ファイルの指定

Quartus II / Qsys にてプロジェクトをコンパイルした際に自動生成されるハンドオフ・ファイルを指定します。本資料では SoC EDS に付属のリファレンス・デザイン内に格納されるハンドオフ・ファイルを利用します。

<SoC EDS Install DIR>¥embedded¥examples¥hardware¥cv_soc_devkit_ghrd¥hps_isw_handoff¥soc_system_hps_0

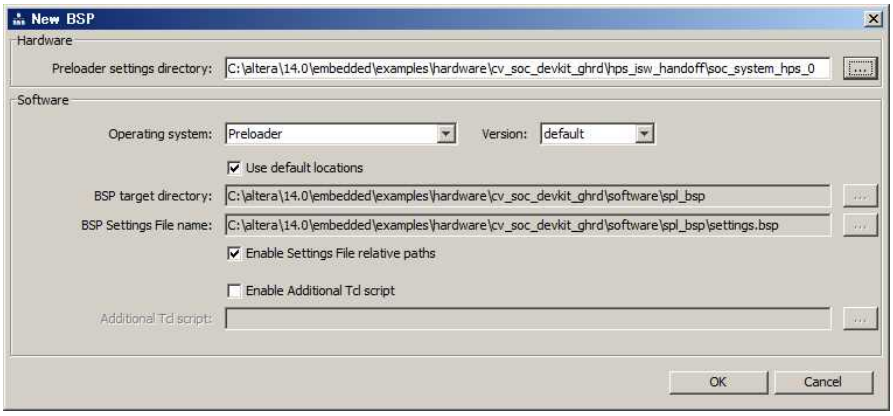

図 2-4 ハンドオフ・ファイルの指定

Operating System : に Preloader が Version : に default が指定されていることを確認します。BSP target directory にて bsp プロジェクトを生成するロケーションを指定します。

デフォルトでは、"*<Quartus II Project DIR>*¥software¥spl_bsp" が指定されますが、use default locations のチェックを外すことで任意のディレクトリを指定することが出来ます。

2-5. Preloader ユーザ・オプションの設定

Preloader に生成時に与えるユーザ・オプションの各種設定を行います。設定には大きく分けて “Common” と “Advanced” があります。それぞれ必要に応じて設定を変更します。

各設定をマウスオーバーすると設定の解説が表示されますので合わせてご確認ください。以降ではそれぞれの設定について解説します。

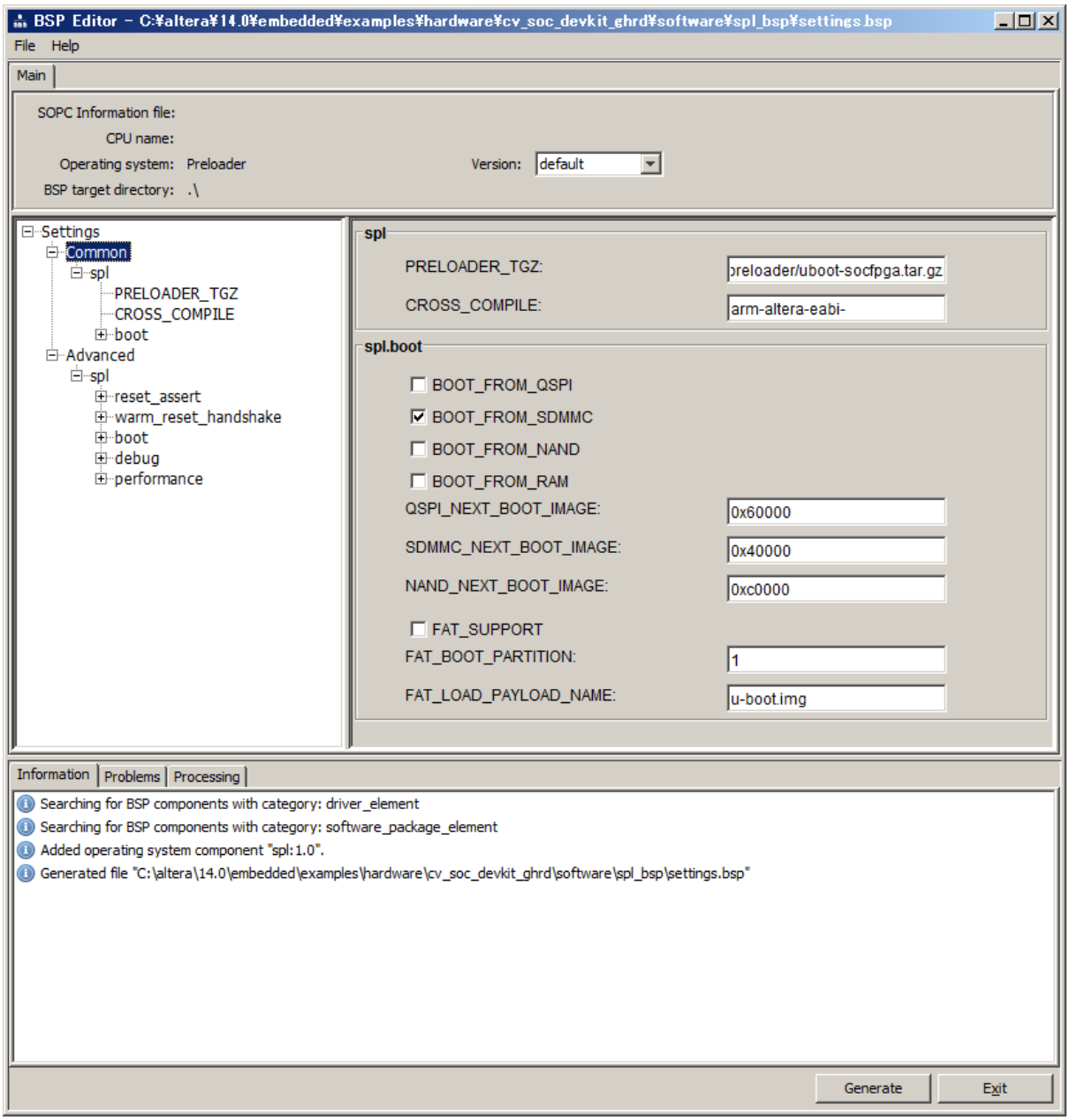

図 2-5 ユーザ・オプションの設定

2-5-1. Common オプション設定

Common オプションの設定では、Preloader に関する基本的なオプションに関して設定を行います。以下に設定内容を示します。

表 2-1 Common オプション設定一覧

大項目	小項目	概要
spl	PRELOADER_TGZ	Preloader ソース・ファイルのアーカイブを指定します。指定したアーカイブ・ファイルを解凍し利用します。基本的に変更する必要はありません。
	CROSS_COMPILE	Preloader のビルドに使用するクロスコンパイラを指定します。SoC EDS 付属のコンパイラが指定されておりますので、基本的に変更する必要はありません。
spl.boot	BOOT_FROM_QSPI	Preloader に続くブート・イメージを QSPI フラッシュからロードする場合にチェックを入れます。
	BOOT_FROM_SDMMC	Preloader に続くブート・イメージを SD カードからロードする場合にチェックを入れます。
	BOOT_FROM_NAND	Preloader に続くブート・イメージを NAND フラッシュからロードする場合にチェックを入れます
	BOOT_FROM_RAM	Preloader に続くブート・イメージを RAM からロードする場合にチェックを入れます。
	*上記、BOOT_FROM_XXX の設定は、いずれか 1 つのみ指定してください（複数にチェックを入れないでください）。	
	QSPI_NEXT_BOOT_IMAGE	BOOT_FROM_QSPI が有効の際、Preloader がロードするブート・イメージが格納されるアドレスを指定します。
	SDMMC_NEXT_BOOT_IMAGE	BOOT_FROM_SDMMC が有効の際、Preloader がロードするブート・イメージが格納されるアドレスを指定します。
	NAND_NEXT_BOT_IMAGE	BOOT_FROM_NAND が有効の際、Preloader がロードするブート・イメージが格納されるアドレスを指定します。
	FAT_SUPPORT	SD カード内の FAT パーティションに対するアクセスを有効化します。この設定は BOOT_FROM_SDMMC が指定されている時のみ有効です。FAT パーティションに次ステージのブート・イメージが格納されている場合に利用します。
	FAT_BOOT_PARTITION	次ステージのブート・イメージが格納される FAT パーティションの番号を指定します。この設定は、FAT_SUPPORT オプションが指定されている時のみ有効です。
	FAT_LOAD_PAYLOAD_NAME	次ステージのブート・イメージのファイル名を指定します。この設定は、FAT_SUPPORT オプションが指定されている時のみ有効です。

2-5-2. Advanced オプション設定

Advanced オプション設定では、Preloader に関する高度な機能について設定します。基本的に変更の必要はありませんが状況に応じて設定を変更してください。

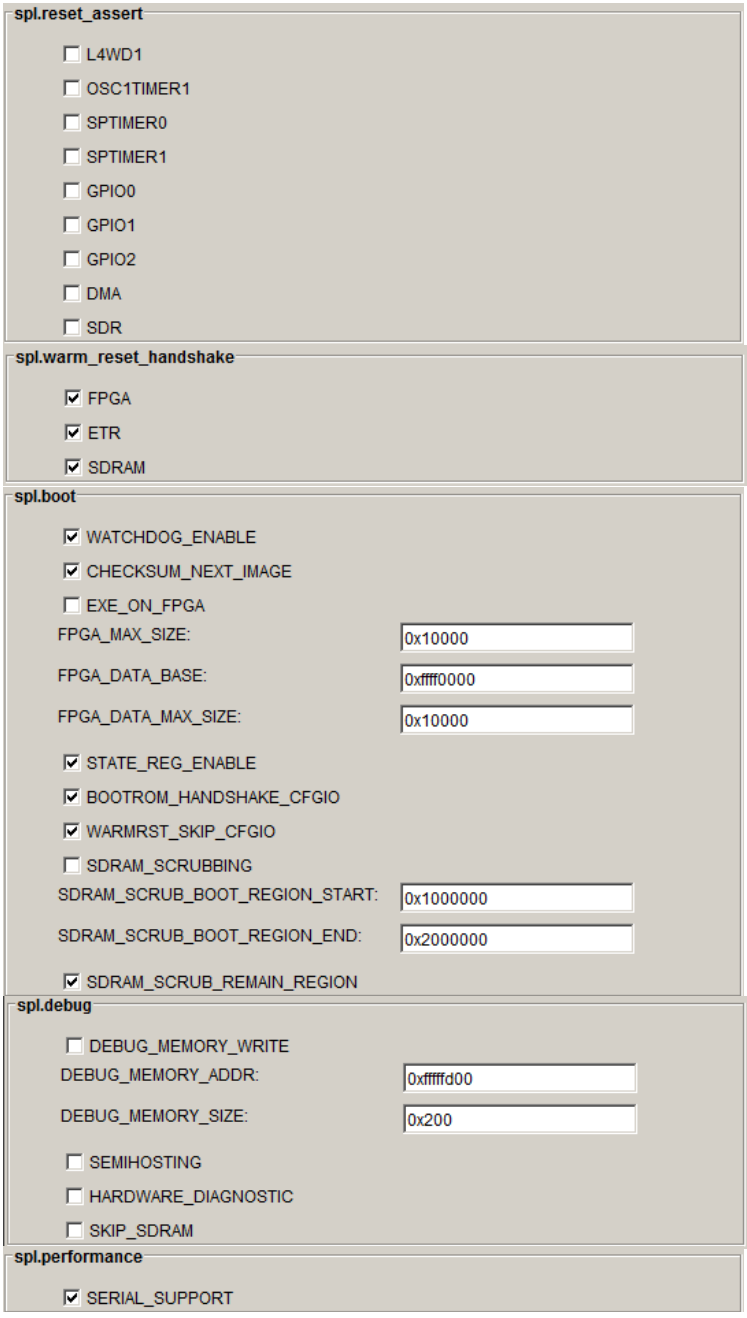

図 2-6 Advenced オプション設定

表 2-2 Advanced オプション設定一覧

大項目	小項目	概要
spl.reset_assert		Preloader 実行時に該当ペリフェラルのリセットの解除を実行するか指定します。チェックを入れるとリセット状態のままとなります。
spl.warm_reset_handshake		WarmReset 時に該当ペリフェラルとのハンドシェイク機能の使用有無を指定します。
spl.boot	WATCHDOG_ENABLE	ウォッチドッグ・タイマの使用有無を指定します。各種デバッグ時には無効化することを推奨します。
	CHECKSUM_NEXT_IMAGE	次ステージのブート・イメージのロード時にチェックサムによるエラーチェックの実施有無を指定します。
	EXE_ON_FPGA	Preloader を FPGA メモリ上で実行する再にチェックを有効にします。このオプションは、BSEL を FPGA に指定した場合に利用します。
	FPGA_MAX_SIZE	FPGA メモリに配置可能な.text、.rodata の最大サイズを指定します。Preloader のビルド時にこのサイズよりも実際のコードサイズが大きい場合にビルドエラーを返します。
	FPGA_DATA_BASE	EXE_ON_FPGA オプション有効時に、.data、.bss、malloc、stack を配置するベースアドレスを指定します。
	FPGA_DATA_MAX_SIZE	Preloader のビルド時にこのサイズよりも実際のデータ（.data 等）が大きい場合にビルドエラーを返します。
	STATE_REG_ENABLE	Preloader 実行時に STATE R レジスタに STATE_VALID データを書き込みます。これは BootROM に対して Preloader が正常にロードされたことを示します。
	BOOTROM_HANDSHAKE_CFGIO	IOCSR および Pin MUX の初期化時に BootROM とのハンドシェイクを実行します。本設定が有効時に WarmReset が発生した場合、Preloader が初期化を実施していても BootROM にて再設定を実行します。
	WARMRST_SKIP_CFGIO	本設定が有効な場合、Preloader は IOCSR および Pin MUX の設定をスキップします。この設定は、BOOTROM_HANDSHAKE_CFGIO が無効な場合にのみ有効となります。
	SDRAM_SCRUBBING	SDRAM ECC 有効時にメモリの初期化を実行します。
	SDRAM_SCRUB_BOOT_REGION_START	SCRUB を実行するメモリの先頭アドレスを指定します。
	SDRAM_SCRUB_BOOT_REGION_END	SCRUB を実行するメモリの終了アドレスを指定します。
	SDRAM_SCRUB_REMAIN_REGION	次ステージのブート・イメージをロード中に上記 SCRUB オプションで指定されない残りのメモリ領域の初期化を実行するか指定します。
spl.debug	DEBUG_MEMORY_WRITE	デバッグ情報をメモリに書き出すオプションの使用有無を指定します。UART が使用出来ないシステムに有効です。

	DEBUG_MEMORY_ADDR	デバッグ情報を書き出すメモリの先頭アドレスを指定します。
	DEBUG_MEMORY_SIZE	デバッグ情報を書き出すメモリに割り当てるメモリサイズを指定します。
	SEMIHOSTING	セミホスティング機能の使用有無を指定します。このオプションはデバッガと接続されていることが前提のオプションとなるため注意してください。UART が使用出来ないシステムにおける Preloader のデバッグに有効です。
	HARDWARE_DIAGNOSTIC	SDRAM メモリの簡易テストを実行します。このオプションを利用するためには少なくとも 1 GB のメモリ領域が必要です。
	SKIP_SDRAM	Preloader の実行時に SDRAM メモリの初期化およびキャリブレーションをスキップします。
spl.performance	SERIAL_SUPPORT	UART を使用したログ出力を有効化します。

2-6. bsp プロジェクトの生成

各種設定が完了後、bsp プロジェクトを生成します。生成される bsp プロジェクトには *.c 、 *.h 、 Makefile を含む Preloader を生成（ビルド）するために必要なファイル群がすべて生成されます。これらのファイルは、「 2-4 ハンドオフ・ファイルの指定 」で “BSP target directory” に指定したロケーションに出力されます。

bsp-editor 右下の “Generate” ボタンを押下しプロジェクトを生成します。本資料では、以下のディレクトリが該当します。

<*SoC EDS Install DIR*>¥embedded¥examples¥hardware¥cv_soc_devkit_ghrd¥software¥spl_bsp

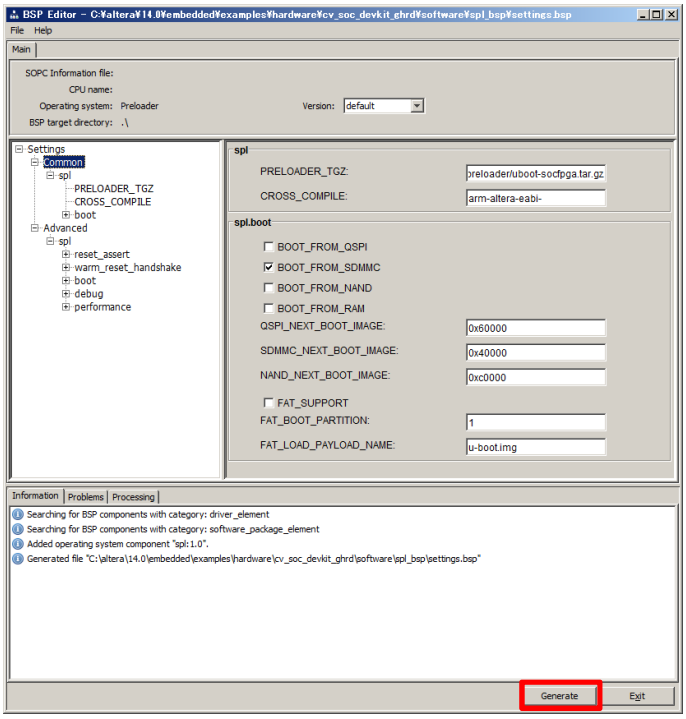

図 2-7 bsp プロジェクトの生成

3. Preloader の生成方法

このセクションでは、前述の手順で生成した bsp プロジェクトを利用し Preloader を生成する手順について解説します。

3-1. Embedded Command Shell の起動

Embedded Command Shell を起動します。起動方法については前述の「 2-1 Embedded Command Shell の起動 」を参照ください。

3-2. bsp プロジェクト・ディレクトリへの移動

cd コマンドを利用して Preloader Generator (bsp-editor) で生成した bsp プロジェクトのルートディレクトリに移動します。本資料の場合は以下のディレクトリが該当します。

<*SoC EDS Install DIR*>¥embedded¥examples¥hardware¥cv_soc_devkit_ghrd¥software¥spl_bsp

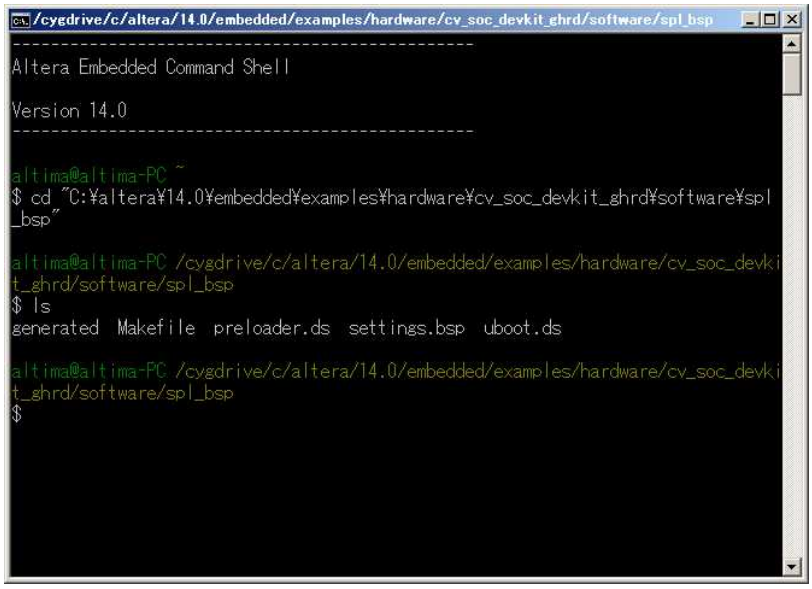

図 3-1 bsp プロジェクト・ディレクトリへの移動

3-3. Preloader のビルド

”make all” コマンドを実行し Preloader を生成します。

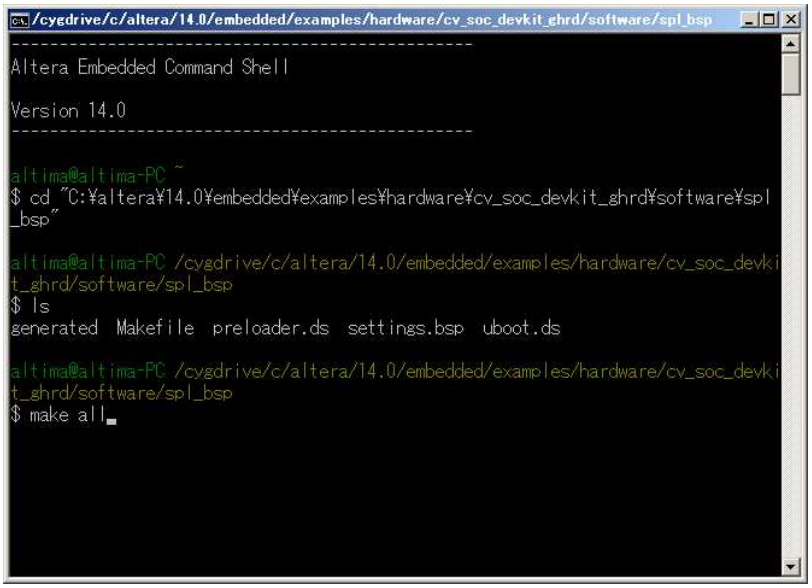

図 3-2 Preloader のビルド

実行後、”<*Quartus II Project Top DIR*>¥software¥spl_bsp” 以下に、preloader-mkpimage.bin という名称のバイナリ・ファイルが生成されていることを確認します。このファイルは BootROM にて参照される Preloader 用のヘッダ情報を付加したバイナリとなっています。SD カードおよび QSPI フラッシュメモリ等への書き込みはこのバイナリ・ファイルを利用します。

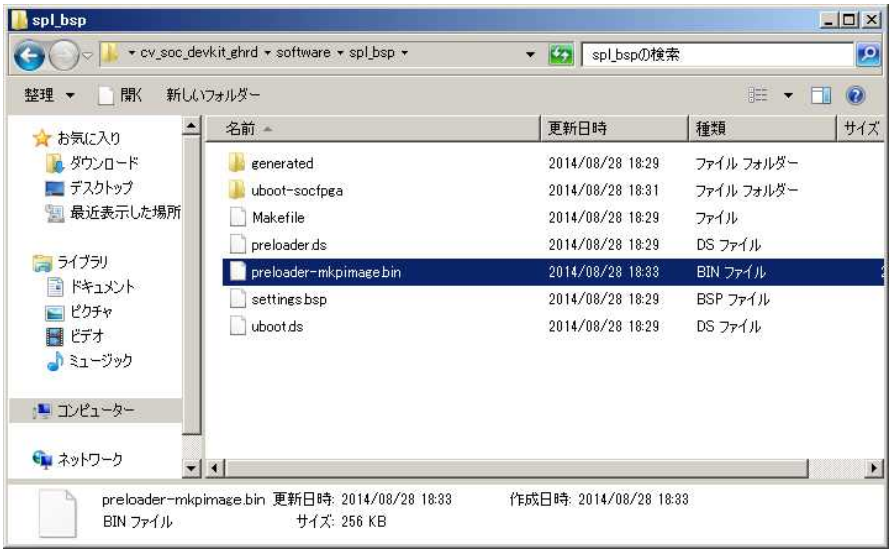

図 3-3 生成された Preloader バイナリ・ファイル

【注記】

本資料では、Windows® 7 Professional を使用して動作の確認を行っております。

ホスト PC の OS が Windows® 10 の場合、 Preloader の生成でエラーが発生する場合が確認されております。
もしご使用の OS が Windows® 10 でエラーが発生する場合は、下記のアルティマ技術サポートのコンテンツページで紹介している、bsp プロジェクト内 Makefile の編集が必要となりますのでご注意ください。

Windows® 10 における Preloader のビルドエラー

3-4. u-boot のビルド（必要に応じて）

”make uboot”コマンドを実行し u-boot を生成します。

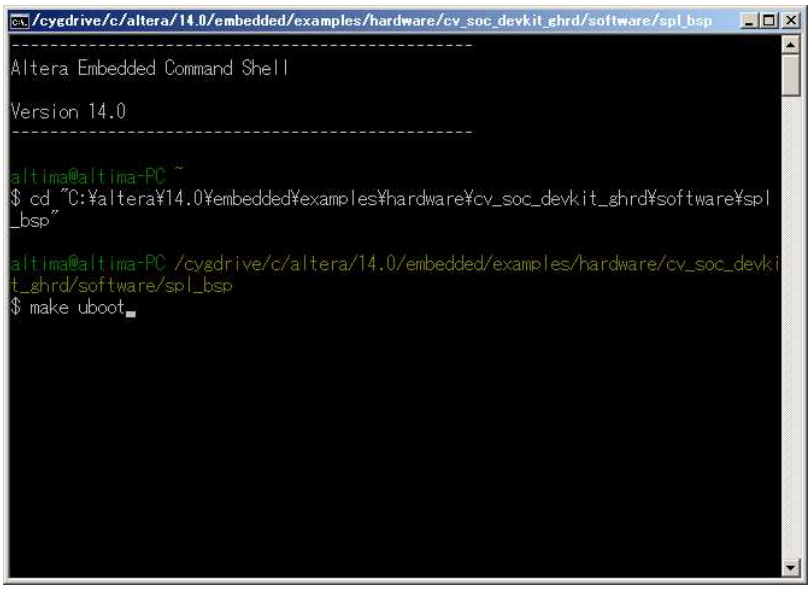

図 3-4 u-boot の生成

上記コマンドを実行すると、以下のディレクトリに Preloader にてロード時に参照されるヘッダを付加した u-boot バイナリ・イメージである u-boot.img が生成されます。

<Quartus II Project Top DIR>¥software¥spl_bsp¥uboot-socfpga¥u-boot.img

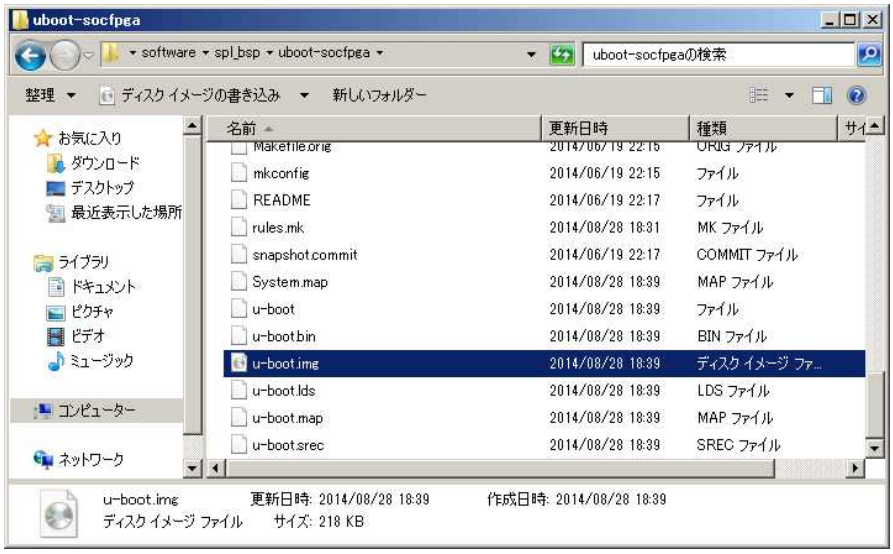

図 3-5 生成された u-boot イメージ・ファイル

4. Preloader / u-boot の更新方法

このセクションでは前述の手順で新規に作成した Preloader バイナリ・ファイルをセットアップ済みの SD カードおよびオンボード上の QSPI Flash メモリに書き込む方法について解説します。本資料で紹介する手順はあくまでも一例であり、他の方法で書き換えることも可能です。

4-1. SD カード上イメージの書き換え

SD カード上イメージの書き換えは、RocketBoards.org に公開されている SD カードイメージ、もしくは SoC EDS 付属の SD カードイメージが書き込まれていることを前提として解説を進めます。

SD カードのセットアップが完了していない場合には、「 1-2 関連文書 」で紹介した Altera SoC Embedded Design Suite User Guide の Getting Started Guides を参考にセットアップを進めてください。なお、SoC EDS に付属の SD カードイメージは以下に保存されています。

<*SoC EDS Install DIR*>¥embedded¥embeddedsw¥socfpga¥prebuilt_images¥sd_card_linux_boot_image.tar.gz

SoC EDS v14.0 には Windows マシンから直接 SD カードの中身を書き換えるためのツール（ALTERA Boot Disk Utility）が同梱されており、上記いずれかの SD カードイメージでセットアップ済みの SD カードに対し、Preloader / u-boot のデータのみを部分的に書き換えることが可能です。

ALTERA Boot Disk Utility を実行する際は、Embedded Command Shell を管理者権限で起動する必要がありますので注意してください。

Preloader / u-boot を書き換える場合には以下のコマンドを使用します。

- Preloader: `$ alt-boot-disk-util -p preloader-mkpimage.bin -a write -d D`
- u-boot: `$ alt-boot-disk-util -b uboot-socfpga/u-boot.img -a write -d D`

* 上記コマンドの -d オプションで指定するドライブレターはホスト PC 環境により異なります。お使いの環境に合わせて読み替えてください。

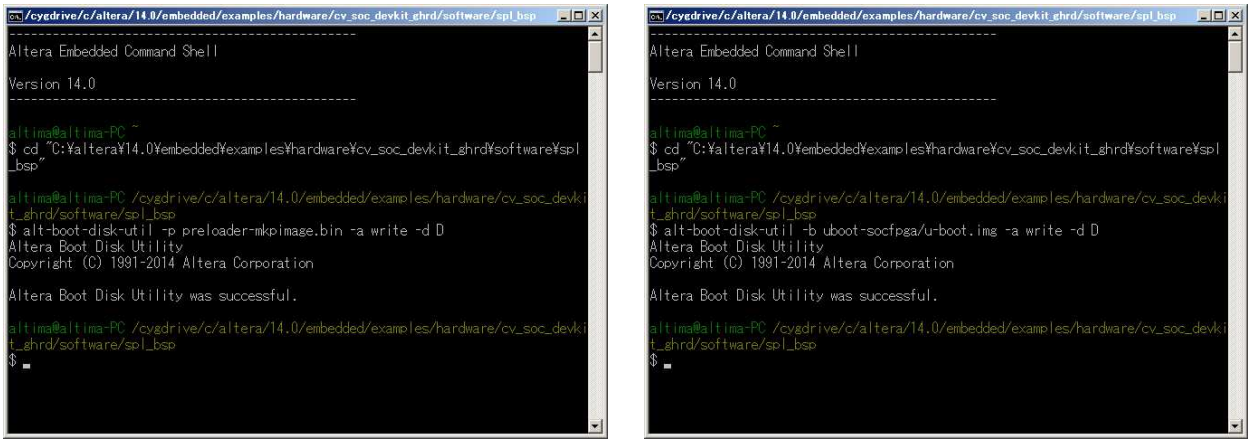

図 4-1 SD カードイメージの書き換え

Linux マシンであれば dd ユーティリティを使用することで同様の処理を実現可能です。Linux マシンにおける書き換え方法の詳細は以下のリンクをご参照ください。

GSRD v14.0 - SD Card

Updating Individual Elements on the SD card

4-2. QSPI Flash メモリの書き換え

オンボード上 の QSPI Flash メモリの場合、SoC EDS に付属の HPS Flash Programmer が使用出来ます。このツールは JTAG を介し QSPI Flash メモリに直接書き込みを実行します。

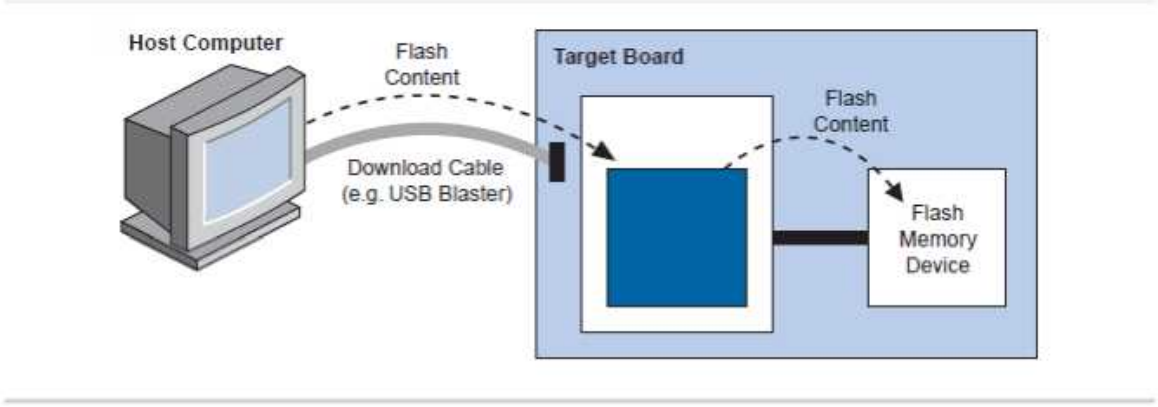

図 4-2 HPS Flash Programmer による Flash 書き込み

作業の開始前に BSEL ピンが QPSI ブートを選択していることを確認してください。

BOOTSEL Field Value	Flash Device
0x0	Reserved
0x1	FPGA (HPS-to-FPGA bridge)
0x2	1.8 V NAND flash memory
0x3	3.3 V NAND flash memory
0x4	1.8 V SD/MMC flash memory with external transceiver
0x5	3.3 V SD/MMC flash memory with internal transceiver
0x6	1.8 V SPI or quad SPI flash memory
0x7	3.3 V SPI or quad SPI flash memory

図 4-3 BSEL ピンの設定

HPS Flash Programmer にて Preloader / u-boot を書き換えるコマンドは以下の通りです。

- `quartus_hps -c 1 -o p preloader-mkpimage.bin`
- `quartus_hps -c 1 -o p -a 0x60000 uboot-socfpga/u-boot.img`

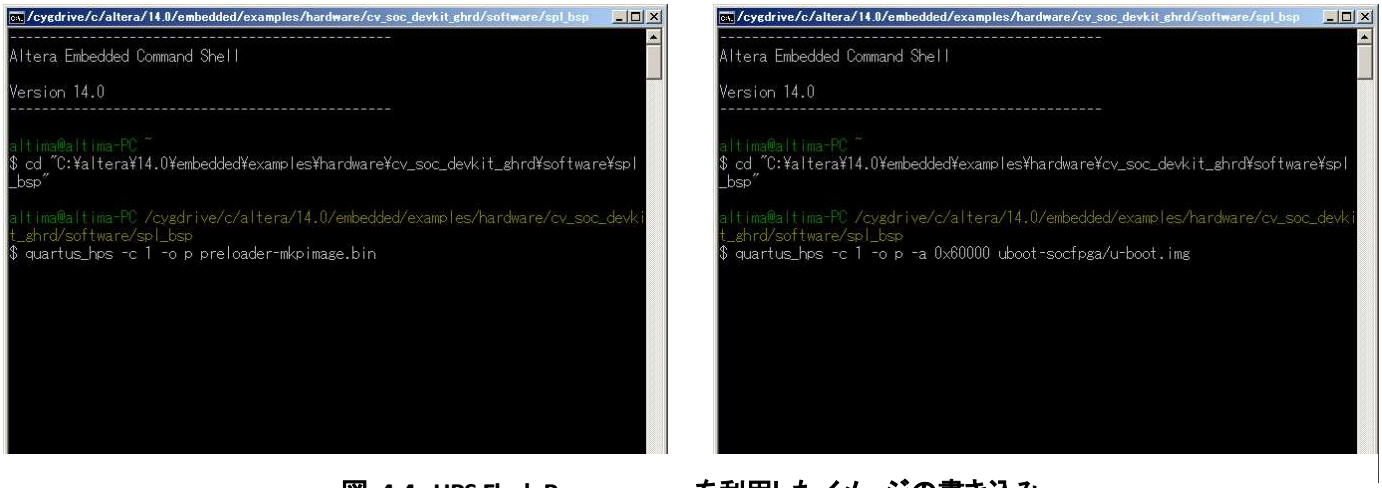

図 4-4 HPS Flash Programmer を利用したイメージの書き込み

HPS Flash Programmer に関する詳細は、「 1-2 関連文書 」で紹介した Altera SoC Embedded Design Suite User Guide を参照ください。

改版履歴

Revision	年月	概要
1	2014 年 9 月	新規作成
2	2018 年 9 月	① 書式変更 ② Windows® 10 使用の際の Preloader 生成における注記を追加 ③ リンク URL 修正
3	2020 年 1 月	① 15 ページ【**注記**】を修正 ② 20 ページ「免責およびご利用上の注意」内のリンク修正