インテル® SoC FPGA の QSPI ベアメタルアプリ・ブート（Cyclone V SoC / Arria V SoC 編）

Ver.18.1

2019 年 3 月 Rev.1

インテル® SoC FPGA の QSPI ベアメタルアプリ・ブート
（Cyclone V SoC / Arria V SoC 編）

目次

1. はじめに

本資料では Cyclone® V SoC 開発キットまたは、Arria® V SoC 開発キットに搭載されている QSPI フラッシュからベアメタルサンプル・アプリケーション **Altera-SoCFPGA-HardwareLib-Unhosted-CV-GNU** をスタンドアローン実行する例を説明しています。

このベアメタルサンプル・アプリケーションは、UART 経由で“Hello World!”メッセージを表示するだけのシンプルなアプリケーションです（ハードウェア・デザイン（sof ファイル）を FPGA にダウンロードする必要はありません）。

本資料では以下の内容を説明しています。

① ハードウェア開発での重要な生成物（ハンドオフファイル）

② SoC FPGA のブートフロー

③ ベアメタルサンプル・アプリケーションを DS-5 でビルドする方法
- DS-5 の起動
- ベアメタルサンプル・アプリケーションのインポート
- ベアメタルサンプル・アプリケーションのビルド

④ QSPI フラッシュブート用 Preloader（プリローダー）の生成方法
- Preloader（プリローダー）とは？
- QSPI フラッシュブート用 Preloader の生成手順

⑤ ベアメタル・アプリケーションを QSPI フラッシュからスタンドアローン実行する例
- Preloader とアプリケーション・イメージを QSPI フラッシュに書き込む方法
- スタンドアローン実行の動作確認

Note:
本資料では主に Cyclone® V SoC をターゲットに説明していますが、ハード・プロセッサー・システム (HPS) 部分はほぼ同一であるため Arria® V SoC にも適用されます。

本資料の説明で使用している主な開発環境を以下に示します。

【表 1-1】この資料の説明で使用している主な環境

項番	項目	内容
1	ホスト PC	Microsoft® Windows® 7 以降が動作するホスト PC 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。 **Note**: Linux も同様のコマンドで使用できます。
2	インテル® Quartus® Prime 開発ソフトウェア（以降、Quartus Prime）	SoC FPGA のハードウェアを開発するためのツールです。 この資料では、Quartus Prime 開発ソフトウェア・スタンダード・エディション v18.1 を使用しています。 ■ Quartus Prime スタンダード・エディション v18.1 **注記：** 使用するターゲットボードに搭載されている SoC FPGA に対応した Device データをインストールしておく必要があります。 Quartus Prime のインストール方法については以下のサイトをご参照ください。 Quartus® Prime & ModelSim® インストール方法（v18.x）
3	インテル® SoC FPGA エンベデッド開発スイート スタンダード・エディション（以降、SoC EDS）	SoC FPGA のソフトウェアを開発するためのツールです。 SoC EDS に含まれる Arm® Development Studio 5 Intel® SoC FPGA Edition （以降、DS-5）を使用して、アプリケーション・ソフトウェアをビルドしデバッグすることができます。 この資料では、SoC EDS スタンダード・エディション v18.1 を使用しています。 ■ SoC EDS スタンダード・エディション v18.1 **注記：** インテル® FPGA ダウンロード・ケーブル II (USB-Blaster II) を使用したベアメタル・アプリケーションのデバッグには、Arm® Development Studio 5 Intel® SoC FPGA Edition（有償版）が必要になります。 SoC EDS のインストール方法に関しては以下のサイトをご参照下さい。 SoC EDS のインストール方法（v18.x）
4	Cyclone V SoC または Arria V SoC 開発キット	本資料の説明でターゲットボードとして使用する開発キットです。 ■ Cyclone V SoC 開発キット ■ Arria V SoC 開発キット
5	ベアメタルサンプル・アプリケーション	本資料の説明で使用するベアメタルサンプル・アプリケーションです。 このベアメタル・アプリケーションは、UART 経由で “Hello World!” メッセージを表示するだけのシンプルなアプリケーションです。 実際に動作確認を行う場合は、本資料と併せて以下のファイルを取得してください。 Altera-SoCFPGA-HardwareLib-Unhosted-CVGNU.tar.gz 本資料の説明では、Altera-SoCFPGA-HardwareLib-Unhosted-CVGNU.tar.gz を **C:¥Temp** に格納したものとして説明しています。 **注記：** このベアメタルサンプル・アプリケーションを実行する際は、ハードウェア・デザイン（sof ファイル）を FPGA にダウンロードする必要はありません。
6	ターミナル・エミュレーション・ソフトウェア	このサンプルを使用するためには、シリアル・ターミナル・ソフトが必要です。 この資料では、「Tera Term」と呼ばれるフリーウェア・ソフトを使用しています。 ■ Tera Term のダウンロード URL **注記：** Tera Term では、ターゲットボードの UART と接続した際の有効な COM ポートに対して、以下の設定を行ってください。 ・ ボーレート 115200 bps ・ 8 ビットデータ ・ パリティなし ・ 1 ストップビット ・ フロー制御なし

Note:
本資料は、Quartus Prime、 SoC EDS、 bsp-editor（Preloader Generator）、および DS-5 の基本的な知識を前提としています。

参考:

- SoC FPGA のブートに関する基本操作については、以下のユーザーガイドが参考になります。
  - 『HPS SoC Boot Guide - Cyclone V SoC Development Kit AN-709』（英語版）
- SoC FPGA の QSPI ブート情報については、以下のページを参照ください。
  - 『GSRD v13.1 - Booting from QSPI』（英文ページ）
  - 『CV SoC and AV Soc QSPI Boot』（英文ページ）
- SoC FPGA のベアメタルに関する基本操作については、以下のユーザーガイドが参考になります。
  - 『Bare Metal User Guide UG-01165』（英語版）
  - 『ベアメタルのユーザーガイド UG-01165』（日本語版）
  - 『UG-01165: Bare Metal User Guide --> Errata - Intel』（英文ページ）
  - 『SoC はじめてガイド - DS-5 によるベアメタル・アプリケーション・デバッグ』（日本語版）
- SoC FPGA のベアメタル開発者向け情報については、以下のページを参照ください。
  - 『Intel SoC FPGA Bare-metal Developer Center』（英文ページ）
- SoC FPGA のベアメタル・プログラミングとハードウェア・ライブラリに関する無償オンライン・トレーニングは、以下のページを参照ください。
  - 『SoC Bare-metal Programming and Hardware Libraries - Intel』（英語、28 分）

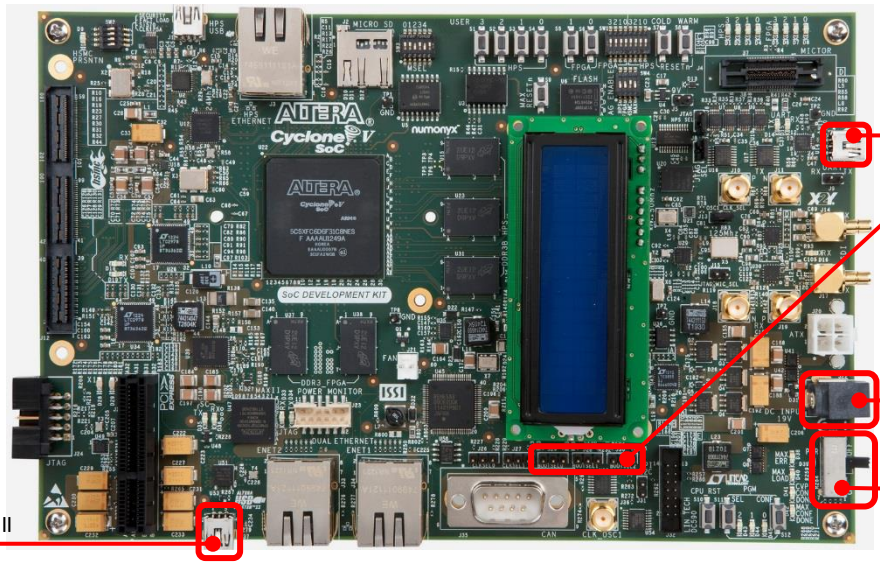
2. 事前準備

本資料では ターゲットボードとして Cyclone V SoC 開発キットを例として説明しています。
ここでは、上記ボードを使用する際に必要なボード設定および FPGA デザインファイルについて説明します。

2-1. ボードの設定

2-1-1. ボードレイアウト

Cyclone V SoC 開発キットのレイアウト図を以下に示します。



【図 2-1】 Cyclone V SoC 開発キットレイアウト図

2-1-2. 電源およびケーブルの接続

AC アダプターの接続や各種ケーブルは以下の通り接続してください。

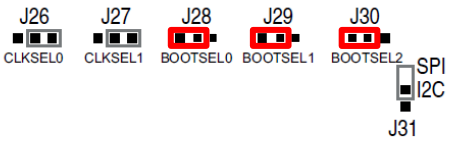
- 電源（AC アダプター）を DC 入力（J22）に接続します。
- Mini USB ケーブルでホスト PC とオンボード USB-Blaster II コネクター（J37）を接続します。
- Mini USB ケーブルでホスト PC と UART コネクター（J7）を接続します。

2-1-3. BSEL（BOOTSEL）ピンの設定

J28－J30（BSEL ピン）が以下の通り設定されていることを確認します。
この設定により、HPS は QSPI ブートとなります。

【表 2-1】BSEL ピンの設定

ボード・リファレンス	信号名	設定
J28	BOOTSEL0	左
J29	BOOTSEL1	左
J30	BOOTSEL2	左



【図 2-2】 BSEL ピンの設定

参考:

- Cyclone V SoC 開発キットに関する情報については、以下の資料が参考になります。
  - 『Cyclone V SoC Development Kit User Guide』（英語版）
  - 『Cyclone V SoC 開発キット・ユーザーガイド』（日本語版）
  - 『Cyclone V SoC Development Board Reference Manual』（英語版）
  - 『Cyclone V SoC 開発ボード リファレンス・マニュアル』（日本語版）
- Arria V SoC 開発キットに関する情報については、以下の資料が参考になります。
  - 『Arria V SoC Development Kit User Guide』（英語版）
  - 『Arria V SoC Development Board Reference Manual』（英語版）

2-2. ハードウェア・デザインファイル

「5. QSPI フラッシュブート用 Preloader（プリローダー）の生成方法」で説明する Preloader を生成するためには、ハードウェア開発で生成した“ハンドオフファイル”が必要になります。

本資料の説明では C:¥intelFPGA¥18.1 にインストールした SoC EDS に付属のハードウェア・デザインを使用しています。

2-2-1. ハードウェア・デザインファイルの格納先

SoC EDS をインストールするとデフォルトでは以下の場所に格納されます。

① Cyclone V SoC 開発キット
C:¥intelFPGA¥18.1¥embedded¥examples¥hardware¥cv_soc_devkit_ghrd

② Arria V SoC 開発キット
C:¥intelFPGA¥18.1¥embedded¥examples¥hardware¥av_soc_devkit_ghrd

Note:
本資料の説明では SoC EDS を C:¥intelFPGA¥18.1 にインストールしたものとして説明しています。

2-2-2. ハードウェア開発での重要な生成物（ハンドオフファイル）

ベアメタル・アプリケーションの開発およびデバッグでは、ハードウェアの開発において最終的に生成されたフォルダーとファイルを使用します。

これらのフォルダーとファイルを「ハンドオフファイル」と呼びます。

開発キットのリファレンス・デザインを使用した場合は、正しく生成されていれば **<Quartus プロジェクト>¥hps_isw_handoff¥soc_system_hps_0** フォルダーの中にツールによって生成されたハードウェア・ソフトウェアのハンドオフファイルがあります。

これらのファイルは、「5-2. Preloader の生成手順」に利用します。Preloader 生成のために使用する bsp-editor（Preloader Generator）ツールで、この hps_isw_handoff¥soc_system_hps_0 フォルダーのパスを指定するので覚えておいてください。

3. SoC FPGA のブートフロー

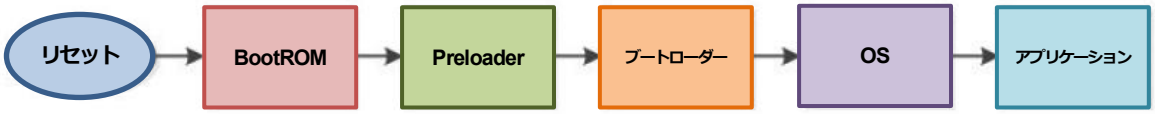
まず、はじめに SoC FPGA のブートフローについて説明します。

> **参考:**
>
> ブートフローに関する詳細については、以下のユーザーガイドが参考になります。
> 『HPS SoC Boot Guide - Cyclone V SoC Development Kit AN-709』（英語版）

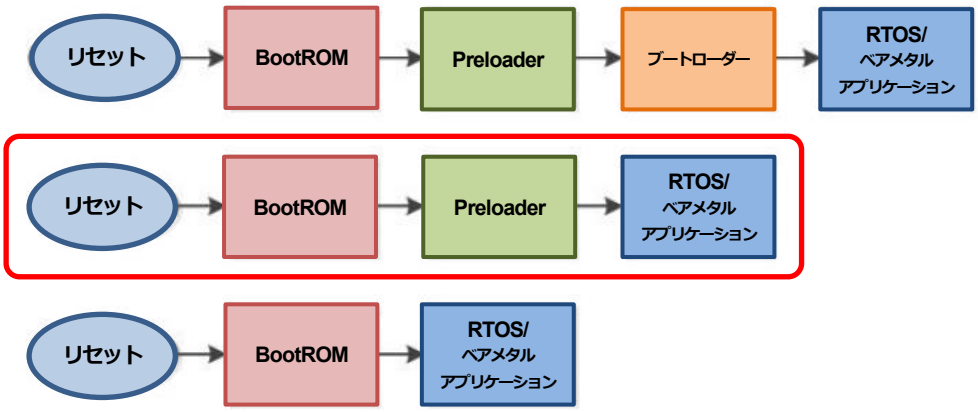
以下の図の通り、SoC FPGA のブートフローには複数のステージが存在します。

ベアメタル・アプリケーションの場合の多くは、以下赤枠で示した Preloader から直接ベアメタル・アプリケーションを起動する方法が用いられます。

本資料でもこのブートフローを実現するための仕組みについて解説しています。



【図 3-1】 一般的なブートフロー



【図 3-2】 追加のブートフロー

- **BootROM**

  SoC FPGA の内蔵オンチップ ROM に焼き込まれているブートコードです（ユーザーによる変更不可）。

- **Preloader**

  ハンドオフファイルの情報を元に、HPS IO や SDRAM コントローラの初期化など動作するために必要な処理を実行します。

- **ブートローダー**

  ユーザー・ブートローダー・コード（U-Boot など）です。アプリケーションや OS に依存します。

- **ベアメタル・アプリケーション**

  OS を使わないアプリケーションをベアメタル・アプリケーションと呼びます。インテル® SoC FPGA のハードウェア・ライブラリ（HWLib）を使用して、直接ハードウェアを読み書きするベアメタル・アプリケーションを作成することができます。

4. ベアメタルサンプル・アプリケーションを DS-5 でビルドする方法

この章では、ベアメタルサンプル・アプリケーション・プロジェクトを DS-5 にインポートして、ビルドする方法について説明します。

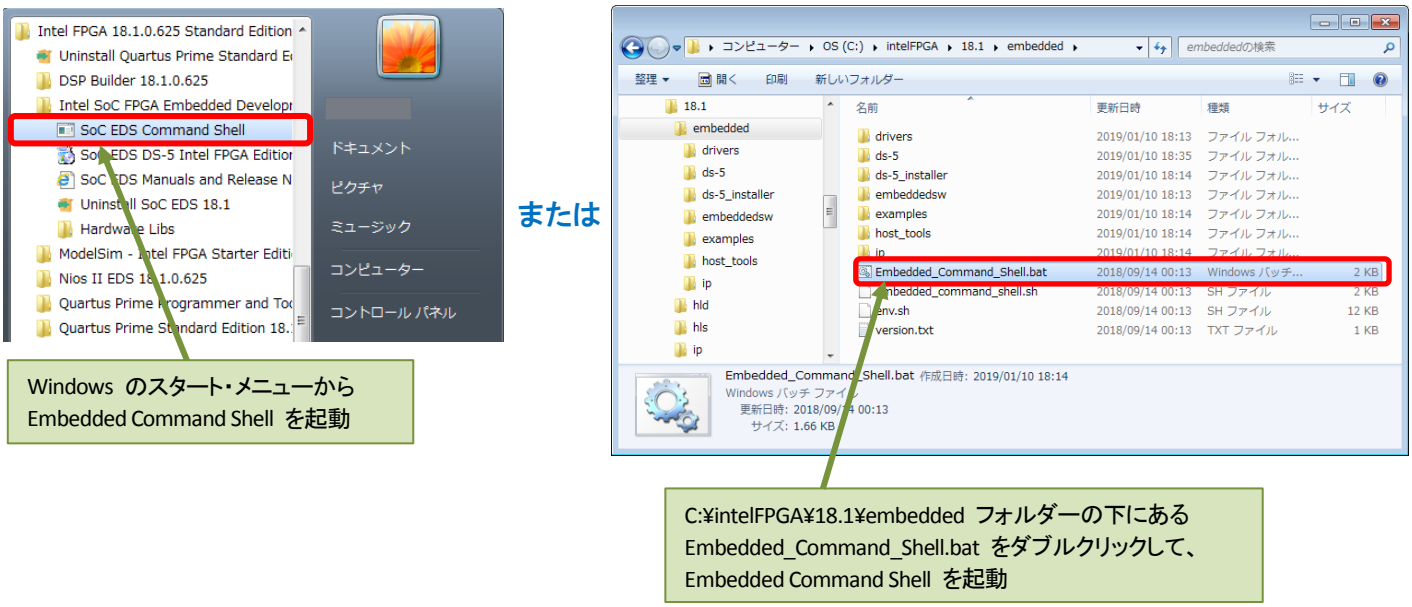
4-1. DS-5 の開始

SoC EDS に含まれている Arm® Development Studio 5 Intel® SoC FPGA Edition (DS-5) を起動します。

SoC EDS に対する各種環境設定を自動的に実施するために、DS-5 は次の Embedded Command Shell から起動してください。

4-1-1. Embedded Command Shell の起動

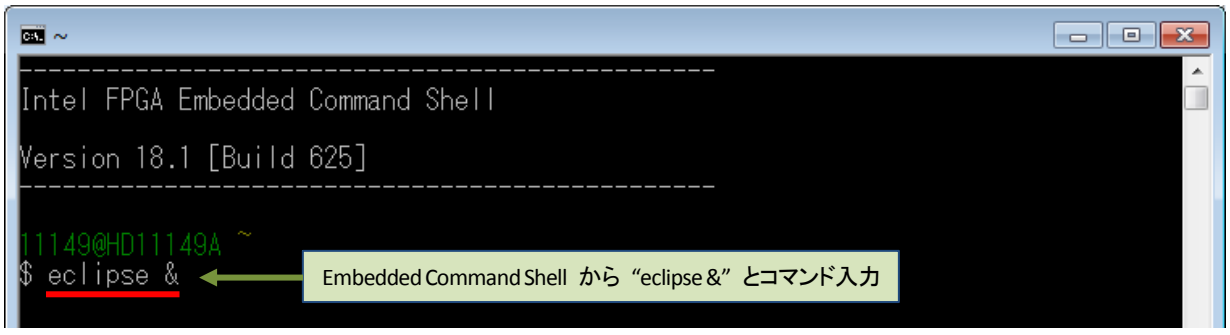
Windows のスタート・メニュー または SoC EDS のインストール・フォルダー（embedded フォルダー）下に格納されている起動用スクリプトを実行し、Embedded Command Shell を起動します。



【図 4-1】 Embedded Command Shell の起動

4-1-2. DS-5 の起動

（1）下図のように Embedded Command Shell のウィンドウが開いたら **eclipse &** ↵ とコマンド入力して DS-5 を起動します。



【図 4-2】 DS-5 の起動

(2) ワークスペース・フォルダーの入力を求められます。ソフトウェア・プロジェクトのために固有のワークスペースを選択または作成します。

パスを指定して［**OK**］をクリックします（この例では、ワークスペースに C:¥Work¥DS-5 Workspace を指定しています。フォルダーが存在しない場合は自動的に作成されます）。

【図 4-3】 DS-5 のワークスペースの指定

(3) DS-5 ウェルカム画面が表示される場合は、［**閉じる**］（**× マーク**）をクリックします。

DS-5 ウェルカム画面は、ドキュメント、チュートリアルやビデオにアクセスするために使用することができます。

【図 4-4】 DS-5 ウェルカム画面

⚠ 注記：

ここで、「**ツールキットが選択されていません**」のプロンプトが出た場合は、［**後で通知する**］をクリックして先に進みます。

また、「**Windows セキュリティの重要な警告**」のプロンプトが出た場合は、［**アクセスを許可する(A)**］をクリックします。

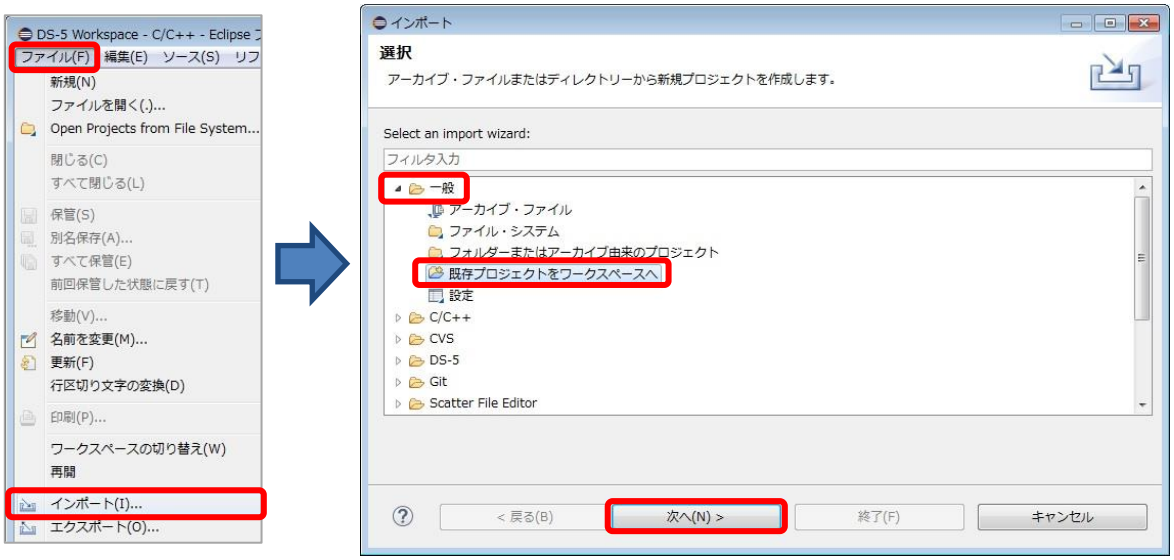
4-2. ベアメタルサンプル・アプリケーションのインポート

この例では、事前にダウンロードしておいたベアメタルサンプル・アプリケーション **Altera-SoCFPGA-HardwareLib-Unhosted-CV-GNU** を DS-5 にインポートします。

このベアメタル・アプリケーションは、UART 経由で“Hello World!”メッセージを表示するだけのものです。

(1) DS-5 のメニューから「**ファイル(F)**」⇒「**インポート(I)...**」を選択します。

(2)「**一般**」⇒「**既存プロジェクトをワークスペースへ**」を選択し、[**次へ(N)**] をクリックします。

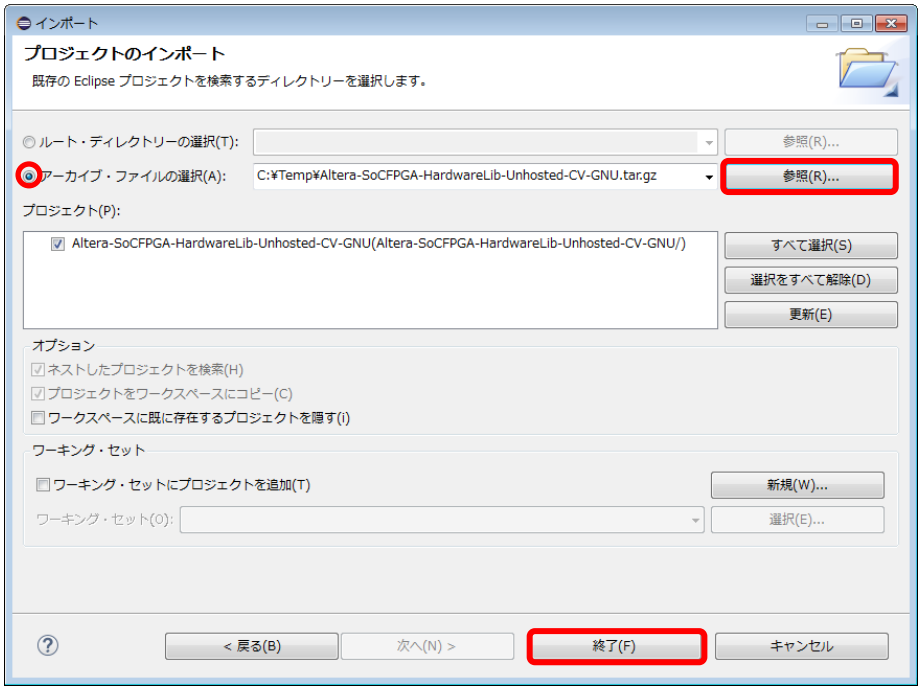


【図 4-5】 既存プロジェクトのインポート

(3)「**アーカイブ・ファイルの選択(A):**」オプションを選択し、[**参照(R)**] ボタンより **Altera-SoCFPGA-HardwareLib-Unhosted-CV-GNU.tar.gz** 選択後、[**終了(F)**] ボタンを押します。

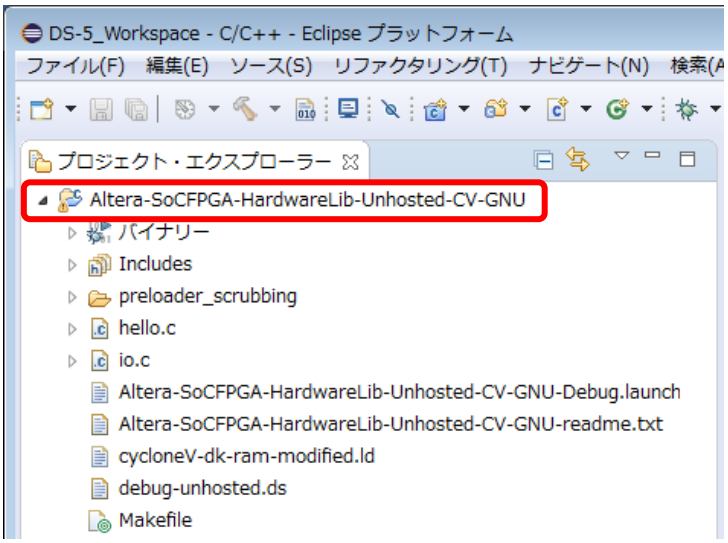
Note:

*本資料の説明では、**Altera-SoCFPGA-HardwareLib-Unhosted-CV-GNU.tar.gz** を **C:¥Temp** に格納したものとして説明しています。*



【図 4-6】 サンプル・アプリケーションの選択

(4) DS-5 画面左側の**プロジェクト・エクスプローラー**パネルにインポートしたベアメタルサンプル・アプリケーション・プロジェクト **Altera-SoCFPGA-HardwareLib-Unhosted-CV-GNU** が追加され、Altera-SoCFPGA-HardwareLib-Unhosted-CV-GNU を展開すると、プロジェクトに含まれる各種ファイルが表示されます。



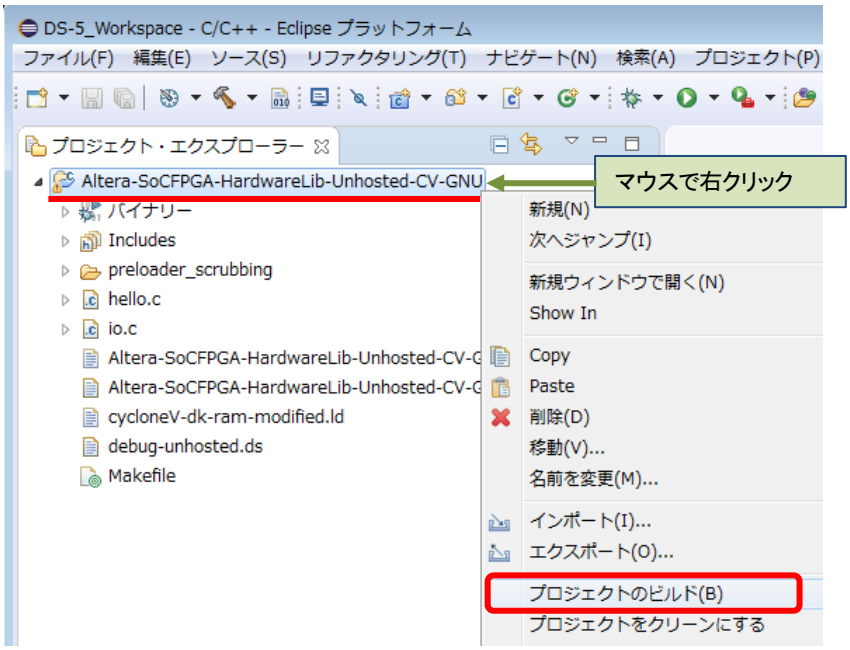
【図 4-7】 インポートにより追加されたプロジェクト

4-3. ベアメタルサンプル・アプリケーションのビルド

次にインポートしたベアメタルサンプル・アプリケーション・プロジェクトをビルドして実行できるようにします。

4-3-1. プロジェクトのビルド

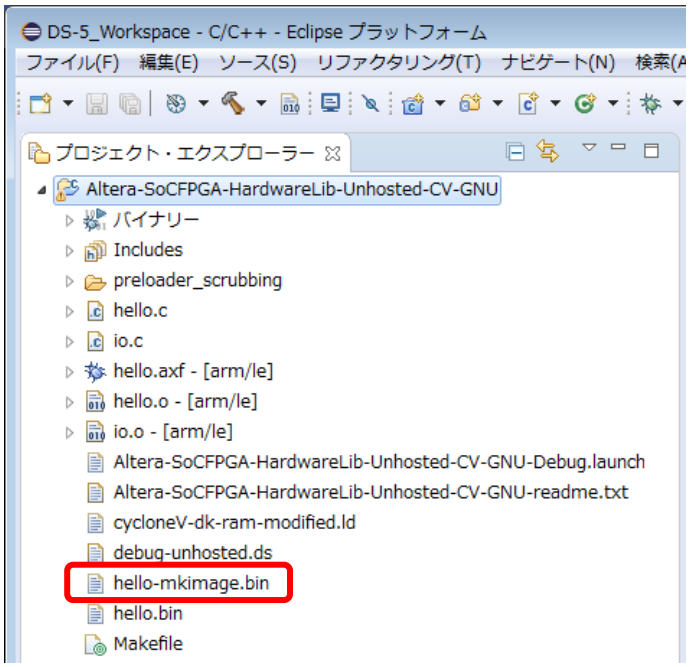
DS-5 プロジェクト（この例では、Altera-SoCFPGA-HardwareLib-Unhosted-CV-GNU）をハイライトし、右クリックして「**プロジェクトのビルド(B)**」を実行します。



【図 4-8】 プロジェクトのビルド

この資料の説明で使用しているベアメタルサンプル・アプリケーション・プロジェクト Altera-SoCFPGA-HardwareLib-Unhosted-CV-GNU では、Makefile 内での指示によりビルドが完了すると、ベアメタル・アプリケーションの hello.bin ファイルから、次節「4-3-2. mkimage ヘッダーの追加」で説明する mkimage ヘッダーを追加した **hello-mkimage.bin** ファイルが生成されるようになっています。これは、Preloader によってロードされるベアメタル・アプリケーション・イメージです。

このアプリケーション・イメージと、「5. QSPI フラッシュブート用 Preloader（プリローダー）の生成方法」で説明する Preloader を QSPI フラッシュに書き込みます。



【図 4-9】 生成されたベアメタル・アプリケーション・イメージ hello-mkimage.bin

4-3-2. mkimage ヘッダーの追加

Preloader からロード・実行する次のステージのブートイメージ・ファイルには、U-Boot ユーティリティ・ツール mkimage を使用して mkimage ヘッダー情報を付加する必要があります。

Preloader は、次のステージのブートイメージをロードする前に、ブートイメージに付加されたヘッダーを検証します。

この資料の説明で使用しているベアメタルサンプル・アプリケーション・プロジェクト Altera-SoCFPGA-HardwareLib-Unhosted-CV-GNU では、Makefile 内での指示により自動的にベアメタル・アプリケーションに mkimage ヘッダー情報を付加した hello-mkimage.bin が生成されましたが、通常は Embedded Command Shell から、次の例のようなコマンドを入力して、バイナリーファイルに mkimage ヘッダー情報を付加します。

```
$ mkimage -A arm -T standalone -C none -a 0x100040 -e 0 -n "baremetal image"
-d hello.bin hello-mkimage.bin
```

注記:
上記コマンド例は、アプリケーションのスタート・アドレスが 0x100040 から作成されている場合の例です。
アプリケーションのスタート・アドレスは、リンカ・スクリプト・ファイル（.ld）内の **ORIGIN** で定義されています。

5. QSPI フラッシュブート用 Preloader（プリローダー）の生成方法

この章では、QSPI フラッシュからブートするために必要な Preloader の生成手順について説明します。

5-1. Preloader（プリローダー）とは？

Preloader は U-boot second program loader（以後、u-boot spl）をベースに、SoC FPGA 向けにカスタマイズが加えられたブートローダーです。

① Preloader の役割は次の通りです。

- 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 へ次ステージのプログラムの展開・ジャンプ

② Preloader は Quartus Prime / Platform Designer 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、ユーザー側で初期化用ソフトウェアの構築をすることなく Quartus Prime / Platform Designer 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。

③ ユーザーの SoC FPGA を搭載したカスタムボードを動かすためには、まずこの Preloader を必ず生成してください。

5-2. Preloader の生成手順

以降に Preloader の生成手順を説明します。

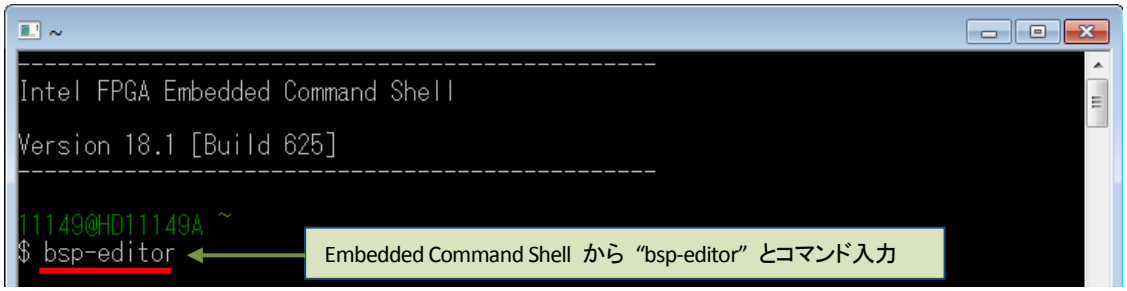
> **参考:**
> Preloader の生成方法についての更に詳しい説明は、以下の資料をご覧ください。
> 『Preloader Generator の使用方法』

5-2-1. Embedded Command Shell の起動

「4-1-1. Embedded Command Shell の起動」と同じ手順で起動します。
Windows のスタート・メニューまたは、SoC EDS のインストール・フォルダー（embedded フォルダー）に格納されている起動用スクリプトを実行し、Embedded Command Shell を起動します。

5-2-2. bsp-editor（Preloader Generator）の起動

下図のように Embedded Command Shell のウィンドウが開いたら **bsp-editor** ↵ とコマンド入力して、bsp-editor（Preloader Generator）の GUI を起動します。

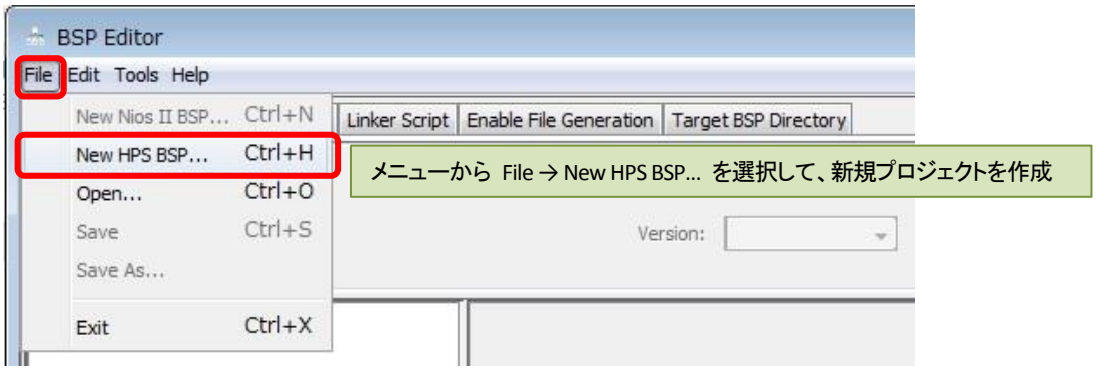


【図 5-1】 bsp-editor（Preloader Generator）の起動

5-2-3. 新規 bsp プロジェクトの作成

図のように bsp-editor（Preloader Generator）の GUI が起動したら、
メニューから「**File**」⇒「**New HPS BSP…**」を選択して、新規プロジェクトを作成します。

> **注記:**
> **SoC EDS v15.0 より前** のバージョンでは、「**File**」⇒「**New BSP…**」を選択します。



【図 5-2】 新規 bsp プロジェクトの作成

5-2-4. ハンドオフファイルの指定

（1）ハードウェア開発で生成した、ハンドオフファイル・フォルダーのパス
<Quartus プロジェクト>¥hps_isw_handoff¥soc_system_hps_0 を指定します。

図のように Preloader settings directory: の並びにある [...] を押してフォルダーを指定します。

Note:

本資料の説明では、C:¥intelFPGA¥18.1 にインストールした SoC EDS に付属のハードウェア・デザインを使用しているため、下図のパスを設定しています。

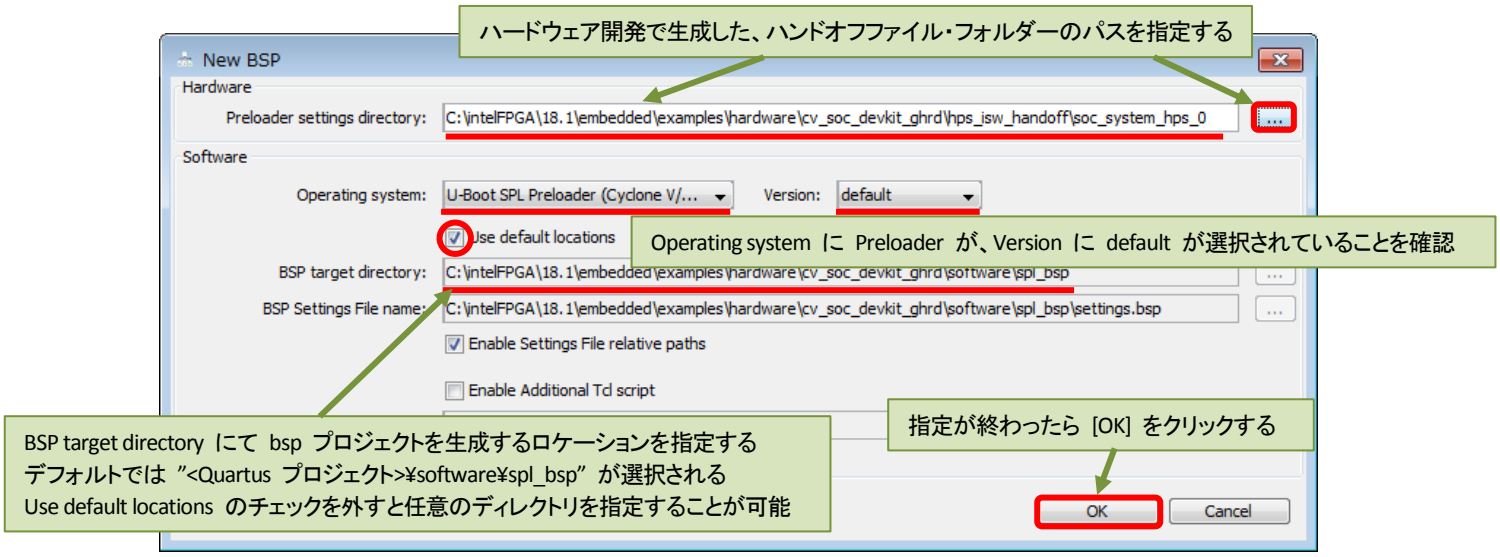
- Cyclone V SoC 開発キットの場合は、

  C:¥intelFPGA¥18.1¥embedded¥examples¥hardware¥**cv_soc_devkit_ghrd**¥hps_isw_handoff¥**soc_system_hps_0**

- Arria V SoC 開発キットの場合は、

  C:¥intelFPGA¥18.1¥embedded¥examples¥hardware¥**av_soc_devkit_ghrd**¥hps_isw_handoff¥**ghrd_5astfd5k3_hps_0**

（2）全ての指定が終わったら［**OK**］をクリックします。



【図 5-3】 ハンドオフファイルの指定

5-2-5. Preloader のユーザーオプション（Common ⇒ spl ⇒ boot）の設定

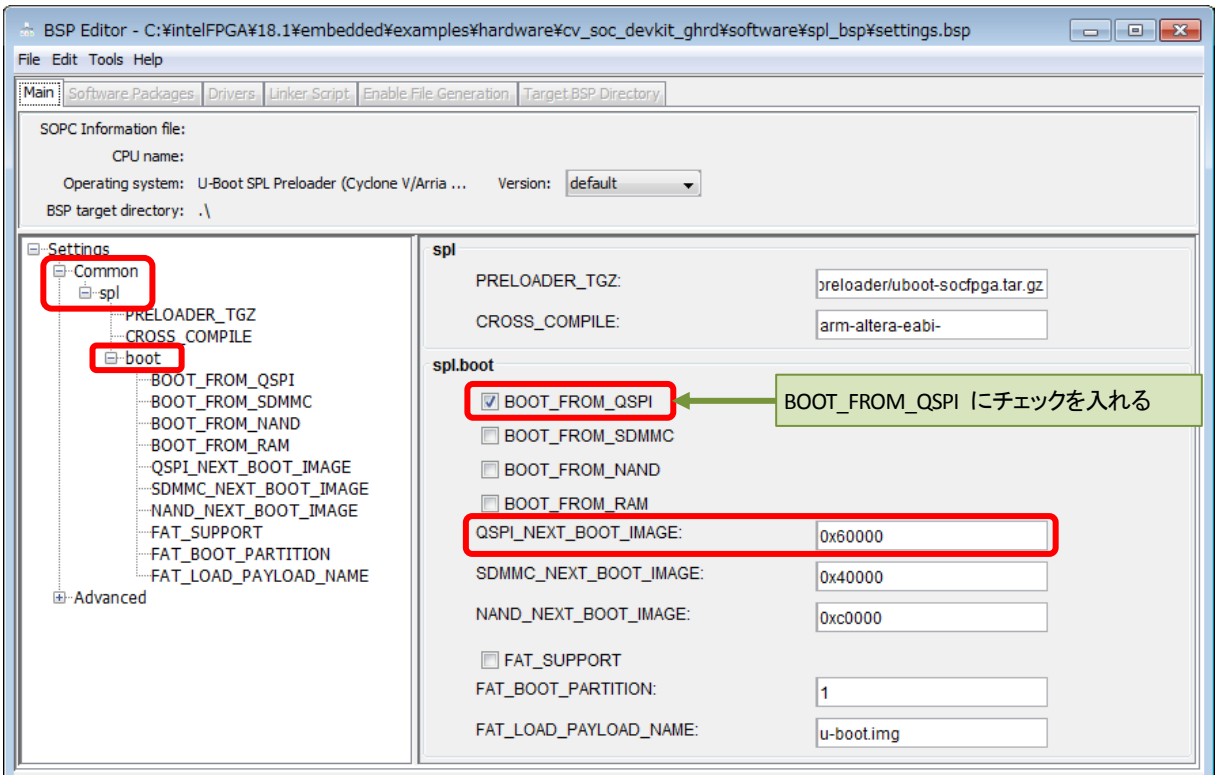
Common ⇒ spl ⇒ boot では Preloader に続くブートイメージを QSPI からロードする設定と、Preloader がロードするブートイメージの格納アドレスを指定します。

(1) オプション **BOOT_FROM_QSPI** にチェックを入れます。この設定により Preloader に続くブートイメージを QSPI からロードします。

(2) その他のブートオプション（**BOOT_FROM_RAM**、**BOOT_FROM_SDMMC**、**BOOT_FROM_NAND**）のチェックを外します。

> **注記:**
> BOOT メモリの選択は、いずれか 1 つにのみチェックを入れてください（複数にチェックを入れない）。

(3) **QSPI_NEXT_BOOT_IMAGE** = **0x60000** であることを確認します。これは、ブートイメージ（ベアメタル・アプリケーション・イメージ）を保存する必要があるアドレスです。 Preloader がこのアドレスに格納されているブートイメージをロードします。



【図 5-4】 Preloader のユーザーオプション（Common ⇒ spl ⇒ boot）の設定

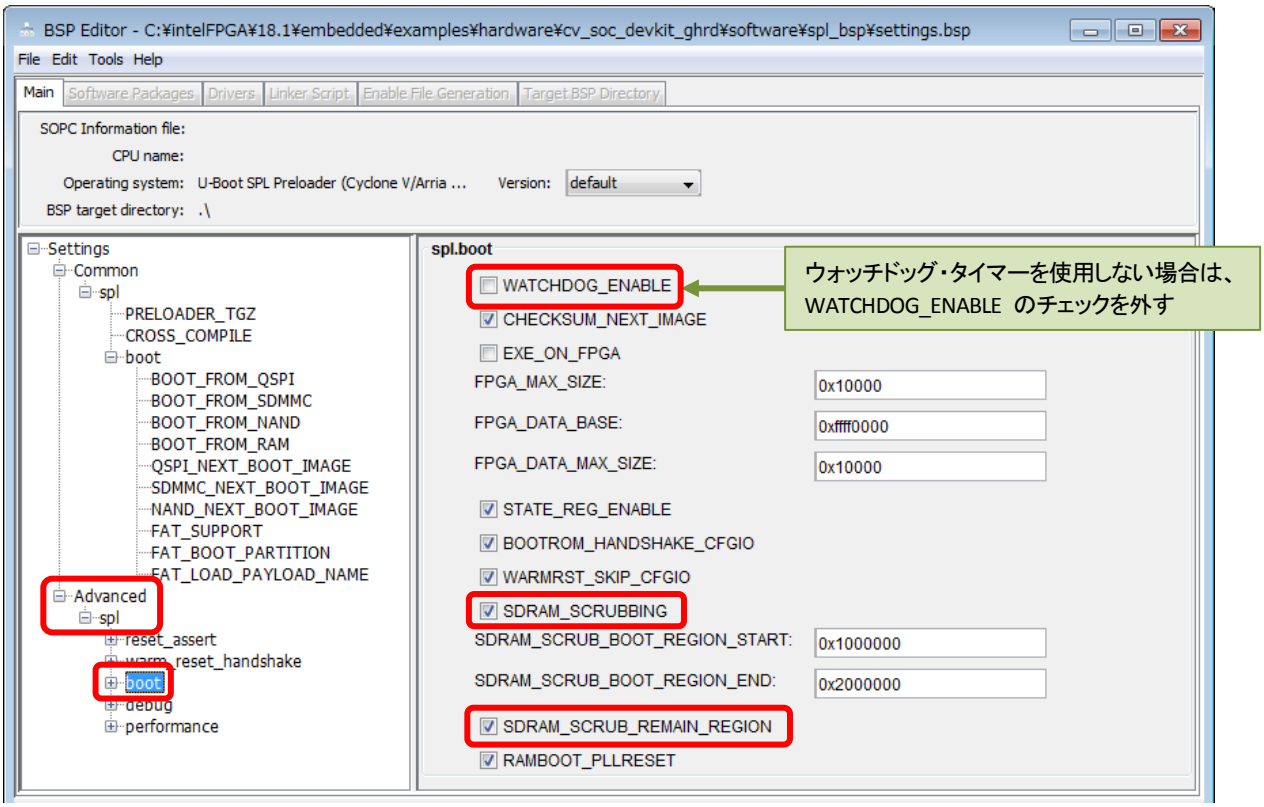
5-2-6. Preloader のユーザーオプション（Advanced ⇒ spl ⇒ boot）の設定

Advanced ⇒ spl ⇒ boot ではウォッチドッグ・タイマーの Disable などブート時の挙動に関して設定を行います。

(1) オプション **WATCHDOG_ENABLE** のチェックを外します。これは、ベアメタル・アプリケーションでウォッチドッグ・タイマーをキックしていないからです。

> **注記:**
> ベアメタル・アプリケーションでウォッチドッグ・タイマーを使用する場合は、ウォッチドッグ・タイマーが正常に動作するようにプログラム・コードを追加してください。

(2) オプション **SDRAM_SCRUBBING** と **SDRAM_SCRUB_REMAIN_REGION** にチェックを入れます。これにより SDRAM がゼロクリアーされ、ベアメタル・プログラムの実行中に ECC エラーが発生するのを防ぎます。



【図 5-5】 Preloader のユーザーオプション（Advanced ⇒ spl ⇒ boot）の設定

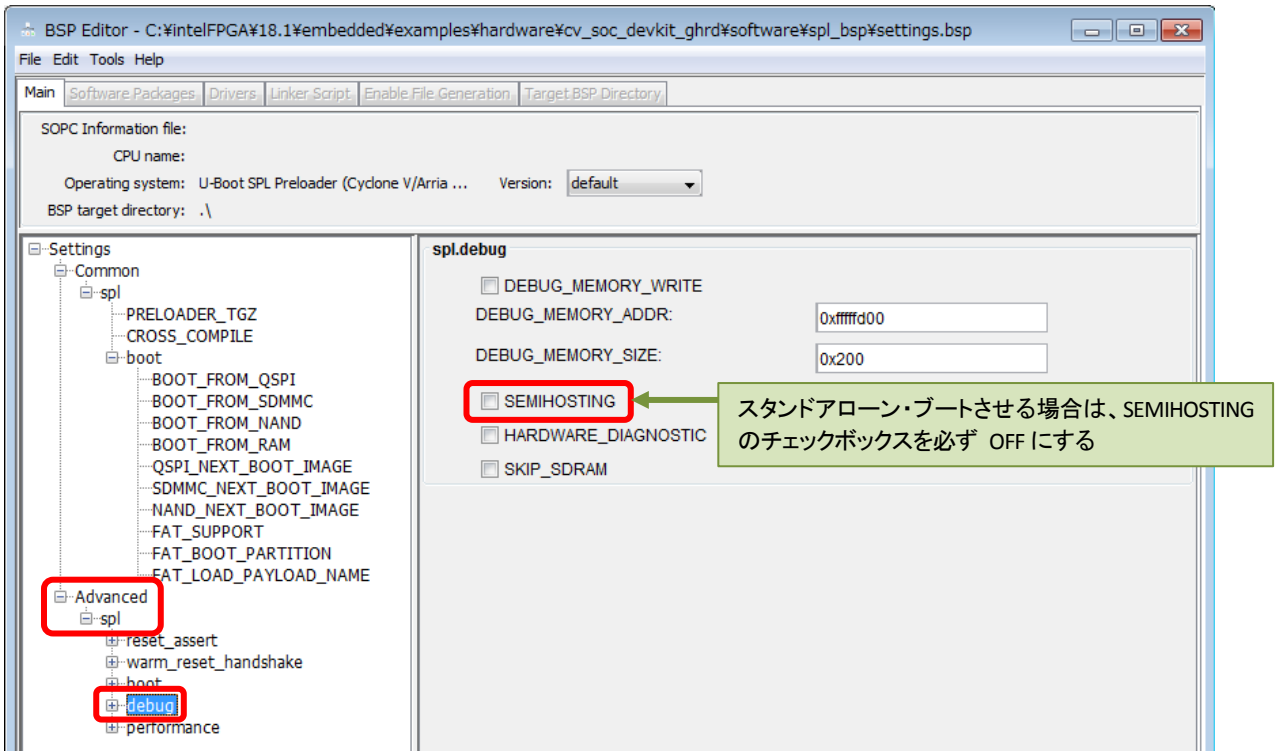
5-2-7. Preloader のユーザーオプション（Advanced ⇒ spl ⇒ debug）の設定

Advanced ⇒ spl ⇒ debug では DS-5 の Semihosting 機能のサポート有無等、デバッグ関連の設定を行います。

ベアメタル・アプリケーションをスタンドアローン・ブートさせる場合は、**SEMIHOSTING** のチェックボックスを **OFF** にします。

Arm® Development Studio 5 Intel® SoC FPGA Edition の Semihosting 機能を使用して Preloader をデバッグする場合は、SEMIHOSTING のチェックボックスを ON にします。

> **注記:**
> DS-5 を使用せずに、スタンドアローン・ブートさせる場合は、**SEMIHOSTING** のチェックボックスを必ず **OFF** にしてください。



【図 5-6】 Preloader のユーザーオプション（Advanced ⇒ spl ⇒ debug）の設定

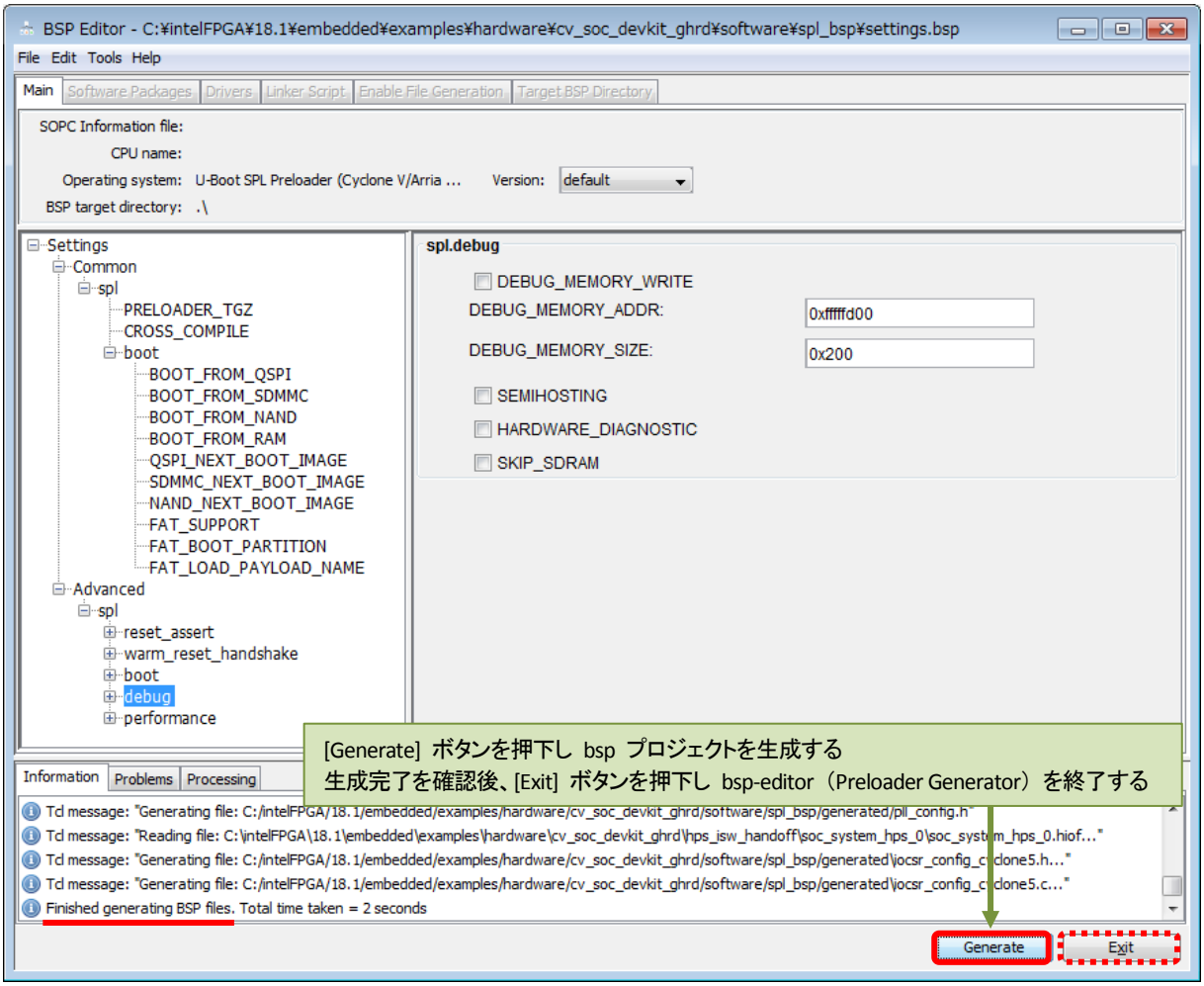
5-2-8. bsp プロジェクトの生成（Generate）

右下の［**Generate**］ボタンを押下し bsp プロジェクトを生成します。

生成する 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が保存されます。

これらのファイルは、「5-2-4. 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生成されます（例では、**<Quartus プロジェクト>¥software¥spl_bsp**）。

生成完了を確認後、[**Exit**］ボタンを押下し bsp-editor（Preloader Generator）を終了します。



【図 5-7】 bsp プロジェクトの生成

5-2-9. Preloader のビルド

（1）Embedded Command Shell のカレント・ディレクトリを、bsp-editor（Preloader Generator）で作成した bsp プロジェクトのディレクトリに移動します。

Embedded Command Shell から 以下のようにコマンド入力します。

```
$ cd "<quartus プロジェクト>¥software¥spl_bsp" ↵
```

Note:

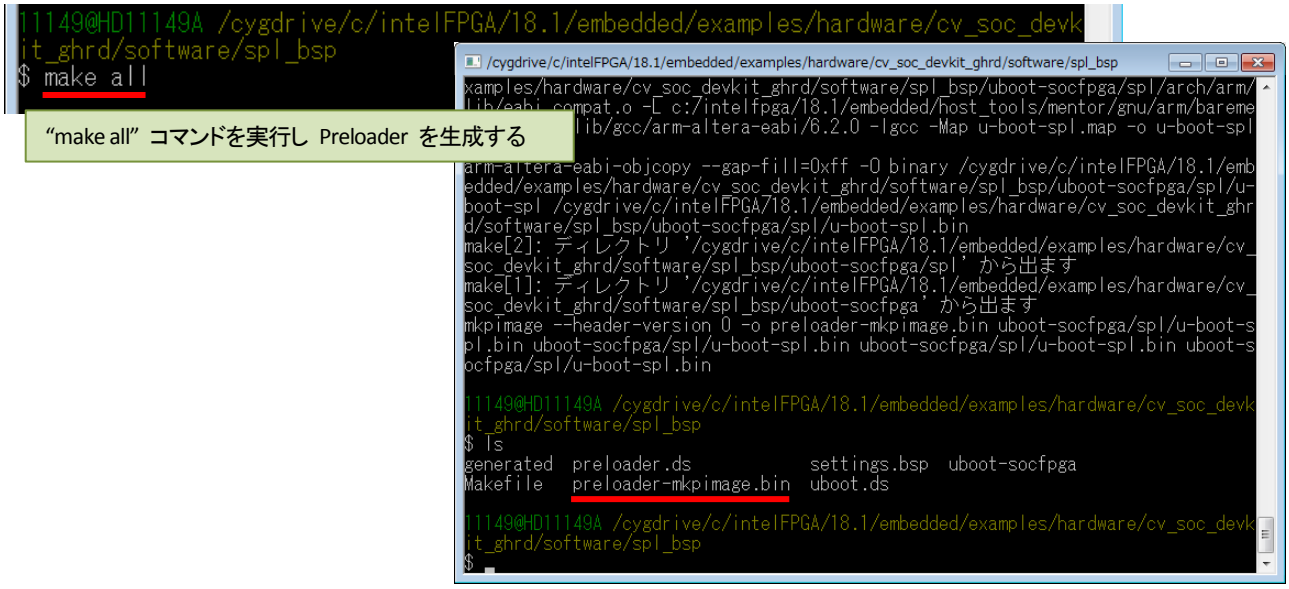
本資料の説明では、C:¥intelFPGA¥18.1 にインストールした SoC EDS に付属のハードウェア・デザインを使用しているため、下図のディレクトリに移動しています。

- Cyclone V SoC 開発キットの場合は、
  C:¥intelFPGA¥18.1¥embedded¥examples¥hardware¥**cv_soc_devkit_ghrd**¥software¥spl_bsp
- Arria V SoC 開発キットの場合は、
  C:¥intelFPGA¥18.1¥embedded¥examples¥hardware¥**av_soc_devkit_ghrd**¥software¥spl_bsp

【図 5-8】 bsp プロジェクトのディレクトリに移動

（2）**make all** ↵ コマンドを実行し Preloader を生成します。

ls ↵ コマンドにて **preloader-mkpimage.bin** 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 用のヘッダー情報を付加したバイナリーファイルで、QSPI フラッシュへ書き込むファイルとなります。



【図 5-9】 “make all” コマンドを実行

注記:

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。

【エラー内容】

この問題は、SoC EDS ツールを使用してプリローダーを生成するときに発生します。
新しい HPS および BSP 設定ファイルを作成した後、以下のように make コマンドが失敗します。

```
tar zxf /cygdrive/c/intelFPGA/18.0/embedded/host_tools/altera/preloader/uboot-socfpga.tar.gz
tar: Error opening archive: Failed to open '/cygdrive/c/intelFPGA/18.0/embedded/host_tools/altera/preloader/uboot-socfpga.tar.gz'
make: *** [uboot-socfpga/.untar] Error 1
```

もしご使用の OS が Windows® 10 でエラーが発生する場合は、以下の参考情報サイトで説明されている対策が必要となりますのでご注意ください。

【参考情報サイト】

- Intel® Knowledge Base - Unable to make preloader in Windows 10
- MACNICA フォーラム「SoC はじめてガイド - DS-5 によるベアメタル・アプリケーション・デバッグについて」

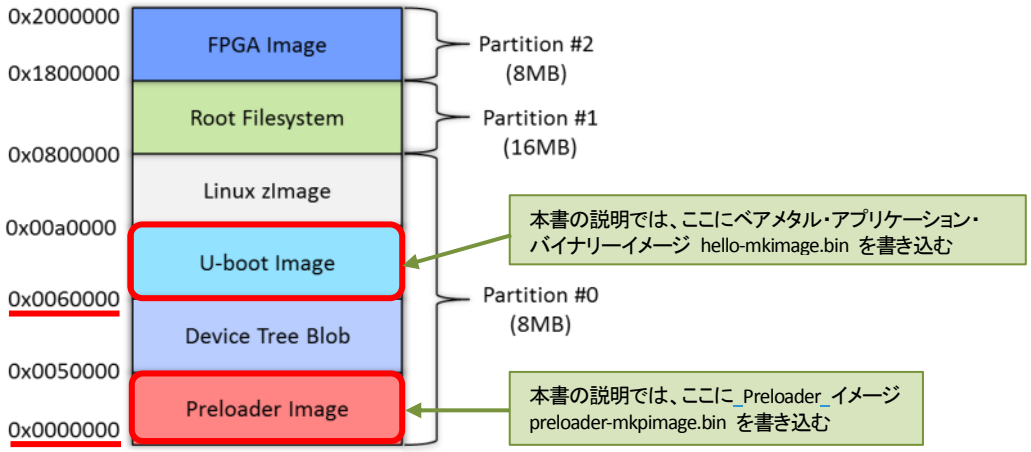
6. ベアメタル・アプリケーションを QSPI フラッシュからスタンドアローン実行する例

この章では、ベアメタル・アプリケーションを QSPI フラッシュからスタンドアローン実行できるようにするために必要な手順について説明します。

6-1. QSPI フラッシュのレイアウト

次の図は Cyclone V SoC 開発キットの QSPI フラッシュレイアウトを詳細に示したものです。 図の中で注意すべき項目は次のとおりです。

- 本書の説明では、Preloader イメージ 0（0x0 番地）に、**preloader-mkpimage.bin** を書き込みます。
- 本書の説明では、次のブートイメージである U-Boot イメージ領域（0x60000 番地）に、ベアメタル・アプリケーション・バイナリーイメージ **hello-mkimage.bin** を書き込みます。



【図 6-1】 QSPI フラッシュのレイアウト

6-2. BSEL（BOOTSEL）ピン設定の確認

「2-1-3. BSEL（BOOTSEL）ピンの設定」に従って、J28 ～J30 の BSEL ピンが QSPI ブートに設定されていることを確認してください。

6-3. Preloader とアプリケーション・イメージを QSPI フラッシュに書き込む方法

QSPI フラッシュへの書き込みには、**HPS フラッシュプログラマー・ユーティリティー**を使用します。

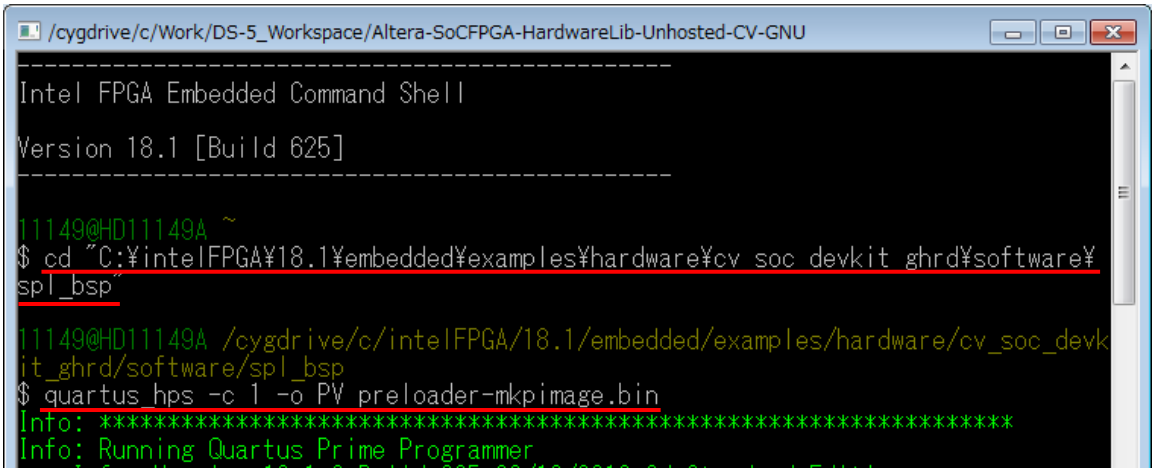
HPS フラッシュプログラマーは、フラッシュの消去、ブランクチェック、プログラミング、検証、検査が可能です。このユーティリティーは、必要な「.bin」拡張子を持つバイナリーファイルを受け入れます。

以下は、HPS フラッシュプログラマーのコマンドライン・シンタックスです。

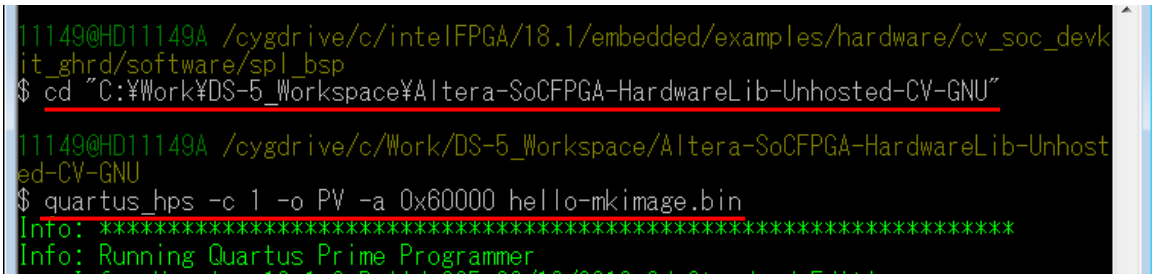
```
quartus_hps <options> <file.bin>
```

Embedded Command Shell から 次のコマンドを実行し、Preloader とベアメタル・アプリケーション・イメージを QSPI フラッシュに書き込みます：

```
$ quartus_hps -c 1 -o PV preloader-mkpimage.bin
$ quartus_hps -c 1 -o PV -a 0x60000 hello-mkimage.bin
```



【図 6-2】 QSPI フラッシュへの Preloader の書き込み例

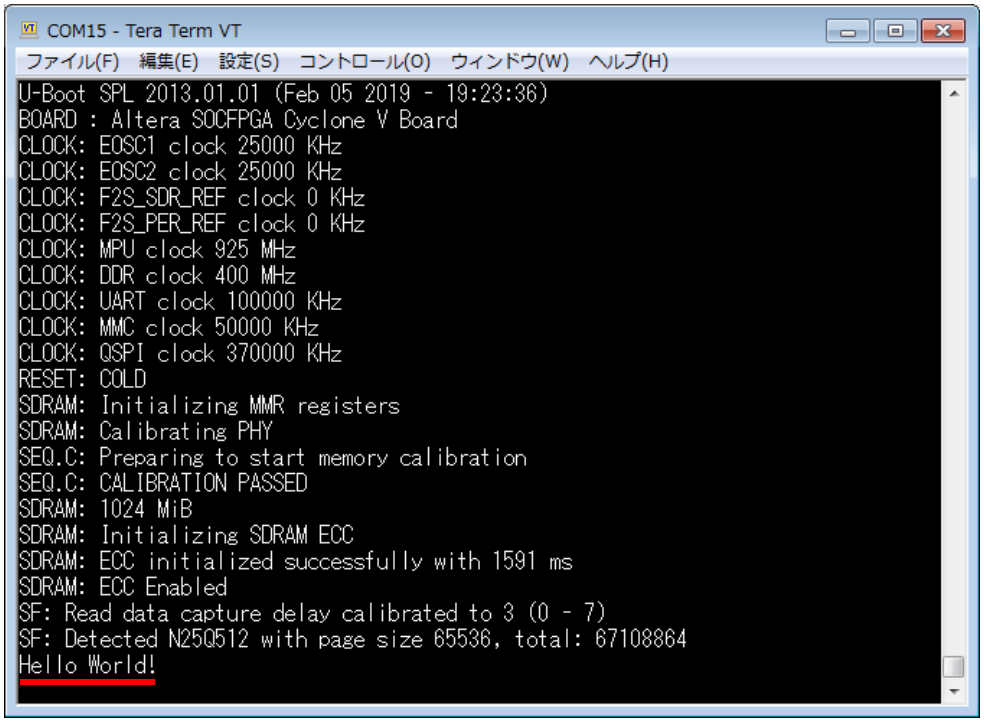


【図 6-3】 QSPI フラッシュへのベアメタル・アプリケーション・イメージの書き込み例

6-4. スタンドアローン実行の動作確認

ボードの電源を入れ直すか、COLD リセットボタン（S7）を押して HPS をリセットします。

ボードがブートし、PC のシリアル端末に Preloader メッセージが表示されてから、“Hello World!” がベアメタル・アプリケーションによって表示されます。



【図 6-4】 QSPI フラッシュからのブート

参考:

SoC EDS、DS-5、Preloader ジェネレータ、および HPS フラッシュプログラマー・ユーティリティーの詳細については、以下のユーザーガイドを参照ください。

『Altera SoC エンベデッド・デザイン・スイート（EDS）ユーザーガイド ug-1137』

改版履歴

Revision	年月	概要
1	2019 年 3 月	初版作成