インテル® SoC FPGA の QSPI ベアメタルアプリ・ブート（インテル® Arria® 10 SoC 編）

Ver.18.1

2019 年 9 月 Rev.2

インテル® SoC FPGA の QSPI ベアメタルアプリ・ブート（インテル® Arria® 10 SoC 編）

目次

1. はじめに

本資料ではインテル® Arria® 10 SoC 開発キットに搭載可能な QSPI（Quad SPI）ブート・フラッシュ・ドーターカードから、ベアメタルサンプル・アプリケーション ALT-HWLib-HelloWorld-Unhosted-A10-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 from Arria 10 SoC!!!”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。また、このサンプルに含まれるファイル io.c は、printf() の出力を JTAG ではなく UART にリダイレクトするスタンドアローン・アプリケーションにも役立ちます。

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 Arm® Development Studio 5 Intel® SoC FPGA Edition（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 2nd ステージ・ブートローダー（U-Boot）の生成方法
- 2nd ステージ・ブートローダーとは？
- QSPI フラッシュブート用 2nd ステージ・ブートローダーの生成手順

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
- RBF ファイルを QSPI フラッシュに書き込む方法
- 2nd ステージ・ブートローダー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、2nd ステージ・ブートローダーとして主に U-Boot を使用した例を説明しています。
non-GPL ライセンスのブートローダー・ソースとして UEFI（Unified Extensible Firmware Interface）ブートローダーを使用することもできます。

UEFI ブートローダーについては、『Intel® Arria® 10 SoC UEFI BootLoader User Guide』（英語版）を参照ください。

Note:

本資料の説明においてハードウェア・デザインについては、既存の Arria® 10 SoC 開発キット向け QSPI ブート用デザインを使用しています。

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	Linux が動作しているホスト PC（Windows PC 上に仮想マシン（VM）環境を構築して Linux を使用することでも可能です） 本資料では、Windows® 7 Professional 上に、Oracle® VM VirtualBox（以下、VirtualBox）と CentOS 6.9（以下、CentOS 6）の組み合わせによる仮想マシン環境を構築して動作の確認を行っております。 ⚠ **注記**: U-Boot のコンパイルは Linux ホストマシンでのみサポートされています。 Windows ではサポートされていません。 仮想マシン環境の構築方法については以下のサイトをご参照ください。 VirtualBox と CentOS 6 による仮想マシン環境の構築
2	インテル® Quartus® Prime 開発ソフトウェア・スタンダード・エディション（またはプロ・エディション）（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（Linux 版） ⚠ **注記**: この資料で説明しているデザインファイル a10_soc_devkit_ghrd_qspi.tgz を実際にコンパイルする場合は、Quartus® Prime プロ・エディションが必要になります。 ⚠ **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 (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（Linux 版） ⚠ **注記**: インテル® FPGA ダウンロード。ケーブル (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	Arria® 10 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。 QSPI ブート・フラッシュ・ドーターカードを取り付けて使用します。 ■ Arria 10 SoC 開発キット
5	Arria® 10 SoC 開発キット向け QSPIブート用コンテンツ	この資料で説明している動作確認を実際に行う場合は、本資料と併せて以下のハードウェア・デザインファイルをダウンロードしてください。 **A10_SoC_DevKit_GHRD_QSPI.tgz** 本資料の説明では、ダウンロードした上記ファイルを **/home/Student/Temp** に格納したものとして説明しています。 ⓘ **Note**: 上記の **A10_SoC_DevKit_GHRD_QSPI.tgz** ファイルは、以下のページのコンテンツを参考に作成しています。 GSRD tagging information ■ Arria 10 QSPI boot hardware（v17.1: a10_soc_devkit_ghrd_qspi.tar.gz） ■ Arria 10 QSPI boot precompiled binaries（v17.1: linux-socfpga-qspi-17.1-a10.tar.gz）

6	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 from Arria 10 SoC!!!” 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アプリケーション・ファイルを取得してください。 **ALT-HWLib-HelloWorld-Unhosted-A10-GNU.tgz** 本資料の説明では、ダウンロードした上記ファイルを **/home/Student/Temp** に格納したものとして説明しています。
7	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（2nd ステージ・ブートローダー・ジェネレーター）、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。

- 『Arria 10 SoC Boot User Guide』（英語版）
- 『Arria 10 SoC ブート・ユーザーガイド』（日本語版）
- 『Intel® Arria® 10 SoC UEFI BootLoader User Guide』（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。

- 『A10 GSRD 16.1 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・ライブラリー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

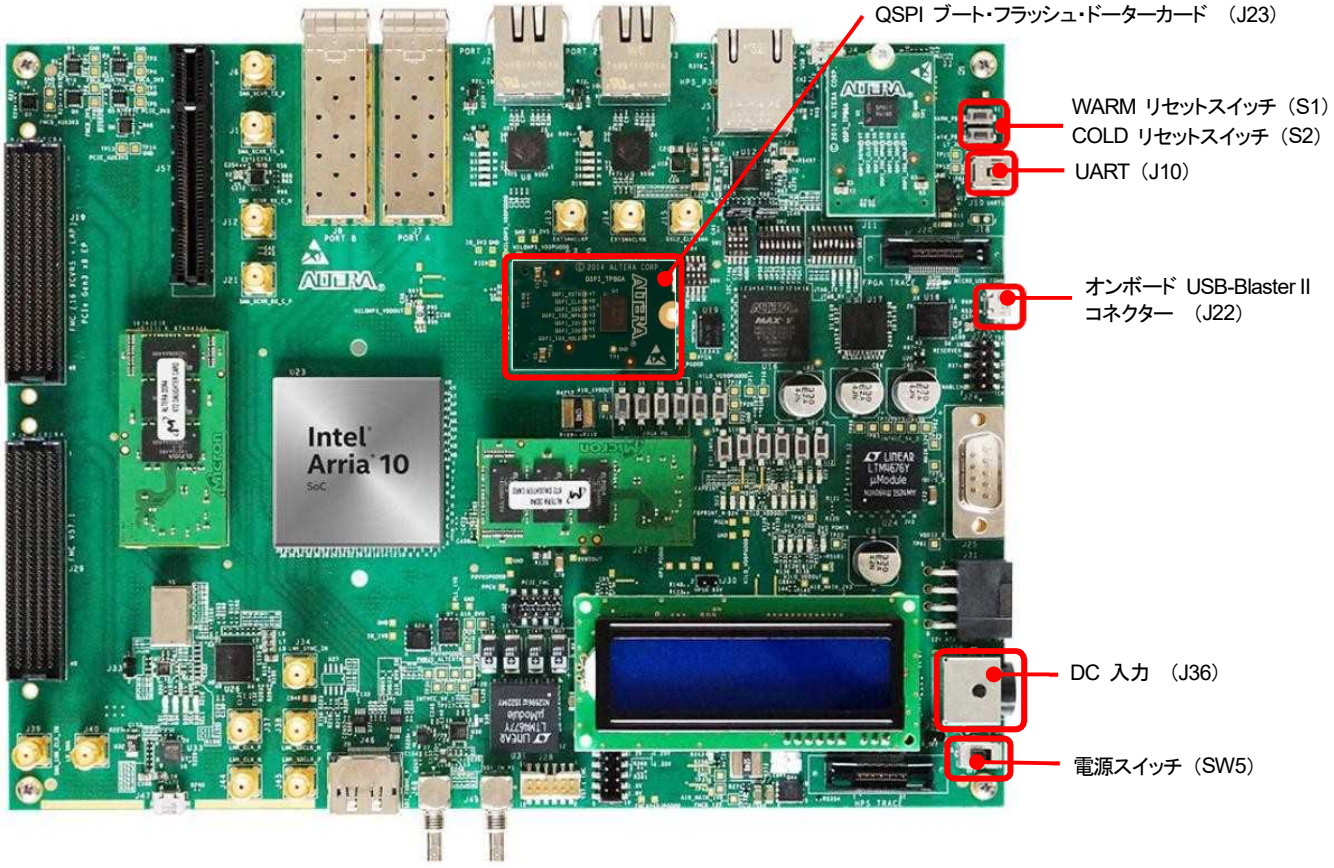
本資料では ターゲットボードとしてインテル® Arria® 10 SoC 開発キットを例として説明しています。

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ハードウェア・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

インテル® Arria® 10 SoC 開発キットのレイアウト図を以下に示します。



【図 2-1】 インテル® Arria® 10 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。

- ドーターカード・コネクタ（J23）に QSPI ブート・フラッシュ・ドーターカードを取り付けます。
- Micro USB ケーブルでホスト PC とオンボード USB-Blaster II コネクター（J22）を接続します。
- Mini USB ケーブルでホスト PC と UART コネクター（J10）を接続します。
- 電源（AC アダプター）を DC 入力（J36）に接続します。

2-1-3. BSEL（BOOTSEL）ピンの設定

インテル® Arria® 10 SoC 開発キットに QSPI ブート・フラッシュ・ドーターカードを搭載することで、BSEL ピンが QSPI ブートの設定となります。 BSEL に関するジャンパなどの設定は必要ありません。

> **参考:**
> インテル® Arria® 10 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 10 SoC Development Kit User Guide』（英語版）
> - 『Arria 10 SoC 開発キット・ユーザーガイド』（日本語版）

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 2nd ステージ・ブートローダー（U-Boot）の生成方法」で説明する 2nd ステージ・ブートローダー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。

2-2-1. ハードウェア・デザインファイルの入手先

この資料で説明している動作確認を実際に行う場合は、インテル® Arria® 10 SoC 開発キット向け QSPI ブート用ハードウェア・デザインファイル **A10_SoC_DevKit_GHRD_QSPI.tgz** をダウンロードして使用します。

本資料をダウンロードしたページと同じページからダウンロードしてください。

本資料の説明では、ダウンロードした上記ファイルを **/home/Student/Temp** に格納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。

ハンドオフファイルには、（XML ファイルとして）FPGA ハードウェア・デザイン情報が含まれており、適切な FPGA ハードウェアの初期化とランタイムアクセスに必要なブートローダー・デバイスツリーを生成するために使用されます。

正しく生成されていれば、**hps_isw_handoff** 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. 2nd ステージ・ブートローダーの生成手順」に利用します。

2nd ステージ・ブートローダー生成のために使用する **bsp-editor**（2nd ステージ・ブートローダー・ジェネレーター）ツールで、この hps_isw_handoff 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。

2-3. ベアメタルサンプル・アプリケーション

「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」で説明しているサンプル・アプリケーション **ALT-HWLib-HelloWorld-Unhosted-A10-GNU.tgz** が必要になります。

本資料をダウンロードしたページと同じページからダウンロードしてください。

本資料の説明では、ダウンロードした上記ファイルを **/home/Student/Temp** に格納したものとして説明しています。

3. SoC FPGA のブートフロー

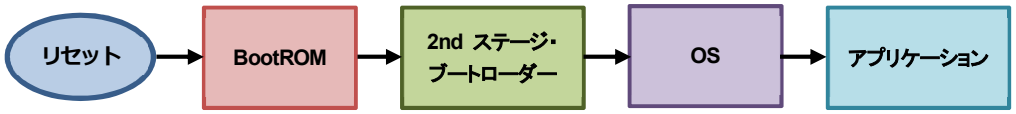
まず、はじめに SoC FPGA のブートフローについて説明します。

> **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。
> - 『Arria 10 SoC Boot User Guide』（英語版）
> - 『Arria 10 SoC ブート・ユーザーガイド』（日本語版）
> - 『Intel® Arria® 10 SoC UEFI BootLoader User Guide』（英語版）

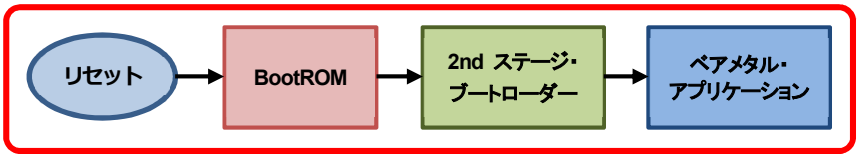
以下の図の通り、SoC FPGA のブートフローには複数のステージが存在します。

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 2nd ステージ・ブートローダー（U-Boot / UEFI ブートローダー）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。

本資料でもこのベアメタル・アプリケーション・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】 ベアメタル・アプリケーション・ブートフロー

- **BootROM**

  インテル® 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は不可）。

  Boot ROM コードはブートソースを決定し、リセット後にハード・プロセッサー・システム (HPS) を初期化し、そして 2nd ステージ・ブートローダーにジャンプします。

- **2nd ステージ・ブートローダー**

  ハンドオフファイルの情報を元に、初期化など動作するために必要な処理を実行します。

  一般的なブートフローの 2nd ステージ・ブートローダーの例は U-Boot です。

  また本書では詳述していませんが、non-GPL ライセンスのブートローダー・ソースとして UEFI（Unified Extensible Firmware Interface）ブートローダーを使用することもできます。

  2nd ステージ・ブートローダーは、OS、ベアメタル・アプリケーションなどをロードすることができます。

- **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んでいます。インテル® SoC FPGA のハードウェア・ライブラリー（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。

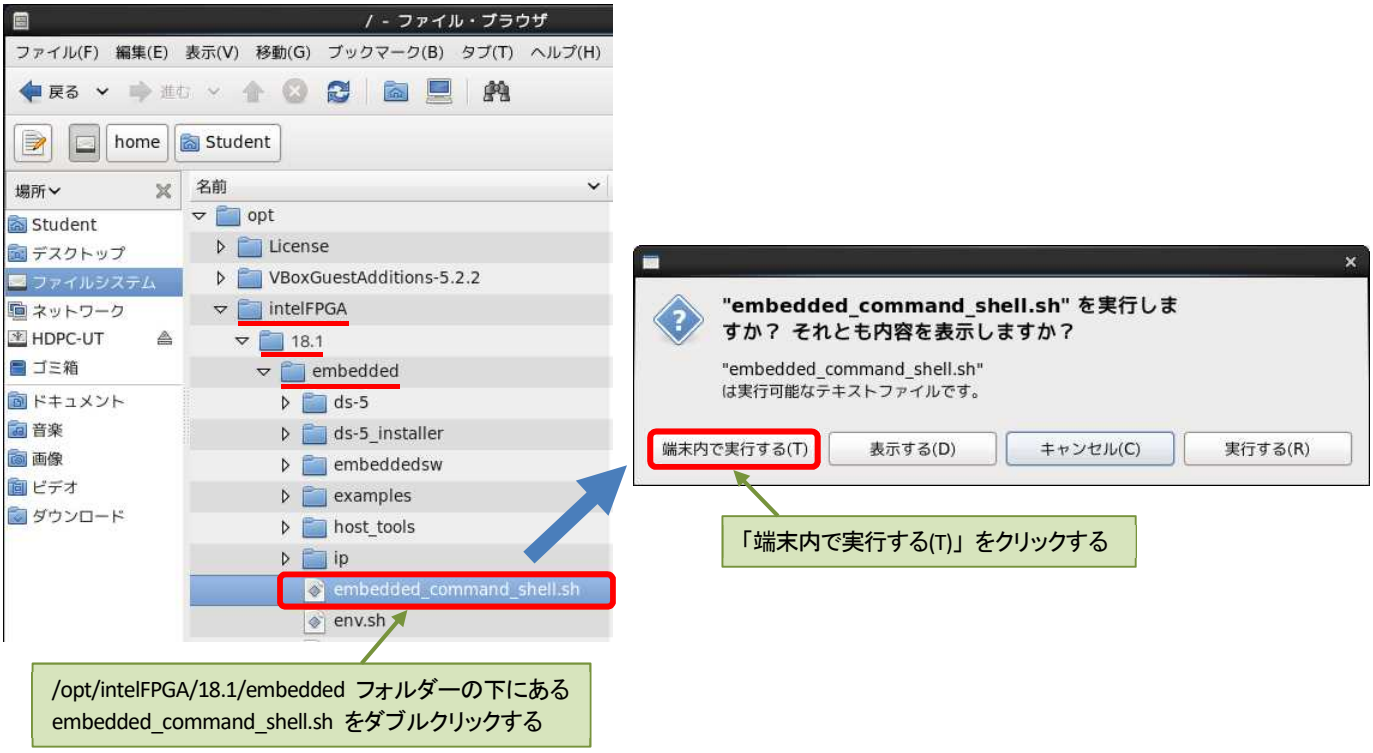
4-1. DS-5 の開始

SoC EDS に含まれている DS-5 Intel® SoC FPGA Edition 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

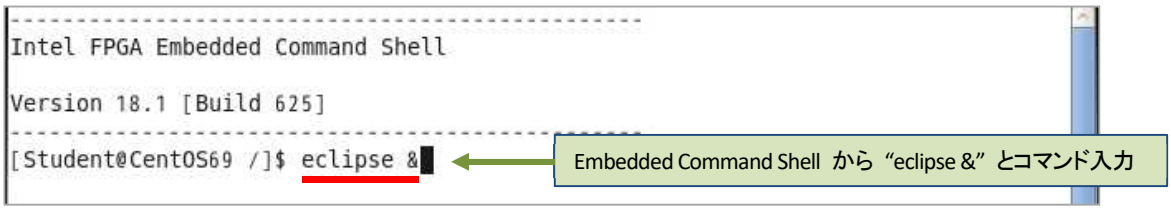
SoC EDS のインストール・フォルダー（embedded フォルダー）下に格納されている起動用スクリプト **embedded_command_shell.sh** を実行し、Embedded Command Shell を起動します。



【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

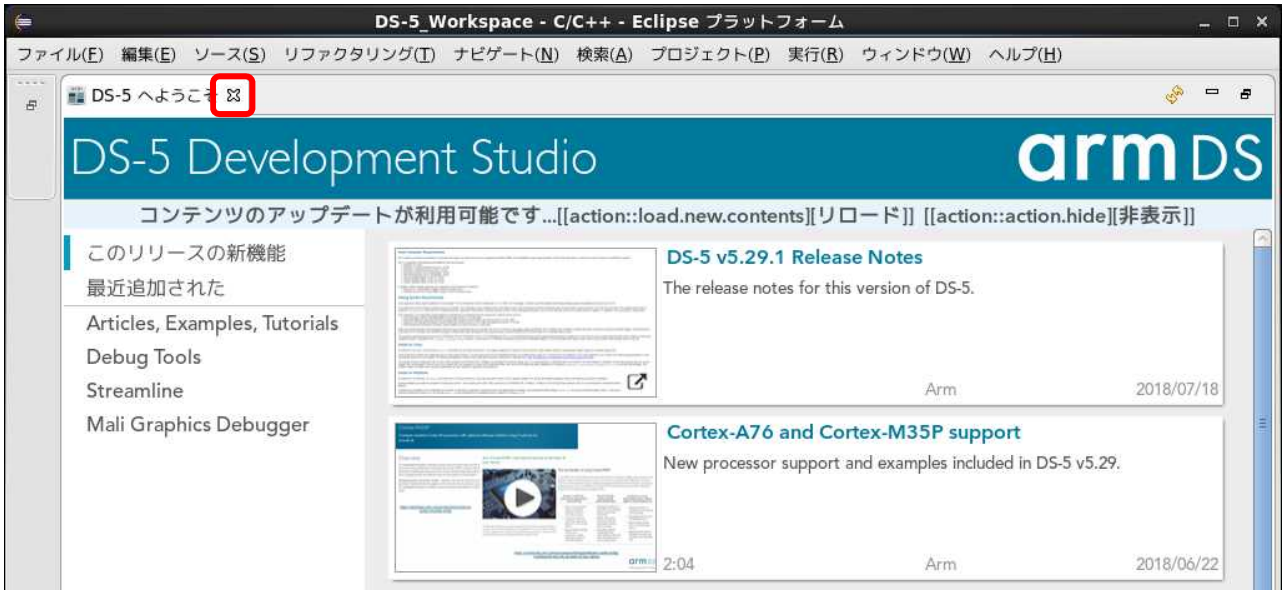
(2) ワークスペース・フォルダーの入力を求められます。ソフトウェア・プロジェクトのために固有のワークスペースを選択または作成します。

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 /home/Student/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

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 **ALT-HWLib-HelloWorld-Unhosted-A10-GNU** を DS-5 にインポートします。

このベアメタルサンプル・プロジェクトの特徴は以下の通りです。

- “Hello from Arria 10 SoC!!!” 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。
- io.c ファイルにより、printf() の出力を UART にリダイレクトしており、スタンドアローン・アプリケーションにも役立ちます。
- startup.s ファイルにより、アプリケーションの先頭で割り込みを禁止する処理を追加しています。これにより、U-Boot において一部のペリフェラルに対して割り込みを有効化している場合に、アプリケーションに期待しない割り込みが入り例外を検出することを防止しています。
- U-Boot では Executable and Linkable Format（ELF）のロードはオプション機能のため、Makefile にて .axf（ELF）ファイルから .bin（プレーンバイナリー）ファイルに変換し、hello.bin を生成しています。

Note:

U-Boot にて割り込みが有効化されてしまうため、ベアメタルアプリの起動時に割り込みのディセーブル処理が必要となります。
以下のリンクは Cyclone® V SoC 向けとなっていますが概念としては同様ですので参考としてご覧ください。

『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』
～ ベアメタル・アプリケーションの SD カードからのスタンドアローン実行例 ～

(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

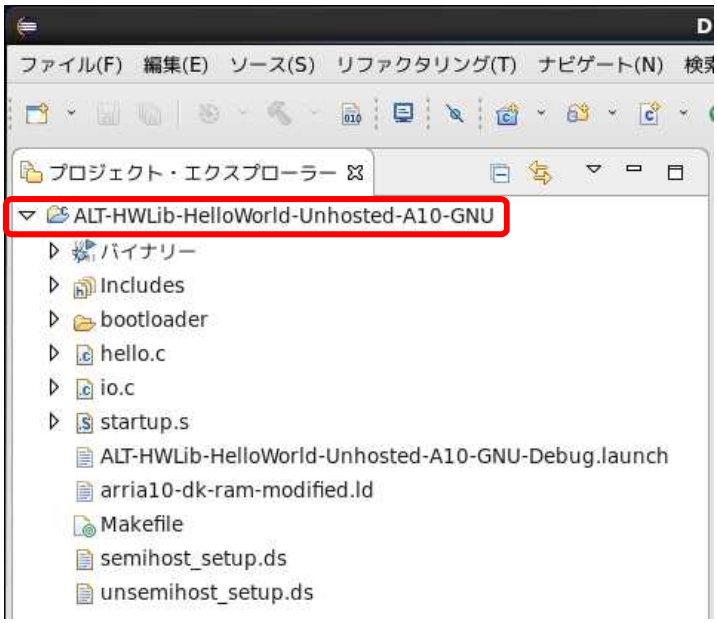
(3)「**アーカイブ・ファイルの選択(A):**」オプションを選択し、［**参照(R)**］ボタンより **ALT-HWLib-HelloWorld-Unhosted-A10-GNU.tgz** 選択後、［**終了(F)**］ボタンを押します。

Note:

本資料の説明では、ALT-HWLib-HelloWorld-Unhosted-A10-GNU.tar.gz を /home/Student/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 **ALT-HWLib-HelloWorld-Unhosted-A10-GNU** が追加され、AlteALT-HWLib-HelloWorld-Unhosted-A10-GNU 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

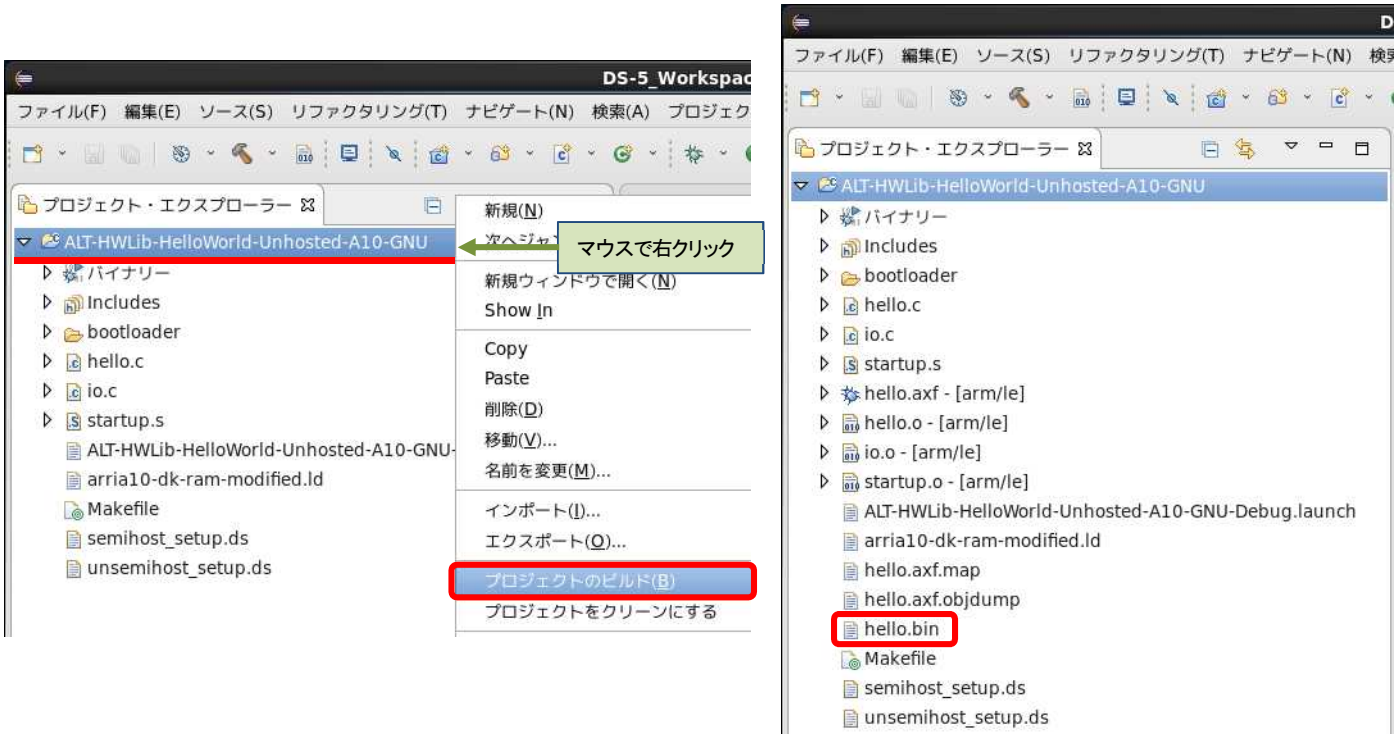
4-3. ベアメタルサンプル・アプリケーションのビルド

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、ALT-HWLib-HelloWorld-Unhosted-A10-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。

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 ファイルが生成されます。



【図 4-8】プロジェクトのビルド

hello.bin は、2nd ステージ・ブートローダー（U-Boot）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 2nd ステージ・ブートローダー（U-Boot）の生成方法」で説明する 2nd ステージ・ブートローダーを QSPI フラッシュに書き込みます。

5. QSPI フラッシュブート用 2nd ステージ・ブートローダー（U-Boot）の生成方法

この章では、インテル® Arria® 10 SoC において QSPI フラッシュからベアメタル・アプリケーションをブートするために必要な 2nd ステージ・ブートローダーの生成手順について説明します。

5-1. 2nd ステージ・ブートローダーとは？

2nd ステージ・ブートローダーはカスタマイズ可能で、通常は HPS の外部の不揮発性フラッシュベース・メモリーまたは FPGA 内のオンチップ RAM に格納されます。

2nd ステージ・ブートローダーは、OS、ベアメタル・アプリケーション、あるいは 3rd ステージ・ブートローダーをロードすることができます。

2nd ステージ・ブートローダーの例としては、 U-Boot および UEFI（Unified Extensible Firmware Interface）ブートローダーがあります。

① インテルが提供する 2nd ステージ・ブートローダー役割は次のとおりです。

- 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 non-GPL ライセンスのブートローダー・ソースとして UEFI ブートローダーを使用することもできます。

UEFI ブートフローは完全に HPS のオンチップメモリーで実行され、ベアメタル・アプリケーションと RTOS を起動するためのデフォルトの選択です。

UEFI は、Cyclone® V および Arria® V デバイスにおける MPL ブートローダーに代わるものです。

Note:

本資料では、2nd ステージ・ブートローダーとして主に U-Boot を使用した例を説明しています。
UEFI ブートローダーについては、『Intel® Arria® 10 SoC UEFI BootLoader User Guide』（英語版）を参照ください。

③ 2nd ステージ・ブートローダーは 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。

④ ユーザーのインテル® 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 2nd ステージ・ブートローダーを必ず生成してください。

5-2. 2nd ステージ・ブートローダーの生成手順

以降に 2nd ステージ・ブートローダーの生成手順を説明します。

SoC EDS には、FPGA デザイン用のブートローダーを生成することを可能にする 2nd ステージ・ブートローダー・サポートパッケージ（BSP）ジェネレーター・ツールが含まれています。

このツールを使用して、2nd ステージ・ブートローダーの設定と生成を行います。

5-2-1. Embedded Command Shell の起動

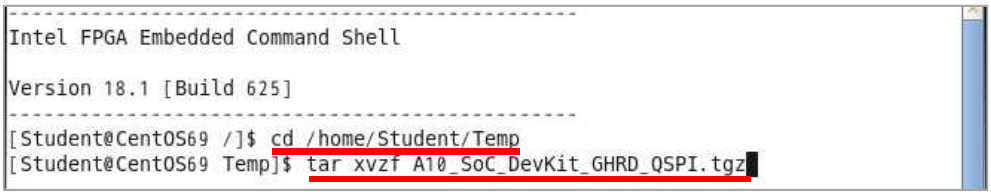
「4-1-1. Embedded Command Shell の起動」と同じ手順で起動します。
インテル® SoC FPGA エンベデッド開発スイート のインストール・フォルダー（embedded フォルダー）下に格納されている起動用スクリプト **embedded_command_shell.sh** を実行し、Embedded Command Shell を起動します。

5-2-2. ハードウェア・デザインファイルの解凍

本資料の説明では、ダウンロードしたハードウェア・デザインファイル **A10_SoC_DevKit_GHRD_QSPI.tgz** を **/home/Student/Temp** に格納したものとして説明しています。

Embedded Command Shell から次のコマンドを入力して A10_SoC_DevKit_GHRD_QSPI.tgz を解凍します。

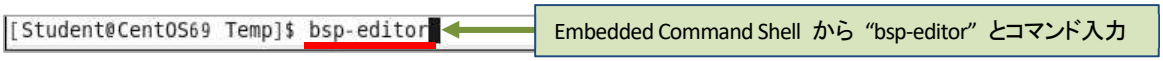
```
$ cd /home/Student/Temp
$ tar xvzf A10_SoC_DevKit_GHRD_QSPI.tgz
```



【図 5-1】 ハードウェア・デザインファイルの解凍

5-2-3. bsp-editor（2nd ステージ・ブートローダー・ジェネレーター）の起動

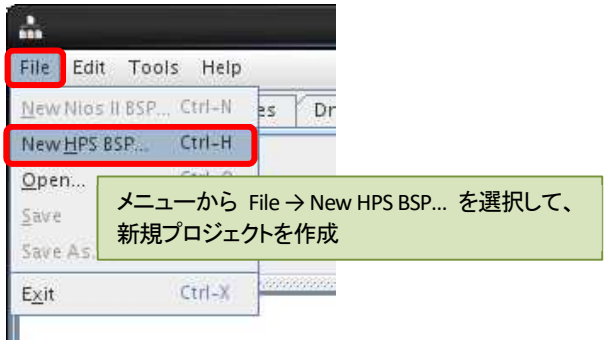
下図のように Embedded Command Shell のウィンドウが開いたら **bsp-editor** とコマンド入力して、bsp-editor（2nd ステージ・ブートローダー・ジェネレーター）の GUI を起動します。



【図 5-2】 bsp-editor（2nd ステージ・ブートローダー・ジェネレーター）の起動

5-2-4. 新規 bsp プロジェクトの作成

図のように bsp-editor 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。



【図 5-3】 新規 bsp プロジェクトの作成

5-2-5. 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 Prime プロジェクト>/hps_isw_handoff を指定します。

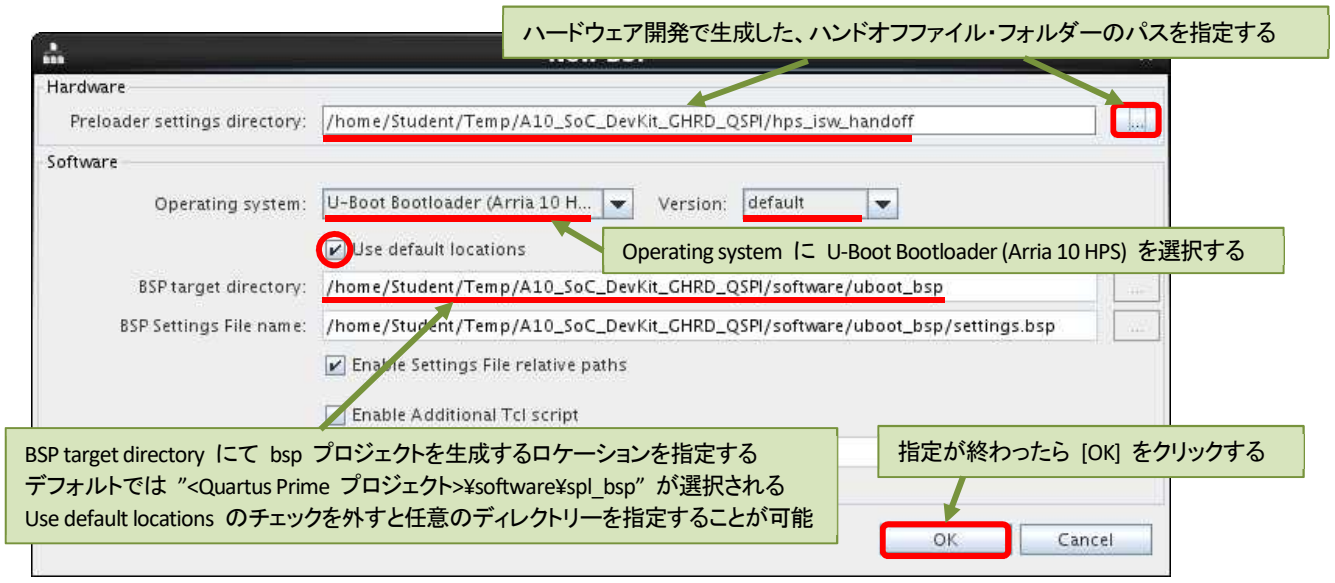
図のように Preloader settings directory: の並びにある [...] を押してフォルダーを指定します。

本資料の説明では、以下のパスを指定します。

/home/Student/Temp/A10_SoC_DevKit_GHRD_QSPI/hps_isw_handoff

（2）Operating systems: に **U-Boot Bootloader (Arria 10 HPS)** を選択します。

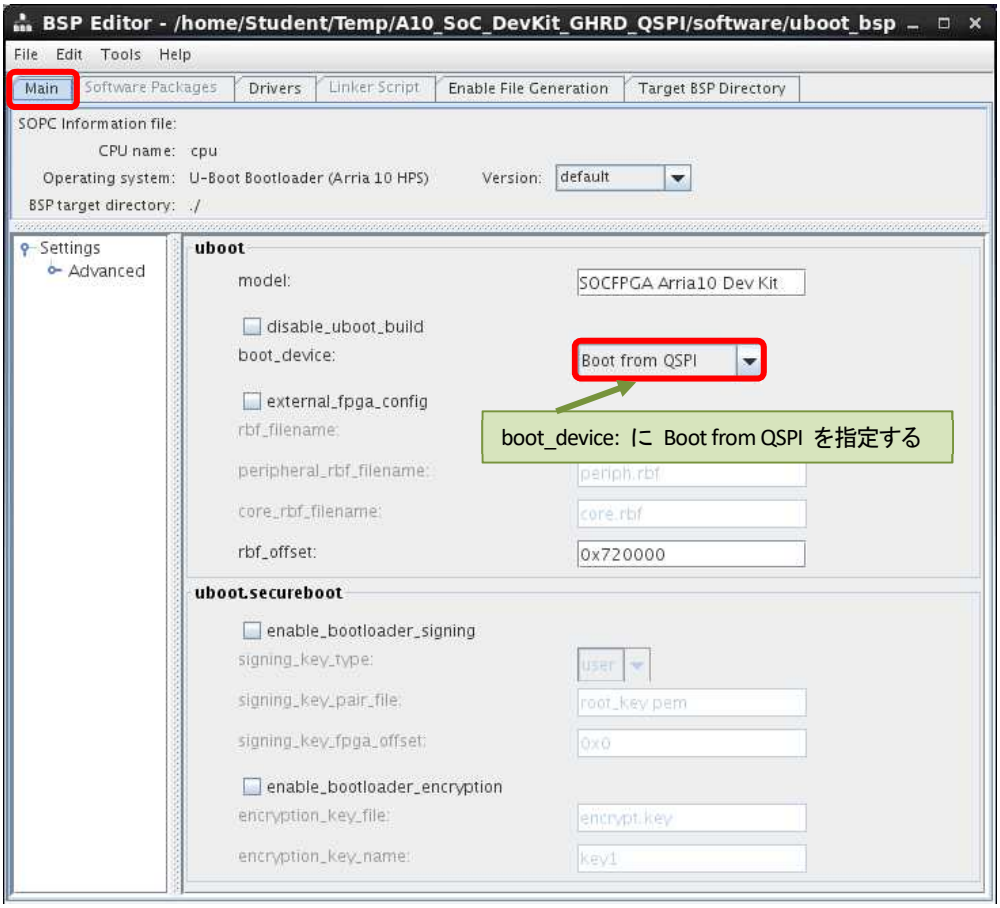
（3）全ての指定が終わったら［**OK**］をクリックします。



【図 5-4】 ハンドオフファイルの指定

5-2-6. 2nd ステージ・ブートローダーのオプションの設定

BSP Editor ウィンドウの Main メニュータブで、boot_device: に **Boot from QSPI** を指定します。



【図 5-5】 2nd ステージ・ブートローダーのオプション設定

5-2-7. bsp プロジェクトの生成（Generate）

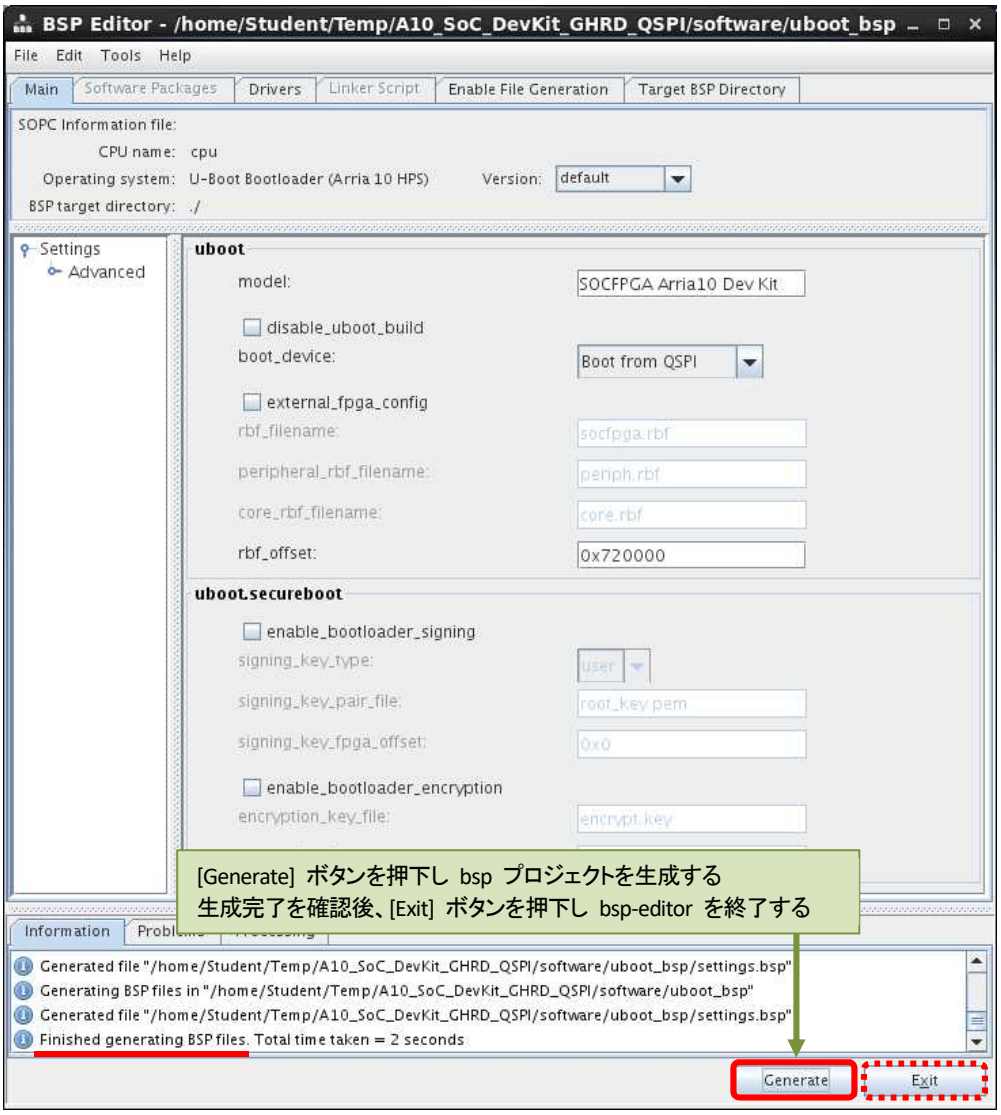
右下の［**Generate**］ボタンを押下し bsp プロジェクトを生成します。

生成する bsp プロジェクトには *.c 、 *.h 、 Makefile を含む 2nd ステージ・ブートローダーを生成（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、「5-2-5. 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。 本書の例では、以下の場所に生成されます。

/home/Student/Temp/A10_SoC_DevKit_GHRD_QSPI/software/uboot_bsp

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



【図 5-6】 bsp プロジェクトの生成

5-2-8. 2nd ステージ・ブートローダーのビルド

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

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/uboot_bsp ↵
```

本資料の説明では、以下のディレクトリーに移動しています。

/home/Student/Temp/A10_SoC_DevKit_GHRD_QSPI/software/uboot_bsp

```
[Student@CentOS69 Temp]$ cd /home/Student/Temp/A10_SoC_DevKit_GHRD_QSPI/software
/uboot_bsp
[Student@CentOS69 uboot_bsp]$
[Student@CentOS69 uboot_bsp]$ ls
Makefile  config.mk  devicetree.dts  settings.bsp  uboot.ds
[Student@CentOS69 uboot_bsp]$
```

bsp-editor で作成した bsp プロジェクトのディレクトリーに移動する

【図 5-7】 bsp プロジェクトのディレクトリーに移動

(2) **make all** ↵ コマンドを実行し 2nd ステージ・ブートローダーを生成します。

ls ↵ コマンドにて **uboot_w_dtb-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 2nd ステージ・ブートローダー用のヘッダ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。

```
[Student@CentOS69 uboot_bsp]$
[Student@CentOS69 uboot_bsp]$ make all
```

"make all" コマンドを実行し 2nd ステージ・ブートローダーを生成する

```
Student@CentOS69:~/Temp/A10_SoC_DevKit_GHRD_QSPI/software/uboot_b
ファイル(F)  編集(E)  表示(V)  検索(S)  端末(T)  ヘルプ(H)
  OBJCOPY examples/standalone/hello_world.bin
  LDS     u-boot.lds
  LD      u-boot
  OBJCOPY u-boot.bin
  MKIMAGE u-boot.img
  OBJCOPY u-boot.srec
  DTC     arch/arm/dts/socfpga_arria10.dtb
  DTC     arch/arm/dts/socfpga_arria10_qspi.dtb
  DTC     arch/arm/dts/socfpga_arria10_nand.dtb
  DTC     arch/arm/dts/socfpga_arria10_qspi_reva.dtb
  SHIPPED dts/dt.dtb
  COPY    u-boot.dtb
  CAT     u-boot-dtb.bin
make[1]: Leaving directory '/home/Student/Temp/A10_SoC_DevKit_GHRD_QSPI/software
/uboot_bsp/uboot-socfpga'
cat uboot-socfpga/u-boot.bin devicetree.dtb > u-boot_w_dtb.bin
mkpimage --header-version 1 -o uboot_w_dtb-mkpimage.bin u-boot_w_dtb.bin u-boot_
w_dtb.bin u-boot_w_dtb.bin u-boot_w_dtb.bin
[Student@CentOS69 uboot_bsp]$
[Student@CentOS69 uboot_bsp]$ ls
Makefile        devicetree.dts    uboot-socfpga
config.mk       settings.bsp      uboot.ds
devicetree.dtb  u-boot_w_dtb.bin  uboot_w_dtb-mkpimage.bin
[Student@CentOS69 uboot_bsp]$
```

【図 5-8】 "make all" コマンドを実行

(3) 現時点で生成された **uboot_w_dtb-mkpimage.bin** を QSPI フラッシュへ書き込んでも、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させることはできません。

U-Boot でのロードおよびブートに関してはコマンドベースで行われます。

これを自動化した環境変数を用意しておくことが一般的になっており、インテル® Arria 10 SoC では、生成された uboot-socfpga ディレクトリーの以下のファイルに定義されています。

/A10_SoC_DevKit_GHRD_QSPI/software/uboot_bsp/uboot-socfpga/include/configs/**socfpga_arria10.h**

例えば QSPI が選択されている場合のデフォルトコマンドは以下の 4 つのコマンド（環境変数）で構成されます。

- run qspirbfcore_rbf_prog
- run qspiload
- run set_initswstate
- run qspiboot

環境変数の実体としては以下のように定義されています。

```
"qspirbfcore_rbf_prog=" ¥
    "fpga loadfs 0 qspi 0:0 ${qspirbfcoreimage} core¥0" ¥
"core_rbf_prog=fpga loadfs 0 mmc 0:1 ${rbfcoreimage} core¥0" ¥

"qspiload=sf probe ${qspiloadcs};" ¥
    "sf read ${loadaddr} ${qspibootimageaddr} ${bootimagesize};" ¥
    "sf read ${fdtaddr} ${qspifdtaddr} ${fdtimagesize};¥0" ¥

"qspiboot=setenv bootargs " CONFIG_BOOTARGS ¥
    " root=${qspiroot} rw rootfstype=${qspirootfstype};" ¥
    "fpgabr 1;" ¥
    "bootz ${loadaddr} - ${fdtaddr}¥0" ¥
```

【図 5-9】 QSPI 環境変数の実体

上記の通り、コマンドにより記述されており、デフォルトでは Linux Kernel のロードとブートを前提としています。

(4) この QSPI 環境変数の実体部分を自作したベアメタル・アプリケーションの格納位置や、DDR 上の展開位置などに合わせて変更し、bootz の代わりに go コマンドで直接エントリーポイントにジャンプするように変更します。

Embedded Command Shell から 以下のようにコマンド入力して、**socfpga_arria10.h** ファイルを開きます。

```
$ gedit ./uboot-socfpga/include/configs/socfpga_arria10.h
```

本資料の説明では、以下のように 2 つの行を変更して、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させるようにします。

```
"qspirbfcore_rbf_prog=" ¥
   "fpga loadfs 0 qspi 0:0 ${qspirbfcoreimage} core¥0" ¥
"core_rbf_prog=fpga loadfs 0 mmc 0:1 ${rbfcoreimage} core¥0" ¥

"qspiload=sf probe ${qspiloadcs};" ¥
   "sf read ${loadaddr} ${qspibootimageaddr} ${bootimagesize};" ¥ ⇒ "sf read ${qspifdtaddr} ${qspibootimageaddr} ${bootimagesize};" ¥
   "sf read ${fdtaddr} ${qspifdtaddr} ${fdtimagesize};¥0" ¥

"qspiboot=setenv bootargs " CONFIG_BOOTARGS ¥
   " root=${qspiroot} rw rootfstype=${qspirootfstype};" ¥
   "fpgabr 1;" ¥
   "bootz ${loadaddr} - ${fdtaddr}¥0" ¥ ⇒ "go ${qspifdtaddr}¥0" ¥
```

${loadaddr}（0x8000）を ${qspifdtaddr}（0x100000）に変更する

bootz ${loadaddr} - ${fdtaddr} を go ${qspifdtaddr} に変更する

【図 5-10】 ベアメタル・アプリケーションをスタンドアローン実行するための QSPI 環境変数の変更

(5) **socfpga_arria10.h** ファイルの変更が終わったら、セーブしてファイルを閉じます。

(6) Embedded Command Shell から 以下のようにコマンド入力して、上記の変更を反映させた 2nd ステージ・ブートローダーを再生成します。

```
$ make clean
$ make all
$ ls
```

```
[Student@CentOS69 uboot_bsp]$ ls
Makefile        devicetree.dts    uboot-socfpga
config.mk       settings.bsp      uboot.ds
devicetree.dtb  u-boot_w_dtb.bin  uboot_w_dtb-mkpimage.bin
[Student@CentOS69 uboot_bsp]$
```

【図 5-11】 生成された 2nd ステージ・ブートローダー

uboot_w_dtb-mkpimage.bin が生成されていることを確認します。

このファイルを次章の手順に従って QSPI フラッシュへ書き込むことで、ベアメタル・アプリケーションをスタンドアローン実行することが可能となります。

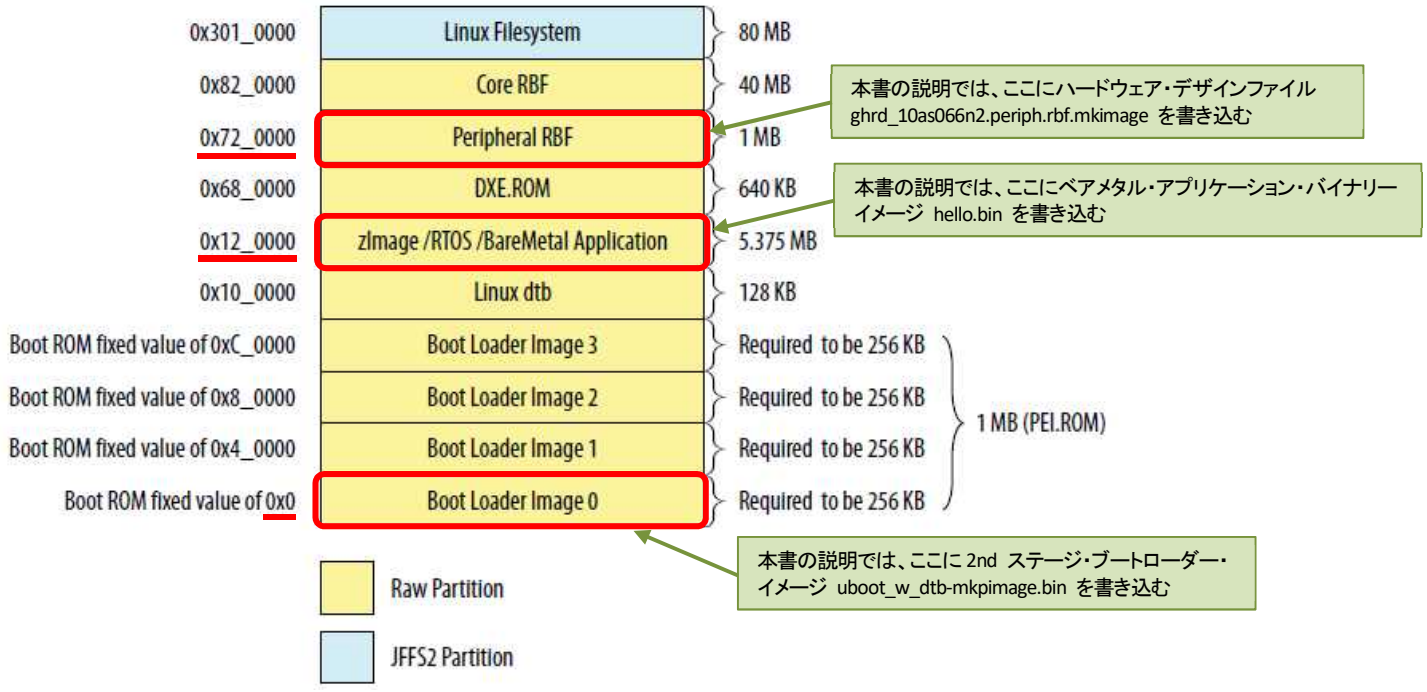
6. ベアメタル・アプリケーションを QSPI フラッシュからスタンドアローン実行する例

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 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。

- 通常は、ブートローダー・イメージ 0、1、2、3 に、 2nd ステージ・ブートローダー（U-Boot）を書き込みます。
  本書の説明では、ブートローダー・イメージ 0（0x0 番地）にのみ、 2nd ステージ・ブートローダー・イメージ **uboot_w_dtb-mkpimage.bin** を書き込みます。
- 次のブートイメージ（zImage、 RTOS バイナリーイメージ、またはベアメタル・アプリケーション・バイナリーイメージのいずれか）を 0x120000 番地に書き込みます。
  本書の説明では、ベアメタル・アプリケーション・バイナリーイメージ **hello.bin** を書き込みます。
- ハードウェア・デザイン（Peripheral RBF）を 0x720000 番地に書き込みます。
  本書の説明では、既存のハードウェア・デザイン **ghrd_10as066n2.periph.rbf.mkimage** を書き込みます。



【図 6-1】 QSPI フラッシュのレイアウト

注記:

QSPI フラッシュへの書き込みには、インテル® FPGA ダウンロード・ケーブル (USB-Blaster II) による接続が必要になります。

RedHat Linux Enterprise 5 以降で USB-Blaster II をはじめて使用する場合は、USB-Blaster II ドライバーのセットアップが必要になります。

セットアップ方法については、「7. 補足: RedHat Linux Enterprise 5 以降での USB-Blaster II のセットアップ」を参照ください。

6-2. QSPI ブート・フラッシュ・ドーターカードの取り付け確認

「2-1-3. BSEL（BOOTSEL）ピンの設定」に従って、インテル® Arria® 10 SoC 開発キットに QSPI ブート・フラッシュ・ドーターカードが搭載されていて、QSPI ブートが可能であることを確認してください。

6-3. ハードウェア・デザインを 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。

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>
```

本資料の説明では、既存のハードウェア・デザイン **ghrd_10as066n2.periph.rbf.mkimage** を QSPI フラッシュに書き込みます。

Embedded Command Shell から次のコマンドを入力します。

```
$ cd /home/Student/Temp/A10_SoC_DevKit_GHRD_QSPI/output_files
$ quartus_hps -c 1 -o PV --boot=18 -a 0x720000 ghrd_10as066n2.periph.rbf.mkimage
```

```
[Student@CentOS69 output_files]$ quartus_hps -c 1 -o PV --boot=18 -a 0x720000 gh
rd_10as066n2.periph.rbf.mkimage
Info: ******************************************************************
Info: Running Quartus Prime Programmer
    Info: Version 18.1.0 Build 625 09/12/2018 SJ Standard Edition
    Info: Copyright (C) 2018  Intel Corporation. All rights reserved.
    Info: Your use of Intel Corporation's design tools, logic functions
    Info: and other software and tools, and its AMPP partner logic
    Info: functions, and any output files from any of the foregoing
    Info: (including device programming or simulation files), and any
    Info: associated documentation or information are expressly subject
    Info: to the terms and conditions of the Intel Program License
    Info: Subscription Agreement, the Intel Quartus Prime License Agreement,
    Info: the Intel FPGA IP License Agreement, or other applicable license
    Info: agreement, including, without limitation, that your use is for
    Info: the sole purpose of programming logic devices manufactured by
    Info: Intel and sold by Intel or its authorized distributors.  Please
    Info: refer to the applicable agreement for further details.
    Info: Processing started: Wed Feb 27 10:37:54 2019
Info: Command: quartus_hps -c 1 -o PV --boot=18 -a 0x720000 ghrd_10as066n2.perip
h.rbf.mkimage
Current hardware is: USB-BlasterII [1-2]
   Successfully change hardware frequency to 16Mhz
Found HPS at device 2
   Double check JTAG chain
HPS Device IDCODE: 0x4BA00477
AHB Port is located at port 0
APB Port is located at port 1
   Double check device identification ...
Warning: Device is Arria 10 SoC
Setup non-secure transaction ...
Warning: Cold Reset ...
Halt CPU by setting Watchpoint
   Successfully halted CPU 0 (Synchronous Watchpoint Event) [0x0308602B]
Boot Info: 1.8V QSPI Flash
Clock Select: 0
Start HPS Quad SPI flash programming ...
Initialize QSPI peripheral and flash controller ...
   Assuming QSPI controller system clock is 50Mhz
   QSPI controller baudrate setting: 32 (15)
Read Silicon ID of Quad SPI flash ...
   Quad SPI Flash silicon ID is 0x1021BB20
      Flash device matched
         Manufacturer: MICRON
         Device: QSPI_1024
         Extended ID: 0x00000044
   Assuming QSPI controller system clock is 50Mhz
   QSPI controller baudrate setting: 2 (0)
Enable Four Byte Addressing ...
Sector Erase Quad SPI flash ...
   Sector Erase Info: Start Addr at 0x00720000 for 6 sector(s)
      Sector Erase Quad SPI flash at 0x00720000
      Sector Erase Quad SPI flash at 0x00730000
      Sector Erase Quad SPI flash at 0x00740000
      Sector Erase Quad SPI flash at 0x00750000
      Sector Erase Quad SPI flash at 0x00760000
      Sector Erase Quad SPI flash at 0x00770000
Program Quad SPI flash ...
Verify Quad SPI flash ...
Info: Quartus Prime Programmer was successful. 0 errors, 0 warnings
    Info: Peak virtual memory: 219 megabytes
    Info: Processing ended: Wed Feb 27 10:40:51 2019
    Info: Elapsed time: 00:02:57
    Info: Total CPU time (on all processors): 00:00:05
[Student@CentOS69 output_files]$
```

【図 6-2】 ハードウェア・デザインの書き込み成功例

6-4. 2nd ステージ・ブートローダー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

Embedded Command Shell から 次のコマンドを実行し、
2nd ステージ・ブートローダー **uboot_w_dtb-mkpimage.bin** を QSPI フラッシュに書き込みます。

```
$ cd /home/Student/Temp/A10_SoC_DevKit_GHRD_QSPI/software/uboot_bsp
$ quartus_hps -c 1 -o PV --boot=18 -a 0 uboot_w_dtb-mkpimage.bin
```

```
[Student@CentOS69 uboot_bsp]$ quartus_hps -c 1 -o PV --boot=18 -a 0 uboot_w_dtb-
mkpimage.bin
Info: ******************************************************************
Info: Running Quartus Prime Programmer
    Info: Version 18.1.0 Build 625 09/12/2018 SJ Standard Edition
    Info: Copyright (C) 2018  Intel Corporation. All rights reserved.
    Info: Your use of Intel Corporation's design tools, logic functions
    Info: and other software and tools, and its AMPP partner logic
    Info: functions, and any output files from any of the foregoing
    Info: (including device programming or simulation files), and any
    Info: associated documentation or information are expressly subject
    Info: to the terms and conditions of the Intel Program License
    Info: Subscription Agreement, the Intel Quartus Prime License Agreement,
    Info: the Intel FPGA IP License Agreement, or other applicable license
    Info: agreement, including, without limitation, that your use is for
    Info: the sole purpose of programming logic devices manufactured by
    Info: Intel and sold by Intel or its authorized distributors.  Please
    Info: refer to the applicable agreement for further details.
    Info: Processing started: Wed Feb 27 10:22:06 2019
Info: Command: quartus_hps -c 1 -o PV --boot=18 -a 0 uboot_w_dtb-mkpimage.bin
Current hardware is: USB-BlasterII [1-2]
   Successfully change hardware frequency to 16Mhz
Found HPS at device 2
   Double check JTAG chain
HPS Device IDCODE: 0x4BA00477
AHB Port is located at port 0
APB Port is located at port 1
   Double check device identification ...
Warning: Device is Arria 10 SoC
Setup non-secure transaction ...
Warning: Cold Reset ...
Halt CPU by setting Watchpoint
   Successfully halted CPU 0 (Synchronous Watchpoint Event) [0x0308602B]
Boot Info: 1.8V QSPI Flash
Clock Select: 0
Start HPS Quad SPI flash programming ...
Initialize QSPI peripheral and flash controller ...
   Assuming QSPI controller system clock is 50Mhz
   QSPI controller baudrate setting: 32 (15)
Read Silicon ID of Quad SPI flash ...
   Quad SPI Flash silicon ID is 0x1021BB20
      Flash device matched
         Manufacturer: MICRON
         Device: QSPI_1024
         Extended ID: 0x00000044
   Assuming QSPI controller system clock is 50Mhz
   QSPI controller baudrate setting: 2 (0)
Enable Four Byte Addressing ...
Sector Erase Quad SPI flash ...
   Sector Erase Info: Start Addr at 0x00000000 for 16 sector(s)
      Sector Erase Quad SPI flash at 0x00000000
      Sector Erase Quad SPI flash at 0x00010000
      Sector Erase Quad SPI flash at 0x00020000
      Sector Erase Quad SPI flash at 0x00030000
      Sector Erase Quad SPI flash at 0x00040000
      Sector Erase Quad SPI flash at 0x00050000
      Sector Erase Quad SPI flash at 0x00060000
      Sector Erase Quad SPI flash at 0x00070000
      Sector Erase Quad SPI flash at 0x00080000
      Sector Erase Quad SPI flash at 0x00090000
      Sector Erase Quad SPI flash at 0x000A0000
      Sector Erase Quad SPI flash at 0x000B0000
      Sector Erase Quad SPI flash at 0x000C0000
      Sector Erase Quad SPI flash at 0x000D0000
      Sector Erase Quad SPI flash at 0x000E0000
      Sector Erase Quad SPI flash at 0x000F0000
Program Quad SPI flash ...
Verify Quad SPI flash ...
Info: Quartus Prime Programmer was successful. 0 errors, 0 warnings
    Info: Peak virtual memory: 217 megabytes
    Info: Processing ended: Wed Feb 27 10:28:40 2019
    Info: Elapsed time: 00:06:34
    Info: Total CPU time (on all processors): 00:00:16
[Student@CentOS69 uboot_bsp]$
```

【図 6-3】 2nd ステージ・ブートローダーの書き込み成功例

同様に、ベアメタル・アプリケーション **hello.bin** を QSPI フラッシュに書き込みます。

```
$ cd /home/Student/DS-5_Workspace/ALT-HWLib-HelloWorld-Unhosted-A10-GNU
$ quartus_hps -c 1 -o PV --boot=18 -a 0x120000 hello.bin
```

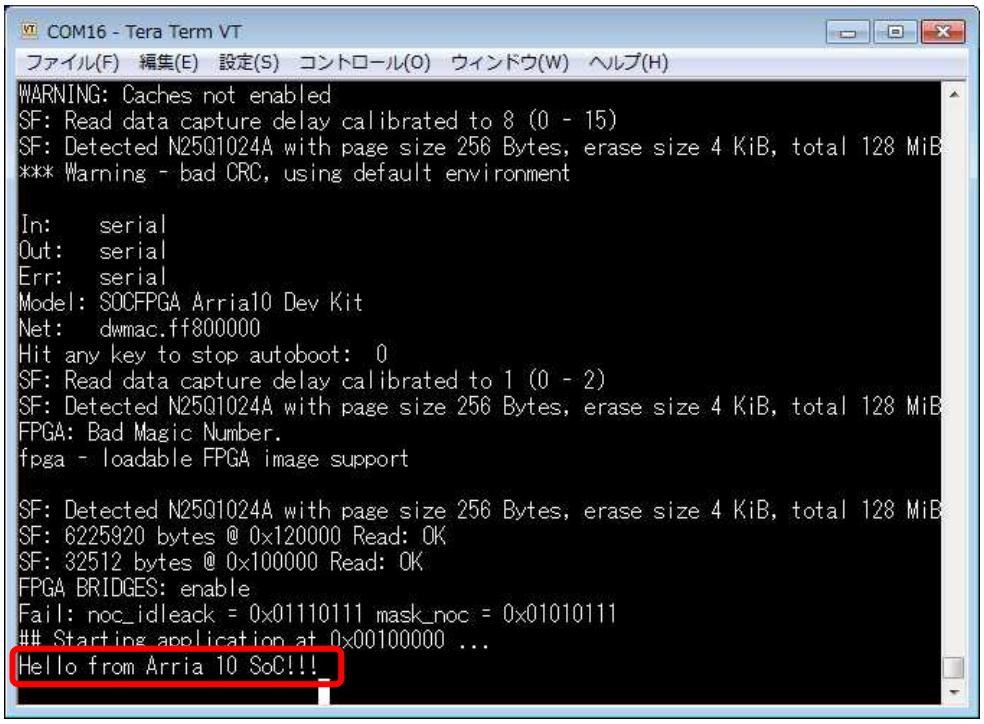
```
[Student@CentOS69 ALT-HWLib-HelloWorld-Unhosted-A10-GNU]$ quartus_hps -c 1 -o PV
 --boot=18 -a 0x120000 hello.bin
Info: ******************************************************************
Info: Running Quartus Prime Programmer
    Info: Version 18.1.0 Build 625 09/12/2018 SJ Standard Edition
    Info: Copyright (C) 2018  Intel Corporation. All rights reserved.
    Info: Your use of Intel Corporation's design tools, logic functions
    Info: and other software and tools, and its AMPP partner logic
    Info: functions, and any output files from any of the foregoing
    Info: (including device programming or simulation files), and any
    Info: associated documentation or information are expressly subject
    Info: to the terms and conditions of the Intel Program License
    Info: Subscription Agreement, the Intel Quartus Prime License Agreement,
    Info: the Intel FPGA IP License Agreement, or other applicable license
    Info: agreement, including, without limitation, that your use is for
    Info: the sole purpose of programming logic devices manufactured by
    Info: Intel and sold by Intel or its authorized distributors.  Please
    Info: refer to the applicable agreement for further details.
    Info: Processing started: Wed Feb 27 10:35:28 2019
Info: Command: quartus_hps -c 1 -o PV --boot=18 -a 0x120000 hello.bin
Current hardware is: USB-BlasterII [1-2]
   Successfully change hardware frequency to 16Mhz
Found HPS at device 2
   Double check JTAG chain
HPS Device IDCODE: 0x4BA00477
AHB Port is located at port 0
APB Port is located at port 1
   Double check device identification ...
Warning: Device is Arria 10 SoC
Setup non-secure transaction ...
Warning: Cold Reset ...
Halt CPU by setting Watchpoint
   Successfully halted CPU 0 (Synchronous Watchpoint Event) [0x0308602B]
Boot Info: 1.8V QSPI Flash
Clock Select: 0
Start HPS Quad SPI flash programming ...
Initialize QSPI peripheral and flash controller ...
   Assuming QSPI controller system clock is 50Mhz
   QSPI controller baudrate setting: 32 (15)
Read Silicon ID of Quad SPI flash ...
   Quad SPI Flash silicon ID is 0x1021BB20
      Flash device matched
         Manufacturer: MICRON
         Device: QSPI_1024
         Extended ID: 0x00000044
   Assuming QSPI controller system clock is 50Mhz
   QSPI controller baudrate setting: 2 (0)
Enable Four Byte Addressing ...
Sector Erase Quad SPI flash ...
   Sector Erase Info: Start Addr at 0x00120000 for 1 sector(s)
      Sector Erase Quad SPI flash at 0x00120000
Program Quad SPI flash ...
Verify Quad SPI flash ...
Info: Quartus Prime Programmer was successful. 0 errors, 0 warnings
    Info: Peak virtual memory: 217 megabytes
    Info: Processing ended: Wed Feb 27 10:35:57 2019
    Info: Elapsed time: 00:00:29
    Info: Total CPU time (on all processors): 00:00:01
[Student@CentOS69 ALT-HWLib-HelloWorld-Unhosted-A10-GNU]$
```

【図 6-4】 ベアメタル・アプリケーションの書き込み成功例

6-5. 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（S2）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 2nd ステージ・ブートローダーメッセージが表示されてから、“Hello from Arria 10 SoC!!!” 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 6-5】 QSPI フラッシュからのブート

参考:

SoC EDS、DS-5、Preloader ジェネレーター、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』

7. 補足： RedHat Linux Enterprise 5 以降での USB-Blaster II のセットアップ

RedHat Linux Enterprise 5 以降で インテル® FPGA ダウンロード・ケーブル（USB-Blaster II）をはじめて使用する場合は、USB-Blaster II ドライバーのセットアップが必要になります。以下の手順に従って設定を行ってください。

(1) Embedded Command Shell から 次のコマンドを実行し、**/etc/udev/rules.d/51-usbblaster.rules** ファイルを作成して以下の行を記述します。

```
$ sudo gedit /etc/udev/rules.d/51-usbblaster.rules
```

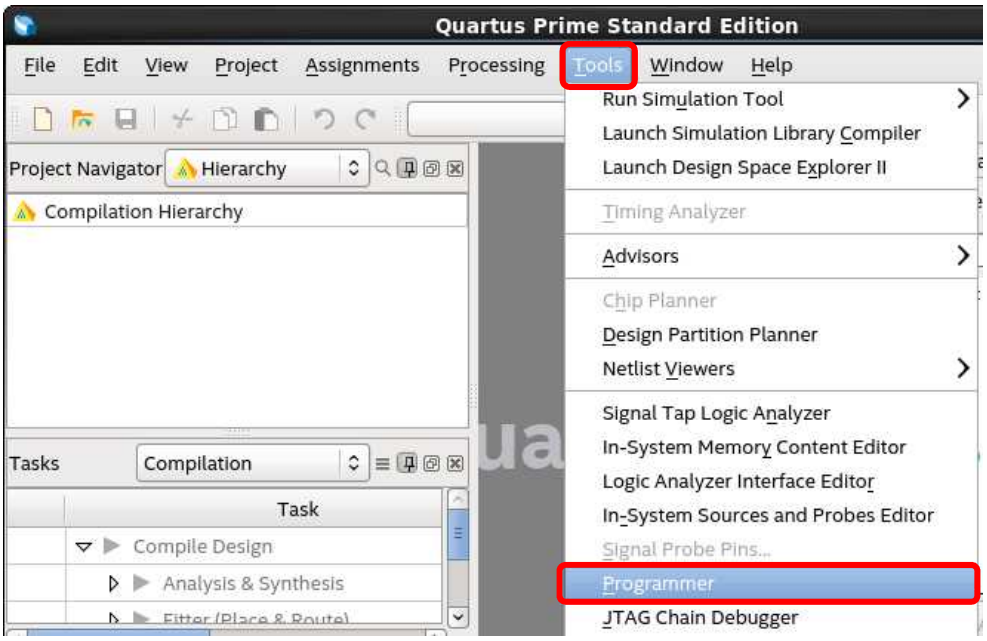
```
# USB-Blaster
BUS=="usb", SYSFS{idVendor}=="09fb", SYSFS{idProduct}=="6001", MODE="0666"
BUS=="usb", SYSFS{idVendor}=="09fb", SYSFS{idProduct}=="6002", MODE="0666"
BUS=="usb", SYSFS{idVendor}=="09fb", SYSFS{idProduct}=="6003", MODE="0666"

# USB-Blaster II
BUS=="usb", SYSFS{idVendor}=="09fb", SYSFS{idProduct}=="6010", MODE="0666"
BUS=="usb", SYSFS{idVendor}=="09fb", SYSFS{idProduct}=="6810", MODE="0666"
```

【図 7-1】 51-usbblaster.rules ファイルに記述する内容

(2) 記述が終わったら、51-usbblaster.rules ファイルをセーブして閉じます。

(3) **Quartus® Prime** を起動し、**Programmer** を起動します。

```
$ cd /opt/intelFPGA/18.1/quartus/bin
$ ./quartus
```

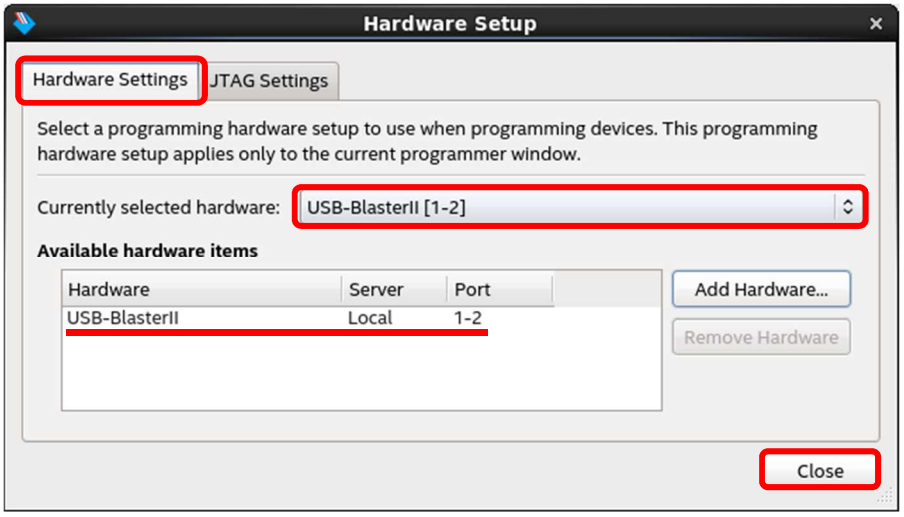


【図 7-2】 Programmer を起動

(4) Programmer が起動したら、[**Hardware Setup…**] をクリックします。

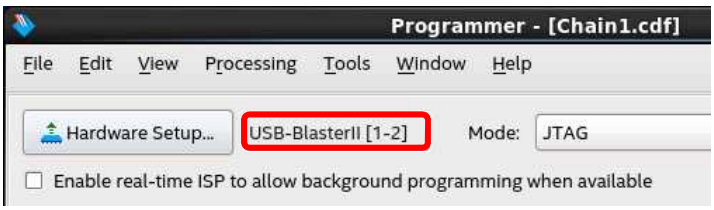
【図 7-3】［Hardware Setup...］をクリック

(5) Hardware Setup ウィンドウが開いたら、「Hardware Settings」タブの Currently selected hardware: の並びのプルダウンから、**USB-Blaster II** を選択して、[**Close**] をクリックします。



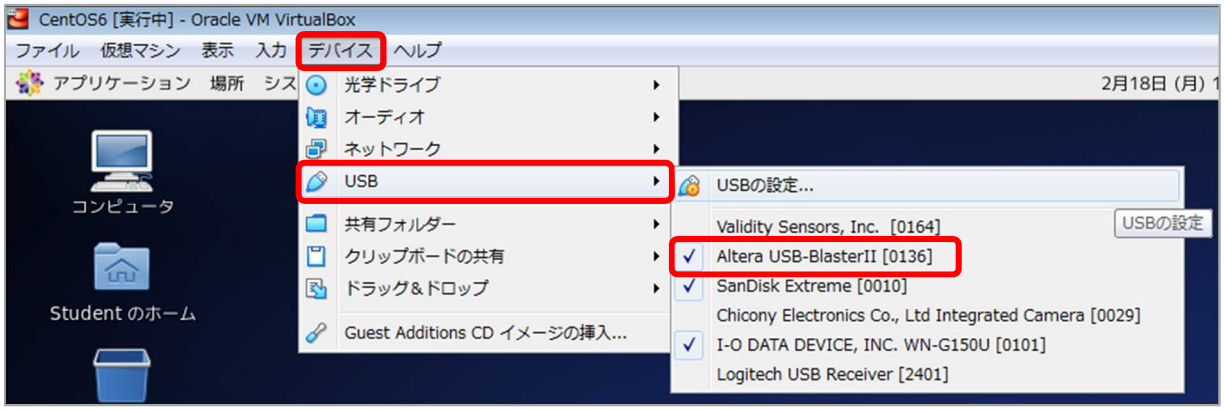
【図 7-4】プルダウンから USB-Blaster II を選択して［Close］をクリック

(6) Programmer ウィンドウに戻るので、**USB-Blaster II** が設定されていることを確認します。正しく設定されていれば Programmer ウィンドウを閉じます。



【図 7-5】 USB-Blaster II が設定されていることを確認

(7) Virtual Box の USB 設定から USB-Blaster II の接続を有効にします。



【図 7-6】 Virtual Box の USB 設定から USB-Blaster II の接続を有効にする

改版履歴

Revision	年月	概要
1	2019 年 3 月	初版
2	2019 年 9 月	『5-2-1. Embedded Command Shell の起動』の説明記述を訂正