

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 (Arria V SoC / Cyclone V SoC 編)

Ver.18



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 (Arria V SoC / Cyclone V SoC 編)

<u>目次</u>

| 1. | はじめに | 4 |
|----|--|-----|
| 2. | 事前準備 | 7 |
| | 2-1. ボードの設定 | . 7 |
| | 2-1-1. ボード・レイアウト | . 7 |
| | 2-1-2. 電源およびケーブルの接続 | . 7 |
| | 2-1-3. SW10 の設定 | . 7 |
| | 2-2. FPGA デザイン・ファイル | . 8 |
| | 2-2-1. FPGA デザイン・ファイルの入手先 | . 8 |
| | 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) | . 8 |
| | 2-2-3. FPGA デザイン・ファイルをターゲット・ボードへダウンロードする方法 | . 9 |
| 3. | SoC FPGA のブート・フロー | 13 |
| 4. | ベアメタル・サンプル・アプリケーションを DS-5 で実行する方法 | 14 |
| | 4-1. SoC EDS に付属のサンプル・アプリケーションの紹介 (参考) | 14 |
| | 4-2. DS-5 の起動 | 15 |
| | 4-2-1. Embedded Command Shell の起動 | 15 |
| | 4-2-2. DS-5 の起動 | 15 |
| | 4-3. ベアメタル・サンプル・アプリケーションのインポート | 17 |
| | 4-4. ベアメタル・サンプル・アプリケーションのビルド | 19 |
| | 4-4-1. Makefile の例(参考) | 19 |
| | 4-4-2. プロジェクトのビルド | 20 |
| | 4-4-3. ベアメタル・サンプル・アプリケーション・プロジェクトのファイル構成構成 | 21 |
| | 4-5. ベアメタル・サンプル・アプリケーションのデバッグ | 22 |
| | 4-5-1. デバッグの実行 | 22 |
| | 4-5-2. デバッグの構成(参考) | 26 |
| | 4-5-3. デバッガ・スクリプト・ファイルでの実行内容 (参考) | 27 |
| 5. | 新規にベアメタル・アプリケーションを作成して DS-5 で実行する方法 | 31 |
| | 5-1. ベアメタル・アプリケーションの新規作成 | 31 |
| | 5-2. ベアメタル・アプリケーションのビルド | 36 |
| | 5-3. ベアメタル・アプリケーションのデバッグ | 37 |
| 6. | FPGA レジスタの確認方法 | 41 |



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 (Arria V SoC / Cyclone V SoC 編)

| 7. カスタム・ボードへの対応方法 | 45 |
|--|------|
| 7-1. Arm プロセッサを含むハードウェアの設計を行う | . 45 |
| 7-2. Preloader(プリローダ)とは? | . 45 |
| 7-3. Preloader の生成手順 | . 46 |
| 7-3-1. Embedded Command Shell の起動 | . 46 |
| 7-3-2. bsp-editor(Preloader Generator)の起動 | . 46 |
| 7-3-3. 新規 bsp プロジェクトの作成 | . 46 |
| 7-3-4. ハンドオフ・ファイルの指定 | . 47 |
| 7-3-5. Preloader のユーザ・オプション(Common)の設定 | . 48 |
| 7-3-6. Preloader のユーザ・オプション(Advanced spl boot)の設定 | . 49 |
| 7-3-7. Preloader のユーザ・オプション(Advanced spl debug)の設定 | . 50 |
| 7-3-8. bsp プロジェクトの生成(Generate) | . 51 |
| 7-3-9. Preloader のビルド | . 52 |
| 7-4. カスタム・ボード向けに生成した Preloader を DS-5 デバッグで使用する方法 | . 54 |
| 7-4-1. Preloader ファイルの差し替え | . 54 |
| 7-4-2. Makefile の編集 | . 55 |
| 8. ベアメタル・アプリケーションを SD カードからスタンドアローン実行する例 | 56 |
| 8-1. SD カードの準備 | . 56 |
| 8-1-1. SoC EDS v13.1 以前のバージョン で Preloader を SD カードに書き込むには | . 57 |
| 8-1-2. SoC EDS v14.0 以降のバージョン で Preloader を SD カードに書き込むには | . 57 |
| 8-2. DS-5 プロジェクトへの追加ファイル | . 58 |
| 8-3. DS-5 プロジェクトの Makefile の修正 | . 59 |
| 8-4. SD カード実行のためのソース・ファイルの変更 | . 60 |
| 8-5. ベアメタル・アプリケーションのビルド | . 60 |
| 8-6. ベアメタル・アプリケーションの実行バイナリの作成と起動 | . 61 |
| 8-6-1. アプリケーションを SD カードの「FAT 領域」に配置して "u-boot から起動" する方法 | . 61 |
| 8-6-2. アプリケーションを SD カードの「u-boot 領域」に格納して"Preloader から起動"する方法 | . 63 |
| 改版履歴 | 65 |

1. <u>はじめに</u>

本資料では Arm[®] DS-5 (以下、DS-5)を利用したインテル[®] SoC FPGA (以下、SoC FPGA)向けベアメタル・アプリケーションの開発およびデバッグ手法について解説しています。

また、本資料では Cyclone[®] V SoC FPGA 評価キット「DEO-Nano-SoC Kit / Atlas-SoC Kit」(以下、Atlas-SoC ボード)、または「DE10-Nano Kit」(以下、DE10-Nano ボード)向けの ベアメタル・サンプル・アプリケーション Atlas-Blinking-LED-Baremetal-GNU を例として説明しています。

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 DS-5 で実行する方法として、
 - インテル[®] SoC FPGA エンベデッド開発スイート(以下、SoC EDS)に付属のサンプル・アプリケーションの紹介
 - ・ 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およびデバッグ
 - ・ デバッガ・スクリプト・ファイルでの実行内容(参考)
- ④ 新規にベアメタル・アプリケーションを作成して DS-5 で実行する方法として、
 - ・ ベアメタル・アプリケーションの新規作成
 - ・ ベアメタル・アプリケーションのビルドおよびデバッグ
- ⑤ FPGA レジスタの確認方法
- ⑥ カスタム・ボードへの対応方法として、
 - ・ Arm プロセッサを含むハードウェアの設計を行う
 - Preloader (プリローダ) とは?
 - Preloader の生成手順
 - ・ カスタム・ボード向けに生成した Preloader を DS-5 デバッグで使用する方法
 - ・ 生成した Preloader を SD カードに書き込む方法
- ⑦ ベアメタル・アプリケーションを SD カードからスタンドアローン実行する例
 - ・ SD カードの準備
 - ・ アプリケーションを SD カードの「FAT 領域」に配置して "u-boot から起動" する方法
 - ・アプリケーションを SD カードの「u-boot 領域」に格納して "Preloader から起動" する方法



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 Professional SP1(64 bit) 搭載の 64 bit マシン 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。 |
| 2 | インテル® Quartus® Prime ス | SoC FPGA のハードウェアを開発するためのツールです。 ソフトウェア開発に必要なハンドオフ・ファイルの生成も行います。 |
| | ダンダート・エティ ション開発ソフトウ | 本資料では、Quartus Prime スタンダード・エディション開発ソフトウェア v18.0 を使用しています。 |
| | エア | ■ Quartus Prime スタンダード・エディション v18.0 (以降、Quartus Prime) http://fpgasoftware.intel.com/18.0/?edition=standard&download_manager=dlm3&platform=windows |
| | | ▲ 注記: 本資料の説明では Cyclone® V を使用していますので、Device データとして Cyclone® V 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。 https://service.macpica.co.ip/library/118817 |
| 3 | インテル® SoC | Soc FPGA のソフトウェアを開発するためのツールです。 |
| | FPGA エンベデッ ド開発スイート (SoC EDS)スタン | ハンドオフ・ファイルを使用して、ターゲット・ボード固有の Preloader(プリローダ)を作成します。また SoC EDS に含まれる DS-5 Intel® SoC FPGA Edition を使用して、アプリケーション・ソフトウェアをコンパイル しデバッグすることができます。 |
| | ダート・エティンヨ ン | 本資料では、SoCEDS スタンダード・エディション v18.0 を使用しています。 |
| | (DS-5 Intel [®] SoC | ■ SoC EDS スタンダード・エディション v18.0 |
| | FPGA Edition) | http://fpgasoftware.intel.com/soceds/18.0/?edition=standard&platform=windows&download_manager=dlm3 |
| | | |
| 4 | Atlas-SoC ボード または | 本資料の説明でターゲット・ボードとして使用する、Cyclone® V SoC を搭載した Terasic 社 Atlas-SoC / DE10-Nano ボードです。 |
| | DE10-Nano ボード | ■ Atlas-SoC ボード https://rocketboards.org/foswiki/view/Documentation/AtlasSoCDevelopmentPlatform |
| | | http://www.terasic.com.tw/cgi- |
| | | bin/page/archive.pl?Language=English&CategoryNo=205&No=941&PartNo=4#Alt |
| | | ■ DE10-Nano ボード https://rocketboards.org/foswiki/view/Documentation/DE10NanoDevelopmentBoard |
| | | https://www.terasic.com.tw/cgi- bin/page/archive.pl?Language=English&CategoryNo=205&No=1046&PartNo=4 |
| 5 | ベアメタル・サンプ ル・アプリケーショ | 本資料の「4. ベアメタル・サンプル・アプリケーションを DS-5 で実行する方法」の説明で使用する、 Atlas-SoC/DE10-Nano ボード上で動作する、LED 点滅ベアメタル・サンプル・アプリケーションです。 |
| | ン | Arm プロセッサから FPGA ファブリック側に実装された PIO ペリフェラルにアクセスし LED の点灯、消 灯を制御します。 |
| | | 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。 Atlas-Blinking-LED-Baremetal-GNU.zip |
| | | このファイルを解凍すると、Atlas-Blinking-LED-Baremetal-GNU.tar.gz ファイルが含まれています。 |
| | | 本資料の説明では、Atlas-Blinking-LED-Baremetal-GNU.tar.gz を C:¥Temp に格納したものとして説明して います。 |
| | | ▲ 注記: 本サンプルを実行する前に、ターゲット・ボード(Atlas-SoC / DE10-Nano)に .sof ファイルをダウンロード してください。 |

| à | | Division Company | |
|---|---|---|---|
| | 6 | Cyclone® V 向け GCC リンカ・スクリ プト・ファイル | 本資料の「5. 新規にベアメタル・アプリケーションを作成して DS-5 で実行する方法」の説明で使用す る、Cyclone® V 向け GCC リンカ・スクリプト・ファイルです。 このリンカ・スクリプトでは、64KB の内部 RAM (On-Chip RAM) にプログラムのダウンロードを指定する とともに、DS-5 のセミホスティング機能の利用をリンカへ指示します。 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。 CycloneV-dk-oc-ram-hosted_ld.zip このファイルを解凍すると、cycloneV-dk-oc-ram-hosted.ld ファイルが含まれています。 本資料の説明では、cycloneV-dk-oc-ram-hosted.ld を C:¥Temp に格納したものとして説明しています。 |
| | 7 | スタンドアローン 実行用オプション・ ファイル | 本資料の「8. ベアメタル・アプリケーションを SD カードからスタンドアローン実行する例」の説明で使用する、スタンドアローン実行用オプション・ファイルです。 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。 Atlas-Blinking-LED-Baremetal-GNU Additional files for stand-alone.zip このファイルを解凍すると、 altera-socfpga-unhosted.ld Makefile startup.s ファイルが含まれています。 |

参考:

- 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・ガイドが参考になります。
 - 『<u>Bare Metal User Guide UG-01165</u>』(英語版)
 - ・『<u>ベアメタルのユーザ・ガイド UG-01165</u>』(日本語版)
 - ・『<u>UG-01165: Bare Metal User Guide --> Errata Intel</u>』(英文ページ)
- 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。
 - ・『<u>Intel SoC FPGA Bare-metal Developer Center</u>』(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。
 - 『<u>Soc Bare-metal Programming and Hardware Libraries Intel</u>』(英語、28 分)

2. <u>事前準備</u>

本資料では ターゲット・ボードとして Atlas-SoC ボード または DE10-Nano ボード 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
 - Atlas-SoC ボードのレイアウト図を以下に示します。DE10-Nano ボードも基本的には同じです。



【図 2-1】 Atlas-SoC ボード・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)を DC 入力 (J14) 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(J13)を接続します。

2-1-3.SW10 の設定

SW10 (MSEL 設定スイッチ) 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、FPGA は FPPx32 モードとなります。

| ボード・リファレンス | 信号名 | 設定 |
|------------|-------|-----------|
| SW10. 1 | MSELO | ON ("0") |
| SW10. 2 | MSEL1 | OFF ("1") |
| SW10. 3 | MSEL2 | ON ("0") |
| SW10.4 | MSEL3 | OFF ("1") |
| SW10. 5 | MSEL4 | ON ("0") |
| SW10.6 | N/A | N/A |

【表 2-1】SW10 の設定

0 1 2 3 4 5 6 "] 0 1 2 3 4 MSEL



2-2. FPGA デザイン・ファイル

本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 Atlas-Blinking-LED-Baremetal-GNU を実行する前に、ターゲット・ボードに FPGA デザイン・ファイル (.sof) をダウンロードしておく必要があります。

2-2-1. FPGA デザイン・ファイルの入手先

Terasic 社の以下のページから CD-ROM イメージをダウンロードします。

- ① Atlas-SoC ボード
 <u>http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=205&No=941&PartNo=4#Alt</u>
 CD-ROM イメージ: DE0-Nano-SoC CD-ROM (rev.D0 Board)
- ② DE10-Nano ボード
 <u>https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=205&No=1046&PartNo=4</u>

 CD-ROM イメージ: DE10-Nano CD-ROM (rev. B2 Hardware)

① Note: <u>本資料の説明では、入手したハードウェア・デザインを</u> C:¥Work 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とファイルを使用します。

これらのフォルダとファイルを「ハンドオフ・ファイル」と呼びます。ベアメタル開発において、特に重要な「ハンドオフ・ファイル」は次の 3 つです。

(1) <Quartus プロジェクト>¥output_files フォルダ

正しく生成されていれば、通常は output_files フォルダの中に **.sof** という拡張子のファイルが出力さ れているはずです (Quartus プロジェクトの設定や構成に依存します)。このファイルを SoC FPGA にプロ グラムして FPGA をコンフィグレーションします。

1 注記:

この説明では、Terasic 社のページからダウンロードした CD-ROM イメージを使用して、ハンドオフ・ファイルを 生成しています。

Atlas-SoC ボードの場合は、以下の例のように output_files フォルダが無く、DE0_NANO_SOC_GHRD フォルダ の直下に .sof ファイルが生成されることに注意してください。

<u>Atlas-SoC ボード</u>の場合は、

 $C: \verb"work" DE0-Nano-Soc".v. 1.3.0_HWrevD0_SystemCD" Demonstrations" Soc_FPGA" DE0_NANO_SOC_GHRD" Soc".system.sof" Demonstrations" Desc." Des$

<u>DE10-Nano ボード</u>の場合は、

 $C: \label{eq:c:work} E10-Nano_v. 1.2.2_HWrevB2_SystemCD \\ \end{tabular} Demonstrations \\ \end{tabular} Soc_FPGA \\ \end{tabular} Demonstrations \\ \end{tabular} Soc_FPGA \\ \end{tabular} Demonstrations \\ \end{tabular} Soc_GHRD \\ \end{tabular} so \\ \end{tabular} Soc_GHRD \\ \end{tabular} so \\ \end{ta$

(2) <Quartus プロジェクト>¥hps_isw_handoff¥soc_system_hps_0 フォルダ

正しく生成されていれば、hps_isw_handoff¥soc_system_hps_0 フォルダ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、「7-3. 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 フォルダのパスを指定するので覚えておいてください。

(3) <Quartus プロジェクト>¥<Platform Designer プロジェクト>¥synthesis フォルダ

正しく生成されていれば、synthesis フォルダの中に **.svd** という拡張子のファイルが出力されているは ずです。この .svd ペリフェラル記述ファイルは、「6. FPGA レジスタの確認方法」に利用します。

DS-5 Intel[®] SoC FPGA Edition で、この <Platform Designer プロジェクト>¥synthesis フォルダのパスを指定 するので覚えておいてください。

1 注記:

この説明では、Terasic 社のページからダウンロードした CD-ROM イメージを使用して、ハンドオフ・ファイルを生成して います。.svd ファイルは下記の場所に生成されます。

<u>Atlas-SoC ボード</u>の場合は、

C:¥Work¥DE0-Nano-SoC_v.1.3.0_HWrevD0_SystemCD¥Demonstrations¥SoC_FPGA¥DE0_NANO_SOC_GHRD¥soc_system¥synthesis¥soc_system_hps_0_hps.svd DE10-Nano ボードの場合は、

C:\U00e4Work\U00e4Demonstrations\U00e4Soc_FPGA\U00e4Demonstrations\U00e4Soc_GHRD\U00e4soc_system\u00e4synthesis\u00e4soc_system\u00e4ps_0\u00e4ps.svd

2-2-3. FPGA デザイン・ファイルをターゲット・ボードへダウンロードする方法

上記で入手したハードウェア・デザイン (sof ファイル)を FPGA にダウンロードします。

「2-1. ボードの設定」のセクションを参照し、ボードのセットアップが完了していることを再度確認してください。 セットアップに問題がなければ、J14 に AC アダプタを接続してください。

- ____1. Quartus Prime メニューの「Tools」⇒「Programmer」、または Programmer アイコン 🌺 をクリックし、 Programmer を起動します。
- ____2. Programmer 内にある [Hardware Setup] ボタンをクリックし、Hardware Setup ウィンドウ内の Currently selected hardware のプルダウンリストから DE-SoC を選択し、ウィンドウを Close します。

| Hardware Setup | DE-SoC [USB-1] | | | Mode: JTAG |
|--|--|---|------------------|-------------------------|
| Enable real-time ISP to | o allow background programming whe | n available | | |
| | Ardware Setup | | | |
| ▶ [™] Start | | | | _ |
| Stop | Hardware Settings JTAG Se | ttings | | |
| | Coloct a programming bardware | setup to use when | n programming de | vices. This programming |
| Auto Datast | Select a programming nardware | setup to use when | P0 | |
| Auto Detect | hardware setup applies only to | the current program | nmer window. | |
| Auto Detect | hardware setup applies only to the Currently selected hardware | the current program | nmer window. | • |
| Auto Detect Control Delete Add File | Currently selected hardware: | the current program DE-SoC [USB-1] | nmer window. | • |
| Auto Detect Auto Detect Auto Delete Add File Change File | Currently selected hardware Available hardware items | the current program DE-SoC [USB-1] Server | Port | Add Hardware |
| Auto Detect Auto Detect Auto Detect Auto Detect Colored Change File Save File | Available hardware DE-SoC | Server Local | Port USB-1 | Add Hardware |
| Auto Detect Auto Detect Add File Change File Save File | Available hardware DE-SoC | DE-SoC [USB-1] Server Local | Port USB-1 | Add Hardware |
| Auto Detect Auto Detect Add File Change File Add Device | Available hardware DE-SoC | DE-SoC [USB-1] Server Local | Port USB-1 | Add Hardware |

[2-3] Hardware Setup

___3. [Auto Detect] ボタンをクリックし、基板上の JTAG チェインに接続されている FPGA を検出します。

____4. Select Device ウィンドウから

<u>Atlas-SoC ボードの場合は 5CSEMA4</u>

<u>DE10-Nano ボードの場合は</u>5CSEBA6

を選択し、[OK] をクリックします。

Atlas-SoC ボードの場合

DE10-Nano ボードの場合

| Quartus Prime Programmer Standard Edition - [Chain1.cdf] | Quartus Prime Programmer Standard Edition - [Chain1.cdf]* | | | | | |
|--|--|--|--|--|--|--|
| File Edit View Processing Tools Window Help | File Edit View Processing Tools Window Help | | | | | |
| | • | | | | | |
| Hardware Setup DE-SoC [USB-1] | Hardware Setup) DE-SoC [USB-1] Mode: JTA | | | | | |
| Enable real-time ISP to allow background programming when available | Enable real-time ISP to allow background programming when available | | | | | |
| Select Device | Select Device | | | | | |
| Found devices with shared JTAG ID for device 2. Please select your device. | Found devices with shared JTAG ID for device 2. Please select your device. | | | | | |
| ScSEBA4 | SCSEBAG SCSEBAG | | | | | |
| SCSXFC4C6 | Delete O 5CSEMAG | | | | | |
| Mdd File | Add File O SCSTFD6D5 | | | | | |
| Change File | Change File © 5CSXFC6C6 | | | | | |
| 🕒 Save File | Save File © 5CSXFC6C6ES | | | | | |
| Add Device | Add Device O 5CSXFC6D6 | | | | | |
| 1 ¹ Up | T [™] Up © SCSXFC6D6ES | | | | | |
| СК | СК Down | | | | | |

【図 2-4】デバイスの選択

___5. 以下のダイアログ・ボックスが表示された場合は、[Yes] を選択します。



【図 2-5】ダイアログ・ボックス

これにより、JTAG チェイン 上に SOCVHPS と 5CSMA4 / 5CSEBA6 が表示されます。SOCVHPS は Hard Processor System (以降 HPS) 側、5CSMA4 / 5CSEBA6 は FPGA 側が認識されたことをそれぞれ示しています。

___6. ダウンロードするファイルを選択します。

Device 欄の 5CSEMA4 / 5CSEBA6 上で右クリックし、「Change File」をクリックします。

Select New Programming File ダイアログ・ボックスにおいて、

<u>Atlas-SoC ボードの場合は</u>、

 $C: \verb"work" DE0-Nano-SoC_v.1.3.0_HWrevD0_SystemCD" Demonstrations" SoC_FPGA" DE0_NANO_SOC_GHRD \\ \verb"soc" system.sof" Soc" \\ \verb"soc" system.sof" \\ "soc" syst$

<u>DE10-Nano ボードの場合は、</u>

 $C: \verb"work" DE10-Nano_v.1.2.2_HWrevB2_SystemCD" Demonstrations" Soc_FPGA" DE10_NANO_Soc_GHRD" sot_GHRD" Soc_GHRD.sof Demonstrations and the set of the se$

を選択します。





____7. 「Program/Configure」にチェックを入れた後、[Start] ボタンをクリックしてコンフィグレーションを行いま す。この動作により FPGA 側に動作イメージが書き込まれた状態となります。

| Juartus Prime Pr | rogrammer Standard Edi | tion - [Chain1.cor] | 1 | | | | | | | | | 25 |
|--|--|--|--|---|-------------------------------|--------|----------------------|----------------------|--------------------------------|--|-----------------|----|
| <u>E</u> dit <u>V</u> iew | P <u>r</u> ocessing <u>T</u> ools <u>W</u> in | dow <u>H</u> elp | | | | | | | Sear | rch altera | .com | 6 |
| | | | | | | | | | | | | _ |
| Hardware Setup | DE-SoC [USB-1] | | | Mode: | JTAG | | • | Progress: | | | | |
| Enable real-time | ISP to allow background pr | ogramming when a | available | | | | | | | | | |
| M Start | File | Device | Checksum | Usercode | Program/ Configure | Verify | Blank- Check | Examine | Security Bit | Erase | ISP CLAMP | |
| Stop | <none></none> | SOCVHPS | 00000000 | <none></none> | | | | | | | | |
| Auto Detect | C:/Work/DE0-Nano-So | 5CSEMA4U23 | 02089284 | 02089284 | | | - | - | | - | | |
| K Delete | | | | | | | | | | | | _ |
| Add File | | | | | | | | | | | | - |
| nange File | (intel) | | iteD | | | | | | | | | |
| Save File | | \rightarrow | | | | | | | | | | |
| dd Device | | | | | | | | | | | | |
| T ^m Up | TDO SOCVHP | S 5CSEN | MA4U23 | | | | | | | | | |
| " Down | 10.50 | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| uartus Prime Pr Edit View | rogrammer Standard Edi P <u>r</u> ocessing <u>T</u> ools <u>W</u> in | tion - [Chain1.cdf] Idow <u>H</u> elp |]* | | - | | | | Sear | rch altera. | _ Com | |
| iartus Prime Pr <u>E</u> dit <u>V</u> iew | rogrammer Standard Edi P <u>r</u> ocessing <u>T</u> ools <u>W</u> in | tion - [Chain1.cdf] dow <u>H</u> elp |]* | | - | | | | Sear | rch altera | _ D | |
| Jartus Prime Pr Edit ⊻iew Hardware Setup | rogrammer Standard Edi Processing Tools Win | tion - [Chain1.cdf] Idow <u>H</u> elp |]* | Mode: | JTAG | | • | Progress: | Sear | rch altera. 0% (Succ | .com | |
| artus Prime Pr Edit ⊻iew Hardware Setup nable real-time | rogrammer Standard Edi P <u>r</u> ocessing <u>T</u> ools <u>Win</u> <u>DE-SoC [USB-1]</u> ISP to allow background pr | tion - [Chain1.cdf] .dow <u>H</u> elp ogramming when a |]* available | Mode: | JTAG | | • | Progress: | Sear | rch altera. 0% (Succ | | |
| artus Prime Pr Edit View lardware Setup able real-time | rogrammer Standard Edi Processing Tools Win DE-SoC [USB-1] ISP to allow background pr File | tion - [Chain1.cdf] Idow Help ogramming when a Device |]* available Checksum | Mode: Usercode | JTAG Program/ Configure | Verify | T Blank- Check | Progress: Examine | Sear 100 Security Bit | cch altera 0% (Succ Erase | com essful) | |
| tus Prime Pr dit View rdware Setup ble real-time Start | rogrammer Standard Edi Processing <u>Tools Win</u> <u>DE-SoC [USB-1]</u> ISP to allow background pr File <none></none> | tion - [Chain1.cdf] dow <u>H</u> elp ogramming when a Device SOCVHPS |]* available Checksum 00000000 | Mode: Usercode <none></none> | JTAG Program/ Configure | Verify | Blank- Check | Progress: Examine | Sear 100 Security Bit | rch altera 0% (Succ Erase | essful) | |
| rtus Prime Pr Edit View ardware Setup able real-time Start Stop uto Detect | rogrammer Standard Edi Processing Tools Win DE-SoC [USB-1] ISP to allow background pr File <none> C:/Work/DE0-Nano-So</none> | tion - [Chain1.cdf] dow Help ogramming when a Device SOCVHPS 5CSEMA4U23 |]* available Checksum 0000000 02089284 | Mode: Usercode <none> 02089284</none> | JTAG Program/ Configure | Verify | Blank- Check | Progress: Examine | Security Bit | 0% (Succ Erase | essful) | |
| artus Prime Pr Edit View tardware Setup nable real-time Start Stop Auto Detect | rogrammer Standard Edi Processing Tools Win DE-SoC [USB-1] ISP to allow background pr File <none> C:/Work/DEO-Nano-So</none> | tion - [Chain1.cdf] idow Help ogramming when a Device SOCVHPS 5CSEMA4U23 | available Checksum 00000000 02089284 | Mode: Usercode <none> 02089284</none> | JTAG Program/ Configure | Verify | Blank- Check | Progress: Examine | Sear 100 Security Bit | rch altera 0% (Succ Erase | com essful) | |
| artus Prime Pr Edit View ardware Setup able real-time No Stop Auto Detect Colete Add File | rogrammer Standard Edi Processing Tools Win DE-SoC [USB-1] ISP to allow background pr File <none> C:/Work/DEO-Nano-So</none> | tion - [Chain1.cdf] dow Help ogramming when a Device SOCVHPS 5CSEMA4U23 | 2* available Checksum 0000000 02089284 | Mode: Usercode <none> 02089284</none> | JTAG Program/ Configure | Verify | Blank- Check | Progress: Examine | Sear 100 Security Bit | CO% (Succ Erase | essful) | |
| artus Prime Pr Edit View ardware Setup able real-time Start Stop Auto Detect Collete Add File hange File | rogrammer Standard Edi Processing Tools Win DE-SoC [USB-1] ISP to allow background pr File <none> C:/Work/DEO-Nano-So</none> | tion - [Chain1.cdf] dow <u>H</u> elp ogramming when a Device SOCVHPS 5CSEMA4U23 | available Checksum 0000000 02089284 | Mode: Usercode <none> 02089284</none> | JTAG Program/ Configure | Verify | Blank- Check | Progress: Examine | Security Bit | Constant of the second se | essful) | |
| artus Prime Pr <u>E</u> dit <u>V</u> iew Hardware Setup hable real-time Start Stop Auto Detect Add File hange File Save File | rogrammer Standard Edi Processing Tools Win DE-SoC [USB-1] ISP to allow background pr File <none> C:/Work/DEO-Nano-So</none> | tion - [Chain1.cdf] idow Help ogramming when a Device SOCVHPS 5CSEMA4U23 | available Checksum 00000000 02089284 | Mode: Usercode <none> 02089284</none> | JTAG Program/ Configure | Verify | Blank- Check | Progress: Examine | Sear 100 Security Bit | Ch altera 0% (Succ | ccom essful) | |
| Jartus Prime Pr Edit View Hardware Setup Inable real-time Stop Auto Detect Change File Save File Add Device | rogrammer Standard Edi Processing Tools Win DE-SoC [USB-1] ISP to allow background pr File <none> C:/Work/DEO-Nano-So</none> | tion - [Chain1.cdf] dow Help ogramming when a Device SOCVHPS 5CSEMA4U23 | available Checksum 00000000 02089284 | Mode: Usercode <none> 02089284</none> | JTAG Program/ Configure | Verify | Blank- Check | Progress: Examine | Sear 100 Security Bit | Comparison of the second secon | essful) | |
| Luartus Prime Pr Edit View Hardware Setup Enable real-time Start Stop Auto Detect Add File Change File Save File Add Device Mup | rogrammer Standard Edi Processing Tools Win DE-SoC [USB-1] ISP to allow background pr File <none> C:/Work/DEO-Nano-So TDI SOCVHP TDO</none> | tion - [Chain1.cdf] dow Help ogramming when a Device SOCVHPS 5CSEMA4U23 SCSEMA4U23 | available Checksum 00000000 02089284 | Mode: Usercode <none> 02089284</none> | JTAG Program/ Configure | Verify | Blank- Check | Progress: Examine | Security Bit | Comparison of the second secon | essful) | |
| Quartus Prime Pri <u>E</u> dit <u>V</u> iew Hardware Setup Enable real-time Start Auto Detect Auto Detect Add File Change File Add Device <u>1</u> ^M Up <u>1</u> ^M Down | rogrammer Standard Edi Processing Tools Win DE-SoC [USE-1] ISP to allow background pr File <none> C:/Work/DEO-Nano-So TDI TDI SOCVHP TDO</none> | tion - [Chain1.cdf] dow Help ogramming when a Device SOCVHPS 5CSEMA4U23 SCSEMA4U23 | available Checksum 00000000 02089284 | Mode: Usercode <none> 02089284</none> | JTAG Program/ Configure | Verify | Blank- Check | Progress: Examine | Sear | Chaltera 0% (Succ | com essful) | |

【図 2-7】sof のダウンロード (Atlas-SoC ボードの場合の例)

3. <u>SoC FPGA のブート・フロー</u>

まずはじめに SoC FPGA のブート・フローについて説明します。

以下の図の通り、Arm のブート・フローには複数のステージが存在します。 ブート・フローに関する詳細は、アプリケーション・ノート 709 (AN 709) をご確認ください。

https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an709.pdf

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】SoC FPGA 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。

Bootloader

ユーザ・ブートローダ・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。

Baremetal Application

インテル[®]のハードウェア・ライブラリ(HWLib)を使用した、直接ハードウェアを読み書きするアプリケーションです。

4. <u>ベアメタル・サンプル・アプリケーションを DS-5 で実行する方法</u>

この章では、既存のベアメタル・サンプル・アプリケーション・プロジェクトを DS-5 にインポートしてデバッグ/ 実行する方法について説明します。

4-1. SoC EDS に付属のサンプル・アプリケーションの紹介(参考)

SoC EDS をインストールすると、SoC EDS に付属しているサンプル・アプリケーションが下記のフォルダに格納されます。

<intel FPGA installation directory>¥embedded¥examples¥software

例) C:¥intelFPGA¥18.0¥embedded¥examples¥software

これらのサンプル・アプリケーションは、DS-5 Intel® SoC FPGA Edition からインポートしてビルドおよびデバッグ することが可能です。

これらのファイルの中で、名前が "Altera-SoCFPGA-HardwareLib-" や "Altera-SoCFPGA-HelloWorld-Baremetal-"となっているものが、ベアメタル対応アプリケーションです。

また、名前に "GNU" と付いているものは GNU コンパイラ (GCC) 版、 "ARMCC" と付いているものは Arm コンパイラ (ARMCC) 版となります。

| a 퉲 18.0 | 名前 | 更新日時 | 種類 | サイズ |
|---------------------|---|------------------|----------|--------|
| 4 鷆 embedded | Altera SoCEDCA Blinking LED Linux CNUL bases | 2019/04/26 07:20 | C7 77 41 | EV |
| drivers | Altera-SocEPCA-binking-LED-Liflux-GNU.ldf.gz | 2010/04/20 07:29 | GZ 7711 | 2 KI |
| b ds-5 | Altera SocePicA Hardwarel ib 16550 CV CNU targe | 2018/04/20 07:29 | GZ 77710 | 23 KI |
| de-5 installer | Altera-SoCEPCA-Hardwarel ib-ECCI 2-CV_APMCC targa | 2010/04/20 07:29 | GZ 77470 | 15 VI |
| p i us-5_iristaller | Altera SocePicA Hardwarel in ECCL2 CV -ARMCC.tdl.gz | 2010/04/26 07:29 | GZ JF1N | 15 KI |
| embeddedsw | Altera-SoCEPCA-Hardwarel ib-EDCA-CV-APMCC targa | 2010/04/20 07:29 | GZ 77711 | 10 KI |
| 🧧 🍌 examples | Altera-SocEPCA Hardwarel ib-EPCA-CV-ARMOC.tdl.yz | 2010/04/26 07:29 | GZ JF1N | 13 1 |
| 🛛 📙 hardware | Altera-SocEDCA-Hardwarel ib-SDL-CV-ARMCC targe | 2010/04/20 07:29 | GZ 77710 | 12 KI |
|) software | Altera-SocEPGA-Hardwarel ib-SPI-CV-ARMCC.tdl.yz | 2010/04/26 07:29 | GZ 7771 | 202 12 |
| b host_tools | Altera-SocEPGA-Hardwarel ib-Timer-A10-ARMCC targz | 2010/04/26 07:20 | GZ JRAN | 15 KI |
| > 📔 ip | Altera-SocEPGA-Hardwarel ib-Timer-A10-GNU tar oz | 2018/04/26 07:20 | G7 7741 | 14 KI |
| b 🌗 hld | Altera-SocEPGA-Hardwarel ib-Timer-AV-ARMCC tar oz | 2018/04/26 07:29 | G7 7741 | 14 KI |
| hlc | Altera-SocEPGA-Hardwarel ib-Timer-AV-GNU targz | 2018/04/26 07:29 | G7 7741 | 14 KI |
| | Altera-SocFPGA-HardwareLib-Timer-CV-ARMCC.tar.oz | 2018/04/26 07:29 | GZ ファイル | 14 KI |
| Þ 📗 ip | Altera-SocFPGA-HardwareLib-Timer-CV-GNU.tar.07 | 2018/04/26 07:29 | GZファイル | 14 KI |
| 🛚 🎳 licenses | Altera-SocFPGA-HelloWorld-Baremetal-ARMCC.tar.gz | 2018/04/26 07:29 | GZファイル | 5 KI |
| 🍌 logs | Altera-SocFPGA-HelloWorld-Baremetal-GNU.tar.gz | 2018/04/26 07:29 | GZ ファイル | 5 KI |
| Imodelsim_ase | Altera-SoCFPGA-HelloWorld-Linux-GNU.tar.gz | 2018/04/26 07:29 | GZ ファイル | 3 KI |
| 🛛 🍌 nios2eds | ☐ Altera-SoCFPGA-Push-Button-Linux-GNU.tar.oz | 2018/04/26 07:29 | GZファイル | 4 KI |
| - | · | | | |

【図 4-1】SoC EDS 付属のサンプル・アプリケーション

4-2. DS-5 の起動

SoC EDS に含まれている DS-5 Intel[®] SoC FPGA Edition を起動します。

SoCEDS 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。

4-2-1. 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 (embedded フォルダ) 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 4-2】 Embedded Command Shell の起動

4-2-2.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 & J とコマンド入力して DS-5 Intel® SoC FPGA Edition を起動します。







_2. ワークスペース・フォルダ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。フォルダが存在しない場合は自動的に作成されます)。

| 🖨 Eclipse Launcher | × |
|--|---|
| Select a directory as workspace | |
| Eclipse $J \exists y \land J \exists - L$ uses the workspace directory to store its preferences and development artifacts. | |
| | |
| ワークスペース(W): C:¥Work¥DS-5 Workspace ▼ 参照(B) | |
| | |
| | |
| この選択をデフォルトとして使用し、今後この質問を表示しない(U) | |
| Recent Workspaces | |
| | |
| | _ |
| OK キャンセル | |
| | |

【図 4-4】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 4-5】DS-5 ウェルカム画面

4-3. 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、事前に用意された Atlas-SoC / DE10-Nano ボード向け LED 点滅ベアメタル・サンプル・アプリケ ーション Atlas-Blinking-LED-Baremetal-GNU を DS-5 にインポートします。

- ____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。

| 0 | S-5 Workspace - C/C++ - Eclipse J | ● インボート | - • • |
|----------|--|---|-------|
| [ファ | イル(F) 編集(E) ソース(S) リフ 新規(N) ファイルを聞く(.) Open Projects from File System | 選択 アーカイブ・ファイルまたはディレクトリーから新規プロジェクトを作成します。 | Ľ |
| | 閉じる(C) すべて閉じる(L) | | |
| | 保管(S) 別名保存(A) すべて保管(E) 前回保管した状態に戻す(T) | ・・システム ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ | F |
| 1 | 移動(V) 名前を変更(M) 更新(F) 行区切り文字の変換(D) | $ \begin{array}{c} & \searrow \\ & & \bigcirc \\ & & & &$ | |
| 4 | ^{印刷(P)} ワークスペースの切り替え(W) 再開 | Scatter File Editor | |
| 2 | インボート(I) エクスボート(0) | (P) 次へ(N) > 終了(F) | キャンセル |

【図 4-6】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ます。 [参照(R)] ボタンより、以下のサンプル・ プロジェクトを指定します。 Atlas-Blinking-LED-Baremetal-GNU.tar.gz 選択後、 [終了(F)] ボタンを押します。

() Note:

<u>本資料の説明では、Atlas-Blinking-LED-Baremetal-GNU.tar.gz</u>をC:¥Tempに格納したものとして説明 しています。

| ● インポート | | |
|---|---|-------------|
| プロジェクトのインボート 既存の Eclipse プロジェクトを検索する | るディレクトリーを選択します。 | |
| | | - |
| ◎ ルート・ディレクトリーの選択(T): | | ▼ 参照(R) |
| アーカイブ・ファイルの選択(A): | C:¥Temp¥Atlas-Blinking-LED-Baremetal-GNU.tar.gz | ▼ 参照(R) |
| プロジェクト(P): | | |
| Atlas-Blinking-LED-Baremeta | l-GNU(Atlas-Blinking-LED-Baremetal-GNU/) | すべて選択(S) |
| | | 選択をすべて解除(D) |
| | | 更新(E) |
| オプション | | |
| マネストしたプロジェクトを検索(H) | | |
| ワークスペースに既に存在するプロ | ビー(C) ジェクトを隠す(I) | |
| ワーキング・セット | | |
| ワーキング・セットにプロジェクト | ~を追加(T) | 新規(W) |
| ワーキング・セット(0): | | ▼ 選択(E) |
| | | |
| | | |
| | | |
| ? < 戻る(B) | 次へ(N) > 終了(F) | キャンセル |
| | | |
| 【図 4- | 7】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 Atlas-Blinking-LED-Baremetal-GNU が追加され、Atlas-Blinking-LED-Baremetal-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。







4-4. 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。

4-4-1. Makefile の例(参考)

サンプル・アプリケーションの Makefile の例を下図に示します。

() Note:

ご利用のサンプル・アプリケーションにより、Makefile の記述内容が異なる場合があります。

| là *Makefile 🛛 | |
|---|-----------------|
| 36 37 ALT_DEVICE_FAMILY ?= soc_cv_av 38 | |
| 35 SOCEDS_ROOT ?= \$(SOCEDS_DEST_ROOT) 40 HWLIBS_ROOT = \$(SOCEDS_ROOT)/ip/altera/hps/altera_hps/hwlib | |
| 42 HWLIBS_SRC := alt_clock_manager.c alt_generalpurpose_io.c alt_globaltmr.c alt_interrupt.c alt_timers.c alt_watchdog.c 43 EXAMPLE_SRC := hulib.c 44 C_SRC := \$(EXAMPLE_SRC) \$(HWLIB5_SRC) | アイルの指定 |
| 45 44 47 リンカ・ファイルの指定 | |
| <pre>48 MULILLBFLAGS := -mcpu=cortex-a9 =mtloat=ab1=sottp =mtpu=neon 49 CFLAGS := -g =00 =Wall -kerror -std=c99 {(MULTLIBFLAGS) -1\$(HWLIBS_ROOT)/include -1\$(HWLIBS_ROOT)/include/\$(ALT_DEVICE_FAMILY) -D\$(ALT_DEVICE_FAMILY) 50 LDFLAGS := -T\$(LINKER_SCRIPT) \$(MULTLIBFLAGS) -1\$(HWLIBS_ROOT)/include -1\$(HWLIBS_ROOT)/include/\$(ALT_DEVICE_FAMILY) -D\$(ALT_DEVICE_FAMILY) 51 - 1</pre> | |
| 52 CROSS_COMPILE := arm-altera-eabi- 53 CC := \$(CROSS_COMPILE)gcc 54 LD := \$(CROSS_COMPILE)gr+ 55 NM := \$(CROSS_COMPILE)nm 56 OD := \$(CROSS_COMPILE)nm 56 OD := \$(CROSS_COMPILE)objdump 56 OD := \$(CROSS_COMPILE)objcopy 56 NM := rm -rf 60 CP := cp -f | |
| <pre>61 62 ELF ?= \$(basename \$(firstword \$(C_SRC))).axf 63 SPL := u-boot-spl.axf 64 083 := \$(patsubst %.c,%.o,\$(C_SRC)) 65 65 65 65 65 65 65 65 65 65 65 65 65</pre> | |
| 66.PHONY: all 67 all: \$(ELF) \$(SPL) 68 | |
| 69.PHDNY: clean 70 clean: 71 = {/DM\ \$(FFF) \$(SPF) \$(ORT) * objdumn * man \$(HW TRS SPC) | |
| 72 73 73 73 73 73 73 74 74 73 75 75 75 75 75 75 75 75 75 75 75 75 75 | |
| 74 s(1): s(2) | |
| 77 78 ALL_HWLIBS_SRC = \$(wildcard \$(HWLIBS_ROOT)/src/hwmgr/*.c) \$(wildcard \$(HWLIBS_ROOT)/src/hwmgr/\$(ALT_DEVICE_FAMILY)/*.c) | |
| <pre>//9 80 \$(foreach file,\$(ALL_HWLIBS_SRC),\$(eval \$(call SET_HWLIBS_DEPENDENCIES,\$(notdir \$(file)),\$(file)))) 91 93 94 94 94 94 94 94 94 94 94 94 94 94 94</pre> | |
| 22\$(OBJ): %.o: %.c Makefile 83 \$(CC) \$(CFLAGS) -c \$< -o \$@ | |
| S5\$(ELF): \$(OBJ) 86 \$(LD) \$(LDELAGS) \$(OBJ) -o \$@ 87 \$(OD) -d \$@ > \$@.objdump 88 \$(NM) \$@ > \$@.map | |
| 9g \$(SPL): \$(SOCEDS_ROOT)/examples/hardware/cv_soc_devkit_ghrd/software/preloader/uboot-socfpga/spl/u-boot-spl 91 \$(CP) \$< \$@ 92 \$(D0) -d \$@ > \$@.objdump | u-boot-spl)の⊐ピー |
| | <i>,</i> |



Makefile の中では主に次のものが定義されています。

- SoC FPGA デバイス・ファミリの指定
- SoC EDS のパス設定
- ビルド対象のソース・ファイルの指定
- リンカ・ファイルの指定
- クロス・コンパイラの指定
- HWLib ソース・ファイルのコピー
- Preloader 実行可能バイナリ (u-boot-spl.axf) のコピー

1 注記:

クロス・コンパイラの指定は、ご使用の SoC EDS のバージョンにより以下のように異なりますのでご注意ください。

- ・ v13.1 までは、 CROSS_COMPILE := arm-none-eabi-
- ・v14.0 からは、CROSS_COMPILE := arm-altera-eabi-

4-4-2. プロジェクトのビルド

DS-5 プロジェクト(この例では、Atlas-Blinking-LED-Baremetal-GNU サンプル・プロジェクト)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 .axf ファイル (この例では、atlas_main.axf) が生成されます。



【図 4-10】 プロジェクトのビルド



4-4-3. ベアメタル・サンプル・アプリケーション・プロジェクトのファイル構成

サンプル・アプリケーションのファイル構成例を下図に示します。



【図 4-11】サンプル・アプリケーションのファイル構成例

この中で特に重要なファイルとしては次のものがあります。

- Makefile : プロジェクトをビルドする際の指示書
- debug-hosted.ds : デバッグ時に実行するスクリプト・ファイル
- <アプリケーション名>.axf : サンプル・アプリケーションの実行可能バイナリ
- u-boot-spl.axf : Preloader の実行可能バイナリ

ALTIMA

4-5. ベアメタル・サンプル・アプリケーションのデバッグ

次にビルドしたベアメタル・サンプル・アプリケーションをデバッグします。

デバッグを実行する前に「2. 事前準備」が完了していることを確認してください。

4-5-1. デバッグの実行

____1. DS-5 プロジェクト(この例では、Atlas-Blinking-LED-Baremetal-GNU サンプル・プロジェクト)をハイライトし、 右クリックして「デバッグ(D)」⇒「デバッグの構成(B)」を選択します。

| ● DS-5 Workspaces - C/C++ - Eclipse プラットフォーム | | | | |
|---|------------|---|-------------------------|--|
| ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H) | | | | |
| | 10 | - 🚳 - C - C - k - O - 4 - B - R I - R - R - R - R - R - R - R - R - |] • 🌾 🗇 • 🔿 • | |
| 🏷 プロジェクト・エクスプローラー 🛙 | | | | |
| ■ 🊰 Atlas-Blinking-LED-Baremetal-GNU ▷ 繰 バイナリー ▷ 劒 Includes | J • | マウスで右クリック 新規(N) 次ヘジャンプ(I) | | |
| ▶ is atlas_main.c ▶ is atlas_main.axf - [arm/le] ▶ is atlas main.o - [arm/le] | | 新規ウィンドウで開く(N) Show In ・ | | |
| [*] u-boot-spl.axf - [arm/le] [*] altera-socfpga-hosted.ld [*] atlas_main.axf.map [*] atlas_main.axf.objdump [*] Atlas-Blinking-LED-Baremetal-D | | Copy Paste 削除(D) 移動(V) 名前を変更(M) | | |
| 📄 debug-hosted.ds 🗋 Makefile | | インボート(I) エクスボート(0) | | |
| | | プロジェクトのビルド(B) プロジェクトをクリーンにする | | |
| | | 更新(F) F5 プロジェクトを閉じる(S) 無関係なプロジェクトを閉じる(U) | | |
| | | Make ターゲット ト インデックス ト | | |
| | | ビルド構成・ | | |
| | | リモートシステムビューで表示 実行(P) | | |
| | | デバッグ(D) | □ 1 □−カル C/C++ アプリケーション | |
| | | プロファイル(P) | | |
| | | ローカル履歴から復元(Y) | テハック の構成(B) | |

【図 4-12】「デバッグ(D)」 「デバッグの構成(B)」を選択

____2. デバッグ構成ウィンドウにある左側のパネルから、

「**DS-5 デバッガ」** 「Atlas-Blinking-LED-Baremetal-Debug」を選択します (表示されない場合は、 DS-5 デバッガの横にある (+) をクリックしてください)。

ターゲット接続は、インテル[®] FPGA ダウンロード・ケーブル(USB-Blaster™)を利用し、 「Altera」 「Cyclone V SoC (Dual Core)」 「Bare Metal Debug」 「Debug Cortex-A9_0」と なるように設定されています。

_3. 接続セクションの右側にある [参照] ボタンを押下し、USB-Blaster™ 接続の選択画面を表示させます。

| ● デバッグ構成 | |
|---|---|
| 構成の作成、管理、および実行 | - All |
| ○ ● ※ ● 許 ▼ フィルタ入力 ○ C/C++ アプリケーション ○ C/C++ アプリケーションへのアタッチ ○ C/C++ ポストモーテム・デバッガー ○ C/C++ リモート・アプリケーション ◆ DS-57/(ッガ) ▲ Atlas-Blinking-LED-Baremetal-Debug ④ TronPython unittest ⑦ Java アプリケーション Ø Java アプリケーション Ø Java アプリケーション Ø Java アプレット Ju JUnit Ø Jython run Ø PyDev Django Ø PyDev Google App Run Ø Python Run Ø Python Run Ø Python unittest Ø Jut - ト Java アプリケーション ▷ 起動グループ | 名前(N): Atlas-Blinking-LED-Baremetal-Debug ◆ 振然 |
| フィルター一致: 19 / 19 項目 | (前回保管した状態に戻す(∨) 適用(Y) |
| ? | デバッグ(D) 閉じる |

【図 4-13】 デバッグの構成

____4. 接続ブラウザ・ウィンドウで、目的の USB-Blaster™(この例では DE-SoC on localhost)をハイライトして、 [選択] をクリックします。

| ● 接続ブラウザ | | X |
|----------------|----------------|----------|
| 接続ブラウザ | | |
| ターゲット接続を | 選択します | |
| DE-SoC USB-1 | | |
| DE-SoC on loca | ilhost [USB-1] | |
| | | |
| ? | 選択 | キャンセル |
| | | |

【図 4-14】 デバッグ・ケーブルの選択



__5. デバッグ構成ウィンドウの右下にある [**デバッグ(D)**] ボタンをクリックします。

| C 🗈 🗙 🗖 🛱 🕶 | 名前(N): Atlas-Blinking-LED-Baremetal-Debug | |
|--|--|--|
| フィルタ入力 | 🖇 接続 🛛 ファイル 🖓 デバッガ 🏀 OS 認識機能 🚧 引数 🚾 環境 🛃 エクスポート | |
| C/C++ アプリケーション C/C++ アプリケーションへのアタッチ C/C++ ポストモーテム・デパッガー C/C++ リモート・アプリケーション | ターゲットの選択 使用する製造元、ボード、プロジェクトのタイプ、およびデバッグ操作を選択します。 現在の選択内容: Altera / Cyclone V SoC (Dual Core) / Bare Metal Debug / Debug Cortex-A9_0 | |
| ▲ 券 DS-5デバッガ | ブラットフォームのフィルタ | |
| | Altera Arria 10 SoC Arria V SoC Cyclone V SoC (Dual Core) Bare Metal Debug Debug Cortex-A9_0 | |
| e ^y Jython unittest Ø PyDev Django A PyDev Google Ann Run | Debug Cortex-A9_1 ターゲット接続 USB-Blaster ・ | |
| e Python Run | DS-5 Debugger will connect to an Altera USB-Blaster to debug a bare metal application. | |
| ₽ Python unittest 型 リモート Java アプリケーション | 接続 | |
| ▶ 起動グループ | Bare Metal Debug Connection DE-SoC on localhost [USB-1]:DE-SoC USB-1 参照 DTSL オプション 編集 USB-Blaster トレースまたはその他のターゲット オプションを構成します。"default" コ | |
| | (前回,尽管し,た)状態に定す(V)) (適田(V)) | |

【図 4-15】 デバッグの実行

___6. Eclipse は、デバッグ パースペクティブに切り替えるかどうかを尋ねます。 [はい(Y)] をクリックしてそれを 受け入れてください。



【図 4-16】パースペクティブスイッチの確認

Windows ファイアウォールの警告が出た場合は、[アクセスを許可する(A)]をクリックします。



【図 4-17】 セキュリティの警告



ダウンロード時にエラーが発生した場合は、以下の確認を行ってください。

- (1) DS-5 のライセンスが紐づけられているネットワーク・インタフェース(例えば USB-Ethernet Interface ア ダプタ)が有効になっているか確認してください。
- (2) 評価ボードの電源入切および PC の再起動で復旧しないか確認してください。評価ボードの電源を切った場合は、再度 FPGA のデータをダウンロードすることを忘れないでください。

デバッガは起動スクリプトの指示に従いセミホスティング機能を有効にした後、JTAG を経由してアプリケーションをボードにダウンロードします。

プログラム・カウンタ が main 関数に到達するとブレークされデバッグが開始できる状態となります。 この段階では、DS-5 のすべてのデバッグ機能を使用することができます(レジスタや変数の表示と編集、逆 アセンブリ・コードの参照、など)。

____7. 緑色の「続行」 🕨 ボタンをクリックして(または F8 キーを押して)アプリケーションを実行します。

これにより、**アプリケーションコンソール** に Hello from Atlas. メッセージ が表示されます。 更に LED [0] ・・・ LED [f] メッセージ が表示され、Atlas-SoC ボード上のユーザ LED (LED [3:0])の点 灯状態が変化することを確認します。

| 😑 DS-5 Workspaces - DS-5 デバッグ - Atlas-Blinking-LED-Baremetal-GNU/atlas_main.c - Eclipse プラットフォーム | | | | |
|---|---|--|--|--|
| ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナピゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H) | | | | |
| · □ ▼ 圖 幅 晶 : 巻 ▼ 目 ■ 图 : ● ペ ▼ : ♪ : ● ペ ▼ : ♪ : ● ペ ▼ : ♪ : ● □ = □ = □ = □ = □ = □ = □ = □ = □ = □ | | | | |
| ※デバッグコ ※ № プロジェク 湯 リモート・ □ □ | 🖬 コマンド 🛙 📷 履歴 🎉 スクリプト 🛛 🕫 🗖 | (M= 変数 X2 💁 ブレークボイ 🚥 レジスタ XV 式 f() 関数 🖳 🗖 | | |
| □ \$\$\$ \$\$\$ \$\$\$ \$\$ \$\$ \$\$ \$\$ \$\$ \$\$ \$\$ \$\$ \$\$ | 📓 📴 📮 🎜 🕈 🏶 | 9x 😫 🔗 🔶 マ | | |
| ▽ ▲ 入tlas-Blinking-LED-Baremetal-Debug 接続 | write of (7%) (Rads-billioning-LED-balemetal-Debug | An 値 型 カウント サイズ 場所 アクセス | | |
| 篇 Cortex-A9_0 #1 stepi で停止 (SVC) | S:080200ABAC CMN r0,#1 wait continue interrupt SVC モード S:080200ABAC で実行停止で実行が停止しました S:080200ABAC CMN r0,#1 | ▶ 変数: 0 ▶ 変数: 0/0 ▶ 変数: 0/0 | | |
| ステータス: 接続 | マンド:コンテンツアシストを表示するには、Ctrl+スペー送信 | <u>変数の追加</u> | | |
| ▲ atlas_main.c ☆ | - | 3 111 逆飛い 22 毛メモリ 三 29か 日イベ 陸 アウ □ □ | | |
| <pre>20 * Copyright Altera 2013 7 8 #include <stdio.h> 9 #include "socal.h" 10 11 #define LED_BASE_ADDR (0xFF210040) 12 130 int main(int argc, char** argv) 14 { 15 int i; 16 printf("Hello from Atlas. \n"); 18 19 while(1) 20 { 21 for(i=0; i < 16; i++){ 22 alt_write_word(LED_BASE_ADDR,i); 23 printf("LED [%x] \n",i); 24 } 25 } 26 return 0; 28 } </stdio.h></pre> | | ・ マスの命令> ・ マスの命令> ・ マスの命令> ・ マスの命令> ・ ロの ・ マスの命令> ・ ロの ・ マスの命令> ・ ロの ・ マスの命令> ・ ロック ・ マスのの令 ・ マスのの令 ・ マスのの令 ・ ロック ・ マスのの令 ・ ロック ・ マスのの令 ・ ロック ・ マスのの令 ・ ロック ・ ロック ・ ロック ・ ロック ・ ロック ・ ロック ・ マスのの ・ ロック ・ ・ ロッ ・ ・ ・ | | |
| | | | | |

【図 4-18】 アプリケーションの実行/デバッグ

- ____8. 「ターゲットから切断」 🔌 ボタンをクリックして CPU との接続を切断し、「すべての接続の削除」 🙀 ボタンをクリックしてターゲットを削除します。
- ____9. メニュー・バーのショートカット・ボタン 屁 をクリックして元の C/C++ パースペクティブに切り替えます。

4-5-2. デバッグの構成(参考)

Atlas-Blinking-LED-Baremetal-GNU サンプル・プロジェクトに含まれる、デバッグ構成におけるその他の設定を以下に紹介します。

①「ファイル」タブ

ターゲット・ボードにダウンロードするアプリケーション実行ファイルを指定しています。

このサンプルでは、アプリケーション実行ファイルに atlas_main.axf を指定しています。

| デバッグ構成 | | |
|--|---|---|
| 構成の作成、管理、および実行 DS-5 デバッグセッションを開始するためのコンフィ | ギュレーションを作成、編集、または選択します。 | Ť |
| C/C++ アブリケーション | 名前(N): Atlas-Blinking-LED-Baremetal-Debug 接続 📓 ファイル 🏘 デバッガ 🎕 OS 認識機能 🗠 引数 📼 環境 🛃 エクスポート | |
| C/C++ アプリケーションへのアタッチ C/C++ ポストモーテム・デバッガー C/C++ リモート・アプリケーション | ターゲットコンフィギュレーション ダウンロードするホスト上のアプリケーション: | |
| ▲ 襟 DS-5デバッガ 登 Atlas-Blinking-LED-Baremetal-Debug ● ¹ IronPython Run | \${workspace_loc:/Atlas-Blinking-LED-Baremetal-GNU/atlas_main.axf} ファイルシステム ワークスペース 図シンボルをロードします | |

【図 4-19】 デバッグ構成ウィンドウの「ファイル」 タブ

②「デバッガ」タブ

デバッグの開始方法や、デバッガ・スクリプト・ファイルを指定しています。

このサンプルでは、main 関数からデバッグを開始し、デバッガ・スクリプト・ファイルとして debug-hosted.ds を指定しています。

| ● デバッグ構成 | | | |
|--|---|--|--|
| 構成の作成、管理、および実行 DS-5 デバッグセッションを開始するためのコンフィ | ギュレーションを作成、編集、または選択します。 | | |
| | 名前(N): Atlas-Blinking-LED-Baremetal-Debug ◆ 接続 → ファイル → デバッガ → OS 認識機能 → 引数 → 現境 → エクスポート 実行制御 ● 接続のみ ● エントリポイントからデバッグします ● シンボルからデバッグします main | | |
| クレンティング ひら-5デバッガ な Atlas-Blinking-LED-Baremetal-Debug の IronPvthon Run | | | |

【図 4-20】 デバッグ構成ウィンドウの「デバッガ」タブ



4-5-3. デバッガ・スクリプト・ファイルでの実行内容(参考)

DS-5 では、デバッガ・コマンドを含むデバッガ・スクリプト・ファイルを使用することにより、デバッグ操作を自動 化することができます。

デバッガ・スクリプト・ファイルでは、アプリケーションを実行する前に Preloader を実行して、HPS I/O ピンや SDRAM コントローラの初期化設定を行い、アプリケーションをロードして実行するような一連の操作がデバッガ・ コマンドで書かれています。

サンプル・アプリケーションにおけるデバッガ・スクリプト・ファイルでの実行内容の例を下図に示します。

() Note:

<u>ご利用のサンプル・アプリケーションにより、デバッガ・スクリプト・ファイルの記述内容が異なる場合があります。</u>



【図 4-21】 デバッガ・スクリプト・ファイルでの実行内容の例

DS-5 デバッガ・スクリプト・ファイルは、テキスト・ベースのファイルです。

DS-5 デバッガ・スクリプト・ファイルを作成する際の注意事項としては以下が挙げられます。

- ファイル拡張子には .ds を使用することになっています。
- 1 行につき 1 つのコマンドのみを含める必要があります。
- コマンドでは大文字と小文字が区別されません。
- 必要に応じて、# を文字の先頭に付けることでコメント文を入れることもできます。

一般的な、DS-5 デバッガ・スクリプト・ファイルでは、以下のようなデバッガ・コマンドが記述されています。



【図 4-22】 デバッガ・スクリプト・ファイル内のデバッグ・コマンド例(1)





[【]図 4-23】 デバッガ・スクリプト・ファイル内のデバッグ・コマンド例(2)

新 参考:

DS-5 でのデバッガ・スクリプト・ファイルおよびデバッガ・コマンドに関する更に詳しい情報は、以下のページが参考になります。

- 『DS-5 活用テクニック ~ デバッガ・コマンドの使い方』 <u>https://service.macnica.co.jp/library/129405</u>
- DS-5 デバッガ・ユーザガイド
 『<u>Arm DS-5 Debugger User Guide</u>』(最新英語版)
 『<u>ARM® DS-5 デバッガユーザガイド バージョン 5.26</u>』(日本語版)
- DS-5 デバッガ・コマンド・リファレンス
 『Arm DS-5 Debugger Command Reference』(最新英語版)
 『ARM® DS-5 デバッガコマンドリファレンス バージョン 5.26』(日本語版)

5. 新規にベアメタル・アプリケーションを作成して DS-5 で実行する方法

この章では、ベアメタル・アプリケーション・プロジェクトを新規に作成して DS-5 でデバッグ / 実行する方法に ついて説明します。

DS-5 が起動していない場合は、前出の「4-2. DS-5 の起動」の手順に従って DS-5 を起動しておきます。

5-1. ベアメタル・アプリケーションの新規作成

___1. DS-5 のメニューより「ファイル(F)」⇒「新規(N)」⇒「プロジェクト(R)...」を実行します。



【図 5-1】「ファイル(F)」⇒「新規(N)」⇒「プロジェクト(R)…」を実行

____2. 新規プロジェクト画面にて「C/C++」⇒「C Project」を選択して [次へ(N)] をクリックします。

| ● 新規プロジェクト | - • × |
|--|-------|
| ウィザードを選択 新規 C プロジェクトの作成 | |
| ウィザード(W): フィルタ入力 | |
| ▶ @ 一般 ● @ C/C++ ⓒ Cプロジェクト ⓒ CDT プロジェクト Makefile Project with Existing Code ▶ @ CVS ▶ @ Java ▶ @ PyDev | |
| (P) (P) キャ (P) (P) | ·ンセル |

【図 5-2】新規プロジェクト画面にて「C/C++」⇒「C Project」を選択

- ____3. プロジェクト名(P): には「HelloWorld」を指定します。
- ____4. プロジェクトの種類:には「実行可能」 「空のプロジェクト」を選択します。
- ____5. ツールチェイン: には「Altera Baremetal GCC」を選択します。
- ____6. [終了(F)]をクリックするとプロジェクトが生成されます。

| ●Cプロジェクト | | |
|---|--|-------|
| c プロジェクト 選択した種類の C プロジェクトを作成 | | |
| プロジェクト名(P): HelloWorld | | |
| ☑ デフォルト・ロケーションの使用(D) | | |
| ロケーション(L): C:¥Work¥DS-5 Workspaces¥HelloWorld | | 参照(R) |
| ファイル・システムを選択(Y): デフォルト ▼ プロジェクトの種類: ▶ | ツールチェイン: Altera Baremetal GCC Arm Compiler 5 (DS-5 built-in) Arm Compiler 6 (DS-5 built-in) Cygwin GCC GCC 4.x [arm-linux-gnueabihf] (DS-5 built-in) MinGW GCC | |
| | olchain を表示 > 終了(F) | キャンセル |

【図 5-3】HelloWorld プロジェクトの生成

____7. プロジェクト・エクスプローラ上の HelloWorld をハイライトさせた状態で、 DS-5 のメニューより「プロジェクト(P)」 「プロパティ(P)」を起動します。



Ver.18 / Rev. 3 2019 年 1 月

- ____8. プロパティ画面左側のツリーより「C/C++ ビルド」 「設定」を選択します。
- ____9. 設定画面の ツール設定 タブ上で「GCC C リンカ」 「イメージ」を選択し、リンカのスクリプト指定 欄の右側にある [参照(B)...] ボタンをクリックします。
- ____10. 次の場所に用意したリンカ・スクリプト・ファイルを選択し、 [開く(O)] をクリックします。 C:¥Temp¥cycloneV-dk-oc-ram-hosted.ld

() Note:

<u>本資料の説明では、cycloneV-dk-oc-ram-hosted.ld を C:¥Temp に格納したものとして説明しています。</u>

このリンカ・スクリプトでは、64KB の内部 RAM (On-Chip RAM) をターゲットに指定するとともに、セミホ スティングの利用をリンカへ指示します。

様々な領域を持ったリンカ・スクリプトのフォーマットを理解したい場合は、スクリプト・ファイルを開いて内 容を参照してみてください。

____11. [OK] ボタンをクリックし、プロパティ画面を閉じます。

| ● プロパティ: HelloWorld | | |
|---|--|--|
| フィルタ入力 | 設定 | ↓ ↓ ↓ ↓ |
| > リソース ▲ C/C++ ビルド Tool chain エディター ビルド変数 ロギング 環境 設定 > C/C++ 一般 ビルダー プロジェクト参照 実行/デバッグ設定 | 構成: [アパッウ" [アクティブ] ③ ツール設定 ♪ ビルド・ステップ ④ ビルド成果物 m バイナリー・パーサー ④ エラー・パーサー ▲ ③ GCC C コンパイラ ◎ ターゲット ◎ ブリプロセッサ ◎ ジメ [*] ト ◎ インクルード ◎ 最適化 | 構成の管理 参照(B) |
| | ※ デバッグ ※ 音 ※ GCC Pセンブラ ※ クーグット ※ クーグット ※ かかい ※ インクルード ※ デバッグ ※ 音 ※ GCC C リンカ ※ テイブラリ ※ ドキュメント ※ マク州 ※ マク州 ※ マク州 ※ コンピューター 、 OS (C:) 、 Temp ※ A (1) ※ マク州 ※ マク州 ※ マクリンカ ※ 音 ※ GCC C リンカ ※ 音 ※ GCC C (1) ※ コンピューター ※ 音 ※ 音 ※ 音 ※ 音 ※ 子の州 ※ 音 ※ 子の州 ※ 音 ※ 音 ※ 子の州 ※ 音 ※ 音 ※ 音 ※ 音 ※ 音 ※ 音 ※ 子の州 ※ 音 ※ 音 ※ 子の州 ※ 音 ※ 音<td>● ・ 4 Tempの検索 の 目目 ・ 1 ② 更新日時 種類 サイズ 2014/06/19 23:01 LD ファイル 9 KB</td> | ● ・ 4 Tempの検索 の 目目 ・ 1 ② 更新日時 種類 サイズ 2014/06/19 23:01 LD ファイル 9 KB |
| | ファイル名(N): cycloneV-dk-oc-ram-hosted.ld | |
| ? | | 0K キャンセル |

【図 5-5】設定画面の ツール設定 タブ



__12. DS-5 のメニューより「ファイル(F)」 「新規(N)」 「ソース・ファイル」を実行します。

____13. ソース・ファイル(E): にファイル名「HelloWorld.c」と入力し「終了(F)」をクリックします。

| • | ● DS-5 Workspaces - C/C++ - Eclipse プラットフォーム | | | | | |
|----|--|-----------------|-----|--------------------------------------|--|--|
| ファ | マイル(F) 編集(E) ソース(S) リファクタリ | リング(T) ナビゲート(N) |)検 | 索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H) | | |
| | 新規(N) | Alt+シフト+N♪ | | C++ プロジェクト | | |
| ~ | ファイルを開く(.) | | | | | |
| | open projects from File System | | | Makefile Project with Existing Code | | |
| | 閉じる(C) | Ctrl+W | | プロジェクト(R) | | |
| | すべて閉じる(L) | Ctrl+シフト+W | | | | |
| | 保管(S) | Ctrl+S | C++ | C/C++ フロシエクトに変換 (C/C++ ネーチャーを追加) | | |
| | 別名保存(A) | | | フォルダー | | |
| 0 | すべて保管(E) | Ctrl+シフト+S | C | ソース・ファイル | | |
| | □休官しに状態に戻9(Ⅰ) | | h | ヘッダー・ファイル | | |
| Ę | ● 新規ソース・ファイル □ ■ □ ■ □ ■ | | | | | |
| - | 新しいソース・ファイルを作成します。 | | | | | |
| | 9-X·JAIUS(D): Helloworld | | | ≶#R(B) | | |
| | ソース・ファイル(E): HelloWorld.d | | | | | |
| 1 | テンプレート(T): デフォルト C ソース・ラ | テンプレート | | ▼ 構成… | | |
| | | | | | | |
| | ? | | | 終了(F) キャンセル | | |

【図 5-6】「ファイル(F)」 「新規(N)」 「ソース・ファイル」を実行

_14. HelloWorld.c をダブルクリックして開き、以下のようにプログラムをタイプ入力します。





__auto_semihosting シンボルは、現在の実行可能イメージがセミホスティング・サービスを必要とすることを、 デバッガへ伝える効果を持ちます。このシンボルは、2 つのアンダースコアで始まります。

__15. 編集後の HelloWorld.c を保存します(Ctrl+S キー または「**ファイル(F)」 「保管(S)**」を実行)。

| • | S-5 Workspaces - C/C++ - HelloWo | orld/HelloWorld.c - Eclipse プラッ |
|------------|---|---------------------------------|
| ファ | ・イル(F) 編集(E) ソース(S) リフ: | ァクタリング(T) ナビゲート(N) |
| C , | 新規(N) ファイルを開く(.) Open Projects from File System | Alt+シフト+N▶ |
| | 閉じる(C) すべて閉じる(L) | Ctrl+W Ctrl+シフト+W |
| | 保管(S) | Ctrl+S |
| 6 | 別名保存(A) | |

【図 5-8】編集後の HelloWorld.c を保存



5-2. ベアメタル・アプリケーションのビルド

次に作成したベアメタル・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。

____1. HelloWorld プロジェクトをハイライトし、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 5-9】 HelloWorld ベアメタル・アプリケーション・プロジェクトのビルド

__2. コンソール・ビューにて、プロジェクトのビルドが完了した事を確認します。 ビルドが完了すると、「デバッグ」フォルダの下に HelloWorld.axf ファイルが生成されます。



【図 5-10】 ビルドの完了

5-3. ベアメタル・アプリケーションのデバッグ

- ____1. メニューの「実行(R)」 「デバッグの構成(B)...」を選択します。ここから、ターゲット・ボード上で Hello World アプリケーションを実行するための設定を行います。
- ____2. デバッグ構成画面左側のリストより「DS-5 デバッガ」を選択し、右クリック して現れるメニューから 「新規(W)」をクリックします。



【図 5-11】新規デバッグ構成の作成

- ____3. 名前(N): に「HelloWorldConfig」と指定します。
- ____4. 「接続」タブのターゲットに「Altera」 「Cyclone V SoC (Dual Core)」 「Bare Meta Debug」 「Debug Cortex-A9_0」を選択します。
- ____5. ターゲット接続に「USB-Blaster」を選択します。
- ____6. 接続の [参照...] ボタンをクリックし、接続ブラウザ画面にて「DE-SoC on localhost」を選択します。

| ● デバッグ構成 | | — × |
|---|---------------------------------|--|
| 構成の作成、管理、および実行 | | 1 |
| ⊗ [デバッガ]: シンボルからデバッグを行いました | が、 [ファイル] タブでシンボルファイルが定義されていません | ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ |
| ○ () () () () () () () () () () () () () | 名前(N): HelloWorldConfig ● 接続 | : ・ E 、 、 、 、 、 、 、 、 、 、 、 、 、 |
| フィルター一致: 19 / 19 項目 | 前回保管した状態に戻す(V) 適用(Y) | |
| ? | デバッグ(D) 閉じる | |

【図 5-12】新規デバッグ構成の「接続」タブの設定

- ____7. 「ファイル」タブを開きます。
- ____8. ダウンロードするホスト上のアプリケーション:の[**ワークスペース...**]ボタンをクリックし、「HelloWorld」 「デバッグ」 「HelloWorld.axf」を選択して、[OK]をクリックします。



【図 5-13】新規デバッグ構成の Files タブの設定

___9. [**デバッグ(D)**] ボタンをクリックすることで、ボードに対してアプリケーションのダウンロードを行い、デバッ グ・セッションを開始します。



【図 5-14】 デバッグ・セッションの開始

____10. DS-5 デバッグ・パースペクティブ・スイッチの確認 プロンプトが表示されたら [はい(Y)] をクリックします。 Windows ファイアウォールの警告が出た場合は、 [アクセスを許可する] をクリックします。

| ● パースペクティブスイッチの確認 | |
|---|--|
| この 起動 は、DS-5 デバッグ パースペクティブに関連付けられ | にています。 |
| このパースペクティブを開きますか? | |
| □ 常にこの設定を使用する(R) | |
| | (はい(Y) いいえ(N) |
| Windows セキュリティの重要な習告 | |
| このプログラムの機能のいくつかが Windows ファイアウォールでプロックさます | ☆ Windows セキュリティの重要な書告 このプログラムの機能のいくつかが Windows ファイアウォールでプロックされてい ます |
| 3/(10/10/99/ステアンジンアイティキアンラン、WHOUWS ソイア・オールより eclipseere Uver コウガブロック社ではます。 名前(1): Celipseere 発行元(ア): 不明 パス(H): C¥altera¥ 14.0¥embedded¥ds=5¥sw¥eclipse¥eclipsee | すべてのパブリックネットワークとプライベート ネットワークで、Windows ファイアウォールにより vstrm_serverd_rddiexe の機能の(いつかがフロックなれています。 を動いた 発行元(P) 不明 |
| eclipse.exe にこれらのネットワーク上での過剰含な許可する: 同 ブライベート ネットワーク (ホーム ネットワークや社内ネットワークなど)(R) | パス(H): CY4)ters#141(#mbeddef#ds=D¥sw¥debughw¥debug_server Yostm_serverd_rddiaxe にこれらのネットワーク上での過信を終っする。 |
| ☑ パブリック ネットワーク (空港、喫茶店など) (非推奨)(U) このようなネットワークは多くの場合、セキュリティが低いかセキュリティが設定されていません) | □ プライベート ネットワーク (ホーム ネットワークや社内ネットワークなど)(R) □ (プリック ネットワーク (空): 「柴菜(K)() (こんびぶよットワーズ(客): (小菜(加)()、(小菜(和)()、(小菜)) |
| フログラムにファイアウォールの経由を計画することの危険性の詳細 アクセス参計画する(A) | プログラムにファイアウォールの経由を許可することの危険性の詳細 |
| | アクセスを許可する(A) キャンセル |

【図 5-15】DS-5 デバッグ・パースペクティブ・スイッチの確認

この時点でアプリケーションは main()の先頭に停止します。

___11. 「続行」
▶ ボタンにてアプリケーションを実行します。

App Console にメッセージが表示されることが確認できるはずです。

| n App Console 🛛 🖬 夕 | ーゲットコンソール 🥺 エラー・ログ | i. 🕞 🔐 🕞 — 🗖 |
|-----------------------|-----------------------------|--------------|
| | 🔄 リンク済み: HelloWorldConfig 🔻 | |
| Hello World from SoC! | | |
| | | |
| | | |
| | | |
| | | |

【図 5-16】 アプリケーションの実行

- ____12. 「ターゲットから切断」 🗽 ボタンをクリックしてアプリケーションを切断し、「すべての接続の削除」 🙀 ボタンをクリックしてターゲットを削除します。
- ____13. メニュー・バーのショートカット・ボタン 🔂 をクリックして元の C/C++ パースペクティブに切り替えます。
- ____14. DS-5 を終了します(任意)。

6. FPGA レジスタの確認方法

DS-5 にペリフェラル記述ファイル (.svd)を追加して、レジスタ・ビューから FPGA / Soft IP レジスタを見るには、 以下の手順を行います。

- ____1. 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ある対象プロジェクトをハイライトし、右クリック して「デバッグ(D)」→「デバッグの構成(B)」を選択します(「4-5-1. デバッグの実行」を参照)。
- ____2. 「デバッグの構成」ウィンドウから、「ファイル」タブをクリックします。
- ____3. 「ディレクトリからペリフェラル記述ファイルを追加します」を選択します。
- ____4. [ファイルシステム]ボタンをクリックして、

 - () Note:

<u>本資料の説明では、C:¥Work に格納した</u> Atlas-SoC ボード用のハードウェア・デザインを使用している ため、下図のパスを指定しています。

Atlas-SoC ボードの場合は、

 $C: \verb"work" DE0-Nano-SoC_v.1.3.0_HW revD0_SystemCD" Demonstrations \verb"sSoC_FPGA" DE0_NANO_SOC_GHRD" system" Synthesis and the statement of the$

DE10-Nano ボードの場合は、

| ● デバッグ構成 | |
|---|--|
| 構成の作成、管理、および実行 DS-5 デバッグセッションを開始するためのコンフィ | ギュレーションを作成、編集、または選択します。 |
| | 名前(N): Atlas-Blinking-LED-Baremetal-Debug ④ 接続 「 協 ファイル」 参 デバッガ 「 ③ OS 認識機能」 (0)・引数 「 二 項境 」 ユ エクスポート |
| C (C++ アブリケーションへのアタッチ C (C++ ポストモーテム・デバッガー C (C++ ポストモーテム・デバッガー C (C++ リモート・アブリケーション ▲ DS-5デバッガ | ターゲットコンフィギュレーション ダウンロードする木スト上のアプリケーション: \${workspace_loc:/Atlas-Blinking-LED-Baremetal-GNU/atlas_main.axf} |
| | ファイルシステム···· ▽-クスペース··· ▽ シンボルをロードします ファイル |
| □ Java アプリケーション ◎ Java アプレット Ju JUnit a³ Jython run a³ Wython unittest | ディレクトリからペリフェラル記述ファイルを追加します C:¥Work¥DE0-Nano-SoC_v.1.3.0_HWrevD0_SystemCD¥Demonstrations¥SoC_FPGA¥DE0_NANO_SOC_GHRD¥soc_system¥synthesis ファイルシステム… ワークスペース… |





____5. [デバッグ] ボタンをクリックして、デバッガ・スクリプトの実行を開始します。

| | 名前(N): Atlas-Blinking-LED-Baremetal-Debug | |
|--|--|---|
| フィルタ入力 | 🐢 接続 🛛 ファイル 🏘 デバッガ 🚳 OS 認識機能 🔤 引数 🚾 環境 🛃 エクスポート | |
| ■ C/C++ アプリケーション | ターゲットの選択 | |
| C/C++ アブリケーションへのアタッチ C/C++ アブリケーションへのアタッチ C/C++ デブリケーションへのアタッチ C/C++ アブリケーションへのアタッチ C/C++ アブリケーションへのアクタッチ C/C++ アブリケーションへのアクタッチ C/C++ アブリケーションへのアクタッチ C/C++ アブリケーションへのアクター C/C++ アブリケーシー C/C++ P/C++ アブリケーシー C/C++ P/C++ P/C+++ P/C++ P/C+++ P/C++ P/C+++ P/C+++ P/C+++ P/C+++ P/C+++ P/C++++ P/C+++ P/C++++ P/C++++ P/C++++ P/C++++++++++ | 使用する製造元、ボード、プロジェクトのタイプ、およびデバッグ操作を選択します。 現在の選択内容: | |
| | Altera / Cyclone V SoC (Dual Core) / Bare Metal Debug / Debug Cortex-A9_0 | |
| ▲ 参 DS-5デバッガ | | 1 |
| Atlas-Blinking-LED-Baremetal-Debug | 0.05 × 100 × | |
| ₽ ^I IronPython Run | Artia 10 CoC | |
| e ^V IronPython unittest | Arria V SoC | |
| Java アプリケーション | Cyclone V SoC (Dual Core) | |
| International Java アプレット | Bare Metal Debug | |
| Ju Junit | Debug Cortex-A9_0 | |
| "Jython unittest | Debug Cortex-A9_1 | |
| PyDev Django | ターゲット接続 USB-Blaster ▼ | |
| 🔈 PyDev Google App Run | | - |
| er Python Run | DS-5 Debugger will connect to an Altera USB-Blaster to debug a bare metal application. | |
| Vython unittest | 接続 | |
| 宮, リモート Java アノリクーション ▶ 記動グループ | | |
| | Bare Metal Debug Connection DE-SoC on localhost [USB-1]:DE-SoC USB-1 | |
| | DTSL オプション 編集 USB-Blaster トレースまたはその他のターゲット オプションを構成します。 | " |
| | | |
| | | |
| | 前回保管した状態に戻す(V) 適用(Y) | |

【図 6-2】 [デバッグ] ボタンをクリック

____6. デバッグ パースペクティブへの切り替えのプロンプトが表示されたら [はい] を選択します。 また Windows ファイアウォールの警告が出た場合は、[アクセスを許可する] をクリックします。

| ● パースペクティブスイッチの確認 | |
|---|---|
| この 起動 は、DS-5 デバッグ パースペクティブに関連付けられ | ています。 |
| このパースペクティブを開きますか? | |
| 常にこの設定を使用する(R) | |
| | (はい(Y) いいえ(N) |
| Windows セキュリティの重要な警告 | |
| このプログラムの機能のいくつかが Windows ファイアウォールでブロックさ | Windows セキュリティの重要な警告 |
| 👽 इन | このプログラムの機能のいくつかが Windows ファイアウォールでブロックされてい |
| すべてのパブリック ネットワークとプライベート ネットワークで、Windows ファイアウォールにより eclipse.exe の機能 | |
| つかがブロックされています。 名前(N): eclipse.exe | すべてのパブリック ネットワークとプライベート ネットワークで、Windows ファイアウォール(ことり vstrm_serverd_rddiexe |
| 100 発行元(P): 不明 | の機能のいべつがかりロックされています。 名前(N): vstrm_serverd_rddiexe |
| パス(H): C:¥altera¥14.0¥embedded¥ds=5¥sw¥eclipse¥eclipse.exe | ■== 発行元(P): 不明 |
| eclipselexe にこれらのネットワーク上での通信を許可する: | パス(H): C¥altera¥14.0¥embedded¥ds-5¥sw¥debughw¥debug_server ¥vstrm_serverd_rddi.exe |
| 🥅 プライベート ネットワーク (ホーム ネットワークや社内ネットワークなど)(R) | vstrm_serverd_rddiexe (これらのネットワーク上での通信を許可する: |
| 図パブリックネットワーク(空港、喫茶店など)(非推奨)(U) | □ ブライベート ネットワーク (ホーム ネットワークや社内ネットワークなど)(R) |
| (このようなネットノークは多くの場合、セキュリティか良いがセキュリティが設定されていません) | 図 パブリックネットワーク(空港、喫茶店など)(非推奨)(U) (このようなネットワーク)(空港、喫茶店など)(非推奨)(U) |
| プログラムにファイアウォールの経由を許可することの危険性の詳細 | (CANANALALA, NG SUMMER, CUTVANISM ALCUTVANISMACTICA SECTION |
| アクセスを許可する(A) | プログラムにファイアウォールの経由を許可することの危険性の詳細 |
| | アクセスを許可する(A) キャンセル |

【図 6-3】 デバッグ パースペクティブへの切り替えとファイアウォールのアクセス許可



7. レジスタ・ビューを選択して FPGA と HPS のペリフェラル・レジスタを表示します。

| (X)= 変数 鸟 ブレーク: | ポイント 🔟 レジスタ 🛙 | ^{X+y} 式 | f() 関数 |
|-----------------|---------------------|------------------|--------------|
| | 🔄 リンク済み: Atlas-Blir | nking-LE | D-Baremetal- |
| レジスター セット: す | べてのレジスター | | |
| 名前 | 値 | サイス* | アクセス |
| 🖽 🗁 Core | 47/47 レジスタ | | |
| 🕀 🗁 CP15 | 141/141 レジスタ | | |
| 🕀 🗁 VFP | 69/69 レジスタ | | |
| 🕀 🗁 SIMD | 16/16 レジスタ | | |
| 🗄 🗁 Peripherals | 2,767/2,767 レジスタ | | |

【図 6-4】 レジスタ・ビューを選択

__8. 「+」記号をクリックして Core レジスタを展開します。コア・レジスタがすべて表示されて編集できるように なります。

| ⊫変数 🔒 ブレーク | ポイント 🚥 レジスタ 🛙 | (※)= 変数 🔏 ブレー | クポイント 竝 レジスタ 🛙 | ^{X+V} 式 | ()関数 |
|-----------------|---------------------|-----------------|--------------------|------------------|------------|
| | 🔄 リンク済み: Atlas-Blin | | ⑤ リンク済み: Atlas-Bli | nking-LED |)-Baremeta |
| ジスターセット: す | すべてのレジスター | レジスター セット: | すべてのレジスター | | |
| 名前 | 値 | 名前 | 値 | ታ ፈን | アクセス |
| 🔁 Core 🗕 🚽 | 47/47 1 277 5 | 🗕 🕞 🗁 Core | 47/47 レジスタ | | |
| - 🗁 CP15 | 141/141 レジスタ | - 🛛 R0 | 0x00000001 | 32 | R/W |
| E 🗁 VFP | 69/69 レジスタ | – 🛛 R1 | 0x0200BEF0 | 32 | R/W |
| | 16/16 レジスタ | - 🛛 R2 | 0x00000000 | 32 | R/W |
| - 🗁 Peripherals | 2,767/2,767 レジスタ | - 🛛 R3 | 0x0200BED4 | 32 | R/W |
| | | - • R4 | 0x00000000 | 32 | R/W |
| | | - • R5 | 0x00000000 | 32 | R/W |
| | | - • R6 | 0x0000BDC8 | 32 | R/W |
| | | - • R7 | 0x0200B404 | 32 | R/W |
| | | - • R8 | ØxFFFFFFF | 32 | R/W |
| | | - • R9 | 0x00000005 | 32 | R/W |
| | | - • R10 | 0x00000000 | 32 | R/W |
| | | - • R11 | 0x00000000 | 32 | R/W |
| | | - • R12 | 0x00000000 | 32 | R/W |
| | | - SP | 0x0201BDB8 | 32 | R/W |
| | | – ● LR | 0x0200A890 | 32 | R/W |
| | | - PC | 0x02000234 | 32 | R/W |
| | | E S CPSR | 0x200001D3 | 32 | R/W |
| | | 🕀 🗁 IRQ | 3/3 レジスタ | | |
| | | 🕀 🗁 FIQ | 8/8 レジスタ | | |
| | | 🕀 🗁 UND | 3/3 レジスタ | | |
| | | 🕀 🗁 ABT | 3/3 レジスタ | | |
| | | B 🔁 SVC | 3/3 レジスタ | | |
| | | 🕀 🗁 USR | 7/7 レジスタ | | |
| | | 🕀 🗁 MON | 3/3 レジスタ | | |
| | | 🕀 🗁 CP15 | 141/141 レジスタ | | |
| | | DE VEP | 69/69 レジスタ | | |
| | | B 🗁 SIMD | 16/16 レジスタ | | |
| | | E > Peripherals | 2,767/2,767レジスタ | | |

【図 6-5】 Core レジスタを展開

____9. Peripherals レジスタ・グループを展開してリストの最後までスクロールします。 altera_avalon_ という接頭辞が付いたものが FPGA 内のソフト IP レジスタです。

「+」記号をクリックしてレジスタを展開することで詳細を確認することができます。

| 🕪= 変数 🎭 ブレークポイント 🚥 レジスタ 🛿 💥 式 f() 関数 | 🖃 🛷 0× 🔗 |
|---|--------------|
| 💲 リンク済み: Atlas-Blinking-LED-Baremet | tal-Debug 🔻 |
| レジスター セット: すべてのレジスター | |
| 名前 | 値 |
| 🕀 🗁 uart1 | 29/29 レジスタ |
| 🕀 🗁 i2c0 | 41/41 レジスタ |
| 🕀 🥦 i2c1 | 41/41 レジスタ |
| 🕀 🗁 i2c2 | 41/41 レジスタ |
| 🕀 🧀 i2c3 | 41/41 レジスタ |
| 🖶 sptimer0 | 9/9 レジスタ |
| 🖶 🗁 sptimer1 | 9/9 レジスタ |
| 🕀 🗁 sdr | 37/37 レジスタ |
| 🖶 🧁 osc1timer0 | 9/9 レジスタ |
| 🖶 🗁 osc1timer1 | 9/9 レジスタ |
| 🖶 🧀 l4wd0 | 13/13 レジスタ |
| 🖶 🗁 l4wd1 | 13/13 レジスタ |
| 🖻 🗁 clkmgr | 41/41 レジスタ |
| 🖶 rstmgr | 8/8 レジスタ |
| 🕀 🗁 sysmgr | 253/253 レジスタ |
| 🖶 🗁 dmanonsecure | 1/1 レジスタ |
| 🕀 🗁 dmasecure | 1/1 レジスタ |
| 🕀 🗁 spis0 | 21/21 レジスタ |
| 🖶 🦻 spis1 | 21/21 レジスタ |
| 🕀 🗁 spim0 | 25/25 レジスタ |
| 🕀 🗁 spim1 | 25/25 レジスタ |
| 🕀 🗁 scanmgr | 6/6 レジスタ |
| 🕀 🗁 mpuscu | 1/1 レジスタ |
| 🕀 🗁 mpul2 | 1/1 レジスタ |
| 🖶 🗁 altera_avalon_jtag_uart_jtag_uart_avalon_jtag_slave | 2/2 レジスタ |
| 🖶 🗁 altera_avalon_sysid_sysid_qsys_control_slave | 2/2 レジスタ |
| 🖶 🗁 altera_avalon_pio_led_pio_s1 | 6/6 レジスタ |
| 🖶 🗁 altera_avalon_pio_dipsw_pio_s1 | 6/6 レジスタ |
| 🗄 🗁 altera_avalon_pio_button_pio_s1 | 6/6 レジスタ |

【図 6-6】 FPGA 内のソフト IP レジスタ

1 注記:

SoC FPGA において、HPS と FPGA 間のバスは、以下の 3 つのブリッジから構成されています。

- ① FPGA-to-HPS ブリッジ: FPGA が HPS 内のスレーブに対してトランザクションを発行可能にするバス
- ② HPS-to-FPGA ブリッジ: HPS が FPGA 内のスレーブに対してトランザクションを発行可能にするバス
- ③ Lightweight HPS-to-FPGA ブリッジ: 通常、FPGA 内のソフト IP の control および status レジスタ (CSR) アクセス用に使用されます

これらのブリッジを初期化して開通することで、HPS と FPGA 間のアクセスが可能となります。

これらのブリッジを初期化していない状態で、レジスタ・ビューから FPGA 内のソフト IP レジスタを開かないでください。開いた場合、Eclipse の接続が不安定になることがあります。

7. カスタム・ボードへの対応方法

この章では、ユーザのカスタム・ボードでベアメタル・アプリケーションを動かすために必要な手順について説 明します。

7-1. Arm プロセッサを含むハードウェアの設計を行う

本資料では SoC FPGA 向けハードウェア設計に関する詳細な説明はしませんが、概要としては以下のような 手順となります。

- ____1. Quartus Prime および Platform Designer を使用し、Arm プロセッサを含むユーザのボードのハードウェアの設計を行います。
- ____2. Hard Processor System(以下、HPS)コンポーネントを Platform Designer システムへ追加しますが、HPS の設定には以下のような多くの重要な設定が含まれていますので、ユーザのボードに合わせた正しい設 定を行ってください。
 - AXI ブリッジの有効化
 - HPS に含まれるペリフェラルの選択と有効化
 - HPS クロック設定
 - SDRAM パラメータの設定
- _____3. Platform Designer システムが完成したら Generate して生成します。
- ____4. Quartus Prime で、ピン・アサインメントの設定とプロジェクトのコンパイルを行います。
 - 7-2. Preloader (プリローダ) 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。

7-3. Preloader の生成手順

以降に Preloader の生成手順を説明します。Preloader の生成方法についての更に詳しい説明は、本資料を 入手したサイト内から以下の資料をご覧ください。

新考:

『Preloader Generator の使用方法』 https://service.macnica.co.jp/library/117865

7-3-1. Embedded Command Shell の起動

「4-2-1. 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 (embedded フォルダ) 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。

7-3-2. 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** しとコマンド入力して、 bsp-editor (Preloader Generator)の GUI を起動します。

| | | - • × |
|-----------------------|--|--------|
| Intel FPGA Embedded C | ommand Shell | * E |
| Version 18.0 [Build 6 | 14] | |
| 11149@HD11149A ~ | | |
| \$ bsp-editor 🗕 ┥ | Embedded Command Shell から "bsp-editor" とコマンド入力 | |

【図 7-1】bsp-editor(Preloader Generator)の起動

7-3-3. 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 SoC | 記: EDS v15.0 より | 前 のバー | ·ジョンでは、「File」 「New BSP」を選択します。 |
|------------|--------------------|----------|---|
| - | BSP Editor | | |
| File | Edit Tools Help | 01 L N | 1 |
| | New Nios II BSP | , Ctrl+N | Linker Script Enable File Generation Target BSP Directory |
| | New HPS BSP | Ctrl+H | 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 |
| | Open | Ctrl+0 | |
| | Save | Ctrl+S | Version: |
| | Save As | | |
| | Exit | Ctrl+X | |





7-3-4. 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 _--- を押してフォルダを指定します。

() Note:

<u>本資料の説明では、C:¥Work に格納した Atlas-SoC ボード用のハードウェア・デザインを使用している</u> ため、下図のパスを設定しています。

<u>Atlas-SoC ボードの場合は</u>、

C:\U00e4Work\U00e4De0-Nano-SoC_v.1.3.0_HWrevD0_SystemCD\U00e4Demonstrations\U00e4SoC_FPGA\U00e4De0_NANO_SOC_GHRD\u00e4hps_isw_handoff\u00e4soc_system_hps_0

<u>DE10-Nano ボードの場合は、</u>

 $C: \label{eq:c:work} Work \eqref{eq:c:work} Demonstrations \eqref{eq:systemCD} Soc_FPGA \eqref{eq:c:work} Demonstrations \eqref{eq:system_hps_0} Soc_GHRD \eqref{eq:system_hps_0} \eqref{eq:system_h$

__2. 全ての指定が終わったら [OK] をクリックします。



【図 7-3】 ハンドオフ・ファイルの指定



7-3-5. Preloader のユーザ・オプション(Common)の設定

Common では Preloader に関する基本的な設定を行います。

(1) spl:

- PRELOADER_TGZ:
 Preloader ソース・ファイルのアーカイブ・ファイルを指定します。基本的に変更する必要はありません。
- CROSS_COMPILE: 使用するクロス・コンパイラを指定します。基本的に変更する必要はありません。

(2) **boot**:

- BOOT_FROM_QSPI:
 Preloader に続くブートイメージを QSPI 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 SDMMC 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。FPGA 側に実装し たメモリからのブートにはこの設定を利用します。
- QSPI_NEXT_BOOT_IMAGE: BOOT_FROM_QSPI チェック時に、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。
- SDMMC_NEXT_BOOT_IMAGE:
 BOOT_FROM_SDMMC チェック時、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。

BOOT メモリ選択は、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)。

| 📩 BSP Editor - C:¥Work¥DE0-Nano-SoC_v.1.3.0_HWrevD0_SystemCD¥Demonstrations¥SoC_FPGA¥DE0_NANO_SOC_GHRD¥software¥spl_bsp¥settings.bsp 📃 🔳 💌 | | | |
|--|---|---|--|
| File Edit Tools Help | | | |
| Main Software Packages Drivers Linker Script Enable | File Generation Target BSP Directory | | |
| SOPC Information file: CPU name: Operating system: U-Boot SPL Preloader (Cyclone 1 BSP target directory: .\ | //Arria Version: default 🔹 | | |
| Settinge PRELOADER_TGZ -CROSS_COMPILE -CROSS_COMPILE -BOOT_FROM_QSPI -BOOT_FROM_SDMMC -BOOT_FROM_NAND -BOOT_FROM_NAND -BOOT_FROM_NAND -BOOT_FROM_SDMMC -BOOT_FROM_SDMMC -BOOT_FROM_SDMMC -BOOT_FROM_SDMMC -BOOT_FROM_SDMMC -BOOT_FROM_SDMMC -BOOT_FROM_SDMMC -BOOT_FROM_SDMMC -SDMMC_NEXT_BOOT_IMAGE -NAND_NEXT_BOOT_IMAGE -FAT_BOOT_PARTITION -FAT_BOOT_P | SpI PRELOADER_TGZ: CROSS_COMPILE: SpI.boot BOOT_FROM_QSPI BOOT_FROM_SDMMC BOOT_FROM_NAND BOOT_FROM_NAND BOOT_FROM_RAM QSPI_NEXT_BOOT_IMAGE: SDMMC_NEXT_BOOT_IMAGE: NAND_NEXT_BOOT_IMAGE: FAT_SUPPORT FAT_BOOT_PARTITION: FAT_LOAD_PAYLOAD_NAME: | preloader/uboot-socfpga.tar.gz arm-altera-eabi- 0x60000 0x40000 0xc0000 1 1 u-boot.img | |

【図 7-4】Preloader のユーザ・オプション(Common)の設定



7-3-6. Preloader のユーザ・オプション(Advanced spl boot)の設定

Advanced spl boot ではウォッチドッグ・タイマの Disable などブート時の挙動に関して設定を行います。ベアメタル・アプリケーションを使用する場合は、WATCHDOG_ENABLE のチェックを外します。

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

| 📩 BSP Editor - C:¥Work¥DE0-Nano-SoC_v.1.3.0_HWrevD0_SystemCD¥Demonstrations¥SoC_FPGA¥DE0_NANO_SOC_GHRD¥software¥spl_bsp¥settings.bsp 📃 🖃 💌 | | | |
|--|-------------------------------------|--------------------------|--|
| File Edit Tools Help | | | |
| Main Software Packages Drivers Linker Script Enable P | ile Generation Target BSP Directory | | |
| SOPC Information file: | | | |
| CPU name: | | | |
| Operating system: U-Boot SPL Preloader (Cyclone V/ | Arria Version: default 👻 | | |
| BSP target directory: .\ | | | |
| □-Settings | spl.boot | | |
| ⊟spl | WATCHDOG_ENABLE | リオッチトック・タイマを使用しない場合は、 | |
| PRELOADER_TGZ | CHECKSUM_NEXT_IMAGE | WATCHDOG_ENABLE のチェックを外す | |
| CROSS_COMPILE | EXE_ON_FPGA | | |
| BOOT_FROM_QSPI | FPGA_MAX_SIZE: | 0x10000 | |
| BOOT_FROM_SDMMC | | 0.##0000 | |
| BOOT_FROM_RAM | In or <u>ebrancehoe</u> . | | |
| QSPI_NEXT_BOOT_IMAGE | FPGA_DATA_MAX_SIZE: | 0x10000 | |
| | STATE_REG_ENABLE | | |
| FAT_SUPPORT | BOOTROM HANDSHAKE CFGIO | | |
| EAT LOAD PAYLOAD NAME | WARMEST SKIP CEGIO | | |
| □ Advanced | | | |
| ⇒ spl | SDRAM SCRUB BOOT REGION START: | 0×1000000 | |
| reset_handshake | | | |
| | SDRAM_SCRUB_BOOT_REGION_END: | 0x2000000 | |
| | SDRAM_SCRUB_REMAIN_REGION | | |
| | RAMBOOT_PLLRESET | | |
| J <u>I</u> | - | | |

【図 7-5】Preloader のユーザ・オプション(Advanced spl boot)の設定



7-3-7. Preloader のユーザ・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。DS-5 Intel[®] SoC FPGA Edition のセミホスティング機能を使用する場合は、SEMIHOSTING のチェック・ボック スを ON にします。

1 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。

| 🖮 BSP Editor - C:¥Work¥DE0-Nano-SoC_v.1.3.0_HWrevD0_SystemCD¥Demonstrations¥SoC_FPGA¥DE0_NANO_SOC_GHRD¥software¥spl_bsp¥settings.bsp 💿 💽 | | | |
|--|--|--|--|
| File Edit Tools Help | | | |
| Main Software Packages Drivers Linker Script Enable File Generation Tar | get BSP Directory | | |
| SOPC Information file: CPU name: Operating system: U-Boot SPL Preloader (Cyclone V/Arria Version BSP target directory: .\ | i: default 🗸 | | |
| Spl.debug ⇒Settings ⇒Spl →PRELOADER_TGZ →CROSS_COMPILE ⇒boot →BOOT_FROM_QSPI →BOOT_FROM_SDMMC →BOOT_FROM_NAND →BOOT_FROM_NAND →BOOT_FROM_NAND →BOOT_FROM_RAM → QSPI_NEXT_BOOT_IMAGE →ANND_NEXT_BOOT_IMAGE →FAT_SUPPORT →FAT_BOOT_PARTITION EAT_LOAD_PAYLOAD_NAME → hoot → reset_assert → warm_reset_handshake → hoot → reformance | JG_MEMORY_WRITE MEMORY_ADDR: 0xffffd00 MEMORY_SIZE: 0x200 HOSTING WARE_DIAGNOSTIC _SDRAM DS-5 のセミホスティング機能を使用する場合は、 SEMIHOSTING のチェック・ボックスを ON にする | | |

【図 7-6】 Preloader のユーザ・オプション(Advanced spl debug)の設定

7-3-8. 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、「7-3-4. 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。

| 🕆 BSP Editor - C:¥Work¥DE0-Nano-SoC_v.1.3.0_HWrevD0_SystemCD¥Demonstrations¥SoC_FPGA¥DE0_NANO_SOC_GHRD¥software¥spl_bsp¥settings.bsp 💿 💽 | | |
|---|--|--|
| File Edit Tools Help | | |
| Main Software Packages Drivers Linker Script Enable. | File Generation Target BSP Directory | |
| SOPC Information file: CPU name: Operating system: U-Boot SPL Preloader (Cyclone V BSP target directory: .\ | /Arria Version: default 👻 | |
| -Settings -Common -spl -pRELOADER_TGZ -CROSS_COMPILE -boot -BOOT_FROM_QSPI -BOOT_FROM_SDMMC -BOOT_FROM_NAND -BOOT_FROM_NAND -BOOT_FROM_NAND -BOOT_FROM_NAND -QSPL_NEXT_BOOT_IMAGE -SDMMC_NEXT_BOOT_IMAGE -FAT_SUPPORT -FAT_BOOT_PARTITION -FAT_LOAD_PAYLOAD_NAME -Advanced -spl -reset_assert -spl -spl -spl -spl -spl -spl -spl -spl | spl.debug DEBUG_MEMORY_WRITE DEBUG_MEMORY_ADDR: 0xffffd00 DEBUG_MEMORY_SIZE: 0x200 Image: SemiHosting HARDWARE_DIAGNOSTIC Image: SkiP_SDRAM SkiP_SDRAM | |
| Information Problems Processing Information Problems Proces | | |
| Searching for BSP components with category: software_package_element Added operating system component "spl:1.0". Generated file "C:\Work\DE0-Nano-SoC_v.1.3.0_HWrevD0_SystemCD\Demonstrations\SoC_FPGA\DE0_NANO_SOC_GHRD\software\spl_bsp\settings.bsp" | | |
| | Generate | |

【図 7-7】 bsp プロジェクトの生成

7-3-9. 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:

<u>本資料の説明では、C:¥Work に格納した Atlas-SoC ボード用のハードウェア・デザインを使用している</u> ため、下図のディレクトリに移動しています。

<u>Atlas-SoC ボードの場合は</u>、

C:\U00e4Work\U00e4DE0-Nano-SoC_v.1.3.0_HWrevD0_SystemCD\U00e4Demonstrations\U00e4SoC_FPGA\U00e4DE0_NANO_SOC_GHRD\u00e4software\u00e4spl_bsp

<u>DE10-Nano ボードの場合は、</u>



【図 7-8】 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。

IS J 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 用のヘッダ情報を付加したバイナリ・ファイルで、SD カードや QSPI フラッシュ・メモリへ書き込むファイルとなります。

| 11149@HD11149A /cygdrive/c/Work/DEO-Nano-SoC_v.1.3.0_HWrevDO_SystemCD/Demonstrat |
|--|
| IONS/SOU_FPGA/DEU_NANU_SUU_GHRD/s I/cygdrive/c/Work/DE0-Nano-SoC_v.1.3.0_HWrevD0_SystemCD/Demonstrations/SoC_FPGA/DE0_NAN |
| arm-altera-eabi-objcopygap-fill=0xff -0 binary /cygdrive/c/Work/DE0-Nano-SoC |
| "make all" コマンドを実行し Preloader を生成する 「winework」 and software/spl u-boot-spl /cygdrive/c/Work/DED-Nano-SoC v.1.3.0 HWrevDD_S "make all" コマンドを実行し Preloader を生成する U-boot-spl /cygdrive/c/Work/DED-Nano-SoC v.1.3.0 HWrevDD_S t-socfpga/spl/u-boot-spl /cygdrive/c/Work/DED-Nano-SoC v.1.3.0 HWrevDD_S t-socfpga/spl/u-boot-spl /cygdrive/c/Work/DED-Nano-SoC v.1.3.0 HWrevDD_S t-socfpga/spl/u-boot-spl /cygdrive/c/Work/DED-Nano-SoC v.1.3.0 HWrevDD_S |
| make[2]: ディレクトリ '/cygdrive/c/Work/DEO-Nano-SoC_v.1.3.0_HWrevDO_SystemCD/De monstrations/SoC_FPGA/DEO_NANO_SOC_GHRD/software/spl_bsp/uboot-socfpga/spl' から 出ます |
| make[1]: ディレクトリ '/cygdrive/c/Work/DEO-Nano-SoC_v.1.3.0_HWrevDO_SystemCD/De monstrations/SoC_FPGA/DEO_NANO_SOC_GHRD/software/spl_bsp/uboot-socfpga' から出ま す |
| mkpimageheader-version 0 -o preloader-mkpimage.bin uboot-socfpga/spl/u-boot-s pl.bin uboot-socfpga/spl/u-boot-spl.bin uboot-socfpga/spl/u-boot-spl.bin uboot-s ocfpga/spl/u-boot-spl.bin |
| 11149@HD11149A /cygdrive/c/Work/DEO-Nano-SoC_v.1.3.0_HWrevDO_SystemCD/Demonstrat ions/SoC_FPGA/DEO_NANO_SOC_GHRD/software/sp1_bsp \$ 1s |
| generated preloader.ds settings.bsp uboot-socfpga Makefile preloader-mkpimage.bin uboot.ds |
| 111490HD11149A /cygdrive/c/Work/DEO-Nano-SoC_v.1.3.0_HWrevDO_SystemCD/Demonstrat ions/SoC_FPGA/DEO_NANO_SOC_GHRD/software/sp1_bsp \$ |
| |

【図 7-9】 "make all" コマンドを実行

1 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 <u>https://www.intel.com/content/altera-www/global/en_us/index/support/support-resources/knowledgebase/embedded/2018/unable-to-make-preloader-in-windows-10.html</u>

MACNICA フォーラム
 https://forum.macnica.co.jp/t/topic/1191/11



7-4. カスタム・ボード向けに生成した Preloader を DS-5 デバッグで使用する方法

ユーザのカスタム・ボードでベアメタル・アプリケーションを動かすために必要な手順としては、主に以下の 2 つがあります。

(1) Preloader ファイルの差し替え

カスタム・ボード向け Preloader を生成し、生成したバイナリでサンプル・プロジェクトの Preloader ファ イルを差し替えます。差し替え対象のファイルは **u-boot-spl.axf** です。

(2) Makefile の編集

Makefile 内にデフォルトの Preloader に関係する処理がある場合は、Makefile から削除します。

- 7-4-1. Preloader ファイルの差し替え
 - DS-5 でプログラムを実行/デバッグするには、Arm Executable and Linkable Format (ELF) ファイルをロードします。
 この形式は、Arm ELF 仕様書で説明されており、.axf というファイル拡張子を使用します。

この形式は、AFMELF 江稼音で読明されており、dxl というアイル拡張士を使用しよ

(2) 「7-3. Preloader の生成手順」で Preloader を生成しましたが、このとき、

<Quartus プロジェクト>¥software¥spl_bsp¥uboot-socfpga¥spl フォルダの下に u-boot-spl ファイルが生成されます。このファイルが Preloader の ELF ファイルとなります。



【図 7-10】 生成された u-boot-spl ファイル

(3) カスタム・ボード向けに生成した Preloader の実行可能バイナリ・ファイル u-boot-spl をコピーして、その ファイル名を .axf 拡張子をつけて u-boot-spl.axf とリネームします。

そのファイルを例えば Atlas-Blinking-LED-Baremetal-GNU ベアメタル・サンプル・プロジェクト内にある既存の u-boot-spl.axf ファイルと入れ替えれば、ユーザのカスタム・ボード向けの Preloader を使用して DS-5 で実行 / デバッグを行うことが可能です。



【図 7-11】 u-boot-spl ファイルをコピーして u-boot-spl.axf とリネームし差し換える

7-4-2. Makefile の編集

必要に応じて Makefile を編集して Preloader に関係する部分をコメント・アウトおよび編集します。

1 Note:

ご利用のサンプル・アプリケーションにより、Makefileの記述内容が異なる場合があります。

- (1) SoC EDS に格納される Cyclone® V SoC 開発キットの Preloader をコピーしないように変更します。
- (2) make clean を実行した際にコピーしたカスタム・ボード向け Preloader 実行可能バイナリ u-boot-spl.axf が削除されないように変更します。



【図 7-12】 Makefile の編集例

8. <u>ベアメタル・アプリケーションを SD カードからスタンドアローン実行する例</u>

この章では、ベアメタル・アプリケーションを SD カード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。

8-1. SD カードの準備

SD カードを作成し、SD カード内の Preloader を生成したものに入れ替えます。

1: 注記 1:

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、bsp-editor (Preloader Generator) にて "SEMIHOSTING"のチェック・ボックスを必ず OFF にして生成した Preloader を使用してください。

1 注記 2:

ベアメタル・アプリケーションを使用する場合は、bsp-editor (Preloader Generator) にて "WATCHDOG_ENABLE"のチェック・ボックスを OFF にして生成した Preloader を使用してください。 また、ベアメタル・アプリケーションでウォッチドッグ・タイマを使用する場合は、ウォッチドッグ・タイマが 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。

Cyclone[®] V SoC FPGA 評価キット(インテル純正ボード、3rd ベンダ製ボード)では、通常 Linux を起動させる ための SD カード・イメージ・ファイルが提供されています。

インテルの Cyclone[®] V SoC FPGA Development Board の SD カード・イメージ・ファイルは、 以下の RocketBoards.org のページで公開されています。 ダウンロードして解凍すると、SD カード・イメージ・ファイル (.img) があります。 https://releases.rocketboards.org/release/2017.10/gsrd/bin/linux-socfpga-gsrd-17.1std-cv.tar.gz

https://releases.rocketboards.org/release/2018.10/gsrd/cv_gsrd/sdimage.tar.gz

(1) SD カードの場合、Preloader は Windows のファイル・システムではアクセス不可能な領域に格納されています。

従って、まずは RocketBoards.org に公開されている SD カード・イメージを microSD カードに書き込ん でください。これにより、microSD カード内に Preloader を書き込むためのパーティションが作成されます。

SD カード・イメージ・ファイルを microSD カードに書き込むには、フリーソフト「Win32 Disk Imager」など をご利用ください。

Win32 Disk Imager

https://sourceforge.net/projects/win32diskimager/

SD カード・イメージの書き込み方法については、下記リンクのビデオを参照ください。 https://www.youtube.com/watch?v=0rckwOGTH5U&t=3s

- (2) Preloader バイナリ・ファイルの書き込み方法は、SoC EDS ツールのバージョンにより異なります。
 - SoC EDS v13.1 以前のバージョン 基本的に Linux の dd ユーティリティを使用します (Linux OS がインストールされた PC が必要です)。
 - SoC EDS v14.0 以降のバージョン インテルが提供する alt-boot-disk-util というツールを使用して書き込むことができます。

8-1-1. Soc EDS v13.1 以前のバージョン で Preloader を SD カードに書き込むには

- Soc EDS v13.1 以前では Soc EDS 自体に Preloader バイナリを書き込むためのツールが無いため、基本 的に Linux の dd ユーティリティ・コマンドを使用します。そのため Linux OS がインストールされた PC が必要です(仮想マシンでも可能)。
- (2) Preloader を部分的にアップデートするには、Linux PC に SD カードをマウントし dd コマンドを実行します。書き込む Preloader のバイナリ・ファイルは「7-3-9. Preloader のビルド」の手順で生成したヘッダ 情報が付加された preloader-mkpimage.bin です。
- ___1. Linux PC に SD カードを接続すると、この例では sdb が見えるようになります (PC の SD カード・スロット で見えない場合は、USB カード・リーダ経由で試してください)。

\$ sudo cat /proc/partitions 4

| 【 Ubuntu での表示例 】 | | | |
|------------------|----|-----------|------|
| 11 | 0 | 1048575 | sr0 |
| 8 | 0 | 100663296 | sda |
| 8 | 1 | 96468992 | sda1 |
| 8 | 2 | 1 | sda2 |
| 8 | 5 | 4191232 | sda5 |
| 8 | 16 | 7822336 | sdb |
| 8 | 17 | 512000 | sdb1 |
| 8 | 18 | 1536000 | sdb2 |
| 8 | 19 | 10240 | sdb3 |

| [CentOS | での表 | 示例 】 | |
|----------|-----|----------|------|
| 8 | 0 | 33554432 | sda |
| 8 | 1 | 512000 | sda1 |
| 8 | 2 | 33041408 | sda2 |
| 253 | 0 | 30973952 | dm-0 |
| 253 | 1 | 2064384 | dm-1 |
| 8 | 16 | 7822336 | sdb |
| 8 | 17 | 512000 | sdb1 |
| 8 | 18 | 1536000 | sdb2 |
| 8 | 19 | 10240 | sdb3 |

___2. dd コマンドを実行して Preloader バイナリを書き込みます。

\$ sudo dd if=preloader-mkpimage.bin of=/dev/sdb3 bs=64k seek=0

8-1-2. Soc EDS v14.0 以降のバージョン で Preloader を SD カードに書き込むには

- (1) SoC EDS v14.0 からは、インテルが提供する alt-boot-disk-util というツールを使用して書き込むことができます。
- (2) このツールは Windows 上から Embedded Command Shell のウィンドウからコマンド入力することで実行 可能です。
- Preloader を部分的にアップデートするには、Embedded Command Shell から 下記のコマンドを実行します。

A 注記:

下記コマンドにおいて、-dFは Windows PC に挿した SD カードのドライブが F ドライブである場合の 指定例です。SD カードのドライブは実際のご使用環境に合わせて指定してください。

\$ alt-boot-disk-util -p preloader-mkpimage.bin -a write -d F

8-2. DS-5 プロジェクトへの追加ファイル

ベアメタル・アプリケーションをスタンドアローン実行できるようにするために、次に説明する 2 つのファイル (startup.s、 altera-socfpga-unhosted.ld) を用意して、DS-5 プロジェクト (この例では、Atlas-Blinking-LED-Baremetal-GNU ベアメタル・サンプル・プロジェクト) へ追加します。

(1) startup.s

スタートアップ・ルーチンにおいて割り込み禁止にすることで、ユーザ・アプリケーション実行時に一旦割り 込みを無効化してハングアップを防ぎます (u-boot で有効化した割り込みが、ユーザ・アプリケーション実 行後に入らないようにするための対処です)。

startup.s のソース・コードの例を以下に示します。テキスト・エディタで記述して startup.s という名前で保存し、DS-5 プロジェクトに追加してください。

| .globalreset | |
|-----------------------|-----------------------------------|
| .text | |
| reset: | (* laterau (direkte */ |
| mrs ru,cpsr | /* Interrupt disable */ |
| orr r0,r0,#0xC0 | / Interrupt disable / |
| msr cpsr_cxsf,r0 | /* Interrupt disable */ |
| | |
| mrc p15,0,r0,c1,c0,2 | /* read CPACR */ |
| orr r0,r0,#0x00f00000 | /* Enable NEON/VFP user access */ |
| mcr p15,0,r0,c1,c0,2 | /* Write to CPACR */ |
| isb | |
| mov r0.#0x4000000 | /* Enable VEP/NEON Hardware */ |
| | /* Write to fpeyc */ |
| | / WITCE TO TPEAC / |
| b _start | |

【リスト 8-1】 startup.s のソース・コードの例

(2) altera-socfpga-unhosted.ld

例として、Atlas-Blinking-LED-Baremetal-GNU ベアメタル・サンプル・プロジェクト内にある既存の altera-socfpga-hosted.ld リンカ・スクリプト・ファイルをコピーして、altera-socfpga-unhosted.ld にファイル名 を変更し、以下の 2 箇所を変更します。

これにより、上記 (1) で追加したスタートアップ・ルーチン startup.s の __reset を参照するようになります。





8-3. DS-5 プロジェクトの Makefile の修正

ベアメタル・アプリケーションをスタンドアローン実行できるようにするために、DS-5 プロジェクト(この例では、 Atlas-Blinking-LED-Baremetal-GNU サンプル・プロジェクト)の **Makefile** を追記・修正します。

これにより、startup.s アセンブラ・ソース・コードを追加して、リンカ・スクリプト・ファイルとして altera-socfpgaunhosted.ld を使用するようになります。

既存の Makefile に対して以下の箇所を追記・修正します。

1 注記:

オリジナルの Makefile は、Malefile_org のようにリネームして保存しておくことをお奨めします。

| ASM_SRC := startup.s | # 追記 |
|--|--|
| ASFLAGS := -march=armv7-a -mcpu=cortex-a9 | # 追記 |
| #LINKER_SCRIPT := altera-socfpga-hosted.ld LINKER_SCRIPT := altera-socfpga-unhosted.ld | # オリジナルをコメント・アウト # 追記 |
| AS := \$(CROSS_COMPILE)as | # 追記 |
| <pre>#OBJ := \$(patsubst %.c,%.o,\$(C_SRC)) C_OBJ := \$(patsubst %.c,%.o,\$(C_SRC)) ASM_OBJ := \$(patsubst %.s,%.o,\$(ASM_SRC)) OBJ := \$(C_OBJ) \$(ASM_OBJ)</pre> | # オリジナルをコメント・アウト # 追記 # 追記 # 追記 |
| #\$(OBJ): %.o: %.c Makefile \$(C_OBJ): %.o: %.c Makefile \$(CC) \$(CFLAGS) -c \$< -o \$0 | # オリジナルをコメント・アウト # 追記 |
| \$(ASM_OBJ): %.o: %.s Makefile \$(AS) \$(ASFLAGS) -c \$< -o \$@ | # 追記 # 追記 |

【リスト 8-3】Makefile の追記・修正箇所

8-4. SD カード実行のためのソース・ファイルの変更

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際、必要に応じて アプリケーションのソース・コードを 修正します。

ここでは例として、Atlas-Blinking-LED-Baremetal-GNU サンプル・プロジェクト内にある既存の atlas_main.c ソ ース・ファイルを一部追記・修正しています。

() Note:

- ※ printf() 文を使用してコンソールに文字出力するには、UART 出力関数の作成・追加が必要となりますが、 この例では作成・追加を行いませんので、printf() 文によるコンソール文字出力は実行されません。
- ※ printf() 文が機能しない状態では、LED の点滅が速く目視で確認しづらくなるため、 この例では for (j=0; j < 0x10000; j++); によるディレイを追加しています。</p>

```
int main(int argc, char** argv)
{
    int i;
    int j; /* 追加 */
    printf("Hello from Atlas. ¥n");
    while(1)
    {
        for(i=0; i < 16; i++){
            alt_write_word(LED_BASE_ADDR,i);
            printf("LED [%x] ¥n",i);
            for (j=0; j < 0x10000; j++); /* ディレイを追加 */
        }
    }
    return 0;
}</pre>
```

【リスト 8-4】atlas_main.c ソース・ファイルの変更

8-5. ベアメタル・アプリケーションのビルド

ここまでの手順が全て終わったら、DS-5 プロジェクト(この例では、Atlas-Blinking-LED-Baremetal-GNU サンプ ル・プロジェクト)を右クリックして [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 .axf ファイル (この例では、atlas_main.axf) が生成されます。



8-6. ベアメタル・アプリケーションの実行バイナリの作成と起動

ベアメタル・アプリケーションをスタンドアローン実行させるには、次の2つの方法があります。

- (1) SD カードの FAT 領域 にアプリケーションの実行バイナリ・ファイルを配置して u-boot から起動 する方 法 (7-6-1 節で説明)
- (2) SD カードの u-boot 領域 にアプリケーションの実行バイナリ・ファイルを格納して Preloader から起動 する方法 (7-6-2 節で説明)

8-6-1. アプリケーションを SD カードの「FAT 領域」に配置して "u-boot から起動" する方法

u-boot からユーザ・アプリケーションを起動する場合は、アプリケーションの実行バイナリ・ファイルを生成し、 それを Windows から認識可能な FAT パーティションに対してコピーすることになります。

以下に手順を説明します。

1. Embedded Command Shell から 以下のコマンドを実行して、DS-5 プロジェクト (この例では、 Atlas-Blinking-LED-Baremetal-GNU サンプル・プロジェクト) ディレクトリに移動して、ベアメタル・アプリケーション の実行バイナリ・ファイル (この例では、atlas_main.bin)を作成します。

\$ cd "C:¥Work¥DS-5 Workspace¥Atlas-Blinking-LED-Baremetal-GNU" \$ arm-altera-eabi-objcopy -v -0 binary atlas_main.axf atlas_main.bin

- ____2. 生成されたベアメタル・アプリケーション実行バイナリ・ファイル(この例では、atlas_main.bin)を SD カー ドの FAT 領域にコピーします。
- ____3. アプリケーションが FPGA 側にアクセスする場合は、FPGA をコンフィグレーションしておきます。
- ____4. SD カードをボードに取り付けて、ボードを WARM リセットします。 <mark>u-boot のコンソール</mark>にオート・ブート のカウント・ダウンが表示されるので、 Enter キーを入力してカウント・ダウンを停止させます。
- ____5. u-boot のプロンプトから 以下のコマンドを実行するとアプリケーションが起動します。

▲ 注記:

下記コマンド例は、アプリケーションのスタート・アドレスが 0x02000000 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。

fatload mmc 0 0x02000000 atlas_main.bin]

go 0x02000000

\Lambda ALTIMA

前述の u-boot のコマンドを自動化する場合には、u-boot のスクリプト機能が使用可能です。 6.

コマンドをテキスト・ファイルとして保存し、mkimage ツールを使ってヘッダ情報を付加したものを FAT パ ーティションに格納しておきます。 u-boot がロード対象とするスクリプト名は u-boot.scr となっています。

① 例えば、下記のような内容でテキスト・ファイル script.txt を作成します。

run bridge_enable_handoff; fatload mmc 0 0x02000000 atlas_main.bin; go 0x02000000;

【リスト 8-5】 script.txt の記述例

② mkimage ツールを実行して script.txt から u-boot.scr を生成します。 u-boot で実行するファイルは、mkimage コマンドで u-boot 用のフォーマットに変換する必要がありま す。

Embedded Command Shell から 以下のコマンドを実行します。

\$ mkimage -A arm -O linux -T script -C none -a 0 -e 0 -n "My script" -d script.txt u-boot.scr J



SoC EDS v13.1 以前のバージョン では、デフォルトで mkimage ツールのパスが通っていないため、 DS-5 プロジェクト(この例では、Atlas-Blinking-LED-Baremetal-GNU ベアメタル・サンプル・プロジェクト) 内に mkimage.exe ファイルをコピーしておく必要があります。

mkimage.exe は、bsp-editor で生成した以下のフォルダにあります。 spl_bsp/uboot-socfpga/tools/mkimage (spl_bsp は bsp-editor で指定したフォルダです)

また、プロジェクト内にコピーした mkimage.exe を実行する際は、コマンド・ラインにおいて mkimage と入力してください。

- ③ SD カードを PC の SD カード・スロットに取り付けて、生成された u-boot.scr を SD カードの FAT 領 域にコピーします。
- ④ SD カードをボードに取り付けて、ボードを WARM リセットすると、u-boot 実行後にベアメタル・アプリ ケーションが自動的に起動します。

8-6-2. アプリケーションを SD カードの「u-boot 領域」に格納して "Preloader から起動" する方法

Preloader からユーザ・アプリケーションを起動する場合は、アプリケーションの実行バイナリ・ファイルを生成 し、それを SD カードの u-boot 領域に対してコピーすることになります。

以下に手順を説明します。

- ____1. Embedded Command Shell から 前述「8-6-1. ____1」のコマンドを実行してベアメタル・アプリケーションの実行バイナリ・ファイル(この例では、atlas_main.bin)を作成します。
- ____2. 以下のコマンドを実行して u-boot 領域に格納するベアメタル・アプリケーション実行バイナリ・ファイル(この例では、atlas_main.img.bin)を作成します。

1 注記:

下記コマンド例は、アプリケーションのスタート・アドレスが 0x02000000 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。

\$ mkimage -A arm -O u-boot -T standalone -C none -a 0x02000000 -e 0 ¥
-n "baremetal image" -d atlas_main.bin atlas_main.img.bin

____3. SD カードの u-boot 領域にベアメタル・アプリケーション実行バイナリ・ファイル (この例では、 atlas_main.img.bin)を書き込みます。

ご使用の SoC EDS のバージョンに応じて、下記 ① または ② の方法で SD カードの u-boot 領域に 実行バイナリ・ファイルを書き込みます。

① Soc EDS v13.1 以前 のバージョンでは、Soc EDS 自体に バイナリ・ファイルを書き込むためのツール が無いため、基本的に Linux の dd ユーティリティ・コマンドを使用します。 そのため Linux OS がインストールされた PC が必要です。

てのため Linux OS がインストールCAL/こ PC が必要です。

u-boot 領域を部分的にアップデートするには、Linux PC に SD カードをマウントし dd コマンドを実行 します。

a) Linux PC に SD カードを接続して以下のコマンドを実行すると、この例では sdb が見えるように なります (PC の SD カード・スロットで見えない場合は、USB カード・リーダ経由で試してください)。

\$ sudo cat /proc/partitions .

 8
 16
 7822336
 sdb

 8
 17
 512000
 sdb1

 8
 18
 1536000
 sdb2

 8
 19
 10240
 sdb3

b) 以下の dd コマンドを実行して SD カードの u-boot 領域にベアメタル・アプリケーションの実行 バイナリ・ファイルを書き込みます。

\$ sudo dd if=atlas_main.img.bin of=/dev/sdb3 bs=64k seek=4 d

② SoC EDS v14.0 以降のバージョンでは、Embedded Command Shell から以下の alt-boot-disk-util コマンドを実行して、SDカードの u-boot 領域に実行バイナリ・ファイル (この例では、atlas_main.img.bin)を書き込みます。

1 注記:

下記コマンドにおいて、-d F は Windows PC に挿した SD カードのドライブが F ドライブである場合の 指定例です。SD カードのドライブは実際のご使用環境に合わせて指定してください。

\$ alt-boot-disk-util -b atlas_main.img.bin -a write -d F

- ____4. アプリケーションが FPGA 側にアクセスする場合は、FPGA をコンフィグレーションしておきます。
- ____5. SD カードをボードに取り付けて、ボードを WARM リセットすると、 Preloader 実行後に ベアメタル・アプ リケーションが自動的に起動します。



<u> 改版履歴</u>

| Revision | 年月 | 概要 |
|----------|------------|---|
| 1 | 2016 年 5 月 | 初版、15.0 |
| 2 | 2018 年 9 月 | 書式変更 Windows[®] 10 使用の際の Preloader 生成における注記を追加 リンク URL 修正 |
| 3 | 2019 年 1 月 | 18.0 に対応 DS-5 デバッガ・スクリプトの内容について追記 演習ファイルも 18.0 対応とし Atlas /DE10-Nano で使えるように変更 全体の文書構成を見直し |

免責およびご利用上の注意

弊社より資料を入手されましたお客様におかれましては、下記の使用上の注意を一読いただいた上でご使用ください。

- 1. 本資料は非売品です。許可無く転売することや無断複製することを禁じます。
- 2. 本資料は予告なく変更することがあります。
- 本資料の作成には万全を期していますが、万一ご不明な点や誤り、記載漏れなどお気づきの点がありましたら、本資料を入手されました下記代理店までご一報いただければ幸いです。
 株式会社マクニカ アルティマ カンパニー https://www.alt.macnica.co.jp/ 技術情報サイト アルティマ技術データベース https://www.alt.macnica.co.jp/
- 4. 本資料で取り扱っている回路、技術、プログラムに関して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。
- 5. 本資料は製品を利用する際の補助的な資料です。製品をご使用になる際は、各メーカ発行の英語版の資料もあわせてご利用ください。