

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.15



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)

1.	はじめに	4
2.	ARM プロセッサを含むハードウェアの設計を行う	6
3.	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)	7
4.	アルテラ SoC のブート・フロー	8
5.	SoC EDS 付属のサンプル・アプリケーション	9
6.	DS-5 の起動	. 10
6	-1. Embedded Command Shell の起動	. 10
6	-2. DS-5 の起動	. 10
7.	サンプル・アプリケーションのインポート	. 12
8.	サンプル・アプリケーションのファイル構成	. 13
9.	Makefile の例	. 14
10.	デバッガ・スクリプト・ファイルでの実行内容	. 15
11.	Preloader(プリローダ)とは?	. 16
12.	Preloader の生成手順	. 17
1	2-1. Embedded Command Shell の起動	. 17
1	2-2. bsp-editor (Preloader Generator)の起動	. 17
1	2-2-1. 新規プロジェクトの作成	. 17
1	2-2-2. ハンドオフ・ファイルの指定	. 18
1	2-2-3. Preloader のユーザ・オプション(Common)の設定	. 19
1	2-2-4. Preloader のユーザ・オプション(Advanced → spl → boot)の設定	. 20
1	2-2-5. Preloader のユーザ・オプション(Advanced → spl → debug)の設定	. 21
1	2-2-6. bsp プロジェクトの生成(Generate)	. 22
1	2-3. Preloader のビルド	. 23
13.	カスタム・ボードへの対応方法	. 24
1	3-1. 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	. 24
1	3-2. Makefile の編集	. 25
1	3-3. 生成した Preloader を SD カードに書き込むには	. 26
1	3-3-1. 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	. 26
1	3-3-2. 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	. 27



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)

14. FPGA レジスタの確認方法	
15. 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 32
15-1. SD カードの準備	
15-2. DS-5 プロジェクトへの追加ファイル	
15-3. DS-5 プロジェクトの Makefile の修正	
15-4. ソース・ファイルの変更	
15-5. ベアメタル・アプリケーションのビルド	
15-6. 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36
15-6-1.SD カードの FAT 領域 にアプリケーションを配置して	u-boot から起動 する方法36
15-6-2.SD カードの u-boot 領域 にアプリケーションを格納し	て Preloader から起動 する方法38
改版履歴	

1. <u>はじめに</u>

本資料では ARM[®] DS-5[™] を利用したアルテラ SoC 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 Mpression Helio ボード(以下、Helio と呼ぶ)向けの ベアメタ ル・サンプル・アプリケーション Helio-BlinkingLED-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。

- ・ 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)
- ・ アルテラ SoC のブート・フロー
- ・ SoC EDS 付属のサンプル・アプリケーション
- ・DS-5 の起動
- サンプル・アプリケーションのインポート
- サンプル・アプリケーションのファイル構成
- ・ Makefile の例
- ・ デバッガ・スクリプト・ファイルでの実行内容
- ・ Preloader の生成手順
- ・ カスタム・ボードへの対応方法
- ・ FPGA レジスタの確認方法
- ベアメタル・アプリケーションの SD カードからの実行例

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	アルテラ	アルテラ SoC FPGA のハードウェアを開発するためのツールです。
	Quartus [®] II	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。
	開発ソフトウェ	本資料では、Quartus II 開発ソフトウェア v15.0 を使用しています。
	ア	■ Quartus II 開発ソフトウェア
		http://fpgasoftware.intel.com/15.0/?edition=subscription
3	アルテラ SoC	アルテラ SoCFPGA のソフトウェアを開発するためのツールです。
	EDS サブスクリ	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
	プション・エディ	に含まれる ARM DS-5 Altera 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
	ション(ARM DS-	
	5 Altera Edition)	本資料では、アルテラ SoC EDS v15.0 を使用しています。
		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、アルテラ SoC EDS のサブスクリプ ション・エディション(有償版)が必要になります。
		■ アルテラ SoC エンベデッド・デザイン・スイート
		http://fpgasoftware.intel.com/soceds/15.0/?edition=subscription&download_manager=dlm3&platform=windows

À			
	4	Helio ボード	本資料の説明でターゲット・ボードとして使用する、アルテラ Cyclone® V SoC を搭載した Mpression Helio ボードです。
			Helio には複数のリビジョンが存在しますが、この資料では、Rev.1.2 または Rev.1.3 を使用して動作確認を行 っています。
			■ Helio ボード Rev.1.2
			https://rocketboards.org/foswiki/Documentation/HelioResourcesForRev12
			■ Helio ボード Rev.1.3
			https://rocketboards.org/foswiki/Documentation/HelioResourcesForRev13
	5	ベアメタル・サ	本資料の説明で使用する、Helio 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。
		ンプル・アプリ	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。
		ケーション	Helio-BlinkingLED-Baremetal-GNU.tar.gz
			ARM DS-5 Altera Edition を使用して、本アプリケーションをインポートし、コンパイルしてデバッグすることができます。

2. <u>ARM プロセッサを含むハードウェアの設計を行う</u>

本資料では アルテラ SoC 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 II 開発ソフトウェアおよび Qsys システム統合ツールを使用し、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 と呼ぶ)コンポーネントを Qsys 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) Qsys システムが完成したら Generate して生成します。
- (4) Quartus II 開発ソフトウェアで、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。

3. 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、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。このファイルを SoC FPGA にプログラムして FPGA をコンフィギュレーションします。

(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、「12. 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>¥<Qsys 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、「14. FPGA レジスタの確認方法」に利用します。 DS-5 Altera Edition で、この <Qsys 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。

4. <u>アルテラ SoC のブート・フロー</u>

以下の通り、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、アルテ ラのアプリケーション・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 4.1】 アルテラ SoC のブート・フロー

5. <u>SoC EDS 付属のサンプル・アプリケーション</u>

アルテラ 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。

<Altera installation directory>¥embedded¥examples¥software

例) C:¥altera¥15.0¥embedded¥examples¥software

これらのサンプル・アプリケーションは、DS-5 Altera 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。

. → ライブラリに追加 →	共有 ▼ 書き込む 新しいフォルダー		3=	• 🖽
J 15.0	名前	更新日時	種類	サイズ
🗼 embedded	Itera-SoCFPGA-Blinking-LED-Linux-GNU.tar.gz	2015/05/11 6:04	GZ ファイル	5
📕 drivers	Land Altera-SoCFPGA-HardwareLib-16550-CV-ARMCC.tar.gz	2015/05/11 6:04	GZ ファイル	22
🍌 ds-5	🐉 Altera-SoCFPGA-HardwareLib-16550-CV-GNU.tar.gz	2015/05/11 6:04	GZ ファイル	22
📕 ds-5_installer	Altera-SoCFPGA-HardwareLib-ECCL2-CV-ARMCC.tar.gz	2015/05/11 6:04	GZ ファイル	15
embeddedsw	Altera-SoCFPGA-HardwareLib-ECCL2-CV-GNU.tar.gz	2015/05/11 6:04	GZ ファイル	15
examples	🐉 Altera-SoCFPGA-HardwareLib-FPGA-CV-ARMCC.tar.gz	2015/05/11 6:04	GZ ファイル	13
hardwara	👃 Altera-SoCFPGA-HardwareLib-FPGA-CV-GNU.tar.gz	2015/05/11 6:04	GZ ファイル	12
	Altera-SoCFPGA-HardwareLib-MPL.tar.gz	2015/05/11 6:04	GZ ファイル	92
J software	Altera-SoCFPGA-HardwareLib-SPI-CV-ARMCC.tar.gz	2015/05/11 6:04	GZ ファイル	12
host_tools	Altera-SoCFPGA-HardwareLib-SPI-CV-GNU.tar.gz	2015/05/11 6:04	GZ ファイル	203
🌗 ip	🔑 Altera-SoCFPGA-HardwareLib-Timer-CV-ARMCC.tar.gz	2015/05/11 6:04	GZ ファイル	10
퉬 hld	🐉 Altera-SoCFPGA-HardwareLib-Timer-CV-GNU.tar.gz	2015/05/11 6:04	GZ ファイル	10
🚹 ip	Altera-SoCFPGA-HelloWorld-Baremetal-ARMCC.tar.gz	2015/05/11 6:04	GZ ファイル	5
licenses	🐉 Altera-SoCFPGA-HelloWorld-Baremetal-GNU.tar.gz	2015/05/11 6:04	GZ ファイル	7
logs	👪 Altera-SoCFPGA-HelloWorld-Linux-GNU.tar.gz	2015/05/11 6:04	GZ ファイル	3
ings	👪 Altera-SoCFPGA-Push-Button-Linux-GNU.tar.gz	2015/05/11 6:04	GZ ファイル	3
modelsim_ase				

【図 5.1】SoC EDS 付属のサンプル・アプリケーション

6. <u>DS-5 の起動</u>

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 Embedded Command Shell から起動してください。

6-1. 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 6-1.1】Embedded Command Shell の起動

6-2. DS-5 の起動

(1) 図のように Embedded Command Shell のウインドウが開いたら eclipse & とコマンド入力し て ARM DS-5 Altera Edition を起動します。

 ~		
Altera Embedded Com	mand Shell	* III
Version 15.0		
elsfae@elsfae03 ~		
\$ eclipse & <	Embedded Command Shell から "eclipse &" とコマンド	人力 ·



(2) Eclipse ツールを使用するワークスペース・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:¥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。)

● ワークスペース・ランチャー		
ワークスペースの選択 Eclipse ブラットフォーム は、ワークスペースと呼ばれるフォルダにプロジェクトを保存しま このセッションに使用するワークスペース・フォルダを選択してください。	3 .	
ワークスペース(W): C:¥Work¥DS-5 Workspace	▼ 参照(8)	
この違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)		
	ОК	キャンセル

【図 6-2.2】DS-5 のワークスペースの指定

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



【図 6-2.3】DS-5 ウェルカム画面

7. <u>サンプル・アプリケーションのインポート</u>

この例では、事前に用意された Helio 向け LED 点滅ベアメタル・サンプル・アプリケーションを DS-5 にイン ポートします。この作業を実施すると、Eclipse 左側の プロジェクト・エクスプローラー パネルにプロジェクトに含 まれる各種ファイルが表示されます。

- (1) ファイル(F) メニュー ⇒ 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。

ファ	イル(F) 編集(E) ソース(S) リフ: 新現(N) ファイルを間く(.)	更が 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。
	閉じる(C) すべて閉じる(L)	インポート・ソースの違訳(S): フィルタ入力
	保管(5) 別名保存(A) すべて保管(E) 前回保管した状態に戻す(T)	・ 一般 ・ 一般 ・ アーカイブ・ファイル ・ ファイル・システム ・ ごの方「コジェクトをワークスペースへ 日安方
٤)	容動(v) 名前変更(M) 更新(F) 行区切り文字の変換(D)	 ▷ ≥ C ▷ ≥ CVS ▷ ≥ Scatter File Editor ▷ ≥ Target Configuration Editor
8	印刷(P) ワークスペースの切り替え(W)	 ▷ → → ンストール ▷ → → → → → → → → → → → → → → → → → → →
<u>ک</u>	中間 インポート(I) エクスポート(O)	

【図 7.1】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。 Helio-BlinkingLED-Baremetal-GNU.tar.gz 選択後、[終了(F)] ボタンを押します。

● インポート		
プロジェクトのインポート 既存の Edipse プロジェクトを検索す	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¥Helio-BlinkingLED-Baremetal-GNU.tar.gz	▼ 参照(R)
プロジェクト(P):		
I Helio-Blinking-LED-Baremeta	l-GNU (Helio-BlinkingLED-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 ⑦ Search for nested projects ⑦ プロジェクトをワークスペースにコ	ピ−(C)	
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)	
ワーキング・セット(0):	÷	暹択(E)
? < 戻る(B)) 次へ(N) > 終了(F)	キャンセル

【図 7.2】サンプル・アプリケーションの選択

8. <u>サンプル・アプリケーションのファイル構成</u>

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。

ビルド前

ビルド後



【図 8.1】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以下のようなものがあります。

- ・ Makefile : プロジェクトをビルドする際の指示書です。
- ・ debug-hosted.ds : デバッグ時に実行するスクリプト・ファイルです。
- ・<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です。
- ・u-boot-spl.axf : Preloader の実行可能バイナリです。



9. Makefile の例

サンプル・アプリケーションの Makefile の例を下図に示します。

※ 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。

Ta *Makefile 🛛	
36 37 ALT_DEVICE_FAMILY ?= soc_cv_av SoC デバイス・ファミリの指定	
as SOCEDS_ROOT ?= \$(SOCEDS_DEST_ROOT) 49 HMLIBS_ROOT = \$(SOCEDS_ROOT)/ip/altera_hps/ahlib 41 HMLIBS_ROOT = \$(SOCEDS_ROOT)/ip/altera_hps/ahlib	
42 HNLIBS_SRC := alt_clock_manager.c alt_generalpurpose_io.c alt_globaltmr.c alt_interrupt.c alt_timers.c alt_watchdog.c 43 EXAMPLE_SRC := hwlib.c 44 C_SRC := \$(EXAMPLE_SRC) \$(HWLIBS_SRC)	掾のソース・ファイルの指定
45 44 LINKER_SCRIPT := cycloneV-dk-ram-hosted.ld 47 リンカ・ファイルの指定	
<pre>48 MULTILIBFLAGS := -mcpu=cortex-a9 -mfloat-abi=softtp -mtpu=neon 49 CFLAGS := -mcpu=cortex-a9 -mfloat-abi=softtp -mtpu=neon 49 CFLAGS := -mtpu=soft=soft=soft=soft=soft=soft=soft=soft</pre>	DEVICE_FAMILY)
52 CROSS_COMPILE := arm-altera-eabi- 53 CC := \$(CROSS_COMPILE)gc 54 LD := \$(CROSS_COMPILE)g++ 55 MM := \$(CROSS_COMPILE)nm 56 GD := \$(CROSS_COMPILE)objdump	
5/0C := \$(CROS <u></u> COMPILE)objcopy 58 59 RM := rm -rf 60 CP := cp -f	
61 62ELF = \$(basename \$(firstword \$(C_SRC))).axf 635PL := u-boot-spl.axf (C_SRC)).axf	
<pre>c=uou := \$(partumor #.c,#.o,\$(C_SKL)) 65 66.PPDOM: all 67all \$(FEP \$(SPL))</pre>	
68 69 .PHONY: clean 70 clean:	
71 \$(RM) \$(ELF) \$(SPL) \$(083) *.objdump *.map \$(HullBS_SRC) 72 73 74 74 74 74 75 74 75 76 76 77 77 74 77 74 77 77 74 77 74 77 74 74	
7 ⁴ \$(1); \$(2) ⁷⁴ \$(CP) \$(2) \$(1) ⁷⁴ endef \$(SOCEDS_ROOT)/ip/altera/hps/altera_hps/hwlib/src/hwmgr/ 以下から	コピーされる
77 78 ALL_HWLIBS_SRC = \$(wildcard \$(HWLIBS_ROOT)/src/hwmgr/*.c) \$(wildcard \$(HWLIBS_ROOT)/src/hwmgr/\$(ALT_DEVICE_FAMILY)/*.c) 70	
<pre>//5 80 \$(foreach file,\$(ALL_HWLIBS_SRC),\$(eval \$(call SET_HWLIBS_DEPENDENCIES,\$(notdir \$(file)),\$(file)))) 81</pre>	
82\$(OBJ): %.o: %.c Makefile 83 \$(CC) \$(CFLAGS) -c \$< -o \$@ 84	
- 5\$\$(ELF): \$(0BJ) 86 \$(LD) \$(LDFLAGS) \$(0BJ) -o \$@ 7 \$(00) -d \$@ > \$@.objdump 88 \$(NM) \$@ > \$@.map	
96 \$(SPL): \$(SOCEDS_ROOT)/examples/hardware/cv_soc_devkit_ghrd/software/preloader/uboot-socfpga/spl/u-boot-spl 91 \$(CP) \$< 98 93 \$(OD) -d 98 > \$8.objdump	可能バイナリ (u-boot-spl) のコピー

【図 9.1】Makefile の例

Makefile の中では主に次のようなものが定義されています。

- ・SoC 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

※ 注記:

クロス・コンパイラの指定は、ご使用のアルテラ 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-

10. デバッガ・スクリプト・ファイルでの実行内容

ARM 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。

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。

※ 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。



【図 10.1】デバッガ・スクリプト・ファイルでの実行内容の例

11. <u>Preloader(プリローダ)とは?</u>

- Preloader は U-boot second program loader(以後、u-boot spl)をベースに、アルテラ SoC 向けにカスタ マイズが加えられたブートローダです。
- (2) 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
- (3) Preloader は Quartus II / Qsys の設計時に自動生成されるハンドオフ・ファイルを用いることで自動生成 されます。このため、ユーザ側で初期化用ソフトウェアの構築をすることなく Quartus II / Qsys で設定した 内容を HPS ブロックに反映することができます。
- (4) ユーザのアルテラ SoC 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。

12. 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。

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

12-1.Embedded Command Shell の起動

「6-1. Embedded Command Shell の起動」の項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。

12-2.bsp-editor (Preloader Generator)の起動

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

 ~	
Altera Embedded Comma	nd Shell
Version 15.0	
alafaaMalafaad? ~	
\$ bsp-editor <	Embedded Command Shell から "bsp-editor" とコマンド入力

【図 12-2.1】bsp-editor (Preloader Generator)の起動

12-2-1. 新規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 メニュー ⇒ New HPS BSP... を選択して、新規プロジェクトを作成します。

※ 注記:		
SoC EDS v15.0 より前 の	ヾージョンでは、 "File → New BSP…"	を選択します。



Edit Tools Help		
New Nios II BSP	Ctrl+N	Linker Script Enable File Generation Target BSP Directory
New HPS BSP	Ctrl+H	
Open	Ctrl+0	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
Save	Ctrl+S	Version: 🚽
Save As		
Exit	Ctrl+X	

【図 12-2-1.1】新規プロジェクトの作成

12-2-2. ハンドオフ・ファイルの指定

- 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。
- (2) 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。

ſ	+ New BSF	,	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	を指定する
	Hardware			
	Preload	ler settings directory:	C:\Work\helio_ghrd_5csxc6es_v15.0\hps_isw_handoff\soc_system_hps_0	
	Software	Operating system	に Preloader が、Version に default が選択されていることを確認	
		Operating system:	U-Boot SPL Preloader (Cyclone V/ 👻 Version: default 👻	
		(Use default locations	
		BSP target directory:	C:\Work\helio_ghrd_5csxc6es_v15.0\software\spl_bsp	
	BS	P Settings File name:	C:\Work\helio_ghrd_5csxc6es_v15.0\software\spl_bsp\settings.bsp	
			Enable Settings File relative paths	
BSP target d	irectory にて は	osp プロジェクトを生成 ⁻	するロケーションを指定する	
デフォルトて	ごは			
" <quartus< th=""><td>プロジェクト>¥so</td><td>oftware¥spl_bsp"</td><td>指定が終わったら [OK]</td><td> をクリックする</td></quartus<>	プロジェクト>¥so	oftware¥spl_bsp"	指定が終わったら [OK]	をクリックする
が選択され	る			
Use default l	locations のチュ	ニックを外すと任意のデ	ィレクトリを指定することが可能	
			OK Car	ncel

【図 12-2-2.1】 ハンドオフ・ファイルの指定



12-2-3. 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。

※ 注記: 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)。

Software Packages Drivers Linker Script Enab	e File Generation Target BSP Directory	
OPC Information file: CPU name: Operating system: U-Boot SPL Preloader (Cyclone BSP target directory: .\	V/Arria Version: default 👻	
Settings	SPI PRELOADER_TGZ: CROSS_COMPILE:	preloader/uboot-socfpga.tar.gz arm-altera-eabi-
	spl.boot □ BOOT_FROM_QSPI □ BOOT_FROM_SDMMC □ 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:	0x60000 0x40000 0xc0000 1 u-bootimg

【図 12-2-3.1】Preloader のユーザ・オプション(Common)の設定



12-2-4. Preloader のユーザ・オプション(Advanced → spl → boot)の設定

Advanced \rightarrow spl \rightarrow 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¥helio_ghrd_5csxc6es_v1 File Edit Tools Help	5.0¥software¥spl_bsp¥settings.bsp	- 8 - 2	
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 ▼	チドッグ・タイマを使用しない場合は、	
- Settings	spl.boot WATC	HDOG_ENABLE のチェックを外す	
Generation	CHECKSUM_NEXT_IMAGE		
BOOT_FROM_QSPI	FPGA_MAX_SIZE:	0x10000	
BOOT_FROM_SDMMC BOOT_FROM_NAND	FPGA_DATA_BASE:	0xffff0000	
QSPI_NEXT_BOOT_IMAGE	FPGA_DATA_MAX_SIZE:	0x10000	
	STATE_REG_ENABLE		
FAT_SUPPORT	BOOTROM_HANDSHAKE_CFGIO		
FAT_LOAD_PAYLOAD_NAME	WARMRST_SKIP_CFGIO		
l⊟ Advanced ⊟ spl	SDRAM_SCRUBBING		
reset_assert Hereset_bandsbake	SDRAM_SCRUB_BOOT_REGION_STAR	T: 0x1000000	
	SDRAM_SCRUB_BOOT_REGION_END:	0x2000000	
⊕ performance	SDRAM_SCRUB_REMAIN_REGION		
	RAMBOOT_PLLRESET		

【図 12-2-4.1】Preloader のユーザ・オプション(Advanced → spl → boot)の設定



12-2-5. 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 Altera 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。

※ 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¥helio_ghrd_5csxc6es_v Elia Edit Toola Hela	15.0¥software¥spl_bsp¥settings.bsp	
Main Software Packages Drivers Linker Script, Enabl	e File Generation Target BSP Directory	
SOPC Information file: CPU name: Operating system: U-Boot SPL Preloader (Cyclone BSP target directory: .\	V/Arria Version: default 🗸	
	Spi.dedug DEBUG_MEMORY_WRITE DEBUG_MEMORY_ADDR: DEBUG_MEMORY_SIZE: SEMIHOSTING HARDWARE DIAGNOSTIC DS-5 のセミ SEMIHOSTIN	0xffffd00 0x200 ホスティング機能を使用する場合は、 NG のチェック・ボックスを ON にする

【図 12-2-5.1】Preloader のユーザ・オプション(Advanced → spl → debug)の設定



12-2-6. 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、「12-2-2. 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¥helio_ghrd_5csxc6es	_v15.0¥software¥spl_bsp¥settings.bsp	
File Edit Tools Help		
Main Software Packages Drivers Linker Script En	able File Generation Target BSP Directory	
SOPC Information file: CPU name: Operating system: U-Boot SPL Preloader (Cyclo BSP target directory: .\	ne V/Arria Version: default 👻	
Settings Common PRELOADER_TGZ CROSS_COMPILE boot BOOT_FROM_QSPI BOOT_FROM_QSPI BOOT_FROM_NAND BOOT_FROM_NAND BOOT_FROM_RAM QSPI_NEXT_BOOT_IMAGE SDMMC_NEXT_BOOT_IMAGE FAT_SUPPORT FAT_BOOT_PARTITION FAT_LOAD_PAYLOAD_NAME Advanced Forset_assert Warm_reset_handshake boot Gebug Formance	spl.debug DEBUG_MEMORY_WRITE DEBUG_MEMORY_ADDR: DEBUG_MEMORY_SIZE: SEMIHOSTING HARDWARE_DIAGNOSTIC SKIP_SDRAM	0xfffffd00 0x200
Information Problems Processing Searching for BSP components with category: drive Searching for BSP components with category: softw Added operating system component "sol: 1.0"	[Generate] ボタンを押下し bsp プロジェクトを生 生成完了を確認後、[Exit] ボタンを押下し bsp-e arepackage_element	E成する editor (Preloader Generator) を終了する
Generated file "C: Work helio_ghrd_5csxc6es_v15.	0\software\spl_bsp\settings.bsp"	Generate Exit

【図 12-2-6.1】bsp プロジェクトの生成



12-3.Preloader のビルド

 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" 🚽

elsfae@elsfae03 ~ \$_cd ~C:¥Work¥helio_ghrd_5csxc6es_v15.0¥software¥spl_bsp~				
elsfae@elsfae03 /cy, 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する				
\$ Is generated Makefile preloader.ds settings.bsp uboot.ds				
elsfae@elsfae03 /cygdrive/c/Work/helio_ghrd_5csxc6es_v15.0/software/spl_bsp \$;_				

【図 12-3.1】 bsp プロジェクトのディレクトリに移動

make all J コマンドを実行し Preloader を生成します。

コマンド実行後、エラーがなく終了したことを確認します。

確認後 1s 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。



【図 12-3.2】"make all" コマンドを実行

※ 注記:

ホスト PC の OS が Windows® 10 の場合、Preloader の生成でエラーが発生する場合が確認されてお ります。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 MACNICA フォーラムで 説明されている対策として、bsp プロジェクト内 Makefile の編集が必要となりますのでご注意ください。 https://forum.macnica.co.jp/t/topic/1191/11

13. カスタム・ボードへの対応方法

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) 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 : u-boot-spl.axf
- Makefile の編集
 デフォルトの Preloader に関係する処理を Makefile から削除します。

13-1.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

- ARM 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 仕様書に記載されており、.axf というファイル拡張子を使用します。
- ■「12. 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 13-1.1】 生成された u-boot-spl ファイル

■ 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 とリネームして、

例えば Helio-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 13-1.2】 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

13-2.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。

- ※ 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 アルテラ Cyclone V SoC 開発キットの Preloader のコピーをさせないように変更し ます。
- (2) make clean にてコピーしたカスタム・ボード向け Preloader 実行可能バイナリが削除されないように変更 します。



【図 13-2.1】Makefile の編集例

13-3.生成した Preloader を SD カードに書き込むには

- 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。
- まずは RocketBoards.org に公開されている SD カード・イメージ、または SoC EDS 付属の 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=SnJrEv17u0M

■ Preloader バイナリ・ファイルの書き込みは、SoCEDS 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: アルテラが提供する 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。

13-3-1. 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。
- 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「12-3. 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

【 Ubuntu での表示例 】			[CentO	S での表	表示例】		
11	0	1048575	sr0	8	0	33554432	sda
8	0	100663296	sda	8	1	512000	sda1
8	1	96468992	sda1	8	2	33041408	sda2
8	2	1	sda2	253	0	30973952	dm-0
8	5	4191232	sda5	253	1	2064384	dm-1
8	16	7822336	sdb	8	16	7822336	sdb
8	17	512000	sdb1	8	17	512000	sdb1
8	18	1536000	sdb2	8	18	1536000	sdb2
8	19	10240	sdb3	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

- 13-3-2. 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
 - SoC EDS v14.0 からは アルテラ 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。
 - このツールは Windows 上から Embedded Command Shell のウイ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。

書き込む Preloader のバイナリ・ファイルは「12-3. 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。

```
※ 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

14. FPGA レジスタの確認方法

- ペリフェラル記述ファイル(.svd)を追加して DS-5 のレジスタ・ビューから FPGA / Soft IP レジスタを見るには、DS-5 の「デバッグの構成」から、
- (1) 「ファイル」タブをクリックして、
- (2) ディレクトリからペリフェラル記述ファイルを追加 を選択し、
- (3) 「ファイルシステム」ボタンをクリックして、
- (4) <Quartus プロジェクト>¥<Qsys プロジェクト>¥synthesis パスを指定します。

按統	🗟 ファイル	🏘 デバッス	ガ) 🍓 OS の認調	載機能 []] (≍)= 引数	76 環境
ター !	デットコンフィ	ギュレーショ	>			
ダウン	レロードする木	スト上のアプ	リケーション:			
ר דר דיר	イルシステム ハル	. ワークス	ぺ−ス]□≥	ンポル	をロード	します
	ディレクトリカ	らペリフェき	ラル記述ファイル	レを追加	します	•
100 million -	C:¥Work¥helio	_ghrd_5csx	c6es_v15.0¥so	c_syste	em¥synt	thesis
-			h7~-7			
8	ファイルシスラ	FA	77. · 7			

【図 14.1】DS-5 の「デバッグの構成」でペリフェラル記述ファイルを追加

(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。

● デバッグ構成	
構成の作成、管理、および実行	- T
 ・ ・ ・	名前(N): Helio-Blinking-LED-Baremetal-Debug ● 接続 ● ファイル 参 デバッガ ● OS の認識機能 ● 引数 ■ 素塊 ターグットの選択 使用する製造元、ポード、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 (Single Core) / Bare Metal Debug / Debug Cortex-A9_0 「ファトフォームのフィルタ ● Cyclone V SoC (Dual Core) ● Cyclone V SoC (Single Core) ● Bare Metal Debug Debug Cortex-A9_0 ● Cyclone V SoC (Single Core) ● Bare Metal Debug Debug Cortex-A9_0 ● DTSL オプション 編集 USB-Blaster トレースまたはその他のターケットオプションを設定します。"default" コンフィギュレーションオプショ DFS Debugger will connect to an Altera USB-Blaster to debug a bare metal application. 接続 Bare Metal Debug Connection Helio on localhost [USB-1]:Helio USB-1
フィルター一致: 19 / 19 項目	適用(Y) 前回保管した状態に戻す(V) デパッグ(D) 閉じる

【図 14.2】「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。

【図 14.3】 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。

(M= 変数 💊 ブレークポイ	ント 🎟 レジスタ 🛛 ^{とい} 式 f() 関数	۶.	
	名前	値	サイズ アクセス
🖶 🗁 Core			
🖶 🗁 CP15			
🖶 🗁 VFP			
🖶 🗁 NEON			
🛓 🗁 Peripherals			

【図 14.4】 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 鸟 ブレークポイント 💷 レジスタ 🛙	(×)= 変数	🎭 ブレークポイント 🚥	レジスタ 🛙	¥+¥ <u>द</u> ्र	f() 関数
名前		名前	値	サイズ	アクセス
E Core	👎 🗁 O	ore			
🖶 🗁 CP15	- 0	R0	0x00000000	32	R/W
🕀 🗁 VFP	- •	R1	0x022664A4	32	R/W
🕀 🗁 NEON	- •	R2	0x00000001	32	R/W
	- •	R3	0x00000000	32	R/W
	- •	R4	ØxFFFF8684	32	R/W
	- •	R5	0x02000000	32	R/W
		R6	0x00256490	32	R/W
		R7	0x020230DC	32	R/W
	- •	R8	ØxFFFFFFFF	32	R/W
		R9	0x00000004	32	R/W
	- •	R10	0xFFD02000	32	R/W
		R11	0x02266594	32	R/W
	- •	R12	0x00000000	32	R/W
		SP	0x02266580	32	R/W
		LR	0x02000538	32	R/W
		PC	0x020008D4	32	R/W
	🕒 🕀	CPSR	0x600001D3	32	R/W
	🕒 🖻	IRQ			
	🕀 ն	FIQ			

【図 14.5】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 ff) 関数			
名前	値	サイズ	アクセス
🖶 🗁 l4wd1			
🖶 🗁 lwhps2fpgaregs			
🖶 🗁 mpul2			
🖶 🗁 mpuscu			
🖶 🗁 nandregs			
🖶 🥟 osc1timer0			
🕀 🧁 osc1timer1			
🕀 🗁 qspiregs			
🖶 🗁 rstmgr			
🖶 🗁 scanmgr			
🖽 🧁 sdmmc			
🖽 🗁 sdr			
🖽 🗁 spim0			
🖶 🗁 spim1			
🖶 🗁 spis0			
🖶 🗁 spis1			
🖶 🗁 sptimer0			
🖶 🗁 sptimer1			
🖶 🗁 stm			
🖶 🗁 sysmgr			
🖶 🗁 uart0			
🖶 🗁 uart1			
🖶 🗁 usb0			
🖶 🗁 usb1			
🖶 🗁 altera_avalon_sysid_sysid_qsys_control_slave	1		
🖶 🧁 altera_avalon_pio_led_pio_s1			
🖶 🧁 altera_avalon_pio_dip <i>s</i> w_pio_s1			
🖶 🧁 altera_avalon_pio_button_pio_s1			
🕀 🗁 altera_avalon_jtag_uart_jtag_uart_avalon_jtag_slave	J		

【図 14.6】FPGA 内のソフト IP レジスタ

※ 注記:

ブリッジを初期化していない状態で、FPGA 内のソフト IP レジスタを開かないでください。開くと Eclipse の接続が不安定になることがあります。

15. 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

15-1.SD カードの準備

- RocketBoards.org に公開されている SD カード・イメージ、または SoC EDS 付属の 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=SnJrEv17u0M
- (2) 「13-3. 生成した Preloader を SD カードに書き込むには」の手順に従って、SD カード内の Preloader を生成したものに入れ替えてください。

※ 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。

※ 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。

15-2.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、Helio-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: mrs r0, cpsr /* Interrupt disable */ orr r0, r0, #0xC0 /* Interrupt disable */ msr cpsr_cxsf, r0 /* Interrupt disable */
<pre>mrc p15, 0, r0, c1, c0, 2</pre>
mov r0,#0x40000000 /* Enable VFP/NEON Hardware */ mcr p10, 7, r0, cr8, cr0, 0 /* Write to fpexc */
D_START

【リスト 15-2.1】startup.s のソース・コードの例

(2) altera-socfpga-unhosted.ld

例として、Helio-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。

```
/*GROUP(-lgcc -lc -lcs3 -lcs3hosted -lcs3arm)*/
GROUP(-lgcc -lc -lcs3 -lcs3unhosted -lcs3arm)
/* PROVIDE(_cs3_reset = __cs3_reset_generic); */ /* オリジナルをコメント・アウト */
PROVIDE(_cs3_reset = __reset); /* 追加した startup.s の __reset を参照するように変更 */
```

【リスト 15-2.2】altera-socfpga-unhosted.ld リンカ・スクリプト・ファイル(変更箇所)

15-3.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、 Helio-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。

※ 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>#0BJ := \$ (patsubst %. c, %. o, \$ (C_SRC)) C_0BJ := \$ (patsubst %. c, %. o, \$ (C_SRC)) ASM_0BJ := \$ (patsubst %. s, %. o, \$ (ASM_SRC)) 0BJ := \$ (C_0BJ) \$ (ASM_0BJ)</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 \$@	# 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 15-3.1】Makefile の追記・修正箇所

15-4.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、Helio-Blinking-LED-Baremetal-GNU サンプル・プロジェクト内にある既存の helio_main.c ソース・ファイルを一部追記・修正します。
 - ※ 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) ł int i; int j; /* 追加 */ printf("Hello from Helio ¥n"); while(1) for (i=0; i < 8; 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; }

【リスト 15-4.1】helio_main.c ソース・ファイルの変更

15-5.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、Helio-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、helio_main.axf)が生成されます。

15-6.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。

- 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(15-6-1 節で説明)
- 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(15-6-2 節で説明)

15-6-1. SD カードの FAT 領域 にアプリケーションを配置して 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、Helio-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、helio_main.bin)を作成します。

\$ cd "C:¥DS-5 Workspace¥Helio-Blinking-LED-Baremetal-GNU" {
\$ arm-altera-eabi-objcopy -v -0 binary helio_main.axf helio_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、helio_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 をコンフィギュ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。u-boot のコンソール 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 helio_main.bin

(6) 上記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。上記コマンドをテキ 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 helio_main.bin; go 0x02000000;

【リスト 15-6-1.1】 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 🛽

※ 注記:

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、Helio-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。

15-6-2.SD カードの u-boot 領域 にアプリケーションを格納して 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。

- Embedded Command Shell から 前述 15-6-1. (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、helio_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、helio_main.img.bin)を作成します。

※ 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。

\$ mkimage -A arm -O u-boot -T standalone -C none -a 0x02000000 -e 0 -n "baremetal image" -d helio_main.bin helio_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、 helio_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。

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=helio_main.img.bin of=/dev/sdb3 bs=64k seek=4
```


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、 helio_main.img.bin)を書き込みます。

※ 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 helio_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 をコンフィギュ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。

改版履歴

Revision	年月	概要
1	2016 年 5 月	初版
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

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。