Arm® Development Studio

バージョン **2019.0**

スタートガイド

101469_1900_00_en

Arm® Development Studio

スタートガイド

リリース情報

本書には以下の変更が加え られています。

変更履歴

発行	日付	機密保持ステータス	変更点
1800-00	27 November 2018	非機密扱い	Arm® Development Studio 初版
1800-01	18 December 2018	非機密扱い	Arm® Development Studio 2018.0 改訂版 1
1800-02	31 January 2019	非機密扱い	Arm® Development Studio 2018.0 改訂版 2
1900-00	11 April 2019	非機密扱い	Arm® Development Studio 2019.0 改訂版

著作権

機密保持ステータス

この文書は機密事項ではありません。 この文書を使用、コピー、および開示する権利は、Arm およびこの文書を配布した当事者によって締結された契約の条項に従って、ライセンスの制限を受ける場合があります。

無制限アクセスは、Arm の内部分類です。

製品ステータス

この文書に記載されている情報は最終的な、すなわち開発された製品のためのものです。

Web アドレス

http://www.arm.com

目次
Arm® Development Studio スタートガイド

序章

本書について 8

Chapter 1 Arm® Development Studio の紹介

1.1 Arm® コンパイラ 1-11
1.2 Arm® デバッガ 1-12
1.3 デバッガプローブ 1-13
1.4 固定仮想プラットフォーム モデル 1-15
1.5 Arm Streamline パフォーマンスアナライザ 1-16
1.6 Mali™ グラフィックデバッガ 1-17

Chapter 2 インストール

2.1 ハードウェア及び host プラットフォーム 2-19
2.2 デバッガシステム 2-20
2.3 Windows へのインストール 2-21
2.4 Linux へのインストール 2-23
2.5 追加の Linux ライブラリ 2-24

Chapter 3 Arm® Development Studio のライセンス

3.1 セットアップを使用してライセンスを追加する 3-26
3.2 ライセンスの表示と管理 3-27

Chapter 4 統合開発環境の紹介

4.1 統合開発環境（IDE）の概要 4-30
4.2 IDE の使用方法 4-31
4.3 言語設定 4-35

Chapter 5 チュートリアル

5.1 チュートリアル: Hello World 5-37

Appendix A 用語とショートカット

A.1 用語 Appx-A-53
A.2 キーボードショートカット Appx-A-54

図の一覧
Arm® Development Studio スタートガイド

Figure 3-1 Adding a license in preferences dialog box. 3-27
Figure 3-2 Deleting a license in preferences dialog box. 3-28
Figure 4-1 IDE in the Development Studio perspective. 4-30
Figure 4-2 Workspace Launcher dialog box 4-31
Figure 4-3 Open Perspective dialog box 4-32
Figure 4-4 Adding a view in an area 4-33
Figure 4-5 Adding a view in |armds| 4-33
Figure 5-1 Screenshot of the IDE after creating a new project 5-39
Figure 5-2 Select VE_Cortex_A9x1 model 5-40
Figure 5-3 Edit configuration Connection tab 5-42
Figure 5-4 Edit configuration Files tab 5-43
Figure 5-5 Select helloworld.axf file 5-44
Figure 5-6 Debug from symbol main 5-45
Figure 5-7 Debug Control View 5-45
Figure 5-8 main () in code editor 5-46
Figure 5-9 Target console output 5-47
Figure 5-10 Commands view 5-47
Figure 5-11 Code Editor view 5-48
Figure 5-12 Disassembly view 5-48
Figure 5-13 Memory view 5-49
Figure 5-14 Disconnecting from a target using the Debug Control view 5-49
Figure 5-15 Disconnecting from a target using the Commands view 5-50

表の一覧
Arm® Development Studio スタートガイド

Table 2-1 Linux kernel version requirements 2-20

序章

この前書きでは『*Arm® Development Studio Getting Started Guide*』を紹介します。

このドキュメントは、次で構成されています。

- *本書について* page 8.

本書について

本書では、Arm®Development Studio を使い始める方法について説明します。 これは、Arm®Development Studio のインストールおよびライセンス発行のプロセスを案内し、Arm®Development Studio を初めて使用するときに発生する可能性があるいくつかの一般的なタスクを案内します。

本書の構成

本書は以下の章から構成されています。

Chapter 1 Arm® Development Studio の紹介
Arm Development Studio は、ベアメタル組み込みシステムおよび Linux ベースのシステム用のプロフェッショナルなソフトウェア開発ソリューションです。 パフォーマンス分析を含む、ブートコード、カーネルの移植からアプリケーションおよびベアメタルデバッグまで、開発のすべての段階を網羅しています。

Chapter 2 インストール
Arm Development Studio は、Windows および Linux オペレーティングシステムで利用できます。 この章では、インストール要件とインストールプロセスについて説明します。

Chapter 3 Arm® Development Studio のライセンス
Arm Development Studio は FlexNet ライセンス管理ソフトウェアを使用して、特定のエディションに対応する機能を有効にします。

Chapter 4 統合開発環境の紹介
Arm Development Studio の統合開発環境（IDE）は Eclipse ベースであり、Eclipse Foundation の Eclipse IDE と Arm ツールのコンパイルおよびデバッグ技術を組み合わせたものです。

Chapter 5 チュートリアル
Arm Development Studio を使い始めるのに役立つチュートリアルが含まれています。

Appendix A 用語とショートカット
Arm Development Studio の新規ユーザーのための補足情報です。

用語集

Arm®用語集は、Arm のマニュアルで使用されている用語の一覧とそれらの用語の定義です。 Arm の用語が一般に受け入れられている意味と異なる場合を除き、Arm の用語集には業界標準の用語は含まれていません。
詳細については、*Arm® Glossary* を参照してください。

表記規約

italic
特別な用語を紹介し、相互参照を示し、引用を示します。

bold
メニュー名などのインタフェース要素を強調表示します。 信号名を表します。 必要に応じて、説明的なリストの用語にも使用されます。

`monospace`
コマンド、ファイル名、プログラム名、ソースコードなど、キーボードから入力できるテキストを示します。

`monospace`
コマンドまたはオプションに使用できる省略形を示します。 完全なコマンド名またはオプション名の代わりに下線付きのテキストを入力できます。

`monospace italic`
引数が特定の値に置き換えられるモノスペーステキストへの引数を示します。

`monospace bold`
サンプルコードの外部で使用された場合、言語キーワードを表します。

`<and>`
コードまたはコードの一部に現れるアセンブラ構文の置き換え可能な用語を囲みます。
例：

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS
Arm®用語集で定義されている特定の技術的意味を持ついくつかの用語の本文での使用を表します。 例えば、 IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, UNPREDICTABLE などです。

フィードバック

この商品に関するフィードバック

この製品についてのご意見やご提案がありましたら、購入元にご連絡ください:

- 製品名.
- 製品リビジョンもしくはバージョン
- できるだけ多くの情報を提供する説明。 必要に応じて症状と診断手順を含めます。

コンテンツに関するフィードバック

内容に質問やコメントがある場合は下記の情報を添えて errata@arm.come にご連絡ください:

- 件名： *Arm Development Studio Getting Started Guide*.
- 101469_1900_00_en という番号.
- 該当する場合は、参照するページ番号
- あなたの質問やコメント

Arm は追加や改良に関する一般的な提案も歓迎します

Note

Arm は PDF を Adobe Acrobat および Acrobat Reader でのみテストします。他の PDF リーダーと組み合わせて使用した場合、表示される文書の品質を保証することはできません。

その他の情報

- *Arm® Developer*.
- *Arm® Information Center*.
- *Arm® Technical Support Knowledge Articles*.
- *Technical Support*.
- *Arm® Glossary*.

Chapter 1
Arm® Development Studio の紹介

Arm Development Studio は、ベアメタル組み込みシステムおよび Linux ベースのシステム用のプロフェッショナルなソフトウェア開発ソリューションです。 パフォーマンス分析を含む、ブートコード、カーネルの移植からアプリケーションおよびベアメタルデバッグまで、開発のすべての段階を網羅しています。

含まれているもの:

Arm コンパイラ 5 と Arm コンパイラ 6 の ツールチェーン.
組み込みアプリケーションとベアメタル組み込みアプリケーションをビルドできます。

Arm デバッガ.
Arm プロセッサベースのターゲットおよび Fixed Virtual Platform（FVP）ターゲットでのソフトウェア開発をサポートするグラフィカルデバッガ。

Fixed Virtual Platform (FVP) targets.
アーキテクチャ Armv6-M、Armv7-A / R / M、および Armv8-A / R / M のシングルコアおよびマルチコアシミュレーションモデル。 これにより、ハードウェアなしでソフトウェアを開発することができます。

Streamline.
サンプリングデータとシステムトレースを視覚的形式と統計的形式の両方でデータを表示するレポートに変換できるグラフィカルパフォーマンス分析ツール

Mali™ グラフィックデバッガ.
Mali グラフィックデバッガを使用すると、グラフィック開発者はアプリケーション内で OpenGL ES、Vulkan、および OpenCL API の呼び出しをトレースできます。

Arm Development Studio ツールを使い始めるのに役立つ、専用の例、アプリケーション、およびサポート資料。
いくつかの他社製コンパイラは Arm Development Studio と互換性があります。 たとえば、GNU Compiler ツールを使用すると、Arm ターゲット用にベアメタル、Linux カーネル、および Linux アプリケーションをコンパイルできます。

以下のセクションから構成されています。

- *1.1 Arm® コンパイラ* page 1-11.
- *1.2 Arm® デバッガ* page 1-12.
- *1.3 デバッグプローブ* page 1-13.
- 1.4 *固定仮想プラットフォームモデル* page 1-15.
- *1.5 Arm Streamline パフォーマンスアナライザ* page 1-16.
- *1.6 Mali™ グラフィックデバッガ* page 1-17.

1.1 Arm® コンパイラ

Arm コンパイラツールを使用すると、ベアメタル組み込みシステムに適したアプリケーションとライブラリを構築できます。
Arm Development Studio は、組み込みアプリケーションとベアメタル組み込みアプリケーションをコンパイルするための 2 つのバージョンの Arm コンパイラを提供します。

- Arm コンパイラ 5 - Armv4 から Armv7 までの全ての Arm アーキテクチャをサポートします。

Note

Armv4 より前の全てのアーキテクチャは廃止されており、
Arm コンパイラ 5 ではサポートされていません。

- Arm コンパイラ 6 - Armv6-M、Armv7 および Armv8 アーキテクチャをサポートします。

Arm コンパイラ 6 は *SVE architectural extensions.*もサポートしています。
このツールチェーンは、High Performance Computing（HPC）用の SVE アーキテクチャ拡張を備えた Armv8-A をターゲットとしたソフトウェア開発にお勧めします。

Arm Development Studio IDE 内またはコマンドラインからコンパイラを実行できます。

Note

Arm コンパイラで利用できる機能は、個々のライセンスの種類によって異なります。

たとえば、ライセンスは:

- Arm コンパイラの使用を特定のプロセッサに限定します
- 作成できるイメージのサイズに上限を設けます

Arm Development Studio のフルのライセンスを購入することで、Arm コンパイラの追加機能を有効にすることができます。 詳細についてはツール販売店にお問い合わせください。

関連情報

Add a compiler to Arm Development Studio

1.2 Arm® デバッガ

Arm デバッガは、Arm Development Studio IDE またはコマンドラインを使用してアクセスでき、Arm プロセッサベースのターゲットおよび FVP（Fixed Virtual Platform）ターゲットでのソフトウェア開発をサポートしています。

IDE を介して Arm Debugger を使用すると、次のような包括的で直感的なビューで、ベアメタルおよび Linux アプリケーションをデバッグできます:

- Synchronized と逆アセンブリ
- スタック
- メモリ
- レジスタ
- 関数
- 変数
- モジュール
- ブレークポイント
- トレース

デバッグコントロールビューを使用すると、ソースレベルまたは命令レベルでアプリケーションをシングルステップ実行したり、コードが実行されると他のビューが更新されたりすることを確認できます。ブレークポイントまたはウォッチポイントを設定するとアプリケーションが停止し、アプリケーションの動作を調べることができます。 ターゲットでサポートされている場合は、このビューを使用して、イベントのシーケンスを示すタイムラインを使用してアプリケーション内の関数実行を追跡することもできます。

Arm DS Command Prompt コマンドラインコンソールを使用してデバッグすることもできます。これにより、スクリプトを介したデバッグおよびトレースアクティビティの自動化が可能になります。

関連情報

Debug control view
Overview of Arm Debugger

1.3 デバッグプローブ

Arm Development Studio は、さまざまなデバッグアダプタと接続をサポートしています。

デバッグアダプタ

デバッグアダプタは複雑さと機能が異なります。 Arm Development Studio でこれらを使用すると、次のような高度なデバッグ機能が提供されます:

- レジスタの `read/write`
- ブレークポイントの設定
- メモリからの `read`
- メモリへの `write`

サポートされている Arm デバッグアダプタは次のとおりです:

- Arm DSTREAM.
- Arm DSTREAM-ST.
- Keil® ULINK™ 2.
- Keil® ULINK pro.
- Keil® ULINK pro D.

デバッグ接続

デバッグ接続により、デバッガはさまざまなターゲットをデバッグできます.

サポートされているデバッグ接続は次のとおりです:

- CADI (モデル用のデバッグ インタフェース).
- gdbserver へのイーサネット接続.
- CMSIS-DAP.
- DTS adviceLUNA (JTAG ICE).
- Altera USB-Blaster II.

  Note

  Arm デバッガは、Altera USB-Blaster および USB-Blaster II デバッグユニットを使用して、Altera Arria V SoC、Arria 10 SoC、Cyclone V SoC および Stratix 10 ボードに接続できます。
  接続を有効にするには、環境変数 `QUARTUS_ROOTDIR` が設定され、Altera Quartus ツールのインストールディレクトリへのパスが含まれていることを確認します:
  — Windows では、この環境変数は通常 Quartus ツールのインストーラによって設定されます。
  — Linux では、環境変数を Altera Quartus ツールのインストールパスに手動で設定する必要があります。 例えば、 `~/altera/13.0/qprogrammer`.です。
  USB-Blaster および USB-Blaster II 用のデバイスドライバのインストールについては、Altera Quartus ツールのドキュメントを参照してください。

ハードウェア構成のデバッグ

Arm Development Studio のデバッグハードウェア構成ビューを使用して、開発ターゲットとワークステーション間のインタフェースを提供するデバッグハードウェアアダプタを更新および構成します。

Arm Development Studio は以下のビューを提供します:

- **デバッグハードウェア設定 IP**
  このビューを使用して、デバッグハードウェアアダプタの *IP アドレスの設定* をします
- **デバッグハードウェアファームウェアインストーラ**
  このビューを使用して、デバッグハードウェアアダプタの *ファームウェアのアップデート* をします

Note

これらのビューは DSTREAM ファミリのデバイスのみをサポートします。

1.4 固定仮想プラットフォームモデル

固定仮想プラットフォーム(FVPs)は、プロセッサ、メモリ、周辺機器など、Arm システムの完全なシミュレーションです。 FVP ターゲットは、プログラマの視点から、ソフトウェアを構築およびテストするための包括的なモデルを提供します。

FVP を使用する場合、絶対的なタイミング精度は、高速シミュレートされた実行速度を達成するために犠牲にされます。 つまり、モデルを使用してソフトウェアの機能を確認することはできますが、サイクル数の正確さ、低レベルのコンポーネント間のやり取り、またはその他のハードウェア固有の動作に頼ってはいけません。

Arm Development Studio は Cortex®ファミリの様々なプロセッサをカバーする複数の FVPs を提供しています。 CADI を実装する他のさまざまな Arm およびサードパーティのシミュレーションモデルに接続することもできます。

Arm Development Studio には、SVE アーキテクチャ拡張をサポートする Armv8-A FVP 実行可能ファイルが含まれています。実行ファイルは、`<install_directory>\bin\...`にあります。これらを使用して、コマンドラインまたは Arm Development Studio IDE 内からアプリケーションを実行できます。

関連情報

About the Component Architecture Debug Interface (CADI)

1.5 Arm Streamline パフォーマンスアナライザ

Arm Streamline パフォーマンスアナライザは、グラフィカルなパフォーマンス分析ツールです。 サンプリングデータ、命令トレース、およびシステムトレースを、視覚的形式と統計的形式の両方でデータを表示するレポートに変換できます。

Streamline は、ハードウェアパフォーマンスカウンタとカーネルメトリックスを使用して、システムリソースを正確に表現します。

1.6 Mali™ グラフィックデバッガ

Mali グラフィックデバッガは OpenGL® ES、EGL™、OpenCL™ 、 Vulkan™ の開発者が API レベルで分析することによってアプリケーションを最大限に活用できるようにするためのツールです。

Mali グラフィックデバッガを使用すると、開発者はアプリケーション内で OpenGL ES、Vulkan、および OpenCL の API 呼び出しをトレースし、フレームごとのアプリケーションへの影響を把握して、考えられる問題を特定することができます。Mali ベースのシステムの改善のための推奨事項として、API の誤用の試行が強調されています。 トレース情報は、1 つのシステム上のファイルに取り込まれ、後で分析することもできます。 基礎となる GPU サブシステムの状態は、いつでも観察可能です。

Chapter 2
インストール

Arm Development Studio は Windows および Linux オペレーティングシステムで利用できます。この章では、インストール要件とインストールプロセスについて説明します。

以下のセクションから構成されています。

- *2.1 ハードウェア及び host プラットフォーム* page 2-19.
- *2.2 デバッガシステム* page 2-20.
- *2.3 Windows へのインストール* page 2-21.
- *2.4 Linux へのインストール* page 2-23.
- *2.5 追加の Linux ライブラリ* page 2-24.

2.1 ハードウェア及び host プラットフォーム

Arm Development Studio を最大限に活用するには、ハードウェアとホストプラットフォームが最小要件を満たす必要があります。

ハードウェア

Arm Development Studio をインストールして使用するには、
ワークステーションに少なくとも以下のものが必要です:

- デュアルコア x86 2GHz プロセッサ (または同等のもの)
- 2GB の RAM
- 約 3GB のハードディスク空き容量

パフォーマンスを向上させるために、Arm は以下の場合に最低 4GB の RAM を推奨します :

- 大きなイメージのデバッグ
- シミュレートされた大きなメモリマップを持つモデルの使用
- Streamline の使用

Hos プラットフォーム

Arm Development Studio は、以下のホストプラットフォームをサポートしています:

- Windows 7 SP1 Professional Edition
- Windows 7 SP1 Enterprise Edition
- Windows 10
- Red Hat Enterprise Linux 6 Workstation
- Red Hat Enterprise Linux 7 Workstation
- Ubuntu Desktop Edition 16.04 LTS
- Ubuntu Desktop Edition 18.04 LTS

Note

Arm Development Studio は 64 ビットホストプラットフォームのみをサポートします。

Arm® コンパイラ host プラットフォーム

Arm Development Studio には、最新バージョンの Arm コンパイラ 5 および Arm コンパイラ 6 が含まれています。各コンパイラバージョンのリリースノートには、ホストプラットフォームの互換性に関する情報が記載されています:

- *Arm Compiler 5*
- *Arm Compiler 6*

Arm Development Studio に Arm コンパイラ 5 および Arm コンパイラ 6 の他のバージョンを追加する方法については、*register a compiler toolchain* を参照してください。

2.2 デバッガシステム

ベアメタルおよび Linux ターゲットをデバッグするときは、追加のソフトウェアとハードウェアが必要です。

ベアメタル

ベアメタルターゲットを Arm Development Studio に接続するにはデバッグユニットが必要です。

Arm Development Studio は、以下のデバッグユニットをサポートしています:

- DSTREAM
- DSTREAM-ST
- ULINK pro
- ULINK pro D
- ULINK 2

Linux アプリケーションと Linux カーネル

Linux アプリケーションのデバッグには、ターゲットに gdbserver バージョン 7.0 以降が必要です。

gdbserver に加えて、特定のアーキテクチャーおよびデバッグ機能には、最低限の Linux カーネルバージョン要件があります。 これを以下の表に示します:

Table 2-1 Linux kernel version requirements

アーキテクチャまたはデバッグ機能	最小 Arm Linux カーネルバージョン
Arm デバッガでのデバッグ	2.6.28
対称型マルチプロセッシング (SMP) システムでのアプリケーションデバッグ	2.6.36
VFP 及び Arm Neon™ レジスタへのアクセス	2.6.30
Streamline	3.4

ファームウェアアップデートの管理

- DSTREAM の場合は *debug hardware firmware installer view* を使用してファームウェアを確認し必要に応じて更新します。更新されたファームウェアは `<install_directory>/sw/debughw/firmware` にあります。
- Arm デバッガで ULINK 2 デバッグプローブを使用するには、CMSIS-DAP 互換ファームウェアでアップグレードする必要があります。 Windows では, the `UL2_Upgrade.exe` プログラムを使用して ULINK 2 ユニットをアップグレードできます。 プログラムと指示は `<install_directory>/sw/debughw/ULINK2` にあります。
- ULINK pro および ULINK pro D の場合、Arm Development Studio がファームウェアのインストールを管理します。

2.3 Windows へのインストール

Arm Development Studio をインストールする方法は 2 つあります。インストールウィザードとコマンドラインのどちらでも使用できます。

Note

Windows プラットフォームには複数のバージョンの Arm Development Studio をインストールできます。 これを行うには、異なるルートインストールディレクトリを使用する必要があります。

このセクションには、以下のサブセクションがあります:

- *2.3.1 インストールウィザードを使用する* page 2-21.
- *2.3.2 コマンドラインを使用する* page 2-21.

2.3.1 インストールウィザードを使用する

インストールウィザードを使用して Windows に Arm Development Studio をインストールするには、次の手順に従います・

前提条件

- Arm Development Studio インストールパッケージを*ダウンロード*してください。

手順

1. ダウンロードした `.zip` ファイルを解凍します
2. 解凍場所から **armds-<version>.exe** を実行します
3. 画面の指示に従います

Note

- インストール中に、デバイスドライバをインストールするように求められる場合があります。これらのドライバをインストールすることをお勧めします。DSTREAM と Energy Probe ハードウェアユニットへの USB 接続が可能になります。またシミュレーションモデルのネットワーキングをサポートします。 これらのドライバはこれらの機能を使用するために必要です。
- ドライバをインストールすると、ドライバソフトウェアに関する警告が表示されることがあります。 あなたは安全にこれらの警告を無視することができます。

2.3.2 コマンドラインを使用する

コマンドラインを使用して Windows に Arm Development Studio をインストールするには、次の手順に従います。

前提条件

- Arm Development Studio インストールパッケージを*ダウンロード*してください。
- コマンドラインからインストールするには、あなたのマシンの管理者権限が必要です。

手順

1. 管理者権限でコマンドプロンプトを開きます
2. Microsoft のインストーラ `msiexec.exe`. を実行します

Note

- **msiexec** の引数として`.msi` ファイルの場所を指定する必要があります。
- **msiexec** オプションの全リストを表示するには、**msiexec /?** をコマンドラインから実行します。

Example 2-1 Examples

msiexec を使用して Arm Development Studio をインストールする方法の例は次のとおりです:

```
msiexec.exe /i <installer_locationdatainstall.msi> EULA=1 /qn /l*v install.log
```

コマンド	定義
`/i`	インストールを実行します。
`<installer_locationdatainstall.msi>`	インストールする `.msi` ファイルのフルパス名を指定します。
`/EULA=1`	これは Arm 特有のオプションです。使用許諾契約書（EULA）に同意するには、EULA を 1 に設定します。コマンドラインで EULA を受け入れる前に EULA を読む必要があります。これは、GUI インストーラ、インストールファイル、または Arm Development Studio のダウンロードページにあります。
`/qn`	クワイエットモードを指定します。 インストールはユーザーの操作を必要としません。 **Note** デバイスドライバのインストールにはユーザーの操作が必要です。 USB ドライバを必要としない場合、またはユーザが USB ドライバを操作しないようにインストールしたい場合は、コマンドラインで SKIP_DRIVERS=1 オプションを使用します。
`/l*v<install.log>`	インストールからのすべての出力を表示するためのログファイルを指定します。

2.4 Linux へのインストール

Linux に Arm Development Studio の 1 つ以上のバージョンをインストールできます。

Arm Development Studio を Linux にインストールするには、インストールパッケージを *ダウンロード*した後に次の手順に従います。

Note

プラットフォームには、複数のバージョンの Arm Development Studio をインストールできます。 これを行うには、異なるルートインストールディレクトリを使用する必要があります。

手順

1. `<install_location>/sw/dependency_check/dependency_check_linux-x86_64.sh` に移動します。 `dependency_check_linux-x86_64.sh` スクリプトを実行してください。このスクリプトは、続行する前にインストールする必要がある不足している依存関係の一覧を出力します。

   Note

   64 ビットインストーラによってインストールされたツールは、32 ビットシステムライブラリに依存します。 Arm Development Studio ツールは実行に失敗する可能性があります。または 32 ビット互換ライブラリがインストールされていないと、欠落ライブラリに関するエラーを報告する可能性があります。 ライブラリの一覧は*追加の Linux ライブラリ* page 2-24 にあります。

2. `armds-<version>.sh` を実行します。 画面の指示に従います。

次のステップ

デバイスドライバとデスクトップショートカットはオプション機能です。 デバイスドライバを使用すると、USB 接続で DSTREAM などのハードウェアユニットをデバッグできます。 サポートされている Linux プラットフォームでは、デスクトップメニューは *http://www.freedesktop.org/*メニューシステムを使用して作成されます。

root 権限を使用してインストール後にオプション機能をインストールするには、次のコマンドを実行します:

```
run_post_install_for_Arm_DS_IDE_<version>.sh
```

このスクリプトはインストールディレクトリにあります。

Note

`suite_exec` を使用して、`Arm Development Studio` 用に環境変数を正しく構成します。たとえば、`<install_directory>/bin/suite_exec <shell>`を実行して、PATH およびその他の環境変数が正しく設定されたシェルを開きます。さらにヘルプを表示するには引数無しで `suite_exec` を実行します。

2.5 追加の Linux ライブラリ

Arm Development Studio を Linux にインストールするには、システムにインストールされていない可能性がある追加のライブラリをいくつかインストールする必要があります。

インストールが必要な特定のライブラリは、実行している Linux のディストリビューションによって異なります。`dependency_check_linux-x86_64.sh` スクリプトは、インストールする必要があるライブラリを識別します。詳しくは *Linux へのインストール* page 2-23 を参照してください。

Note

必要なライブラリがインストールされていないと、一部の Arm Development Studio ツールは実行に失敗する可能性があります。

次のようなエラーメッセージが表示されることがあります:

- armcc: No such file or directory
- arm-linux-gnueabihf-gcc: error while loading shared libraries: libstdc++.so.6: cannot open shared object file: No such file or directory

必要なライブラリ

Arm Development Studio は、以下のライブラリに依存しています:

- `libasound.so.2`
- `libatk-1.0.so.0`
- `libc.so.6 *`
- `libcairo.so.2`
- `libfontconfig.so.1`
- `libfreetype.so.6`
- `libgcc_s.so.1 *`
- `libGL.so.1`
- `libGLU.so.1`
- `libgthread-2.0.so.0`
- `libgtk-x11-2.0.so.0`
- `libncurses.so.5`
- `libnsl.so.1`
- `libstdc++.so.6 *`
- `libusb-0.1.so.4`
- `libX11.so.6`
- `libXext.so.6`
- `libXi.so.6`
- `libXrender.so.1`
- `libXt.so.6`
- `libXtst.so.6`
- `libz.so.1 *`

Note

64 ビットインストールでは、アスタリスクが付いているライブラリには追加の 32 ビット互換ライブラリが必要です。

一部のコンポーネントは、ブラウザライブラリを使用してレンダリングすることもできます。 Arm は、すべてのコンポーネントが正しくレンダリングされるように、これらのライブラリのいずれかをインストールすることをお勧めします:

- `libwebkit-1.0.so.2`
- `libwebkitgtk-1.0.so.0`
- `libxpcom.so`

Chapter 3
Arm® Development Studio のライセンス

Arm Development Studio は FlexNet ライセンス管理ソフトウェアを使用して、特定のエディションに対応する機能を有効にします。

Arm Development Studio のエディションを比較するには、 *Compare editions* を参照してください。

以下のセクションから構成されています。

- *3.1 セットアップを使用してライセンスを追加する* page 3-26.
- *3.2 ライセンスの表示と管理* page 3-27.

3.1 セットアップを使用してライセンスを追加する

初めて Arm Development Studio を開くと**製品セットアップ** ダイアログボックスが開き、ライセンスを追加するように求められます。

前提条件

Arm Development Studio を購入した場合は、ライセンスサーバのアドレスとポート番号、またはライセンスファイルが必要です。

手順

1. **製品ライセンスの追加** を選択して**次へ(N)>**をクリックします
2. ライセンスを追加してください:
   - ライセンスサーバの場合は、 **ライセンスサーバ** を選択し、ライセンスサーバのアドレスとポート番号を入力します。`<port number>@<server address>.` の形式で指定し、**次へ(N)>**をクリックします
   - ライセンスファイルの場合は、 **ライセンスファイル、参照...** の順に選択して、ワークステーションからファイルを選択します
     次へ(N)>をクリックします
   - 評価ライセンスの場合:
     1. **評価ライセンスの取得**を選択して**次へ(N)>**.をクリックします
     2. Arm アカウントに ログインして**次へ(N)>**をクリックします
     3. ネットワークインタフェイスを選択して**終了(F)** をクリックします。
        評価ライセンスが生成されます
3. アクティブにする製品を選択して**終了(F)** をクリックします

3.2 ライセンスの表示と管理

Arm Development Studio 内でライセンス情報を表示するには、**ヘルプ(H)** > **Arm License Manager** の順に選択します。

このセクションには、以下のサブセクションがあります:

- *3.2.1 ライセンスを追加する* page 3-27.
- *3.2.2 ライセンスを削除する* page 3-28.

3.2.1 ライセンスを追加する

Arm License Manager を使用して Arm Development Studio にライセンスを追加できます。

前提条件

ライセンスサーバのアドレスとポート番号、またはライセンスファイルが必要です。

手順

1. ライセンス情報を表示するには、**ヘルプ(H)** > **Arm License Manager** の順にクリックします
2. **加算**をクリックして**製品セットアップ**ウィザードを開きます

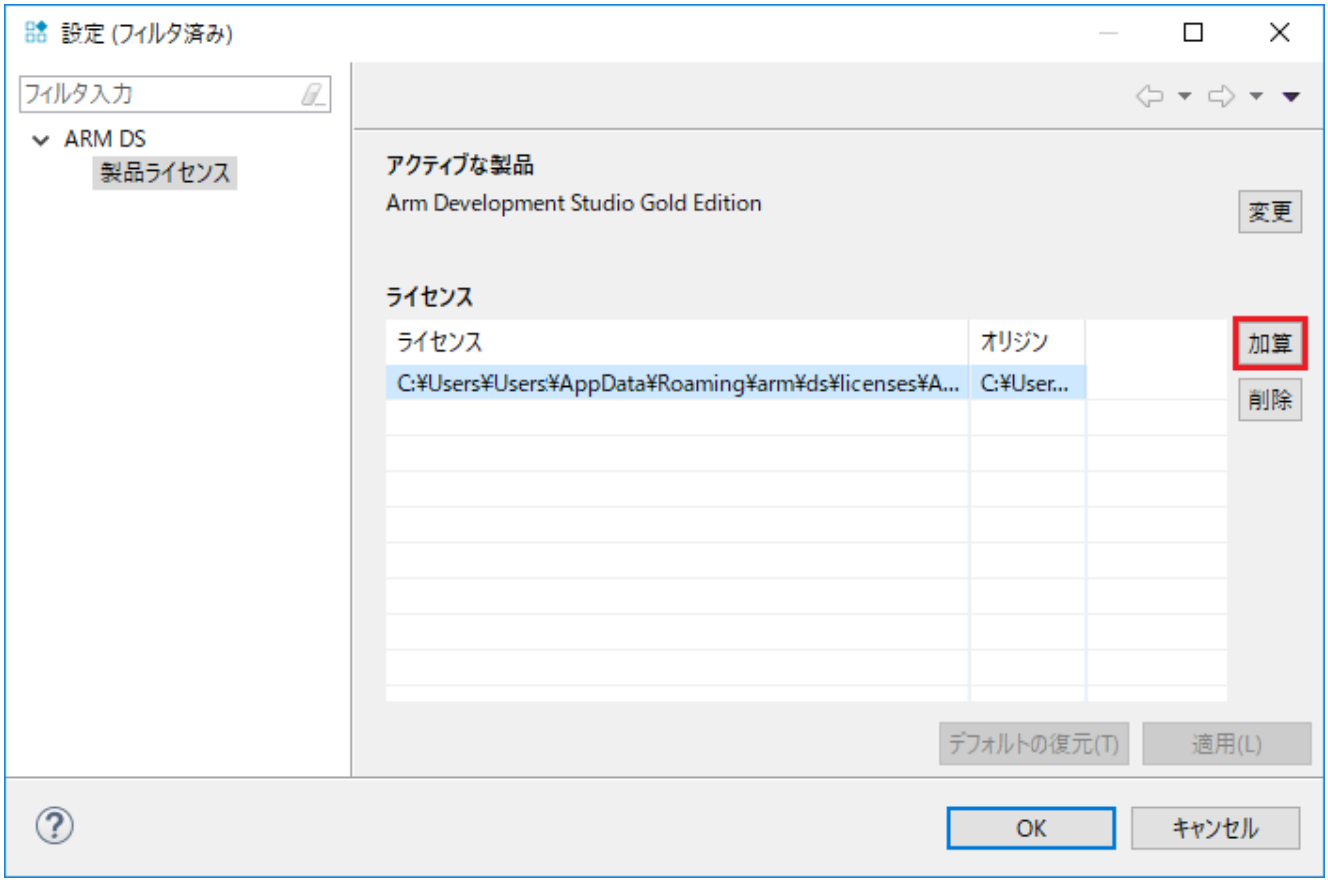


FIGURE 3-1 Adding a license in preferences dialog box.

3. ライセンスを追加するには *3.1 セットアップを使用してライセンスを追加する* page 3-26. の手順に従ってください

関連タスク

3.1 セットアップを使用してライセンスを追加する page 3-26.

3.2.2 ライセンスを削除する

Arm License manager を使用して、Arm Development Studio から不要なライセンスを削除できます。

手順

1. ライセンス情報を表示するには、**ヘルプ(H)**> **Arm License Manager** の順にクリックします
2. 削除するライセンスを選択して**削除**を選択します

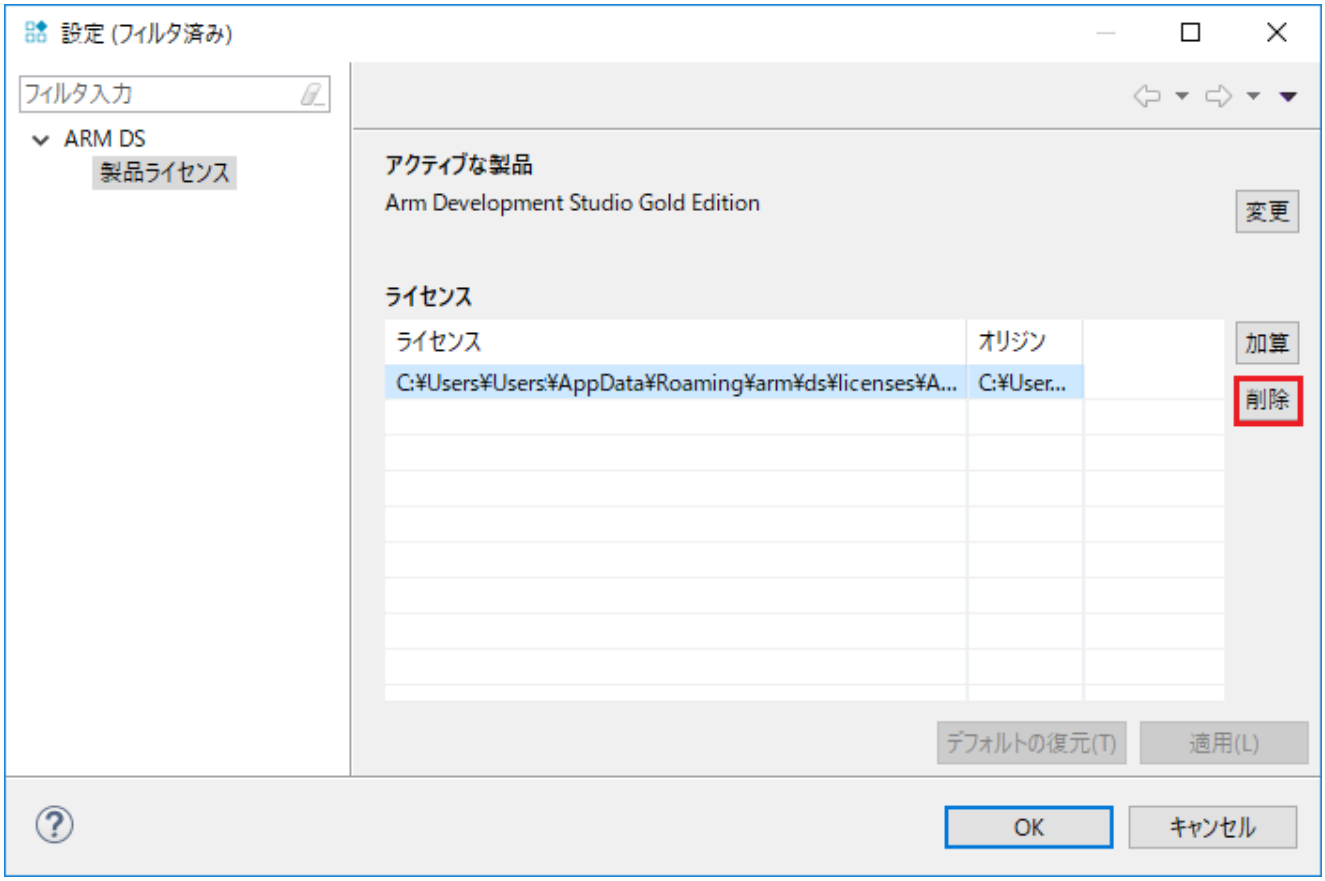


Figure 3-2 Deleting a license in preferences dialog box.

Chapter 4
統合開発環境の紹介

Arm Development Studio の統合開発環境（IDE）は Eclipse ベースであり、Eclipse Foundation 製の Eclipse IDE と Arm ツールのコンパイルおよびデバッグ技術を組み合わせたものです。

含まれるもの:

Project Explorer

project explorer を使用すると、ファイルやプロジェクトへの依存関係の追加や削除、プロジェクトのインポート、エクスポート、作成、ビルドオプションの管理など、さまざまなプロジェクトタスクを実行できます。

エディター

エディタを使用すると、C / C ++または Arm アセンブリ言語のソースファイルを読み書きしたり、修正したりできます。

パースペクティブとビュー

パースペクティブは、特定の種類の環境に合わせてカスタマイズされたビュー、メニュー、およびツールバーを提供します。Arm Development Studio はデフォルトで **Development Studio** パースペクティブを使用します。 パースペクティブを切り替えるには、メインメニューから**ウィンドウ(W)** > **Perspective** を選択します。

以下のセクションから構成されています。

- *4.1 統合開発環境（IDE）の概要* page 4-30.
- *4.2 IDE の使用方法* page 4-31.
- *4.3 言語設定* page 4-35.

4.1 統合開発環境（IDE）の概要

IDE には、特定のパースペクティブに関連付けられている一連のビューが含まれています

Arm Development Studio は、デフォルトとして **Development Studio** パースペクティブを使用します。

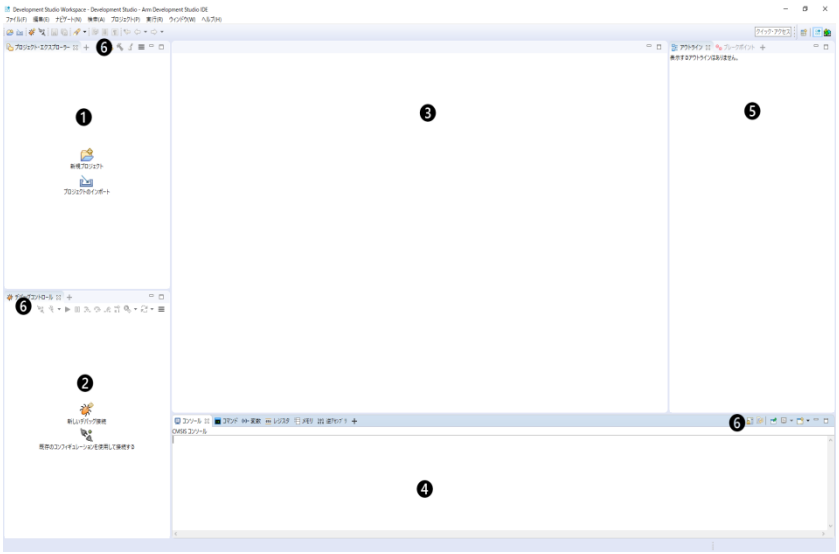


Figure 4-1 IDE in the Development Studio perspective.

1. Project Explorer - このビューを使用して、プロジェクトを作成、構築、および管理します。
2. デバッグコントロール - このビューを使用して、デバッグ接続を作成および制御します。
3. エディターウィンドウ - このビューを使用して、ソースコードファイルの内容を表示および変更します。 エディタ領域のタブには、現在編集用に開かれているファイルが表示されます。
4. デバッグビュー - Arm Debugger に固有のビュー。 複数のビューを 1 つの領域に重ねてタブを作成することができます。
5. 追加のデバッグビューを含む領域。 すべての領域は、任意のビューを含むように完全にカスタマイズ可能です。
6. メニューとツールバー - メインメニューとツールバーは IDE ウィンドウの上部にあります。 特定の機能に関連する他のツールバーは、各パースペクティブまたはビューの上部にあります。

終了時に、設定は自動的に保存されます。 次に Arm Development Studio を開くと、ウィンドウは同じパースペクティブとビューに戻ります。

ビューの詳細については、ビュー内をクリックして F1 キーを押し、**ヘルプ**ビューを開いてください。

IDE をカスタマイズする

レイアウト、キー割り当て、ファイルの関連付け、配色を変更して IDE をカスタマイズできます。 これらの設定は**ウィンドウ(W)** > **設定(P)** にあります。変更はワークスペースに保存されます。 別のワークスペースを選択した場合、これらの設定は異なる可能性があります。

4.2 IDE の使用方法

Arm Development Studio IDE はカスタマイズできます。 このセクションの指示に従って、表示できるビューを選択することができます。

このセクションには、以下のサブセクションがあります:

- *4.2.1 デフォルトのワークスペースを変更する* page 4-31.
- *4.2.2 パースペクティブの切り替え* page 4-32.
- *4.2.3 ビューを追加する* page 4-32.

4.2.1 デフォルトのワークスペースを変更する

ワークスペースは、プロジェクトに関連するファイルやフォルダ、および IDE 設定を保存するためのファイルシステム上の領域です。 Arm Development Studio が初めて起動すると、デフォルトのワークスペースが `C:\Users\<user>\Development Studio Workspace`. に自動的に作成されます。

Note

プロジェクト専用のワークスペースフォルダを選択することをお勧めします。 プロジェクトに関連しないリソースを含む既存のフォルダを選択した場合、Arm Development Studio からそれらにアクセスすることはできません。 これらのリソースは、後でプロジェクトを作成およびビルドするときにも競合を引き起こす可能性があります。

Arm Development Studio は最後に使用したワークスペースに自動的に開きます。

手順

1. **File** > **Switch Workspace** > **Other...** を選択します。
2. The **Eclipse Launcher** ダイアログボックスが開きます。

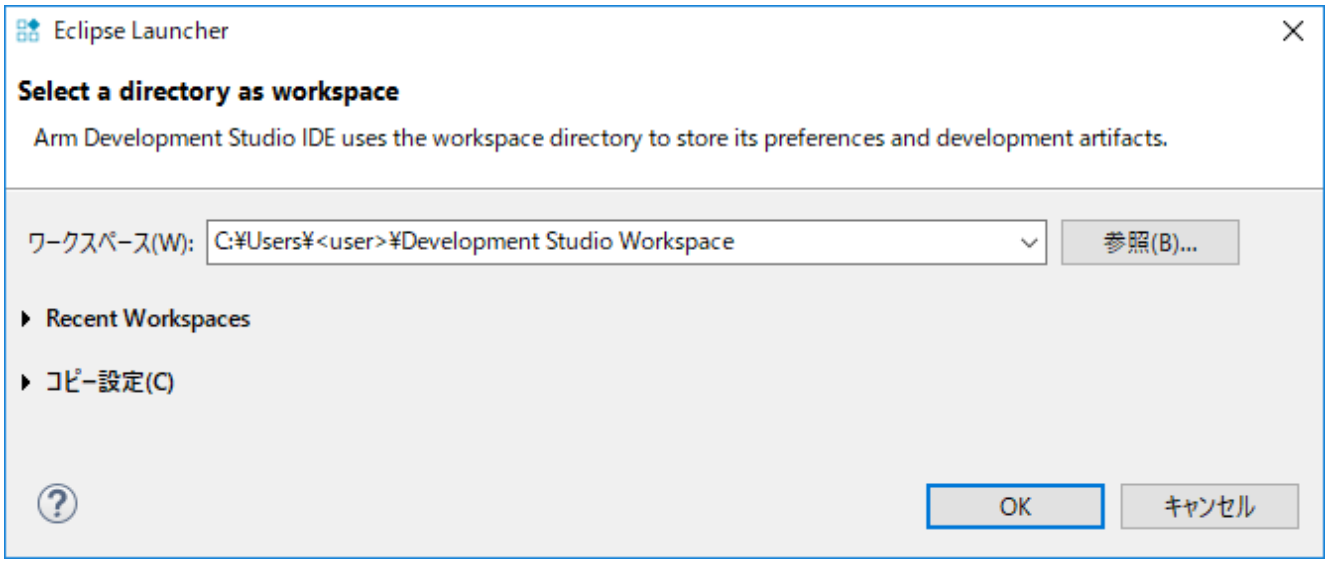


Figure 4-2 Workspace Launcher dialog box

2. **Browse…** をクリックしてワークスペースを選択し **OK**.をクリックします。

Arm Development Studio が新しいワークスペースで再起動します。

4.2.2 パースペクティブの切り替え

パースペクティブは、Arm Development Studio IDE で選択したビューとエディターのレイアウトを定義します。 各パースペクティブには、それぞれ独自のメニューとツールバーが関連付けられています。

手順

1. **ウィンドウ(W)** > **Perspective** > **パースペクティブを開く(O)** > **その他(O)...** 選択します。
   これにより **パースペクティブを開く**ダイアログボックスが開きます。
2. 開くパースペクティブを選択して、**OK** をクリックします。

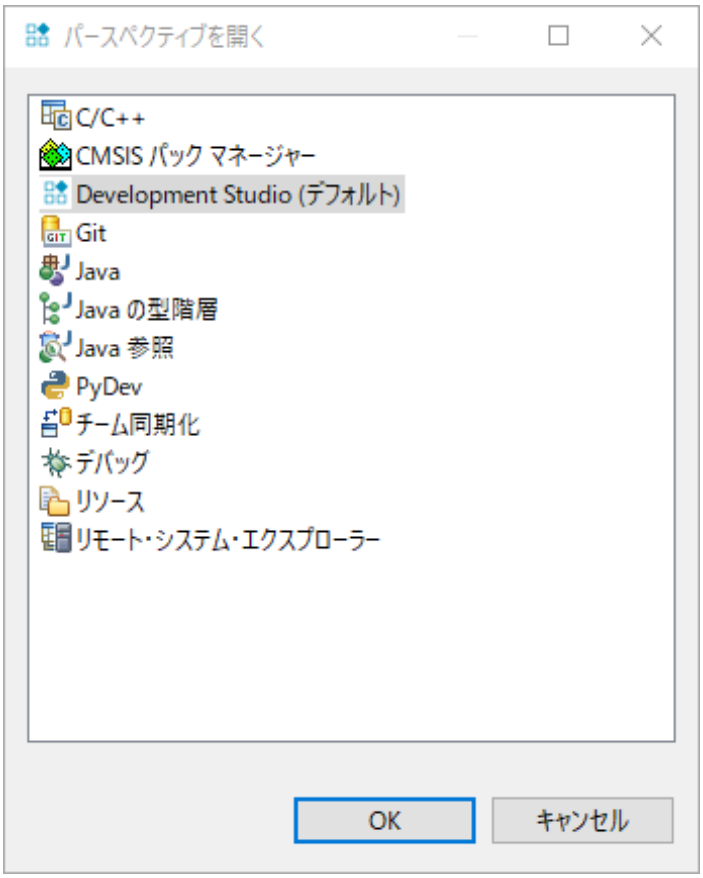


Figure 4-3 Open Perspective dialog box

ワークスペースにパースペクティブが開きます。

関連情報

Arm Debugger perspectives and views

4.2.3 ビューを追加する

ビューは、アクティブなデバッグ接続に対応して、特定の機能に関する情報を提供します。 各パースペクティブには一連のデフォルトビューがあります。 ワークスペースをカスタマイズするために、ビューを追加、削除、または再配置することができます。

手順

1. ビューを追加したいエリアの+ボタンをクリックします。

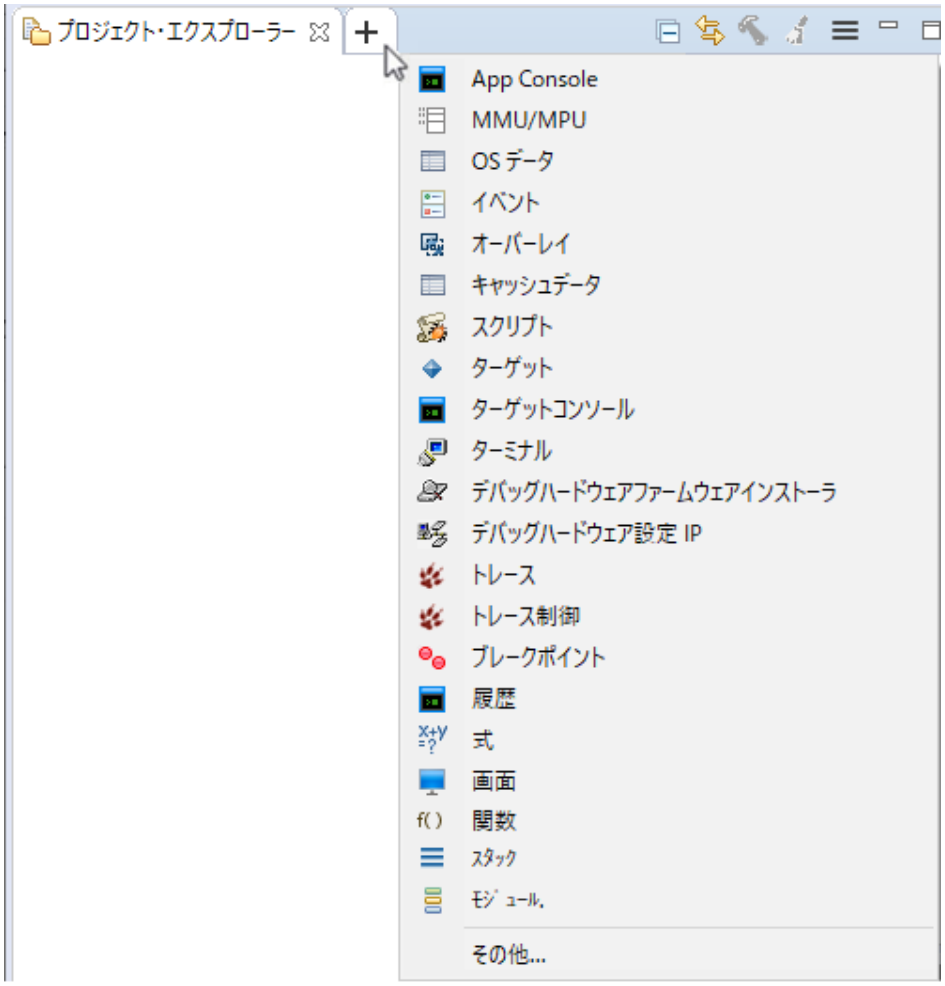


Figure 4-4 Adding a view in an area

2. 追加するビューを選択するか **その他...** をクリックして**ビューの表示**ダイアログボックスを開き使用可能なビューの一覧を確認します。.

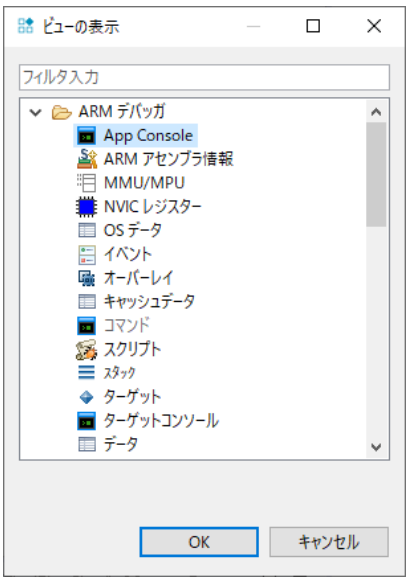


Figure 4-5 Adding a view in |armds|

3. 開くビューを選択して、 **OK**. をクリックします

選択した領域にビューが開きます。

関連情報

Arm Debugger perspectives and views

4.3 言語設定

IDE の一部の機能は、異なる言語に翻訳されています。 Eclipse Web サイトから言語パックをダウンロードできます。

異なる言語パックで IDE を起動するには、2 つの方法があります:

- 使用する言語でオペレーティングシステムが実行されている場合、IDE は翻訳された機能を自動的に表示します。
- オペレーティングシステムが使用する言語で実行されていない場合は、IDE の起動時に-nl コマンドライン引数を指定する必要があります。
  たとえば、日本語の言語パックを使用するには、次のように入力します:

```
armds_ide -nl ja
```

Note

Arm コンパイラ 6 はソースコードで日本語の文字をサポートしていません。

Chapter 5
チュートリアル

Arm Development Studio の使用を開始するのに役立つチュートリアルが含まれています。

以下のセクションから構成されています。

- *5.1 チュートリアル:Hello World* page 5-37.

5.1 チュートリアル: Hello World

Hello World チュートリアルは新規ユーザー向けで、最初のプロジェクトを立ち上げて実行するための各ステップを紹介します。

このセクションには、次のサブセクションが含まれます:

- *5.1.1 Arm® Development Studio を初めて開く* page 5-37.
- *5.1.2 C/C++ でプロジェクトを作成する* page 5-38.
- *5.1.3 プロジェクトを構成する* page 5-39.
- *5.1.4 プロジェクトをビルドする* page 5-40.
- *5.1.5 デバッグセッションを構成する* page 5-40.
- *5.1.6 Arm Debugger を使用したアプリケーションデバッグ* page 5-46.
- *5.1.7 ターゲットの切断* page 5-49.

5.1.1 Arm® Development Studio を初めて開く

Arm Development Studio を初めて開くと、ライセンスの詳細を追加するように求められます。このセクションのタスクを完了すると、Arm デバッガを使用する準備が整います。

Arm Development Studio は *Linux と Windows プラットフォーム* page 2-19.の両方で使用できます。

前提条件

- 次のいずれかの Arm Development Studio をダウンロードしてインストールします:
  — Linux: *Linux へのインストール* page 2-23.
  — Windows: *Windows へのインストール* page 2-21.
- Arm Development Studio を購入した場合、ライセンスファイル、または接続するライセンスサーバのアドレスとポート番号のいずれかが必要です。

手順

1. Arm Development Studio を開きます:
   - Windows では、**Windows menu** > **Arm Development Studio <version>** を選択します
   - Linux では:
     — GUI: Linux バリアントのメニューシステムを使用して、Arm Development Studio を見つけます。
     — コマンドライン: `<installation_directory>/bin/armds_ide` を実行します
2. Arm Development Studio を初めて開くと **製品セットアップ** ダイアログが開き、製品ライセンスを追加するように求められます。 次のいずれかを実行できます:
   - **製品ライセンスの追加** - Arm Development Studio を購入した場合はこのオプションを選択します。
   - **評価ライセンスの取得** - 製品を評価する場合は、このオプションを選択します。
3. **次へ(N)>**をクリックします。
4. **製品ライセンスの追加**を選択した場合:
   a. ライセンスファイルの場所、またはライセンスサーバのアドレスとポート番号を入力し、**次へ(N)>**をクリックします。
   b. 使用する資格がある Arm Development Studio のエディションがリストされます。必要なエディションを選択し、**次へ(N)>**をクリックします。
   c. 概要ページで詳細を確認してください。 正しい場合は、**終了(F)**をクリックします。
5. **評価ライセンスの取得**を選択した場合:
   a. Arm Developer アカウントのメールアドレスとパスワードを使用して、Developer アカウントにログインします。アカウントがない場合は、**アカウントを作成**をクリックします。
   b. ライセンスをロックするネットワークインタフェイスを選択します。.

c. **終了(F)**をクリックします。

Arm Development Studio が開きます。ユーザーインタフェイスの主な機能については、*IDE の使用方法* page 4-31 を参照してください。

Note

ワークスペースは、デフォルトで次のいずれかに自動的に設定されます:

- Windows: `<userhome>\Development Studio Workspace`
- Linux: `<userhome>/developmentstudio-workspace`

ファイル(F) > **ワークスペースの切り替え(W)**を選択して、デフォルトの場所を変更できます。

5.1.2 C/C++でプロジェクトを作成する

Arm Development Studio をインストールしてライセンスを取得したら、簡単な Hello World C プロジェクトを作成し、ターゲットのベース RAM アドレスを指定する方法を示します。このチュートリアルの残りの部分では、Arm Compiler 6 ツールチェーンを使用し、ターゲットは Arm Development Studio で提供される Cortex®-A9 固定仮想プラットフォームです。

前提条件

Arm® Development Studio を初めて開く page 5-37.を完了します。

Procedure

1. 新しい C プロジェクトを作成します:
   a. 開いているプロジェクトがない場合は、**プロジェクト・エクスプローラー**ビューで**新規プロジェクト**をクリックします。
   b. それ以外の場合は、 **ファイル(F)** > **新規(N)** > **プロジェクト(R)...** を選択します。
2. **C/C++**メニューを展開し、**C プロジェクト**を選択して**次へ(N)>**をクリックします。
3. **C プロジェクト** ダイアログで:
   a. **プロジェクト名(P)**フィールドに `HelloWorld` と入力します。
   b. **プロジェクトの種類**で**実行可能** > **Hello World ANSI C Project** を選択します。
   c. **ツールチェーン**で **Arm Compiler 6** を選択します。
   d. **終了(F)**をクリックします。

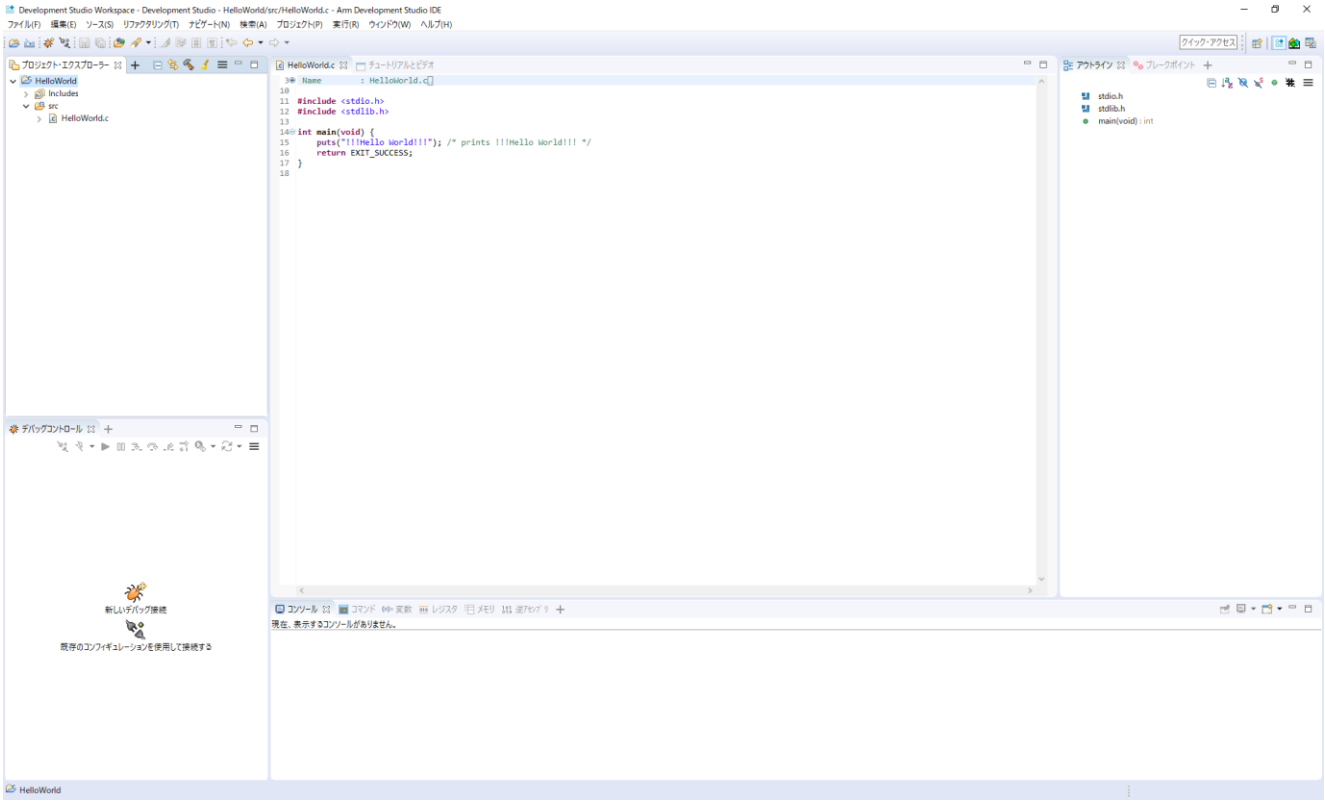


Figure 5-1 Screenshot of the IDE after creating a new project

5.1.3 プロジェクトを構成する

HelloWorld プロジェクトをビルドする前に、FVP ターゲットの RAM ベースアドレスをリンカに伝える必要があります。 これにより、アプリケーションがビルドされ、ターゲットに正しくロードされます。 また、イメージファイルにデバッグシンボルを追加するように Arm Debugger に指示する必要があります。これにより、イメージをデバッグできます。

前提条件

C/C++ でプロジェクトを作成する page 5-38.を完了します。

手順

1. **プロジェクト・エクスプローラー**ビューで、`HelloWorld` プロジェクトを右クリックし **プロパティ(R)** を選択します。
   プロパティ:HelloWorld ダイアログボックスが開きます。
2. イメージファイルにデバッグシンボルを追加します:
   a. **C/C++ ビルド**を展開し、**ビルド変数**を選択します。
   b. **構成** を **Debug [アクティブ]**に選択します。
3. リンカのベース RAM アドレスを **C/C++ ビルド**を展開した**設定**を選択して設定します:
   a. **ツール設定** > **すべてのツールの設定** > **ターゲット**を選択します。
   b. **ターゲット CPU** ドロップダウンから、**Cortex-A9** を選択します。
   c. **ターゲット FPU** ドロップダウンから、**No FPU** を選択します。
   d. **ツール設定** > **Arm リンカ 6** > **イメージレイアウト**を選択します。
   e. **RO ベースアドレス**フィールドに、`0x8000000` を入力します。
4. **OK** をクリックします。
5. インデックスを再構築するように求められたら、**Yes** をクリックします。

5.1.4 プロジェクトをビルドする

これで HelloWorld プロジェクトをビルドできます！

前提条件

次のタスクを完了します:

- *C/C++ でプロジェクトを作成する* page 5-38.
- *プロジェクトを構成する* page 5-39.

手順

1. **プロジェクト・エクスプローラー**ビューで、HelloWorld プロジェクトを右クリックし **プロジェクトのビルド(B)**を選択します。

プロジェクトがビルドされたら、**プロジェクト・エクスプローラー**ビューの **Debug** の下で、HelloWorld.axf ファイルを見つけます。

.axf ファイルには、Arm Debugger がソースレベルのデバッグを実行できるようにするオブジェクトコードとデバッグシンボルが含まれています。

Note

デバッグシンボルはビルド時に追加されます。Arm Compiler 6 でコンパイルするときに-g オプションを使用して、これを手動で指定できます、またこれをデフォルトの動作に設定できます。詳細については*プロジェクトを構成する* page 5-39.を参照してください。

5.1.5 デバッグセッションを構成する

Arm Development Studio では、**新しいデバッグ接続** ウィザードを使用してターゲットへのデバッグ接続を作成することにより、デバッグセッションを構成します。

要件に応じて、ハードウェア、Linux アプリケーション、またはモデルへの接続を構成できます:

- *ハードウェア接続*は、ハードウェア上でアプリケーションを直接実行およびデバッグするための接続を構成するためのものです。
- *Linux アプリケーション接続*は、ターゲット上で実行されている Linux アプリケーションをデバッグするための接続を構成するためのものです。
- *モデル接続* は、CADI 準拠モデルでアプリケーションを直接実行およびデバッグするための接続を構成するためのものです。

次の例では、このチュートリアルの前のセクションで作成したプロジェクトを使用して、Cortex®-A9 固定仮想プラットフォーム（FVP）への**モデル接続** 構成します。

手順

1. メインメニューから、**ファイル(F)**>**新規(N)** > **モデル接続**を選択します
2. **モデル接続**ダイアログボックスで、接続の詳細を指定します:
   a. デバッグ接続名に **HelloWorld** などを入力します
   b. **デバッグ接続を既存のプロジェクトに関連付けます**オプションを選択し、 前のセクション*プロジェクトをビルドする* page 5-40.で作成およびビルドしたプロジェクトを選択します。
   c. **次へ(N)>**をクリックします。
3. **ターゲット選択**ダイアログボックスで、ターゲットの詳細を指定します:
   a. **Arm FVP (Installed with Arm DS)** > **VE_Cortex_A9x1** を選択します。

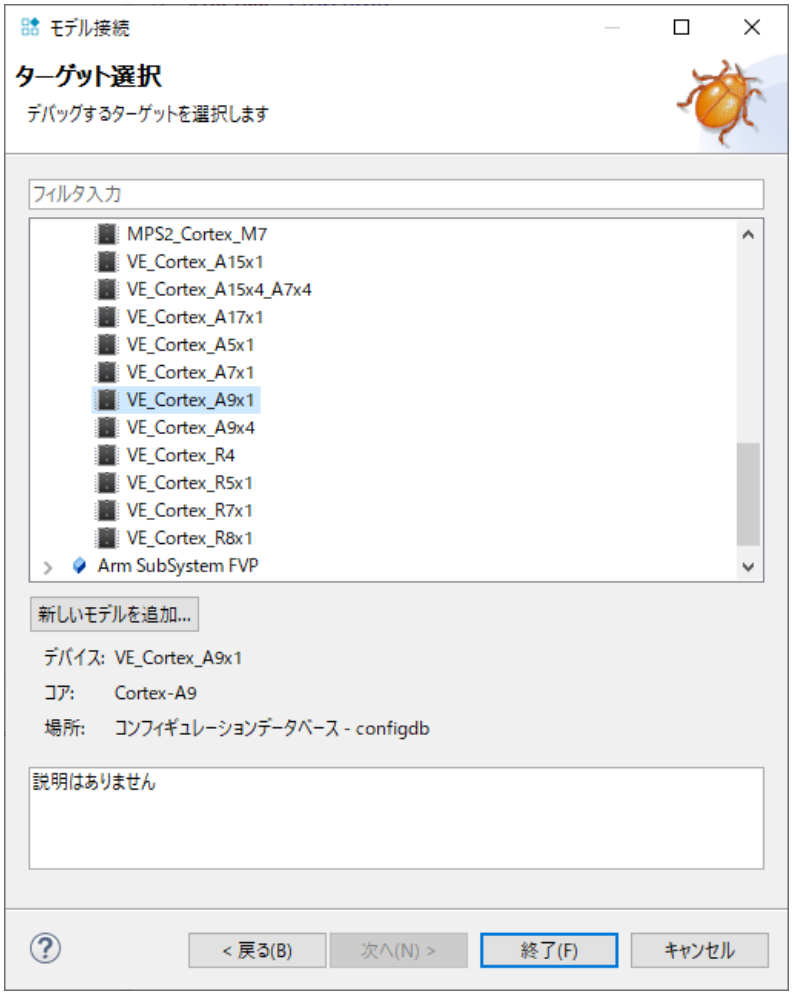


Figure 5-2 Select VE_Cortex_A9x1 model

b. **終了(F)**をクリックします。

4. **構成の編集**ダイアログで、正しいターゲットが選択されていること、適切なアプリケーションファイルが指定されていること、およびデバッガがどこからデバッグを開始するかを確認します:

   a. **接続**タブで、**Arm FVP (Installed with Arm DS)** > **VE_Cortex_A9x1** > **Bare Metal Debug** > **Debug Cortex-A9** が選択されていることを確認します。

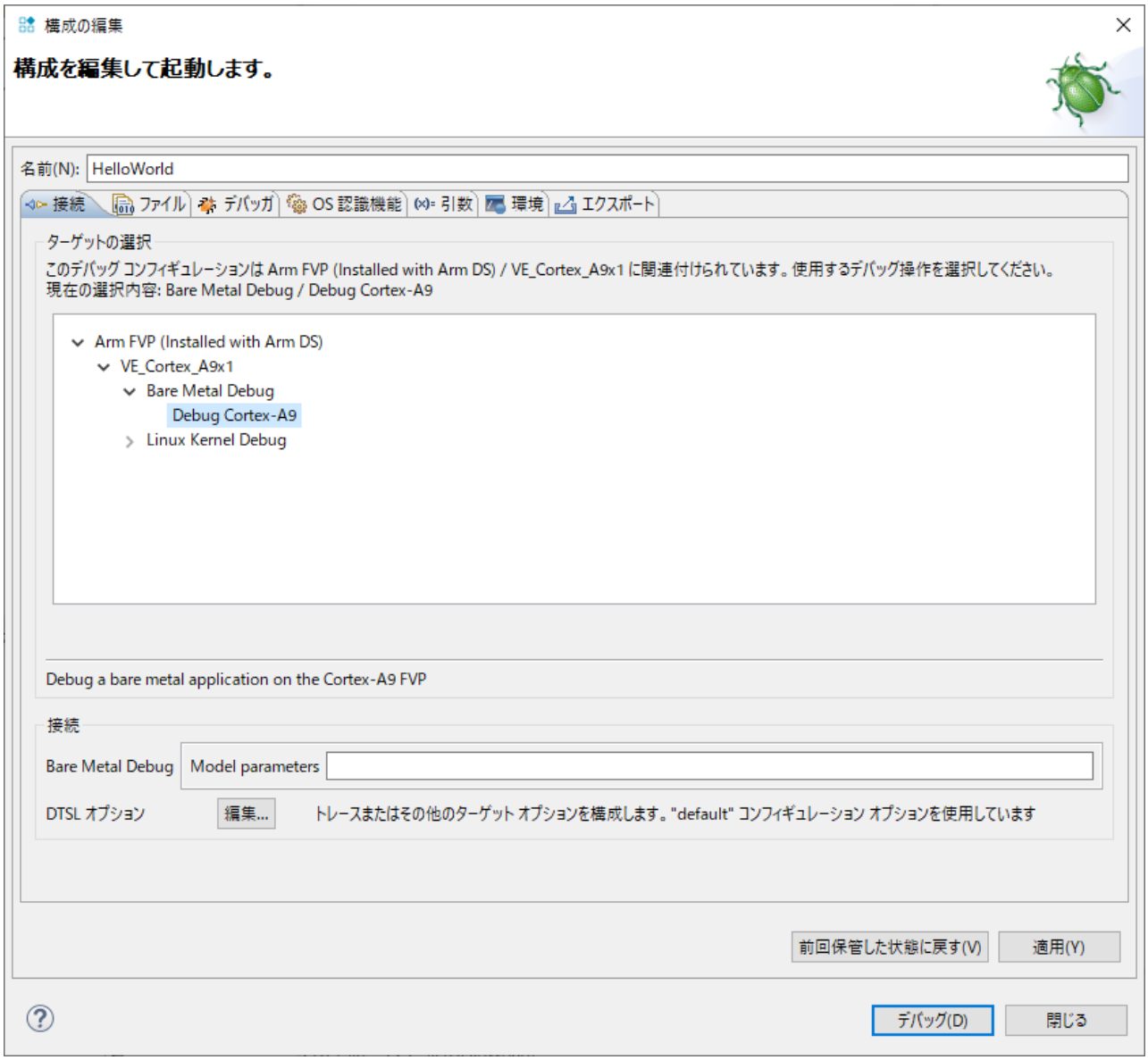


Figure 5-3 Edit configuration Connection tab

b. **ファイル**タブで、**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** > **ワークスペース...** を選択します

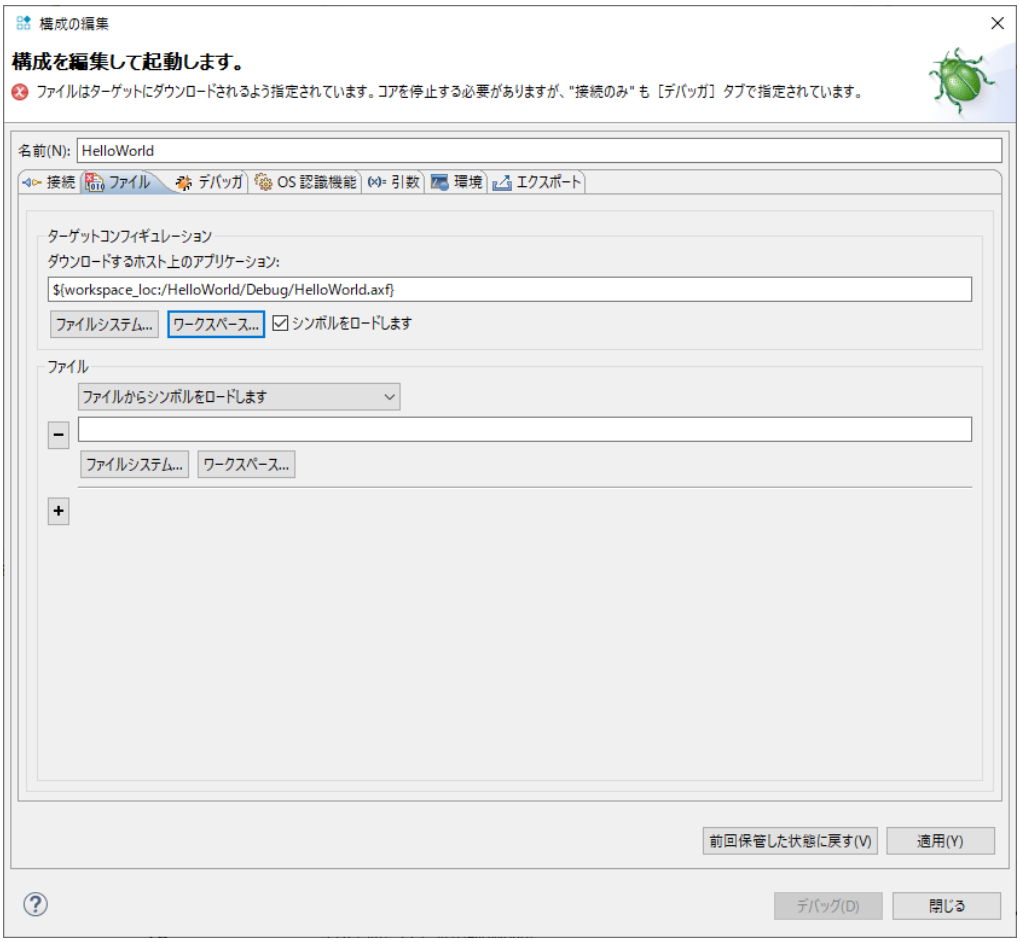


Figure 5-4 Edit configuration Files tab

c. **HelloWorld** プロジェクトをクリックして展開し、**Debug** フォルダーから `HelloWorld.axf` を選択して **OK** をクリックします。

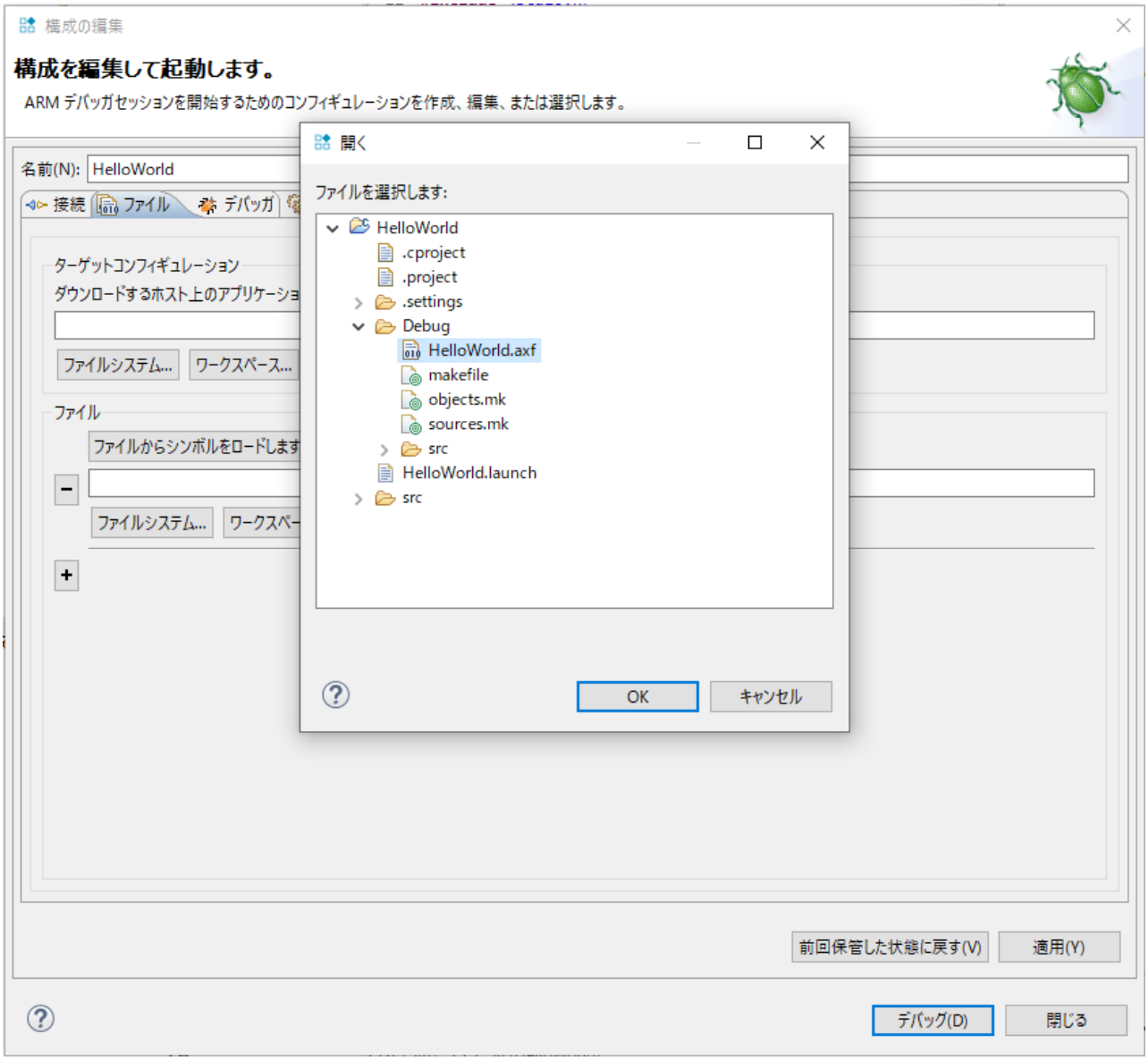


Figure 5-5 Select helloworld.axf file

d. **デバッガ**タブで、**シンボルからデバッグします**を選択します。

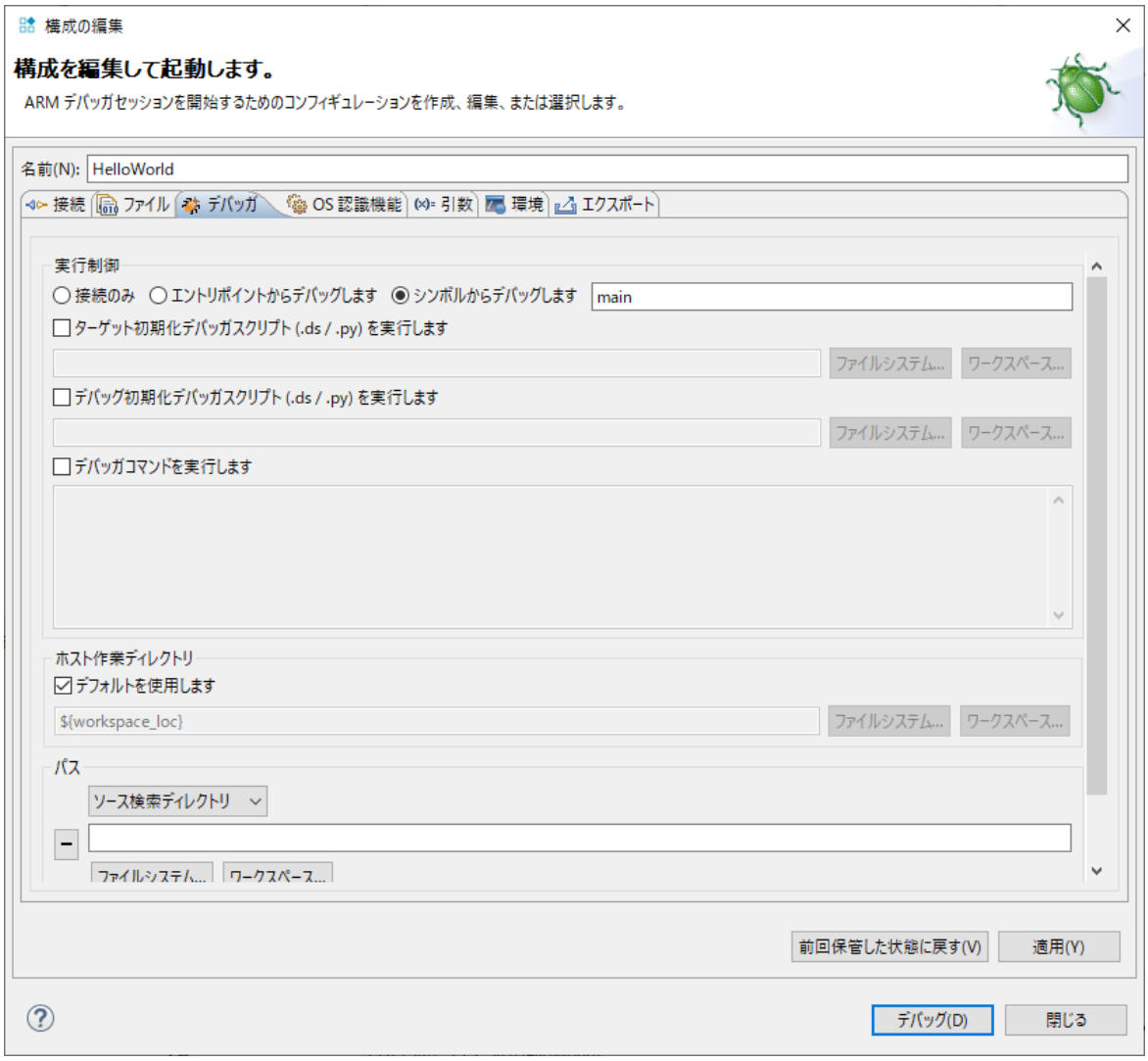


Figure 5-6 Debug from symbol main

5. **デバッグ(D)**をクリックして、アプリケーションをターゲットにロードし、デバッグ情報をデバッガにロードします。

Arm Development Studio がモデルに接続し、**デバッグコントロール**ビューに接続ステータスを表示します。

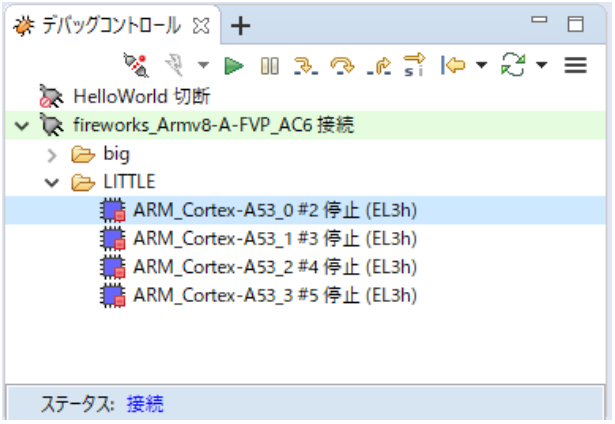


Figure 5-7 Debug Control View

アプリケーションはターゲットにロードされ、main()関数で停止し、実行準備が整います。

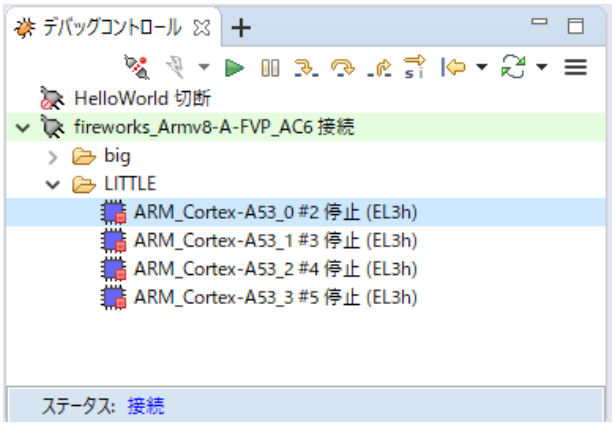
Figure 5-8 main () in code editor

5.1.6 Arm Debugger を使用したアプリケーションデバッグ

デバッグ構成を作成し、アプリケーションがターゲットにロードされたので、デバッグを開始してアプリケーションをステップスルーします。

アプリケーションの実行とステップ実行

デバッグコントロールビューにあるコントロールを使用して、アプリケーションをデバッグします。 デフォルトでは、これらのコントロールはソースレベルのステップ実行を行います。



 - クリックして、ターゲットにアプリケーションをロードした後、アプリケーションの実行を続けます。

 - クリックして、コードの実行を中断または一時停止します。

 - クリックしてコードをステップ実行します。

 - クリックしてソース行をステップオーバーします。

 - クリックしてステップアウトします。

 - これは切り替えです。 これをクリックして、ステップ実行手順とステップ実行ソースコードを切り替えます。 これは、上記のステップコントロールに適用されます。

他のビューには、デバッグ接続に関連する情報が表示されます

- **ターゲットコンソール**ビューには、アプリケーションの出力が表示されます。

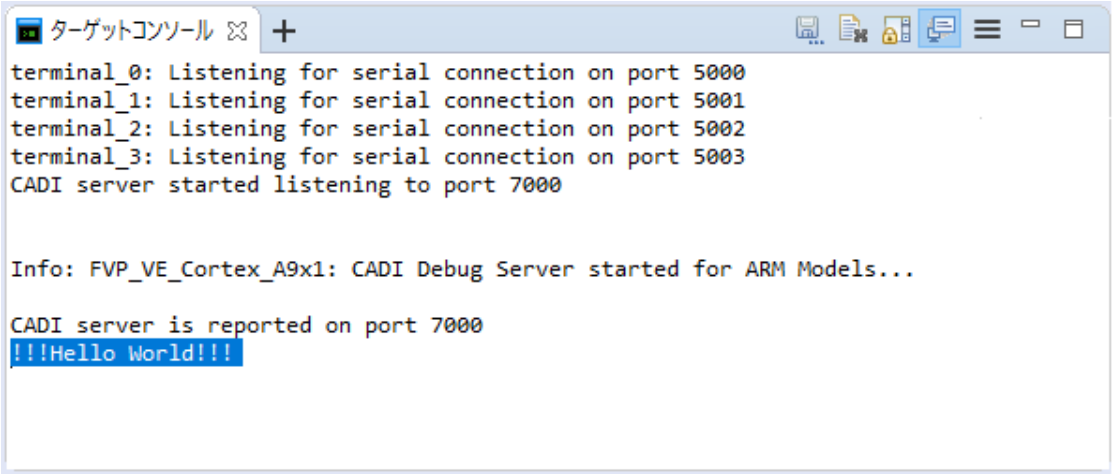


Figure 5-9 Target console output

- **コマンド**ビューには、デバッガによって出力されたメッセージが表示されます。
  また、このビューを使用して、Arm Debugger コマンドを入力します。

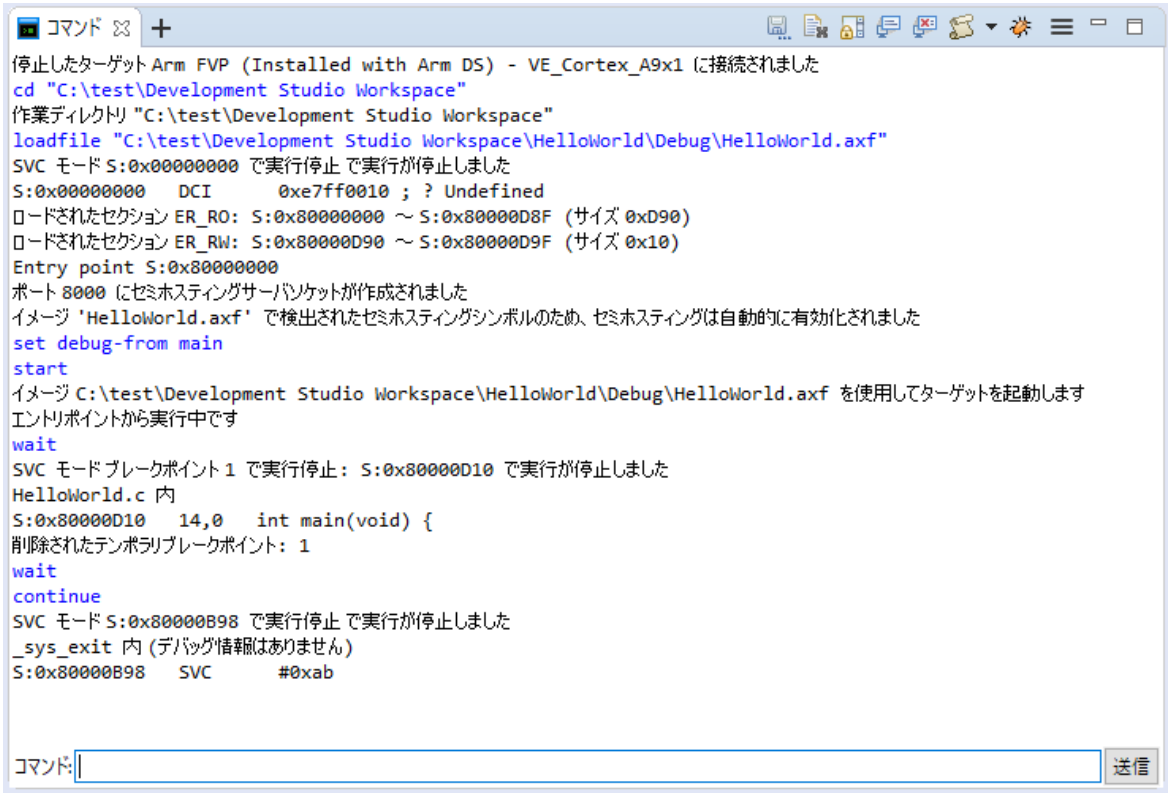


Figure 5-10 Commands view

- C / C ++ Editor ビューには、アクティブな C、C ++、または Makefile が表示されます。
  これらのファイルを編集すると、ビューが更新されます。

Figure 5-11 Code Editor view

- **逆アセンブリ**ビューには、ビルドされたプログラムがアセンブリ命令として表示され、そのメモリの場所が表示されます。

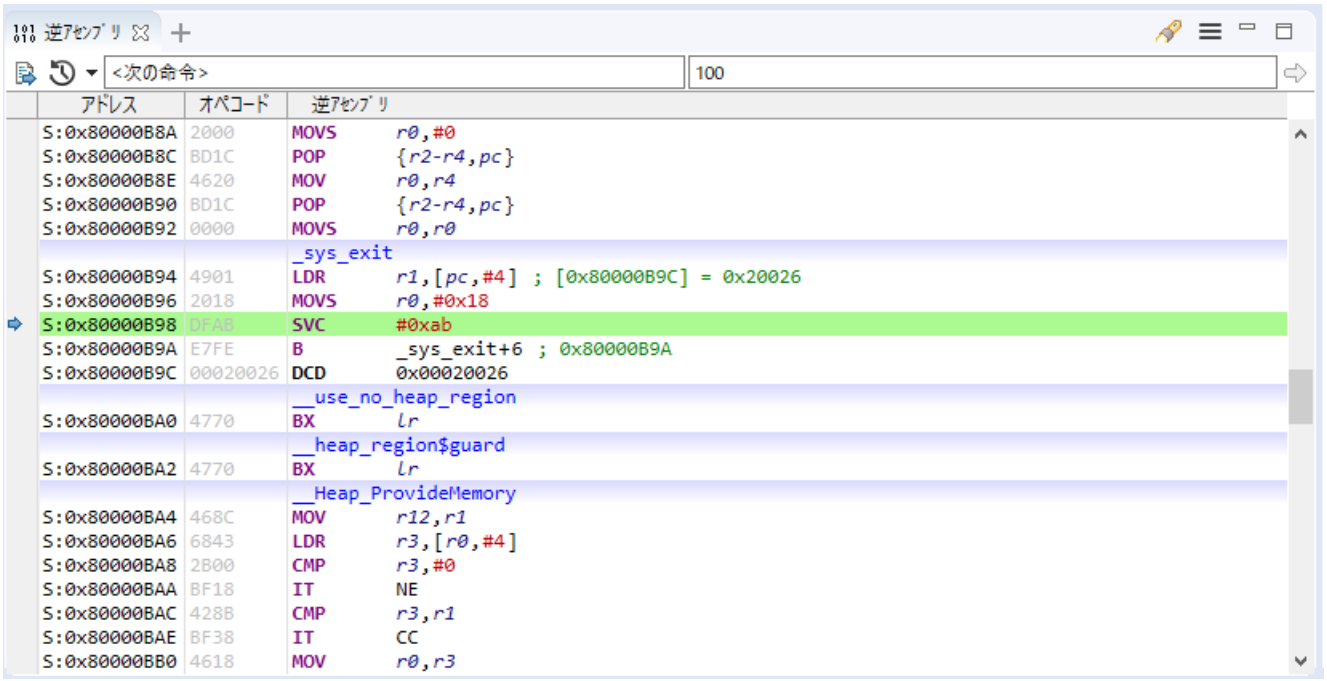


Figure 5-12 Disassembly view

プログラムが停止するコード内の場所を示します。 この場合、それは `main()` 関数にあります。

- **メモリ**ビューには、ターゲットメモリでコードがどのように表示されるかが表示されます。たとえば、アプリケーションの文字列 Hello World がメモリでどのように表されるかを表示するには、次のようにします:
  1. **メモリ**ビューを開きます。
  2. **アドレス**フィールドに`&main` と入力し、キーボードの **Enter** キーを押します。 ビューには、ターゲットのメモリの内容が表示されます。
  3. Hello World という単語を選択して強調表示します。

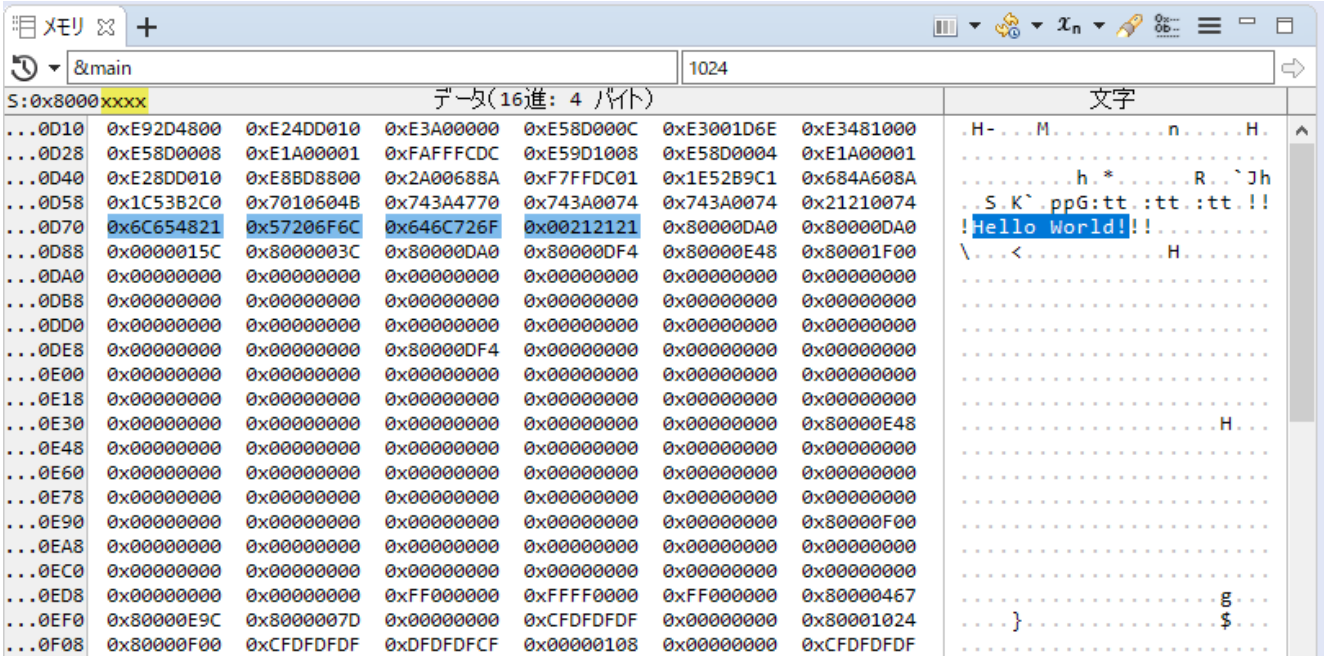


Figure 5-13 Memory view

上記の例では、**メモリ**ビューにコードの 16 進値とメモリ値の ASCII 文字エンコーディングが表示され、コードの詳細を表示できます。

デバッグアクティビティを完了した後、*ターゲットの切断* 5-49.を実行できます。

5.1.7 ターゲットの切断

ターゲットから切断するには、**デバッグコントロール**または**コマンド** ビューを使用できます。

- **デバッグコントロール**ビューを使用している場合は、ツールバーの **ターゲットから切断** をクリックします。

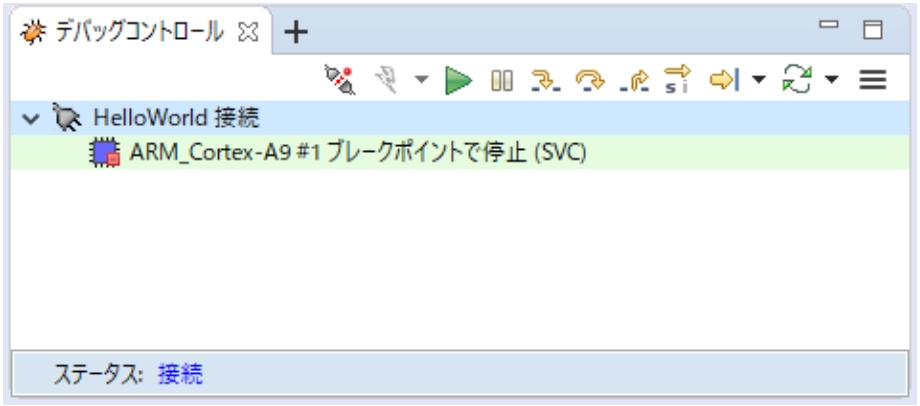


Figure 5-14 Disconnecting from a target using the Debug Control view

- **コマンド**ビューを使用している場合は、**コマンド**フィールドに **quit** と入力し**送信**をクリックします。

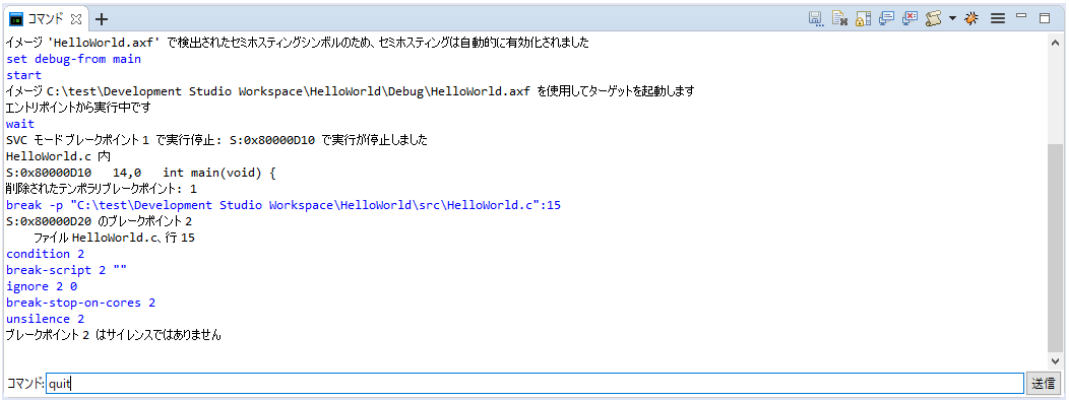


Figure 5-15 Disconnecting from a target using the Commands view

切断プロセスにより、次の場合を除き、ターゲットの状態が変更されないことが保証されます:

- ターゲットへのダウンロードはキャンセルされ、停止されている場合。
- ターゲット上のブレークポイントはすべてクリアされ、Arm Development Studio で維持されている場合。
- DAP（デバッグアクセスポート）の電源が切れている場合。
- DSC（デバッグステータスコントロール）レジスタのデバッグビットがクリアされている場合。

トレースキャプチャセッションが進行中の場合、Arm Development Studio がターゲットから切断された後でも、トレースデータはキャプチャされ続けます。

Appendix A
用語とショートカット

Arm Development Studio の新規ユーザー向けの補足情報。

以下のセクションから構成されています。

- *A.1 用語* page Appx-A-52.
- *A.2 キーボードショートカット* page Appx-A-53.

A.1 用語

以下の用語が本書で使用されています。

デバイス

デバッグの対象となるアプリケーションが組み込まれているターゲット内のコンポーネント。

ダイアログボックス

デバッグの対象となるアプリケーションが組み込まれているターゲット内のコンポーネント。

エディタ

ソースファイルなどのファイルのコンテンツを表示および変更できるビュー。 エディター領域のタブには、現在編集のために開いているファイルが表示されます。

フラッシュプログラム

フラッシュデバイスにデータを保存することを表す用語。

IDE

統合開発環境。 パースペクティブ、メニュー、およびツールバーを含むウィンドウ。これは、個々のプロジェクト、関連するサブフォルダー、およびソースファイルを管理できるメインの開発環境です。 各ウィンドウは 1 つのワークスペースにリンクされています。

パネル

ダイアログボックスやタブ内で編集可能なフィールドをグループ化した小領域。

パースペクティブ

関連するビュー、エディタ、メニュー、およびツールバーの組み合わせを表示する Eclipse ウィンドウ内のページ。

プロジェクト

Eclipse 内の関連するファイルとフォルダのグループ。

リソース

プロジェクト、ファイル、フォルダ、それらの組み合わせなどの総称。

送信

ターゲットにファイルを送信すること。

タブ

関連する情報をグループ化して表示するためにダイアログボックス内に重ねて表示される小さなページ。パネルや編集可能なフィールドを含みます。下になっているページのタブをクリックすると、上に表示されます。

ターゲット

プリント基板上の開発プラットフォーム、または ARM® ハードウェアの動作をエミュレートするソフトウェアモデル。

ビュー

ビューは、エディター内のアクティブなファイルに対応する特定の機能の関連情報を提供します。 また、独自の関連メニューとツールバーもあります。

ウィザード

よく実行されるタスク、例えば、新しいファイルやプロジェクトの作成過程を案内するための一連のダイアログボックス。

ワークスペース

プロジェクトに関係するファイルやフォルダを保管するためにファイルシステム内に確保する領域。

A.2 キーボードショートカット

以下は Arm Development Studio で最もよく使用されるキーボードショートカットです

F3

アセンブリ命令をクリックして F3 キーを押すと、その命令に関するヘルプ情報が表示されます。

F10

矢印キーと組み合わせてメインメニューにアクセスします。

Alt+F4

Arm Development Studio を終了します。

Alt+左矢印

ナビゲーション履歴をさかのぼります。

Alt+右矢印

ナビゲーション履歴を前に進みます。

Ctrl+セミコロン

ARM® アセンブラエディタでは、アクティブファイル内の選択したコードブロックにコメントマークを追加します。

Ctrl+Home

エディタのフォーカスをコードの先頭に移動します。

Ctrl+End

エディタのフォーカスをコードの末尾に移動します。

Ctrl+B

ワークスペース内で、前回のビルド以降に変更されたすべてのプロジェクトをビルドします。

Ctrl+F

アクティブエディタ内のコードを検索するための［検索］または［検索/置換］ダイアログボックスを表示します。読み出し専用のエディタではこの機能は無効になっています。

Ctrl+F4

エディタに表示されているアクティブなファイルを閉じます。

Ctrl+F6

エディタに表示されるファイルを、開いている複数のファイル間で切り替えます。

Ctrl+F7

ビューの表示を使用可能なビューの間で切り替えます。

Ctrl+F8

表示を使用可能なパースペクティブの間で切り替えます。

Ctrl+F10

矢印キーと組み合わせてドロップダウンメニューにアクセスします。

Ctrl+L

アクティブファイル内の指定された行へ移動します。

Ctrl+Q

アクティブファイル内の最後に編集した場所へ移動します。

Ctrl+Space

エディタ内のカーソル位置にオートコンプリートで挿入できる項目のリストを表示します。

Shift+F10

矢印キーと組み合わせてコンテキストメニューにアクセスします。

Ctrl+Shift+F

設定ダイアログボックス内のコードスタイルの設定をアクティブにして、アクティブなファイルに適用します。

Ctrl+Shift+L

すべてのキーボードショートカットの一覧を開きます。

Ctrl+Shift+R

リソースを開くを開くためのダイアログボックスを開きます。

Ctrl+Shift+T

要素を開く ダイアログボックスを開きます。

Ctrl+Shift+/

C / C ++エディタで、アクティブファイル内の選択したコードブロックの先頭と末尾にコメントマーカーを追加します。