Arm® Cortex®-M1 DesignStart™ FPGA-Xilinx edition

User Guide

Copyright © 2018, 2019 Arm Limited or its affiliates. All rights reserved.

Release Information

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date</th>
<th>Confidentiality</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000-00</td>
<td>27 September 2018</td>
<td>Non-Confidential</td>
<td>First release for r0p0.</td>
</tr>
<tr>
<td>0001-00</td>
<td>18 January 2019</td>
<td>Non-Confidential</td>
<td>First release for r0p1.</td>
</tr>
</tbody>
</table>

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with © or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at http://www.arm.com/company/policies/trademarks.

Copyright © 2018, 2019 Arm Limited (or its affiliates). All rights reserved.


110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349
Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Web Address

http://www.arm.com
3.5 Cortex®-M1 processor signals ................................................................. 3-35

Chapter 4 Working with the Cortex®-M1 DesignStart™ example design
4.1 Editing the A7 example design .................................................................. 4-38
4.2 Debug ....................................................................................................... 4-39
4.3 Memory map ............................................................................................ 4-40
4.4 QSPI multiplexing for the V2C-DAPLink board ........................................... 4-43
4.5 Interrupt mapping .................................................................................... 4-44
4.6 Constraints ............................................................................................... 4-45
4.7 Loading the pre-built bitstream .................................................................. 4-46
4.8 Loading the flash file ................................................................................ 4-47
4.9 Bit file regeneration .................................................................................. 4-49
4.10 Simulation ............................................................................................... 4-50

Chapter 5 V2C-DAPLink board
5.1 V2C-DAPLink adaptor board features ....................................................... 5-52
5.2 V2C-DAPLink configuration ...................................................................... 5-54
5.3 Flash download requirements .................................................................... 5-55
5.4 V2C-DAPLink board layout ........................................................................ 5-56
5.5 Conditions to enable the DAP interface .................................................... 5-58
5.6 DAP drivers ............................................................................................... 5-59
5.7 Programming the V2C-DAPLink QSPI using drag and drop ....................... 5-60
5.8 Using the μVision debugger to communicate through V2C-DAPLink ........... 5-62
5.9 Using the μVision debugger to download projects through the flash programming utility .......................................................... 5-64
5.10 Recovering the DAP connection ............................................................... 5-67

Chapter 6 Example software design
6.1 Example software design for Arty A7 ........................................................ 6-70
6.2 Example software design directory structure ............................................. 6-71
6.3 Example design reference files .................................................................. 6-72
6.4 Generating the Arty A7 board support package .......................................... 6-73
6.5 Building the example software design ....................................................... 6-78
6.6 Software update flow ................................................................................ 6-79

Appendix A Revisions
A.1 Revisions .................................................................................................. Appx-A-82
Preface

This preface introduces the Arm® Cortex®-M1 DesignStart™ FPGA-Xilinx edition User Guide. It contains the following:

- About this book on page 7.
- Feedback on page 9.
About this book

This book describes how to use the Cortex®-M1 DesignStart™ FPGA-Xilinx edition to design your system using the Cortex-M1 processor. This book also describes an example design for the Digilent Arty Artix 7 (A7) development board.

Product revision status

The rm|pn identifier indicates the revision status of the product described in this book, for example, r1p2, where:

rm  Identifies the major revision of the product, for example, r1.
pn  Identifies the minor revision or modification status of the product, for example, p2.

Intended audience

The intended audience is system designers, system integrators, and verification engineers who want to implement the processor in a Field-Programmable Gate Array (FPGA) using the Xilinx Vivado tools.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction
The Cortex-M1 DesignStart™ FPGA-Xilinx edition package provides an easy way to use the Cortex-M1 processor in the Xilinx Vivado design environment. The Cortex-M1 processor is intended for deeply embedded applications that require a small processor to be integrated into an FPGA. The processor implements the Armv6-M architecture and is closely related to the Cortex-M0 and Cortex-M0+ processors that are intended for ASIC implementation.

Chapter 2 Installing the Cortex®-M1 DesignStart™ example design
This chapter describes the Cortex-M1 DesignStart example design installation process.

Chapter 3 Cortex®-M1 processor IP configuration
After installing the Arm IP Integrator (IPI) repository, you can find the Cortex-M1 processor package in the Vivado IP catalog.

Chapter 4 Working with the Cortex®-M1 DesignStart™ example design
This chapter describes how to work with an example design targeting a low-cost evaluation board, Digilent Arty Artix 7 (A7). This example design is provided to demonstrate the integration and software development using the Cortex-M1 processor. The example is based on the Digilent Arty A7-35T board, and uses some of the standard Xilinx peripherals to connect to some of the features on the board. The example is intended to show typical usage, rather than a completely minimal Cortex-M1 processor design.

Chapter 5 V2C-DAPLink board
The optional V2C-DAPLink adaptor board provides a debug flow that is familiar to anyone who is used to working with Cortex-M microcontrollers. It allows Arty FPGA boards to be used with mbed OS 2 Classic. This chapter describes the optional V2C-DAPLink adaptor board and how it is used.

Chapter 6 Example software design
This chapter describes an example software design, and describes how to build and debug it.

Appendix A Revisions
This appendix describes the technical changes between released issues of this document.

Glossary
The Arm® Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.
See the Arm® Glossary for more information.

**Typographic conventions**

*italic* Introduces special terminology, denotes cross-references, and citations.

**bold** Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

**monospace** Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

**monospace** Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

**monospace italic** Denotes arguments to monospace text where the argument is to be replaced by a specific value.

**monospace bold** Denotes language keywords when used outside example code.

<and> Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
ADD Rd, SP, #<imm>
```

**SMALL CAPITALS** Used in body text for a few terms that have specific technical meanings, that are defined in the Arm® Glossary. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

**Additional reading**

This book contains information that is specific to this product. See the following documents for other relevant information.

**Arm publications**

- PrimeCell® Infrastructure AMBA® 2 AHB to AMBA® 3 AXI Bridges (BP136) Technical Overview (DTO0008).

The following confidential book is only available to licensees:

Cortex®-M1 Integration Manual (D110167).

**Other publications**

Feedback

Feedback on this product
If you have any comments or suggestions about this product, contact your supplier and give:
• The product name.
• The product revision or version.
• An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content
If you have comments on content then send an e-mail to errata@arm.com. Give:
• The number 100211_0001_00_en.
• If applicable, the page number(s) to which your comments refer.
• A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note
Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.
Chapter 1
Introduction

The Cortex-M1 DesignStart™ FPGA-Xilinx edition package provides an easy way to use the Cortex-M1 processor in the Xilinx Vivado design environment. The Cortex-M1 processor is intended for deeply embedded applications that require a small processor to be integrated into an FPGA. The processor implements the Armv6-M architecture and is closely related to the Cortex-M0 and Cortex-M0+ processors that are intended for ASIC implementation.

This chapter describes the Cortex-M1 DesignStart FPGA-Xilinx edition features and directory structure.

It contains the following sections:
• 1.1 Cortex®-M1 DesignStart™ FPGA-Xilinx edition package on page 1-11.
• 1.2 Directory structure on page 1-12.
1.1 **Cortex®-M1 DesignStart™ FPGA-Xilinx edition package**

An example system design is provided to target a low-cost development platform, with example integration tests.

The Cortex-M1 DesignStart FPGA-Xilinx edition package includes:

- A Cortex-M1 processor that has:
  - 1, 8, 16, or 32 interrupts.
  - Configurable endianness, only little-endian is supported in the example system.
  - Configurable OS extensions.
  - Configurable embedded debug support.
  - Configurable multiplier (small or fast).
  - *Instruction Tightly Coupled Memory* (ITCM), up to 1MB.
  - *Data Tightly Coupled Memory* (DTCM), up to 1MB.
  - ITCM Alias support
  - *Serial Wire* (SW), JTAG, or combined SWJ-DP debug port.
- Integrated AHB to AXI bridge, which allows the packaged Cortex-M1 processor to connect directly to standard Vivado components.
- Optional V2C-DAPLink board support, which:
  - Provides Cortex-M debug flow.
  - V2C-DAPLink USB to the *Serial Wire Debug* (SWD) interface.
  - V2C-DAPLink USB UART endpoint.
  - Local *Quad Serial Peripheral Interface* (QSPI), flash for code download (8MB) independent of FPGA image.
  - User accessible microSD card support.
  - Pass-through connections for shield adapter boards.
- Example designs for *Arty Artix 7* (A7) 35T and *Arty Spartan 7* (S7) 50T development boards.
  - Integrates the processor with standard Xilinx peripherals.
  - Example software tests.
- *Cortex Microcontroller Software Interface Standard* (CMSIS) compatible *Board Support Package* (BSP) generation that is done through Xilinx Vivado *Software Development Kit* (SDK).
- Support for simulation and FPGA implementation. The encrypted design can be:
  - Simulated in the Xilinx Vivado and Mentor QuestaSim simulators.
  - Implemented for FPGA in Xilinx Vivado.

**Note**

The Cortex-M1 DesignStart FPGA-Xilinx edition package:

- Can be used with any suitable Xilinx FPGA, but the example system design only supports two specific development boards. If you are using your own hardware and software, you only require version 2018.2 or later of the Xilinx Vivado tool.
- Targets Windows development environment and uses Arm Keil *Microcontroller Development Kit* (MDK) for software development.

To use the example system designs, you require:

- A Digilent Arty A7 development board.
- The board files provided by Digilent for this board.
- Xilinx Vivado.
- Arm Keil MDK.
1.2 Directory structure

The expected directory structure after you download and unpack the Arm IP deliverables is:

```plaintext
<installation_directory>
  /docs
  _hardware/
    _m1_for_arty_a7/
      _block_diagram/
      _constraints/
      _m1_for_arty_a7/
      _testbench/
    _m1_for_arty_s7/
      _block_diagram/
      _constraints/
      _m1_for_arty_s7/
      _testbench/
  _software/
    _m1_for_arty_a7/
      _Build_Keil/
      _flash_downloader/
      _m1_for_arty_s7/
      _Build_Keil/
  _vivado/
  _Arm_ipi_repository/
    _CM1DbgAXI/
    _DAPLink_to_Arty_shield/
  _Arm_sw_repository/
  Cortex
```

--- Important ---

The deliverable supports building the Cortex-M1 example design on both the Digilent Arty Artix 7 (A7) board with Artix FPGA and Spartan 7 (S7) with Spartan FPGA. Throughout this document, the A7 is used as the example. However, the same files and methods apply to the S7 project. To use the S7 project, replace any reference to `m1_for_arty_a7` with `m1_for_arty_s7`.

The following table describes the directory structure.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/docs</td>
<td>Contains this document and example design diagram.</td>
</tr>
<tr>
<td>hardware/m1_for_arty_a7/block_diagram/</td>
<td>Example block diagram.</td>
</tr>
<tr>
<td>hardware/m1_for_arty_a7/constraints/</td>
<td>Constraint files.</td>
</tr>
<tr>
<td>hardware/m1_for_arty_a7/m1_for_arty_a7/</td>
<td>Vivado project root.</td>
</tr>
<tr>
<td>hardware/m1_for_arty_a7/testbench/</td>
<td>Simulation testbench.</td>
</tr>
<tr>
<td>software/m1_for_arty_a7/</td>
<td>Example software application.</td>
</tr>
<tr>
<td>software/m1_for_arty_a7/Build_Keil/</td>
<td>Compilation directory for example code, which compiles under MDK and uses Xilinx drivers.</td>
</tr>
<tr>
<td>software/flash_downloader/</td>
<td>Flash downloader.</td>
</tr>
<tr>
<td>vivado/Arm_ipi_repository/CM1DbgAXI/</td>
<td>Cortex-M1 processor debug and AXI interface.</td>
</tr>
<tr>
<td>vivado/Arm_ipi_repository/ DAPLink_to_Arty_shield/</td>
<td>Interface block to the Arty adaptor board.</td>
</tr>
<tr>
<td>vivado/Arm_sw_repository/</td>
<td>Cortex-M1 processor software files for Board Support Package (BSP) and example application development.</td>
</tr>
</tbody>
</table>
Before you can use the deliverables, you must configure your Vivado installation to:

- Reference the Arm IP.
- Install the Digilent board files, if you want to use the provided example design.

Note

If you have already downloaded other versions of the Cortex-M1 DesignStart FPGA-Xilinx edition, then these have a similar directory structure. Arm recommends that you merge the directory structure between the installs to simplify their use. At a minimum, Arm recommends that you merge the directories under /vivado so that Vivado only needs to be assigned one directory location to read Arm hardware and software repositories.
Chapter 2
Installing the Cortex®-M1 DesignStart™ example design

This chapter describes the Cortex-M1 DesignStart example design installation process.

--- Attention ---
If you only use the provided example design for software development, then you can skip 2.6 Downloading QSPI memory models on page 2-22 and 2.7 Configuring simulation in Vivado on page 2-24. You can use the steps described in 4.8 Loading the flash file on page 4-47 to load the FPGA image.

It contains the following sections:
- 2.1 Installing board files on page 2-15.
- 2.2 Setting local drive for Windows on page 2-17.
- 2.3 Installing Arm IP repository on page 2-18.
- 2.4 Installing Arm software repository on page 2-19.
- 2.5 Installing shell models on page 2-21.
- 2.6 Downloading QSPI memory models on page 2-22.
2.1 Installing board files

The Digilent Arty *Artix 7 (A7)* board uses a board file to enable easy connectivity from the Xilinx *IP Integrator* (IPI) tool to the board pins. To use the board file in the tool, you must copy the board file into the Vivado installation.

--- Caution ---

If you have opened the example design before the board files were installed, then Vivado has already modified the project to only target the device and not the board. In this scenario, when the example design block diagram is opened, Vivado reports errors because it does not have the board I/O connections. To resolve this, you must copy the Xilinx project file (*m1_for_arty_a7.xpr*) again from the archive.

---

**Procedure**

- The board file download and installation instructions are found at [https://reference.digilentinc.com/learn/software/tutorials/vivado-board-files/start](https://reference.digilentinc.com/learn/software/tutorials/vivado-board-files/start). As a minimum you must install the /arty directory.
- To use the board files in a shared environment, you can add a reference to the location as part of your design. For example, if you uncompress the Digilent files to `<install_dir>/vivado/Digilent`, you can use the following command in the Tcl console:

```
set_param board.repoPaths ../../vivado/Digilent_board_files/vivado-boards-master/new/board_files/arty/
```

- Alternatively, the Vivado project has the parameter `board.repoPaths` ready within it. Open the Vivado project, `<install_dir>/hardware/m1_for_arty_a7/m1_for_arty_a7/xpr`, and uncomment the following line:

```
<!-- Option Name="BoardPartRepoPaths" Val="/PPRDIR/../../../vivado/Digilent_board_files/vivado-boards-master/new/board_files" -->
```

When the design is opened in Vivado and if the board files are not correctly installed, the following error message is displayed.
Critical Messages

There was one critical warning message while opening project.

Messages

1. [Board 49-67] The board_part definition was not found for digilentinc.com:arty/part0:1.1. This can happen sometimes when you use custom board part. You can resolve this issue by setting 'board.repoPaths' parameter, pointing to the location of custom board files. Valid board_part values can be retrieved with the 'get_board_parts' Tcl command.

Next Steps

You must now proceed to 2.2 Setting local drive for Windows on page 2-17.
2.2 Setting local drive for Windows

Some Vivado projects can have issues with long path names to instances deep within the hierarchy because of Windows limitations on path length. This can become apparent when running simulations and other processes.

To resolve this, when running in Windows, Arm recommends that you assign a drive letter to the root of the current design. Using this method, all subsequent paths are relative to this drive letter. To map a local drive letter to the current path:

**Prerequisites**

You must complete the steps in 2.1 Installing board files on page 2-15.

**Procedure**

1. Open Vivado.
2. Open the Tcl console window.
3. The current directory location can be checked using the Unix command `pwd`.
4. Navigate to your `<installation_directory>` folder. This is the folder where the Cortex-M1 package was installed.
5. To map the `<installation_directory>` folder to the drive `V:`, type the following command in the prompt:
   ```
   exec subst V: .
   ```

      **Attention**

   In the `exec subst V: .` command, you must add a space between `V:` and `. characters.

The package `<installation_directory>` folder maps to drive `V:` and the rest of this book assumes that this folder maps to drive `V:`. If you map to a different drive, you must use the different drive in the instructions as appropriate. If the drive mapping is successful, you should have the directories `V:/hardware, V:/software, V:/vivado, and V:/docs`.

**Next Steps**

You must now proceed to 2.3 Installing Arm IP repository on page 2-18.
2.3 Installing Arm IP repository

After downloading and unpacking the deliverable, the Arm IP Integrator (IPI) repository must be added to the list of Vivado IP repositories. This makes the processor available in any new designs.

To add Arm IPI repository to the list of Vivado IP repositories:

**Prerequisites**

You must complete the steps in:
- 2.1 Installing board files on page 2-15.
- 2.2 Setting local drive for Windows on page 2-17.

**Procedure**

1. Open Vivado.
2. From Tools → Settings, select IP Defaults.
3. In the list of Default IP repository search paths, add the path to the /Arm_ipi_repository. 

   Vivado only reads the IPI repository during design creation. If the repository is updated, or an existing design must use the Cortex-M1 processor, then you must refresh the project repository. To do this, navigate to Tools → Settings → IP → Repository → Refresh all.

**Next Steps**

You must now proceed to 2.4 Installing Arm software repository on page 2-19.
2.4 Installing Arm software repository

The Arm software repository must also be added to the list of available Vivado repositories. To add the Arm software repository to the list of Vivado software repositories:

**Prerequisites**

You must complete the steps in:

- 2.1 Installing board files on page 2-15.
- 2.2 Setting local drive for Windows on page 2-17.
- 2.3 Installing Arm IP repository on page 2-18.

**Procedure**

1. Open Vivado.
2. From File, select *Launch SDK*.
3. Set the default *Exported location* to `V:/software` and the default *Workspace* to `V:/software/m1_for_arty_a7/sdk_workspace`.

![Launch SDK](image)

Figure 2-2  Launch SDK

4. Vivado issues a warning regarding the exported hardware file being out of date. This is because you have not built the project. Select Yes to proceed.
5. Once the SDK opens, select Xilinx → Repositories and add the path to the `V:vivado/Arm_sw_repository/` to the Global Repositories.

Next Steps

To use the Cortex-M1 software on existing designs, you might be required to rescan the Software Development Kit (SDK) repositories. In the SDK, select Xilinx → Repositories → Rescan Repositories.

You must now proceed to 2.6 Downloading QSPI memory models on page 2-22.
2.5 Installing shell models

If you do not want to perform simulation with Quad Serial Port Interface (QSPI) memory models or any simulation at all, the Cortex-M1 DesignStart FPGA-Xilinx edition package has support to allow you to not have to download the QSPI files from the vendor websites, while allowing you to not have the warnings associated with opening the project when these files have not been downloaded. To achieve this, the package contains empty shell files which can be extracted to the correct locations. These empty shell files do not have any functionality. However, when these empty shell files are extracted, and when you open the project, the warnings do not occur. Additionally, these shell files compile correctly under simulation, so you do not get a simulation warning when you include them. If you require correct simulation of the QSPI devices that the example design uses, you must download the correct files from the respective vendor websites. For more information, see 2.6 Downloading QSPI memory models on page 2-22.

Prerequisites

You must complete the steps in 2.2 Setting local drive for Windows on page 2-17.

Procedure

1. Navigate to v:/hardware/m1_for_art_7/testbench.
2. Extract the testbench_shell_files.zip to the current directory (not to a further directory). This results in the following file structure.

   V:/m1_for_arty_a7
   |   _hardware
   |     |   _sfdp.vmf
   |     |   _testbench
   |     |     |   _Micron_N25Q128A13E
   |     |     |   |   _code
   |     |     |     |   _N25Qxxx.v
   |     |     |   _S25fl128s
   |     |     |   _model
   |     |         |   _s25fl128s.v

After you extract the shell memory models, if you require installing the correct QSPI memory models from the respective vendor websites, then before installing these files, you must delete the file structure shown in this section, including the three files, and all associated directories.

If you require correct simulation of QSPI devices that the example design uses, you must download the correct files from the respective vendor websites. For this you must proceed to 2.6 Downloading QSPI memory models on page 2-22.
2.6 Downloading QSPI memory models

If you want to simulate the example design, then the testbench can also simulate the Quad Serial Port Interface (QSPI) devices that are fitted to the Arty Artix 7 (A7) baseboard (a Micron device) and the V2C-DAPLink board (a Cypress device).

Prerequisites

--- Note ---

- It is only necessary to download the QSPI memory models if you want to simulate the example design when you are operating on the Arty A7 board, and optionally, with the V2C-DAPLink board fitted. If you do not want to simulate the design, you can ignore this section.
- If you do not want to perform simulation with Quad Serial Port Interface (QSPI) memory models or any simulation at all, empty shell files can be extracted to the correct locations to support this feature. For more information, see 2.5 Installing shell models on page 2-21.

--- Caution ---

If you do not download the QSPI memory models, then you get warnings every time you open the Vivado project. The following figure shows these warnings. If you do not intend to simulate the QSPI models, then these warnings can be ignored. If you want to remove these warnings, then you can install the shell files following the instructions in 2.5 Installing shell models on page 2-21.

![Critical Messages](image)

You must complete the steps in:

- 2.1 Installing board files on page 2-15.
- 2.2 Setting local drive for Windows on page 2-17.
• 2.3 Installing Arm IP repository on page 2-18.
• 2.4 Installing Arm software repository on page 2-19.

Procedure
• To simulate the QSPI devices that are fitted, you must download the appropriate models from Micron and Cypress websites.
• When the QSPI memory models are correctly installed, you can enable using the Verilog define at the top of V:/testbench/tb_m1_for_arty.v.

If the V2C-DAPLink board is fitted and QSPI device models included, then code execution is from the QSPI device on the V2C-DAPLink board.

Next Steps
You must first refer to the information in either of the following depending on the QSPI model that you choose to install:
• 2.6.1 Micron QSPI model on page 2-23.
• 2.6.2 Cypress QSPI model on page 2-23.

After you have downloaded and installed the required QSPI model, you must proceed to 2.7 Configuring simulation in Vivado on page 2-24.

2.6.1 Micron QSPI model
The Micron device used on the Digilent Arty Artix 7 (A7) base board is N25Q128A13E.
A Verilog simulation model for this device is available in the Micron website.
The archive file that you must download is N25Q128A13E_3V_MicronXIP_VG12.tar. When the archive is downloaded, it must be expanded to a directory named /Micron_N25Q128A13E. This directory must be located under the V:/hardware/m1_for_arty_a7/testbench directory. To enable the correct configuration of the QSPI memory, the /Micron_N25Q128A13E/sim/sfdp.vmf file must be copied to the V:/hardware/m1_for_arty_a7/testbench directory.
If you are using the Micron model, ensure to add the include directory for it to the design. This is done in the Tcl console using the following command:
set_property INCLUDE_DIRS [get_property DIRECTORY [current_project]]/../testbench/Micron_N25Q128A13E [get_filesets sim_1]

2.6.2 Cypress QSPI model
The Cypress (Spansion) QSPI device used on the V2C-DAPLink board is S25fl128S.
A Verilog simulation model for this device is available at the Cypress website.
The archive file that you must download is s25f1128s.zip. This archive is a self-installing executable. Run the executable, and extract the files to the V:/hardware/m1_for_arty_a7/testbench directory. This copies the model files to a folder called /S25f1128s in this location.
2.7 Configuring simulation in Vivado

To configure simulations in Vivado, you must have either the Vivado or a third-party simulator installed. The paths to the simulator must be configured in Vivado.

To configure the paths to the simulator in Vivado, navigate to Tools → Settings → Tool Settings → 3rd Party simulators.

Prerequisites

- 2.1 Installing board files on page 2-15.
- 2.2 Setting local drive for Windows on page 2-17.
- 2.3 Installing Arm IP repository on page 2-18.
- 2.4 Installing Arm software repository on page 2-19.
- 2.6 Downloading QSPI memory models on page 2-22.
Chapter 3
Cortex®-M1 processor IP configuration

After installing the Arm IP Integrator (IPI) repository, you can find the Cortex-M1 processor package in the Vivado IP catalog.

This package is a version of Cortex-M1 r1p0 processor with debug and the BP136 AHB to AXI bridge r0p1 pre-integrated.


This chapter describes the four Cortex-M1 processor IP configuration tabs, each with details on individual configuration categories.

Note

• For more information about the Cortex-M1 processor configuration options, see, the Configurable options section in the Cortex®-M1 Technical Reference Manual.

• For more information on the BP136 AHB to AXI bridge, see the PrimeCell® Infrastructure AMBA®2 AHB to AMBA®3 AXI Bridges (BP136) Technical Overview. This document is superseded, indicating that the documentation is no longer maintained, but the current content is still relevant.

It contains the following sections:
• 3.1 Configuration tab on page 3-26.
• 3.2 Debug tab on page 3-28.
• 3.3 Instruction Memory tab on page 3-30.
• 3.4 Data Memory tab on page 3-32.
• 3.5 Cortex®-M1 processor signals on page 3-35.
3.1 Configuration tab

The following figure shows the configuration tab.

![Configuration tab](image)

**Number of interrupts**

This indicates the number of interrupt sources the Cortex-M1 processor supports. The number of interrupts port is automatically set to match the size of the vector that is connected to the IRQ input.

**Note**

The valid values for the number of interrupts are 1, 8, 16, and 32. If the vector connected to the IRQ pin has a width that is different from any of the valid values, the IRQ port is set to the next highest valid value. When editing the IPI block diagram to modify the width of the IRQ port, you must first update the vector that is connected to the IRQ pin to the new desired width. Run Validate Design on the block diagram and the IRQ port is updated to match the width of the input vector.
OS Extensions
Enable OS extensions if the Cortex-M1 processor is defined to include the Nested Vectored Interrupt Controller (NVIC) and Core OS extensions such as SVC and SysTick.

Small Multiplier
Enable Small Multiplier if the Cortex-M1 processor is to use a small but slower multiplier for fabrics that do not have dedicated multiplier resources.

Big Endian
Enable Big Endian if the Cortex-M1 processor is defined to have BE8 big-endian byte ordering.

Note
The example design provided only supports little-endian, but you can choose the Big Endian option if you are using the Cortex-M1 processor in any other system.
3.2 Debug tab

The following figure shows the Debug tab.

On this tab you can select the following:

**Debug port select**
You can select either JTAG, *Serial Wire* (SW), JTAG and SW, or No Debug.

*Note*

Any debug port that is implemented on the processor needs to be connected to a debug probe using I/O pins. This is generally a separate interface to the FPGA JTAG port.

If the optional V2C-DAPLink board is fitted, the example design connects *Serial Wire Debug* (SWD) to this board.
Small Debug

If small debug is enabled the processor debug logic has reduced functionality, but with the benefit of reduced resource usage.

The differences are:

- The full debug configuration has four breakpoint comparators and two watchpoint comparators.
- The small debug configuration has two breakpoint comparators and one watchpoint comparator.

No debug

When No Debug is selected the debug port is removed from the processor core. This allows a resource-optimized build to be created of the core. Also, when No Debug is selected, consider the following:

- The Small Debug option is disabled.
- All debug pins are removed from the processor instance (JTAG, SW, and debug resets).
- The ability to drag-and-drop new code using the V2C-DAPLink board is no longer supported. For more information, see 5.7 Programming the V2C-DAPLink QSPI using drag and drop on page 5-60. However, if the V2C-DAPLink J2 jumper (Cfg) is fitted, existing code is still run from the V2C-DAPLink QSPI device.
- The ability to debug the processor core is removed. For more information, see 5.8 Using the µVision debugger to communicate through V2C-DAPLink on page 5-62.
- The ability to download software projects through the V2C-DAPLink board is no longer supported.
3.3 Instruction Memory tab

The following figure shows the Instruction Memory tab.

On this tab you can select the following:

**ITCM Size**

The range is 8KB to 1MB. Select the optimal size for your code base.

**Note**

- Currently the flow to update a bitstream with new Instruction Tightly Coupled Memory (ITCM) data only supports memory sizes in the range 16KB to 128KB. If you require sizes outside that range, contact Arm for support. For more information on this flow, see *Software Update flow* on page 6-79.
**Initialize ITCM**

If you require the instruction memory to be initialized when the design is built:

1. Select Initialize ITCM.
2. Specify the filename, see the example design as a reference.

--- **Note** ---

- The filename must not have quote marks around it.
- The filename must be added to the design and marked as a memory initialization file.
- Vivado reads the memory file during synthesis. It is not possible to update the memory file and to run just implementation or generate bitstream. To incorporate software updates into an existing bit file, see *Software Update flow on page 6-79.*

ITCM aliasing is controlled at reset by the state of the **CFGITCMEN[1:0]** signal. For more information on **CFGITCMEN[1:0]**, see the The upper and lower aliases can be enabled independently, that is, either one alias, both aliases, or none of the aliases. For more information about processor memory regions, see the *Cortex®-M1 Technical Reference Manual.*

To boot the processor from ITCM, you must:

1. Enable ITCM lower alias.
2. Initialize the ITCM.

If the processor does not boot from ITCM, you must provide memory at address **0x00000000** on the external AXI interface which contains the initial stack pointer and vector table.

Instruction fetch latency is lower from ITCM than from the AXI interface. If you boot from AXI memory, you can copy code to ITCM at the upper alias and then execute from there to get better performance.
3.4 Data Memory tab

The following figure shows the data memory tab.

**Figure 3-4 Data Memory tab**

**DTCM size**

The range is 2KB to 1MB. Select the optimal size for your code base. You can also choose not to include a DTCM (0KB). When selecting the smaller DTCM sizes, you must ensure that your software project is correctly configured to match the DTCM size that is available. It is possible to configure a software project with a larger data memory allocation than that available in the hardware because the DTCM is often uninitialized. This leads to runtime failures.
Initialize DTCM

If you require the data memory to be initialized when the design is built:
1. Select Initialize DTCM checkbox.
2. Specify the filename, see the example design as a reference.

--- Note ---
- The filename must not have quote marks around it.
- The filename must be added to the design and marked as a memory initialization file.
- Vivado reads the memory file during synthesis. It is not possible to update the memory file and to run just implementation or generate bitstream. To incorporate software updates into an existing bit file, see 6.6 Software update flow on page 6-79.

No DTCM option

With the No DTCM configuration, software must be compiled to divide the ITCM memory between instruction and data. The following figure shows an example Keil project configuration. This configuration is for a 16KB ITCM (0x4000 address range). The first 12KB (0x3000) is allocated to instruction memory (IROM), and the top 4KB (0x1000) is allocated to data memory (IRAM). The data memory area starts at 0x3000 which is the top part of the instruction memory, and within the memory region of the ITCM. By default, if the DTCM is included, then data memory (IRAM) must start at 0x20000000. For more information on the memory map, see 4.3 Memory map on page 4-40.

--- Figure 3-5 Example Keil configuration ---
----- Caution ----- 

The No DTCM configuration prevents programming the V2C-DAPLink QSPI using drag and drop, as described in 5.7 Programming the V2C-DAPLink QSPI using drag and drop on page 5-60 and also using the µVision debugger to download projects through the flash programming utility, as described in 5.9 Using the µVision debugger to download projects through the flash programming utility on page 5-64. The reason for this is because the V2C-DAPLink firmware requires the DTCM memory area for the software download process.
3.5 Cortex®-M1 processor signals

For details of the Cortex-M1 signals, see the Signal descriptions appendix in the Cortex®-M1 Technical Reference Manual.

The external AHB-Lite interface is not exported, and the AXI interface replaces it. For more information, see the AHB master bus to AXI bridge signal connections figure in the PrimeCell® Infrastructure AMBA®2 AHB to AMBA®3 AXI Bridges (BP136) Technical Overview.

The AHB-AP interface is not exported, it is replaced by the Serial Wire (SW) or JTAG interface pins that are described in the Arm® CoreSight™ SoC-400 Technical Reference Manual.

——— Note ————

The PrimeCell® Infrastructure AMBA®2 AHB to AMBA®3 AXI Bridges (BP136) Technical Overview document is a superseded, indicating that the documentation is no longer maintained, but the current content is still relevant.
This chapter describes how to work with an example design targeting a low-cost evaluation board, Digilent Arty Artix 7 (A7). This example design is provided to demonstrate the integration and software development using the Cortex-M1 processor. The example is based on the Digilent Arty A7-35T board, and uses some of the standard Xilinx peripherals to connect to some of the features on the board. The example is intended to show typical usage, rather than a completely minimal Cortex-M1 processor design.

The board provides the Digilent Pmod™ peripheral module headers for peripherals, and shield expansion headers to support additional expansion. You can use the optional Arm V2C-DAPLink board with these headers to use Cortex-M1 for easy debug and software development. If you do not use the V2C-DAPLink board, you can still connect a Serial Wire Debug (SWD) probe (Arm Keil® ULINK™ or similar) to J4 (nSRST on I/O[39], SWDIO on I/O[40], and SWCLK on I/O[41]).

Some features of the example design detect the presence of the V2C-DAPLink board, and adapt accordingly. The V2C-DAPLink board includes pass-through headers for an additional shield board to be connected on top.

The block diagram of the design is available in /docs/m1_for_arty_a7_example_design.pdf.

The example design has the following functions:

- UART to output to either the Arty onboard USB connector, or the V2C-DAPLink board, when fitted.
- GPIO_0 connected to the four DIP switches, SW[3:0], and the four green LEDs LD[7:4].
- GPIO_1 connected to the four push button switches, BTN[3:0], and the four multicolor LEDs.
- QSPI_0 connected to the Arty on-board Quad Serial Port Interface (QSPI) flash memory.
- BRAM ctrl 0 connected to 64KB of internal FPGA BRAM.
The following peripherals are connected to the V2C-DAPLink adaptor board using J4.

- QSPI 1 connected to the adaptor board QSPI flash memory.
- SPI 0 connected to the adaptor board SD card memory.

A number of pre-built files are provided with the example design. For more information, see 6.3 Example design reference files on page 6-72.

Note

The example design files are modified by the Vivado tool when you open the design, so it might be useful to copy the /hardware directory before working with it. For more information on the directory structure, see 1.2 Directory structure on page 1-12.

It contains the following sections:

- 4.1 Editing the A7 example design on page 4-38.
- 4.2 Debug on page 4-39.
- 4.3 Memory map on page 4-40.
- 4.4 QSPI multiplexing for the V2C-DAPLink board on page 4-43.
- 4.5 Interrupt mapping on page 4-44.
- 4.6 Constraints on page 4-45.
- 4.7 Loading the pre-built bitstream on page 4-46.
- 4.8 Loading the flash file on page 4-47.
- 4.9 Bit file regeneration on page 4-49.
- 4.10 Simulation on page 4-50.
4.1 Editing the A7 example design

When loading the Arty Artix 7 (A7) example design for the first time, if warning messages are issued about either missing IP blocks (Cortex-M1 processor) or board files (Digilent board files), then the design must be closed and the instructions for installation of the IP repository and Digilent board files must be followed. For more information on these installations, see Chapter 2 Installing the Cortex®-M1 DesignStart™ example design on page 2-14. In this scenario, it is possible that Vivado has modified the design file. Therefore, after correct installation of the IP repository and board files is complete, the original design must be installed from the archive.

Procedure

1. Open Vivado.
2. Select Open Project on the splash screen, and select /hardware/m1_for_arty_a7/m1_for_arty_a7/m1_for_arty_a7.xpr.
3. In the sources tab, navigate down the hierarchy to the m1_for_arty_a7_i instance, marked with a block diagram symbol. Double-clicking this opens the block diagram that is shown in /docs/m1_for_arty_a7_example_design.pdf.
4. The design can now be navigated to understand the connectivity and configuration. Double-clicking on any of the IP blocks brings up the configuration for that block.

Note

If you want to change the memory map, this is done in the address editor. However, you must not modify the addresses of the V2C-DAPLink interface peripherals. Additionally, the example hardware memory map matches the pre-compiled software memory map. Therefore, if other peripheral addresses are modified, the equivalent changes must be made in the software.
4.2 Debug

The example design uses Serial Wire Debug (SWD). There is no dedicated Arm debug connector on the Arty Artix 7 (A7) board, therefore, SWD is only connected to the expansion connector. When the V2C-DAPLink adaptor board is fitted, the SWD ports are connected directly to this board and are accessible through the USB connector as part of the V2C-DAPLink interface.

To use JTAG debug, you must use a suitable debug probe, and route the JTAG connections to the expansion headers.
4.3 Memory map

The following figure shows the memory map of the example Cortex-M1 DesignStart FPGA-Xilinx edition system.

![Memory map diagram]

Figure 4-1 Example system memory map

The following table shows the example Cortex-M1 DesignStart FPGA-Xilinx edition memory map.
<table>
<thead>
<tr>
<th>Type</th>
<th>Start</th>
<th>End</th>
<th>Peripheral</th>
<th>Instance name</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>0x00000000</td>
<td>0x0000FFFFF</td>
<td>Instruction Tightly Coupled Memory (ITCM) (lower)</td>
<td>daplink_if_0/axi-_xip_quad_0</td>
<td>1MB</td>
<td>Boot region when CFGITCM[0] is 1. This indicates that there is no V2C-DAPLink board.</td>
</tr>
<tr>
<td></td>
<td>0x00000000</td>
<td>0x0000FFFFF</td>
<td>Quad Serial Peripheral Interface (QSPI)</td>
<td></td>
<td></td>
<td>Boot region when CFGITCM[0] is 0. This indicates that there is a V2C-DAPLink board.</td>
</tr>
<tr>
<td></td>
<td>0x10000000</td>
<td>0x1000FFFFF</td>
<td>ITCM (upper)</td>
<td>Integrated in the Cortex-M1 processor.</td>
<td>1MB</td>
<td>Upper ITCM alias. CFGITCM[1] is always HIGH in the example design.</td>
</tr>
<tr>
<td></td>
<td>0x10100000</td>
<td>0x1010FFFFF</td>
<td>External</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRAM</td>
<td>0x20000000</td>
<td>0x2000FFFFF</td>
<td>Data Tightly Coupled Memory (DTMC)</td>
<td>Integrated in the Cortex-M1 processor.</td>
<td>Configurable</td>
<td>eXecute-never (XN) region</td>
</tr>
<tr>
<td></td>
<td>0x20100000</td>
<td>0x2010FFFFF</td>
<td>External</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Peripheral</td>
<td>0x40000000</td>
<td>0x4000FFFFF</td>
<td>QSPI eXecute In Place (XIP)</td>
<td>daplink_if_0/axi-_xip_quad_0</td>
<td>64KB</td>
<td>Provides code execution from QSPI on the V2C-DAPLink board.</td>
</tr>
<tr>
<td></td>
<td>0x40010000</td>
<td>0x4001FFFFF</td>
<td>GPIO 0</td>
<td>daplink_if_0/axi_gpio_0</td>
<td>64KB</td>
<td>Control for QSPI peripheral multiplexer. Bit [0] selects between the two QSPI peripherals.</td>
</tr>
<tr>
<td></td>
<td>0x40020000</td>
<td>0x4002FFFFF</td>
<td>QSPI</td>
<td>daplink_if_0/quad_spi_0</td>
<td>64KB</td>
<td>Provides programming control from QSPI on the V2C-DAPLink board.</td>
</tr>
<tr>
<td></td>
<td>0x40030000</td>
<td>0x4003FFFFF</td>
<td>SPI</td>
<td>daplink_if_0/axi-_single_spi_0</td>
<td>64KB</td>
<td>Single SPI on a dedicated connector.</td>
</tr>
<tr>
<td></td>
<td>0x40040000</td>
<td>0x4004FFFFF</td>
<td>Unused</td>
<td>-</td>
<td>-</td>
<td>Unused peripheral region</td>
</tr>
<tr>
<td></td>
<td>0x40100000</td>
<td>0x4010FFFFF</td>
<td>UART</td>
<td>axi_uartlite_0</td>
<td>64KB</td>
<td>Baseboard UART or V2C-DAPLink USB, when fitted.</td>
</tr>
<tr>
<td></td>
<td>0x40110000</td>
<td>0x4011FFFFF</td>
<td>GPIO 1</td>
<td>axi_gpio_0</td>
<td>64KB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x40120000</td>
<td>0x4012FFFFF</td>
<td>GPIO 2</td>
<td>axi_gpio_1</td>
<td>64KB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x40130000</td>
<td>0x4013FFFFF</td>
<td>QSPI</td>
<td>axi_quad_spi_0</td>
<td>64KB</td>
<td>Provides read/write access to QSPI on V2C-DAPLink board.</td>
</tr>
<tr>
<td></td>
<td>0x40140000</td>
<td>0x4014FFFFF</td>
<td>Unused</td>
<td>-</td>
<td>-</td>
<td>Unused peripheral region</td>
</tr>
</tbody>
</table>

The V2C-DAPLink firmware uses this region. Therefore, you must not modify it to retain compatibility with the V2C-DAPLink board.
<table>
<thead>
<tr>
<th>Type</th>
<th>Start</th>
<th>End</th>
<th>Peripheral</th>
<th>Instance name</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
<td>0x60000000</td>
<td>0x60001FFF</td>
<td>BlockRam</td>
<td>axi_bram_ctrl_0</td>
<td>8KB</td>
<td>Additional area of RAM. This also supports code execution.</td>
</tr>
<tr>
<td></td>
<td>0x60002000</td>
<td>0x9FFFFFFF</td>
<td>Unused</td>
<td>-</td>
<td>-</td>
<td>Unused RAM region.</td>
</tr>
<tr>
<td>External</td>
<td>0xA00000000</td>
<td>0xDFFFFFFF</td>
<td>Unused</td>
<td>-</td>
<td>-</td>
<td>Unused external device region.</td>
</tr>
<tr>
<td>System</td>
<td>0xE00000000</td>
<td>0xE0000FFF</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0xE00010000</td>
<td>0xE0001FFF</td>
<td>Data Watchpoint and Trace (DWT)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0xE00020000</td>
<td>0xE0002FFF</td>
<td>Breakpoint Unit (BPU)</td>
<td>Integrated in the Cortex-M1 processor.</td>
<td>4KB</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0xE00030000</td>
<td>0xE000DFFF</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0xE000E0000</td>
<td>0xE00EFFF</td>
<td>System Control Space (SCS)</td>
<td>Integrated in the Cortex-M1 processor.</td>
<td>4KB</td>
<td>Nested Vectored Interrupt Controller (NVIC), Debug, and system control registers.</td>
</tr>
<tr>
<td></td>
<td>0xE00EF0000</td>
<td>0xE00EFFF</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0xE00FF0000</td>
<td>0xE00FFFFF</td>
<td>ROM table</td>
<td>Integrated in the Cortex-M1 processor. Modification is not supported when using the Cortex-M1 DesignStart FPGA-Xilinx edition.</td>
<td>4KB</td>
<td>-</td>
</tr>
<tr>
<td>Reserved</td>
<td>0xE0100000</td>
<td>0xFFFFFFFF</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

All the AXI peripherals that are detailed in the example design are mapped to either of the following:
- Peripheral region (0x40000000 to 0x5FFFFFFF).
- SRAM region (0x60000000 to 0x9FFFFFFF) in the case of the block RAM controller.

If the V2C-DAPLink board is not fitted, then the ITCM RAM, implemented in FPGA memory, is mapped to both 0x00000000 and 0x10000000. Code that is preloaded into the ITCM RAM is executed from address 0x00000000 from boot-up.

If the V2C-DAPLink board is fitted, then the ITCM RAM is only mapped to 0x10000000. For code execution, the V2C-DAPLink board contains a QSPI AXI peripheral configured to eXecute In Place (XIP) mode. This peripheral is named qspi_xip and is mapped to address 0x00000000. Code is executed from this XIP QSPI device on boot-up.

The DTCM is always mapped starting at 0x20000000. In contrast to other Cortex-M processors, which do not have a TCM, the DTCM is XN.
4.4 QSPI multiplexing for the V2C-DAPLink board

The Quad Serial Port Interface (QSPI) device, that is fitted to the V2C-DAPLink board, has two Xilinx QSPI AXI controllers. A single GPIO signal from a GPIO peripheral can select one of the two controllers to use. One of the controllers is configured in eXecute In Place (XIP) mode, the other controller is configured in normal mode, which is required to write to the memory device.

For more information on the peripherals and their memory map, see Table 5-1 Interface type on page 5-55

--- Caution ---

If software is intended to be run from the V2C-DAPLink board, then the software must not switch the GPIO signal across to the controller in normal mode. If this happened, then the processor can no longer read the code and the processor enters LOCKUP state.
4.5 Interrupt mapping

The following table shows the interrupts that the example system uses.

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>UART0_IRQn</td>
<td>UART 0 interrupt</td>
</tr>
<tr>
<td>1</td>
<td>GPIO0_IRQn</td>
<td>GPIO 0 interrupt</td>
</tr>
<tr>
<td>2</td>
<td>GPIO1_IRQn</td>
<td>GPIO 1 interrupt</td>
</tr>
<tr>
<td>3</td>
<td>QSPI0_IRQn</td>
<td>Quad Serial Port Interface (QSPI) 0, (Arty board) interrupt</td>
</tr>
<tr>
<td>4</td>
<td>DAP_QSPI0_IRQn</td>
<td>V2C-DAPLink board QSPI 0 interrupt</td>
</tr>
<tr>
<td>5</td>
<td>DAP_SPI0_IRQn</td>
<td>V2C-DAPLink board SPI 0 interrupt</td>
</tr>
<tr>
<td>6</td>
<td>DAP_QSPI_XIP_IRQn</td>
<td>V2C-DAPLink board QSPI eXecute In Place (XIP) interrupt</td>
</tr>
</tbody>
</table>

If you use CMSIS for your software flow, these interrupts are enumerated in the ARTY_CM1.h and startup_ARTY_CM1.s files.

Note

Additionally, IRQ[31] is connected to DAPLINK_fitted_n. This is used as a level-detect non-interrupt signal to determine if the V2C-DAPLink is fitted.
4.6 Constraints

Two constraint files for the example design are included in the /hardware/m1_for_arty_a7/constraints folder.

The constraints include internal timing constraints for the Cortex-M1 processor, particularly asynchronous clock domain crossing paths. These constraints must be included in any design that uses the Cortex-M1 processor. The majority of the I/O connections are made using the board file connections, which automatically populate the I/O pad and I/O voltage standard. The exception is the shield connector, which goes to the V2C-DAPLink adaptor board. This uses a tristate port due to the mix of signal direction. Since this does not map directly onto the board file, the I/O pad and I/O standards for the shield connector are defined in the synthesis constraint file.
4.7 Loading the pre-built bitstream

The design is provided with a prebuilt bit file in V:/hardware/m1_for_arty_a7/m1_for_arty_a7/m1_for_arty_a7_reference.bit. This bit file allows you to program the Arty Artix 7 (A7) board with the example design, which can be used to demonstrate correct connection, programming, and operation of the Arty A7 board. This file loads the volatile memory in FPGA RAM. Therefore, the FPGA programming is only valid while the board is powered on. Additionally, if Prog is pressed, then the flash program image is loaded into the FPGA, overwriting any existing FPGA image.

--- Caution ---

If you have not programmed the flash, then the Digilent example design is the image in the flash, and this is loaded into the FPGA. In this instance, the board is not running a Cortex-M processor.

---

In these instructions, V: is used to refer to the package install directory.

The bitstream includes a software image that is preloaded into Instruction Tightly Coupled Memory (ITCM).

To load the pre-built bitstream:

**Procedure**

1. Open Vivado.
2. On the splash screen, from Flow → Open Hardware manager, select V:/hardware/m1_for_arty_a7/m1_for_arty_a7/m1_for_arty_a7.xpr.
3. Connect the Arty board using the micro-USB connection, not the V2C-DAPLink connector.
4. Connect a terminal application (for example, TeraTerm) to the USB UART port. This is automatically created when Arty A7 board is connected.
5. Set the terminal to: Baud rate 115,200 8 bits One stop No parity.
6. Open the hardware manager, and select Open Target.
7. Right click on the Digilent A7 board's xc7A35t device.
8. Select Program Device and locate the m1_for_arty_a7_reference.bit bitstream file.
9. Wait while the bitstream is downloaded.
10. If Reset is pressed on the Arty A7 board, the following message appears on the splash screen and displayed on the terminal.

```
**********************************************
Arm Cortex-M1 Revision 0 Variant 1
Example design for Digilent A7 board
V2C-DAPLink board not detected
Use DIP switches and push buttons to control LEDS
Version 1.1
**********************************************
Bram readback correct
Base SPI readback correct
```

11. Test the operation of the LEDs using the DIP switches and the push buttons.

If PROG is pressed on the Arty A7 board, then the built-in Digilent reference design is loaded. This displays a different splash screen on the terminal, using the same UART board rates. This reference design has different functions for the DIP and push button switches. To return to the Arty reference design, you must reprogram the board using the instructions in this section. To make the Arm reference design persistent, follow the steps in 4.8 Loading the flash file on page 4-47 to load the design in flash.
4.8 Loading the flash file

A flash file is provided that you can use to program the Arty board with the example design and a simple test program. This flash file can be used to demonstrate correct connection, programming, and operation of the Arty Artix 7 (A7) board. The non-volatile flash image is used to load the FPGA on board powerup, and also when Prog is pressed.

Note
The board is provided with a Digilent example design. Programming the flash overwrites this design.

In these instructions, $V:$ is used to refer to the package install directory.

The flash file includes a software image that is preloaded into Instruction Tightly Coupled Memory (ITCM).

To load the pre-built flash file:

Procedure
1. Open Vivado.
2. On the splash screen, select Open Project, and select $V:/hardware/m1_for_arty_a7/m1_for_arty_a7/m1_for_arty_a7.xpr$.
3. Connect the Arty board using the micro-USB connection, not the V2C-DAPLink connector.
4. Connect a terminal application (for example, TeraTerm) to the USB UART port. This is automatically created when Arty A7 board is connected.
5. Set the terminal to: Baud rate 115,200 8 bits One stop No parity.
6. Open the Hardware manager, and select Open Target. Select Auto Connect.
7. Right-click on the Digilent A7 board’s xc7a35t, and select Add configuration memory device.
8. Select mt25ql128-spi-x1_x2_x4. Select OK. The following figure shows the resultant hardware tab in Vivado.

![Arty A7 board hardware tab in Vivado](image)

9. In the Do you want to program the configuration device now prompt, click OK.
10. Select $V:/hardware/m1_for_arty_a7/m1_for_arty_a7/m1_for_arty_a7_reference.mcs$ for the configuration file.
11. Click OK to program the flash.
When the flash is programmed, press Prog to load the FPGA with the example design.

Note

The Arty Spartan-7 (S7) board has a different flash device to the Arty A7 board. For the Arty S7, select device

s25fl128sxxxxx0-spi-x1_x2_x4.
4.9 Bit file regeneration

You can regenerate the bit file using Run Implementation and Generate Bitstream. Any new bitstream is located in the Vivado numbered implementation directory, for example, m1_for_arty_a7/m1_for_arty_a7.runs/impl_1/.
4.10 Simulation

A testbench is provided which instantiates the example design. The testbench allows for simulation with both the V2C-DAPLink board fitted and not fitted. This is controlled with a Verilog define in `/testbench/tb_m1_for_arty.v`. Additionally, the testbench allows simulation of the V2C-DAPLink peripherals that are present, but with the V2C-DAPLink fitted link removed. This configuration allows faster simulation because the code is executed from the Instruction Tightly Coupled Memory (ITCM) instead of the V2C-DAPLink Quad Serial Port Interface (QSPI) flash device model. The testbench stimulates the pushbutton and DIP switches fitted to the host board. It also has a behavioral UART receiver to display the output of the UART onto the simulation console.

To run simulations from Vivado, the Vivado simulator or a third-party simulator has to be installed. This is selected under Tools → Settings → Simulation → Target Simulator.

The Cortex-M1 IP encryption supports the in-built Vivado simulator and the Questa Advanced simulator. If you already have the Questa Advanced simulator installed in the path, then no other settings are required. However, if the Questa Advanced simulator is not on your path, then the path can be set within Vivado.

This is selected under Tools → Settings → 3rd Party Simulators.

4.10.1 Testbench conditionals

The testbench conditional compilation options are controlled by defines at the top of `tb_m1_for_arty_a7.v`.

<table>
<thead>
<tr>
<th>Option name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`INCLUDE_QSPI_MODEL</td>
<td>Set this option if the Quad Serial Port Interface (QSPI) Verilog models have been installed.</td>
</tr>
<tr>
<td>`INCLUDE_DAPLINK</td>
<td>Set this option to enable inclusion of the V2C-DAPLink peripherals. Supports lower external stimulus, longer resets, and drivers for Serial Wire Debug (SWD).</td>
</tr>
<tr>
<td>`DAPLINK_LINK_NF</td>
<td>If <code>INCLUDE_DAPLINK option is set, code is normally executed from the V2C-DAPLink QSPI model, and UART output directed to the V2C-DAPLink UART ports. If </code>DAPLINK_LINK_NF is also set, then code is executed from Instruction Tightly Coupled Memory (ITCM) and UART outputs are directed to the base board UART ports.</td>
</tr>
</tbody>
</table>

4.10.2 Executing code from QSPI

The Quad Serial Port Interface (QSPI) on the V2C-DAPLink is configured as an eXecute-In-Place (XIP) controller. Within the testbench, the V2C-DAPLink QSPI device model, S25fl128S, is preloaded with code from the qspi-a7.hex file. If `INCLUDE_DAPLINK is defined, and `DAPLINK_LINK_NF is not defined, then code is executed from the QSPI model.

Note: Code execution from the QSPI model is approximately ten times slower than the execution from the Instruction Tightly Coupled Memory (ITCM) RAM. This is because of the access of the QSPI and the subsequent data transfer through the AXI interconnect.

4.10.3 Wave files

By default, when Vivado activates the simulator window, it only shows the top-level signals. For QuestaSim, two preconfigured wave files are included, `wave_daplink.do` and `wave_no_daplink.do`. For the Vivado default simulator, `wave_daplink.wcfg` is provided.
Chapter 5
V2C-DAPLink board

The optional V2C-DAPLink adaptor board provides a debug flow that is familiar to anyone who is used to working with Cortex-M microcontrollers. It allows Arty FPGA boards to be used with mbed OS 2 Classic. This chapter describes the optional V2C-DAPLink adaptor board and how it is used.

It contains the following sections:

• 5.1 V2C-DAPLink adaptor board features on page 5-52.
• 5.2 V2C-DAPLink configuration on page 5-54.
• 5.3 Flash download requirements on page 5-55.
• 5.4 V2C-DAPLink board layout on page 5-56.
• 5.5 Conditions to enable the DAP interface on page 5-58.
• 5.6 DAP drivers on page 5-59.
• 5.7 Programming the V2C-DAPLink QSPI using drag and drop on page 5-60.
• 5.8 Using the μVision debugger to communicate through V2C-DAPLink on page 5-62.
• 5.9 Using the μVision debugger to download projects through the flash programming utility on page 5-64.
• 5.10 Recovering the DAP connection on page 5-67.
5.1 V2C-DAPLink adaptor board features

The board supports the following features:

- Allows Arty Artix 7 (A7) and Spartan-7 (S7) FPGA boards to be used with mbed OS 2 Classic.
- V2C-DAPLink Serial Wire Debug (SWD) over USB.
- UART over USB.
- Dedicated Quad Serial Port Interface (QSPI) flash for code image.
- Micro-SD card for application use (SPI mode only).
- Allows stacking of standard Shield expansion boards.
- DAPLink USB Composite Device:
  - USB Mass Storage Device Class (MSC) for programming software images to block RAM and QSPI.
  - USB Communication Device Class (CDC) for UART debug with nSRST support.
  - USB Human Interface Device (HID) for CMSIS-DAP software debug.

The following figure shows the V2C-DAPLink adaptor board, the Arty header breakout pins, and the point where they are interfaced together (this is depicted in orange).

![V2C-DAPLink adaptor board diagram]

A dedicated microcontroller on the V2C-DAPLink board provides the interface between a micro-USB connector and the UART and Serial Wire Debug (SWD) interfaces. This is pre-loaded with firmware that is configured to permit drag-and-drop software download onto the on-board QSPI. Using this programming interface requires that the Xilinx QSPI controllers are implemented as shown in the example design (at the same memory locations). The flash programming routine is loaded into target RAM at address 0x10000000, which is the Instruction Tightly Coupled Memory (ITCM) upper alias. The
V2C-DAPLink firmware is not intended to work with any processor except a single Cortex-M1 instance as demonstrated in the example design. For more information on the flash programming routine and download requirements, see 5.3 Flash download requirements on page 5-55.

The V2C-DAPLink board has a reset switch for the Cortex-M1 processor, CS_nSRST, this reset is also driven from the V2C-DAPLink chip. CS_nSRST must be used to reset the processor nSYSRESET and peripherals, but not the processor DBGRESETn or the Debug Access Port (DAP) resets.
5.2 V2C-DAPLink configuration

The V2C-DAPLink board has a configuration jumper, J2. This is used to drive a detect signal to the example design, and has the following effects when used with the example design.

Jumper open

The processor boots from the Instruction Tightly Coupled Memory (ITCM) lower alias. The ITCM initialization is performed as part of FPGA programming on powerup. A debugger sees the ITCM at both \(0x00000000\) and \(0x10000000\). The QSPI on the V2C-DAPLink can be written or accessed using the normal mode peripheral at \(0x40020000\). The UART connection to the V2C-DAPLink is unused in this configuration.

Jumper closed

The processor boots from Quad Serial Peripheral Interface (QSPI) eXecute In Place (XIP). The upper ITCM alias at \(0x10000000\) is still initialized at FPGA powerup, but is available for application use. Breakpoints cannot be placed directly in the QSPI image. There is no built-in process to copy any code from the QSPI XIP region into ITCM.

The UART connection to the V2C-DAPLink is connected to the example design UART in this configuration.

Note

For more information on the memory map, see 4.3 Memory map on page 4-40.
5.3 Flash download requirements

The DAPLink processor on the V2C-DAPLink is pre-programmed with a flash download routine. This is used for drag-and-drop programming and debugger code download. To maintain compatibility with the pre-programmed image, you must retain the following components in your system.

<table>
<thead>
<tr>
<th>Base address</th>
<th>Interface path in example design</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>Daplink_if_0/axi_xip_quad_spi_0/AXI_FULL</td>
<td>Code execution from dedicated Quad Serial Port Interface (QSPI) on V2C-DAPLink memory interface.</td>
</tr>
<tr>
<td>0x40000000</td>
<td>Daplink_if_0/axi_xip_quad_spi_0/AXI_LITE</td>
<td>Configuration interface that is used to set QSPI clock polarity and clock phase for eXecute-In-Place (XIP) execution.</td>
</tr>
<tr>
<td>0x40020000</td>
<td>daplink_if_0/axi_quad_spi_0</td>
<td>Normal mode QSPI controller used to read, write, and verify code to the dedicated QSPI on the V2C-DAPLink memory interface.</td>
</tr>
</tbody>
</table>
| 0x40010000   | Daplink_if_0/axi_gpio_0 | Bit [0] is used to control muxing of the QSPI interface.  
 0 QSPI XIP mode. QSPI is read-only through the axi_xip_quad_spi_0. This is the setting for executing code from the V2C-DAPLink. This is default option.  
 1 QSPI read, write, and verify through the normal mode axi_quad_spi_0 controller. |

Note

There is another peripheral, axi_single_spi_0 on the V2C-DAPLink board. This is a normal mode SPI controller that is used to write to the V2C-DAPLink SD card slot. In the example design, this has a base address of 0x40030000. The address of this peripheral is not fixed, however, Arm recommends that you do not change the address unless required.

Caution

Bit [0] of axi_gpio_0 must be held LOW while V2C-DAPLink code is executing. If your code must be run from V2C-DAPLink, then you must ensure that your code does not set this signal HIGH.
5.4 V2C-DAPLink board layout

The V2C-DAPLink adaptor board layout is based on the Arduino Shield form factor. The following figure shows the board layout.

![V2C-DAPLink board layout](image)

**Figure 5-2 V2C-DAPLink board layout**

The optional V2C-DAPLink board features are supported on the inner row expansion pins. The shield adaptor pins pass through to a shield board above the optional V2C-DAPLink board. The following table shows the Shield I/O pin mapping.

<table>
<thead>
<tr>
<th>I/O pin</th>
<th>Artix 7 bank</th>
<th>SPARTAN-7 bank</th>
<th>V2C-DAPLink signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>14</td>
<td>14</td>
<td>SD_nSS</td>
</tr>
<tr>
<td>27</td>
<td>14</td>
<td>14</td>
<td>SD_MISO</td>
</tr>
<tr>
<td>28</td>
<td>14</td>
<td>14</td>
<td>SD_MOSI</td>
</tr>
<tr>
<td>29</td>
<td>14</td>
<td>14</td>
<td>SD_SCLK</td>
</tr>
<tr>
<td>30</td>
<td>14</td>
<td>14</td>
<td>QSPI_Q0</td>
</tr>
<tr>
<td>31</td>
<td>14</td>
<td>14</td>
<td>QSPI_Q1</td>
</tr>
<tr>
<td>32</td>
<td>14</td>
<td>14</td>
<td>QSPI_Q2</td>
</tr>
<tr>
<td>33</td>
<td>14</td>
<td>14</td>
<td>QSPI_Q3</td>
</tr>
<tr>
<td>34</td>
<td>CONFIG</td>
<td>14</td>
<td>RSVD (V2C-DAPLink fitted)</td>
</tr>
<tr>
<td>35</td>
<td>14</td>
<td>14</td>
<td>QSPI_CLK</td>
</tr>
<tr>
<td>36</td>
<td>14</td>
<td>14</td>
<td>QSPI_nS</td>
</tr>
<tr>
<td>I/O pin</td>
<td>Artix 7 bank</td>
<td>SPARTAN-7 bank</td>
<td>V2C-DAPLink signal</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
<td>----------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>37</td>
<td>14</td>
<td>CONFIG</td>
<td>UART_RX</td>
</tr>
<tr>
<td>38</td>
<td>14</td>
<td>CONFIG</td>
<td>UART_TX</td>
</tr>
<tr>
<td>39</td>
<td>14</td>
<td>CONFIG</td>
<td>CS_nSRST</td>
</tr>
<tr>
<td>40</td>
<td>14</td>
<td>14</td>
<td>CS_DIO</td>
</tr>
<tr>
<td>41</td>
<td>14</td>
<td>14</td>
<td>CS_CLK</td>
</tr>
</tbody>
</table>

Table 5-2 Shield I/O mapping (continued)
5.5 Conditions to enable the DAP interface

The V2C-DAPLink board provides a USB interface to the Cortex-M1 design Serial Wire Debug (SWD) connections.

For the V2C-DAPLink board to work, the implementation in the Arty Artix 7 (A7) board must contain a Cortex-M1 processor supporting SWD with the Quad Serial Port Interface (QSPI) flash interfaces present. For more information on memory map configuration, see Chapter 4 Working with the Cortex®-M1 DesignStart™ example design on page 4-36.

The Cortex-M1 processor is an integral part to program QSPI using the Debug Access Port (DAP). To debug or program using the DAP, the processor must be in a valid state of execution. Corrupt software can cause the system to lock. If this happens, you might need to perform a recovery procedure. For more information, see 5.10 Recovering the DAP connection on page 5-67.
5.6 DAP drivers

The Debug Access Port (DAP) device issues USB codes for many devices to the host PC.

- Mbed VFS USB drive (Microsoft drivers).
- USB Serial Device (Mbed or Microsoft drivers).
- DAP interface.

Useful references

- https://os.mbed.com/handbook/CMSIS-DAP.
5.7 Programming the V2C-DAPLink QSPI using drag and drop

To program the V2C-DAPLink Quad Serial Port Interface (QSPI) using the drag and drop mechanism:

**Procedure**

1. Configure the Arty Artix 7 (A7) board with a valid Cortex-M1 processor design. Program the Arty A7 board with the reference MCS flash file. For more information on loading the flash file, see 4.8 Loading the flash file on page 4-47. This is required for step 6 in this procedure which causes a suitable image to be loaded into the FPGA which supports V2C-DAPLink.

2. Connect the V2C-DAPLink board to the Arty A7 board headers.

3. You must ensure that the V2C-DAPLink jumper is connected to J2, Cfg.

4. You must power the Arty A7 board by connecting the USB to the host.

5. You must power the V2C-DAPLink board by connecting the USB to the host.

   a. You can now connect a UART terminal program to both USB serial ports that the base Arty board and V2C-DAPLink board create. Both UARTs have settings of Baud rate 115,200 8 bits One stop No parity. With the J2 CFG jumper fitted, the terminal output from the FPGA is directed to the V2C-DAPLink UART. With J2 removed, the output is directed to the Arty board UART.

6. Press PROG on the Arty A7 board to ensure that it has configured the FPGA.

7. Press nRST on the V2C-DAPLink board to perform a clean reboot of the software that is programmed to the V2C-DAPLink QSPI device. The V2C-DAPLink might be programmed with a Cortex-M-compatible software image, however, this might not match the hardware design which you are using.

8. The host displays a file window similar to the following figure:

   ![File window](image)

   **Figure 5-3 File window**

9. The V2C-DAPLink QSPI can be programed with the qspi_a7.bin file generated as part of the software compilation flow. For more information, see 6.5.1 Software design post processing on page 6-78. This .bin file is automatically produced when the software is compiled and it is located in /software/m1_for_artxy_a7/Build_keil/qspi_a7.bin.

10. Drag and drop qspi_a7.bin onto This PC\MBED V2C.

   The drive for This PC\MBED V2C disappears, the V2C-DAPLink QSPI is programmed, and the drive reappears. If there are any errors, they are reported in a text file, Fail.txt. After the drag and drop file
transfer has completed, the new software runs when the processor is reset. For example, when the nRST button on the V2C-DAPLink board is pressed.

——— Caution ————

You cannot program the V2C-DAPLink QSPI device using drag and drop if either of the following Cortex-M1 configuration options have been selected:

- No Debug. See 3.2 Debug tab on page 3-28.
- No DTCM. See 3.4 Data Memory tab on page 3-32.
5.8 Using the μVision debugger to communicate through V2C-DAPLink

To set up a μVision project to communicate through V2C-DAPLink:

Procedure
1. Load the project and then go to Options for Target <name of executable> (alt+F7).
2. Select the Debug tab.
3. On the right-hand side of the screen deselect Load Application at Startup.
4. Select Use:, and then select CMSIS-DAP Debugger from the drop-down menu.

5. Click Settings and select the subsequent Debug tab.

Figure 5-4 Debug tab
6. Ensure that SWJ is ticked and select CMSIS-DAP from the drop-down menu. The IDCODE must read a valid value and the device name must indicate ARM CoreSight SW-DP.

7. Click OK in the Cortex-M Target Driver Setup and Options for Target <name of executable> screens.

8. You can now connect the debugger to the target by clicking on the debug icon.
5.9 Using the μVision debugger to download projects through the flash programming utility

To set up a μVision project to download projects through the flash programming utility, you must have the correct driver installed.

The file S25FL128S_V2C.FLM must first be copied to C:\Keil_v5\ARM\Flash, or wherever your Keil installation is. This file can be found in the V:\software\flash_downloader directory.

**Procedure**

1. Load the project and then go to *Options for Target <name of executable>* (alt+F7)
2. Select the Debug tab.
3. Click on Settings and select the subsequent Flash Download tab
4. Click Add and select the driver file S25FL128S_V2C.
5. Click Add and check that the Start and Size text boxes are filled with \texttt{0x10000000} and \texttt{0x1000} respectively.

6. Click OK.

7. Select the Utilities tab from the \textit{Options for Target <name of executable>} window and select \textit{Use Target Driver for Flash Programming} and tick \textit{Use Debug Driver}.

--- Caution ---

You cannot use the \muVision debugger to download projects through the flash programming utility if either of the following Cortex-M1 configuration options have been selected:

- No Debug. See \textit{3.2 Debug tab} on page 3-28.
- No DTCM. See \textit{3.4 Data Memory tab} on page 3-32.
8. Click OK.
9. Click on the download icon to download flash.

---

Figure 5-9 Utilities tab

Figure 5-10 Download flash icon
5.10 Recovering the DAP connection

If you program the Quad Serial Port Interface (QSPI) with software that causes the processor to lock up, the QSPI might become unaccessible. To recover the Debug Access Port (DAP) connection, a valid image must be programmed into the V2C-DAPLink QSPI or the device must be erased.

Procedure
1. Configure the Arty Artix 7 (A7) board with a valid Cortex-M1 processor design.
2. Connect the V2C-DAPLink to the Arty boards headers.
3. Ensure the V2C-DAPLink jumper is removed from J2, Cfg.
4. Connect the USB to the host to power:
   - The Arty board.
   - The V2C-DAPLink board.
5. Press PROG on the Arty board to ensure it has configured the FPGA.
6. Connect to the DAP with the μVision debugger.
7. Load the project and then go to the Options for Target <name of executable> (alt-F7).
8. Select the Debug tab.
9. Click on Settings and go to the Flash Download tab.
10. Ensure Program and Verify are unticked.

![Cortex-M Target Driver Setup](image)

Figure 5-11 Flash Download tab

11. Click OK in the Cortex-M Target Driver Setup and Options for Target <name of executable> screens.
12. Click the download icon to erase the device.
13. Replace the J2, Cfg link on the V2C-DAPLink and press nRST.
Chapter 6
Example software design

This chapter describes an example software design, and describes how to build and debug it.

Software for the Cortex-M1 processor can be run either from the Instruction Tightly Coupled Memory (ITCM), initialized as part of the FPGA image, or from an external AXI memory.

It contains the following sections:

- 6.1 Example software design for Arty A7 on page 6-70.
- 6.2 Example software design directory structure on page 6-71.
- 6.3 Example design reference files on page 6-72.
- 6.4 Generating the Arty A7 board support package on page 6-73.
- 6.5 Building the example software design on page 6-78.
- 6.6 Software update flow on page 6-79.
6.1 Example software design for Arty A7

An example software design is provided which demonstrates the basic functionality of the processor and some peripherals.

The example design software design is compiled using Arm µVision Microcontroller Development Kit (MDK) 5.24 onwards. A project file for the example design is in V:/software/m1_for_arty_a7/Build_Keil/m1_for_arty_a7.uvprojx. The example software design uses compiler options for the Cortex-M0 processor. This is the correct choice if your toolchain does not provide explicit support for the Cortex-M1 processor.

The software demonstrates:

- UART output to either the Arty onboard USB connector or the V2C-DAPLink board when fitted.
- GPIO_0, the LEDs mirror the state of the DIP switches. When each switch is turned on, the appropriate LED is lit.
- GPIO_1, as each pushbutton is pressed, the appropriate LED rotates around eight possible colors (seven lit states and a single off state).
- QSPI_0, read and write accesses are testing during powerup.
- BRAM ctrl 0, read and write memory accesses are tested during powerup.

The peripherals on the V2C-DAPLink board are not covered by the software tests.

The example software design relies on the Xilinx Board Support Package (BSP) for the example design. You must generate the BSP before you build the software design.
6.2 Example software design directory structure

The software structure provided uses the Xilinx software framework for the AXI peripherals and combines this with Arm CMSIS software for the Cortex-M1 processor.

```plaintext
<installation_directory>
  |_software/
     |_m1_for_arty_a7/
        |_Build_Keil/
        |_cmsis/
        |_gpio/
        |_main/
        |_spi/
        |_uart/
        |_sdk_workspace/
```

The following table describes the directory structure.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>software/m1_for_arty_a7/Build_Keil/</td>
<td>Build directory.</td>
</tr>
<tr>
<td>software/m1_for_arty_a7/cmsis/</td>
<td>Cortex-M1 CMSIS included files and bootfiles.</td>
</tr>
<tr>
<td>software/m1_for_arty_a7/gpio/</td>
<td>User GPIO routines that reference Xilinx GPIO driver.</td>
</tr>
<tr>
<td>software/m1_for_arty_a7/main/</td>
<td>Top-level files.</td>
</tr>
<tr>
<td>software/m1_for_arty_a7/spi/</td>
<td>SPI routines that reference the SPI driver.</td>
</tr>
<tr>
<td>software/m1_for_arty_a7/uart/</td>
<td>User UART routines that reference Xilinx UART driver.</td>
</tr>
<tr>
<td>software/m1_for_arty_a7/sdk_workspace/</td>
<td>Location of Software Development Kit (SDK) build Board Support Package (BSP) files.</td>
</tr>
</tbody>
</table>
6.3 Example design reference files

A number of reference design files are provided with the delivery.

The following table describes these example design reference files in hardware `\m1_for_arty_a7\m1_for_arty_a7`.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bram_a7.elf</td>
<td>Example design software binary for Cortex-M processors with debug symbols in Elf_Dwarf format.</td>
</tr>
<tr>
<td>bram_a7.hex</td>
<td>Example design software hex file loaded into FPGA build of Cortex-M Instruction Tightly Coupled Memory (ITCM).</td>
</tr>
<tr>
<td>m1.mmi</td>
<td>Example design memory map information. This is used to merge elf and bit files.</td>
</tr>
<tr>
<td>m1_for_arty_reference.bit</td>
<td>Example design with software included to load into FPGA RAM.</td>
</tr>
<tr>
<td>m1_for_arty_reference.mcs</td>
<td>Example design with software included to load into Arty board configuration flash.</td>
</tr>
</tbody>
</table>

The following table describes these example design reference files in software `\m1_for_arty_a7\Build_Keil`.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bram_a7.elf</td>
<td>Example design software binary for Cortex-M processors with debug symbols in Elf_Dwarf format.</td>
</tr>
<tr>
<td>bram_a7.hex</td>
<td>Example design software hex file loaded into FPGA build of Cortex-M ITCM.</td>
</tr>
<tr>
<td>qspi_a7.bin</td>
<td>Example design software binary in QSPI format. This can be loaded by drag-and-drop using V2C-DAPLink mass storage.</td>
</tr>
<tr>
<td>qspi_a7.hex</td>
<td>Example design software hex file in QSPI format.</td>
</tr>
</tbody>
</table>
6.4 Generating the Arty A7 board support package

Before compiling the example software design that you are provided, a Board Support Package (BSP) is created using the Vivado Software Development Kit (SDK). The example software design includes files and directories that the BSP creates.

To generate a Cortex-M1 BSP for the Arty Artix 7 (A7) board:

Procedure
1. Open Vivado.
2. Open the design found in V:/hardware/m1_for_arty_a7/m1_for_arty_a7/m1_for_arty_a7.xpr.
3. If the original design has been modified, including changing the address map, then proceed and follow steps 4 and 5. If the hardware design is unchanged, proceed to step 6.
4. Select Generate Block Diagram from the left-hand side pane and then select Generate. This directs Vivado to generate the file required files for synthesis, implementation, and simulation for the block diagram.
5. Select File → Export Hardware. Set the Exported location to V:/software. The dialog box that opens prompts that an exported module for the file is already found. Click Yes to overwrite this file.
6. Select File → Launch SDK. Set Exported location to V:/software and workspace to V:/software/m1_for_arty_a7/sdk_workspace. Click OK to proceed.
7. Vivado SDK launches and automatically opens the hardware platform specification for the Arty A7 example design. The following image shows the memory map that is displayed. The memory map displayed aligns with the map described in 4.3 Memory map on page 4-40.

![Figure 6-2 Launch SDK](image_url)
Figure 6-3  Memory map

a. Confirm that under Xilinx → Repositories, the global repository list includes V:/vivado/Arm_sw_repository.

9. Set the design name to standalone_bsp_0.
10. Click Finish.
On the Standalone tab, ensure that stdin and stdout are set to use axi_uartlite_0.

**Caution**

The SDK does not read the stdin and stdout values unless they are changed. This is a known issue, and therefore, you must set stdin and stdout to none, and then set them back to axi_uartlite_0.

Click Finish. The SDK generates the required BSP files.

The following directory structure now exists as V:/software/m1_for_arty_a7/sdk_workspace/standalone_bsp_0/CORTEX_M1_0/. The common Xilinx include files are in the /include directory. The driver files for the selected peripherals and the standalone BSP core files are in the /libscr directory.
12. The xpseudo_asm_rcvt.h and xpseudo_asm_rcvt.c files must be manually copied from V:/vivado/Arm_sw_respository/Cortex/bsp/standalone_v6_7/src/arm/cortexm1/armcc to V:/software/m1_for_arty_a7/sdk_workspace/standalone_bsp_0/CORTEX_M1_0/include directory because of differences between the Vivado SDK and Arm Keil Microcontroller Development Kit (MDK).

The BSP is complete and is now ready fo use by the example software design for compilation.

------- Note -------

The BSP header file, xparameters.h, is located in the V:/software/m1_for_arty_a7/sdk_workspace/standalone_bsp_0/CORTEX_M1_0/include. This header file includes definitions for all memory addresses and peripheral configurations. It is automatically generated from the hardware platform specification. To enable tightly coupled hardware and software configurations Arm recommends that you use the configuration definitions from this file.

------- Caution -------

If the xparameters.h file does not contain entries for STDIN_BASEADDRESS or STDOUT_BASEADDRESS, then the stdin and stdout locations are not correctly set. This results in no UART output. The standalone_bsp_0 directory should be removed, and the BSP regenerated.

Next Steps

You must now proceed to 6.5 Building the example software design on page 6-78.
6.5 Building the example software design

The example software design is built using the Arm Keil μVision Microcontroller Development Kit (MDK) tool.

To build the example software design:

**Prerequisites**

- You must complete the steps in 6.4 Generating the Arty A7 board support package on page 6-73.
- The example design Keil project uses a post processing batch file, `make_hex_a7.bat`, which is automatically run after compilation. This batch file creates the necessary software files in the desired formats and copies the files to their respective locations. This batch file requires the executable `fromelf.exe` to be in the users path. This executable is located in `<Keil install path>/ARM/ARMCC/bin`. This location should be added to the users path before opening the Keil project.

**Procedure**

1. Open Arm Keil μVision MDK and navigate to Project -> Open Project.
2. Select V:/software/m1_for_arty_a7/Build_Keil/m1_for_arty_a7.uvprojx.
3. Ensure that the target is `m1_for_arty_a7`.
4. Navigate to Project -> Rebuild. This rebuilds all target files.

**6.5.1 Software design post processing**

The target file, `m1_for_arty_a7.axf`, is generated in `/Build_Keil/objects`. There is a post-process batch file, `make_hex_a7.bat` that the design calls automatically when the target is built. The batch file converts the `.axf` file to suitable `.hex`, `.bin`, and `.elf` files. The batch file automatically copies the relevant output files to the appropriate hardware project directories.

Therefore, when the design is rebuilt in Arm Keil μVision Microcontroller Development Kit (MDK), new `.elf` and `.hex` files are present in the filepath V:/hardware/m1_for_arty_a7/m1_for_arty_a7.

**Note**

For V2C-DAPLink drag and drop operation, `qspi_a7.bin` is created as part of the software design post processing process. This file is present in the `/Build_Keil` directory. This file can be directly copied to the V2C-DAPLink drive. The `.hex` file that the batch file generates is intended for use with the Vivado tools, and it does not work for drag and drop programming.

**Caution**

If the example design has no output to the UART, but the rest of design runs correctly on the board, that is, the LEDs respond to the push button changes, the cause is the generation of the standalone BSP, in particular, the setting of the `stdin` and `stdout` locations. For more information on changing the `stdin` and `stdout` locations, see 6.4 Generating the Arty A7 board support package on page 6-73. You must delete the current `standalone_bsp_0` directory and regenerate.
6.6 Software update flow

To avoid rebuilding the FPGA each time the software is modified, you can update the BRAM memories content in an existing bit file with the new software content.

This mechanism requires the following:
- A bit file of latest hardware design, containing the Cortex-M1 processor data and instruction memories inferred as RAM36 primitives.
- A Memory Map Information (MMI) file. The MMI file lists the mapping of the Cortex-M1 buses to the RAM36 primitives, and their location. The MMI file only changes when the hardware has been rebuilt. It does not require regeneration for each software iteration.
- A Software .elf file output from the software compilation tool flow.
- A batch file to combine these three files and produce a new bit file.

6.6.1 Generating the MMI file

The Memory Map Information (MMI) file maps the bit lanes from the data and instruction buses in the Cortex-M1 processor to specific RAM36 primitives and their locations.

The MMI file is updated whenever the FPGA design is rebuilt and a new bit file generated.

Note
It is not necessary to produce an MMI file each time the software is rebuilt. The MMI file reflects the current hardware build within the FPGA, and as such it is paired with each bit file.

You must generate the MMI file manually following these steps:

Procedure
1. In Vivado, after a bit file is produced, open the implemented design.
2. Open the TCL console.
3. Navigate to V:/hardware/m1_for_arty_a7/m1_for_arty_a7.
4. To create the file m1.mmi in the current directory, at the prompt type source make_mmi_file.tcl.

6.6.2 Generating bit and flash files

In the V:/hardware/m1_for_arty_a7/m1_for_arty_a7 folder, there is a Windows batch file, make_prog_bit.bat. The make_prog_bit.bat file combines the m1_for_arty_a7_wrapper.bit, m1.mmi, and bram_a7.elf files into a bit file, m1_for_arty_a7.bit and a flash file m1_for_arty_a7.mcs.

To create a new m1_for_arty_prog.bit bit file in the current directory:

Prerequisites
The make_prog_bit.bat batch file requires that:
- A m1_for_arty_a7_wrapper.bit bit file is in \m1_for_arty_a7_35.runs\impl_1\.
- An m1.mmi file is in the current directory.
- A bram_a7.elf file is in the current directory.

The Vivado executable must be in your path. To test this, open a command window or console, and type the following:

vivado

Procedure
1. Open a command window in the V:/hardware/m1_for_arty_a7/m1_for_arty_a7 folder.
2. Check that the make_prog_bit.bat file is configured.
3. At the prompt, execute the `make_prog_bit.bat` file.
4. Check the console messages to ensure that both `m1_for arty_a7.bit` and `m1_for arty_a7.mcs` files have been generated.

### 6.6.3 Programming

To program the example software design:

**Procedure**

1. In Vivado, open the hardware manager and auto-connect to the Arty Artix 7 (A7) board.
2. Select *Program Device*. By default, Vivado selects the original bit file created `m1_for_arty_a7_wrapper.bit`.
3. Navigate to `V:/hardware/m1_for_arty_a7/m1_for_arty_a7`.
4. Select the `m1_for_arty_prog.bit` file generated in 6.6.2 Generating bit and flash files on page 6-79.
5. Select *Program*. The bit file with the latest software updates is now programmed on the board.
Appendix A
Revisions

This appendix describes the technical changes between released issues of this document.

It contains the following section:
• \textit{A.1 Revisions} on page Appx-A-82.
A.1 Revisions

This appendix describes the technical changes between released issues of this book.

Table A-1 Issue 0000_00

<table>
<thead>
<tr>
<th>Change</th>
<th>Location</th>
<th>Affects</th>
</tr>
</thead>
<tbody>
<tr>
<td>First release for r0p0</td>
<td>-</td>
<td>None</td>
</tr>
</tbody>
</table>

Table A-2 Differences between Issue 0001_00 and Issue 0000_00

<table>
<thead>
<tr>
<th>Change</th>
<th>Location</th>
<th>Affects</th>
</tr>
</thead>
<tbody>
<tr>
<td>First release for r0p1</td>
<td>Document history table.</td>
<td>First documentation release for r0p1</td>
</tr>
<tr>
<td>Functionality for no simulation with QSPI</td>
<td>2.5 Installing shell models on page 2-21</td>
<td>First documentation release for r0p1</td>
</tr>
<tr>
<td>models and no simulation at all support added.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The ability to select the number of interrupts using the configuration tab has been removed.</td>
<td>3.1 Configuration tab on page 3-26</td>
<td>First documentation release for r0p1</td>
</tr>
<tr>
<td>No DTCM option added, and the DTCM size range changed.</td>
<td>3.4 Data Memory tab on page 3-32</td>
<td>First documentation release for r0p1</td>
</tr>
<tr>
<td>No debug option added</td>
<td>3.2 Debug tab on page 3-28</td>
<td>First documentation release for r0p1</td>
</tr>
<tr>
<td>Additional pre-requisite added.</td>
<td>6.5 Building the example software design on page 6-78</td>
<td>First documentation release for r0p1</td>
</tr>
</tbody>
</table>