ARM Cortex™-M0

r0p0-02rel0

Release Note

25 September 2009

Release Note - ARM Cortex-M0 r0p0-02rel0

Proprietary notice

Document confidentiality status

This document is confidential except disclosure permitted to “Designers”.

Product status

The information in this document is for a product at Early Access quality status.

Web address

http://www.arm.com

Feedback

ARM limited welcomes feedback on both the product, and the documentation.

Feedback on the Cortex-M0 processor

If you have any comments or suggestions about this product or its deliverables, please contact support-cores@arm.com giving:

- The product name
- A concise explanation of your comments.

Feedback on this document

If you have any comments on about this document, please send email to errata@arm.com giving:

- The document title
- The document number
- The page number(s) to which your comments refer
- A concise explanation of your comments

General suggestions for additions and improvements are also welcome.

Contents

1 PRODUCT DELIVERABLES

1.1 Product Release Status

This is the Early Access release of the ARM Cortex-M0 processor at revision r0p0.

Early Access release status has a particular meaning to ARM of which the recipient must be aware. A deliverable so designated has satisfactorily achieved all criteria for its promotion to a Mature Release status. It may be delivered in accordance with the contract and be expected to perform as described in the data-sheet. However, there remain some elements of uncertainty, which cannot finally be validated until the deliverable has been successfully deployed by customers. Accordingly, the recipient of a deliverable with Early Access Release status may be directly contributing to the final stage of approval of that deliverable.

In due course, after the Early Access Release the product status will change to Mature. It should be noted that Support for the Early Access release of the deliverable will only be provided by ARM to a recipient who has a current support and maintenance contract for the deliverable.

NOTE:
- These deliverables may only be used as described in the terms of your legal agreement.

1.2 About the Cortex-M0 Processor

Cortex-M0 is a highly deterministic, low gate count, 32-bit processor that implements the ARMv6-M architecture with zero deviation instruction determinism and zero jitter interrupts for zero wait-state memory systems. It has a 3-stage pipeline and very low area and power (half that of Cortex-M3) and approximately 0.89 DMIPS/MHz performance. The Cortex-M0 programmer's model is compatible with Cortex-M1 and Cortex-M3 for portability.

Cortex-M0 is a synthesizable Verilog RTL design that is written for ASIC implementation. Note that Verilog 2001 language constructs have been used in the RTL. Cortex-M0 includes the following features:

- Configurable multiplier (either 1-cycle or 32-cycle)
- System Control Block
- Optional SysTick system timer
- Nested Vector Interrupt Controller
- Configurable debug
- Single 32-bit wide AHB-Lite memory interface for instruction and data transactions and all debugger accesses to the external system
    - The *AMBA 3 AHB-Lite Protocol* (ARM IHI 0033A) is available on the ARM documentation website http://infocenter.arm.com/

Full details of the Cortex-M0 processor are contained in the Technical Reference Manual (TRM) document and other technical publications documents, see section 3.1.

During the development of Cortex-M0, a coverage-driven verification methodology has been applied using both random and directed stimuli and has been supplemented by formal property checking and hardware prototyping. The design has undergone significant CPU core validation and verification commensurate with an ARM Early Access release and the deliverables have been subjected to extensive out of the box testing. This release successfully boots μC/OS-II, ThreadX, RTX and uCLinux operating systems.

1.3 About the Cortex-M0 Release Note

This release note contains information about the usage, limitations and quality status of the accompanying deliverables. Table 1 lists the ARM part numbers for the individual deliverables included in the release of this ARM product. With the exception of the Architecture documents, the deliverables are provided under their individual part numbers of the format AT510-xx-nnnnn-r0p0-nnrel0. Except where indicated, these deliverables are released together as a single deliverables bundle, AT510-BU-50000-r0p0-02rel0.

TECHNICAL DOCUMENTATION

Part number	Description	Format	Revision
AT510-DC-06003	Cortex-M0 Release Note (this document)	PDF	r0p0
AT510-DA-00001	Cortex-M0 Technical Reference Manual	FrameMaker	r0p0
AT510-DA-03001	Cortex-M0 Technical Reference Manual	PDF	r0p0
AT510-DA-00005	Cortex-M0 User Guide Reference Material	FrameMaker	r0p0
AT510-DA-03005	Cortex-M0 User Guide Reference Material Example	PDF	r0p0
AT510-DA-04005	Cortex-M0 User Guide Reference Material	XML Source	r0p0
AT510-DC-70047	Cortex-M0 Integration & Implementation Manual	PDF	r0p0
AT510-MN-70030	Cortex-M0 IP-XACT - SPIRIT description	XML	r0p0
AT510-DC-11001	Cortex-M0 Errata List [1]	PDF	r0p0
AR085-DA-70000	ARMv6-M Architecture Reference Manual [1]	PDF	r0p0
AR085-DC-11001	ARMv6-M Architecture Errata List [1]	PDF	r0p0

IMPLEMENTATION

Part number	Description	Format	Revision
AT510-MN-22110	Cortex-M0 Synthesizable Verilog	Text	r0p0
AT510-MN-70005	Integration Synthesizable RTL Verilog	Text	r0p0
AT510-RM-70000	Cadence Reference Implementation Flow [1]	Text, PDF	r0p0
AT510-RM-00002	Synopsys Reference Implementation Flow [1]	Text, PDF	r0p0

INTEGRATION

Part number	Description	Format	Revision
AT510-MN-70009	Cortex-M0 Integration Kit	Text	r0p0

VALIDATION

Part number	Description	Format	Revision
AT510-MN-22010	Vector Capture and Replay Test Bench	Text	r0p0
AT510-VE-09001	Portable Power Indicative Test Source	Text	r0p0
AT510-VE-70006	Portable Functional Test Source	Text	r0p0
AT510-VE-70007	Portable Speed Indicative Test Source	Text	r0p0
AT510-VE-70025	Portable Maximum Power Test Source	Text	r0p0

Table 1: Part numbers for the ARM Cortex-M0 Deliverables

Notes to Table 1

[1] These parts are not released as part of the deliverables bundle AT510-BU-50000 and must be merged into the deliverables installation area by the licensee (see Section 2.1.2)

2 INSTALLATION

2.1 Introduction

Intellectual Property (IP) deliverables are delivered as one or more UNIX zipped tar files which have been security encrypted. These installation instructions cover the Unix operating system only.

2.1.1 Unpacking the Deliverables

Information on how to download and unpack the deliverables is contained in the release email notification from ARM. Click on the Connect http links in the release email and click on "Add to Download" for each bundle or deliverable. After all items have been selected for download, click on the "download" button and wait for the transaction to be built. The window will be refreshed to show the size of the transaction, a checksum number and a link called "Download Now" at the bottom of the page which can be used to download the file. Save the `arm-download-nnnnn.tgz` file and use the GNU `gtar` utility to unpack it with the following Unix command:

```
% gtar xzf arm-download-nnnnn.tgz
```

For each download from the ARM Connect IP Delivery Server, two extra files are created as listed below. In the names of the delivered files, `nnnnn` is a unique delivery number. These files should be used to view the contents (parts or files) of the delivery or to investigate possible download corruption problems.

1. **`ARM_DELIVERY_nnnnn.TXT`** lists the downloaded parts and the constituent parts of any downloaded bundle.
2. **`ARM_MANIFEST_nnnnn.TXT`** contains a manifest of all the files included in the transaction, together with their checksums. The checksums provided are calculated using the RSA Data Security, Inc. MD5 Message-Digest Algorithm. The checksums can be used to verify the integrity of the data using the `md5sum` tool (which is part of the GNU `textutils` package) by running (in Unix):

   ```
   % md5sum --check ARM_MANIFEST_nnnnn.TXT
   ```

2.1.2 Merging the Deliverables

After unpacking the deliverables using GNU `gtar`, each bundle or separate deliverable will be contained in its own directory. An example list is given below although the exact version numbers (eg r0p0-00rel0) may be higher than those shown.

```
AT510-BU-50000-r0p0-02rel0/
AT510-DC-11001-r0p0-00rel0/
AT510-RM-00002-r0p0-00rel0/
AT510-RM-70000-r0p0-00rel0/
AR085-DA-70000-r0p0-02rel0/
AR085-DC-11001-r0p0-02rel0/
```

Use the following steps to merge the deliverables into a single installation directory:

1. The errata PDF document contained in `AT510-DC-11001-r0p0-00rel0/` must be copied into the `doc/` directory of `AT510-BU-50000-r0p0-02rel0/`. Use the following Unix command:

   ```
   % cp AT510-DC-11001-r0p0-00rel0/* AT510-BU-50000-r0p0-02rel0/doc
   ```

2. The ARMv6-M Architecture Reference Manual PDF documents contained in `AR085-DA-70000-r0p0-00rel0/`must be copied into the `doc/` directory of `AT510-BU-50000-r0p0-02rel0/`.

   ```
   % cp -r AR085-DA-70000-r0p0-02rel0/* AT510-BU-50000-r0p0-02rel0
   ```

3. The ARMv6-M Architecture Errata List PDF documents contained in `AR085-DC-11001-r0p0-02rel0/`must be copied into the `doc/` directory of `AT510-BU-50000-r0p0-02rel0/`.

   ```
   % cp AR085-DC-11001-r0p0-02rel0/doc/errata/* AT510-BU-50000-r0p0-02rel0/
        doc/ARMv6-M_architecture
   ```


4. The contents of each supplied implementation -RM- deliverable must be copied into the downloaded bundle directory so that it appears at the same directory level as `logical/` (see Figure 1 on page 6). For example, using the following Unix command:

```
% cp -r AT510-RM-70000-r0p0-00rel0/* AT510-BU-50000-r0p0-02rel0
```

NOTE:

- When copying more than one RM deliverable, some errors similar to that below may be seen. These are expected and occur when the same files are used in multiple RM deliverables.

```
cp: cannot create regular file `AT510-BU-
50000-r0p0-02rel0/implementation_tsmc90_g_lowk/design_kits/arm/fe_tsmc090g_sc-ad
v_v10_2007q4v2/aci/sc-ad/doc/tsmc090adgrvt.pdf': Permission denied
```

5. If desired, the bundle directory can be renamed to `Cortex-M0-r0p0-02rel0` using:

```
% mv AT510-BU-50000-r0p0-02rel0 Cortex-M0-r0p0-02rel0
```

6. The deliverables are now ready for use.

2.1.3 Directory Structure

Once the deliverables have been correctly combined, the directory structure should appear as shown in Figure 1. The Cortex-M0 Integration and Implementation Manual (ref 3.1) gives more details on files and directory structure.

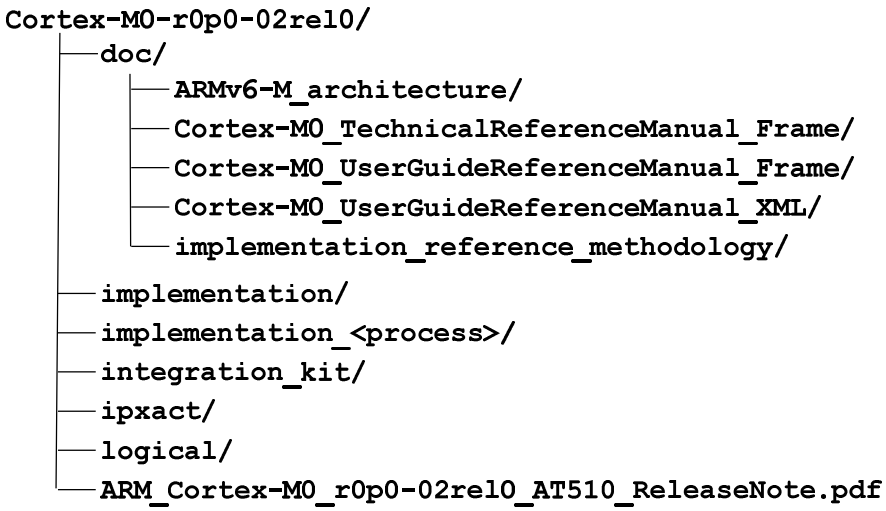

Figure 1: Cortex-M0 Development Environment Directory Structure

Directory	Contents
doc/	Contains the Cortex-M0 Technical Reference Manual, Integration and Implementation Manual and User Guide Reference Material.
doc/ARMv6-M_architecture	Contains the ARMv6-M architecture documentation
doc/implementation_reference_methodology	Contains Cortex-M0 EDA specific implementation documentation associated with the Reference Methodology (RM) deliverables
implementation/	Contains the vector replay testbenches
implementation_<process>/	Contains the Reference Methodology (RM) implementation scripts targeted at a particular process and EDA vendor eg `implementation_tsmc90_g_lowk`. Documentation for the RM scripts is located in the `doc/` directory.
integration_kit/	Contains the Cortex-M0 integration kit, vector capture testbench and test vector source files
ipxact/	Contains the Cortex-M0 IP-XACT description based on version 1.2 of the SPIRIT specification
logical/	Contains the Verilog RTL of the Cortex-M0 processor , the integration level and example components like the PMU and reset controller

Table 2: Cortex-M0 Release Directory Content Descriptions

3 DOCUMENTATION

3.1 Cortex-M0 Technical Publications Documents

The following technical publications are included in the `doc/` directory of this release.

1. Cortex-M0 Technical Reference Manual (TRM) ARM DDI0432B

   The Technical Reference Manual provides details for programming the Cortex-M0 processor.
2. Cortex-M0 Integration and Implementation Manual (IIM) ARM DII0238A

   The Integration and Implementation Manual provides some implementation guidelines and gives details for configuring the RTL, performing vector capture and replay, using the integration kit and information for integrating Cortex-M0 into a system.
3. Cortex-M0 User Guide Reference Material (UGRM) ARM DUI0467B

   The User Guide Reference Material provides reference material that can be configured and included in a User Guide or similar document for a product that is based on the Cortex-M0 processor.

3.2 Architecture Documents

The following architecture documents should accompany this release. ARM recommends that they are copied into the `doc/ARMv6-M_architecture` directory of this release as described in section 2.1.2

1. DDI0419B_arm_architecture_v6m_reference_manual_errata_markup_1_0.pdf

   This is a marked up version of revision B of the ARMv6-M Architecture Reference Manual showing fixes to known errata. This is the latest version of the ARMv6-M ARM and must be used by all licensees
2. DDI0419B_arm_architecture_v6m_reference_manual.pdf

   This is revision B of the ARMv6-M Architecture Reference Manual and is for reference only. It has been superseded by the marked up PDF described in 1 above.
3. ARMv6-M_Architecture_Errata_List.pdf

   This is a simple errata document referencing the new marked up PDF of the ARMv6-M ARM.

NOTE:

- DDI0419B_arm_architecture_v6m_reference_manual_errata_markup_1_0.pdf is the latest version of the ARMv6-M ARM and must be used by all licensees

3.3 README Files

The following README file is contained in this release:

- `README_IPXACT` is located in the `ipxact/` directory and gives details on the Cortex-M0 IP-XACT descriptions in this release

3.4 Errata Document

`Cortex-M0_Errata_vX.0.pdf` (where X refers to the version number) released under part number AT510-DC-11001 contains the latest information on known errata with this product. This deliverable is not released as part of the deliverables bundle, but should accompany this release. If new errata are found subsequent to this release, then the errata document will be updated and released to all licensees.

4 TOOL VERSIONS

Table 3 lists the tool versions used during the development of Cortex-M0 and are therefore known to work. Other tool versions may be used, but they have not been tested by ARM.

Tools used to develop the ARM Cortex-M0		
Platform OS	Red Hat Linux	Enterprise 4
ARM RealView Tools	RealView Debugger Suite (RVDS)	4.0
	RealView Compiler Tools (RVCT)	4.0
	RealView MDK	3.42b, 3.42c, 3.5
	CMSIS (Cortex Microcontroller Software Interface Standard) [1]	V1.10
Simulation Tools	Cadence IUS	8.10.007, 8.20.009
	Mentor ModelSim MTI	6.1f, 6.4b
	Synopsys VCS	2008.09
Cadence Implementation Tools	Conformal	7.2
	Encounter Test	6.2.5
	Encounter Timing System	7.1
	Fire & Ice QRC	EXT 7.1
	RTL Compiler	7.2
	SOC Encounter	7.10-USR2-s140_1
	VoltageStorm	ANLS 7.11-s014_1
	CPF	V1.0
Synopsys Implementation Tools	Design Compiler	2008.09-SP2
	IC Compiler	2008.09-SP2
	Formality	2008.09-SP2
	MVRC	2008.12
	MVSIM	2008.12
	PrimeRail	2008.12
	PrimeTime PX	2008.06-SP3
	PrimeTime SI	2008.06-SP3
	Star RCXT	2008.12
	TetraMAX	2008.09-SP2
	UPF	V1.0
Other Tools	Perl	5.8.7
	IP-XACT / SPIRIT	V1.2

Table 3: Cortex-M0 Tools and Versions

Notes to Table 3

[1] The latest version of CMSIS is available for free download from http://www.onarm.com

5 KNOWN LIMITATIONS OF THIS RELEASE

5.1 Expected Synthesis Warnings

Cortex-M0 is highly configurable and implements this configurability using Verilog parameters. These parameters are used throughout the design to allow a synthesis tool to remove logic that is not required in the chosen configuration. As such, the following types of synthesis warning are to be expected:

- Unused registers
- Unused block inputs
- Constant values on outputs or registers

5.2 SRPG Support

Cortex-M0 has been designed for deployment in a multi-power domain system to maximize static power efficiency. This deployment is intended to be carried out via the use of CPF/UPF and example CPF and UPF files have been provided in the release as described in section 5.2.1. The CPF and UPF files specify the same power intent but use different commands as required by different EDA tools.

5.2.1 CPF / UPF File Locations

CPF/UPF files are located in three locations within the full set of Cortex-M0 deliverables:

1. **RTL**: These files, located in `logical/models/cpf` and `logical/models/upf`, describe the power intent of the Cortex-M0 processor. In the future, when hierarchical CPF/UPF is fully supported by the power aware tools, this version of the CPF/UPF file will not require modification and will be referenced by other CPF/UPF files. However, until then some implementation specific details may be present and require changing.
2. **Integration kit**: These files, located in `integration_kit/validation/srpg`, describe the power intent of the Cortex-M0 integration kit example MCU system (CM0IKMCU module). This file is intended as an example which may be modified by the licensee to suit their particular system. The example MCU is divided into 3 power domains as described below. Further detailed information is contained in the CPF / UPF files.
    a. **Always on** power domain. This includes the system RAM, PMU, WIC, clock and reset controller, part of the Cortex-M0 DAP and GPIO blocks to generate an interrupt to wake-up the processor

    NOTE:
    - Some blocks are required to be always on in the integration kit to allow for the required integration testing of the Cortex-M0 processor. In a real system, it may be more appropriate to put some of the GPIO blocks in the system power domain.

    b. **System** power domain. This includes the processor system power domain, the AHB interconnect, the ROM and the STCLKEN generator

    c. **Debug** power domain contains the processor debug power domain and part of the Cortex-M0 DAP
3. **Vector Replay**: These files, located in `implementation/vectors/CORTEXM0IMP/srpg` and `implementation/vectors/CORTEXM0INTEGRATIONIMP/srpg` support power aware replay of vectors on the RTL. The only difference between the vector replay CPF/UPF files and those with the RTL is the top level module name.

NOTE:
- All CPF/UPF files must be consistent. Any relevant changes should be reflected in all versions.

5.2.2 CPF Support

The example CPF (Common Power Format) files have been used extensively in simulation with Cadence IUS and have been used to create functional SRPG (State Retained Power Gated) netlists on a single library with SRPG support. The CPF files provided conform to the CPF 1.0 specification. However, due to incomplete support of this standard by IUS and implementation tools, some aspects of the files cannot be tested fully.

Specifically, the power gate acknowledge signals are not automatically driven in simulation. To work around this limitation, the integration kit Verilog drives the acknowledge signals instead.

5.2.3 UPF Support

The example UPF (Unified Power Format) files have been used extensively in simulation with MVSIM and VCS only and have been used to create functional SRPG (State Retained Power Gated) netlists on a single library with SRPG support. The UPF files provided conform to the UPF 1.0 specification. However, due to incomplete support of this standard by the MVSIM and implementation tools, some aspects of the files cannot be tested fully.

Specifically, the use of Boolean expressions to control the polarity of the power-switch acknowledge signals is not currently supported in MVSIM or the implementation tools. This means that for simulation:

- MVSIM will assume a fixed polarity of the power-switch acknowledge signal and
- The implementation tools will connect the power-switches without adding invertors as required to match the requirements in the UPF against the library cell specification.

As a result:

- Workarounds may be required in UPF simulations to ensure that the power-switch acknowledge signals are correctly interfaced to your system Power Management Unit (PMU)
- UPF RTL simulations and netlist simulations may behave differently

For this reason, SRPG implementation sign-off must include netlist simulations with all workarounds for power-switch control removed.

The UPF supplied assumes:

- A HIGH level on the power-switch input implies a request to power-down
- A HIGH level on the power-switch output implies that the switch has turned the power off

5.3 Integration Kit

- The following EDA tool combinations are supported for multi-voltage RTL simulation of a state-retained power gated (SRPG) implementation
  - Synopsys MVSIM with VCS and UPF
  - Cadence IUS (includes NCSIM) and CPF
- SRPG netlist simulation support:
  - The final SRPG netlist which includes the full power network can be simulated with full power-down behavior without the need for any power-aware tools.
  - Intermediate SRPG netlists may be simulated using power aware tools in the integration kit with the appropriate processed CPF/UPF files but will require modifications to the `integration_kit/validation/RunIK` script and the `integration_kit/logical/tbench/verilog/rtl.vc` file
- Due to incomplete support of the CPF/UPF v1.0 standards by the power aware tools (refer to section 5.2 for details), workarounds are required in the integration kit verilog to ensure correct verification of the RTL in an SRPG implementation. These work-arounds are not required for SRPG netlist simulations. You must therefore ensure that you simulate your SRPG netlist or replay vectors before signing off your SRPG implementation. There are verilog `'defines` to make this process as user-friendly as possible. Details of the available SRPG-related `` `defines `` are contained in the `integration_kit/logical/tbench/verilog/rtl.vc` file.

- The integration kit tests do not exhaustively test the Cortex-M0 processor I/O because not all I/O pins have an effect that is visible to program code and some pins are meant to be static. As such the passing of all integration kit tests must not be used as the only sign-off criteria for successful processor integration.
- MDK version 3.50 and earlier do not support big-endian compilation for Cortex-M0 targets
- Depending on the compiler used, there may be some benign warnings when compiling Dhrystone

5.4 Vector Replay

- The following EDA tool combinations are supported for multi-voltage RTL simulation of a state-retained power gated (SRPG) implementation
    - Synopsys MVSIM with VCS and UPF
    - Cadence IUS (includes NCSIM) and CPF
- For CPF simulations use the `-cpf` switch of the Replay script in conjunction with `-sprg` switch
- No netlist simulations are supported with power aware tools
    - The final SRPG netlist which includes the full power network can be simulated with full power-down behavior without the need for any power-aware tools.
    - Intermediate SRPG netlist may be simulated with the appropriate CPF/UPF files in the Replay testbench, but will require modifications to the `implementation/vectors/tools/Replay` script and the following files:
        - `implementation/vectors/CORTEXM0IMP/tbench/tbench.vc` and
        - `implementation/vectors/CORTEXM0INTEGRATIONIMP/tbench/tbench.vc`

6 REVISION HISTORY

Date	Revision - Version	Description
27 March 2009	r0p0-00rel0	Early Access release r0p0
3 June 2009	r0p0-01rel0	Updated integration kit and integration manual with improved support for SRPG implementations, MDK projects and the addition of a system ROM table example
25 September 2009	r0p0-02rel0	Addition of the User Guide Reference Material deliverables Replacement of the Configuration & Sign-off Guide and the Integration Manual with a combined document - the Integration and Implementation Manual Minor update to the TRM to reflect the new documentation above

Internal document reference: **PR353-PRDC-011235** v**1.0**