µC/FS™ File System for CrossCore® Embedded Studio version 1.1.0 Release Notes

Introduction

This document contains the release notes for µC/FS™ File System for CrossCore® Embedded Studio version 1.1.0. It describes the release in detail and provides latest information that supplements the main documentation.

Users of previous releases should check the "Version Compatibility" section for pertinent instructions on modifying existing applications for this new release.

Support and Assistance

There are several options for contacting support:

- Submit your questions online at http://www.analog.com/support

- E-mail your Processor and DSP software and development tools questions from within CrossCore Embedded Studio.

  Go to "Help->E-mail Support…". This will create a new e-mail addressed to processor.tools.support@analog.com, and will automatically attach your CrossCore Embedded Studio version information (ProductInfo.html).

- E-mail your Processors and DSP applications and processor questions to:
  - processor.support@analog.com OR
  - processor.china@analog.com (Greater China support)
- Post your questions in the Processors and DSP online technical support community in Engineer Zone at http://ez.analog.com/community/dsp

Update Highlights

The focus of µC/FS™ File System for CCES version 1.1.0 is to support the release of ADSP-BF70x processor family.

The version of the Micriµm's file system included with µC/FS™ File System for CCES version 1.1.0 has been upgraded to v4.07.00.

Micriµm's file system documentation has migrated to their website at address http://doc.micrium.com instead of being included with µC/FS™ File System for CrossCore® Embedded Studio. Any documentation specific to Analog Devices processors can still be found within this product.

New supported Processors

µC/FS™ File System for CrossCore® Embedded Studio 1.1.0 supports all the processors supported by µC/FS™ 1.0.2. The newly supported processors are:

- ADSP-BF700, ADSP-BF701, ADSP-BF702, ADSP-BF703, ADSP-BF704, ADSP-BF705, ADSP-BF706, ADSP-BF707
    - Devices supported: RAM Disk, SD Card, USB Mass Storage

New add-ins to support ADSP-BF707 ROM

ADSP-BF70x processors include a Utility ROM which contains the µC/OS-III Real-Time Kernel and µC/LIB and µC/CPU which are required by µC/OS-III. To support the utility ROM, µC/FS includes new add-ins that do not add µC/LIB and µC/CPU to the project.To use µC/FS in a project with µC/OS-III in ROM, you must choose the ROM-specific µC/FS add-ins. If you have uC-CPU from uC-FS (i.e non-ROM uCFS) and the uCOS3 ROM then you will get double definitions and link errors.

Further, using the µC/FS components to support the µC/OS-III ROM configurations without a µC/OS-III ROM add-in results in compilation failures since the µC/LIB and µC/CPU headers and sources are not in the project.

Micriµm software versions

µC/FS™ File System for CrossCore® Embedded Studio version 1.1.0 is based on Micriµm's µC/FS™ File System version 4.07.00

There are several CrossCore Embedded Studio add-ins based on Micriµm's products which share common add-ins. To ensure that the same version of these add-ins is used by all the add-ins that require them, these add-ins are installed in a common location which is distinct from the µC/OS-FS install folder. These common add-ins are

- µC/CPU which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-CPU v1.1.0. This installation includes µC/CPU version 1.30.01.
- µC/LIB which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-LIB v1.1.0. This installation includes µC/LIB version 1.38.00. This is the minimum version of µC/LIB required to build µC/FS 4.07.00. Version 1.38.00 of µC/LIB has deprecated some APIs that were used by previous versions of µC/FS, most notably `Mem_PoolBlkGetUsedAtIx()` and `Mem_PoolBlkIxGet()`. Updating to µC/FS V4.07.00 is required if updating other Micrium products that require µC/LIB V1.38.00.
- µC/CLK which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-CLK v1.1.0. This installation includes µC/CLK version 3.09.03

Version Compatibility

µC/FS™ File System requires CrossCore® Embedded Studio version 1.1.0 or later for all processors.

µC/FS™ File System version 1.1.0 requires requires µC/LIB version 1.1.0 or later. This is the version of µC/LIB included with this product.

µC/FS™ File System for CrossCore® Embedded Studio Software Anomalies

This section enumerates the most significant anomalies which relate to µC/FS™ File System for CrossCore® Embedded Studio. For a comprehensive list of all the public software anomalies visit http://www.analog.com/SoftwareAnomalies

Anomalies fixed in version 1.1.0

Anomalies fixed by the Micriµm upgrade to version 4.07.00

- SD Driver: issue STOP_TRANSMISSION command only once per stop operation.
- Missing err code init in FSDev_Access(Lock|Unlock).
- Mounting logical partition fails if extended partition type is LBA extended (0xF).
- NAND Driver: incorrect data size allocation for Micron ECC and Soft ECC.
- FS_FAT_JournalOpen(): erroneous journal file's cluster count calculation when the journal size is smaller than the cluster size.
- NAND Driver errors in 16 bits defect mark checking.
- NAND Driver: add support for switching to 16 bits width

Known issues with µC/FS™ File System for CrossCore® Embedded Studio 1.1.0

These are the currently known problems which affect µC/FS™ File System for CrossCore® Embedded Studio.

UCFS-217 Upgrading the file system add-in to 1.1.0 results in warnings

CCES produces some unexpected warnings when upgrading the uC-FS component to 1.1.0.

The following warnings can be ignored safely and the resulting project should build without issues.
Property "uccpu-ucfs-compatibility" required by Add-in "..." doesn't exist
Property "uclib-ucfs-compatibility" required by Add-in "..." doesn't exist

µC/FS™ File System for CrossCore® Embedded Studio version 1.0.2 Release Notes

Introduction

This document contains the release notes for µC/FS™ File System for CrossCore® Embedded Studio version 1.0.2. It describes the release in detail and provides latest information that supplements the main documentation.

Users of previous releases should check the "Version Compatibility" section for pertinent instructions on modifying existing applications for this new release.

For product support assistance, please contact our Processor Tools Support Team at processor.tools.support@analog.com.

Update Highlights

The focus of µC/FS™ File System for CCES version 1.0.2 is to support the release of the Mass Storage Class (MSC) component. The MSC file system component is required to use the µC/USB Host Stack add-in for CrossCore® Embedded Studio Mass Storage Class component.

For new projects, the option to align buffers on cache lines is set by default since it provides better performance out of the box. To create a project with the exact same settings as in previous versions, you must uncheck the relevant box in the configuration window.

The version of the Micriµm's file system included with µC/FS™ File System for CCES version 1.0.2 contains major bug fixes in the NAND device support and therefore it is highly recommended that NAND projects are updated. Similarly, the journaling module has been fully redesigned so any journalled volume used under prior versions must be cleanly unmounted before upgrading to µC/FS™ File System for CCES version.

Supported Processors

The new USB Mass Storage Class component is supported in the following processors

- ADSP-BF522, ADSP-BF524, ADSP-BF526, ADSP-BF523, ADSP-BF525, ADSP-BF527
- ADSP-BF542, ADSP-BF542M, ADSPBF547, ADSP-BF547M, ADSP-BF548, ADSP-BF548M, ADSP-BF549, ADSP-BF549M
- ADSP-BF606, ADSP-BF607, ADSP-BF608, ADSP-BF609

Since µC/FS™ File System for CrossCore® Embedded Studio 1.0.2 also supports the same processors as version 1.0.1 the full list of processors supported is:

- ADSP-BF504, ADSP-BF504F, ADSP-BF506F
    - Devices supported: RAM Disk, SD Card
- ADSP-BF512, ADSP-BF514, ADSP-BF516, ADSP-BF518
    - Devices supported: RAM Disk, SD Card
- ADSP-BF522, ADSP-BF524, ADSP-BF526, ADSP-BF523, ADSP-BF525, ADSP-BF527
    - Devices supported: RAM Disk, NAND Flash, USB Mass Storage
- ADSP-BF531, ADSP-BF532, ADSP-BF533, ADSP-BF534, ADSP-BF536, ADSP-BF537, ADSP-BF538, ADSP-BF539
    - Devices supported: RAM Disk only
- ADSP-BF542, ADSP-BF542M, ADSPBF547, ADSP-BF547M, ADSP-BF548, ADSP-BF548M, ADSP-BF549, ADSP-BF549M
    - Devices supported: RAM Disk, SD Card, IDE, NAND Flash, USB Mass Storage
- ADSP-BF544, ADSP-BF544M,
    - Devices supported: RAM Disk
- ADSP-BF561 RAM Disk
    - Devices supported: RAM Disk
- ADSP-BF592-A
    - Devices supported: RAM Disk
- ADSP-BF606, ADSP-BF607, ADSP-BF608, ADSP-BF609
    - Devices supported: RAM Disk, SD Card, USB Mass Storage

Micriµm software versions

µC/FS™ File System for CrossCore® Embedded Studio version 1.0.2 is based on Micriµm's µC/FS™ File System version 4.06.01

There are several CrossCore Embedded Studio add-ins based on Micriµm's products which share common add-ins. To ensure that the same version of these add-ins is used by all the add-ins that require them, these add-ins are installed in a common location which is distinct from the µC/OS-FS install folder. These common add-ins are:

- µC/CPU which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-CPU v1.0.3. This installation includes µC/CPU version 1.29.02.03.

- µC/LIB which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-LIB v1.0.3. This installation includes µC/LIB version 1.37.01.
- µC/CLK which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-CLK v1.0.1. This installation includes µC/CLK version 3.09.03

Version Compatibility

µC/FS™ File System version 1.0.2 requires CrossCore® Embedded Studio version 1.0.2 or later.

The Mass Storage Class component of this release requires the use of µC/USB Host™ Stack which, in turn, requires device drivers which are only available in CrossCore® Embedded Studio version 1.0.3. If you are using CrossCore® Embedded Studio version 1.0.2 then you will also need the USB host device drivers which have been provided separately.

µC/FS™ File System for CCES version 1.0.2 upgrades the release of Micriµm's file system to the latest currently available. This version of µC-FS introduces an incompatibility in the NAND flash support due to changes in the low level formatting. If a NAND device was used with a prior version of µC-FS then you should either:

- Re-format the NAND flash prior to use (including both volume and low level formatting). This is the recommended solution since the new low level formatting settings are the ones preferred by Micriµm for all new development. To help with the formatting the NAND examples provided with the µC/FS™ File System for CrossCore® Embedded Studio version 1.0.2 installation contain the appropriate APIs to execute the formatting. See each example's readme for further information.
- Edit your code to maintain compatibility. To do this you need to:

  a) Define the macro APP_CFG_FS_NAND_UB_CNT_MAX=10 in the compiler preprocessor options
  b) Edit the file fs_app.c and add the following code to the function App_FS_AddNAND. Note that this code is already included in the latest fs_app.c template.

  ```
  #ifdef APP_CFG_FS_NAND_UB_CNT_MAX

   nand_cfg.UB_CntMax = APP_CFG_FS_NAND_UB_CNT_MAX;

  #endif /* APP_CFG_FS_NAND_UB_CNT_MAX */
  ```

Software requirements

In order to use all the add-ins which are part of µC/FS File System version 1.0.2 to a project we recommend that you use CrossCore Embedded Studio version 1.0.2 or later.

In order to add the Mass Storage Class component of µC/FS to a project you need to use µC/USB Host™ Stack version 1.0.0 or later.

µC/FS™ File System for CrossCore® Embedded Studio Software Anomalies

This section enumerates the most significant anomalies which relate to µC/FS™ File System for CrossCore® Embedded Studio. For a comprehensive list of all the public software anomalies visit http://www.analog.com/SoftwareAnomalies

Anomalies fixed in version 1.0.2

TAR-48701/ UCFS-120: uCFS: The file system removes the file app_timing.c when it shouldn't

When all the file system components are removed from a project, the file app_timing.c is also removed. Since this is a file that customer's may have modified uC/FS should leave it in the project so customers can remove it when they want

µC-FS™ File System for CrossCore® Embedded Studio version 1.0.1 Release Notes

Introduction

This document contains the release notes for µC/FS™ File System for CrossCore® Embedded Studio version 1.0.1. It describes the release in detail and provides latest information that supplements the main documentation.

This release includes support for the processors listed in the next section, below. Users of previous releases should check the "Version Compatibility" section, below, for pertinent instructions on modifying existing applications for this new release.

For product support assistance, please contact our Processor Tools Support Team at <processor.tools.support@analog.com>.

Update Highlights

Supported Processors

This release of µC-FS™ File System for CrossCore® Embedded Studio adds support for the following Blackfin processors:

- ADSP-BF504, ADSP-BF504F, ADSP-BF506F
  - Devices supported: RAM Disk, SD Card
- ADSP-BF512, ADSP-BF514, ADSP-BF516, ADSP-BF518
  - Devices supported: RAM Disk, SD Card
- ADSP-BF522, ADSP-BF524, ADSP-BF526, ADSP-BF523, ADSP-BF525, ADSP-BF527
  - Devices supported: RAM Disk, NAND Flash
- ADSP-BF531, ADSP-BF532, ADSP-BF533, ADSP-BF534, ADSP-BF536, ADSP-BF537, ADSP-BF538, ADSP-BF539
  - Devices supported: RAM Disk only
- ADSP-BF542, ADSP-BF542M, ADSPBF547, ADSP-BF547M, ADSP-BF548, ADSP-BF548M, ADSP-BF549, ADSP-BF549M
  - Devices supported:RAM Disk, SD Card, IDE, NAND Flash
- ADSP-BF544, ADSP-BF544M,

    - Devices supported: RAM Disk
- ADSP-BF561 RAM Disk
    - Devices supported: RAM Disk
- ADSP-BF592-A
    - Devices supported: RAM Disk

As with µC-FS™ File System for CrossCore® Embedded Studio 1.0.0, this release also supports the following Blackfin processors:

- ADSP-BF606, ADSP-BF607, ADSP-BF608, ADSP-BF609
    - Devices supported:RAM Disk, SD Card

Micriµm software versions

µC-FS™ File System for CrossCore® Embedded Studio version 1.0.1 is based on Micriµm's µC-FS™ File System version 4.05.01

There are several CrossCore Embedded Studio add-ins based on Micriµm's products which share common add-ins. To ensure that the same version of these add-ins is used by all the add-ins that require them, these add-ins are installed in a common location which is distinct from the µC-FS install folder. These common add-ins are

- µC/CPU which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-CPU v1.0.1. This installation includes µC/CPU version 1.29.01.
- µC/LIB which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-LIB v1.0.1. This installation includes µC/LIB version 1.37.00.
- µC/CLK which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-CLK v1.0.1. This installation includes µC/CLK version 3.09.03

The documentation for all of these add-ins can be found in CrossCore® Embedded Studio Help under µC/FS™ 1.0.1

Supported devices

IDE device support

Configuring the ATAPI interface with the Pin Multiplexing add-in

Double-clicking the system.svc opens the System Configuration utility, which also shows the configuration tabs for each add-in. Select the “Pin Multiplexing” tab to open the configuration options.

The “peripherals” list displays all the peripherals that can be configured. Enable the “ATAPI [ATAPI - data signals muxed with the async bus]” tick box to generate the required code in the Pin Multiplexing add-in that will configure the ATAPI interface for the ADSP-BF548 EZ-Kit.

NAND device support

Supported NAND chips:

- Micron 2Gbit 29F2G08 (ADSP-BF548 EZ-Kit Lite BOM Rev 2.5 and later)
- Micron 4Gbit 29F4G08 (ADSP-BF527 EZ-Kit Lite BOM Rev 3.4 and later)
- ST Micro 2Gbit NAND02 (earlier ADSP-BF548 EZ-Kit Lite)
- ST Micro 4Gbit NAND04 (earlier ADSP-BF527 EZ-Kit Lite)

Configuring the NAND interface with the Pin Multiplexing add-in

Double-clicking the system.svc opens the System Configuration utility, which also shows the configuration tabs for each add-in. Select the “Pin Multiplexing” tab to open the configuration options.

The “peripherals” list displays all the peripherals that can be configured. For ADSP-BF54x, enable the “NAND [NAND Flash Module]” tick box. For ADSP-BF52x, enable the “NAND [NAND Flash Controller on PORTH]” tick box. These options generate the required code, on their respective platforms, in the Pin Multiplexing add-in that will configure the NAND Flash Controller.

Configuring the NAND device with the µC/FS™ add-in

Double-clicking the system.svc opens the System Configuration utility, which also shows the configuration tabs for each add-in. Select the “µC/FS” tab and the "NAND" page to open the configuration options.

Micron 29F2G08 and 29F4G08

- Type of error correction: Micron hardware ECC
- Enable automatic configuration (ONFI): checked

ST Micro NAND02

- Type of error correction: Software ECC(1-bit)
- Free spare area start: 2
- Free spare area length (bytes): 60
- Enable automatic configuration (ONFI): unchecked
- Bus width (bits): 8
- Number of blocks in device: 2048
- Number of pages per block: 64
- NAND page size (bytes): 2048
- Spare area size (bytes): 64

- Factory defect mark type: Spare byte/word 1 in 1st or last page
- Codeword size required for ECC (bytes): 528
- Maximum number of bad blocks: 40
- Maximum number of erases on a single block: 100000
- Number of partial page programming allowed before erase: 4

ST Micro 4Gbit NAND04

- Type of error correction: Software ECC(1-bit)
- Free spare area start: 2
- Free spare area length (bytes): 60
- Enable automatic configuration (ONFI): unchecked
- Bus width (bits): 8
- Number of blocks in device: 4096
- Number of pages per block: 64
- NAND page size (bytes): 2048
- Spare area size (bytes): 64
- Factory defect mark type: Spare byte/word 1 in 1st or last page
- Codeword size required for ECC (bytes): 528
- Maximum number of bad blocks: 80
- Maximum number of erases on a single block: 100000
- Number of partial page programming allowed before erase: 4

Note that NAND Flash and SD Card cannot be used simultaneously on ADSP-BF54x processors, The reason is that both RSI and NFC are treated as a single peripheral by the DMA channel peripheral mapping.

RAM Disk device support

The RAM Disk device support existing in µC/FS version 1.0.0 has been extended to support further Blackfin processors as specified in the supported processors section above.

SD Card device support

Configuring the RSI/SDH interface with the Pin Multiplexing add-in

Double-clicking the system.svc opens the System Configuration utility, which also shows the configuration tabs for each add-in. Select the “Pin Multiplexing” tab to open the configuration options.

The “peripherals” list displays all the peripherals that can be configured. For ADSP-BF60x, enable the “RSI0 [RSI Module]” tick box. For ADSP-BF50x and ADSP-BF51x, enable the "SD-RSI [SD/RSI Module]" tick box. For ADSP-BF54x, enable the "SDH [Secure Digital Host]" tick box. These options generate the required code, on their respective platforms, in the Pin Multiplexing add-in that will configure the SD Card interface.

Note that NAND Flash and SD Card cannot be used simultaneously on ADSP-BF54x processors, The reason is that both RSI and NFC are treated as a single peripheral by the DMA channel peripheral mapping.

Version Compatibility

The add-In framework provided with CCES 1.0.0 and µC-FS 1.0.0 itself have several undocumented issues related to add-In upgrades. Because of this it is recommended that you do not use the CCES add-in framework to upgrade existing projects that use µC/FS 1.0.0 add-ins to use µC/FS 1.0.1. Instead, it is recommended that entirely new projects are created

Software requirements

µC/FS File System for CrossCore Embedded Studio requires CrossCore Embedded Studio version 1.0.1 or later to be installed to build projects.

Adding µC/FS to a project

When a µC/FS device is added to a CrossCore Embedded Studio project, the CCES add-in framework executes the following actions

- Adds links to the file system files in the installation folder
- Adds generated files based on the configuration window settings
- Sets up the required toolchain options.
- Copies fs_app.c to your project. This file is a copy of a micrium template which contains funtions for file system initialization. This file can be edited by the application developer and will not be removed from the project when the file system is removed or overwritten if it is already present in the destination folder.
- Copies app_timing.c to your project. This file contains template time-stamping interface for uC-FS when not using an RTOS. This file can be edited by the application developer and will not be removed from the project when the file system is removed or overwritten if it is already present in the destination folder.

µC/FS for CrossCore Embedded Studio version 1.0.1 contains changes to the files fs_app.c and fs_app.h which are required to support the new NAND flash add-in. For this reason it is recommended that all projects start with a new version of these files. If you already have an older version of fs_app.c and fs_app.h in your project we recommend that you copy new versions to your project from the path where you installed µC/FS. The files are located in uC-FS\common\uC-FS\APP\Template.

MISRA-C Compliance Conformance Checking

The Motor Industry Software Reliability Association (MISRA) published a set of guidelines for the C programming language to promote best practice in developing safety related electronic systems in road vehicles and other embedded systems. The CrossCore® Embedded Studio compiler fully supports the MISRA-C 2004 Guidelines, and can detect violations of the rules at compile-time, link-time, and run-time.

The µC/FS File System for CrossCore Embedded Studio complies with MISRA by documenting known violations of the MISRA Compliance standard. The violations mean that µC/FS sources will not compile with MISRA conformance checking enabled, and any file that includes µC/FS headers will also not compile. The file {µC/FS File System install folder}\uC-FS\Docs\FS MISRA-C 2004 Compliance Matrix.xls contains the list of MISRA-C violations by µC/FS.

Known issues with µC/FS™ File System for CrossCore® Embedded Studio

These are the currently known problems which affect µC/FS™ File System for CrossCore® Embedded Studio.

TAR-49883 ADI_NFC doesn't support DMA on BF52x

The Crosscore Embedded Studio NFC driver in CCES 1.0.1 uses 32-bit DMA operations which cannot be used with the NAND Flash Controller on ADSP-BF52x processors. For this reason, Programmed I/O must be used instead.

In order to use Programmed I/O uC-FS for CCES defines the macros _ADI_NFC_PIO_READS and _ADI_NFC_PIO_WRITES when CCES 1.0.1 is used which instruct its BSP to use the Programmed I/O NFC interface. In later versions of CCES uC-FS will default to use DMA instead. In order to keep on using Programmed I/O applications can define _ADI_NFC_PIO_READS and _ADI_NFC_PIO_WRITES.

Note that the NFC driver is delivered as part of the CCES product.

µC/FS™ File System for CrossCore® Embedded Studio version 1.0.0 Release Notes

What is µC/FS File System for CrossCore Embedded Studio

µC/FS™ File System for CrossCore® Embedded Studio is the result of a partnership between Analog Devices and Micriµm to provide a user-friendly programming environment for µC/FS applications running on Analog Devices' processors. µC/FS™ File System for CrossCore® Embedded Studio provides an integrated environment with CrossCore Embedded Studio which offers the advantage of an industry-standard IDE combined with Analog Devices' advanced optimizing compiler technology.

µC/FS is a compact, reliable, high-performance file system based on clean, consistent ANSI C source code. It supports the FAT file system for interoperability with all major operating systems.

The memory footprint of µC/FS can be adjusted at compile time based on required features and the desired level of run-time argument checking. For applications with limited RAM, features such as cache and read/write buffering can be disabled; for applications with sufficient RAM, enabling these features improves performance.

µC/FS can access multiple media simultaneously, including multiple instances of the same type of medium (since all drivers are re-entrant). In addition, a logical device driver is provided so that a single file system can span several (typically identical) devices.

Features

- Integration with CrossCore® Embedded Studio for Analog Devices Processors Software
- POSIX-compatible interface for file access and directory access
- Processor-independent interface
- Ease of porting to new platforms
- Scalable RAM and ROM requirements
- Full FAT support including FAT12/16/32 and long file names (VFAT)
- Optional journaling component for failsafe FAT operation

Please contact Analog Devices for more information on purchasing the optional journaling component.

- Support for formatting and the creation of DOS partitions on a device

Capabilities

The primary file interface is the familiar POSIX interface, where Micrium-specific functions provide the same functionality as the following standard functions:

clearerr	fread	s_mkdir
closedir	fseek	opendir
fclose	fsetpos	readdir
feof	ftell	remove
ferror	ftruncate	rename
fflush	ftrylockfile	rewind
fgetpos	funlockfile	rmdir
flockfile	fwrite	setvbuf
fopen		

The µC/FS implementation provides the same POSIX interface, but uses a “fs_” prefix for each API. For example, µC/FS provides fs_feof(), which is equivalent to the POSIX feof() API.

Getting Started with µC/FS™ File System for CrossCore® Embedded Studio

Installation

CrossCore® Embedded Studio v.1.0.0 or newer must be installed prior to installing µC/FS™ File System.

Please make sure to close CrossCoreEmbedded Studio before proceeding with the installation. If CrossCore Embedded Studio is left open during the installation, it will have to be restarted after installing µC/FS File System.

µC/FS File System for CrossCore Embedded Studio installs the following:

- µC/FS File System. Its default installation directory is C:\Analog Devices\uCFileSystem-Rel1.0.0.
- µC/LIB . This software is always installed into Common Program Files directory. This location is determined by the %CommonProgramFiles(x86)% environment variable in 64-bit operating systems or by %CommonProgramFiles% in 32-bit operating systems.
- µC/CPU. This software is always installed into Common Program Files directory. This location is determined by the %CommonProgramFiles(x86)% environment variable in 64-bit operating systems or by %CommonProgramFiles% in 32-bit operating systems.

- µC/CLK. This software is always installed into Common Program Files directory. This location is determined by the %CommonProgramFiles(x86)% environment variable in 64-bit operating systems or by %CommonProgramFiles% in 32-bit operating systems.

Analog Devices strongly recommends installing µC/FS File System outside of the Program Files directory to prevent possible permission issues related to UAC (User Access Control). If you have already installed the product under Program Files then we recommend that you uninstall it and re-install it in a different location.

Note: Multiple versions of the µC/FS File System can be installed on the same system. Only a single instance of a specific version of the product can be installed on a system.

License Checking

The installation process checks for a valid license for µC/FS File System. If a valid license is not detected, the installer will start the Manage Licenses utility for entering and activating a license. The installer will fail in a non-interactive mode when valid license is not present.

Installation Logging

The installer does not create a log file by default. If you encounter installation issues, you can generate an installation log file by running the installer from the command prompt.

Change to the directory containing the downloaded installer executable and run the following from the command prompt:

```
ADI_uCFileSystem-Rel1.0.0.exe /v"/l*v c:\temp\installer.log"
```

License

The installation process checks for a valid license for µC/FS™ File System. Refer to the Licensing Guide in your CrossCore® Embedded Studio installation which can also be found in http://www.analog.com/CrossCoreLicensingGuide.

Support and Assistance

There are several options for contacting support:

- Submit your questions online at:

  http://www.analog.com/support
- E-mail your Processor and DSP software and development tools questions from within CrossCore Embedded Studio.

Go to "Help->E-mail Support…". This will create a new e-mail addressed to processor.tools.support@analog.com, and will automatically attach your CrossCore Embedded Studio version information (ProductInfo.html).

- E-mail your Processors and DSP applications and processor questions to:
  - processor.support@analog.com OR
  - processor.china@analog.com (Greater China support)
- Post your questions in the Processors and DSP online technical support community in Engineer Zone at

  http://ez.analog.com/community/dsp

Supported processors

µC/FS File System for CrossCore Embedded Studio 1.0.0 supports the ADSP-BF60x family of Blackfin processors.

Supported Devices

- SD Card, which uses the RSI interface.
- RAM Disk

Software requirements

µC/FS File System for CrossCore Embedded Studio requires CrossCore Embedded Studio version 1.0.0 or later to be installed to build projects.

Getting started with a project that uses the File System

Adding the File System to a project

Creating a new project which includes the File System

In order to create a project you should follow the instructions provided in the CrossCore Embedded Studio help. As part of the project creation, the page "Add-in Selection" contains a list of all the available add-ins for the project that you are creating, based on the products that you have installed and the processor and type of project selected. You can see the µC/FS add-ins in the Middleware section under "File System" and then "Micriµm uC-FS".

Adding the File System to an existing project

Every CrossCore Embedded Studio project contains a **System Configuration** file called system.svc which is located in the root of the project. The file is the IDE's interface for managing the various prewritten software components used in the "system" implemented by a project. Double-clicking any system.svc file in a navigation view opens that file in a System Configuration Utility which allows you to see the add-ins that you currently have in your project. Clicking on Add allows you to select one of the µC/FS add-ins in the Middleware section under “File System” and then “Micriµm uC-FS”.

Add-Ins

There are two add-ins supported with this release:

- uC-FS RAMDisk for Blackfin (1.0.0)
- uC-FS SD Card for Blackfin (1.0.0)

When you add an add-in the CrossCore Embedded Studio IDDE makes the appropriate changes to your project, including the necessary files, links and include folders. The add-ins can be used independently, or used together in the same project.

µC/FS project structure

When adding µC/FS to a CrossCore Embedded Studio project, the µC/FS-specific files get place in the system folder. Please do not change this organization. In the system folder the following structure gets created:

- A **uC-FS** folder

  This folder contains any sources and header files which are part of the µC/FS File System. It contains the folders:

  - **FAT, OS, Source**

    These folders contain the actual RTOS code which should not be modified.

- **Dev**

  This folder contains the files required for the devices used by each of the add-ins selected, including platform-dependant implementation (BSP) files. These files should not be modified.

- **GeneratedSources**

  This folder contains headers and sources generated by the product based on the GUI configurations. These files should not be modified.

- A **uC-CPU** folder

  This folder contains any sources and header files which are required by Micriµm uC/CPU software. uC/CPU provides a processor-independent interface to the supported processors and toolchains that is used in all Micriµm products.

- A **uC-LIB** folder

This folder contains any sources and header files which are required by Micriµm uC/LIB software. uC/LIB provides a clean and organized implementation of some of the most common standard library functions, macros and constants. uC/LIB is required by many Micriµm products including uC/OS-III.

- A **uC-CLK** folder

  This folder contains the sources and headers required by the µC/CLK software. µC/CLK provides the timing functionality required by the µC/FS File System.

- A **uC-Common** folder.

  This folder contains sources and headers which are common to several Micriµm products but that are not part of any Micriµm product themselves. These include for example app_cfg.h which is needed by all Micriµm applications.

System view

CrossCore Embedded Studio provides the System view which is used by µC/FS File System for CrossCore Embedded Studio. Use the System Configuration Overview tab to add a µC/FS add-in to a CrossCore Embedded Studio project.

To access the System Configuration Overview tab, do one of the following:

- In a navigation view, double-click the **system.svc** file of a project. The **System Configuration** utility appears with the overview tab selected.
- If the utility is already open, select the Overview tab.

For more information about the System view, see the CrossCore Embedded Studio help.

Configuration tabs

When a µC/FS™ File System for CrossCore® Embedded Studio add-in gets added to a project, several configuration tabs get added to the System view. These include tabs for Micriµm components which are required by several products like uC-LIB and a configuration tab for µC/FS. These configuration tabs provide an easy mechanism to generate any macro definitions required by the Micriµm products.

If the µC/FS add-in is used by a CrossCore Embedded Studio project, its configuration tab includes the following pages:

- µC/FS General
- FAT
- Build Settings
- Application General

Look in the CrossCore Embedded Studio help in the µC/FS node under µC/FS tab for more information about each of the configuration options.

Configuring the SD Card device

The SD Card device uses the RSI interface. This means that the pins related to the RSI need to be configured on your board before the SD Card device is used. We recommend using the Pin Multiplexing add-in for CrossCore® Embedded Studio to configure the RSI driver for use with the µC/FS SD Card device. When selecting which add-in are used in your application, ensure that the “Pin Multiplexing (1.0.0)” add-in from the “Recommended Add-ins” section is selected.

Configuring the RSI interface with the Pin Multiplexing add-in

Double-clicking the system.svc opens the System Configuration utility, which also shows the configuration tabs for each add-in. Select the “Pin Multiplexing” tab to open the configuration options.

The “peripherals” list displays all the peripherals that can be configured - Enable the “RSI0 [RSI Module]” tick box to generate the required code in the Pin Multiplexing add-in that will configure the RSI.

Configuring the RAMDisk device

The RAM Disk device does not require any hardware-specific configuration. We recommend that external memory is used, as there is unlikely to be enough memory using internal memory alone. To enable external memory use the “Startup Code/LDF” add-in for CrossCore® Embedded Studio, and configure the use of external memory in the LDF section under the Startup Code/LDF tab.

µC/CLK configuration

µC/CLK is an independent clock/calendar management module, with source code for easily managing date and time in a product. µC/FS uses the date and time information from µC/CLK to update files and directories with the proper creation/modification/access time.

µC/CLK can be used with or without an RTOS. µC/CLK relies on a periodic signal to increment an internal timestamp, which can either be provided by an RTOS or an external source.

When not using an RTOS

When not using an RTOS, µC/CLK relies on an external timestamp. The default configuration provided with µC-FS File System for CrossCore Embedded Studio does not enable the external timestamp option, so µC/CLK requires changes to be able to link. To enable external timestamps, use the µC-CLK tab in the **System Configuration** utility and enable “include support for external clock/calendar”.

The file system/uC-FS/app_timing.c contains a framework with all the definitions required for the µC/FS to be used without an RTOS. This file is a template and it **must** be modified so that it produces the correct delays and signals required for µC/CLK.

When using µCOS-III

µC/CLK detects if µC/OS-III Real-Time Kernel for CrossCore Embedded Studio is being used, and includes the correct implementation of the OS-specific sources. In this case, the default configuration of µC/CLK uses the RTOS clock and timestamps are maintained using a µC/CLK-specific task.

The µC/CLK task must be configured with parameters that are appropriate for your application, specifically for task priority and stack size. These fields can be configured by clicking on the µC/FS tab in the System View, and selecting the “Application General” page. The priority of the CLK Task should be set so that the task can run frequently.

Other RTOSs

Like µC/FS, the RTOS interface provided by Micriµm has been designed so that it can be ported onto any other RTOS. However, the Analog Devices CrossCore Embedded Studio software only supports the µCOS-III and no-OS environments. The framework to use another RTOS exists, but requires external software. Please contact your third-party RTOS vendor if another RTOS is required.

Initializing µC/FS™ File System for CrossCore® Embedded Studio

To ensure timely initialization, the IDE adds required code to a global C function named adi_initComponents(). µC/FS must be initialized after all the components that it depends on, like pin muxing or the RTOS. This means that µC/FS must be initialized after the call to adi_initComponents().

When you add any µC/FS component to your project, CrossCore Embedded Studio’s IDE adds fs_app.c and fs_app.h to the project. These files contain skeleton functions which can be used to initialize µC/FS but can be modified to suit each specific application. We recommend that you initialize the file system using the App_FS_Init() interface in fs_app.c, which initializes the file system and all the required devices. It is also possible to make direct calls to FS_Init() to initialize the file system.

µC/FS requires timing functionality provided by µC/CLK. This means that µC/CLK must be initialized before initialising the file system. A call to Clk_Init() must be made after the adi_initComponents() and before the file system initialization.

Header files

In versions prior to 4.04.01 of Micriµm's µC/FS, the inclusion of fs.h was sufficient to include all the required headers for any module that uses the file system. From version 4.04.01 onwards, Micriµm recommend the inclusion of header files specific to the functionality required, so including fs.h no longer includes types and function prototypes to support all µC/FS functionality. The fs_inc.h header includes a list of the most common header files, though it may not include everything that is required.

Examples

µC/FS File System for CrossCore Embedded Studio provides two simple examples which show how to use the File System, one example shows the use of the RAMDisk device and the other the SD Card device. Each example is shipped for ADSP-BF609, and can be used in both Release and Debug configurations.

Notes

- The examples use features that can be disabled in the configuration windows. Disabling any of the used features results in expected link errors.
- Double-clicking on a example from the example browser or the system overview page opens the project in the installation folder without copying it to your workspace. If you want to modify the example in any way, it is recommended that it gets copied to your workspace. If you would like to copy the project to your workspace note that you may have to copy the sources separately. See the Known Issues section for more details.

Location

In order to locate µC/FS examples you can do the following:

- Open CrossCore Embedded Studio's Example Browser which can be found in CrossCore Embedded Studio under Help. Select in the Product section "Micriµm µC/FS v 4.04.00 [1.0.0]" for a full list of examples.
- Import projects located in your µC/FS installation folder under uC-FS/Examples.

Interaction with other CrossCore® Embedded Studio software

µC/OS-III™ Real Time Kernel for CrossCore® Embedded Studio

Although µC/FS File System for CrossCore Embedded Studio does not require an RTOS, if µC/OS-III is in the application then µC/FS uses some RTOS objects to ensure its behavior in a multi-threaded environment. In its first release µC/FS requires semaphores, and 2 task-specific registers if working directory functionality is enabled. Under certain conditions the API OSTimeDlyHMSM() can also be required.

Common Micriµm Components

There are several CrossCore Embedded Studio add-ins based on Micriµm's products which share common components. To ensure that the same version of these components is used by all the add-ins that require them, these components are installed in a common location which is distinct from the add-in install folders. These common components are

- µC/CPU which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-CPU v1.0.0. This installation includes µC/CPU version 1.29.01.
- µC/LIB which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-LIB v1.0.0. This installation includes µC/LIB version 1.36.02.
- µC/CLK which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-CLK v1.0.0. This installation includes µC/CLK version 3.09.00.

The documentation for these components can be found in CrossCore® Embedded Studio Help under µC/FS™ 1.0.0 > Components Shared by Add-ins.

Features not supported

File System Features

MSC Not Supported

The initial release of the µC/FS™ File System for CrossCore® Embedded Studio will not support the MSC Class driver.

Command Line Interface

This is currently not supported for the initial release of the µC/FS File System for CrossCore Embedded Studio.

Toolchain Options

MISRA-C Compliance Conformance Checking

The Motor Industry Software Reliability Association (MISRA) published a set of guidelines for the C programming language to promote best practice in developing safety related electronic systems in road vehicles and other embedded systems. The CrossCore® Embedded Studio compiler fully supports the MISRA-C 2004 Guidelines, and can detect violations of the rules at compile-time, link-time, and run-time.

The µC/FS File System for CrossCore Embedded Studio complies with MISRA by documenting known violations of the MISRA Compliance standard. The violations mean that µC/FS sources will not compile with MISRA conformance checking enabled, and any file that includes µC/FS headers will also not compile. The file {µC/FS File System install folder}\uC-FS\Docs\FS MISRA-C 2004 Compliance Matrix.xls contains the list of MISRA-C violations by µC/FS.