TASKING VX-toolset for ARM Cortex v5.1r1 Release Notes

Scope

These release notes cover the changes between v5.0r1 and v5.1r1 of the TASKING VX-toolset for ARM Cortex.

For previous release notes, please visit the **TASKING ARM support website**.

Contents

- New in v5.1r1
- Software Platform migration tips
- Software Platform Repositories
- Quick start
- License Information
- Fixed Issues in v5.1r1

New in v5.1r1

This section gives an overview of the most important new features and improvements in v5.1r1 (compared to v5.0r1). See the section with fixed issues for a complete list.

Native support for Mac OS X

The VX-toolset for ARM release v5.1 is available now for 64-bit Intel-based Macs with OS X (Mavericks and higher). Debugger support is available through STMicroelectronics ST-LINK/V2.

MISRA C:2012

The compiler now supports the checking of your code against a broad set of rules as defined in the MISRA C:2012 standard. You can enable and disable individual rules from the compiler's command line as well as in the Eclipse IDE. See the User Guide for the list of rules supported by the ARM C compiler.

Eclipse IDE update

This release includes an updated version 4.3.2 of the IDE Eclipse named Kepler. For features, improvements and bug fixes, please refer to the Eclipse community web-site.

Device Support

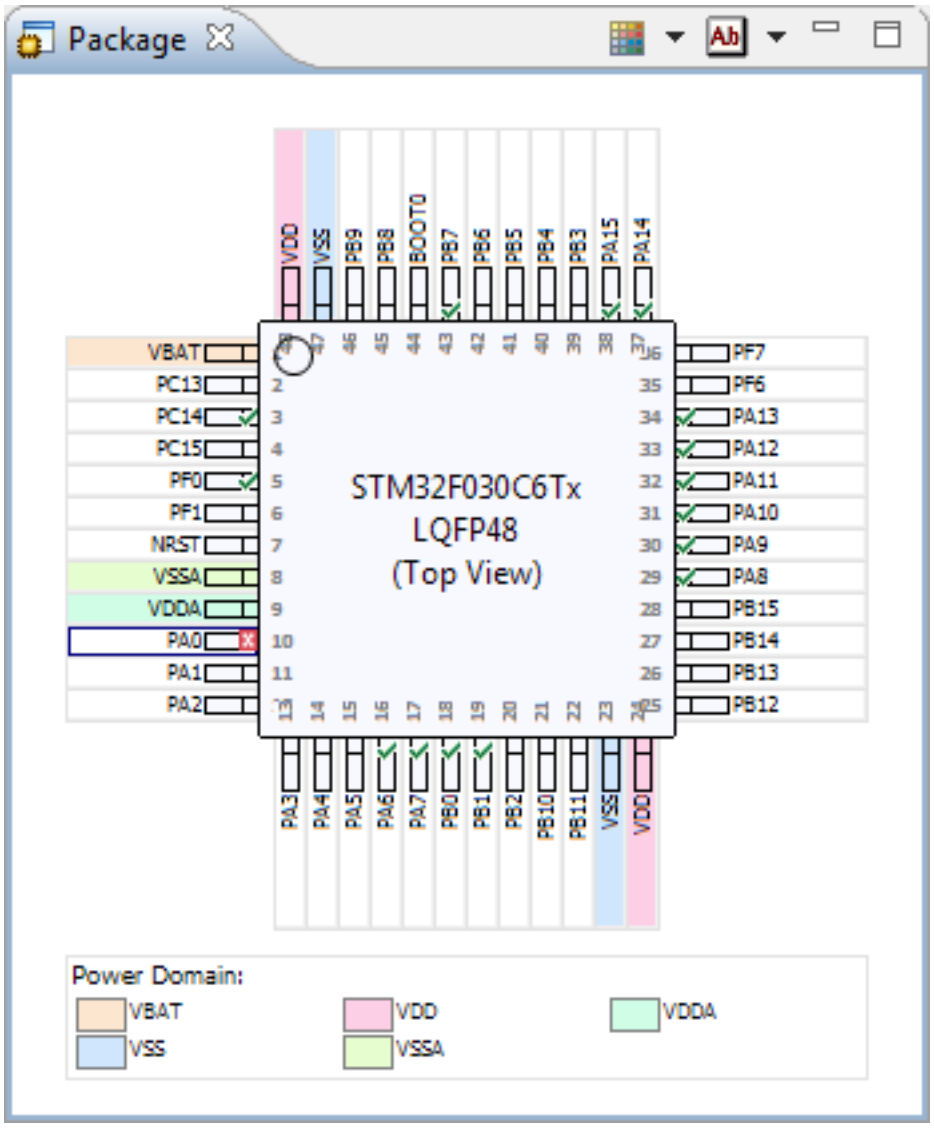
Support has been added for the following device families:

- Freescale Kinetis microcontroller
- Atmel SMART Cortex-M based microcontrollers
- Texas Instruments Tiva C family
- STMicroelectronics STM32L0 series
- Spansion FM0 and FM4
- Silicon Labs EFM32

TASKING Pin Mapper for STM32

Nowadays many microcontrollers are equipped with a large number of on-chip peripheral modules. These microcontrollers are made available in surface mount packages with various numbers of physical pins. The number of

pins usually does not allow all peripheral modules to be used simultaneously. Hardware registers on the microcontroller allow for configurable assignment of peripheral module signals to physical pins. This means that you have to select the proper device for your application and properly initialize hardware registers from software. The purpose of the TASKING Pin Mapper is to assist you in performing those tasks. The Software Platform STM32 repository is adapted to make use of the generated output of the Pin Mapper.



Pin Mapper package view

TASKING Pin Mapper for STM32 is supported in the Premium edition of the toolset. For more information about using the TASKING Pin Mapper, see the Using the TASKING Pin Mapper for ARM guide.

TASKING Software Platform

The original Software Platform repository has been split up into several parts of functionality.

Other improvements:

- Added support for the STMicroelectronics STM32F0xx family.
- Added support for all Silicon Labs EFM32 Gecko families.
- Improved support for local repositories.
- STM32 repository now uses the pin initialization functions generated by the Pin Mapper.

CMSIS-DAP

Added CMSIS-DAP Debugger support.

Shift JIS Kanji

In order to allow for Japanese character support on non-Japanese systems (like PCs), you can use the Shift JIS Kanji Code standard. This standard combines two successive ASCII characters to represent one Kanji character. A valid Kanji combination is only possible within the following ranges:

- First (high) byte is in the range 0x81-0x9f or 0xe0-0xef.
- Second (low) byte is in the range 0x40-0x7e or 0x80-0xfc

Compiler option -Ak and Assembler option --kanji enable support for Shift JIS encoded Kanji multi-byte characters in strings and (wide) character constants. Without this option, encodings with 0x5c as the second byte conflict with the use of the backslash ('\') as an escape character. Shift JIS in comments is supported regardless of this option.

Note that Shift JIS also includes Katakana and Hiragana.

Software Platform migration tips

Converting your STM32 Software Platform Document to v5.1r1

The Software Platform STM32 repository is adapted to make use of the generated output of the Pin Mapper. Projects using the STM32 peripherals should be extended with a Pin Mapper document. For more information about how to add a Pin Mapper document and to generate the peripheral initialization code, see the Using the TASKING Pin Mapper for ARM guide.

Special care is required to specify SPI NSS (chip select) pins. The definition of these pins is moved from the STM32 peripheral to the STM32 SPI Driver. To be able to select an NSS port it is required to add the specific STM32 GPIO peripheral and driver to your Software Platform document. Make sure you also specify the NSS pin as GPIO output in your Pin Mapper document. Select the STM32 SPI Driver and Specify the SPI NSS pin in the Properties view first by selecting the NSS (GPIO) port and next by entering the port pin number.

There are many STM32 Software Platform examples available showing how to address one of the STM32 peripherals. You can, for example, select the 'Samples' icon at the 'Welcome' view or from the File > Import menu, select TASKING Software Platform > Example Projects. From the Import Software Platform Example Project dialog select one of the STMicroelectronics STM32 Evaluation Board examples.

Using the STM32 repository without the Pin Mapper

To support gradual migration to v5.1r1 it is still possible to use the classic non-Pin Mapper version of the STM32 repository:

1. Open the Software Platform document.
2. Select the Software Platform Builder software service. The Software Platform Builder Properties view appears.
3. Add `platform:/plugin/com.tasking.swp.repository.stm32.classic/SoftwarePlatform/` to the value of Local Repositories. Note that this field is case sensitive. You can enter the URL directly or you can click the button to open the Local Repository Locations. Click Add to open the Add Repository dialog. Enter the URL and click OK in all dialogs.
4. From the Software Platform menu, select Reload Repository.

The added classic STM32 repository will take precedence over the default repositories. Plug-ins from the classic STM32 repository are marked as "Deprecated".

Using the STM32 Library Wrappers

Due to a naming conflict between STM32 library wrappers and other drivers, it was necessary to introduce a new naming convention. The stm32_*peripheral*_open() function will only be used for other drivers. The names of all STM32 Library wrapper initialization functions changed from `stm32_peripheral_open()` to `stm32_peripheral_init()`. This way it becomes possible to use Library Wrappers and other drivers simultaneously.

Software Platform Repositories

TASKING Software Platform comes with several repositories.

STM32 Repository

The STM32 repository contains all STM32 specific content, like peripherals, low-level drivers and library wrappers. The Software Platform STM32 repository uses the pin initialization functions generated by the Pin Mapper. Support is

included for the following device families:

- STMicroelectronics STM32F0xx
- STMicroelectronics STM32F10x
- STMicroelectronics STM32F2xx
- STMicroelectronics STM32F30x
- STMicroelectronics STM32F37x
- STMicroelectronics STM32F4xx (no FP support)
- STMicroelectronics STM32L1xx

This repository contains software/firmware from STMicroelectronics which is licensed under MCD-ST Liberty SW License Agreement V2.

EFM32 Repository

The EFM32 repository contains all Silicon Labs EFM32 specific content, like peripherals, low-level drivers and library wrappers. Support is included for the following device families:

- EFM32 Gecko
- EFM32 Giant Gecko
- EFM32 Leopard Gecko
- EFM32 Tiny Gecko
- EFM32 Wonder Gecko
- EFM32 Zero Gecko

This repository contains software/firmware from Silicon Laboratories which is licensed. See the top of each file for detailed information. Basically you are free to use the Silicon Labs code for any project using Silicon Labs devices.

TASKING POSIX Repository

TASKING POSIX implementation (PSE51) supports:

- Multi-threading
- Standard device I/O
- Standard file I/O (PSE52)
- Standard device Shared Memory routines
- File system interface for accessing sockets (PSE53)
- File system interface for accessing shared memory objects
- Basic support for message passing (PSE52)
- Basic support for signals
- FAT File System

For a detailed description of TASKING's implementation, see *TASKING POSIX Implementation*.

Some features have limited functionality for devices without a vendor specific repository.

Generic Repository

The following software is provided as Bonus Technology as described in the Altium EULA. Some third-party software is included which may require special license agreements. Check the generated code for the exact license usage conditions.

USB Host support:

- hid - Human Interrupt Devices profile to support USB mouses and keyboards
- uvc - Universal Video Class profile to support USB Webcams
- hub - Profile to support connection of up to one USB HUB
- msd - Mass Storage Device profile to support USB memory sticks
- netw - Networking profile with support for a range of USB Wi-Fi devices from Ralink/MediaTek
- hci - Host Controller Interface profile to support USB Bluetooth dongles

Other services:

- Graphics Services
- Graphics User Interface
- JPEG encoder / decoder
- HyperText Transfer Protocol Client & Service
- HTTP Secure (MatrixSSL)
- TCP/IP Internet protocol suite (lwIP)
- JSON Parser (yajl)
- Bluetooth Human Interface Device and RFCON profiles

Some features have limited functionality for devices without a vendor specific repository.

Updates

Altium provides a download site for updates to the ARM Eclipse IDE. This is primarily intended for updates to fast changing components such as the Software Platform Repository and Pin Mapper content. You can update your installation by using the Available Updates wizard.

Check for updates

1. From the Help menu, select Check for Updates. The Available Updates wizard appears.
2. Follow the steps in the wizard and click Finish.

Quick start

For a quick start, just start the *ARM Eclipse IDE* from the Start menu. This will start the Eclipse based development environment. You will be asked to select a workspace. In case you used Eclipse before it is recommended to select a new workspace. After clicking OK, you will see the 'Welcome' view at the right of editor area. On this view you will see icons that link to specific information. You can, for example, select the 'Samples' icon and import the ARM generic or ARM Software Platform project examples.

Another icon on the Welcome page, the 'First Steps' icon, links to the 'ARM Getting Started' document. This is a good starting point for exploring the capabilities of the environment and the tools.

License Information

TASKING products are protected with TASKING license management software.

License key

You need a license key when you install a TASKING product on a computer. When you order a TASKING product from Altium or one of its distributors, a license key will be sent to you by email or on paper.

See the Getting Started with the TASKING VX-toolset for ARM guide for information on obtaining a license.

Local TASKING License Server (not applicable to evaluation licenses)

If you have ordered a TASKING product with a floating license, you can have it serviced by the Remote TASKING License Server (the most convenient solution) or through a Local TASKING License Server (in case you have no external network access for example). Consult your Altium representative for assistance on deciding what the best setup would be for your situation.

If you like to setup up a local license server, we kindly refer you for more information to Support for TASKING License Management System (TLM) on our website. Here you can also download the Local TASKING License Server package.

It is advised that you install the Local TASKING License Server before you install products that require this server.

Fixed issues for v5.1r1

Improvements

- ARMVX-38792 - Unreachable license server causes slow Eclipse user interaction
- ARMVX-38878 - The LDR= pseudo-instruction should generate a T2 LDR if a high register is used.

Problems

- ARMVX-38324 - Variables located on odd addresses are handled incorrectly
- ARMVX-38551 - Erroneous do-while loop optimization when integer loop counter wraps around
- ARMVX-38662 - Slow makefile generator in Eclipse
- ARMVX-38733 - Double precision (64-bit) floating point VLDR from literal pool not supported.
- ARMVX-38780 - Allow optimization across volatile access too loose on 64-bit volatiles
- ARMVX-38790 - Many STM32F4xxx SFR's not shown in Register window of debugger
- ARMVX-38795 - Default compiler optimization should set to level 1 for the Eclipse Debug configuration
- ARMVX-38800 - STM32F0xx and STM32F030 SFR's not shown in Register window of debugger
- ARMVX-38811 - Incorrect optimization
- ARMVX-38821 - Flash error for STM32F429ZI when using ST-Link and size is greater than 1MB
- ARMVX-38823 - passing a big struct as stack parameter will cause error S917
- ARMVX-38828 - Incorrect code generation for -O0 -t0 --cpu=ARMv7R
- ARMVX-38868 - LDMIA instruction peephole optimization ignores volatile access sequence
- ARMVX-38870 - ISO C99 macros FE_TONEAREST, FE_UPWARD, etc missing
- ARMVX-38877 - Assembler allows negative constant values with MOVW/MOVT instructions.
- ARMVX-38879 - Generic instructions in an IT-block are not handled properly.
- ARMVX-38880 - Bottom of call stack shown by Debug view may be incorrect
- ARMVX-38887 - ISO C99 macros FP_NAN, FP_INFINITE, etc missing
- ARMVX-38888 - Missing ISO C99 floating point library functions
- ARMVX-38889 - ISO C99 macro INFINITY is missing
- ARMVX-38890 - ISO C99 IEEE 754 floating point functions prototype has changed
- ARMVX-38891 - Function atof does not always correctly convert constants with a "binary-exponent-part" ('p')
- ARMVX-38892 - Incorrect localtime with return from mktime
- ARMVX-38893 - Function scanf does not leave next argument untouched when it should
- ARMVX-38894 - ISO C99 macros/functions MATH_* macros and math_errhandling missing
- ARMVX-38917 - Float value comparison might fail when FPU instructions are used

You can find the list of open issues for v5.1r1 on the internet.

Software Platform improvements

- SWP-18 - Added platform url support for local repository location dialog
- SWP-19 - Cortex M0 support
- SWP-40 - Support for ILI9341 LCD controller (required to address the TFT LCD on the STM32F429 Discovery board)
- SWP-42 - Add STM324xG-EVAL WebClient example
- SWP-45 - Add EXTI example
- SWP-56 - Support for multiple (system) repositories
- SWP-58 - Add support for alternative output path
- SWP-59 - Show resource info about selected software platform items
- SWP-65 - Add STM32 L1-Discovery LCD example
- SWP-78 - Add STM32F429I-DISCO example showing how to initialize the 64 Mbits external SDRAM
- SWP-80 - Add STM32F429I-DISCO example showing how to use multi threading
- SWP-93 - The link "Generated code is out-of-date" behaves unexpectedly
- SWP-96 - Adding a Device Stack to the SWP-document out of focus

Software Platform problems

- SWP-6 - Plugin posix_threads: thread priority not restored after mutex unlock when using mutex priority ceiling
- SWP-7 - Plugin st_stm32f[24]xx_tim: compiler errors when using timer peripheral which does not use all pins
- SWP-8 - Plugin drv_stm32_i2sm: enables DMA1 instead of DMA2 for I2S3/stm32f30x
- SWP-9 - Example STM322xG_Audio_Service: Initialization fails

- SWP-12 - Non-existing Plug-in include path causes unpredictable errors.
- SWP-13 - Negative selection count in 'Add...' dialog
- SWP-16 - Plugin per_stm32f[24]xx_i2s: Incorrect default interrupt numbers for DMA_RX and DMA_TX
- SWP-20 - Plugin pal_arch_cortex_variant_generic: add support for all Cortex-M architectures
- SWP-21 - Plugin drv_stm32_usart: compiler errors when used on top of stm32f10x USART3 peripheral
- SWP-28 - Plugin drv_stm32_usart: function stm32_usart_read() should not block if there are less than nbytes immediately available for reading
- SWP-30 - Plugin drv_stm32_usart: TX/RX Blocking Timeout ignored when using multi-threading
- SWP-35 - Plugin drv_stm32_i2cm: Improve recovery after failure
- SWP-37 - No help information available for several STM32 driver plug-ins
- SWP-38 - Plugin drv_stm32_sdio: Implement 64-bit addressing to support SD cards greater than 4GB
- SWP-43 - Plugin stm32f4xx_eth_lib: Auto negotiation fails because the time waiting for link state to rise up is to short
- SWP-44 - The STM3210E-EVAL examples fail on older evaluation boards based on STM32F103ZET6 cpu
- SWP-50 - Adaptor stm32_exti_to_extint: compilation fails for STM32F0XX, STM32F10X, STM32F30X, STM32F37X and STM32F4XX devices
- SWP-60 - Naming conflict between STM32 library wrapper and generic drivers
- SWP-61 - Plugin stm32f1xx_cmsis: compiler errors for Value Line Devices
- SWP-69 - Restore defaults does not update automatic options
- SWP-70 - Plugin drv_stm32_ltdc: 4 IO-lines need a different alternate function
- SWP-71 - Example STM324x9I_MB1046_Graphics: wrong memory configuration
- SWP-79 - Adaptor graphics_to_textdisplay: char write should also clean the background
- SWP-81 - Plugin agui: compiler errors when used in multi-threading environment
- SWP-87 - Examples using clock() without POSIX requires Software Timing Service plugin
- SWP-103 - ADC examples using the STM32F10X ADC Library wrapper should be adapted for the new external trigger default
- SWP-105 - Plugin drv_stm32_usart: Calling stm32_usart_write() fails for size greater than 1
- SWP-127 - Function modem_open() does not return
- SWP-139 - Plugin timing: Use software timers enabled causes clock() problem
- SWP-153 - Fix procedure for changing FLASH waitstates
- SWP-164 - Plugin per_stm32f0xx_gpio / st_stm32f0xx_gpio: no support for GPIO-E