NXP Semiconductors
Release Notes

Document Number: S32DS2018R1
Rev. 1.0, 12/2018

S32 Design Studio 2018.R1

Contents

1. Release Description

NXP Semiconductors is pleased to announce the release of the S32 Design Studio 2018.R1 for NXP Arm® based devices and hardware accelerators. S32 Design Studio is based on the Eclipse open development platform and integrates the Eclipse IDE, GNU Compiler Collection (GCC), GNU Debugger (GDB), and other software to offer designers a straightforward development tool with no code-size limitations.

1.1. Release content

- Eclipse Neon 4.6 Framework
- GNU Bare-Metal Targeted Tools for Arm® 32-bit Embedded Processors (GCC version 6.3.1 20170509, build 1574 revision g924fb68)
- GNU Bare-Metal Targeted Tools for Arm® 64-bit Embedded Processors (GCC version 6.3.1 20170509, build 1574 revision g924fb68)
- GNU Linux Targeted Tools for Arm® 64-bit Embedded Processors (GCC version 6.3.1 20170509, build 1574 revision g924fb68)
- Libraries NewLib, NewLib Nano, EWL, and EWL Nano [1]
- Semihosting for Arm® 32-bit and 64-bit bare-metal target toolchains
- GDB 7.12.1 with Python support
- S32 Trace tool is integrated to provide software analysis features (profiling, code coverage, and other)
- S32 Debug Probe support provided with S32 Debugger and S32 Trace tool
- PEMicro® hardware debugger support provided with P&E Debugger
- Lauterbach Trace32® support
- S32 Flash Tool is delivered to support Flash/SD/MMC memory programming
- The S32DS Extensions and Updates tool for automatic lookup and on-demand installation of software packages adding support for the NXP Arm® based processor families
- S32 Configuration Tool framework with the Pin, Clock, Peripheral, DCD, IVT, and DDR Configuration tools
- The wizards for creating application projects and library projects for the supported processor families
- The wizard for creating projects from project examples[2]
- EmbSys Registers view
- SDK management
- Import of MCAL configuration to a custom SDK
- The Getting Started page

[1]The availability of libraries depends on the toolchain version. Tools for Arm® 64-bit processors support NewLib libraries only.
[2]The project examples are provided in the device specific software packages.

2. What's New

S32 Design Studio 2018.R1 is the first release of the software product.

3. System Requirements for Windows Host

3.1. Recommended configuration

- PC with 2.6 GHz Intel® Pentium® compatible processor or better
- 4 GB of RAM
- 4.5 GB of disk space (when installing all product features or all updates)
- 3 GB of temporary storage (required only during the product installation)
- USB port for communications with target hardware
- Ethernet port for communications with target hardware (optional)

3.2. Operational minimum configuration

- PC with 1.8 GHz Intel® Pentium® compatible processor
- 2 GB of RAM
- 4.5 GB of disk space (when installing full product or updates)
- 3 GB of temporary storage (required only during the product installation)
- USB port for communications with target hardware

3.3. Host operating system support

- Microsoft® Windows® 7 64-bit
- Microsoft® Windows® 8/8.1 64-bit
- Microsoft® Windows® 10 64-bit

S32 Design Studio 2018.R1 supports all editions of the operating systems listed above and is limited only by the requirements of the Java Runtime Environment.

4. System Requirements for Linux Host

4.1. Recommended configuration

- PC with 2.6 GHz Intel® Pentium® compatible processor or better
- 4 GB of RAM
- 4 GB of disk space for installation files (full product or updates)
- 2.5 GB of temporary storage (required only during the product installation)
- Java 1.8

- GCC 4.9.3
- USB port for communications with target hardware
- Ethernet port for communications with target hardware (optional)

S32 Design Studio 2018.R1 Installation Guide specifies the installation prerequisites for Linux platforms.

4.2. Operational minimum configuration

- PC with 1.8 GHz Intel® Pentium® compatible processor
- 2 GB of RAM
- 4 GB of disk space for installation files (full product or updates)
- 2.5 GB of temporary storage (required only during the product installation)
- Java 1.8
- USB port for communications with target hardware

4.3. Host operating system support

- Ubuntu LTS 16.04 64-bit
- Debian 8 64-bit
- CentOS 7 64-bit

S32 Design Studio 2018.R1 supports all editions of the operating systems listed above and is limited only by the requirements of the Java Runtime Environment.

5. Product Web Page

S32 Design Studio 2018.R1 is available for download from the product Web page at www.nxp.com.

6. Installation and Licensing

The S32 Design Studio 2018.R1 installation package contains the base tools and the installer. Support for the NXP Arm® based processor families is provided with additional software packages. The packages can be installed on the product from the S32DS Extensions and Updates tool or downloaded from the product Web page.

Note: The plug-ins to support third-party compilers or debuggers such as Lauterbach Trace32® are not included in the installation package and have to be installed from the corresponding sites or installation packages.

Run the installation package. The wizard will guide you through the installation process.

Note: Installation of S32 Design Studio 2018.R1 from the command line interface in the console or silent mode is not supported.

During installation, license activation is requested. The following activation types are supported:

- Online activation requires access to the Internet. A license activation request is sent automatically.
- Offline activation requires no Internet access. A license activation request is generated and needs to be passed to the licensing site manually. The activation response is loaded from the site to the installer manually.

New functionality can be added to S32 Design Studio 2018.R1 with additional software packages, updates and patches. Software packages add support for specific NXP Arm® based processor families. Updates and patches correct software defects and extend general functionality of the product.

New functionality can be added directly from the Internet or from a downloaded archive. If your computer is connected to the Internet, select **S32DS Extensions and Updates** on the **Help** menu to find and install all updates and packages available. If your computer does not have access to the Internet, you can download software packages, updates, and patches from the product page and install them from the archive files using the S32DS Extensions and Updates tool.

7. Starting S32 Design Studio

To start S32 Design Studio and begin to work with it:

- In Windows, double-click the product icon on the desktop or go to the Start menu and click **NXP S32 Design Studio** > **S32 Design Studio 2018.R1**. To run the product from the command line, open the terminal, navigate to the `<install_dir>\eclipse` directory, type `s32ds.bat`, and press `Enter`.
- In Linux, open the `NXP S32 Design Studio/S32 Design Studio 2018.R1` directory on the desktop and double-click the product icon. To run the product from the command line, open the terminal, navigate to the `<install_dir>/eclipse` directory, type `./s32ds.sh`, and press `Enter`.

8. Technical Support

- S32 Design Studio general issues are tracked through the S32DS Public NXP Community space:

  https://community.nxp.com/community/s32/s32ds

- For confidential cases and cases which cannot be publicly shared on NXP Community please follow the steps described here:

  https://community.nxp.com/docs/DOC-329745

- Submit a support request for P&E Microcomputer Systems:

  http://www.pemicro.com/support/index.cfm

Appendix A. Known issues and Workarounds

- **Conditional watchpoints and breakpoints**: Conditional breakpoints and watchpoints, including those using ignore counts, may not work in some cases.

Workaround: Avoid using conditions for breakpoints and watchpoints, instead check for condition in the code and set a normal breakpoint.

- **Disassembly view** content might be destroyed occasionally.

  Workaround: Close the Disassembly view and open it again by selecting **Window** > **Show View** > **Disassembly** on the menu.
- **Hyperlinks** may not work correctly on Windows 10 when Microsoft Edge is configured as the default browser.

  Workaround: Set a different browser as the default one.
- **Views** are not updated instantly after executing a command in the GDB console.

  Workaround: Use the `step` command to see refreshed views.
- The **Getting Started** page may be displayed blank in Linux.

  Workaround: Install Webkit1 for GTK2 using the following command: `sudo apt-get install libwebkitgtk-1.0-0`
- The **Getting Started** page may fail to load content at startup of S32 Design Studio 2018.R1. “ The page can't be displayed ” error message may be shown instead.

  Workaround: Refresh the **Getting Started** page by pressing the F5 key.
- **Duplicated error dialog**: When adding more watchpoints than supported by the device, the popup box with the “ Not enough hardware resources for processing ” error message is displayed twice.

  Workaround: Close the popup box, then close it again.
- **Peripheral register indexes are displayed incorrectly**: When several instances of a peripheral are available, the **EmbSys Registers** view fails to display similar registers of different instances indexed properly (like “ reg_A [0] ” , “ reg_A [1] ” , and so on). Instead, the view displays these registers with the “ [err] ” index value. For instance, all SLBR registers in the REG_PROT group are displayed as “ SLBR [[err]] ” .

  Workaround: No workaround available.
- **Stepping over the try-catch block fails on a VDK**: When debugging a project on a VDK, an attempt to step over the try-catch block fails.

  Workaround: The issue is specific for projects compiled with NewLib. Recreate a project and select NewLib Nano as the library.
- **A wrong message when terminating a debug session with S32 Debugger**: When debugging with S32 Debugger and terminating the debug session, the message about launching the debug configuration (like “ Launching my_project_S32Debug ”) is displayed at the bottom of the perspective.

  Workaround: The message does not indicate a real problem and can be ignored.
- **Watchpoints set on complex data types are not hit**: When debugging on a target connected with S32 Debug Probe, a debug session ignores a watchpoint set on a variable of a complex data type (such as a structure or other). A watchpoint set on an item of a basic data type inside a complex variable works correctly.

  Workaround: Avoid setting watchpoints at complex data types.

- **A target application fails when being loaded with the Flash tool via a COM port**: When loading an initializing application to the target with the Flash tool via a COM port, a failure may be reported with the “ Response error ” message.

  Workaround:

  1. Hold the reset button on the board.
  2. Run the command to start the Flash tool and to load the initializing application, for instance:

     ```
     S32FlashTool.exe -t targets\S32V234.bin -a flash
     \Cypress_S26KL512S2_algo.bin -i uart -p COM37
     ```

  3. Wait for a second, then release the reset button.
- **USB connection failure when debugging with S32 Debug Probe on a Linux VM**: A debug session fails on a Linux virtual machine with S32 Debug Probe connected through USB. Debugging with S32 Debug Probe connected through Ethernet can be done successfully.

  Workaround: Connect the probe to USB. Set up the debug configuration to use the Ethernet connection rather than USB. Specify the virtual IP address of the probe in the connection settings of the debug configuration. To obtain the required value, run the following command:

  ```
  sudo ifconfig -a
  ```

  The displayed output includes a section for the virtual Ethernet interface created on Linux for communication with the probe over the physical USB link. The *inet addr* value in this section is a local IP address starting with “ 169.254 ” . Take the *HWAddr* value in the section and convert the last two bytes into decimal notation. Append the resulting numbers to “ 169.254 ” to obtain the required virtual IP address.

Note: There are some issues which are introduced by Eclipse CDT and are therefore reproduced in the S32 Design Studio. These issues might be fixed when the fix is available in the future version of Eclipse CDT and the S32 Design Studio migrates to the updated CDT.

Appendix B. Performance Considerations

The following suggestions will help you keep the S32 Design Studio tools running at a respectable performance level.

1. To maximize the performance, the S32 Design Studio tools should be installed on a computer with the recommended configuration. While the tools will operate on a computer with the operational minimum configuration, limited hardware will restrict its ability to function at desired performance levels.
2. Close unused projects. Eclipse caches files for all open projects in the workspace. If you need multiple projects open, try to limit the number of projects to no more than 10.
3. The Eclipse IDE provides several options that provide user assistance tools. These options, however, use memory and CPU resources. If the performance is slow, and you do not need these options, turn them off.

4. Scalability mode options configure how Eclipse deals with large source files and may affect the overall performance of the IDE. Turn off these options to maximize the performance. Following scalability mode options are available:
   - Editor live parsing: impacts parsing while typing, Outline View, semantic highlighting, folding, etc.
   - Semantic highlighting: C/C++ identifiers are colored
   - Syntax coloring: coloring of keywords, comments and literals
   - Parsing–based content assist proposals: content assist proposals which require parsing the file
   - Content assist auto activation: content assist activated automatically on trigger sequences, like '.', '::' or '–>'.

   To disable:

   a. Choose **Window** > **Preferences**
   b. Expand **C/C++** > **Editor** > **Scalability**
   c. Uncheck **Enable all scalability mode options**
5. Content Assist Auto Activation can reduce the number of keystrokes one must type to create the code. The Content Assist plug-in consists of components that predict what keystroke you may type next, based on the current context, scope and prefix. Turn off this predictor to maximize the performance.

   To disable:

   a. Choose **Window** > **Preferences**
   b. Expand **C/C++** > **Editor** > **Content Assist**
   c. Uncheck all the options for **Auto-Activation**

How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

Document Number: S32DS2018R1

Rev. 1.0

12/2018