NXP Semiconductors
Release Notes

Document Number: S32DS34
Rev. 1.0, 12/2020

S32 Design Studio for S32 Platform 3.4

Contents

1. Release Description

NXP Semiconductors is pleased to announce the release of the S32 Design Studio for S32 Platform 3.4 for NXP Arm® based devices and hardware accelerators. S32 Design Studio for S32 Platform 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 2019-12 framework
- GNU Bare-Metal Targeted Tools for Arm® 32-bit Embedded Processors
  - # GCC version 6.3.1 20170509, build 1620 revision g01b30c3
  - # GCC version 9.2.0 20190812, build 1649 revision gaf57174
- GNU Bare-Metal Targeted Tools for Arm® 64-bit Embedded Processors
  - # GCC version 6.3.1 20170509, build 1620 revision g01b30c3
  - # GCC version 9.2.0 20190812, build 1649 revision gaf57174
- GNU Linux Targeted Tools for Arm® 64-bit Embedded Processors
  - # GCC version 6.3.1 20170509, build 1620 revision g01b30c3
  - # GCC version 9.2.0 20190812, build 1649 revision gaf57174
- Libraries: NewLib, NewLib Nano, EWL, and EWL Nano [1]
- Semihosting for Arm® 32-bit and 64-bit bare-metal target toolchains
- MSYS2 32bit version 1.0.0
- GDB 9.2.0 with Python support
- S32 Debugger (with S32 Debug Probe) support for ARM cores
- S32 Debugger support for the CMSIS-DAP on the GreenBox II board (is supported by particular devices)
- S32 Debugger provides OS Awareness support for FreeRTOS and OSEK
- S32 Trace for A53 cores with S32 Debugger
- S32 Flash Tool
- PEmicro® debugger support
- Lauterbach Trace32® support
- IAR debugger support
- iSystem debugger support
- Segger debugger support
- Green Hills compiler support
- IAR compiler support
- Diab compiler support[2]

[1] The availability of libraries depends on the toolchain version and the core type. Tools for Arm® 64-bit processors support NewLib libraries only. Tools for Arm® 32-bit processors do not support EWL libraries for the Cortex-R52 and Cortex-M33 cores.

[2] Wind River Diab Compiler does not support Debian OS.

- The S32DS Extensions and Updates tool
- The SDK integration is provided with additional software packages. The SDK packages can be installed and updated with the S32DS Extensions and Updates tool.
- SDK management
- Support for importing MCAL configuration to a custom SDK
- The migration support is provided to upgrade:
    # SDK version attached to the project,
    # GCC based toolchain for project.
- S32 Configuration Tool framework with the Pin, Clock, Peripheral, DCD, IVT, DDR and QuadSPI Configuration tools
- The wizards for creating application, library projects and projects from project examples for the supported processor families[3]
- The Getting Started page
- EmbSys Registers view
- Peripheral Registers view
- ARM System Registers view
- Watch registers view
- Memory Spaces view

2. What's New

This release comes with several new features, improvements and bug fixes.

2.1. New features

- New Eclipse version.
- New S32 Configuration Tool version.
- The Diab toolchain support is provided by the project wizard (available for particular devices).
- The latest versions of PEmicro debugger plug-in and drivers are provided.
- S32 Debugger provides preliminary support for secure debugging with the Password and Challenge/Response authentication methods.
- Updated GDB client.
- Eclipse TM Terminal is provided.[4]
- Peripheral Registers view - search option updated to search by address.
- ARM System Registers view.
- Watch registers view.
- Memory Spaces view.

[3] Support for wizards and the project examples are provided in the device specific software packages.

[4] For more information on standard Eclipse plugins refer to Eclipse documentation.

2.2 Bug fixes

- For multicore project after creating the boot core will be set and focused automatically in the project explorer.
- Migration for Library and I/O support options is supported when migrate from gcc 6.3 to gcc 9.2.
- Fully qualified domain name supported for S32 Debugger's debug probe hostname.
- S32 Configuration Tool Peripherals context menu fixed (non relevant items removed).
- Registers with side effects are switched to "Read on demand" mode by default and cannot be switched to "Read always" (readAction attribute is read from .svd file).
- "Test connection" button fixed specially for S32S device.
- Hyperlinks now work correctly on Windows 10 when Microsoft Edge is configured as the default browser.

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
- 30 GB of disk space (when installing all product features or all updates)
- 24 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
- 25 GB of disk space
- 20 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 for S32 Platform 3.4 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
- 25 GB of disk space for installation files (full product or updates)
- 20 GB of temporary storage (required only during the product installation)
- Java Runtime Environment 1.8 64-bit
- GCC 4.9.2 or newer
- USB port for communications with target hardware
- Ethernet port for communications with target hardware (optional)

S32 Design Studio for S32 Platform 3.4 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
- 20 GB of disk space
- 16 GB of temporary storage (required only during the product installation)
- Java Runtime Environment 1.8 64-bit
- GCC 4.9.2 or newer
- USB port for communications with target hardware

4.3. Host operating system support

- Ubuntu LTS 16.04 64-bit
- Ubuntu LTS 18.04 64-bit
- Debian 8 64-bit
- CentOS 7 64-bit

S32 Design Studio for S32 Platform 3.4 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 for S32 Platform 3.4 is available for download from the product Web page at www.nxp.com.

6. Installation and Licensing

The S32 Design Studio for S32 Platform 3.4 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 for S32 Platform 3.4 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 for S32 Platform 3.4 with additional software packages, updates and patches. Software packages add support for specific NXP Arm® based processor families. Updates and patches extend general functionality of the product and correct software defects.

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 S32DS

To start S32 Design Studio for S32 Platform 3.4 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 for S32 Platform 3.4**. To run the product from the command line, open the terminal, navigate to the directory with the installed product, type `s32ds.bat`, and press `Enter`.
- In Linux, to run the product from the command line, open the terminal, navigate to the directory with the installed product, type `./s32ds.sh`, and press `Enter`.

8. Technical Support

- S32 Design Studio for S32 Platform 3.4 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 PEmicro:

  https://www.pemicro.com/support/

9. Related Documentation

To learn more about the included tools, refer to the following Release Notes:

- S32 Debugger Release Notes located in `/S32DS/tools/S32Debugger/Debugger/`
- S32 Configuration Tool Release Notes located in `/Release_Notes/`
- S32 Flash Tool Release Notes located in `/S32DS/tools/S32FlashTool/doc/`
- GNU Bare-Metal Targeted Tools for Arm 32-bit Embedded Processors Release Notes located in:

  ```
  # /S32DS/build_tools/gcc_b1620/gcc-6.3-arm32-eabi/
  # /S32DS/build_tools/gcc_v9.2/gcc-9.2-arm32-eabi/
  # /S32DS/tools/gdb_arm/arm32-eabi/
  ```

- GNU Bare-Metal Targeted Tools for Arm 64-bit Embedded Processors Release Notes located in:

  ```
  # /S32DS/build_tools/gcc_b1620/gcc-6.3-arm64-eabi/
  # /S32DS/build_tools/gcc_v9.2/gcc-9.2-arm64-eabi/
  # /S32DS/tools/gdb_arm/arm64-eabi/
  ```

- GNU Linux Targeted Tools for Arm 64-bit Embedded Processors Release Notes located in:

  ```
  # /S32DS/build_tools/gcc_b1620/gcc-6.3-arm64-linux/
  # /S32DS/build_tools/gcc_v9.2/gcc-9.2-arm64-linux/
  # /S32DS/tools/gdb_arm/arm64-linux/
  ```

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 when Microsoft Edge (early versions) 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 for S32 Platform 3.4. “ 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.

- **Uninstallation of P&E drivers**: The P&E Device Drivers are not uninstalled with uninstallation of S32 Design Studio. After that, an attempt to uninstall them manually can cause errors.

  Workaround: If you do not need the P&E drivers for other products and devices, uninstall them before the product uninstallation.

- **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.

- **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.

- **USB connection failure when debugging with S32 Debug Probe on a Linux VM and Debian**: A debug session fails on a Linux virtual machine and Debian 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.

- **S32 Debug Probe USB connection is not recognized in S32DS 3.2 after S32DS 3.3 is installed**: S32 Debug Probe driver upgraded to RNDIS 6.0 in S32DS 3.3 instead of RNDIS 5.0 version used in S32DS 3.2.

  Workaround: Install new RNDIS driver to S32DS 3.2 manually as described in readme.txt in `<S32DS3.2_install_dir>/S32DS/tools/S32Debugger/Debugger/drivers/usb`.

- **Unable to install the VP Explorer plug-in**: When launched, the S32 Design Studio for S32 Platform installer looks for the existing Virtualizer Runtime installation to install the VP Explorer plug-in. The latest version of Virtualizer Runtime is not detected.

  Workaround: Install the VP Explorer plug-in manually. On the main menu, choose **Help** > **Install New Software** from the menu bar, click **Add**, then click **Archive**. Navigate to the `<synopsys_version>/VirtualizerRuntime/SLS/windows/vpexplorer_feature` directory and choose the `vpexplorer-feature.zip` file. The required plug-in will appear in the list, select it and click **Next** to proceed the installation.

- **Debugging on a VDK may fail**: Some errors may occur due to unexpected termination of a debug session or unsupported registers and watchpoints, etc.

  Workaround: Run a debug session in Virtualizer Studio.

- **Creating new project in the root of drive C:\ fails**: When creating a new project in the root directory, the project wizard shows the "Can't create directory: C:\project_name" error message. Windows does not allow user to create files in this location.

  Workaround: Create a subdirectory in the root directory (for example, C:\projects) and specify this location in the project creation wizard.

- **Register variable is not shown in memory view**: If the compiler locates the variable in register the debugger cannot show it in the memory view. "View memory" selection for this variable generates the error message: "Can't view memory on &(counter)".

  Workaround: Information which register is assigned for the variable can be found in:

  # the **Disassembly** view,
  # the error message in the **Expression** view. If "**&counter**" was added in the **Expression** view then the error message contains the information like this:

```
Failed to execute MI command:
-data-evaluate-expression *(&counter)
Error message from debugger back end:
Address requested for identifier "counter" which is in
 register $r<reg_num>
```

- **GHS project with SDK build failed after renaming of the project**: Header files are copied to the project, and rename process affects the paths. GHS build system believes that the files are not changed and does not update them during building - this is the GHS build system specific.

  Workaround: For GHS projects after renaming of the project activate clean and refresh procedures before building.

- **S32 Debugger Program counter does not support HEX-value setting**.

  Workaround: When creating a **Debug Configuration**, at **Startup** tab use one of the following options:

  # use label or function name instead for the **Set program counter at** field in **Runtime Options**,
  # or uncheck the **Set program counter at** and add the following commands to **Run commands** field:

  ```
  set $pc=<HEX-value>
  -exec-until *<HEX-value>
  ```

- **Debugging on S32 Debugger may fail**: **Variables** view may pose a problem when entering some function with uninitialized pointers as it generates random access which may corrupt the debugger. In some cases the invalid access may be generated when debugging optimized code (GHS toolchain) where pointer location shared with some variable.

  Workaround: Close the **Variables** view or make it inactive for critical part of code (until pointers are not initialized in code).

- **Debugging on S32 Debugger**: **Variables** view and **Expressions** view may not display the value of some variables correctly if those variables have been used in macros.

  Workaround: Use the **Details** number format in these views as it is the only format that will display the correct value in this context.

- **Step out work incorrect with dbg_derived_types source**: The GHS toolchain does not generate DWARF Call Frame information that is suitable for unwinding the call stack. For third-party debuggers incapable of unwinding the call stack through an alternate means (disassembling code), this may nullify the ability to walk the call stack or view variables saved in previous frames. As a workaround S32DS removes the debug frames info to force GDB client to recreate the frames. In some cases this method may not work and lead to incorrect behaviour for **Step out** command.

  Workaround: Set a breakpoint at function end and run to breakpoint.

- **Project with huge amount of macros freezes S32DS IDE**: The huge count of macros is critical for indexing sources. Editing file with many macros causes 100% CPU usage and freezes S32DS IDE.

  Workaround: Manually tune `s32d.ini` file - remove the following lines:

  ```
  -Xms256m
  -Xmx2048m
  ```

- **Projects renaming out of the workspace is not supported**: S32DS can't perform its custom rename procedure for a project out of the workspace used in the session. The eclipse default renaming will get the wrong result (problems with launch configurations and other related files).

  Workaround: Use one of the following options:

  # Copy the project to wokspace (use file system options).
  # Import the prioject with copying files to workspace.
  # Change to another workspace with the project to rename (when using several workspaces).

- **GDB 9.2**: "Couldn't determine a path for the index cache directory" warning appears in **Debugger Console** view. Directory variable is not set for GDB 9.2.

  Workaround: Set environment variable before GDB launch:

  ```
  set HOME=%LocalAppData%
  ```

  In the S32DS IDE to set this environment variable go to **Window** > **Preferences** > **C/C++** > **Build** > **Environment**, click **Add**. In the **New variable** window set `HOME` for the **Name** field and `${LOCALAPPDATA}` for the **Value**.

- S32DS has multiuser install issue on Linux.

  Workaround: To install the S32 Design Studio for S32 Platform for multiple users perform the following steps:

  # Under the user with sudo rights:
    # Install the S32 Design Studio for S32 Platform to the `/opt/NXP/S32DS.3.4` folder,
    # Change permission:

      ```
      sudo chmod -R 770 /opt/NXP/S32DS.3.4
      ```

    # Run the S32 Design Studio for S32 Platform from the `/opt/NXP/S32DS.3.4` folder and close,
    # Change permission:

      ```
      sudo chmod 770 /dev/shm/cll_mutex
      sudo chmod -R 770 /opt/NXP/S32DS.3.4/eclipse/
      configuration
      ```

    # Add the second S32DS user without sudo rights to the first user's group:

      ```
      sudo usermod -aG <sudo_user_name> <S32DS_user_name>
      ```

  # Switch user to the <S32DS_user_name>,
  # Run the S32 Design Studio for S32 Platform from the `/opt/NXP/S32DS.3.4` folder.

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 for S32 Platform tools running at a respectable performance level.

1. To maximize the performance, the S32 Design Studio for S32 Platform 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. The following scalability mode options are available: editor live parsing, semantic highlighting, syntax coloring, parsing–based content assist proposals, and content assist auto activation.

   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**
6. For faster serial communication response (for example, UART connection), you can change the OS latency timer settings. The latency timer is a form of time-out mechanism for the read buffer of FTDI devices. The default value for the latency timer is 16 ms. You can decrease the latency value to maximize the performance:

   - On Windows, go to the port properties, open advanced port settings and select the required latency value from the drop-down menu.

- On Linux, open the terminal and set the required value using the `echo` or `setserial` commands. For example:

```
echo 2 > /sys/bus/usb-serial/devices/ttyUSB0/latency_timer
```

How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Revision: 1.0, 12/2020

Document number: S32DS34