onsemi IDE Getting Started Guide

Table of Contents

Chapter 1

Getting Started with the Eclipse-Based onsemi IDE

1.1 SOFTWARE TO DOWNLOAD

IMPORTANT: onsemi acknowledges that this document might contain the inappropriate terms “white list", "master" and "slave”. We have a plan to work with other companies to identify an industry wide solution that can eradicate non-inclusive terminology but maintains the technical relationship of the original wording. Once new terminologies are agreed upon, future products will contain new terminology.

Download the onsemi IDE Installer from www.onsemi.com/RSL10.

1.2 onsemi IDE INSTALLATION PROCEDURES

Install the onsemi IDE by running *onsemi_IDE.msi*. The onsemi IDE is installed in this location by default: *C:\Program Files (x86)\onsemi\IDE_V*<**version**>.

- If you select the J-Link checkbox during the installation process, you will be prompted to install SEGGER J-Link. You need the J-Link software to download and debug applications on the Evaluation and Development Board (EVB).

- The **J-Link Installation Check** screen will guide you through the process of installing J-Link. For the onsemi IDE to detect J-Link automatically, install it in J-Link default location: *C:\Program Files (x86)\SEGGER.*

- After installing J-Link, replace the *C:\Program Files (x86)\SEGGER\JLink**JLinkDevices.xml*** file with the one provided by onsemi. Also, add the **Montana folder** provided by onsemi containing the Code, Data and NVR flash *elf* files to the following location: *C:\Program Files (x86)\SEGGER\JLink\Devices\ONSemiconductor*.

Chapter 2

Connecting to the Hardware

There are multiple ways of power and connecting to the EVB. The board can be powered from a J-Link or FTDI connection. It can also be powered by an external power supply.

The board can be connected to a PC for communicating and debugging via the J-Link and FTDI connections.

2.1 CONNECT VIA J-LINK

The EVB can be connected to a PC for debugging and communication via a J-Link connection. This will require a J-Link debug probe, the appropriate USB cable for the J-Link debug probe, and a 20-pin ribbon cable.

Before connecting the debug probe to the EVB, the jumpers on the board must be configured. See Figure 1 for the appropriate jumper configuration.

Once the J-Link debug probe is connected to the PC, connect the debug probe to the EVB via the 20-pin ribbon cable, taking note of the pin 1 indicator. The header to be connected to is J62. The end of J62 labelled VDDO is the side of the connector associated with pin 1. Also, see Figure 1 for a visual representation of the appropriate location. If the device is being powered by the J-Link debug probe, LED D1 is expected to turn on. If it does not, try the `power on perm` command via J-Link to turn on the debug probe's 5V rail. You can tell if the device is powered correctly via the J-Link command line as well. When initiating a connection, VtRef is expected to be equal to VBAT/VDDO.

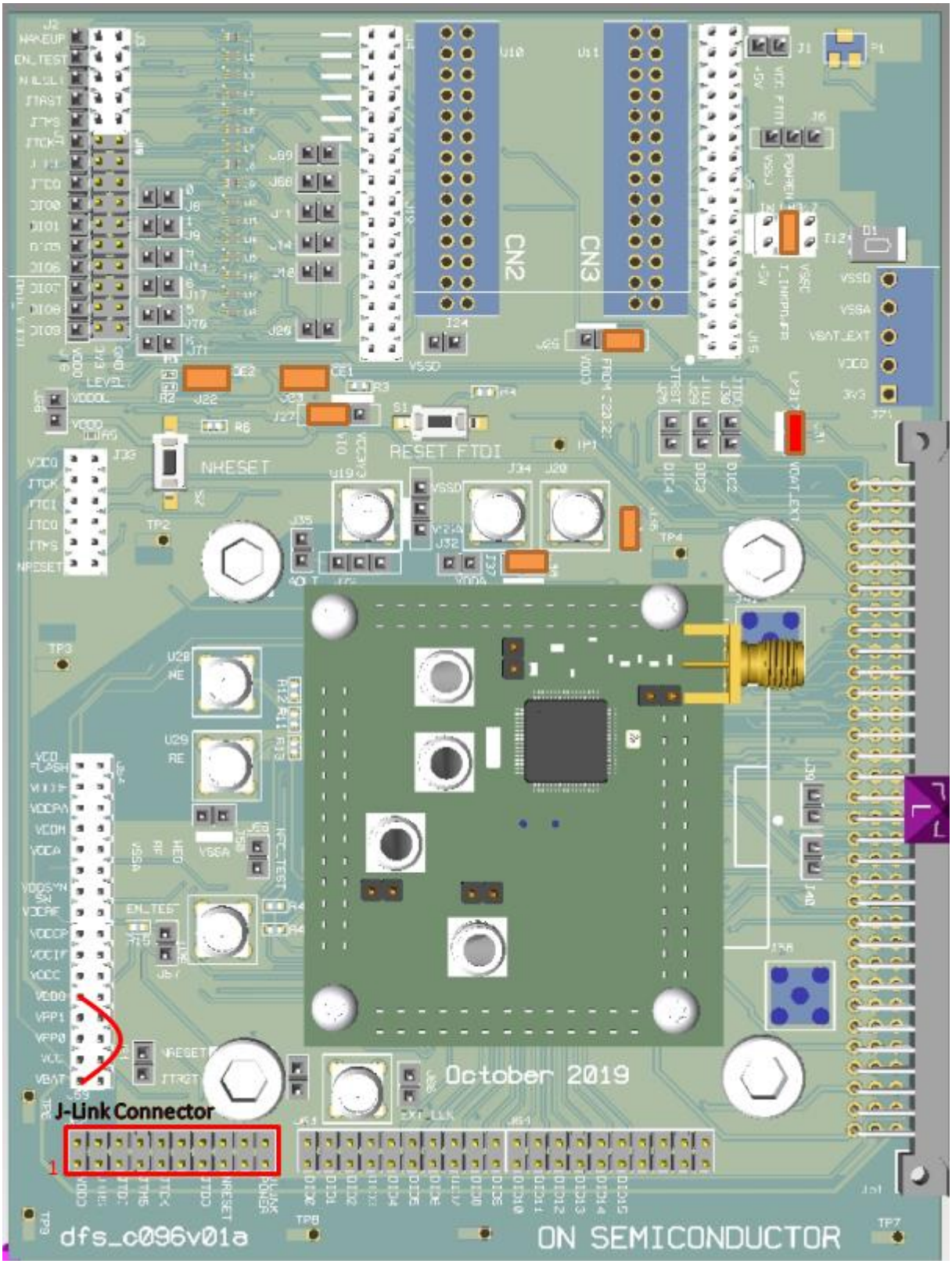

Figure 1: Evaluation and Development Board Configured for J-Link Self-Powered Mode and J-Link Communication

2.2 CONNECT VIA FTDI

The EVB can be connected to a PC for debugging and communication via a JTAG connection. This requires an FT4232H Mini Module to be installed on the EVB in the socket formed by U10 and U11. A Mini USB cable is then connected to the Mini Module. This allows a JTAG connection to the device.

In order to power the device from the USB connection, the appropriate jumpers must be configured on the EVB. See Figure 2 for the appropriate jumper configuration.

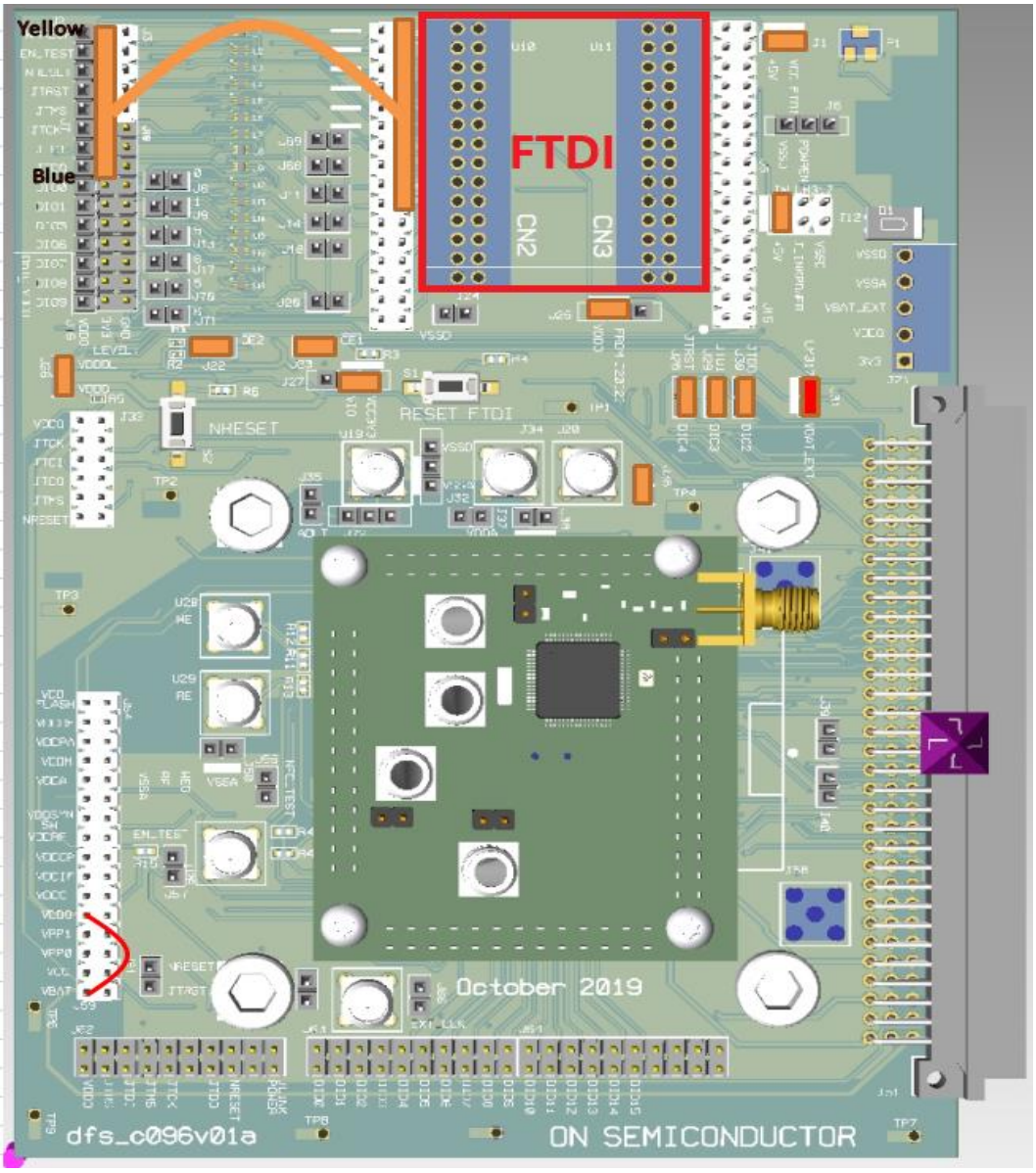

Figure 2: Evaluation Board Configured for FTDI Self-Powered Mode and FTDI Communication

2.3 POWER EXTERNALLY

The EVB can also be powered externally when connecting via FTDI or J-Link. The 5V rails from these connections are no longer needed whenthe board is powered externally. The two primary input voltage rails, VDDO and VBAT, can be connected in two locations. The primary location is the J21 connector. As a secondary connector, J59 can be used. Both locations are labelled in silkscreen, as shown in Figure 3.

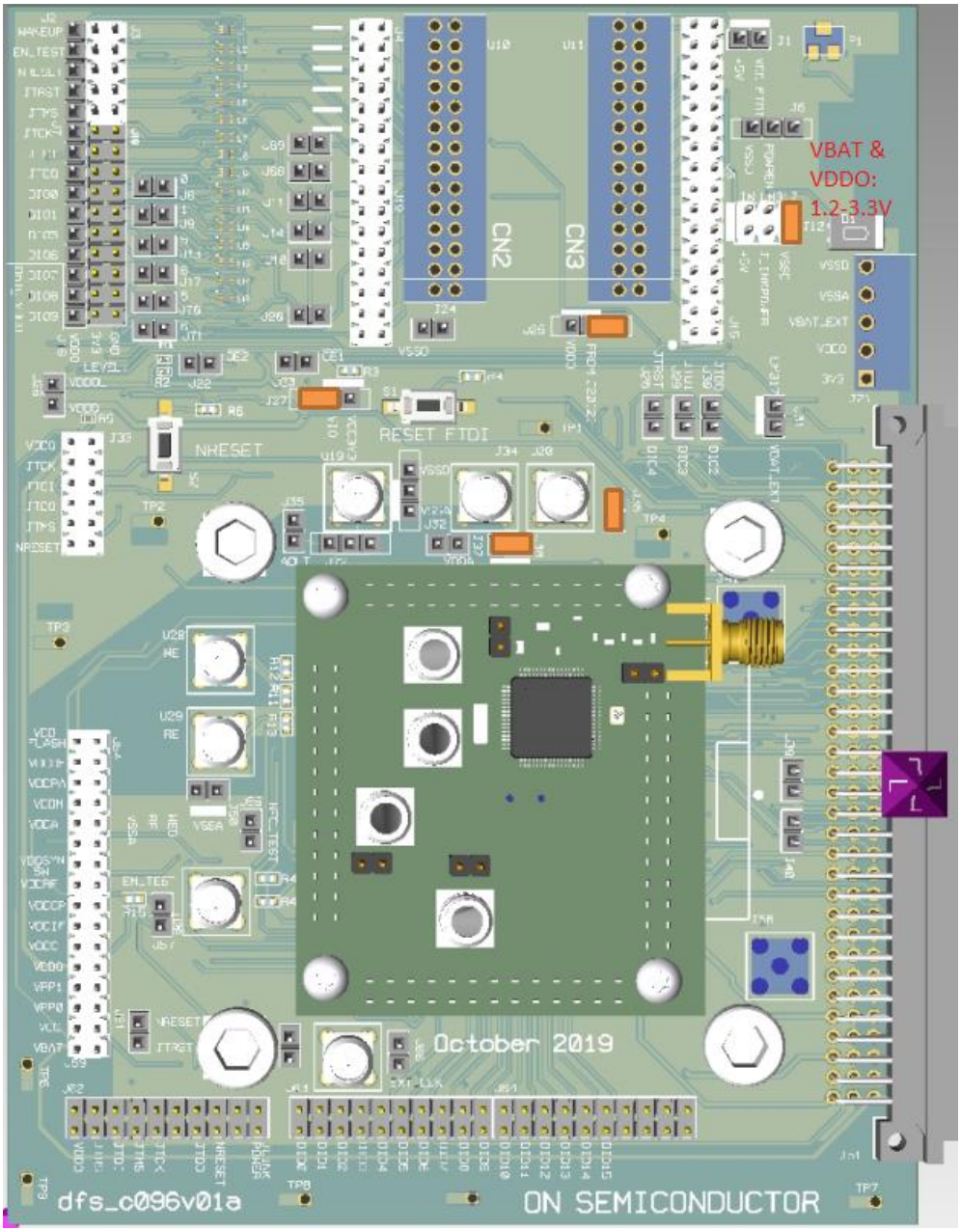

Figure 3: Evaluation Board Configured for External Power Supply

Chapter 3

Building Your First Application with the onsemi IDE

3.1 SOFTWARE TO DOWNLOAD

Download the Montana Software Package and extract the package to any temporary folder. (The temporary folder can be on any drive on your computer)

3.2 MONTANA CMSIS-PACK INSTALLATION PROCEDURES

1. It is important to create a new workspace for each new version of the IDE to ensure compatibility. Create a new workspace — at, for example, *c:\workspace* — using either Windows® Explorer or the onsemi Launcher in step 2.
2. Open the onsemi IDE by going to the Windows Start menu and selecting **onsemi** > **onsemi IDE**. From the onsemi IDE Launcher screen, browse to your new workspace, select it, and click **Launch**.
3. On the top right corner of the Workbench perspective, click on the Open Perspective icon, select **CMSIS Pack Manager**, and click **Open**, as shown in Figure 4.

NOTE: If you cannot see the **CMSIS-Pack Manager** item, re-install the IDE in your user folder (i.e., *C:\Users\<**user_name**>*).

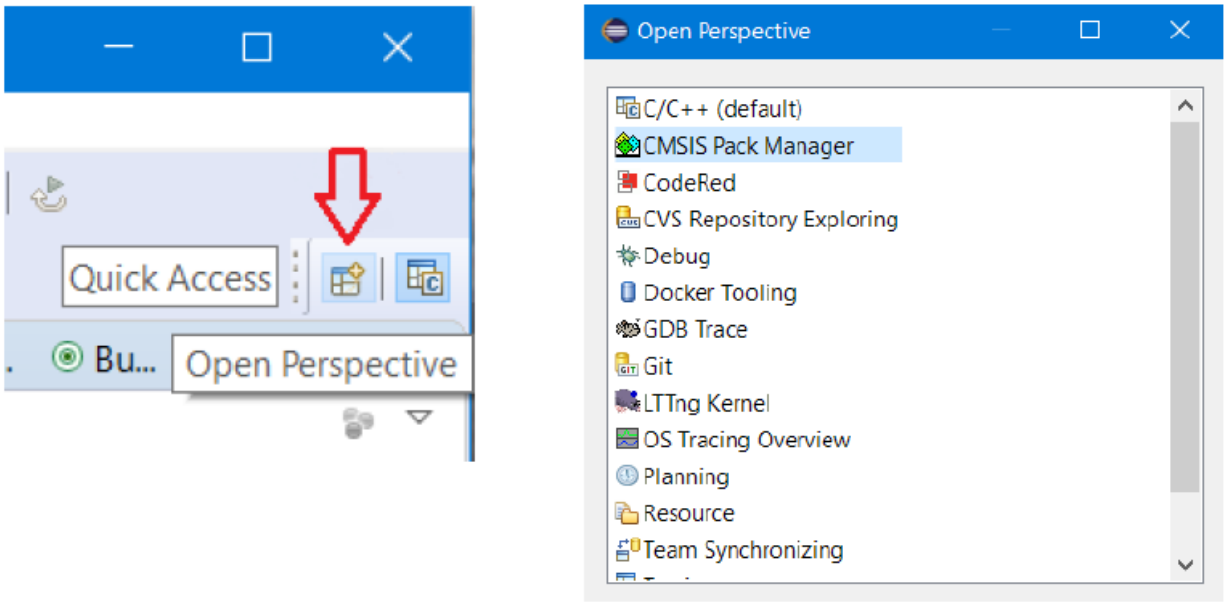

Figure 4: Opening the CMSIS Pack Manager Perspective

4. Click on the Import Existing Packs icon, select your pack file *ONSemiconductor.Montana.<**version**>.pack*, where **<*version*>** is a number such as A.2.0-35, and click **Open**, as shown in Figure 5.

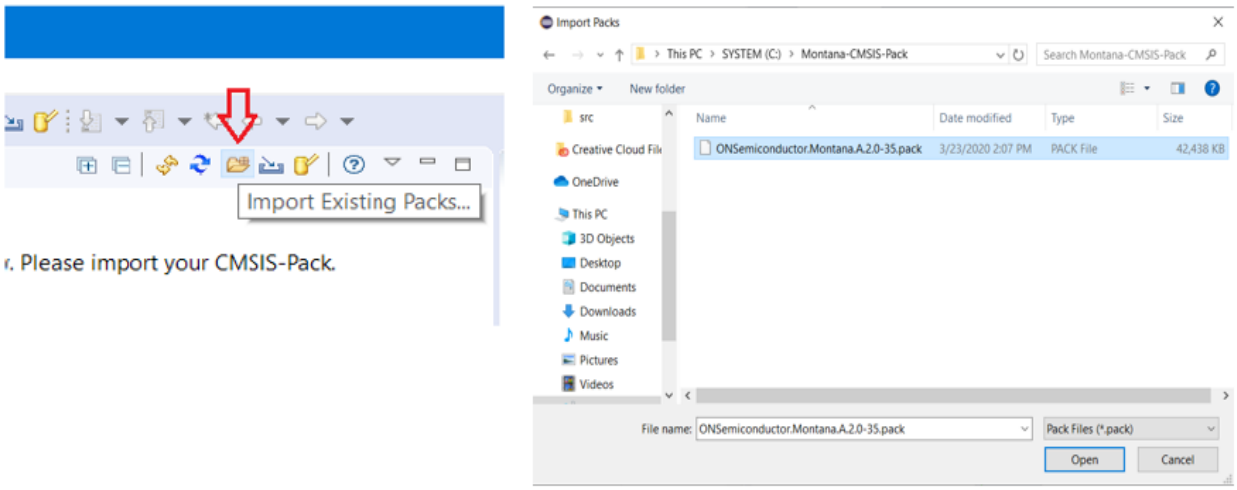

Figure 5: Installing the Montana CMSIS-Pack

NOTE: Default CMSIS pack installation location:

C:\Users\<user_id>\ON_Semiconductor\PACK\ONSemiconductor\Montana\<version>

5. The Montana CMSIS-Pack now appears in the list of installed packs. In the **Devices** tab, if you expand **All Devices** > **ONSemiconductor** > **Montana Series** you can see Montana listed there. You can manage your installed packs in the **Packs** tab. Expanding **ONSemiconductor** > **Montana** makes the **Pack Properties** tab display the details of the Montana CMSIS-Pack, as seen in Figure 6.

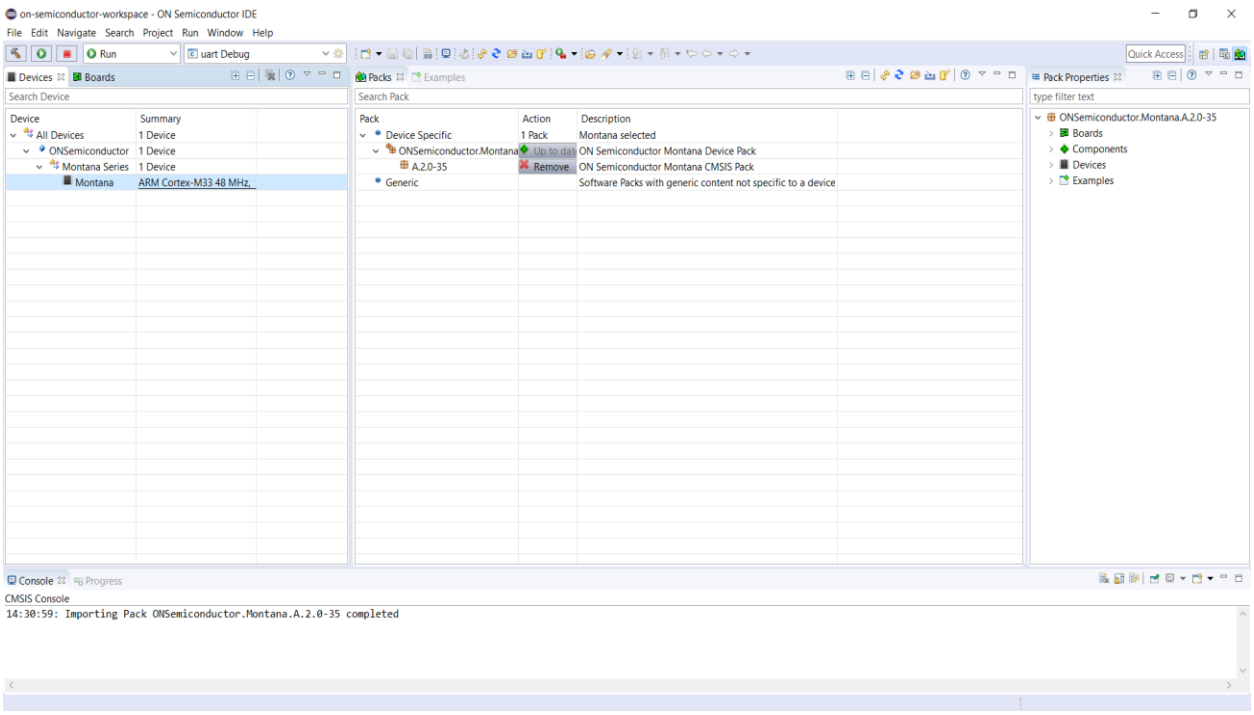

Figure 6: Pack Manager Perspective after Montana CMSIS-Pack is Installed

3.3 IMPORTING THE SAMPLE CODE

Import the sample as follows:

1. In the Pack Manager perspective, click on the **Examples** tab to list all the example projects included in the Montana CMSIS-Pack.
2. Choose the example project called *flash*, and click the **Copy** button to import it into your workspace, as seen in Figure 7.

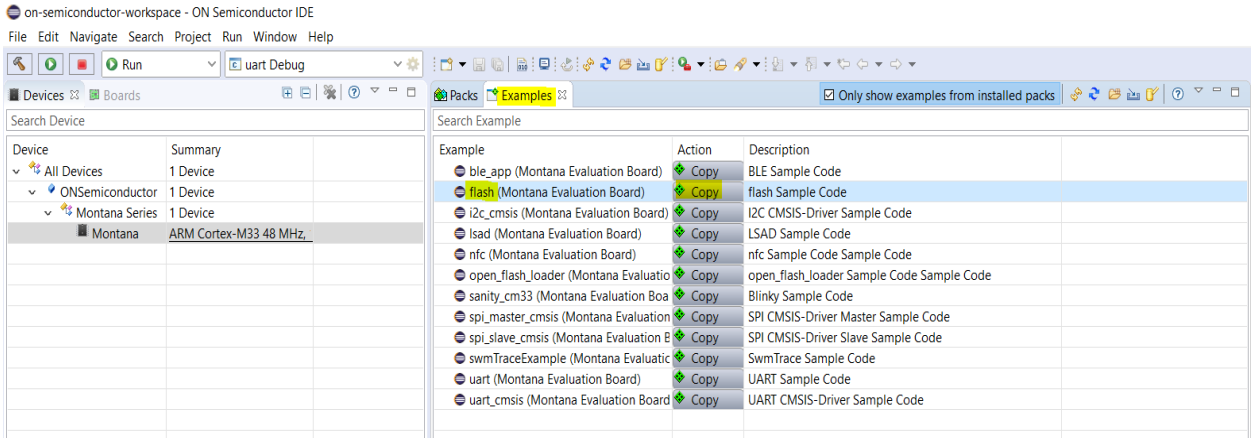

Figure 7: Pack Manager Perspective: Examples Tab

3. The C/C++ perspective opens and displays your newly copied project. In the Project Explorer panel, you can expand your project folder and explore the files inside your project. On the right side, the *flash.rteconfig* file displays software components. If you expand **Device** > **Libraries**, you can see the **HAL library** (libHAL) and the **Startup** (*libcmsis*) components selected for *flash*, as shown in Figure 8.

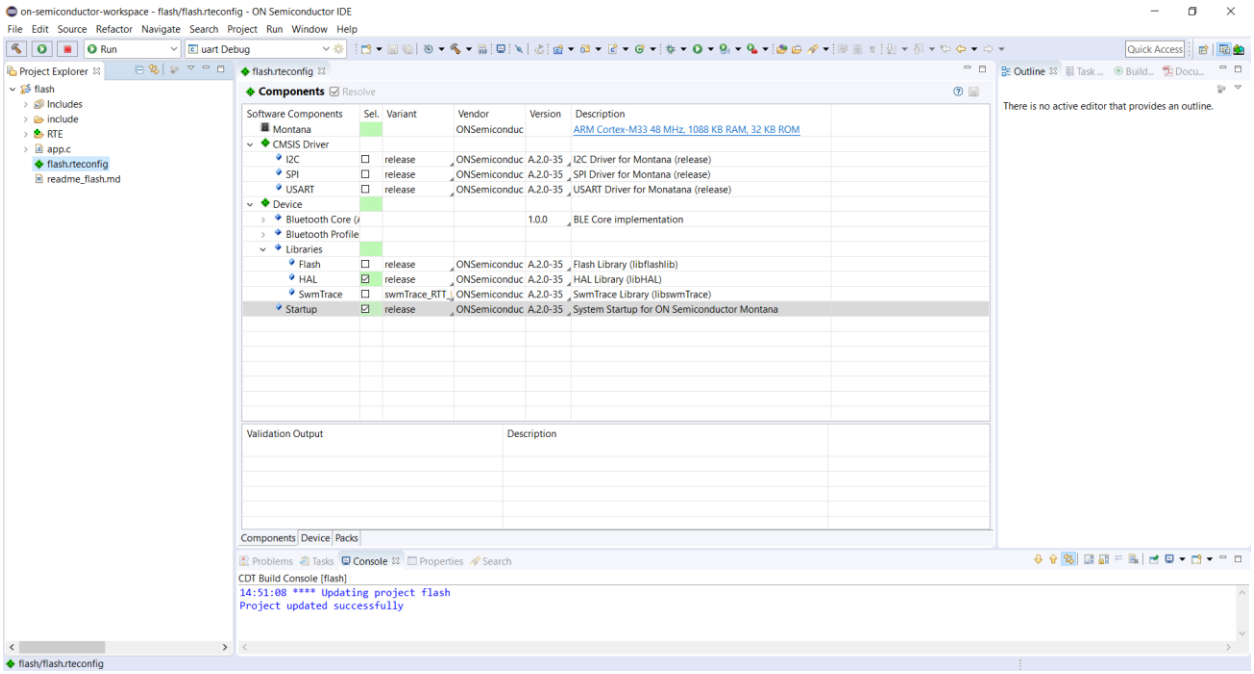

Figure 8: RTE Configuration for the Flash Example Project in the onsemi IDE

3.4 BUILD THE SAMPLE CODE

Follow these steps to build the sample code:

1. Right click on the folder for *flash* and click **Build Project**. Alternatively, you can select the project and click the Build Project icon, which looks like a hammer, as shown in Figure 9.

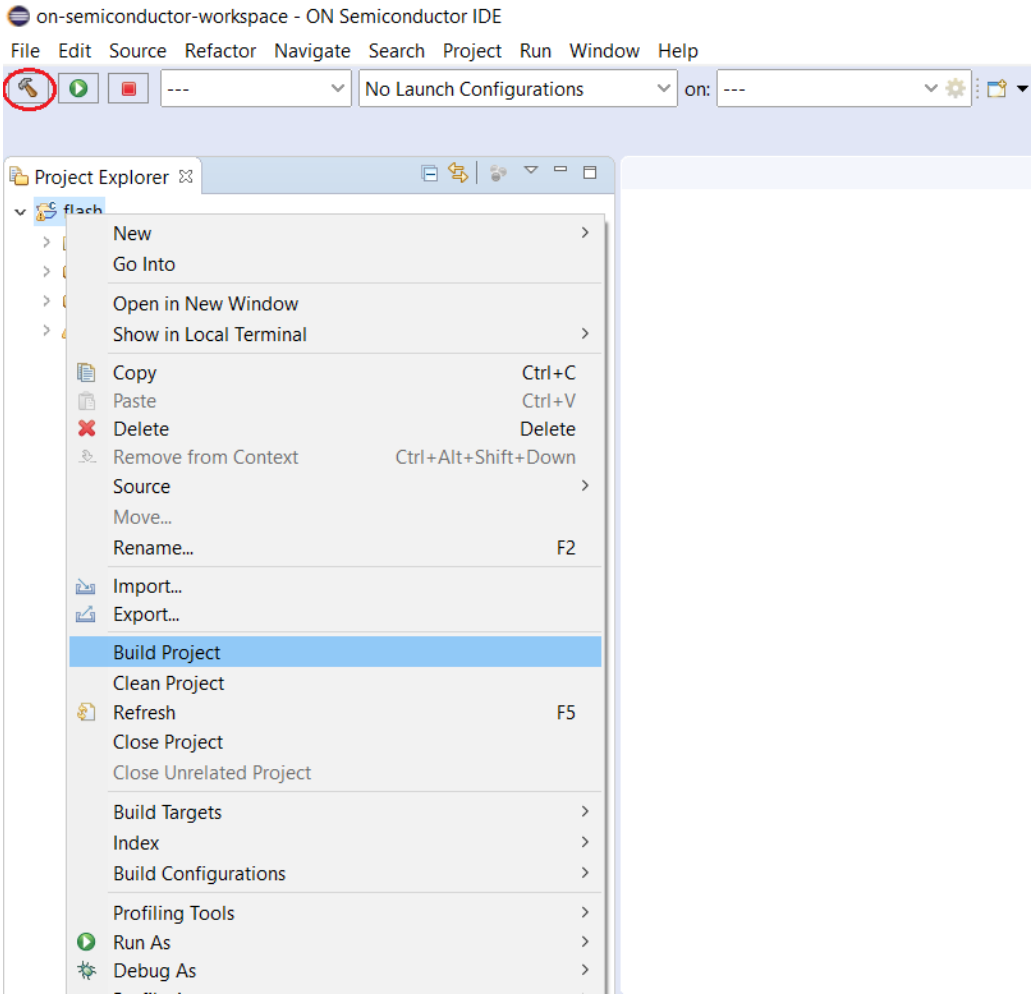

Figure 9: Starting to Build a Project in the onsemi IDE

2. When the build is running, the output of the build is shown in the onsemi IDE C/C++ Development Tooling (CDT) Build Console, as illustrated in Figure 10.

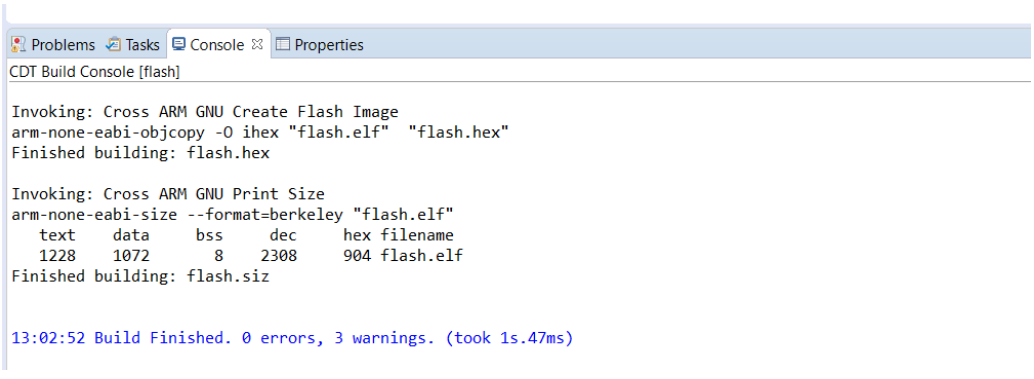

Figure 10: Example of Build Output

3. The key resulting output in Project Explorer, in the Release Folder, includes:

- *flash.hex*: HEX file for loading into Flash memory
- *flash.elf*: Arm executable file, run from RAM, used for debugging
- *flash.map*: map file of the sections and memory usage

These files are shown in Figure 11.

NOTE: You might need to refresh the project to see the three built output files. To do so, right-click on the project name *flash* and choose **Refresh** from the menu, as shown in Figure 11.

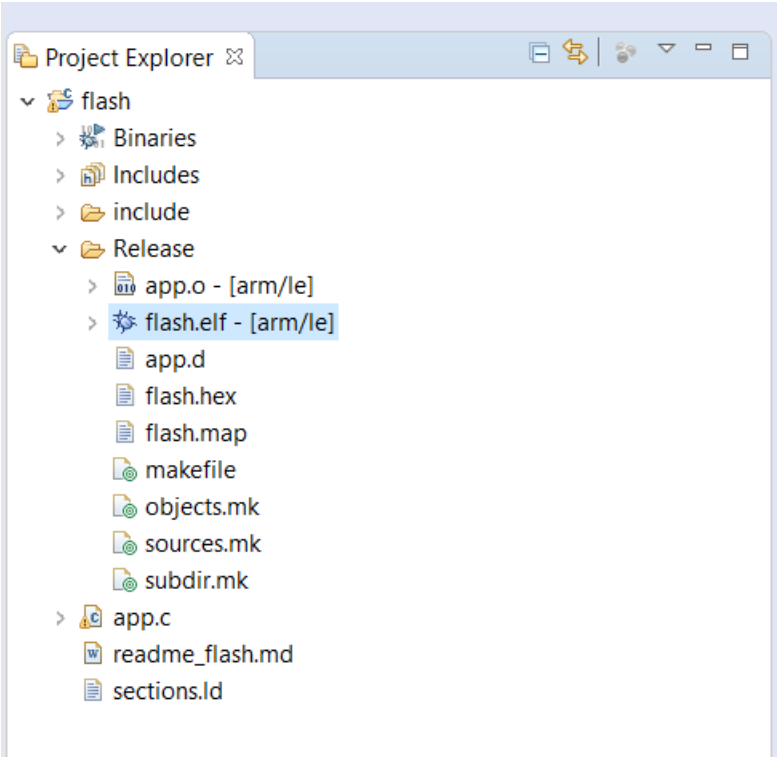

Figure 11: Output Files from Building a Sample Project

3.5 DEBUGGING THE SAMPLE CODE

Debug the application using the *.elf* file as follows:

1. Within the Project Explorer, right-click on the *flash.elf* file and select **Debug As > Debug Configurations…**
2. When the **Debug Configurations** dialog appears, right-click on **GDB SEGGER J-Link Debugging** and select **New Configuration**. A new configuration for *flash* appears under the **GDB SEGGER** heading, with new configuration details in the right side panel.

3. Change to the **Debugger** tab, and enter Montana in the **Device Name** field. Ensure that **SWD** is selected as the target interface (as shown in Figure 12).

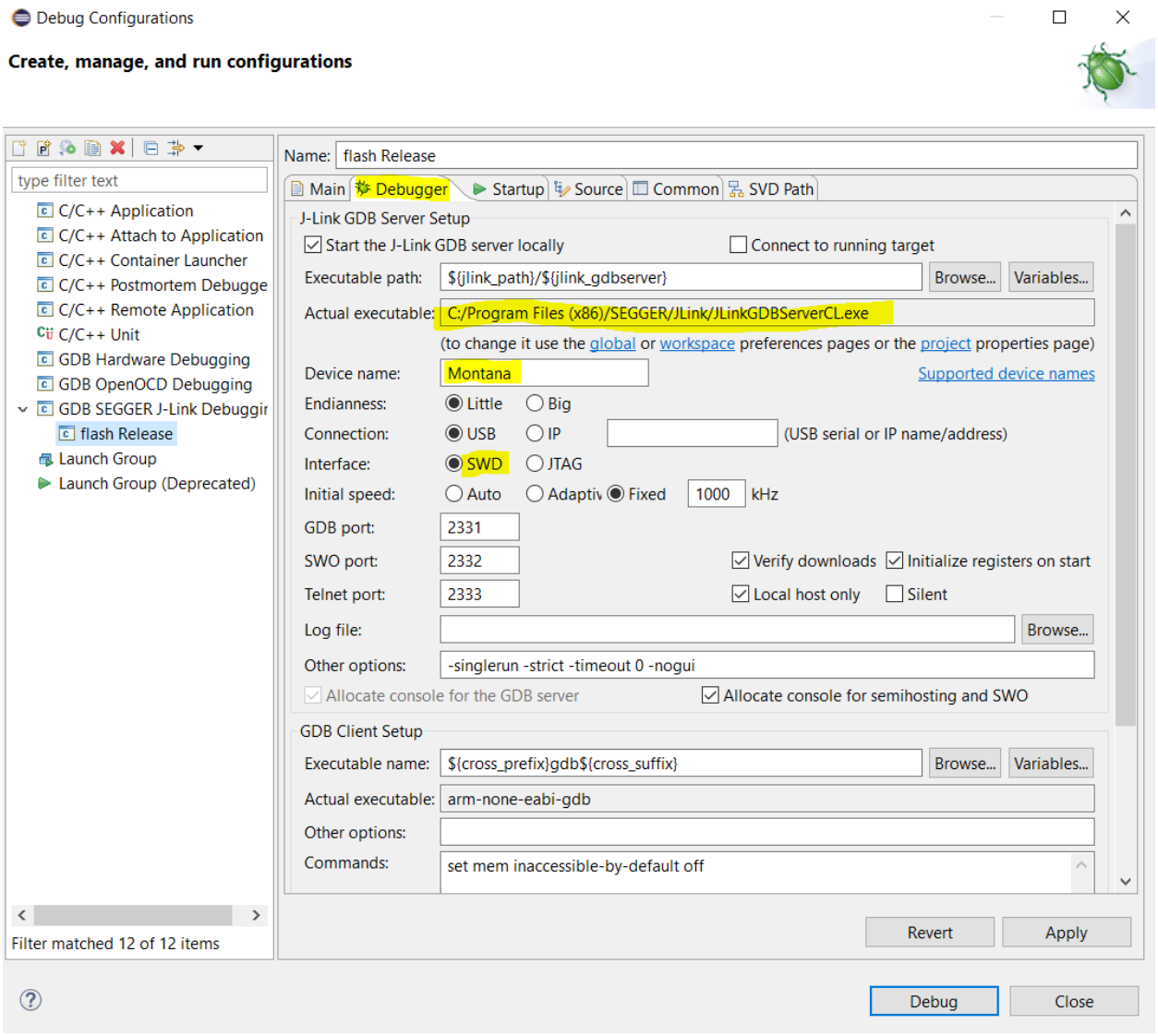

Figure 12: Setting Up a GDB Launch Configuration, Debugger Tab

4. Change to the Startup tab, uncheck the **Initial Reset and Halt** checkbox, and enter the following information in the Pre-run/Restart reset (as shown in Figure 13):
*set $sp = *((int *) &ISR_Vector_Table)*

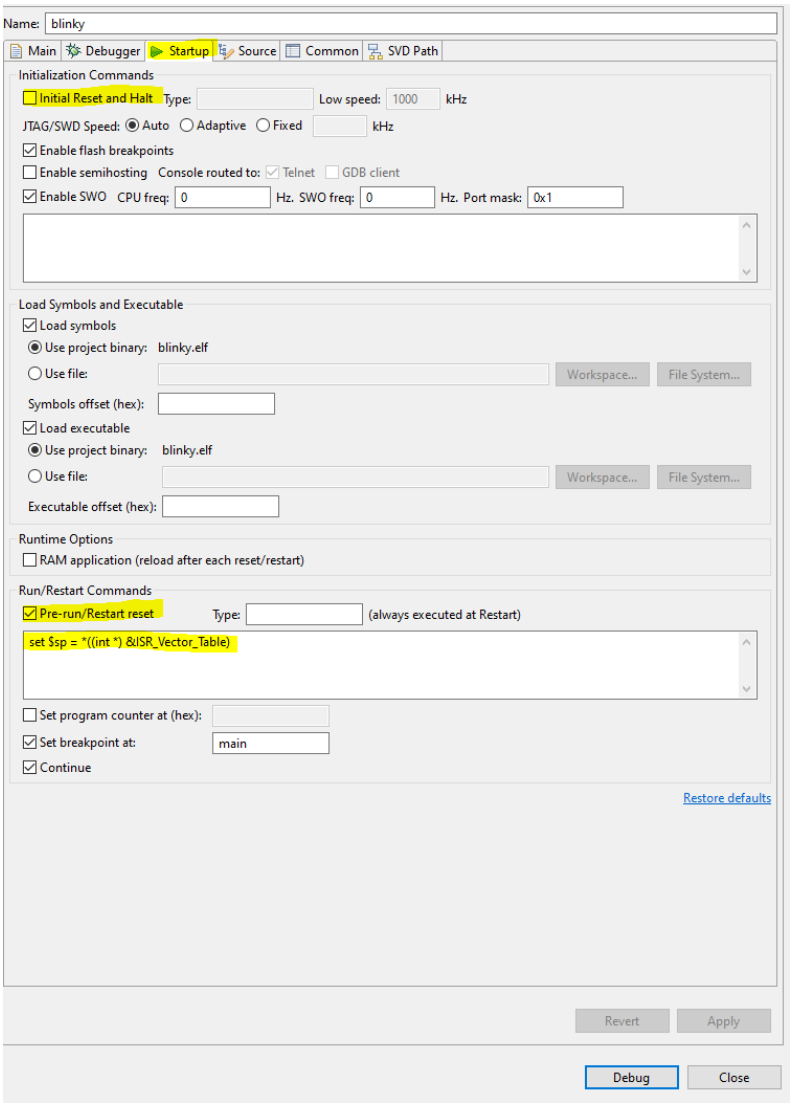

Figure 13: Setting Up a GDB Launch Configuration, Startup Tab

5. Once the updates to the configuration are completed, make sure that the Evaluation and Development Board is connected to the PC, and click **Debug**. J-Link automatically downloads the *flash* sample code to Montana's flash memory.

   NOTE:
   - For running the code from flash, a linker script (*sections.ld*) located in *firmware\configuration* can be used.
   - For running the code from RAM, a linker script (*sections_ram.ld*) located in *firmware\configuration* can be used.
   - When running the code from RAM, check the **RAM application** box in the debug configuration found in the Startup tab's **Runtime Options** area, as shown in Figure 14.

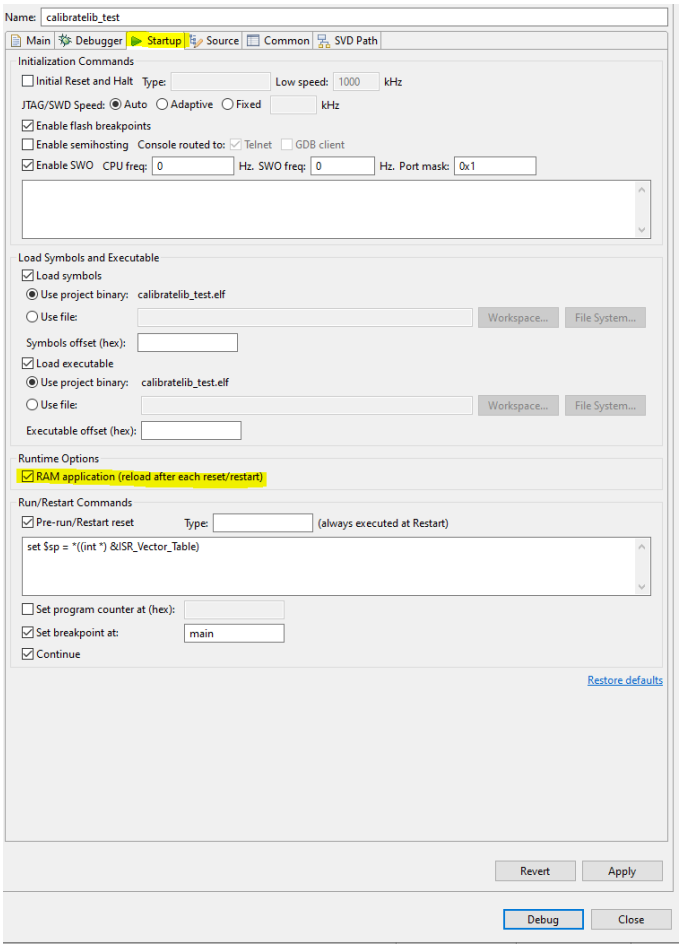

Figure 14: Running Code from RAM, Startup Tab