Freescale Semiconductor, Inc.
Application Note

Document Number: AN5229
Rev. 0, 01/2016

ARM DS-5 Development Studio Debug i.MX6UL-EVK

1. Introduction

This application note is intended to introduce the ARM DS-5 IDE debug functions based on an MX6UL EVK board.

Contents

2. ARM® DS-5 Development Studio Introduction

The ARM DS-5 is a professional software development solution for Linux-based and bare-metal embedded systems, covering all the stages in development, from boot code and kernel porting to application debug.

DS-5 includes the following component tools:

1. Eclipse-based IDE combines software development with the compilation technology of the DS-5 tools.
2. DS-5 Compilation Tools.
3. DS-5 Debugger, together with a supported debug target, enables debugging of application programs and complete control over the flow of program execution to quickly isolate and correct errors.

DS-5 has three different editions: Community, Professional, and Ultimate. This Application Note is based on the ARM DS-5 Ultimate Edition.

3. i.MX 6UltraLite-Low-power, Secure, ARM® Cortex®-A7 Core

Expanding the i.MX 6 series, the i.MX 6UltraLite is a high performance, ultra-efficient processor family featuring an advanced implementation of a single ARM Cortex-A7 core, which operates at speeds up to 528 MHz. The i.MX 6UltraLite applications processor includes an integrated power management module that reduces the complexity of external power supply and simplifies power sequencing. Each processor in this family provides various memory interfaces, including 16-bit LPDDR2, DDR3, DDR3L, raw and managed NAND flash, NOR flash, eMMC, QSPI SPI, and a wide range of other interfaces for connecting peripherals such as WLAN, Bluetooth™, GPS, displays, and camera.

- Features
  - ARM Cortex-A7 @ 528 MHz, 128 KB L2 cache
  - Parallel LCD Display up to WXGA (1366x768)
  - 8/10/16/24-bit Parallel Camera Sensor Interface
  - 16-bit LP-DDR2, DDR3/DDR3L
  - 8/16-bit Parallel NOR FLASH / PSRAM
  - Dual-channel Quad-SPI NOR FLASH
  - 8-bit Raw NAND FLASH with 40-bit ECC
  - 2xMMC 4.5/SD 3.0/SDIO Port
  - 2xUSB 2.0 OTG, HS/FS, Device or Host with PHY
  - Audio Interfaces include 3x I2S/SAI, S/PDIF Tx/Rx
  - 2x10/100 Ethernet with IEEE 1588
  - 2x12-bit ADC, up to 10 input channel total, with resistive touch controller (4-wire/5-

wire)

- Partial PMU Integration
- Security Block: TRNG, Crypto Engine (AES/TDES/SHA/RSA with DPA), Tamper Monitor, Secure Boot, SIMV2/EVMSIM X 2, OTF DRAM Encryption, PCI4.0 pre-certification
- 14x14 289MAPBGA 0.8mm pitch
- 9x9 289 MAPBGA 0.5mm pitch

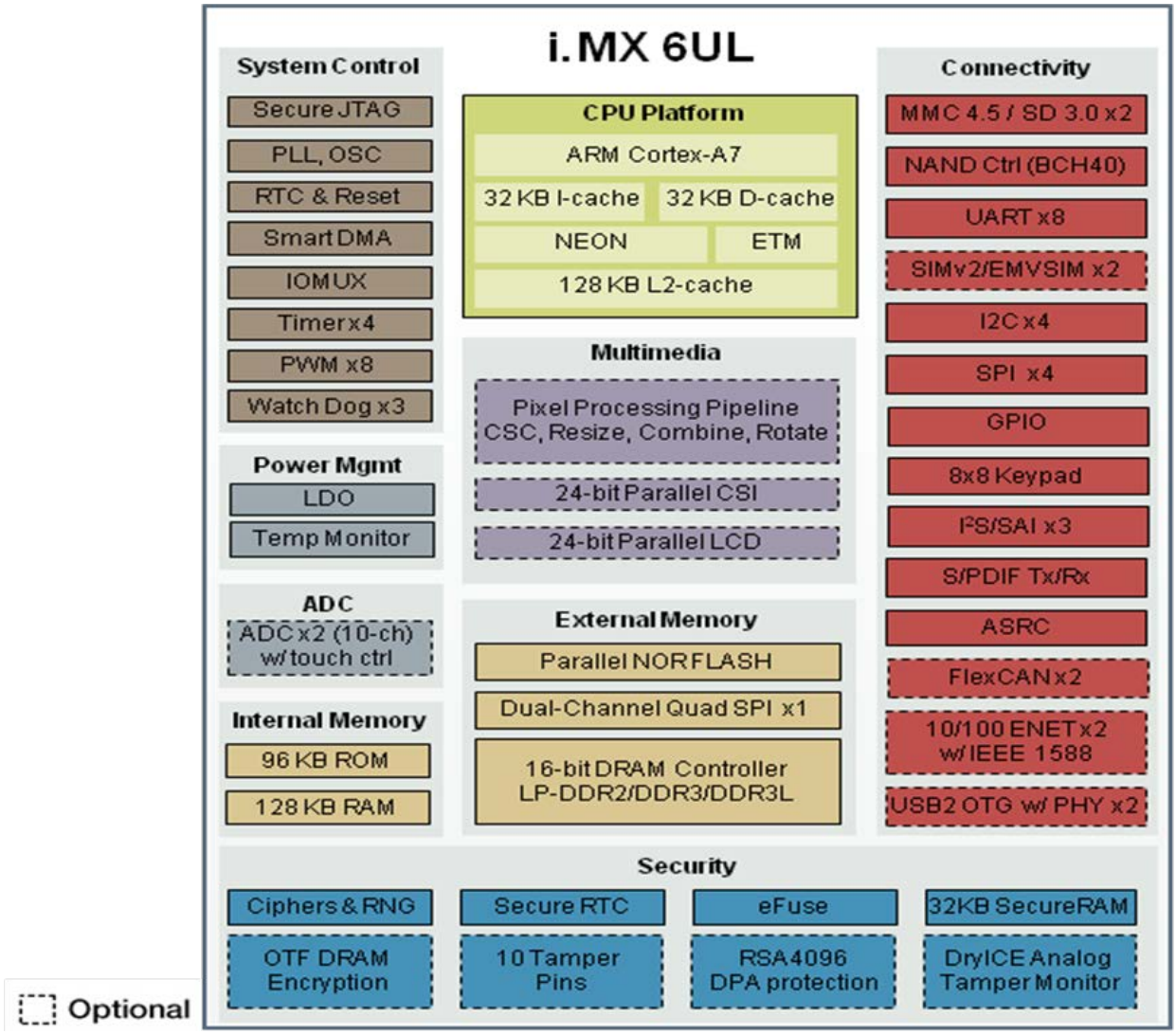

Figure 1. i.MX 6UL MPU Block Diagram

4. Requirements

- ARM DS-5 Software for Linux or Windows
- ARM RealView-ICE Debugger or J-LINK Debugger
- i.MX 6UtraLite EVK Rev.C (HW rework needed)
- JTAT Connection Between Debug and i.MX 6UtraLite EVK
- Ethernet/USB Connection Between Host PC and Debugger

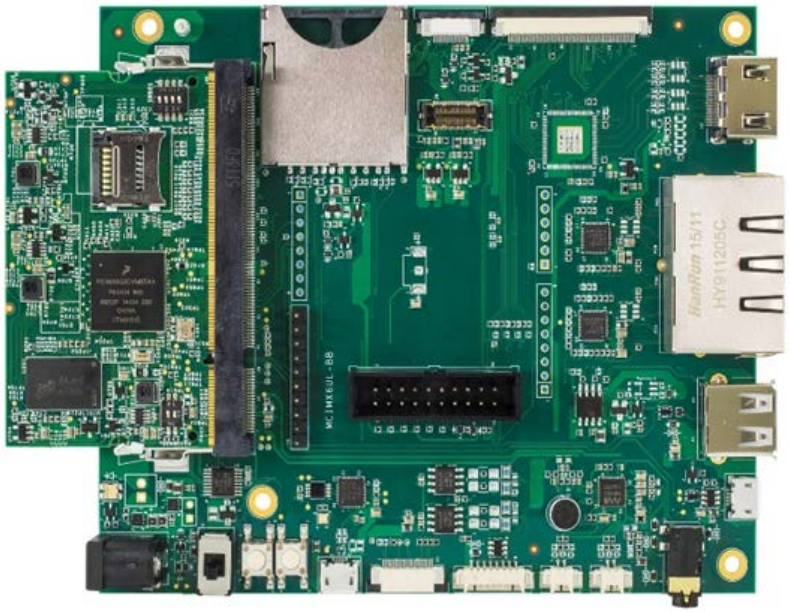

Figure 2. i.MX 6UL-EVK board

NOTE:

The JTAG Port pin is reused by Audio codec WM8960 SAI2 pin on the i.MX 6 UL-EVK board. In order to correctly connect to the JTAG port of the board with the RealView-ICE Debugger or the JLINK debugger, five resistances R1432, R1407, R1431, R1433, and R1434 need to be removed. This is shown in the following figure.

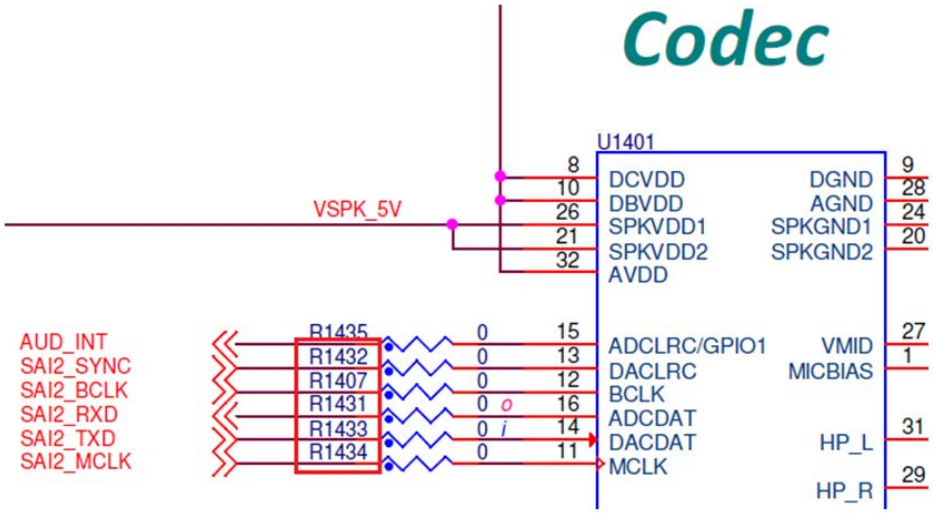

Figure 3. i.MX 6UL-EVK Hardware rework for JTAG pin

The RealView-ICE Debugger is connected to the i.MX 6UL-EVK board via the JTAG port:

Figure 4. i.MX 6UL-EVK connection with RealView-ICE Debugger

5. ARM DS-5 IDE

The ARM Development Studio (DS-5) IDE is based on the Eclipse Platform.

The workbench is the main development environment where you can manage individual projects, associated sub-folders, and source files. It uses a single folder called a workspace to store files and folders related to specific projects. A typical workbench window contains one or more perspectives, a set of related views, editors, menus, and toolbars.

DS-5 uses the C/C++ and DS-5 Debug perspectives.

Further details can be found at: http://ds.arm.com/

5.1. How to install

The ARM DS-5 can be found at: http://ds.arm.com/downloads/. Install it by following the step by step wizard.

Open DS-5 IDE, click the Help menu, select the ARM License Manager, and generate a 30-day free license.

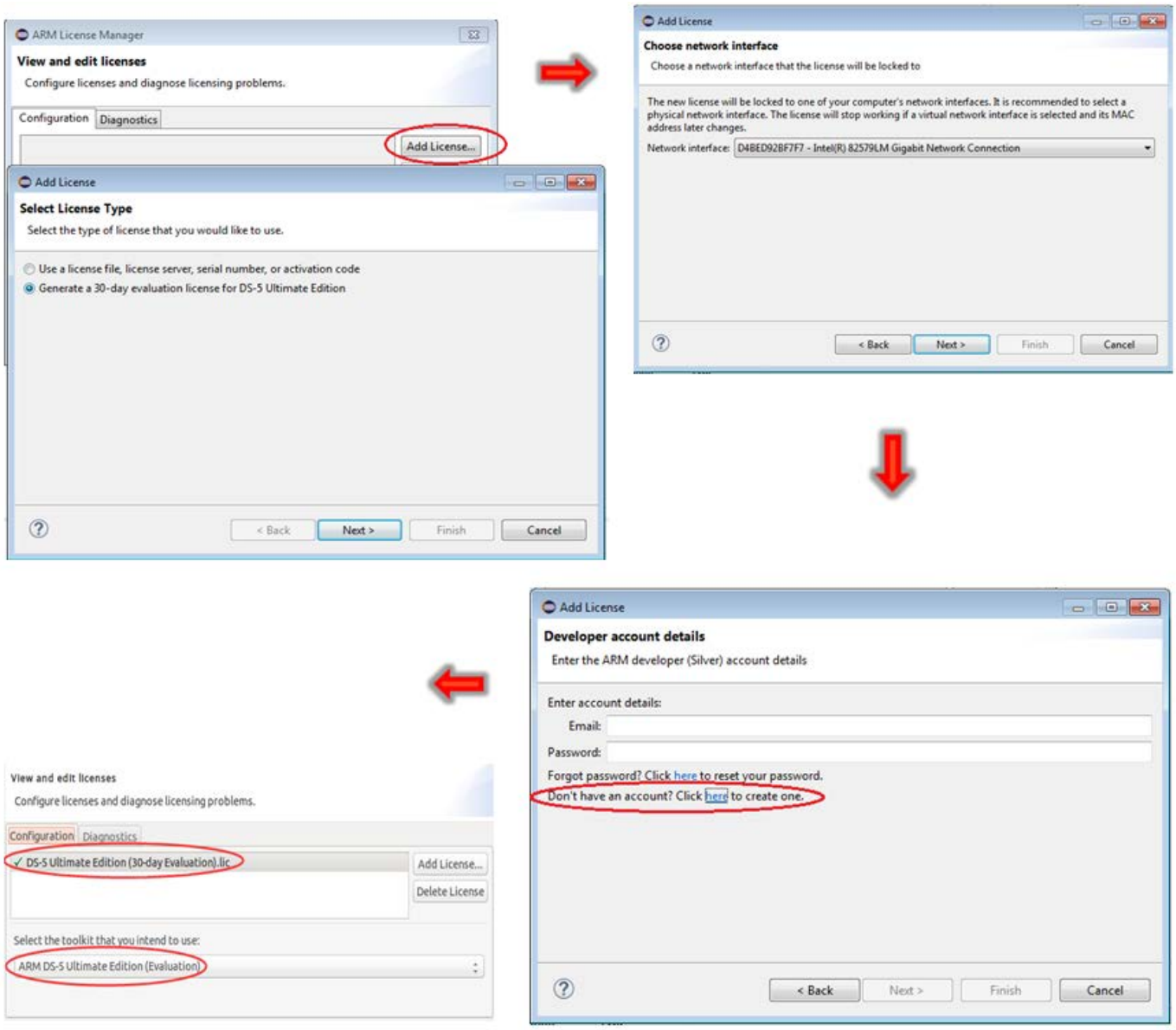

Figure 5. Install the license

When connecting the RealView-ICE Debugger with the Host PC via Ethernet/USB for the first time, you may need to update the debugger firmware. To do this, open the Debug Hardware Update in the Windows Start Menu and update the firmware with the following wizard:

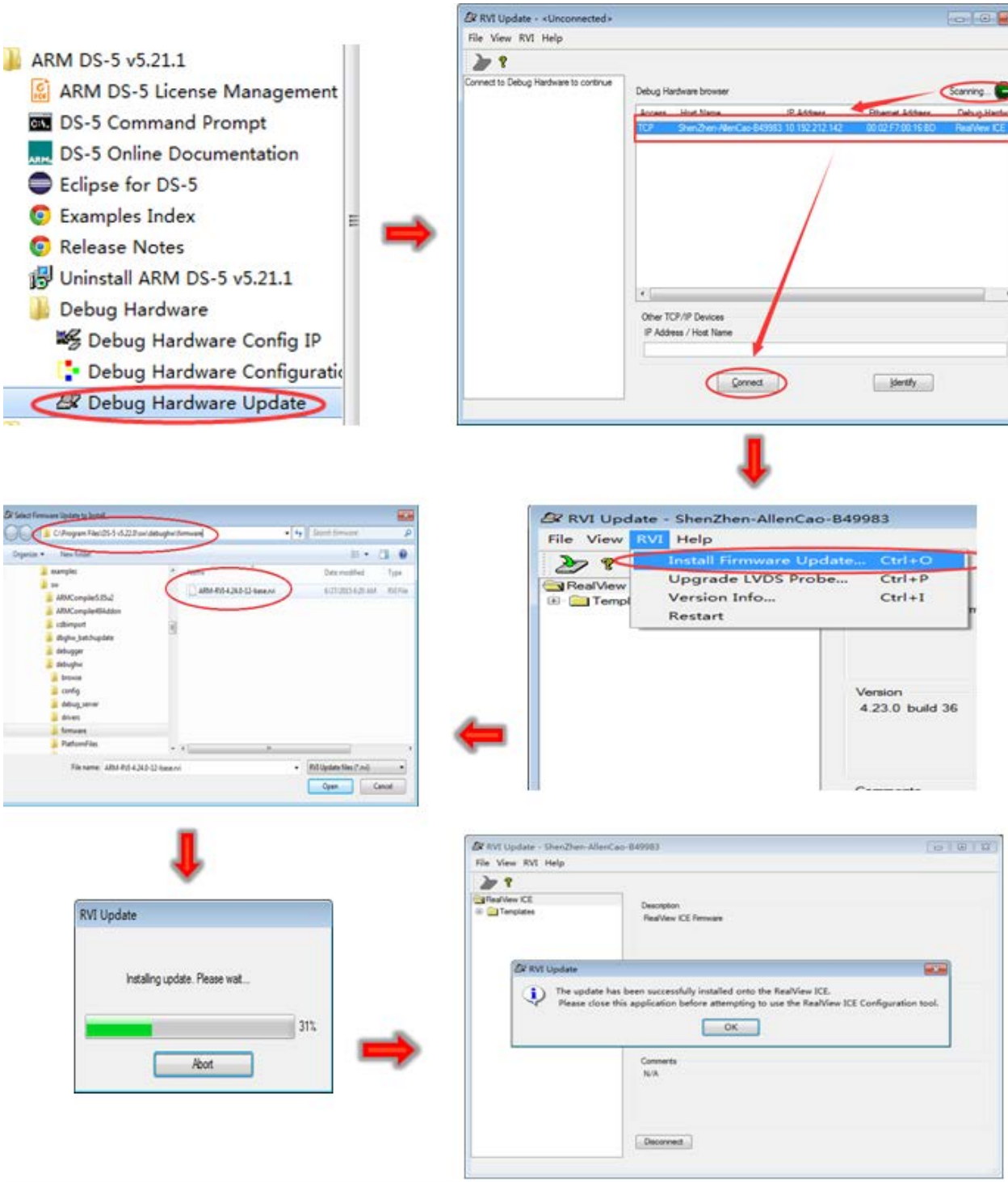

Figure 6. Update the RealView-ICE Debugger firmware

5.2. How to create a project

Eclipse IDE can show a range of programs. There are two classes of project; the first class of project requires ARM DS-5 to build and the binary image. Create the project as follows:

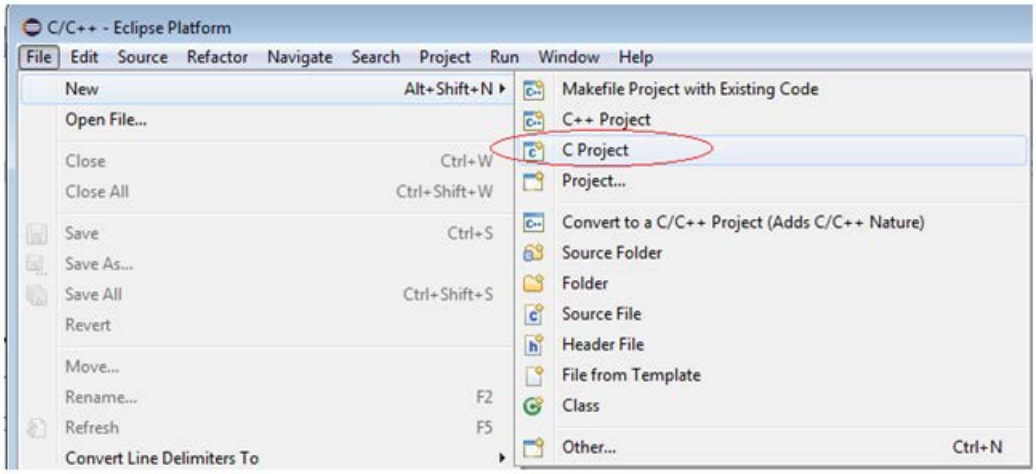

Figure 7. Create a built project

You must select the correct Toolchains to build this project:

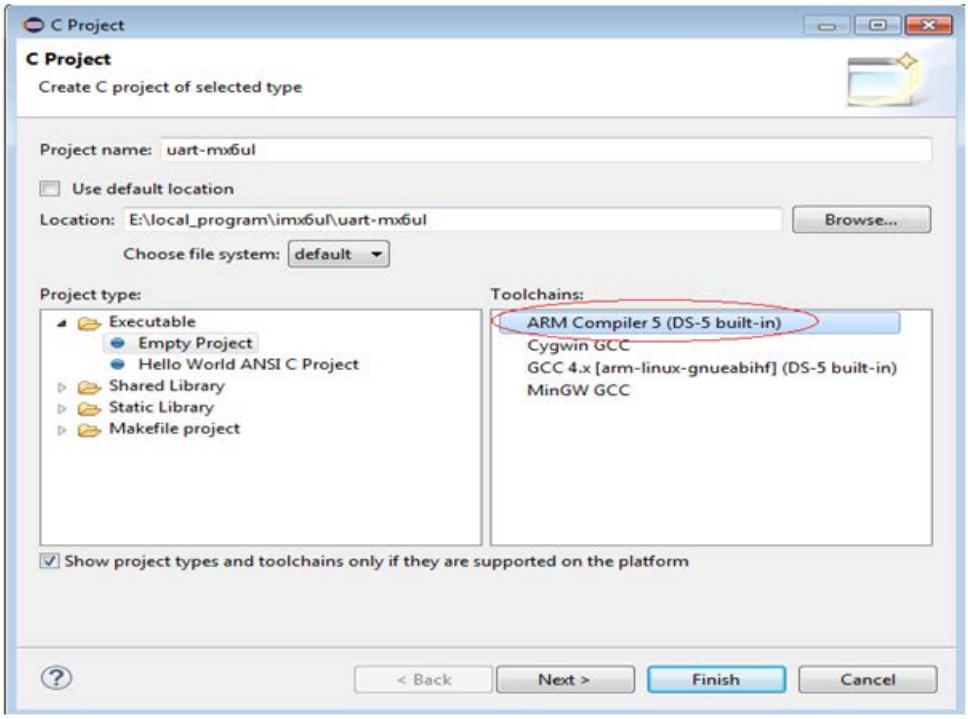

Figure 8. Select ARM Compiler 5 Toolchains for building

The second class of project is when ARM DS-5 is not used to build the project. The project can be built by Linux Host. You must import existing source codes:

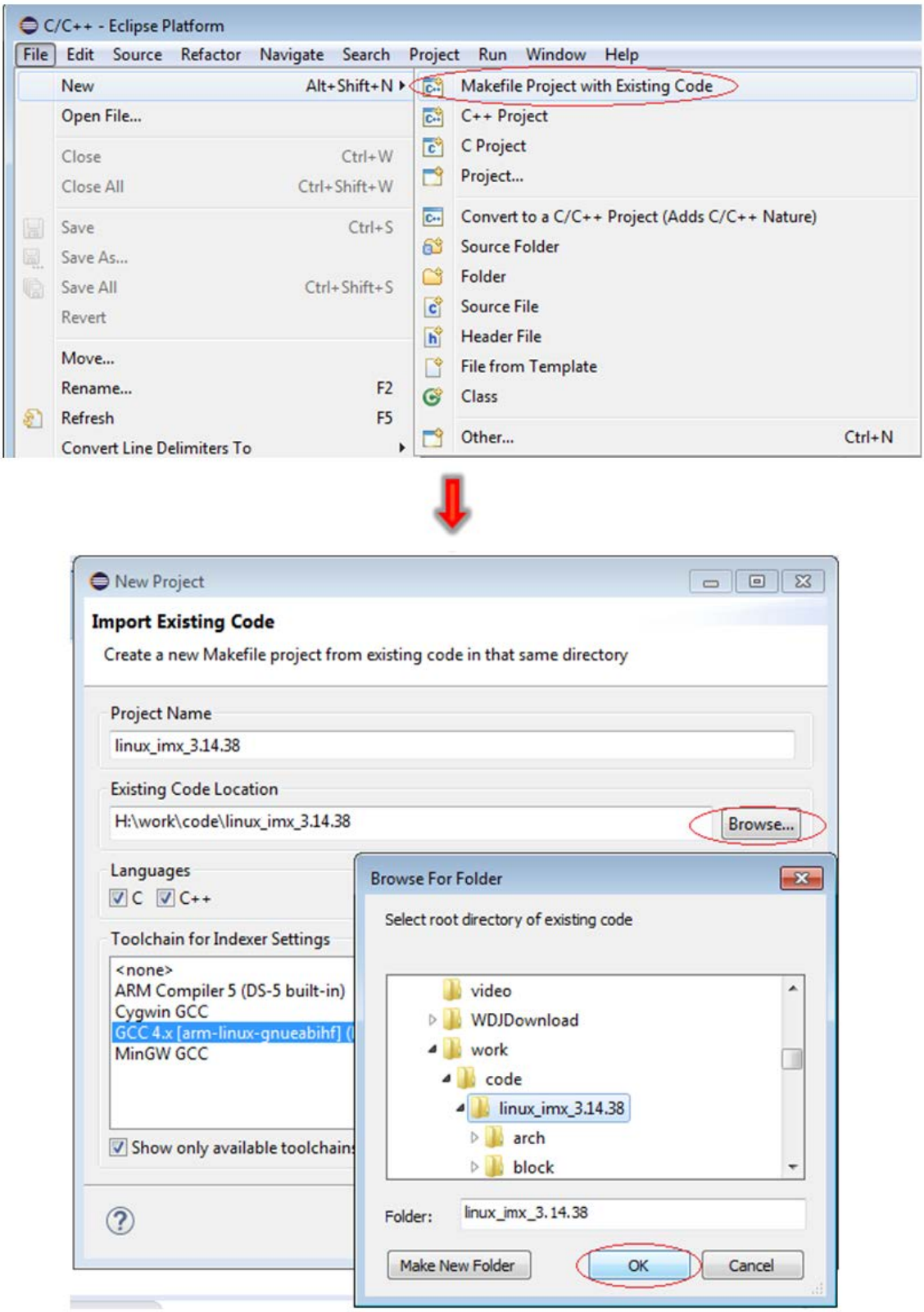

Figure 9. Create a no built project

6. RealView-ICE Debug

Realview-ICE is a powerful ARM core debugger, it supports JTAG port connection.

Figure 10. RealView-ICE debugger

6.1. Debug Bare Metal Code

Open ARM DS-5 and create a built project as shown in section 5.2. Import the source code to the project:

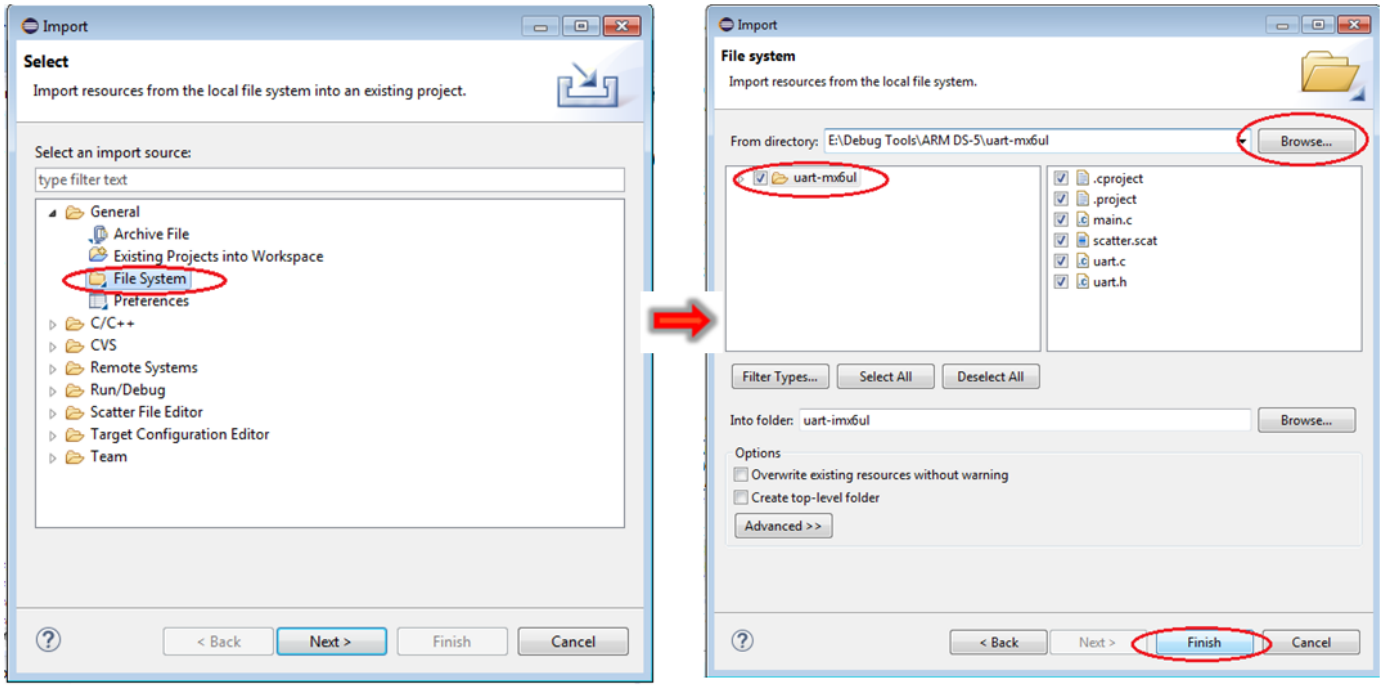

Figure 11. Import the source code to the project

Configure the Project Properties then select the ARM C Compiler Target:

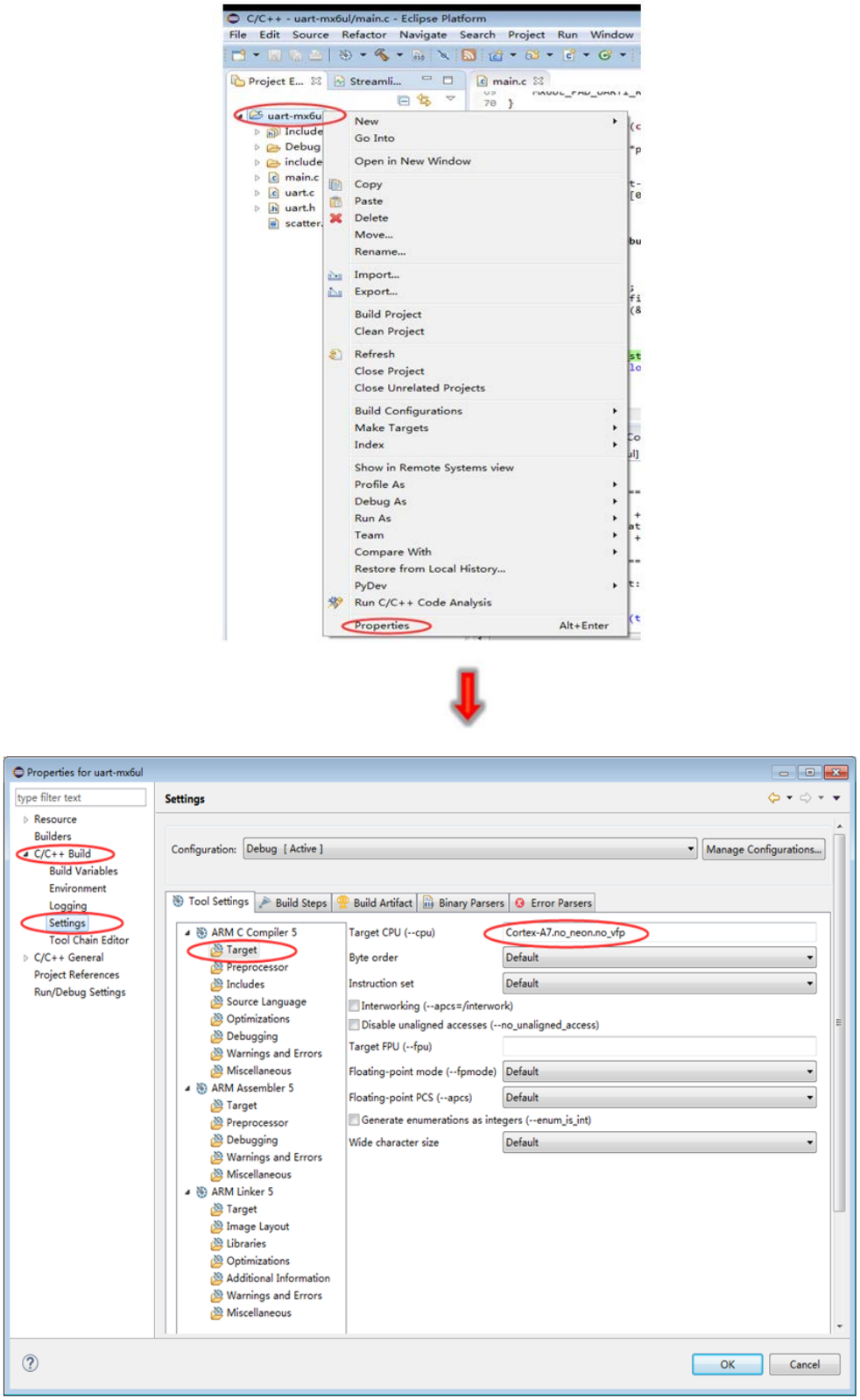

Figure 12. Select the ARM C Compiler 5 Target

Select the ARM Assembler 5 Target Configuration:

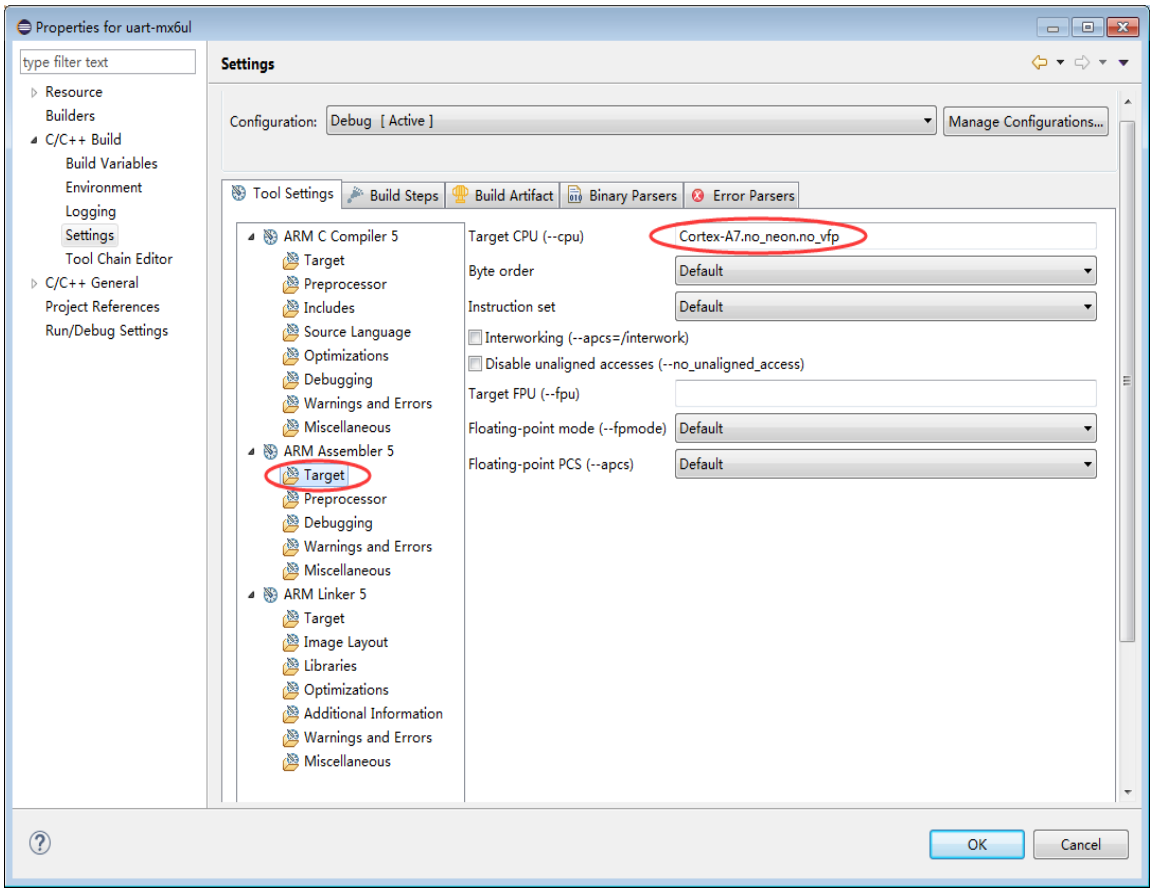

Figure 13. Select the ARM Assembler 5 Target

Select the ARM Linker Target Configuration:

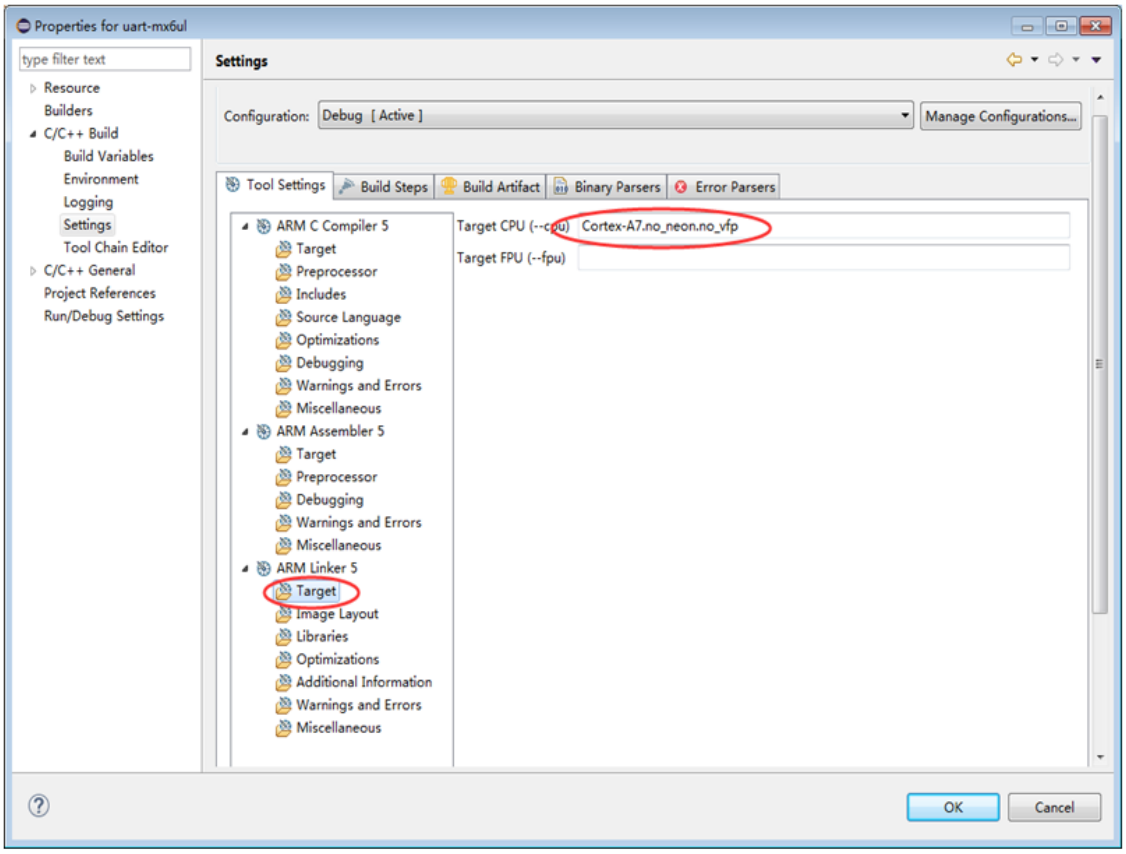

Figure 14. Select the ARM C Linker Target

Select Image Layout (Scatter file) Configuration,

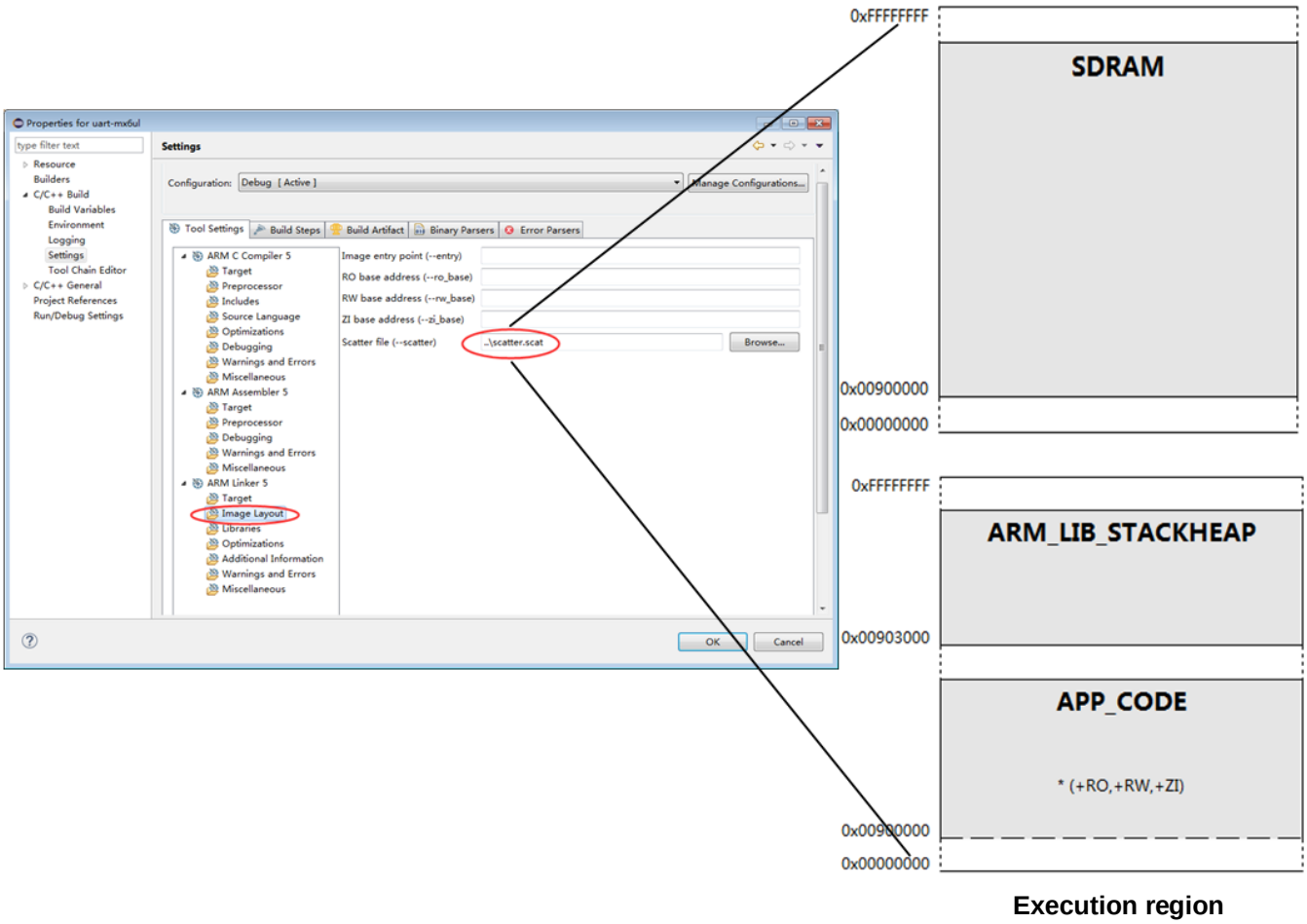

Figure 15. Select the scatter file

Build the project; right click at the project title to select Build Project:

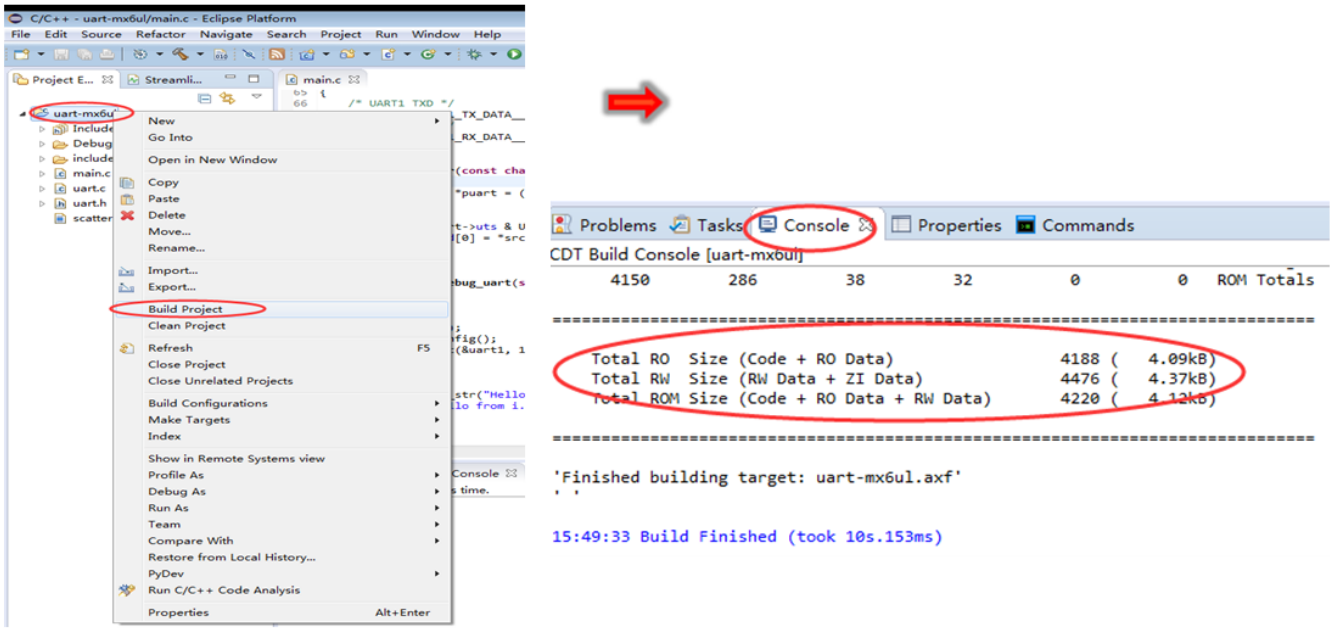

Figure 16. Build the project

Configure the DS-5 debugger connection:

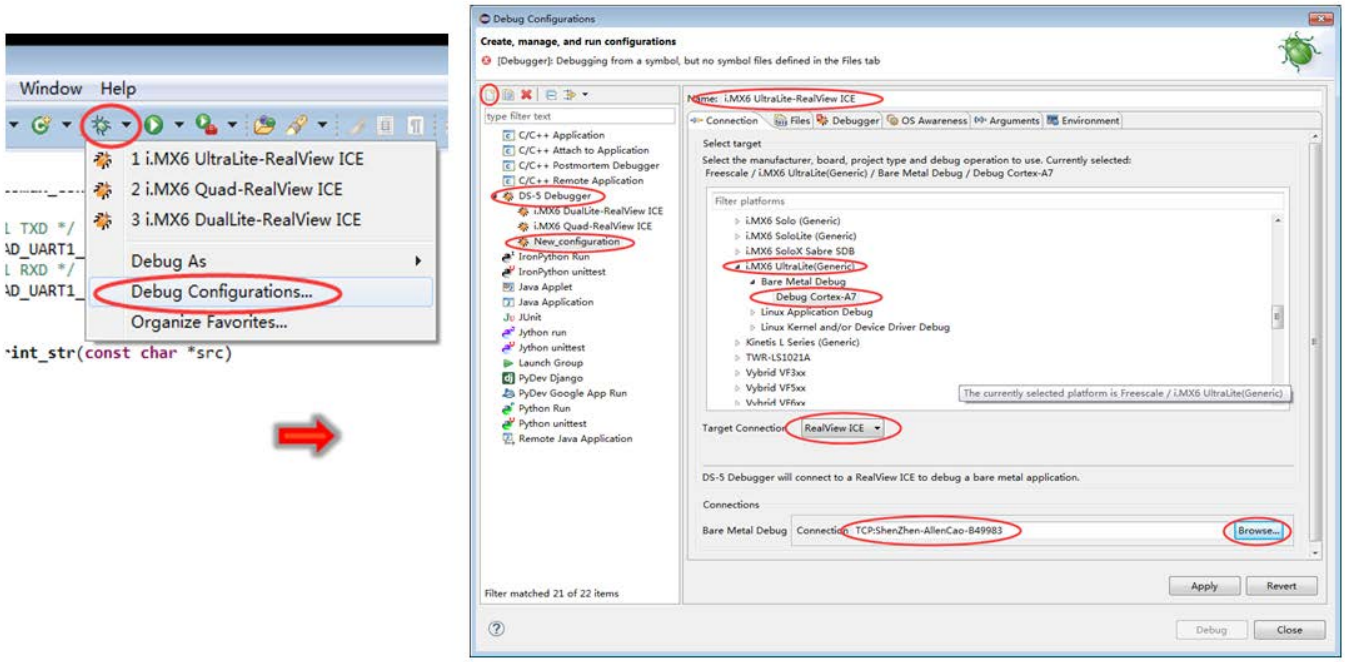

Figure 17. Configure DS-5 debugger connection

NOTE:

if there is no “i.MX6 UltraLite(Generic)” item in Connection->Select Target, copy folder “i.MX6 UltraLite(Generic)” in “mx6ul_evk_ds-5\DS-5-Board-i.MX6 UL\Boards\Freescale” to path “$(ARM DS-5 Install path)\DS-5 v5.22.0\sw\debugger\configdb\Boards\Freescale”.

Configure the DS-5 Debugger Files:

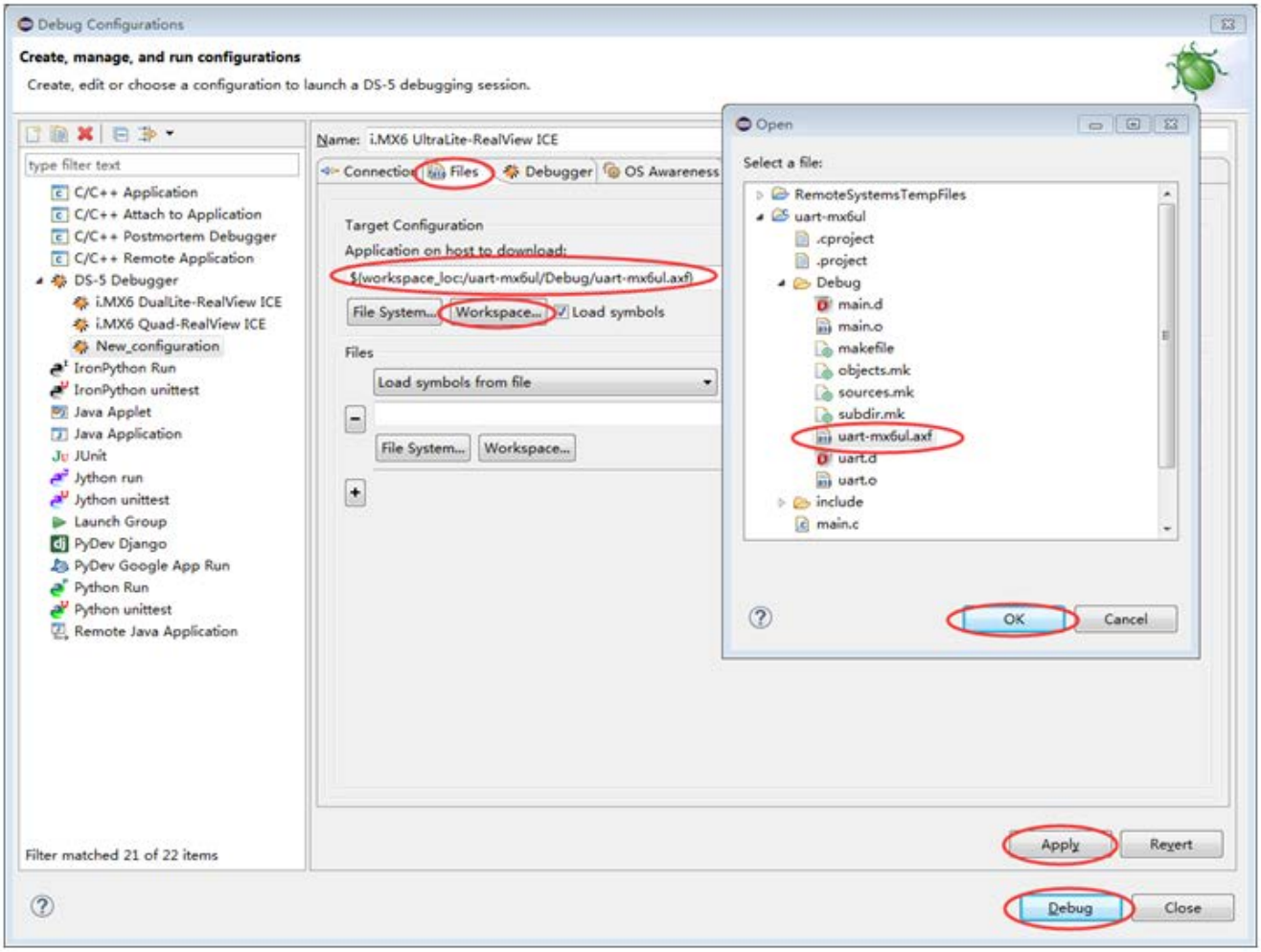

Figure 18. Configure the DS-5 Debugger Files Connection

To initialize the target's chip register, select Debug Configurations-> Debugger-> Run target initialization debugger script.

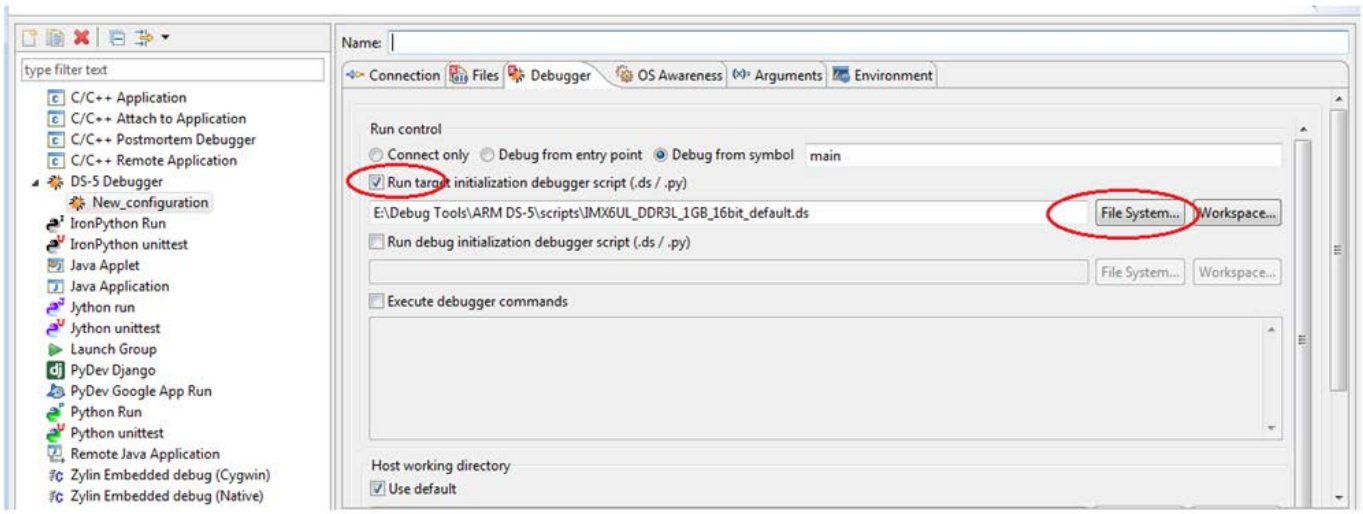

Figure 19. Configure target initialization debugger script

When the debug connects successfully, the DS-5 Debug UI is displayed:

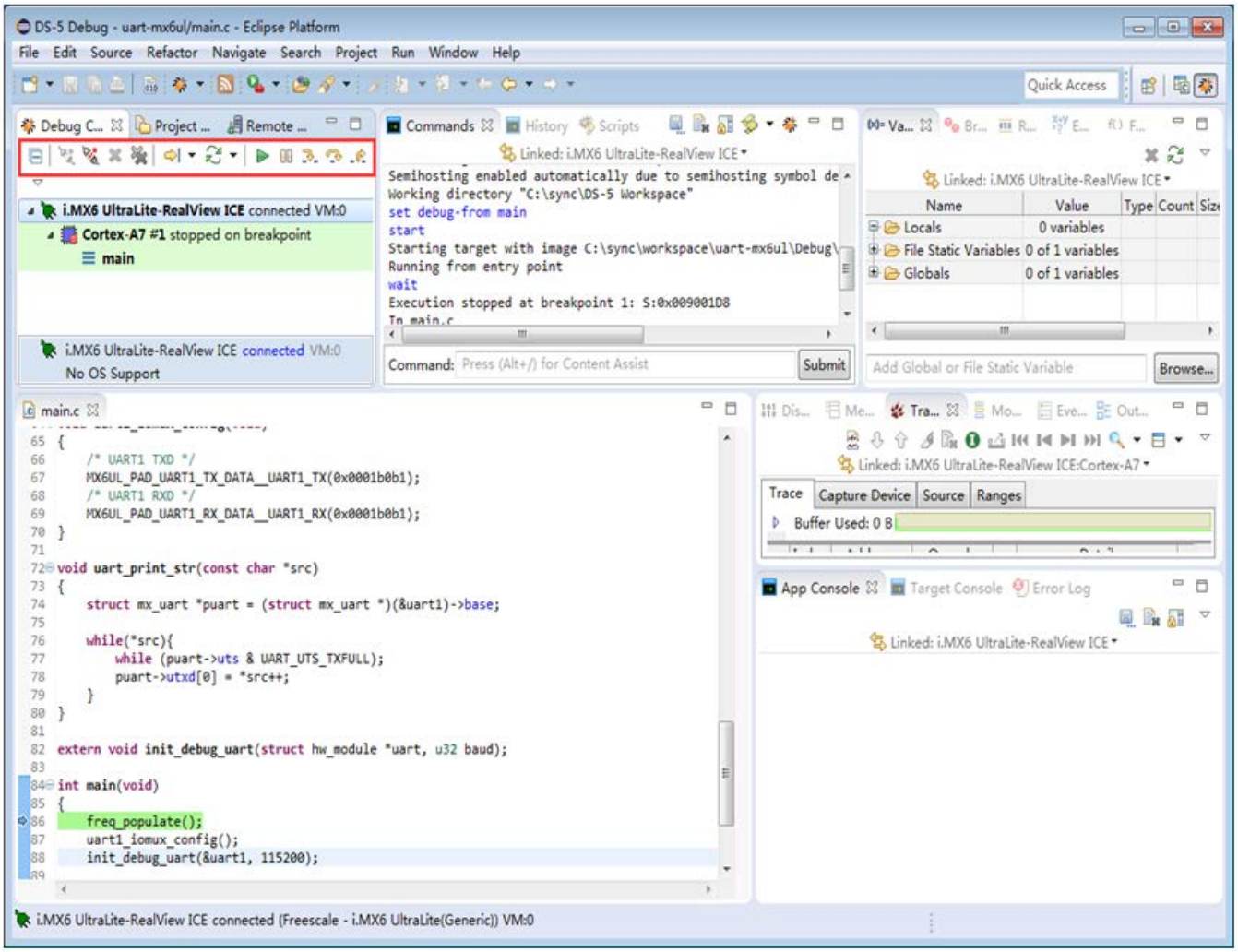

Figure 20. DS-5 Debugger Connected View

Debug and Run:

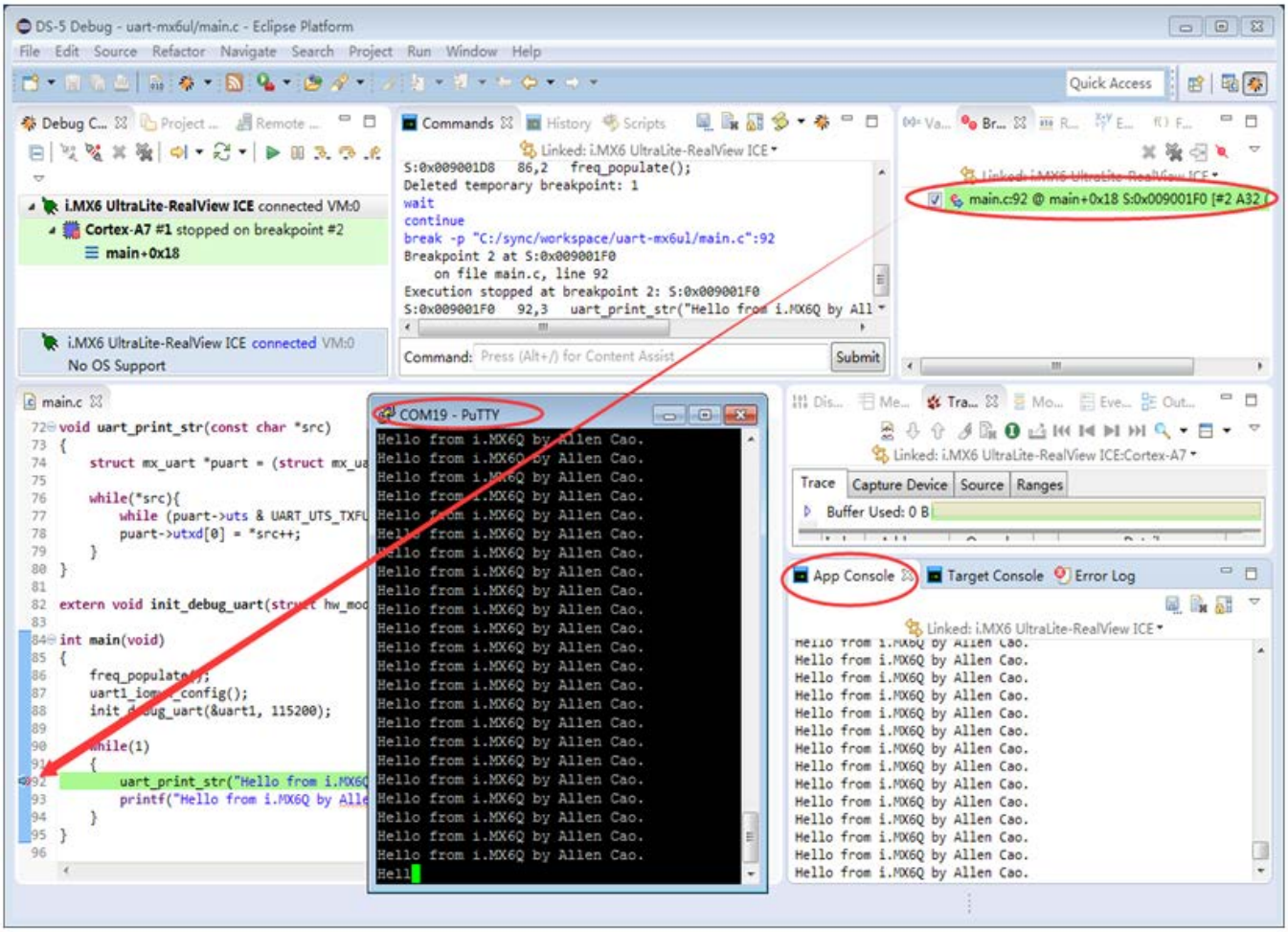

Figure 21. DS-5 Debug view

6.2. Debug U-Boot Code

Before debug, the U-Boot must be compiled on a Linux Host.

Open the ARM DS-5, create a no-built project as in section 5.2, and import your U-Boot source code.

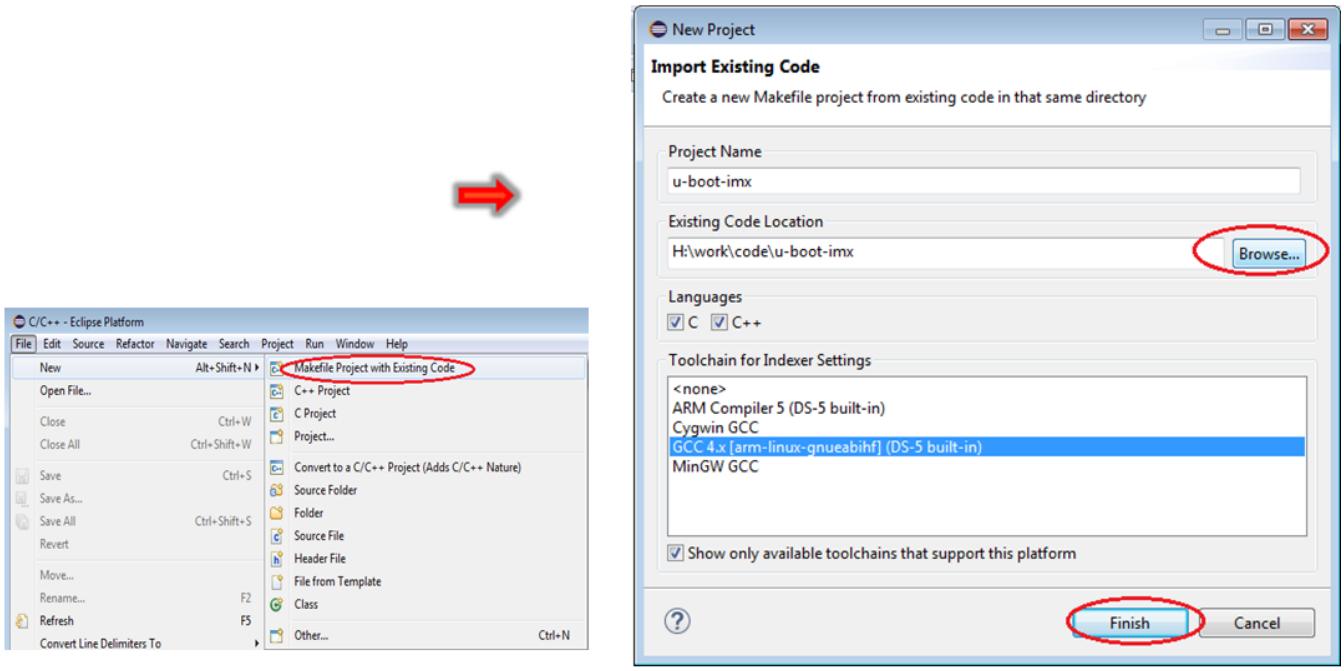

Figure 22. Create U-boot project

After the project has been created, use the following command to setup a boot SD card:

$ sudo dd if=u-boot.imx of=/dev/sdc bs=512 seek=2

Connect your i.MX6UL-EVK board Debug UART to your PC and open the PC's serial terminal. Turn on the power switch to boot the MX6UL EVK board; the serial terminal will then display the U-Boot log if setup has been successful. Quickly type any key in the terminal view to stop the U-Boot at its command line:

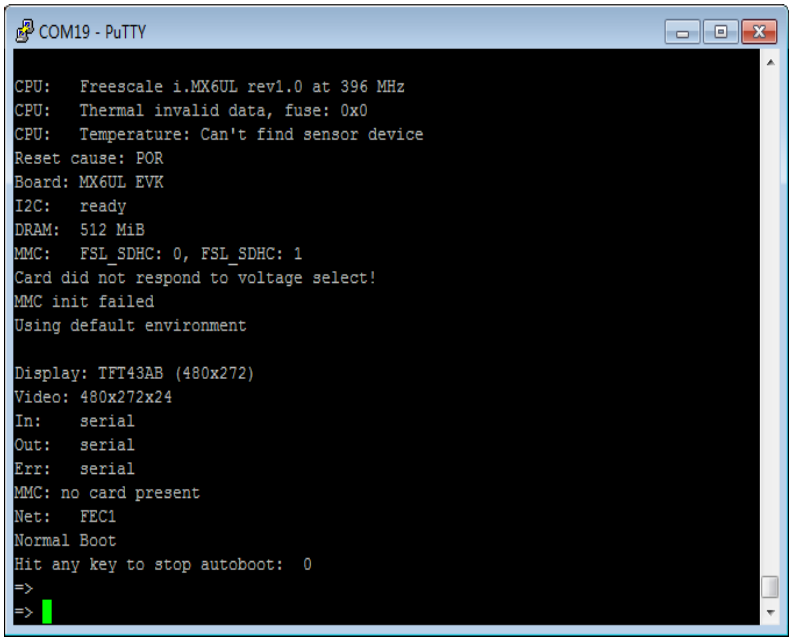

Figure 23. U-boot command prompt line

Configure the DS-5 Debug Connection:

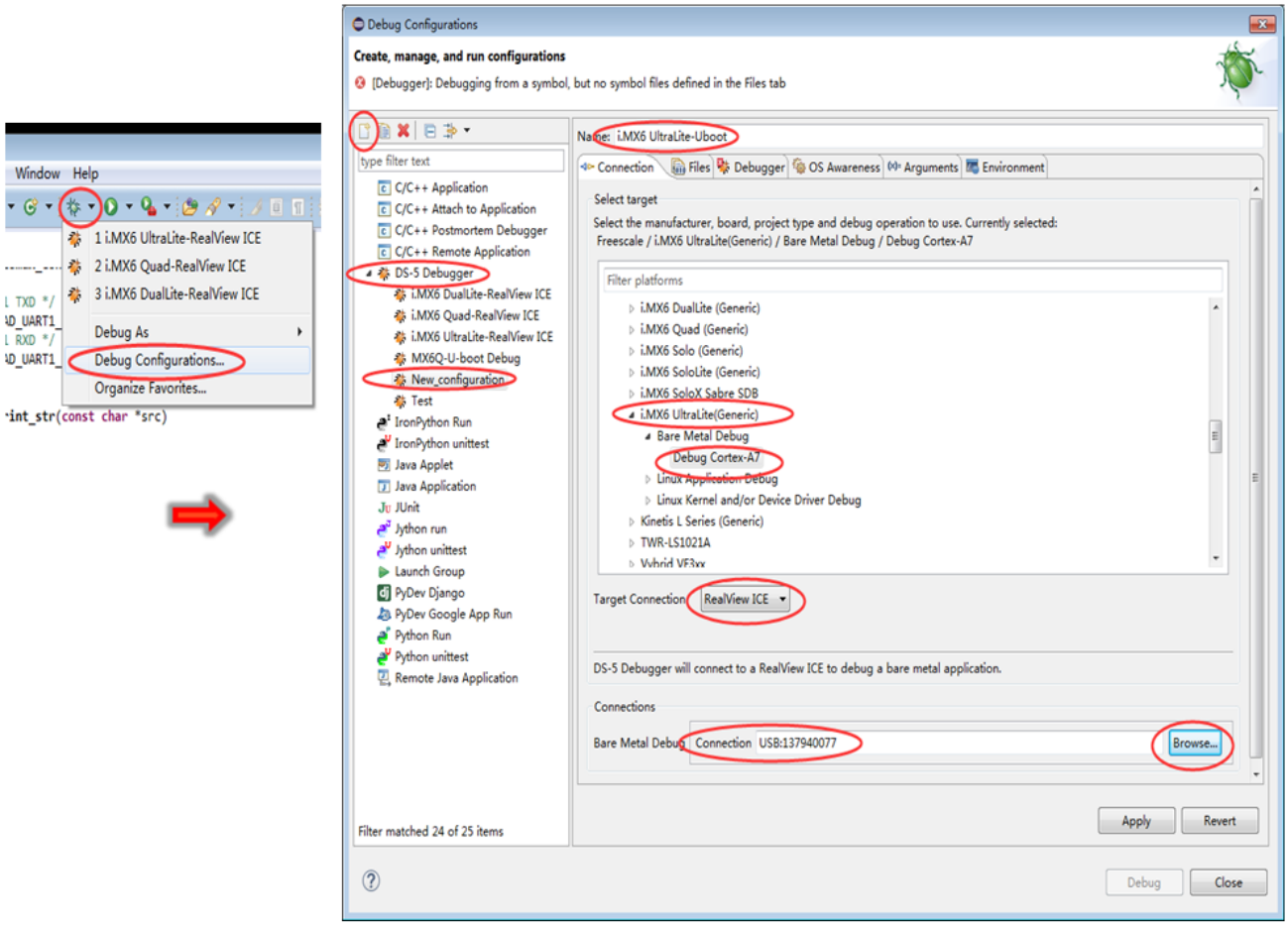

Figure 24. DS-5 Debugger Connections Configuration

Configure the DS-5 Debugger Configuration and click on start Debug at the bottom right:

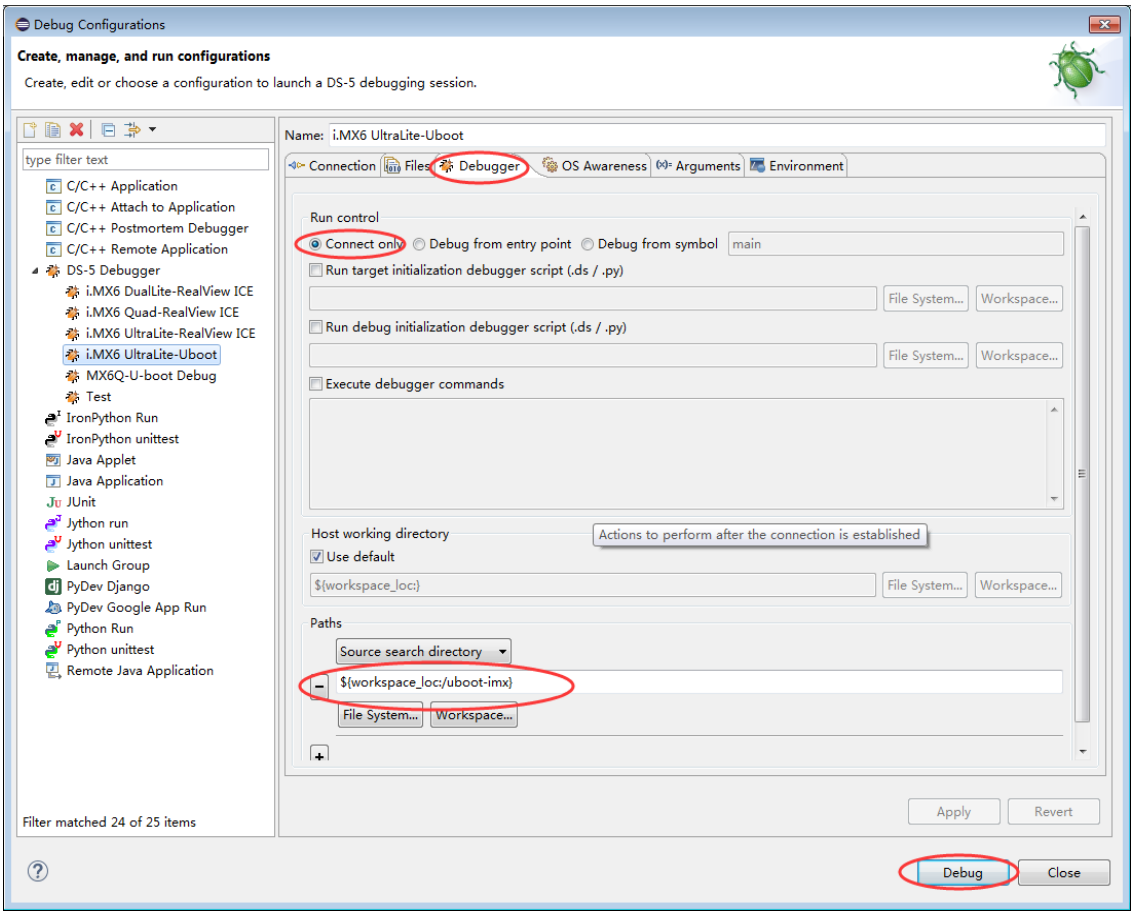

Figure 25. DS-5 Debugger Configurations

When connected successfully, open the DS-5 Debug UI, click on Run and then interrupt it when debug starts:

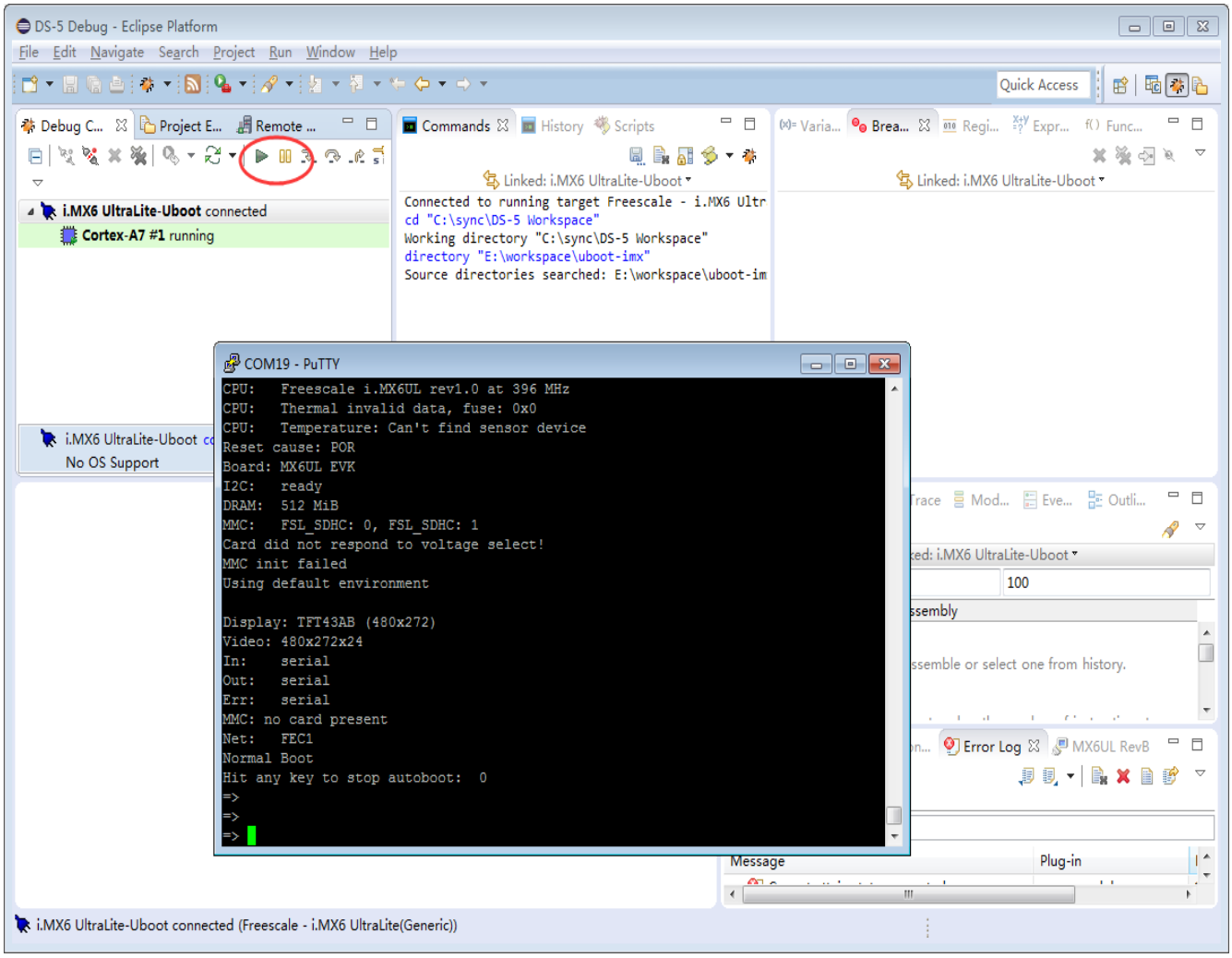

Figure 26. DS-5 Debugger Connected View

After the U-Boot has been interrupted, then Load, Add Symbols File and click on U-Boot:

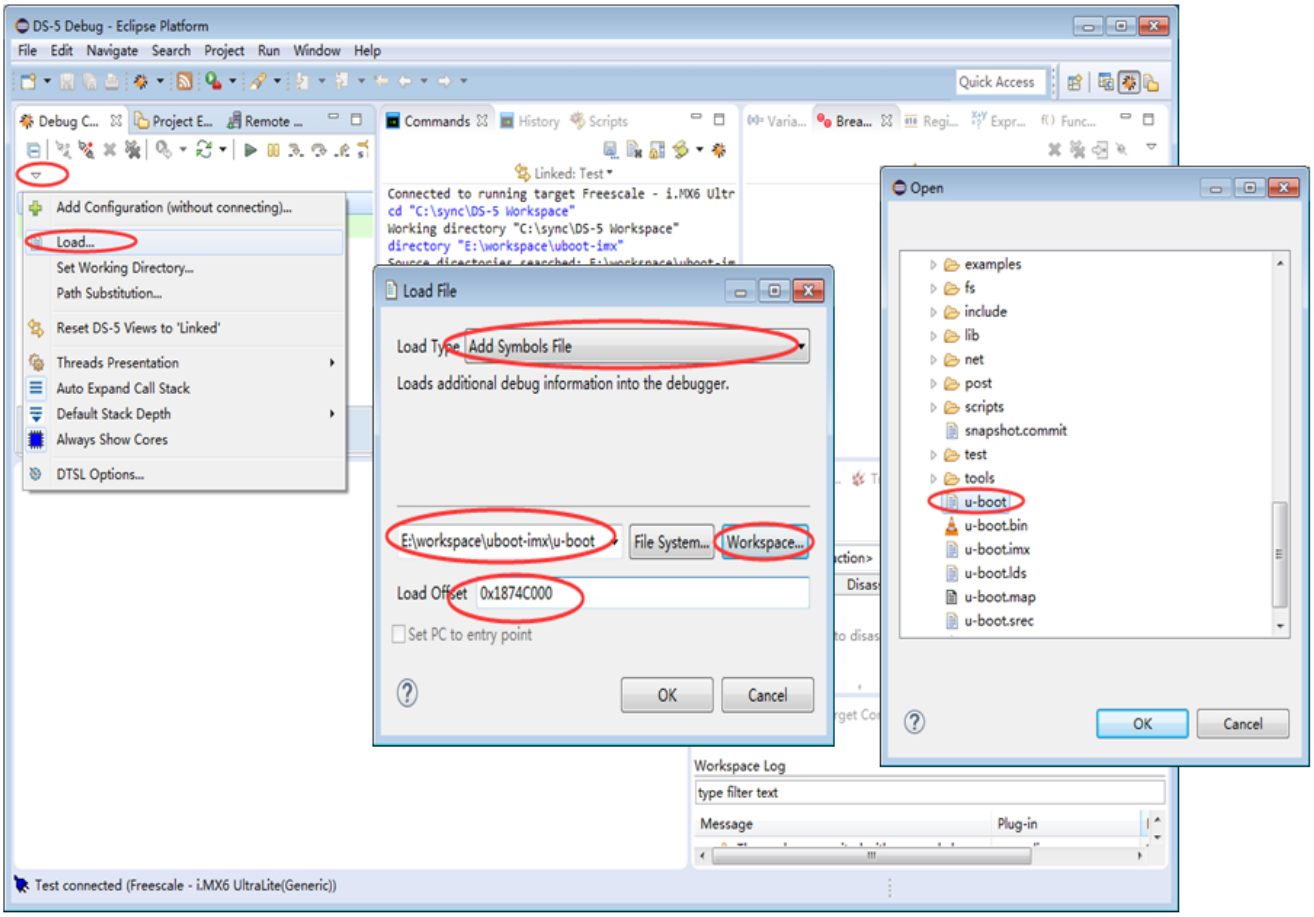

Figure 27. Load Symbol File

Run and then Interrupt the U-Boot, you can then see where Cortex-A7 has stopped and the functions in the stack:

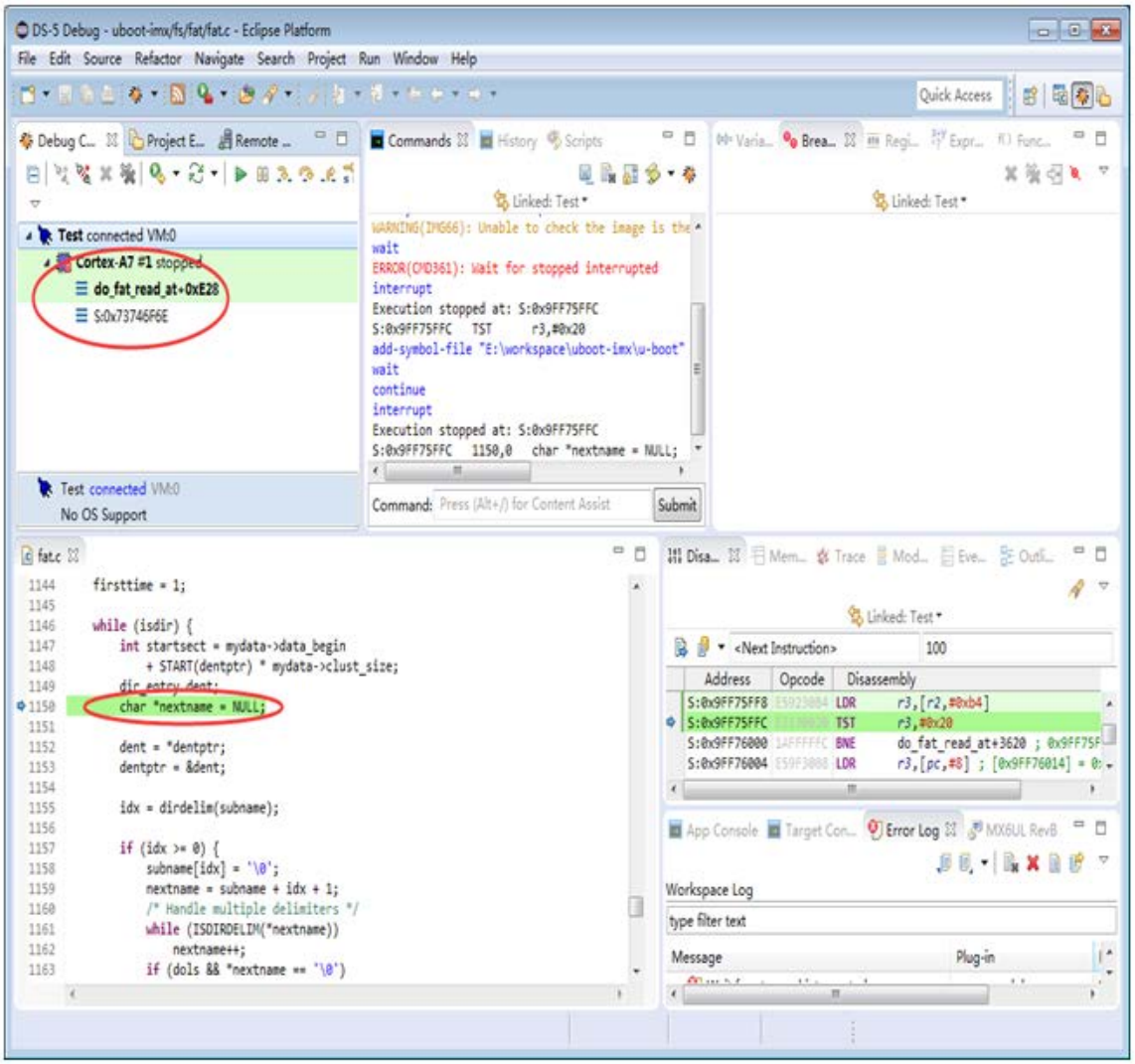

Figure 28. U-boot Debug View

6.3. Debug Linux kernel code

Before debugging, the Linux kernel code should be compiled on a Linux Host.

Enable and compile the kernel debug info options; this will include the symbol information for the debugger in the 'vmlinux' kernel executable:

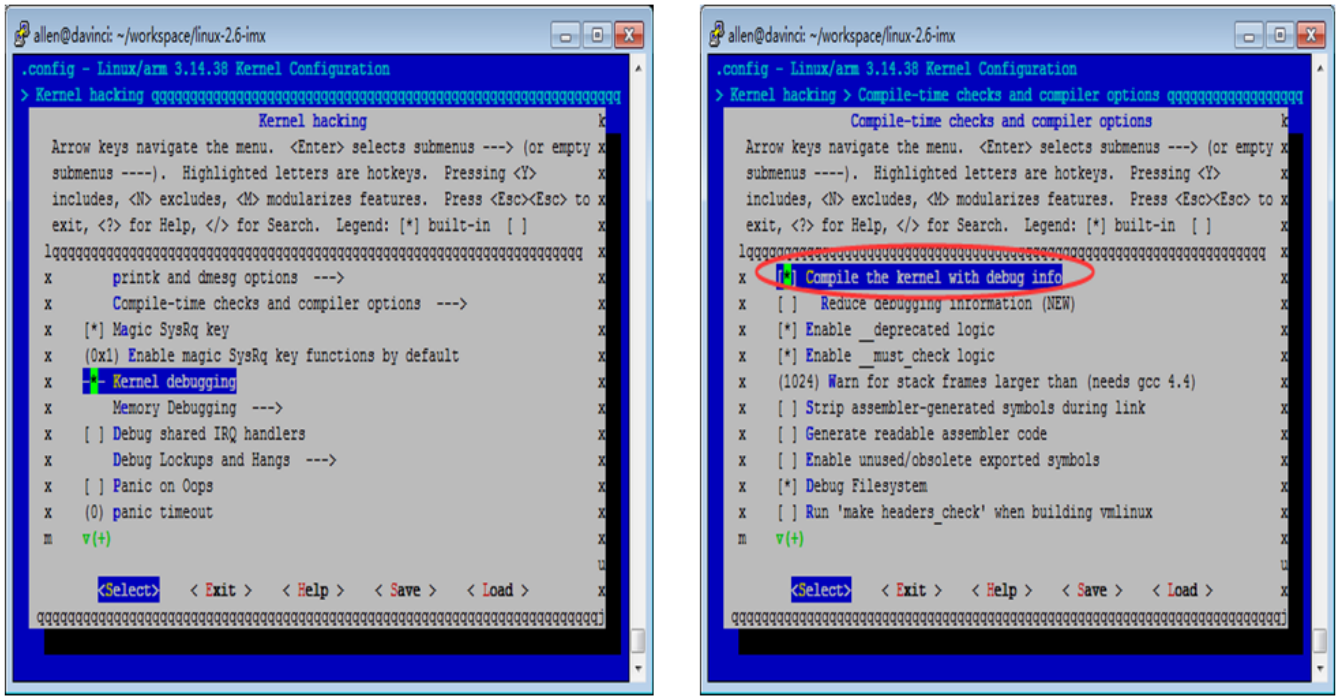

Figure 29. Configure kernel make menu

NOTE:

If your i.MX6UL-EVK board runs Linux kernel version "imx_3.14.38_6ul_ga", you must apply this patch for JTAG: ***0001-For-DS-5-debug-and-uart-console-work-normally.patch***

Then build the kernel on Linux Host.

Setup a boot SD card following the *MX6UL Linux User Guide*.

Connect your i.MX6UL-EVK board Debug UART to a PC, open the serial terminal of the PC, turn on the power switch to boot the MX6UL EVK board, the serial terminal will display the kernel log if setup successfully.

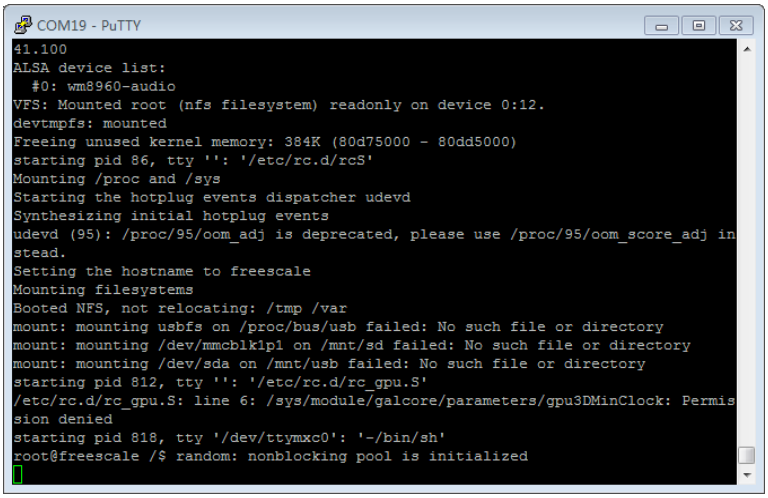

Figure 30. Kernel command prompt line

Configure the DS-5 Debug connection:

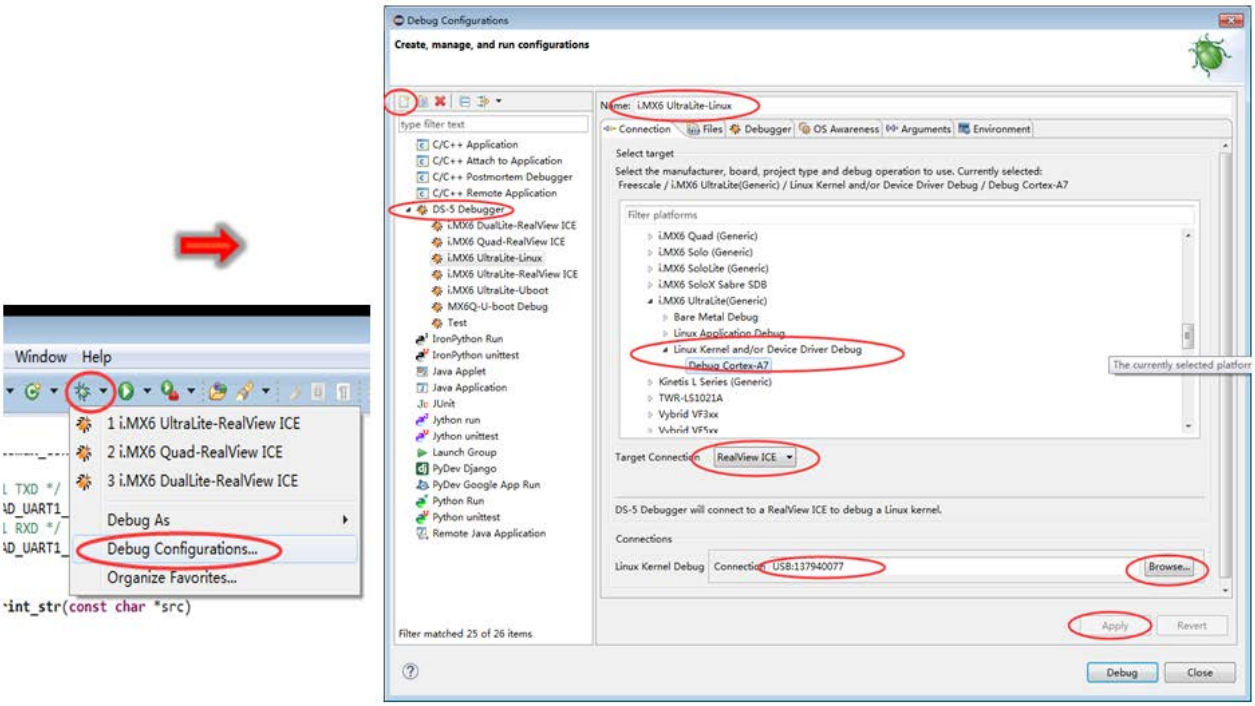

Figure 31. Configure DS-5 Debug Connection

NOTE:

If there is no “i.MX6 UltralLite(Generic)” item in Connection->Select Target, then copy the folder “i.MX6 UltraLite(Generic)” in “mx6ul_evk_ds-5\DS-5-Board-i.MX6 UL\Boards\Freescale” to path “$(ARM DS-5 Install path)\DS-5 v5.22.0\sw\debugger\configdb\Boards\Freescale”.

Configure the DS-5 Debug Debugger Configuration, and start the debug:

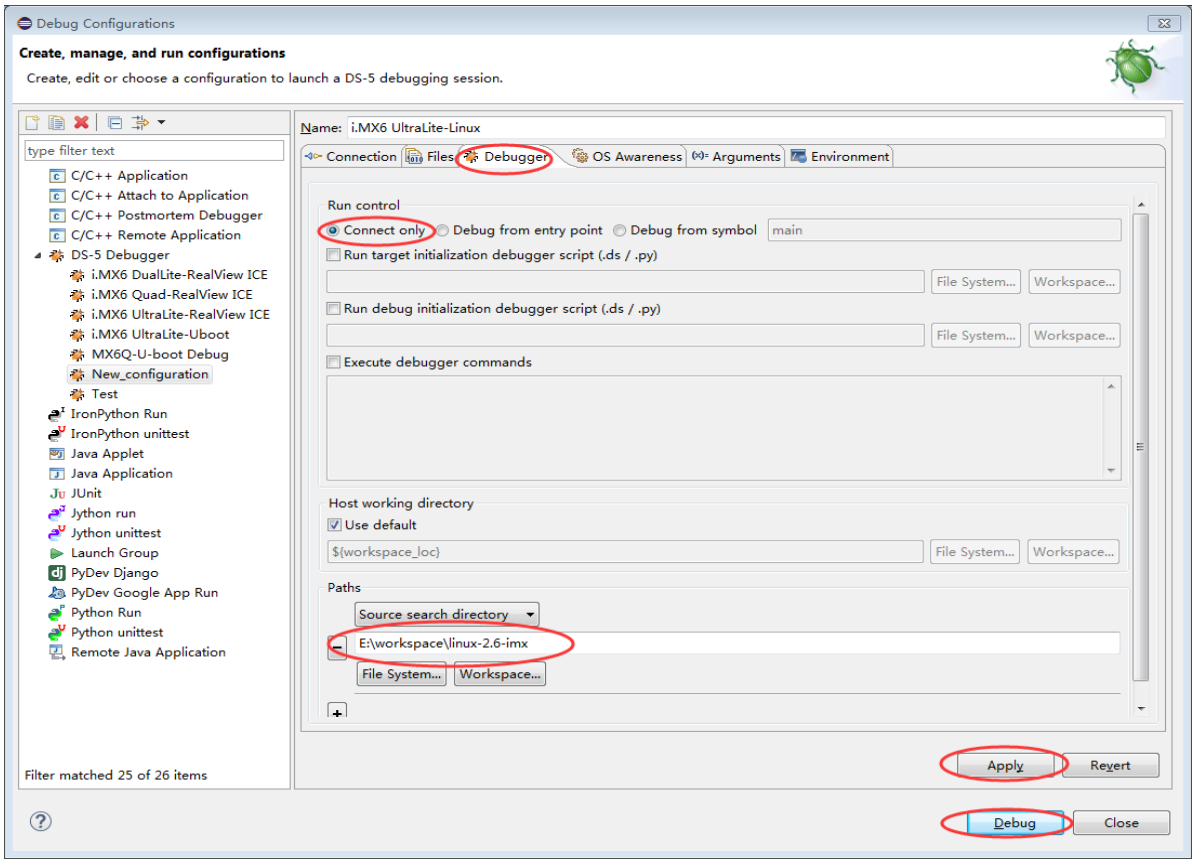

Figure 32. Configure DS-5 Debug Connection

The DS-5 Debug UI will then open, and Try RUN, or Interrupt menu.

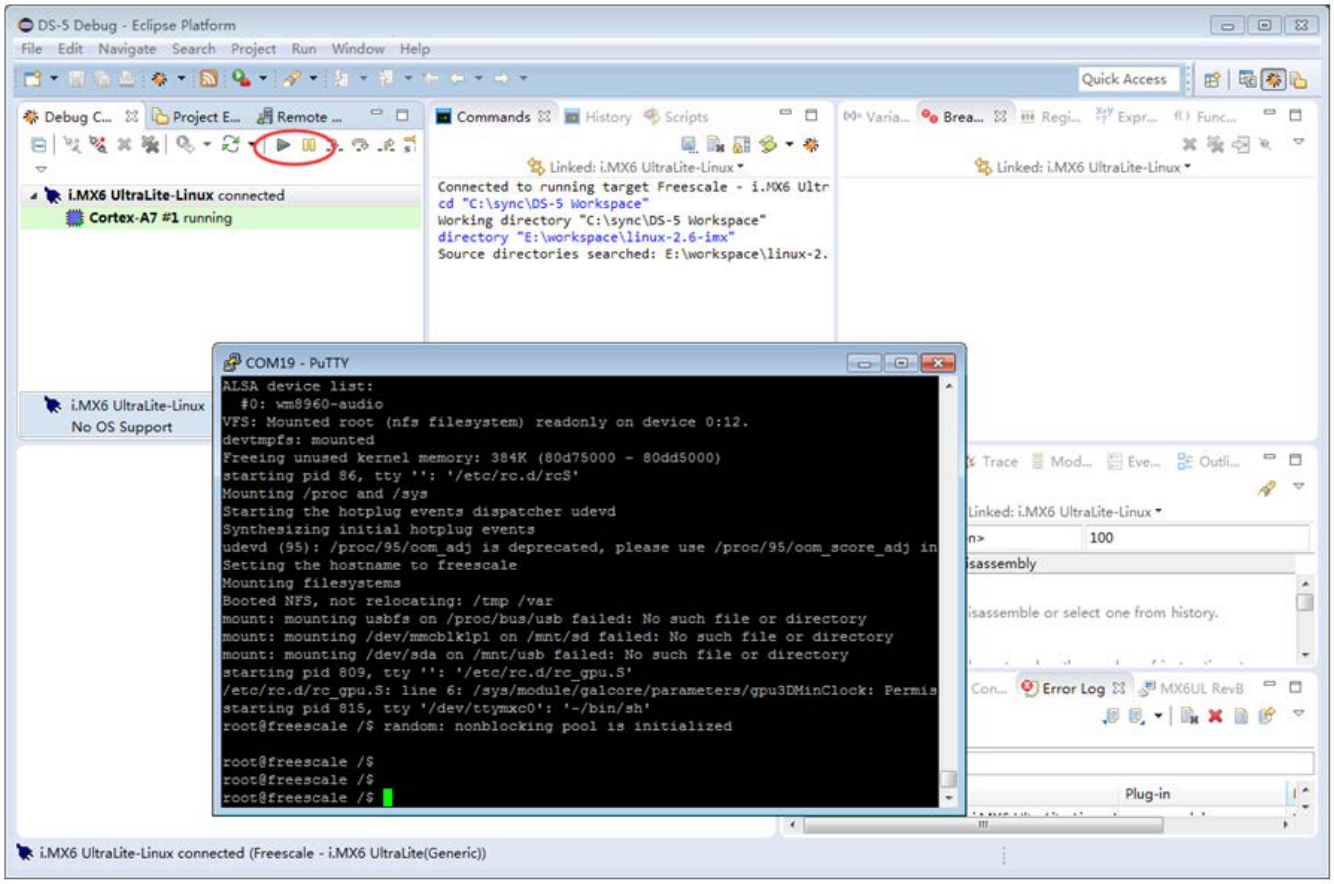

Figure 33. Configure DS-5 Debugger Connected View

Interrupt the Linux and then Load Symbols File:

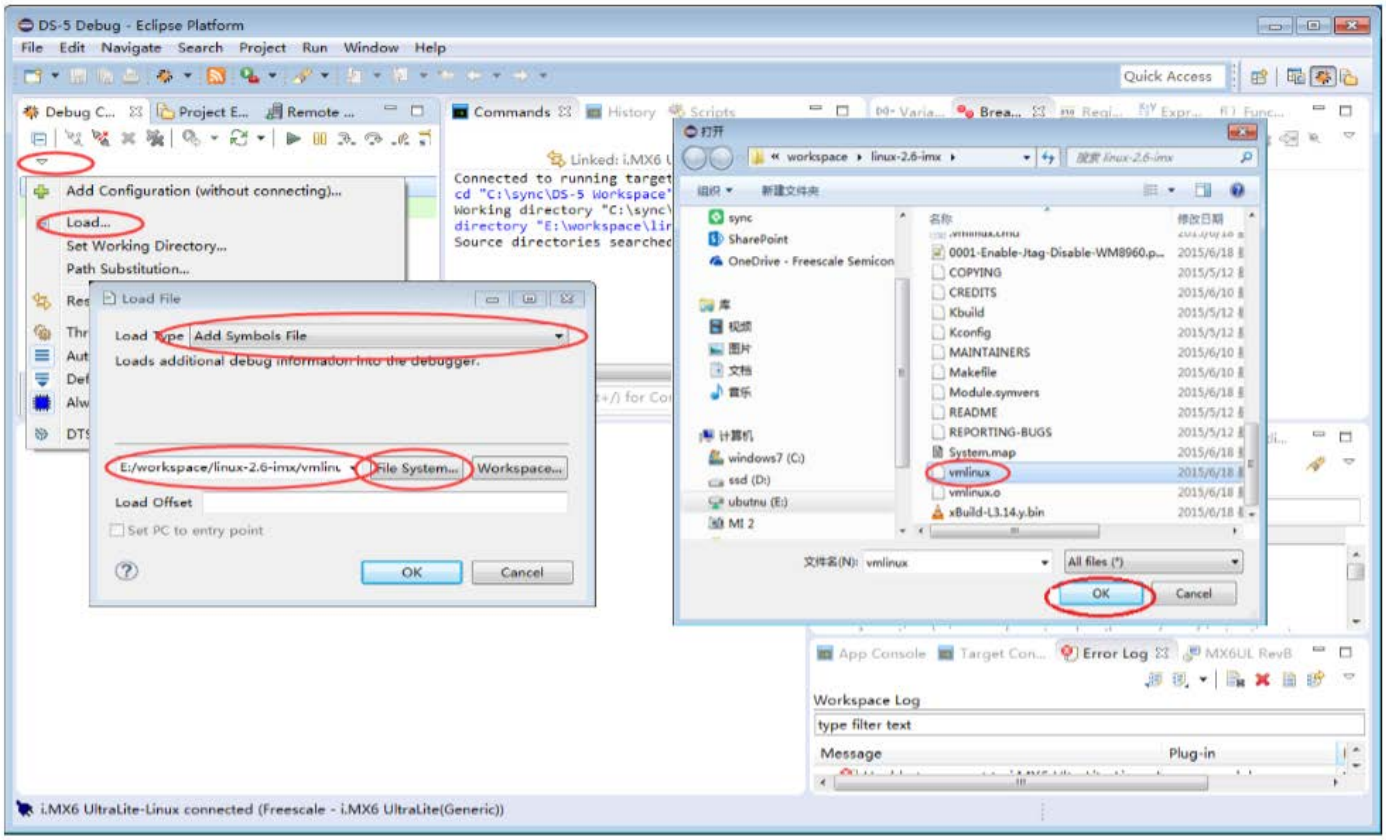

Figure 34. Load Linux kernel symbol file

Run and then interrupt the Linux, you can then see where Cortex-A7 has stopped and the functions in the stack:

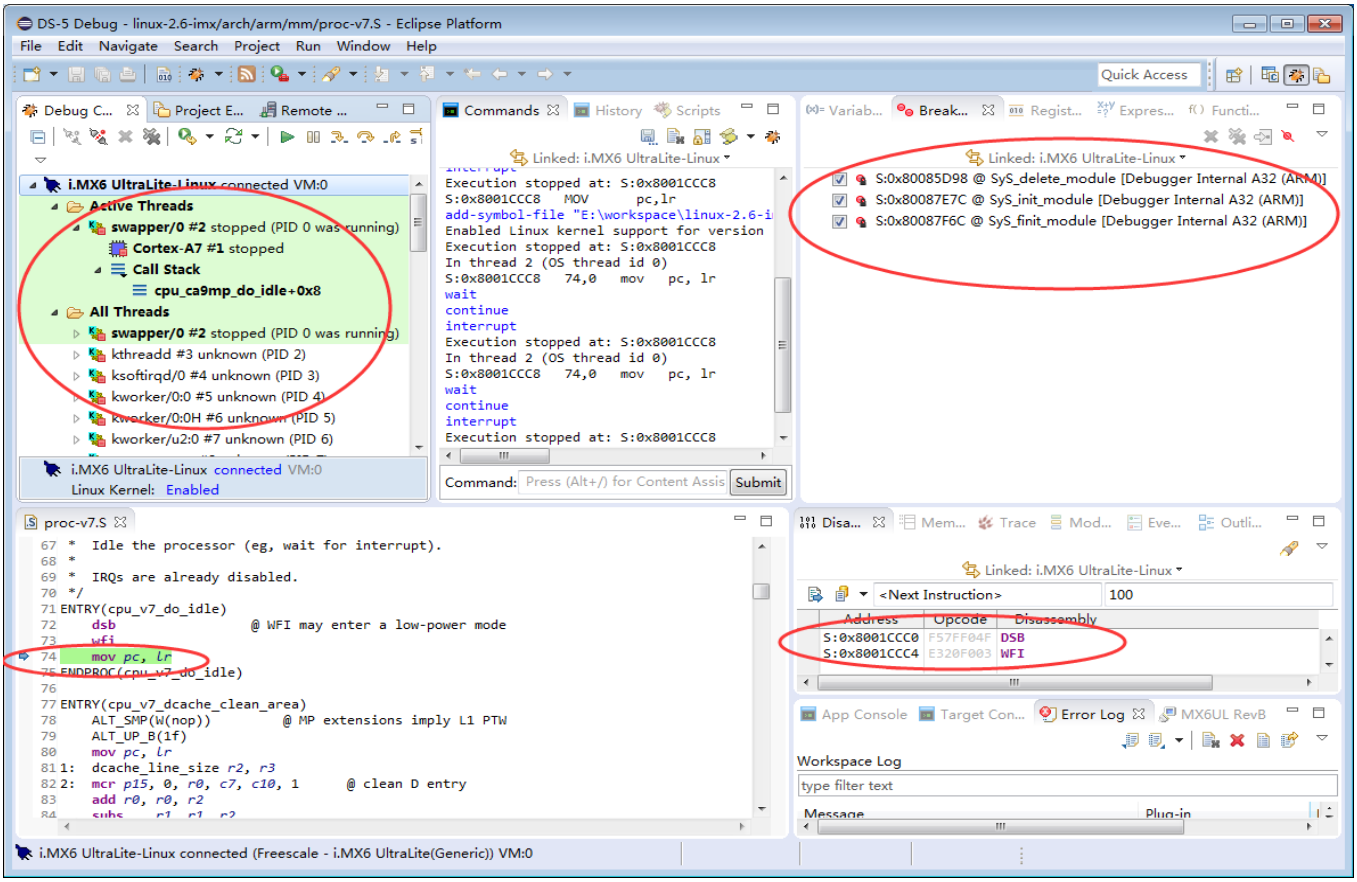

Figure 35. Linux kernel debug view

7. J-LINK Debug

ARM DS-5 IDE supports the J-LINK debugger when Debug Configurations select Zylin Embedded debug(Native) plugin. It based on GDB command between GDB-server and GDB-clinet.

7.1. Install software

Open DS-5, select Help -> Install New Software. To install the zylincdt plugin use the following parameters:

Name: zylincdt
Location: http://opensource.zylin.com/zylincdt

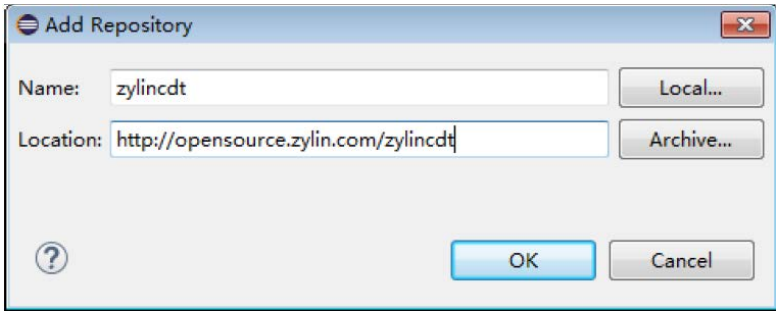

Figure 36. Add zylincdt plugin

Install the J-Link software and documentation pack at https://www.segger.com/jlinksoftware.html

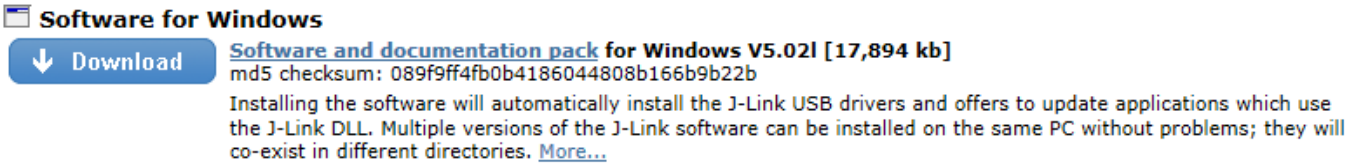

Figure 37. J-Link software

After the installation is complete you will find the J-Link software in the start menu.

Next, install the GDB tools (yagarto-bu-2.23.1_gcc-4.7.2-c-c++_nl-1.20.0_gdb-7.5.1_eabi_20121222.exe) from http://sourceforge.net/projects/yagarto

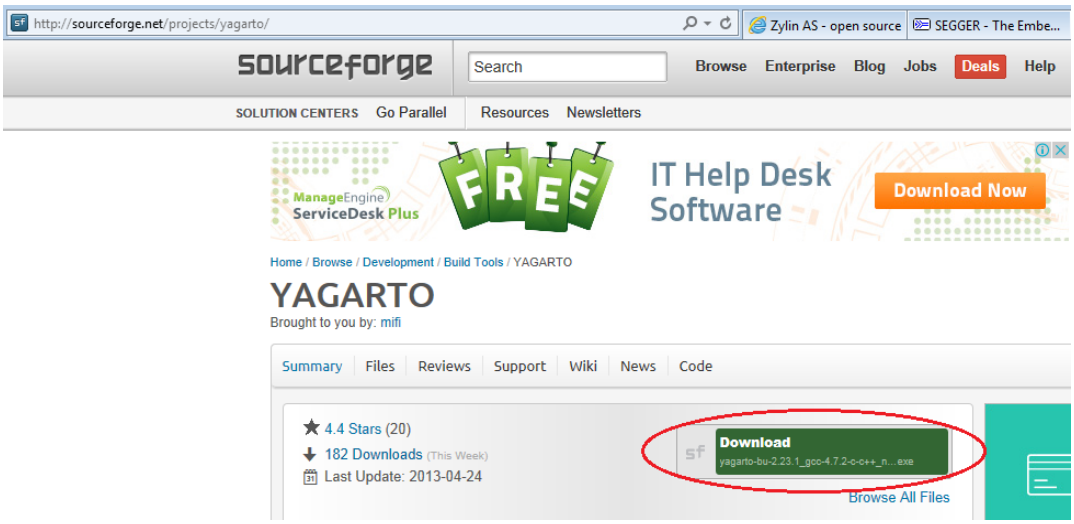

Figure 38. GDB tools Download

7.2. Using J-Link to debug i.MX6UL

Open ARM DS-5, use File -> Import to import the demo code, or you can create a project as in setion 5.2.

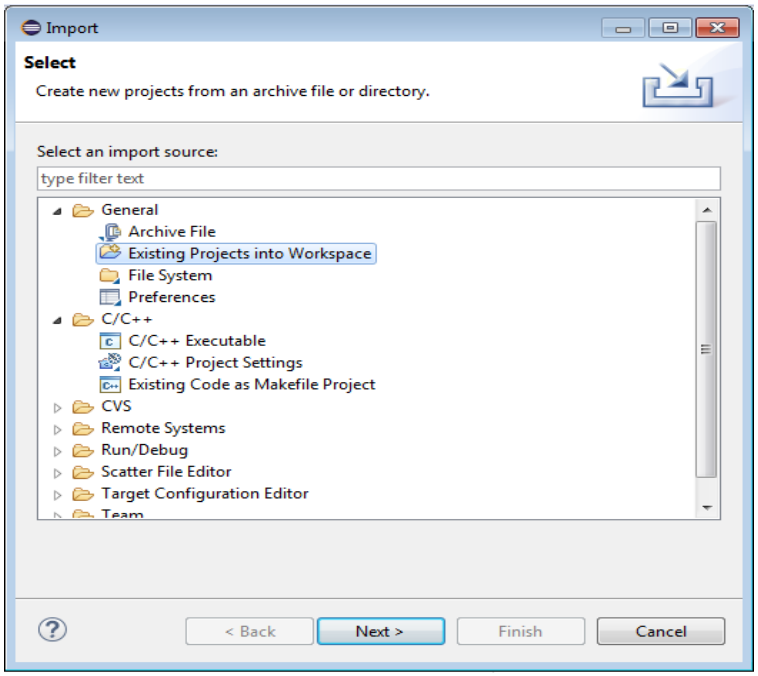

Figure 39. Import the demo project

Configure the Project Properties, select ARM C Compiler 5 Target,

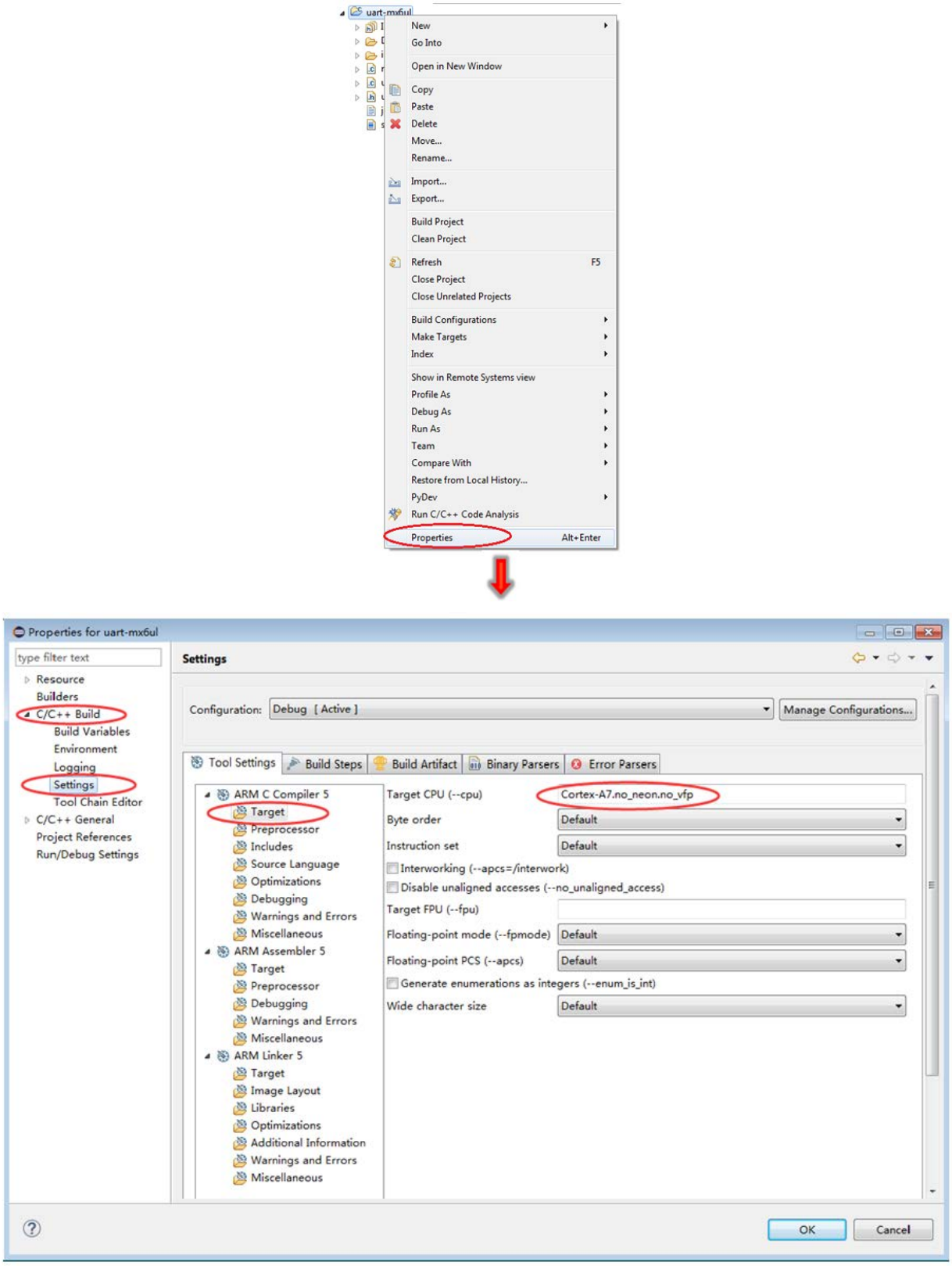

Figure 40. Configure ARM C Compiler 5 target

Configure the ARM Assembler 5 target:

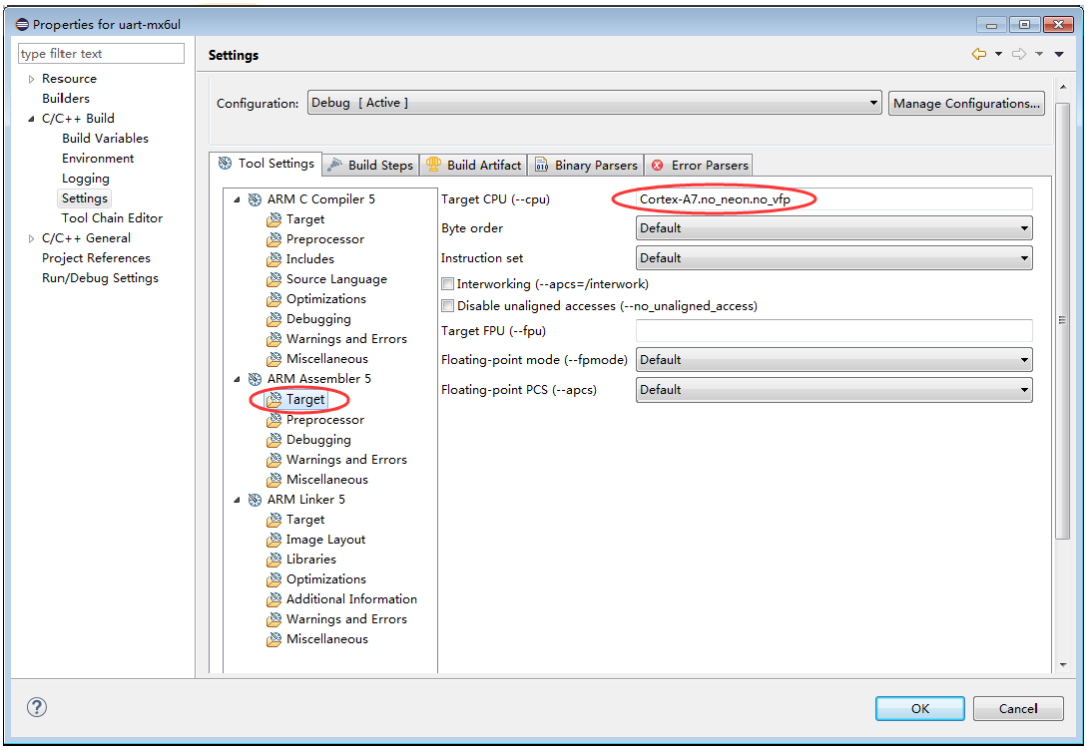

Figure 41. Configure ARM Assembler 5 target

Configure the ARM Linker 5 target:

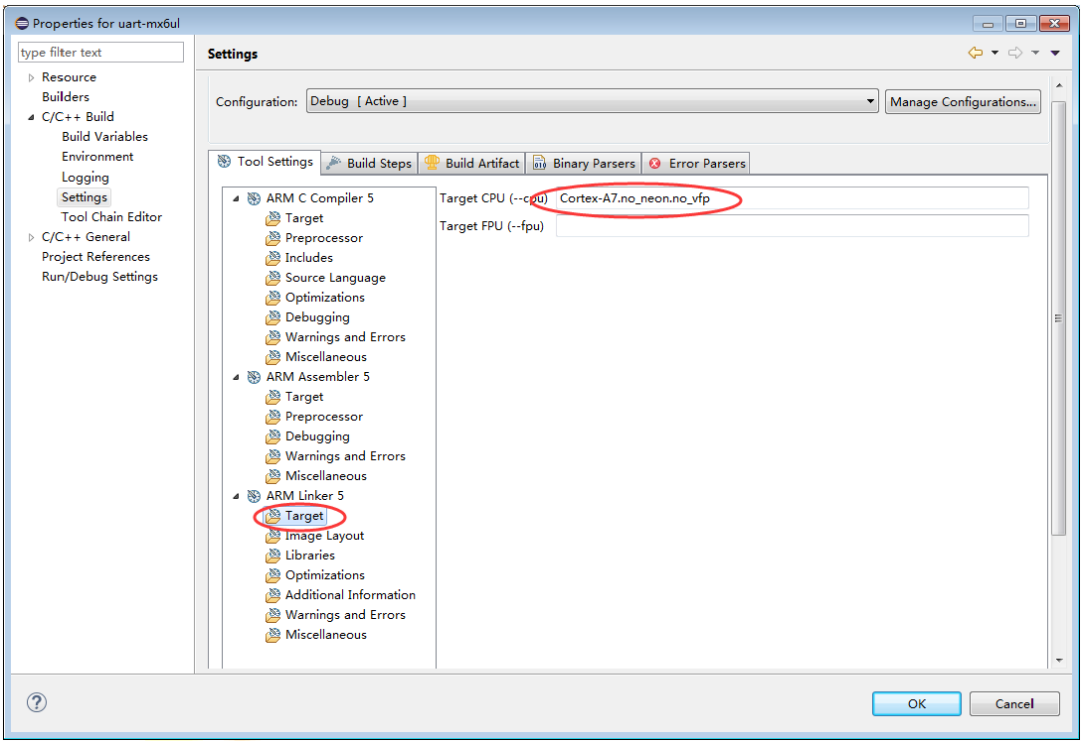

Figure 42. Configure ARM Linker 5 target

Select Image Layout (scatter file) configuration:

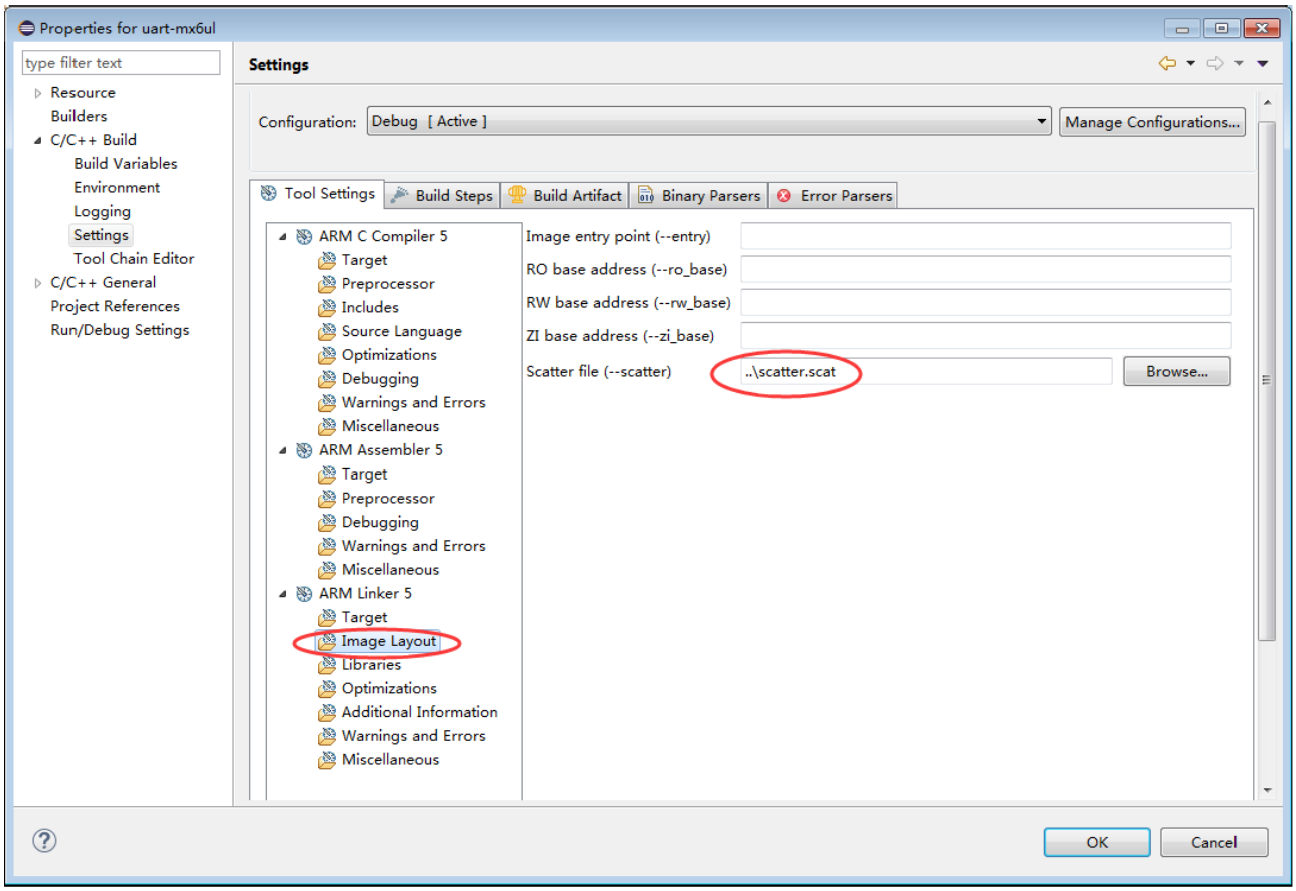

Figure 43. Configure ARM Linker Image Layout

Click the Build button to build the demo code.

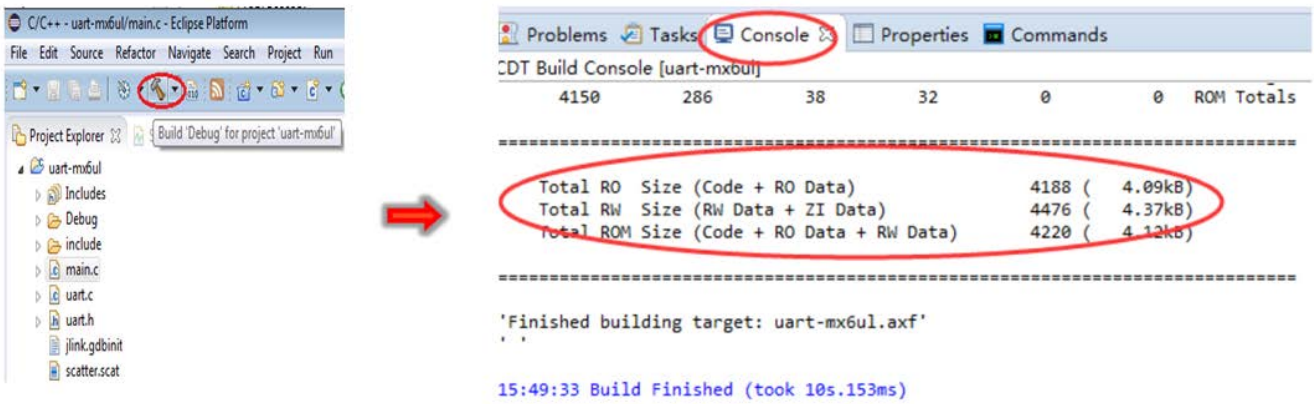

Figure 44. Build the project

DS-5 Debug Connection Configuration and select GDB debugger path installed before:

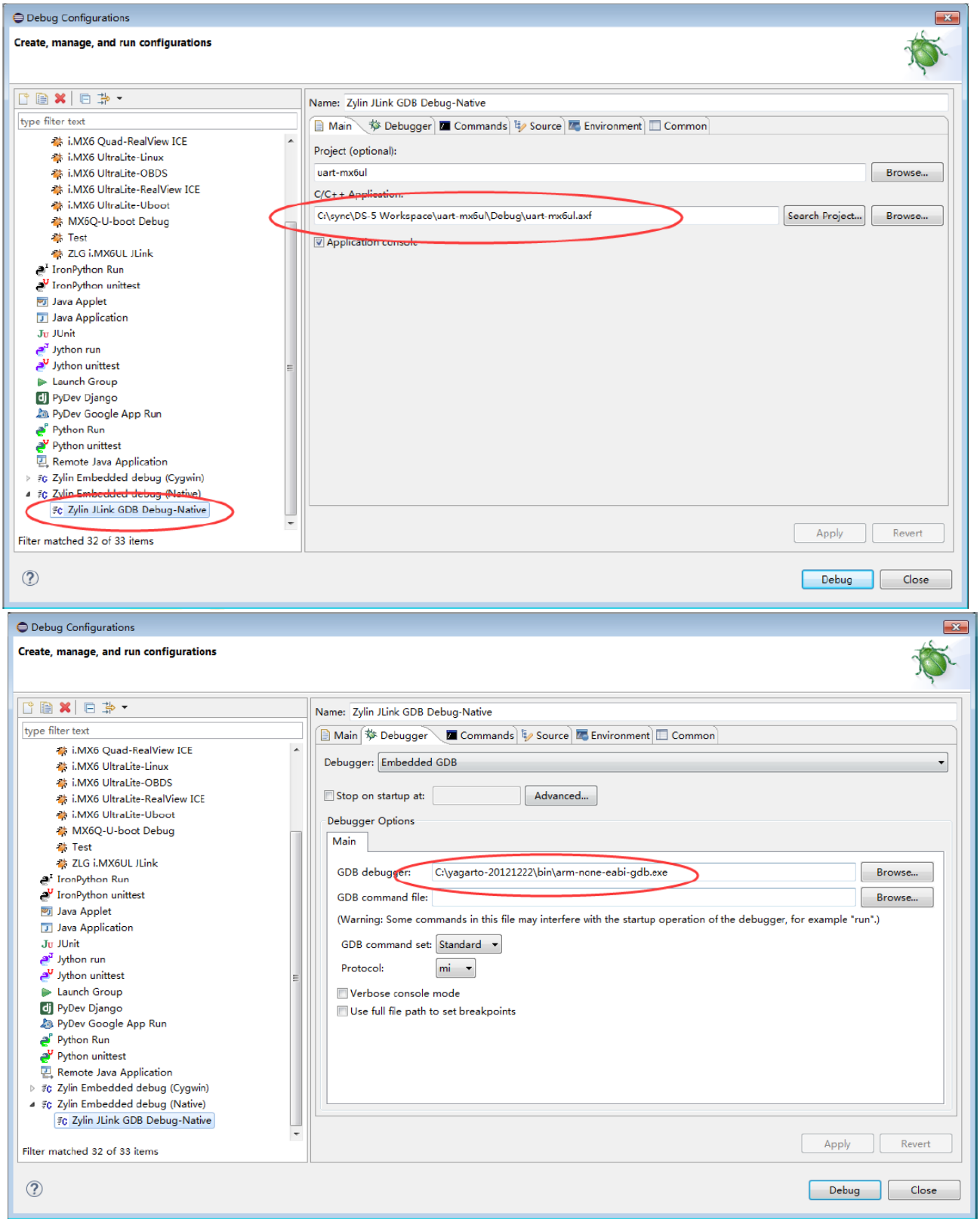

Figure 45. Select GDB debugger path

Input six “Initialize” commands:

- target remote localhost:2331
- monitor reset
- monitor sleep 300
- monitor reg cpsr = 0xd3
- load
- break main

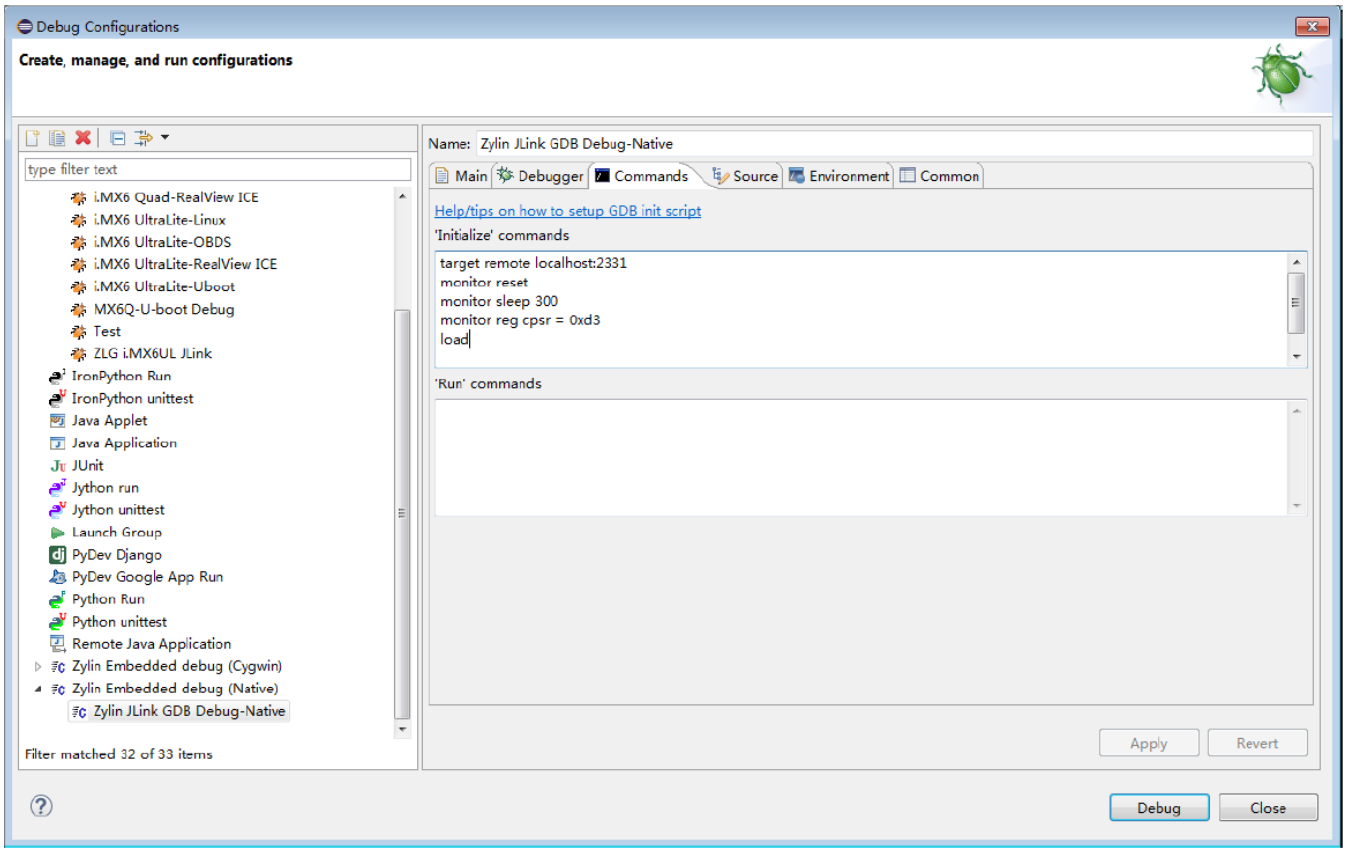

Figure 46. GDB command

Open the Segger J-Link GDB Server:

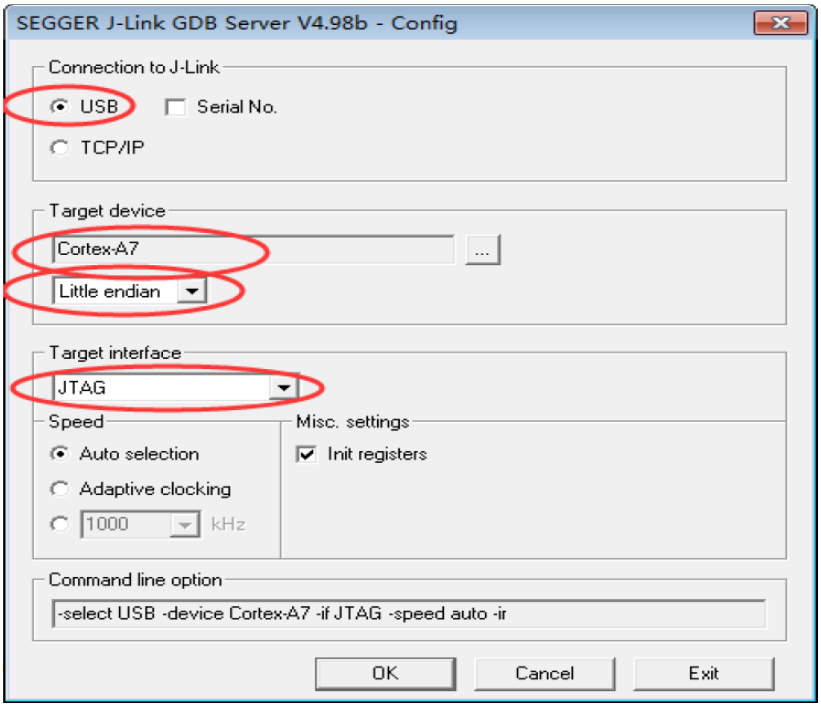

Figure 47. J-Link GDB Server

The following parameters indicate that J-Link has connected with the target board successfully:

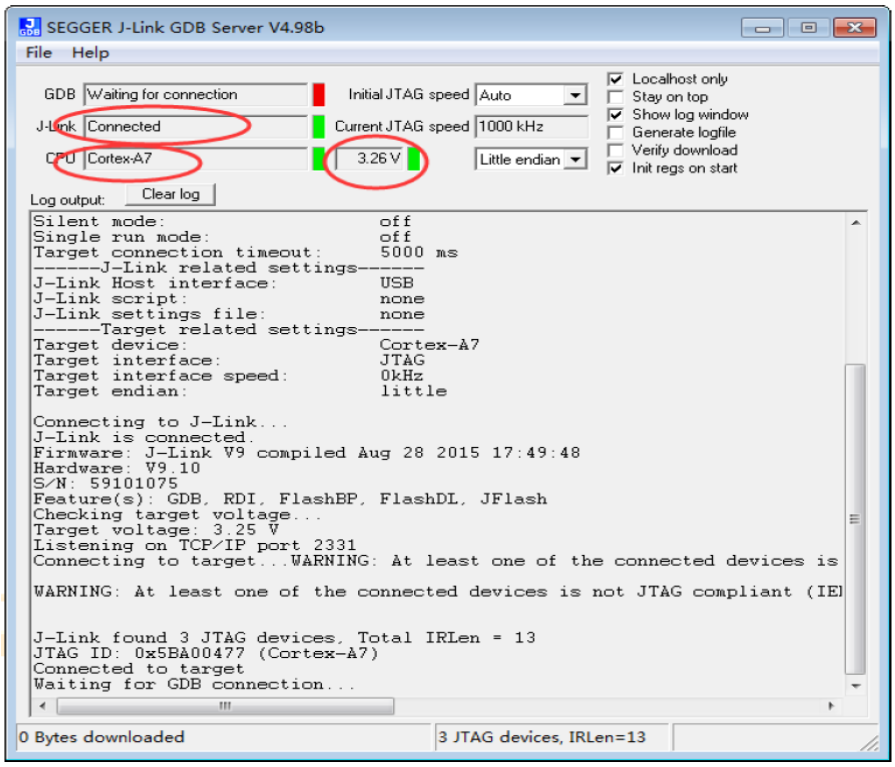

Figure 48. J-Link Connected With Target

Select the Debug button to debug the project:

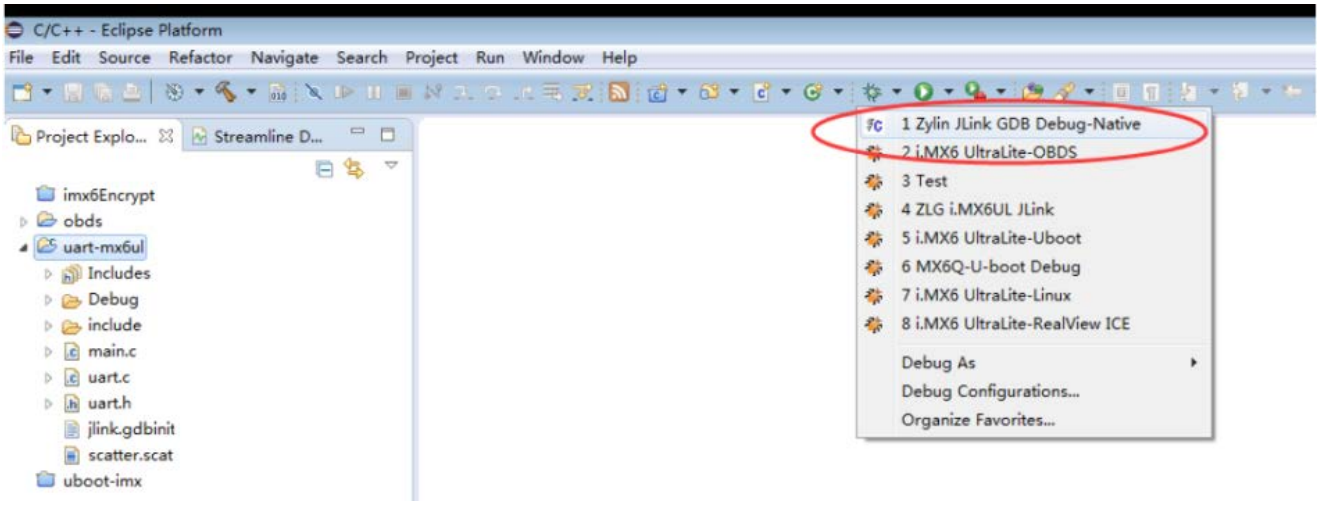

Figure 49. Select J-Link GDB Debug

Stop the Debugger to watch the call stack and some debug information:

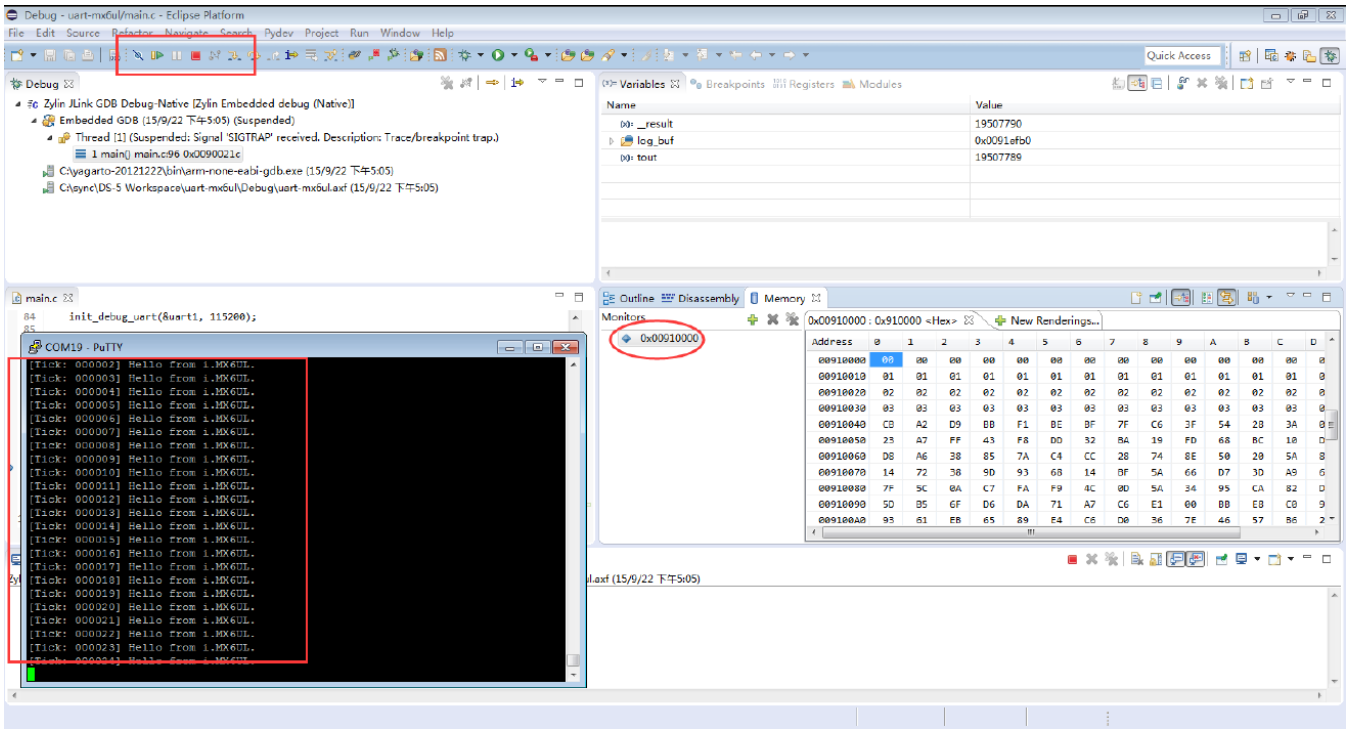

Figure 50. J-Link GDB Debug View

8. Revision history

Table 1. Revision history

Revision number	Date	Substantive changes
0	01/2016	Initial release

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN5229
Rev. 0
01/2016