

J-Link 在基于 AM335x 的 Starterware 开发中的应用

Starterware 是 TI 针对工业客户推出的基于 AM335x 的非操作系统的软件开发包，具有实时性高，灵活性高等特点。随着 AM335x 在业界得到广泛的肯定，目前已经在工业各个领域得到了广泛的应用。J-Link 是市场上很流行的 ARM 仿真器，价格便宜，使用方便。本文主要介绍如何使用 J-Link 在 CCS（Code Composer Studio）和 IAR（IAR Embedded Workbench）集成调试环境中对基于 AM335x 的 Starterware 进行编译调试。

1 Starterware 介绍

Starterware 是一个非操作系统的软件开发包，包含 SOC 启动代码，DDR 内存初始化代码，以及串口，I2C，USB，Ethernet，LCD，中断等驱动。由于没有操作系统复杂平台化功能，所以 Starterware 是一个结构简单的软件包。非常适合于有高实时性，但无并行性需求的应用领域，目前在工控，HMI，工业缝纫机，PLC 等工业方向已经有了很多成功应用。

Starterware 的目录结构及说明可以参考相应的 [wiki](#) 页面，文件夹的布局除了考虑功能模块的划分外，也在板级支持和编译环境两个方面有所考虑，理清了这两点便于理解 Starterware 的架构，说明如下：

- A. 按支持不同开发板划分的目录。在 driver，example，platform 等目录中，子目录 evmAM335x，beaglebone 和 evmskAM335x，分别针对 AM335x 的三个开发板 [GP EVM](#)，[BeagleBone](#) 和 [Starter Kit](#)，存放着相应的板级配置代码。
- B. 按不同编译环境划分的目录。Starterware 把所有编译相关的工程文件放在了 build/armv7a 目录中，其子目录 cgt_ccs，ewarm 和 gcc，存放分别对应 CCS，IAR 和 GCC 编译环境的工程配置文件。

从软件结构来看，Starterware 有两部分组成，Bootloader 和 Application(应用程序)，具体介绍如下：

1.1 BootLoader

BootLoader 被 ROM code 加载到片上 SRAM 中。AM335x 启动后，首先运行 ROM code，ROM code 根据 sys_boot（具体可以参考 [AM335x technical reference manual](#) 中的 Initialization 章节）的配置，从相应的存储器或者外设中得到 Bootloader，并加载到片上的 SRAM 的起始地址处，即 0x402F0400。

BootLoader 中完成如下工作：

- A. ARM core 的配置，包括中断向量表，Cache，MMU 等配置。
- B. PRCM 模块（具体可以参考 [AM335x technical reference manual](#) 中的 PRCM 章节）的配置，主要是对各 PLL，power management 等的配置。

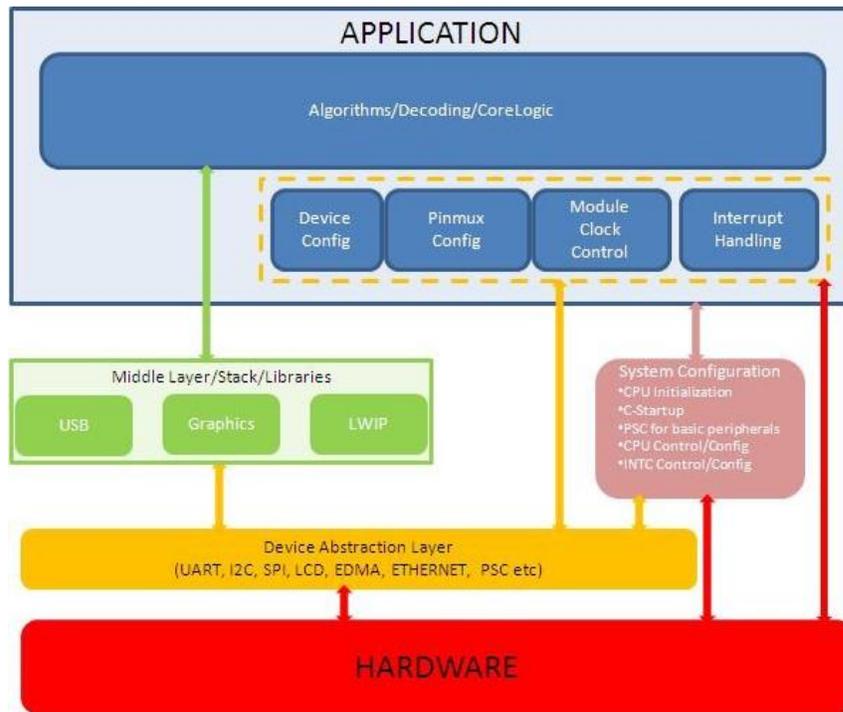
- C. I2C, PMIC 的配置。如果使用了推荐的 PMIC, 首先对 I2C 模块配置, 然后通过 I2C 配置 PMIC。
- D. UART 配置, 主要用于输出调试信息。
- E. DDR 配置。
- F. 根据配置, 会初始化相应的外设或者存储器。
- G. 完成了上述配置后, BootLoader 会相应的外设或存储器中读取 Application, 并加载到 DDR 中。

上述工作过程和 SPL 很接近。但在 Starterware 中, BootLoader 没有和 u-boot 一样复杂的外设工具支持, 编译完成后只有 34K Bytes, 可以放在片上 SRAM 上, 所以只要一二级 boot loader 就可以完成需要的任务了。

1.2 Application

Application 是应用的主体, 实现应用的逻辑, 并调用相应的外设驱动。

Starterware 中在 example 目录。其软件架构如下:



该架构很清晰, 上层应用调用下层驱动或者中间件, 以实现逻辑对底层模块的调用。这样的结构, 在生成可执行文件时, 只是链接需要的库文件, 使得文件比较小, 便于 JTAG 下载调试, 同时, 清晰的结构十分便于调试, 查找问题所在。

Application 中仅对模块的时钟进行配置, 而 PRCM (参加 [technical reference manual](#)) 中相关 PLL, 基本的电源管理功能等是在 BootLoader 或者编译环境的脚本 (后面有详细介绍) 中配置的, 所以, Application 需要在这两者准备好的环境中运行。

2 J-Link 介绍

J-Link 是 SEGGER 公司推出的 JTAG 仿真器，在 ARM 芯片的客户中很流行。可支持 ARM9, Cortex-A8 等主流的 ARM 核，同时也支持 CCS, IAR 等集成开发环境。

目前，J-Link 已经提供了对 AM335x 的支持，由于 AM335x 提供的是 TI 的 [TI JTAG](#) 接口，和 ARM 的 JTAG 接口有区别，所以需要从硬件接口和软件 firmware 上进行升级才可以让 J-Link 和 AM335x 连上，在后面章节有具体介绍。

J-Link 对 AM335x 的调试和 TI 自己的仿真器对 AM335x 的调试相比，区别主要在于调试环境的搭建和 ARM core 的连接，下面会具体介绍。

3 调试环境的准备

3.1 J-Link

如前面提到的，J-Link 默认的接口是 20-Pin ARM JTAG，而 AM335x [GP EVM](#) 所带的接口是 [CTI JTAG](#)，所以需从硬件接口和 firmware 两个方面进行配置，以实现两个接口的兼容：

A. 硬件接口配置

AM335x 可以支持 14-pin 和 20-pin TI JTAG，同时，J-Link 也可以支持 14-pin 和 20-pin ARM JTAG。这 4 种 JTAG 接口的定义如下：

Pin	TI 14-Pin 0.10" 2 row Molex Target card connector: SAM-TSM-17-DV	Compact TI 20-Pin (CTI) Samtec 0.05" 2 row Molex Target card connector: FTR-110-51-S-D-06 Emulator connector: RSM-110-02-S-D	ARM 20-Pin 0.10" 2 row Molex Target card connector: SAM-TSM-110-DV Emulator connector: SSW-110-22-G-D-VS	ARM 14-Pin 0.10" 2 row Molex (Not recommended)
1	TMS	TMS	VTRef	VTRef
2	TRSTn	TRSTn	VSupply	GND
3	TDI	TDI	TRST	TRST
4	TDIS	TDIS	GND	GND
5	VTRef	VTRef	TDI	TDI
6	KEY	KEY	GND	GND
7	TDO	TDO	TMS	TMS
8	GND	GND	GND	GND
9	RTCK	RTCK	TCK	TCK
10	GND	GND	GND	GND
11	TCK	TCK	RTCK	TDO
12	GND	GND	GND	SRST
13	EMU0	EMU0	TDO	VTRef
14	EMU1	EMU1	GND	GND
15		SYSRST#	SRST	
16		GND	GND	
17		EMU2	DBGRRQ	
18		EMU3	GND	
19		EMU4	DBGACK	
20		GND	GND	

从上述表格可见：

- a. ARM JTAG 中，14-pin 接口和 20-pin 接口的区别主要是，20-pin 接口带有 SRST 接口，可以用于通过 JTAG 接口对 SOC 进行 reset。ARM JTAG 的详细文档参考[这里](#)。
- b. TI JTAG 中，14-pin 接口和 20-pin 接口的区别除了 SYSRST (SRST) 接口外，还有 EMU2~EMU4 的区别，这四个 pin 脚是为了 TI 自己的高速 debug 接口而预留的，暂时没有使用。TI JTAG 的详细文档参考[这里](#)。
- c. TI JTAG 比 ARM JTAG 多的几个 pin 脚如下：
 - EMU0~EMU4
EMU0 和 EMU1 对于 J-Link 可以不用连接，具体介绍可以参考[这里](#)。
 - KEY
这个脚悬空，不需要使用。
 - TDIS
这个脚和地相连即可。从 TI JTAG 到 ARM JTAG 转接电路设计可参考[这里](#)。

B. Firmware 的升级

对于已有的 J-Link 需要进行 firmware 的升级以支持 AM335x。首先，到 SEGGER 的[官方地址](#)可以下载最新的 firmware，并安装到 PC 机上，然后将 J-Link 插到 PC 上，运行  升级 firmware。如果升级成功，再次运行该程序，windows 状态栏中会出现 。

3.2 集成调试环境

TI 的 CCS，除了支持 TI 专有的 [XDS560v2](#)，[XDS100v2](#) 等仿真器外，现在也能支持 J-Link。第三方集成调试开发环境 IAR，最新版本也开始提供对 AM335x 平台的支持，相应的仿真器包括 J-Link 和 XDS100v2。本文将分别介绍如何配置 CCS 和 IAR 配合 J-Link 对基于 AM335x 的 Starterware 平台进行调试。

3.2.1 CCS

CCS 5.2.0 或者以上版本可以支持 J-Link，最新版本的 CCS 已经经过验证支持 J-Link。CCS 有 Windows 版本和 Linux 版本，这里介绍的是基于 Windows 版本 5.3.0 的调试。

CCS 针对 J-Link 的插件不在 CCS 安装包中，需要单独安装，安装文件可以在[这里](#)下载，安装路径可以选择 CCS 的安装路径，默认为 C:\ti\ccsv。

3.2.2 IAR

IAR 6.50 及以上版本可以支持 AM335x，可以到 [IAR 官方网站](#) 下载最新的版本。本文调试过程中提到的 IAR 是指 IAR 6.5.0。

3.3 开发板

目前，针对不同的应用，TI 发布了基于 AM335x 的多个开发板，其中，[BeagleBone](#)，[Starter Kit](#) 和 [ICE](#) 上配置了基于 FT2232 的 XDS100v2，[GP EVM](#) 和 [IDK](#) 上设计了 [CTI JTAG](#) 接口。这里以 GP EVM 为平台介绍调试过程。这里，从 CTI JTAG 接口到 ARM JTAG 接口用到了一个转接板，其原理图可以参考[这里](#)。

4 Starterware 代码的准备

4.1 下载代码

在 [TI wiki](#) 中下载最新的 Starterware 源代码。

4.2 Starterware 的编译

Startware 的编译可以分为导入代码工程，选择编译模式，编译工程三个步骤。下面将以 uartEco 为例，按这三个步骤逐一介绍编译过程，并在每个步骤中分别介绍 CCS 和 IAR 的使用。

4.2.1 导入代码工程

如前面所介绍，编译相关的工程文件都在 build/armv7a 中，再根据不同的编译环境选择相应的子目录：

A. 在 CCS 中导入工程 uartEco

- a. 打开 CCS Edit 的界面。菜单 Menu -> Window -> Open Perspective -> Other。
- b. 导入工程。

File->import, 进入该界面后选择 Code Composer Studio -> Existing CCS/CCE Eclipse Project。

在导入工程界面中选择 uartEco 所在的目录：

`C:\ti\AM335X_StarterWare_02_00_00_07\build\armv7a\cgt_ccs\am335x\evmAM335x\uart`

即可导入 uartEco 工程。

B. 在 IAR 中导入工程

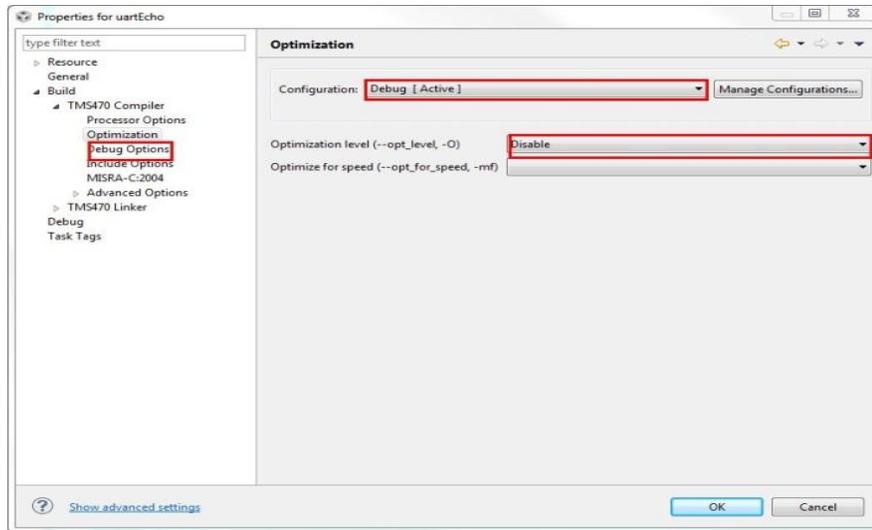
菜单 project->Add Existing Project to Current Workspace, 选择 uartEco IAR 工程所在目录即可。 `C:\ti\AM335X_StarterWare_02_00_00_07\build\armv7a\ewarm\am335x\evmAM335x\uart`

4.2.2 选择编译模式

Starterware 对 CCS 和 IAR 都提供了 Debug 和 Release 两种模式。Debug 模式是为了便于调试而设置的，在该配置中，编译器的优化选项被关闭，因为优化选项打开时，编译器生成的汇编代码其顺序和 C 代码不一致；此外，加入 Debug 选项，使得生成的 image 中带有 symbol 等调试信息。Release 则相反，为了使得最后发布的代码达到最佳的性能文件大小比，加入了最佳的优化选项，并去掉了调试选项，确保生成较小的文件。本文中，主要介绍如何在 CCS 和 IAR 下调试 Starterware，所以需要选择 Debug 模式，下面将分别介绍如何在 CCS 和 IAR 中选择 Debug 模式：

A. 在 CCS 中选择 Debug 模式

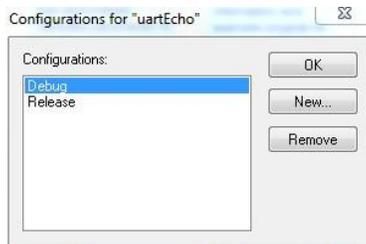
在菜单中，Project->build Configurations->Set Active-> Debug, 可以使能 Debug 模式。另外，可以右键点击 uart 工程，选择 Properties, 然后选择 Debug options 标签，可以得到如下界面：



在红色矩形框内的内容可以看到优化选项都已关闭，还可在此界面下查看其它编译选项。

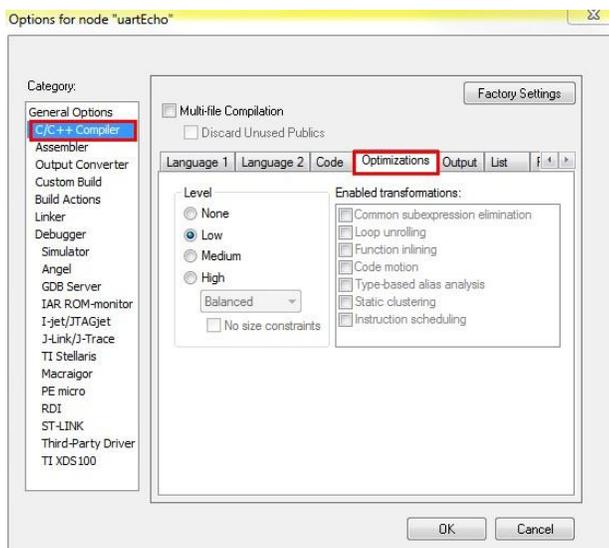
B. 在 IAR 中选择 Debug 模式

在菜单中，Project->Edit Configurations，可以得到如下界面：



在该界面中选择 Debug 即可。

类似 CCS，同样右键选择 uart 工程，菜单中选择 Options，可以得到优化选项的相关内容，可以参考下图：



4.2.3 编译工程

在 CCS 和 IAR 中都可以在 Project 菜单中选择 build all 来编译 uartEco 工程。这里的 uartEco 工程是应用程序，其调用的 driver, platform, utilities 等库不在此处编译。

对于 CCS 而言，这些库在 C:\ti\AM335X_StarterWare_02_00_00_07\binary\armv7a\cgt_ccs 中已经有预编译的库，但为了调试，建议在 Debug 配置下，重新编译这些库。

对于 IAR，这些库没有预编译，所以需要先编译这些库，再编译 uart 工程。需要编译的库如下：

```
C:\ti\AM335X_StarterWare_02_00_00_07\build\armv7a\ewarm\utils  
C:\ti\AM335X_StarterWare_02_00_00_07\build\armv7a\ewarm\mmcsdlib  
C:\ti\AM335X_StarterWare_02_00_00_07\build\armv7a\ewarm\nandlib  
C:\ti\AM335X_StarterWare_02_00_00_07\build\armv7a\ewarm\am335x\drivers  
C:\ti\AM335X_StarterWare_02_00_00_07\build\armv7a\ewarm\am335x\system_config  
C:\ti\AM335X_StarterWare_02_00_00_07\build\armv7a\ewarm\am335x\usblib  
C:\ti\AM335X_StarterWare_02_00_00_07\build\armv7a\ewarm\am335x\evmAM335x\platform
```

编译完成后，生成的库文件在下列目录的相关子目录中

```
C:\ti\AM335X_StarterWare_02_00_00_07\binary\armv7a\ewarm
```

可执行文件在下列目录中，

```
C:\ti\AM335X_StarterWare_02_00_00_07\binary\armv7a\cgt_ccs\am335x\evmAM335x\uart
```

调试所需要的是.out 文件，可以通过 CCS 下载到 EVM 板上运行。直接在板上的 SD card, NandFlash, UART 等启动设备上使用的是 _ti.bin 文件，该类型文件是通过 TI 提供的工具从.out 文件转化而来的，具体可以参考 [wiki 页面](#)。

至此，用 J-Link 调试基于 AM335x 的 Starterware 的准备工作都已经完成了。

5 Starterware 的调试

调试的过程主要分为导入代码工程，CCS 连接 AM335x，代码调试等几个部分，下面将详细介绍在 CCS 和 IAR 中如何通过 J-Link 调试 Starterware。

5.1 导入代码

在编译代码的过程中，已经将代码工程导入到了 CCS 或者 IAR 中，这里不再重复。导入代码的作用是 Debugger 通过 image 中的 symbol 查找源码时，可以通过导入的工程，直接找到并显示源码，便于通过源码调试。

5.2 连接 AM335x

主要分成仿真器的连接，ARM core 连接两部分。

5.1.1 仿真器的连接

将 J-Link 仿真器通过 [转接板](#) 连接到 GP EVM 的 baseboard 的 J2 口即可。注意 JTAG 的口的 pin 脚顺序。接下来是对 CCS 和 IAR 做配置，实现 CCS/IAR 和 J-Link 之间的软件连接，包括 Target 的配置和连接两个步骤：

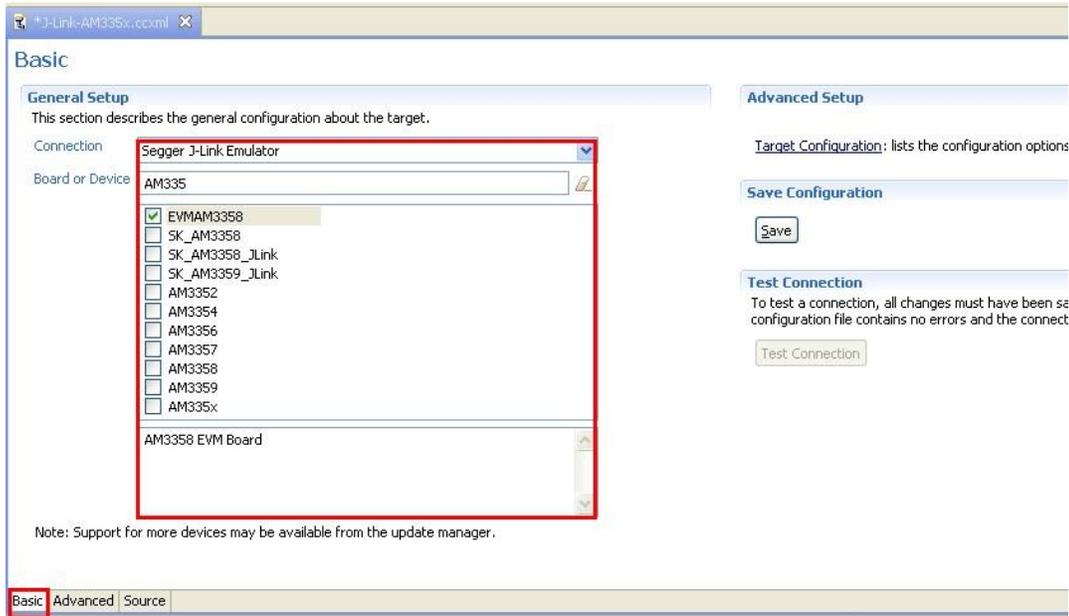
A. Target 配置

a. CCS 中的配置

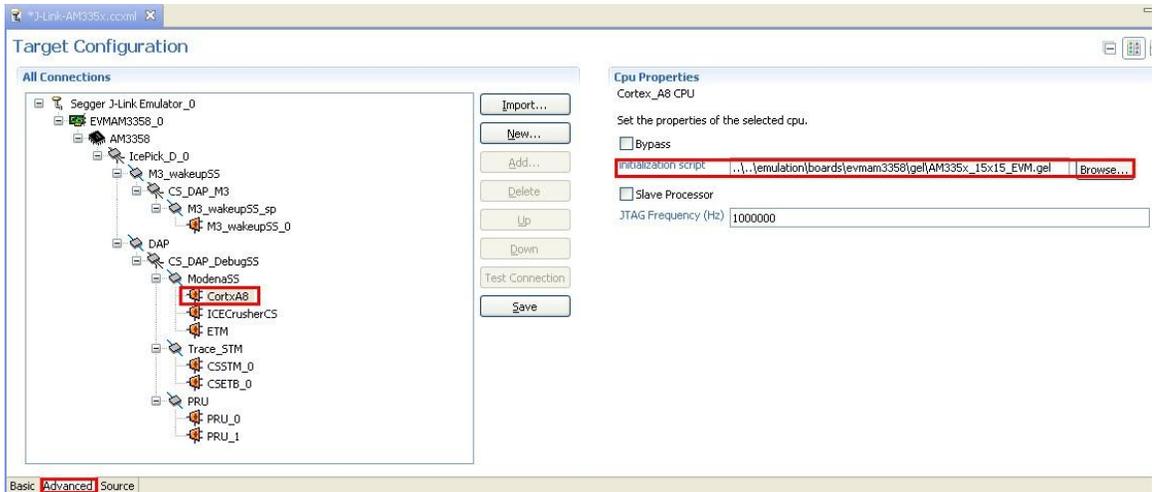
Target 的配置包含两个部分，一个是仿真器（J-Link），另一个就是 SOC（AM3358）。具体操作如下：

- View -> Target Configurations。
- 点右键选择 New Target Configuration。
- 新建一个叫做 AM335_EVM 的 target。
- Connection 中选择 Segger J-Link Emulator。
- 在 Board or Device 中选择 EVMAM3358。
- 点击 Save 保存。
- 点击 Test Connection 看是否能够正常连接。

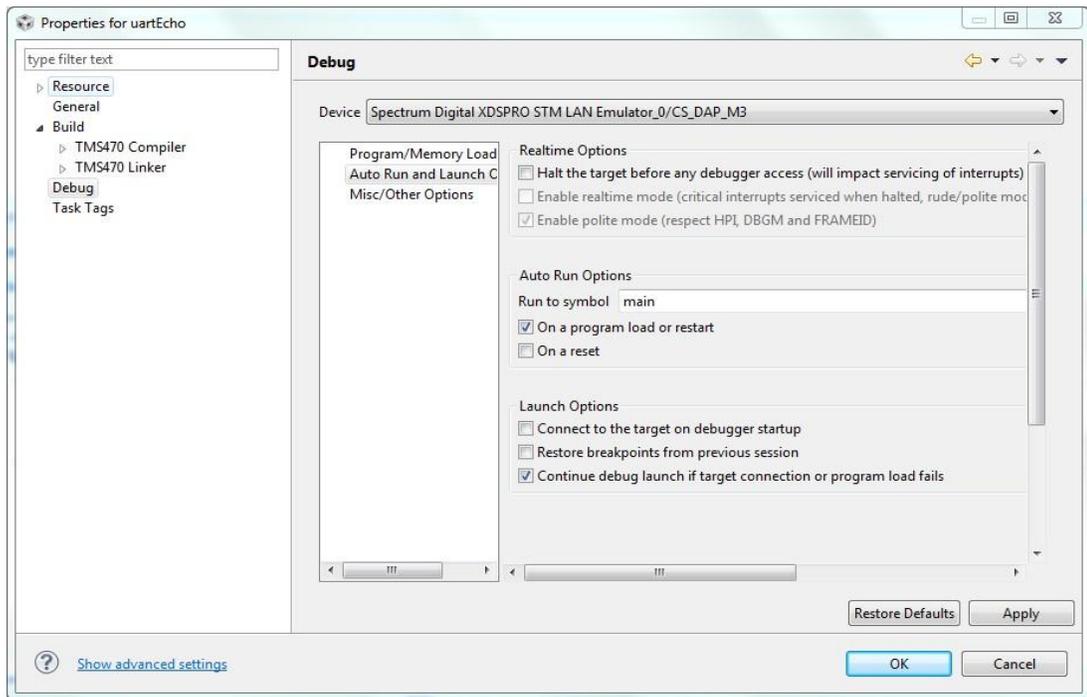
配置 target 成功后可以看到如下界面



这里 EVMAM3358 和 AM3358 的区别是 EVMAM3358 中包含了 GEL 文件，并且该 GEL 文件会在连接 ARM core 时被自动加载。GEL 是 General Extension Language 的缩写，是一种 CCS 所专有的一种类 C 的脚本语言，该语言可以通过 CCS 对 JTAG 进行控制，通常用于初始化内存，时钟等，具体可以参考[这里](#)。点击上述界面的 Advanced 标签，然后点击 CortexA8 可以得到下面的界面，里面有 EVMAM3358 所包含的 GEL 文件的路径，该文件在 CCS 连上 Cortex-A8 core 时会自动运行（也可以手动运行，可以参考[这里](#)）用于初始化 AM3358 的时钟系统以及 [GP EVM](#) 上的 DDR2 等，即是初始化了基于 AM3358 的最小系统。客户移植 Starterware 到产品的电路板时，可以先从移植 GEL 文件开始，这样便于调试。当软件稳定后，需要将 GEL 文件的内容移植到 Bootloader 中。



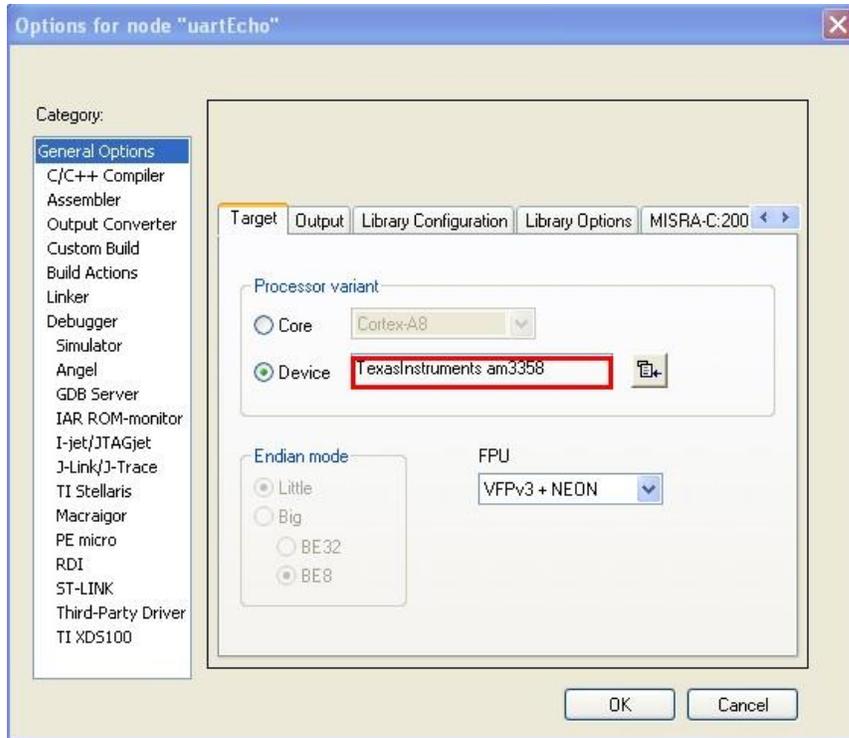
同时，对仿真器连上 core 后行为也可以进行设置，鼠标右键点击 properties，得到以下界面：



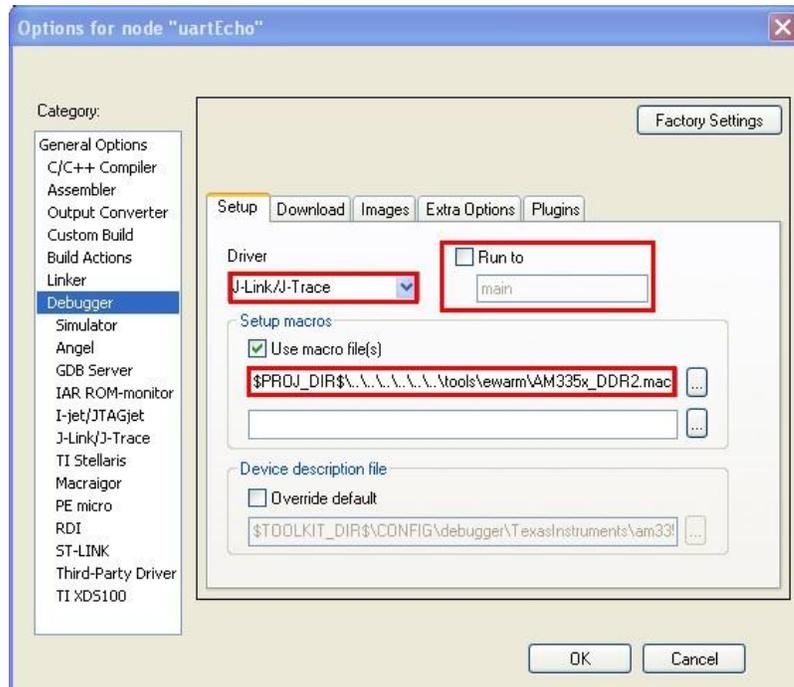
在该页配置中，在调试过程需要调整的就是 Auto Run Options，可以根据需要设置，当加载了 image 后，core 自动运行到指定的 symbol，并且被 JTAG 接口停住 PC 指针。

b. IAR 中的配置

选择 Project->Options，可以得到下面的界面：



在 Device 中选择 AM3358 即可。这里对应与 CCS 中，配置了 SOC。相应的，选择 Debugger 页面，可以得到如下界面，进行 Target 的配置。



在上图的 Driver 中，选择 J-Link/J-Trace 即是选择了 J-Link 仿真器。Use macros file(s)中选择的 AM335x_DDR2.mac(在这里下载)，其功能和上面介绍的 GEL 文件基本一样，都是为了初始化一个基于 AM335x [GP EVM](#) 的最小系统。关于 mac 文件可以参考 [IAR 的文档](#)。

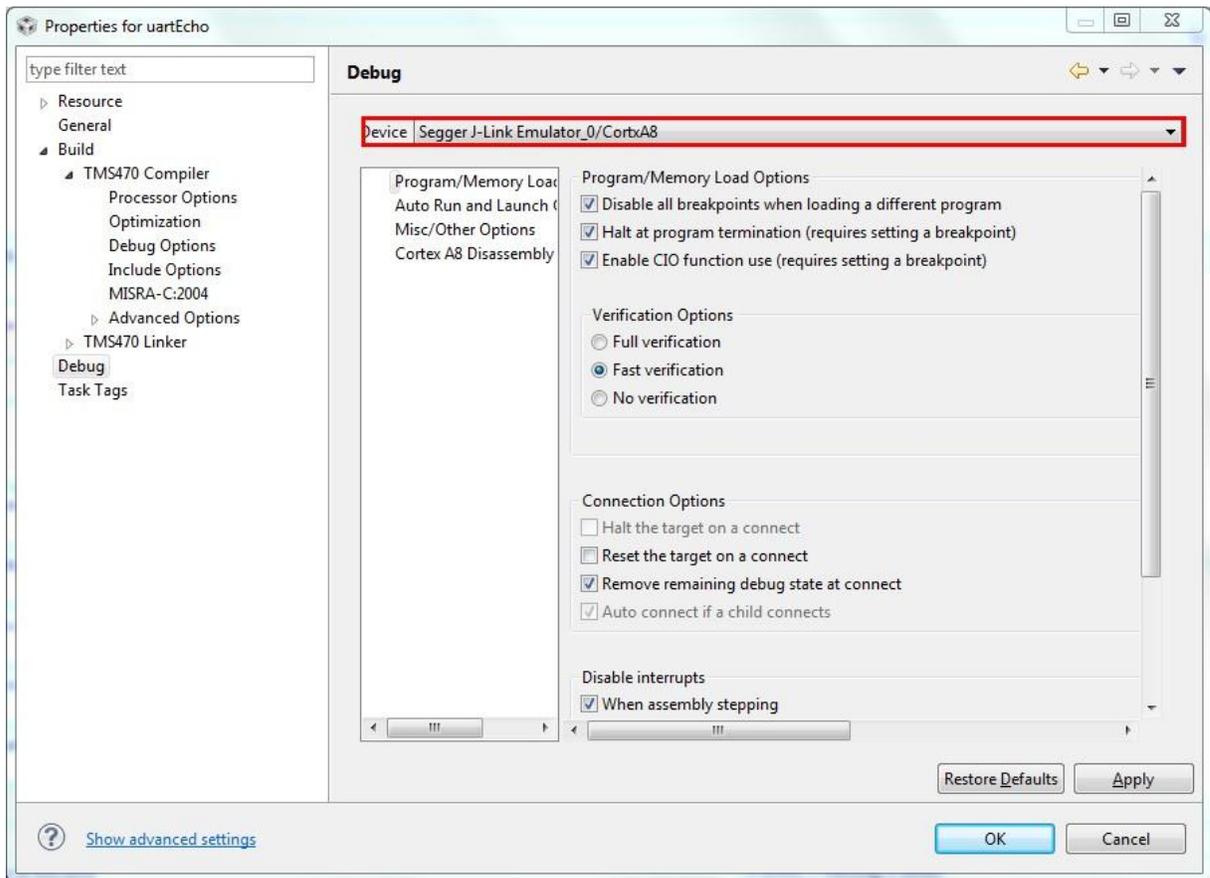
Run to 选项是指加载了 image 后，Cortex-A8 core 所运行到的 symbol。

B. Target 的连接

a. CCS 中仿真器的连接

完成上述配置后，可以连接 target，可选择手动连接或者调试时自动连接两种方式。

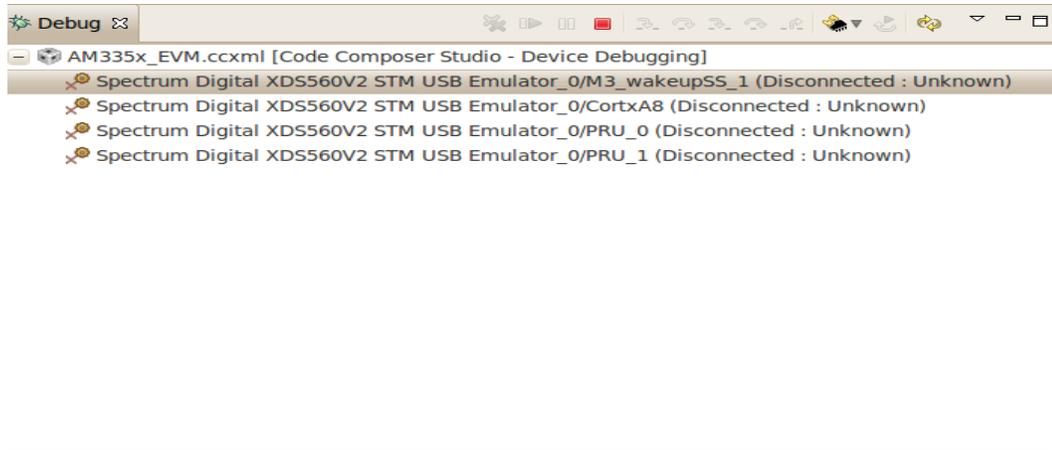
自动连接便于调试，可右键选中 Target Configurations 中已配置好的 target，AM335x_EVM.ccxml，在右键中选择 Set as default，此时 AM335x_EVM.ccxml 就成了默认的调试对象。然后，右键选择导入的 uartEco 工程，选择 Properties，在其页面中选择 Debug，如下图所示：



如红色矩形框中所示，选择 CortexA8。调试 uartEco（点击下图中红色框中的按钮）时，将会自动连接已配置的 target，并连接上 ARM core，并将已编译好的 image 下载到板级的 DDR 上。



接下来介绍手动连接，可以用于直接下载不是在当前 CCS 中编译的 image，具体做法是，右键选中 Target Configurations 中已配置好的 target，AM335x_EVM.ccxml，在右键菜单中选择 Launch Selected Configuration，连接成功后，可以得到下图



此时，PC 和仿真器以及仿真器和 SOC 的 JTAG 连接成功，但是 ARM core 还没有连上。从图中可以看到，有多个 core 的配置选项，这里只关注 ARM core，即 CortexA8。

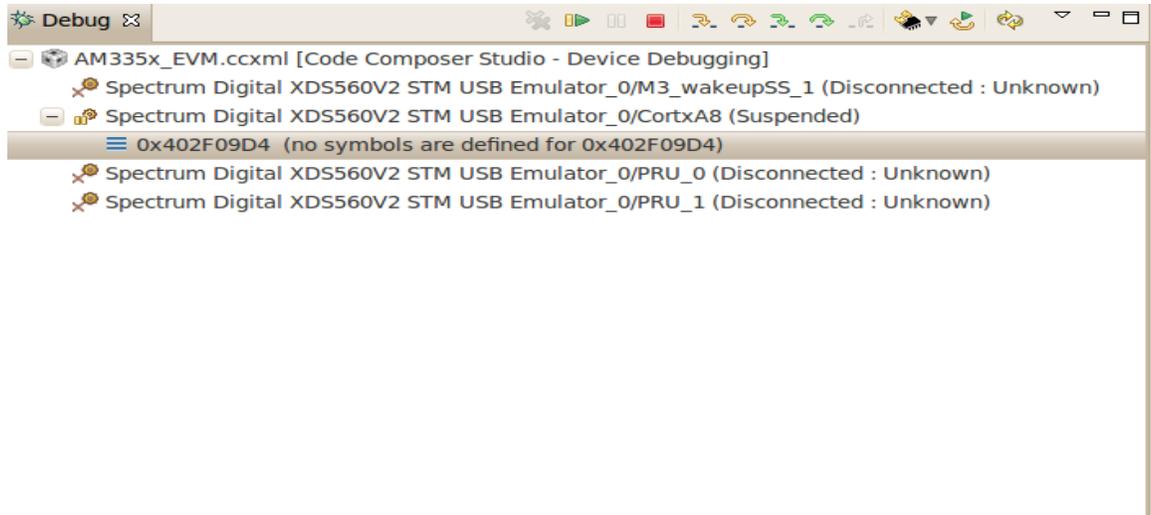
b. IAR 中仿真器的连接

IAR 中不需要单独连接 target，调用 Debug 时，会自动连接 JTAG 和 ARM core。

5.1.2 ARM Core 连接

a. CCS 中 ARM core 的连接。

对于自动连接，如前所述，ARM core 的连接是 Debugger 自动完成的。这里介绍的是手动连接的情况，在 Debug 窗口中，右键点击 CortexA8 core, 选择 Connect Target. 连接成功后,如下图所示：



此时，Debugger 已经成功连上 AM335x 的 Cortex-A8 core 了。

b. IAR 中 ARM core 的连接。

IAR 中不需要单独连接 ARM core，调用 Debug 时，Debugger 会自动连接 JTAG 和 ARM core。

5.3 代码调试

代码的调试主要分加载 image，加载 symbol 和调试三个步骤。

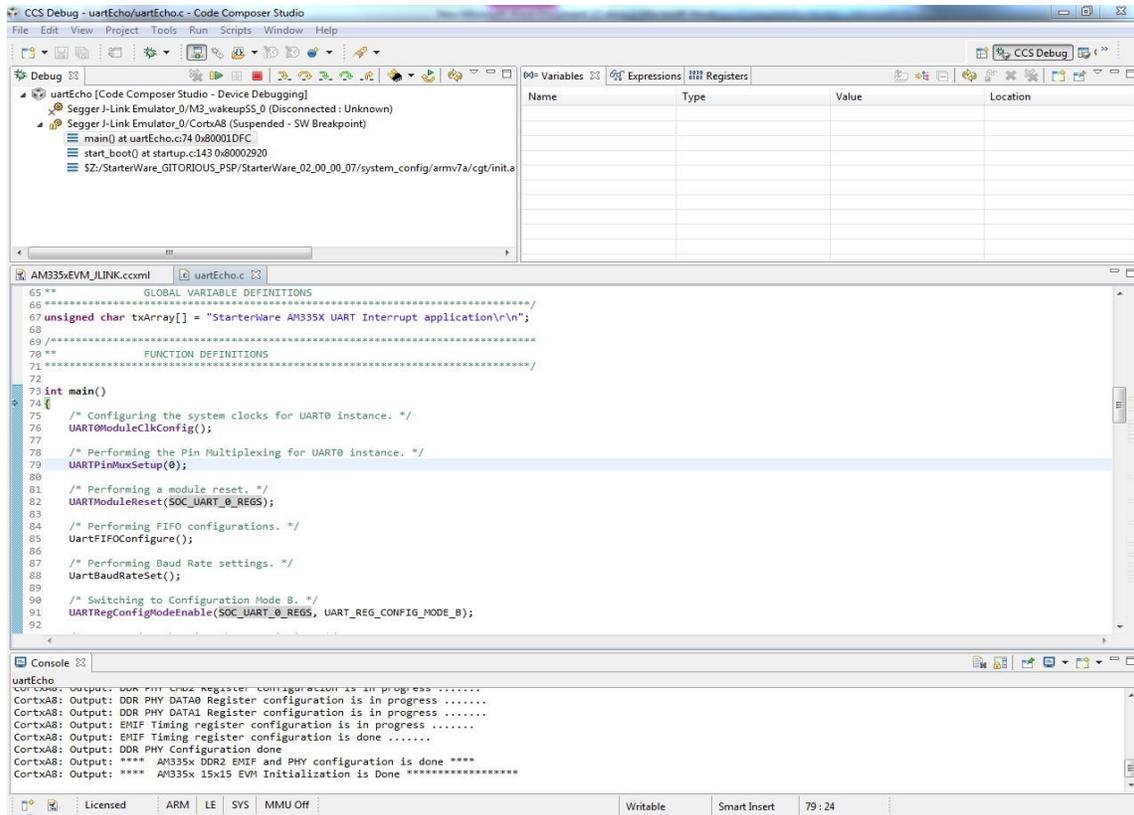
5.3.1 加载 image

Debugger 加载 image 到 AM335x 的片上 SRAM 或者板级 DDR 上的方法有两种：

- A. Debugger 通过 JTAG 加载。
 - a. 如果需要将 image 加载到 DDR 上，需要先初始化 PLL，DDR 等，搭建一个最小系统后，DDR 才能接收 image。如前所述，GEL/mac 文件可以完成这一工作，所以在加载 Application image 时，需要先加载 GEL/mac 文件。相应的 Bootloader 则可以直接加载其 image 到片上 RAM 上即可。
 - b. 加载 image 后，Debugger 就会根据前面的配置，使得 Core 的 PC 停到指定的 symbol 处。
- B. 通过 AM335x 所支持的启动存储介质或者外设来加载。
 - a. AM335x 支持多种启动方式，包括 SD 卡，NandFlash，UART 等，可以将 image 通过这些方式来启动。同样，对于调试 Application，需要先加载 BootLoader，需要通过 BootLoader 来加载 Application。其实，这就是 Starterware 启动的过程，可以参考 [wiki](#)。
 - b. 加载 image 后，需要在 Debugger 中设置需要调试部分的入口地址，并使 Core 的 PC 指针停留需要调试的入口处，从而开始调试。

由于 Starterware 的代码比较少，所以一般直接采取通过 JTAG 的方式下载 image，所以这里只介绍通过 JTAG 下载 image 的方式，具体如下：

- A. 在 CCS 中加载 image
 - a. 自动加载 image 的方式，在仿真器的连接中已有介绍。
 - b. 手动加载 image 的方式。连接好仿真器和 ARM core 后，菜单 Run-> Load-> Load Program，然后选择所需加载的 image 即可。
如前所述，这里推荐自动加载 image 的方式。Image 加载成功后可以得到如下界面：

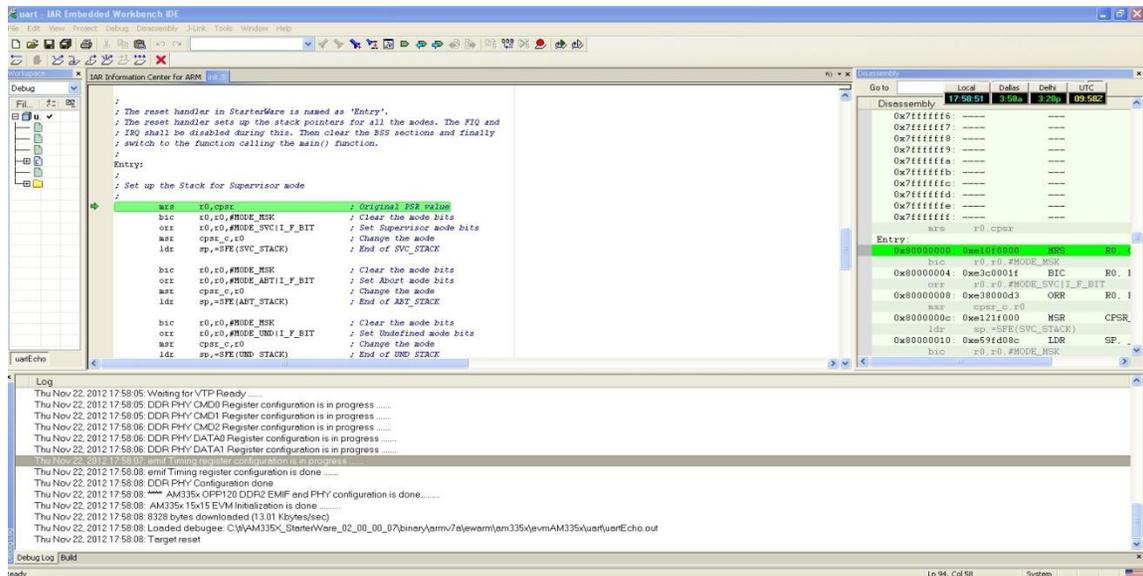


B. 在 IAR 中加载 image

- 配置 Debugger 之后，在工程窗口中选择 uartEco 工程，在工具栏中选择，Download and Debug, 如下图红框中的按钮，就进入 IAR 的 Debugger 模式。



- IAR 在 Debugger 模式中会自动调用前面所配置的 .mac 文件，并加载相应的 image。这里，如果调试的是 BootLoader，注意不要选择 .mac 文件，因为 BootLoader 中会配置 PLL，DDR 等。
- 加载成功后，可以得到如下的界面。PC 指针会根据前面所述的配置停到指定的 symbol 处。



5.3.2 加载 symbol

对于通过 JTAG 的方式加载的 image，因为 image 是通过 Debug 方式生成的，image 中已经包含有相应的 symbol，所以不需要额外加载 symbol 了。

5.3.3 调试

加载 image（包括 symbol）成功后，即可进行单步，设置断点等调试。CCS 和 IAR 中分别有对应的工具栏进行设置：



这些和一般的在线调试基本一致，这里就不再过多介绍了。

6 总结

至此，已经完整介绍了如何使用 J-Link 在 CCS 和 IAR 集成调试环境中对基于 AM335x 的 Starterware 进行编译调试，希望借此能够更进一步加速 Starterware 的开发。