低成本的**ARM**调试方案**——**有关于**Wiggler**、**H-Jtag**、**OpenOCD**、**GDB**、**Insight

2009-05-20 17:26:07

低成本的ARM调试方案——有关于Wiggler、H-Jtag、OpenOCD、GDB、Insight

先给大家介绍个大概情况，现在国内都有什么著名的ARM开发工具和解决方案，价格从高低排：

- BDI1000/2000/3000

  目前我知道的最牛X的调试工具，可以调试ARM、MIPS、PPC、ColdFire、XScale等多种处理器。无需更换硬件，只需要买不同的软件授权就可以调试不同的CPU。JTAG下载速度可以上兆，以太网接口。因为太贵了（BDI2000好像要人民币50000吧），我没怎么研究它到底配合什么软件来调试，不过GDB它是肯定支持的，它一直是我心目中的神话啊。

- J-Link原版

  J-Link是IAR公司为ARM开发的调试工具，支持RDI协议的调试工具，如Keil、ADS、IAR等；支持GDB调试；什么SWD之类的用得很少，有没有都一样；但J-Link不支持ARM10以上的内核。JTAG下载的速度可以达到400~500K，正版价格大约5000人民币（全功能）吧，这么贵基本也不考虑了。

- Multi-ICE原版

  ARM公司的原创调试工具，支持全系列ARM芯片，现在多少钱我也不知道了，反正在2000~3000人民币这个级别。我这里指的是国内做得比较好的那些，比如Realview之类的。仅仅支持ADS、SDT之类的裸奔代码调试，JTAG下载速度130K左右。虽然这几年Multi-ICE是国内ARM调试绝对的霸主，但现在ARM公司已经停止对ADS的维护了，Multi-ICE会开始走向没落。

- Multi-ICE盗版

  国内有很多Multi-ICE的盗版，功能和Multi-ICE原版一样，并口的、USB的都有，价钱几百块人民币，淘宝上到处都有。但是和J-Link盗版相比，不推荐购买。

- J-Link盗版

  最近这段时间，J-Link盗版渐渐开始多起来了，淘宝上也很多，功能和原版没有区别。价格大约在几百人民币左右，从性价比来看，推荐购买。我之后还会写一篇用J-Link调试ARM的文章，当你入门之后，绝对无法忍受今天介绍的这个低成本方案的JTAG下载速度，那时就买个J-Link来爽爽。

- U-Link盗版

U-Link是Keil公司做的用于ARM和某些增强型8051调试的工具，由于Keil公司做U-Link的时候没有加密，导致现在盗版满天飞，只需要100多块钱就可以买到一个。现在Keil已经被ARM收购，U-Link也是ARM一家的了。U-Link正版在盗版的排挤下，根本没有什么买的必要；U-Link仅仅支持Keil，而且JTAG下载速度仅有20~30K。

- Wiggler电缆

  Wiggler是世界上最泛滥的一种调试工具，它非常简单，只需要一片74HC244，一个9013，几个电阻就可以。本来Wiggler是Macraigor（http://www.macraigor.com/）制作的，可以支持Macraigor的OCDRemote这个GDB Server，可以支持ARM、PPC、ColdFire、MIPS、XScale等多种CPU。后来因为它结构太简单，被人破解后搞得全世界都是，于是Macraigor怒了，现在用OCDRemote必须是Macraigor原厂的Wiggler了……尽管如此，后人又在Wiggler的硬件基础上开发了很多的调试工具，例如H-Jtag；另外也有其他的调试工具增加了对Wiggler的支持，例如OpenOCD。Wiggler电缆的成本特别低，当然它的性能也和成本一样低；用H-Jtag下载速度大约20~30KB/s，用Linux虚拟机下的OpenOCD下载速度大约2KB/s。不过对于囊中羞涩的学生们来说，是一个非常不错的入门工具。本文就针对Wiggler进行介绍。

估计看这篇文章的人会有一些是从单片机起步，转到ARM上来的，我先梳理下各种CPU调试的知识。

- 从MCS-51/96、PIC之类的单片机转入ARM

  这条路是学校教学比较传统的路子。条件好点的学校开单片机课的时候都有实验，用实验箱和仿真器做实验，那种仿真器就是一种最早的CPU/MCU仿真器，仿真器通过仿真头连接电路板，完全模拟CPU/MCU的功能；仿真器通过串口或者其他什么口连接计算机，计算机上有集编译器、调试器为一体的集成开发环境，可以监控和运行程序。这种仿真器的成本一般比较高，而且仿真不同的CPU/MCU需要不同的硬件，结构也很复杂，使用软件模拟的断点。解释下软件模拟的断点——就是用特殊的函数调用指令替换断点所在位置的指令，这些特殊的函数具有和仿真器的监控软件交互的功能。应该有很多的同学平时没有条件用上这么奢侈的设备，多半是用的是ISP，采用“点灯****”——就是借助LED、串口之类的调试程序，每修改一次程序就重新下载一次，调试非常的艰苦。我刚开始玩单片机的时候AT的89S52还没有出来，我花一个月的生活费买了个烧写器，每次改程序都把芯片撬下来放到烧写器上，烧完再装回去继续点灯……
  走这条路，要明白的事情有：ARM的寄存器可不是51那寥寥可数的几个，是没有必要也不可能背下来的；ARM芯片一般都内置了JTAG调试逻辑，不需要CPU仿真器，需要的是一个JTAG协议转接器（虽然现在大家还叫这种东西为仿真器）；集成开发环境在使用者看来和单片机的没有任何区别，这点请放心。

- 从AVR、C8051F之类有JTAG的单片机转入ARM

  时代是不断进步的，AVR、C8051F具有JTAG口的单片机。JTAG（Joint Test Action

Group）组织定了一个最初是用于测试生产出来的芯片是不是良品的测试接口和标准，在芯片的各个管脚上放上锁存器，然后串起来构成移位寄存器，可以监控芯片管脚的输入和输出；后来大家发现这东西用来搞芯片的在线调试不错，于是就出现了现在JTAG调试风行的局面。再说的明白些，也就是利用JTAG可以控制CPU内核，每个CPU都可以成为自己的“仿真器”，而不需要专用的设备。“人人都是食神。”——周星星语录。从理论上来说，世界上只需要一种仿真器，哦，确切的说应该叫做JTAG协议转换器，就可以调试所有的兼容JTAG标准的芯片；BDI2000这种超级贵的“仿真器”以及Wiggler这种什么都通吃的便宜货的存在是很合理的事情。
走这条路，应该已经明白了JTAG是什么，所以不用多说了。

- GDB是什么

  正像Windows和Linux的对比，集成开发环境比GDB在嵌入式开发领域，拥有更多的用户，但这并不意味的GDB不好。GDB（GNU Project Debugger）是开源软件组织GNU开发和维护的一种调试工具，它能调试目前所有的能跑Linux的CPU，当然ARM也是其中一员。对于初学者来说，不建议使用GDB，还是先从集成开发环境入手，例如ADS、SDT、Keil、IAR之类的。其实从编译器的层面来讲，集成开发环境和GDB所用的编译器GCC没有什么区别，但集成开发环境里面提供了源文件组织与浏览、工程文件管理、调试等多种功能，用起来很友好。GCC+GDB光学习写相当于工程文件的Makefile就要花很多的时间。但是，一旦你的学习进了一步到了Linux的Loader和内核，集成开发环境就无能为力了。前面已经提到了，本文覆盖了从刚开始的裸奔代码到涉足操作系统的GCC+GDB调试环境的建立方法。本文关于GDB的部分应该是国内挺难找到的HOWTO，转载请注明来自EE小站。关于GDB，可以参考下我之前的这篇文章http://xianzilu.spaces.live.com/blog/cns!4201FDC93932DDAF!268.entry。

在开始之前请先确认你的电脑有并口，如果是笔记本就算了，买个PCMIA转并口的卡的钱够买个盗版U-Link了；要是肯下血本买盗版J-Link，那就看我以后写的文章。

- 首先说代码裸奔怎么做

  你需要的东西有：

  ● 带并口的电脑一台
  ● 并口延长线一根
  ● Wiggler一个
  ● 随便什么ARM7或ARM9的开发板一个

  如果没有并口延长线，可以去电脑城买一根。如果没有Wiggler，你可以选择DIY，下面这张图是Wiggler的一种版本：
  如果不想DIY，上淘宝淘一个去。ARM开发板也可以在淘宝上淘淘，看你的经济能力了。

  你需要的软件有：

● ADS (ARM Developer Suite) V1.2
● H-Jtag

ADS在一般学校的FTP上都有，H-JTAG请访问http://www.hjtag.com。在此再拜一下Twentyone大侠，可以为大家写出这么好的免费软件。
H-Jtag和ADS的使用方法在H-Jtag的网站上的手册里写得很清楚了，我就不再啰嗦了，给出地址http://www.hjtag.com/download/USER%20MANUAL%20(CN).pdf。

- 说说GDB怎么做

如果你对Linux下ARM的开发没有概念，先看我还是菜菜鸟的时候写的这篇文章http://xianzilu.spaces.live.com/blog/cns!4201FDC93932DDAF!121.entry。

GDB使用GDB工具链，调试解决方案的结构是

GDB前端<--->GDB<--->GDB服务程序<--->JTAG协议转换器（仿真器）<--->目标CPU（ARM CPU）
|
控制接口

GDB有一个很大的缺点——文本界面，使用非常不方便。但幸运的是，有很多热心的开发者为GDB写了一些图形“外壳”——GDB前端，大大方便了GDB的使用。因为我们做的是交叉开发（即在x86结构的电脑上开发ARM等非x86结构的CPU程序），所以GDB无法直接调试编译出来的程序，这就需要一个服务程序。这个服务程序可以是一个可以控制目标CPU的程序（可能运行于计算机上；也可能运行于某些仿真器上，例如如BDI2000就是这样），也可以是一个运行于目标CPU上的服务程序，由它来装载被调试的程序。但是后者一般需要目标CPU上已经运行起了Linux内核；调试Bootloader和Linux内核本身，需要前一种服务程序。GDB和GDB服务程序之间的连接方式可以是以太网或者串口，而且GDB服务程序一般还有别的控制接口，例如Telnet接口，可以实现对目标CPU的控制，如初始化和程序文件下载等。比较复杂哦，一会儿说到软件的时候就会用上这些知识。

你需要的东西和裸奔代码一样

你需要的软件有：

● 一个可以运行的Linux

虚拟机里的、真实的都可以，推荐使用Open Suse 10.3，下载地址http://software.opensuse.org/

● 本机GCC编译器

Open Suse自己带的就可以

● 交叉GCC编译器

可以去下载一个，随便给个地址把http://www.linux4sam.org/twiki/bin/view/Linux4SAM/SoftwareTools#Cross_Toolchain值得注意的是U-Boot 1.2.0之后需要使用支持软浮点的交叉编译器，如果没有，可以用Crosstool制作一个，可以看我之前的这篇文章http://xianzilu.spaces.live.com/blog/cns!4201FDC93932DDAF!274.entry

● OpenOCD源码

OpenOCD的主页是http://openocd.berlios.de/web/。OpenOCD是一个运行于PC上的程序，它可以控制包括Wiggler之内的很多JTAG硬件；我们可以将它理解为一种GDB服务程序。OpenOCD的源码只能通过SVN下载，地址是svn://svn.berlios.de/openocd/trunk
，在写这篇文章的时候OpenOCD已经是R818版本了，这个版本对Wiggler的支持有问题，我编译的是r520版本的，如果没有SVN Client，这个版本只能通过曲线的方式获得：先到下载http://www.yagarto.de/download/oldver/openocd-r520-20080322.exe这个由YAGARTO提供的OpenOCD For Cygwin的版本，安装它，在安装目录例如C:\Program Files\openocd-r520\source里找到源码压缩包。

● Insight源码

Insight是一个GDB的图形前端，我感觉它比DDD更适合嵌入式系统程序的调试。
Insight的下载地址http://sourceware.org/insight/downloads.php。

● 随便什么程序的源码，例如U-Boot

U-Boot就不用介绍了，如果不知道可以Google下。
U-Boot的下载地址是http://www.denx.de/wiki/UBoot/SourceCode。

下面开始编译，先是OpenOCD，假设源代码已经解压缩到了/home/lxz/build-openocd，先设定权限
cd /home/lxz/build-openocd
chmod 755 ./bootstrap
./bootstrap
等一会儿，输入
./configure --prefix=/usr/local/arm/openocd --enable-parport
这里--prefix指定的是安装的路径，--enable-parport使能并口，然后
make
sudo make install
输入root密码，等一会儿，安装就完成了

然后是insight，假设源码已经解压缩到了/home/lxz/insight-6.8，然后
cd /home/lxz/insight-6.8
./configure --prefix=/usr/local/arm/arm-linux-insight --target=arm-linux

这里--prefix指定的是安装的路径，--target指的是为ARM编译GDB，等一会儿，输入
make
等一会儿，输入
sudo make install
输入root密码，等一会儿，安装就完成了

然后编译一个U-Boot用于测试，假设源码已经解压缩到了/home/lxz/at91rm9200/u-boot-1.2.0，假设已经修改完了Makefile中的交叉编译器的选项，假设我为AT91RM9200DK开发板编译，然后
cd /home/lxz/at91rm9200/u-boot-1.2.0
make at91rm9200dk_config
make
于是得到了/home/lxz/at91rm9200/u-boot-1.2.0/u-boot这个映像

为了能让OpenOCD正常使用，我们还需要2个脚本，第一个是OpenOCD的配置脚本，这个脚本的作用是配置GDB服务程序、JTAG仿真器。写这个脚本可以看OpenOCD的文档 http://openfacts.berlios.de/index-en.phtml?title=OpenOCD_configuration。我给出我的AT91RM9200DK开发板的配置文件at91rm9200.cfg，每一条配置信息的作用我就不解释了，请仔细阅读OpenOCD的文档。

```
# Daemon configuration
telnet_port 23
gdb_port 2331
daemon_startup reset
# JTAG interface configuration
interface parport
jtag_speed 0
reset_config trst_and_srst
jtag_device 4 0x1 0xf 0xe
# parport options
parport_port 0x378
parport_cable wiggler
# Target configuration
target arm920t little run_and_init 0 arm920t
run_and_halt_time 0 1000
target_script 0 reset at91rm9200_init.script
working_area 0 0x00200000 0x1000 backup
```

我还是提一下，上面这段配置信息中的target_script 0 reset at91rm9200_init.script这句就是指定第二个脚本的，而且让OpenOCD在当前目录下搜索这个脚本。也就是说，如果at91rm9200.cfg在/home/lxz/at91rm9200下，那么你在/home/lxz/at91rm9200下启动OpenOCD服务程序，OpenOCD就会在/home/lxz/at91rm9200下搜索at91rm9200_init.script这个脚本；如果在与at91rm9200.cfg所在路径不同的路径下启动OpenOCD服务程序，OpenOCD就无法找到at91rm9200_init.script，此时，target_script 0 reset at91rm9200_init.script这句就应该写成target_script 0 reset /home/lxz/at91rm9200/at91rm9200_init.script。
第二个脚本的作用是初始化ARM CPU，因为U-Boot往往是在SDRAM里运行的，其连接位

置也都在SDRAM里。用GDB或GDB前端下载程序的时候，必须保证SDRAM是可用的。AT91RM9200这个CPU上电的时候如果从片内Boot ROM启动（不推荐从外部启动，因为如果没有启动程序，AT91RM9200将运行于慢时钟，这样JTAG仿真器可能工作不正常），需要进一步配置PLL，PIO，SDRMC之类的外设之后，SDRAM才可以使用。第二个脚本就是一系列寄存器读写和延时命令的集合，如何编写请看OpenOCD的手册http://openfacts.berlios.de/index-en.phtml?title=OpenOCD_commands，给出我的at91rm9200_init.script。

```
mww 0xfffffc28 0x00000000
mww 0xfffffc2c 0x00000000
mww 0xfffffc20 0x0000ff01
sleep 20
mww 0xfffffc28 0x20263e04
sleep 20
mww 0xfffffc2c 0x10483e0e
sleep 20
mww 0xfffffc30 0x00000000
sleep 20
mww 0xfffffc30 0x00000202
sleep 20
mww 0xfffff870 0xffff0000
mww 0xfffff804 0xffff0000
mww 0xffffff60 0x00000002
mww 0xffffff64 0x00000000
mww 0xffffff98 0x2188c159
mww 0xffffff90 0x00000002
mww 0x20000000 0x00000000
mww 0xffffff90 0x00000004
mww 0x20000000 0x00000000
mww 0x20000000 0x00000000
mww 0x20000000 0x00000000
mww 0x20000000 0x00000000
mww 0x20000000 0x00000000
mww 0x20000000 0x00000000
mww 0x20000000 0x00000000
mww 0x20000000 0x00000000
mww 0xffffff90 0x00000003
mww 0x20000200 0x00000000
mww 0xffffff94 0x000002e0
mww 0x20000000 0x00000000
mww 0xffffff90 0x00000000
mww 0x20000000 0x00000000
arm7_9 sw_bkpts enable
```

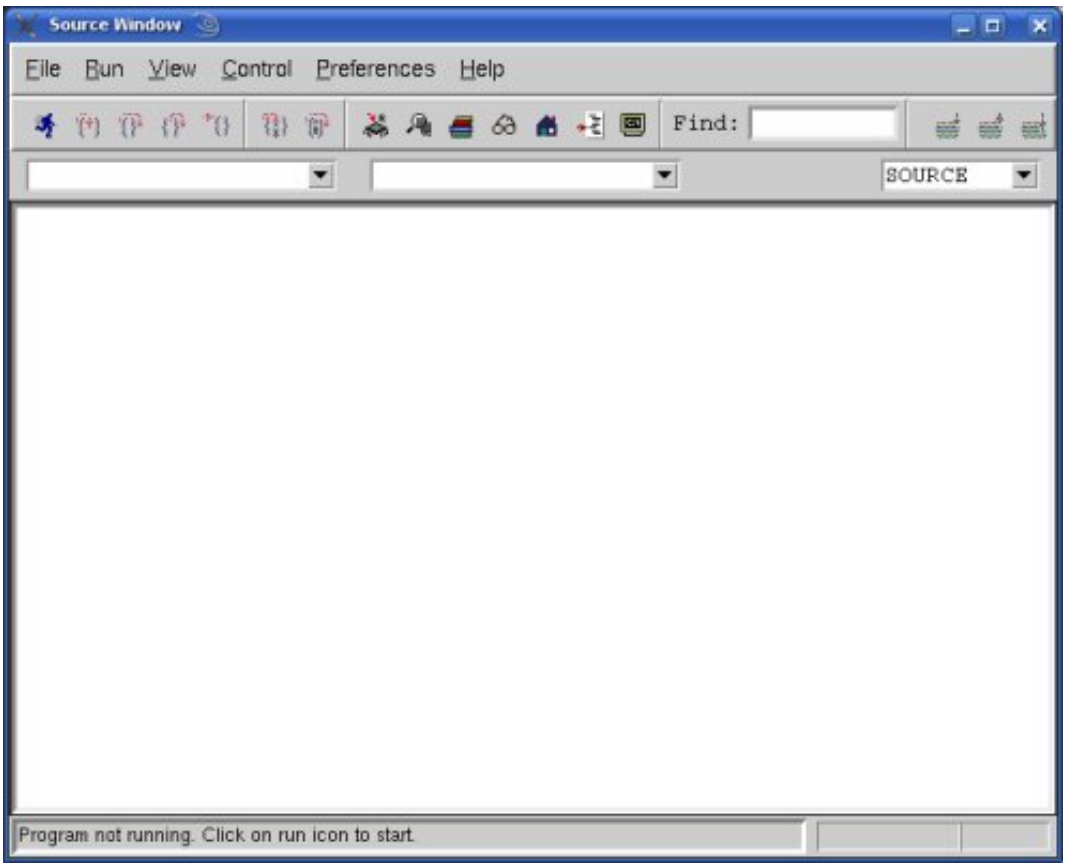
这个脚本写起来很复杂，建议从一些样例代码上把寄存器的数值扒过来。另外，有些CPU，例如S3C2410，它上电的时候，SDRAM是默认可以用的，就不需要这个脚本了。还有一个值得注意的是，由于我们用的是Wiggler这种简单的JTAG协议转换器，初始化脚本里必须加上arm7_9 sw_bkpts enable这句。现在终于可以开始调试了，假设把OpenOCD安装在了/usr/local/arm/openocd，把Insight安装在了/usr/local/arm/arm-linux-insight，两个初始化脚本都放在了/home/lxz/at91rm9200；你已经正确连接了Wiggler，开发板已经上电。接下来还是用命令来说明

```
# cd /home/lxz/at91rm9200
# sudo /usr/local/arm/openocd/bin/openocd -f at91rm9200.cfg
root's password:
Open On-Chip Debugger 1.0 (2008-07-21-20:15) svn:
$URL: http://svn.berlios.de/svnroot/repos/openocd/trunk/src/openocd.c $
Info:    jtag.c:1329 jtag_examine_chain(): JTAG device found: 0x05b0203f (Manufacturer:
0x01f, Part: 0x5b02, Version: 0x0)
Info:    target.c:240 target_init_handler(): executing reset script 'at91rm9200_init.script'
Info:    options.c:50 configuration_output_handler(): software breakpoints enabled
```

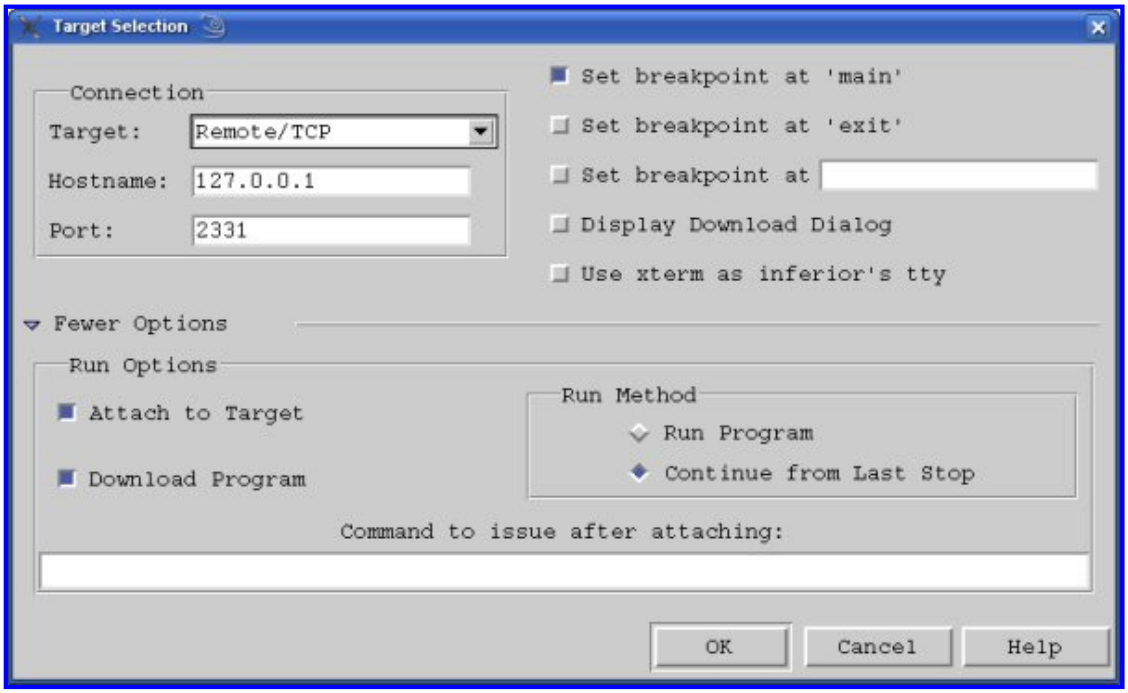
这就说明OpenOCD已经开始工作了。然后启动Insight

```
# cd /home/lxz/at91rm9200/u-boot-1.2.0/
# /usr/local/arm/arm-linux-insight/bin/arm-linux-insight
```

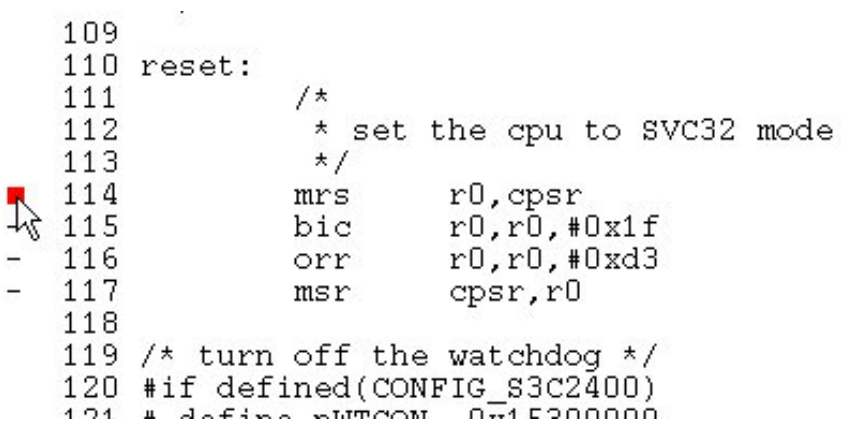
出现下面的窗口



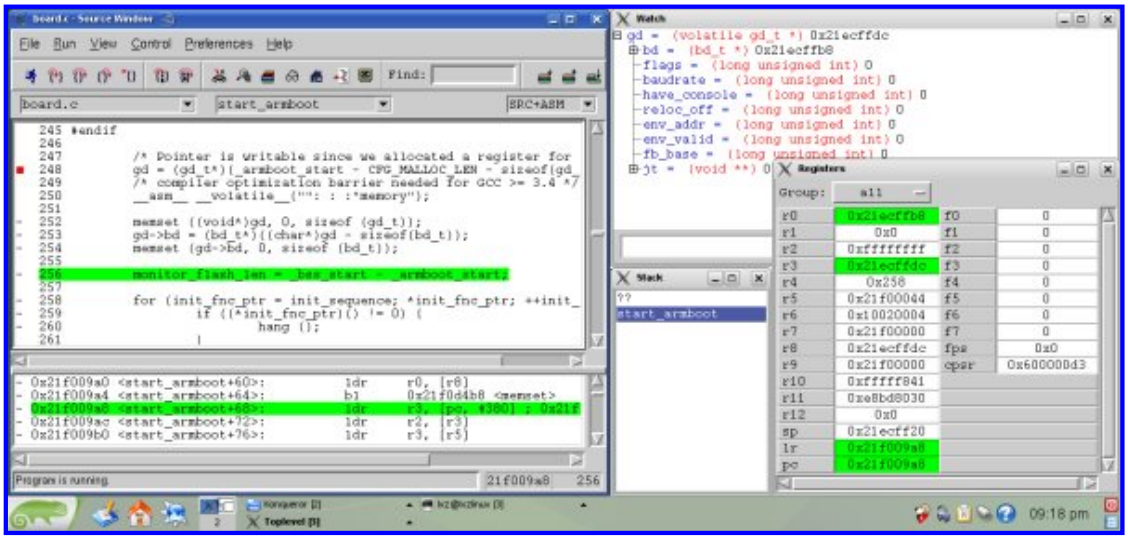
然后选择菜单File>Target Settings...，在出现的窗口中进行如下设置，然后点OK。



选择菜单File>Open，打开/home/lxz/at91rm9200/u-boot-1.2.0/u-boot这个映像；然后选择菜单Run>Download，将U-Boot程序下载到目标CPU。然后在程序运行的必经之路设定一个断点，如下图所示。



选择菜单Control>Continure，程序就会从头开始执行，并停在断点处了。Insight还有很多不错的功能，并且很容易上手，大家研究下就好。补充一点，如果你对你的初始化脚本是否起作用没有信心，可以在启动Insight之后只选择菜单Run>Connect to target，然后选择菜单View>Memory查看各个寄存器和内存。最后给出一张我用Insight调试U-Boot的截图。



在使用的过程中就会发现，用Wiggler下载的速度实在不怎么样，U-Boot的可执行映像至多只有200KB，所以还是可以忍受的。

用同样的方法也可以调试其他Boot Loader，甚至是Linux内核；但是Linux内核的可执行映像一般有2MB之大，用Wiggler调试也是不现实的。我之前已经做了广告了，内核的调试要用J-Link来搞，敬请期待EE小站的后续文章。

我对ARM CPU的在线Flash Download这件事情不是很感冒，所以H-JTAG和OpenOCD的这部分功能EE小站是不会涉及了，请见谅。今天就到这里。