嵌入式系统设计专题实验指导书

——基于W90P710 ARM7处理器实验板的设计与应用

[](http://pic.sogou.com/javascript:void(0))

刘美兰 刘瑞玲

2019年9月

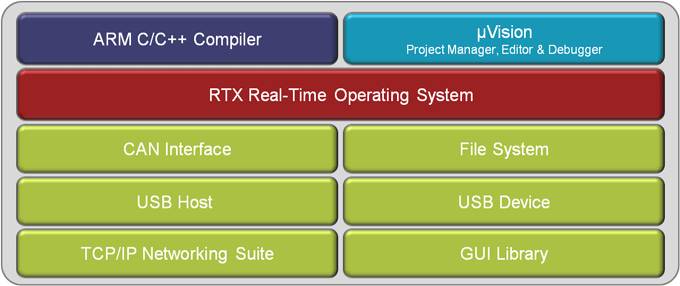
目 录

1. 基于ARM处理器的C语言的编程与调试
2. 嵌入式交叉编译环境的建立和使用
3. 使用gdb在开发板w90p710上调试uClinux内核

**第一章 基于ARM处理器的C语言的编程与调试**

1. **开发工具包简介**

MDK-ARM（ARM微控制器开发工具包）是ARM公司推荐的用于基于ARM处理器的微控制器的完整软件开发环境。MDK-ARM由Keil公司（已被ARM公司收购）提供，利用了该公司先进的µVision集成开发环境，适用于基于 Cortex™-M、Cortex-R4、ARM7™ 和 ARM9™ 处理器的微控制器芯片开发。MDK-ARM 专为微控制器应用程序而设计，易于学习和使用，同时具有强大的功能，能满足大多数要求苛刻的嵌入式应用程序的需求。



**图1 MDK-ARM模块示意图**

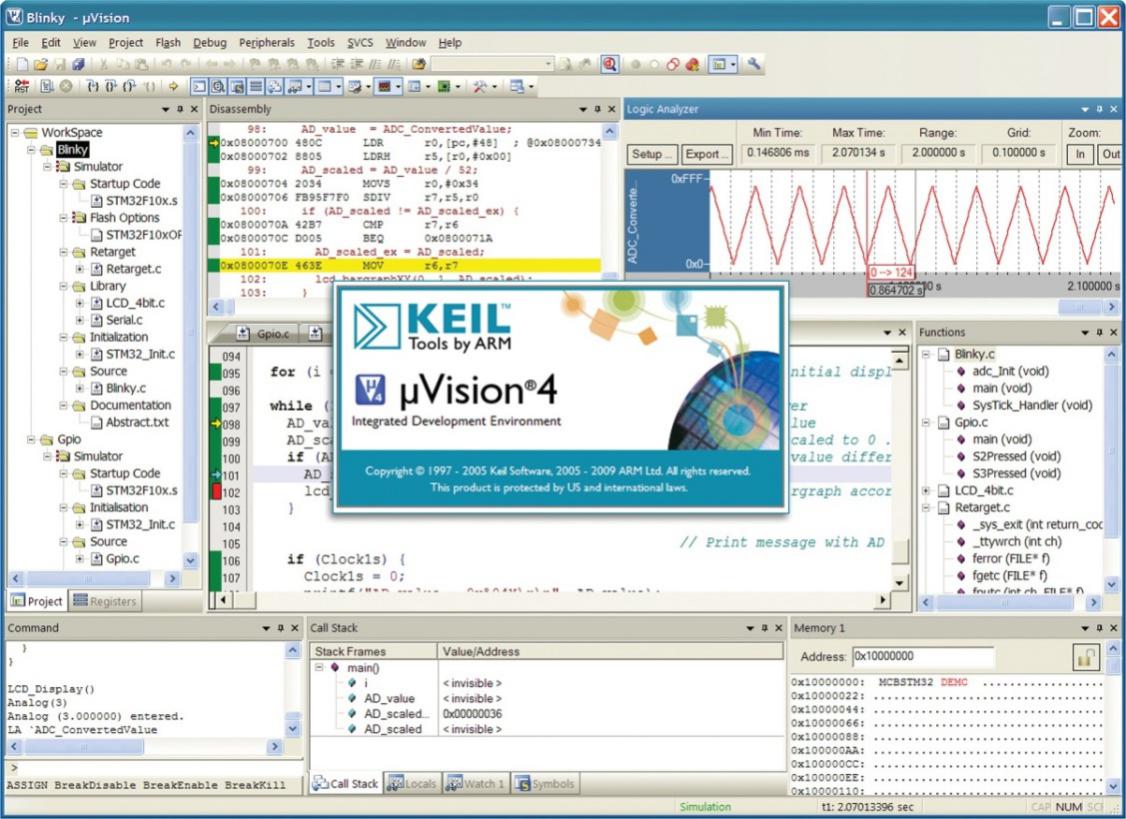
MDK-ARM微控制器开发工具包的主要性能和特征包括：

* 完全支持 Cortex-M、Cortex-R4、ARM7 和 ARM9 微控制器芯片
* 集成行业领先的 ARM C/C++ 编译工具链
* 集成µVision集成开发环境，包括项目管理器、编辑器和调试器
* 集成Keil RTX 确定性、空间占用小的实时操作系统（开源代码）
* 集成TCP/IP 网络套件，提供多种协议和各种应用程序
* 为USB 设备和 USB 主机堆栈配备标准驱动程序类
* ULINKpro支持对正在运行的应用程序进行即时分析并记录执行的每条 Cortex-M 指令
* 可提供执行程序的完整的代码覆盖率信息
* 集成执行性能分析器和性能分析器支持程序优化
* 提供大量示例项目可帮助快速熟悉 MDK-ARM 强大的内置功能
* 符合 CMSIS Cortex 微控制器软件接口标准

MDK-ARM 具有四种版本：MDK-Lite、MDK 基础版、MDK 标准版和 MDK 专业版。所有版本都提供完整的 C/C++ 开发环境和调试支持，MDK 专业版主要包括了丰富的中间件库。其中，MDK-Lite版提供免费下载，不需要序列号或许可证密钥，但是其所能开发的程序大小限制在32KB以内。下载网址在<http://www.keil.com/arm/mdk.asp>。

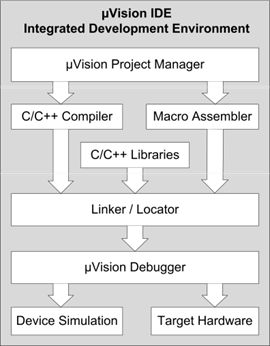
1. **µVision集成开发环境简介**

µVision集成开发环境（Integrated Development Environment, IDE）基于窗口设计，是集项目管理、源代码编辑、编译汇编、程序调试和全功能仿真于一体的集成开发环境，可以帮助开发者快速创建嵌入式程序开始嵌入式软件开发。µVision IDE的基本界面环境如图2所示。



**图2 µVision IDE软件环境界面**

如图3所示，µVision IDE集成有多种嵌入式程序开发所需的模块，包括项目管理器、源代码编辑器、Make工具链、程序仿真调试器等，并能完成闪存Flash编程功能。以下，我们重点介绍其中的器件数据库、Make工具链和程序仿真调试器。



**图3 µVision IDE嵌入式程序模块**

* 器件数据库

µVision器件数据库提供了一种便捷的方式来选择和配置ARM微控制器芯片与项目参数。它包括了预配的设置，因此可以使软件开发者完全集中于应用需求。此外，我们还可以添加自己的芯片器件，或者改变已有的设置。

* Make工具链

针对MDK-ARM平台，µVision IDE在Make工具链中集成了完整的ARM编译工具（之前也被称为ARM RealView编译工具），包括：

* armcc：ARM C/C++编译器，将C/C++程序源代码翻译成可执行的机器语言，可以同时支持ARM和Thumb指令的生成，具有代码大小和性能优化功能，支持内嵌汇编器、内联函数、CPU寄存器参数传递和可重入运行时类库，并支持IEEE-754兼容的单双精度浮点数据处理。
* MicroLib：Microlib类库，专门针对基于ARM的嵌入式应用C语言程序开发的高度优化的库。与ARM编译工具链集成的标准C库相比，Microlib提供了对于多种嵌入式系统需要的重要的代码大小优势。
* armasm：ARM宏汇编器，将汇编语言程序翻译成可执行的机器语言，支持标准的宏处理和条件汇编控制。
* armLink：ARM连接器，将多个将一个或多个由编译器或汇编器生成的目标文件外加库连接成为可执行文件或新的库文件，支持静态栈分析。
* armar：ARM库管理工具，可以在标准格式ar库内有效管理和维护多个ELF目标文件集合。
* fromelf：ARM ELF工具，可以帮助处理由编译器、汇编器和连接器生成的ARM ELF目标文件和映像文件。
* µVision Debugger调试器

µVision Debugger调试器完全集成于µVision IDE集成开发环境之中，可以实现以下功能：

* 对C/C++源代码或汇编级代码进行反汇编，并在各种单步执行和视图模式下，随着程序执行将反汇编代码与源代码相对应。
* 多种断点设置方式，包括访问和复杂断点。
* 支持设置书签，便于快速查找和定义关键点。
* 查看和修改内存、变量和寄存器的值
* 列出包括栈变量在内的程序调用树
* 查看微控制器片上外设的状态
* 支持调试命令或类似C的脚本函数功能
* 执行分析功能可以显示执行时间，以及每条指令所需执行周期
* 为安全攸关应用测试进行代码覆盖率分析
* 多种分析工具、指令跟踪能力和窗口调整功能

µVision调试器支持两种操作模式：仿真器（Simulator）模式和目标（Target）模式。

仿真器模式将µVision调试器配置成纯软件来精确的仿真目标系统，包括指令和大多数片上外设。在此模式下，我们可以在还没有硬件的情况下测试应用代码，便于我们能够快速开发可靠地嵌入式软件。仿真器模式支持：

* 不需要硬件环境在桌面电脑上就可以进行软件测试
* 基于功能的早期软件调试来改善软件可靠性
* 支持硬件调试器无法实现的断点功能
* 理想的输入信号，没有硬件调试器的噪声
* 在信号处理算法过程中单步执行
* 检测出那些会损坏真实硬件外设的失效场景

目标模式将µVision与真实硬件连接起来，可以支持对目标开发板的调试，包括单步执行、设置断点和查看内存等有用的调试方法。本次实验主要使用仿真器仿真对软件进行调试，关于使用目标模式调试开发板在下一次实验中重点介绍。

1. **使用µVision集成开发环境**

本小节，我们开始熟悉µVision IDE的软件界面。通过开始菜单启动Keil µVision 。如图4所示，µVision IDE是一个完全基于窗口的程序开发环境，支持窗口的重排、拖拽等窗口程序基本操作。与通常的Windows软件相似，µVision IDE软件界面包括基本的菜单栏（Menu bar）、工具栏（Toolbars）以及状态栏（Status bar），此外还包括断点和书签设置、命令行提示与输入等功能。



**图4 µVision IDE软件界面指示图**

为了便于理解，我们将µVision IDE中间的主显示区域定义划分为以下三个窗口区域，如图5所示。

* 项目窗口区域：位于屏幕显示的左侧部分，默认显示项目窗口、函数窗口、图书窗口和寄存器窗口。
* 编辑器窗口区域：位于屏幕显示的中间和右侧部分，可以修改源代码、查看性能和分析信息，并检查反汇编代码。
* 输出窗口区域：位于屏幕显示的下部，提供与调试、存储器、定义符号、调用栈、局部变量、命令、浏览信息以及文件查找结果等相关的信息。

窗口的位置和大小可以根据需要调整。必要时，我们可以通过调用Window – Reset Current Layout菜单，将窗口排布恢复到默认状态。



**图5 窗口区域划分示意图**

µVision IDE有两种工作模式：构建（Build）模式和调试（Debug）模式。针对两种不同的模式，屏幕设置、工具栏设置以及项目选项的也相应不同。文件工具栏在两种模式下都会使能，而调试工具栏与构建工具栏分别在各自对应的模式下使能。所有按钮、图表和菜单按照工作模式而使能，当不可用时则显示为灰色。

构建模式为标准工作模式。此模式下，我们可以编写应用代码，配置项目，选择目标硬件和器件。同时，我们还将编译、汇编和连接源程序，修正可能的错误，最终生成可执行的目标文件。

调式模式下主要进行软件代码的仿真与调试。在调试模式下，我们也可以修改一些通用设置并编辑源代码文件，但是这些修改只能在返回构建模式后才能有效。只有那些与调试相关的设置才会立即生效。

µVision IDE的大部分功能都可以通过调用相应的菜单或工具按钮来实现。这些菜单与工具栏的操作与大多数Windows程序很类似，我们将在后面的实验中逐步学习和熟悉。具体的说明请参见参考文献[1]。

1. **创建嵌入式程序**

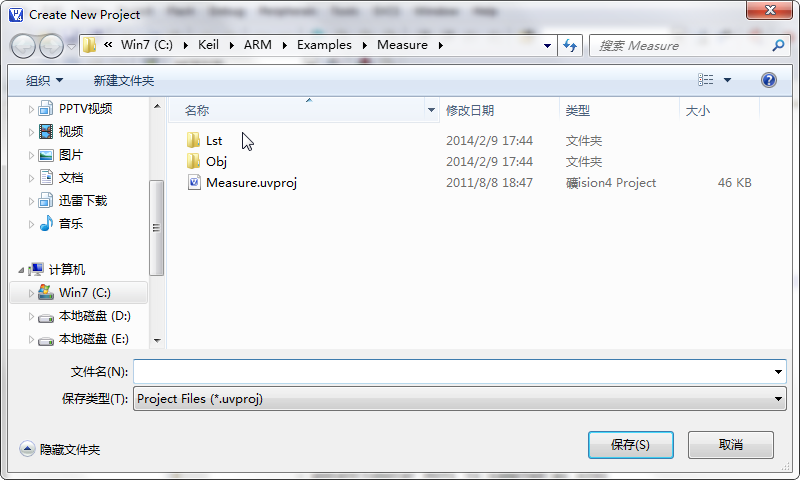
创建一个新的嵌入式项目包括如下几个步骤：

1. 新建项目文件

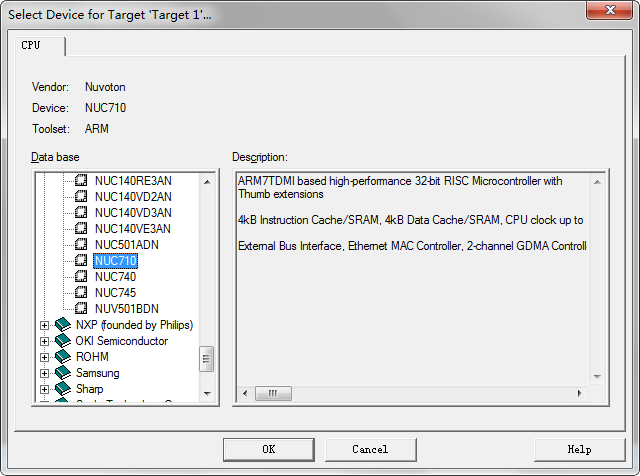
从菜单栏选择Project – New µVision Project，打开一个标准对话框，如图6所示。在对话框中为新建项目选择所在文件夹和项目文件名称，单击Save按钮。建议为每一个新建项目选择一个独立的文件夹，以便于项目文件管理。

单击Save按钮后，µVision弹出Select Device对话框，需要我们选择项目的目标微控制器型号。Select Device对话框中列出了µVision器件数据库中的所有微控制器芯片型号，如图7所示。在项目开发中，我们也可以通过Project – Select Device for Target菜单调出该对话框。为项目选择正确的微控制器芯片类型很重要，因为µVision将根据器件类型定制工具设置、外设以及相应的对话框。

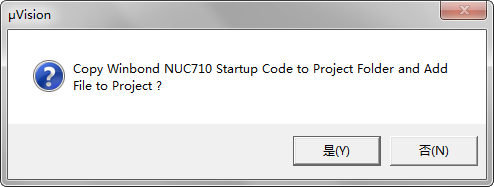
在Select Device对话框点击OK后，µVision会弹出选择拷贝启动代码的对话框，如图8所示。所有的嵌入式程序都需要微控制器初始化和启动的代码来完成与芯片、开发板等相关的硬件配置。µVision中集成了器件数据库中所列出的绝大多数芯片器件的启动代码。通常，µVision将根据器件类型自动选择并拷贝相应的启动代码到项目文件夹中。此处，只需选择Yes即可。



**图6 新建工程示意图**



**图7 选择微控制器型号**



**图8 是否添加启动代码到工程**

1. 使用项目窗口

当我们成功的创建新项目后，项目窗口中会显示该项目的targets、groups和files，如图9所示。默认情况下，target的名称是Target 1，group的名称是Source Group 1。我们可以看到，启动代码文件已经被添加到工程中。



**图9 启动代码添加到工程**

1. 新建源文件

使用File工具栏或选择File – New菜单创建一个新的源文件。在编辑器窗口中，我们可以为新建源文件输入源代码。µVision支持根据文件后缀名的语法彩色高亮功能。为了尽快使用该功能，我们在输入源代码前先将空文件保存。选择File工具栏的保存按钮或使用File – Save菜单，将源文件保存为main.c。

C源代码如下：

#define N 6

int main(void)

{

unsigned char c[6] = {1,2,3,4,5,6};

unsigned char x[6] = {4,8,9,3,5,1};

unsigned char i,f;

for (i=0, f=0; i<N; i++)

{

f = f + c[i]\*x[i];

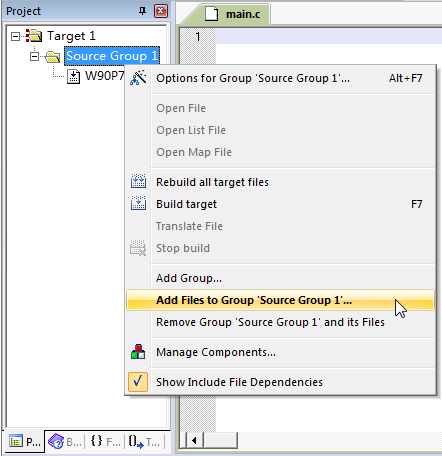
}

return 0;

}

1. 添加源文件到工程

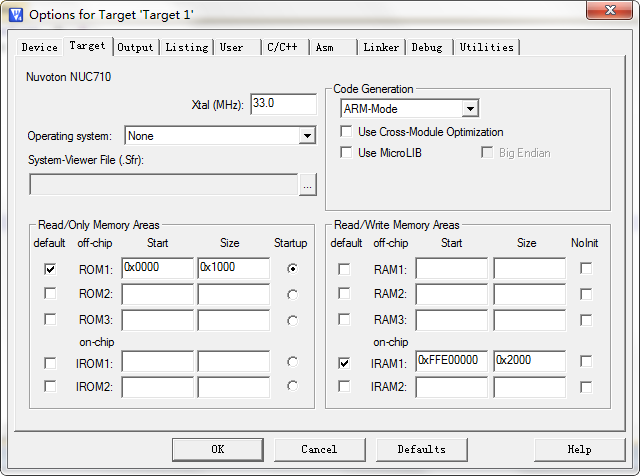
源文件虽然保存在项目目录下，但是由于没有被添加到当前项目中，也不会被编译。我们需要在项目窗口中，用鼠标右键单击Source Group 1，在弹出菜单中选择Add File to Group，按照提示一步步将新建源代码文件添加到当前工程中，如图10所示。



**图10 添加源文件到工程示意图**

1. 设置目标选项

通过Build工具条或从Project菜单打开Options for Target对话框。在对话框中，我们可以更改目标器件类型，设置目标选项，并配置开发工具链。请按照如图11所示设置Target，其余使用默认设置。

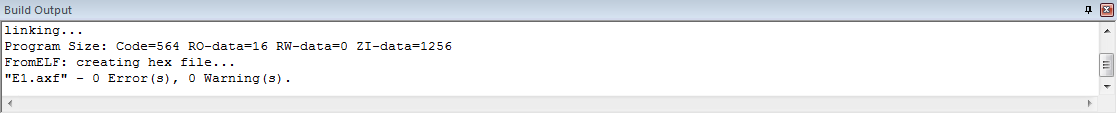


**图11 设置目标选项**

项目Build的过程即是源代码编译、汇编、连接并最终生成目标文件的过程。Build过程可以通过以下三个工具按钮操作实现。

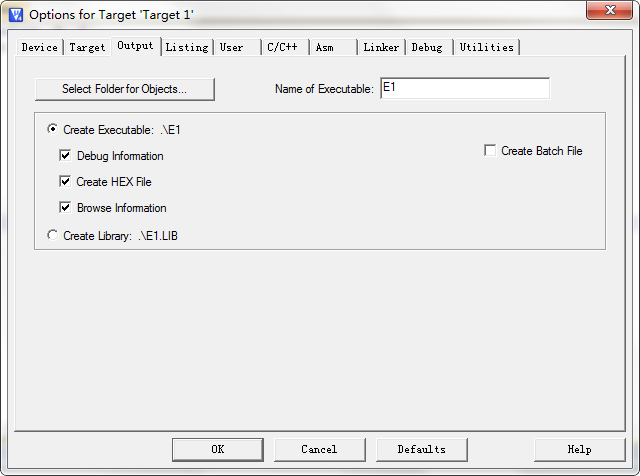
* Translate File -编译或汇编当前源文件。
* Build Target - 编译和汇编所有被修改过的文件，并连接项目。
* Rebuild - 编译和汇编所有文件，无论被修改与否，并连接项目。

在汇编、编译和连接的过程中，µVision会在Build输出窗口显示错误和警告，如图12所示。对于提示的错误或警告，我们可以双击该信息跳转到源代码的对应行查找错误或警告原因。当项目成功完成所有build过程，µVision会显示**0 Error (s), 0 Warnings (s)**。同时，Build输出窗口还会显示程序代码和数据的大小。



**图12 µVision错误警告显示窗口**

我们还可以通过在Options for Target – Output中设置在build过程中自动生成HEX文件，以用于Flash存储器的编程。



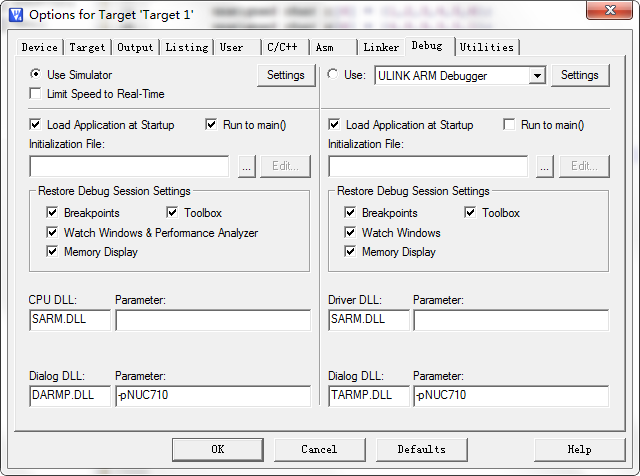
**图13 选择HEX文件输出**

注：HEX全称Intel HEX，是一种文件格式，用以对可编程微控制器、EPROMs和其他类似芯片传送二进制信息。通常，编译器或汇编器将程序源代码转换成机器码并输出到HEX文件。该HEX文件随后被编程器导入，用来将机器码“烧写”或传送到目标系统或ROM中装载并执行。

1. **程序调试**

下面我们学习使用µVision调试器仿真调试之前编写的程序。如前所述，µVision调试器可以配置成仿真器模式或目标调试器模式。本次实验，我们先学习仿真器调试模式，目标调试器模式在下一次结合硬件开发板的实验中学习。通过Build工具条打开Options for Target对话框，切换到Debug标签，按照图14所示设置调试器为仿真器模式。

从Debug工具条选择Start/Stop Debug Session命令按钮进入调试模式。µVision IDE会装载应用，执行启动代码，并按照配置一直执行到main函数入口处暂停。当程序执行暂停后，µVision IDE打开文本编辑器窗口、反汇编窗口以及寄存器窗口、栈调用窗口等调试输出窗口。



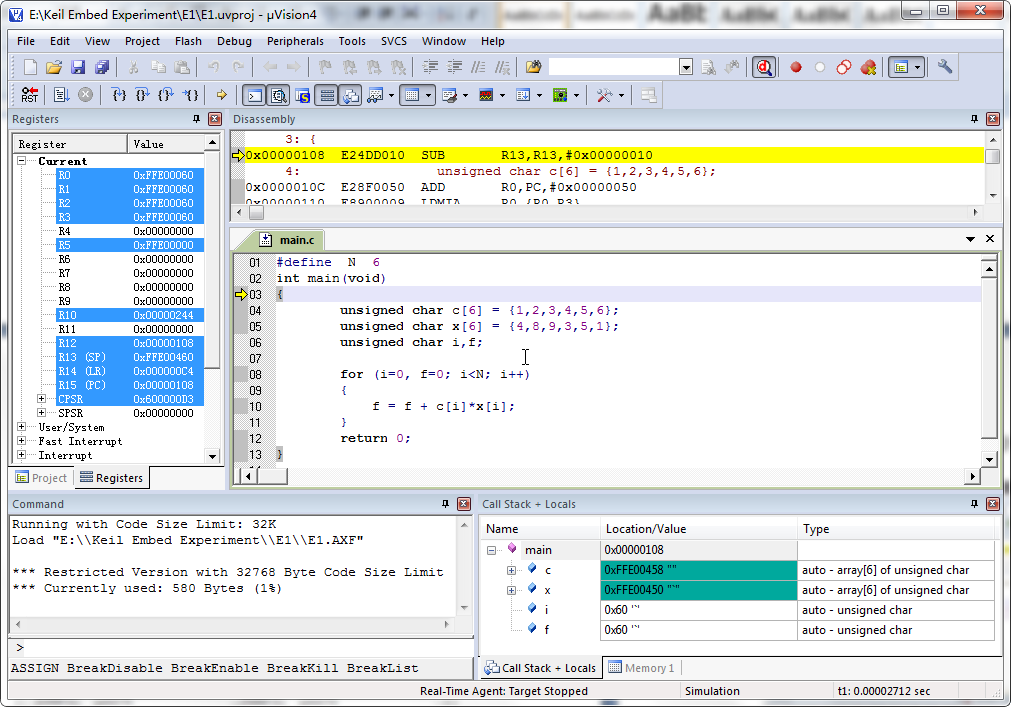
**图14 µVision仿真器模式设置窗口**

µVision IDE进入调式模式后的界面如图15所示。

左边是Register Window，实时显示处理器寄存器中的值。

右边是Disassembly Window和Text Editor Window。其中，上边Disassembly Window，显示C语言等高级语言反汇编后的汇编代码；下边的Text Editor Window，对源程序进行修改或编辑。

右下方是Call Stack Window，查看程序中的变量在栈中的变化情况。



**图15 µVision进入调式模式后的界面**

µVision IDE进入调试模式后，我们可以通过操作Debug工具栏或Debug菜单的调试命令来控制代码的执行。Debug工具栏如图16所示：

C:\Users\Administrator\Desktop\keil截图\15.png

**图16 Debug工具栏**

控制代码执行的常用命令解释如下：

 Run命令：启动执行程序

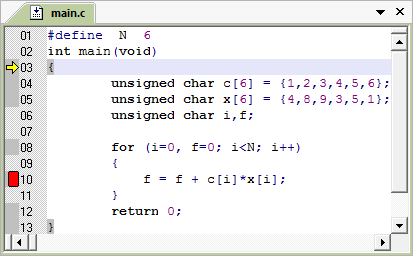
* Stop命令：暂停执行程序

 Reset CPU命令：PC指针复位

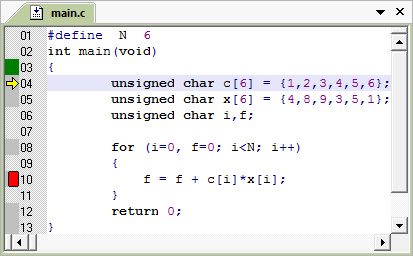
* Step 命令：在程序中单步执行中进入函数内部
* Step Over命令：在程序中单步执行中将函数看做是一条指令
* Step Out命令：从当前函数中跳出进入上一级程序
* 执行到光标行命令：执行程序到当前光标所在代码行

我们还可以通过双击编辑器窗口代码行前面的空白处或Debug –Insert/Remove Breakpoint插入断点，辅助控制程序的执行。图17-19中，代码行前面的红点就是插入的断点，同样也可以双击红色断点取消插入的断点，程序中允许插入多个断点。

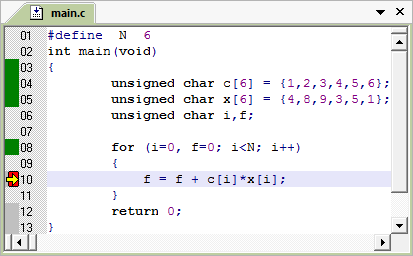
选择Debug按钮进入Debug模式后，源程序从图17中的黄色箭头处开始执行。当点击Debug工具栏上的C:\Users\Administrator\Desktop\keil截图\debug2.png按钮时，源程序会单步向下执行，执行过后的源程序前会变成绿色，如图18所示；在源程序没有执行到断点之前点击Debug工具栏上的Run按钮C:\Users\Administrator\Desktop\keil截图\run.png，程序会自动执行到红色的断点处，如图19所示。



**图17 程序开始单步调试**

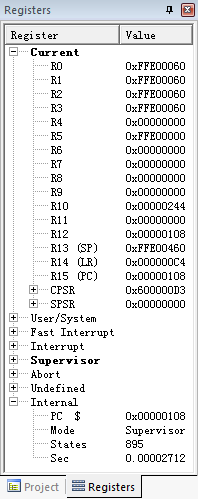


**图18 程序单步调试后**



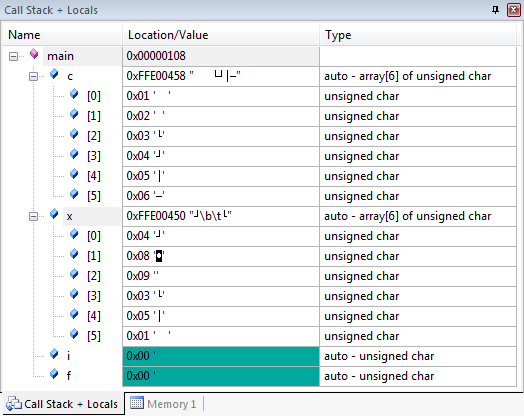
**图19 程序执行到断点处**

程序在执行过程中，相应工作模式下的寄存器的值会发生变化，图20中显示了微控制器寄存器的值。可以通过双击寄存器的值来改变其值或使用F2快捷键选择相应的寄存器后在改变其值。



**图20 寄存器窗口**

可以在Call Stack Window中查看源程序中变量的变化情况，图21反映的是源程序执行到图17中的断点处时，栈中变量的变化情况。此时，数组c[6]和x[6]，变量i和f的初始值都被压入到了栈中，实现对数组和变量的初始化操作。



**图 21 栈显示窗口**

1. W90P710芯片简介

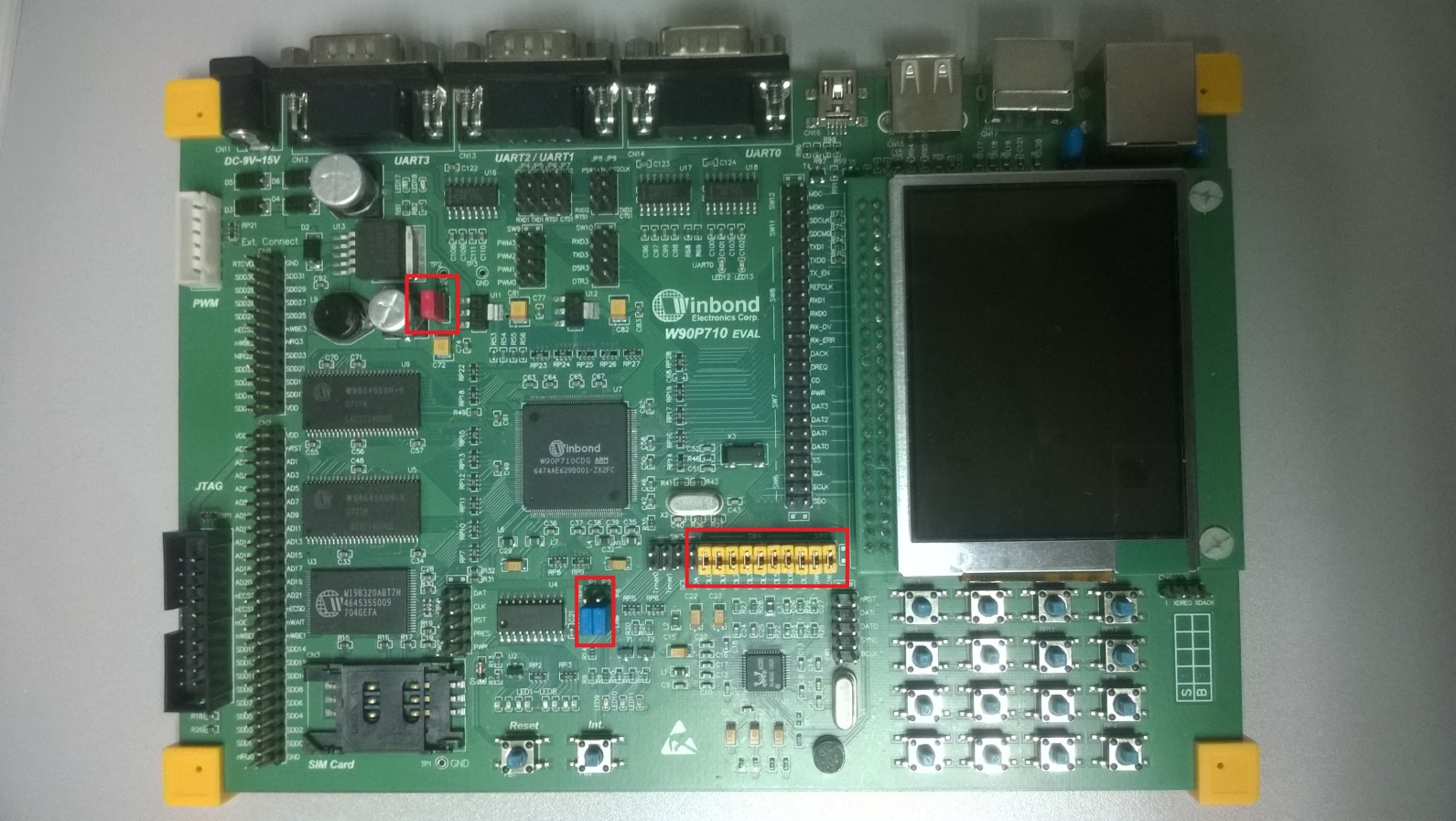
W90P710是华邦电子开发的一款基于ARM7TDMI微处理器内核的SoC芯片。ARM7TDMI微处理器内核是由ARM公司设计的32 位RISC 架构微处理器。W90P710芯片采用0.18um工艺生产，典型工作频率可达80MHz，采用176引脚LQPF封装。2008年，华邦电子分割逻辑产品线后成立新唐科技（Nuvoton Technology Corp.）。之后，W90P710也被称作NUC710。虽然W90P710是一款已逐渐退出市场的芯片产品，但却是一个学习ARM7架构与编程的好平台。W90P710主要的特性与集成外设接口功能如下：

* 片上集成有 4KBytes I-Cache和4KBytes D-Cache，且该存储器可根据实际需求被配置成为 片上RAM使用。
* 片上集成二个带有 8 分频的 24 位定时/计数器、一个看门狗定时器、最多可达71个可编程I/O接口、四组独立的UART串行接口、一个双通道的 GDMA接口和一个高级中断控制器。
* 片上集成一个可配置的LCD控制器，可支持TFT型和低成本的STN型LCD模块。
* 片上集成一个10M/100Mb自适应的以太网MAC层控制器，可实现对以太网通信的支持，降低网络通信的系统成本。
* 片上集成USB1.1主机和设备控制器各一个，二个智能卡主控器。
* 片上集成一个 AC97/I2S控制器，一个SD/SDIO卡主控器和一个PS/2键盘控制器。
* 片上集成一个可灵活配置的外部总线接口（External Bus Interface, EBI）接口。通过 EBI接口可连接SDRAM、ROM、SRAM、Flash存储器以及其他各类 I/O 设备。

关于W90P710芯片更详细的资料请参考附件数据手册中的W90P710CDG文件。

1. W90P710评估板使用说明

本次实验我们选用W90P710评估板作为硬件实验平台，图22为W90P710 评估板的实拍图。整块评估板可以划分为核心功能模块和应用功能模块两部分。其中，核心功能模块主要是围绕W90P710芯片扩展的最小系统，而应用功能模块是通过W90P710芯片的GPIO和EBI接口扩展的应用外设。



**图22 华邦W90P710 评估板实拍图**

核心功能模块以W90P710处理器芯片为核心，外扩16MB SDRAM、4MB Flash、1个JTAG调试口、1个UART调试串口、1个USB主机接口和1个设备接口，组成W90P710评估板的最小系统。评估板核心功能模块的框图如下图23所示，主要由以下各个部分组成：

* MCU：基于 ARM7TDMI内核的华邦W90P710芯片，最高工作频率可达到 80M
* 启动FLASH：评估板最大可支持8MB (4Mx16) FLASH，板载1片 2MBx16 FLASH
* SDRAM：评估板最大可支持32MB (8Mx16x2 banks) SDRAM，板载共有 16MB
* USB：一个 USB1.1主接口和一个USB1.1设备接口
* UART：一个只有TX/RX 信号的UART0端口用于调试工作台通信
* JTAG：14管脚JTAG 作为调试接口



**图 23 华邦 W90P710评估板核心功能模块框图**

如图24所示，应用功能模块通过GPIO和EBI总线扩展了丰富的应用外设接口，主要包括：

* LCD：通过 W90P710的LCD 控制器接口支持AUO 960\*240 TFT 液晶显示屏
* SD卡：通过W90P710的SD接口支持4位模式SD卡
* I2C ：板载一片64KB EEPROM，连接至W90P710的 I2C接口
* 智能卡：两个由W90P710的SMC控制器支持的 7816-3 端口
* 三个由W90P710支持的UART端口（公型）
* PS/2 ：一个由W90P710的PS/2接口支持的PS/2 端口
* RTC：由W90P710的实时钟控制器支持的实时时钟
* 矩阵键盘：由W90P710的KPI控制器支持的2 行 \* 8列键盘
* SPI ：一个由W90P710的SPI总线支持的32M -bit串行FLASH
* Audio：由通过音频解码器ALC-203 和W90P710的AC-97端口支持的音频输入输出功能
* LED调试显示：八个连接至W90P710的EBI bank0端口的LED
* PWM：由W90P710的PWM控制器支持的四通道PWM连接器
* 超时LED显示：两个连接至W90P710定时器输出的LED



**图24 华邦 W90P710评估板应用功能模块框图**

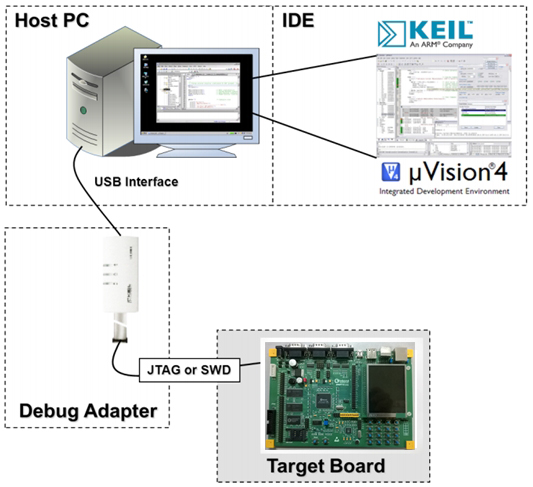
在应用功能模块中，W90P710 除了地址总线引脚和数据总线引脚之外，各个功能模块通过GPIO的引脚来实现相应的功能。由于处理器芯片的引脚总数有限，所以很多部件都是共享某些特定的引脚。如 GPIO[0:4] 是由 GPIO、PWM、UART3 以及 AC-97 来共享，但是某一个特定的时刻，只能有一种模块使用该引脚。W90P710芯片通过相应的寄存器来设定当前时刻由那个模块来使用该 GPIO 引脚。

注意：在使用W90P710评估板之前，需要对其进行跳线设置来配置相应的功能。本次实验中所涉及到的跳线设置在图22中已经用矩形框标出，请按照图示设置。实验中，若要重启系统，请直接按下W90P710评估板下方的Reset按钮。关于W90P710评估板的详细介绍和使用说明请参考附件2：W90P710CDG芯片产品数据手册。

1. MDK-ARM的目标调试功能

在W90P710评估板上开发和调试程序需要使用MDK-ARM的调试器。在进行实验一时，我们主要是使用了MDK-ARM调试器的仿真功能。本次实验，我们学习使用调试器的目标调试功能。使用MDK-ARM的目标调试器调试评估板，需要调试适配器的配合。实验中，我们使用J-LINK调试适配器。整个系统的连接如图25所示，J-LINK的USB端连接在PC机上，将另一端的连接在W90P710的JTAG端口上，等待PC机上的驱动自动安装完成之后，J-LINK就可以正常工作了。

J-LINK调试适配器可以辅助完成：（1）下载目标程序；（2）查看存储器与寄存器；（3）控制程序单步执行；（4）插入多个断点；（5）实时执行程序；（6）烧写Flash存储器。利用J-LINK调试适配器，我们可以像使用仿真器一样实现对在评估板运行程序的控制与状态查看。关于对J-LINK调试的设置，将在后续的实验过程中学习。



**图25 调试环境连接图**

1. LED灯闪烁实验

**【实验目的】**

学会使用Keil MDK-ARM进行ARM内核应用处理器的I/O接口编程与调试方法。

**【实验设备】**

1. 硬件：W90P710实验系统、PC机、J-LINK仿真器
2. 软件：PC机操作系统、Keil MDK-ARM 4.22

**【实验内容及要求】**

在W90P710评估板平台编写程序，控制八个发光二极管逐一循环点亮。

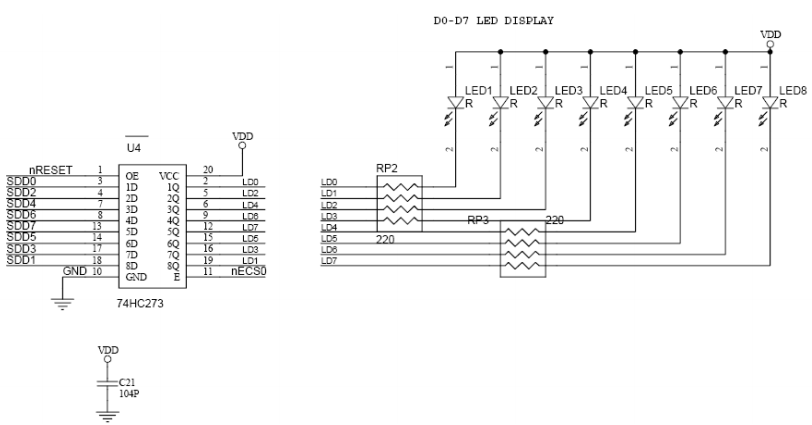
**【实验原理】**

W90P710评估实验板上LED灯硬件电路原理图如图26所示。CPU数据总线的低8位连接到锁存器74HC273的信号输入端，CPU 外部总线接口（EBI）的nECS0片选信号连接到锁存器的锁存控制信号输入端。这样当CPU访问BANK0地址空间内的任一地址时，数据总线低8位的数据将被锁存器锁存后输出，作为控制8个LED 灯的发光状态的控制信号。

本实验中，设置CPU外部总线BANK0的基地址为0x78000000。因此，如向该基地址写数据0x00时，8个LED 灯都将点亮；而写数据0xFF时，则8个LED灯都将熄灭。有关应用处理器相关寄存器的详细设置，请参考附件2：W90P710CDG芯片产品数据手册。有关评估板详细的电路连接与设置，请参考附件3：W90P710评估板硬件应用说明。

**【实验步骤】**

1. 打开Keil，点击Project->New uVision Project，新建名称为led的工程，并将工程保存在实验目录的新建LEDBlinky文件夹中。同时，将实验提供文件中的Ext\_SDRAM.ini文件拷贝到LEDBlinky文件夹下。
2. 在弹出选择ARM芯片窗口，选择Nuvoton系列下的NUC710芯片，点击“OK”，提示“将NUC710启动代码添加到工程”，请选择“是”。
3. 点击File->New，新建名称为led.c的文件，并将led.c文件保存在LEDBlinky文件夹中。将LED灯闪烁实验的C语言程序源代码通过编辑器输入到led.c文件中，并保存。C语言源代码如下所示。



**图26 LED控制电路原理图**

//LED灯闪烁程序C代码

#define WriteReg(Address,Value) \*(unsigned int volatile \*)(Address) = Value

#define U32 unsigned int

#define REG\_EXT0CON 0xFFF01018

#define EBILED\_ADDRESS 0x78000000

//子函数声明

void EBILedSet(U32 Value);

void Delay(U32 t);

//主函数

int main(void)

{

U32 i;

WriteReg(REG\_EXT0CON, 0xF0078003);

while(1)

{

for(i = 0 ;i < 8; i++)

{

EBILedSet(0x1<<i);

Delay(1000000);

}

}

return 0;

}

//延时

void Delay(U32 t)

{

do

{

t--;

}while(t);

}

//控制LED 灯的亮灭

void EBILedSet(U32 Value)

{

WriteReg(EBILED\_ADDRESS, ~Value);

}

1. 右键选择Project窗口下的Target1->Source Group1，在弹出的窗口中选择Add Files to Group，将led.c文件添加到工程中。
2. 点击Keil工具栏的Target Option按钮，在Options for Target窗口中设置J-LINK调试环境。

**Target标签页**设置如图27所示。W90P710芯片的指令地址空间从0x00000000开始。而评估板板载SDRAM存储器大小为16MB。其中，分配8MB给只读存储器空间，用于存储指令；另外8MB分配给随机读写存储器空间，用于存储数据和栈等。因此，设置程序编译的地址映射如图30所示。

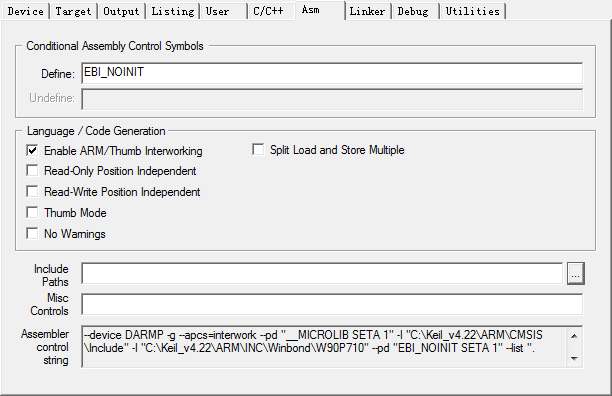
**ASM标签页**设置与程序汇编有关的选项。如图28所示，在Define框内填写EBI\_NOINIT，避免启动代码中对EBI的配置和相关配置冲突。

**Debug标签页**设置与程序调试器有关的设置。请按照图29所示，设置选择目标调试J-LINK/J-TRACE,选择初始化文件（Initialization File）为使用项目目录下的Ext\_SDRAM.ini文件配置评估板。

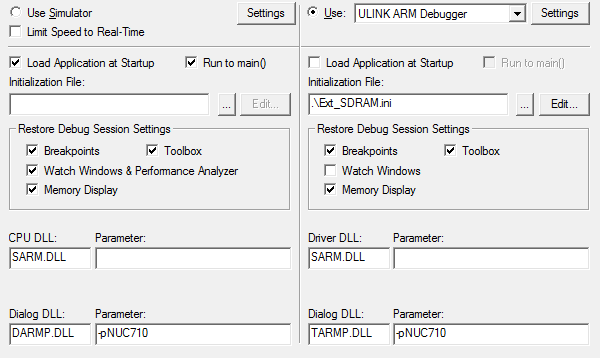
**Utilities标签页**设置与程序下载到Flash存储器有关的选项。请按照如图30所示选择，并点击Setting按钮，如图31所示，删除在Programming Algorithm框中的Flash芯片，点击OK保存。



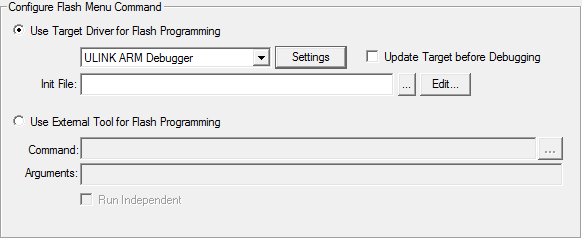
**图27 Target标签页设置**



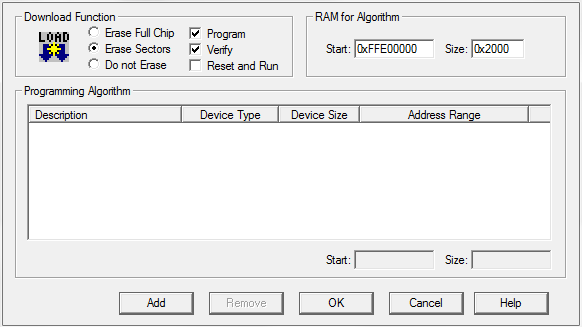
**图28 ASM标签页设置**



**图29 Debug标签页设置**



**图30 Utilities标签页设置**



**图31 Utilities标签页Settings窗口**

1. 编译-连接工程，生成后缀名.axf的镜像文件。
2. 修改并分析调试配置初始化文件Ext\_SDRAM.ini。该文件的部分内容如下所示。**最重要的是**要将装载的axf文件替换成自己工程产生的axf文件。axf文件名称与工程名称相同，此处为led.axf。

// <o2.0> EBI Setup

// <o2.1> Enable Flash Programming

// <e2.2> Download Program

FUNC void Download (void) {

if (Setup & (1 << 2)) {

// <s0.80> Command for Loading***（替换自己的.axf）***

exec("LOAD ***led.axf*** INCREMENTAL");

}

}

// </e>

// <e0.3> Setup Program Counter to Entry Point

// <o1> Program Entry Point <0x0-0xFFFFFFFF>

// </e>

// <o0.4> Execute Program untill Main Function

Setup = 0x1D;

Entry = 0x00000000;

EBI\_Setup(); //设置外部总线

EnableFlashProgramming(); //Flash编程使能

Download(); //程序代码下载至目标板SDRAM中

PC\_Setup(); //设置PC指针为0x00000000，开始执行

GoMain(); //一直执行到main函数入口处

1. 将J-LINK仿真器的USB端连接到PC上，另一端连接到评估板的JTAG端口上，然后打开电源。
2. 点击Keil软件的Debug按钮，J-LINK会自动地将.axf后缀的镜像文件下载到SDRAM中。选择Run按钮或者单步调试按钮，观察程序的执行和LED流水灯效果。
3. 阵列键盘扫描实验

**【实验目的】**

1. 掌握W90P710 键盘控制器的工作原理与扫描方法。
2. 掌握I/O 端口扫描键盘的寄存器设定及程序设计方法。

**【实验设备】**

1. 硬件：W90P710实验系统、PC机、J-LINK仿真器
2. 软件：PC机操作系统、Keil MDK-ARM 4.22

**【实验内容及要求】**

编写基于中断的KPI 控制程序。首先接好键盘跳线器，并设置键盘中断及相应GPIO 接口，进入中断后能用键盘控制LED灯的显示。

每个按键从0-F依次编号，每按一个键，LED灯按照相应的键值显示。比如：按下BP7，相应的编号或者键值是0x6，对应的二进制数为0110，则LED灯低4位中的第2位和第3位亮，即LED2和LED3亮，其余LED全部不亮。

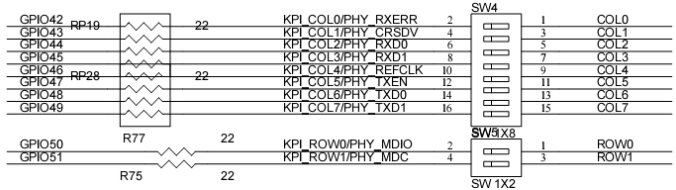
**【实验原理】**

在键盘中按键数量较多时，为了减少I/O口的占用，通常将按键排列成矩阵形式，在矩阵键盘中每条水平线和垂直线在交叉处不直接相连，而是通过一个按键相连接，这样在由N条水平线和M条垂直线最多可以有N \*M 个按键，大大的减少了对于芯片I/O的占用。

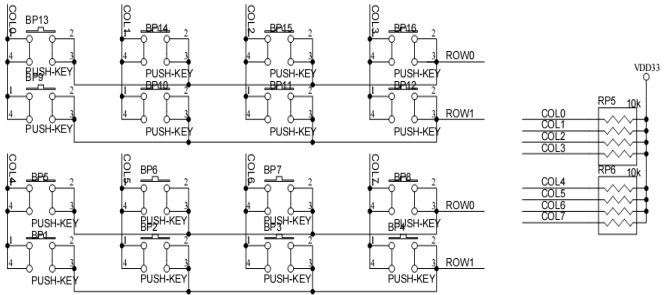
矩阵键盘由行信号和列信号组成矩阵。每个按键连接到行信号和列信号的交叉点。通过矩阵键盘控制器输出行信号扫描信号（低电平）顺序逐行的对行扫描，读取列信号线的状态。

如果有按键按下，对应的行输出扫描信号会拉低列输入信号，触发键盘中断。键盘中断服务程序会将键盘状态寄存器的值读出，判断哪个按键被按下。

W90P710评估板线路上采用GPIO50 和 GPIO51 输出行扫描线，键盘的列状态由GPIO42~49 输入，实验接线图见图32。



**图32(a) GPIO 口连接线路图**



**图32(b) 2x8 键盘线路图**

W90P710评估板的键盘设计为2行输出扫描，8列输入的方式。其键盘的模式可以通过相应的寄存器设定(最大支持16x8)，例如设定为2x8 键盘。键值的判断不需要设计者另行判断，系统提供了相应的寄存器存储按键值，只需要读键盘状态寄存器就可以知道按键值。 当有1个或者2个键值按下的时候，键盘的扫描顺序是 ROW0 CL0-1-2-3-4-5-6-7，ROW1 CL0-1-2-3-4-5-6-7 …ROW16 CL0-1-2-3-4-5-6-7。因此，两个键同时按下的时候，只会读到较小的键值。

实验中，键盘键值的判断采用的是中断模式，需要程序打开键盘控制器中断，并设定优先级。在中断服务程序中，读取按键键值并点亮对应的LED灯。注意：W90P710评估板上的键盘的第一行和第三行对应芯片键盘控制器的ROW0扫描行，第二行和第四行对应芯片键盘控制器的ROW1扫描行，而不是直接对应的4x4矩阵键盘顺序划分。因此，读入键值后需要转换以对应物理按键，并控制相应的LED灯。

**【实验步骤】**

实验步骤与“LED灯闪烁实验”步骤一样，实验中的C代码如下。

//矩阵键盘扫描C代码

#define u32 unsigned int

#define REG\_GPIO\_CFG2 0xFFF83020

#define REG\_AIC\_SCR29 0xFFF82074

#define REG\_AIC\_MECR 0xFFF82120

#define REG\_KPICONF 0xFFF88000

#define REG\_KPISTATUS 0xFFF8800C

#define EBILED\_ADDRESS 0x78000000

#define REG\_EXT0CON 0xFFF01018

#define EBI\_EXT0\_VALUE 0xF0078003

#define WriteReg(Address,Value) \*(unsigned int volatile \*)(Address) = Value

#define ReadReg(Address) \*(unsigned int volatile \*)(Address)

void KPIInit(void);

int main (void){

WriteReg(REG\_EXT0CON, EBI\_EXT0\_VALUE);

KPIInit();

while(1);

return 0;

}

void KPIInit(void){

WriteReg(REG\_GPIO\_CFG2, 0x000aaaaa); //将GPIO设置为KPI模式，2X8

WriteReg(REG\_AIC\_SCR29, 0x00000045); //将KPI 中断设置为高电平有效，优先级为5

WriteReg(REG\_AIC\_MECR, 0x20000000);

WriteReg(REG\_KPICONF, 0x00142fff);

}

//KPI中断服务子程序

\_\_irq void IRQ\_Handler(void){

u32 KeyValue, temp;

temp = ReadReg(REG\_KPISTATUS);

temp &= 0x0000000f;

if(temp > 0xB){

KeyValue = temp - 0xC;

}

else if(temp < 0x4){

KeyValue = temp + 0xC;

}

else{

KeyValue = temp;

}

WriteReg(EBILED\_ADDRESS, ~KeyValue);

}

1. LCD显示实验

**【实验目的】**

1. 掌握LCD 液晶模块的基本原理和功能。
2. 掌握LCD 液晶模块和微控制器的硬件接口及程序设计方法。

**【实验设备】**

1. 硬件：W90P710实验系统、PC机、J-LINK仿真器
2. 软件：PC机操作系统、Keil MDK-ARM 4.22

**【实验内容及要求】**

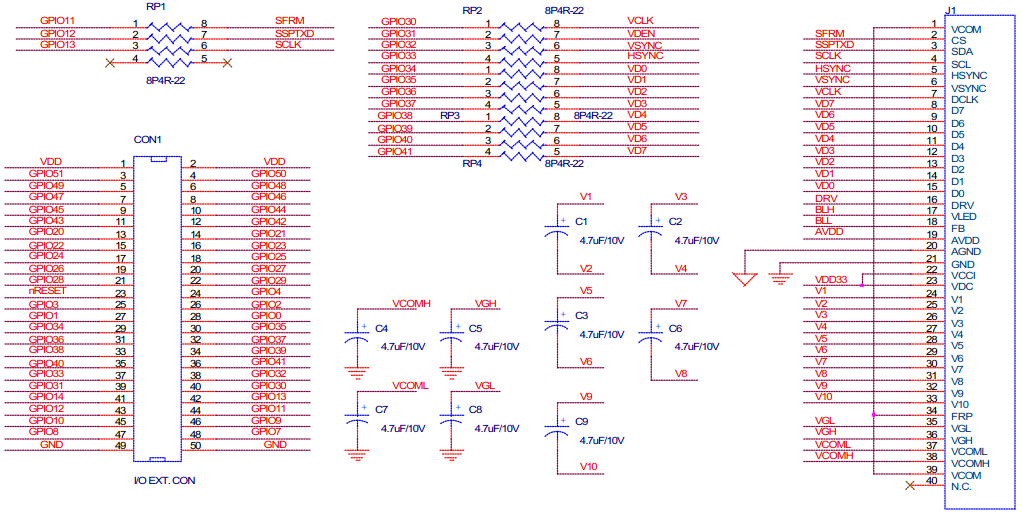
编写LCD 程序：

1. 在LCD上显示一张图片；
2. 在LCD上显示多张图片，加入键盘按键功能，制成电子相册，用键盘按键翻页循环显示；
3. 在LCD上显示一张动态图片（GIF图）。

**【实验原理】**

W90P710 评估板的LCD 控制器的主要目的是将VIDEO/OSD 的原始图像数据显示到外部液晶屏上。LCD控制器可以支持AUO的TFT屏，LG 的 TFT 屏，CASIO 的TFT 屏，实验中使用的是AUO 3.0" TFT A030DL01（960x240，8位数据总线）数字式TFT液晶屏。此外，LCD控制器还能够连接到STN LCD 和电视编码器上，并且支持VIDEO和OSD 两种显示模式。这两种显示模式的设定由不同的寄存器控制，通过设定相应寄存器的值可以实现不同的模式和功能。

AUO 3.0" TFT A030DL01 液晶屏由单个像素点排列组成，向这些像素点配置不同的值，可以显示不同的颜色。但是，由于AUO TFT液晶屏内部没有缓存，需要在SDRAM中开辟一片缓冲区存储一维数组形式的原始图像数据。然后，将存储原始图像数据的SDRAM 缓冲区的首地址赋值给LCD控制器中的FIFO1SDADR寄存器。接着，原始图像数据在时钟脉冲的作用下依次输入到LCD控制器的FIFO1中，直至FIFO1被填满才将FIFO1中的数据输出到LCD颜色发生器。最后，LCD颜色发生器将图像显示到评估板上的液晶屏中。如果在液晶屏上定位显示图片或其他信息，需要计算出所要显示的内容在液晶屏上的位置。然后，在新建的SDRAM缓冲区中存入需要显示的内容即可。LCD硬件连接图如图33所示。



**图33 LCD连接原理图**

LCD模块由 40 个引脚与外界电路相连，由于本身没有寄存器，需要由 W90P710中的LCD 控制器提供相应的寄存器，控制选择对应的液晶类型及显示形式、FIFO 使用、颜色显示模式和时序设置。

LCDCON寄存器比较重要，需要根据实际所使用的硬件，对控制寄存器进行配置。本次实验使用16 bpp 模式、真彩型TFT LCD、没有使用查找表功能，分别使能相应的位实现LCDCON寄存器初始化操作。

**【实验步骤】**

实验步骤与“LED灯闪烁实验”类似，但有两个不同点：在步骤3中不需要新建源文件，直接将文件夹“lcd显示所需文件”中提供的实验文件拷贝到工程新建文件夹中；并将所有文件添加到工程目录中，完成任务a。

然后修改lcd.c主程序，加入键盘按键功能，增加图片，制成电子相册，用键盘按键翻页循环显示完成任务b。步骤如下：

* 准备好几幅图片，320\*240像素，用Image2Lcd工具转换为16位真彩色的\*.c格式文件，并添加到工程中；
* 根据上次实验在主程序中添加按键功能；
* 根据不同键值切换显示图片。

最后完成任务c，连续显示一组动态图片。

**注意事项：如果程序下载不成功，可能是由于W90P710.s文件被损坏，这时先删除原工程中的W90P710.s，再到一个可以正常显示图片的工程下拷贝一个W90P710.s文件，重新添加到该工程中。**

**参考文献：**

1. Getting Started, Creating Applications with µVision®4
2. 嵌入式计算系统设计原理，沃尔夫，李仁发，机械工业出版社，2009年6月.

**附件**

[1] Getting Started, Creating Applications with μVision®4

[2] W90P710 32bit ARM Microcontroller Product Data Sheet

[3] W90P710 Eval. Board Hardware Application Note

[4] W90P710 Schematics

[5] A030DL01 3.0" TFT LCD Module Spec

第二章 嵌入式交叉编译环境的建立和使用

1. **实验目的**

1、熟悉和掌握在PC环境中生成和使用嵌入式交叉编译工具。

2、熟悉和掌握在PC环境中使用调试器对运行在嵌入式平台上的程序进行调试。

1. **实验硬件需求**：

1、w90p710开发板一套（电源，串口）。

2、J-Link调试器。

3、一台PC（Windows XP或者Win7）。

1. **实验软件需求：**

实验前会得到如**图2- 1**所示的软件工具包：



**图2- 1 实验所需软件工具包**

其中每个文件夹的功用如下：

cygwin：cygwin的离线下载安装包目录；

demo：实验中用到的示例程序目录；

software：实验中用到的软件安装程序所在目录；

w90p710内核代码：实验中调试开发板内核所需的代码目录；

交叉编译工具链：实验中搭建ARM开发环境所需的工具和源码所在目录；

1. **实验内容**：

GCC（GNU Compiler Collection，GNU编译器集合）是一套由GNU工程开发的支持多种编程语言的编译器，其不仅应用十分广泛而且源代码开源，可以移植到多种硬件平台。本实验首先通过下载GNU官网提供的生成GCC所需的源代码，在Linux环境中编译生成ARM编译开发工具链，然后使用编译得到的工具链生成可以在ARM平台中运行的程序。

GDB（the GNU Project Debugger）是GNU开源组织发布的一个强大的UNIX/Linux下的程序调试工具，用于实时的跟踪程序运行，观察或者修改程序运行情况，本实验将借助J-Link硬件调试器在w90p710开发板上运行和调试使用交叉编译工具生成的程序。

本实验主要由以下五部分组成：

1、安装和配置Cygwin环境。

2、交叉编译生成ARM开发工具链。

3、使用ARM开发工具链编译示例程序。

4、搭建GDB调试环境。

5、使用GDB工具对运行在w90p710开发板上的程序进行在线调试。

***注意：实验2和实验3中有底纹的字符是要在软件界面中手动输入的命令。***

1. **实验具体步骤**：

1、查看Cygwin是否安装：

首先，在桌面上有Cygwin的快捷方式，如**图2-** 所示：



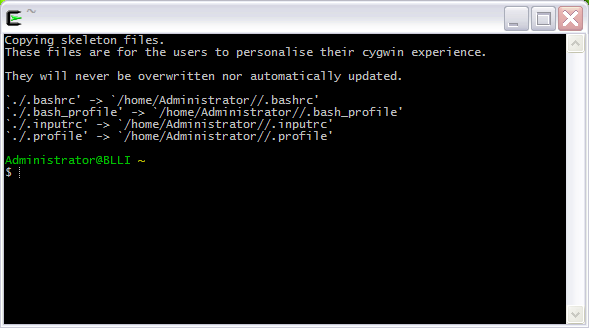
**图2- 2 Cygwin的快捷方式**

其次，在D盘的cygwin目录中有如**图2-**所示文件，该目录便是Cygwin的工作目录，同时该目录也是Windows系统中的一个普通目录，可以使用Windows系统中的拷贝和新建文件夹等常用操作，后续实验中提到的文件复制或者拷贝等操作，便是指Windows系统中常用的文件操作（ctrl+c，ctrl+v等）。



**图2-3 Cygwin的工作目录**

最后，双击桌面的Cygwin termial快捷方式打开Cygwin终端，出现如**图2-** 所示窗口：



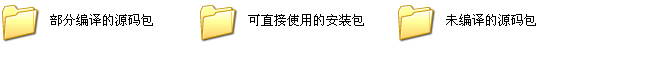
**图2- 4 Cygwin字符终端**

第一次打开时会根据当前PC的用户名，在Cygwin工作目录的home文件夹下新建一个同样名字的文件夹，此时需要同学在该home目录下新建一个名字为root的文件夹，***本实验以后的默认工作目录就是该home目录下面的root文件夹（d:\cgywin\home\root），以后本文提到的“root目录”均指的是该文件夹。***

2、交叉编译生成ARM开发工具链

1）编译安装ARM开发工具链：

编译工具链所需的工具在**图2- 1**所示的“交叉编译工具链”文件夹下。打开该文件夹可以发现有以下三个文件夹，如**图2-** 所示：



**图2- 5交叉编译工具链目录**

本实验使用的是“部分编译的源码包”下的文件。首先将该目录下面的四个压缩包拷贝到root目录下，然后在Cygwin终端使用tar命令解压这四个压缩包，使用命令如下：

cd /home/root

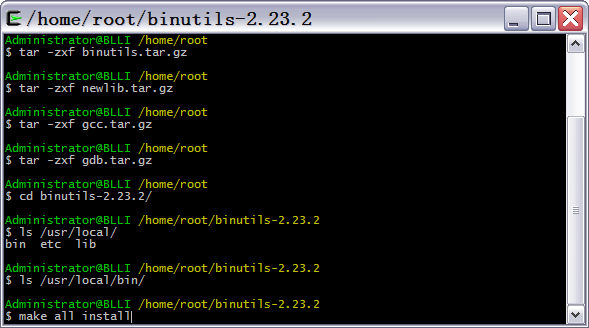
tar–zxf binutils.tar.gz

tar–zxf newlib.tar.gz

tar–zxf gcc.tar.gz

tar–zxf gdb.tar.gz

如**图2-** 所示：



**图2- 6解压并编译binutils**

在执行编译安装过程前，需要确认Cygwin工作目录中的/usr/local/bin目录是空目录，此时可以通过打开Windows平台中对应路径D:\cygwin\usr\local\bin进行查看，也可以在Cygwin终端中使用命令：

ls /usr/local/bin

查看该目录，如**图2-** 所示。如果ls命令没有返回任何信息，则可以确认该目录是个空目录，否则ls命令将返回该目录下存在的文件。此时如果该目录非空则手动删除该目录下所有文件。

a）编译安装binutils

在Cygwin终端中依次输入下列命令：

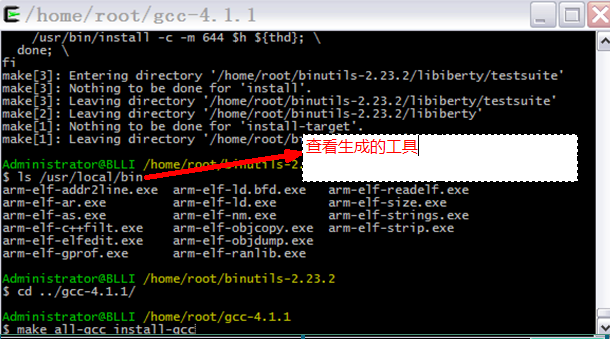
cd /home/root/binutils-2.23.2/

make

make install

完成对binutils的编译（make）和安装（install），其安装路径指定的是/usr/local/bin。

此时再查看/usr/local/bin目录，发现已经生成了**图2-**中所示工具：



**图2-7 binutils生成的工具**

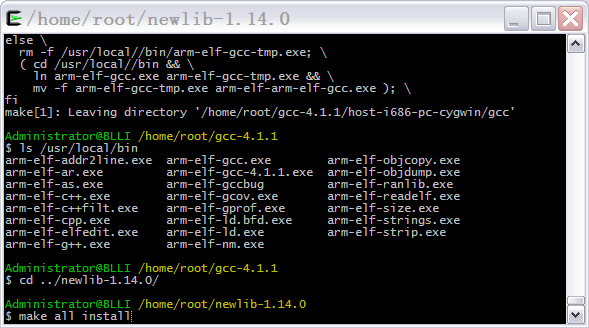
b）预编译和安装gcc

整个gcc的编译可以分为两个阶段：第一个阶段是不依赖于额外库文件的编译和安装，也称为预编译，预编译生成的gcc工具链使用时不能自动识别一些常用的函数，如malloc函数；第二个阶段是配合newlib库的编译，newlib库是一个面向嵌入式系统的C运行库，可移植性强，具有可重入特性、功能完备等特点，已广泛应用于各种嵌入式系统中，编译gcc时加入newlib库的支持可以给嵌入式开发者带来极大的方便。此时在Cygwin终端执行以下命令：

cd /home/root/gcc-4.1.1/

make all-gcc install-gcc

执行后再查看/usr/local/bin目录，发现此时又生成了新的工具，如**图2-**所示：



**图2-8 预编译gcc生成的工具**

c）编译安装gcc所依赖的lib库文件

在Cygwin终端执行以下命令：

cd /home/root/newlib-1.14.0/

make

make install

完成对lib包的编译和安装。

d）编译安装支持newlib库函数的gcc工具链

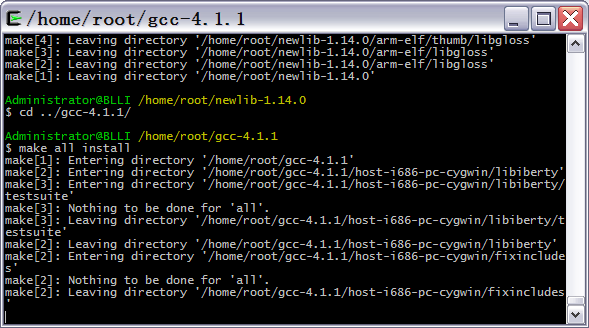
在newlib库编译安装好后，在Cygwin终端执行以下命令：

cd /home/root/gcc-4.1.1/

make

make install

完成对gcc工具链的完整编译和安装，如**图2-** 所示。



**图2- 9 完全编译gcc**

e )编译安装gdb

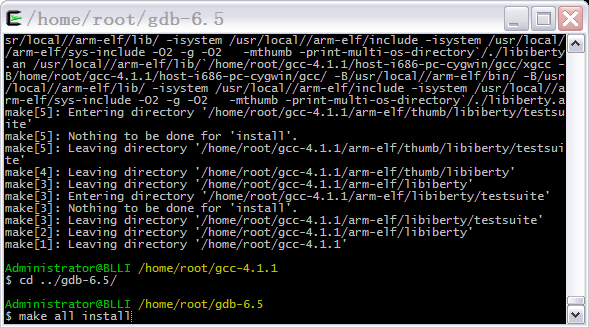
在Cygwin终端执行以下命令：

cd /home/root/gdb-6.5/

make

make install

完成对gdb工具链的编译和安装，如**图2- 2**所示。

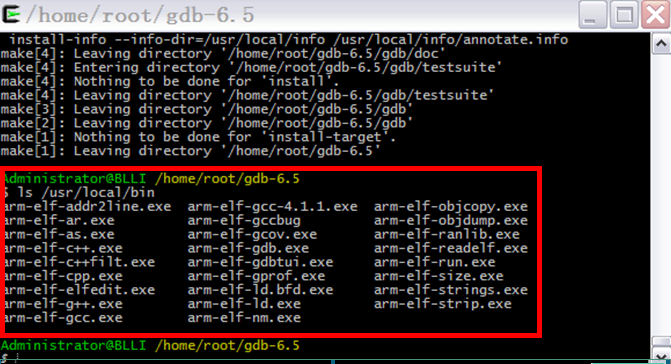


**图2- 2 编译gdb**

至此整个交叉编译环境编译安装完毕。**图2-** 中剩余两个文件夹的作用详见附录2-1。

2）查看所有工具是否完成编译和安装

如果/usr/local/bin目录中出现了如**图2- 3**所示工具，则说明编译安装完成：

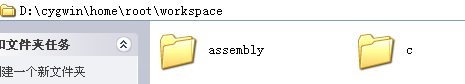


**图2- 3 完全编译后的工具链**

3、使用ARM开发工具链编译示例程序

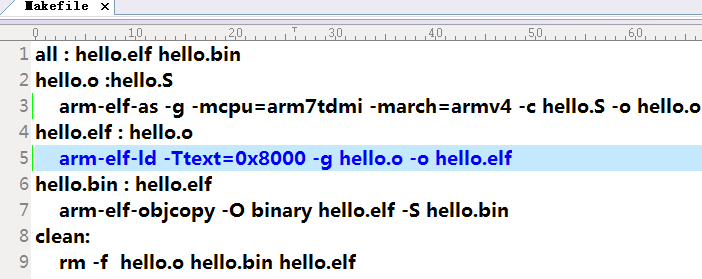
1）使用arm-elf工具链编译示例程序

在root目录下新建一个workspace文件夹，并将**图2- 1**所示软件工具包中demo目录下的assembly和c两个文件夹拷贝到workspace目录下，如**图2- 4**所示：



**图2- 4 代码实例包目录**

assembly文件夹里面是一个简单的汇编程序hello.S（见附录2-2），里面使用arm-elf工具链对该程序进行编译和链接，其中使用的Makefile内容如**图2-** 所示：



**图2- 13 示例的Makefile**

Makefile中首先使用arm-elf-as命令编译汇编程序hello.S，生成未连接的目标文件hello.o，然后使用arm-elf-ld命令将hello.o链接成elf格式的可执行文件hello.elf，最后使用arm-elf-objcopy命令提取hello.elf中的执行代码，生成hello.bin文件。

这里简单介绍各个命令中用到的参数：

-g：表示编译时加入用于调试的符号信息；

-c：表示只将程序编译成目标文件，不进行链接；

-o：表示输出目标；

-mcpu：表示目标CPU型号，因为后续实验要用到的w90p710开发板的CPU型号是arm7tdmi，所以如**图2-** 所示，编译时指定arm7tdmi为mcpu的值。如果所用编译器内部有对相应CPU型号进行过限定或优化，则编译时将调用该限定或优化；如果所用编译器没有对相应CPU型号做出过处理，则该参数无效；

-march：表示目标CPU所属架构，用来限定编译时所用的指令集，因为arm7型号的CPU属于armv4架构，所以如**图2-** 所示，编译时指定armv4为march的值；

-Ttext：表示链接时的地址，也是程序在开发板运行时的内存地址，所以必须是开发板的有效内存物理地址，这里选取的是0x8000；

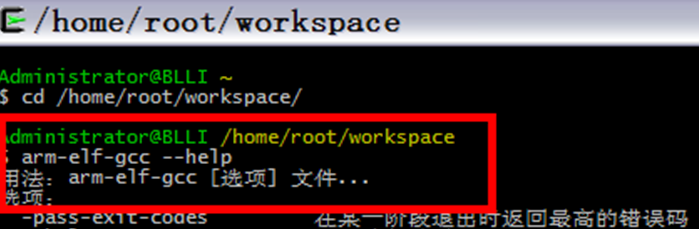
-O binary：表示输出格式是二进制格式；

-S：表示去除elf中所有的调试信息。

可以通过“命令+--help”来获知更加详细的参数信息，例如在Cygwin终端输入以下命令：

arm-elf-gcc --help

可以查看arm-elf-gcc命令的所有参数及其介绍，如图2- 54所示：

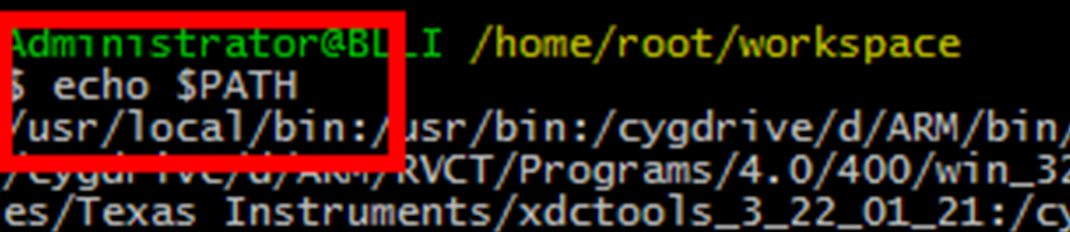


**图2- 64 查看命令参数**

在运行“make”命令编译程序之前，首先确认系统的PATH路径中包含先前编译成功的工具链路径:/usr/local/bin，在Cygwin终端输入以下命令：

echo $PATH

如果输出的信息中包含/usr/local/bin目录，则说明系统的PATH变量包含了先前编译成功的工具链路径，如**图2-** 5所示：



**图2- 15 查看系统默认查找路径PATH变量**

如果PATH变量里面没有预先生成的工具链路径，需要手动添加该路径到PATH中，添加方式可参照附录2-1所示。然后在Cygwin终端输入命令：

cd /home/root/workspace/assembly

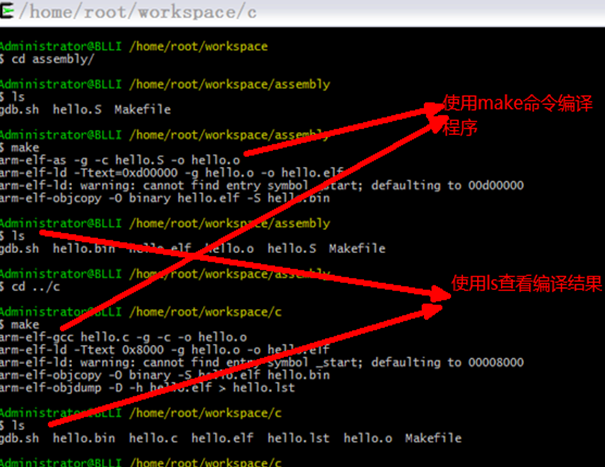
make

编译汇编语言，并生成包含符号信息的hello.elf和可执行二进制文件hello.bin。同理，继续在Cygwin终端中输入命令：

cd /home/root/workspace/c

make

编译示例的c程序，生成包含符号信息的hello.elf和可执行二进制文件hello.bin，其过程如**图2-**6所示：



**图2-16 使用make编译实例程序**

2）查看程序是否编译成功

如**图2-**6所示，可以通过ls命令查看/home/root/workspace的assembly和c目录下是否生成了elf和bin格式的可执行程序，从而确定是否编译成功。

4、搭建GDB调试环境

在将编译生成的程序下载到开发板上用GDB进行调试之前，首先需要安装一些工具软件。

1）安装J-Link驱动

J-Link是一款支持GNU GDB协议的硬件调试器，使用前需要安装对应驱动，打开**图2- 1**所示软件工具包中的software目录，双击Setup\_J-linkARM\_V426.exe安装J-Link驱动。

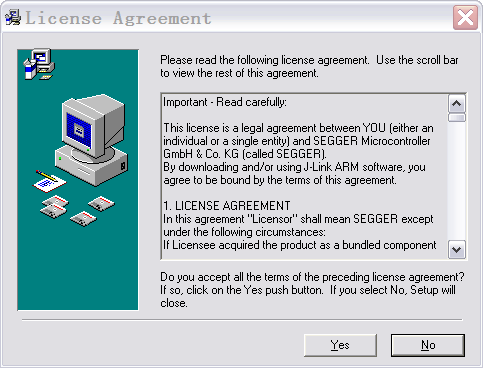
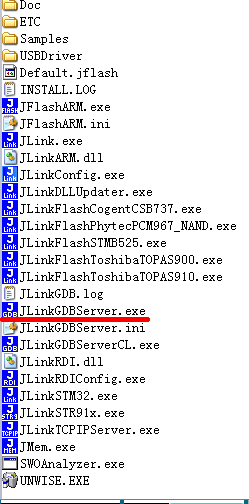


图2- 17 安装J-Link驱动

在图2- 7中选择按钮“Yes”进行安装，后面一直选择“Next”选项即可。

安装完成后在安装路径（默认C:\Program Files\SEGGER\J-linkARM\_V426）下面可以找到对应的驱动程序，如**图2-** 8所示：



**图2- 18 J-Link驱动安装目录**

其中JLinkGDBServer.exe是后续要用到的程序，为了方便起见可以在桌面建立一个指向其的一个快捷方式。

2）建立SecureCRT快捷方式

SecureCRT是一个串口控制程序，其可以接收和发送串口数据，**图2- 1**所示软件工具包中software目录下SecureCRT文件中是绿色免安装版的程序，为了方便起见，可以直接在桌面建立一个指向SecureCRT文件夹里SecureCRT.exe的快捷方式。

5、使用GDB工具在线调试运行在w90p710上的程序

w90p710是新唐科技（Nuvonton）公司生产的基于ARM7TDMI核心的开发板，其CPU最高主频为80MHz，在板有16M 的内存和4M的NOR Flash，下面介绍如何使用J-Link调试器对运行在w90p710上的程序进行调试。

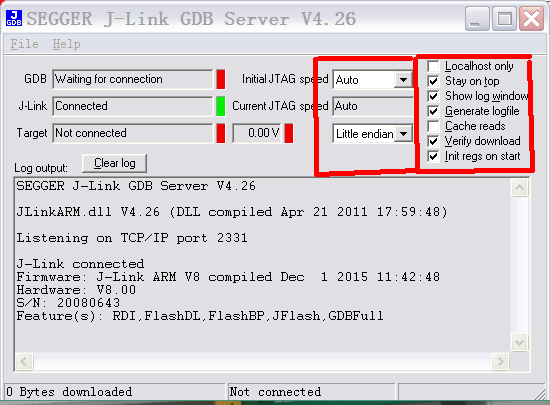
1）首先使用J-Link和串口线正确连接PC和开发板，如**图2-** 9**错误！未找到引用源。**所示：



**图2- 19使用J-Link和串口线连接PC和开发板**

J-Link的另外一端接PC的usb口，串口线的另外一端接PC的串口。

2）双击JLinkGDBServer.exe打开J-Link GDB Server，并进行如**图2-** 所示配置：

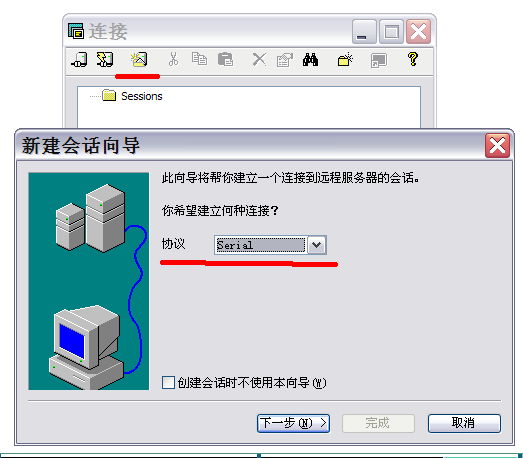


**图2- 20 配置gdbserver**

3）打开secureCRT软件，依次点击**图2-** 中菜单栏的“文件→连接→新建连接”，并进行如**图2-** 所示配置：

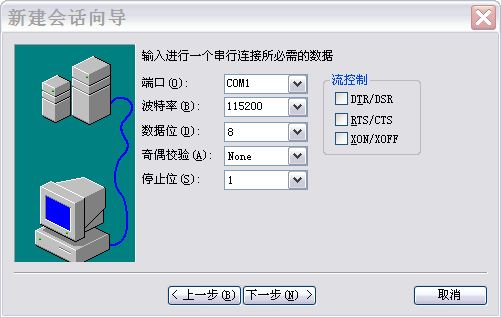


**图2- 21 SecureCRT界面**

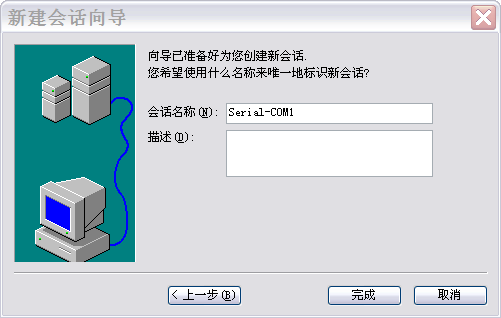


**图2- 22 新建串口终端**

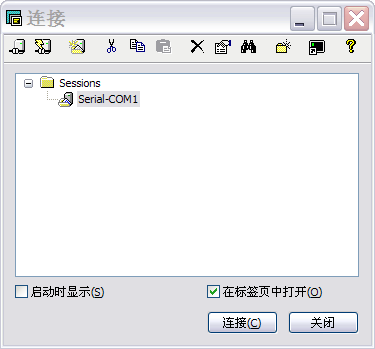
在**图2-** 的下拉菜单中选择serial。并在接下来的弹窗中按照**图2-** 3和**图2-** 4所示进行配置。



**图2- 23 配置串口终端参数**



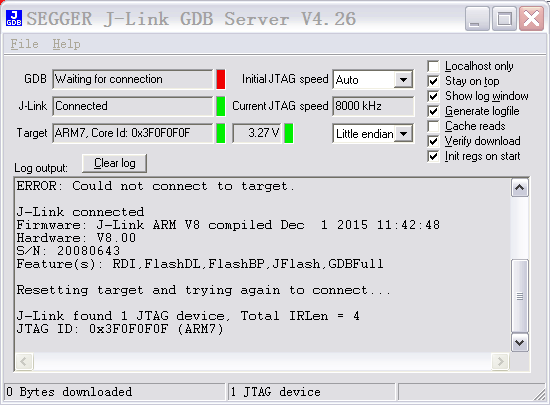
**图2- 24 确定串口终端名称**



**图2- 25 完成新建串口终端**

在**图2-** 5中点击按钮“连接”，完成串口配置。

4）此时插上w90p710电源，查看J-Link gdbserver窗口可以发现J-Link已经可以正确识别设备，如**图2-** 6所示：



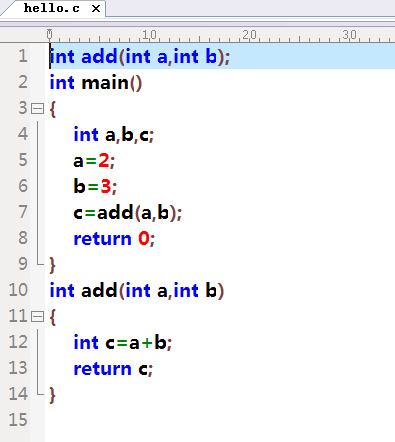
**图2- 26 J-Link对arm7芯片的识别**

5）使用arm-elf-gdb工具进行调试前需要确定两个软件要素：

首先，本机PC上必须有要调试程序的源代码，否则调试器在调试过程中无法显示c/c++代码信息；

其次，要调试的程序在编译时加入了符号信息，即编译时加入了“-g”参数，参见**错误！未找到引用源。**所示Makefile。

这里要调试的是/home/root/workspace/c下面的hello.c程序，其代码如**图2-** 7所示：



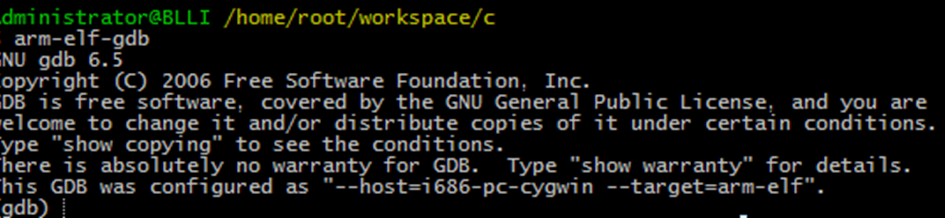
**图2- 27 hello.c代码**

打开Cygwin终端，输入以下命令开始进行调试：

cd/home/root/workspace/c

arm-elf-gdb

此时进入gdb的命令行界面，如**图2-** 8所示：



**图2- 28 arm-elf-gdb调试命令行**

依次在gdb的命令行中输入以下命令（更多的命令见附录3-3）：

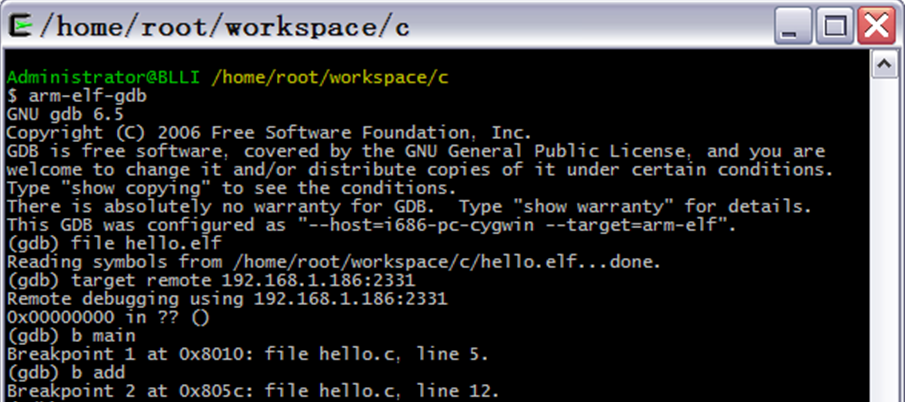
file hello.elf

target remote xxx:2331（xxx是本机PC的ip，例如192.168.1.186等）

b main

b add

如**图2-** 9所示：



**图2- 29 gdb命令示例**

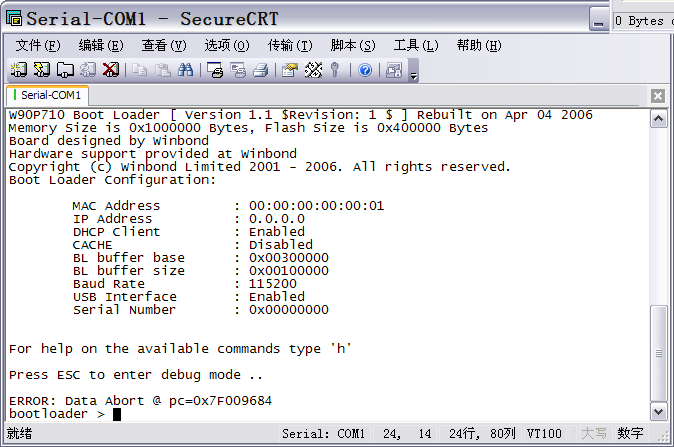
其中file hello.elf导入要调试程序的符号信息；然后使用target remote命令连接PC的J-Link gdbserver，其中192.168.1.186是本机PC的ip地址，2331是J-Link所占用的端口号。接着使用b 命令设置断点，由**图2-** 可知，设置的断点是main函数入口和其调用的add函数入口。

此时符号信息和断点已经设置好，下一步需要通过串口把要调试的程序下载到内存中去。

在gdb命令行里面输入命令：

c

使cpu继续运行。此时快速切换到secureCRT软件的界面，并按下esc键，进入w90p710的bootloader命令行，如**图2- 7**所示：



**图2- 7 w90p710的bootloader**

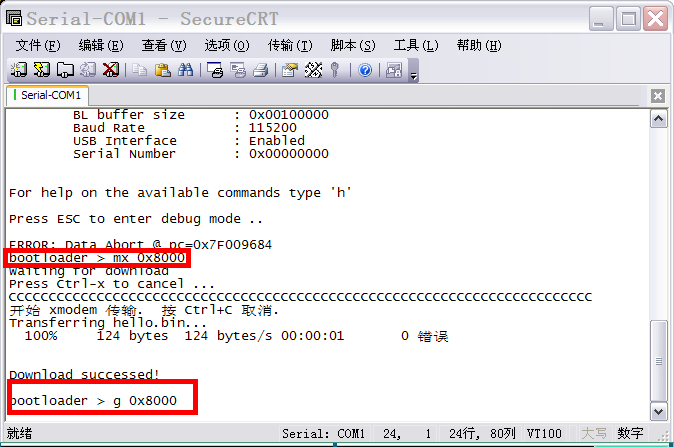
在bootloader命令行中输入命令：

mx 0x8000

然后依次点击secureCRT 菜单栏的“传输→发送Xmodem”，找到预先编译生成的hello.bin文件(D:\cygwin\home\root\workspace\c下面)。点击发送即可把该文件下载到开发板内存的0x8000地址处。然后继续在bootloader命令行输入命令：

g 0x8000

使bootloader运行内存0x8000处的程序，如**图2- 8**所示：



**图2- 8 下载可执行程序**

此时切换到cygwin界面，发现已经触发main函数断点，并显示当前断点处的代码。

依次在gdb命令行中输入下列命令（如**图2- 9错误！未找到引用源。**所示）：

n

p a

p b

n

p b

其中n命令是c语言单步执行，p（print）命令可以打印当前变量的值。此时在gdb命令行中输入命令：

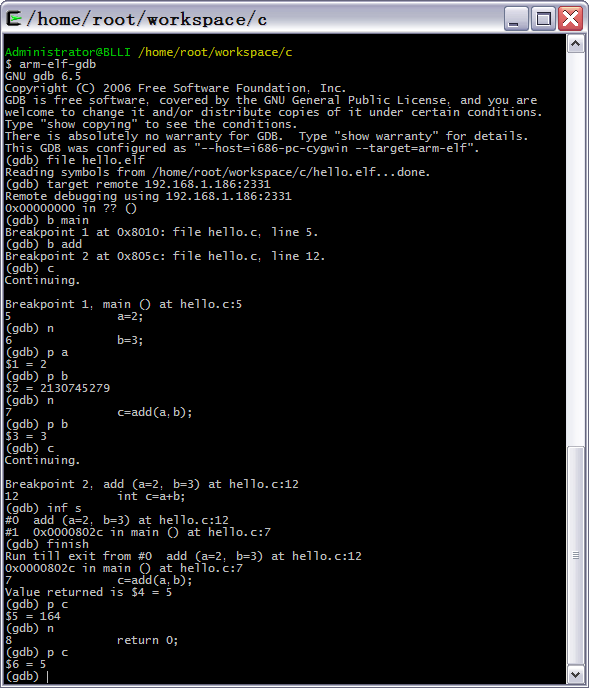
c

使cpu继续运行，发现随后触发了add函数断点，可以通过在gdb命令行中输入下列命令进行调试：

inf s

finish

其中“inf s”命令是打印当前函数调用信息，由**图2- 9**可知，add函数是在main函数的第7行被调用的，而“finish”命令是执行完毕函数并返回上一层函数，由**图2- 9**可知其返回值是5。



**图2- 9 调试示例程序**

**附录 2-1 编译和安装工具链**

**图2-** 中“可直接使用的安装包”文件夹里面的arm-elf-tools.tar.gz是已经编译完成的工具链，包括arm-elf-ld、arm-elf-gcc和arm-elf-gdb等工具，可以拷贝并解压到Cygwin工作目录中，然后添加解压路径到Cygwin的PATH变量中，具体需要在Cygwin终端中输入的命令如下：

cd /opt

mkdir arm\_tools

其中“cd /opt”命令指的是进入Cygwin工作目录下的opt文件夹下；“mkdir arm\_tools”命令是在当前目录下新建一个名为arm\_tools的文件夹，此时打开Windows系统中的“D:\cygwin\opt”目录，发现已经多了一个名为arm\_tools的文件夹，将arm-elf-tools.tar.gz拷贝到该文件夹下，然后继续在Cygwin终端中输入以下命令：

cd arm\_tools

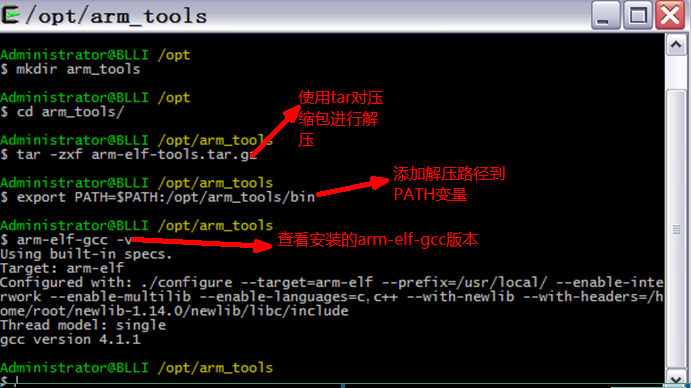
tar–zxf arm-elf-tools.tar.gz

export PATH=$PATH:/opt/arm\_tools/bin

arm-elf-gcc -v

其中使用“cd arm\_tools”命令进入到新建的arm\_tools文件夹下；“tar”命令是用来对arm-elf-tools.tar.gz压缩包进行解压的；解压后的文件将出现在Windows系统中的“D:\cygwin\opt\arm\_tools”文件夹下；然后“export”命令用来设置Cygwin系统的环境变量——PATH，在原来的PATH变量后面加入了解压得到的bin文件夹路径；最后使用“arm-elf-gcc -v”命令查看安装的arm-elf-gcc的版本信息。

整个过程如下图所示：



可直接使用的安装包

**图2-** 中“未编译的源码包”文件夹里面是编译生成arm-elf工具链所需的源码，一共有四个压缩包，如下图所示：

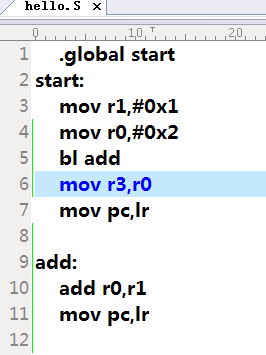


未编译的源码包

其中binutils-2.23.2.tar.bz2是用来生成汇编编译器(arm-elf-as)和链接器(arm-elf-ld)等工具的源码；newlib-1.14.0.tar.gz是生成一些gcc所需库文件的源码；gcc-4.1.1.tar.bz2是用来生成arm-elf-gcc，arm-elf-g++等工具的源码。gdb-6.5a.tar.bz2是用来生成调试工具arm-elf-gdb的源码。

从源码开始编译生成完整工具链的过程可以在当前文件夹下的README.txt中找到，整个编译过程大约耗时3个小时，感兴趣的同学可以自己实验，这里不多做介绍。

**附录 2-2 hello.S 代码**



调试该程序时可以设置断点为“hello.S:2”，然后通过si单步运行汇编程序，然后通过inf r查看cpu的寄存器信息。

**附录2-3 GDB常用命令简介**

file xxx：其中xxx是编译生成的包含符号信息的elf文件，file命令可以导入elf文件中的符号信息到gdb中，其中符号信息主要指的是程序运行时每段代码对应的运行地址。

target remote ip:port：远程连接指定ip和端口的gdbserver，其中port指的是gdbserver所占用的端口号。

break（或者b） xxx：在xxx处设置断点，其中xxx可以是一个函数，也可以是某个文件的某一行，比如b hello.S：2，意思是在hello.S的第二行设置断点。

continue（或者c）：使cpu继续运行。

info(或者inf) s：查看当前代码的函数调用信息。

inf r：查看当前所有的arm寄存器。

inf b：查看当前设置的断点信息。

list（或者l）：查看当前代码附近的c程序代码。

set variable=value：将一个变量variable赋值为value，比如set $pc=0x8000，是设置当前pc寄存器的值为0x8000。

print xxx（或者p）：打印当前变量xxx的值，可以设置打印格式，例如print /x a表示使用十六进制格式打印变量a，print /d a表示使用十进制格式打印变量a。

display(或者disp) xxx：跟踪显示变量xxx的值。

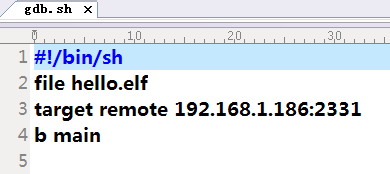
next（或者n）：c语言单步执行，不触发函数调用。

step（或者s）c语言单步执行，如果遇到函数则进入函数内部。

si：汇编语言单步执行。

finish：结束当前函数并返回上一层函数，同时如果函数有返回值则显示返回值。

source xxx：批处理xxx脚本，一般将一些经常使用的命令输入到一个脚本中，然后可以避免反复输入。例如编写一个gdb.sh脚本，如下图所示



然后在gdb命令行里面输入命令：

source gdb.sh

就相当于一次输入了上图中的3条命令。

**第三章 使用gdb在开发板w90p710上调试uClinux内核**

**一、实验目的**

1、熟悉和了解uClinux内核相关知识。

2、熟悉和掌握使用gdb工具调试内核。

**二、实验硬件需求**

1. w90p710开发板一套（电源，串口）。
2. J-Link调试器。

**三、实验软件需求**

1. 安装好arm-elf-gdb的Cygwin环境。
2. segger J-Link gdb server v4.26。
3. 串口终端程序SecureCRT。
4. w90p710内核源码和文件系统。

**四、实验内容**

uClinux表示micro-control linux，即“微控制器领域中的Linux系统”，是开放源码的嵌入式Linux的典范之作。uClinux主要是针对目标处理器没有存储管理单元MMU（Memory Management Unit）的嵌入式系统而设计的，已经被成功地移植到了很多平台上，本节课将使用J-Link在w90p710平台上调试uClinux内核，主要内容由以下四部分：

1. 搭建内核调试环境。
2. 通过加入断点查看内核中的内存信息。
3. 通过加入断点查看cpu的任务调度信息。
4. 通过加入断点查看并调试串口中断。

***注意：实验三和实验四中有底纹的字符是要在软件界面中手动输入的命令。***

**五、实验具体步骤**

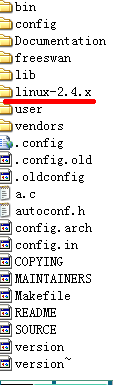
1、搭建内核调试环境：

1. 将包含w90p710的uClinux内核源码的uClinux.tar.gz压缩包（**图2- 1**所示目录中的“w90p710内核代码”文件夹中）拷贝到Cygwin工作目录的/home/root目录下。
2. 打开Cygwin终端，进入/home/root目录下，输入下列命令对uClinux.tar.gz进行解压：

cd /home/root

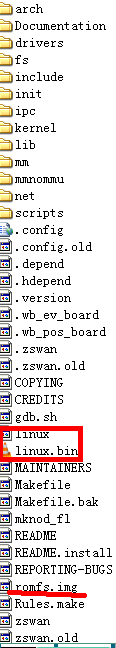
tar–zxf uClinux.tar.gz

解压后目录如**图3- 1**所示：



**图3- 1 uClinux-dist目录**

1. **图3- 1**中linux-2.4.x目录是待调试的内核源码目录，其内部结构如**图3- 2**所示：



**图3- 2 linux-2.4.x目录**

其中，linux文件是提前自行编译好的带调试符号信息的uClinux内核文件，linux.bin是后续要下载到开发板上运行的uClinux内核镜像，romfs.img是文件系统镜像。这里对内核源码中各个文件夹的功用进行简单介绍：

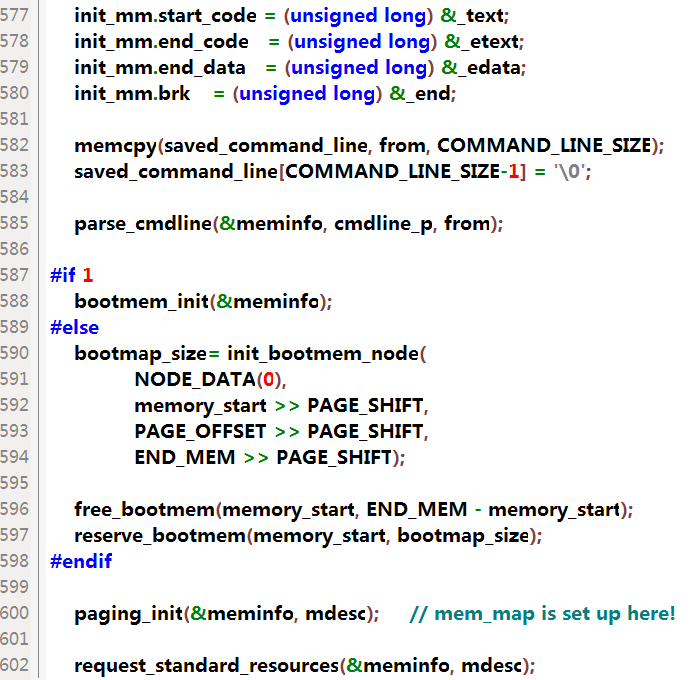
arch文件夹里面主要是内核中与处理器架构相关代码；Documentation文件夹内是内核源码的一些帮助文档；drivers文件夹内是和硬件平台相关的驱动代码；fs文件夹内部是和文件系统相关的代码；include文件夹是整个内核编译所需的头文件目录；init文件夹是内核启动时初始化操作相关代码目录；ipc文件夹是内核中进程间通信相关代码目录；kernel文件夹内是内核中一些与硬件无关的核心代码；lib文件夹是内核编译所需的一些库代码目录；mm和mmnommu文件夹是内核中和内存操作相关的代码目录；net文件夹是内核中网络相关的代码目录；scripts文件夹是内核编译时所需的一些脚本文件目录。

1. 使用J-Link连接好PC和w90p710开发板。
2. 使用串口线连接好PC和w90p710开发板。
3. 打开J-Linkgdb server v4.26界面。
4. 打开串口终端程序SecureCRT。
5. 打开w90p710开发板电源。

2、通过加入断点查看内核中的内存信息：

mm\_struct结构体是Linux内核中记录一个进程所占用的内存信息的结构体，每一个进程都有属于自己的一个mm\_struct信息，其中init\_mm变量是一个特殊的mm\_struct结构体变量，因为它是内核进程所使用的mm\_struct结构体变量。其内部不仅包含内核镜像本身的一些信息，例如内核代码段起始地址，数据段起始地址等，同时包含了内核运行过程中占用的内存地址信息列表（mm\_list）。而meminfo结构体包含有内核能够使用的所有内存信息，例如开发板有几个内存bank，每个bank的内存起始地址和大小等。

首先通过使用记事本打开linux2.4.x/arch/armnommu/kernel/setup.c文件查看其中的setup\_arch函数（详见附录3-1），发现该函数中实现了对init\_mm和meminfo这两个变量的初始化。如 **图3- 3**所示：



**图3- 3 setup\_arch函数中内存初始化相关操作代码**

因此后续将在该函数内部设置断点观察内核中的内存初始化信息。

1. 在Cygwin终端依次输入以下命令开始进行调试：

cd /home/root/uClinux-dist/linux-2.4.x/

arm-elf-gdb

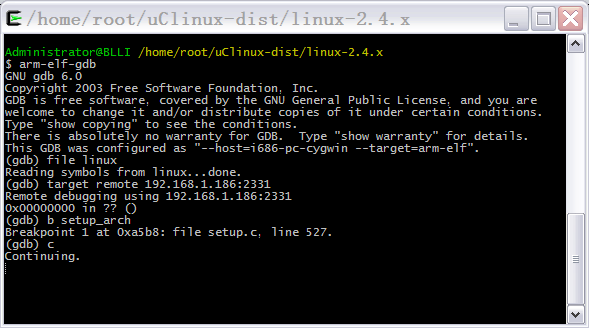
并在出现的gdb命令行中依次输入以下调试命令：

file linux

target remote xxx:2331

b setup\_arch

如**图3- 4**中所示：



**图3- 4 设置setup\_arch断点**

这里设置的断点函数是setup\_arch函数，在gdb命令行输入命令：

c

使cpu继续运行，此时迅速切换到SecureCRT程序并按下esc键进入bootloader命令行界面。

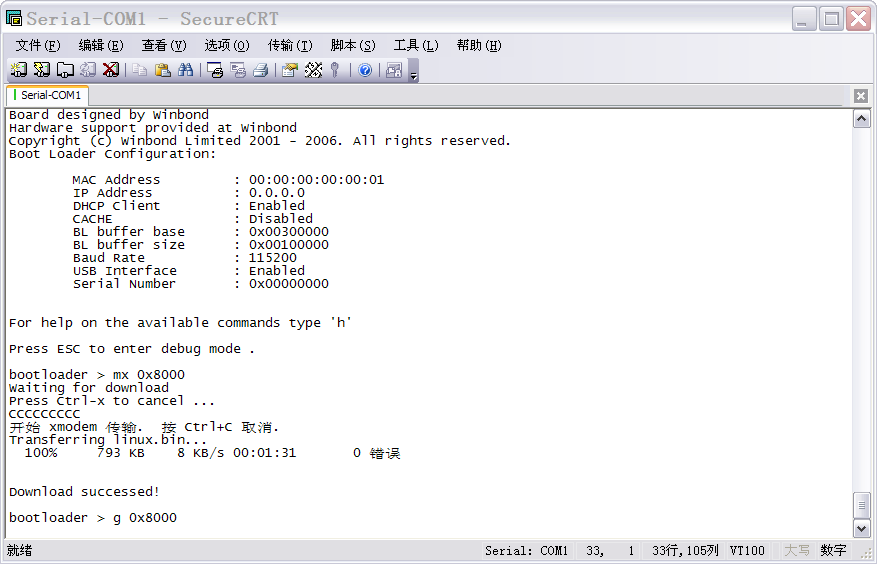
1. 在bootloader命令行依次输入命令：

mx 0x8000

然后依次点击secureCRT 菜单栏的“传输→发送Xmodem”，找到要下载的内核镜像linux.bin（在/home/root/uClinux-dist/linux-2.4.x文件夹下），然后继续在bootloader命令行中输入命令：

g 0x8000

使cpu跳转到开发板内存的0x8000地址执行，如**图3- 5**所示：



**图3- 5 使用串口下载内核到开发板**

1. 此时查看Cygwin终端可以发现cpu已经停在setup\_arch函数中，此时可以在gdb命令输入下列命令：

p init\_mm

p/x meminfo

查看初始化前的init\_mm信息和meminfo信息。然后输入下列命令：

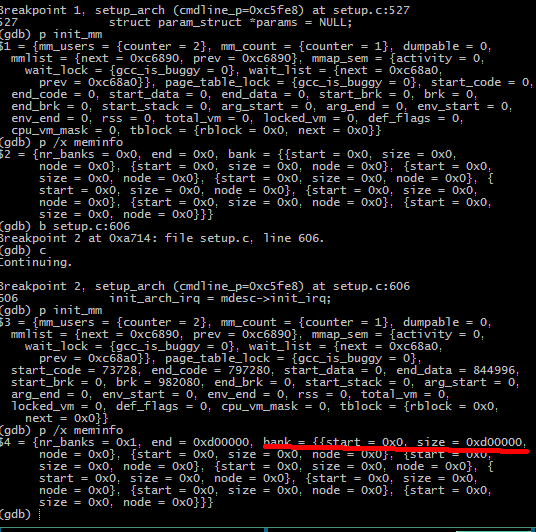
bsetup.c:606

c

p init\_mm

p /x meminfo

设置新的断点在该文件的606行，查看初始化后的内存信息，其过程如**图3- 6**所示：



**图3- 6 查看初始化前后的内存信息**

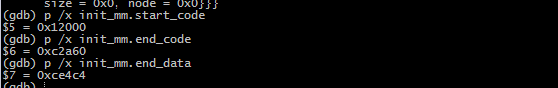
由**图3- 6**中meminfo信息可知，内核能够识别的内存只有一个bank，其起始地址（start)是0x0，然后大小（size）是0xd00000。同时可以结合代码查看init\_mm结构体里面一些具体变量的值，在gdb命令行继续输入命令：

p /x init\_mm.start\_code

p /x init\_mm.end\_code

p /x init\_mm.end\_data

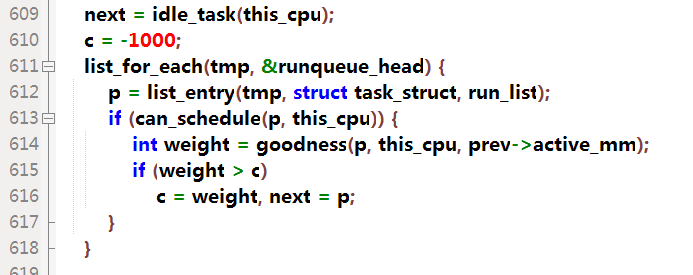
上述三个变量分别代表了内核的代码段起始地址、结束地址和数据段的结束地址，输出结果如**图3- 7**所示：



**图3- 7 查看init\_mm结构体里面的参数**

3、通过加入断点查看cpu的任务调度信息

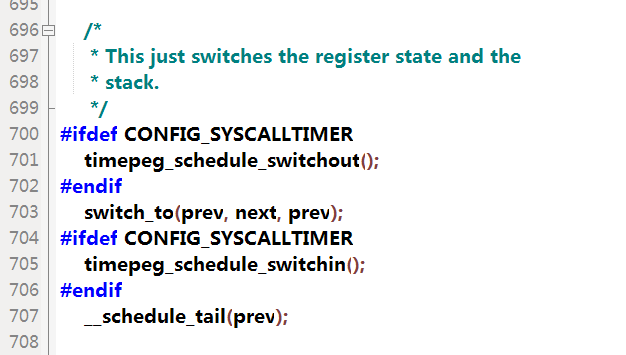
schedule函数是Linux内核中任务调度函数，其代码实现起始于linux-2.4.x/kernel/sched.c文件的552行（见附录3-2）。该函数通过遍历当前CPU的任务链表runqueue，然后调用goodness函数计算链表中每一个进程的weight（权重），从中选取权重得分值最高的进程作为占据CPU的下一个进程，如**图3- 8**所示：



**图3- 8 遍历链表选取下一个占据CPU的进程**

其中goodness函数内部实现了当前内核的任务调度算法，其主要调度机制是判断一个进程是否是实时进程：如果不是实时进程，则返回一个和当前进程已运行时间相关的一个数字作为当前进程的权重；如果是实时进程，则返回一个较大的数，对该函数感兴趣的同学可以自行查阅附录3-3。

在选择好下一个进程后，schedule函数就会调用switch\_to函数进行进程间切换，如**错误！未找到引用源。**中703行所示：



**图3- 9 进程间切换**

后续实验将设置断点在**图3- 9**的703行处，观察切换前后的进程信息。

1. 调试内核里面cpu任务调度需要内核和文件系统协同工作，只有内核初始化完毕进入文件系统后才会有进程之间的切换。

首先在Cygwin终端的gdb命令行中输入命令：

quit

退出调试，然后在Cygwin终端中输入命令：

arm-elf-gdb

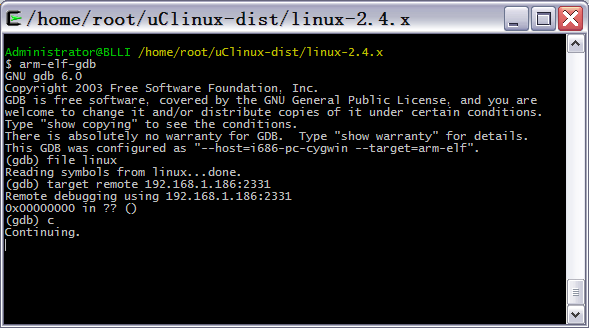
重新开始进行新的调试。并在gdb的命令行输入命令：

file linux

target remote xxx:2331

c

如**图3- 10**所示：



**图3- 10 开始进行CPU调度调试**

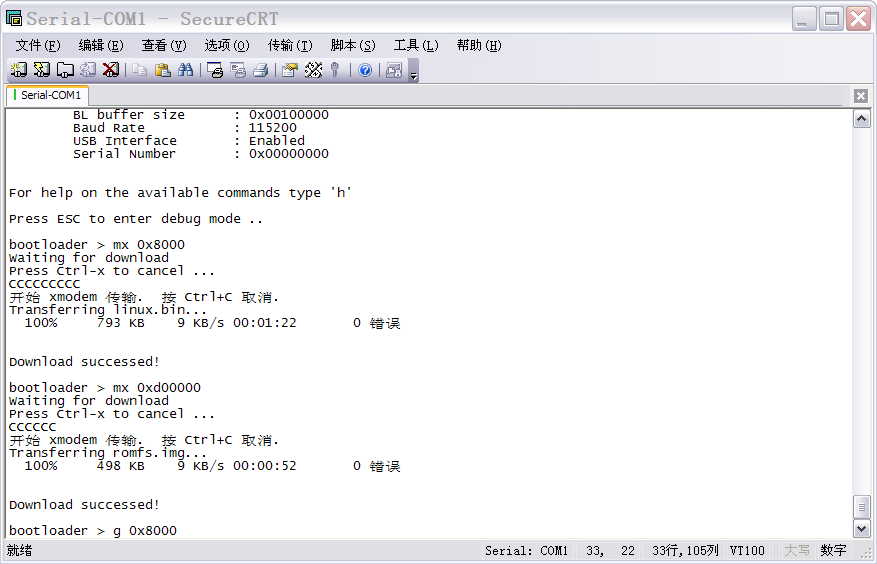
这里可以看到预先没有设置任何断点，直接让cpu自由运行。此时在串口软件SecureCRT里面及时按下“esc”键进入bootloader界面，然后依次在bootloader的命令行输入命令：

mx 0x8000

下载内核镜像linux.bin，然后再bootloader的命令行输入命令：

mx 0xd00000

文件系统镜像romfs.img（这两个文件均在Cygwin的/home/root/uClinux-dist/linux-2.4.x目录下），如**图3- 11**所示：



**图3- 11 使用串口下载内核和文件系统到开发板**

继续在bootloader命令行输入命令：

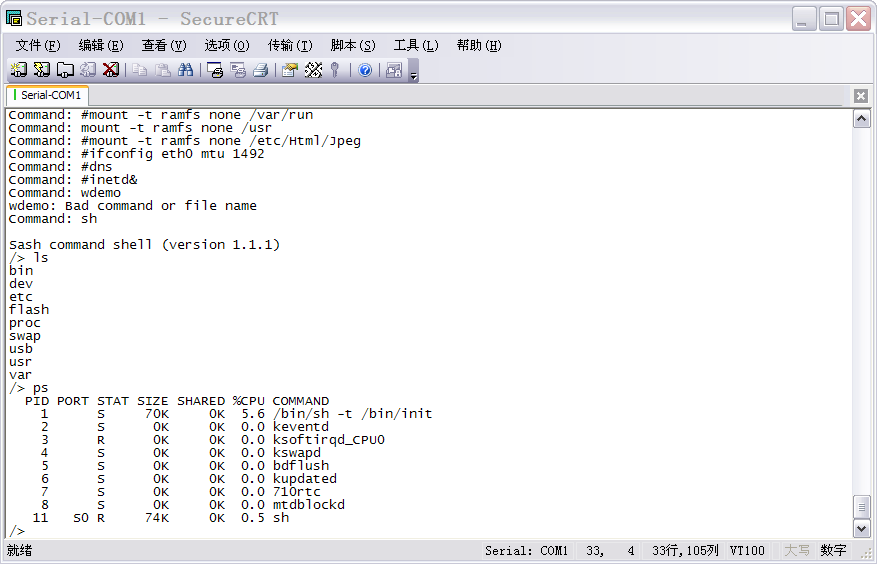
g 0x8000

可以发现内核和文件系统均可以正常启动，最后进入uClinux自带的简易shell界面，如**图3- 12**所示。该shell程序是运行在w90p710上的一个应用程序，类似于Cygwin界面的命令行处理程序（bash）。该shell程序接收PC端发来的字符并将字符再次发送到PC端显示，当用户输入回车键时，该shell程序对回车键之前输入的字符串进行判断，如果该字符串是shell的一个内建命令，例如ls、ps等，则shell对输入命令进行解析并将结果输出到PC进行显示。这里值得注意的是该shell命令行的开头总是“/>”三个字符。而用户直接在命令行输入回车，则被认为是一条空命令，shell处理后直接跳到下一个命令行，即只输出shell命令行开头三个字符“/>”。

此时在shell命令行中输入命令：

ps

查看当前系统运行的进程情况，如**图3- 12**所示：



**图3- 12 使用ps命令查看当前进程信息**

1. 然后在Cygwin终端中的gdb命令行中按下“ctrl+c”组合键，打断cpu的运行，然后依次在gdb命令行输入以下命令设置断点：

b sched.c:703

c

如**图3- 13**所示：



**图3- 13 设置CPU调度断点**

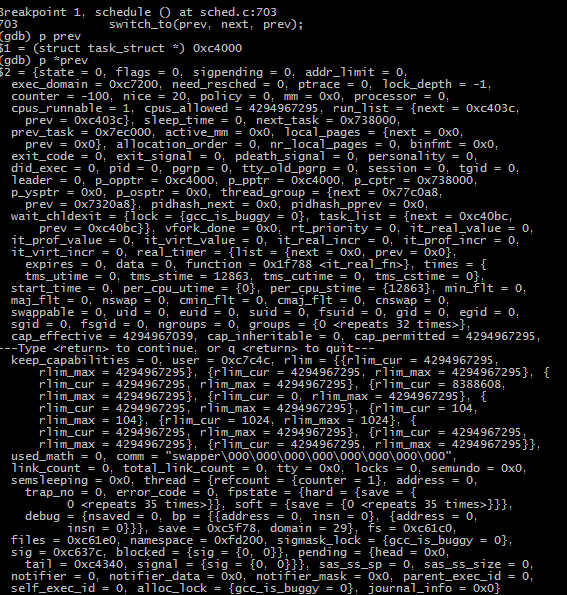
由**图3- 9错误！未找到引用源。**可知，703行位于schedule函数中，此时prev变量代表发生调度前正在运行的进程，next变量代表调度后即将运行的进程。此时在gdb命令行输入命令：

p prev

p \*prev

p \*next

查看prev和next两个进程的信息，其中prev进程的信息如**图3- 14**所示：



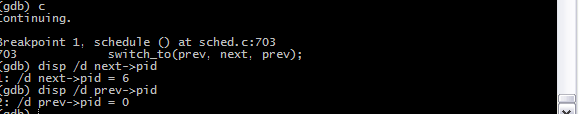
**图3- 14 查看prev结构体信息**

1. 其中可以通过读取结构体中的pid变量获知该结构体的具体信息，所以跟踪打印prev和next结构体pid变量的值，以查看两个切换任务的pid信息，在gdb命令行输入命令：

disp /d next->pid

disp /d prev->pid

结果如**图3- 15**所示：



**图3- 15 查看任务调度前后的pid号**

disp指令可以一直跟踪打印某个变量（使用undisp+id可以取消跟踪打印），这里一直跟踪prev和next的pid信息。

1. 继续在gdb命令行中输入命令：

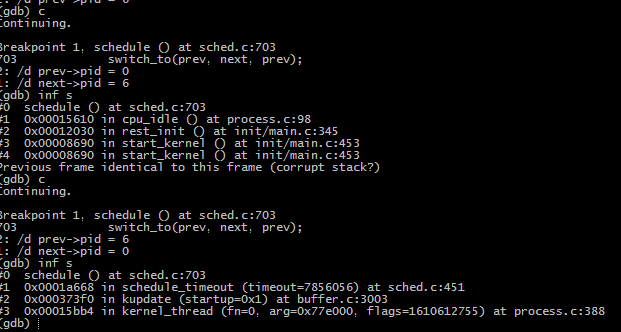
c

inf s

c

inf s

让cpu连续运行，发现在不对SecureCRT串口终端进行操作的情况下（即处理器处于空闲运行状态），每隔固定时间cpu会发生pid为0和pid为6之间的任务切换（如图3- 16所示）：



**图3- 16 pid 0和pid 6之间进行切换**

这是因为操作系统空闲时会进入cpu\_idle函数，通过“inf s”命令打印cpu\_idle函数的调用堆栈可知，该函数是内核启动时创建的，其具体代码信息可以在linux-2.4.x/arch/armnommu/kernel/process.c中查询（见附录3-4），而Linux内核本身在固定时间后就会进行一次调度判断，pid为0的是内核内建的任务调度进程(swapper)，pid为6是内核运行时创建守护进程kupdated，kupdated进程运行时执行的kupdate函数信息可以在linux-2.4.x/fs/buffer.c中查询（见附录3-5）。此时在gdb命令行中输入命令：

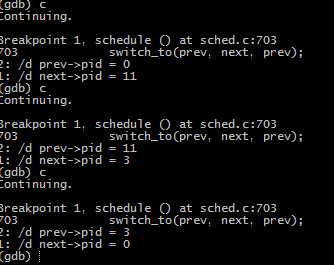
c

使CPU继续运行，然后在SecureCRT的shell命令行中快速按下回车键，则可以发现Cygwin的gdb命令行中触发了预先设置的断点，继续gdb命令行中输入命令：

c

c

出现如**图3- 17**所示的进程切换：



**图3- 17 观察调度前后的pid号**

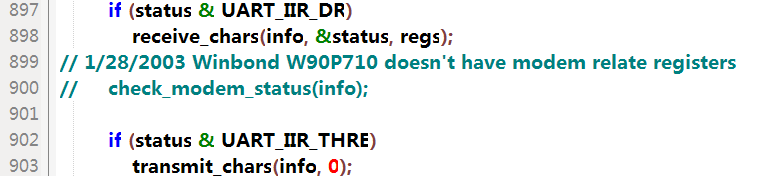
即首先发生了pid为0和pid为11的进程切换，由**图3- 12**可知，pid为11的进程是处理shell命令的sh进程。随后发生了pid为11和pid为3的进程切换，pid为3是ksoftirqd\_CPU0进程，然后又发生了pid为3和pid为0的进程切换。

发生上述进程切换是因为在SecureCRT的shell中输入字符，PC会向开发板的串口发送字符，触发开发板的串口接收中断，然后将中断接到的字符通过串口送往SecureCRT回显。同时接收到的字符也会交由当前的shell程序进行处理，因此触发了pid为11的sh进程。shell程序处理完毕后会将处理结果通过串口发送到PC的SecureCRT软件中的shell界面进行显示。通过串口发送数据会触发串口硬件中断，内核在调用串口中断处理函数后，会继续调用ksoftirqd\_CPU0进程（pid为3）完成这次中断处理，最后串口发送字符完毕后重新返回内核的swapper进程(pid为0)。

接下来可以对串口中断进行调试以验证上述过程。

4、通过加入断点查看并调试串口中断

串口接收到中断信号后会执行内核内部对应的串口中断服务函数，w90p710默认使用uart0和外界进行通讯，uart0的中断服务函数是rs\_interrupt\_single函数（完整代码见附录4-6），该函数位于linux-2.4.x/drivers/char/w90p710\_uart.c文件中，其处理接收和发送字符的相关代码如**图3- 18**所示：



**图3- 18 串口中断处理函数rs\_interrupt\_single**

其中receive\_chars函数处理串口接收字符，transmit\_chars函数处理串口发送字符，其中transmit\_chars函数又调用serial\_out函数（位于w90p710\_uart.c的404行，见附录3-7）将要发送的字符输出，所以后续将在以上函数中设置断点，观察从SecureCRT软件显示的uClinux shell终端中输入回车，到该回车被开发板执行的整个过程。

1）首先设置相应的断点，在gdb命令行中输入以下命令：

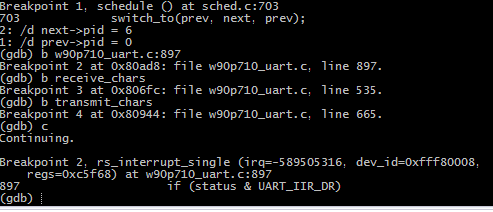
b w90p710\_uart.c:897

b receive\_chars

bw90p710\_uart.c:406

c

接着迅速在SecureCRT软件的shell命令行中按下回车键，此时发现触发设置的rs\_interrupt\_single函数断点，如**图3- 19**所示：



**图3- 19 设置串口调试断点**

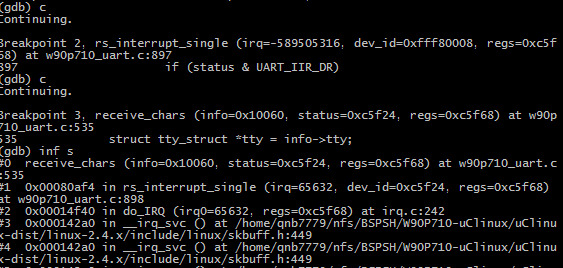
继续在gdb命令行输入命令：

c

让cpu继续运行，发现触发了预先设置的receive\_chars函数断点，说明内核中断开始处理PC发来的字符，此时输入命令：

inf s

来确定函数调用关系，如**图3- 20**所示：



**图3- 20 打印receive\_chars函数调用堆栈**

可见receive\_chars函数确实是由rs\_interrupt\_single函数触发的，而rs\_interrupt\_single函数是被Linux内核的中断入口do\_IRQ函数调用的（后续可能会出现很多\_\_irq\_svc的调用信息，该函数是中断发生时的汇编入口，对arm中断感兴趣的同学可以自行查阅该函数）。

然后在gdb命令行中继续输入命令：

c

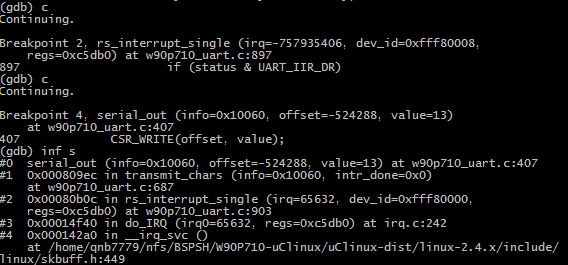
发现又触发了rs\_interrupt\_single函数，说明shell即将要把接收到的字符发送到PC端进行显示，在gdb命令中继续输入命令：

c

发现这次触发了serial\_out函数断点，在gdb中输入命令：

inf s

查看其调用堆栈，发现serial\_out函数确实是由transmit\_chars函数调用而来，如**图3- 21**所示：



**图3- 21 查看serial\_out函数调用堆栈**

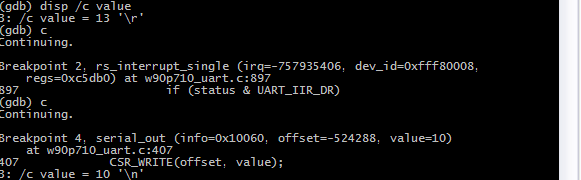
通过阅读serial\_out函数代码可知，该函数将第三个参数value输出到串口发送寄存器，所以可以跟踪打印value的值来查看串口将要输出的字符信息。此时在gdb命令行中输入下列命令观察输出的字符信息：

disp /c value

c

c

发现后续开发板向PC输出的是回车符‘/r’和换行符‘/n’（windows下面的回车键就是回车符和换行符的组合），如**图3- 22**所示：

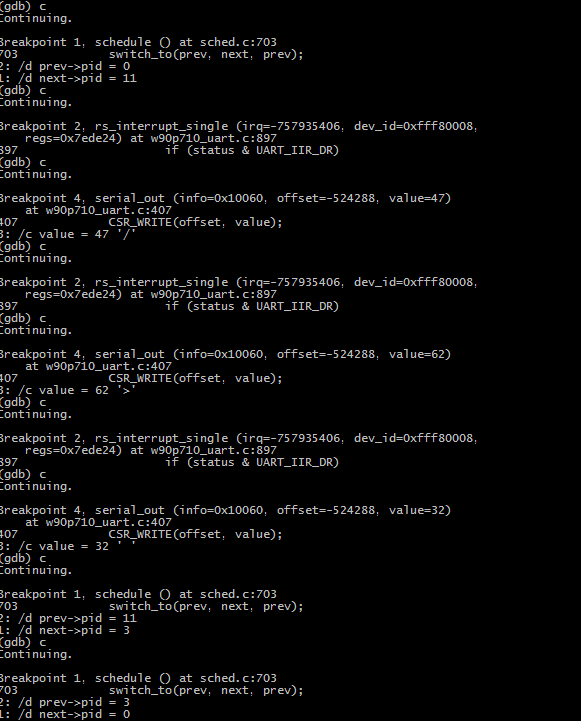


**图3- 22 观察输出的字符信息**

然后继续在gdb命令行中连续输入命令：

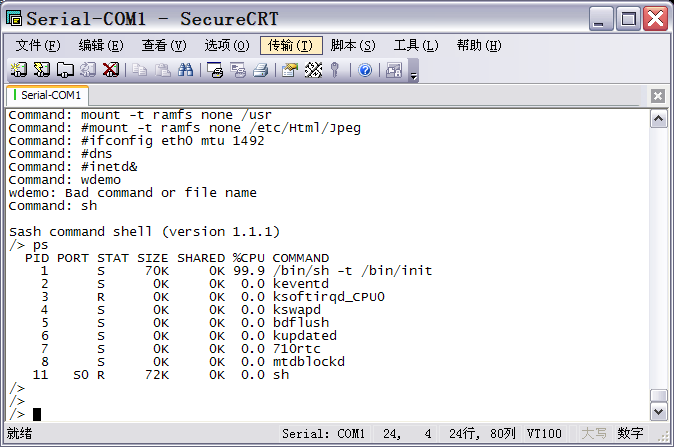
c

可以发现触发了pid为0和pid为11之间的进程切换，进程切换后又向PC发送了三个字符，分别对应斜线、大于号和空格，正如前文所描述的那样，在shell的命令行中直接输入回车键，shell会因为这是一个空命令，直接输出shell的命令行开头字符串“/>”，如**图3- 23**所示：



**图3- 23 调试串口中断**

然后此时观察SecureCRT软件中的shell终端，发现确实输出了上述三个字符，如**图3- 24**所示：



**图3- 24 SecureCRT终端显示**

输出完毕后发生了pid为11和pid为3之间的进程切换，和pid为3和pid为0之间的进程切换，其过程正如上文分析所述。

2）当开发板的内核和文件系统正常工作时，PC通过串口向开发板的shell程序发送一个字符，shell程序后续会通过串口在PC上回显该字符，例如，在SecureCRT终端中输入字母m，则该字母将首先通过串口发送给开发板的shell程序进行处理，然后再通过串口回显到PC的SecureCRT程序中。本实验接下来将在开发板内核的串口中断中通过GDB调试，动态修改开发板回传给PC的字符，实现例如在SecureCRT中输入字母m给开发板，但是最后开发板会回传一个字母n给SecureCRT显示。

首先在Cygwin终端的gdb命令行中输入命令：

quit

退出调试，然后在Cygwin终端中输入命令：

arm-elf-gdb

重新开始进行新的调试。并在gdb的命令行中输入命令：

file linux

target remote xxx:2331

c

然后及时切换到PC串口程序SecureCRT中按下“esc”键进入bootloader界面，然后依次在bootloader的命令行输入命令：

mx 0x8000

mx 0xd00000

下载内核镜像linux.bin和文件系统镜像romfs.img，然后继续在bootloader命令行中输入命令：

g 0x8000

启动下载到内存的内核，后续可以发现内核和文件系统均可以正常启动，进入uClinux自带的简易shell界面。

在Cygwin终端中的gdb命令行中按下“ctrl+c”组合键，打断cpu的运行，然后依次在gdb命令行输入以下命令：

b w90p710\_uart.c:406

c

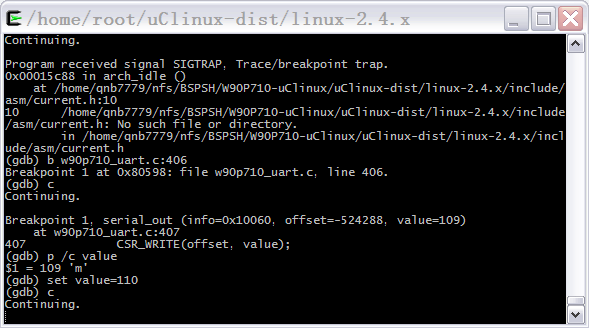
此时在SecureCRT软件中显示的uClinux的shell命令行中输入小写字母m，发现Cygwin终端的gdb命令行中显示cpu触发了serial\_out断点，此时在gdb命令行输入命令：

p /c value

得到串口即将输出的是字母m，即刚才在uClinux的shell命令行中输入的m字符，此时在gdb命令行中输入命令：

set value=110

可以动态修改开发板要回传给PC的字符，如**图3- 25**所示：

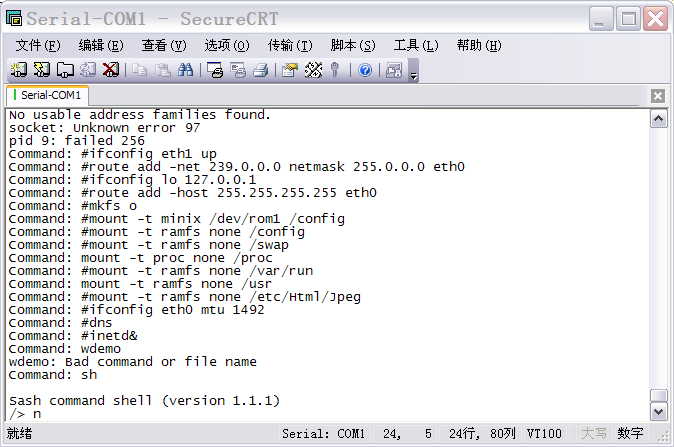


**图3- 25 动态更改要输出的值**

然后在gdb命令行中输入命令：

c

让cpu继续运行，发现串口终端打印出来的是字母n，而不是预先输入的字母m，如**图3- 26**所示：

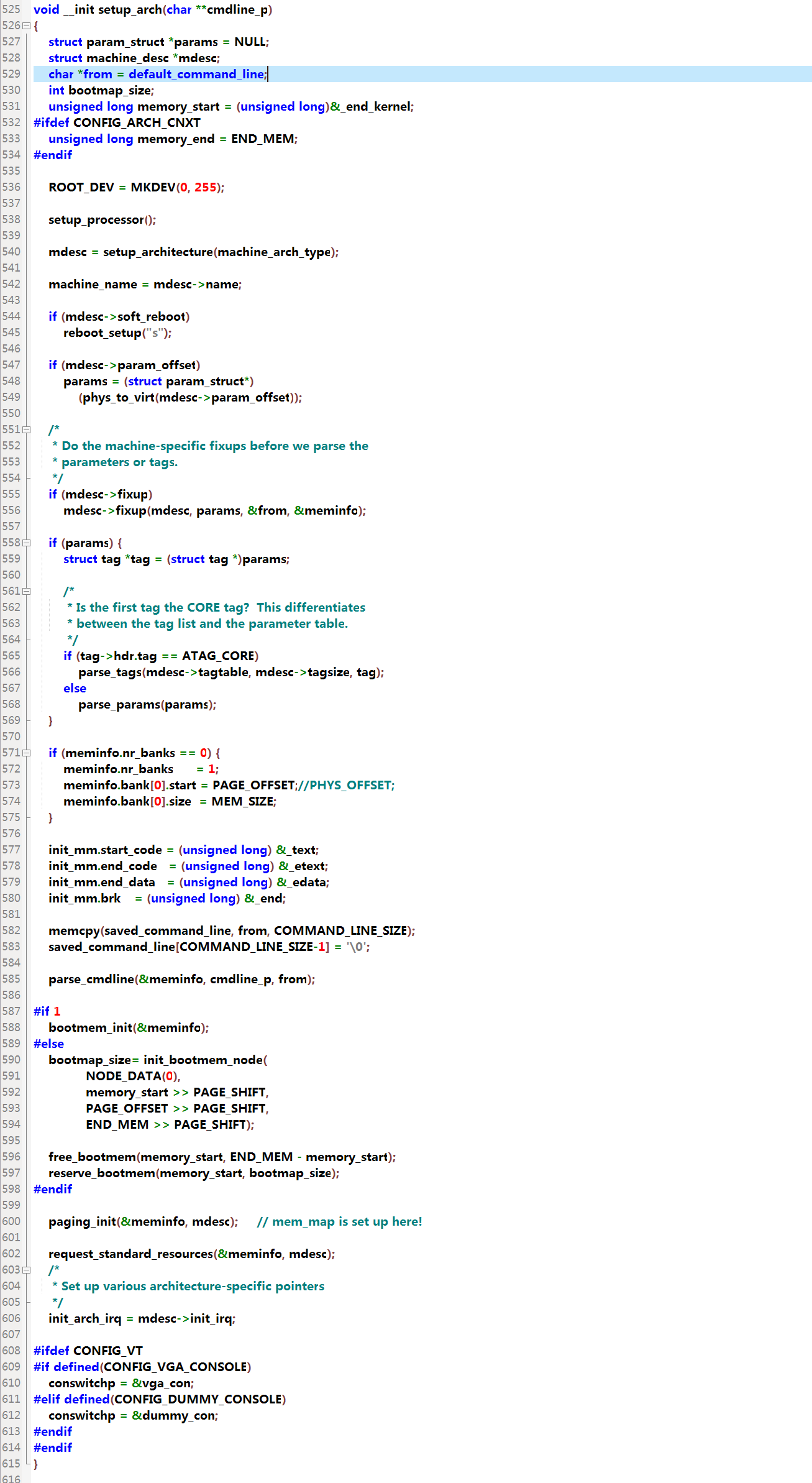


**图3- 26 SecureCRT终端显示**

***注意：结束实验后请务必拔下开发板电源。***

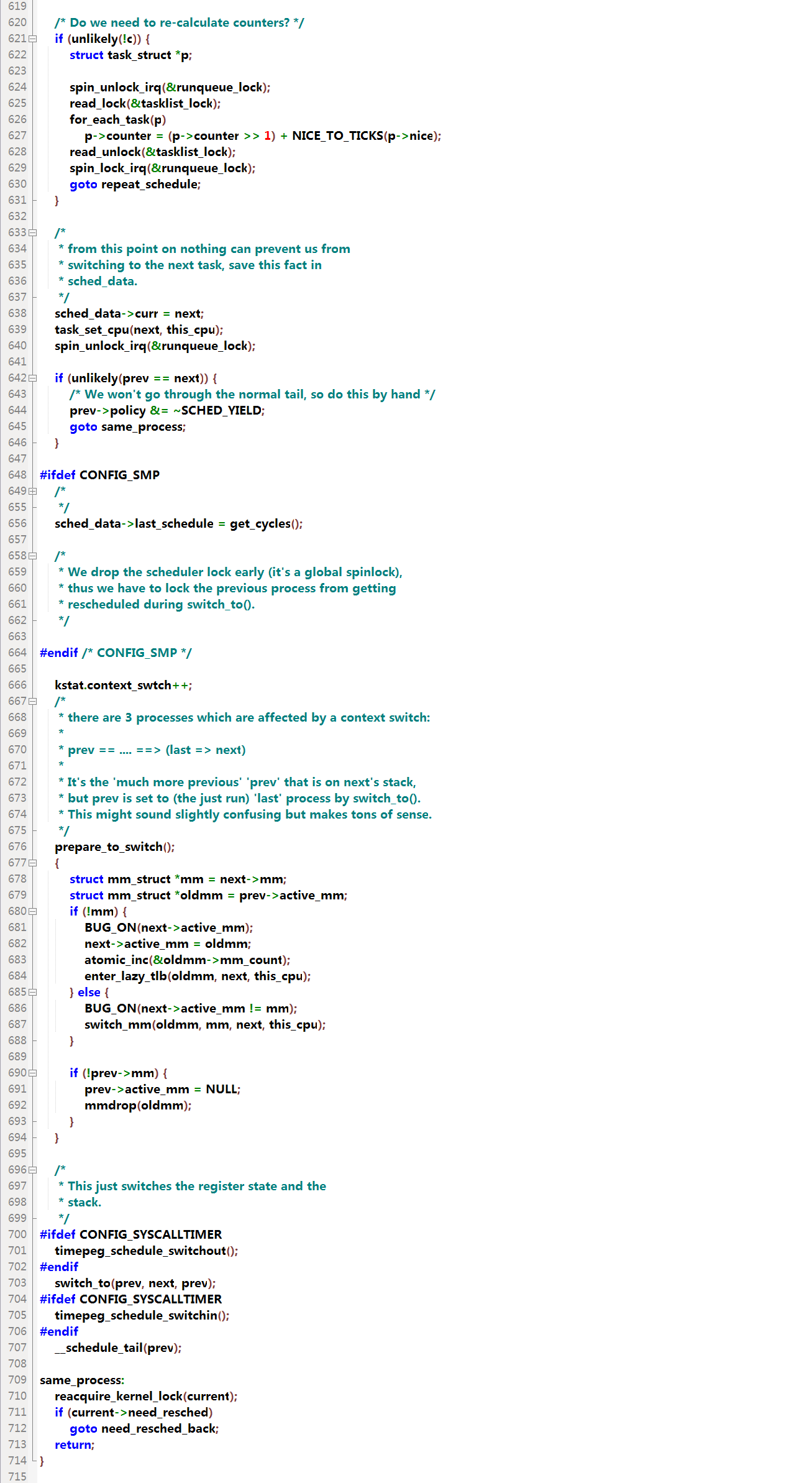
**­**

**附录3-1 setup\_arch函数**

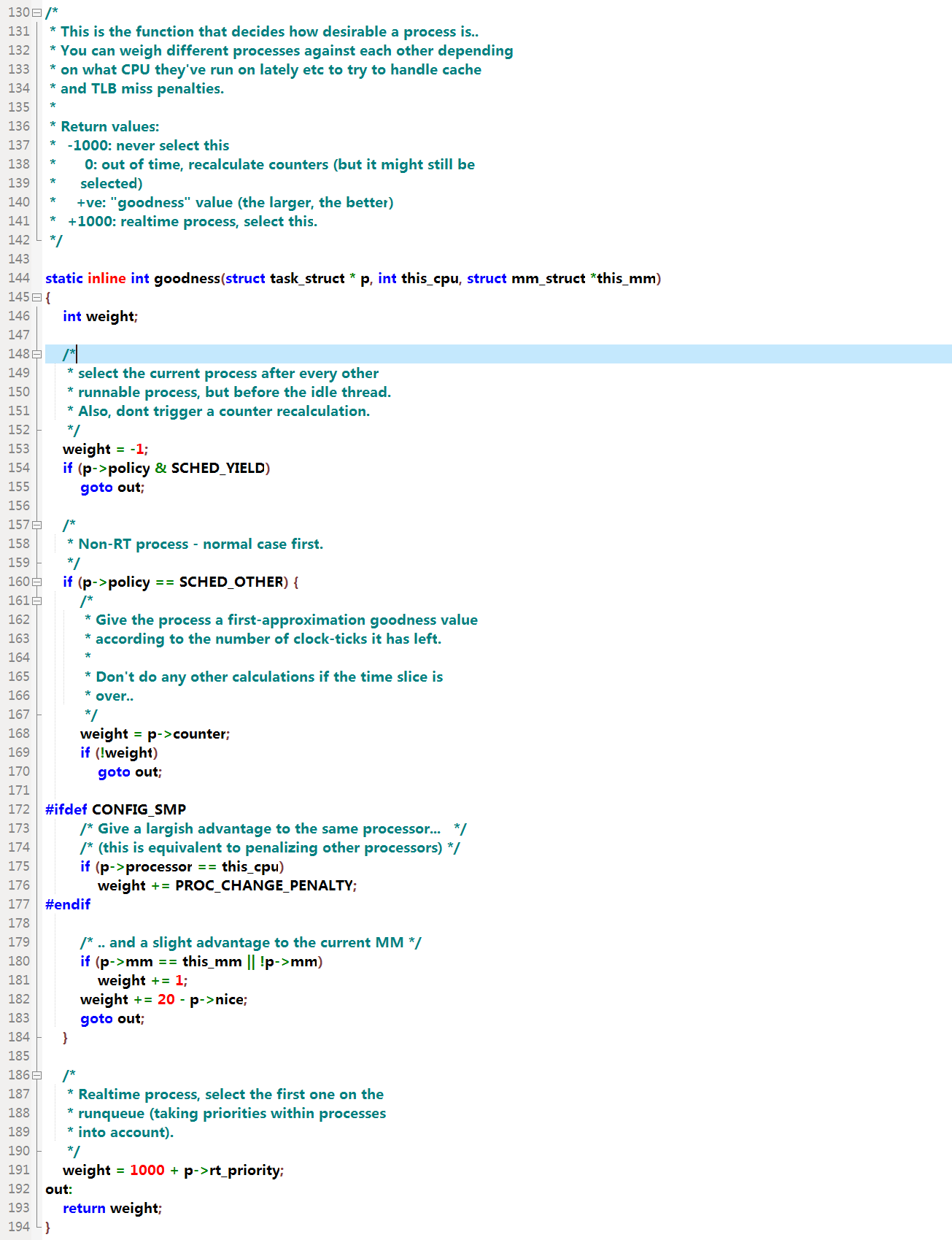


**附录3-2 schedule函数**

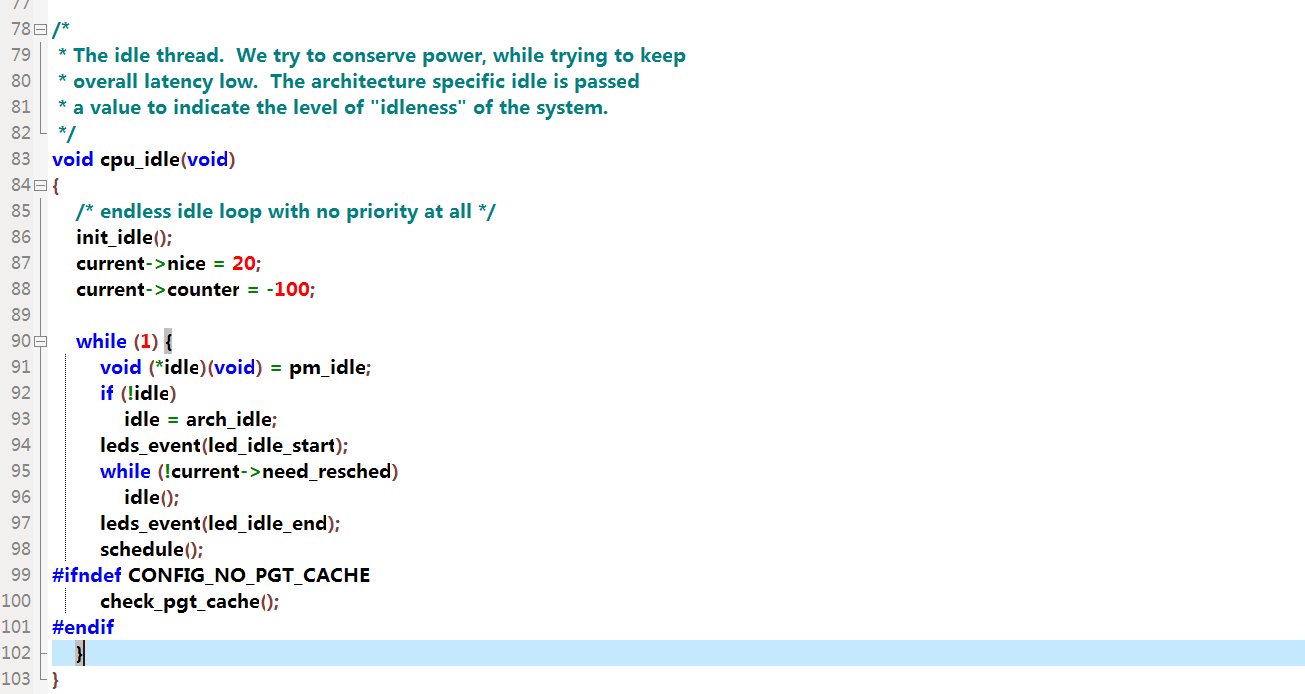




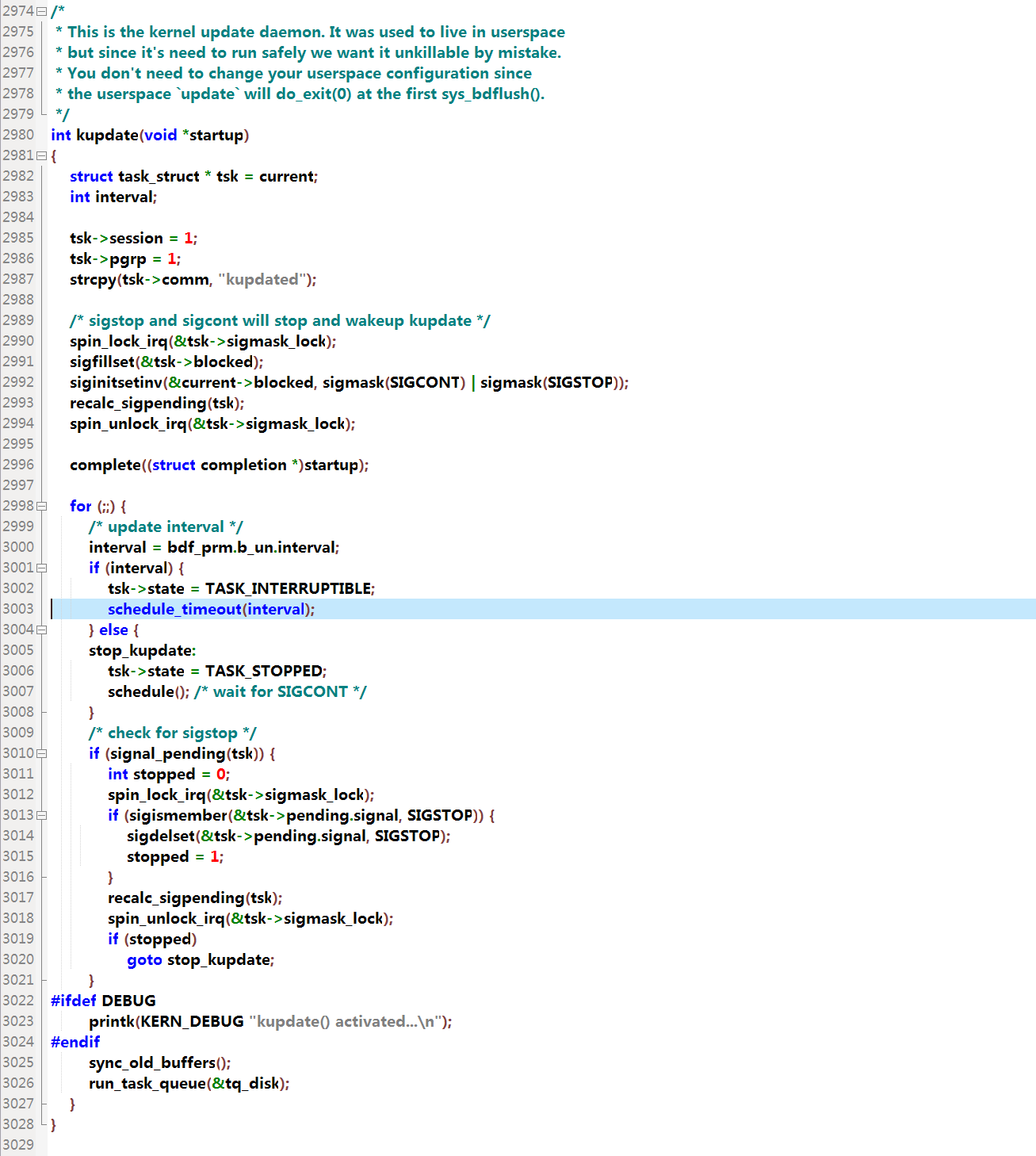
**附录3-3goodness函数**



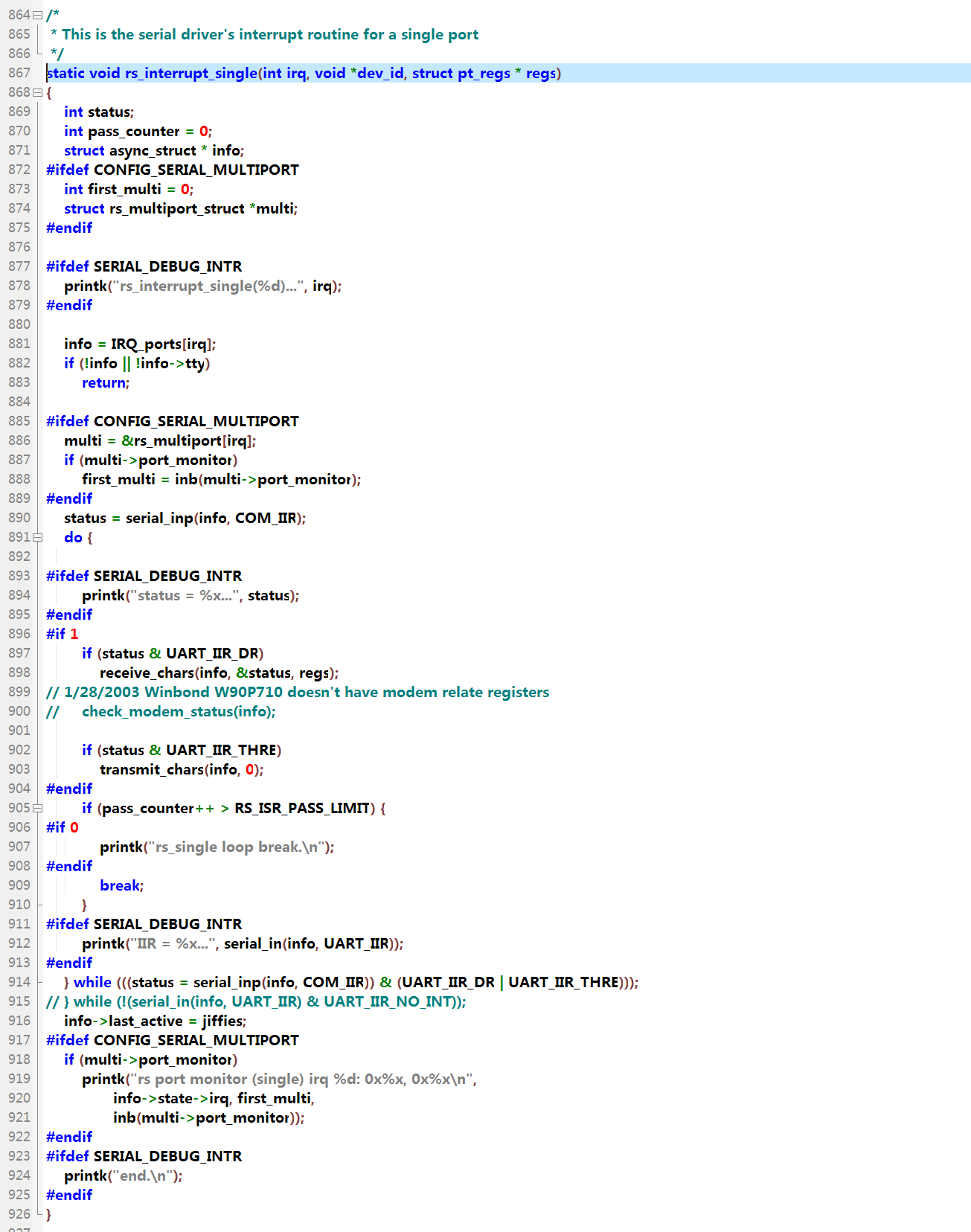
**附录3-4 cpu\_idle函数**



**附录3-5 kupdate函数**



**附录3-6rs\_interrupt\_single函数**



**附录3-7transmit\_chars&serial\_out函数**

