第三节：面向新手的AVR开发工具，及基本知识

互联网上下载AVR STUDIO 4

首先，请放松，我们首先要做的只是下载AVR开发软件和相关的资料。下载前确保这些软件有效，下载的时间取决于你连接互联网的速度。

下载以下文件到你的计算机中（如C:\Temp）

AVR STUDIO 4 (~15MB)		This file contains the AVR Studio 4 Program. This program is a complete development suite, and contains an editor and a simulator that we will use to write our code, and then see how it will run on an AVR device.
Assembly Sample Code (~1kB)		This file contains the Assembly Sample code you will need to complete this guide.
AT90S8515 Datasheet (~4MB)		This is the Datasheet for the AT90S8515 AVR Microcontroller. This is a convenient "Getting Started" device. For now you don't have to worry about the different types of AVR micros. You'll see that they are very much alike, and if you learn how to use one (eg. 8515), you will be able to use any other AVR without any problems.
Instruction Set Manual (~2MB)		This is the Instruction Set Manual. This document is very useful if you want detailed information about a specific instruction.

- AVR STUDIO4：这个文件是一个完整的开发工具，包括编辑、仿真功能，利用这个工具，我们可以编辑源代码，并在AVR器件上运行。这个文件大概15MB。
- 代码实例：这是一个用于演示完整代码，大概1KB。
- AT90S8515数据手册：这是AT90S8515芯片的完整资料。AT90S8515芯片是一款非常容易上手的AVR芯片。现在你不用担心AVR其他型号的产品，以后你会发现AVR系列产品有非常大的相似性，你掌握了一款芯片（比如说8515），学会其他芯片也非常容易。这个文件大概4MB。
- 指令说明：如果你要详细的了解各种指令的话，这个文件非常有用。这个文件大概2MB。

如果以上的文件你都下载了，下面的工作就是安装软件了。

安装AVR Studio 4

AVR Studio现在也提供版本3，不过版本4将逐步替代版本3。

使用Windows NT/2000/XP的用户请注意，安装AVR Studio软件时，必须使用管理员（Administrator）权限登陆，这是Windows系统限定只有管理员才可以安装新器件。

安装过程：

1. 双击AVRSTUDIO.EXE文件。这是一个自解压文件，双击后系统询问将文件解压到哪个目录底下，一般情况下是放在系统默认临时目录下，这个目录是隐藏在你的硬盘中的。为了记忆方便，一般输入指定的目录，如C:\temp。
2. 当所有文件解压缩完成以后，打开temp目录，双击SETUP.EXE文件。好了，开始安装了，推荐适用默认的

安装路径，当然，用户也可以设定自己的安装路径，但是在指南中，我们使用默认路径。

好了，开发工具安装好了，就这么简单。现在你可以开始编写代码，在AVR器件上运行了。注意，将芯片资料和指令说明放在旁边，这些资料以后经常用到。

AVR器件的基本知识

AVR系列产品是采用新架构生产的芯片，整个系列都具有良好的相似性，芯片结构也非常容易理解和掌握。好了，现在我们就来开始看看AVR器件的基本知识吧。

首先，AVR系列产品可以分为3大种类：

- tiny AVR（简化版芯片）
- AVR（常用芯片）
- mega AVR（增强型）

这3类器件的区分是很明显的，譬如tiny芯片在管脚和功能上有所裁减。但是3类器件都采用同样的结构和存储器管理方式，如果将外围管脚以及一些特殊的模块除去，每一个AVR芯片都具有相同的内核，这样的性能保证升级芯片时非常的便利。

有些适用于特殊领域的AVR器件带有SRAM、EEPROM、扩展SRAM接口、ADC、UART等等模块。

选择合适的AVR芯片

从上面的介绍来看3类AVR芯片是有差别的，我们可以从芯片的性能上去区分：高性能的——mega AVR，简化版本的——tiny AVR，在两者之间的——标准AVR。

开始学习编写AVR代码

学习新的知识是非常有意思的，当然，也可能遇到一些小小的麻烦。对于学习AVR编程，一个可行的办法是读完全部的资料，但是这个方法在时间和对芯片的理解上可能有不小的问题。在这里，我们通过一个简单的方法，包括：

- 找一些事先写好、可以工作的代码
- 理解这些代码是如何工作的
- 按照自己的需要修改代码

在这次学习中，我们采用AT90S8515芯片，现在开始花点时间把芯片的说明书看一下。

学会看AVR的芯片说明书

看AVR的芯片说明书可能是一件非常恐怖的事情，因为AVR的芯片说明书有时长达350页。将这样的说明书从头到尾读完并且记忆下来是十分长并痛苦的。现在你不需要这样做，芯片说明书详细记录了芯片的技术资料，你只是在需要了解某个方面的时候才需要翻阅相关的内容。

打开AVR芯片的说明书，你会发现说明书大体上分成以下几个部分

- 第一页讲述关键信息和列表
- 总体介绍芯片架构
- 外围设备介绍
- 存储器编程
- 芯片性能

- 寄存器介绍
- 命令总结
- 封装信息

这样排列有非常大的便利，当你熟悉适用AT90S8515芯片说明书以后，再看其他AVR产品的说明书也会觉得非常容易。在这次整个学习过程中，你需要参看AT90S8515芯片数据手册中有关架构介绍部分的内容（在说明书的开始），这段内容包含了非常重要的信息，包括AVR芯片的存储器、地址以及其他信息。

另外一个非常重要的部分是命令总结。如果你要自己编写AVR代码的话，这部分的内容至关重要。如果你想深入了解命令的话，那就参看芯片数据手册前面的内容。

第四节：利用AVR STUDIO 4 进行开发

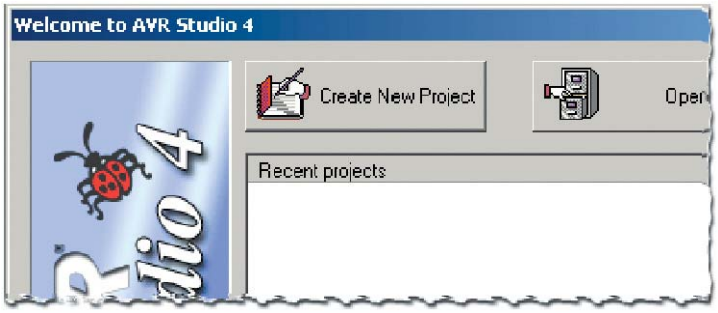
好了，现在你已经安装了开发软件，也知道了AVR的基本知识，也拥有了芯片数据手册，下面，我们就开始进行AVR芯片的开发吧。

AVR Studio 4的界面

提示：如果你还没有安装AVR Studio 4开发软件，你可以参考前面的章节来安装软件。

创建一个新的项目

启动AVR Studio 4 的方式如下：点击 开始-〉程序-〉ATMEL AVR工具。AVR Studio启动后，你将看到一个对话框。我们需要创建一个新的项目，点击“Create New Project”按钮。



配置项目参数

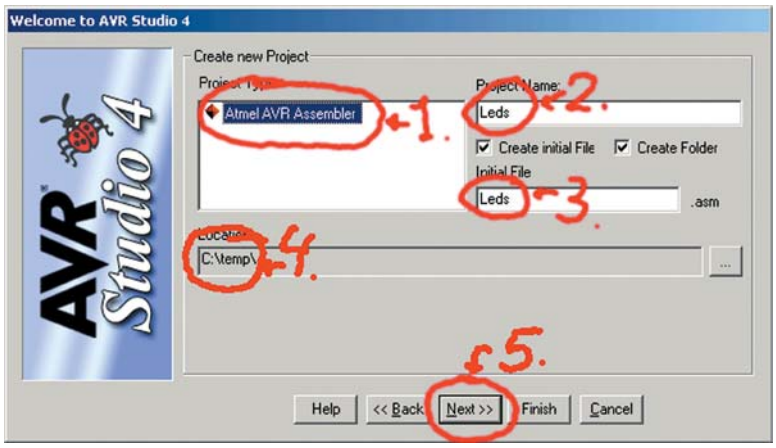
这个步骤包括选择我们要创建什么类型的项目，设定名称以及存放的路径。

这个过程包括五个步骤：

（1）在对话框左边选中Assembly program，表明你要创建一个项目。

（2）输入项目的名称。项目的名称可以随意定义，在例子中我们用了“Leds”。

（3）我们需要AVR Studio自动产生一个汇编文件，在例子中，我们用了“Leds”。

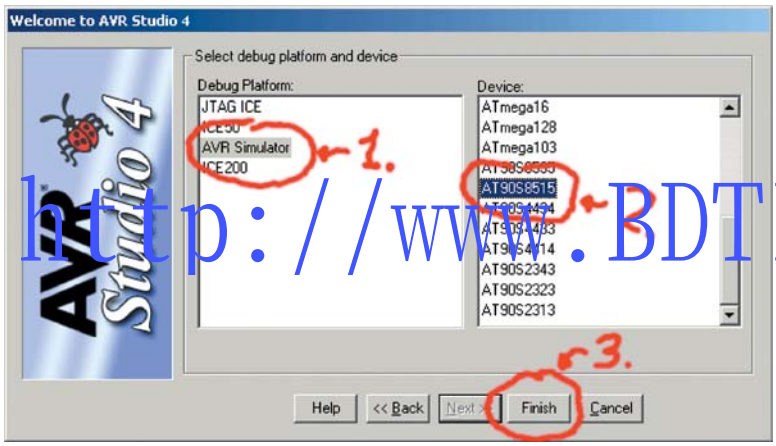


（4）选择你要存放项目的路径

（5）确认所有的选项，确认之后，按“Next”按钮。

选择调试平台

AVR Studio 4软件可以让客户选择多种开发调试工具。

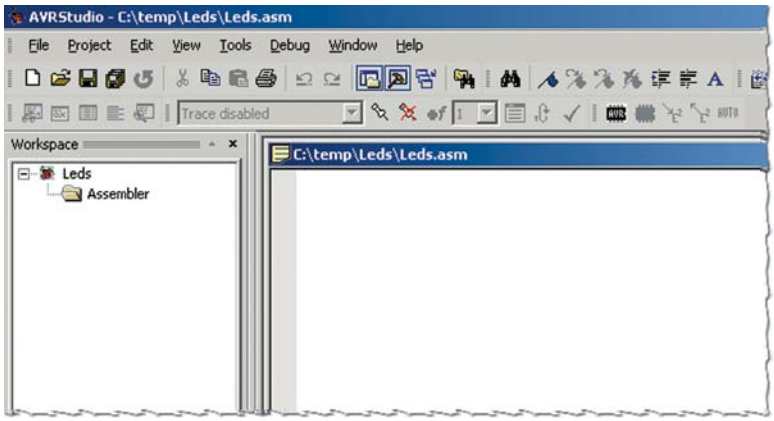


（1）AVR Studio 4允许可以选择多种开发调试工具，在这里我们选用带有仿真功能的AVR Simulator。

（2）芯片我们选用AT90S8515。

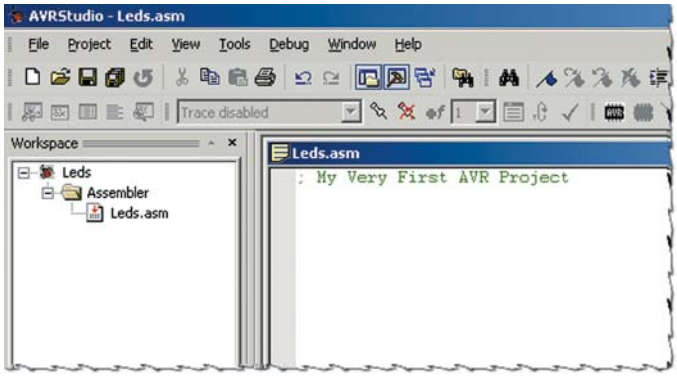
编写你的第一行代码

经过上面的步骤，AVR Studio打开了一个空的文件，文件的名字是Leds.asm。可能你注意到Leds.asm这个文件没有出现在左边的栏目中，这是因为这个文件还没有保存过。现在在文件中输入：“；My Very First AVR Project”，“；”的作用是注释，在编译时，分号以后的内容被忽略。



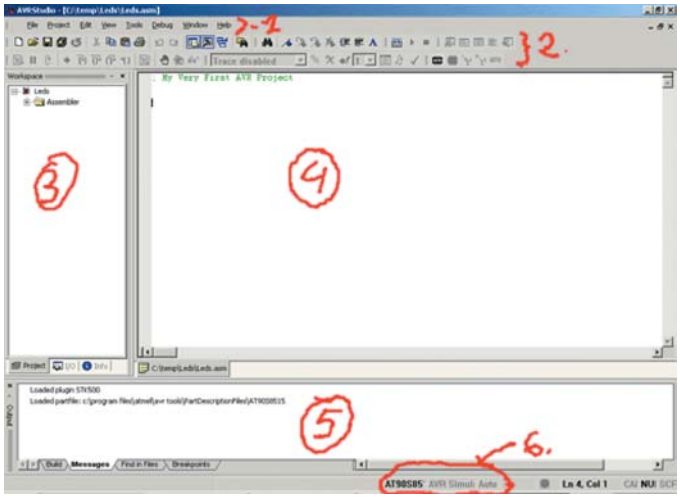
在File菜单中按下Save按钮，文件被保存。同时，文件也出现在左边栏目中。

好了，现在我们的AVR Studio 可以工作了，下面让我们再来关注AVR Studio 4的界面。



AVR Studio4 的用户图形界面（GUI）

让我们来自己观察一下AVR Studio4 的用户图形界面（GUI）。



我们把用户图形界面划分成了6个部分。在AVR Studio 4系统中包括了AVR Studio的帮助文件，在这里，我们着重介绍AVR Studio 4的框架和一些要注意的事项。

（1）第一行是菜单栏。这与标准的Windows程序差不多，包括打开/保存文件、剪贴/复制，这个栏目还包含了Studio的一些特殊功能，如仿真等。

（2）第二行是快捷方式栏，这一栏存储了一些常用命令，包括保存/打开文件，设置断点等等。

（3）第三部分为工作台窗口，在这里显示项目文件、IO状态以及项目选用AVR器件的信息

（4）第四是编辑窗口。在这里可以编辑你的源代码。对于熟练的用户，在这里也可以嵌入C代码

（5）第五是输出窗口，状态信息在这里显示

（6）第六是系统状态条。这里显示AVR Studio软件工作的模式，例如我们选用了AT90S8515芯片在仿真模式下工作，这些信息就会在系统状态条中显示。

用户图形界面深层指南

AVR Studio的用户图形界面制作的非常友好，用户不需要太多的知识就可以使用。但是，我们建议用户还是参看一下AVR Studio自带的HTML帮助文件。用户可以从AVR Studio软件的help->AVR Studio User Guide打开帮助文件。

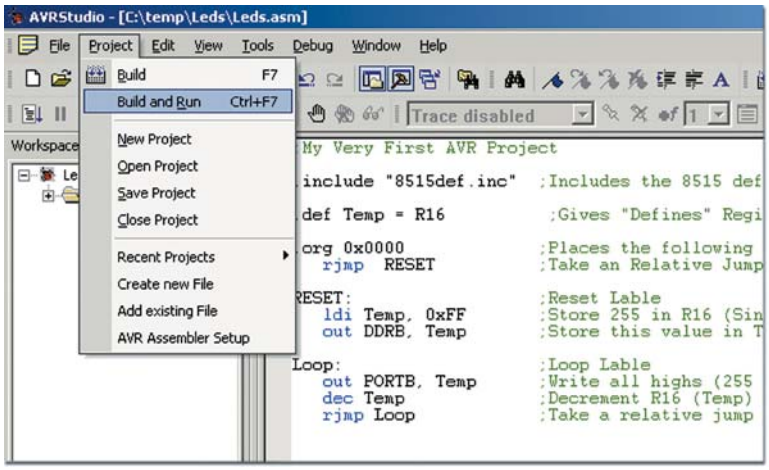
编写你的第一个别AVR程序

下面，我们继续完成我们的第一个AVR程序。到目前为止，我们安装了开发软件，创建了“Leds”项目，下面，我们就开始编写AVR代码了。

在AVR Studio的编辑窗口，继续完成代码，需要添加的代码如下（可以运用复制/粘贴方式将代码复制到编辑窗口）：


```
;My Very First AVR Project
.include "8515def.inc"        ;Includes the 8515 definitions file
.def Temp = R16               ;Gives "Defines" RegisterR16 the name Temp
.org 0x0000                   ;Places the following code from address 0x0000
     rjmp RESET               ;Take a Relative Jump to the RESET Label
RESET:                        ;Reset Label
     ldi Temp, 0xFF           ;Store 255 in R16 (Since we have defined R16 = Temp)
     out DDRB, Temp           ;Store this value in The PORTB Data direction
                              Register
Loop:                         ;Loop Label
     out PORTB, Temp          ;Write all highs (255 decimal) to PORTB
     dec Temp                 ;Decrement R16 (Temp)
     rjmp Loop                ;Take a relative jump to the Loop label
```

注意到在编辑窗口，代码的颜色发生了变化。这是编辑窗口的语法高亮功能，这个功能非常有利于增强代码的可读性能。代码输入完毕后，按CTRL+F7或者是Project菜单-〉Build and Run。



在输出窗口中（在屏幕的左下角），你将会看到项目编译，同时报告没有发现错误，同时，我们也看到我们的程序编译后代码包括6个字（12个Byte）。

恭喜，现在你已经编写了第一个AVR程序了，下面我们着重研究这段代码。

注意：如果你的代码不能编译，请检查汇编文件。特别是你可能将引用文件“8515def.inc”放在了别的目录，请在文件中加入完整路径，如“.include “c:\complete path\8515def.inc” ”。

理解源代码

理解源代码

到目前为止，我们编写的第一个程序编译没有问题，这是非常大的成就。但是我们还是来看看这段代码的含义。完整的代码如下：

```
;My Very First AVR Project
.include "8515def.inc"        ;Includes the 8515 definitions file
.def Temp = R16               ;Gives "Defines" RegisterR16 the name Temp
.org 0x0000                   ;Places the following code from address 0x0000
     rjmp RESET               ;Take a Relative Jump to the RESET Label
RESET:                        ;Reset Label
     ldi Temp, 0xFF           ;Store 255 in R16 (Since we have defined R16 = Temp)
     out DDRB, Temp           ;Store this value in The PORTB Data direction
                              Register
Loop:                         ;Loop Label
     out PORTB, Temp          ;Write all highs (255 decimal) to PORTB
     dec Temp                 ;Decrement R16 (Temp)
     rjmp Loop                ;Take a relative jump to the Loop label
```

现在，让我们一行一行来看这段代码

; My Very First AVR Project

分号（ ； ）开始的行是代码的注释。注释可以加在任何行的后面，如果注释超过了一行，每一行注释都要以分号开头

.include "8515def.inc"

不同的AVR器件还是有一些区别的，比如说PORTB就对应着不同的存储地址，.inc文件存储这种信息，在应用了这个文件以后，就可以把PORTB标号和具体的存储地址相对应（在AT90S8515芯片中，对应是OX18）。

.def Temp=R16

.def命令可以创建一个容易记忆的标号（如Temp）代替寄存器名称（如R16）。在项目中使用大量通用寄存器的时候，这个命令非常有效。（在芯片数据手册中有通用寄存器的介绍）

.org 0x0000

这个命令的作用是将下条指令定位在Flash存储器中地址为0x0000的单元，比如在这个例子中，下条指令RJMP指令就定位在0x0000地址（在Flash的开始）。这样做的原因在于芯片上电复位、复位信号有效后或是看门狗有效以后，芯片从0x0000开始执行指令。当然，这里也可以存放中断跳转指令。在这个例子中，我们没有利用中断，所以就存放了RJMP指令。

Rjmp RESET

前面我们介绍过了指令定位在0x0000，相对跳转指令（RJMP）指令就被存放在0x0000单元中，这条指令被首先执行。如果参看芯片说明书，你会发现AT90S8515不支持JMP指令，芯片只有RJMP指令。这原因是我们不需要JMP指令。如果你比较JMP 和

RJMP指令，会发现JMP指令更长，这将使得器件执行速度变慢、代码变大，而用RJMP也可以访问到所有的Flash存储器，因此，芯片不支持JMP指令。

RESET：

这个是标号。你可以把标号放在代码中任何地方。标号的作用在于区分跳转指令的不同分支。标号使用是必要及方便的，在编译的时候，编译器自动运算标号的正确地址。

Ldi Temp,0xFF

这是一个立即读取（Load Immediate）指令，这个命令将读取一个立即数，写入指定的寄存器。在上面，我们定义了Temp是R16，所以这条指令的动作是将立即数0xff放入寄存器R16。

Out DDRB,Temp

为什么我们不写成“Ldi
DDRB,Temp”？这是个好问题，现在来让我们参看命令手册。找到LDI和OUT指令，用户会发现LDI的语法是：“LDI Rd, K”，这就表明这个命令只能使用通用寄存器的R16到R31。参看OUT命令的语法：“OUT A,
Rr”，则表明这个命令可以使用R0到R31。
执行了这条指令以后，DDRB寄存器被置高。DDRB寄存器被置高，表明PORTB管脚被定义成输出脚。

Loop：

还是一个标号

out PORTB , Temp

现在我们把0xFF写入PORTB，如果现在有个真实的器件的话，我们去测量芯片的PORTB，会发现管脚为5V。值得注意的是这10个IO管脚是芯片经常使用的，所以有必要在芯片说明书中参看这部分管脚的资料。从芯片说明书中我们可以看到PORTB包括3个寄存器：PORTB、PINB和DDRB。PORTB为写入寄存器，PINB为读入寄存器，DDRB寄存器控制管脚是输入还是输出。

Dec Temp

DEC为减法指令，这个指令表明对R16寄存器减一操作，经过这个指令，Temp寄存器（也就是R16寄存器）里面的内容为0xFE。DEC是一条算术指令，AVR芯片支持广泛的算术指令。完整的指令列表请参看芯片说明书中的指令列表。

Rjmp Loop

通过这条指令，我们将程序跳转到Loop标号处，在每一次循环中，都把Temp的数值赋给PORTB。

到目前为止，我猜想你已经知道这段程序大体的功能了：我们做了一个计数器，这个计数器从255递减到0。那么到0后会怎么样呢？

仿真源代码

仿真源代码

AVR Studio 4可以在多种方式下工作，刚才编写代码的时候，我们是在编辑模式，现在我们进入调试模式。先让我们着重看一下软件界面

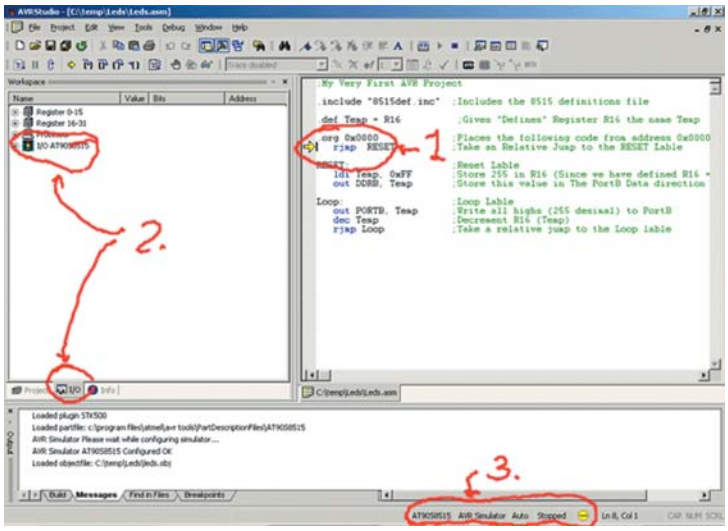
（1）注意到有一个黄色的箭头指向RLMP指令。这个箭头的作用是指向即将被执行的指令。

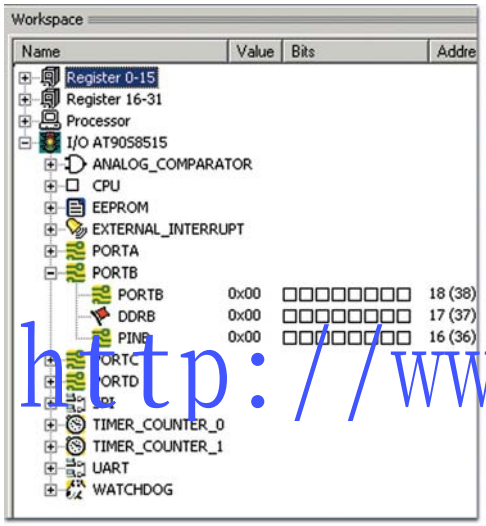
（2）注意到工作台窗口显示项目IO信息。IO信息是项目开发中最经常使用的信息，在下面我们将详细的介绍

（3）在底部状态栏显示当前状态。在本项目中显示：AT90S8515 simulator，Auto，Stopped。这里有一个黄色的图标。现在，最好检查一下显示信息，以确认选用的器件和仿真工具。

展开IO信息

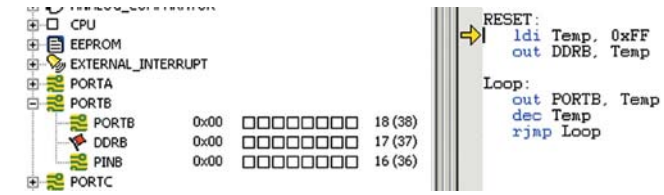
我们编写的项目主要是对PORTB存储器操作，所以我们将把IO信息展开，以观察IO的详细信息。展开IO信息树，将得到如图信息。



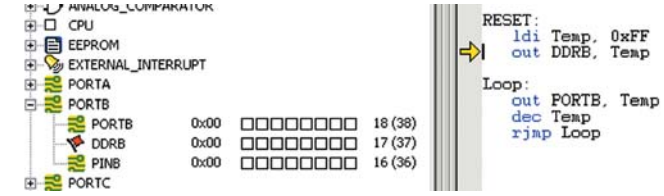


分段调试代码

AVR Studio开发软件支持分段调试代码。软件支持运行到断点，然后返回寄存器信息，并在此等待；也支持单步指令执行。按F11进行单步指令执行。

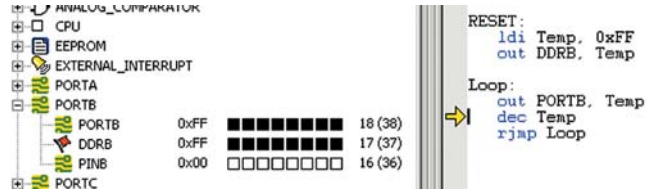


现在我们按一次F11，注意到黄色的箭头指向**LDI Temp, 0xFF**这条指令，表明这条指令即将被执行。

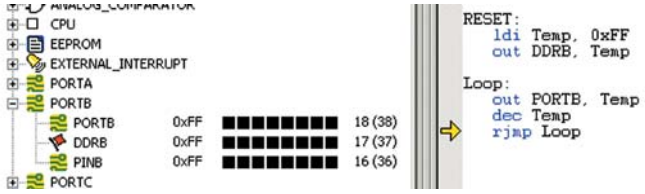


再按一次F11，LDI指令执行完毕，黄色箭头指向OUT指令。Temp寄存器的内容被赋值为0xFF（如果你观察R16寄存器，你会发现R16寄存器的内容也变成了0xFF，因为我们把Temp映射成R16了）。

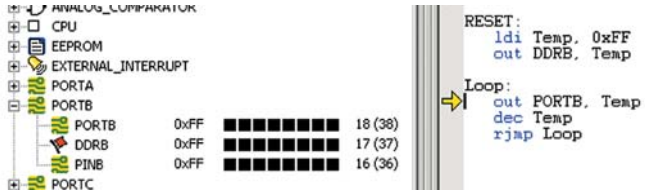
再按一次F11，如图所示，DDRB寄存器的内容被赋值为0xFF。在图中，一个白的方块表示0，黑的方块表示1。DDRB被设置成高电平表示所有的PORTB位设置为输出。



再按一次F11，0xFF写入PORTB寄存器，黄色箭头指向DEC指令。注意到PORTB寄存器内容是0xFF，但是PINB寄存器内容仍然是0x00。



再按一次F11，Temp的内容被减一，变成0xFE。同时注意到PINB寄存器内容变成了0xFF。用户可能变得非常疑惑，让我们来看芯片说明书中关于PORT的章节吧。数据首先锁存在输出管脚上，在延迟一个时钟周期以后，管脚上的数据锁存到PIN寄存器中。真如你所看到了，模拟仿真如实的反映了真实器件的工作步骤。



下一条指令是跳转指令，按一次F11，RJMP指令被执行，黄色箭头重新返回到OUT PORTB ,Temp指令。

按一次F11，新的Temp值再次赋予PORTB寄存器。一直按F11，你会发现PORTB寄存器的值可以一直递减到0x00，如果我连续运行程序，会有什么结果呢？

小结：

通过以上的步骤，用户已经对编写、运行AVR程序有了初步的认识。就像我们以前提到那样，最好的学习方法是寻找可以正常工作的代码实例，理解这些代码是如何工作的。在这里推荐一个网站：www.AVRfreaks.net，在这里有大量收集好的应用例子，值得你去学习。