

# **PicoRV32 IDE Quick Start**

## **应用指南**

**(版本号: V1.0)**

深圳市紫光同创电子有限公司

版权所有 侵权必究

## 文档版本修订记录

[illegible]

## 名词术语解释

[illegible]

## 目录

1.	概述.....	1
1.1.	介绍.....	1
1.2.	主要内容.....	1
1.3.	设计信息.....	1
2.	环境搭建.....	2
2.1.	安装Java JDK .....	2
2.2.	安装Eclipse.....	6
2.3.	安装python .....	6
3.	开发流程.....	7
3.1.	打开IDE软件.....	7
3.2.	打开工程.....	7
3.3.	生成hex文件.....	10
3.4.	工程讲解.....	10
3.5.	生成反汇编文件.....	13

## 图目录

图 1 安装向导 .....	2
图 2 安装路径选择 .....	2
图 3 安装进度显示 .....	3
图 4 JRE 安装路径选择 .....	3
图 5 JRE 安装进度条显示 .....	3
图 6 安装完成 .....	4
图 7 打开环境变量配置 .....	4
图 8 配置JAVA_HOME环境变量 .....	5
图 9 PATH环境变量添加 .....	5
图 10 检测安装是否成功 .....	6
图 11 PYTHON安装向导 .....	6
图 12 安装完成 .....	7
图 13 检测PYTHON安装是否成功 .....	7
图 14 IDE主界面 .....	7
图 15 打开工程 .....	8
图 16 工程文件列表 .....	8
图 17 编译工程 .....	9
图 18 查看*.BIN文件是否存在 .....	9
图 19 利用PYTHON生成*.HEX .....	10
图 20 将*.HEX文件路径放入RTL代码中 .....	10
图 21 主函数 .....	12
图 22 UART.C函数列表 .....	12
图 23 生成反汇编文件 .....	13

## 表目录

表 1 设计信息 .....	1
----------------	---

## 1. 概述

### 1.1. 介绍

本文档为深圳市紫光同创电子有限公司PicoRV32 IDE软件应用文档。主要介绍了PicoRV32集成开发环境的搭建以及使用C语言进行开发。

### 1.2. 主要内容

- 搭建PicoRV32 IDE开发环境
- IDE的简单使用操作
- 代码编译以及hex文件转换操作

### 1.3. 设计信息

表 1设计信息

提供的demo工程	
工程文件	pico_test_1
开发工具支持	
设计工具	Windows x64版本的Eclipse开发工具
	jdk-8u101-windows-x64
	python3x64

## 2. 环境搭建

### 2.1. 安装 Java JDK

在我们提供的文件夹里面找到jdk-8u101-windows-x64文件。

- 双击jdk-8u101-windows-x64.exe，进入安装向导
- 点击下一步



图 1 安装向导

- 选择安装路径



图 2 安装路径选择

- 下一步

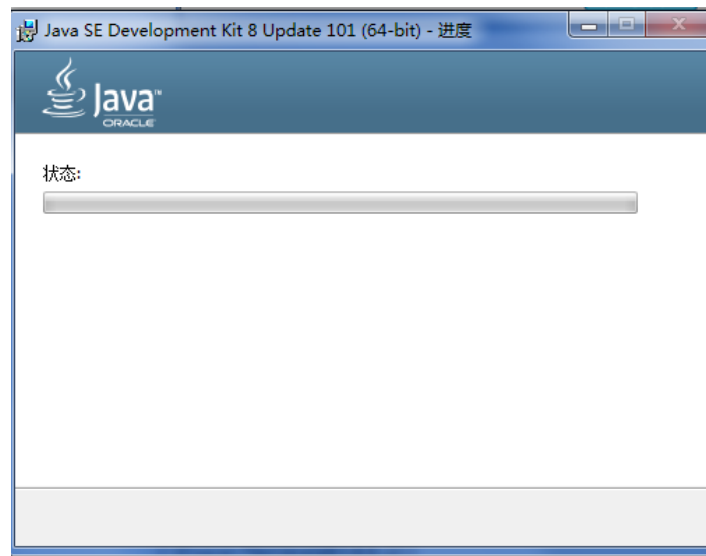


图 3 安装进度显示

➤ 更改jre安装路径



图 4 jre安装路径选择

➤ 下一步



图 5 jre安装进度条显示



- 点击关闭按钮，完成安装



图 6 安装完成

- 设置环境变量

右键计算机->属性->高级系统设置->环境变量->系统变量->新建

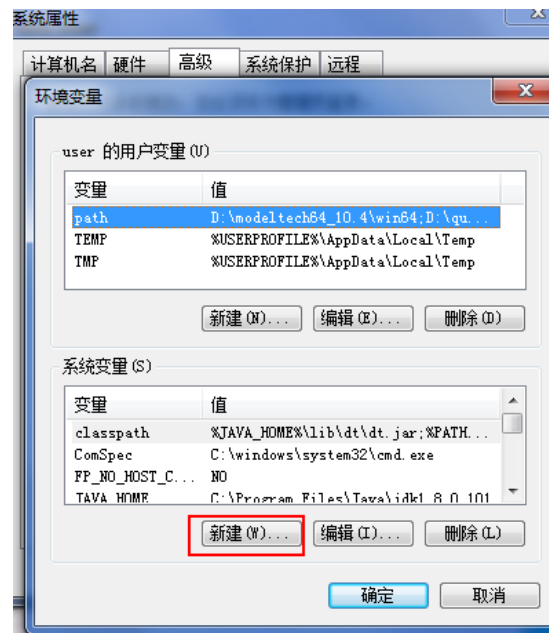


图 7 打开环境变量配置

- JAVA\_HOME环境变量设置

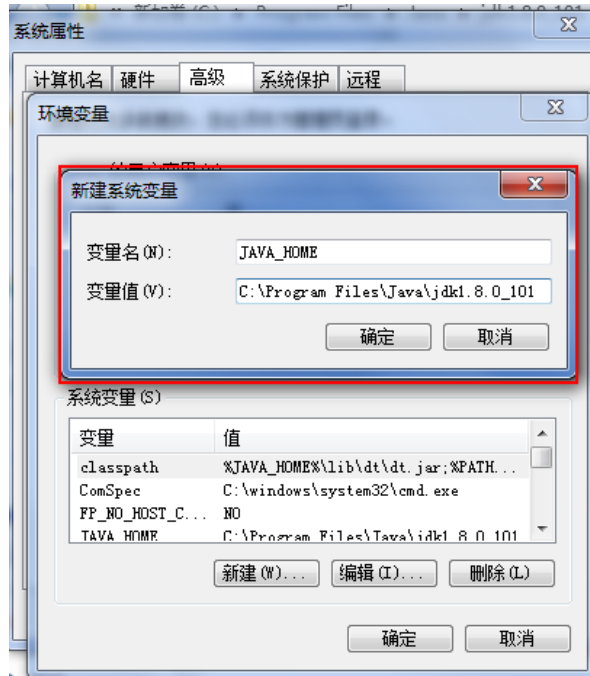


图 8配置JAVA\_HOME环境变量

- PATH指定可执行程序位置，变量值例如： %JAVA\_HOME%\bin;  
即： C:\Program Files\Java\jdk1.8.0\_101\bin; %.....%表示引用的意思

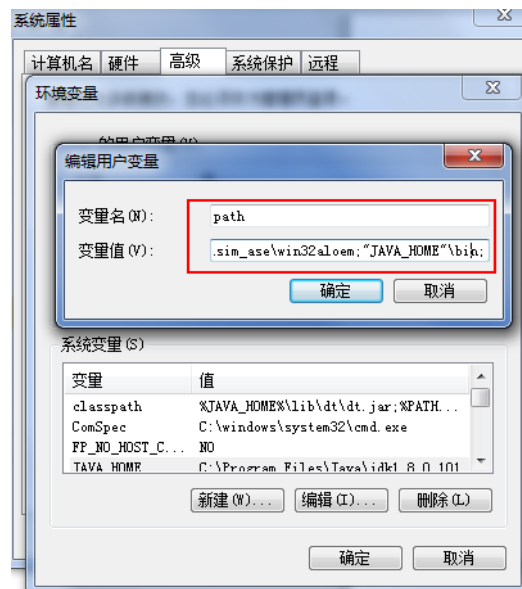


图 9 PATH环境变量添加

- 测试是否安装好java  
开始->运行->cmd 回车，弹出dos窗口  
输入： java -version 会出现版本信息，如出现如图所示信息，安装成功，反之则失败

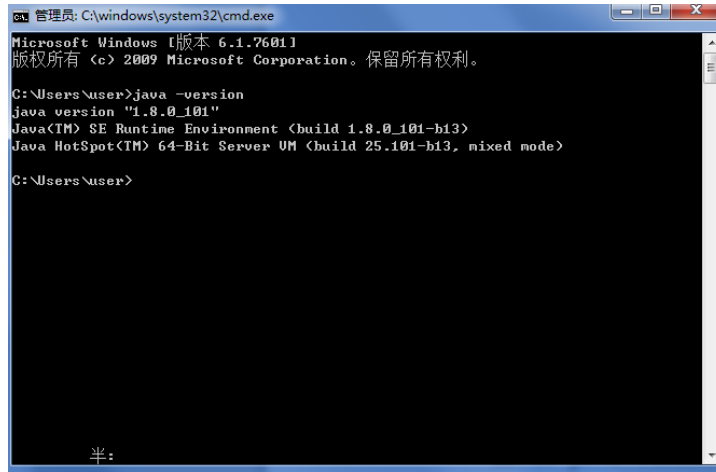


图 10检测安装是否成功

## 2.2. 安装 Eclipse

IDE软件是绿色软件，不需要安装，只需要将我们提供的编译环境安装包直接解压到D盘，文件名称为：Hummingbird，如：D:\Hummingbird，只需要一级目录。

进入：D:\Hummingbird\HBird-Eclipse\_2018\_09\20180110-1243-gnumcueclipse-4.3.1-oxygen-2-win32.win32.x86\_64\eclipse目录，直接双击eclipse.exe打开软件即可。

## 2.3. 安装 python

- 解压python3x64压缩包，进入解压目录，双击python-3.6.3-amd64文件开始安装
- 选择默认安装（Install Now）



图 11 python安装向导

红框处一定要勾选。

- 点击Close，完成安装

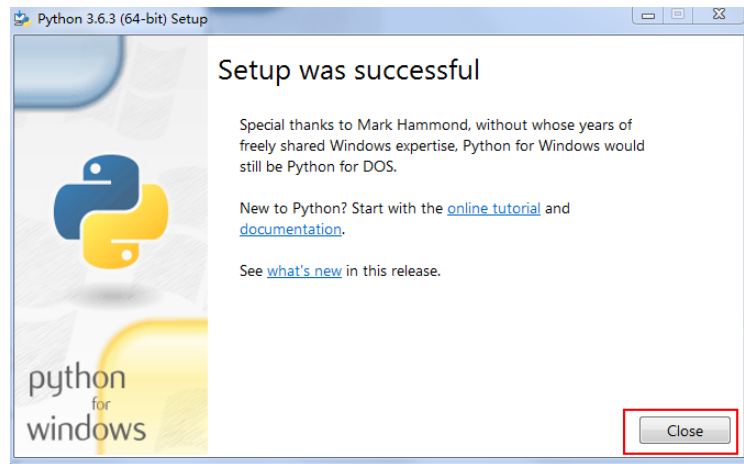


图 12安装完成

➤ 检测python是否安装完成

和检测java一样，首先进入dos界面，输入python，出现如下图所示信息，表示安装成功

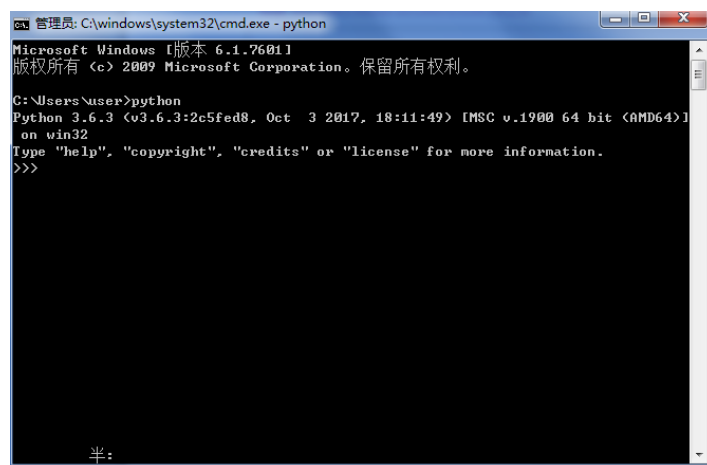


图 13检测python安装是否成功

自此，环境搭建完成。

## 3. 开发流程

### 3.1. 打开 IDE 软件

- D:\Hummingbird\HBird-Eclipse\_2018\_09\20180110-1243-gnumcueclipse-4.3.1-oxygen-2-win32.win32.x86\_64\eclipse目录，直接双击eclipse.exe就可以打开软件

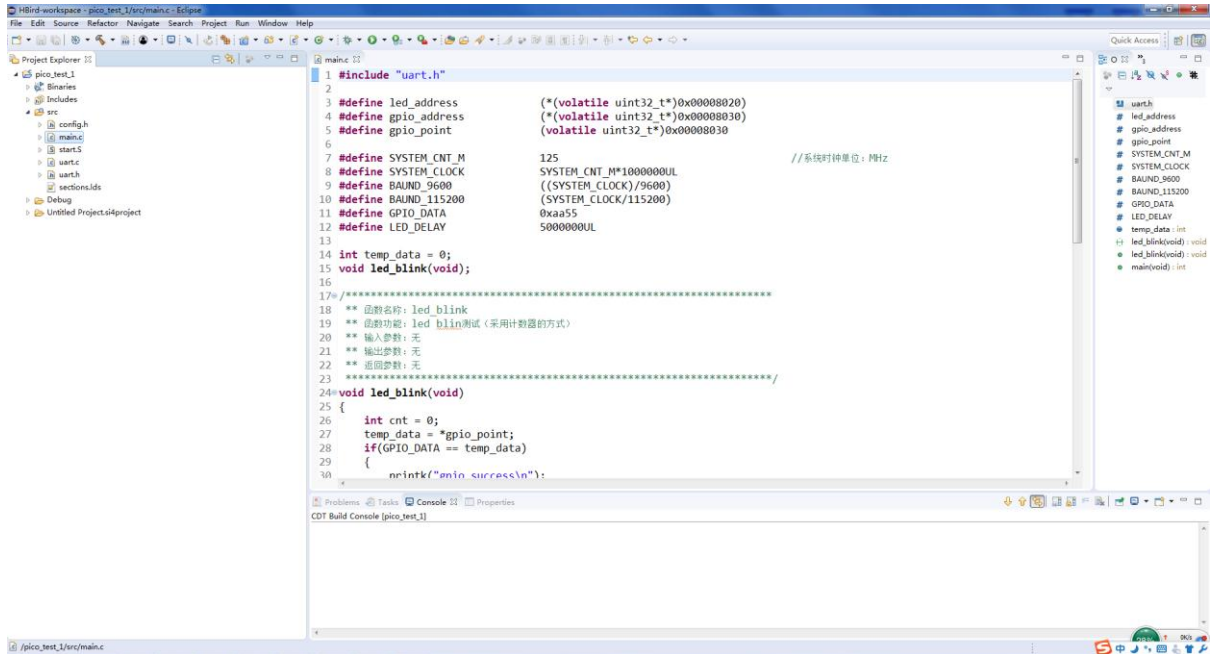


图 14 IDE主界面

### 3.2. 打开工程

我们这里使用的是直接打开一个已经建好的工程，而不是导入工程。

- 鼠标左键点击File菜单选项，在下拉列表中选择Open Project from File System，点击后弹出如下图所示窗口，首先导入工程文件的目录，再点击Finish。

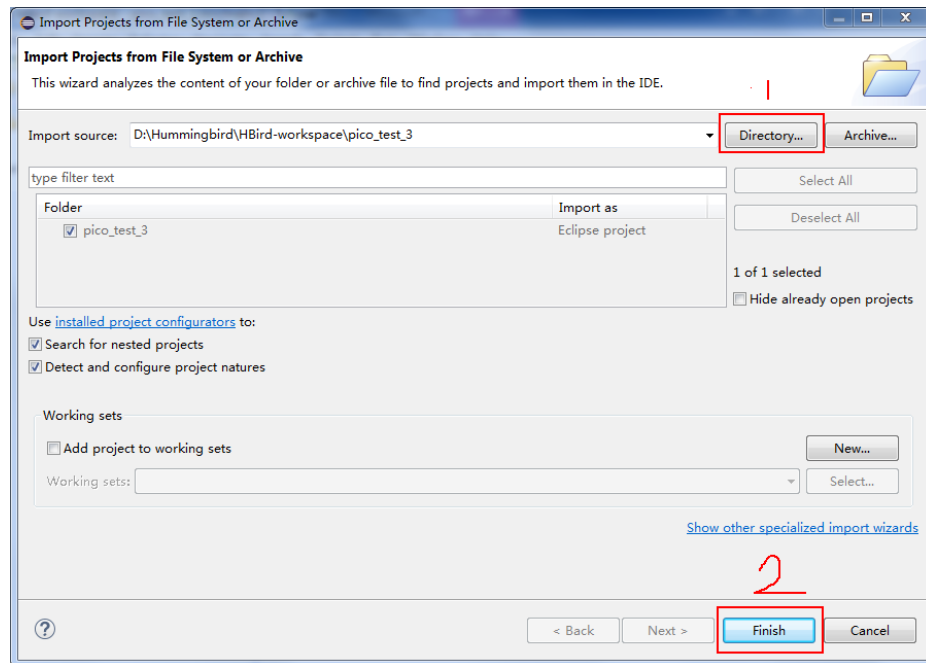


图 15打开工程

- 点击Finish以后，工程文件就被添加到Project Explorer菜单栏中，如下图所示：

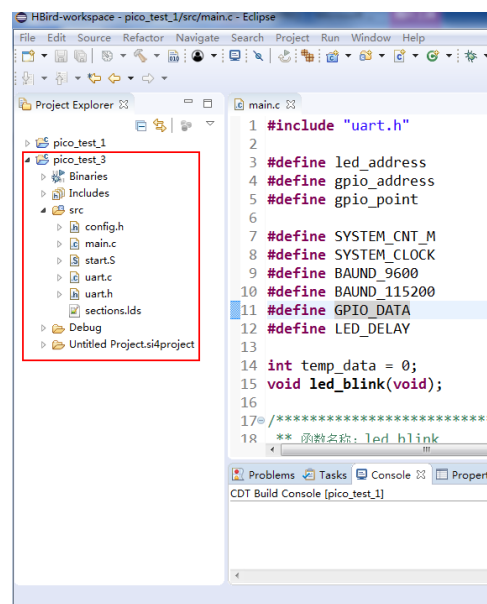


图 16工程文件列表

- 编译工程，鼠标左键点击工程，然后鼠标右键弹出下拉列表，先点击Clean Project，再点击Build Project，如下图所示：

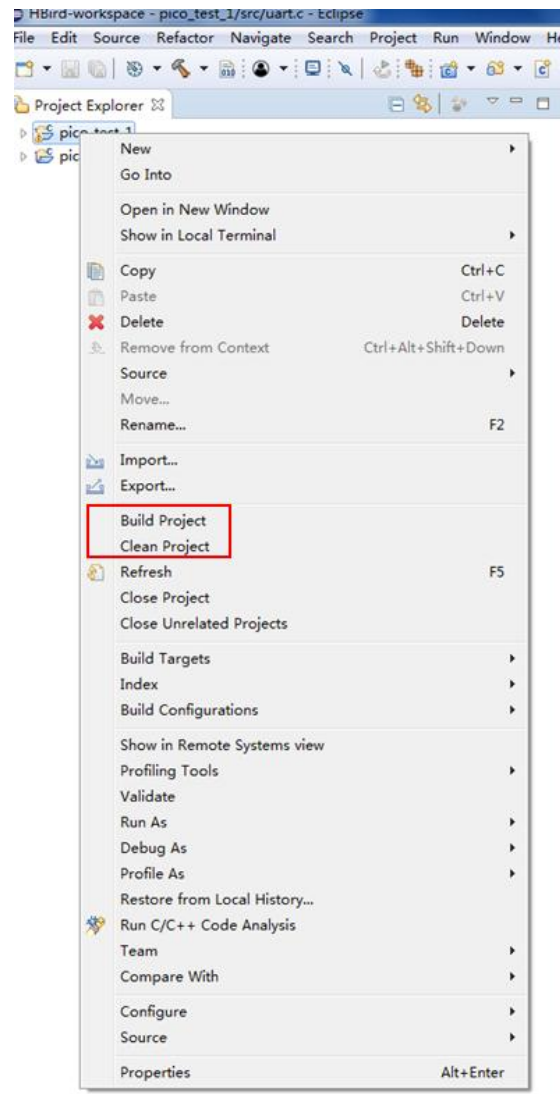


图 17编译工程

- 编译完成，可以在工程目录看到生成了.bin文件，如果没有，左键点击Debug，然后鼠标右键，选择Refresh按钮，bin文件如下图所示：

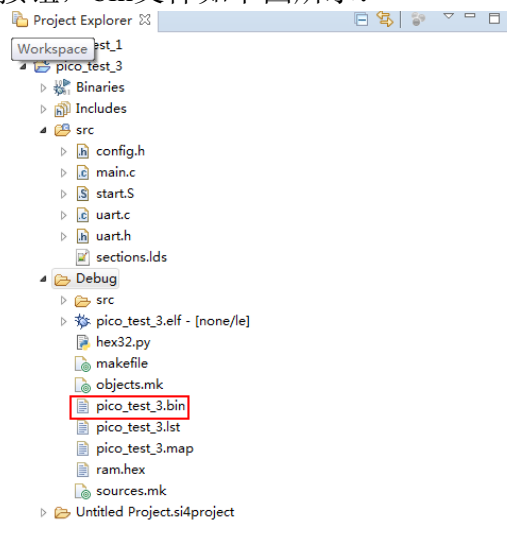


图 18查看\*.bin文件是否存在

### 3.3. 生成 hex 文件

进入d盘：d:;

进入bin文件夹：cd D:\Hummingbird\HBird-workspace\pico\_test\_3\Debug

执行转换操作：hex32.py <bin file name>

操作过程如下图所示：

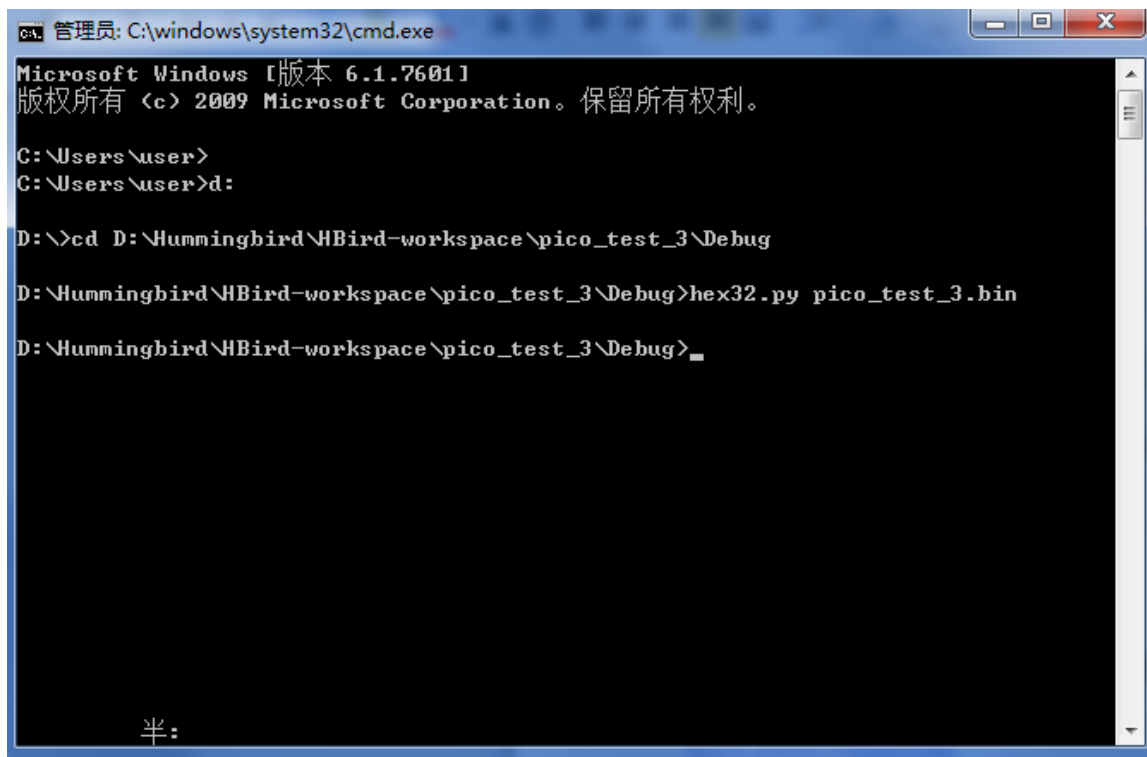


图 19利用python生成\*.hex

可以看到在Debug文件夹目录下面生成了hex文件，

- 在RTL设计中，pico源码文件mem.v文件中（或者叫其他名字），将hex文件的绝对路径放到readmemh语句里面（hex文件也可以放到RTL工程的当前路径），这样就可以在综合时将hex文件导入到mem中。如下图所示：

```
//0-1023 `sram, 1024-8191 cmd
reg [31:0] memory [0:CMD_NUM-1]/* synthesis syn_ramstyle = "block_ram" */;
initial begin
    $readmemh("ram_c.hex", memory, 1024);
end
```

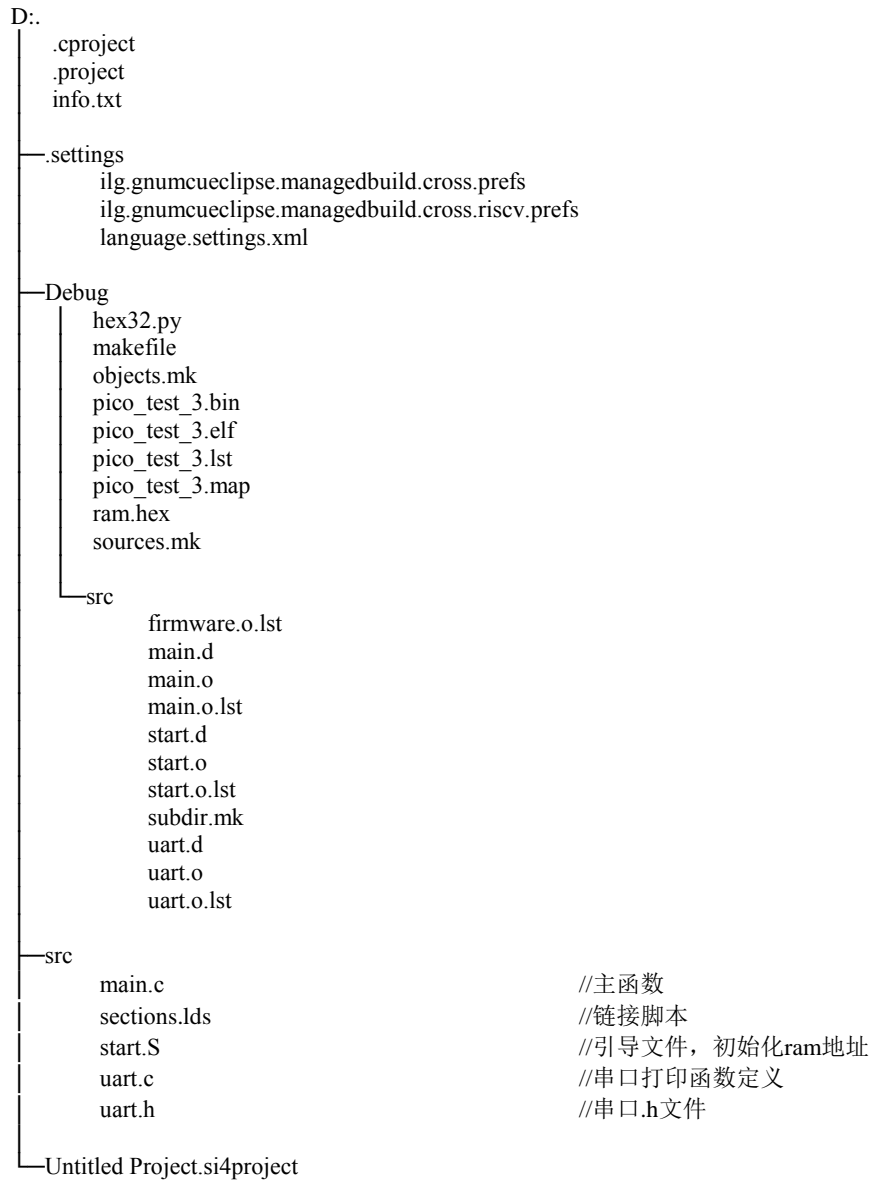
图 20将\*.hex文件路径放入RTL代码中

- 综合以及布局布线完成以后，下载可执行文件到开发板。详细的RTL设计文档请参见《PicoRV32 DEMO应用指南》

### 3.4. 工程讲解

工程文件夹树状图如下：





这里我们只看一部分，主要讲解一下工程文件下面src文件夹下面的文件。查看完整的树形图用户可以在进入工程文件下使用cmd指令：**tree/f**查看完整的树形图。工程目录下面的文件目录.txt文件对工程文件目录有详细讲解。

src文件夹里面包含的是整个项目的工程源码，以及连接脚本。

main.c文件，我们主要实现了打印、led blink以及读取指定寄存器值的功能，如下图所示：

```

main.c uart.c uarth uarth
25 {
26     int cnt = 0;
27     temp_data = *gpio_point;
28     if(GPIO_DATA == temp_data)
29     {
30         printk("gpio success\n");
31     }
32     else
33     {
34         printk("gpio failed\n");
35     }
36
37     led_address = 0xff;
38     while(1)
39     {
40         cnt++;
41         if(cnt == LED_DELAY / 2)
42         {
43             led_address = 0x00;
44         }
45         else if(cnt == LED_DELAY)
46         {
47             cnt = 0;
48             led_address = 0xff;
49             printk("led blink ok\n");
50         }
51     }
52 }
53
54 int main(void)
55 {
56     reg_uart_clkdiv = BAUND_9600;
57
58     printk("Hello Risc-V Pango 2019\n");
59     printk("test data = %d,%f,%s,0x = %x\n", 100,33.456,"2019",100);
60     printk("simple compute : 50*10 = %d,100/3 = %f,100%3 = %d\n", 50*10,(double)100/3,(int)100%3);
61     led_blink();
62     return 0;
63 }
64

```

图 21主函数

主函数主要实现了字符串打印，led灯的闪烁，读取指定地址的值。

字符串打印以及一些简单的计算，我们自己编写了printf函数，这里用printk表示，自带的printf需要消耗很大的资源，所以我们自己编写。

uart.c文件，我们实现了单字符打印、字符串打印、浮点数打印、十进制打印等。

**提示：**建议用户在我们搭建好的工程基础上做开发，可直接在IDE里面复制粘贴工程，不建议重新搭建新的工程。

```

Out... Ta... Bui... Do...
stdint.h
stdbool.h
uart.h
print_num(uint32_t, int) : void
put_char(char) : void
print_str(const char*) : void
print_hex(uint32_t, int) : void
print_hex2(uint32_t) : void
print_dec(uint32_t) : void
print_ftt(double) : void
printk(char*, ...) : void
getchar_prompt(char*) : char
get_char() : char

```

图 22 uart.c函数列表

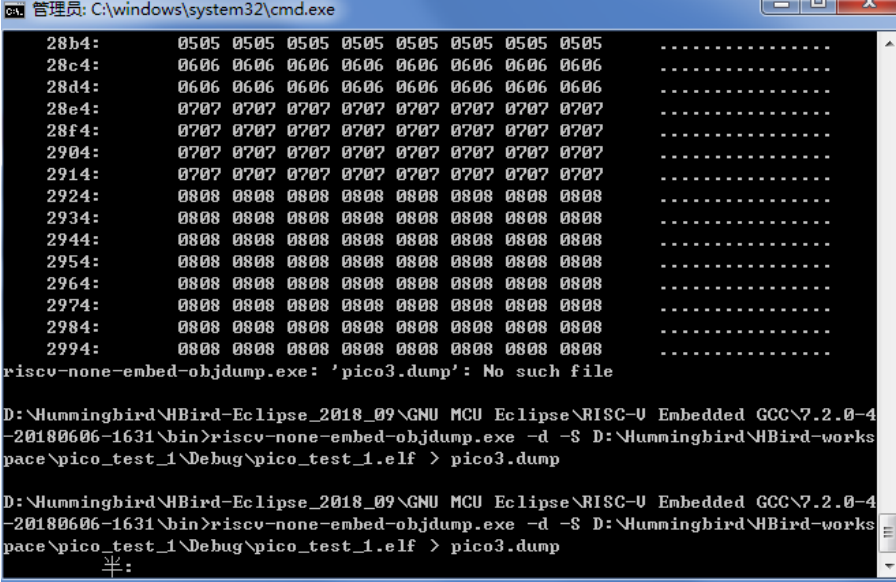
注意：在编写代码中，空函数会被编译器优化掉，例如：`for(i = 0; i < 10; i++);`仅仅写一个for循环语句，仿真出来实际上没有执行相应的时钟周期数量。被编译器优化掉了。在函数调用时，必须对内核或自定义的寄存器地址进行操作。

### 3.5. 生成反汇编文件

查看源码的反汇编指令用户可以进行如下操作：

```
cd D:\Hummingbird\HBird-Eclipse_2018_09\GNU MCU Eclipse\RISC-V Embedded GCC\7.2.0-4-20180606-1631\bin
riscv-none-embed-objdump.exe -d -S D:\Hummingbird\HBird-workspace\pico_test_1\Debug\pico_test_1.elf > pico.dump
```

也可以导出到绝对路径。



```

28b4: 0505 0505 0505 0505 0505 0505 0505 0505 .....
28c4: 0606 0606 0606 0606 0606 0606 0606 0606 .....
28d4: 0606 0606 0606 0606 0606 0606 0606 0606 .....
28e4: 0707 0707 0707 0707 0707 0707 0707 0707 .....
28f4: 0707 0707 0707 0707 0707 0707 0707 0707 .....
2904: 0707 0707 0707 0707 0707 0707 0707 0707 .....
2914: 0707 0707 0707 0707 0707 0707 0707 0707 .....
2924: 0808 0808 0808 0808 0808 0808 0808 0808 .....
2934: 0808 0808 0808 0808 0808 0808 0808 0808 .....
2944: 0808 0808 0808 0808 0808 0808 0808 0808 .....
2954: 0808 0808 0808 0808 0808 0808 0808 0808 .....
2964: 0808 0808 0808 0808 0808 0808 0808 0808 .....
2974: 0808 0808 0808 0808 0808 0808 0808 0808 .....
2984: 0808 0808 0808 0808 0808 0808 0808 0808 .....
2994: 0808 0808 0808 0808 0808 0808 0808 0808 .....
riscv-none-embed-objdump.exe: 'pico3.dump': No such file

D:\Hummingbird\HBird-Eclipse_2018_09\GNU MCU Eclipse\RISC-V Embedded GCC\7.2.0-4-20180606-1631\bin>riscv-none-embed-objdump.exe -d -S D:\Hummingbird\HBird-workspace\pico_test_1\Debug\pico_test_1.elf > pico3.dump

D:\Hummingbird\HBird-Eclipse_2018_09\GNU MCU Eclipse\RISC-V Embedded GCC\7.2.0-4-20180606-1631\bin>riscv-none-embed-objdump.exe -d -S D:\Hummingbird\HBird-workspace\pico_test_1\Debug\pico_test_1.elf > pico3.dump
半:

```

图 23生成反汇编文件