

# RT-Thread Programming Guide

RT-Thread Development Team<sup>1</sup>

2013-01-18

<sup>1</sup>This is the PDF file for the RT-Thread Programming Guide book contents. It is licensed under RT-Thread license.



# 目录

目录	i
1 原生模式编程	1
1.1 工程 . . . . .	1
1.2 用户应用 . . . . .	2
1.3 编译运行 . . . . .	4
2 测试	7



## 第 1 章

# 原生模式编程

从[github](#)上[下载基本工程](#)，这是一个实现最基本功能的工程，对于想接触RT-Thread编程的爱好者，这个是最基本也是最容易入门的工程。

如果喜欢使用Keil MDK，可以通过如下方式转换出Keil MDK的工程文件。

```
scons --target=mdk -s
```

如果喜欢iar，可以通过如下方式获得工程文件

```
scons --target=iar -s
```

这个工程包括的功能：

- \* 基本的内核
- \* 基本的命令行
- \* 如果使用USB连接电脑，可以使用USB虚拟的串口
- \* 如果使用真实串口，可以使用TTL线连接RX0，TX0

这里推荐使用USB线直接连接方式（但使用USB方式会忽略掉启动时信息。以及在系统当机时也不能输出出错信息）。

ART板上也包括一个SWD调试接口，可用于连接支持swd的仿真器，例如常用的ulink2，jlink等仿真器。当使用仿真器连接swd时，可以配合Keil MDK或IAR等工具进行源代码级别的单步调试。

ART swd连接20 pin常规JTAG如下图所示：

请注意图中的20 pin JTAG缺口是朝上的。

### 1.1 工程

假设使用Keil MDK来进行工程的编译和调试，请在rtconfig.py中修改交叉工具链为keil，如下所示：

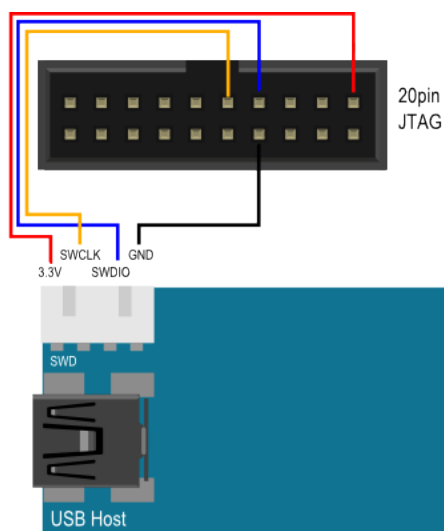


图 1.1: SWD

```
PLATFORM = 'armcc'
```

然后在命令行下执行:

```
scons --target=mdk -s
```

正确的情况下，会在basic目录下生成project.uvproj文件。可以用Keil MDK直接打开这个工程文件。打开后工程会有几个默认的组（Group）：

其中每个组的对应情况：

- \* Applications - 用户自己的应用程序；
- \* Drivers - RT-Thread在ART上基本的驱动程序；
- \* STM32\_StdPeriph - ST的STM32固件库；
- \* Kernel - RT-Thread内核代码；
- \* CORTEX-M4 - RT-Thread在ARM Cortex-M4上的移植；
- \* DeviceDrivers - RT-Thread的设备驱动框架；
- \* finsh - RT-Thread的finsh shell；
- \* Components - RT-Thread的组件管理器，当前主要用于RT-Thread组件的初始化；

## 1.2 用户应用

在RT-Thread中，通常会提供一个applications.c文件，在这个文件中初始化用户自己的应用（组件初始化也会在这里调用），例如这个工程中的applications.c文件：

```
#include "stm32f4xx.h"
#include <board.h>
```

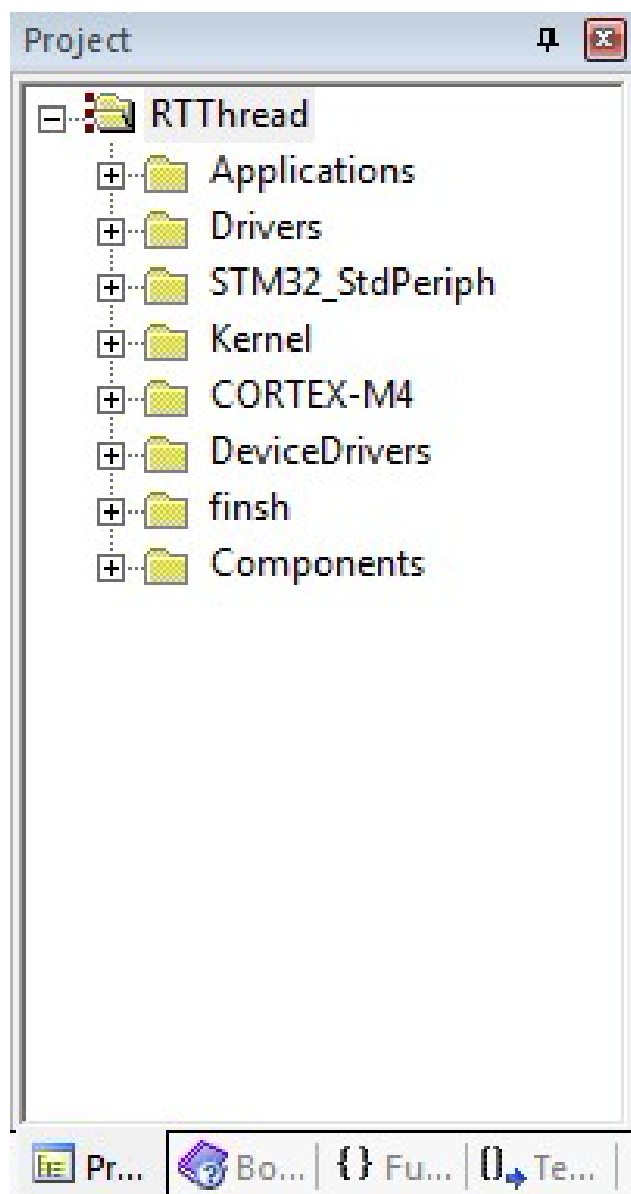


图 1.2: SWD

```

#include <rtthread.h>

#include <components.h>
static void thread_entry(void* parameter)
{
    rt_components_init();

#ifdef RT_USING_USB_DEVICE
    /* usb device controller driver initilize */
    rt_hw_usbd_init();

    rt_usb_device_init("usbd");

    rt_usb_vcom_init();

#endif
#ifdef RT_USING_CONSOLE
    rt_console_set_device("vcom");
#endif
#ifdef RT_USING_FINSH
    finsh_set_device("vcom");
#endif
}

int rt_application_init(void)
{
    rt_thread_t tid;

    tid = rt_thread_create("init",
        thread_entry, RT_NULL,
        2048, RT_THREAD_PRIORITY_MAX/3, 20);

    if (tid != RT_NULL)
        rt_thread_startup(tid);

    return 0;
}

```

函数: `rt_application_init` 是用户任务的入口点, 在这里创建了一个初始化任务用于多任务级别的初始化, 其入口函数是: `thread_entry`。在这个任务中, 它调用了 `rt_components_init` 函数以初始化系统的组件 (在这个工程中用到了 `finsh shell` 以及 `usb device stack`, 所以这两个组件会分别在 `rt_components_init` 函数中进行初始化)。

### 1.3 编译运行

使用生成的 Keil MDK 工程, 我们可以使用 Keil MDK 自己携带的编译器进行编译、下载:

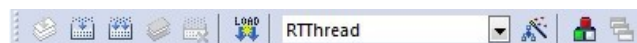
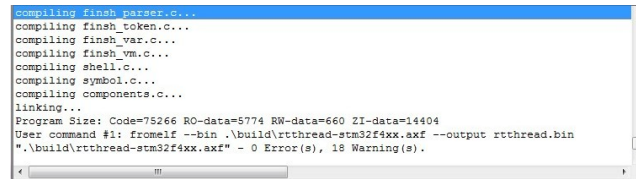


图 1.3: Build

编译完成后, 应该会在 Keil MDK 的输出窗口中显示成功:





```
compiling finsh_parser.c...
compiling finsh_token.c...
compiling finsh_var.c...
compiling finsh_vm.c...
compiling shell.c...
compiling symbol.c...
compiling components.c...
linking...
Program Size: Code=75266 RO-data=5774 RW-data=660 ZI-data=14404
User command #1: fromelf --bin .\build\rtthread-stm32f4xx.axf --output rtthread.bin
".\build\rtthread-stm32f4xx.axf" - 0 Error(s), 18 Warning(s).
```

图 1.4: BuildOk

编译完成后可以下载到ART板子上，同时可以使用PuTTY工具打开ART的虚拟USB串口，进入到RT-Thread的命令行状态。



## 第 2 章

# 测试

代码示例:

```
#include <stdio.h>

int main(int argc, char** argv)
{
    printf("Hello World\n");
    return 0;
}
```