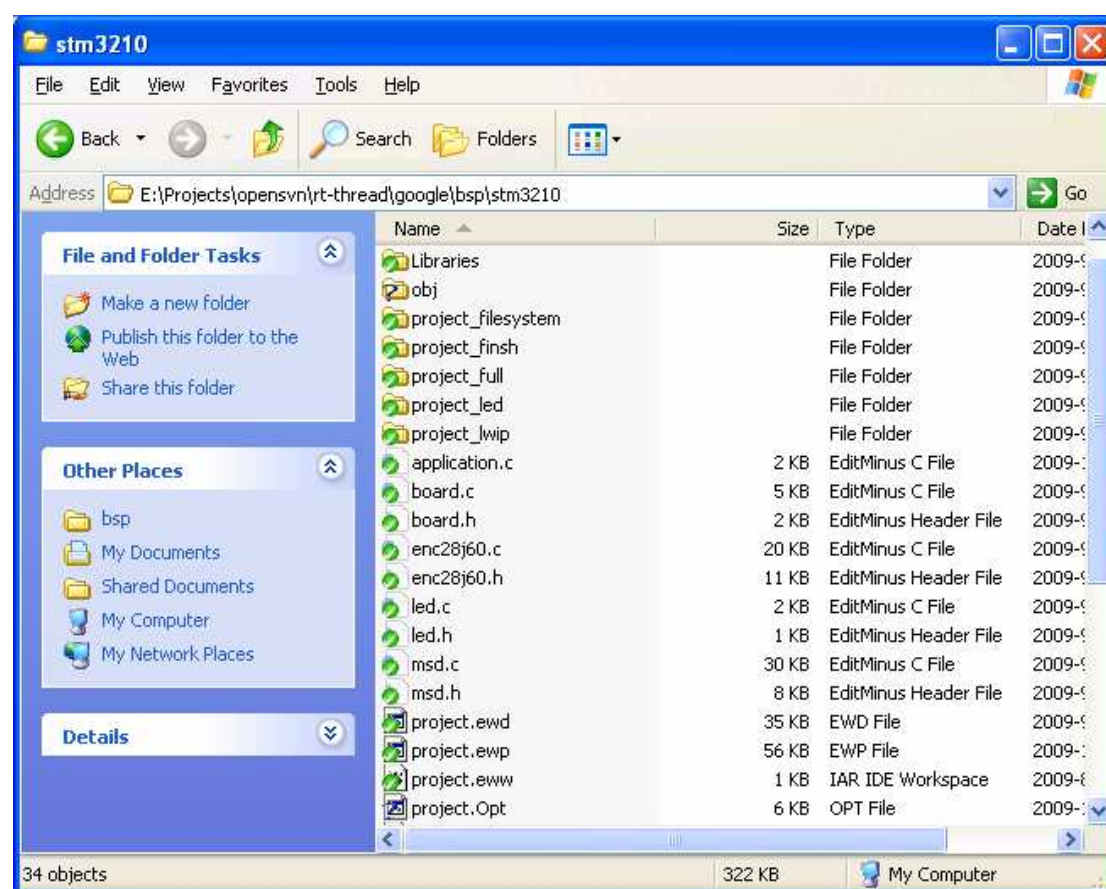


STM32 跑马灯工程详细解析

先下载下来 RT-Thread 的代码，针对于 STM32 的 MDK、IAR 工程放在 bsp/stm3210 目录下：project.Uv2、project.eww。

RT-Thread/STM32 的版本包括了五种工程，分别对应不同的功能特性剪裁。刚下载的发布，bsp/stm3210 目录下的工程文件默认是全功能的版本，即包括了 RT-Thread 的全部特性：实时核心 + Shell + 文件系统 + TCP/IP 协议栈，相当于把所有的特性都打开了。



五个不同的版本分别对应

- project_led
- project_finsh
- project filesystem
- project_lwip
- project_full

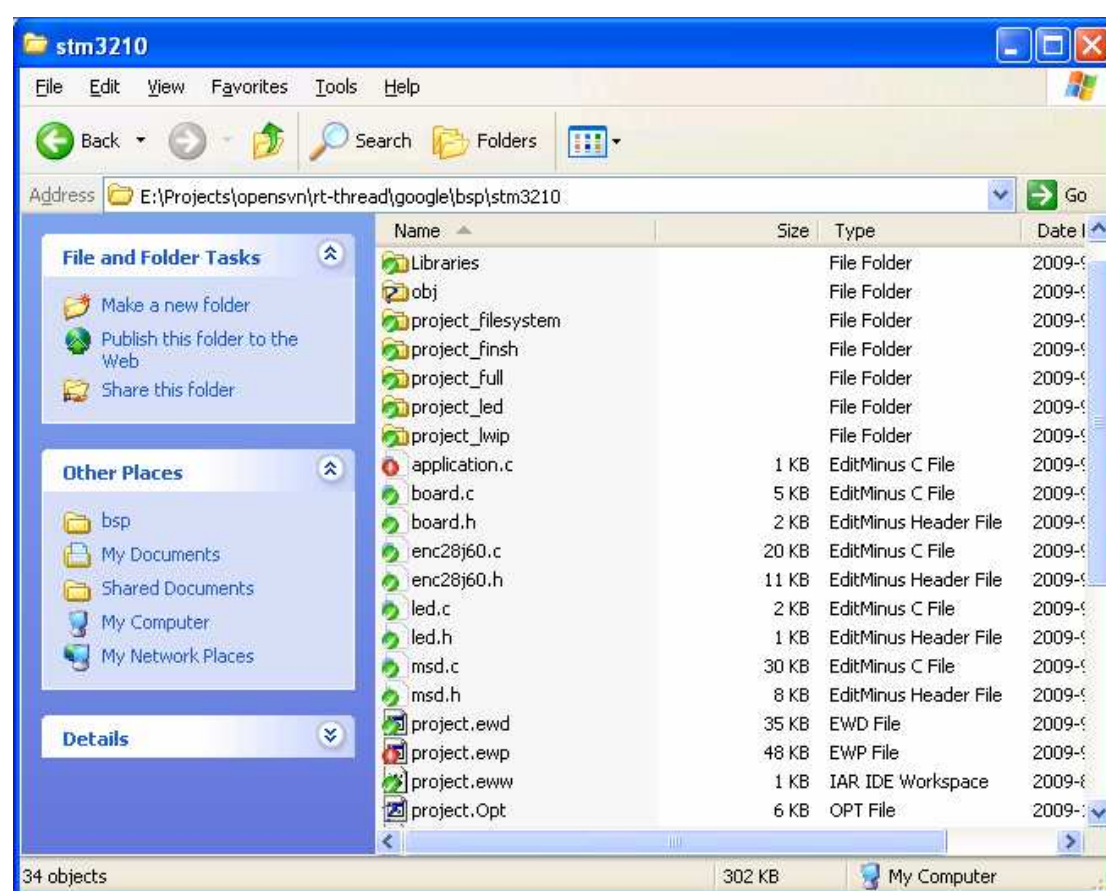
五个不同的子目录。

跑马灯工程

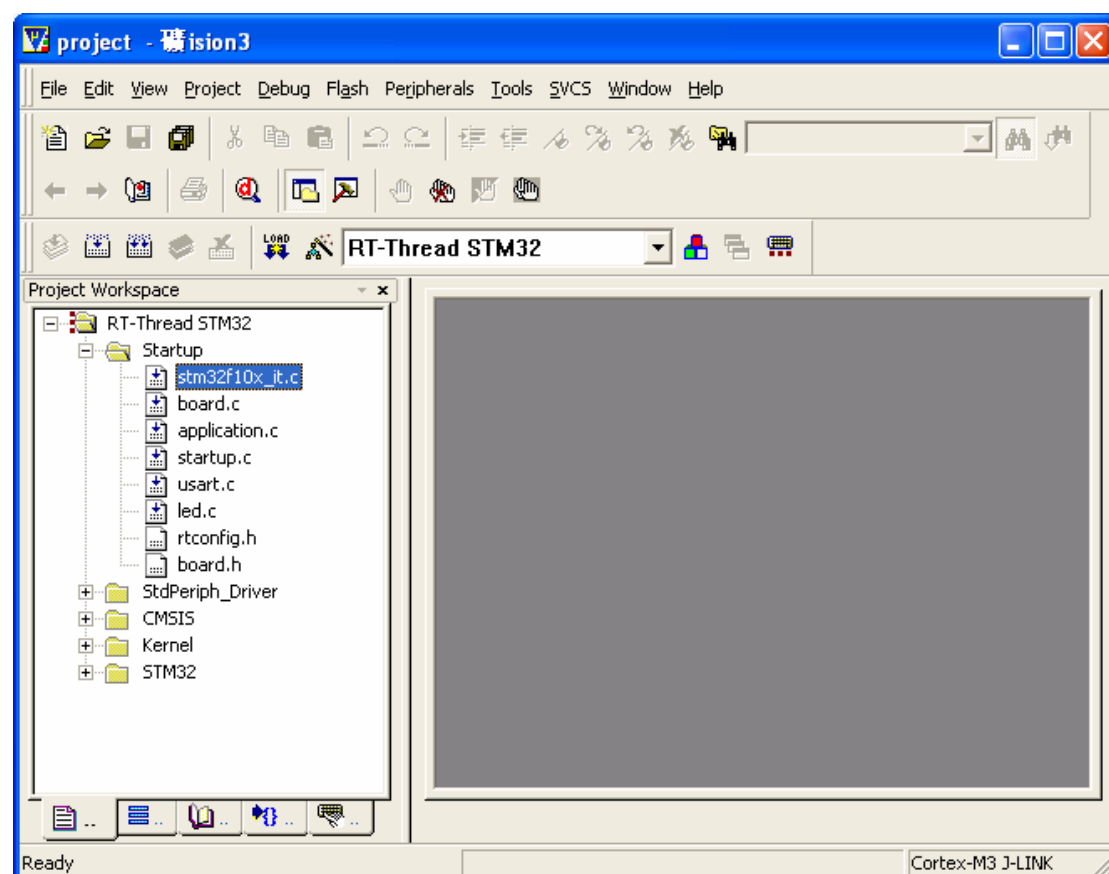
本篇文档说的是 RT-Thread 最基本的版本，一个内核 + 跑马灯的例子：系统跑起来后，板子上的 LED 灯会定时闪烁（0.5 秒）。

使用跑马灯工程文件

为了使用这个例子，需要把 `bsp/stm3210/project_led` 目录下的所有文件复制到 `bsp/stm3210` 目录下(如果是 svn 取下来的版本，.svn 目录就不用复制过去了)。Keil MDK 打开 `project.Uv2` 工程文件，IAR 打开 `project.eww` 工程文件。



用 `project_led` 目录文件覆盖上一层目录文件图例



用 Keil MDK 3.8 打开工程文件的图例

其中包含了五个 Group:

- **Startup** – 用户文件及和开发板密切相关的组，RT-Thread 也是从这里开始启动的；
- **StdPeriph_Driver** – STM32 的 ST 官方固件库，目前是 v3.1.0；
- **CMSIS** – ST 官方给出的 ARM CMSIS 接口；
- **Kernel** – RT-Thread 的实时核心；
- **STM32** – RT-Thread 针对 STM32 的移植。

这五个 Group 中，通常 Kernel，STM32 是不需要修改的，StdPeriph_Driver、CMSIS 也不用修改（但可以根据情况进行剪裁，请参考 ST 官方的文件）。

不同的 STM32 开发板有不同的 LED 配置，请检查你开发板的配置，默认使用的是 U-EasyTech 的 STM32F103ZE 开发板，LED 的配置分别为：PF6，PF7，PF8，PF9 四盏灯。

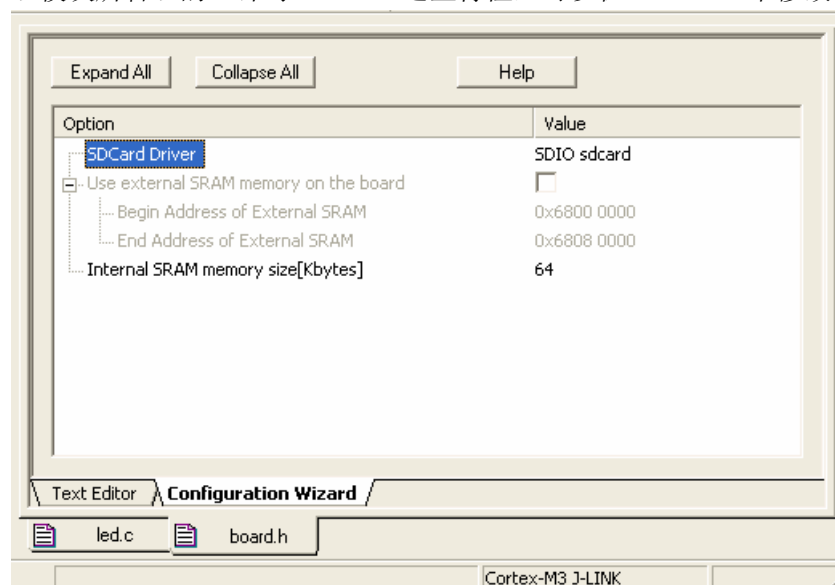
```
04 #define RCC_APB2Periph_GPIO_LED    RCC_APB2Periph_GPIOF
05 #define GPIO_LED                    GPIOF
06 #define GPIO_Pin_LED                GPIO_Pin_6 | GPIO_Pin_7 | GPIO_Pin_8 | GPIO_Pin_9
07
08 static const rt_uint16_t led_map[] = {GPIO_Pin_6, GPIO_Pin_7, GPIO_Pin_8, GPIO_Pin_9};
```

因为 STM32 的 GPIO 都是在 APB 总线 2 上面的，不同的配置只需要修改上述的定义即可。

不同型号 STM32 芯片的配置

我们知道，STM32 是一个芯片的系列，根据不同的资源情况又分成数种不同的芯片。这些不同的芯片在片内 flash，片内 SRAM，外设上都有可能不同。STM32F103VC/D/E, STM32F103ZC/D/E 等芯片甚至支持 FSMC 控制器，能够在开发板上扩展更多的 SRAM 内存。

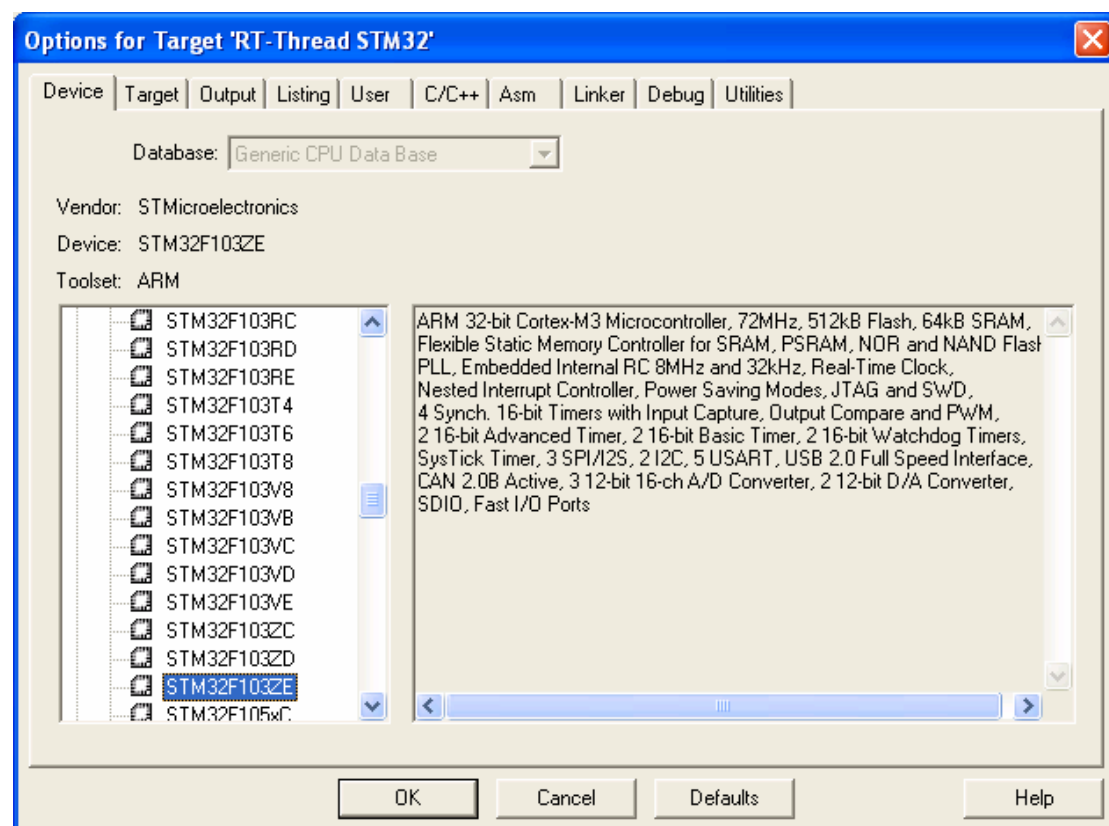
这个 RT-Thread 版本是能够支持不同的 STM32 芯片的，包括 101、103、105、107 等系列，当配置使用了 RT_USING_HEAP 选项时（在 rtconfig.h 中修改，默认使用，更详细的细节调整请参考 **RT-Thread 编程指南**），系统可用内存是受 RT-Thread 动态内存管理模块所管理的。针对 STM32 这些特性，可以在 board.h 中修改：



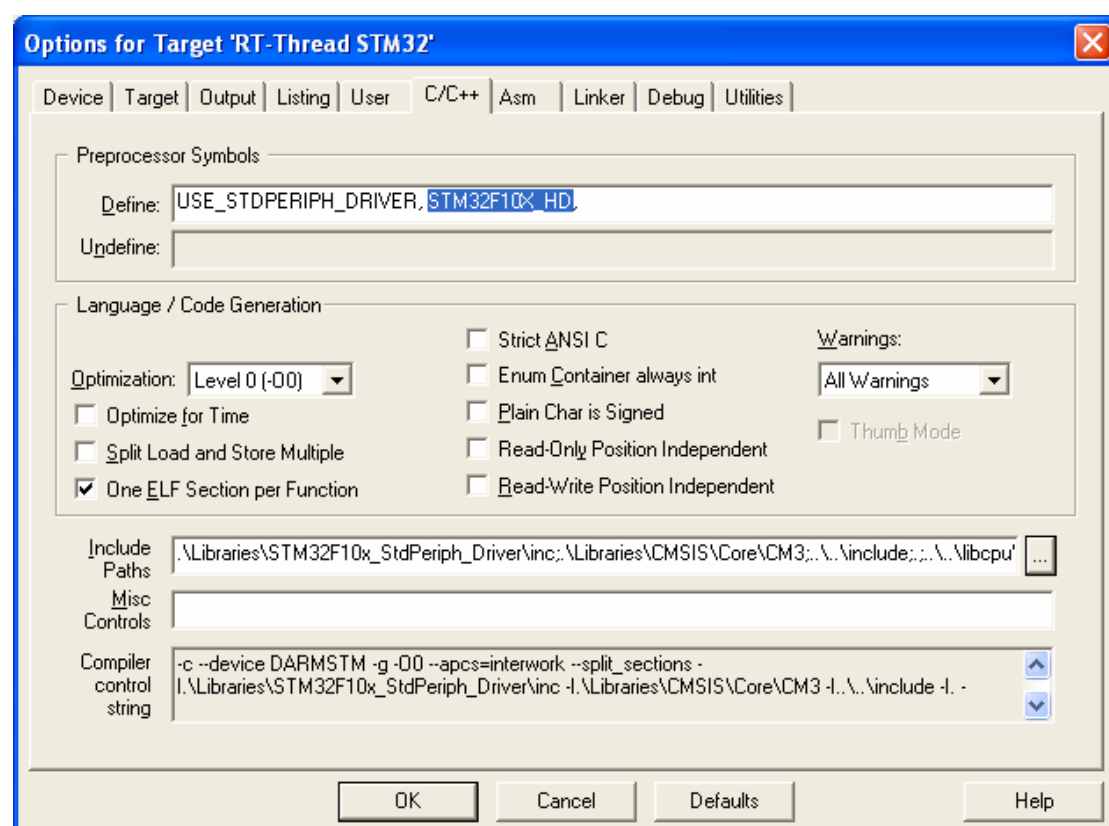
其中主要包括三项：

- **SDCard Driver**: SD 卡驱动，如果不使用文件系统，此项配置是无效的。使用文件系统时，支持 SDIO 接口的 SD 卡驱动，或 SPI 接口的 SD 卡驱动。
- **Use external SRAM memory on the board**: 使用外扩的 SRAM 作为 RT-Thread 的动态内存。使能后，还需要设置外扩 SRAM 的开始地址和结束地址。
- **Internal SRAM memory size[Kbytes]**: 如果不使用外扩 SRAM，那么 RT-Thread 将把片内 SRAM 做为动态内存来使用。针对不同的 STM32 芯片，其片内 SRAM 大小是不相同的，这里就可以配置其大小，单位是 Kbytes。

不同的 STM32 芯片，还需要在工程选项中选择相应的芯片类型，工程选项如下图：



Device 中选择相应的芯片类型



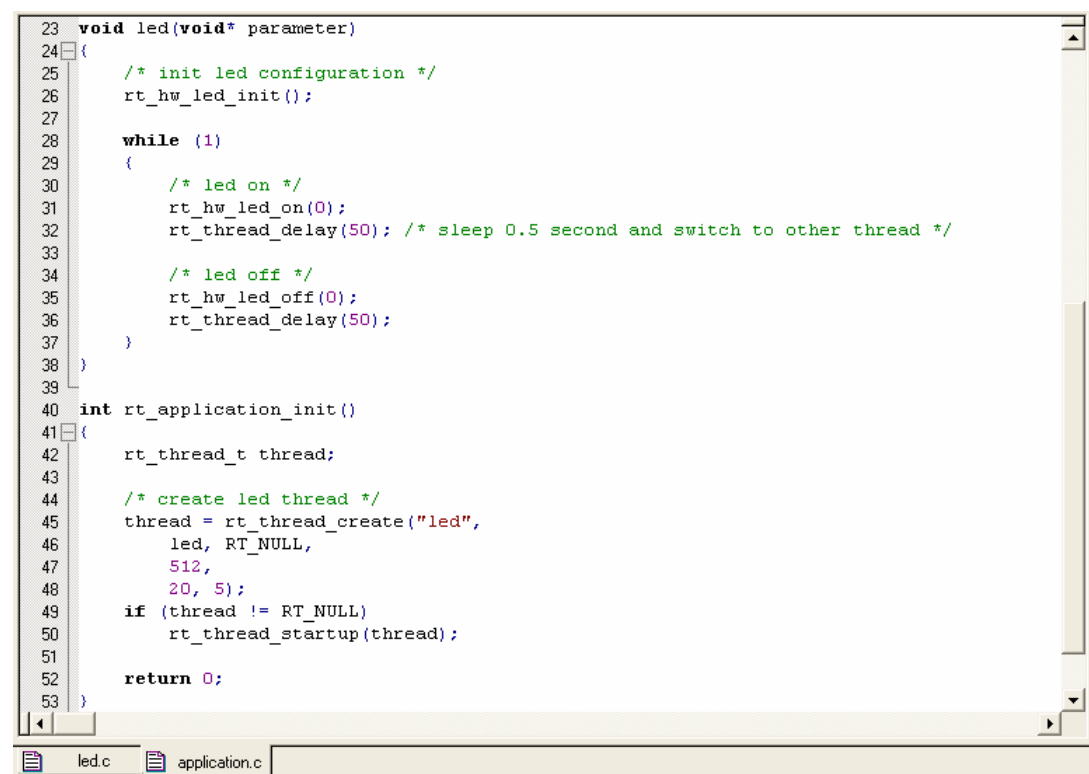
STM32 芯片类别图例

C/C++定义中（Define）中填写不同的 STM32 种类，STM32F10X_LD（低密度），STM32F10X_MD（中密度），STM32F10X_HD（高密度）和 STM32F10X_CL（连接类型）。

跑马灯代码

跑马灯的例子实现了开发板上的 LED 灯闪烁，这里用到了一个线程，让这个线程周期性的更新 LED 等状态：点亮，熄灭。

代码例子如下：



```
23 void led(void* parameter)
24 {
25     /* init led configuration */
26     rt_hw_led_init();
27
28     while (1)
29     {
30         /* led on */
31         rt_hw_led_on(0);
32         rt_thread_delay(50); /* sleep 0.5 second and switch to other thread */
33
34         /* led off */
35         rt_hw_led_off(0);
36         rt_thread_delay(50);
37     }
38 }
39
40 int rt_application_init()
41 {
42     rt_thread_t thread;
43
44     /* create led thread */
45     thread = rt_thread_create("led",
46                               led, RT_NULL,
47                               512,
48                               20, 5);
49     if (thread != RT_NULL)
50         rt_thread_startup(thread);
51
52     return 0;
53 }
```

rt_application_init 函数是用户的初始化函数，可以创建一些用户自己的线程，当系统调度启动后，这些用户线程得到运行。在这个例子中，创建了一个名称为“led”的线程，入口是 **led** 函数，栈大小是 512 字节，优先级是 20，时间片长度是 5（只在相同优先级线程存在是才有效）。

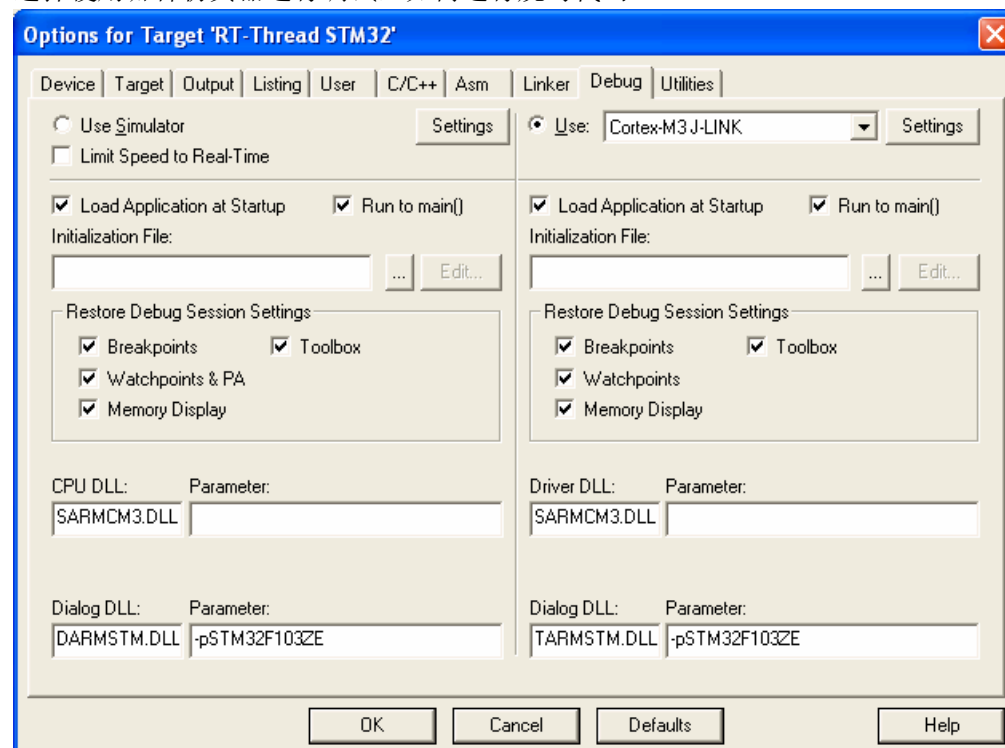
led 函数是“led”线程主函数，代码非常简单，先调用 **rt_hw_led_init** 函数对 led 的 GPIO 进行初始化配置，然后进入一个死循环：点亮灯，延时 0.5 秒，熄灭灯，再延时 0.5 秒。（延时并不意味着 led 线程的忙等待，会自动切换到其他线程）

rt_hw_led_init, **rt_hw_led_on**, **rt_hw_led_off** 等函数都在 **led.c** 中实现。

运行跑马灯例子

为了运行这个跑马灯例子，需要根据自己仿真器情况配置工程的 **Debug**, **Utilities** 选项，

选择使用哪种仿真器进行调试，如何进行烧写代码。



在文档里为了演示这个工程，暂时使用 **Simulator** 运行。运行时把 **Peripherals->General Purpose I/O -> GPIOF** 菜单打开，并选上 **View->Periodic Window Update**，将可以看到 PF6 在置 1、清 0。

