嵌入式实时操作系统原理-FreeRTOS

北京麦克泰软件技术有限公司

2017年4月

课程安排

- 1 学习和掌握一种RTOS
- 2 FreeRTOS的原理
- 3 TraceAlyzer工具介绍
- 4 IAR EWARM集成开发环境
- 5 基于NUCLEO-F401RE的OS实验

BMR
麦克泰技术

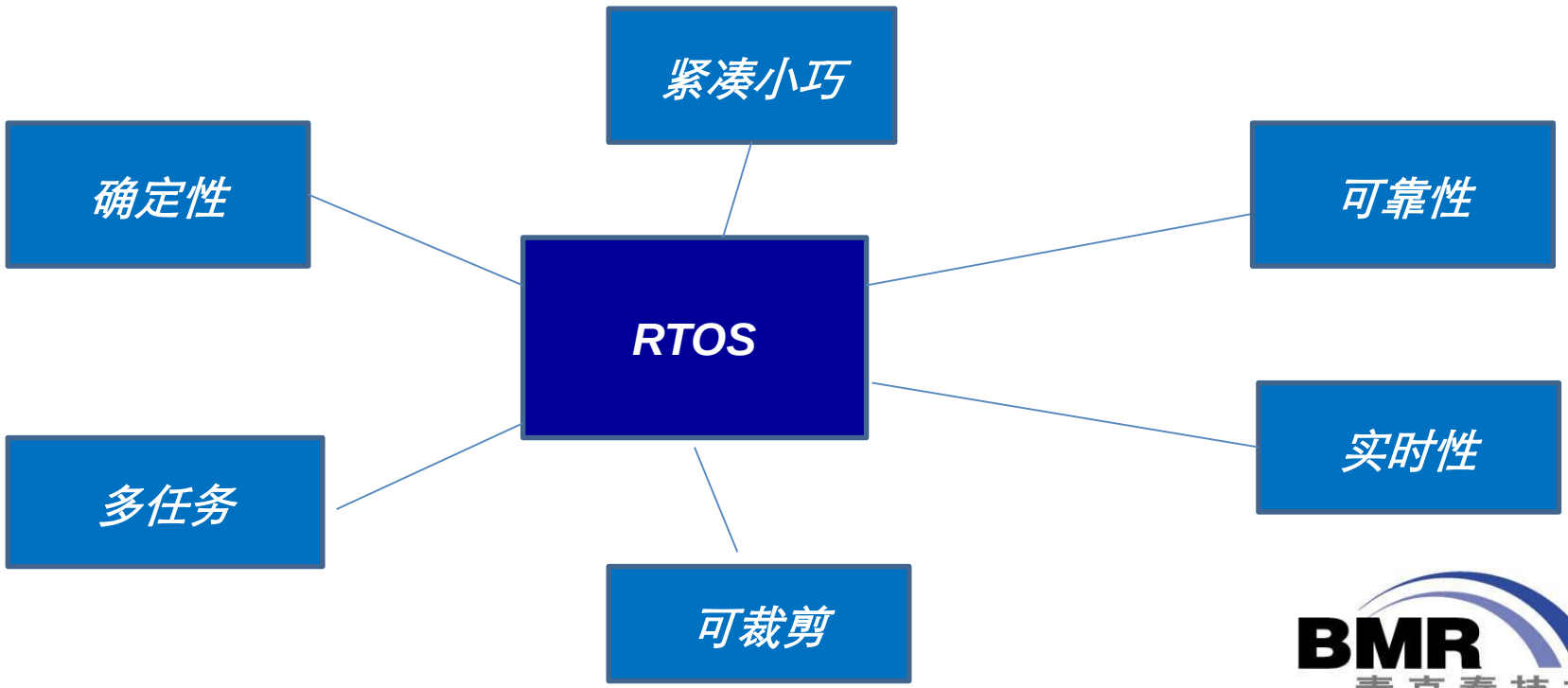
学习和掌握一种RTOS

北京麦克泰软件技术有限公司

2017年4月

什么是RTOS？

- R(real) T(time) OS 实时多任务操作系统
  - RTOS一种操作系统，属于嵌入式操作系统
  - RTOS种类很多；有商业的、DIY和开源的。



什么样OS 是RTOS ？

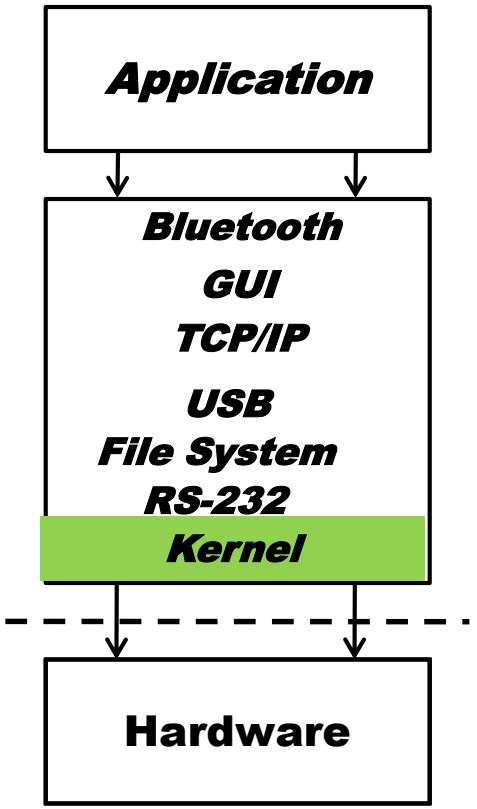
- 那么什么样OS能称为RTOS呢?

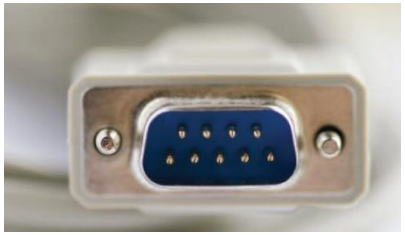
IEEE的实时UNIX分委会认为应具备

- 异步的事件响应
- 确定的切换时间和中断延迟时间优先级中断和调度
- 抢占式调度
- 内存锁定
- 连续文件
- 同步
- ……

RTOS vs. RTOS Kernel

- 操作系统(OS)是一系列软件的集合，提供资源管理和应用代码服务的能力
- RTOS 已经包含了一系列的软件库（中间件，比如vxwork，QNX）
- RTOS kernel 只包含 OS 基本服务（比如FreeRTOS, uc/OS-III）





RTOS 的历史

- RTOS 已经有超过30年的历史
- 比较著名的商业产品有；（按照时间顺序）
  - VRTX　Microtec　（Mentor 公司收购）
  - pSOS　Wind RiverSystem　wrs.com （WRS 公司收购）
  - OS-9　Microware　Microware.com （Metorworks 收购）
  - SMX　Micro Digtal　www.smxrtos.com
  - VxWorks　Wind RiverSystem　wrs.com （Intel 公司收购）
  - LynxOS　lynuxwork　ynuxworks.com
  - QNX　QNX　www.qnx.com （黑莓收购）
  - CMX　CMX system　www.cmx.com
  - Nucleus　ATI www.mentor.com/esd （Mentor收购）
  - THREADX　Expresslogic　www.rtos.com
  - uC/OS －II/III　Micrium　www.micrium.com
  - INTEGRITY　Gree Hill　www.ghs.com
  - 全球超过100多种，中国几种，更有许多用户自己设计RTOS

开源的RTOS

- RTEMS
  - 实时多处理器系统，最早运用在美国防系统
  - 由OAR 公司维护，广泛用在航空航天和军工

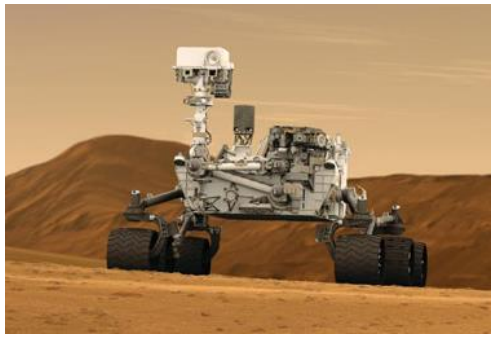
- FreeRTOS
  - 比较清晰的表现其目标和专注点在支持8-16-32位 MCU ，但整体缺乏系统性和配套
- eCOS
  - 基于GNU 的RTOS，含TCP/IP和文件系统，Redhad 曾拥有，eCOcentric维护，消费电子应用
- Contiki
  - 起源于无线传感网络的的RTOS ，有超低功耗管理和IPV6支持。
- Zephyr
  - Linux基金会宣布了一个微内核项目,由Intel 主导，风河提供技术。

为什么要学习RTOS（today）？

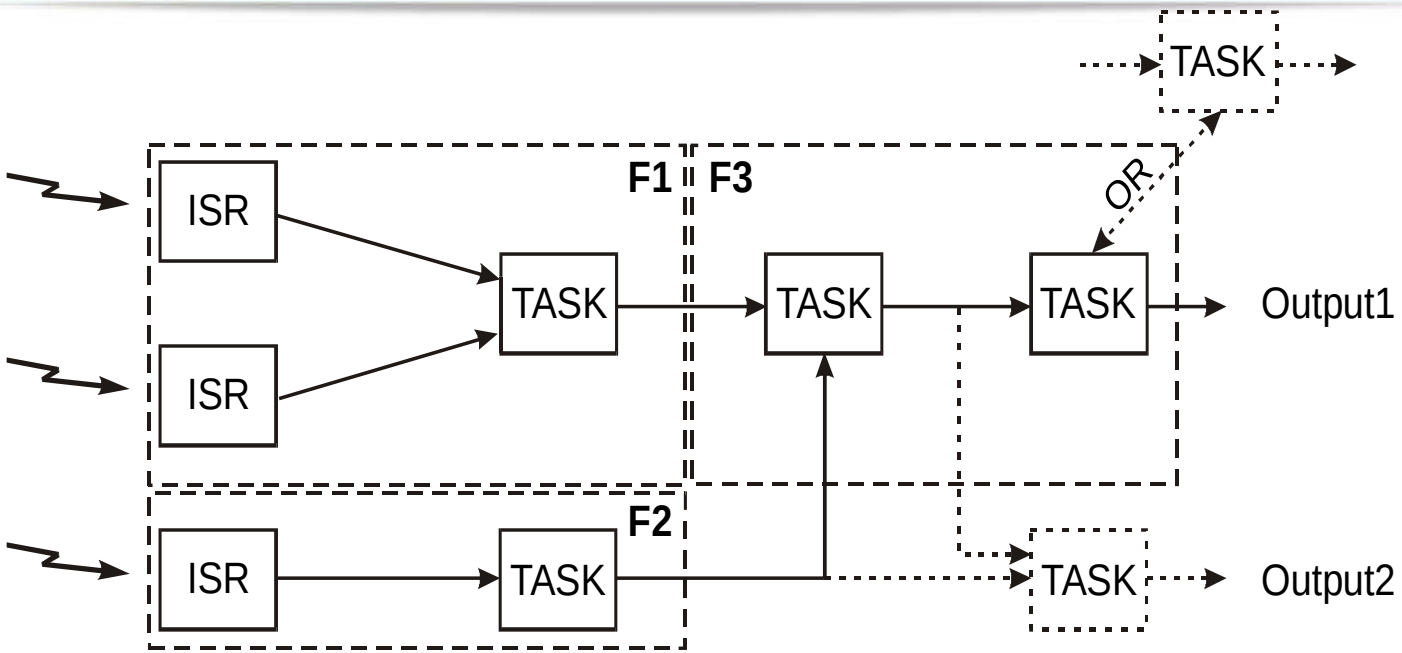
- 物联网的大潮，MCU 迎来一个新的发展机遇：
  - 物联网系统表现为是一个分布式的嵌入式计算系统，大量的MCU替代过去运行Linux 的嵌入式处理器的中心计算模式。
  - 物联网应用需要动态功耗效率，以使系统能够在一定的频率范围（50 到 300+ MHz）内，以最低的功耗水平运行。高功耗效率（DMIPS/mW）意味着应用可以在较低的频率下运行，从而降低有效功耗。
  - 物联网系统是互联系统，需要高度的安全性。互联和安全设计是物联网设备区别传统的嵌入式系统，需要RTOS 的支撑。
  - 软件在物联网中越来越重要，SDN(软件定义网络）OS是软件基础和核心。

RTOS 的应用

- 工业控制装置
- 通信设备
- 消费电子产品
- 仪器仪表
- 军事电子设备
- 航空航天系统
- 计算机外设
- 医疗电子产品
- …….



RTOS 的精髓 - 多任务系统



- 任务独立，基于优先级任务调度
- 任务间通信，异步处理
- F1，F2，F3三个功能模块接口清晰
- 易于加入任务

掌握RTOS内核三大要素

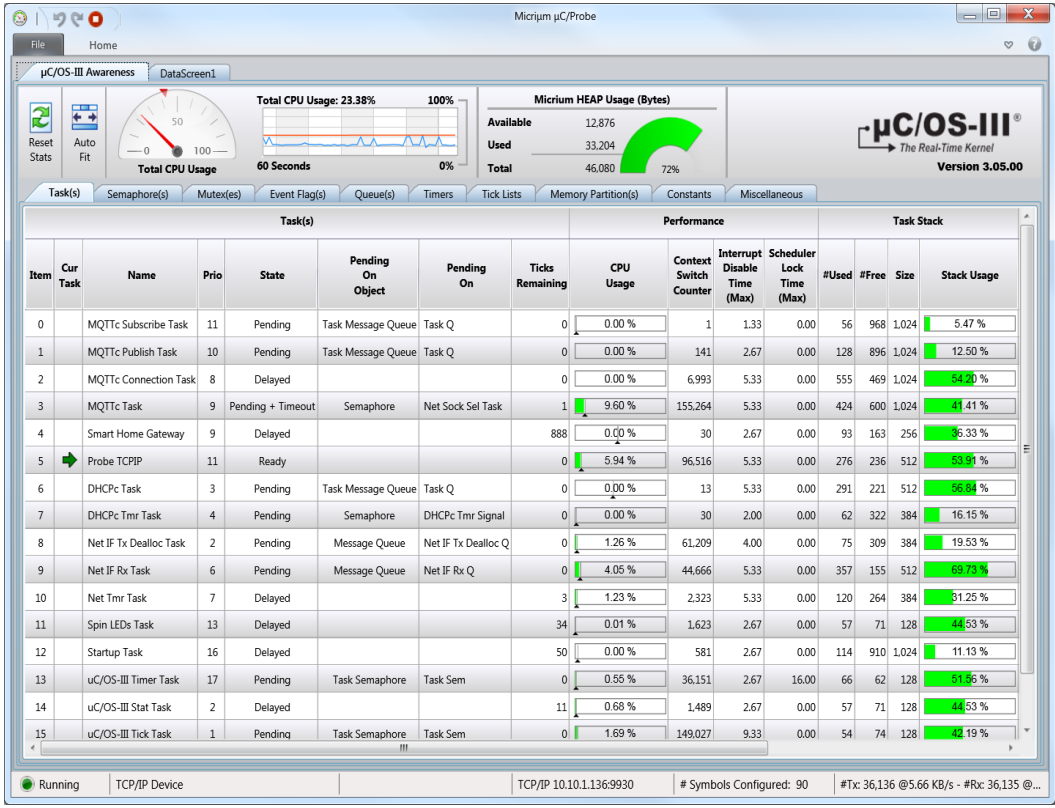
- 事件驱动
  - 中断机制和多任务
  - 优先级抢占和时间片轮转调度
- 资源共享
  - 任务间通信和同步互斥
  - 提供的机制有信号量、邮箱、消息队列、事件标志、互斥
- 性能测量和优化
  - 工具
  - 编译器优化

RTOS 应用的调试

- RTOS 在开发和学习中经常会遇到这样一些问题:
  - 我的任务是在运行吗?
  - 这些任务使用了多少堆栈空间?
  - 每个任务占用了多少 CPU 时间?
  - 我应该如何优化我的代码?
  - 中断关闭多少时间?
  - 有优先级反转的问题?
  - 有死锁的问题?
  - 查看变量、数据结构、队列和 I/O 设备
- 静态调试帮不上忙，因为嵌入式系统都是动态系统
- 有专门的工具可以帮助基于RTOS系统的可视化分析

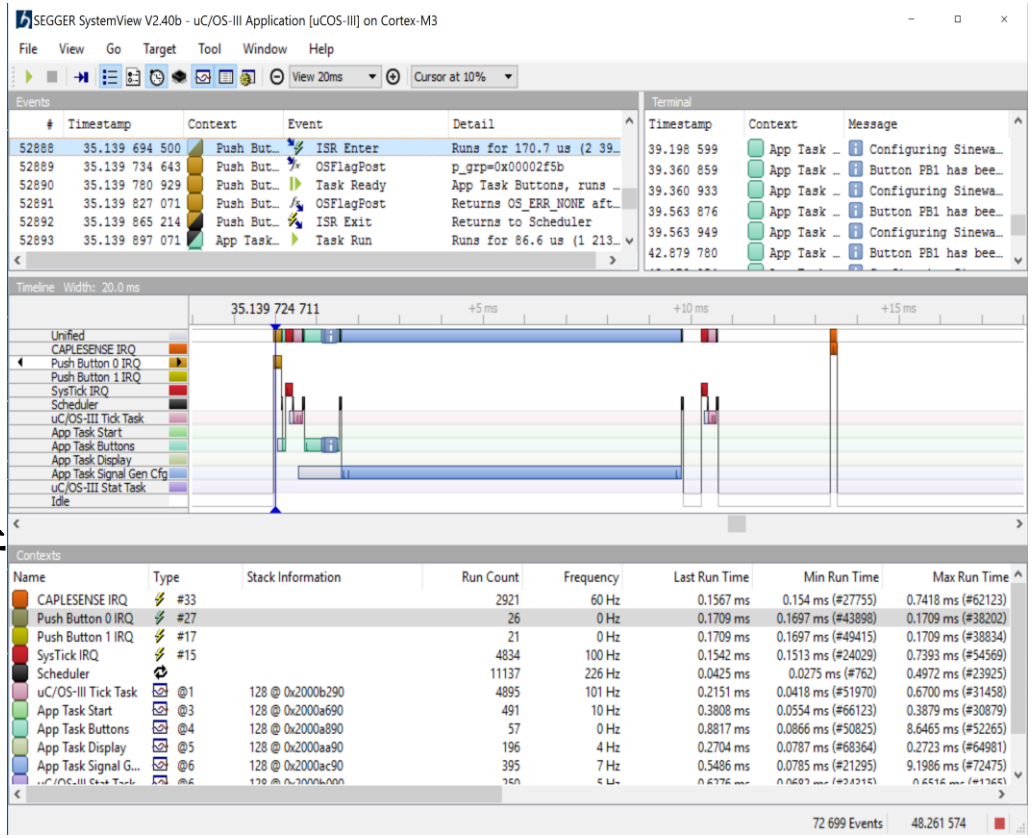
Micriμm 的 μC/Probe

- 基于Windows软件
  - 通过 JTAG 、USB和ETH 连接目标板
    - 不需要 CPU 介入
  - 在运行时显示RTOS 状态
    - 每个任务运行计算
    - 每个任务的 CPU占用时间
    - 每个任务中断关闭时间
    - 堆栈使用情况
    - 任务相应时间
    - 任务状态
    - .....
- 支持RTOS
  - μC/OS-II, μC/OS-III 和 μC/OS 5



SEGGER 的 SystemView

- 在时间轴上显示RTOS 事件
  - 动态实现
  - 中断（ISR)
  - 任务
- 以优先级为序
- 触发
  - 以任务和ISR结束为开始
  - 以用户事件为开始
- 跟踪信息可保存下来用作后分析
- 需要有 SEGGER J-Link
- 支持RTOS
  - **µC/OS-III 和 µC/OS 5**
  - **FreeRTOS**



Percepio 的 Tracealyzer

- 以时间轴记录 RTOS 事件
  - ISRs
  - 任务
- 跟踪信息可保存下来用作后分析
- 超过25种不同的数据展示方式
  - 展示的内容有相关性
- 支持:
  - SEGGER J-Link
  - TCP/IP
  - USB-CDC
- 支持 **RTOS**
  - **µC/OS-III** 和 **µC/OS 5**
  - **FreeRTOS**
  - **Linux**和**vxwork...**

学习RTOS 方法

- 任务管理是重点
  - 掌握任务建立，调度，通信和互斥等机制
  - 掌握内存管理方式（静态作为重点）
  - 学习RTOS 内核和硬件相关部分－中断和时钟管理
- 简单的驱动编写，比如串口
  - 移植过去是重点，现在芯片公司参与多些
- RTOS应用编程接口
  - CMSIS-RTOS-ARM 制定Cortex MCU RTOS接口标准
  - POSIX－UNIX 标准
  - uITRON 日本标准
  - OSEK/VDX－汽车电子和交通标准
- RTOS 编程语言和开发工具
  - C/C++，IAR/KEIL/GCC，未来Python和JS 。

CMSIS COMPLIANT
ARM® Cortex™ Microcontroller Software Interface Standard

BMR 麦克泰技术

RTOS的组件

- OS 组件越来越多、越来越重要
  - 协议—TCP/IP
    - 开源LWIP
    - 商业的USB 协议,蓝牙协议和CANOPEN的价格相对要贵
  - 文件系统，Flash NAND， SD/MMC，USB 盘支持优化
  - 图形模块，uC/GUI （emWin）和TouchFX ,纯软件模块对于MCU 消耗大，软硬结合方案，多点触屏和2D/3D图形是未来趋势。
  - 芯片公司开始提供RTOS 的组件(源代码和二进制)
- 大型的RTOS 基本包括了基本组件
  - 比如 Vxwork,QNX 包含TCP/IP,FILE和GUI
- 小型的RTOS 组件是外加的
  - uC/OS-III 有uC/GUI,uC/FS,uC/TCP-IP等

应用决定需要那些组件，组件也决定了使用和选择哪种RTOS

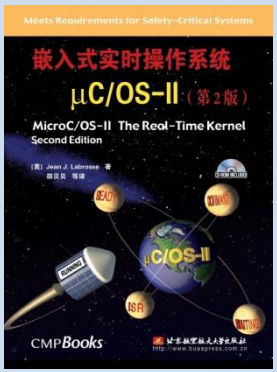
如何选择一个RTOS？

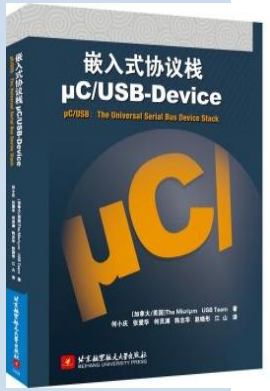
- 你需要 RTOS kernel 还是RTOS?
  - FreeRTOS 自己只有kernel ，其他第三方，uc/OS-II/II 相当完整组件。
- RTOS 还是Linux（Android） ?
  - 你的硬件设计使用的是MCU还是AP
  - 两者都可以使用RTOS ，但是后者可以支持Linux 或者Android
- 嵌入式安全的需求
  - 借助MCU/AP 的 MPU和MMU 可以实现系统内存保护，从而获得安全认证和预认证的安全产品，比如 SaftRTOS
- 芯片和硬件平台的支持
  - 每家芯片公司的平台都会支持1-2 RTOS 或者Linux/Android
- 价格、技术支持和服务升级
  - 开源？一次性版式，还是unit/per CPU/side ，技术服务？

Comparing microcontroller real-time operating systems
Sergey Kolesnik, Telecard-Pribor Ltd. - December 08, 2013

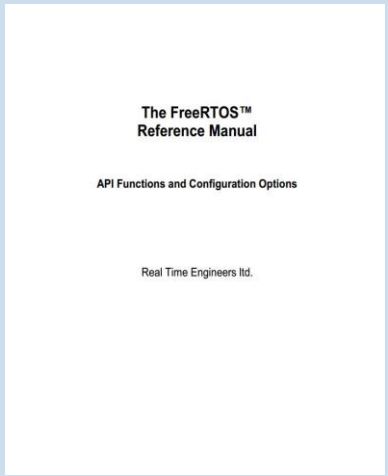
学习资料和工具

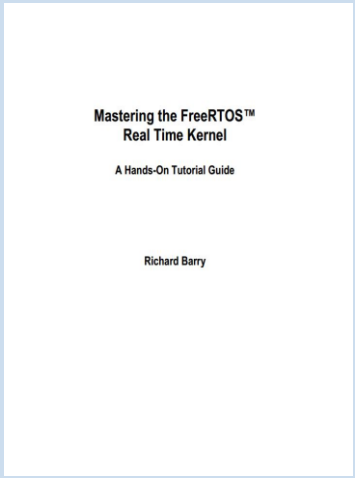
- uC/OS-II 和 III 有非常好中英文文图书（官方）





- FreeRTOS 资料比较少，官方只有英文的PDF手册
- 芯片公司的BSP 比较多
- 视频（ATMEL 工程师）
  - 1 RTOS 介绍
  - http://v.youku.com/v_show/id_XNTgyMTEzOTU2.html
  - 2 RTOS 特性和API
  - http://v.youku.com/v_show/id_XNTgyMTE4NjQw.html
  - 3 FreeRTOS使用
  - http://v.youku.com/v_show/id_XNTgyMTE4MDg4.html
  - 4 深入了解FreeRTOS
  - http://v.youku.com/v_show/id_XNTgyMTIyODgw.html





嵌入式操作系统风云录：历史的演进与物联网未来

- 这书全面回顾了嵌入式操作系统演进历史，主流的嵌入式操作系统技术特点、成长历程以及背后的商业故事，展望了嵌入式操作系统未来的技术路径、市场发展趋势和物联网时代的新机遇。本书以时间轴讲述了从RTOS、开源嵌入式操作系统到物联网操作系统发展历程，以技术为视角剖析了嵌入式操作系统的实时性、安全性和云计算等重要技术，从手机、通信、汽车和可穿戴几个市场讨论了嵌入式操作系统的应用，从嵌入式操作系统知识产权讨论了商业模式的问题。
- 共15章 20万字，2016年底出版。
- www.hexiaoqing.net 有样章，京东、亚马逊、天猫大网站有销售

BMR 麦克泰技术

BMR
麦克泰技术

FreeRTOS原理

北京麦克泰软件技术有限公司

2017年4月

FreeRTOS介绍

- 开源、免费的实时内核（FreeRTOS.org）
- 由Real Time Engineers Ltd开发维护
- 代码目录结构

```
FreeRTOS
    |
    +-Demo
    |
    +-Source
    |
      +-tasks.c
      +-queue.c
      +-list.c
    +-portable
```

作者Richard Barry

FreeRTOS 特征

- 支持35种处理器架构。
- 支持任务和协程（co-routine）
- 支持可抢占式/协作式任务调度
- 内核对象可以动态或静态分配
- 通信和同步机制包含任务通知、队列(信号量/互斥量/递归式互斥量)、事件标志
- 互斥量支持优先级继承
- 软件定时器
- FreeRTOS-MPU
- 紧凑的尺寸（6-12KB）

FreeRTOS资料

- **PDF手册链接**

  http://www.freertos.org/Documentation/RTOS_book.html

FreeRTOS Documentation

PDF files

The unprecedented demand for FreeRTOS is keeping us very busy - so much so that finding time to complete our latest book "Mastering the FreeRTOS Real Time Kernel" is proving challenging! Complimentary pre-release copies have been provided to purchasers of the older books for some time - and now we have extended that offer to everybody. Use the links below to download your copy.

Mastering the FreeRTOS Real Time Kernel - a Hands On Tutorial Guide

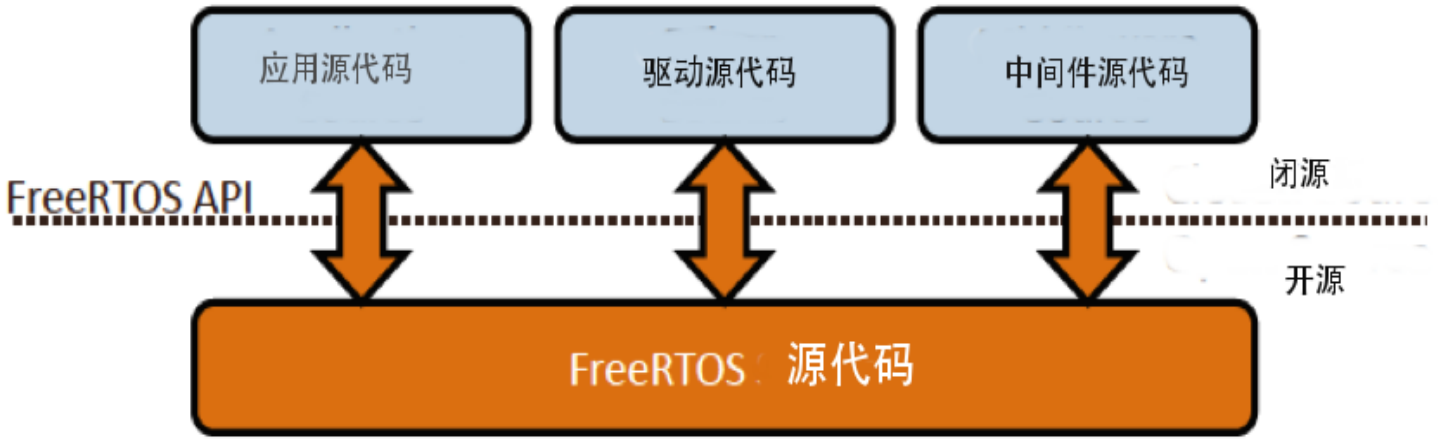
FreeRTOS V9.0.0 Reference Manual


Book companion source code

FreeRTOS授权

- Real Time Engineers Ltd公司书写并拥有FreeRTOS代码
- GPL授权方式

  用户无需公开自己与FreeRTOS 内核不直接相关模块



OpenRTOS

- OpenRTOS是FreeRTOS的商业版本
- 英国WHIS 安全系统公司提供（www.highintegritysystems.com）

内　容	FreeRTOS 开源授权	OpenRTOS 商业授权
免费？	是	不是
能在商业应用中使用 FreeRTOS?	是	是
免版税？	是	是
使用了 FreeRTOS 服务的应用程序必须开源？	不是，只有代码提供的功能与 FreeRTOS 提供的不同	不是
需要提供 FreeRTOS 内核的修改文件？	是	不需要
需要有文档说明使用了 FreeRTOS?	是，只需一个 Web 的链接	不需要
需要提供给用户 FreeRTOS 代码？	是	不需要
能在商业应用中得到专业的技术支持？	不能，可通过在线社区	可以
产品有维护？	没有	有
提供法律保护？	没有	有

SAFERTOS

- **SAFERTOS**
  - **安全认证版的FreeRTOS，它提供了卓越的性能和预认证的可靠性，同时使用用最少的资源。**

    --WHIS 安全系统公司研制

    --支持广泛的国际开发标准

    --基于FreeRTOS的功能模型

    --由TÜV南德意志集团通过预认证IEC 61508-3 SIL 3

BMR 麦克泰技术

FreeRTOS实时内核

任务管理

- 任务实现

```
void  ATaskFunction (void *pvParameters)
{

        Task initialization;
        for (;;)
         {

             /*The code to implement the task functionality.*/

         }
        //vTaskDelete(NULL);

}
```

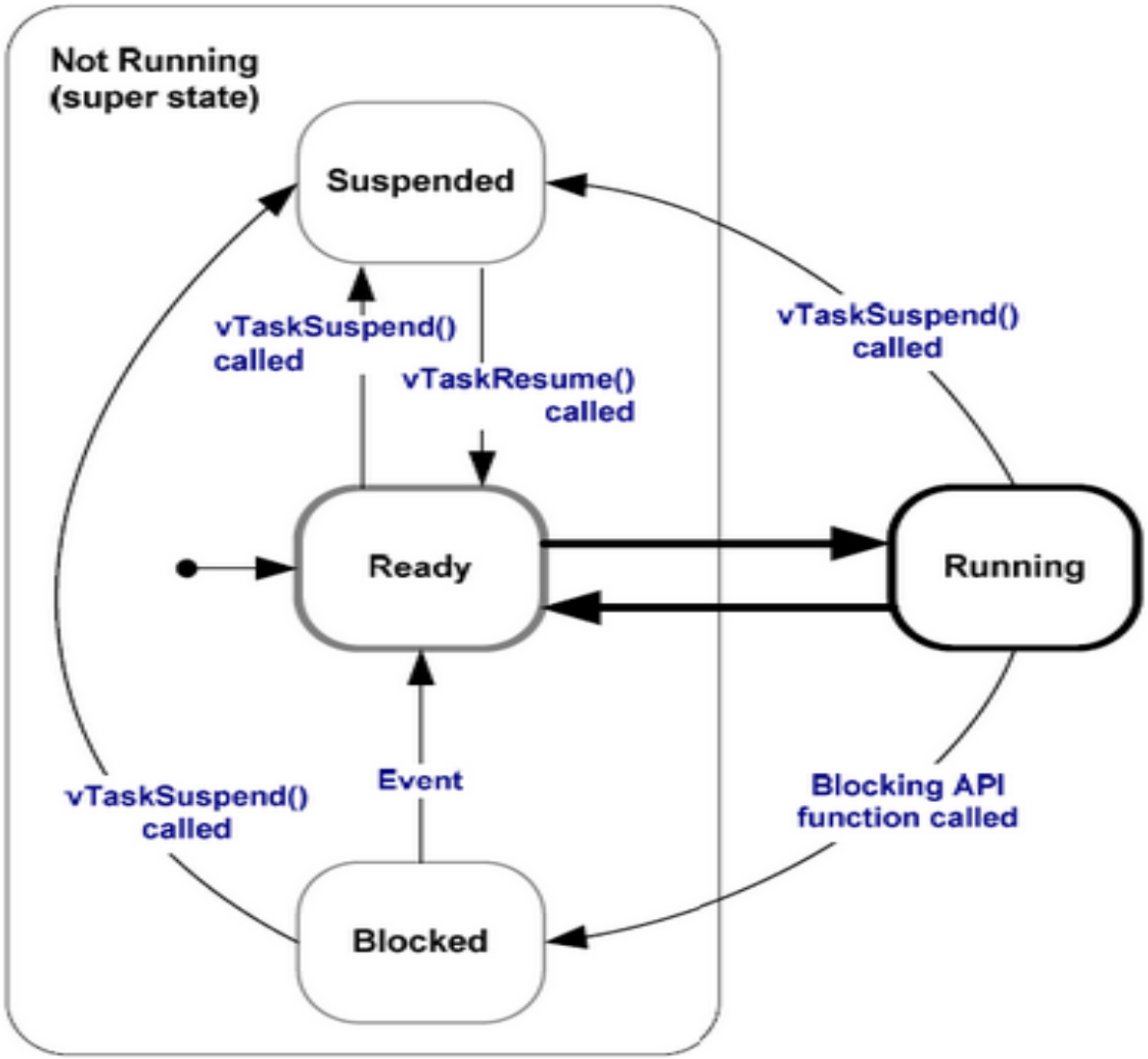
任务管理

- 任务创建

```
BaseType_t xTaskCreate( TaskFunction_t pvTaskCode,
                        const char * const pcName,
                        uint16_t usStackDepth,
                        void *pvParameters,
                        UBaseType_t uxPriority,
                        TaskHandle_t *pxCreatedTask );
```

任务管理

- FreeRTOS任务状态切换



任务管理

- 时间管理
  xTaskDelay()-延时任务
  xTaskDelayUntil()-更精确的延时

任务管理

- vTaskDelete()-删除任务
- vTaskPrioritySet()-改变任务的优先级
- xTaskSuspend()-挂起一个任务
- xTaskResume()-恢复任务
- uxTaskPriorityGet()-查询任务的优先级

队列管理

- FreeRTOS中所有任务间的通信与同步机制都是基于队列实现的
  - 数据存储
  - 允许多个任务访问
  - 阻塞队列访问
  - 通常FIFO

队列管理

- 创建队列

  xQueueCreate()

  - 使用之前，必须创建
  - 从堆空间中分配内存单元

```
xQueueHandle xQueueCreate(
    unsigned portBASE_TYPE uxQueueLength,
     unsigned portBASE_TYPE uxItemSize );
```

队列管理

- 数据发送

```
BaseType_t xQueueSendToFront( QueueHandle_t xQueue,
                              const void * pvItemToQueue,
                              TickType_t xTicksToWait );

BaseType_t xQueueSendToBack( QueueHandle_t xQueue,
                             const void * pvItemToQueue,
                             TickType_t xTicksToWait );
```

队列管理

数据接收

```
BaseType_t xQueueReceive( QueueHandle_t xQueue,
                          void * const pvBuffer,
                          TickType_t xTicksToWait );
```

```
BaseType_t xQueuePeek( QueueHandle_t xQueue,
                       void * const pvBuffer,
                       TickType_t xTicksToWait );
```

队列管理

- 中断服务中使用队列
  - xQueueSendToFrontFromISR()
  - xQueueSendToBackFromISR()
  - xQueueReceiveFromISR()

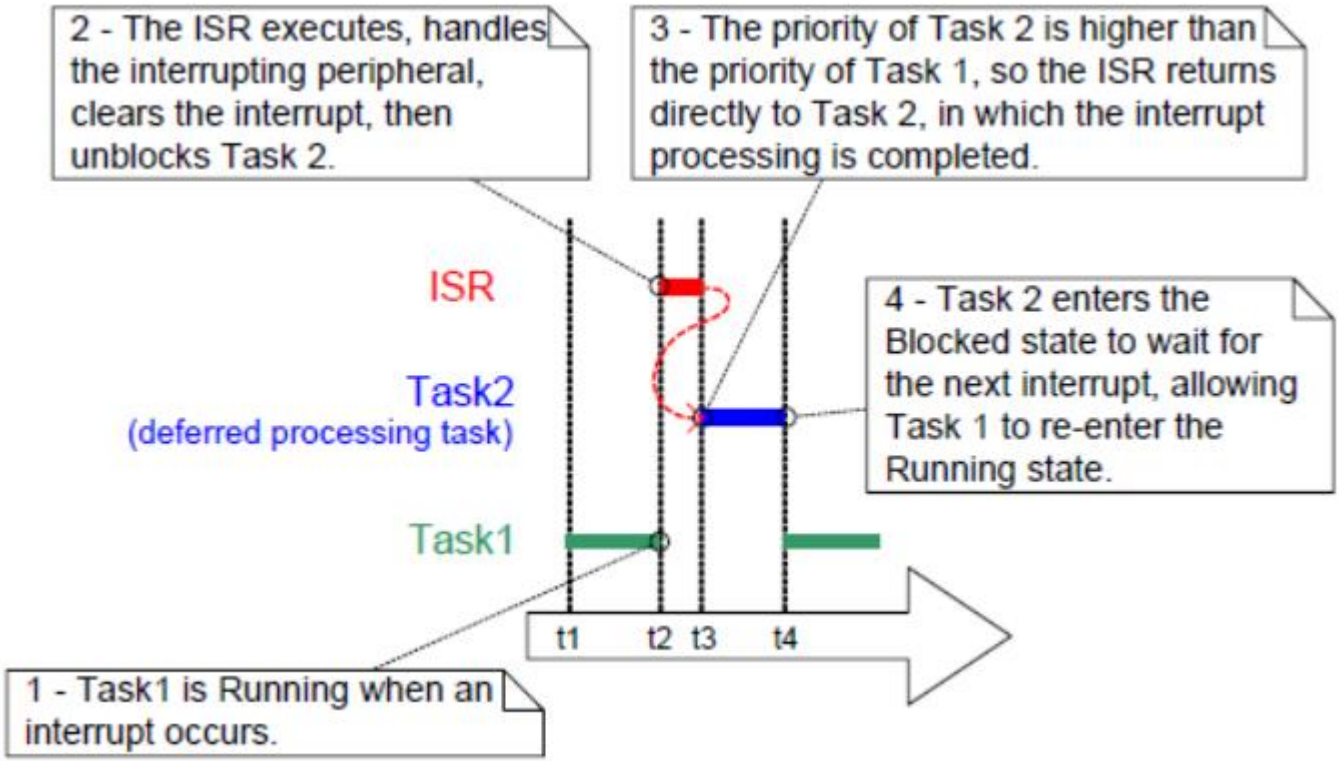
队列管理

- uxQueueMessageWaiting()
  - 查询队列中当前有效数据单元的个数
  - 不能在中断例程中调用

```
unsigned portBASE_TYPE uxQueueMessagesWaiting(
                                    xQueueHandle xQueue );
```

中断管理

- 直接中断处理
- 延迟中断处理



资源管理

- 保护机制
  - 临界段

    关中断

    锁调度器
  - 二值信号量
  - 互斥
  - Gatekeeper任务

任务同步

- 同步机制
  - 信号量
  - 队列
  - 事件组
  - 任务通知

内存管理

- 动态分配
  - malloc()/pvPortMalloc()
  - Free()/vPortFree()
- 实现
  - Heap_1.c
  - Heap_2.c
  - Heap_3.c
  - Heap_4.c
  - Heap_5.c

FreeRTOS启动过程

```
int main( void )
{
        prvSetupHardware();

      /* Start the tasks defined within this file/specific to this demo. */
     xTaskCreate( vLCDTask, "LCD", configMINIMAL_STACK_SIZE, NULL,
tskIDLE_PRIORITY, NULL );

      /* Start the scheduler. */
      vTaskStartScheduler();

     /* Will only get here if there was not enough heap space to create the idle task. */
    return 0;
}
```

FreeRTOS Trace 工具-TraceAlyzer 介绍

北京麦克泰软件技术有限公司

2017年4月

展示RTOS运行时的世界

Percepio Tracealyzer™是一个业界领先的RTOS可视化trace工具，能够帮助我们更好、更快的分析我们的实时操作系统的性能。

可视化包括：

操作系统级别的行为，

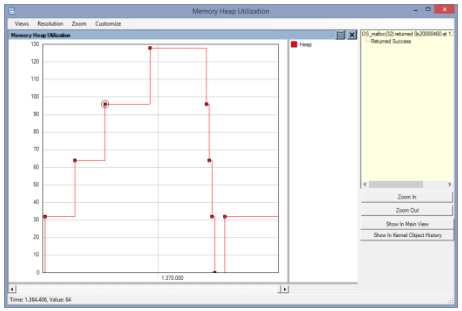
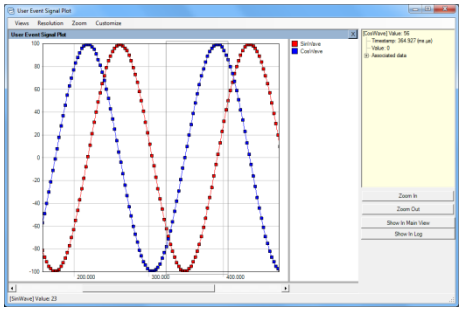
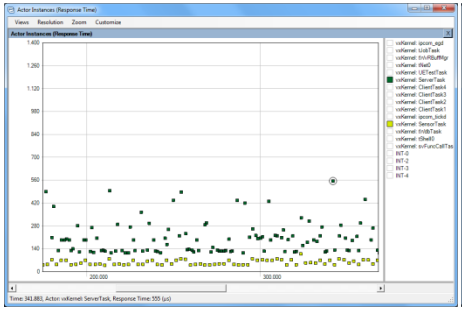
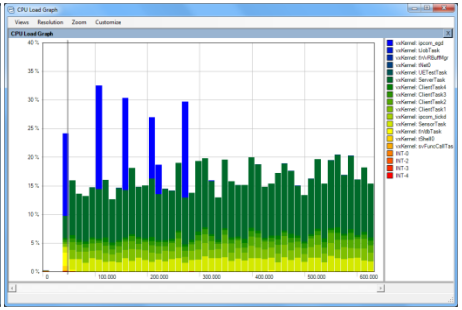
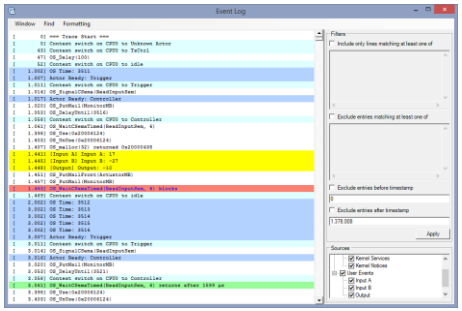
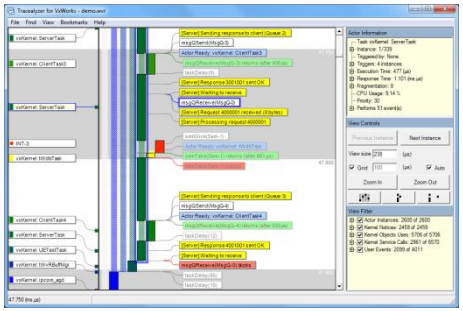
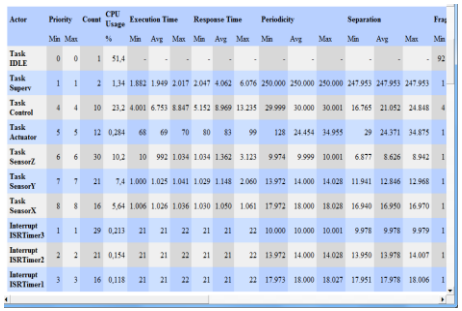
任务调度和系统调用，中断

应用程序级别的行为，

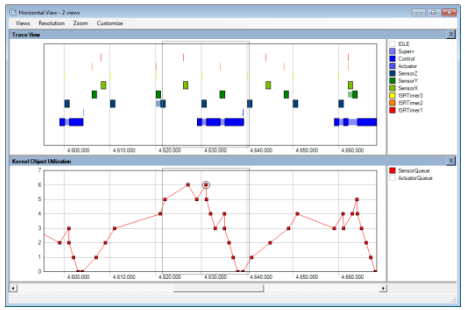
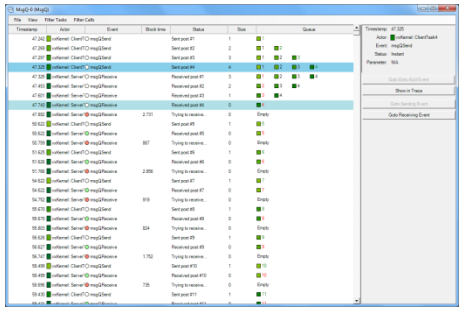
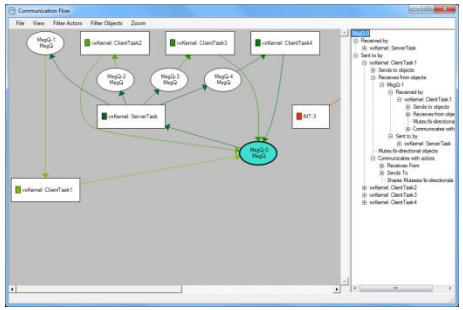
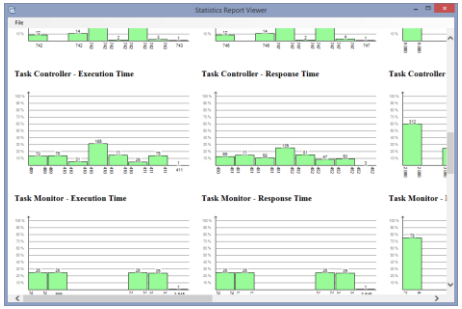
事件、日志记录和数据，绘制

作用：调试会话期间作为的实验室工具，在部署产品使用时，作为崩溃记录仪

TraceAlyzer可视化

Actor	Priority		Count	CPU Usage	Execution Time			Response Time			Periodicity			Separation			Frq
	Min	Max		%	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min
Task IDLE	0	0	1	51,4	-	-	-	-	-	-	-	-	-	-	-	-	92
Task Supervy	1	1	2	1,34	1.882	1.949	2.017	2.047	4.062	6.076	250.000	250.000	250.000	247.953	247.953	247.953	1
Task Control	4	4	10	23,2	4.001	6.753	8.847	5.152	8.969	13.235	29.999	30.000	30.001	16.765	21.052	24.848	4
Task Actuator	5	5	12	0,284	68	69	70	80	83	99	128	24.454	34.955	29	24.371	34.875	1
Task SensorZ	6	6	30	10,2	10	992	1.034	1.034	1.362	3.123	9.974	9.999	10.001	6.877	8.626	8.942	1
Task SensorY	7	7	21	7,4	1.000	1.025	1.041	1.029	1.148	2.060	13.972	14.000	14.028	11.941	12.846	12.968	1
Task SensorX	8	8	16	5,64	1.006	1.026	1.036	1.030	1.050	1.061	17.972	18.000	18.028	16.940	16.950	16.970	1
Interrupt ISRTimer3	1	1	29	0,213	21	21	22	21	21	22	10.000	10.000	10.001	9.978	9.978	9.979	1
Interrupt ISRTimer2	2	2	21	0,154	21	21	22	21	21	22	13.972	14.000	14.028	13.950	13.978	14.007	1
Interrupt ISRTimer1	3	3	16	0,118	21	21	22	21	21	22	17.973	18.000	18.027	17.951	17.978	18.006	1

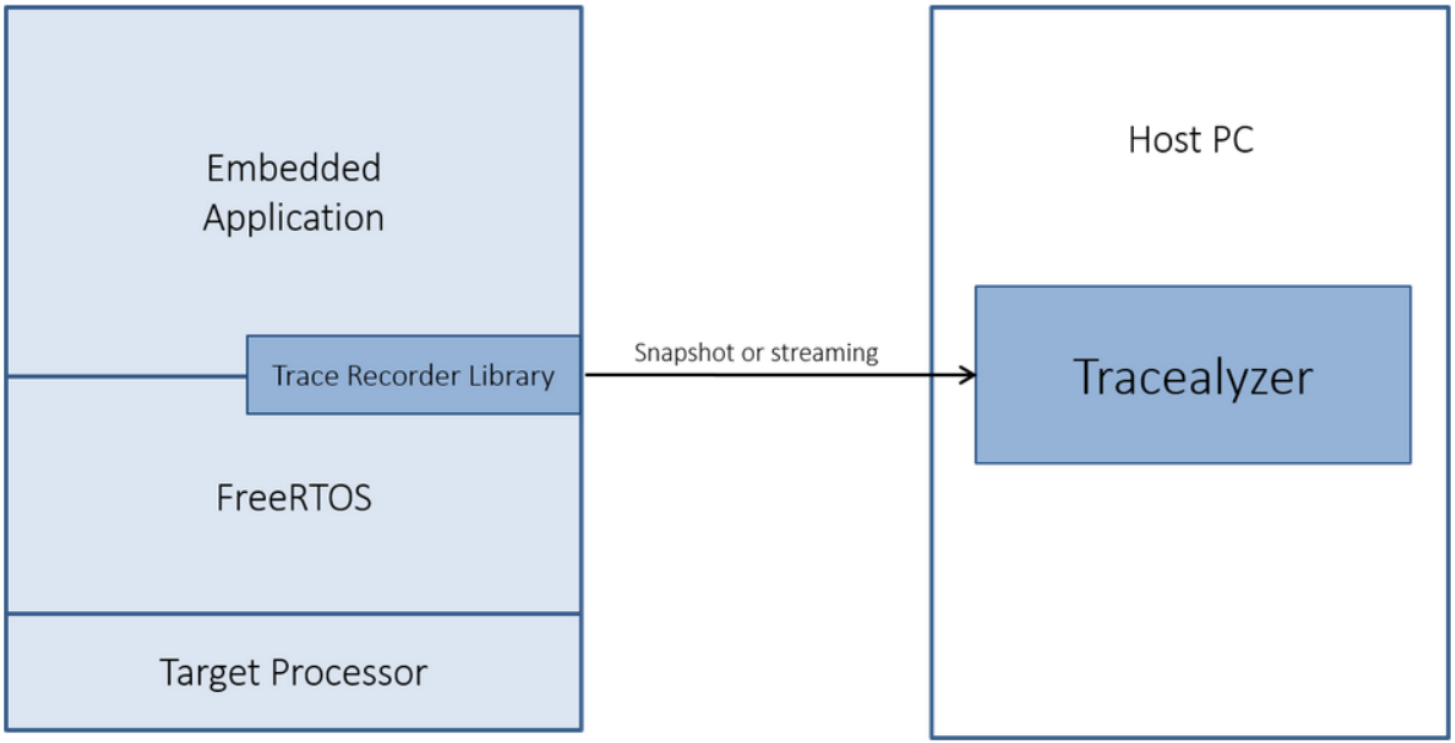





BMR 麦克泰技术

TraceAnlyzer支持的平台

目前支持FreeRTOS, µC/OS, VxWorks 和Linux等。

以FreeRTOS操作系统为例，Tracealyzer 主要包括两部分

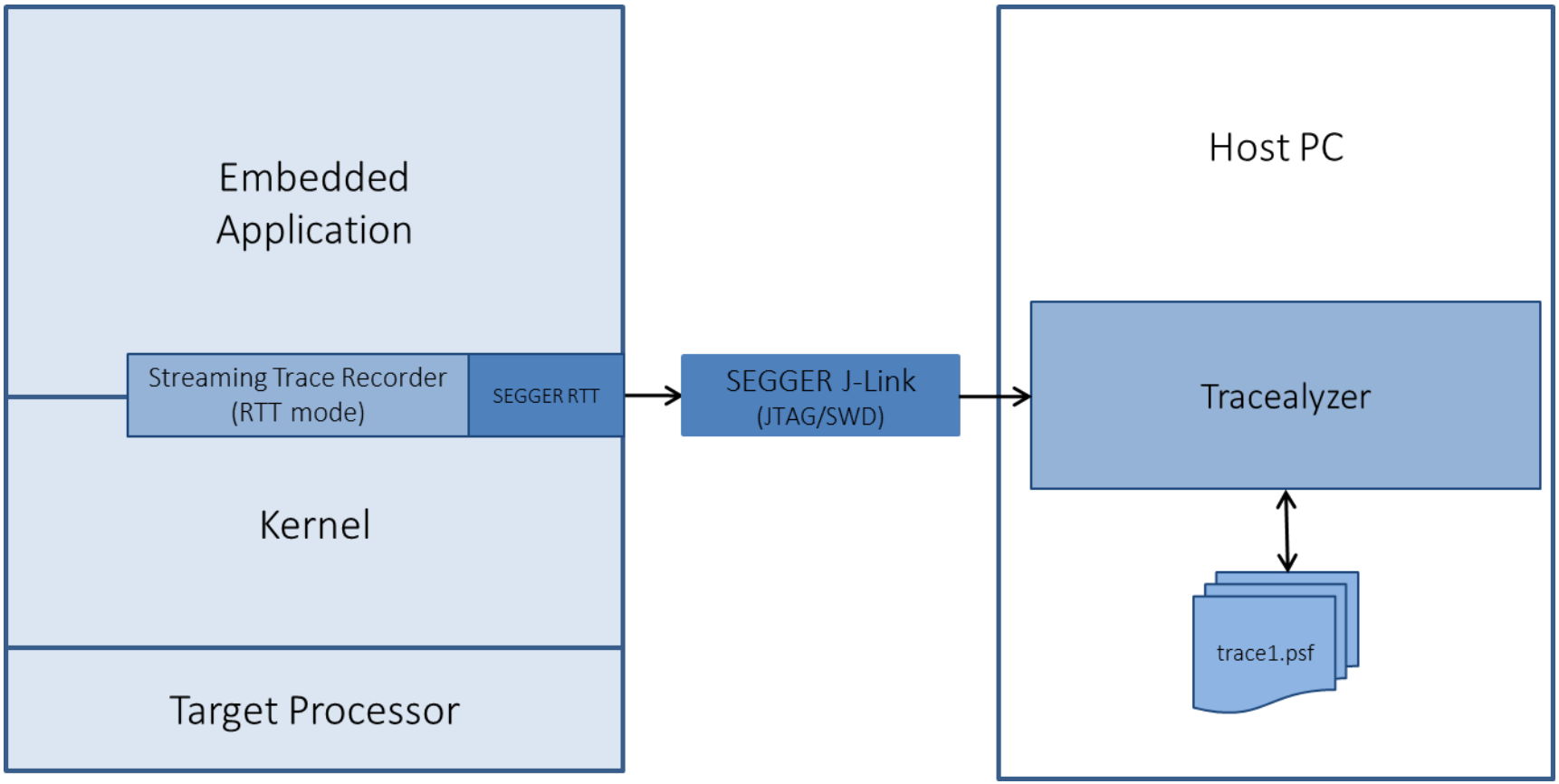


实现Trace的技术手段

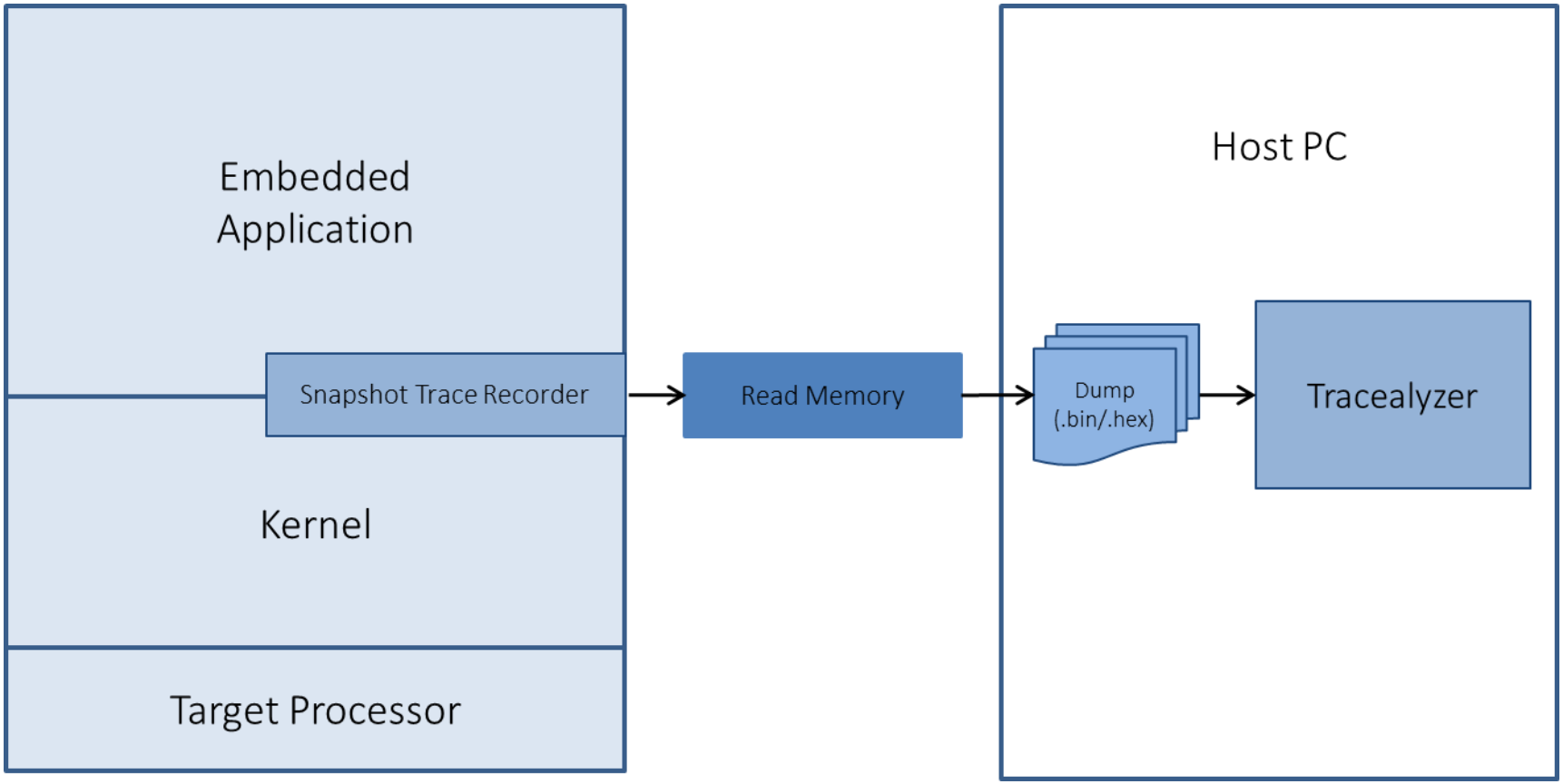
1. 数据流记录--streaming recorder（RTT）

2. 快照记录---snapshot recorder

Segger J-Link Streaming



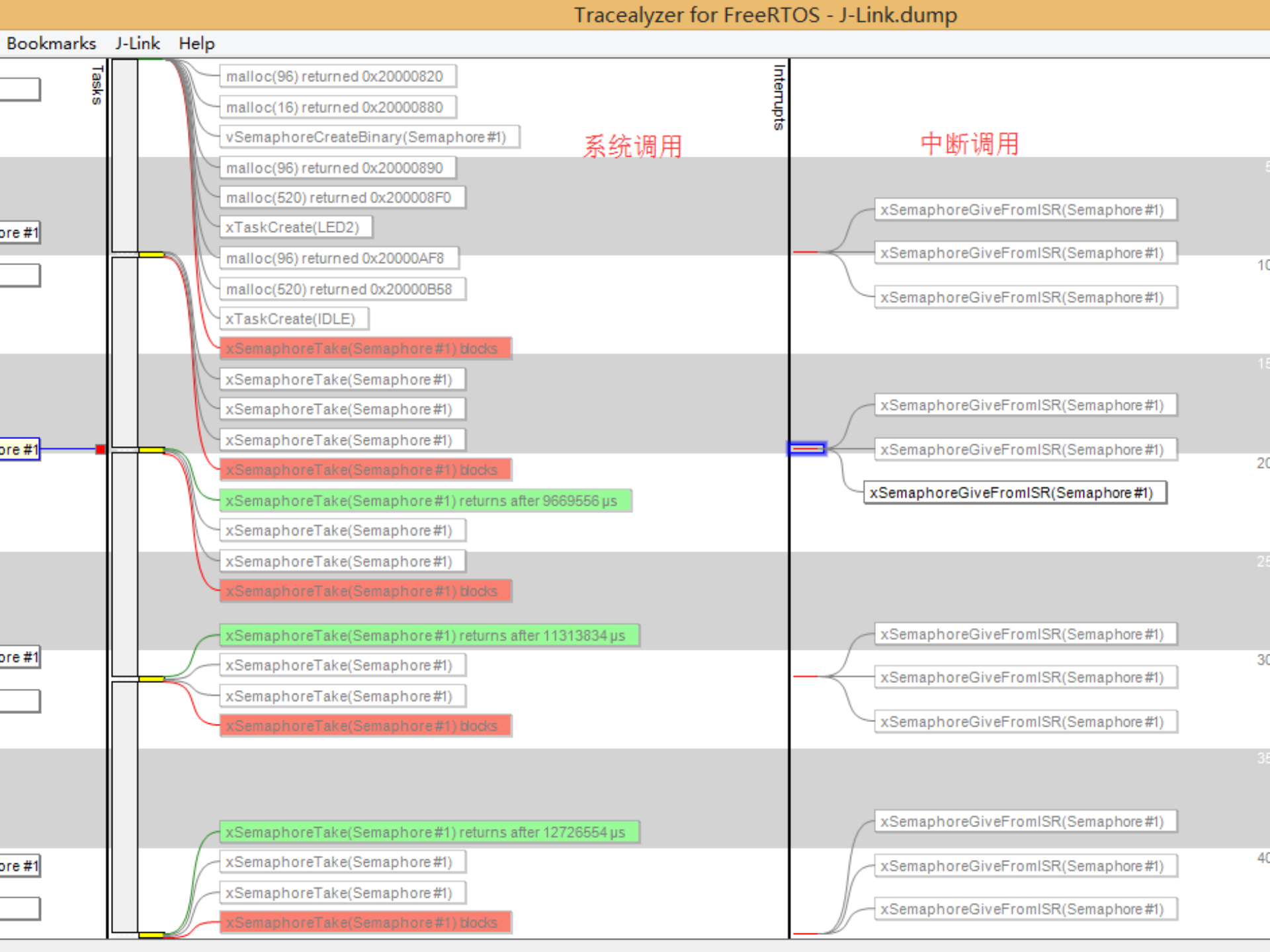
Snapshot方式



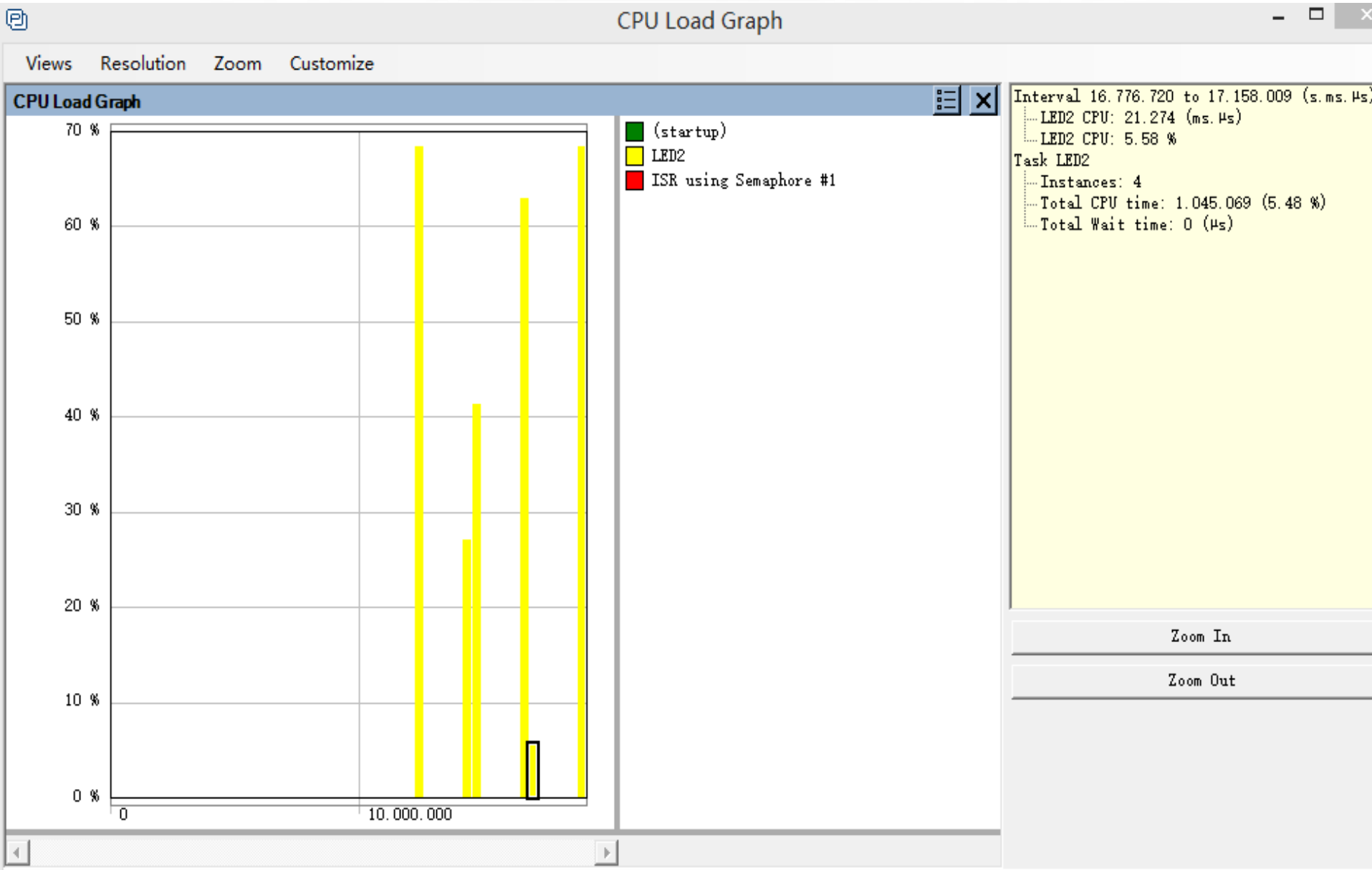
TraceAnalyzer的主要功能：

功能如下：

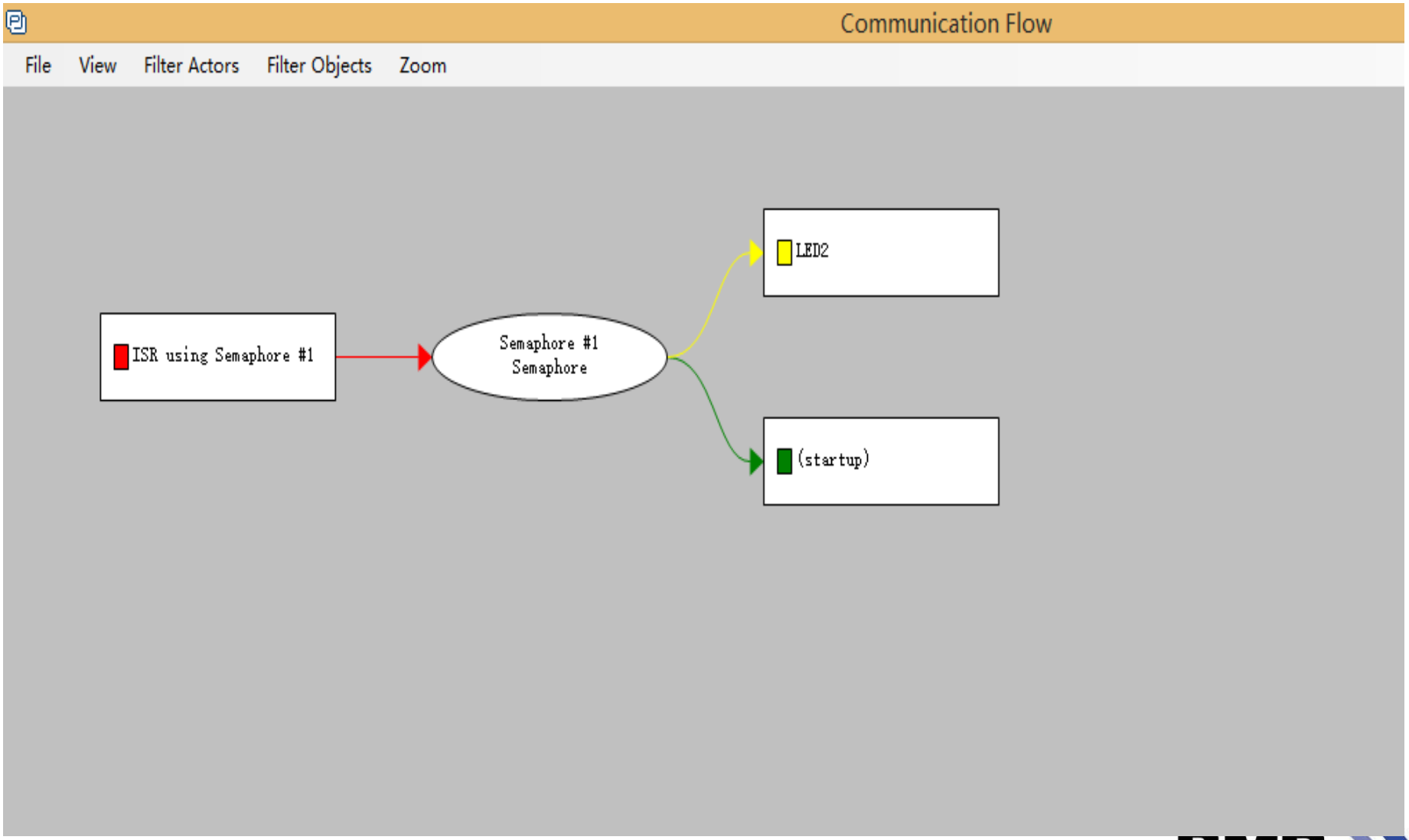
- 分析任务以及中断的执行情况
- 研究CPU的负载，以便进行优化。
- 能够更清楚的了解任务执行流程图
- 可以在应用代码里调用相关的打印API，帮助我们调试


Tracealyzer for FreeRTOS - J-Link.dump
Bookmarks
J-Link
Help
Tasks
Interrupts
系统调用
中断调用
malloc(96) returned 0x20000820
malloc(16) returned 0x20000880
vSemaphoreCreateBinary(Semaphore#1)
malloc(96) returned 0x20000890
malloc(520) returned 0x200008F0
xTaskCreate(LED2)
malloc(96) returned 0x20000AF8
malloc(520) returned 0x20000B58
xTaskCreate(IDLE)
xSemaphoreTake(Semaphore#1) blocks
xSemaphoreTake(Semaphore#1)
xSemaphoreTake(Semaphore#1)
xSemaphoreTake(Semaphore#1)
xSemaphoreTake(Semaphore#1) blocks
xSemaphoreTake(Semaphore#1) returns after 9669556 µs
xSemaphoreTake(Semaphore#1)
xSemaphoreTake(Semaphore#1)
xSemaphoreTake(Semaphore#1) blocks
xSemaphoreTake(Semaphore#1) returns after 11313834 µs
xSemaphoreTake(Semaphore#1)
xSemaphoreTake(Semaphore#1)
xSemaphoreTake(Semaphore#1) blocks
xSemaphoreTake(Semaphore#1) returns after 12726554 µs
xSemaphoreTake(Semaphore#1)
xSemaphoreTake(Semaphore#1)
xSemaphoreTake(Semaphore#1) blocks
xSemaphoreGiveFromISR(Semaphore#1)
xSemaphoreGiveFromISR(Semaphore#1)
xSemaphoreGiveFromISR(Semaphore#1)
xSemaphoreGiveFromISR(Semaphore#1)
xSemaphoreGiveFromISR(Semaphore#1)
xSemaphoreGiveFromISR(Semaphore#1)
xSemaphoreGiveFromISR(Semaphore#1)
xSemaphoreGiveFromISR(Semaphore#1)
xSemaphoreGiveFromISR(Semaphore#1)
xSemaphoreGiveFromISR(Semaphore#1)
xSemaphoreGiveFromISR(Semaphore#1)
xSemaphoreGiveFromISR(Semaphore#1)
5
10
15
20
25
30
35
40

CPU 负载情况


CPU Load Graph
Views
Resolution
Zoom
Customize
CPU Load Graph
70 %
60 %
50 %
40 %
30 %
20 %
10 %
0 %
0
10.000.000
(startup)
LED2
ISR using Semaphore #1
Interval 16.776.720 to 17.158.009 (s.ms.µs)
LED2 CPU: 21.274 (ms.µs)
LED2 CPU: 5.58 %
Task LED2
Instances: 4
Total CPU time: 1.045.069 (5.48 %)
Total Wait time: 0 (µs)
Zoom In
Zoom Out

通信流程



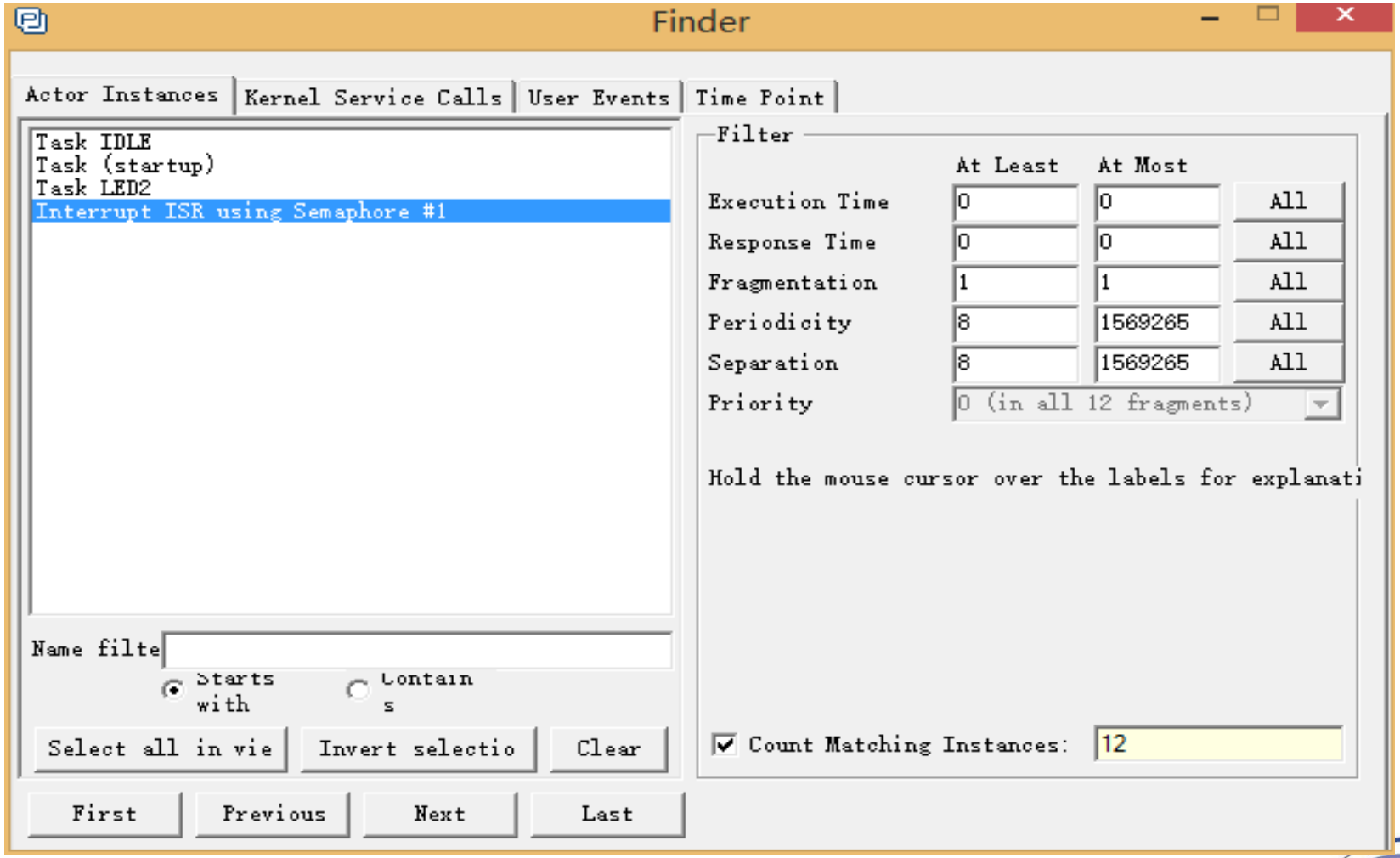
系统事件以及用户事件

Event Log

Window Find Formatting

```
[          7] === Trace Start ===
[          7] Context switch on CPU 0 to (startup)
[         15] malloc(96) returned 0x20000570
[         24] malloc(16) returned 0x200005D0
[         35] vSemaphoreCreateBinary(Semaphore #1)
[         43] malloc(96) returned 0x200005E0
[         51] malloc(520) returned 0x20000640
[         74] xTaskCreate(LED2)
[         84] malloc(96) returned 0x20000848
[         93] malloc(520) returned 0x200008A8
[        114] xTaskCreate(IDLE)
[        134] [Default Channel] Task1 starting...
[        146] xSemaphoreTake(Semaphore #1) blocks
[        157] Context switch on CPU 0 to IDLE
[38.169.704] Context switch on CPU 0 to ISR using Semaphore #1
[38.169.704] xSemaphoreGiveFromISR(Semaphore #1)
[38.169.704] Context switch on CPU 0 to IDLE
[38.169.712] Actor Ready: LED2
[38.169.719] Context switch on CPU 0 to ISR using Semaphore #1
[38.169.719] xSemaphoreGiveFromISR(Semaphore #1)
[38.169.719] Context switch on CPU 0 to IDLE
[38.169.728] Context switch on CPU 0 to ISR using Semaphore #1
[38.169.728] xSemaphoreGiveFromISR(Semaphore #1)
[38.169.728] Context switch on CPU 0 to IDLE
[38.169.736] Context switch on CPU 0 to LED2
[38.169.745] xSemaphoreTake(Semaphore #1)
[38.169.763] [message] ISR_TASK synchronization
[38.169.779] [Default Channel] Task1 starting...
[38.169.786] xSemaphoreTake(Semaphore #1)
[38.169.806] [message] ISR_TASK synchronization
[38.169.822] [Default Channel] Task1 starting...
[38.169.829] xSemaphoreTake(Semaphore #1)
[38.169.849] [message] ISR_TASK synchronization
[38.169.865] [Default Channel] Task1 starting...
[38.169.876] xSemaphoreTake(Semaphore #1) blocks
[38.169.888] Context switch on CPU 0 to IDLE
[40.218.144] Context switch on CPU 0 to ISR using Semaphore #1
```

Finder



TraceAlyzer API

- **void vTraceEnable(int startOption)**
- **void vTraceStop(void)**

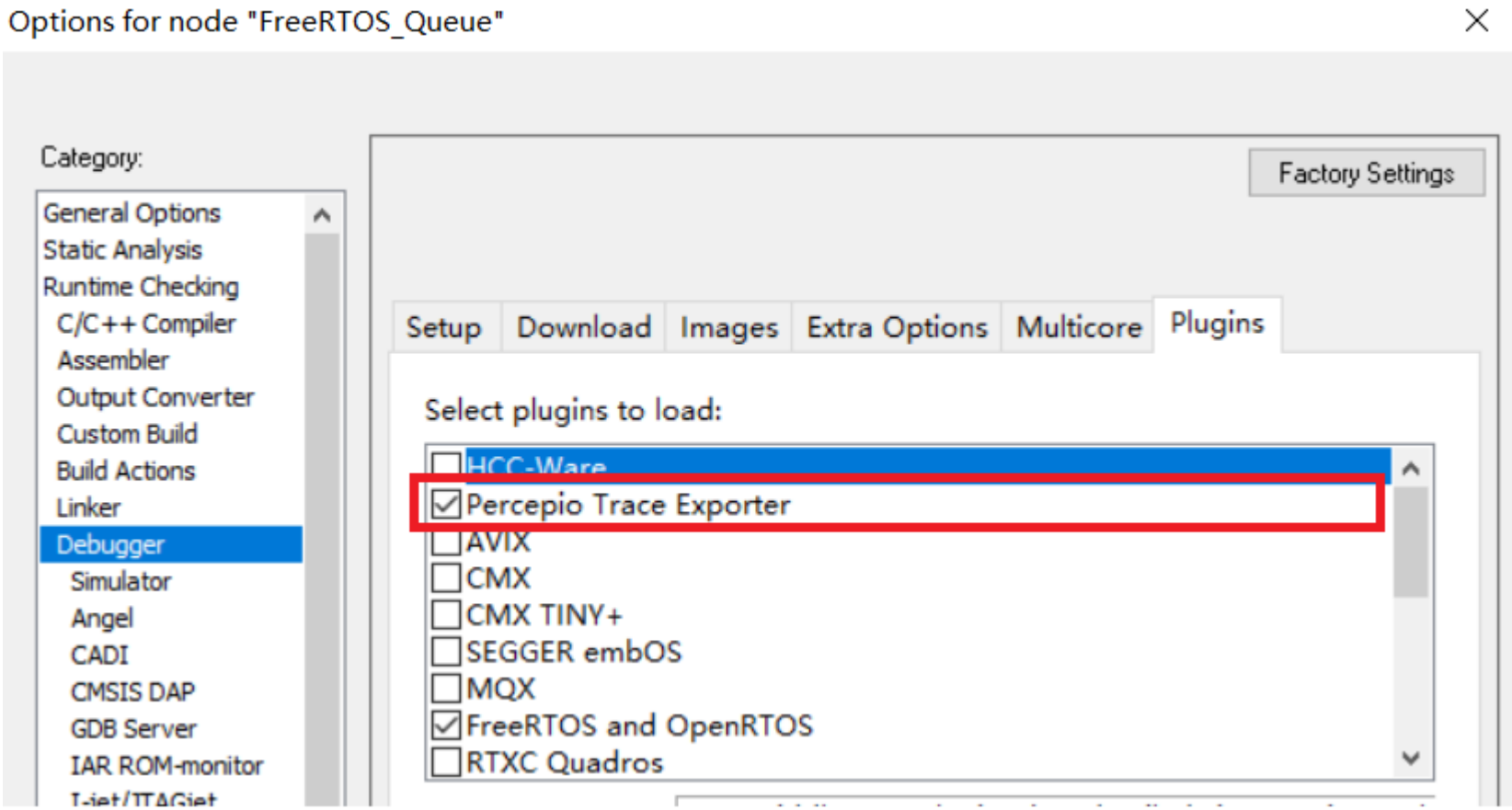
关于traceanalyzer的使用

流程：

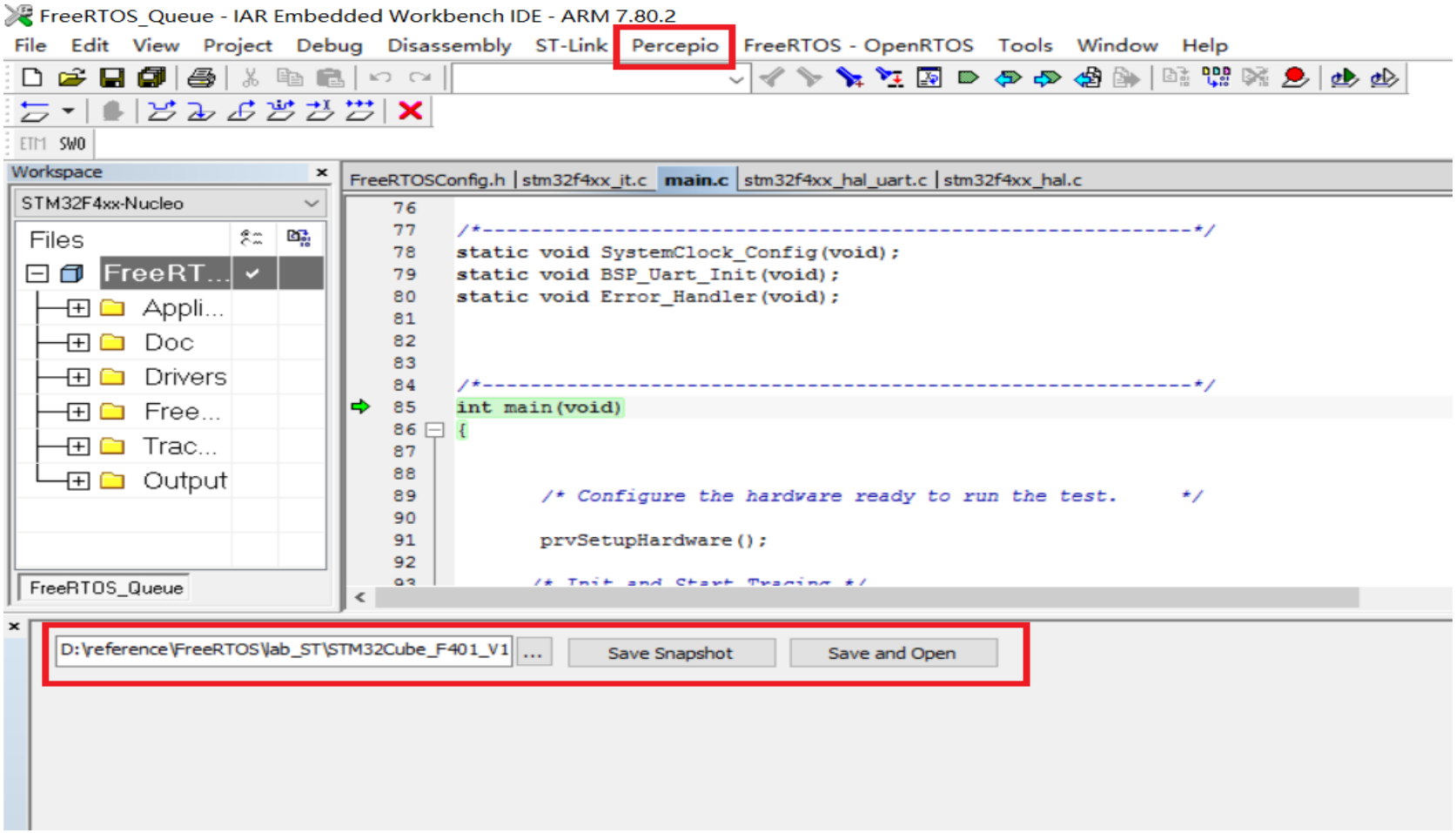
- 将Trace Recorder Library 添加到工程中
- 配置trconfig.h
- 配置FreeRTOSConfig.h
- 在main函数里初始化并启用

  void vTraceEnable(int startOption)

Traceanalyzer与IAR 配合



Traceanalyzer与IAR 配合



BMR 麦克泰技术

FreeRTOS 开发工具-IAR EWARM 使用

北京麦克泰软件技术有限公司

2017年4月

IAR Embedded Workbench for ARM

- **EWARM: 完全集成的ARM嵌入式软件开发环境**

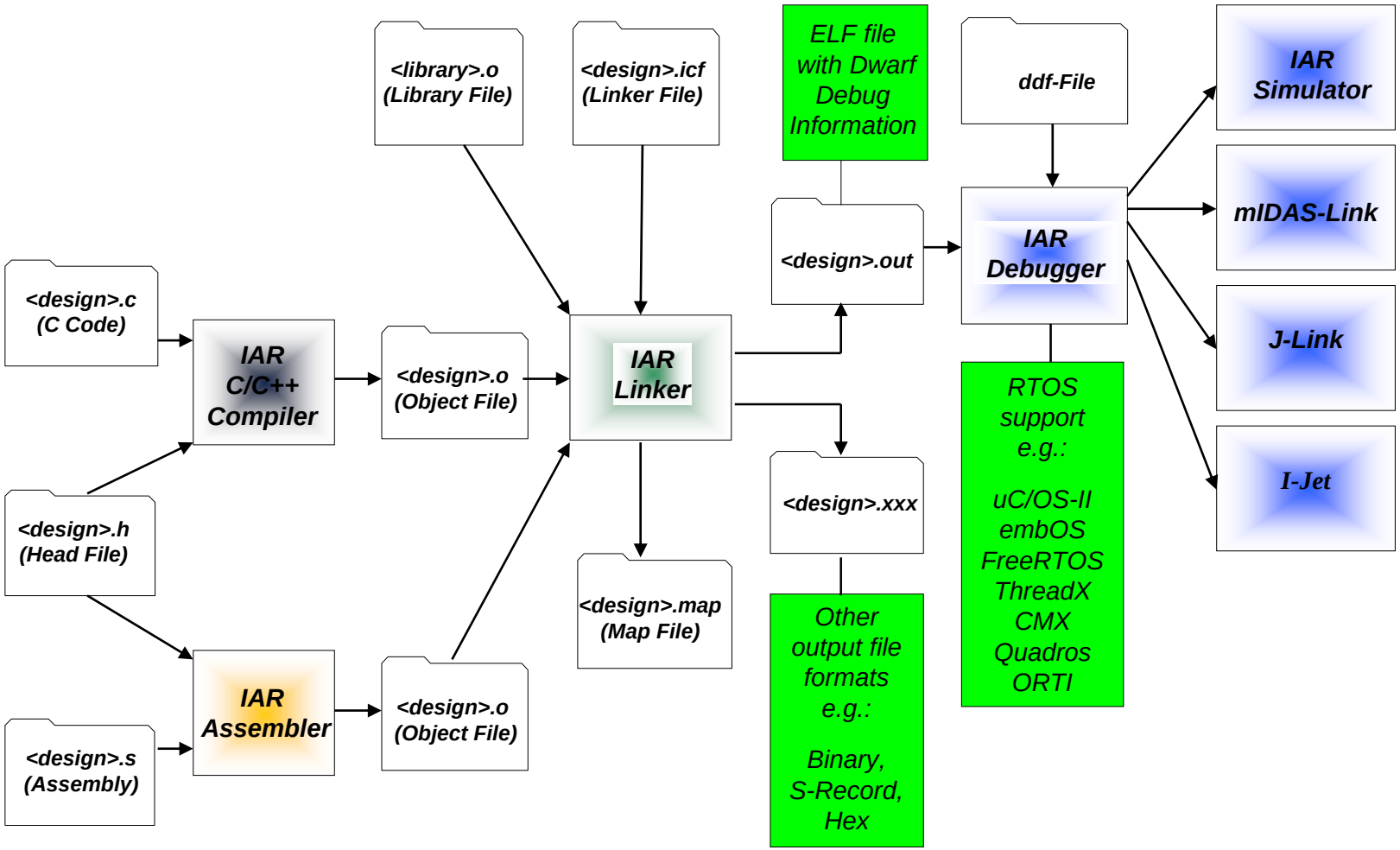
- **支持所有ARM内核和绝大多数基于ARM内核的处理器**
  ARM7, ARM9, ARM10, ARM11, SecurCore
  Cortex-M0/M1/M3/M4, Cortex-R4, Cortex-A5/A8/A9/A15

- **支持ARM CMSIS软件接口标准**
  包含CMSIS DSP Software Library，便于在Cortex-M3/M4处理器上进行数字信号处理软件开发

- **C/C++编译器**
  业界领先的代码优化性能：Around 35% Smaller than GCC
  支持C语言的C89/C99两代国际标准
  内置MISRA-C:1998和MISRA-C:2004编程规则检查器

- **调试环境C-SPY**
  RTOS内核识别插件: uC/OS-II& uC/OS-III, embOS, ThreadX, FreeRTOS, …
  ARM CoreSight: SWO Trace & ETM Trace

EWARM: 工具架构



EWARM: 评估与试用

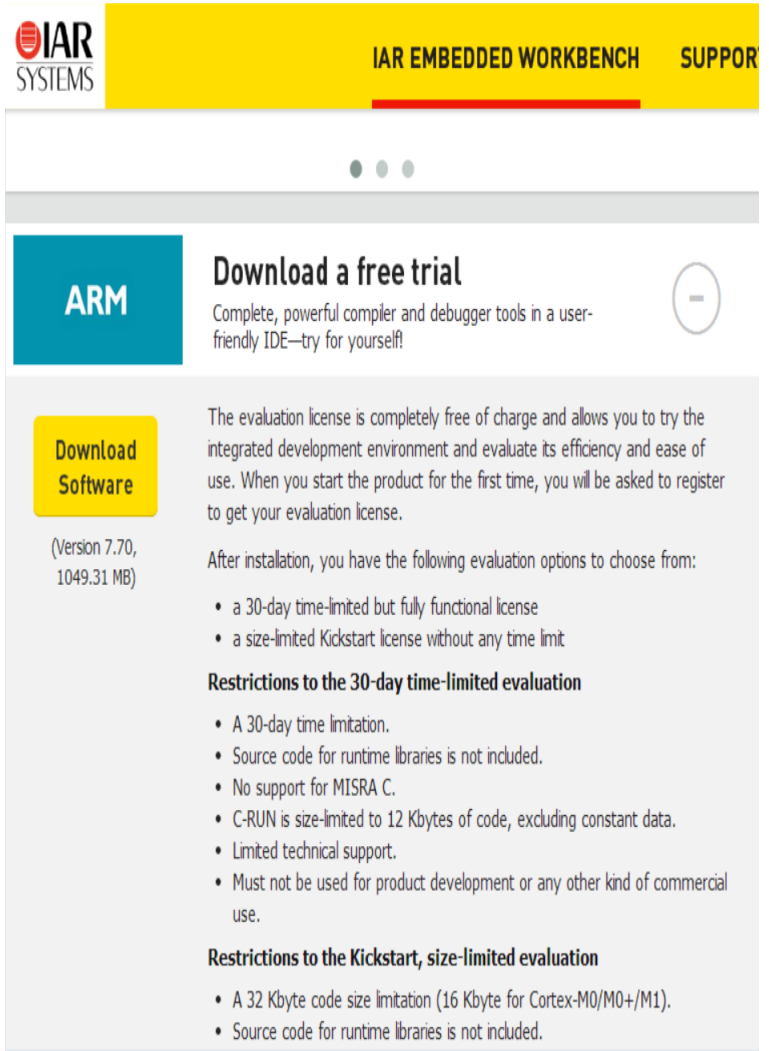
www.iar.com/iar-embedded-workbench/#!?architecture=ARM

下载并安装EWARM

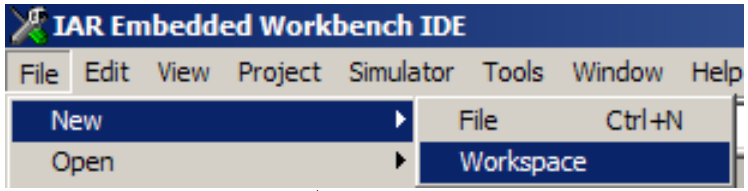
填写注册信息，并发送

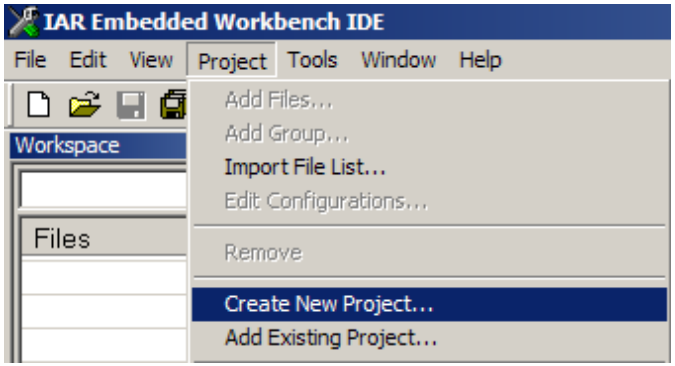
收取E-mail，获取license

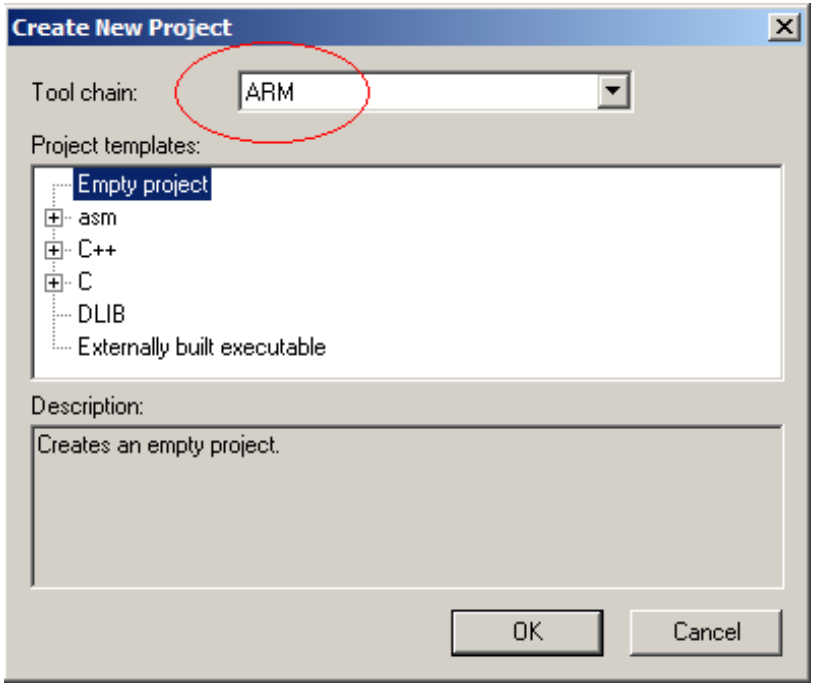
激活license

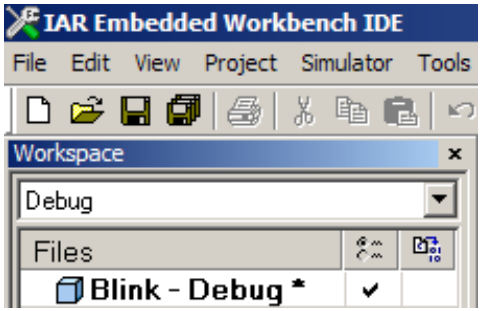


创建新的工程



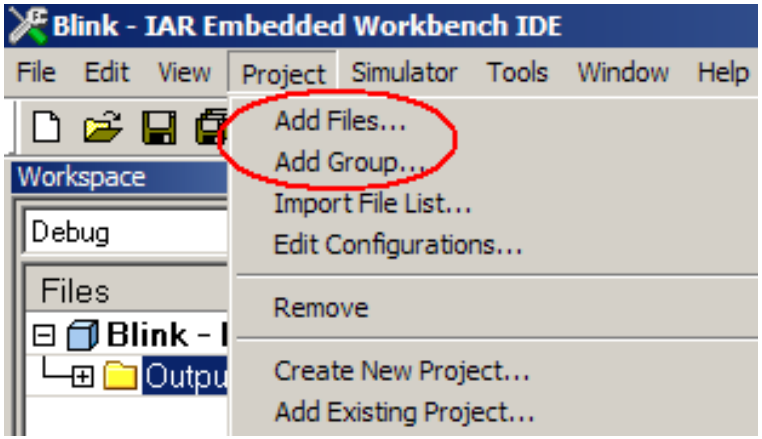






以程序Blink为例，先将原有的工程文件Blink.ewp和Blink.ewd删除，然后创建一个新的Blink Project。

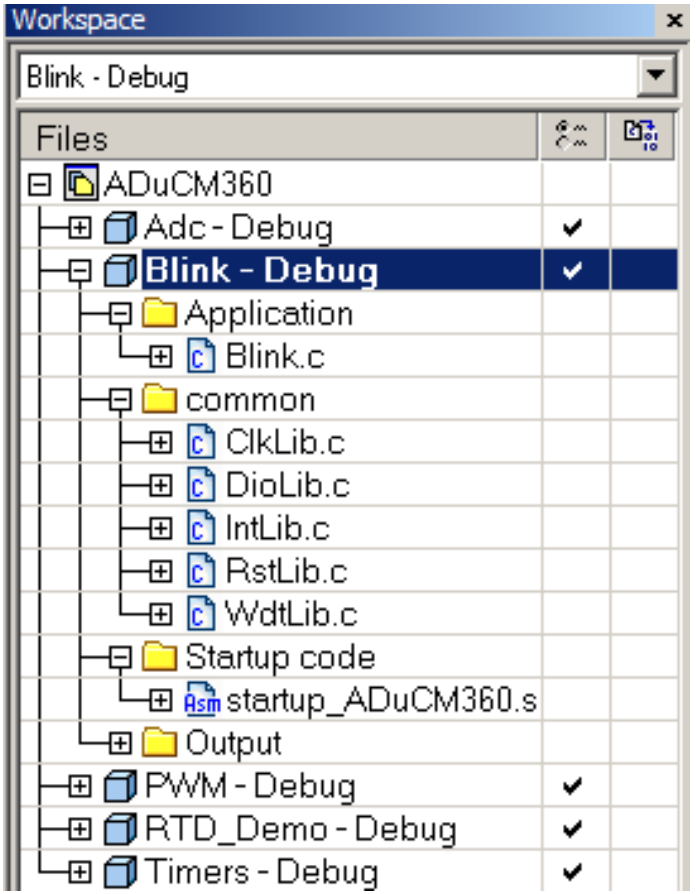
添加文件和组



Highlight the Project or a Group then:

Project menu → *Add Files*

→ *Add Group*



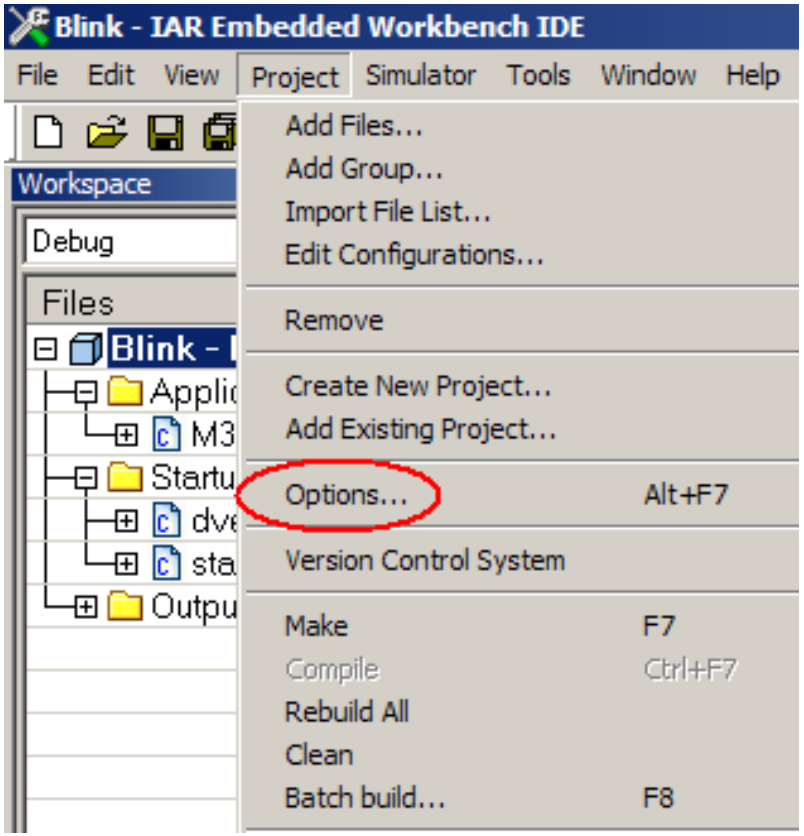
添加图示源文件到该Project中；
可以创建文件组例如Application和Common等，以使层次结构清晰。

配置工程Options

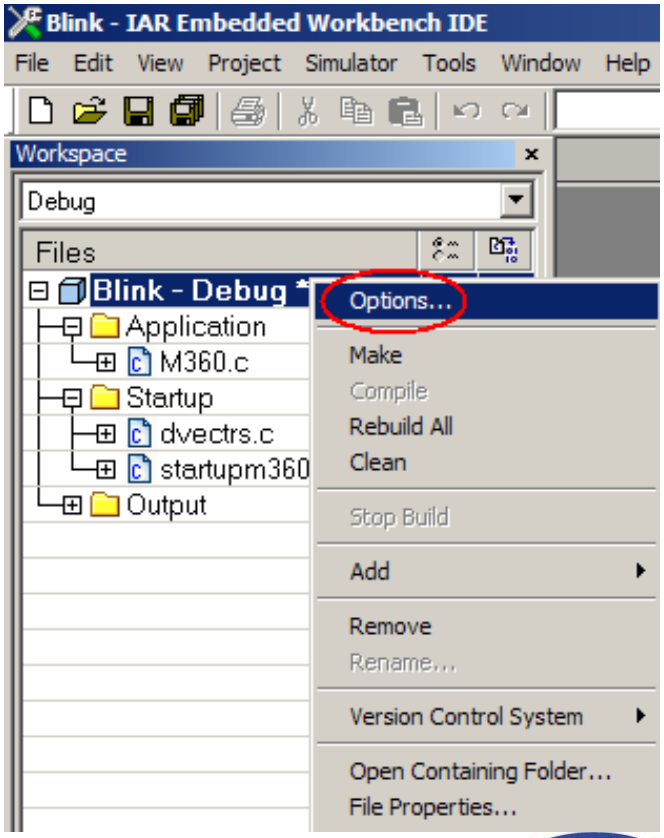
- Highlight the Project name, then:

  Main menu -> Project -> Options…

  Context menu -> Options…



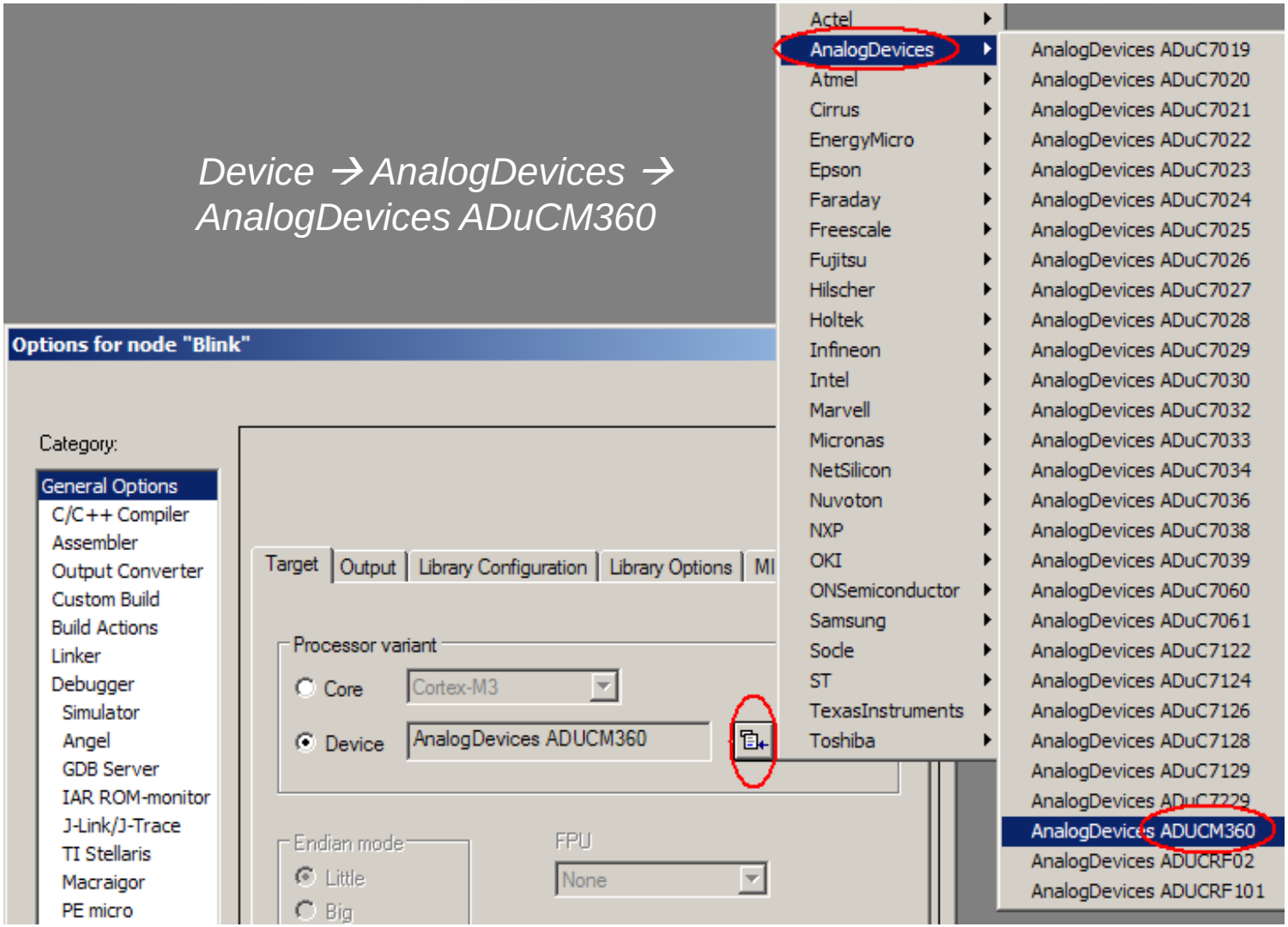
或



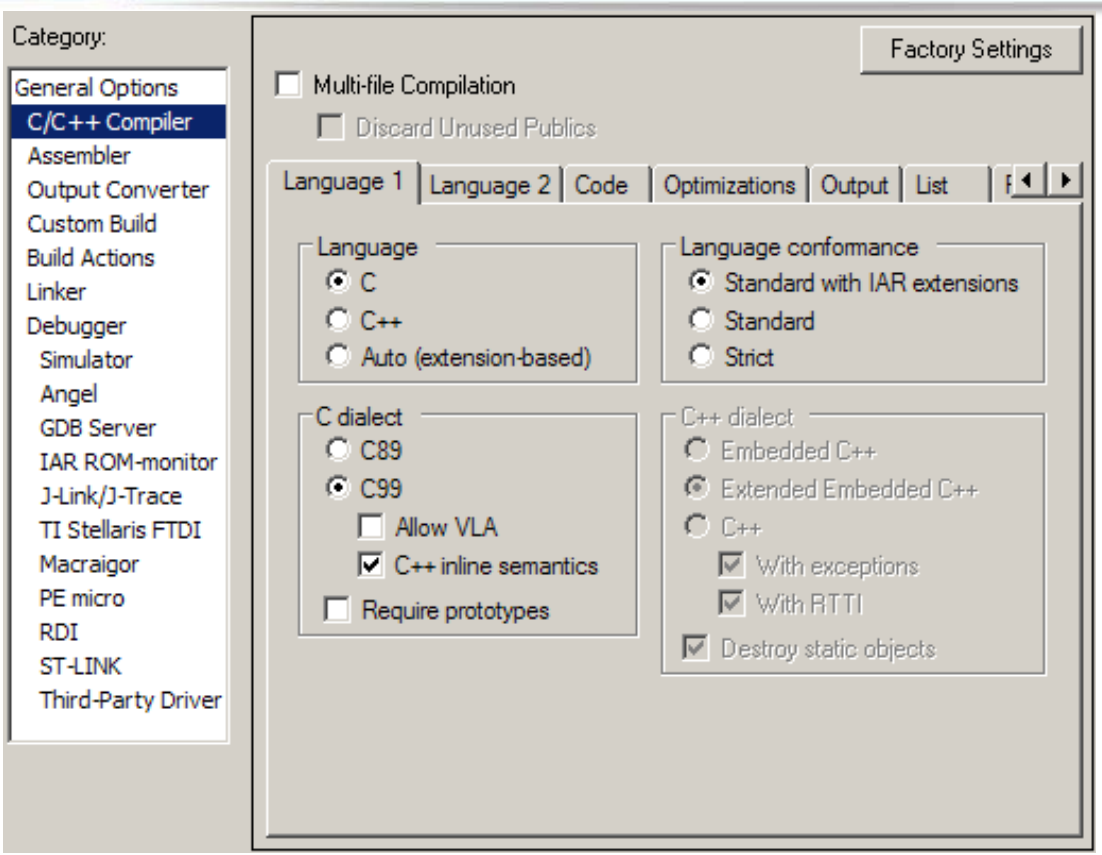
BMR 麦克泰技术

General: 处理器选择

Device → AnalogDevices → AnalogDevices ADuCM360



C/C++ Compiler: 语言选择



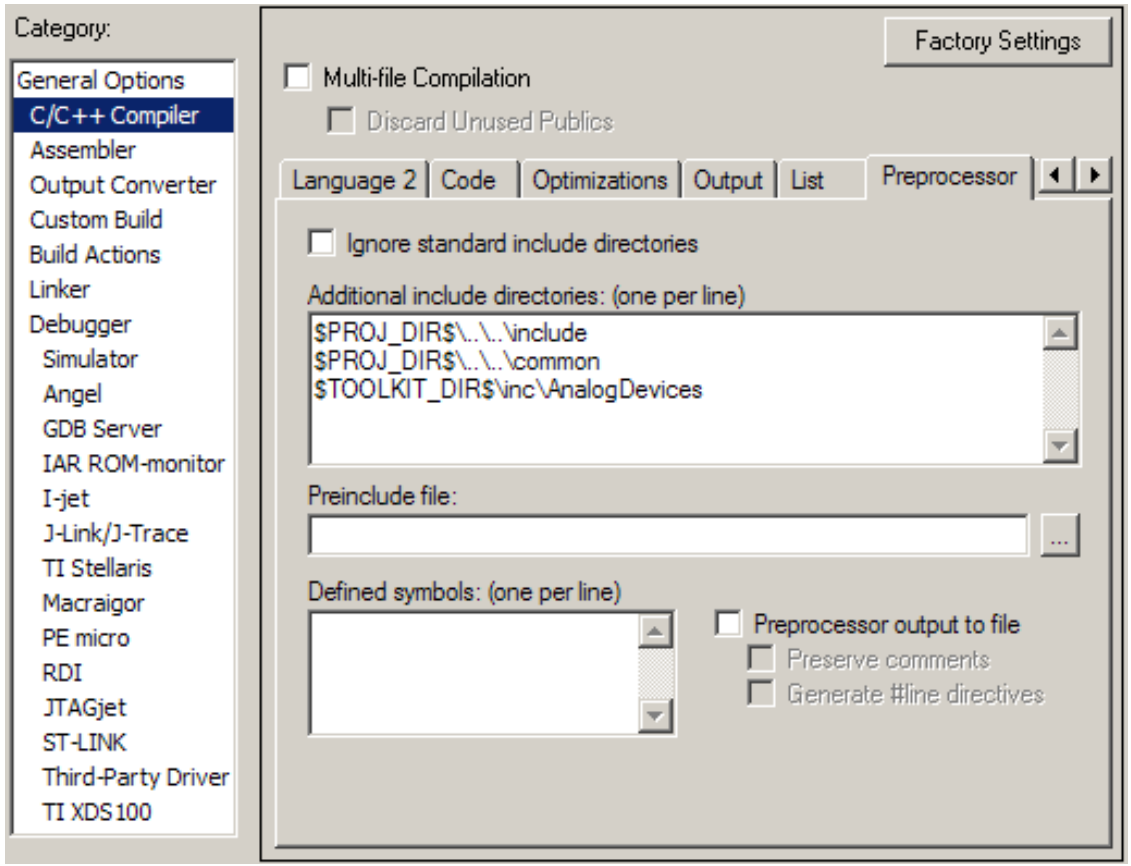
- C

  C89 or C99

- C++

  Embedded C++, Extended Embedded C++ or Full C++

C/C++ Compiler: Preprocessor

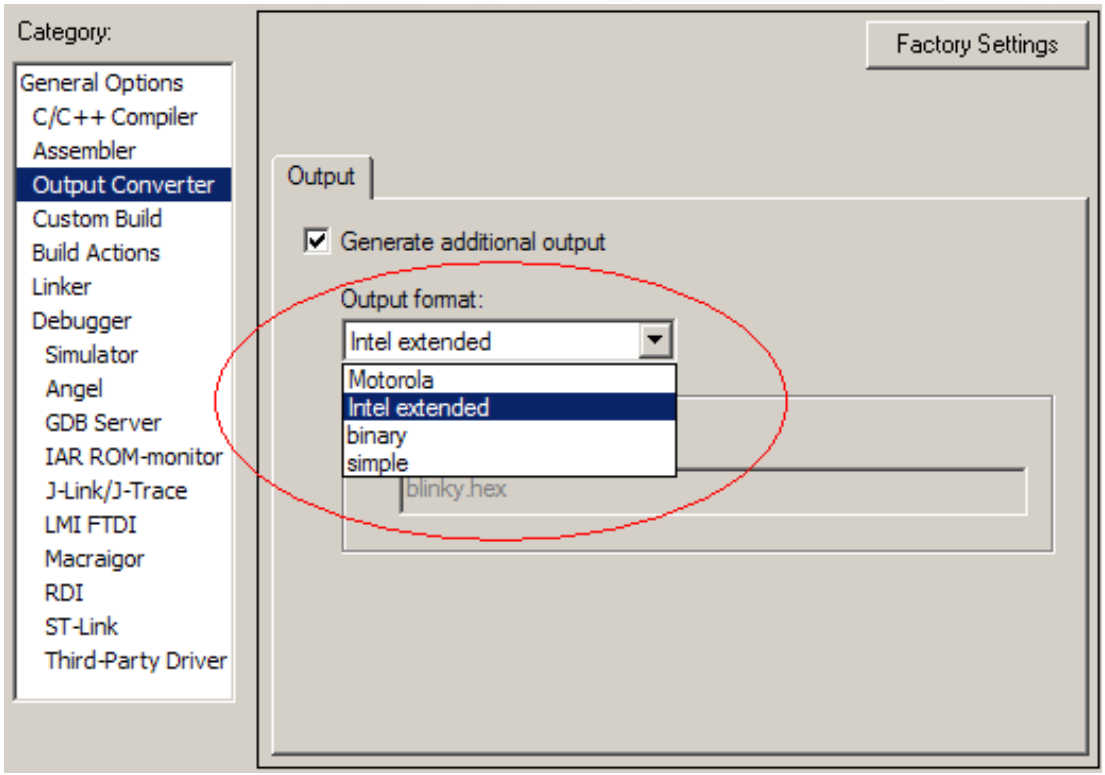


- 增加可以找到头文件的 **include**目录

$PROJ_DIR$: 当前工程的路径

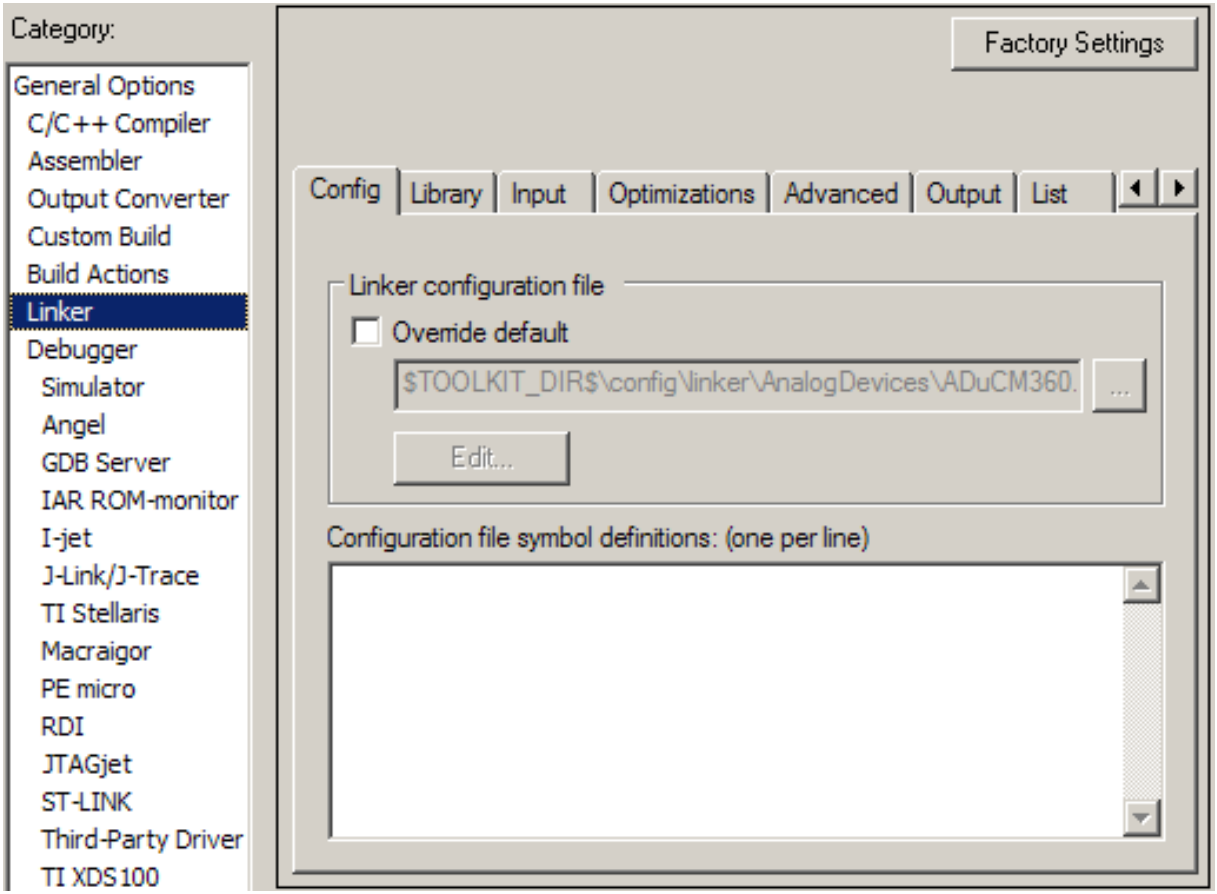
$TOOLKIT_DIR$: EWARM 安装文件夹.

Output Converter: 输出格式



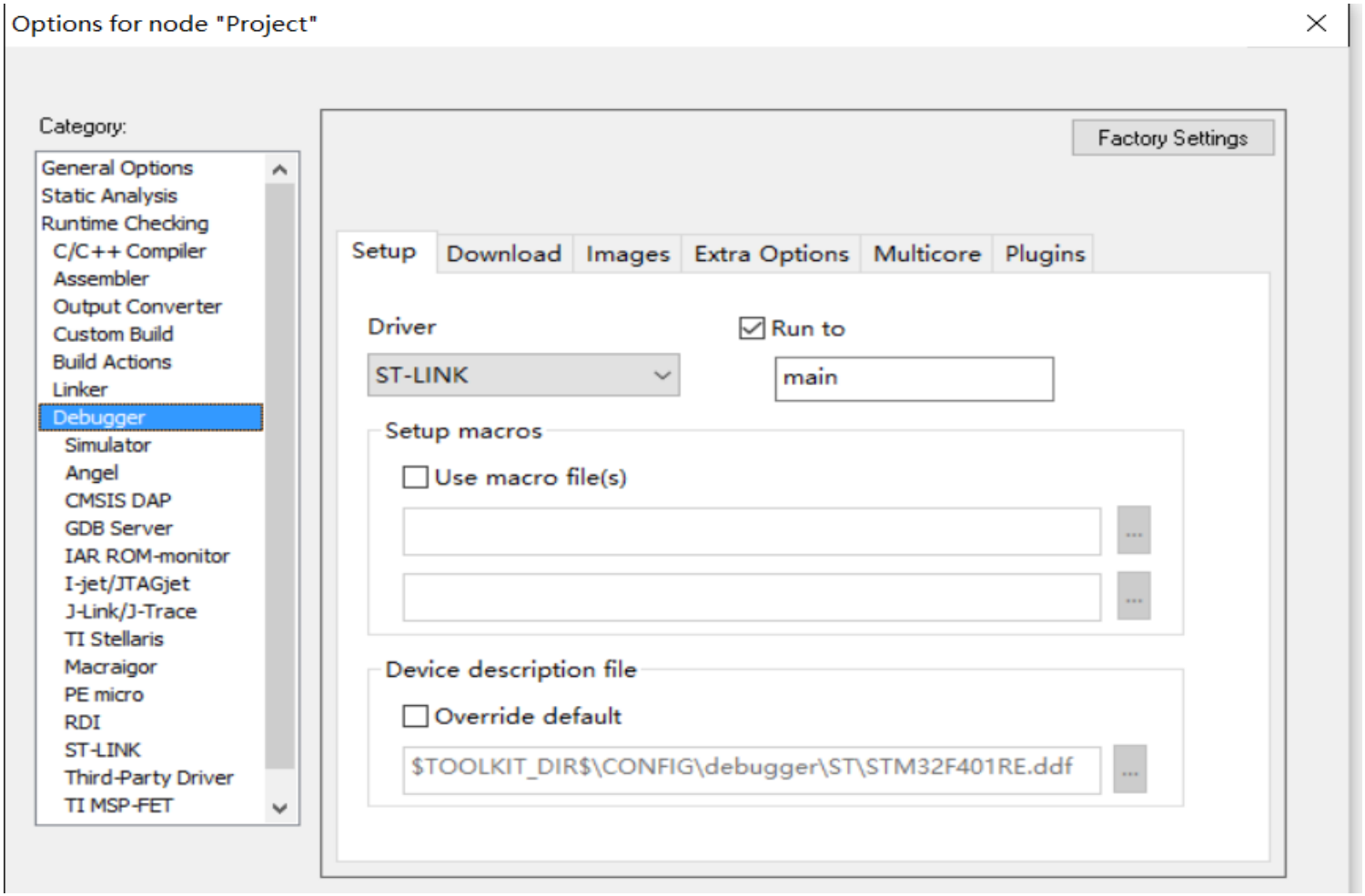
- Motorola: S-Record format
- Intel extended: Hex format
- Binary: Bin format

Linker: 配置文件

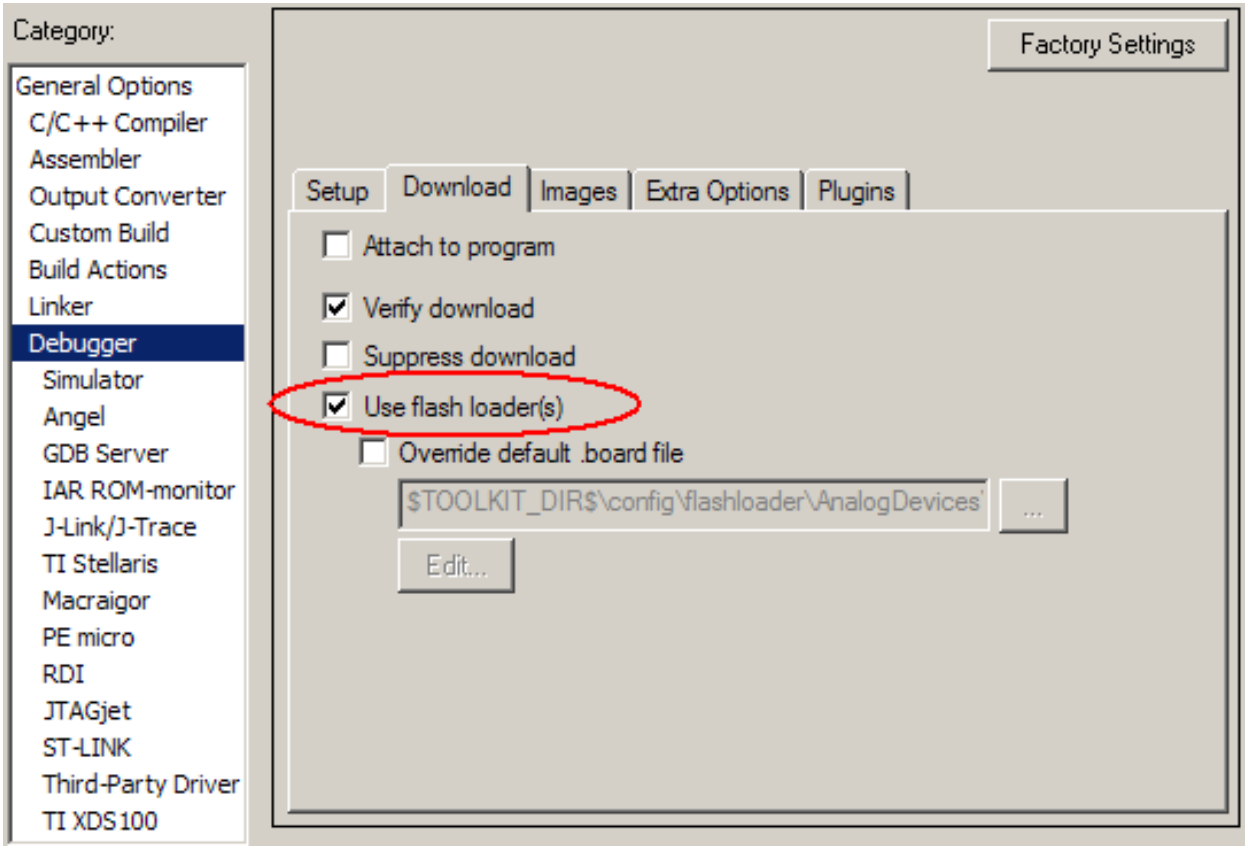


*.icf: 代码和数据定位在目标板（FLASH)的文件
$PROJ_DIR$\stm32f401xe_flash.icf

Debugger: 选择调试工具

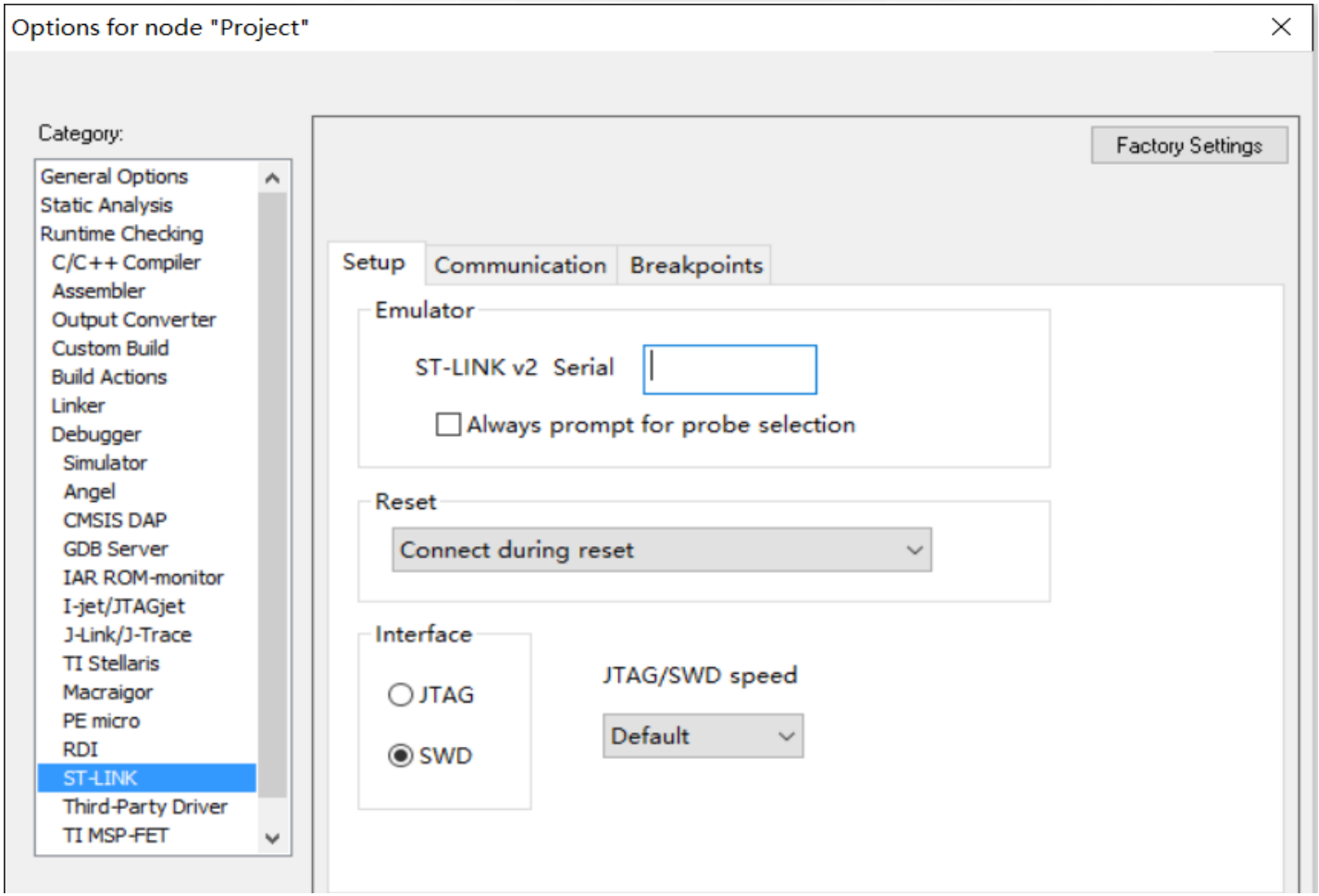


Debugger: Flash Programming



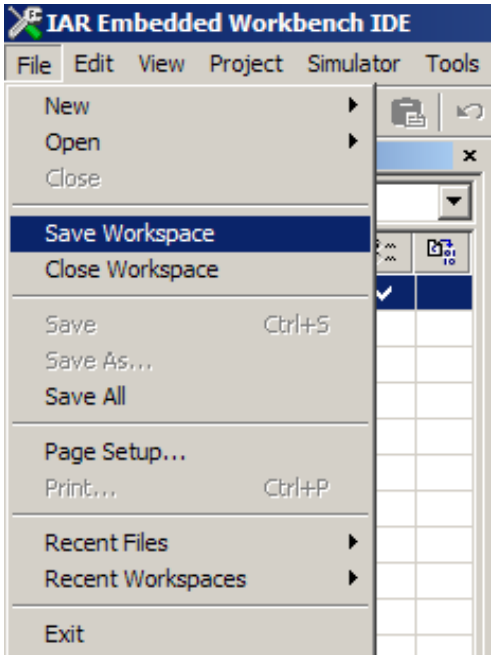
Use flash loader(s): 直接将应用程序代码下载到片上的FLASH 存储器里面

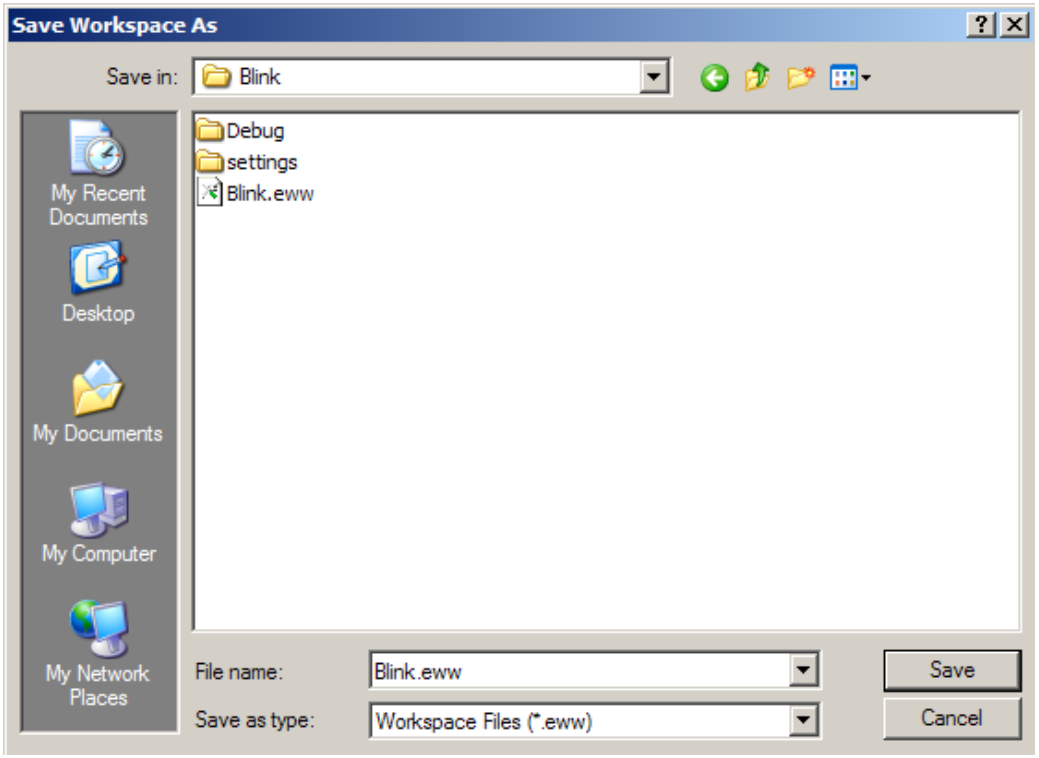
ST-Link: 连接



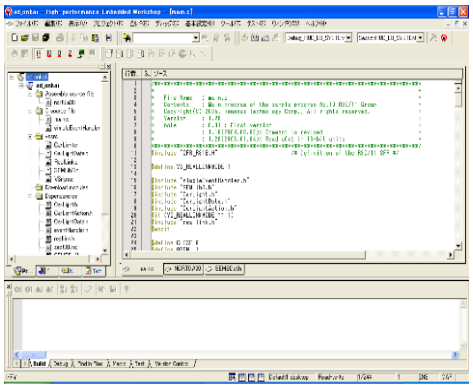
* ST-Link Connection: 选择 “SWD” 支持STM32 NUCLEO 401 devices.

保存工作区

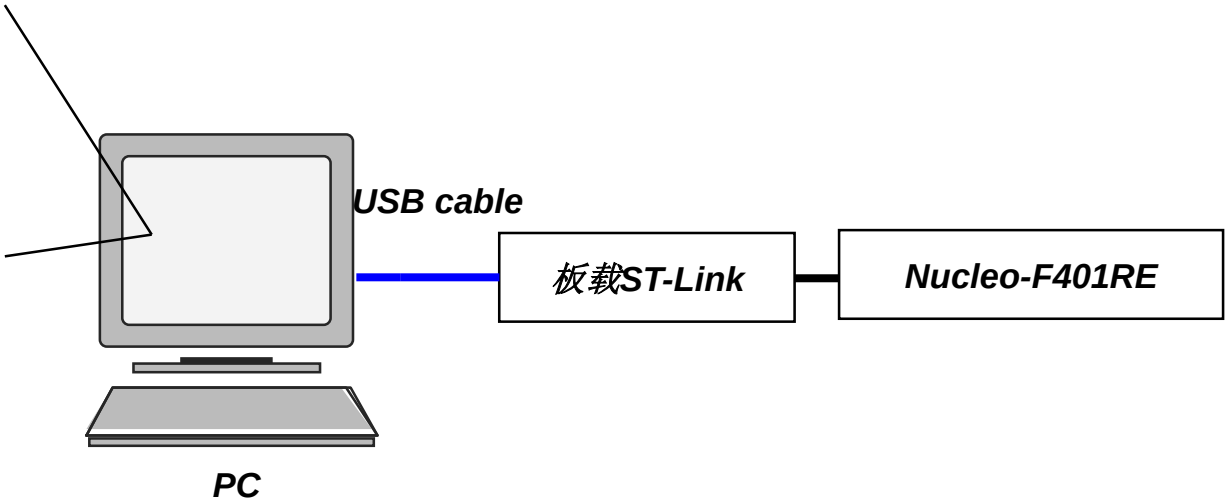




硬件连接



EWARM



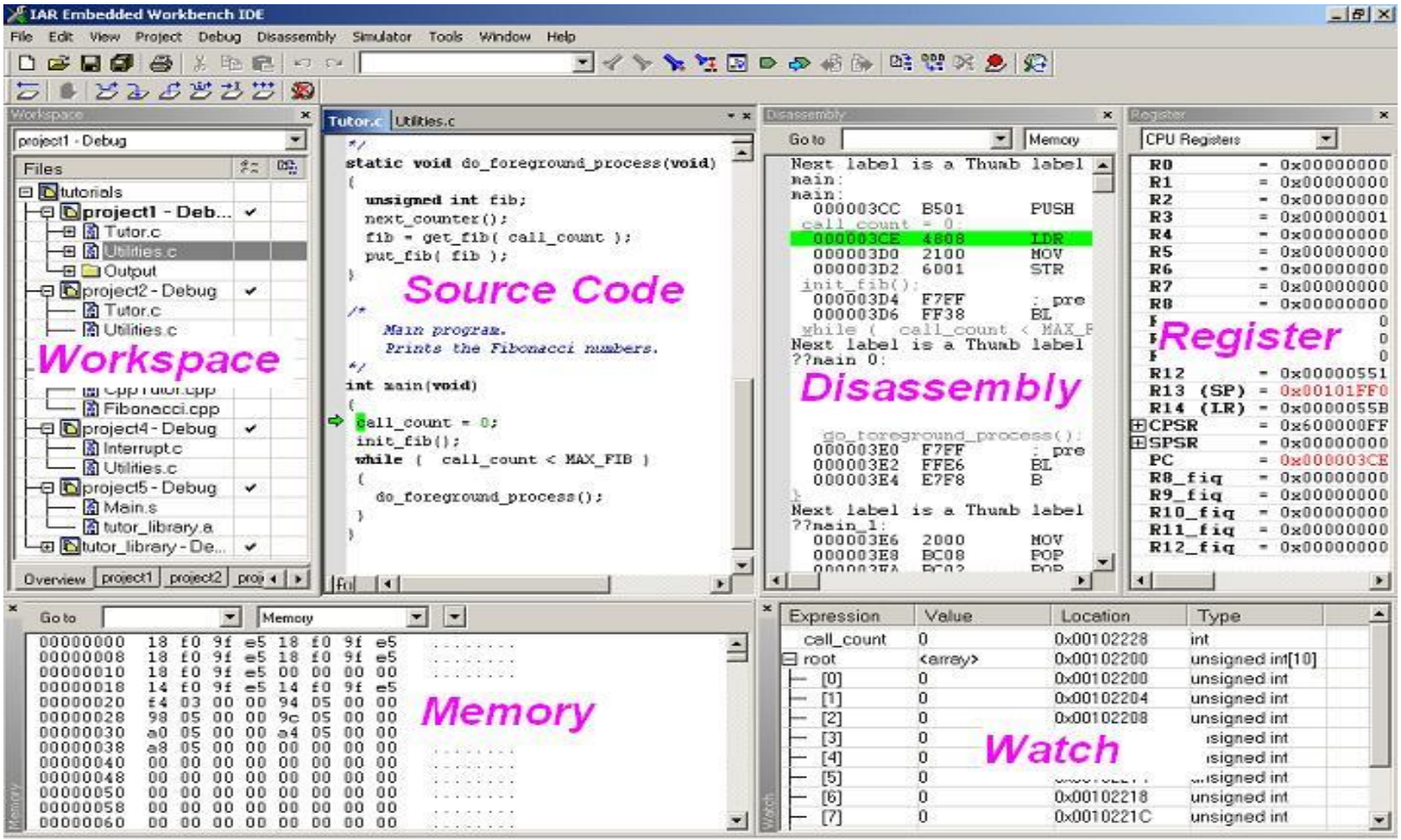
- 连接PC和**Nucleo-F401RE**
- 第一次连接ST-Link时，Windows会提示安装相应的驱动。

编译和调试

- Make or Rebuild all –编译和链接
  - **Project → Make**
  - **Project → Rebuild all**

- *Download and Debug- 下载和调试*
  - ***Project → Download and Debug***

C-SPY调试器: 窗口

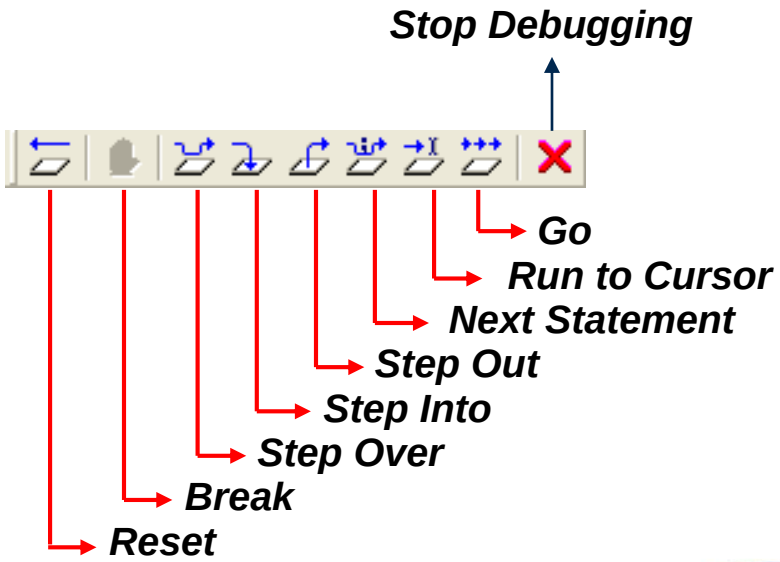


C-SPY调试器: 菜单与工具栏

* **Debug Menu and Toolbar –调试窗口和小工具**

  - **Debug Menu→ Go, Break, Reset**
    Stop Debugging
    Step Into/Over/Out
    Next Statement, Run to Cursor

  - **Toolbar:**

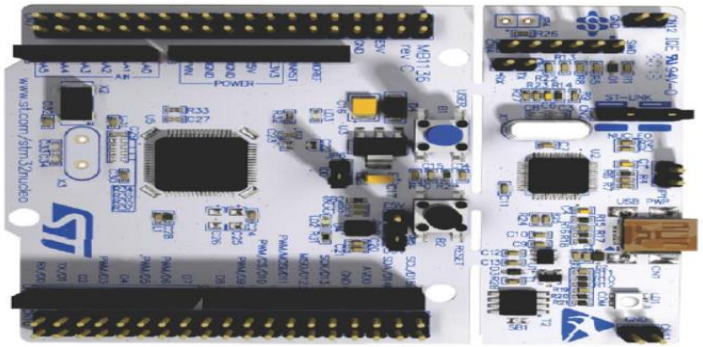


FreeRTOS实验：基于NUCLEO-F401RE

北京麦克泰软件技术有限公司

实验准备

ST NUCLEO Board



NUCLEO-F401RE

IAR EWARM

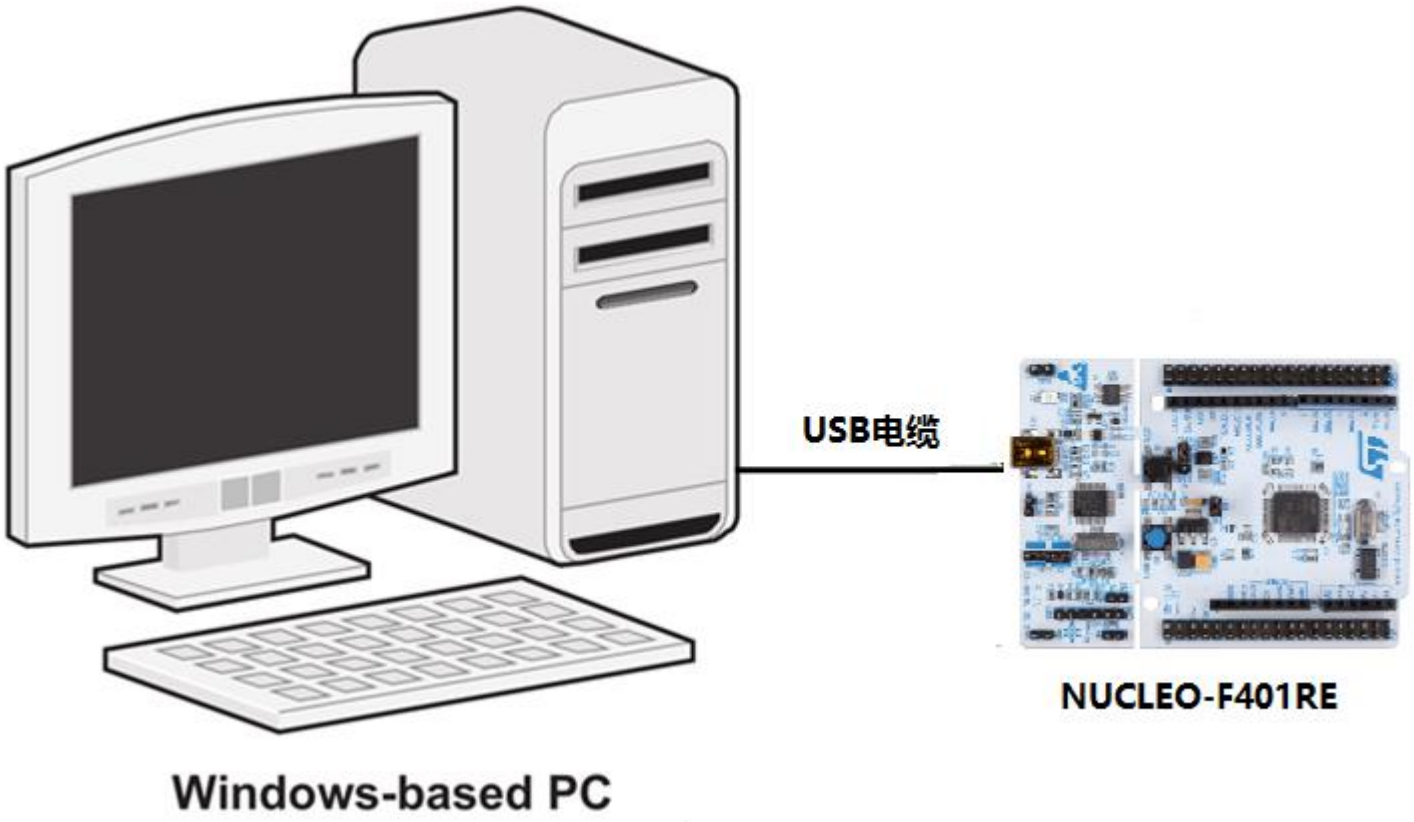
iar.com

RTOS

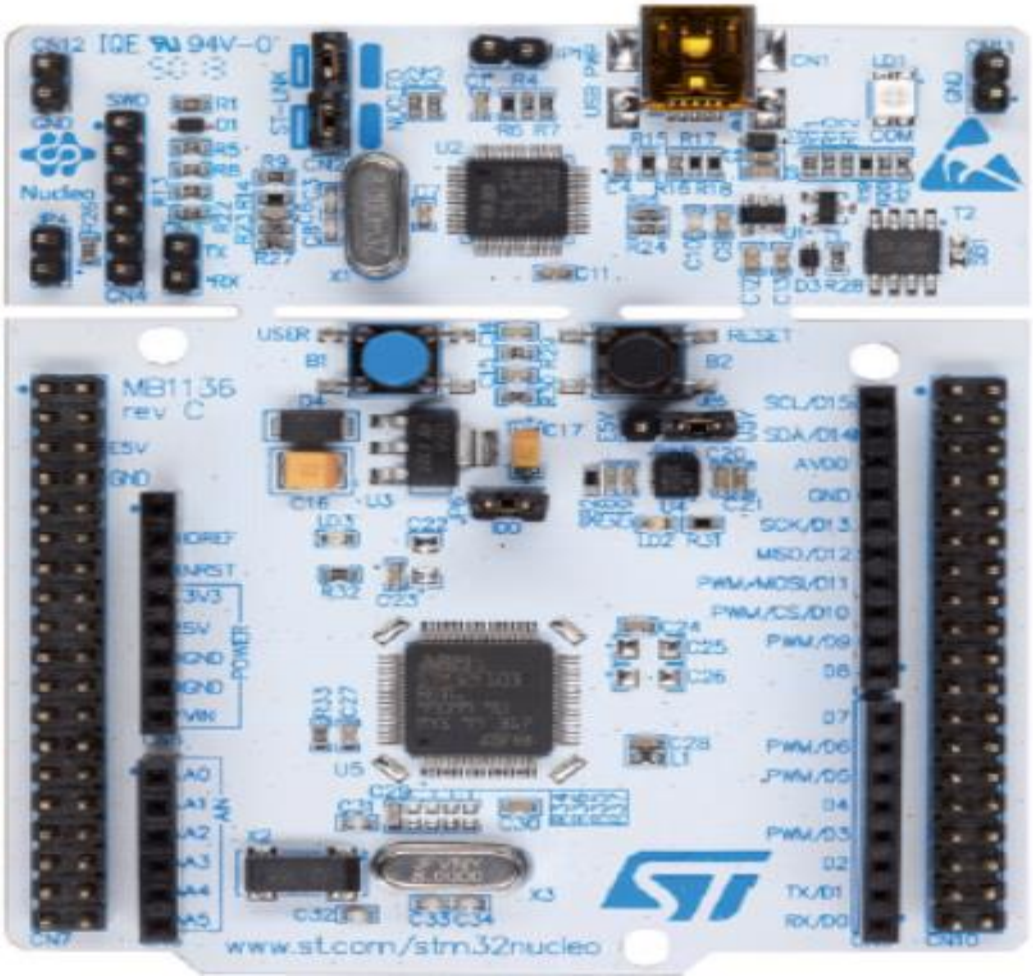
Percepio Tracealyzer

连接

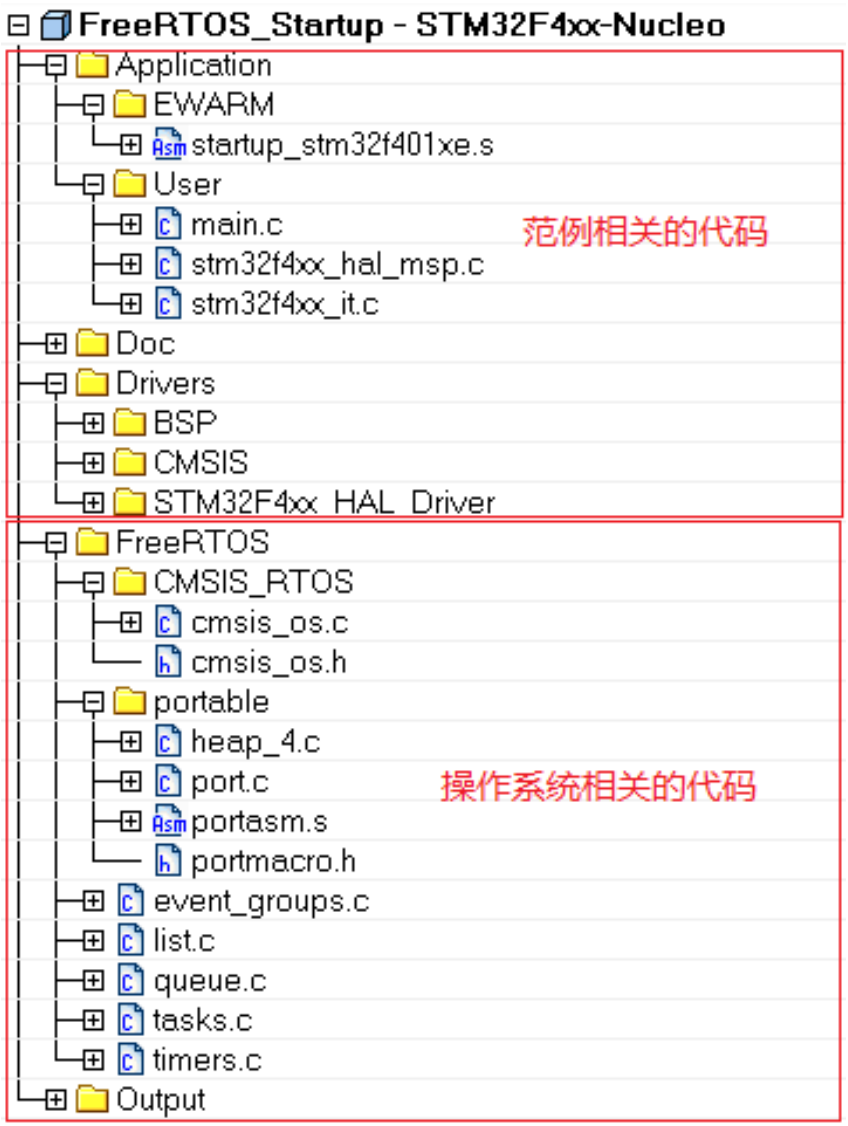
- 连接PC与NUCLEO-F401开发板



NUCLEO-F401RE开发板



实验工程架构



四个实验

- RTOS启动
- 任务管理
- 任务同步
- 任务通信