

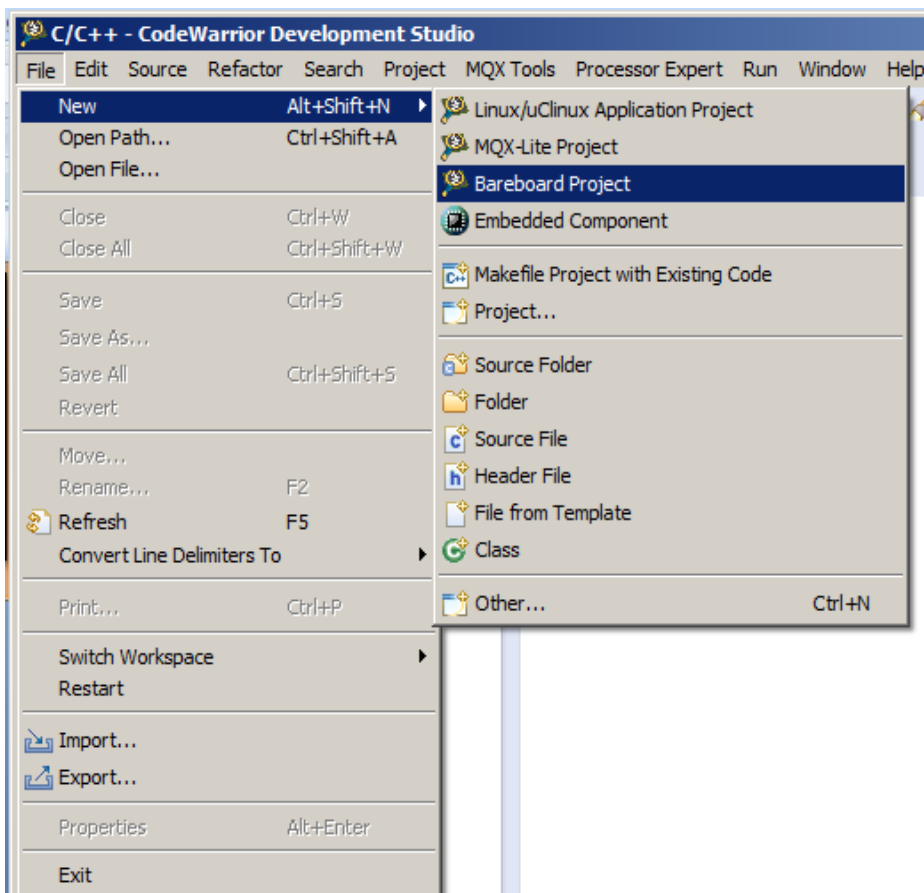
Freescal CodeWarrior 10.6 集成 开发环境（IDE）使用手册

本手册详细介绍了利用Freescal CodeWarrior 10.6 IDE 处理器专家系统（Processor Expert）快速建立KEA工程和调试的步骤，以及该IDE常用的编程及调试技巧，旨在帮助用户快速熟悉和掌握CodeWarrior 10.6的使用，利用处理器专家系统快速搭建应用工程进行产品原型验证。

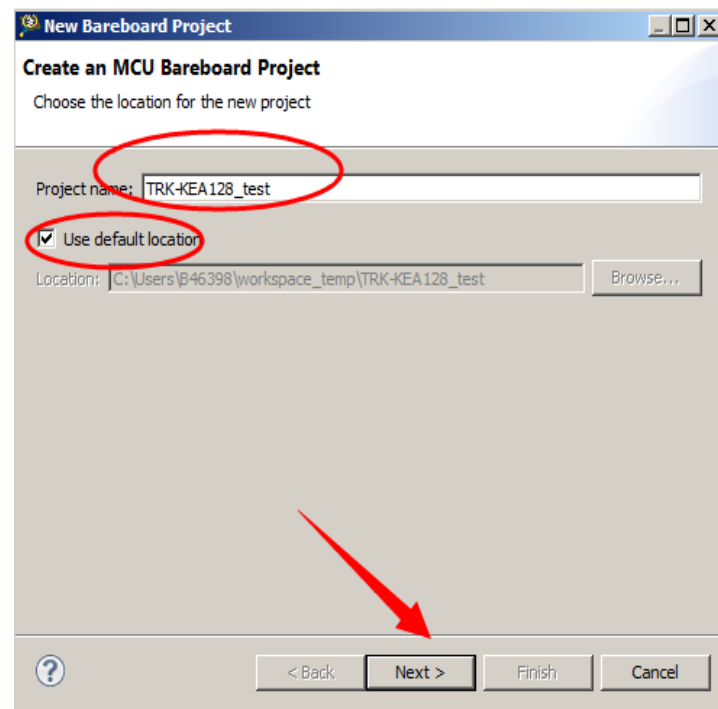
作者：胡恩伟（飞思卡尔中国汽车电子现场应用工程师）
日期：2014年10月26日星期日
版本：1.0.0

1. 利用工程向导快速创建KEA工程

a. 点击菜单File→BareBoard Project(裸板工程)



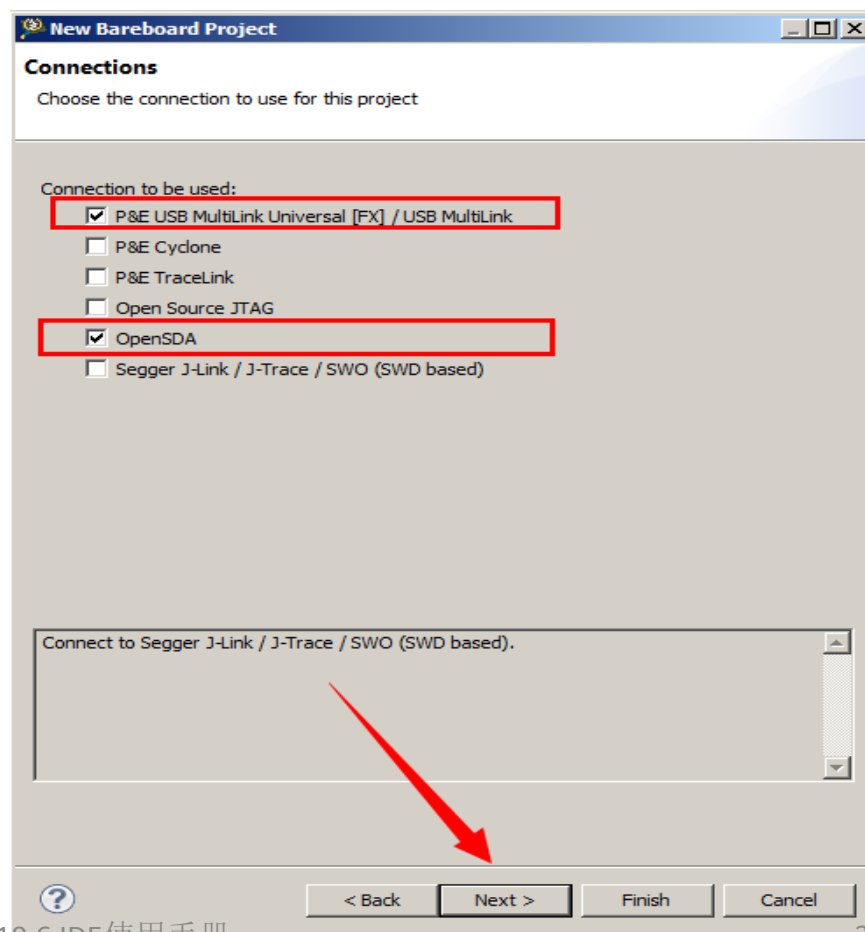
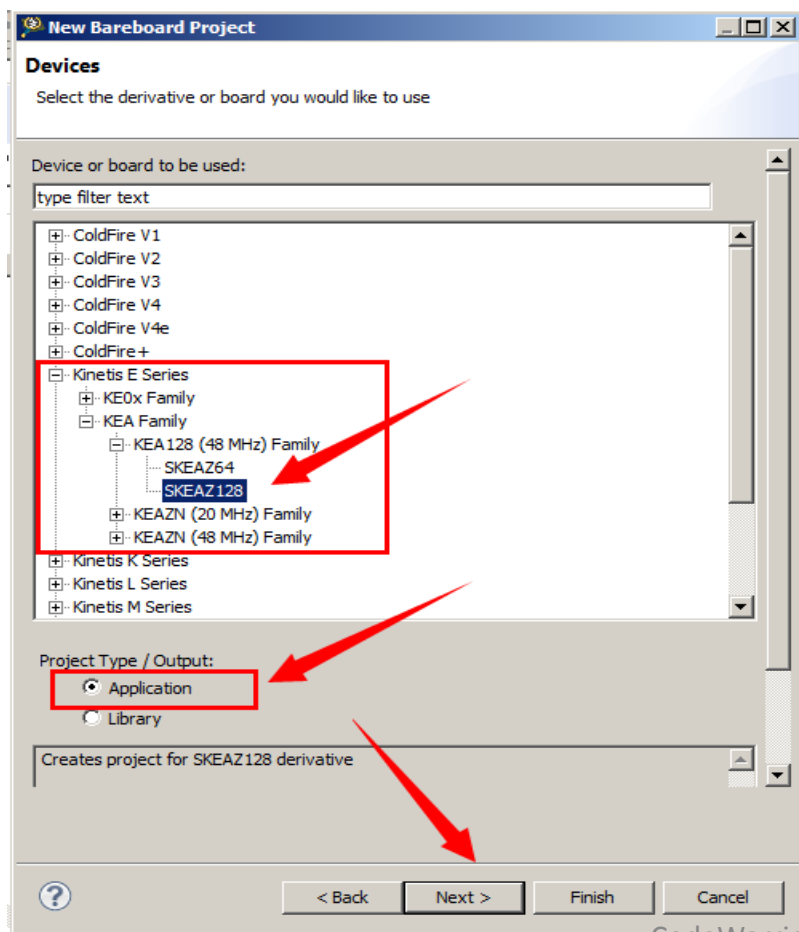
b. 输入工程名称（默认将该工程创建在当前工作空间（workspace），用户也可以将其放到其他工作空间）



利用工程向导快速创建KEA工程

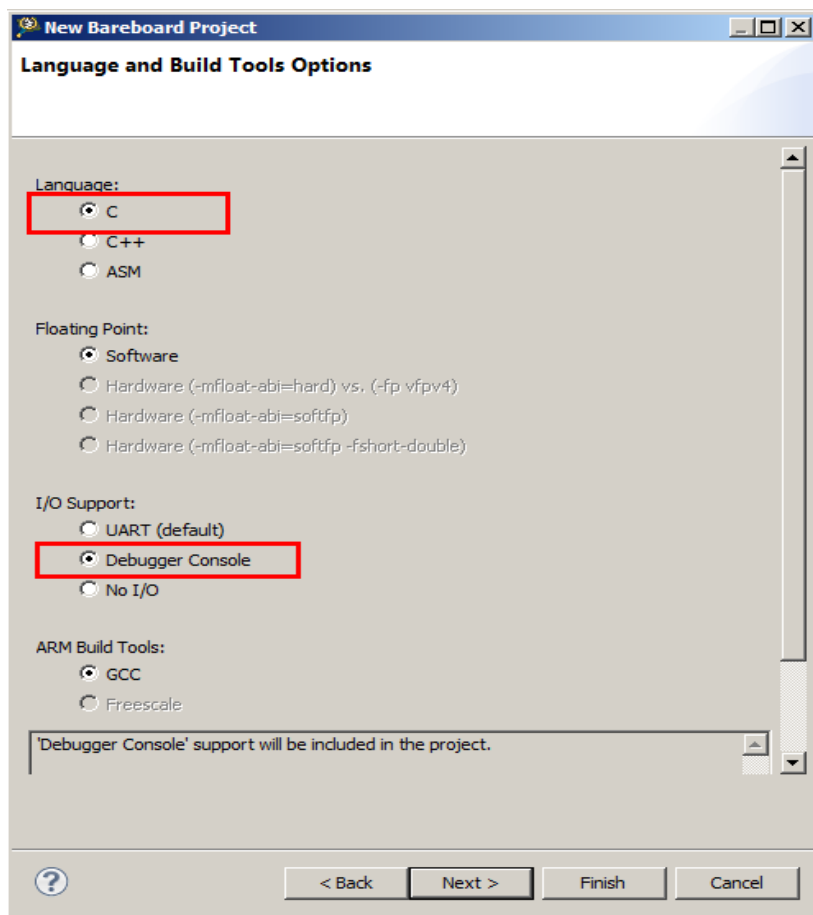
c. 选择器件，这里KEA属于Kinetis E系列，故选择如下：

d. 选择调试工具，这里必须选择TRK-KEA128板载的OpenSDA作为本工程的调试工具：

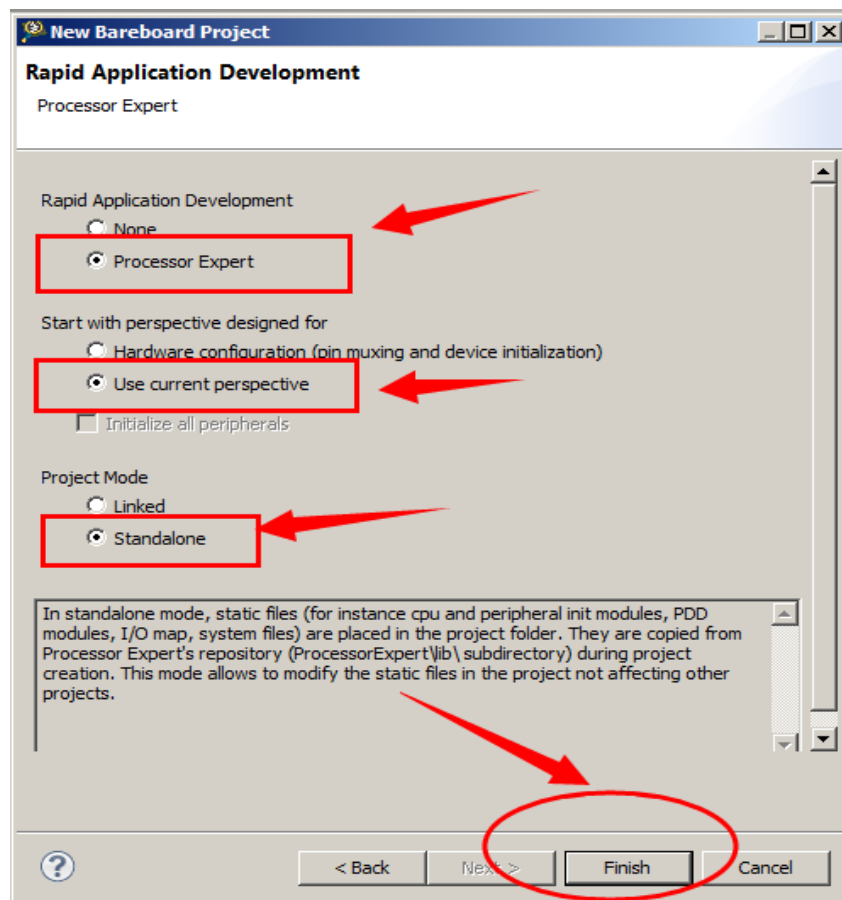


利用工程向导快速创建KEA工程

e.选择编程语言和浮点数支持以及控制台（console）硬件支持：



f.选择是否使用处理器专家系统以及工程外设driver的使用模式：



利用工程向导快速创建KEA工程

□ 处理器专家系统工程介绍

工程及文件窗口

处理器专家视窗

用户可编程代码放在Source文件夹下，其中：
Event.c和Event.h:存在中断相关的callback函数；
Main.c:为main()函数实体；

调试器相关设置

链接文件

启动代码

处理器专家为每一个组件（component）生一个对应的.h和.c文件，包含该组件图形化配置对应的驱动程序

Generated_Code

- CPU_Config.h
- Cpu.c
- Cpu.h
- Init_Config.h
- IO_Map.h
- PE_Const.h
- PE_Error.h
- PE_LDD.c
- PE_LDD.h
- PE_Types.h
- Pins1.c
- Pins1.h
- Vectors_Config.h

Project_Settings

- Debugger
 - init_kinetis.td
 - mass_erase.td
 - SKEAZ128.me
- Linker_Files
 - ProcessorExpert.ld
- Startup_Code
 - __arm_end.c
 - __arm_start.c
 - runtime_configuratic

Components - TRK-K...

- Generator_Configura
- FLASH
- OSs
- Processors
 - Cpu:SKEAZ128MLK4
- Components
 - Pins1:PinSettings
- PDD

Problems Console

Processor Expert

User working directory = C:\ProgramData\Processor Expert\CMP

Internal cache directory = C:\ProgramData\Processor Expert\F

CPU组件介绍及配置

❑使用外部晶振作为时钟源进行clock配置

❑使用内部振荡器作为时钟源进行clock配置

The screenshot shows the 'Component Inspector - Cpu' window with the 'Properties' tab selected. The 'Clock settings' section is expanded, showing the following configuration:

- Internal oscillator**: Slow internal reference clock [kHz] is 32.768.
- System oscillator**: Enabled (indicated by a red box and the text '使能外部晶振').
- Clock source**: External crystal.
- Clock frequency [MHz]**: 8.0 (indicated by a red box and the text '输入外部晶振频率').
- Clock source settings**: 1.
- Clock source setting 0**:
 - ICS settings**:
 - ICS mode: FEE (indicated by a red box and the text '选择FLL工作模式为FEE').
 - ICS external ref. clock [MHz]: 8.0.
 - FLL settings**:
 - FLL module: Enabled.
 - FLL output [MHz]: 40.0 (indicated by a red box and the text '配置FLL倍频输出, 处理器专家自动选择配置系数').
 - ICS output prescaler: Auto select.
 - ICS output clock: 40.0.
- Clock configurations**: 1.
- Clock configuration 0**:
 - Clock source setting**: configuration 0.
 - System clocks**:
 - Core clock: 40.0.
 - Bus clock: 20.0.
 - Timer clock: 40.0.

The screenshot shows the 'Component Inspector - Cpu' window with the 'Properties' tab selected. The 'Clock settings' section is expanded, showing the following configuration:

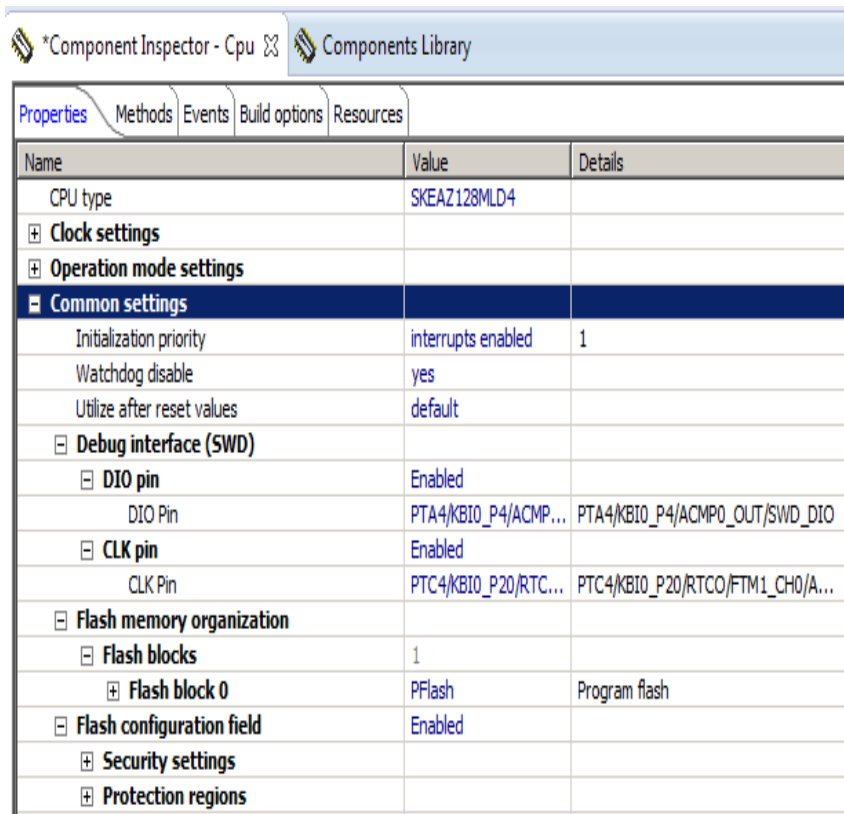
- Internal oscillator**: Slow internal reference clock [kHz] is 32.768.
- System oscillator**: Disabled (indicated by a red box and the text '除能外部晶振').
- Clock source settings**:
 - Clock source setting 0**: Settings of the system oscillator. If enabled, the system oscillator can be used for peripherals.
 - ICS settings**:
 - ICS mode: FEE (indicated by a red box and the text '选择FLL工作模式为FEE').
 - ICS external ref. clock [MHz]: 0.0.
 - FLL settings**:
 - FLL module: Enabled.
 - FLL output [MHz]: 41.94304.
 - Reference clock source**: Slow internal clock.
 - ICS output prescaler: Auto select.
 - ICS output clock: 41.94304.
- Clock configurations**: 1.
- Clock configuration 0**:
 - Clock source setting**: configuration 0.
 - System clocks**:
 - Core clock: 41.94304.
 - Bus clock: 20.97152.
 - Timer clock: 41.94304.

注意：这里内核/总线/定时器时钟频率不能配合为整数的原因是，FLL用的是内部32.768KHz的时钟源，其整数倍频不为整数

CPU组件介绍及配置

❑在CPU组件的属性设置中还包括常规设置（common settings）

其中包含了默认看门狗、SWD调试口以及Flash Memory的设置

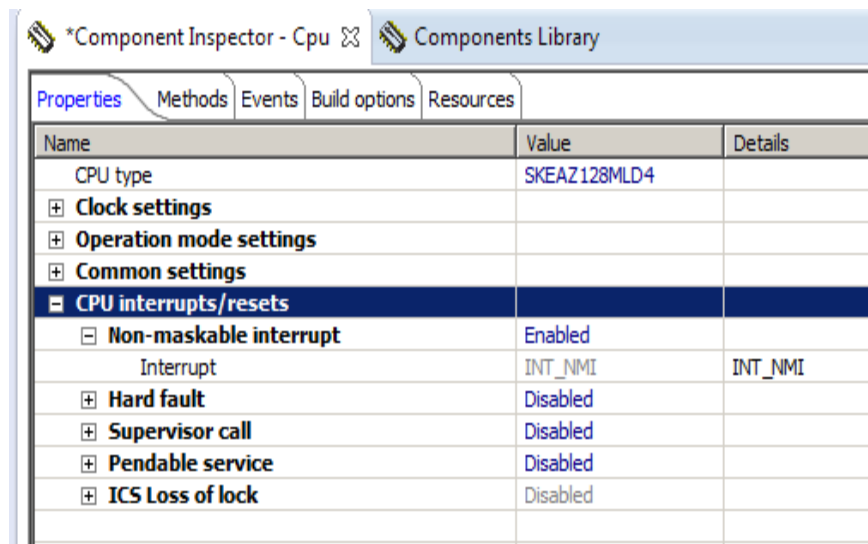


The screenshot shows the 'Component Inspector - Cpu' window with the 'Common settings' tab selected. The table lists various configuration options for the CPU component.

Name	Value	Details
CPU type	SKEAZ128MLD4	
⊕ Clock settings		
⊕ Operation mode settings		
⊖ Common settings		
Initialization priority	interrupts enabled	1
Watchdog disable	yes	
Utilize after reset values	default	
⊖ Debug interface (SWD)		
⊖ DIO pin	Enabled	
DIO Pin	PTA4/KBIO_P4/ACMP...	PTA4/KBIO_P4/ACMP0_OUT/SWD_DIO
⊖ CLK pin	Enabled	
CLK Pin	PTC4/KBIO_P20/RTC...	PTC4/KBIO_P20/RTC0/FTM1_CH0/A...
⊖ Flash memory organization		
⊖ Flash blocks	1	
⊕ Flash block 0	PFlash	Program flash
⊖ Flash configuration field	Enabled	
⊕ Security settings		
⊕ Protection regions		

❑在CPU组件的属性设置中还包括CPU内核中断/复位设置（CPU interrupt/reset）

其中包含了CPU内核系统级中断（ARM Cortex M0+实现的异常）：不可屏蔽中断NMI、硬件错误异常Hard fault（当CPU执行非法指令、非对其地址访问时触发该异常，可以用于捕获程序跑飞时的场景）、超级调用Supervisor Call和可请求服务异常（用于RTOS系统任务切换），以及内部时钟失锁（ICS Loss of lock）。所有这些中断的优先级都高于外设中断。

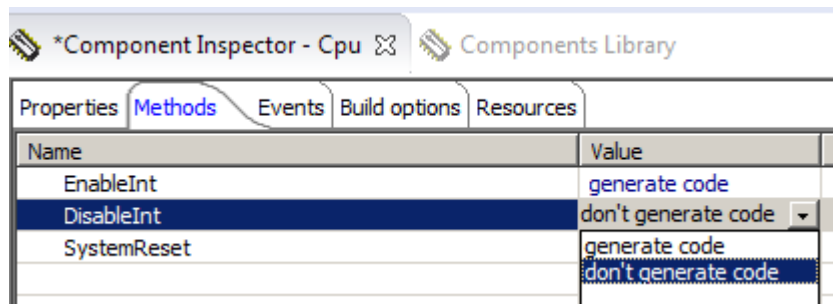


The screenshot shows the 'Component Inspector - Cpu' window with the 'CPU interrupts/resets' tab selected. The table lists various configuration options for CPU interrupts and resets.

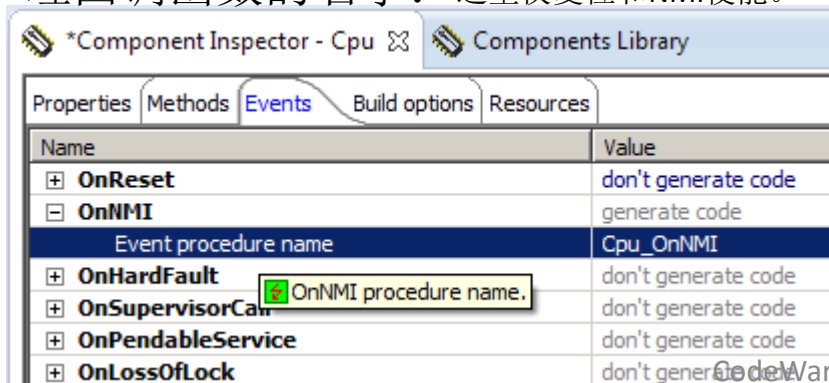
Name	Value	Details
CPU type	SKEAZ128MLD4	
⊕ Clock settings		
⊕ Operation mode settings		
⊕ Common settings		
⊖ CPU interrupts/resets		
⊖ Non-maskable interrupt	Enabled	
Interrupt	INT_NMI	INT_NMI
⊕ Hard fault	Disabled	
⊕ Supervisor call	Disabled	
⊕ Pendable service	Disabled	
⊕ ICS Loss of lock	Disabled	

CPU组件介绍及配置

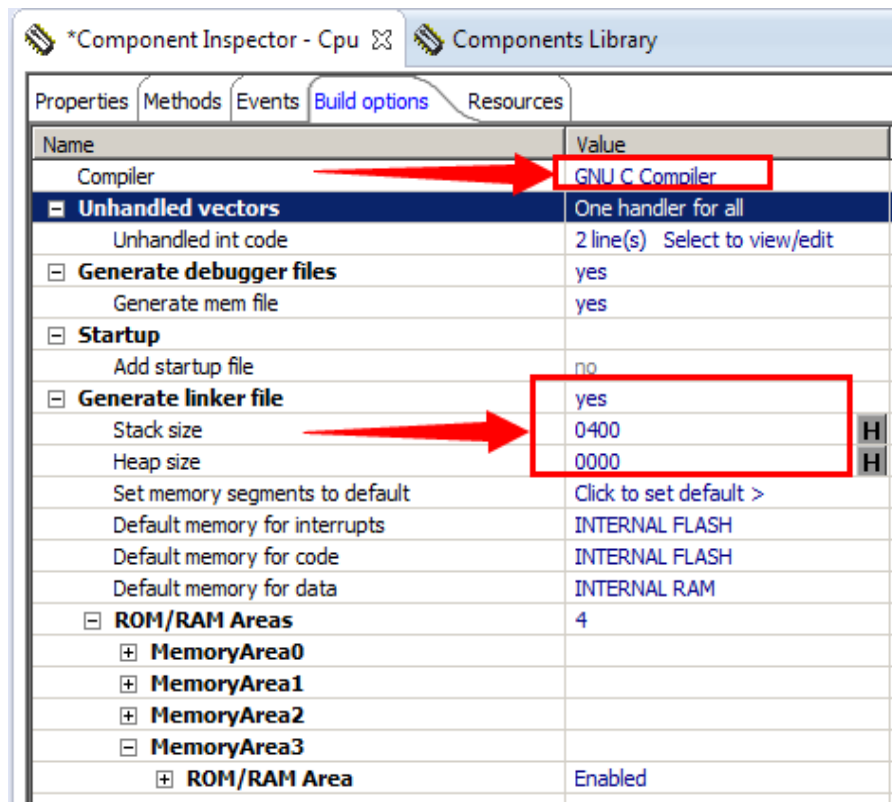
❑在CPU组件的方法（Methods）设置中包含了该组件属性配置所产生的API函数：这里可以配置是否生产使能/除能外设中断以及系统复位函数



❑在CPU组件的事件（Eventss）设置中包含了该组件属性配置所产生的中断处理回调函数的名字：这里仅复位和NMI使能。



❑在CPU组件的编译选型（Build options）设置可进行memory资源的分配：特别是堆栈的设置以及为处理中断向量的处理，处理器专家依据此配置生成过程的链接文件

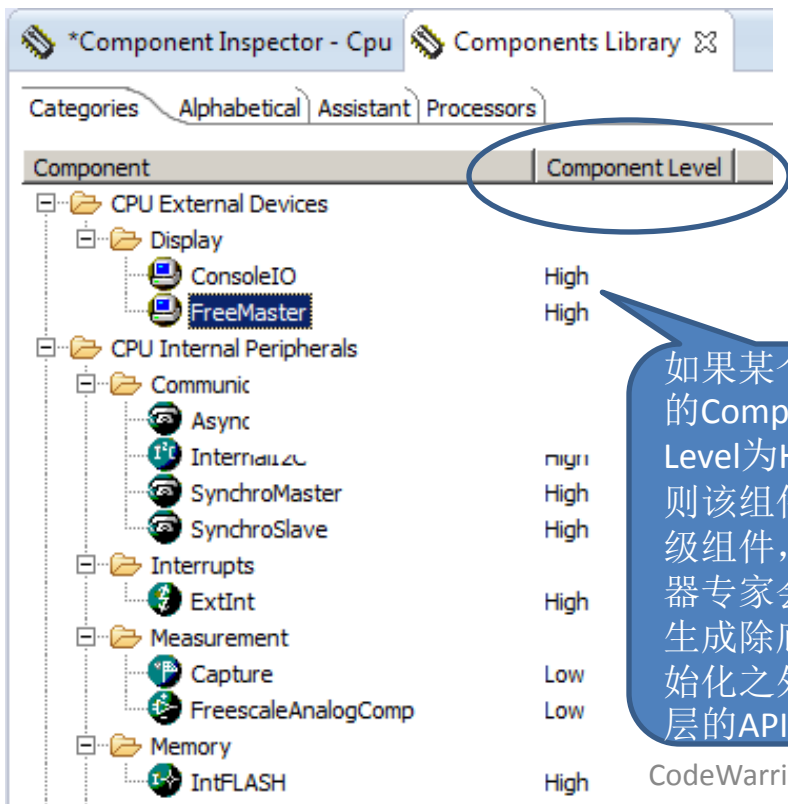


处理器专家组件库（Component Library）介绍

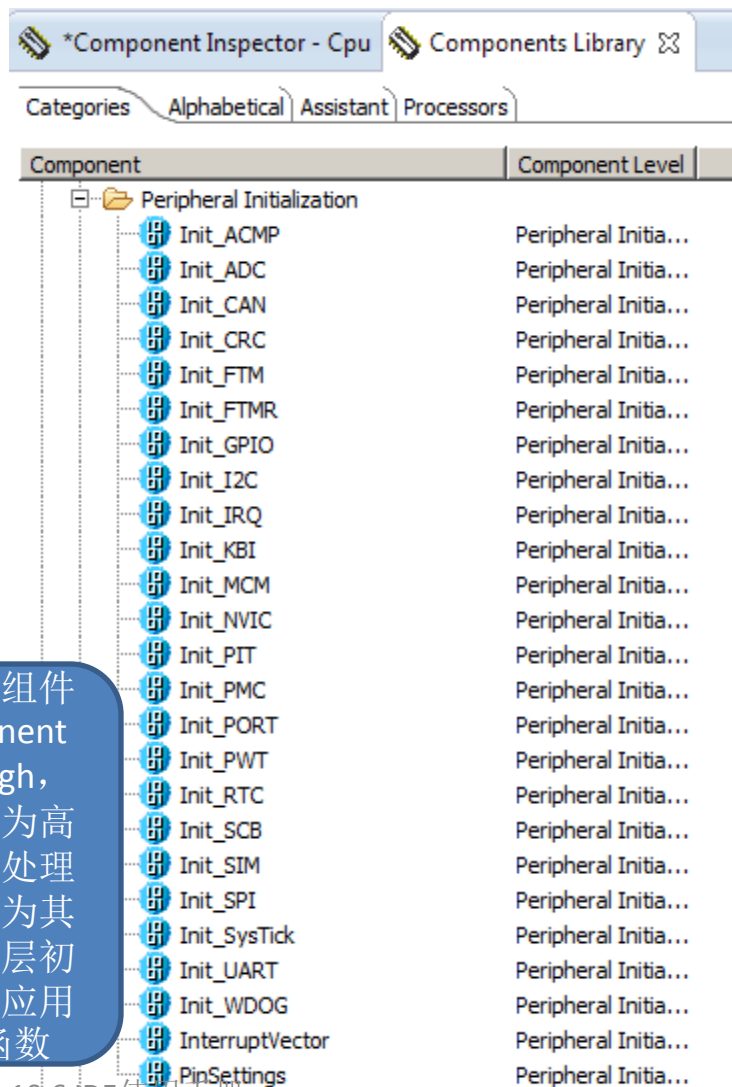
❑CPU外部设备：控制台IO和FreeMaster组件

❑CPU内部外设：

- 通信：同步/异步串行通信
- 中断：外部中断
- 测量：捕捉（timer 输入捕捉）和模拟比较器（AMCP）
- 存储器：Flash初始化及驱动
- 外设初始化：各MCU内部外设的初始化组件



如果某个组件的Component Level为High，则该组件为高级组件，处理器专家会为其生成除底层初始化之外应用层的API函数



处理器专家组件库（Component Library）介绍

❑CPU内部外设：

➤I/O输入输出：单bit I/O和多bit I/O以及并行I/O

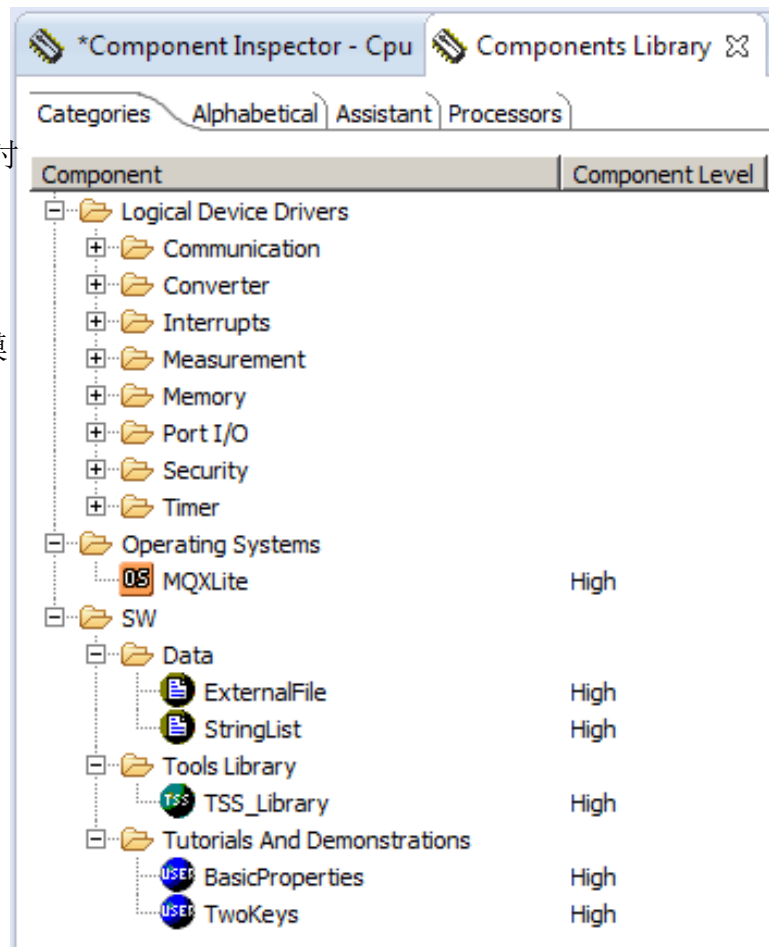
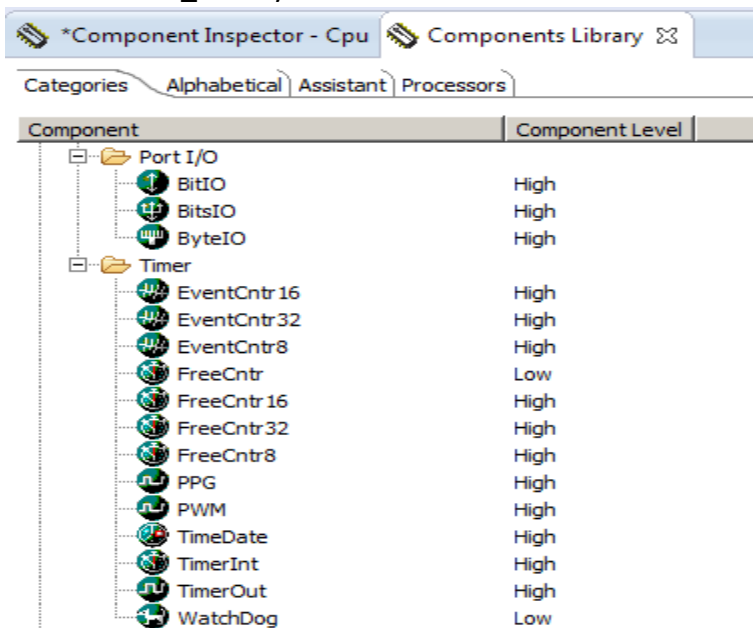
➤定时器：timer资源的各种应用，如定时事件

（EventCnt16/32/8）、自由计数器（FreeCnt16/32/8）、脉宽调整输出（PWM）、软件日历功能（TimeDate）、定时器中断（TimerInt）、定时器输出（TimeOut）以及看门狗

❑逻辑设备驱动：各种片上资源/外设的逻辑层driver

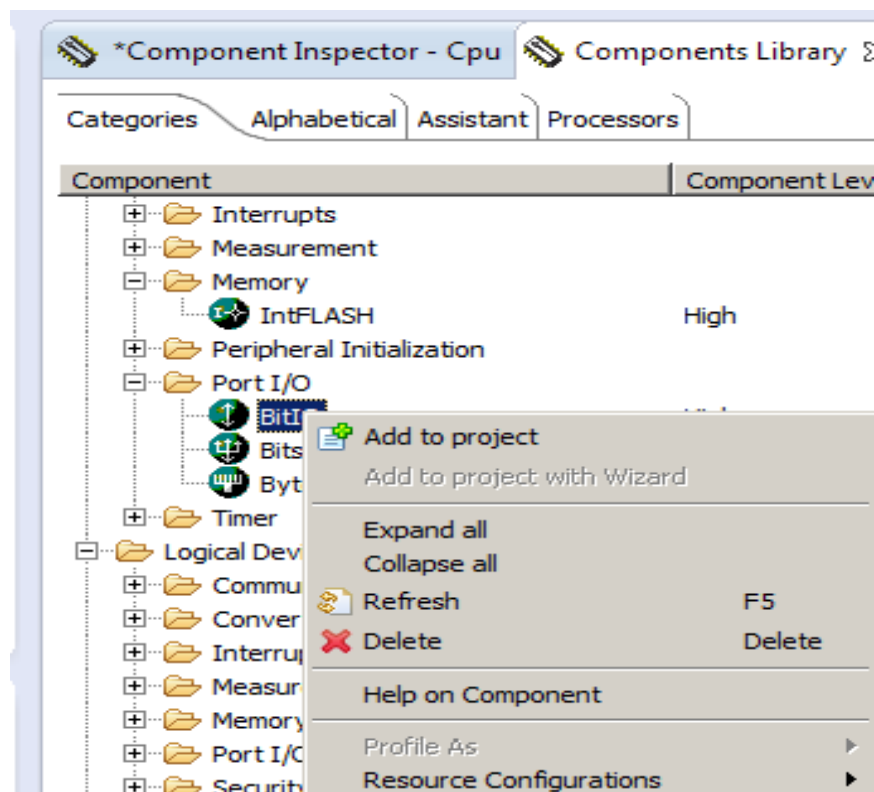
❑操作系统：支持KEA MCU的MQX Lite

❑软件模拟：包含以下外部文件字符的输入接口，触摸感应输入库TSS_Library等

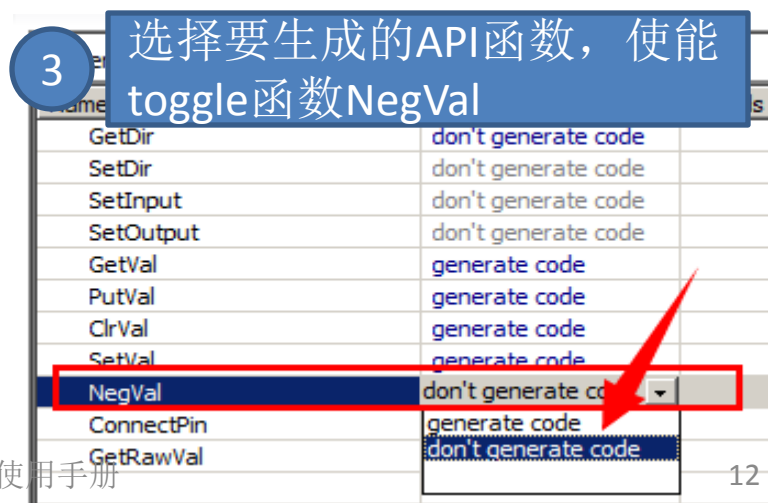
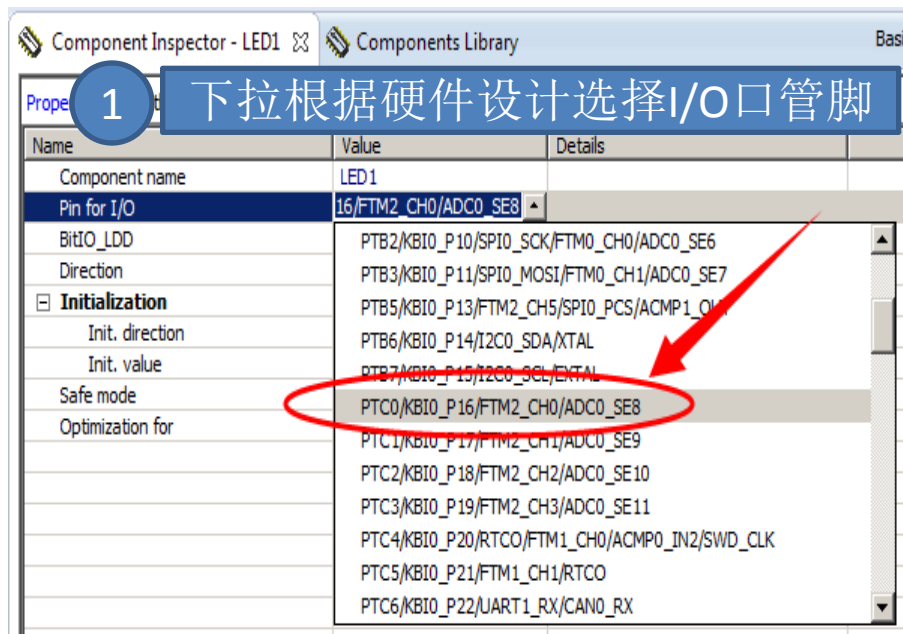


添加单bit I/O组件

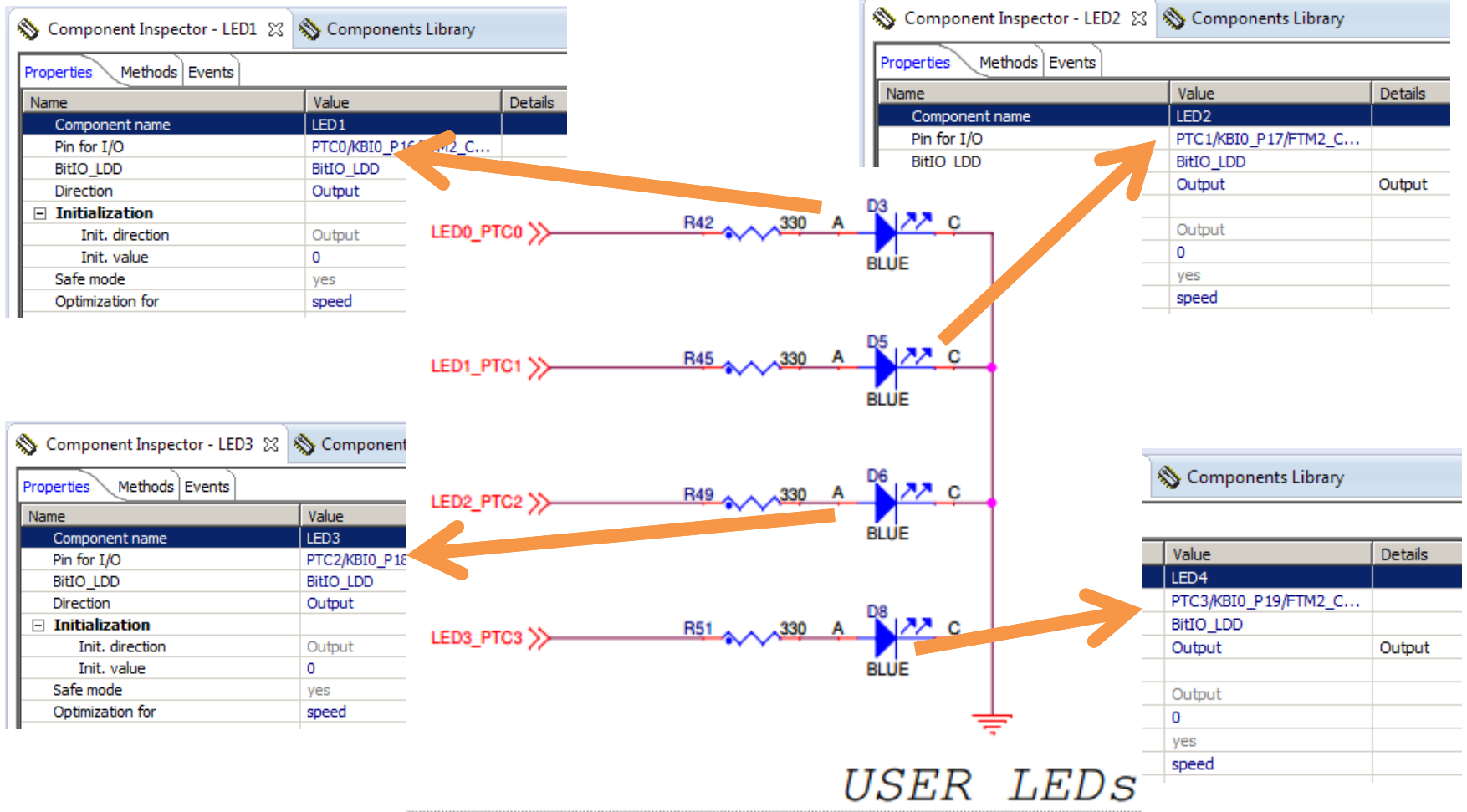
❑用户可通过在组件库中选中要添加的组件，右键→选中“Add to Project”快速添加该组件到用户工程，如下图所示：



bit I/O组件配置



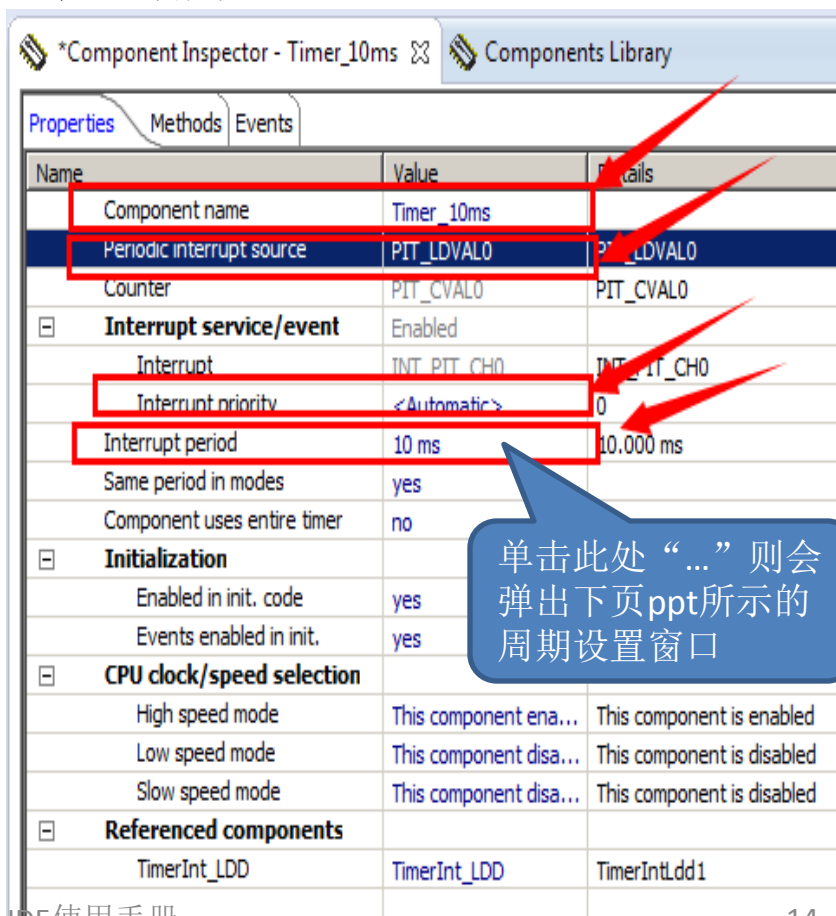
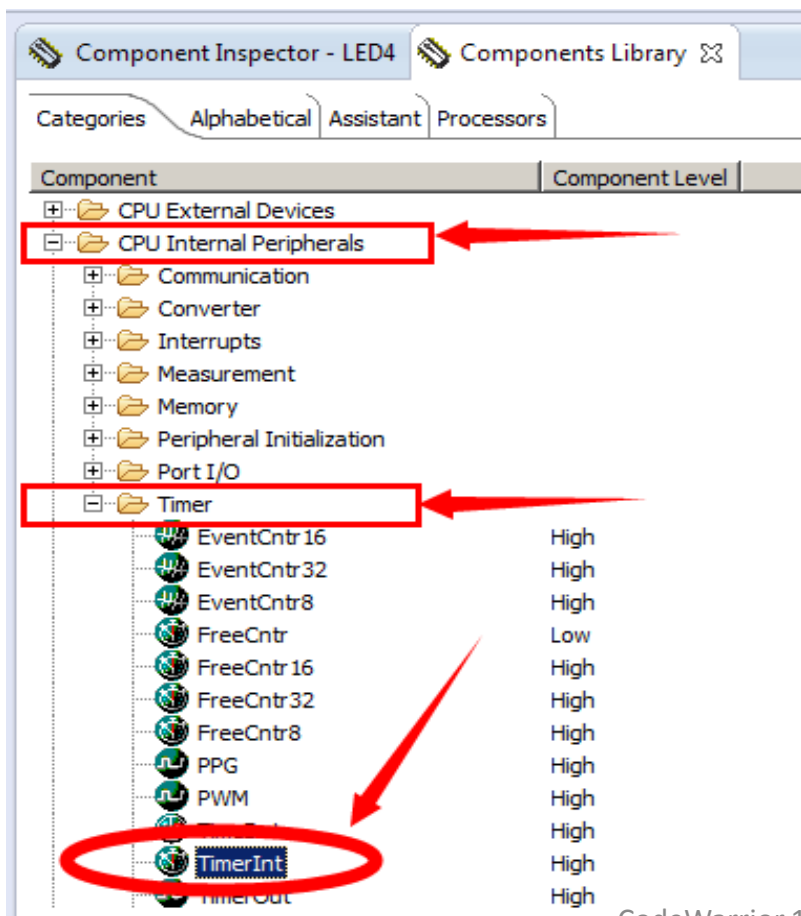
添加单bit I/O组件控制TRK-KEA128板载的4颗LED



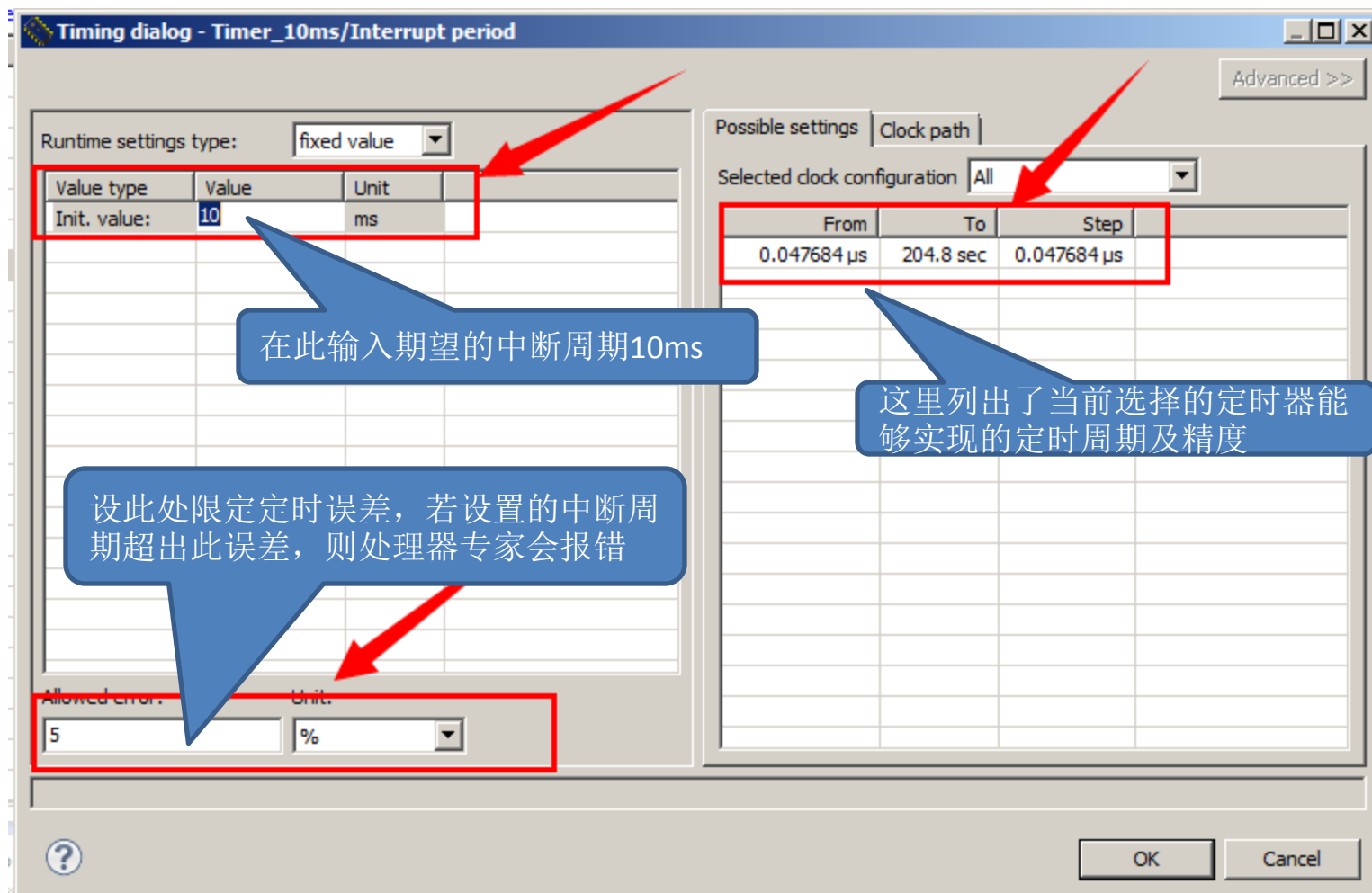
添加和配置定时器中断组件

□选择CPU内部外设→定时器→TimerInt
添加定时器中断组件，如下图所示：

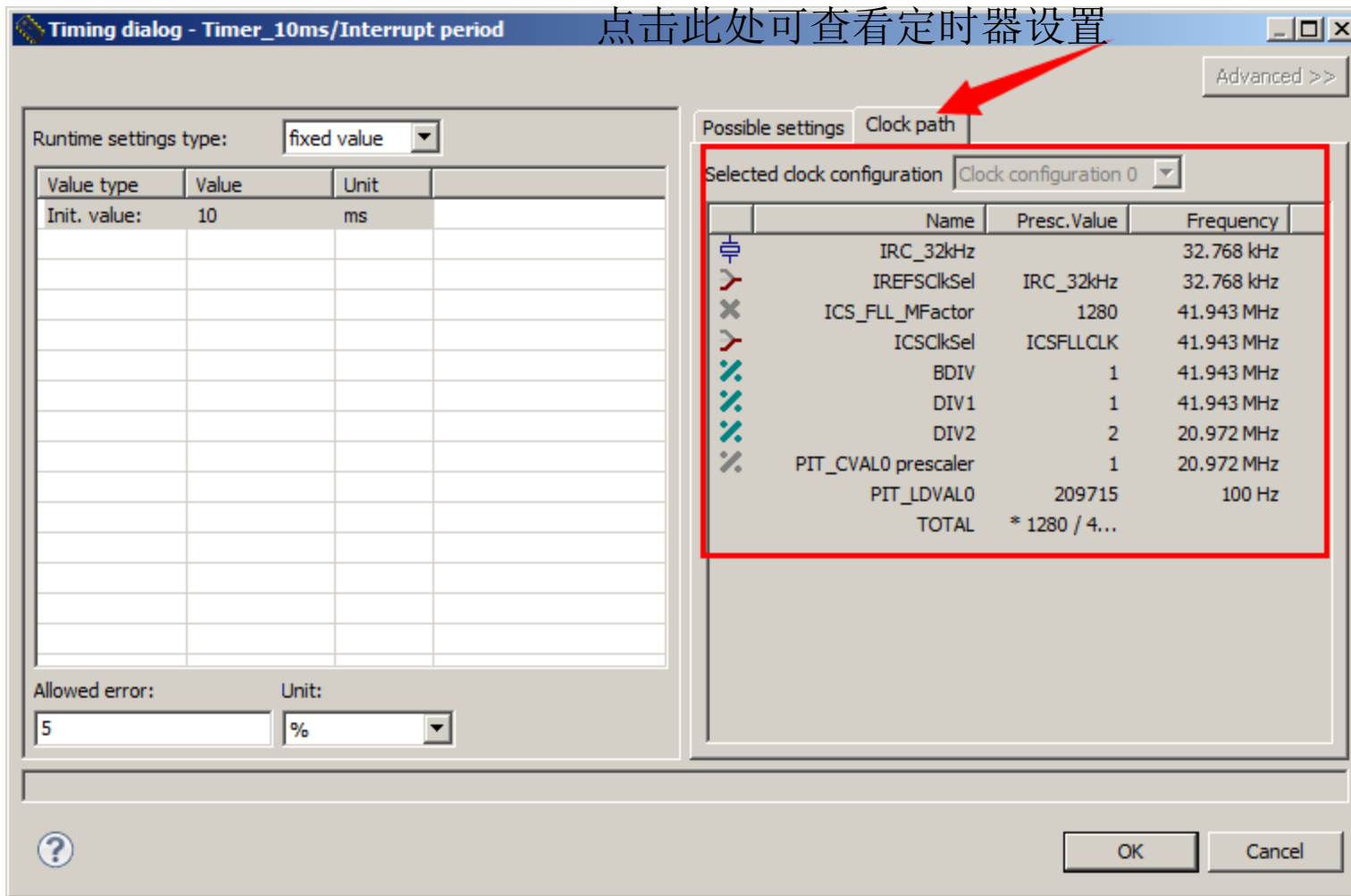
❑ 设置组件名为“Timer_10ms”,选择PIT1作为该定时器硬件中断源,设置中断周期为10ms



设置定时器中断周期为10ms

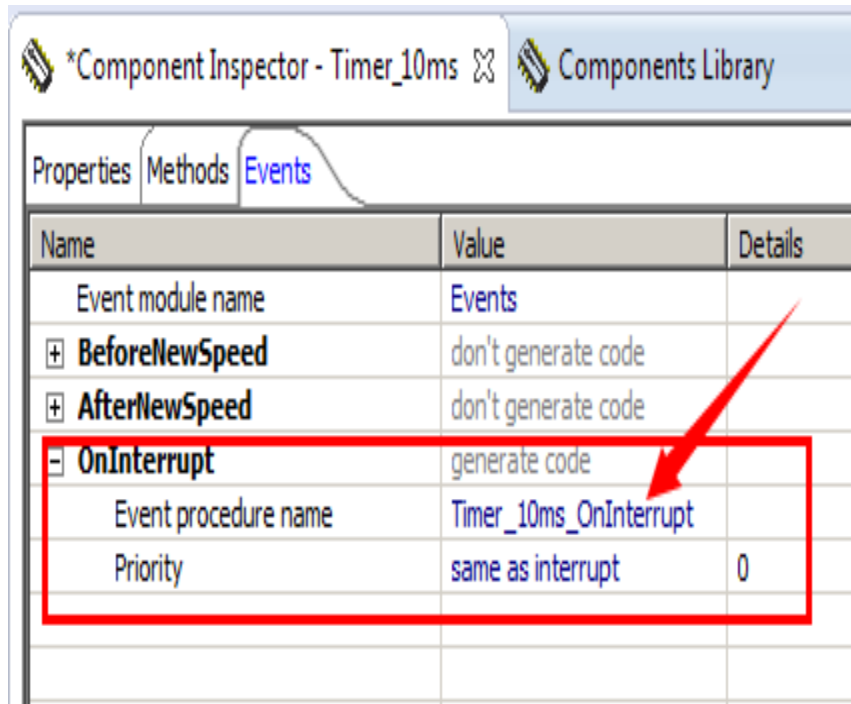


查看定时器设置源及分配设置

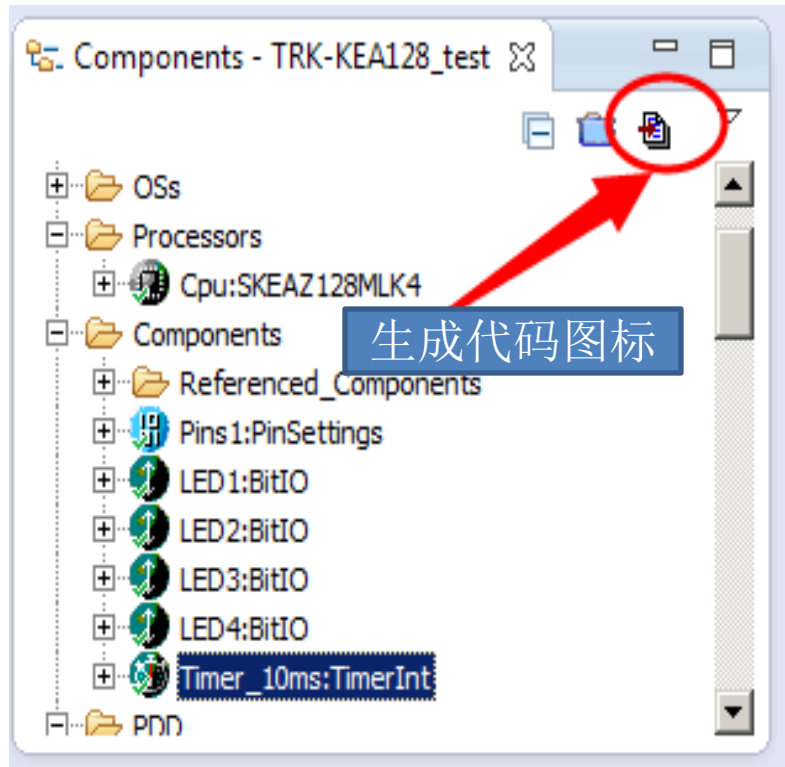


添加和配置定时器中断组件

❑ 在事件（Events）栏，可修改中断处理回调函数名称，其默认如下图所示：

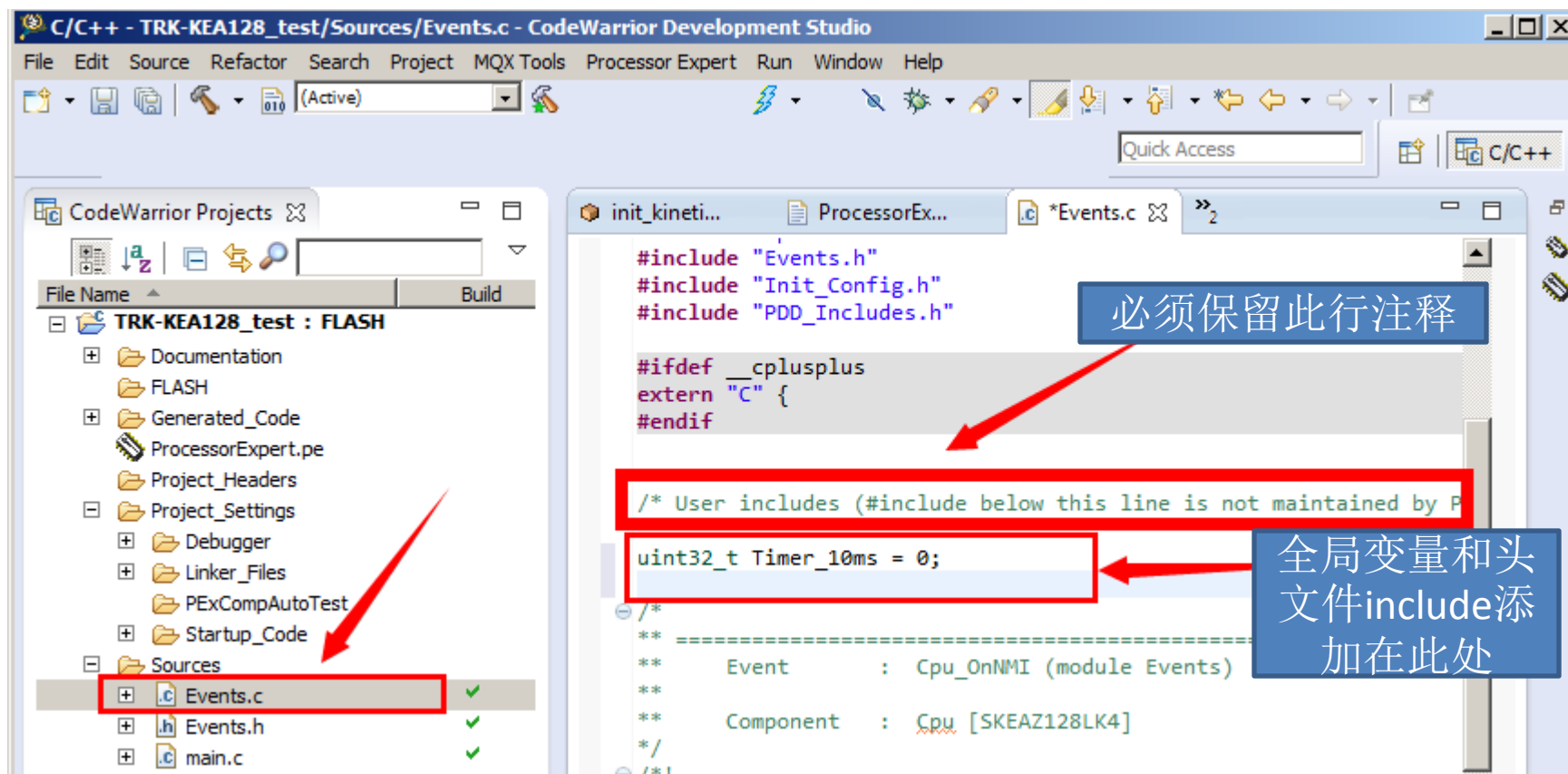


❑ 设置完成后可以看到定时器组件已出现在Component文件夹下，点击右上角的生成代码图标生成代码：



添加和配置定时器中断组件

- 完成以上设置并生成代码之后，选择Event.c添加如下全局变量Timer_10ms；
注意：这里添加全局变量和头文件include必须加在处理器专家指定的位置



添加和配置定时器中断组件

- 最后在中断回调函数中添加中断处理，这里为全局中断计数器加1；
注意：用户的中断处理代码必须加在处理器专家指定的位置



```

}

/*
** =====
**      Event      :  Timer_10ms_OnInterrupt (module Events)
**
**      Component   :  Timer_10ms [TimerInt]
**      Description :
**          When a timer interrupt occurs this event is called (only
**          when the component is enabled - <Enable> and the events are
**          enabled - <EnableEvent>). This event is enabled only if a
**          <interrupt service/event> is enabled.
**      Parameters  :  None
**      Returns     :  Nothing
**      =====
*/

void Timer_10ms_OnInterrupt(void)
{
    /* Write your code here ... */
    Timer_10ms++;
}

```

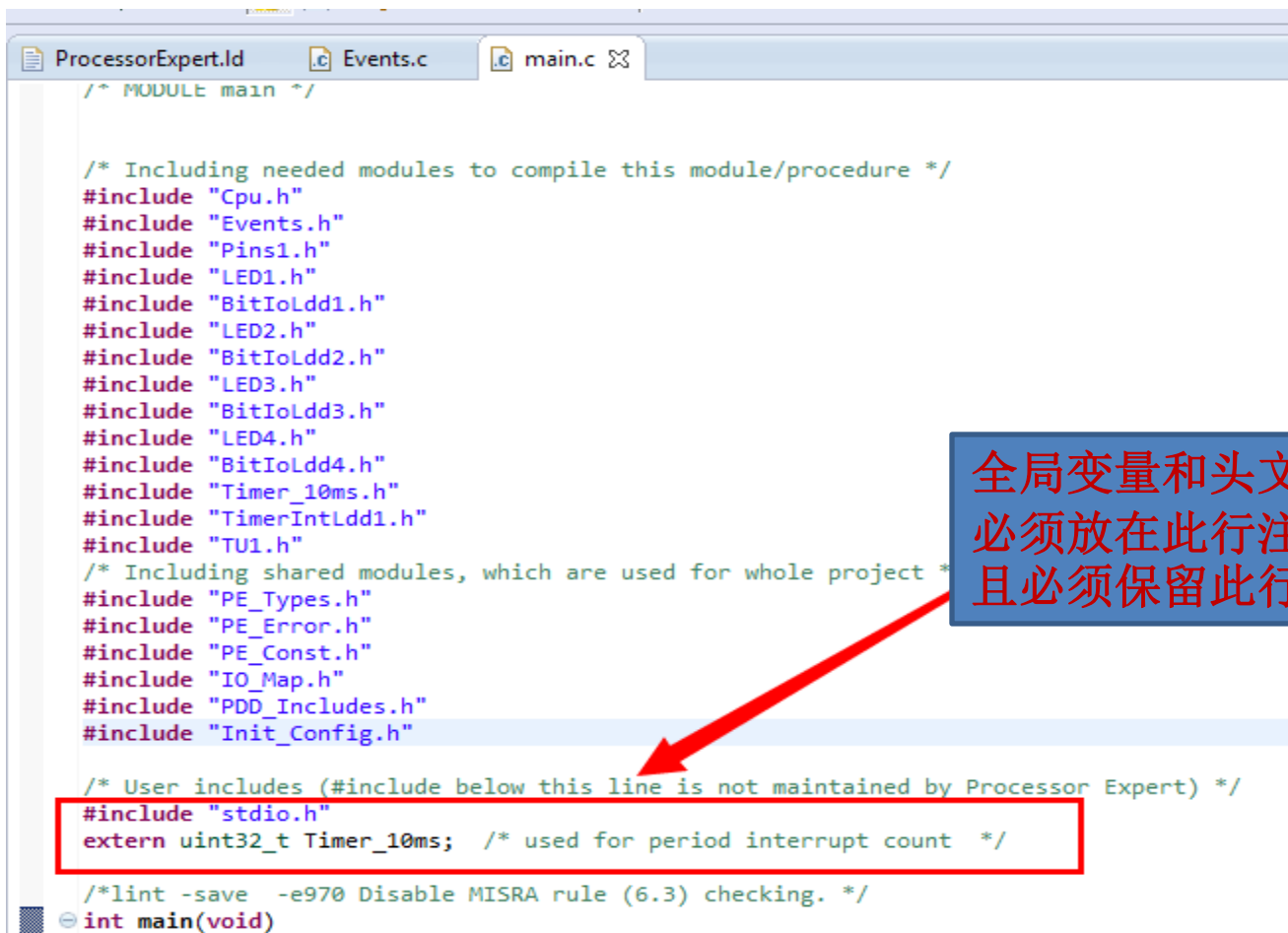
必须保留

在此添加用户代码

在Main()函数添加用户代码

□在main.c中引用全局变量Timer_10ms;

注意：这里添加全局变量和头文件include也必须加在处理器专家指定的位置



```
ProcessorExpert.ld  Events.c  main.c x

/* MODULE main */

/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "Pins1.h"
#include "LED1.h"
#include "BitIoLdd1.h"
#include "LED2.h"
#include "BitIoLdd2.h"
#include "LED3.h"
#include "BitIoLdd3.h"
#include "LED4.h"
#include "BitIoLdd4.h"
#include "Timer_10ms.h"
#include "TimerIntLdd1.h"
#include "TU1.h"
/* Including shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include "PDD_Includes.h"
#include "Init_Config.h"

/* User includes (#include below this line is not maintained by Processor Expert) */
#include "stdio.h"
extern uint32_t Timer_10ms; /* used for period interrupt count */

/*lint -save -e970 Disable MISRA rule (6.3) checking. */
int main(void)
```

全局变量和头文件include
必须放在此行注释之后并
且必须保留此行注释

添加main()函数主体

- 在main()函数中如下代码，实现对TRK-KEA128 demo板板载4可LED的控制；
注意：这里添加全局变量和头文件include也必须加在处理器专家指定的位置

```
ProcessorExpert.ld  Events.c  main.c
int main(void)
/*lint -restore Enable MISRA rule (6.3) checking. */
{
    /* Write your local variable definition here */
    uint8_t Toggled_LED = 0; /* used to record the toggled LED index */

    /** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! */
    PE_low_level_init();
    /** End of Processor Expert internal initialization.
    /* Write your code here */
    /* For example: for(;;) { } */

    printf("hello the world~!\n");

    while(1)
    {
        if((Timer_10ms%1)==0) /* toggle LED1 every 10ms */
        {
            LED1_NegVal(); Toggled_LED = 1;
        }
        if((Timer_10ms%2)==0) /* toggle LED2 every 20ms */
        {
            LED2_NegVal(); Toggled_LED = 2;
        }
        if((Timer_10ms%3)==0) /* toggle LED3 every 30ms */
        {
            LED3_NegVal(); Toggled_LED = 3;
        }
        if((Timer_10ms%4)==0) /* toggle LED4 every 40ms */
        {
            LED4_NegVal(); Toggled_LED = 4;
        }

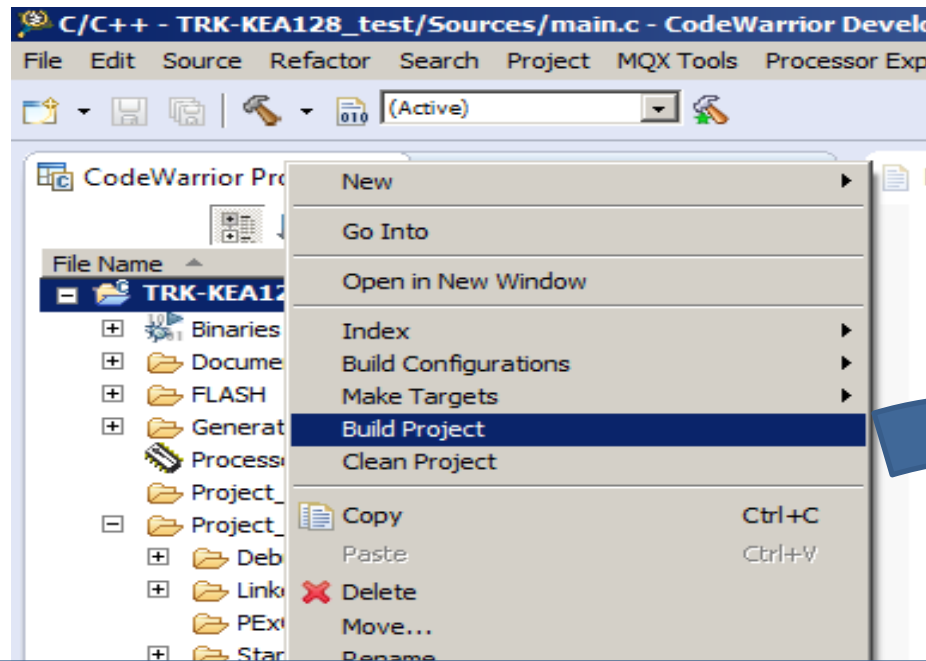
        /* print Timer_10ms value */
        printf("Timer_10ms = %d\n",Timer_10ms);
    }
}
```

添加局部变量Toggled_LED


添加LED控制以及控制台console打印输出定时器中断计数器值：
每10ms/20ms/30ms/40ms
分别toggle一下
LED1/LED2/LED3/LED4,每次
循环打印一次Timer_10ms
计数值

编译工程

□选中该工程，右键→**Build Project**，对其进行编译，编译结果（错误和警告）会在问题窗口列出，单击可快速跳转到引起该错误或警告的C代码位置：



一个工程编译之后，错误error为0才会生成最终的可执行文件，才能进行下载和调试；警告warning视情况而定，可以忽略不管

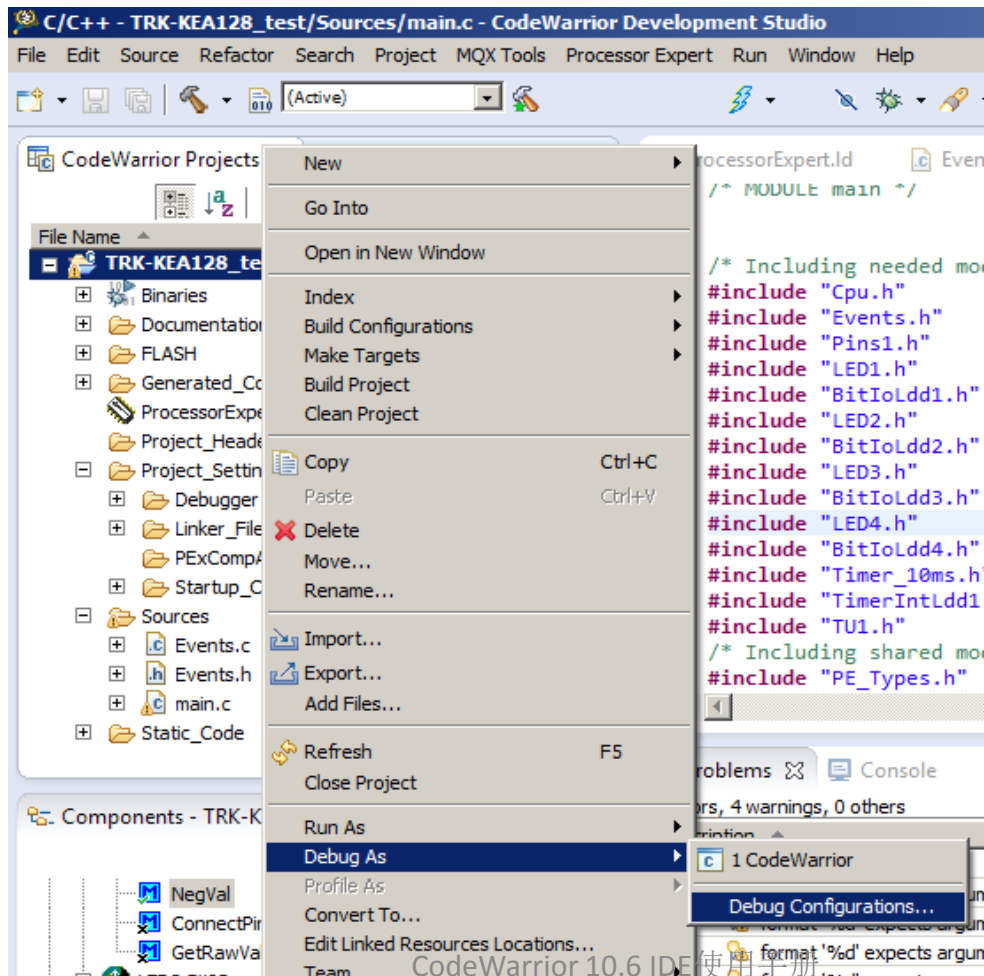


0 errors, 4 warnings, 0 others

Description	Resource	Path	Location	Type
Warnings (4 items)				
format '%d' expects argument of type 'int', but argument 2 has type 'uint32_t' [-Wformat]	main.c	/TRK-KEA128_te...	line 73	C/C++ Pro...
format '%d' expects argument of type 'int', but argument 2 has type 'uint32_t' [-Wformat]	main.c	/TRK-KEA128_te...	line 77	C/C++ Pro...
format '%d' expects argument of type 'int', but argument 2 has type 'uint32_t' [-Wformat]	main.c	/TRK-KEA128_te...	line 81	C/C++ Pro...
format '%d' expects argument of type 'int', but argument 2 has type 'uint32_t' [-Wformat]	main.c	/TRK-KEA128_te...	line 85	C/C++ Pro...

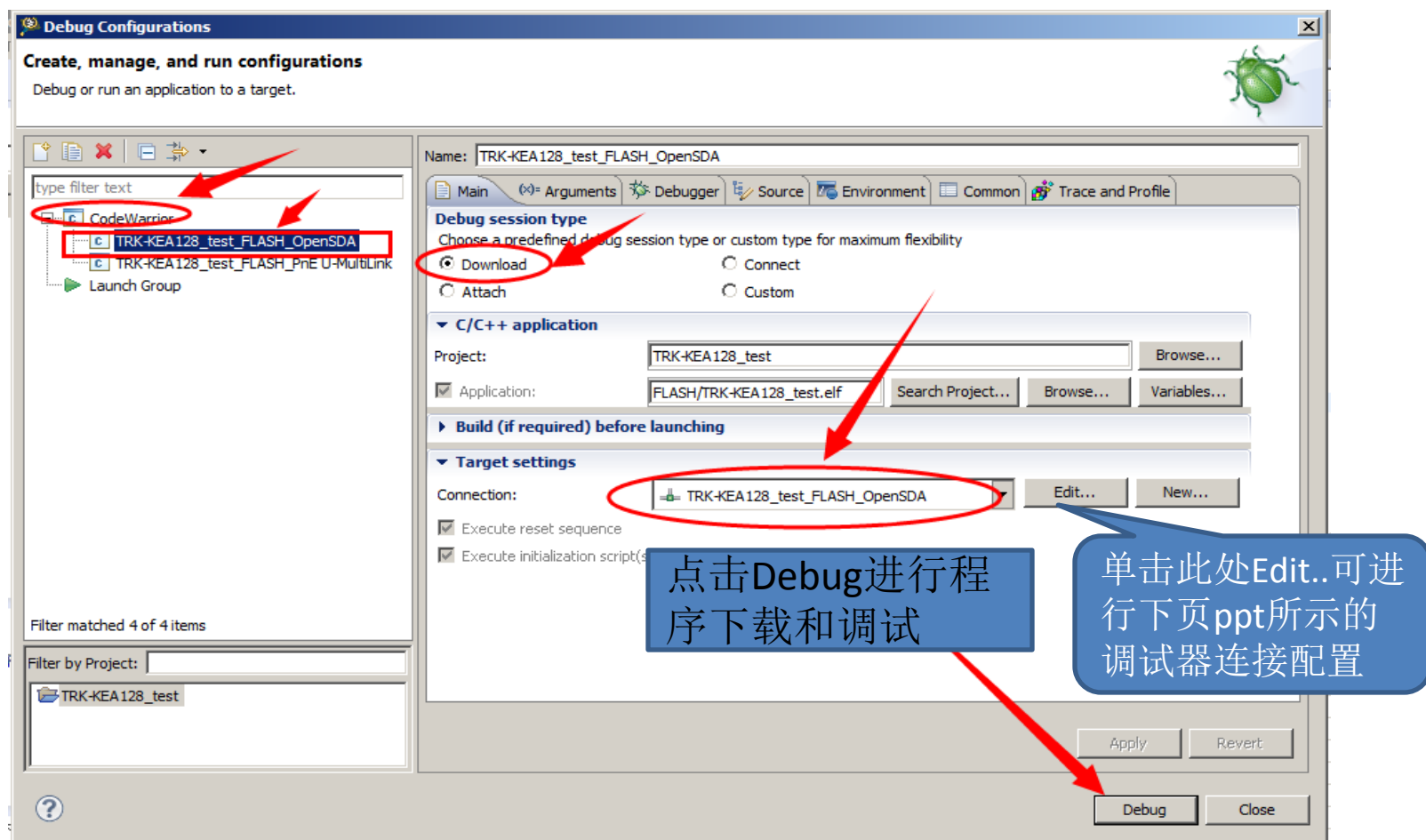
启动调试

□选中该工程，右键→**Debug As**→**Debug Configuration**，启动调试配置界面
每个工程编译之后第一次调试都要进行这样的配置



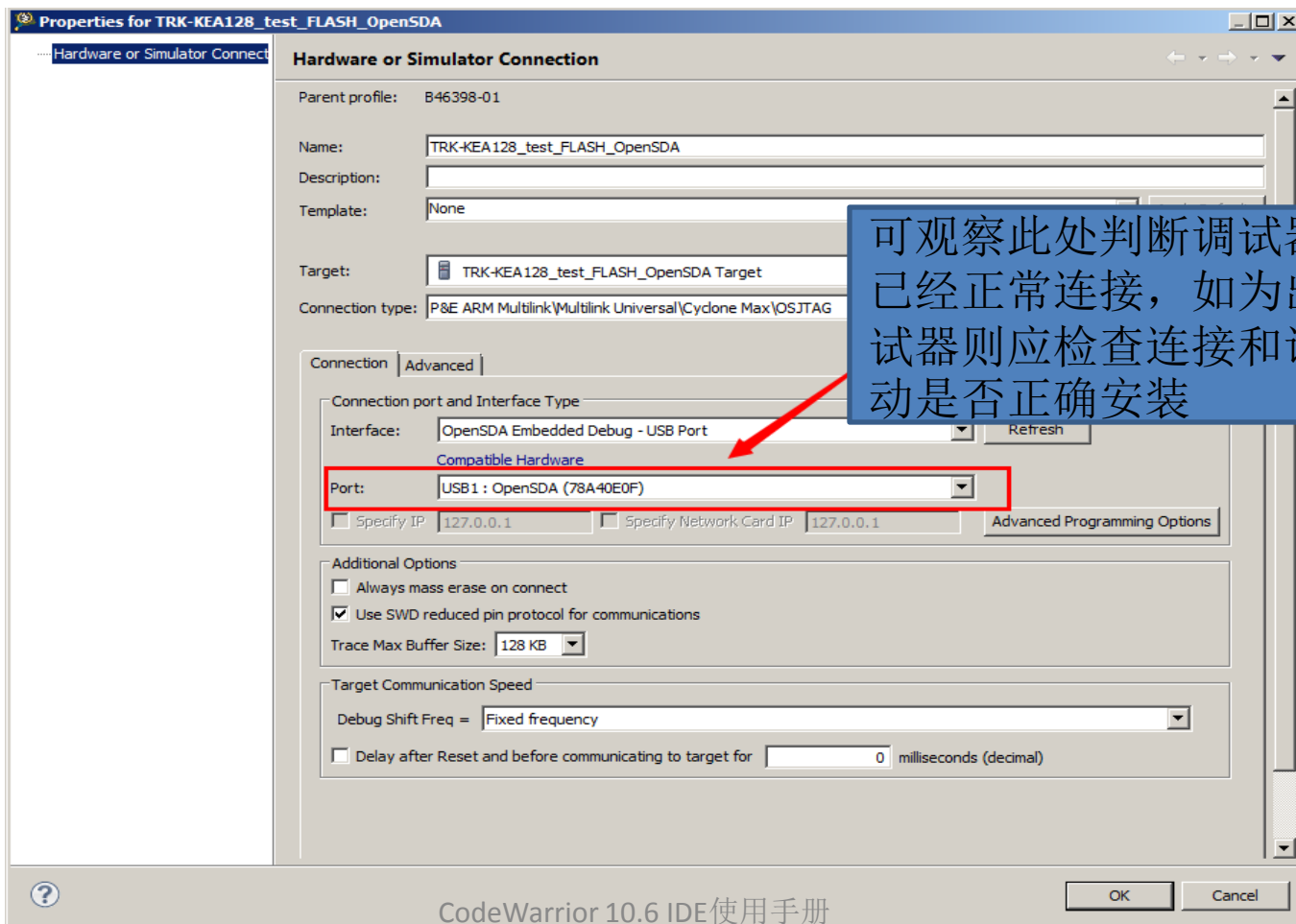
调试配置

❑ 在弹出的调试配置窗口中，点击**CodeWarrior**，选择**TRK-KEA128_test_FLASH_OpenSDA**，对其进行如下配置



调试器连接设置

□这里选择TRK-KEA128板载OpenSDA作为本 Demo的调试器，查看并确保TRK-KEA128板已经正常连接到电脑



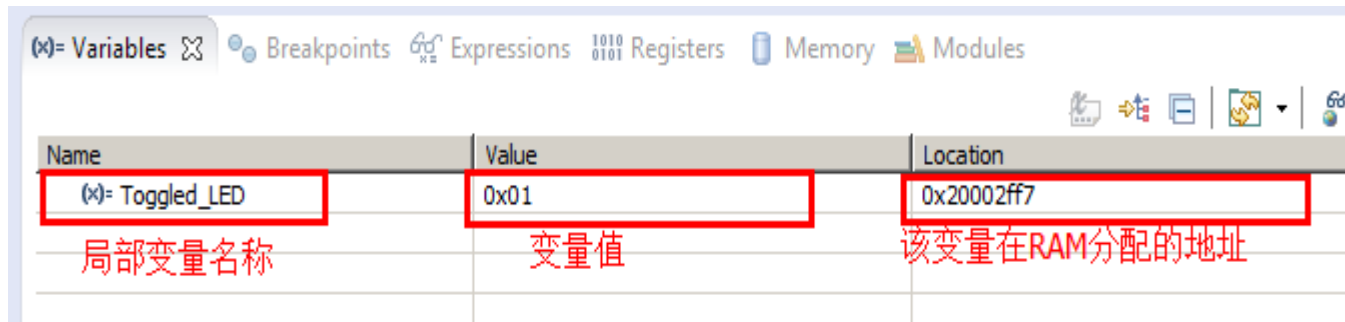
调试界面介绍

The screenshot shows the CodeWarrior 10.6 IDE interface with several callouts explaining its components:

- 全速运行** (Full Speed Run): A button in the top toolbar.
- 暂停** (Pause): A button in the top toolbar.
- 单击选中此处, 则进入汇编语言单步执行状态** (Click here to enter assembly language single-step execution state): A callout pointing to a specific button in the top toolbar.
- 复位** (Reset): A button in the top toolbar.
- 单步执行, 跳出函数** (Single-step execution, exit function): A button in the top toolbar.
- 单步执行, 跳过函数** (Single-step execution, skip function): A button in the top toolbar.
- 单步执行, 进入函数** (Single-step execution, enter function): A button in the top toolbar.
- 进程窗口: 可查看当前函数的地址及函数调用和执行关系** (Process Window: View current function address and call/execution relationship): A callout pointing to the Process Window on the left.
- 变量、断点, 表达式、寄存器以及存储器查看窗口, 将在下面几页ppt进行详细介绍** (Variables, breakpoints, expressions, registers, and memory viewing windows, will be introduced in detail in the next few slides): A callout pointing to the Breakpoints, Expressions, Registers, and Memory windows on the right.
- C语言代码窗口: 可查看当前执行C代码函数** (C Language Code Window: View current executing C code function): A callout pointing to the C Code Window on the left.
- 汇编语言代码窗口: 可查看当前执行C代码函数对应的汇编代码** (Assembly Language Code Window: View assembly code corresponding to the current executing C code function): A callout pointing to the Disassembly Window on the right.
- TRK-KEA128 Demo板的实际运行效果, 四颗板载LED按照程序控制进行10ms/20ms/30ms/40ms的周期闪烁** (Actual running effect of the TRK-KEA128 Demo board, four on-board LEDs controlled by the program to flash in 10ms/20ms/30ms/40ms cycles): A callout pointing to the image of the TRK-KEA128 board on the bottom right.
- 控制台信息, 可在可以看到程序printf函数打印输出的定时器中断计数值** (Console information, can see the timer interrupt counting value printed by the program's printf function): A callout pointing to the Console Window at the bottom.

查看变量和断点

❑ CodeWarrior 会将当前执行函数中的所有局部自动添加到变量查看窗口，其中的变量随程序的执行而不断改变和更新

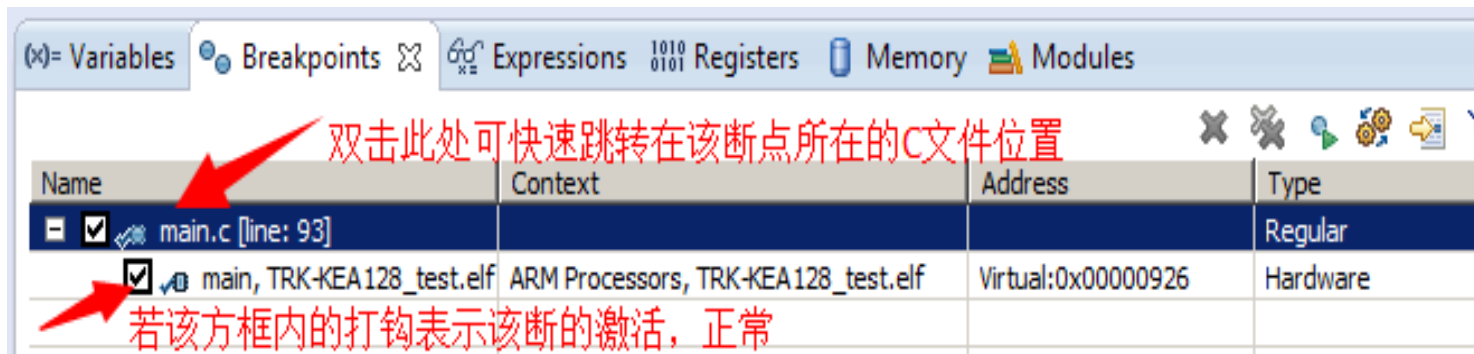


Name	Value	Location
(*)= Toggled_LED	0x01	0x20002ff7

局部变量名称 变量值 该变量在RAM分配的地址

❑ 在断点查看窗口能够查看到当前工程所有的断点

❑ 注意：KEA MCU 仅支持 2 个硬件断点，如果设置多于 2 个断点，则无效



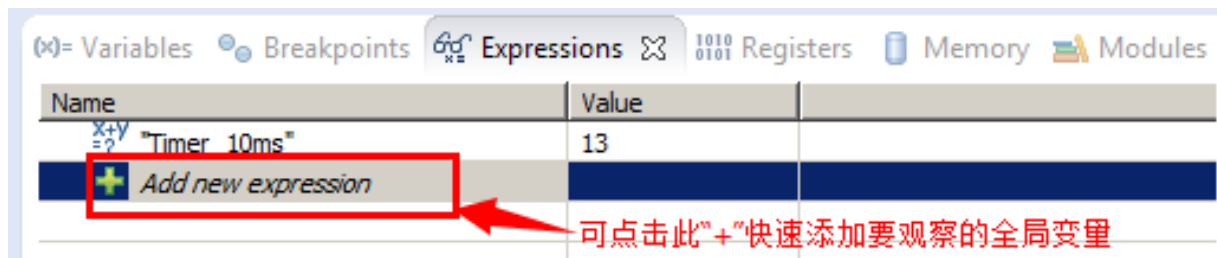
Name	Context	Address	Type
<input checked="" type="checkbox"/> main.c [line: 93]			Regular
<input checked="" type="checkbox"/> main, TRK-KEA128_test.elf	ARM Processors, TRK-KEA128_test.elf	Virtual:0x00000926	Hardware

双击此处可快速跳转在该断点所在的C文件位置

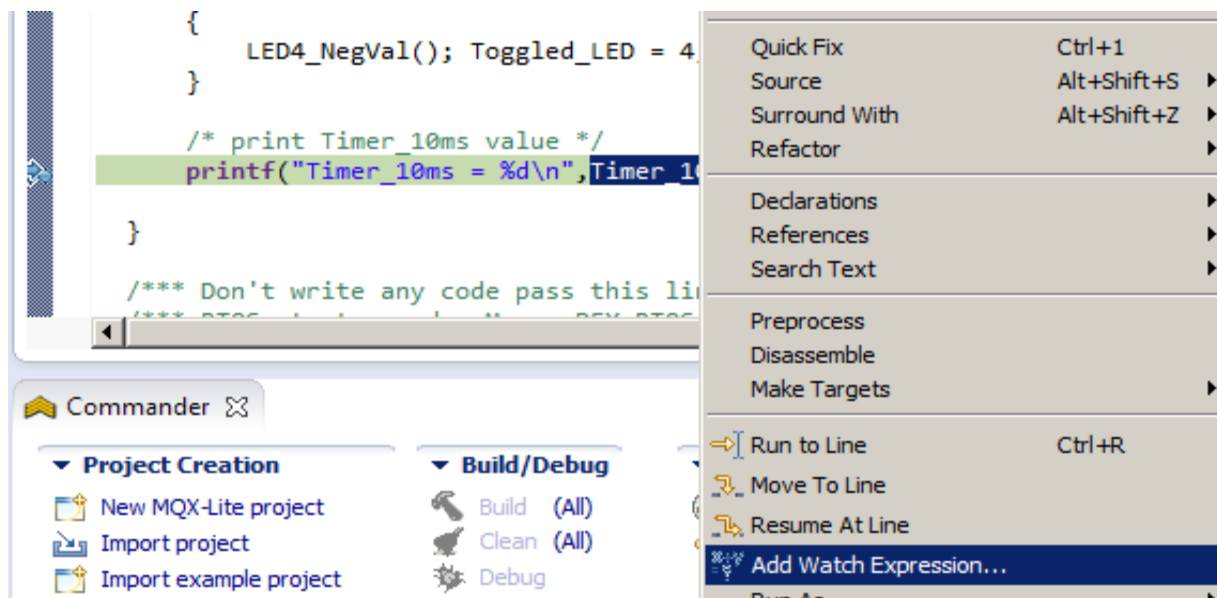
若该方框内的打钩表示该断的激活，正常

查看全局变量/表达式

□可以通过点击“+”添加要查看的全局变量或表达式：

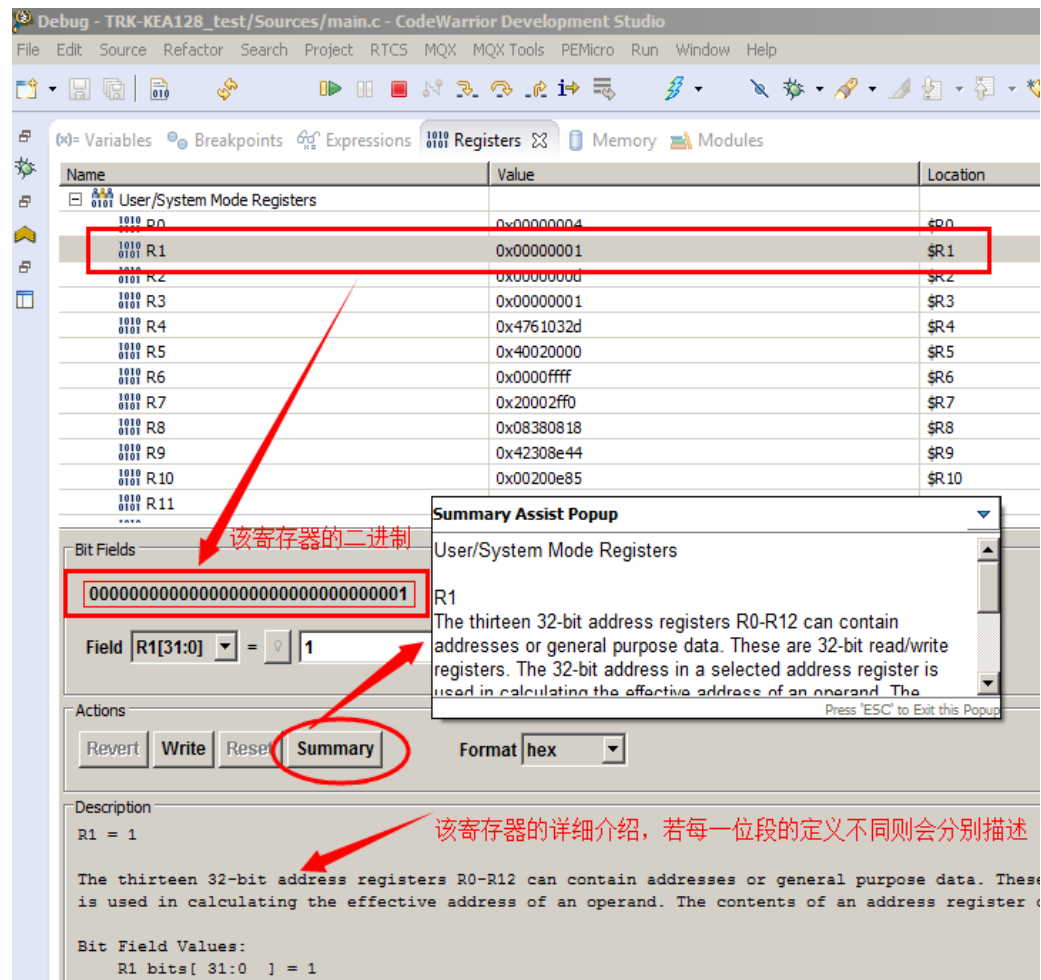
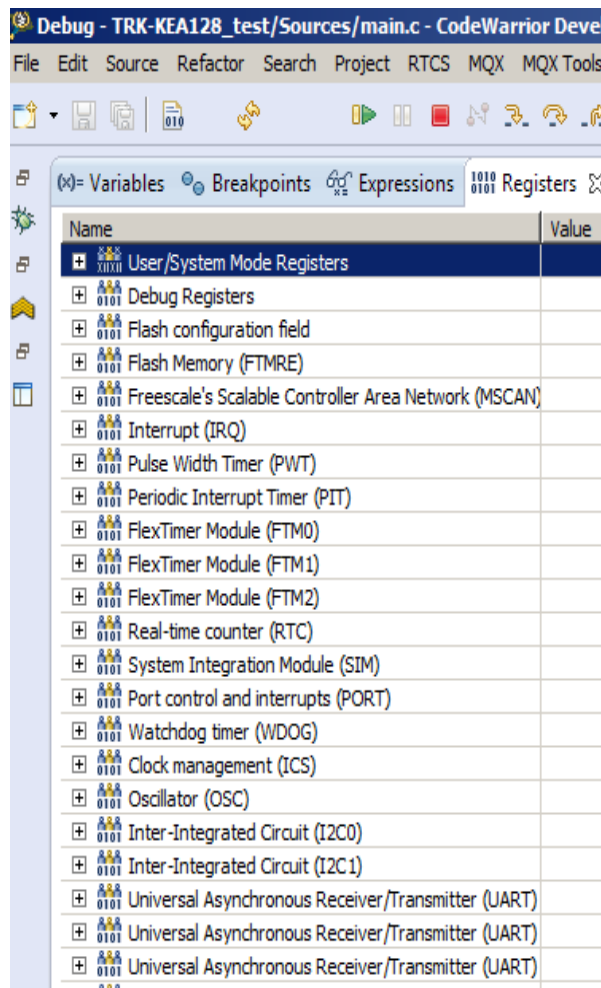


□也可以在C代码窗口中选择要查看的全局变量或表达式右键快捷添加：

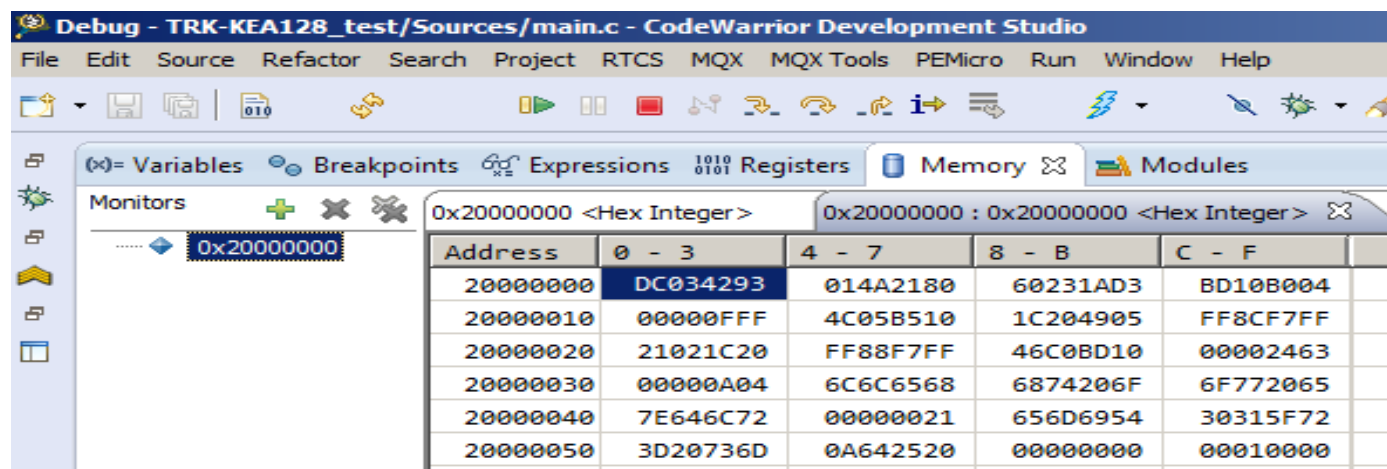
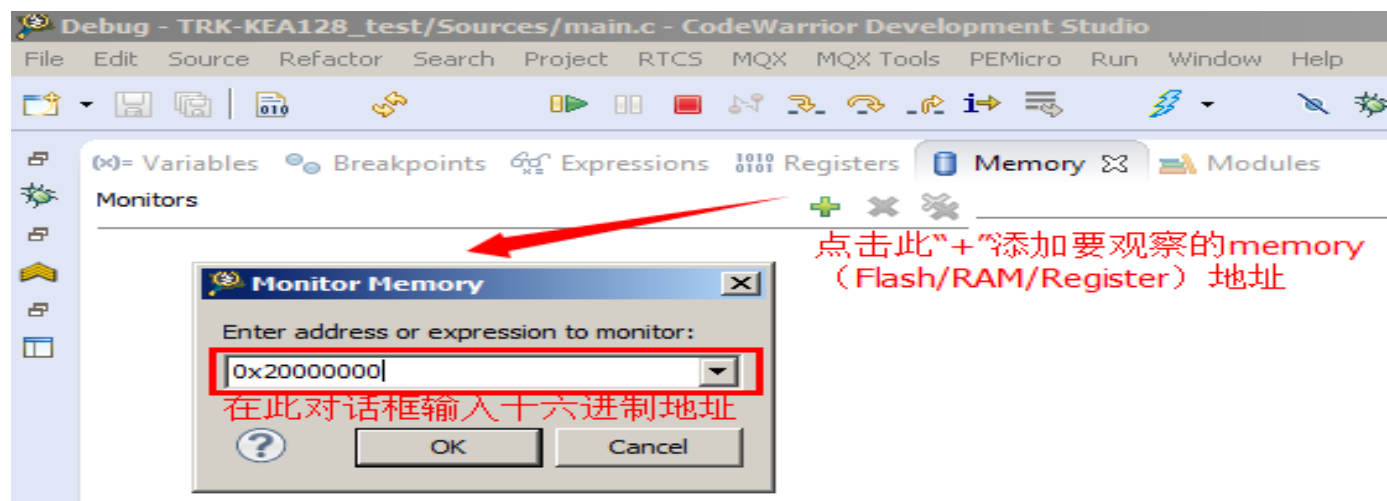


查看MCU内核及外设寄存器

□通过寄存器查看窗口可以查看CPU内核及MCU外设的所有寄存器值



观察指定地址的memory或变量

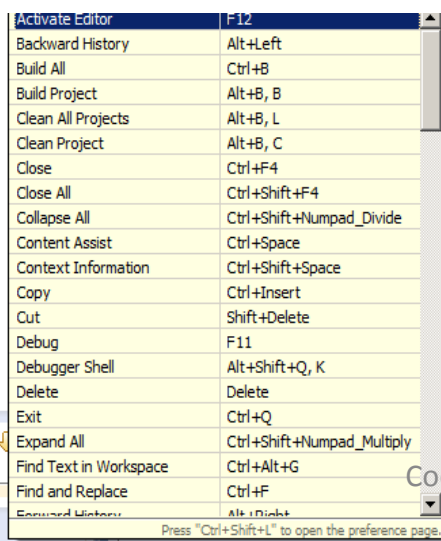


一些使用技巧

□ 快捷键

- 跳转到函数/变量/宏定义: F3
- C代码行注释/反注释: Ctrl + /
- C代码段注释: Ctrl + Shift + /
- 快速跳转至上一编辑处: Alt + Left
- 快速跳转至下一编辑处: Alt + Right
- 在资源管理器中快速打开CodeWarrior工程中的某一文件: 选中该文件→右键→选择最后一个选项“Show In Windows Explorer”

□ 快速浏览CodeWarrior的快捷键: Ctrl + Shift + L, 如下图所示:



Activate Editor	F12
Backward History	Alt+Left
Build All	Ctrl+B
Build Project	Alt+B, B
Clean All Projects	Alt+B, L
Clean Project	Alt+B, C
Close	Ctrl+F4
Close All	Ctrl+Shift+F4
Collapse All	Ctrl+Shift+Numpad_Divide
Content Assist	Ctrl+Space
Context Information	Ctrl+Shift+Space
Copy	Ctrl+Insert
Cut	Shift+Delete
Debug	F11
Debugger Shell	Alt+Shift+Q, K
Delete	Delete
Exit	Ctrl+Q
Expand All	Ctrl+Shift+Numpad_Multiply
Find Text in Workspace	Ctrl+Alt+G
Find and Replace	Ctrl+F
Source Window	Alt+Right

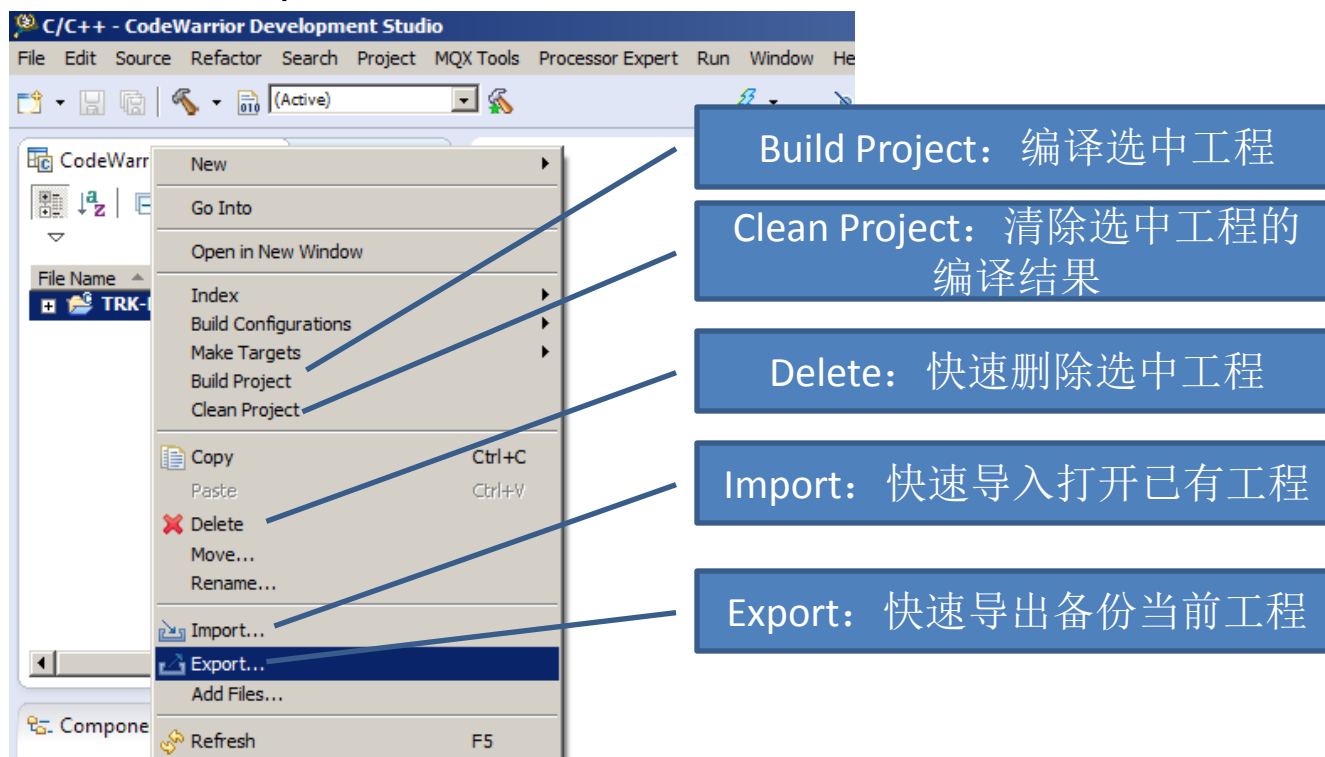
Press "Ctrl+Shift+L" to open the preference page.

2. CodeWarrior 10.6强大的工程管理功能介绍

- 快速导入导出CodeWarrior工程
- 快速打开关闭CodeWarrior工程
- CodeWarrior工作空间视窗管理
- CodeWarrior本地/在线使用帮助

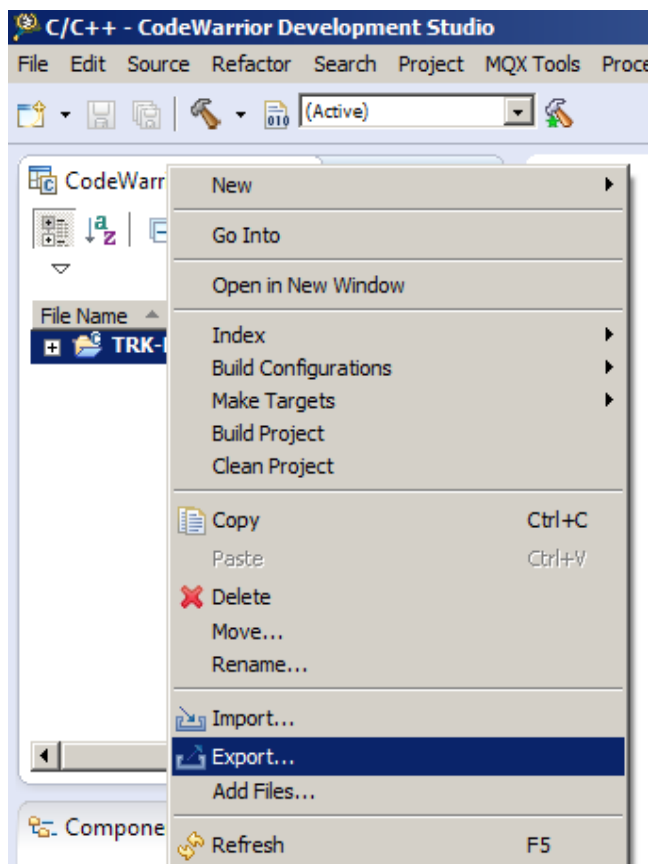
快速导入导出CodeWarrior工程

- ❑ CodeWarrior 10.6集成了强大的工程导入导出功能，但区别于经典版本的CodeWarrior（如5.9、6.3和2.10），其使用导入（Import）功能来打开工程，使用导出（Export）功能来打包备份工程；

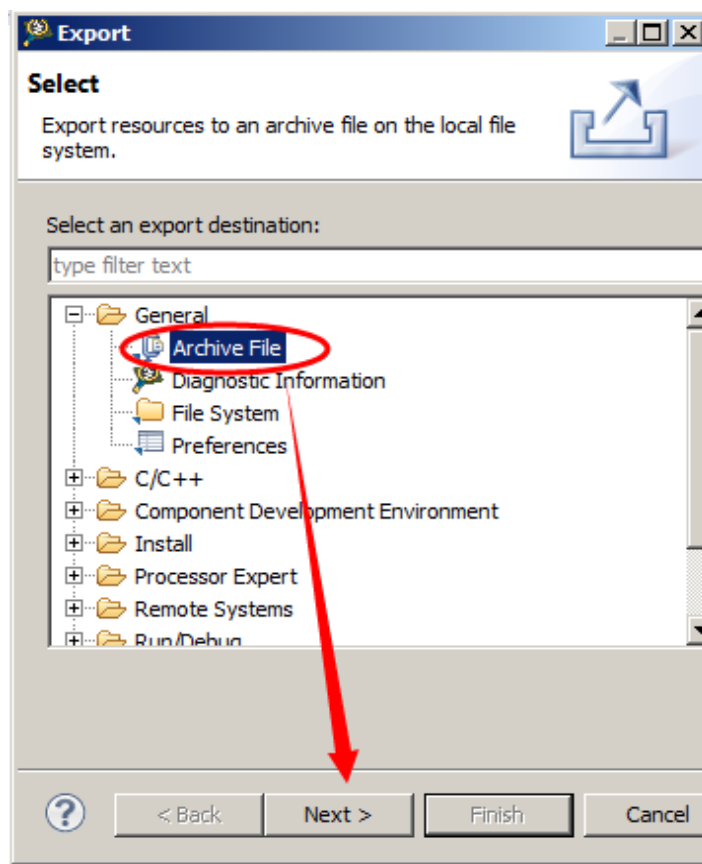


快速导出CodeWarrior工程

- ❑ 选中要导出的工程，右键→导出（Export）



- ❑ 选择General→Archive File(压缩文件)→Next

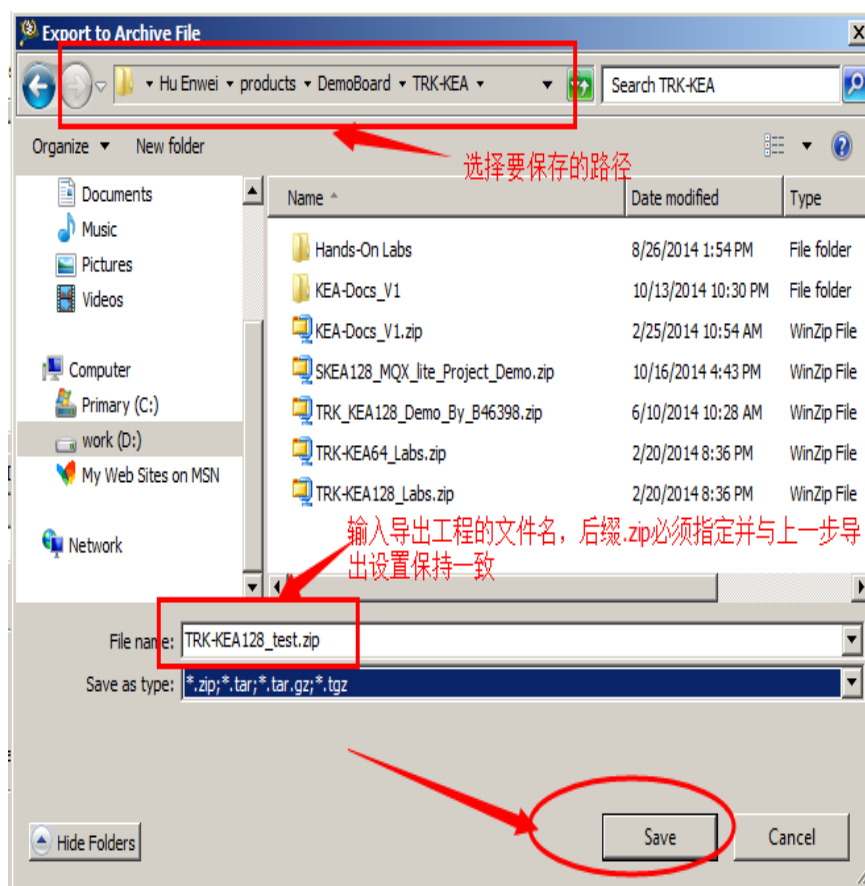


快速导出CodeWarrior工程

- ❑ 选择要导出工程的内容及导出选项设置

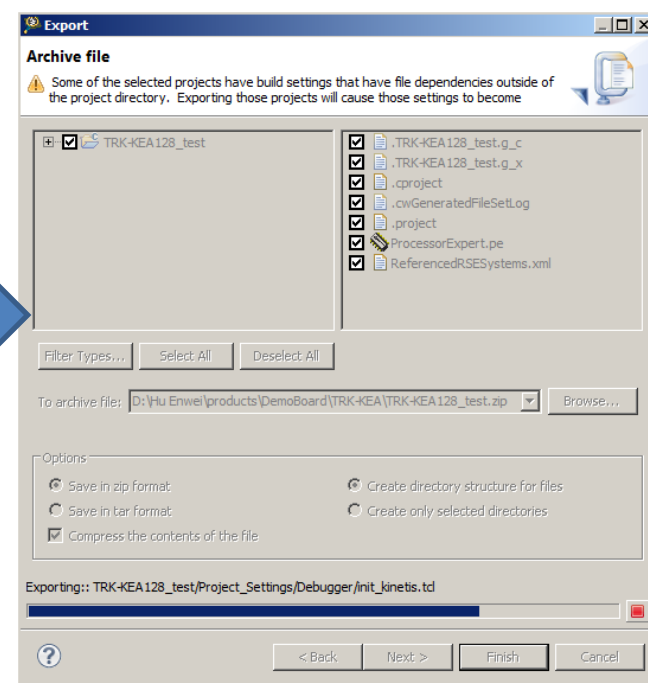
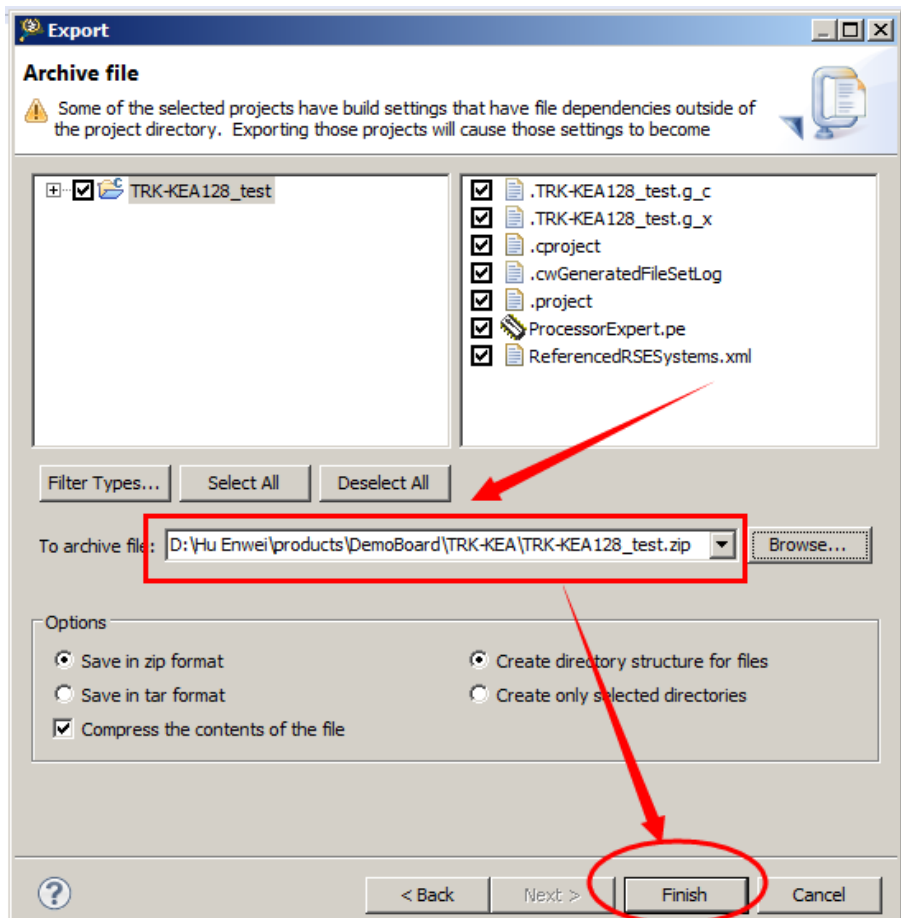


- ❑ 选择要导出工程的存储位置及压缩包名称
注意：文件后缀名一定得与上一步设置的导出压缩包格式一致



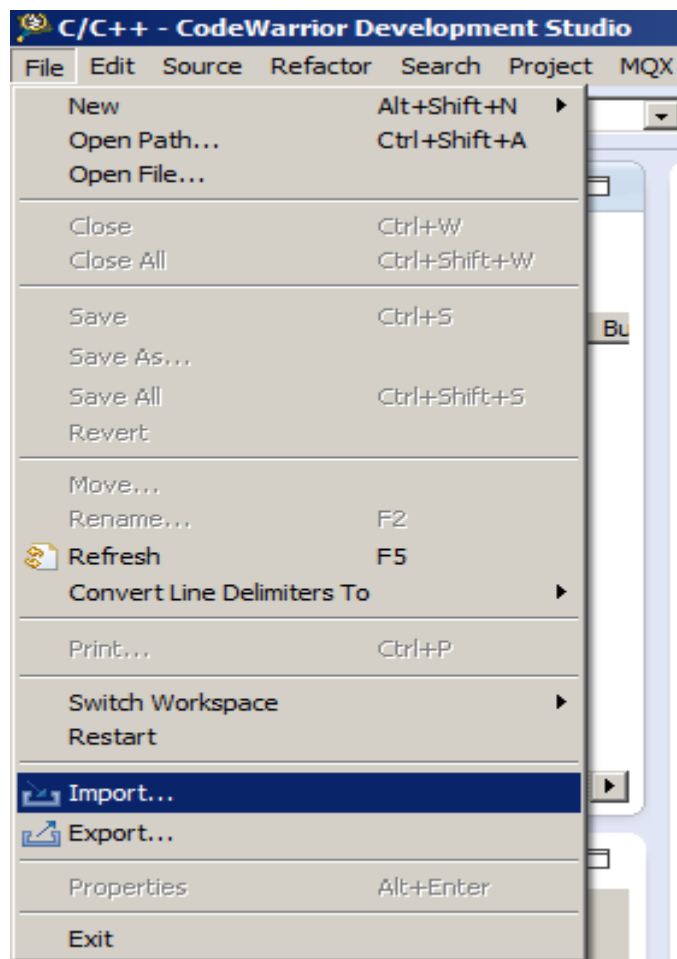
快速导出CodeWarrior工程

❑ 最后Finish完成工程导出：

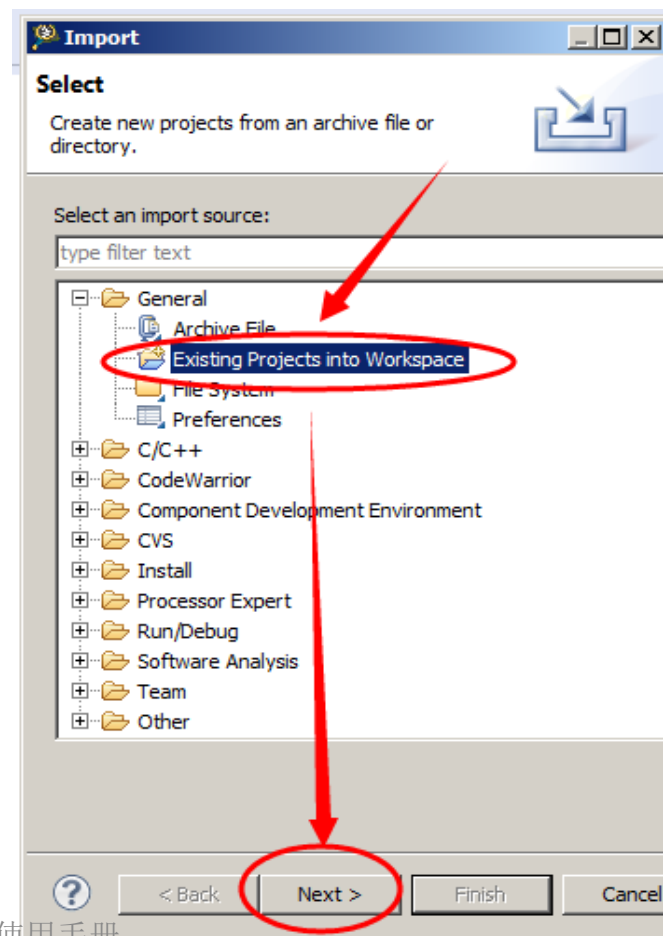


快速导入CodeWarrior工程

❑ File→Import,打开工程导入向导

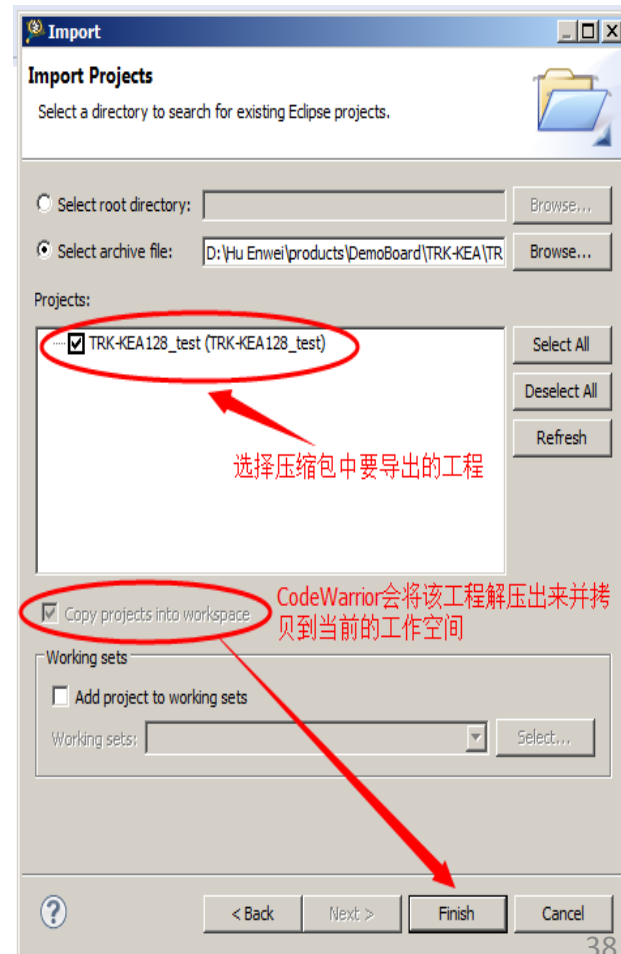
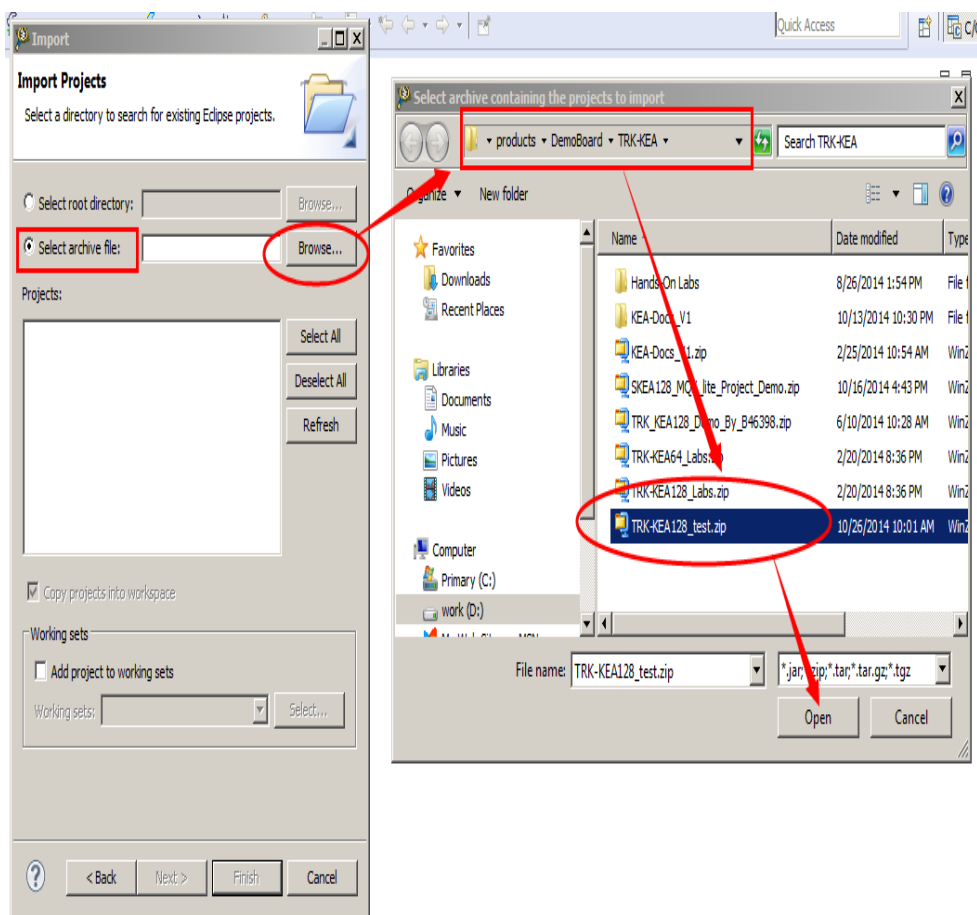


❑ 选择→Existing projects Into Workspace
导入已有工程到工作空间



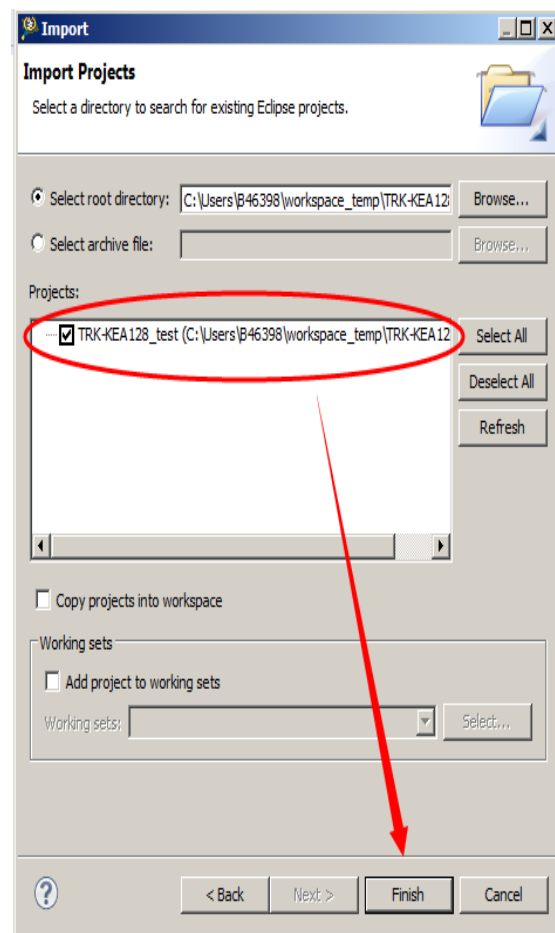
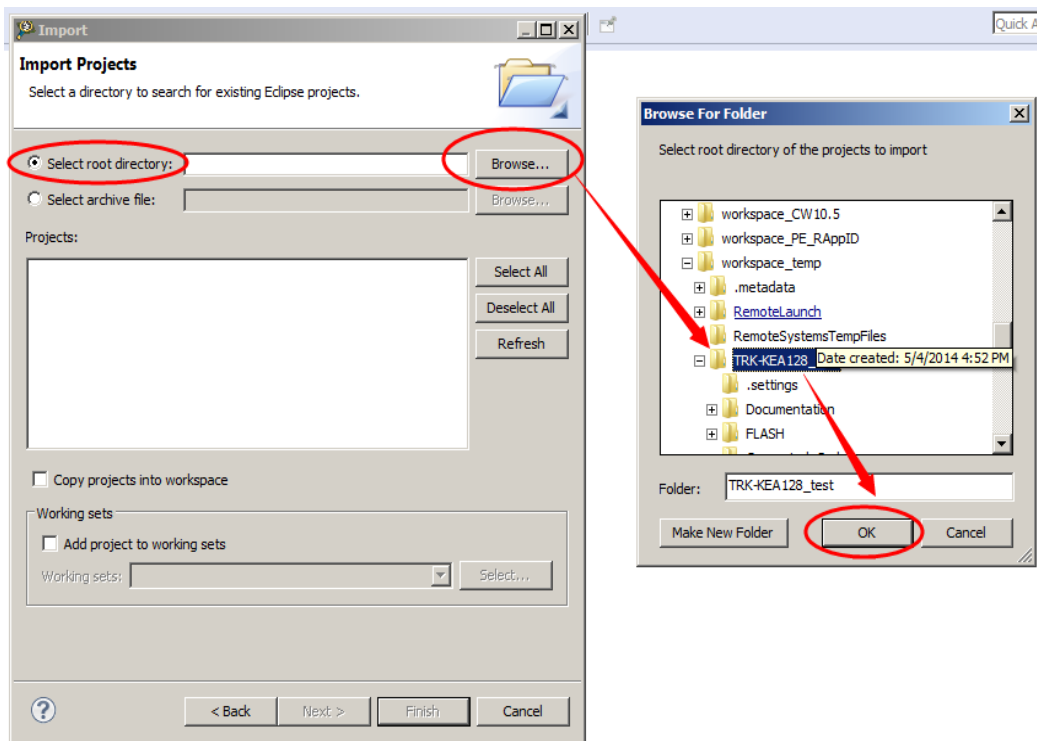
快速导入CodeWarrior工程（压缩包工程）

- ❑ CodeWarrior 10.6能够自动识别压缩包中的CodeWarrior工程并将其拷贝到当前工作空间



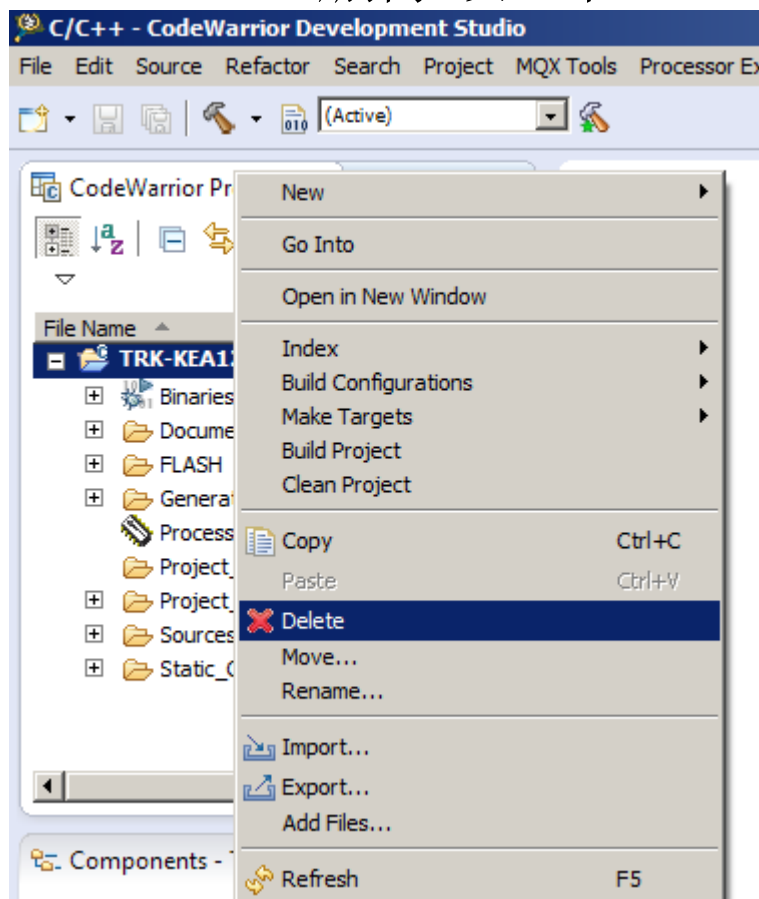
快速导入CodeWarrior工程（解压后的工程）

- ❑ CodeWarrior 10.6也能够自动识别已经解压的CodeWarrior工程并将其拷贝到当前工作空间

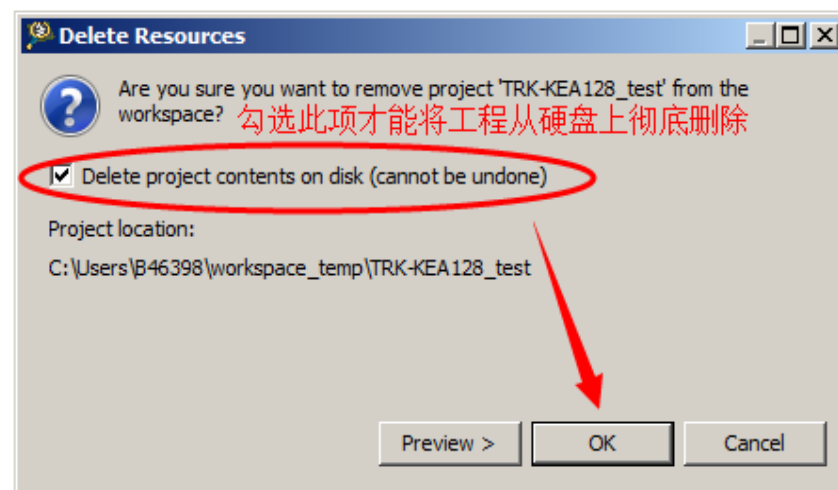


删除工作空间中的CodeWarrior工程

- ❑ 在当前工作空间选择要删除的CodeWarrior工程，右键
→Delete删除该工程

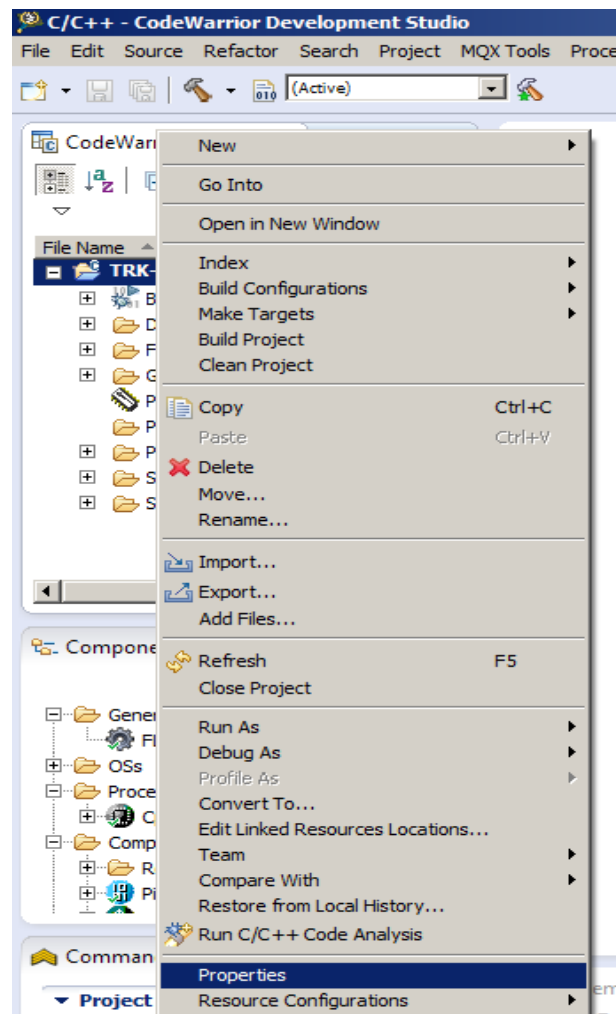


注意：每个工程在工作空间中都对应有一个与工程同名文件夹保存，若一个工作空间已经存在一个**CodeWarrior**工程了，则不能导入同名的另外一个工程，可提供彻底删除来解决这样的问题



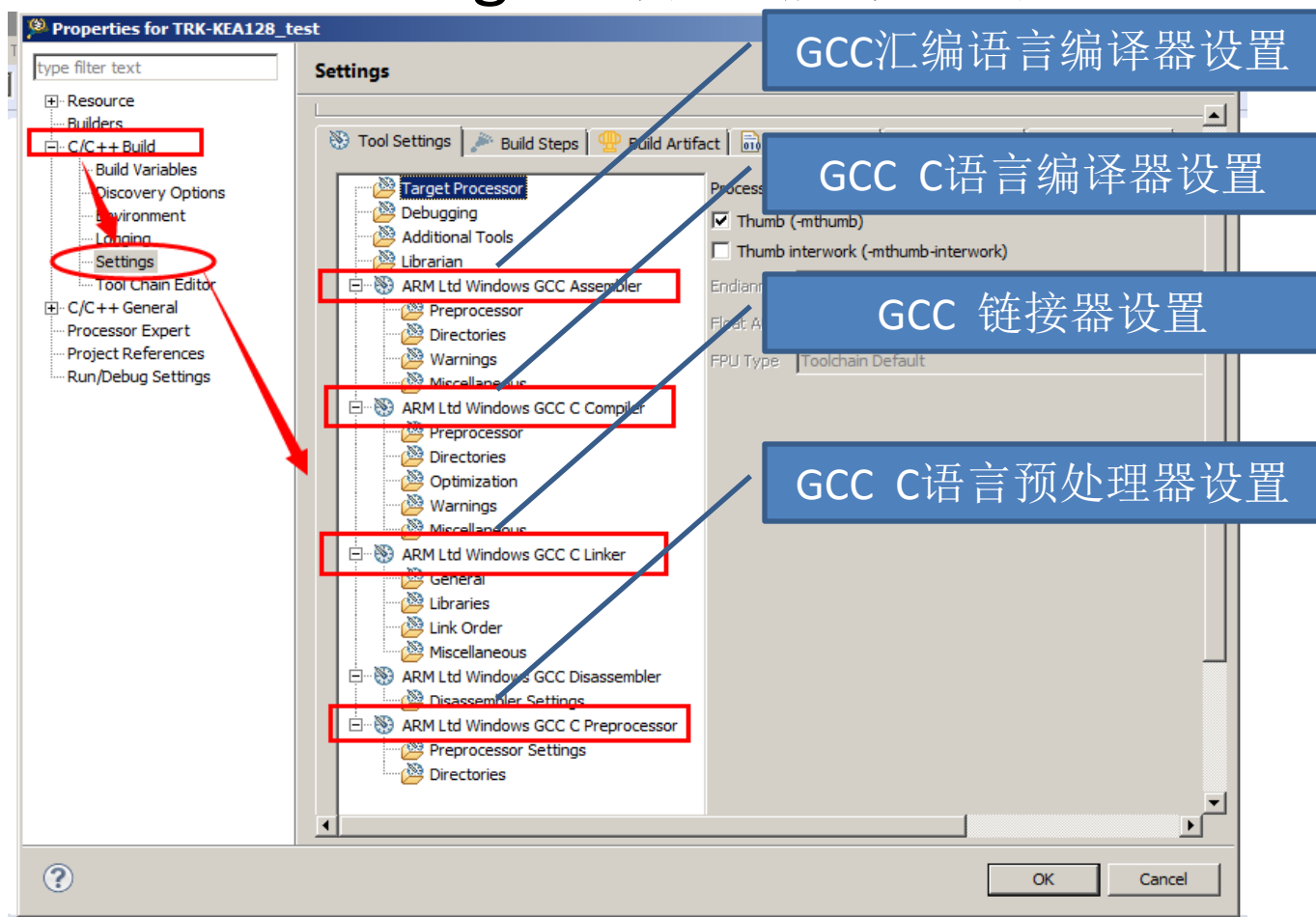
CodeWarrior工程属性（Properties）设置

- ❑ 对于CodeWarrior 10.6中每一个工程来说，属性设置是其能够正常被编译、链接和调试的关键。
- ❑ CodeWarrior工程属性的设置主要包括：
 - 预编译器设置
 - 汇编/C编译器设置
 - 链接器设置
 - 调试器设置
- ❑ 启动CodeWarrior工程属性设置的方法为：选择要设置的工程，右键→Properties(属性)，如右图：



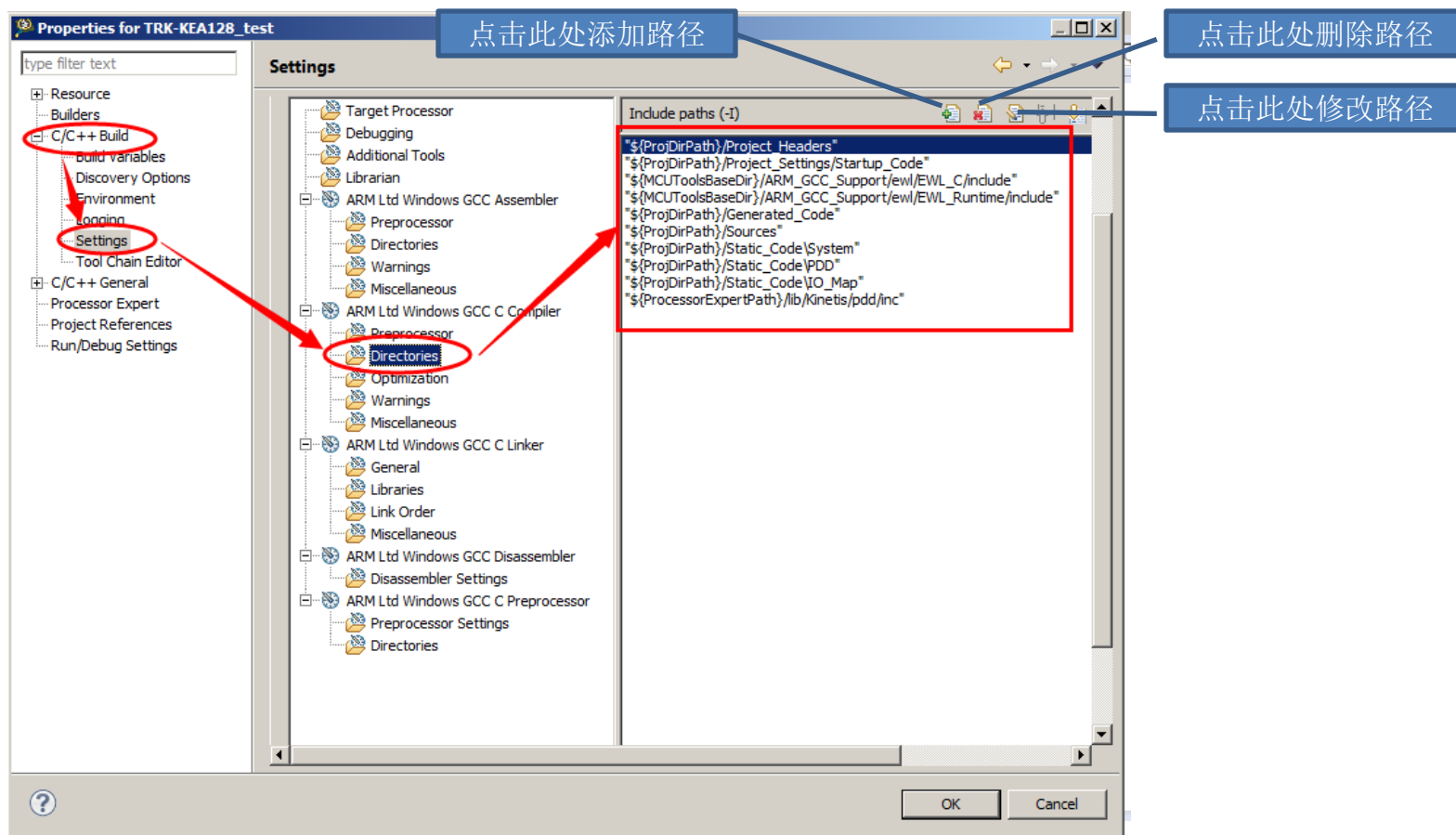
CodeWarrior工程属性（Properties）设置

□ C/C++ Build → Settings 查看和修改工程属性设置：



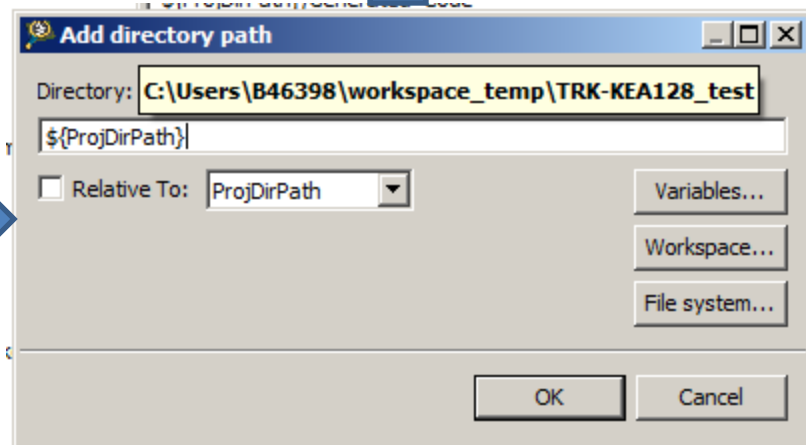
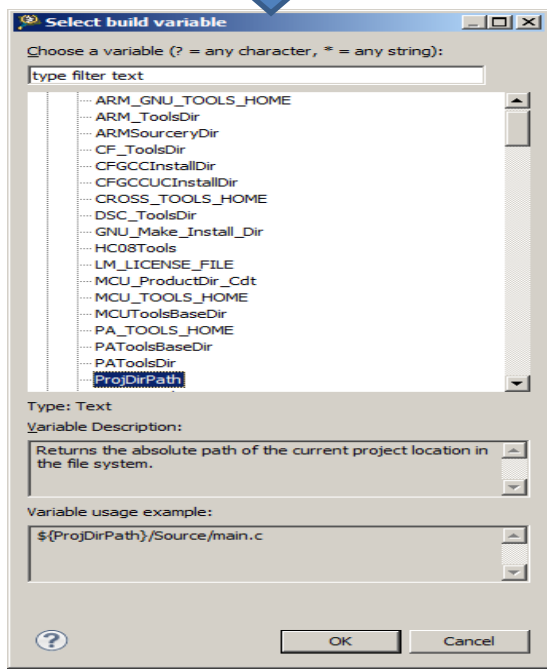
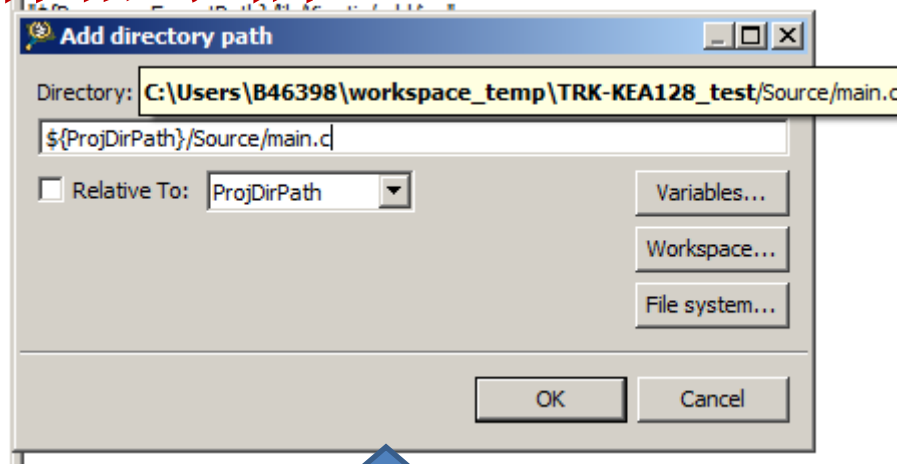
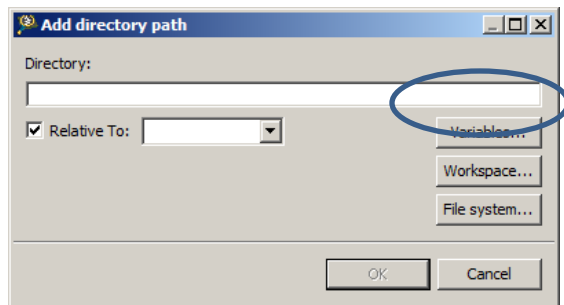
CodeWarrior工程GCC C语言编译器设置

- 设置工程C文件包含路径，只有在GCC C语言编译器设置处添加了该目录，GCC C语言编译器才能找到该路径下的C文件（.h和.c文件）



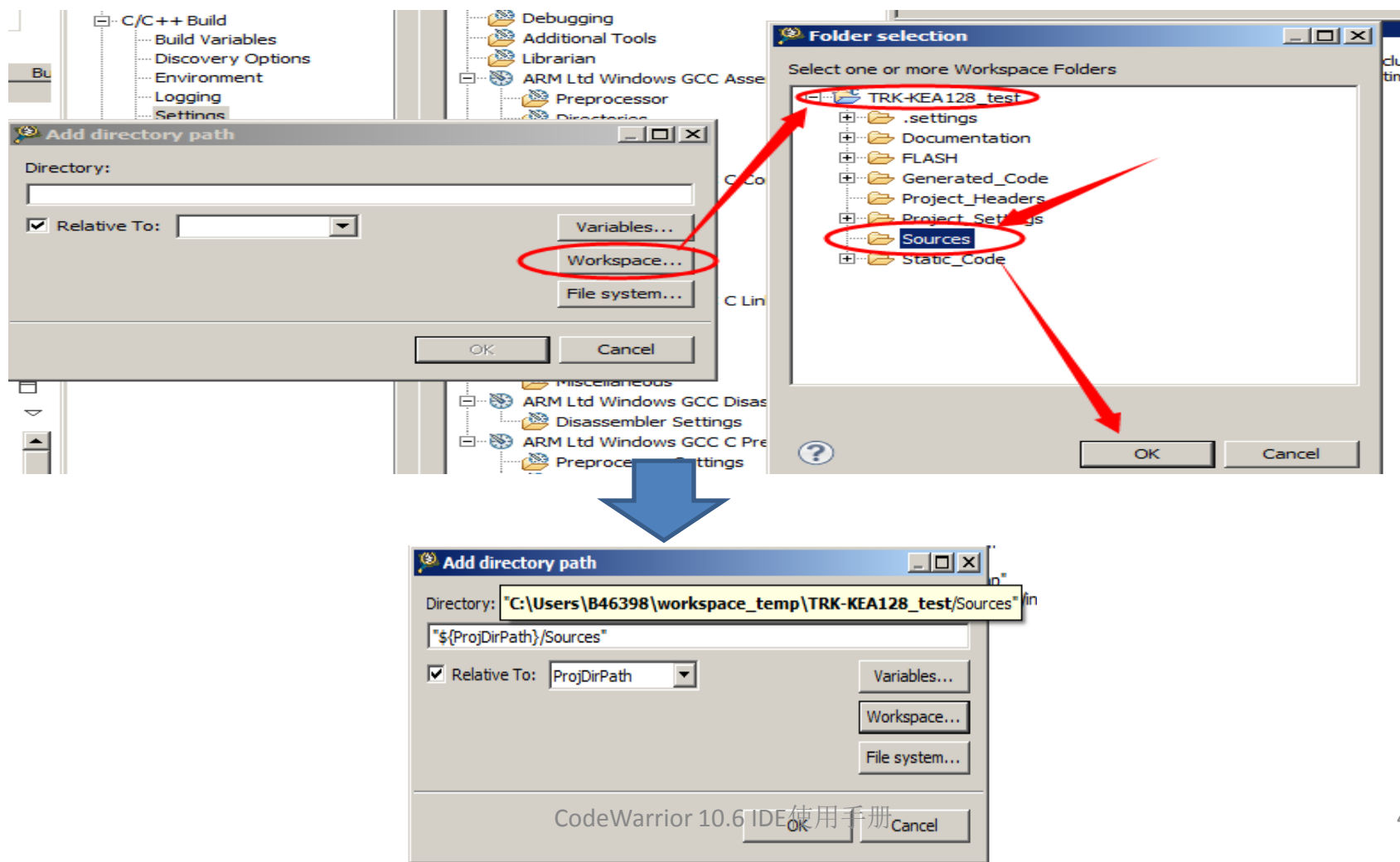
CodeWarrior工程GCC C语言编译器设置

□通过环境变量添加文件相对路径



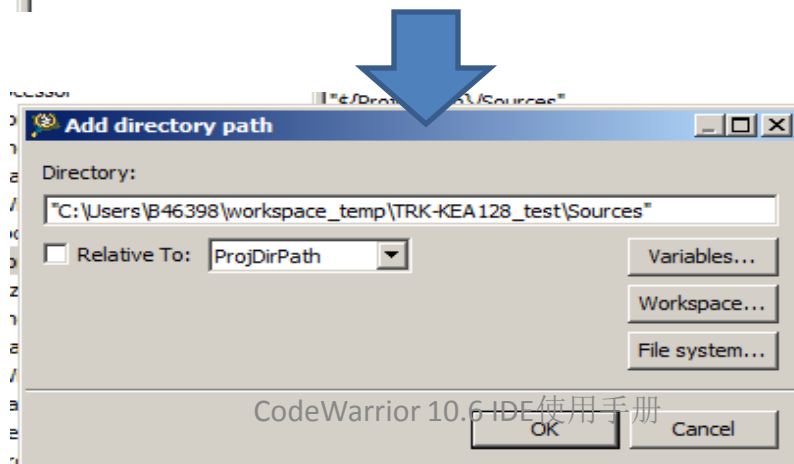
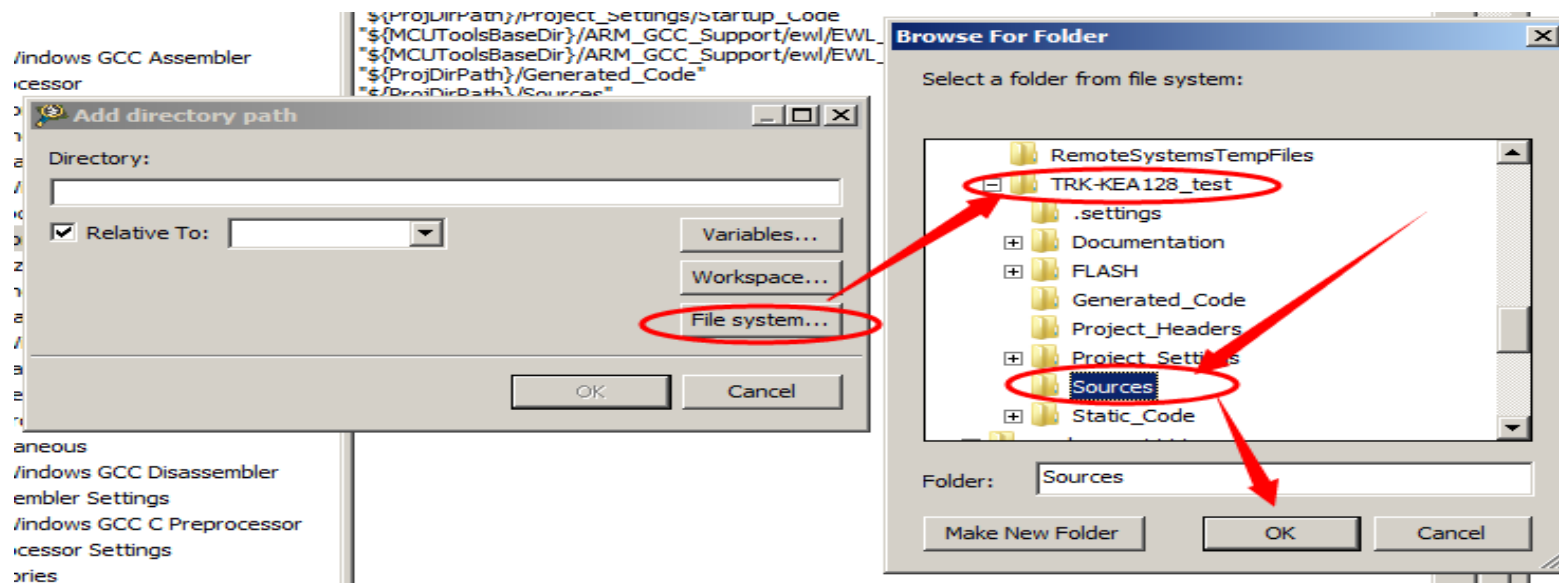
CodeWarrior工程GCC C语言编译器设置

□通过工作空间快捷添加文件相对路径



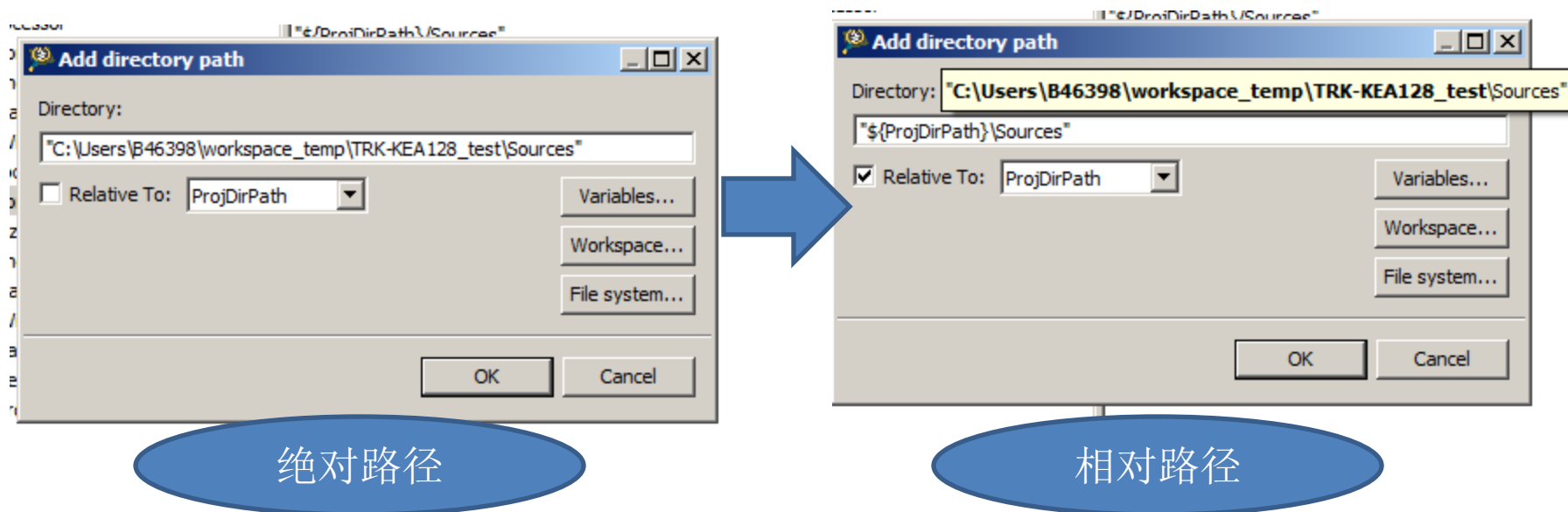
CodeWarrior工程GCC C语言编译器设置

□通过文件系统快捷添加文件绝对路径



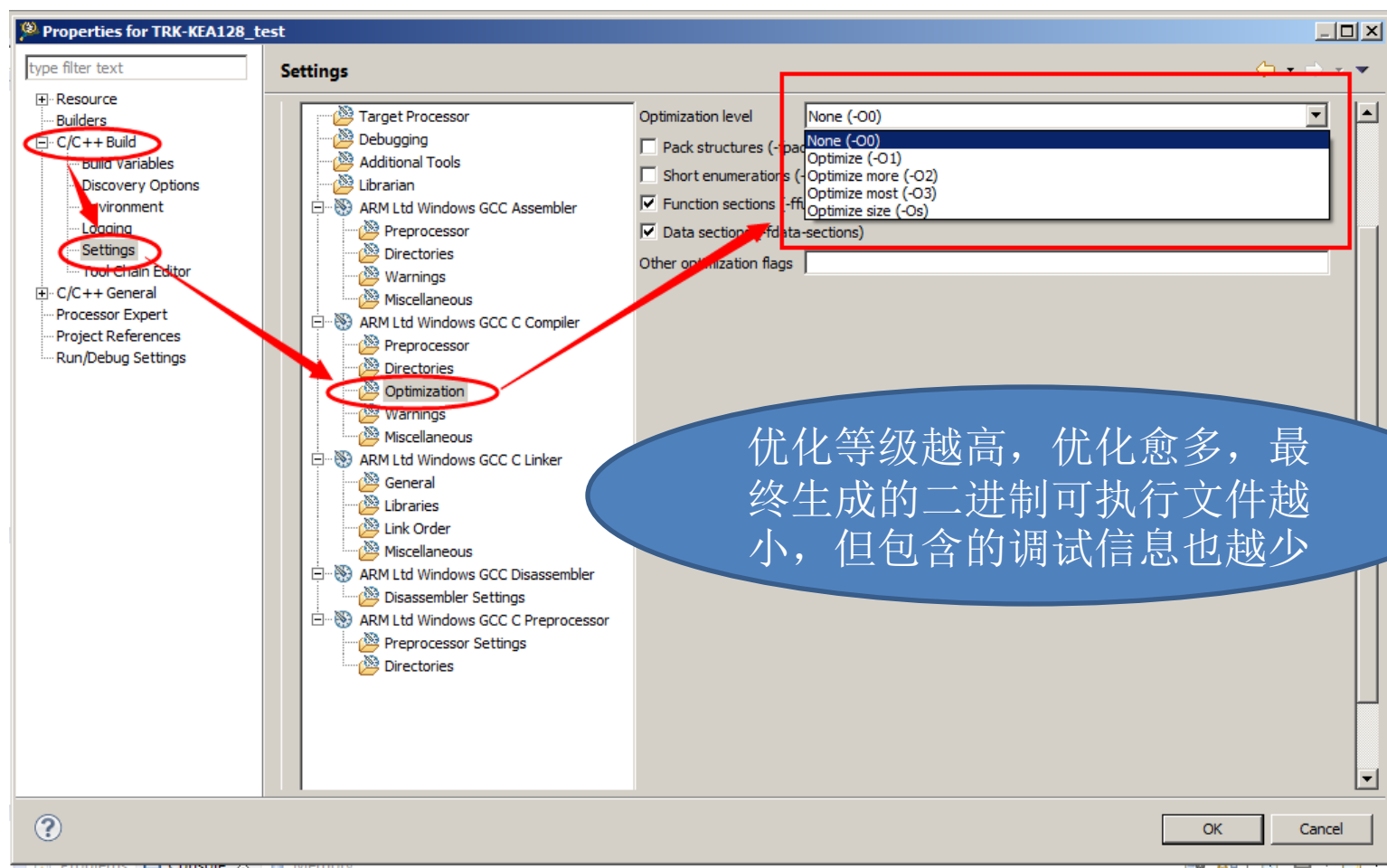
CodeWarrior工程GCC C语言编译器设置

- ❑ 通过勾选“**Related To**”快速将绝对路径转换为相对路径
- ❑ 在设置文件路径是尽量使用相对路径，以保证**CodeWarrior**工程在不同电脑上的正常使用（编译、链接和调试）



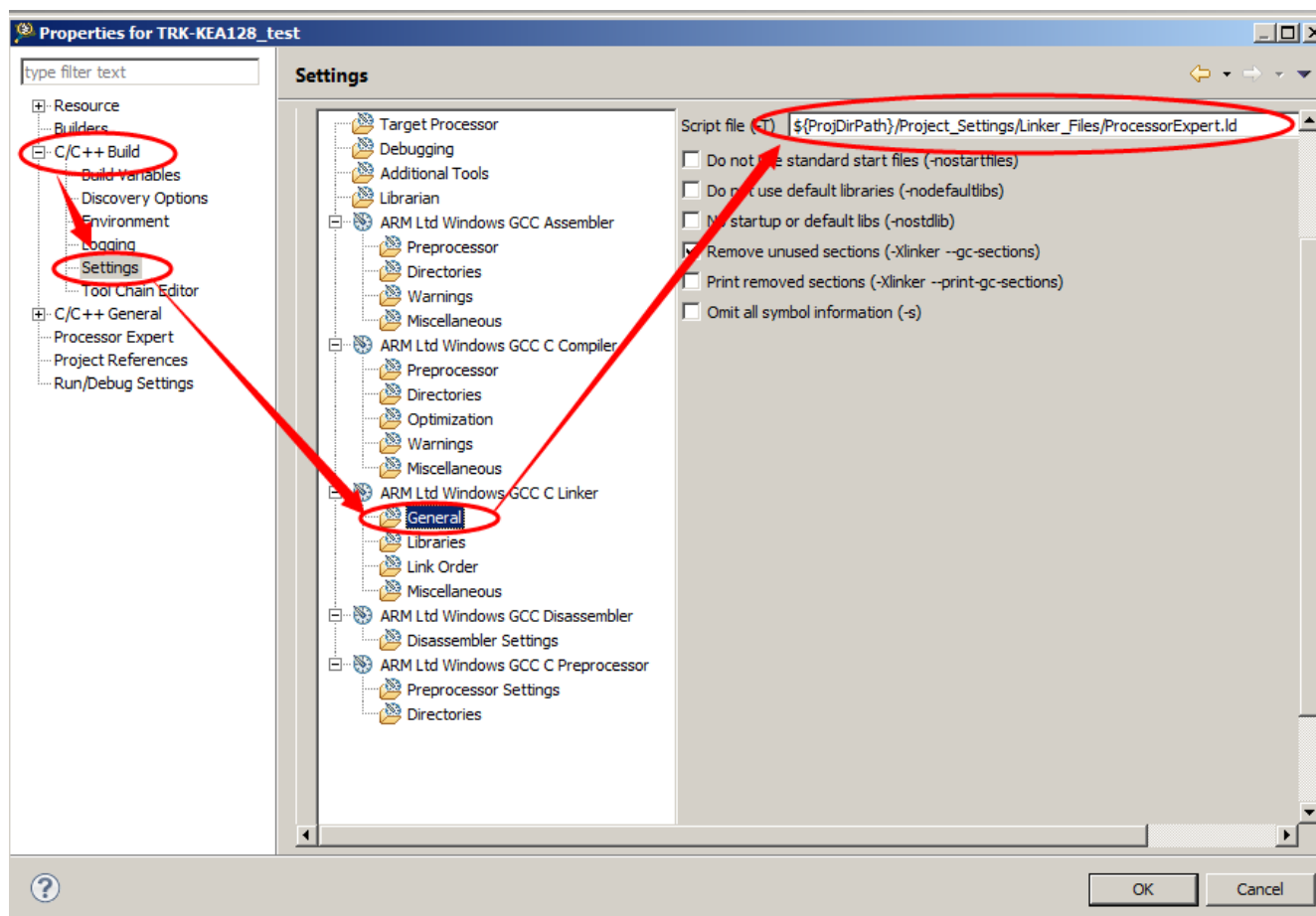
CodeWarrior工程GCC C语言编译器设置

❑ 在GCC C语言编译器设置中还可以设置工程的优化等级



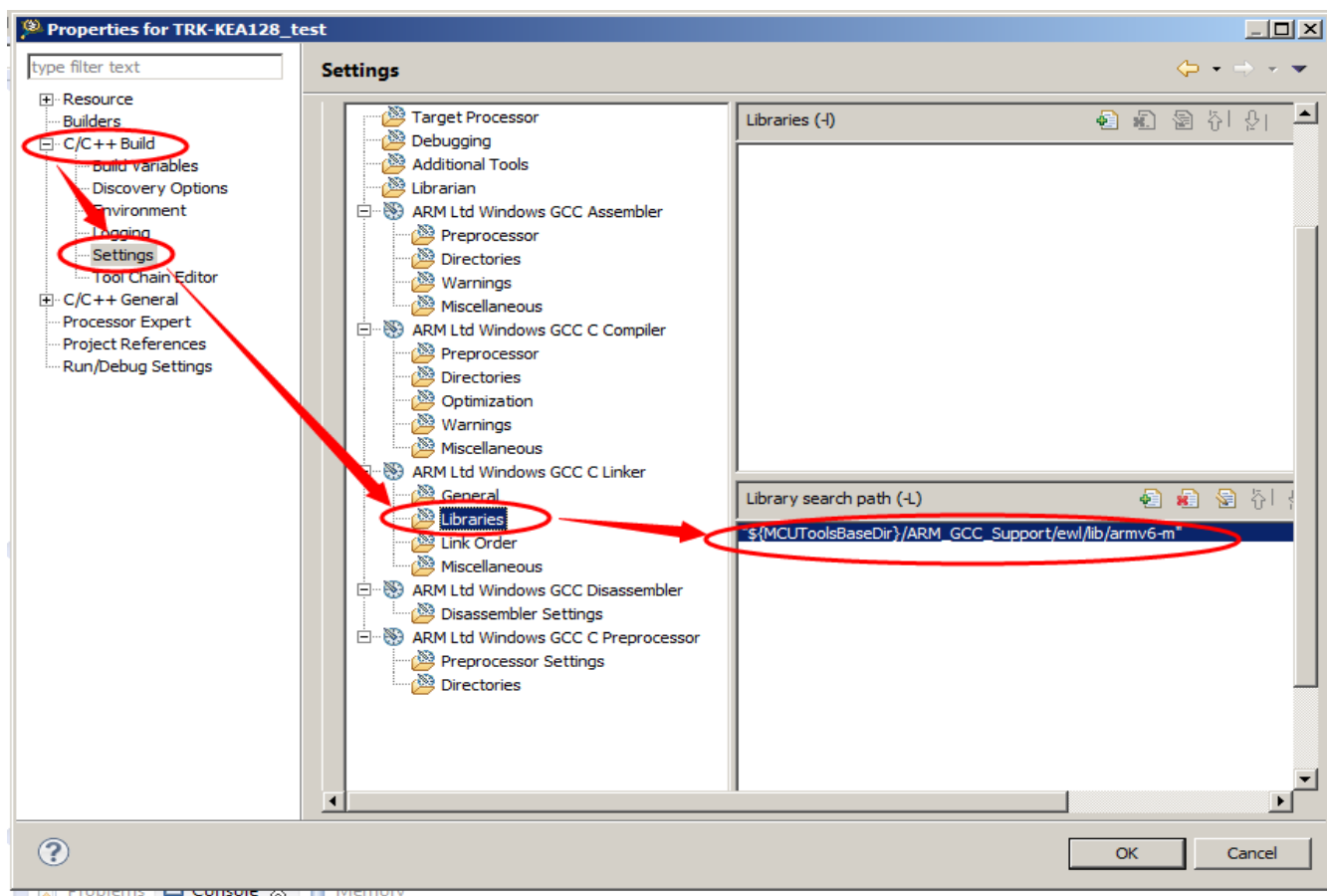
CodeWarrior工程GCC C语言链接器设置

□ 设置工程的链接文件



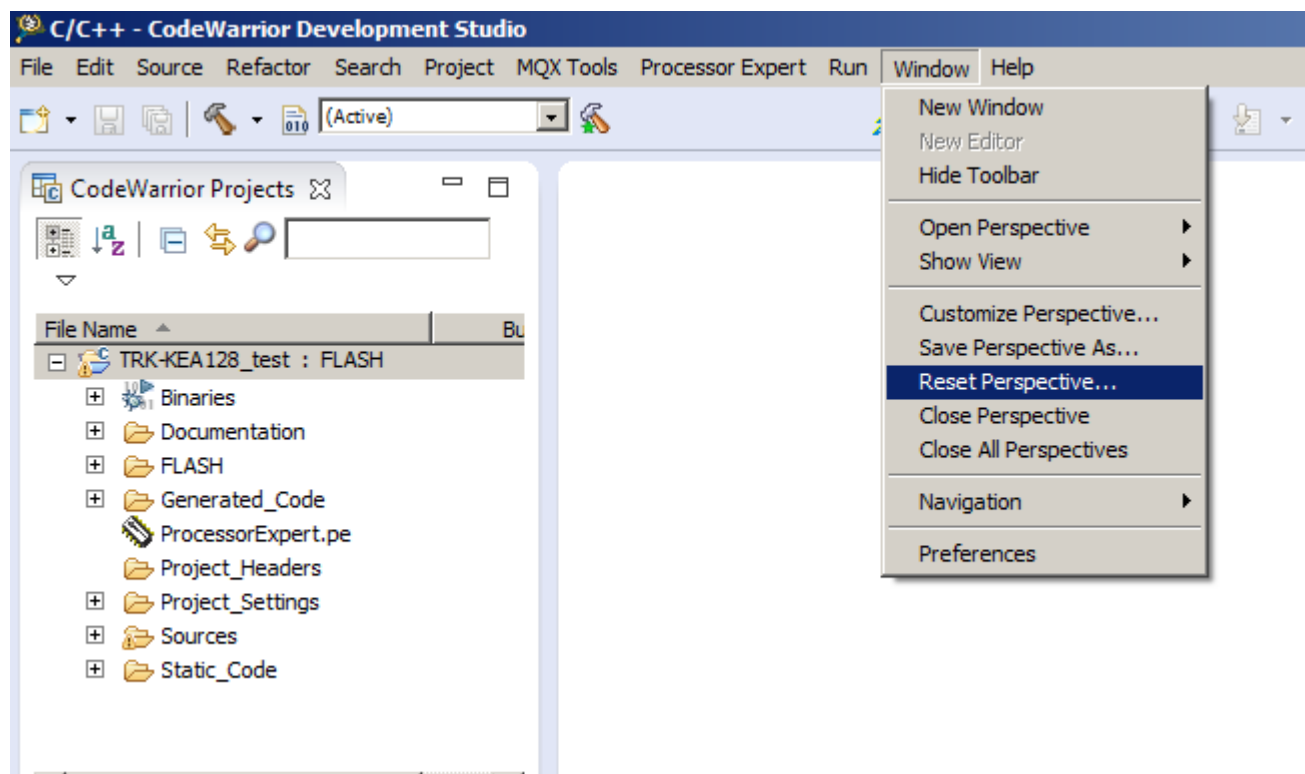
CodeWarrior工程GCC C语言链接器设置

- ❑ 在GCC C语言链接器设置还可以指定使用库的路径



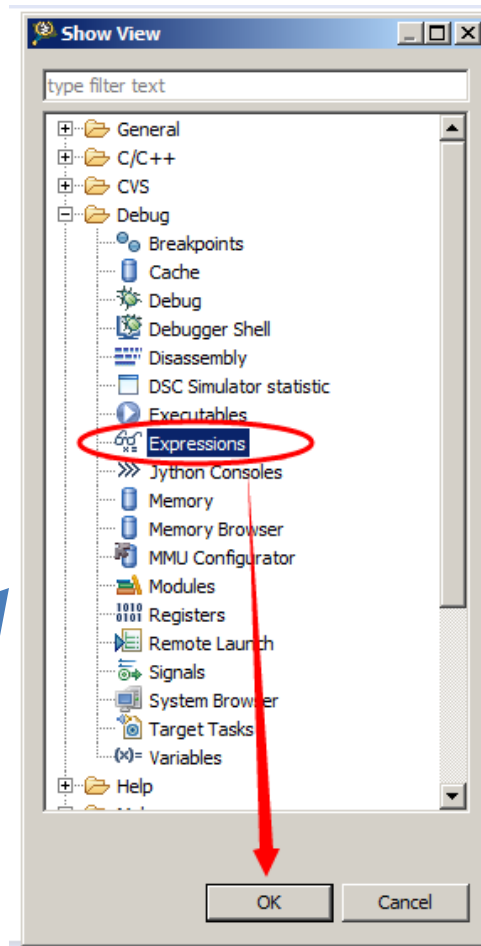
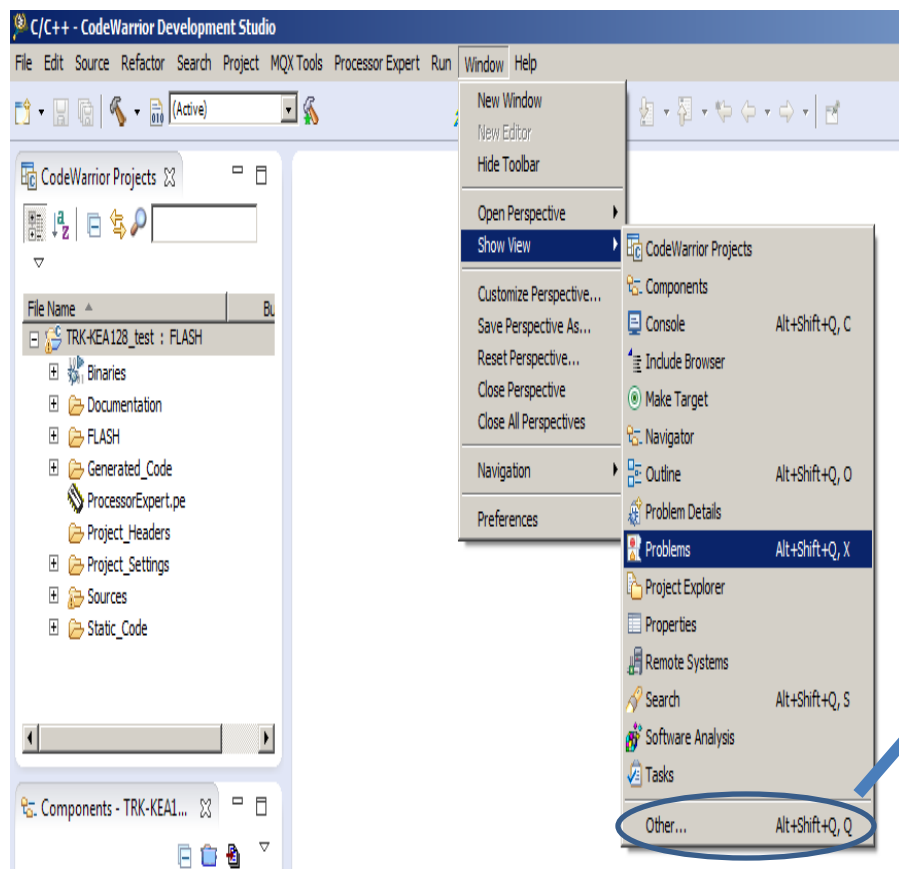
快速恢复默认的CodeWarrior窗口分布

- 如果想要恢复CodeWarrior默认的窗口布局，可以通过菜单Window→Reset Perspective完成：



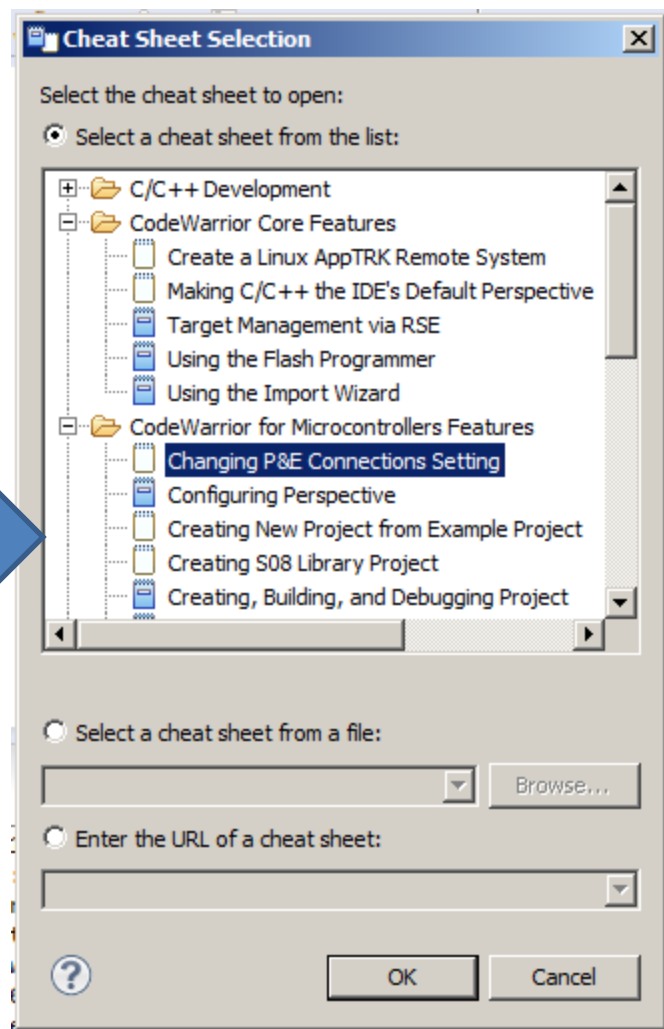
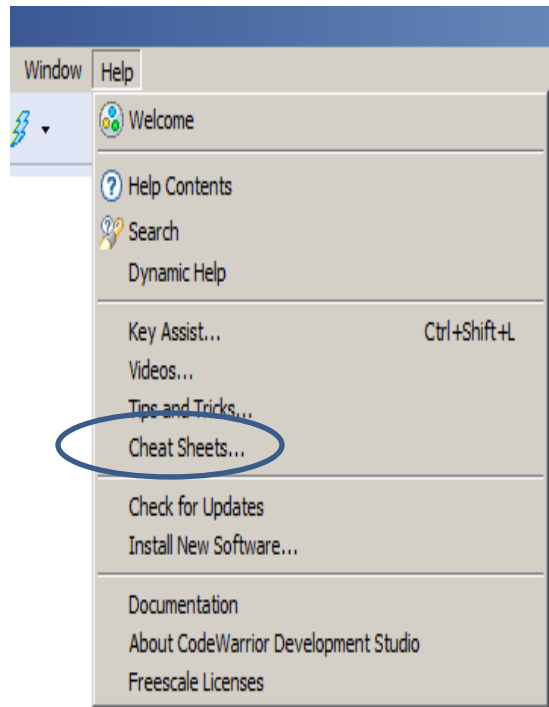
打开CodeWarrior视窗

- 有时如果不注意关掉了某些窗口，则可以通过菜单Window→Show View→选择要打开的视窗/窗口，重新打开：



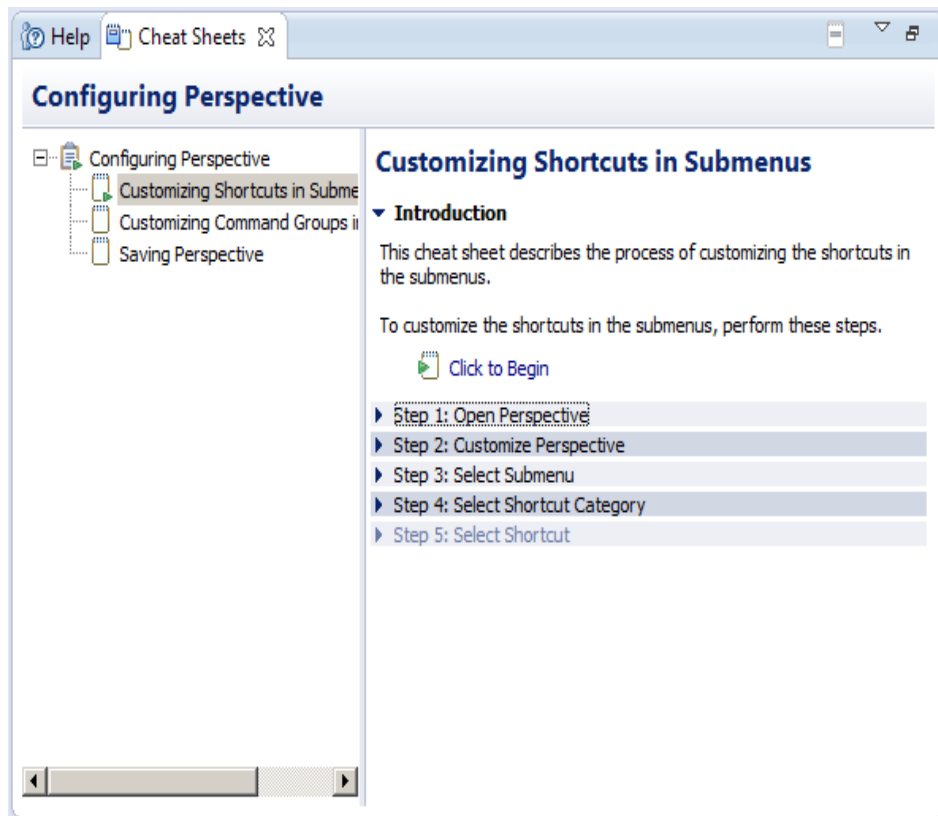
快速查看CodeWarrior帮助(Cheat Sheet)

- ❑ 用户可以通过菜单Help→Cheat Sheet
- ❑ 在弹出的窗口中选择要学习的CodeWarrior使用内容



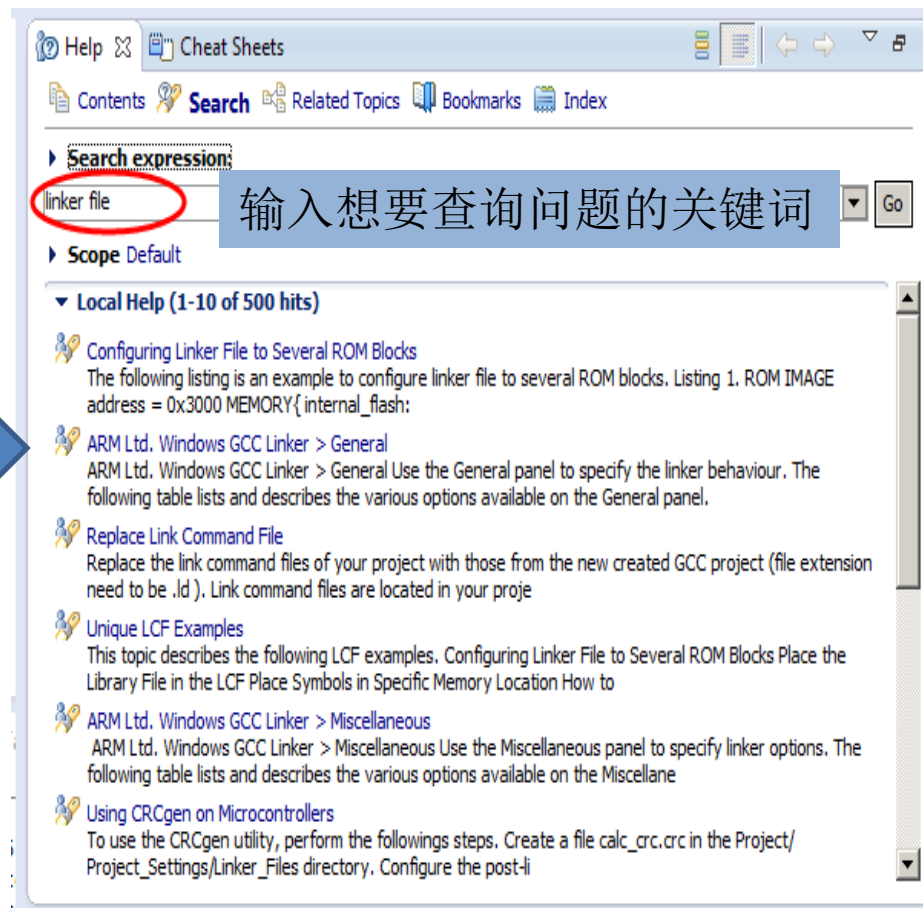
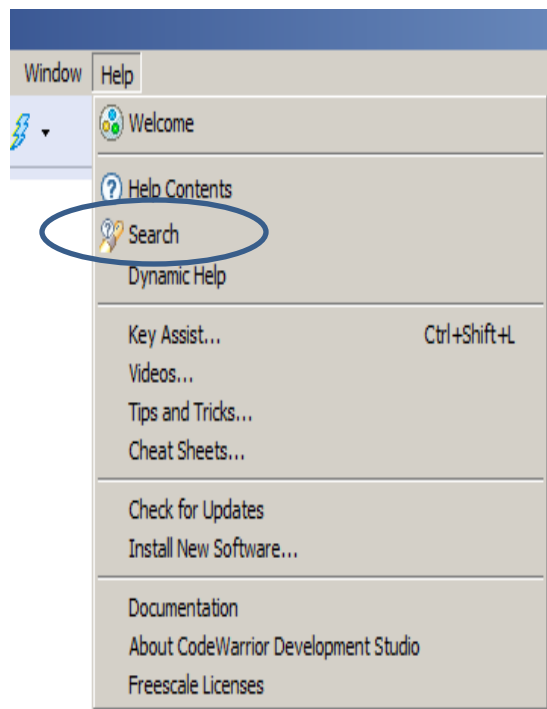
快速查看CodeWarrior帮助(Cheat Sheet)

❑ 通过Cheat Sheet窗口可以step by step来学习CodeWarrior 10.6 的使用



快速查看CodeWarrior帮助(Help Search)

❑ 用户还可以通过菜单Help→Search打开帮助查询窗口，查找帮助：



更多的CodeWarrior本地帮助手册

- ❑ 用户可以在CodeWarrior 10.6的安装目录下\MCU\Help\PDF 如：C:\Freescall\CW MCU v10.6\MCU\Help\PDF，找到关于CodeWarrior 10.6 IDE的使用、编译器、汇编器、连接器、MQX/MQXLite操作系统等的使用帮助手册/用户手册：

