AN4701
应用笔记

STM32F4系列微控制器的专有代码读取保护

前言

软件提供商开发了复杂的中间件解决方案（知识产权代码，或IP-Code），保护它们对于微控制器而言至关重要。

为了应对这一重要需求，STM32F4xx MCU具有以下功能：

- 读保护（RDP）：防止进行读取操作
- 写保护：防止进行不需要的写入或擦除操作
- 专有代码读取保护（PCROP）：防止进行读写操作

本应用笔记对这些闪存保护技术进行了说明，重点是专有代码读取保护（PCROP），并提供了PCROP保护的基本示例。

本文档随附的X-CUBE-PCROP固件封装包包含了PCROP示例的源代码，以及运行示例所需的所有固件模块。

本应用笔记必须与*表 1*中所列的参考手册一起阅读。这些文档及其对应的数据手册均可在*www.st.com*上获取。

表1. 参考文档

产品线	参考手册
STM32F401	RM0368
STM32F410	RM0401
STM32F411	RM0383
STM32F412	RM0402
STM32F413	RM0430
STM32F42x/STM32F43x	RM0090
STM32F446	RM0390
STM32F469/479	RM0386

目录

表格索引

图片目录

1 读保护（RDP）

读取保护是全局闪存读保护，可保护嵌入式固件代码，避免复制、逆向工程、使用调试工具读出或其他方式的入侵攻击。该保护应在二进制代码载入嵌入式闪存后，由用户进行设置。

1.1 读保护级别0

级别0是默认级别，闪存完全打开，可在所有引导配置（调试功能，从RAM、从系统内存引导加载程序或从闪存启动）下进行全部内存操作。这种模式下，没有保护（主要用于开发和调试）。

1.2 读保护级别1

激活读保护级别1时，即使是从SRAM或系统内存引导加载程序来启动，也不能使用调试功能（如串行线路或JTAG）访问（读取，擦除和编程）闪存或备用SRAM。

但是，当从闪存启动时，允许用户代码访问该内存和备用SRAM。

对受保护的闪存进行任何读取请求都会产生一个总线错误。

将RDP选项字节重新编程为级别0，可禁用RDP级别1保护，这会导致整体擦除。

1.3 读保护级别2

激活RDP级别2时，级别1所支持的所有保护均有效，芯片受到全面保护。RDP选项字节和所有其他选项字节都会被冻结，不能再修改。JTAG、SWV（单线查看器）、ETM 和边界扫描被禁用。

从闪存启动时，用户代码可以访问内存内容。但是，不再能从SRAM或从系统内存引导加载程序启动。

这种保护是不可逆的（JTAG熔断），所以不能回到保护级别1或0。

*表 2*描述了从内部闪存启动，或在调试，或从SRAM或系统内存引导加载程序启动时，对闪存、备用SRAM、选项字节和一次性可编程字节（OTP）的不同访问。

表2. 闪存内容访问 vs. RDP级别

存储区	保护级别	调试，从RAM或从系统内存引导加载程序启动			从 Flash 自举（用户代码）		
		读取	写入	擦除	读取	写入	擦除
主存储器和BKP	级别 1	无		无(1)	有		
	级别 2	无			有		
选项字节	级别 1	有			有		
	级别 2	无			无		
OTP	级别 1	无		NA(2)	有		NA(2)
	级别 2	无		NA(2)	有		NA(2)

1. 仅当RDP从级别1变为级别0时进行擦除。OTP 区域保持不变。
2. NA = 不适用。

1.4 RDP保护的STM32F4xx微控制器内部闪存内容更新

当RDP保护激活时（级别1或级别2），内部闪存内容不能通过调试进行更新，当从SRAM或系统内存引导加载程序启动时也不能更新。

因此对最终产品的一个重要要求就是，能够将内部闪存中的嵌入式固件升级为新的固件版本，添加新功能并修正潜在问题。

该要求可以通过实现用户专用固件来执行内部闪存的应用内编程（IAP）来解决，使用诸如USART的通信协议来进行重新编程过程。

关于IAP的更多详细内容，请参考应用笔记AN3965，可在*www.st.com*上获取。

2 写保护

写保护用来保护指定扇区内容，避免代码更新或擦除。这种保护可应用于扇区。

任何写请求都会产生一个写保护错误。如果要擦除/编程的地址属于闪存中处于写保护状态的区域，则通过硬件将WRPERR标志置位。

例如，如果闪存中至少有一个扇区是写保护的，则不能对其进行整体擦除，并且WRPERR标志置位。

根据nWPRi选项位的保护模式选择（SPRMOD选项位），闪存扇区可为写保护或读&写（PCROP）保护模式。要激活每个闪存扇区i的写保护（SPRMOD=0），可使用一个选项位（nWRPi）。当设置扇区 i（选项位nWRPi = 0）为写保护时，该扇区不能被擦除或编程。

可通过嵌入式用户代码或使用STM32 ST-Link Utility软件和调试接口，进行使能或禁用写保护管理。

3 专有代码读保护（Proprietary Code Read Out Protection，PCROP）

3.1 PCROP保护概述

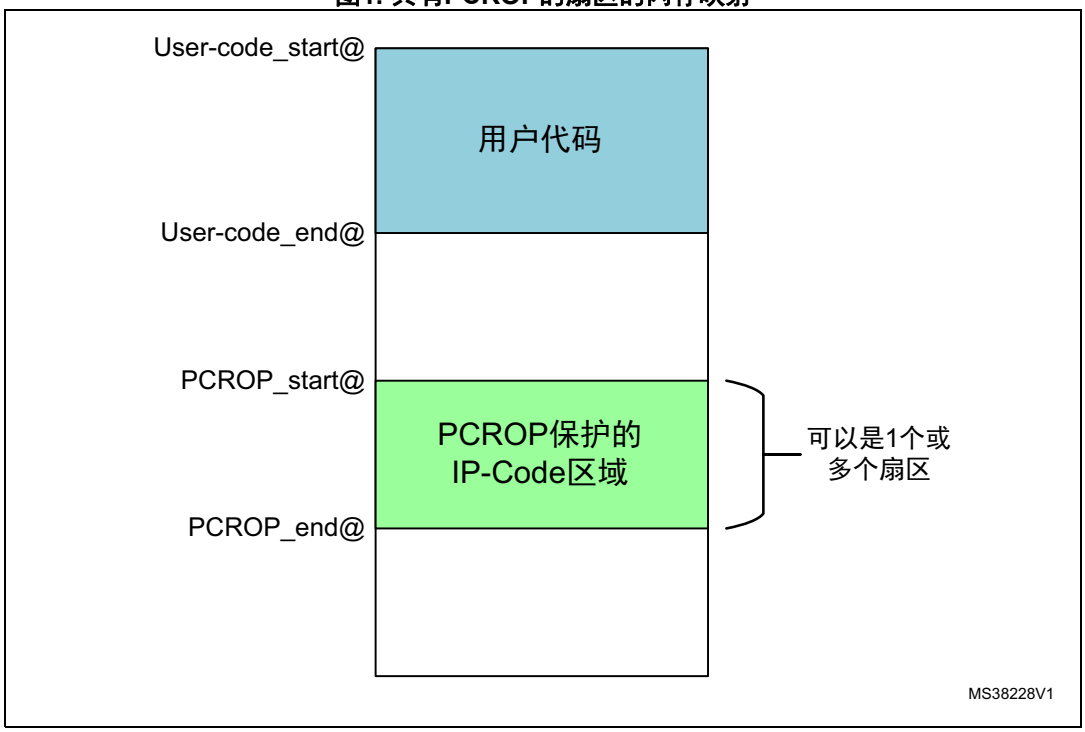
PCROP是闪存中IP-Code的读写保护，应用于扇区，保护专有代码不被最终用户代码、调试器工具或RAM Trojan代码所修改或读取。

任何读或写请求都会产生一个读或写保护错误：

- 如果要擦除/编程的地址属于闪存中PCROP的区域，则通过硬件将WRPERR标志置位。
- 当通过D总线对PCROP的扇区执行读访问时，RDERR标志置位。
- 当从PCROP的扇区检测到读或写请求且无法运行该请求时，会产生一个闪存操作错误中断，并且FLASH_SR寄存器中的OPERR标志置位。这仅在FLASH_CR寄存器中ERRIE位被置位时有效。

受保护的IP-Code可以很容易地被最终用户应用程序所调用，并且仍能受到保护，不可直接访问IP-Code本身。PCROP不会阻止执行受保护的代码。

图1. 具有PCROP的扇区的闪存映射



3.2 如何使能PCROP保护

要激活PCROP，必须激活SPRMOD选项位，这会改变nWRP选项位的功能。SPRMOD选项位有效的情况下，利用与写保护同样的选项字节，可选择PCROP扇区。每个扇区可以单独进行PCROP保护，还可以保护附加扇区（当RDP设为级别0或1时），而不必进行全片擦除，这与禁用PCROP保护不同。

激活PCROP功能时应采取一些预防措施。当激活PCROP模式时，nWRPi位的有效值会被反转，因此如果SPRMOD = 1且nWRPi = 1，那么用户扇区 i 是PCROP保护的。

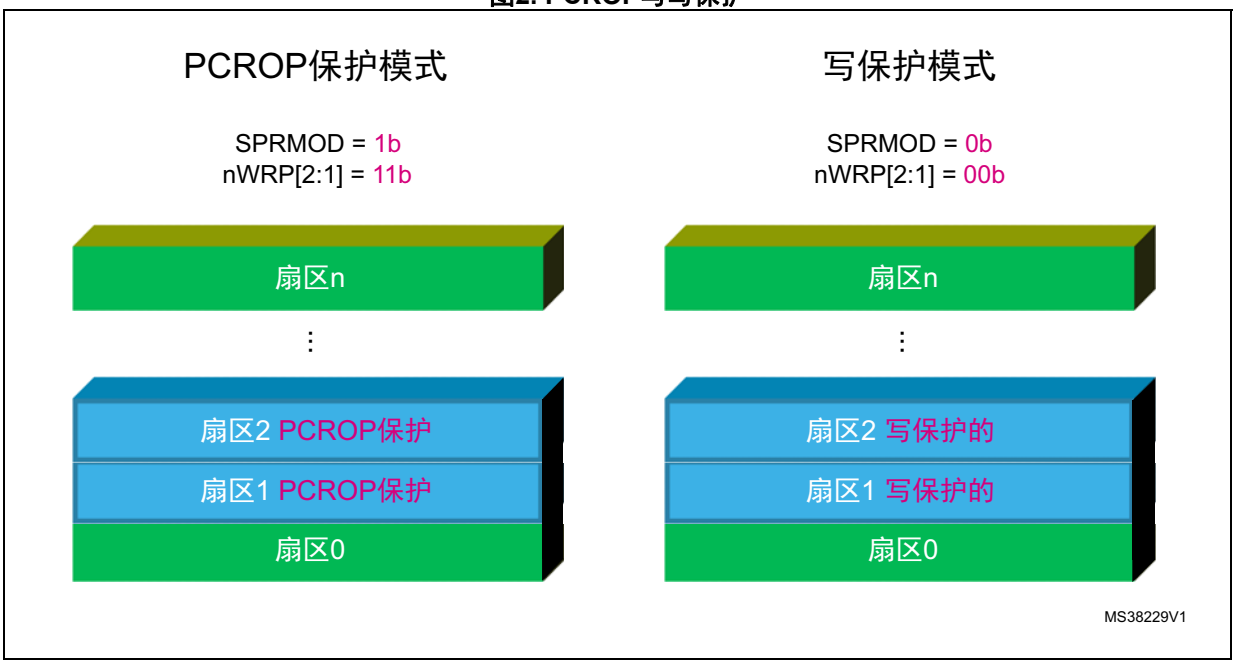
要正确激活扇区 i 上的PCROP，必须按照以下详细步骤：

1. 清除所有的nWRP位，除了那些已经被PCROP保护的扇区（对于具有两个Flash存储器组的STM32F4产品，如STM32F429ZIT，bank2的nWRP位也必须清零）；
2. 将需要PCROP保护的扇区i的nWRPi置位；
3. 通过置位SPRMOD位使能PCROP保护。

关于实现PCROP使能的更多详细内容，请参考所提供的FW包（*Step1-ST_Customer_level_n project main.c*文件）中所述的*PCROP_Enable()*函数。

要重点注意的是，不能同时存在一个写保护扇区和另一个PCROP保护扇区。因此用户或者处于写保护模式，对扇区进行写保护，或者处于PCROP模式，扇区为PCROP保护。*图 2*显示了PCROP和写保护之间的区别。

图2. PCROP与写保护



3.3 如何禁用PCROP保护

根据RDP级别，如果RDP级别为1或0，则PCROP可被禁用，但是如果RDP设置为级别2，PCROP不能被禁用。当RDP设为级别2，所有选项字节都会被冻结，不能修改。因此，PCROP保护的扇区不能再被擦除或修改，这样就成为永久性保护。

在受保护扇区上禁用PCROP的唯一方法是，将RDP级别从1改为0，同时清除SPRMOD位。

应用程序开发过程中，用户可能需要禁用PCROP或全局RDP保护，而不必花时间开发或禁用保护功能。STM32 ST-LINK Utility工具是一个非常简单的禁用或使能保护的方式，利用调试接口如JTAG或SWD即可实现，而无需开发专门的功能。

关于如何使用STM32 ST-LINK Utility软件的更多详细信息，请参考用户手册UM0892，可在www.st.com上获取。

注： *禁用PCROP或/和全局RDP保护会导致全片擦除。*

3.4 存放和执行PCROP的IP-Code

如前所述，PCROP不会阻止执行受保护的IP-Code，并且其函数可方便地被用户代码所调用。

PCROP保护扇区不被D-Code总线读访问，因此这里重点指出，只允许进行代码执行（通过I-Code总线取指令），而禁止读取数据。因此，受保护的IP-Code将无法访问存储在同一区域的相关数据值（如文字池、分支表或常量，执行过程中它们通过D-Code总线从闪存中取出）。

PCROP的代码必须是只可执行的代码，不能包含任何数据。

用户必须配置编译器来生成只可执行的IP-Code，并且避免任何数据读取，所需编译器配置在*第 4 节*中有详细说明。

3.4.1 不能PCROP保护向量表扇区

中断向量表包含了每个中断处理程序的进入点地址，它们可由CPU通过D-Code总线进行读取。通常中断向量表位于首地址0x08000000的第一个扇区（例外是在某些情况下，它会被重新定位到其他区域，如SRAM）。

将代码置于闪存中进行保护时，必须遵守以下规则：

- 向量表所在的第一个闪存扇区不能PCROP保护
- 待PCROP保护的代码不能位于第一个扇区中。

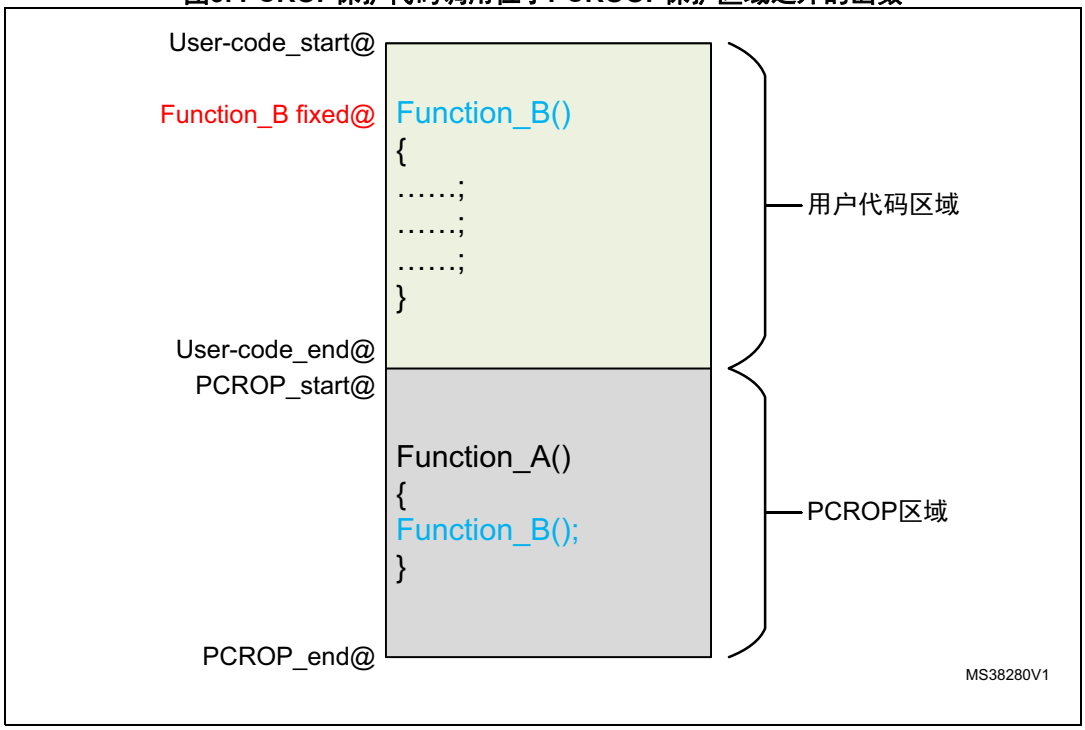
3.4.2 PCROP的IP-Code依赖关系

受保护的IP-Code可以从位于用户代码区域中以及PCROP保护区之外的库中调用函数。这种情况下，IP-Code中包含了相关函数地址，允许PC（程序计数器）在执行IP-Code时跳转到这些函数。一旦IP-Code被PCROP保护，则这些地址不能更改。因此，每个调用函数必须位于（PCROP保护区域之外）PCROP保护IP-Code中写入的相应固定地址，否则PC跳转到无效地址，IP-Code将无法正常工作。

要完全独立，受保护IP-Code必须与其所有关联函数放在一起。

*图 3*显示一个示例，其中PCROP保护的Function_A()调用位于PCROP保护区域之外的固定地址的Function_B()。

图3. PCROP保护代码调用位于PCROOP保护区域之外的函数



3.5 STM32F4系列的PCROP功能可用性

*表 3*给出了STM32F4微控制器的PCROP保护可用性、闪存大小和扇区数的概述。

表3. STM32F4系列的PCROP可用性概述

产品线	闪存（字节）	扇区数	PCROP可用性
STM32F401	最大 512 K	多达8	有
STM32F405/415	最大 1 M	多达12	无
STM32F407/417	最大 1 M	多达12	无
STM32F411	最大 512 K	多达8	有
STM32F427/437	最大 2 M	多达24	有
STM32F429/439	最大 2 M	多达24	有
STM32F446	最大 512 K	多达8	有
STM32F469/479	最大 2 M	多达24	有

4 PCROP示例

本应用笔记提供的固件示例对PCROP保护的使用情形进行了说明。开发此固件所需的所有步骤均在本章详细说明。

4.1 示例要求

4.1.1 硬件要求

运行此示例所需的硬件如下：

- 嵌入了STM32F429ZIT MCU的STM32F429I-DISCOVERY板
- 微型USB数据线，用来为NUCLEO板上电，并连接嵌入式探索STLINK进行调试和编程。

4.1.2 软件要求

需要下列软件工具：

- IAR Embedded Workbench® 或Keil® µvision IDE
- STM32 STLink Utility主要用于使能或禁用PCROP保护。

4.2 示例概述

本例描述了ST用户级别n向ST用户级别n+1提供具有关键IP-Code的预编程STM32F429ZIT MCU的用例。

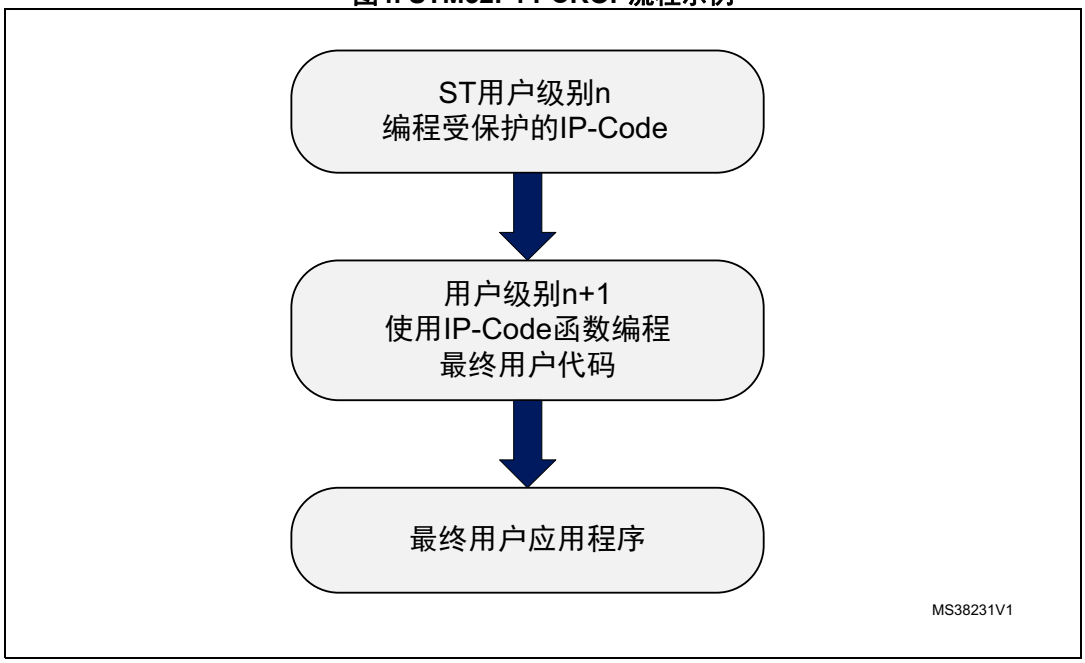
IP-Code必须通过激活PCROP来保护，允许ST用户级别n+1使用其函数（不能读取或修改它）来对最终用户应用程序进行编程。

ST用户级别n应将下列输入与预加载的STM32 MCU一起提供：

- 闪存映射，定义了准确的PCROP保护IP-Code位置和可用的编程扇区
- ST用户级别n+1项目必须包含的头文件，其中含有最终用户代码中调用的IP-Code函数定义
- 符号定义 文件，包含PCROP保护的IP-Code函数符号。

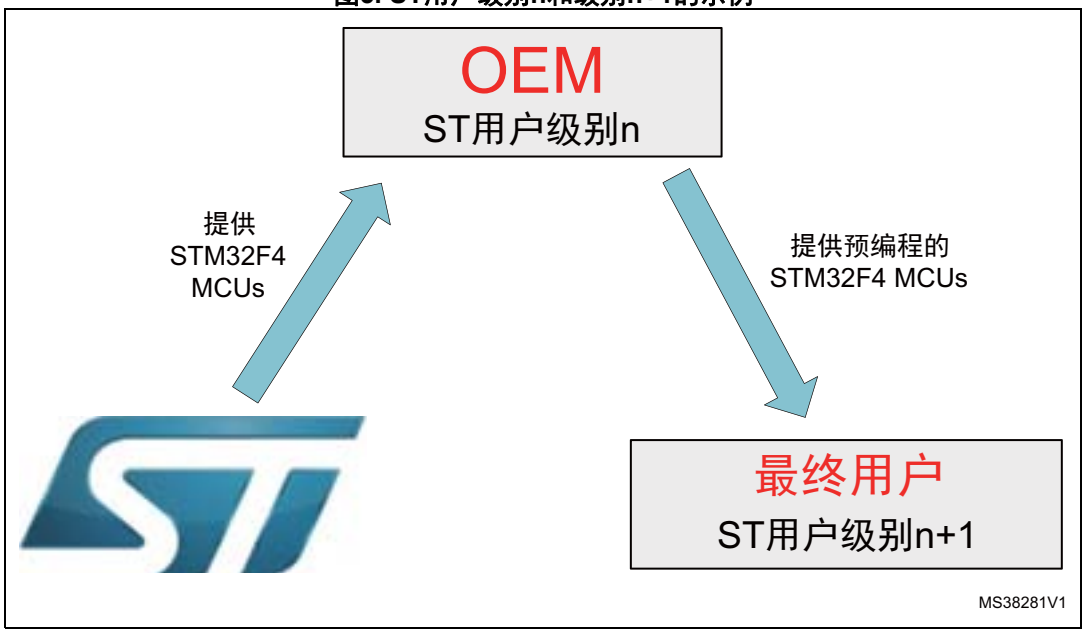
所述用例如*图 4*中所示意。

图4. STM32F4 PCROP流程示例



OEM（原始设备制造商）可以是使用STM32F4微控制器的ST用户级别n。然后OEM提供预编程MCU给ST用户级别n+1，这可能是制造最终用户产品的END CUSTOMER，如*图 5*中所示。

图5. ST用户级别n和级别n+1的示例



4.2.1 软件设置

本应用笔记提供了两个项目：

- 项目1：*STEP1-ST_Customer_level_ n*

  此项目显示了ST用户级别n如何存放、保护和执行其IP-Code，以及如何生成IP-Code关联文件如头文件和符号定义文件，并提供给ST用户级别n+1的示例。

  此项目包括两种不同的项目配置：

  – PCROP-IP-Code-XO：此配置下，编译器配置为生成只执行IP-Code，避免对其进行数据读取。

  – PCROP-IP-Code：此配置下，编译IP-Code，不阻止数据（文字池）生成。此配置专门用来进行测试，显示PCROP保护的IP-Code必须为只执行代码。

- 项目2：*STEP2-ST_Customer_level_n+1*

  此项目显示了ST用户级别n+1，在得到预编程了PCROP保护的IP-Code 的STM32F429ZIT后，如何利用这些受保护的IP-Code函数来创建其自己的最终用户应用程序的示例。

4.3 PCROP保护的IP-Code：FIR低通滤波器

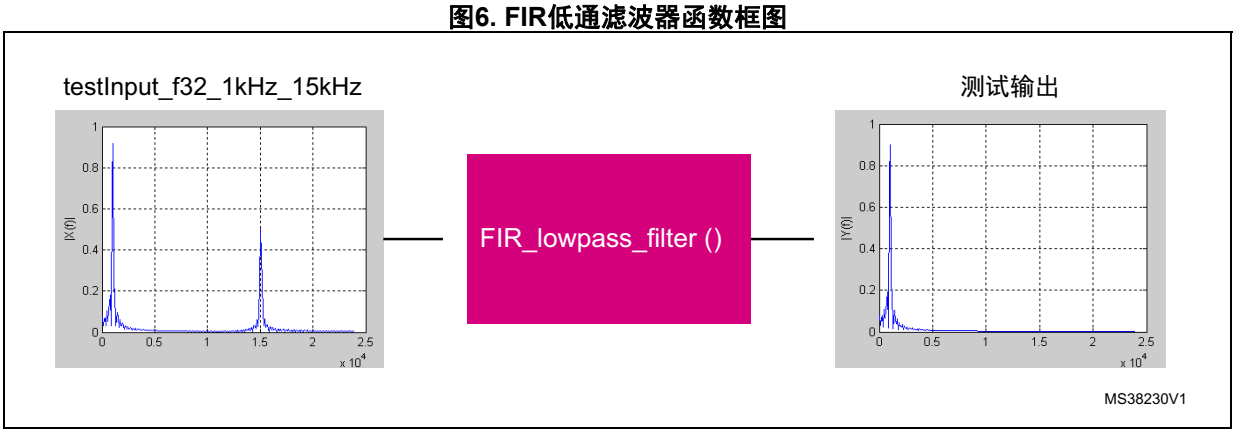
作为待保护的IP-Code示例，将对来自CMSIS-DSP的FIR低通滤波器算法进行描述，重点介绍在不提供函数本身详细信息的情况下，如何保护和调用此IP-Code函数。

FIR低通滤波器从输入中滤除高频信号分量。

输入信号是频率为1 KHz和15 KHz的两个正弦波之和。低通滤波器（其预配置截止频率设为6 KHz）滤除15 KHz信号，在输出端留下1 KHz正弦波。

*图 6*显示了FIR低通滤波器框图。

图6. FIR低通滤波器函数框图



所用CMSIS DSP软件库函数：

- *arm_fir_init_f32()*：配置滤波器的初始化函数，在*arm_fir_init_f32.c*文件中有描述；
- *arm_fir_f32()*：表示FIR滤波器的初等函数，在*arm_fir_f32.c*文件中有描述。

以下函数利用上述CMSIS DSP函数来创建：

- *FIR_lowpass_filter()*：表示FIR滤波器的全局函数，在*fir_filter.c*文件中有描述。

嵌入到STM32F4微控制器中的FPU和DSP用来进行信号处理和浮点计算，以输出正确信号。

关于FIR函数的更多详细信息，用户可参考相关软件包里
“Drivers/CMSIS/Documentation/DSP”目录中的CMSIS文档。

4.4 STEP1：ST用户级别n

此阶段中，ST用户级别n将：

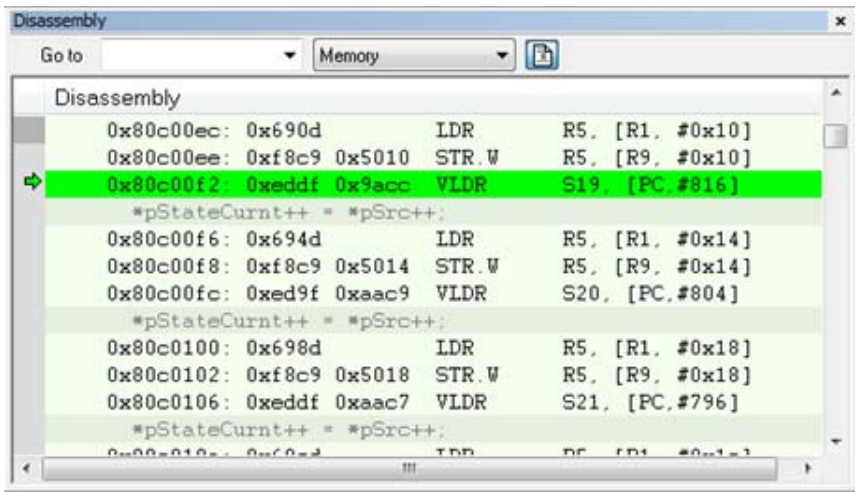
- 生成只执行IP-Code；
- 将IP-Code置于内部闪存的扇区2中；
- 保护置于扇区2中的IP-Code；
- 通过在主代码中调用其函数，执行IP-Code；
- 创建头文件，并生成符号定义文件，用于
STEP2-ST_Customer_level_n+1项目。

4.4.1 生成只执行IP-Code

每个工具链都有其自己的选项，来防止编译器生成文字池和分支表。例如，Keil®具有
Execute-only Code选项，而IAR有
No data reads in code memory选项。

*图 7*显示了包含文字池的汇编代码，其中指令格式为*VLDR <variable>, [PC + <offset>]*。

图7. 包含文字池的汇编代码示例



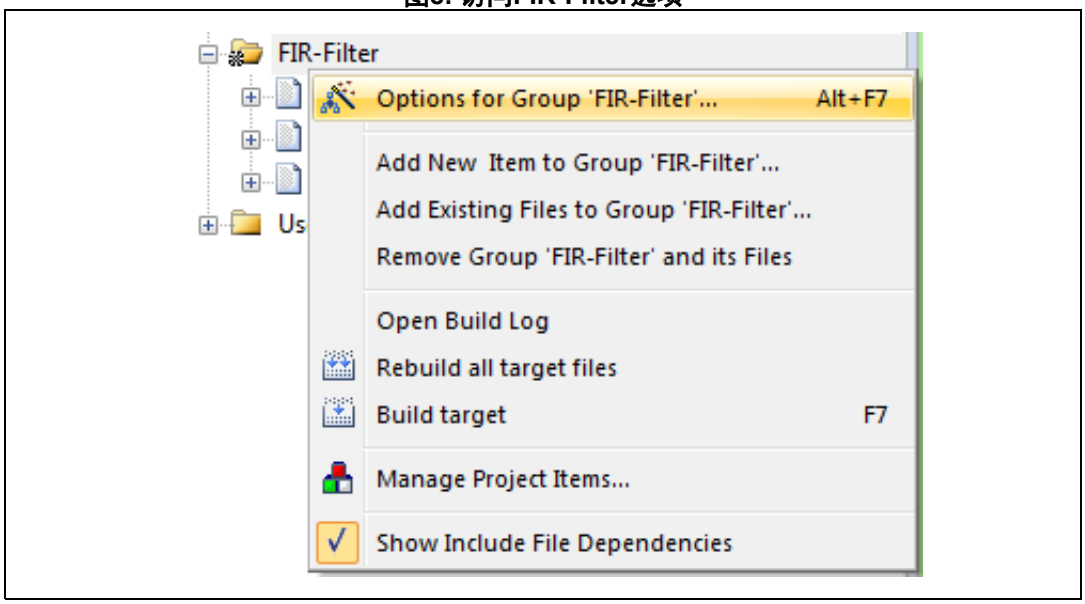
Keil®：使用*Execute-Only*命令

对于Keil®，必须使用命令*--execute_only*来生成无文字池和分支表的代码，防止编译器对代码扇区生成任何数据访问。

*--execute_only*命令必须用于PCROP保护的代码。

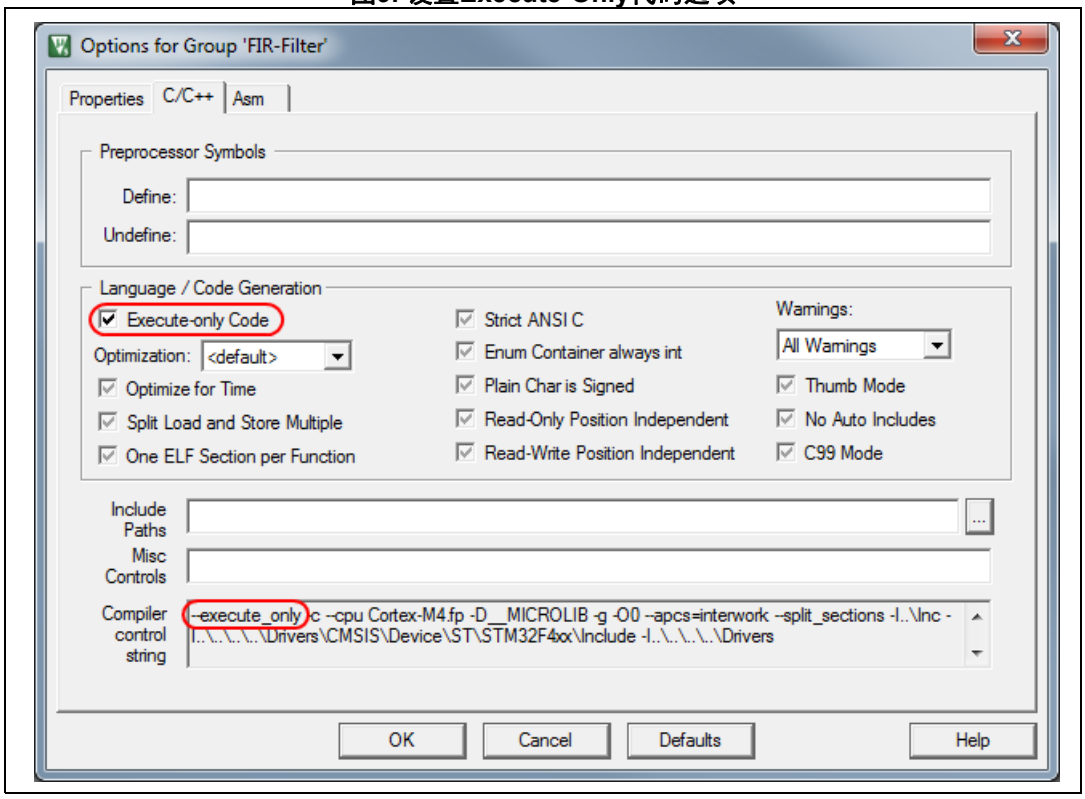
要设置此命令，请右键单击组文件或IP-Code，并选择**Options for Group 'FIR-Filter'**，如*图 8*中所示。

图8. 访问FIR-Filter选项



在以下窗口中（见*图 9*）选择选项**Execute-only Code**，然后*–execute_only*命令会自动被添加到编译器控制字符串字段中。

图9. 设置Execute-Only代码选项



然后在分散文件中，用*execute-only +XO*替换选项*read only +RO*。

更新Keil®分散文件

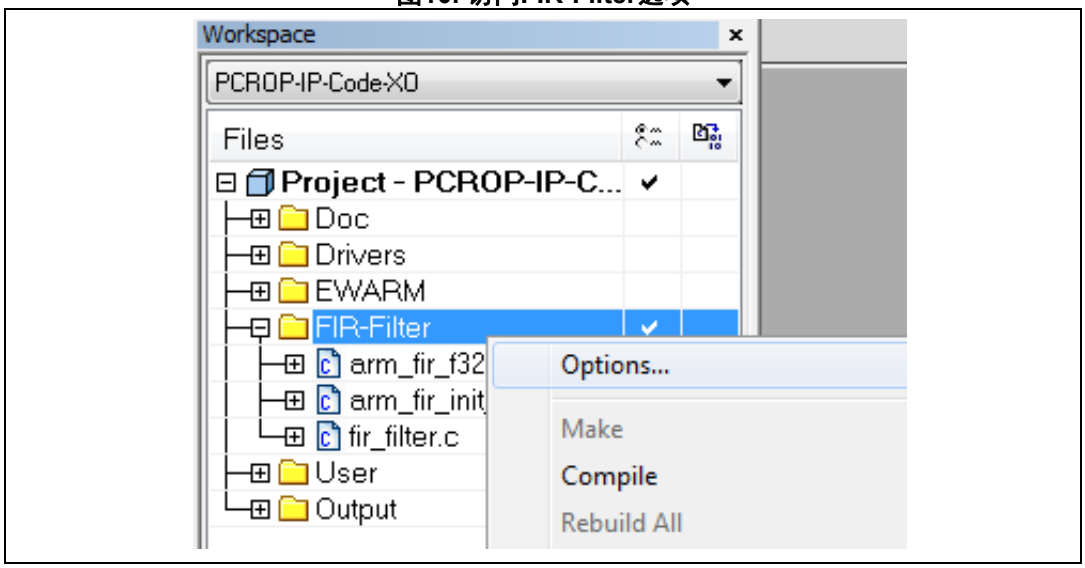
必须更新分散文件，才能将IP-Code文件设置为仅执行代码，如下所示

```
LR_PCROP 0x08008000 0x00004000  {
  ER_PCROP 0x08008000 0x00004000  { ; load address = execution address
        arm_fir_f32.o (+XO)
        arm_fir_init_f32.o (+XO)
        FIR_Filter.o (+XO)
        }
}
```

IAR：代码内存中不进行数据读取

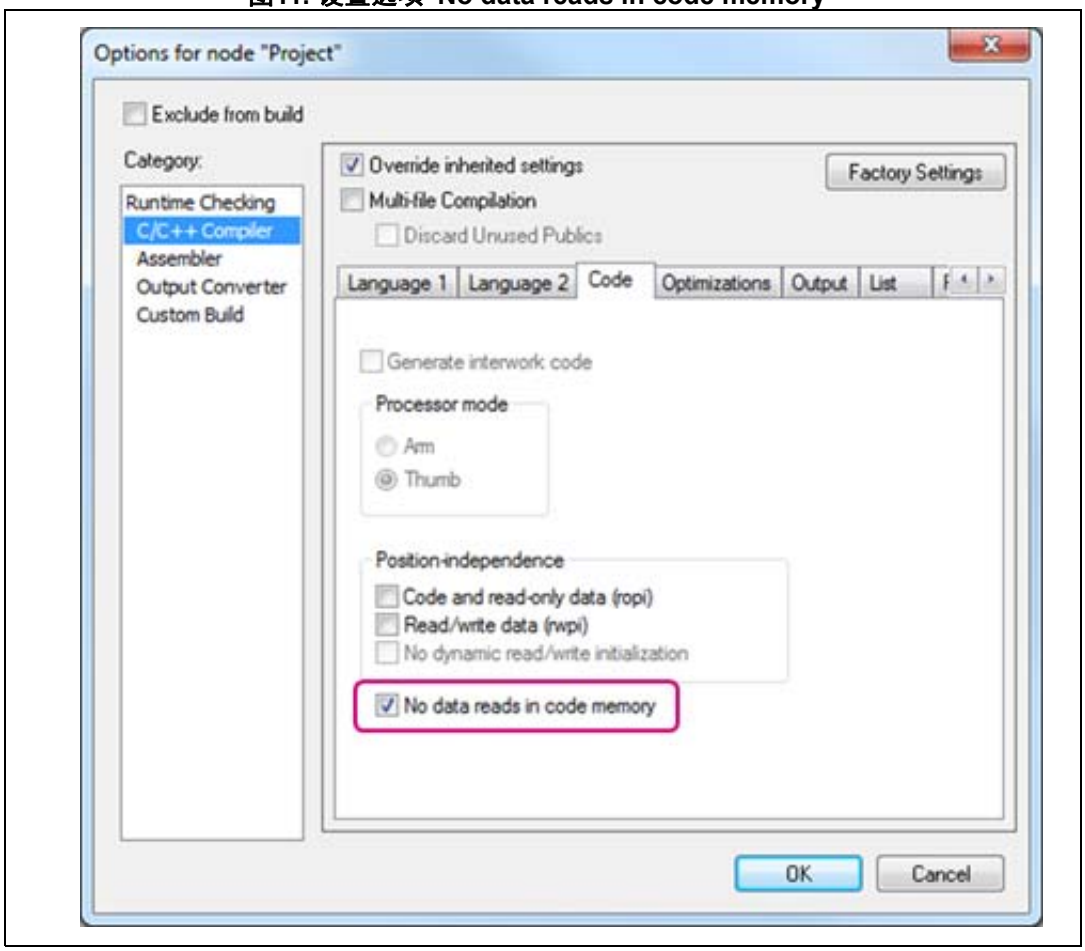
对于IAR，必须使用选项“No data reads in code memory”来防止编译器对代码扇区生成任何数据访问。要激活此选项，请右键单击包含IP-Code源文件的文件组“*FIR-Filter*”，然后（见*图 10*）选择*Options*。

图10. 访问FIR-Filter选项



然后，在以下窗口中，选择选项“*No data reads in code memory*”，如*图 11*中所示。

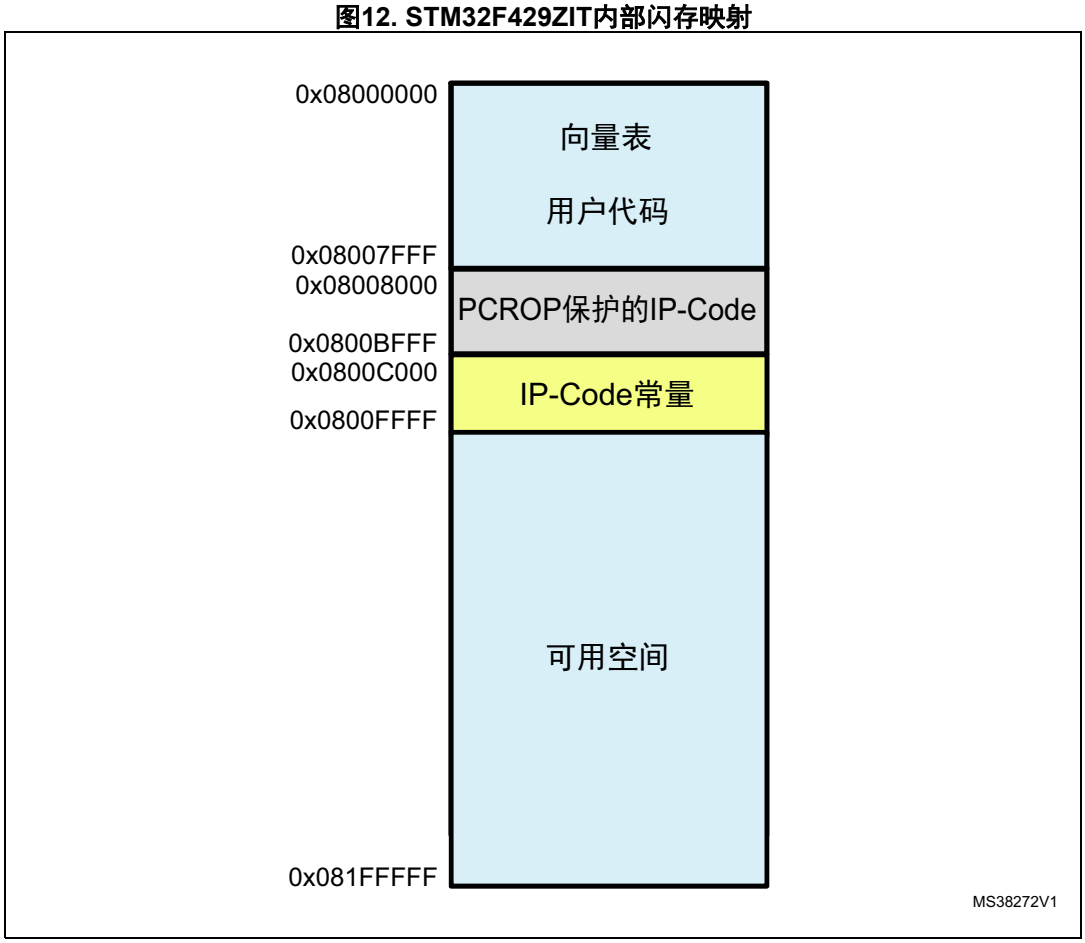
图11. 设置选项“No data reads in code memory”



4.4.2 存放IP-Code

如前所述，PCROP保护的代码不能在中断向量表所在的第一个扇区中。这就是为什么我们要把IP-Code放在内部闪存的Sector 2 BANK1中，而用户代码和向量表则从第一个扇区开始存放，如*图 12*所示。

图12. STM32F429ZIT内部闪存映射



Keil®分散文件

要将IP-Code放入Sector 2，需按照以下顺序：

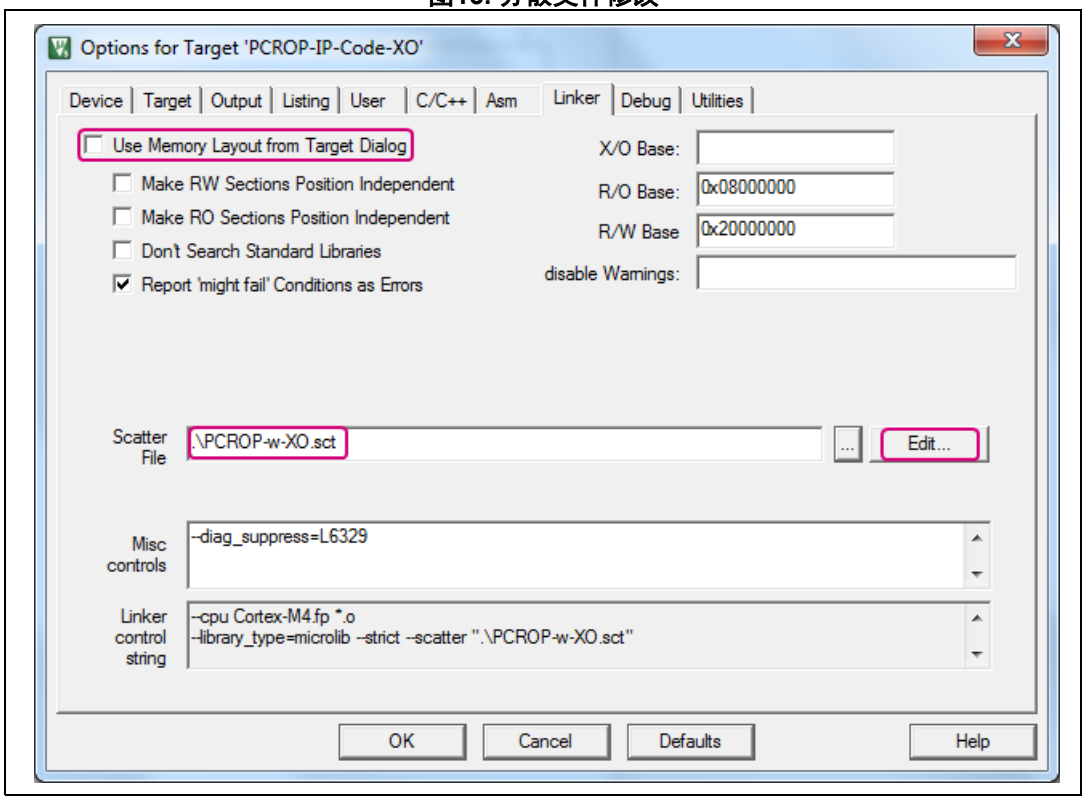
进入*Project* ➞ *Options for Target*，选择连接器选项卡，然后从*Target Dialog*中取消选中*Use Memory Layout*，如*图 13*中所示，然后点击*Edit*按钮来修改 PCROP-w-XO.sct分散文件。

然后打开分散文件。IP-Code对象文件应放在名为LR_PCROP的新的专用加载区域中，如下所示：

```
LR_PCROP 0x08008000 0x000040000  {  ; IP Code dedicated load region
  ER_PCROP 0x080080000 0x00004000  {
        arm_fir_f32.o (+XO)
        arm_fir_init_f32.o (+XO)
        FIR_Filter.o (+XO)
```

```
        }
}
```

图13. 分散文件修改



IAR ICF文件

要使用IAR将IP-Code放在Sector 2中，请打开ICF文件并添加新的加载区域，如下所示：

define symbol __ICFEDIT_region_PCROP_start__ = 0x08008000;
define symbol __ICFEDIT_region_PCROP_end__ = 0x0800BFFF;

define region PCROP_region = mem:[from __ICFEDIT_region_PCROP_start__ to __ICFEDIT_region_PCROP_end__];

place in PCROP_region { ro object arm_fir_f32.o,
ro object arm_fir_init_f32.o,
ro object FIR_Filter.o};

4.4.3 将常量放在PCROP保护扇区之外

所有IP-Code常量(例如，整数，浮点数或字符串）必须放在PCROP保护的区域之外，因为它们不能被D-Code总线进行访问。

链接器文件可用来将常量放在特定内存区域中，如以下示例所示。

示例：IAR ICF文件：

```
define symbol __ICFEDIT_region_CONST_start__      = 0x0800C000;
define symbol __ICFEDIT_region_CONST_end__      = 0x0800FFFF;
define region CONST_region      = mem:[from __ICFEDIT_region_CONST_start__   to
__ICFEDIT_region_CONST_end__];
place in CONST_region {section .rodata};
```

对于IAR使用@ *operator*定义常量时，以及对于Keil®使用*__attribute__((at(address)))*定义常量时，另一种方法是直接将常量放在用户代码中的固定地址。本例中使用了这个方法，在*fir_filter.c*文件中进行常量定义，类似于下面的例子。

利用Keil®，将常量放在固定地址

```
const int k __attribute__((at(0x0800C000))) = 500; /* 将常量k放在闪存的0x0800C000 */
```

利用IAR，将常量放在固定地址

```
const int k @ 0x0800C000 = 500; /* 将常量k放在闪存的0x0800C000 */
```

注意： 存放IP-Code常量时，用户应该考虑到这些常量可能会被ST用户级别n + 1删除或修改，因此IP-Code函数将会无效。因而建议将这些常量放在专用扇区，其中不编程用户代码。然后这个内存区域应在提供给ST用户级别n+1的闪存映射中突出显示。

4.4.4 保护IP-Code

使用*PCROP_Enable()*函数激活PCROP

使用*Step1-ST_Customer_level_n project main.c*文件中定义的*PCROP_Enable()*函数，激活Sector 2上的PCROP，从而实现IP-Code保护

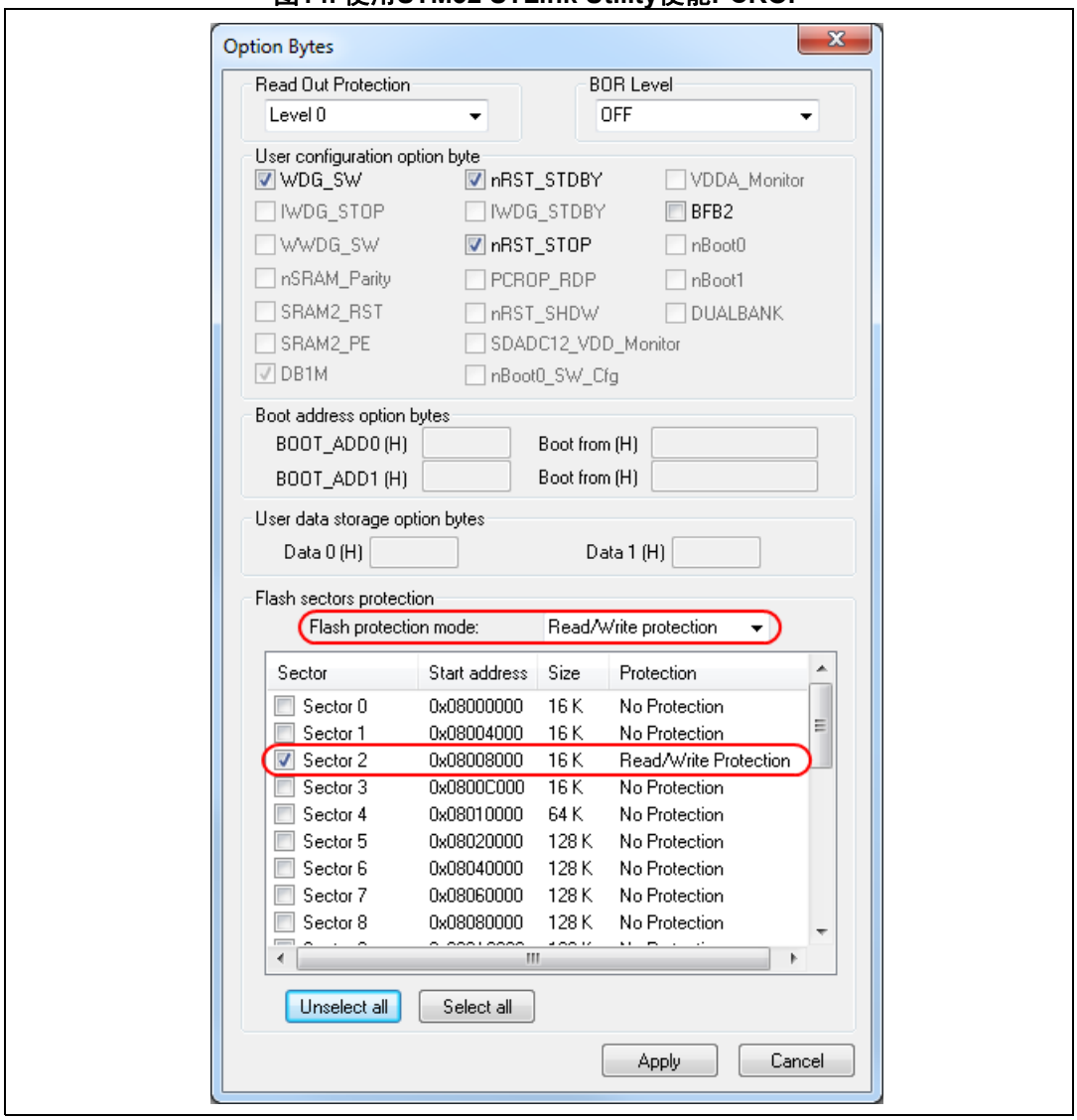
当此函数至少执行了一次时，Sector 2变为读/写保护的，任何IP-Code修改都需要禁用PCROP，然后全部闪存内容都会被擦除。

使用STM32 STLink Utility激活PCROP

要利用STM32 STLink激活Sector 2上的PCROP，用户必须按照下面所示的顺序：

- 板子上电，然后在STLink Utility接口进入*Target* ➞ *Option bytes*。
- 下面的窗口中，将闪存保护模式设置为读/写保护，并使能Sector 2上的保护，如*图 14*中所示，然后点击*Apply*按钮。

图14. 使用STM32 STLink Utility使能PCROP

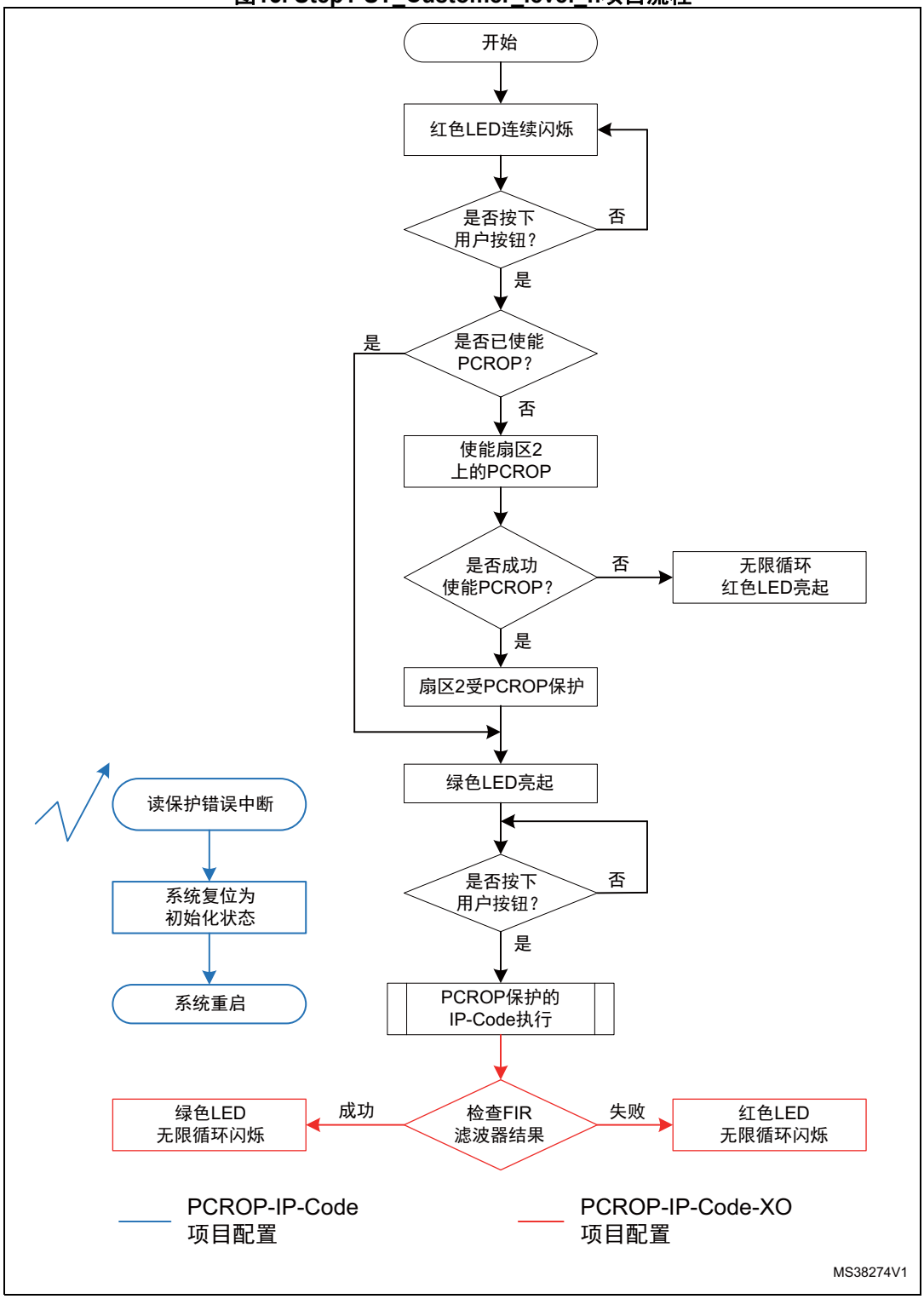


4.4.5 Step1-ST_Customer_level_n项目流程

*图 15*说明了采用两种项目配置的Step1-ST_Customer_level_n项目流程：

- PCROP-IP-Code-XO（红色表示）：PCROP保护的IP-Code不包含任何文字池，然后FIR滤波器算法成功运行，注意如果从/向PCROP保护区域发送读/写请求，则会产生一个读错误中断并且系统复位。
- PCROP-IP-Code（蓝色表示）：IP-Code包含文字池，当开始执行PCROP保护的IP-Code时，会产生一个读操作错误中断，然后启动系统复位，系统重新开始。

图15. Step1-ST_Customer_level_n项目流程



4.4.6 执行PCROP保护的IP-Code

一旦IP-Code被编程在Sector 2上并且受到保护，则必须通过调用用户代码中的函数来对其进行测试。

本章我们将介绍，如何执行具有两种配置的*Step1-ST_Customer_level_n*项目中PCROP保护的IP-Code，注意PCROP-IP-Code配置仅用于测试，不能用于STEP2。PCROP-IP-Code-XO才是正确的配置，这里编译器被设置为产生只执行IP-Code。这是用来运行*Step2-ST_Customer_level_n+1*项目的配置。

PCROP-IP-Code-XO（必须在STEP2之前使用）

编译器配置为生成只执行IP-Code，避免对其进行数据读取（避免文字池）。

1. 打开位于*Step1-ST_Customer_level_n*目录中的项目，选择您首选的工具链
2. 选择PCROP-IP-Code-XO配置
3. 重建所有文件。
4. 按照以下顺序运行示例：
   a) 为板子上电，在载入代码之前，检查是否存在任何PCROP保护或写保护的扇区。如果有，则使用STM32 STLink Utility禁用该保护，然后载入代码。程序加载完毕后，红色LED应连续闪烁；
   b) 按下用户按钮键来激活PCROP保护，当此完成时，绿色LED会亮起，Sector 2被PCROP保护，否则红色Led亮起，PCROP激活失败；
   c) 按下用户按钮键来执行在*main.c*文件中调用的PCROP保护IP-Code，绿色LED应连续闪烁。

PCROP-IP-Code（仅用于测试，不能用于STEP2）

没有使用特殊的编译器选项，仅用于测试目的，说明对于PCROP保护的代码，必须避免代码中的数据（如文字池和分支表）。

1. 在位于*Step1-ST_Customer_level_n*目录的同一个项目中，选择*PCROP-IP-Code*配置
2. 重建所有文件。
3. 按照以下顺序运行示例：
   a) 为板子上电，在载入代码之前，检查是否存在任何PCROP保护或写保护的扇区。如果有，则利用STM32 STLink Utility禁用该保护，然后载入代码；当程序加载完毕时，红色LED应连续闪烁；
   b) 按下用户按钮键来激活PCROP保护，当此完成时，绿色LED会亮起，Sector 2被PCROP保护，否则红色LED亮起，PCROP激活失败；
   c) 按下用户按钮键来执行在*main.c*文件中调用的PCROP保护IP-Code，会产生一个错误操作中断，启动系统复位，并且红色LED连续闪烁。

解释

低通滤波器函数计算*testInput_f32_1kHz_15kHz*输入信号，并输出1 KHz正弦波。输出数据*testOutput*随后与MATLAB已计算出的参考*refOutput*进行比较，如果二者匹配，则绿色LED连续闪烁，否则红色LED连续闪烁。

对于PCROP-IP-Code配置，其中PCROP保护的IP-Code包含文字池：执行IP-Code（FIR_lowpass_filter()函数）时，不能通过D-Code总线访问文字池，然后RDERR标志置位。OPERR标志也被置位，并产生一个读操作错误中断，然后在*HAL_FLASH_OperationErrorCallback()*函数中启动系统复位，红色LED连续闪烁。

但是对于PCROP-IP-Code-XO配置，IP-Code可正确执行，绿色LED会连续闪烁。

注： *更多详细内容，请参考固件包中的readme.txt。*

4.4.7 创建头文件并生成符号定义文件

要提供给ST用户级别n+1的头文件包含要使用的IP-Code函数定义。本例中包含在*main.c*文件中的是*fir_filter.h*文件。

利用IAR产生符号定义文件

IDE中，要导出PCROP保护的IP-Code符号，请选择*Project*➞*Options*➞*Build Actions*，并在Post-build命令行文本字段中指定以下命令行：

```
$TOOLKIT_DIR$\bin\isymexport.exe --edit "$PROJ_DIR$\steering_file.txt" "$TARGET_PATH$" "$PROJ_DIR$\fir_filter.o"
```

这里，创建*steering_file.txt*用作选项，使生成的符号文件中只有IP-Code函数符号。

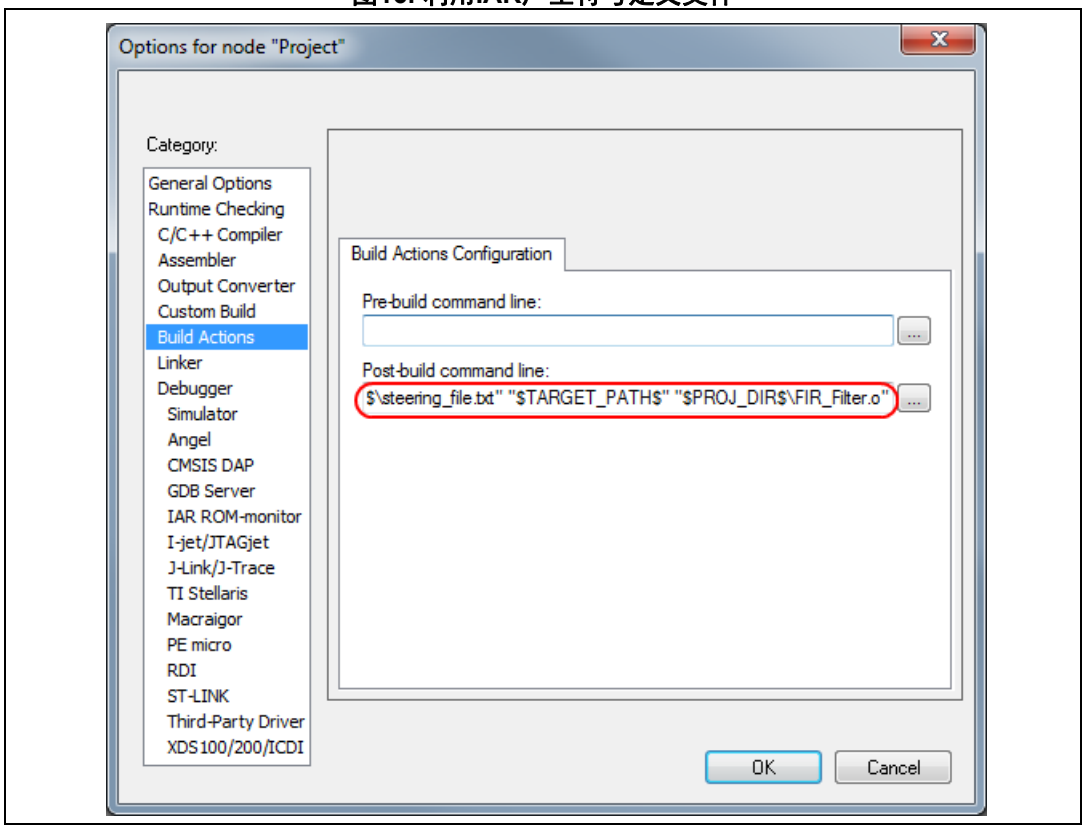
要使符号文件中只有IP-Code函数，则必须创建*steering_file.txt*，如下所示：

```
show arm_fir_f32
show arm_fir_init_f32
show FIR_lowpass_filter
```

这里“show”命令用来保留所生成符号定义文件中的首选函数。

所生成符号定义文件fir_filter.o可通过将其添加到STEP2-ST_Customer_level_n+1 IAR项目中来使用。

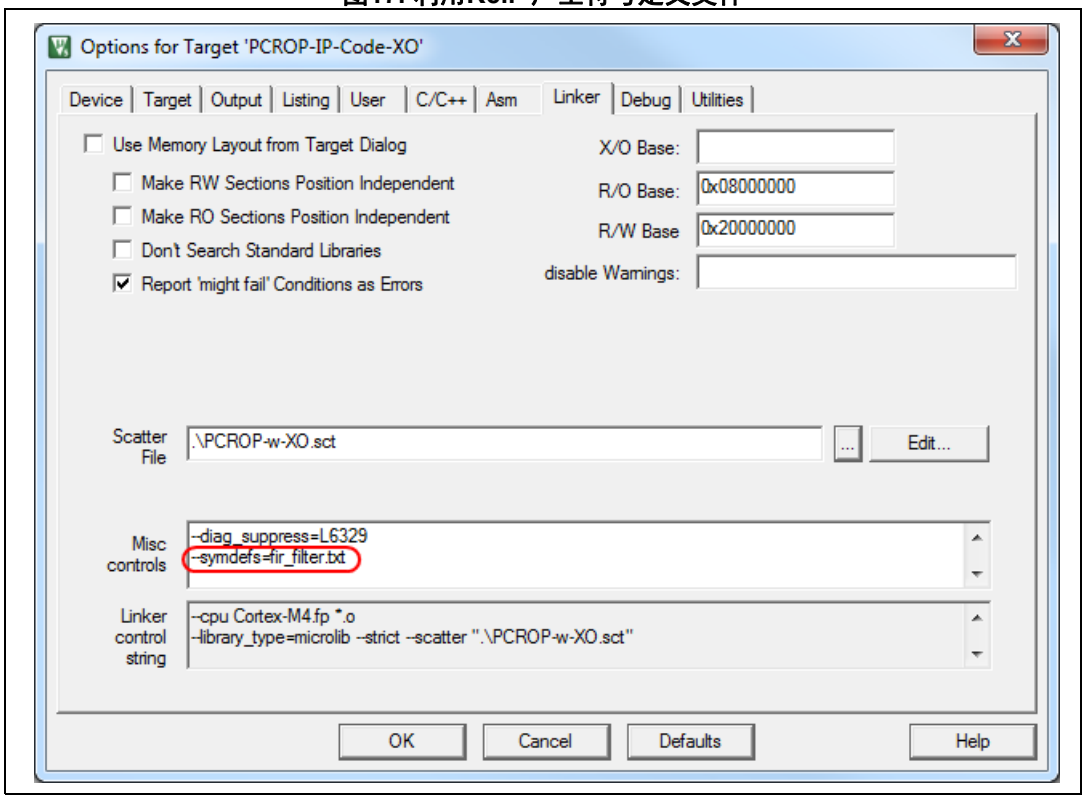
图16. 利用IAR产生符号定义文件



利用Keil®产生符号定义文件

要利用Keil®生成符号定义文件，请进入Linker选项卡中的Project选项，添加命令--*symdefs=fir_filter.txt*并重新构建项目。

图17. 利用Keil®产生符号定义文件



名为*fir_filter.txt*的符号定义文件随后会在Step1-ST_Customer_level_n\MDK-ARM\PCROP-IP-Code-XO目录中创建。

该文件包含项目的所有符号，因此只能保留最终用户所调用的那些IP-Code函数。所有其他函数都应该被删除，然后产生的符号定义文件为

符号定义文件fir_filter.txt：

#<SYMDEFS># ARM Linker, 5050106: Last Updated: Sat May 16 20:25:52 2015

0x08008001 T FIR_lowpass_filter

0x0800804d T arm_fir_f32

0x08008483 T arm_fir_init_f32

4.5 STEP2：ST用户级别n+1

ST用户级别n+1具有来自用户级别n的，预加载PCROP保护的IP-Code的STM32F429ZIT MCU，提供的符号定义文件以及头文件。

然后参考由ST用户级别n提供的闪存映射，ST用户级别n+1应：

- 创建最终项目；
- 在其项目中包含ST用户级别n提供的的头文件以及添加其提供的符号定义文件；
- 调用PCROP保护的IP-Code函数；
- 执行并调试最终用户应用程序。

注意： ST用户级别n+1必须使用与ST用户级别n完全相同的工具链和编译器版本，来开发和编程IP-Code，否则ST用户级别n+1不能使用该IP-Code。
在所提供的示例中，对于两个项目STEP1-ST_Customer_level_n和STEP2-ST_Customer_level_n+1，用户必须使用同样的工具链和编译器版本。
例如，如果使用了MDK-ARM工具链V5.14来运行STEP1-ST_Customer_level_n项目，则必须也使用它来运行STEP2-ST_Customer_level_n+1项目。

4.5.1 创建一个最终用户项目

ST用户级别n+1将根据*表 4*中所述的闪存映射来存放其代码。

Sector 2（灰色线）不可用，因为它是PCROP保护的，而Sector 3（浅黄色线）不能使用，因为它包含IP-Code常量。因此，在执行链接器文件时，用户必须小心，避免将任何代码放在这些区域。

表4. 预编程STM32F429ZIT内部闪存映射

扇区编号 (Sector number)	扇区大小	起始地址	内容
扇区0	16 KB	0x08000000	可提供
扇区1	16 KB	0x08004000	
扇区2	16 KB	0x08008000	PCROP保护的 IP-Code
扇区3	16 KB	0x0800C000	IP-Code常量
扇区4 ... ... 扇区11	16 KB ... ... 128 KB	0x08000800 ... ... 0x080E0000	可提供
扇区12	16 KB	0x08100000	
... ...	... ...	... ...	
扇区22	128 KB	0x081C0000	
扇区23	128 KB	0x081E0000	

PCROP保护的IP-Code函数随后用来创建最终用户应用程序。位于Step2-ST_Customer_level_n+1目录中的项目是一个示例，其中PCROP保护的FIR Filter函数在*main.c*文件中被调用。

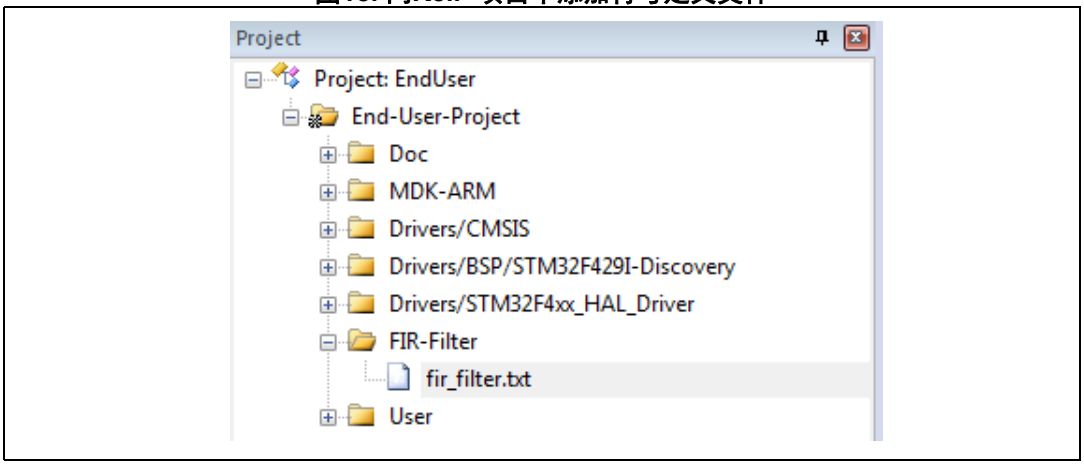
4.5.2 包含头文件并添加符号定义文件

ST用户级别n所提供的符号定义文件应添加为传统源文件，这是在链接Step2-ST_Customer_level_n+1项目中的PCROP保护IP-Code所必需的。

向Keil®项目中添加符号定义文件

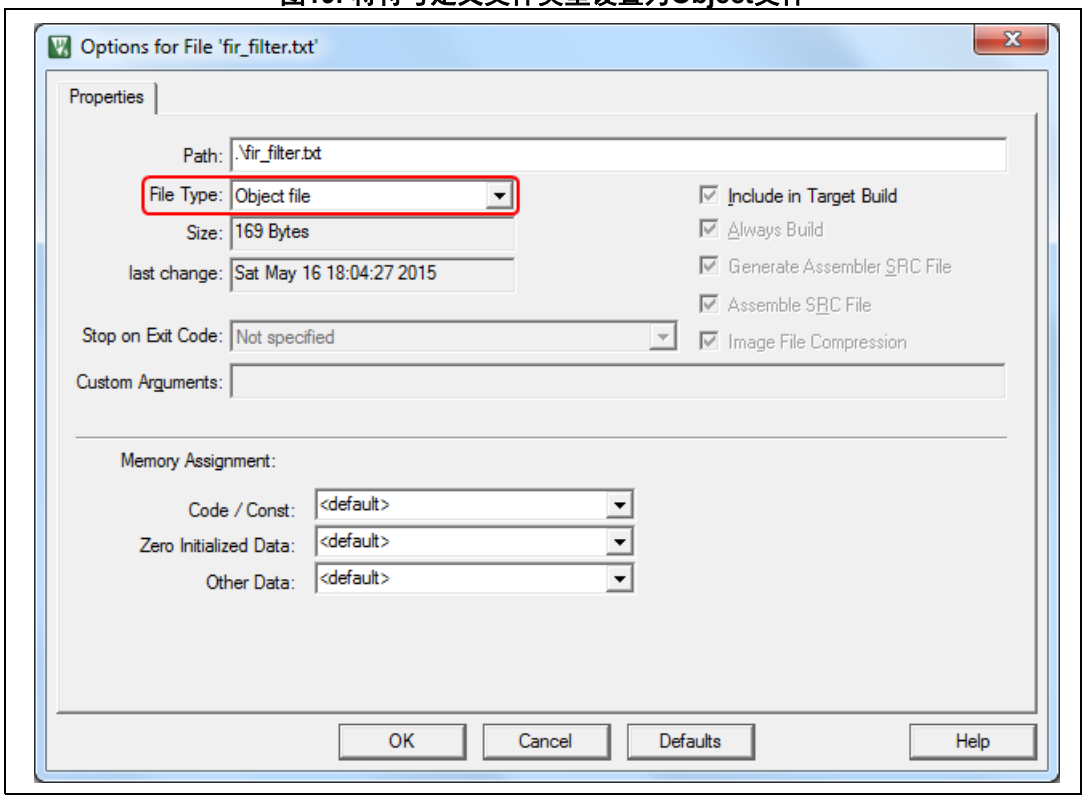
符号定义文件由*第 4.4.7 节*中所述的ST用户级别n提供，本例中名为*fir_filter.txt*，它必须添加到FIR-Filter组中，如*图 18*中所述。

图18. 向Keil®项目中添加符号定义文件



所添加的*fir_filter.txt*文件类型必须改为Object文件，而不是文本文档文件，如*图 19*中所示。

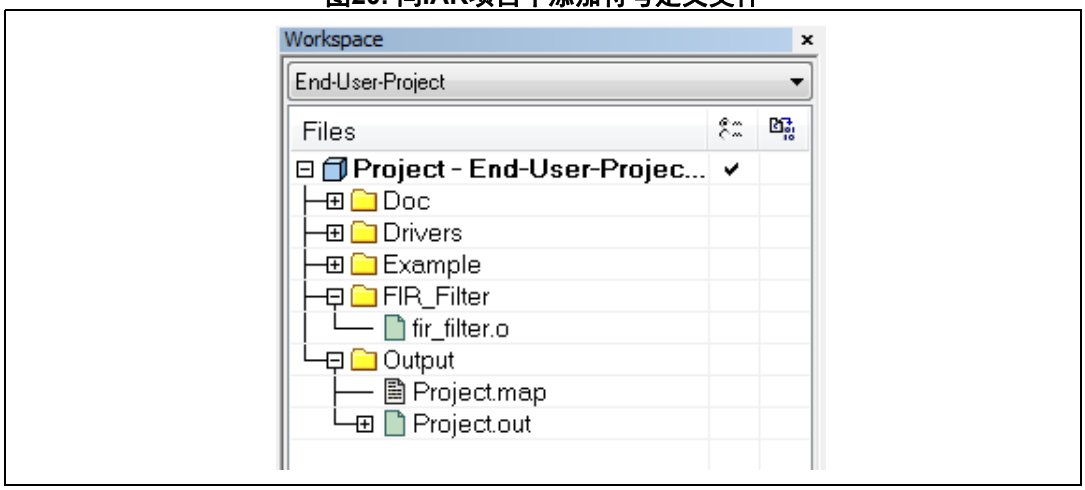
图19. 将符号定义文件类型设置为Object文件



向IAR项目中添加符号定义文件

必须将*fir_filter.o*文件作为目标文件添加到组*FIR_Filter*中，如*图 20*中所示。

图20. 向IAR项目中添加符号定义文件



包含头文件

头文件*fir_filter.h*包含在*main.c*文件中，然后用户可以调用FIR Filter函数。

4.5.3 调用PCROP保护的IP-Code函数

当*fir_filter.h*头文件包含在*main.c*文件中，并且符号定义文件已添加到项目中时，可调用PCROP保护的IP-Code函数。

*FIR_lowpass_filter()*函数（在fir_filter.c文件中有描述）在主文件中被调用，如下所示：

FIR_lowpass_filter(inputF32, outputF32, TEST_LENGTH_SAMPLES)

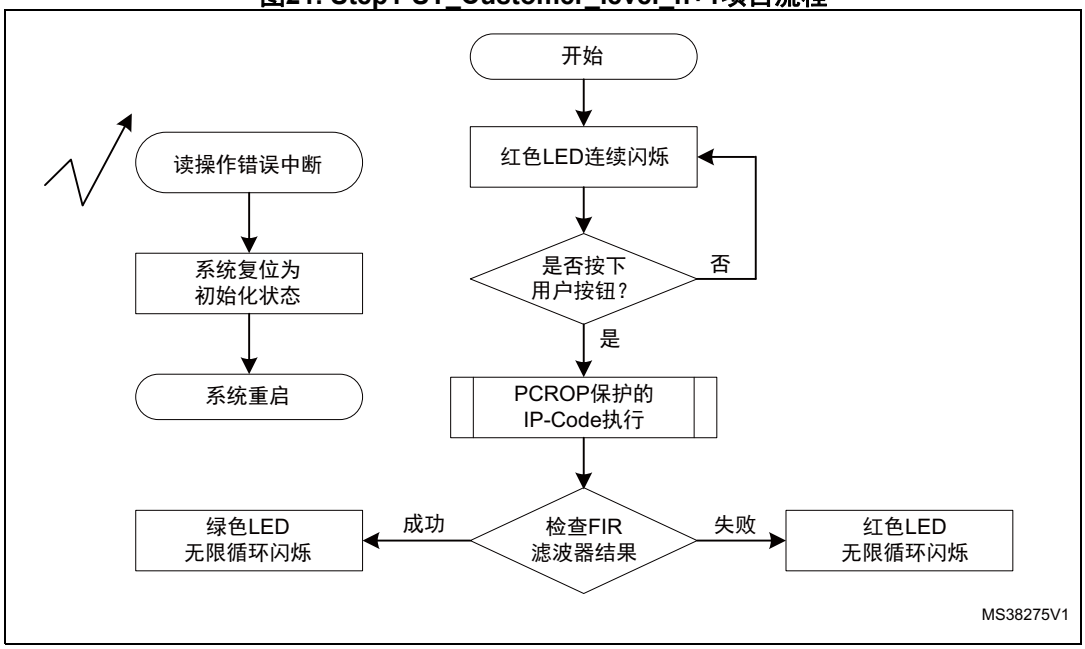
其中：

- *inputF32*：指针，指向包含输入信号数据的表；
- *outputF32*：指针，指向已处理输出信号数据所填充的表；
- *TEST_LENGTH_SAMPLES*：待处理输入中的采样数。

4.5.4 Step2-ST_Customer_level_n+1项目流程

*图 21*说明了Step2-ST_Customer_level_n+1项目流程。如果从/向PCROP保护的IP-Code发送任何读/写请求，则会产生一个读操作错误中断，启动系统复位，然后系统重启，红色LED连续闪烁。

图21. Step1-ST_Customer_level_n+1项目流程



4.5.5 运行最终用户应用程序

在此阶段IP-Code已经由STM32F429ZIT MCU进行预加载并PCROP保护，因此Step1-ST_Customer_level_n项目（PCROP-IP-Code-XO配置）必须在运行该项目前进行加载和执行。

要使程序工作，请按照下述步骤操作：

1. 打开位于Step2-ST_Customer_level_n+1目录中的项目，选择与STEP1中相同的工具链。
2. 重建所有文件。
3. 按照以下顺序运行示例
   a) 为板子上电，并加载代码（这里仅加载用户代码）
   b) 按下用户按钮键来执行在*main.c*文件中调用的PCROP保护IP-Code，绿色LED应连续闪烁。

注： *更多详细内容，请参考固件包中的readme.txt。*

4.5.6 调试模式中的PCROP保护

在开发其最终用户应用程序时，ST用户级别n+1需要调试其代码。因此，在调试最终用户应用程序时，PCROP保护会阻止任何对ST用户级别n所开发的IP-Code的读访问。

本章中，我们将介绍在调试用户代码时，PCROP如何保护预编程IP-Code不被读取。

下述调试示例在Keil®项目上完成。

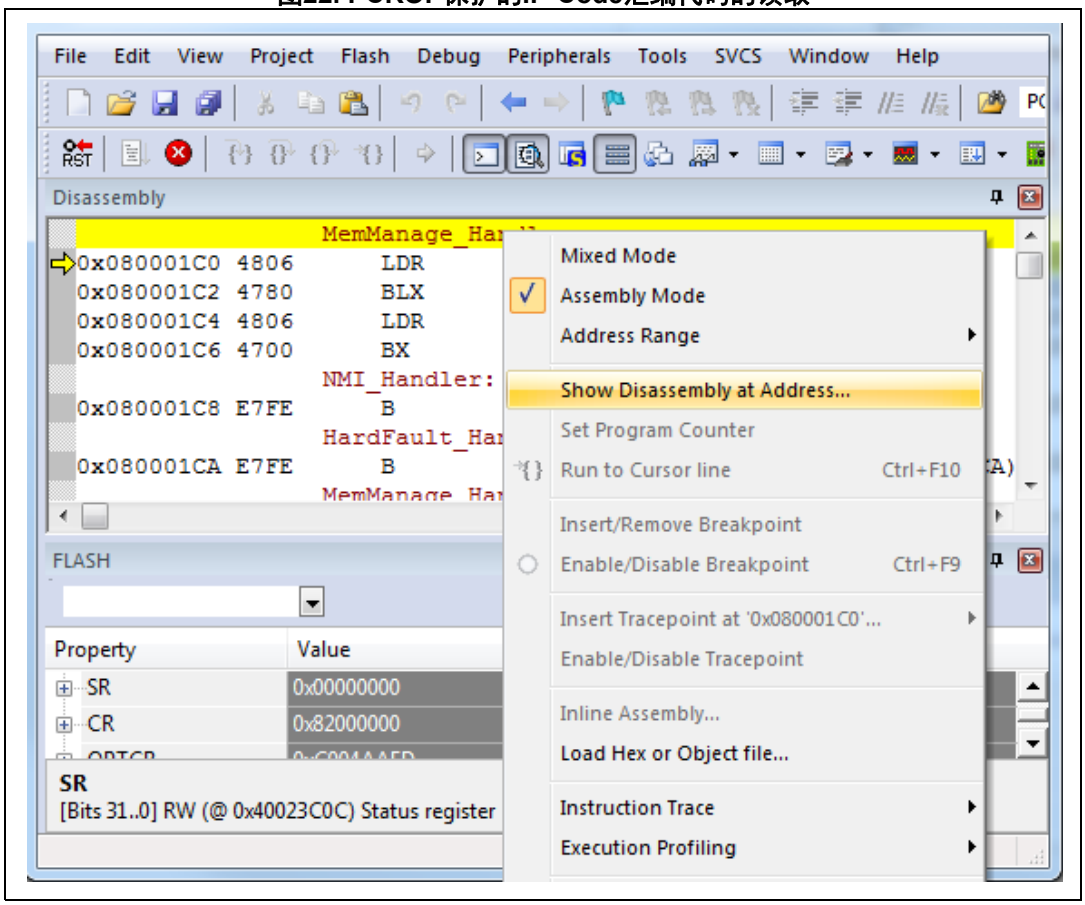
调试最终用户应用程序

调试最终用户应用程序之前，Step1-ST_Customer_level_n项目（PCROP-IP-Code-XO工作空间）必须加载并执行，以使STM32F429ZIT具有预编程且PCROP保护的IP-Code。

要调试Step2-ST_Customer_level_n+1项目，请按照下述步骤操作：

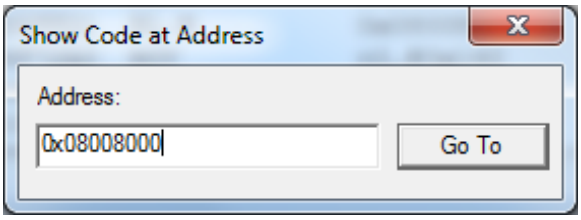
1. 打开位于Step2-ST_Customer_level_n+1目录中的项目，选择与STEP1中相同的工具链。
2. 重建所有文件。
3. 点击Start/Stop Debug会话来开始调试。
4. 点击Run来运行程序，红色LED会连续闪烁。
5. 按下User按钮来执行IP-Code，绿色LED会连续闪烁。
6. 要访问PCROP保护的IP-Code，右键点击Disassembly窗口，然后选择Show Disassembly at Address，如*图 22*中所示。

图22. PCROP保护的IP-Code汇编代码的读取



7. 在地址栏填写PCROP保护扇区起始地址，并单击**Go To**按钮，如*图 23*中所示。

图23. 填写PCROP保护扇区起始地址

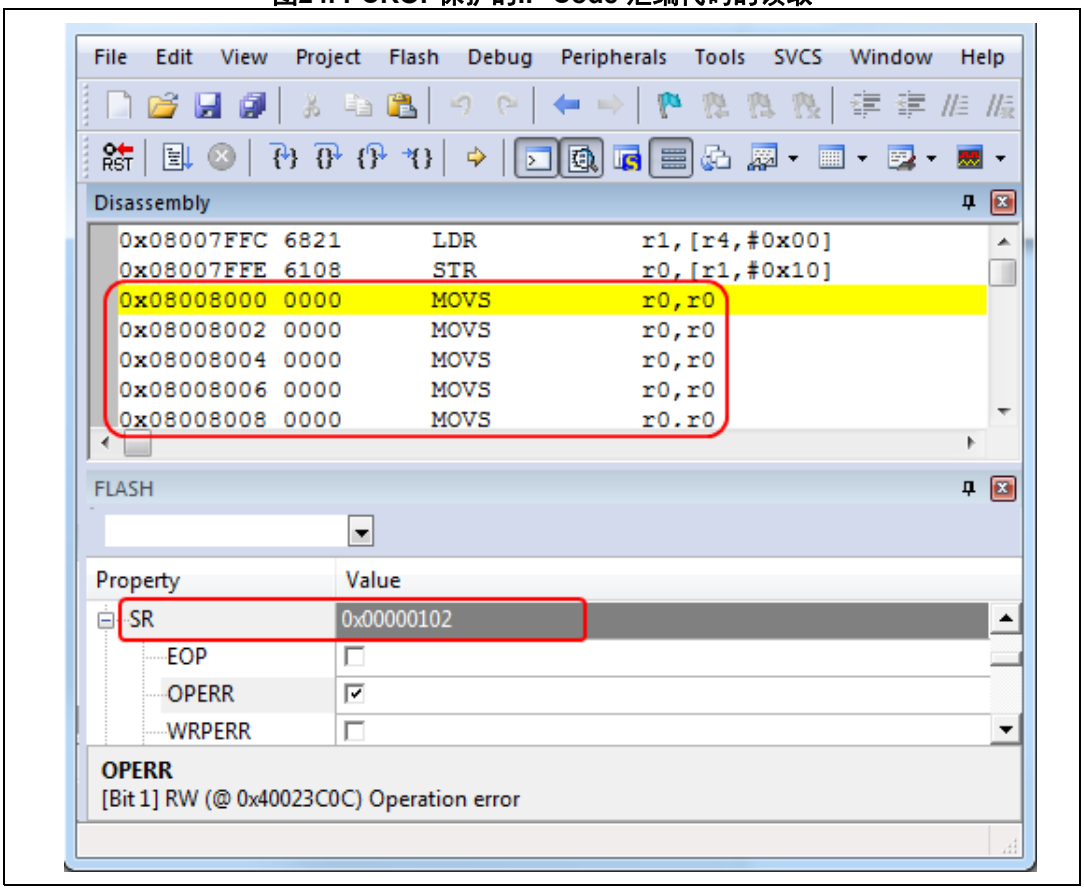


如*图 24*中所示，载入到Sector 2中的PCROP保护IP-Code是不可读的，而位于0x08008000地址之前的代码可以被读取。读取PCROP保护扇区会将FLASH_SR寄存器中的RDERR和OPERR标志置位。

由于调试时通过D-Code总线进行了闪存读操作，因此会产生一个闪存操作错误中断。

然后在*HAL_FLASH_OperationErrorCallback()*函数（在*main.c*文件中有描述）中启动软件Reset，Red LED4会连续闪烁。

图24. PCROP保护的IP-Code 汇编代码的读取



5 结论

STM32F4系列微控制器提供了灵活有用的读和/或写保护功能，可用于需要保护的应用。

本应用笔记说明了如何使用STM32F4xx MCU提供的读、写和PCROP保护功能。

6 版本历史

表5. 文档版本历史

日期	版本	变更
2015年7月28日	1	初始版本。
2015年10月12日	2	增加了STM32F469/479产品，因此更新了*前言*。 更新了*Keil®分散文件*和*调试最终用户应用程序*。
2016年11月29日	3	增加了STM32F410、STM32F412和STM32F413产品，因此更新了*前言*并增加了*表 1：参考文档*。

表6. 中文文档版本历史

日期	版本	变更
2017年8月10日	1	中文初始版本。

重要通知 - 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。本文档的中文版本为英文版本的翻译件，仅供参考之用；若中文版本与英文版本有任何冲突或不一致，则以英文版本为准。