AN4758
应用笔记

STM32L4、STM32L4+和STM32G4系列微控制器上的专利代码读取保护

引言

软件提供商正在开发被称为IP（知识产权）代码的复杂中间件解决方案，保护它们对微控制器而言是一个非常重要的问题。

为了满足这一重要要求，STM32L4、STM32L4 +和STM32G4系列MCU可提供以下保护功能：

- 读取保护（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
- 专利代码读取保护（PCROP）：防止在闪存和SRAM存储器上进行读写操作。
- 防火墙：针对外部进程为敏感代码和数据提供访问保护。

本应用笔记对这些闪存保护功能进行了说明，重点介绍了专利代码读取保护（PCROP），并提供了PCROP保护的基本示例。防火墙保护（在STM32L4和STM32L4 +系列上可用）在www.st.com上的“*STM32L0 / L4防火墙概述*”（AN4729）中进行了介绍。

本文档随附的X-CUBE-PCROP固件封装包含了PCROP示例的源代码，以及基于STM32L4系列微控制器运行示例所需的所有固件模块，并且该封装可轻松移植到STM32L4+和STM32G4系列微控制器上。

本应用笔记必须与产品数据手册以及以下参考手册一起阅读，这些参考手册可从www.st.com获得：

- RM0351（STM32L4x5xx、STM32L4x6xx）
- RM0392 (STM32L4x1xx)
- RM0394（STM32L43xxx、STM32L44xxx、STM32L45xxx、STM32L46xxx）
- RM0432（STM32L4Rxxx和STM32L4Sxxx）
- RM0440 (STM32G4xx)

目录

表格索引

图片目录

1 单分区存储器保护说明

基于Arm®[a]内核的STM32L4、STM32L4+和STM32G4系列微控制器采用多种机制，可对全存储器或特定段进行读写保护。

读保护用于保护代码免受外部访问的转储（SW IP保护），而写保护用于保护代码或数据不被意外擦除。除闪存外，这些保护还扩展到STM32L4和STM32L4+系列的SRAM2，以及STM32G4系列的CCM（内核耦合存储器）SRAM。

STM32L4xx MCU还具有防火墙机制，可在存储器中创建受信执行区域。

1.1 读取保护（RDP）

读取保护是全局闪存读保护，可保护嵌入式固件代码，可以预防复制、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。

读取保护适用于：

- 主闪存
- 实时时钟（RTC）中的备份寄存器
- SRAM2（STM32L4/STM32L4+）或CCM-SRAM（STM32G4）
- 选项字节（仅限级别 2）。

以下章节中对三个RDP级别（0，1和2）进行定义和描述。

1.1.1 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。在这种模式下没有保护，该模式可满足开发和调试需求。

1.1.2 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）分别访问（读取，擦除和编程）STM32L4/L4+和STM32G4系列的闪存或SRAM2和CCM-SRAM。在这些情况下，任何对受保护区域的读请求都会生成总线错误。

但是，当从闪存启动时，则允许从用户代码访问闪存和SRAM2（STM32L4/L4+）或CCM-SRAM（STM32G4）。

a. Arm是Arm Limited（或其子公司）在美国和/或其他地区的注册商标。

将RDP选项字节重新编程为级别0，可禁用RDP级别1保护，这会导致闪存被批量擦除；而且SRAM2（STM32L4/L4+）或CCM-SRAM（STM32G4）和备份寄存器会复位。

1.1.3 读保护级别2

激活RDP级别2时，级别1下提供的所有保护均有效，MCU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和边界扫描全部禁用。

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 1*根据保护级别和执行模式总结读取访问权限。

表1. 访问状态 vs 保护级别和执行模式

存储区	保护级别	用户执行（从Flash启动）			从RAM调试/启动/从加载程序启动		
		读取	Write	擦除	读取	Write	擦除
主要 Flash	1 级	有			无		
	级别 2	有			NA(1)		
系统存储器	1 级	有	无		有	无	
	级别 2	有	无		NA(1)		
选项字节	1 级	有			有		
	级别 2	有	无		NA(1)		
备份寄存器	1 级	有		NA(1)	无		
	级别 2	有		NA(1)	NA(1)		
SRAM2 或 CCM-SRAM	1 级	有		NA(1)	无		
	级别 2	有		NA(1)	NA(1)		

1. NA：不适用

1.1.4 受RDP保护的STM32内部闪存内容更新

当Flash 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，或者当从SRAM或系统内存引导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解决，使用诸如USART的通信协议来进行重新编程过程，从而执行内部闪存的应用内编程（IAP）。

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。

1.2 写保护

写保护用来保护指定内存区域的内容，避免更新或擦除代码段或非易失性数据。

1.2.1 闪存写保护

写保护区域的数量取决于闪存架构。

对于STM32L4和STM32L4+系列，每个闪存中可以以2KB粒度定义最多2个区域。

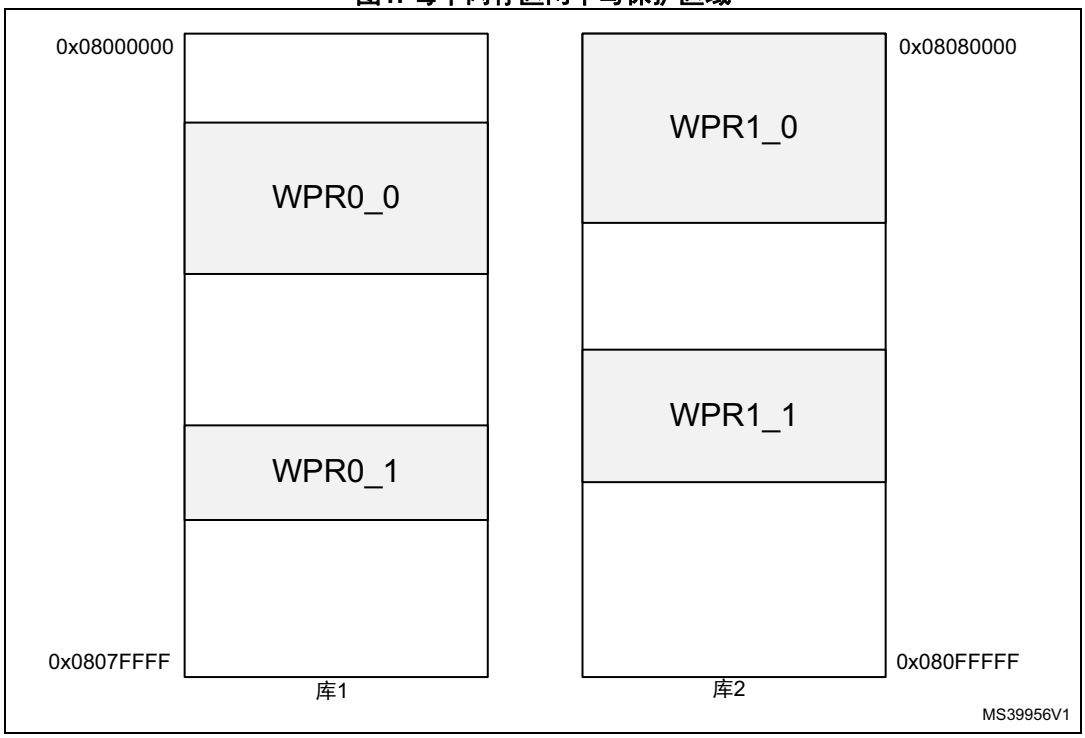
STM32G4 3类设备能够以单分区或双分区工作。

- 在单分区模式（DBANK = 0）中，最多能够以4 KB的粒度定义四个写保护区域。
- 在双分区模式（DBANK = 1）中，最多可以定义两个写保护区域
  每个存储库中2 KB的粒度。

STM32G4 Cat2设备只能在单个闪存库中工作。能够以2 KB粒度定义两个写保护区域。

*图 1*中的灰色区域是具有两个粒度为2 KB的写保护（WRP）区域的双分区结构的示例。

图1. 每个闪存区两个写保护区域



受保护区域无法被擦除和编程，任何写请求都会产生写保护错误。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页是写保护的，则不能对其进行批量擦除，并且设置WRPERR标志。

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。

1.2.2 SRAM2 CCM-SRAM写保护

在STM32L4/L4+上，32KB的SRAM2可以通过1KB页面单独进行写保护。该保护的设置由32位系统配置寄存器进行控制，并在启用后，只有系统复位才能对其进行禁用。

在STM32G4中，CCM-SRAM也可以通过1KB的段进行写保护（3类设备为32KB，2类设备为10 KB）。

1.3 专利代码读取保护（PCROP）

1.3.1 PCROP保护概述

PCROP是用于闪存中IP-Code的读写保护。它可以防止专利代码可能被最终用户代码、调试器工具或RAM特洛伊木马程式修改或读取。

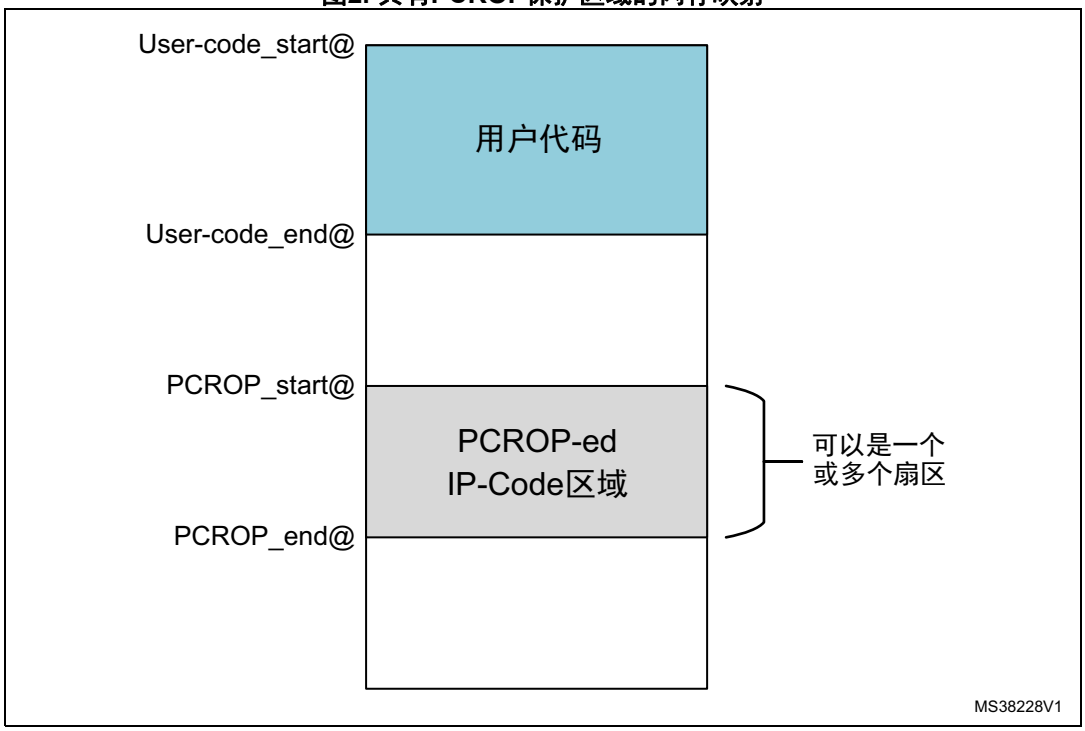
任何读或写请求都会产生一个读或写保护错误：

- 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保护区域执行读访问时，RDERR标志置位。
- 除了这些标志，如果通过FLASH_CR寄存器中的ERRIE位启用，还会引发中断。

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，而不会直接访问IP-Code本身。随后，PCROP不会阻止执行受保护的代码。

PCROP区域设置为具有8字节的精细粒度，因此不会浪费用于最终用户开发的闪存。

图2. 具有PCROP保护区域的闪存映射



1.3.2 如何启用PCROP保护？

对于STM32L4/L4 +系列，每个存储库中可以选择一个具有64位粒度的区域。

在STM32G4系列中，根据DBANK模式，每个库可以定义一个PCROP区域（在双分区模式下）或两个PCROP区域（适于所有存储器）。

每个PCROP区域均由与物理闪存库基址相关的起始页偏移量和结束页偏移量定义。

要激活PCROP，应在闪存选项字节寄存器中设置受保护区域的起始地址和结束地址：

- PCROP1SR：PCROP区域起始地址库1
- PCROP1ER：PCROP区域结束地址库1
- PCROP2SR：PCROP区域起始地址库2
- PCROP2ER：PCROP区域结束地址库2

*表 2*详细说明了寄存器值如何确定读取保护区域：

表2. 保护区域与寄存器值

PCROP寄存器值	PCROP 保护区
PCROPxSR = PCROPxER	存储库受到完整保护
PCROPxSR > PCROPxER	无PCROP区域（没有保护）
PCROPxSR < PCROPxER	PCROPxSR和PCROPxER之间的区域受到保护

另有一个选项位（PCROP_RDP = PCROP1ER [31]）可用来选择在RDP保护从级别1变为级别0时PCROP区域是否被擦除。为两个存储库设置该选项。

关于启用PCROP的更多详细内容，请参阅所提供的固件包（Step1-ST_Customer_level_n project main.c文件）中所述的PCROP_Enable()函数。

1.3.3 如何禁用PCROP保护？

仅当RDP级别为1或0时，才能禁用PCROP。当RDP设为级别2时，无法禁用PCROP；所有选项字节都会被冻结，不能再修改。因此，PCROP保护的区域不能再被擦除或修改，这样就成为永久性保护。

禁用受保护区域上的PCROP的唯一方法是将RDP从级别1降低到级别0，并同时停用已编程的区域（在嵌入式软件中，设置PCROPxSR> PCROPxER）。

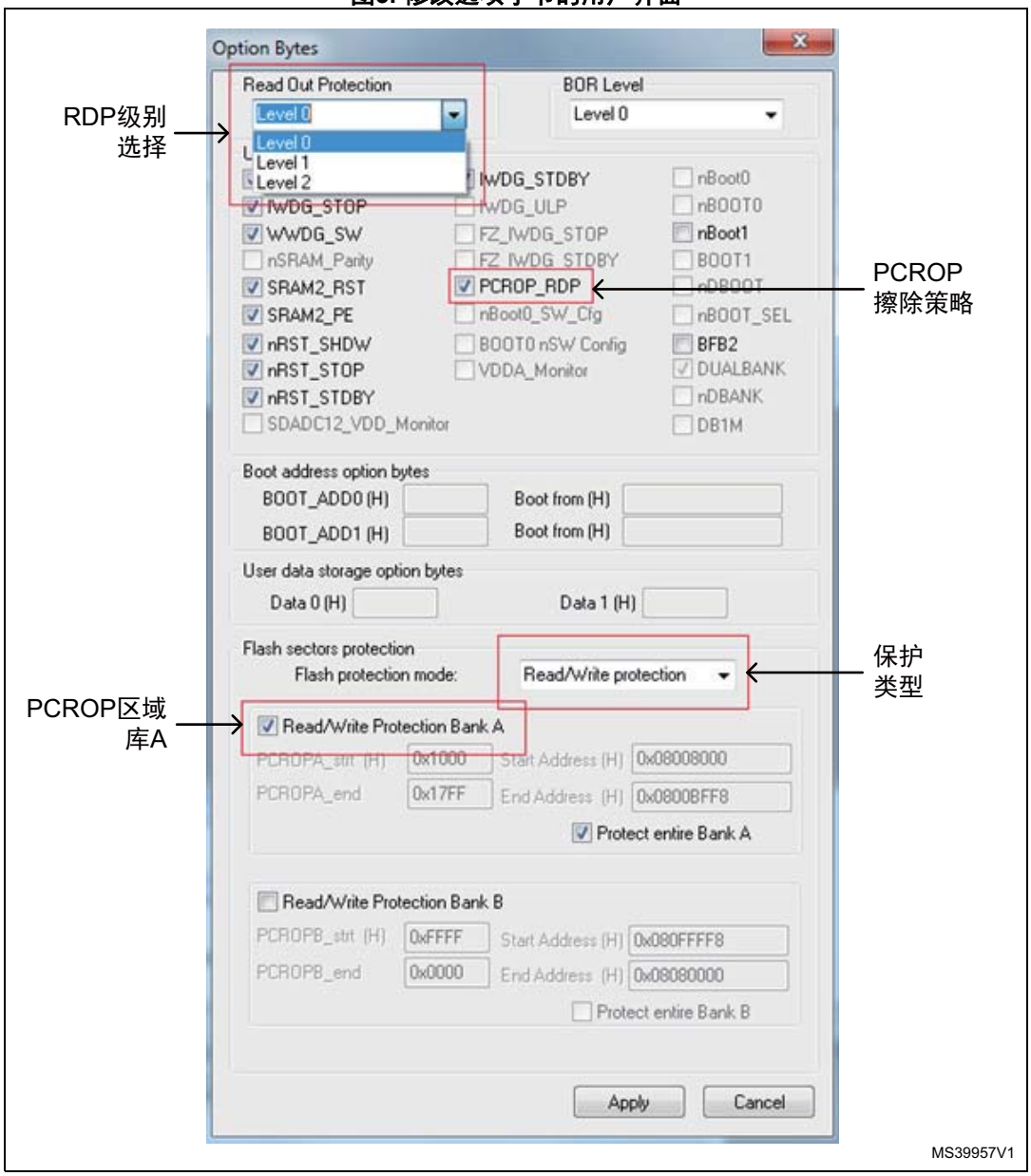
如果已设置PCROP_RDP，则对主闪存进行批量擦除。RTC和SRAM2/CCM-SRAM中的备份寄存器也将被擦除。

如果PCROP_RDP位被清零，则完全批量擦除由部分批量擦除所代替，该部分页面擦除是对PCROP激活的存储区内进行连续页面擦除（PCROP保护的页面除外）。这样做是为了保持PCROP代码。

使用STM32-Link Utility

应用程序开发过程中，用户可能需要使用另一个替代方案（而不是内嵌的闪存代码）来禁用PCROP或全局RDP保护。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。*图 3*显示如何修改选项字节。

图3. 修改选项字节的用户界面



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。

1.3.4 PCROP保护的IP-Code编译

对PCROP保护区域进行保护，以防数据总线（D代码）读取访问，因此仅允许执行代码（通过I代码总线提取指令），而无法进行数据读取。因此，受保护的IP代码将无法访问存储在同一区域的相关数据值（如文字池或常量，执行过程中它们通过D-Code总线从闪存中取出）。

文字池

应使用编译选项以避免使用原本应放在代码段中的文字池。编译选项将在*第 2 节*中进一步详细说明。

常量数据

IP-CODE使用的非易失性数据应放置在PCROP包含区域之外的特定存储区域中。IP-CODE开发人员将为存储器映射提供这些常量数据区域。建议对这些部分进行写保护。

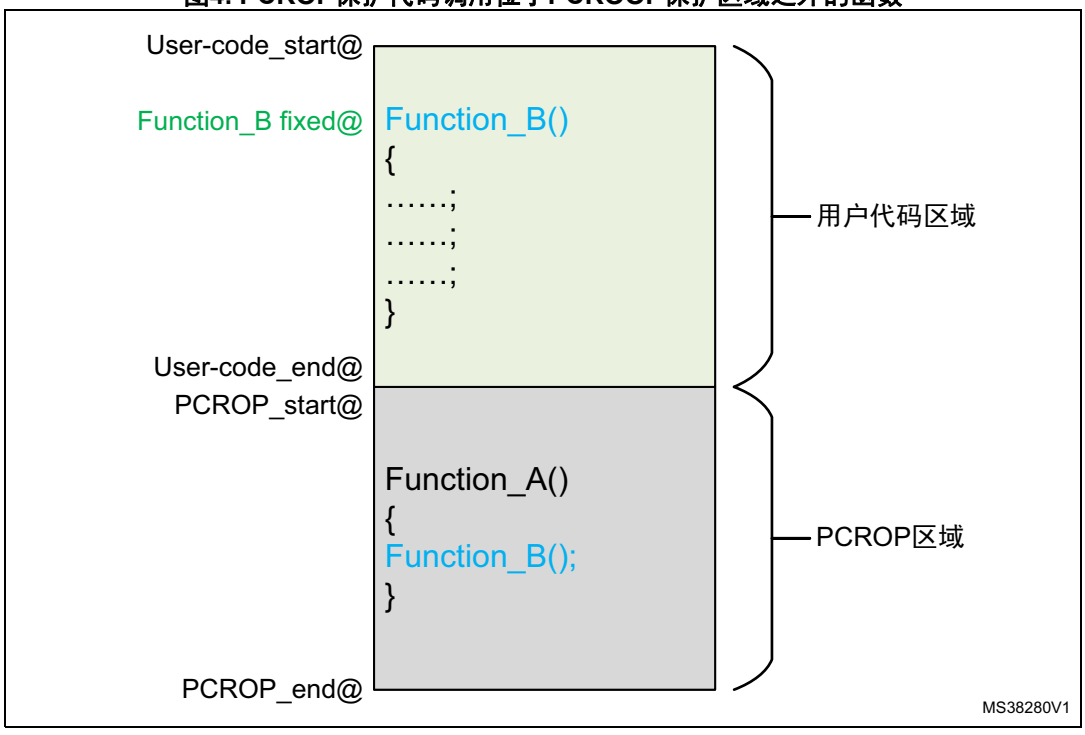
1.3.5 PCROP保护的IP-Code相关性

受保护的IP代码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代码中包含了相关函数地址，允许PC（程序计数器）在执行IP代码时跳转到这些函数。一旦IP代码被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代码中写入的相应固定地址，否则PC跳转到无效地址，IP代码将无法正常工作。

要完全独立，受保护IP代码必须与其所有关联函数放在一起。

*图 4*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（绿色）的Function_B() （浅蓝色）。

图4. PCROP保护代码调用位于PCROOP保护区域之外的函数



1.4 其他保护

1.4.1 防火墙

防火墙是STM32L4/L4+系列可提供的保护功能。防火墙与其他闪存保护相关联，可提高对来自第三方的部分代码和数据的保护级别。

用户可能希望使用存储器映射中某处的机密关联数据（加密密钥、安全算法和关联变量...），对某些敏感算法进行保护。用户可能希望准确管理用户代码何时访问该受信区域，以及该安全区域何时跳回到非受保护用户代码执行。如果在代码执行器件检测到某些意外访问，防火墙将完全执行访问监视（指令提取、读、写操作）的作用并生成复位，立即停止受保护区域内的任何入侵行为。

有关该功能的详细说明，请参阅AN5185。

1.4.2 STM32G4系列的安全存储区

安全存储区定义了一个代码区域，该代码区域只能在启动时执行一次，除非再次发生新的复位，否则不会再执行。

该存储区的主要目的是保护闪存的特定部分免受意外访问。它专用于执行受信代码，如安全密钥存储或安全启动。

2 PCROP示例

本应用笔记提供的固件示例对PCROP保护功能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节中详细说明。

该示例针对STM32L4系列而开发，但可轻松移植到STM32L4+和STM32G4系列。

2.1 要求

2.1.1 硬件要求

运行此示例所需的硬件如下：

- 内嵌STM32L476VG MCU的STM32L4探索板（版本B或C）
- 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。

2.1.2 软件要求

需要下列软件工具：

- IAR Embedded Workbench® (v7.40.3)或Keil® µvision IDE(v5.14.0)
- STM32 STLink Utility（v3.7.0）主要用于使能或禁用PCROP保护。

2.2 说明

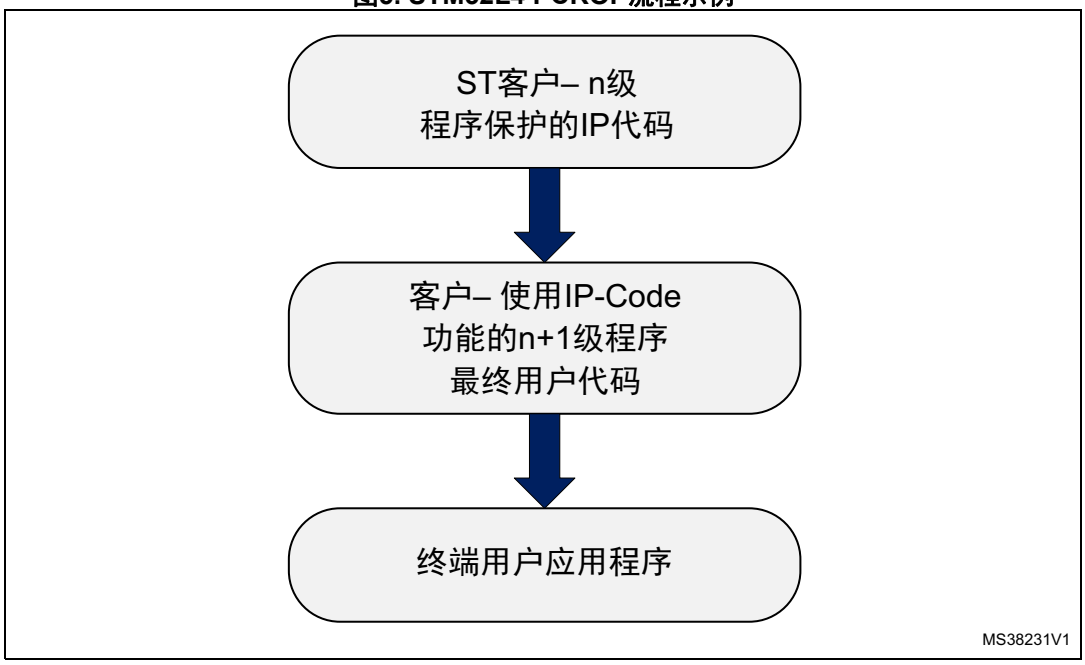
2.2.1 场景描述

本例描述了意法半导体用戶应用层级n向意法半导体用戶应用层级n+1提供具有关键IP代码的预编程STM32L476VG MCU的用例。IP代码必须通过激活PCROP来保护，允许意法半导体用户应用层级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。意法半导体用戶应用层级n应将下列输入与预加载的STM32 MCU一起提供：

- 闪存映射定义了确切受保护的IP-CODE位置以及常量数据位置。
- 意法半导体用戶应用层级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代码函数定义。
- 符号定义文件，包含IP-Code函数符号。

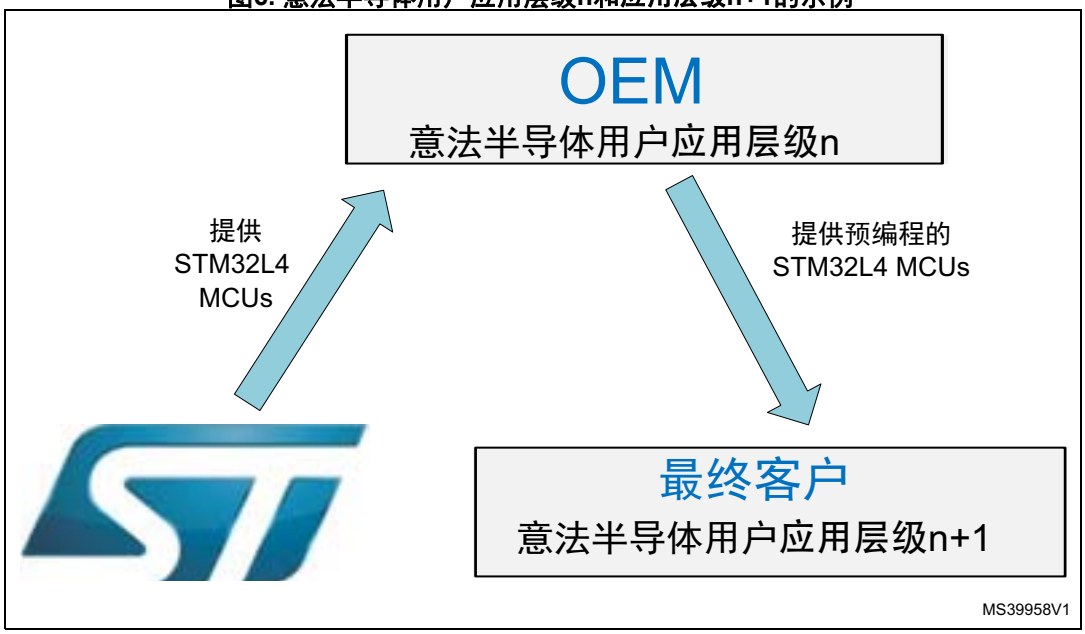
所述用例如*图 5*中所示意。

图5. STM32L4 PCROP流程示例



OEM（原始设备制造商）可以是使用STM32L4微控制器的意法半导体用户应用层级n。然后OEM提供预编程MCU给意法半导体用户应用层级n+1，这可能是制造最终用户产品的END CUSTOMER，如*图 6*中所示。

图6. 意法半导体用户应用层级n和应用层级n+1的示例



2.2.2 PCROP保护的IP代码：FIR低通滤波器

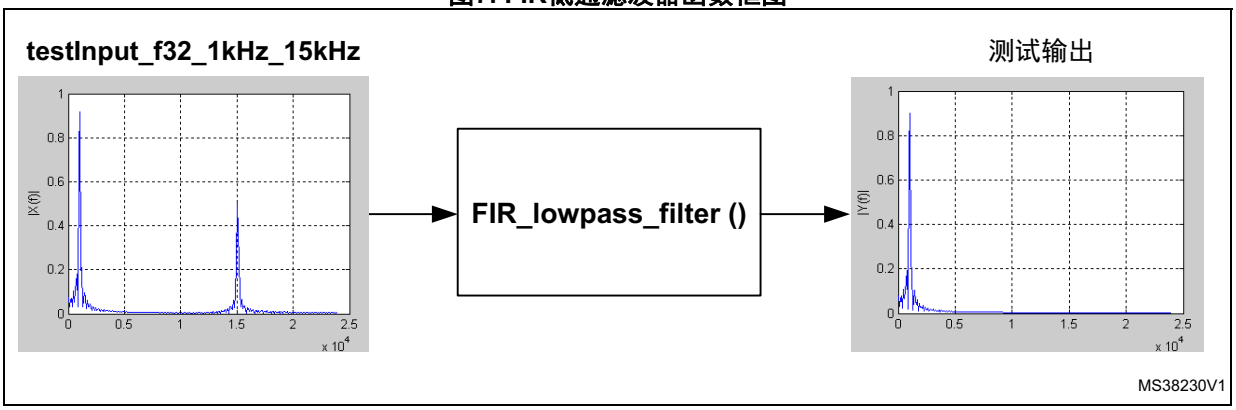
我们可以以CMSIS-DSP库中的FIR低通滤波器算法为例，作为要保护的IP-CODE。在这里，我们只将重点放在详细说明如何保护并调用该IP-CODE示例，并未详细介绍其自身功能。

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（其预配置截止频率为6 KHz）滤除15 KHz信号，在输出端留下1 KHz正弦波。

*图 7*显示了FIR低通滤波器框图。

图7. 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。

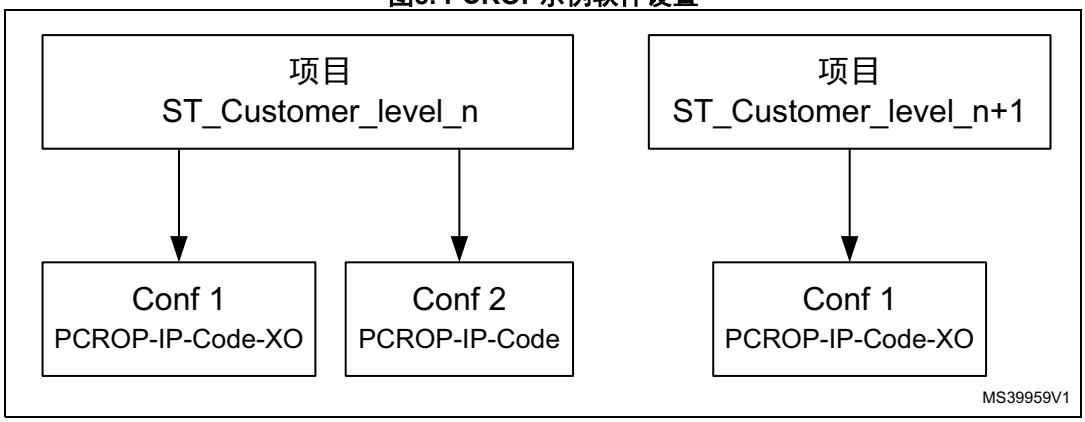
嵌入到STM32L4微控制器中的FPU和DSP用来进行信号处理和浮点计算，以输出正确信号。

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。

2.2.3 软件设置

本应用笔记详细介绍了两个项目（*图 8*）。

图8. PCROP示例软件设置



项目1：STEP1-ST_Customer_level_ n

此项目显示了意法半导体用戶应用层级n如何存放、保护和执行其IP代码，以及如何生成IP代码关联文件如头文件和符号定义文件，并提供给意法半导体用戶应用层级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代码，避免对其进行数据读取。
- PCROP-IP-Code：此配置下，编译IP代码，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代码必须为只执行代码。

项目2：STEP2-ST_Customer_level_n+1

此项目显示了意法半导体用戶应用层级n+1, 在得到预编程了PCROP保护的IP代码的STM32L476VG后，如何利用这些受保护的IP代码函数来创建其自己的最终用户应用程序的示例。

2.3 开发步骤1：意法半导体用戶应用层级n

此阶段中，意法半导体用戶应用层级n将：

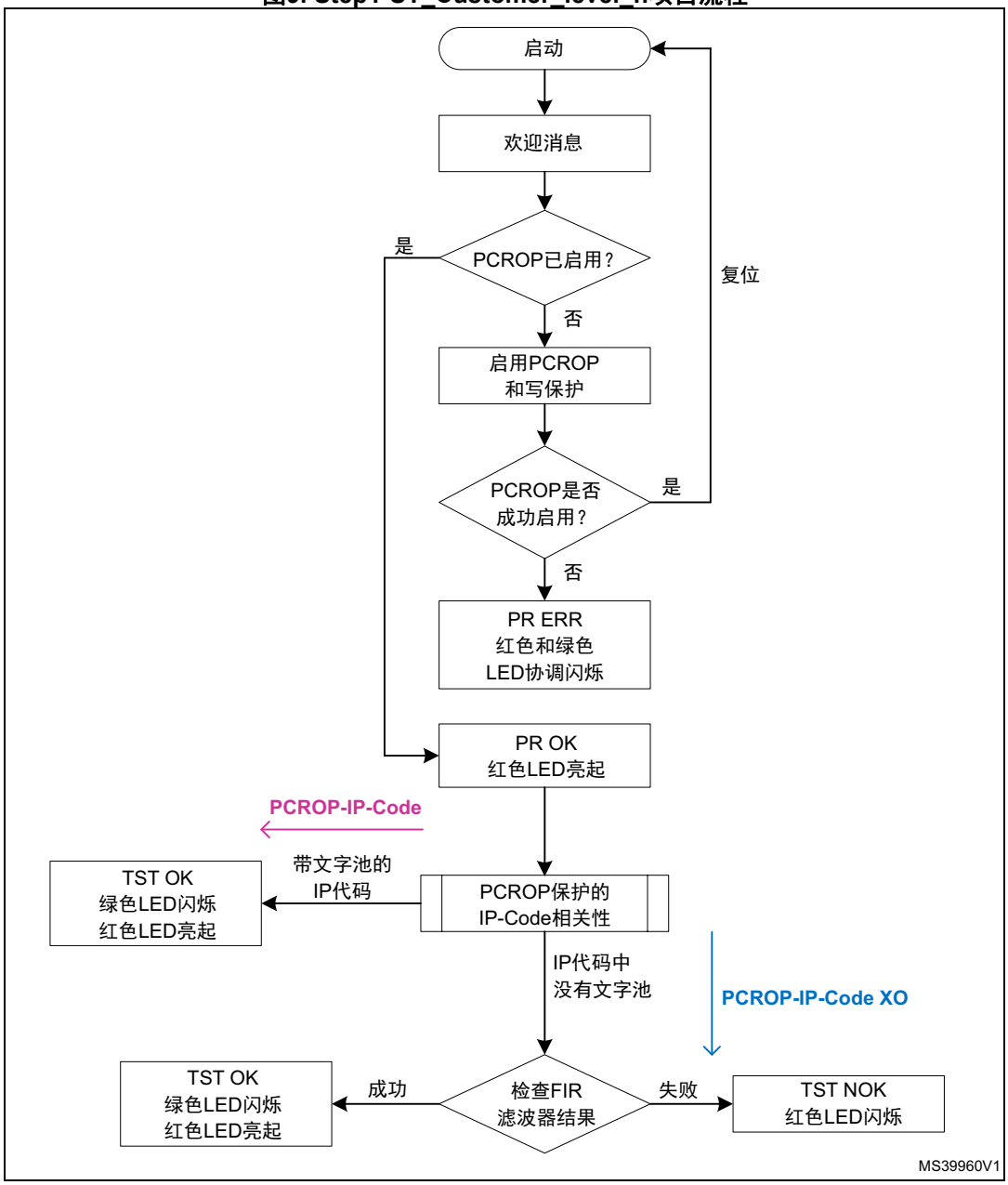
- 生成只执行IP代码；
- 将IP代码和数据段放在闪存的指定位置；
- 写保护数据段；
- 利用PCROP保护IP代码区域；
- 通过在主代码中调用其函数，执行IP代码；
- 创建头文件，并生成符号定义文件，用于STEP2-ST_Customer_level_n+1项目。

2.3.1 项目流程

*图 9*说明了采用两种项目配置的Step1-ST_Customer_level_n项目流程。测试结束根据所选配置而异。

- PCROP-IP-CODE-XO（蓝色箭头）：PCROP保护的IP-CODE不含任何文字池，FIR滤波器算法成功运行。
- PCROP-IP-CODE（红色箭头）： IP-CODE包含文字池，开始执行PCROP保护的IP-CODE时，会发生读取操作错误中断。

图9. Step1-ST_Customer_level_n项目流程

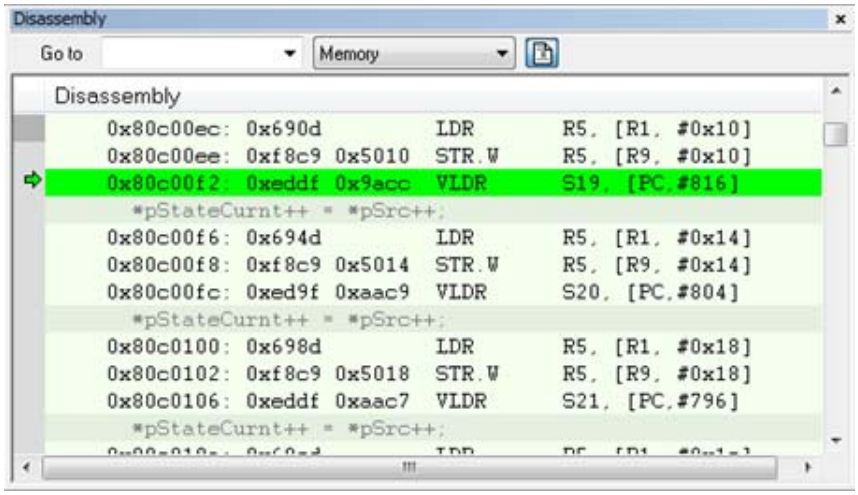


2.3.2 生成只执行IP代码

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 10*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10. 包含文字池的汇编代码示例



Keil®-仅执行编译选项

必须使用编译选项*"-execute_only"*来生成无文字池的代码，防止编译器对代码扇区进行任何数据访问。

对于所有包含文字池的IP代码文件，必须使用该选项。

要设置该命令，请右键单击组文件或IP-Code并（请参阅*图 11*）选择"用户/Fir"组选项：

如*图 12*中所述，在以下窗口中选择选项Execute-only Code：
*–execute_only*选项会自动被添加到编译器控制字符串字段中：

图11. 访问FIR滤波器选项

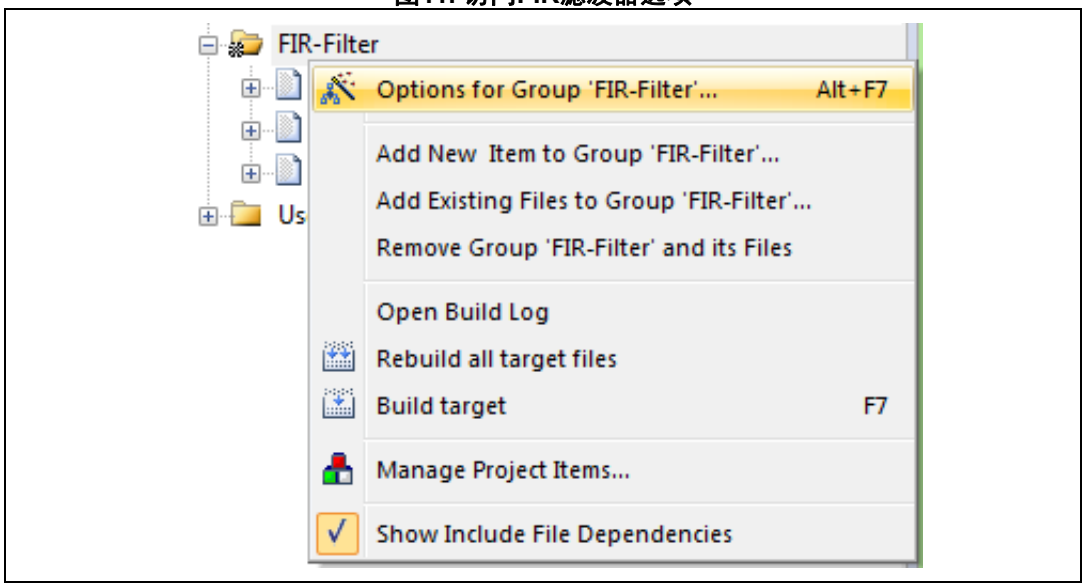
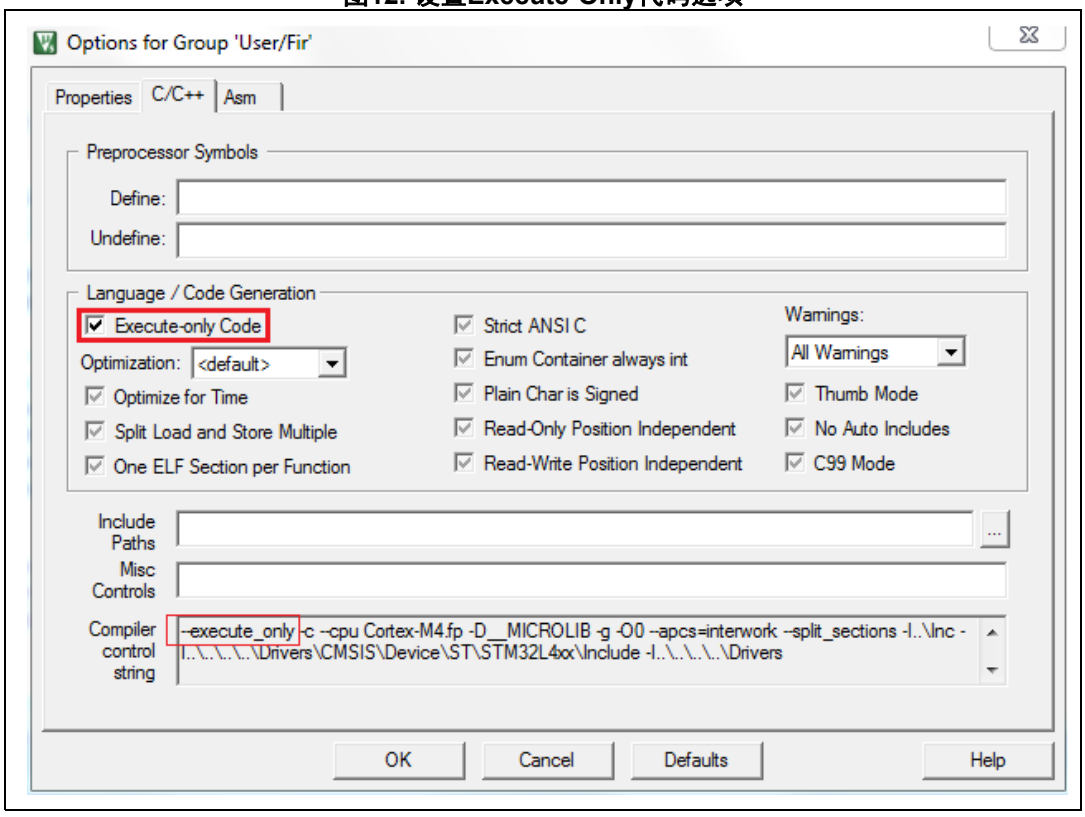


图12. 设置Execute-Only代码选项



定义IP代码内存区域布局的分散文件将访问类型属性设置为+XO。请参考*第 2.3.3 节*。

IAR：代码内存中不进行数据读取

必须在IAR™中使用选项“No data reads in code memory（未在代码存储器中读取数据）”，生成没有文字池的代码。要激活该选项，用户必须右键单击包含IP-CODE源文件的文件组“Fir”（请参阅*图 13*），然后选择“选项”，然后在下面的窗口中选中选项“*No data reads in code memory*”，如*图 14*中所示。

图13. 访问FIR-Filter选项

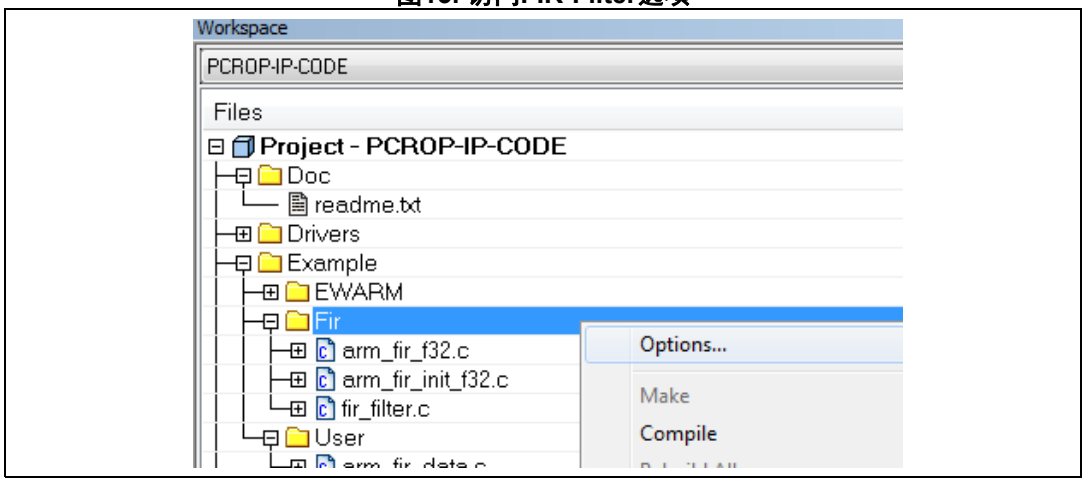
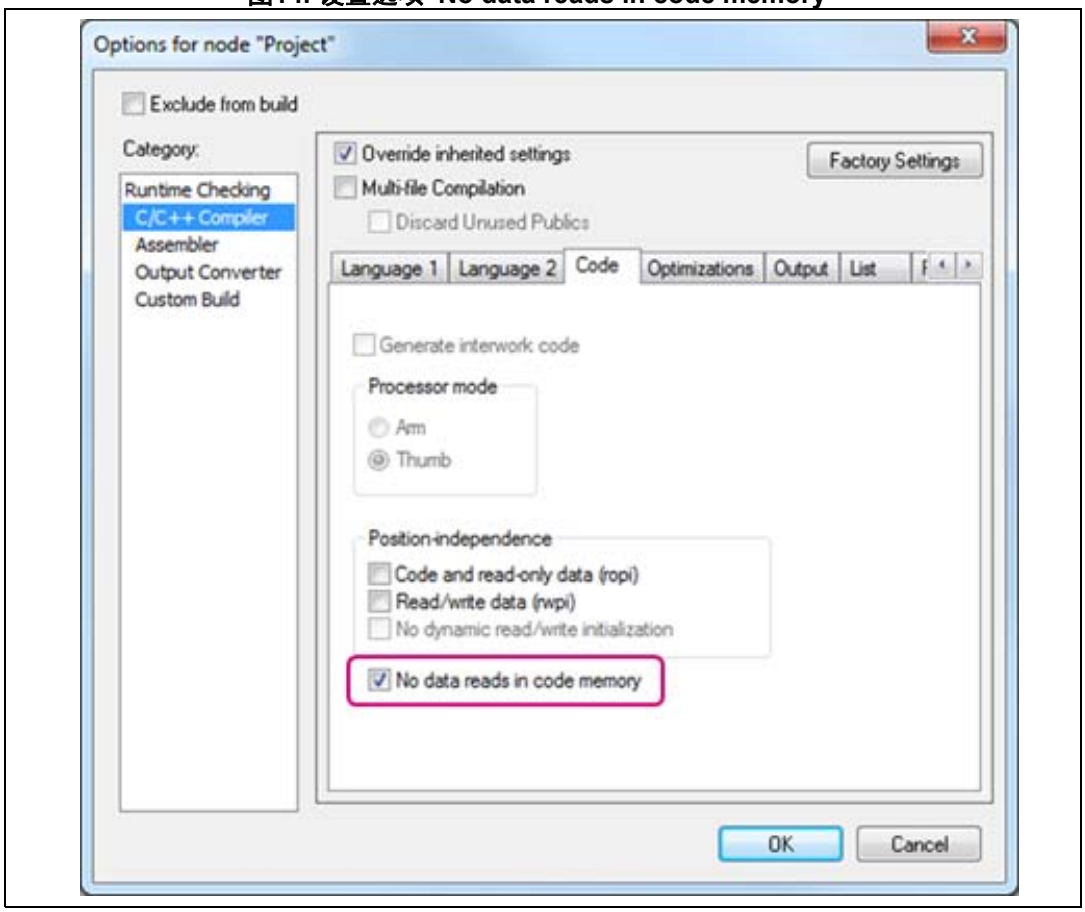


图14. 设置选项“No data reads in code memory”



2.3.3 将IP-Code和数据段放在Flash存储器中

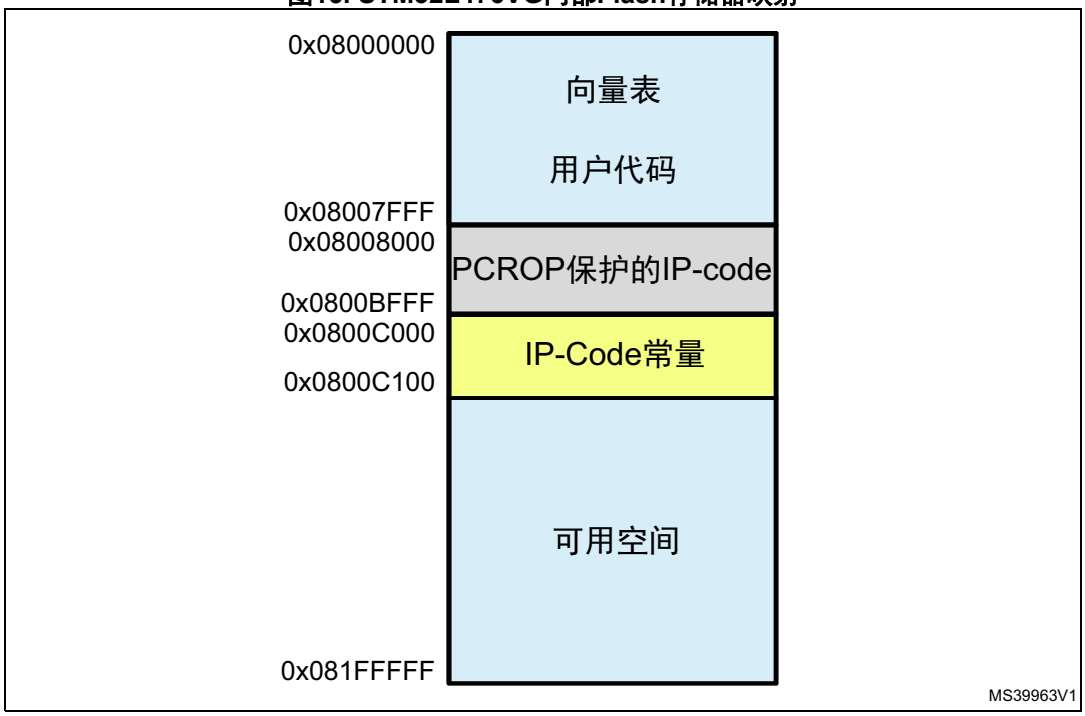
在本应用笔记示例中，将针对IP-CODE段及其关联的常量数据（FIR滤波器系数）对两个特定的存储器段进行定义。

- IP代码段位于Flash存储区1中，地址为0x0800_8000
- 常量数据位于代码扇区的后面，地址为0x0800_C000。

用户代码和向量表的主存储区域位于开端

所产生的Flash存储器映射如*图 15*所示。

图15. STM32L476VG内部Flash存储器映射



存储器布局由Keil®IDE的分散文件和IAR™中的链接器配置文件（ICF）进行处理。

Keil®分散文件

必须更新这两个新区域的分散文件：

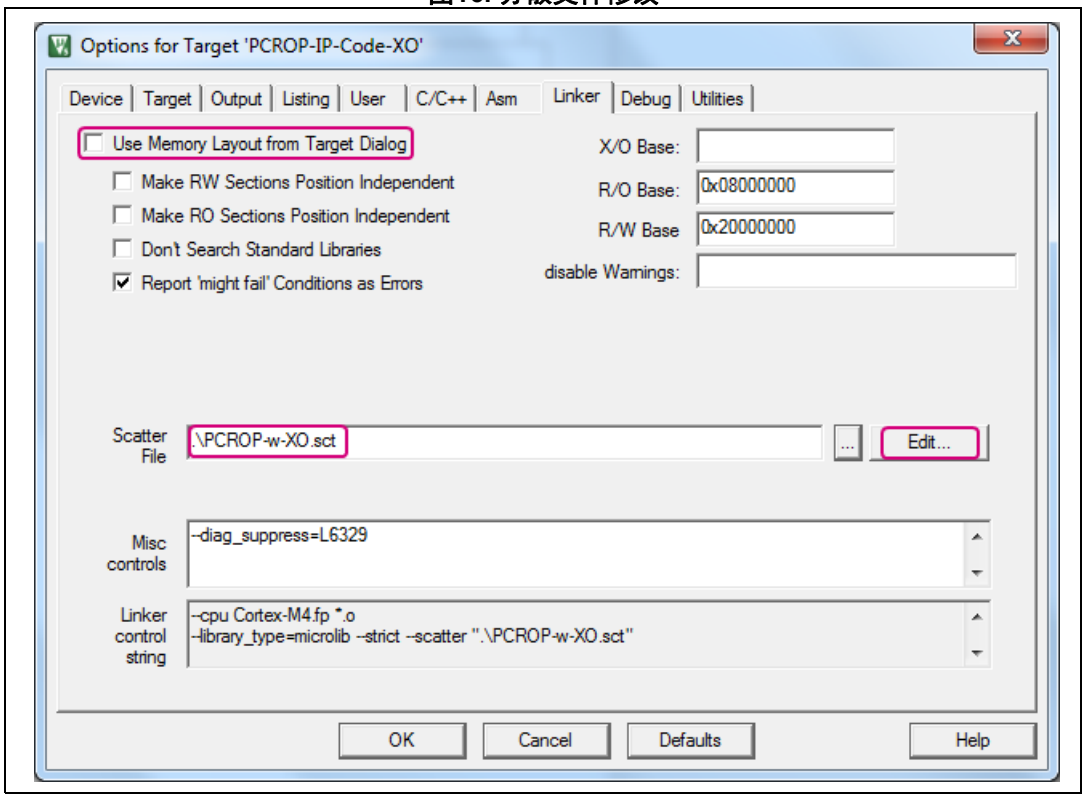
- LR_PCROP是具有仅执行访问属性的IP-CODE区域
- LR_DATA为常量数据存储区域，用于IP-Code。定义一个新扇区，名为“fir_coeff_section”。

```
; *************************************************************
; *** 分散加载描述文件，由uVision生成 ***
; *************************************************************
; 在这里代表ROM和sram存储区域
..
..
```

```
; *********************************
;  PCROP保护区域
; *********************************
LR_PCROP 0x08008000 0x00004000  {
  ER_PCROP 0x08008000 0x00004000  {  ; 载入地址 = 执行地址
     arm_fir_init_f32.o (+XO); +XO用于仅执行访问
     arm_fir_f32.o (+XO)
     FIR_Filter.o (+XO)
  }
}
;  定义FIR系数的2KB的部分
;  本节对应于Flash页数=24
LR_DATA 0x0800C000 0x00000800  {
  fir_coef_section 0x0800C000 0x00000800  {
     FIR_Filter.o (+RO)
  }
}
```

在“Project（项目）”→“Options for Target（目标选项）”中，将该新的分散文件选择到链接器。选择“Linker（链接器）”选项卡，然后取消选中“Use Memory Layout from Target Dialo（使用目标对话框中的存储器布局）”，如*图 16*中所示。

图16. 分散文件修改



IAR ICF文件

定义了两个存储区域

- PCROP_region是IP-CODE区域。
- 由于数组说明中的权利属性，Fir_coef_region是将用于fir系数的存储区域（请参阅*显式数据放置*）。

请注意，编译对象fir_filter.o包含代码和全局常量数据。代码和文件之间的分隔在ICF文件中是显式的。

```
/* 定义PCROP区域起始和结束地址 */
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__];

/* 定义fir系数常量数据的2KB页*/
define symbol __ICFEDIT_region_CONST_start__    = 0x0800C000;
define symbol __ICFEDIT_region_CONST_end__      = 0x0800C800;
define region fir_coef_region    = mem:[from __ICFEDIT_region_CONST_start__   to
__ICFEDIT_region_CONST_end__];

/* 将 IP-CODE 放置在 PCROP 存储区域中*/
place in PCROP_region { ro object arm_fir_f32.o,
ro object arm_fir_init_f32.o,
ro code object FIR_Filter.o};
/* 将 IP-CODE 放置在 PCROP 存储区域中*/
place in fir_coef_region { ro data section fir_coef_section object FIR_Filter.o };
```

显式数据放置

由于下面的编译属性，常量数据将映射到此扇区。以下代码摘要显示了IAR™和Keil®的语法。

```
/* IAR IDE */
/* 采用专门地址进行放置： */
const float32_t firCoeffs32[NUM_TAPS] @ 0x0800C000 = { …};
/* 采用ICF文件中定义的专门扇区进行放置： */
const float32_t firCoeffs32[NUM_TAPS] @ "fir_coef_section" = { ….};

/* KEIL IDE */
/* 采用专门地址进行放置： */
const float32_t firCoeffs32[NUM_TAPS] __attribute__((at(0x0800C000))) ={ …};
/* 采用分散文件中定义的专门扇区进行放置： */
const float32_t firCoeffs32[NUM_TAPS] __attribute__((section("fir_coef_section"))) = {…};
```

2.3.4 常量写保护

用户应该考虑到，IP代码常量可被意法半导体用戶应用层级n+1所删除或修改，函数可能变成无用的，因此建议对这些常量进行保护，避免不需要的写入/擦除。

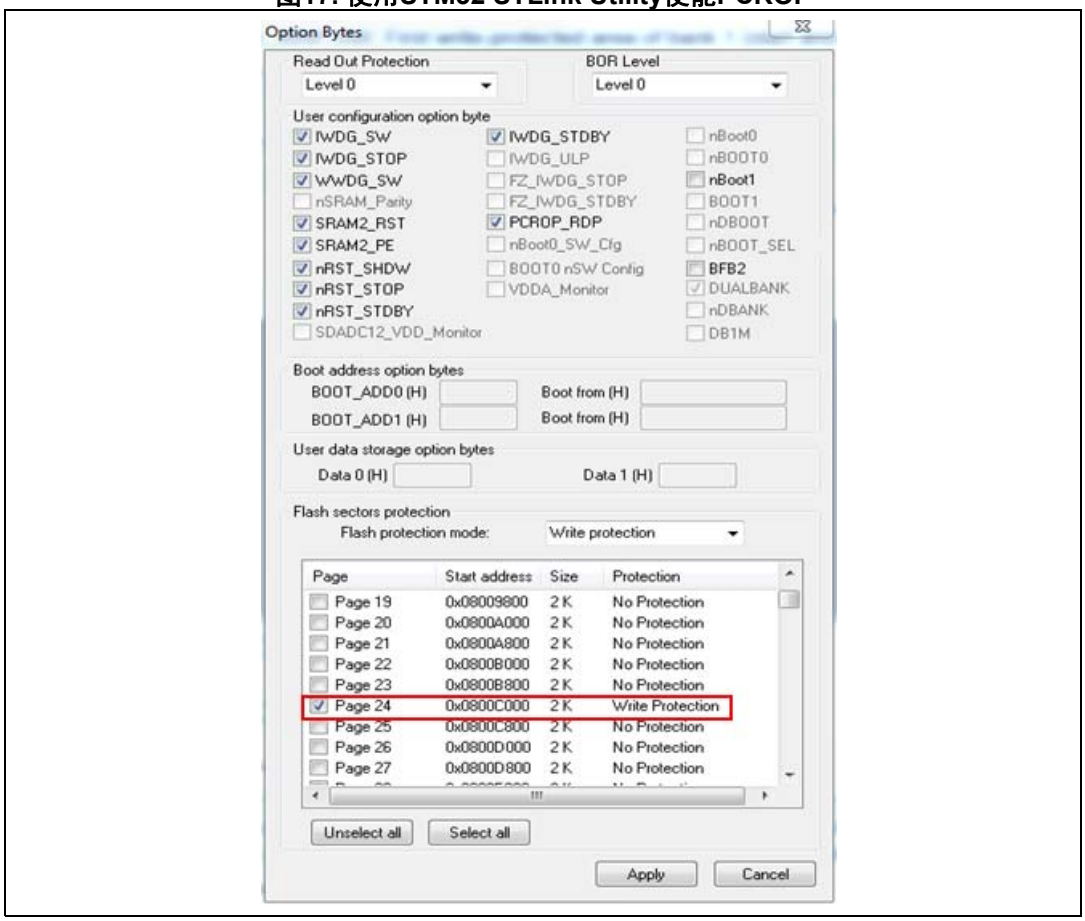
通过以下选项字节寄存器设置写保护区域：

- FLASH_WRP1AR：库1的第一写保护区域（起始页码和结束页码）；
- FLASH_WRP1BR：库1的第二写保护区域；
- FLASH_WRP2AR：库2的第一写保护区域；
- FLASH_WRP2BR：库2的第二写保护区域。

示例的IP代码使用了116个字节作为常量系数。最小页大小为2 KB，因此单个页将被保护（地址0x0800C000至0x0800C100之间）。

写保护可通过专用FW代码在专用寄存器中设置正确值（参见*PCROP_Enable()*函数）来进行设置，或利用STLink-Utility进行设置，如*图 17*中所示。

图17. 使用STM32 STLink Utility使能PCROP



2.3.5 保护IP代码

至于写保护，可以通过专用的固件代码或使用STLink-Utility设置PCROP区域。

使用固件激活PCROP

*PCROP_Enable ()*函数将在第一次运行时设置受保护区域。定义区域后，将生成系统复位以将闪存保护考虑在内。在第二次运行时，代码一直持续到测试结束。

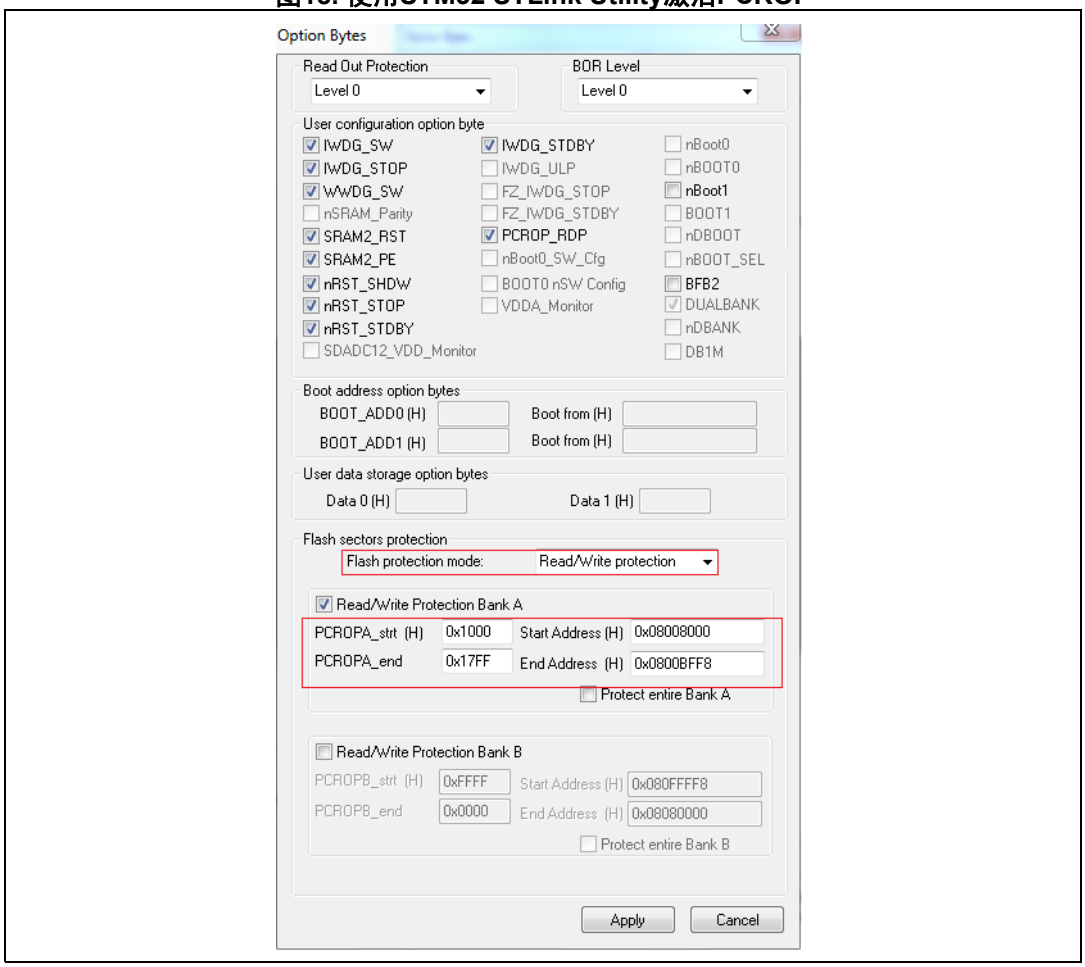
该功能在Step1-ST_Customer_level_n project main.c文件中定义。

使用STM32 STLink Utility激活PCROP

要使用STM32 STLink Utility激活PCROP，用户必须按照下面所示的顺序进行操作：

- 板子上电，然后在STLink Utility接口进入Target ➞ Option bytes
- 在以下窗口中，将闪存保护模式设置为读/写保护，在库A上启用保护，并指定区域的起始地址和结束地址，如*图 18*中所示。
- 点击Apply按钮。

图18. 使用STM32 STLink Utility激活PCROP



2.3.6 执行PCROP保护的IP-Code

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

本节我们将介绍，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。这是在运行Step2-ST_Customer_level_n+1项目之前使用配置。

PCROP-IP-Code-XO（必须在STEP2之前使用）

编译器配置为生成只执行IP代码，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保护或写保护的区域。如果有，则使用STM32 STLink Utility禁用该保护；然后载入代码。
   b) 按下复位按钮开始测试：
      显示欢迎消息“WELCOME PCROP STEP 1”；
      如果已启用PCROP，则设置红灯，否则将对PCROP区域进行编程，并在欢迎消息[a]上循环出现系统复位。
      应用程序测试正在运行并检查结果。如果通过测试，则绿灯将无限切换。

a. 该系统复位使设备与可能的调试会话断开连接。

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保护或写保护的区域。如果有，则使用STM32 STLink Utility禁用该保护，然后载入代码；
   b) 按下复位按钮开始测试：
      显示欢迎消息“WELCOME PCROP STEP 1”；
      如果已启用PCROP，则设置红灯，否则将对PCROP区域进行编程，并在欢迎消息[a]上循环出现系统复位。
      应用程序测试正在运行，但是发生中断，并且开发板显示“IT MEM”错误消息。两个LED都亮起。

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，则通过测试。
（显示“TST OK”消息，并且绿色LED持续切换）。

对于PCROP-IP-Code配置，其中PCROP保护的IP代码包含文字池：执行IP代码（*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。发生中断，并调用*HAL_FLASH_OperationErrorCallback()*函数。

对于PCROP-IP-Code-XO配置，IP-Code可正确执行并且测试正常结束。

2.3.7 创建头文件并生成符号定义文件

IP-CODE开发结束后，应将头文件和符号定义文件提供给以下开发人员/客户。

头文件

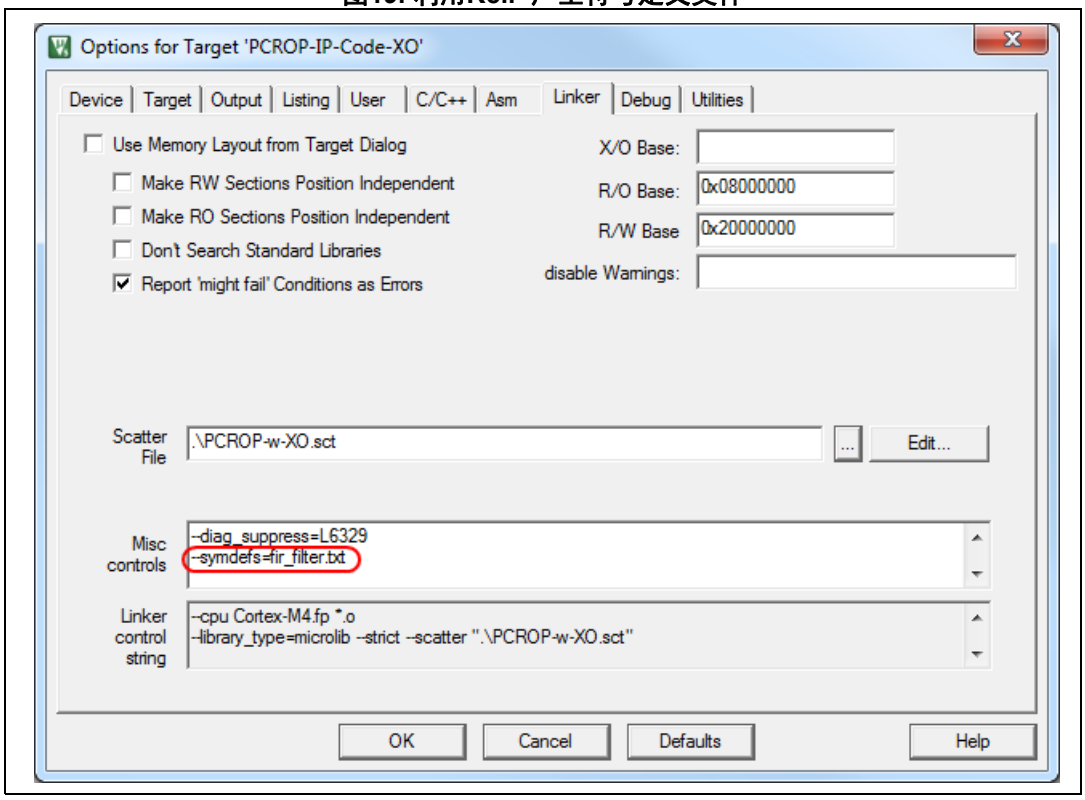
要提供给意法半导体用戶应用层级n+1的头文件包含要使用的IP代码函数定义。本例中包含在*main.c*文件中的是*fir_filter.h*文件。

利用Keil®的符号定义文件

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_filtre.txt*（请参阅*图 19*）。重建项目。

a. 该系统复位使设备与可能的调试会话断开连接。

图19. 利用Keil®产生符号定义文件



名为fir_filtre.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，因此用户必须仅保留最终用户所调用的那些IP-Code函数，并删除所有其他函数。

所得符号定义文件将是：

```
#<SYMDEFS># ARM Linker, 5050106: Last Updated: Tue Sep 01 14:22:05
0x08008001 T FIR_lowpass_filter
0x08008051 T arm_fir_f32
0x0800871b T arm_fir_init_f32
```

利用IAR的符号定义文件

IAR绝对符号导出器*isymexport*用于从ROM图像文件中导出绝对符号。

IDE中，要导出PCROP保护的IP代码符号，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"

*–edit steering_file.txt*选项用于仅保留最终用户代码要调用的功能的符号。在*steering_file.txt*中，命令“show”仅用于保留所选功能：

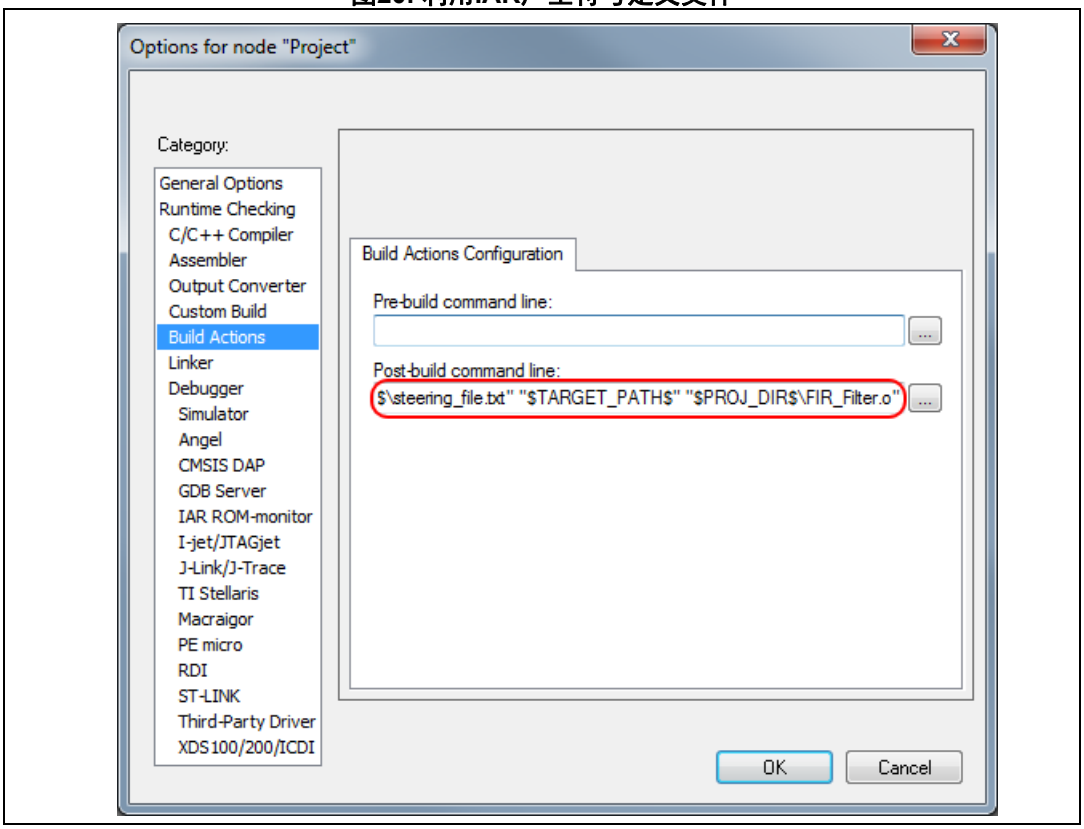
show arm_fir_f32

show arm_fir_init_f32

show FIR_lowpass_filter

最终用户将通过将输出文件*fir_filter.o*添加到项目中来进行使用。

图20. 利用IAR产生符号定义文件



2.4 开发步骤2：意法半导体用户应用层级n+1

意法半导体用户应用层级n+1具有来自用户应用层级n的，预加载PCROP保护的IP代码的STM32L476VG MCU，提供的符号定义以及头文件。

参考由意法半导体用户应用层级n提供的闪存映射，意法半导体用户应用层级n+1应：

- 创建最终项目；
- 在其项目中包含意法半导体用户应用层级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代码函数；
- 执行并调试最终用户应用程序。

注意： 意法半导体用户应用层级n+1必须使用与意法半导体用户应用层级n完全相同的工具链和编译器版本，来开发和编程IP代码，否则意法半导体用户应用层级n+1不能使用该IP代码。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。

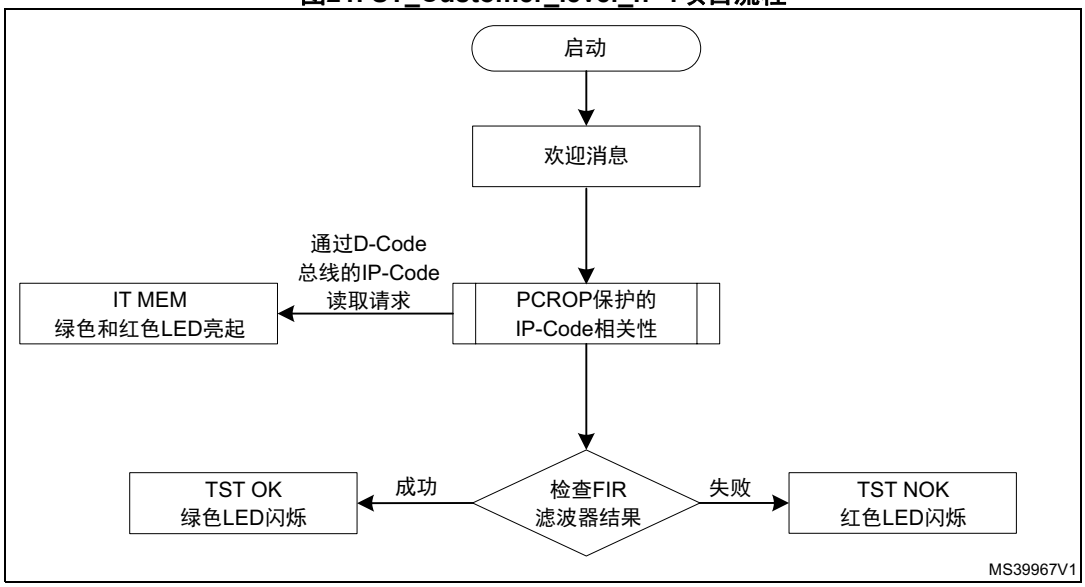
2.4.1 项目流程

*图 21*说明了ST_Customer_level_n+1项目流程。除非对保护区域进行读操作，否则应成功执行代码。该读取操作可以是调试访问或存储器转储。

中断通过特定消息“IT MEM”进行通知。

主要测试可能会失败，并最终显示消息“TSTNOK”。如果常量数据（此处为滤波器系数）已损坏或已擦除，则可能会发生这种情况。

图21. ST_Customer_level_n+1项目流程



2.4.2 创建最终用户项目

受保护的代码和写保护的数据段位于存储器映射（请参阅*图 15*）的已知位置，因此用户必须注意，在实施链接器文件时，应避免将任何代码放置在这些区域。

PCROP保护的IP代码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。

2.4.3 包含头文件并添加符号定义文件

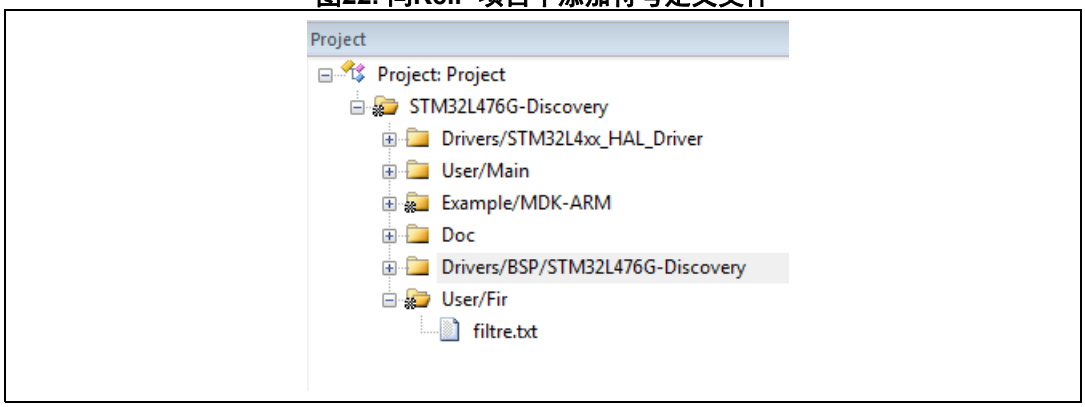
头文件*fir_filtre.h*包含在*main.c*文件中，然后用户可以调用FIR滤波器函数。文件应添加到编译器设置的包含搜索路径中。

意法半导体用戶应用层级n所提供的符号定义文件应添加为传统源文件，必须与其正确链接。用户必须根据所选的IDE遵循该方法。

向Keil®项目中添加符号定义文件

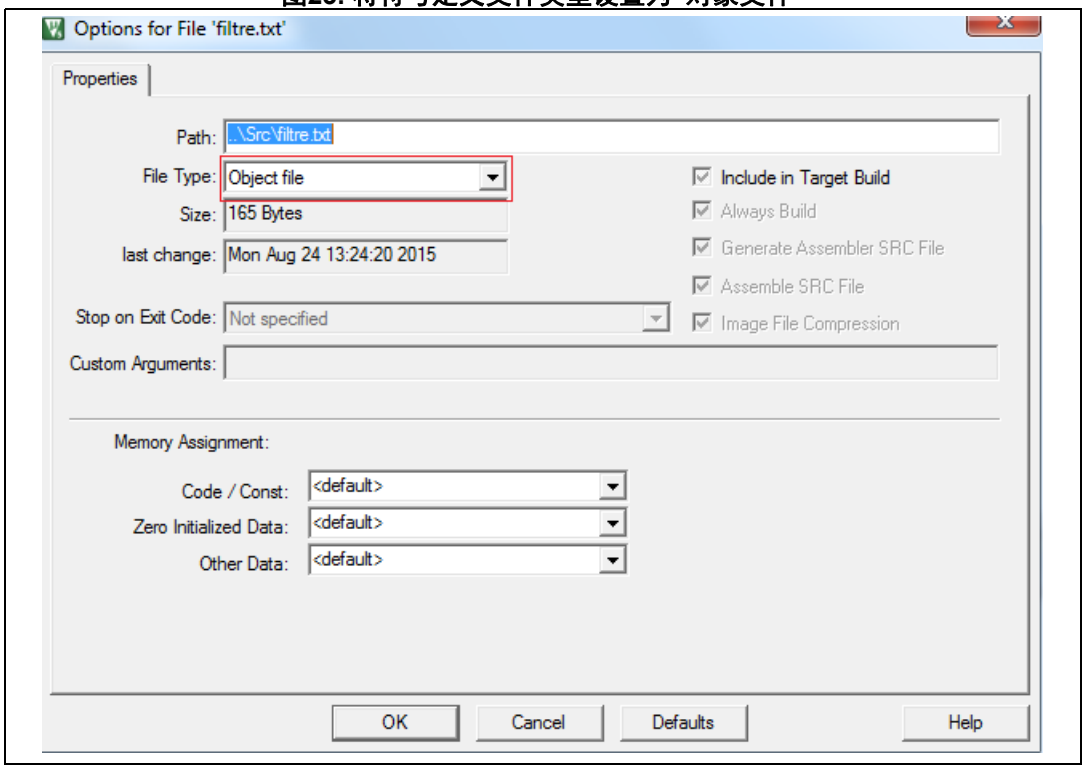
*第 2.3.7 节*中所述的符号定义文件在本例中名为*fir_filter.txt*，将其添加到*图 22*中所述的User/Fir组。

图22. 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（请参阅*图 23*），而不是文本文档文件。

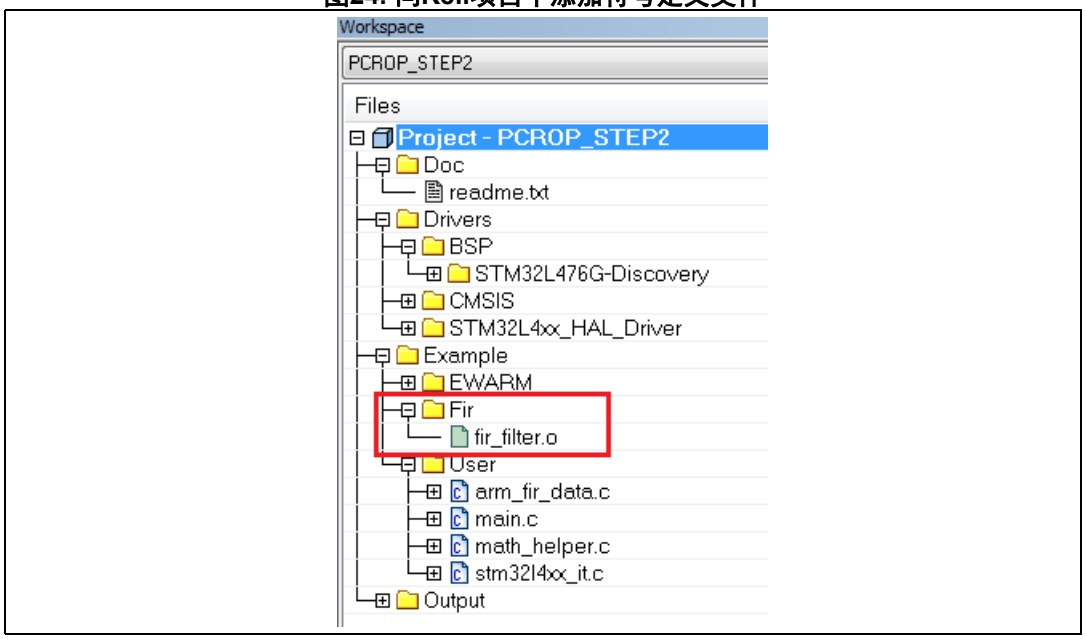
图23. 将符号定义文件类型设置为“对象文件”



向IAR项目中添加符号定义文件

将*第 2.3.7节*中生成的*fir_filter.o*文件作为对象文件添加到*图 24*中所示的“Fir”组中。

图24. 向Keil项目中添加符号定义文件



2.4.4 运行最终用户应用程序

在此阶段IP代码已经由STM32L476VG 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，用户必须按照下述步骤操作：

1. 打开位于Step2-ST_Customer_level_n+1目录中的项目，并选择与步骤1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“reset（复位）”按钮。
   显示欢迎消息“WELCOME PCROP STEP 2”
   应用程序测试正在运行并检查结果。如果通过测试，则绿灯将无限切换。

2.4.5 调试模式中的PCROP保护

在开发其最终用户应用程序时，意法半导体用戶应用层级n+1需要调试其代码。因此，在调试最终用户应用程序时，PCROP保护会阻止任何对意法半导体用戶应用层级n所开发的IP代码的读访问。

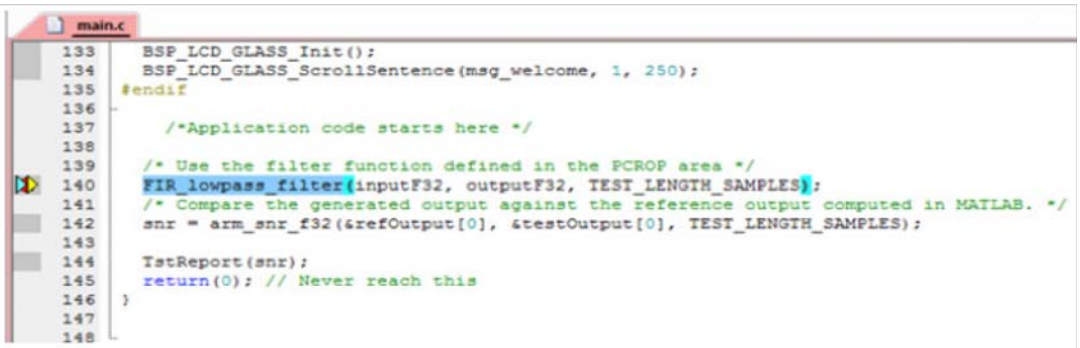
本章中，我们将介绍在调试用户代码时，PCROP如何保护预编程IP代码不被读取。下述调试示例在Keil® IDE上完成。

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，使STM32L476具有预编程且PCROP保护的IP代码。

要调试Step1-ST_Customer_level_n+1项目，用户必须按照下述步骤操作：

1. 打开位于Step1-ST_Customer_level_n+1目录中的项目，并选择与步骤1中使用的相同的工具链。
2. 重建所有文件。
3. 点击Start/Stop Debug会话来开始调试。
4. 在FIR_lowpass_filter（受保护的函数）上放置断点。



5. 单击“Run（运行）”来运行程序，板上将显示欢迎消息“WELCOME PCROP STEP 2”。
6. 要访问PCROP保护的IP代码，右键点击Disassembly窗口，然后选择Show Disassembly at Address，如*图 25*中所示。
7. 在地址栏填写PCROP保护区域起始地址，并单击*图 26*中所示的“Go To（转到）”按钮”：如*图 27*中所示，PCROP保护的IP-Code是不可读的。
8. 点击“next step（下一步）”。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。显示“IT MEM”消息（*图 28*）。

图25. PCROP保护的IP-Code 汇编代码的读取

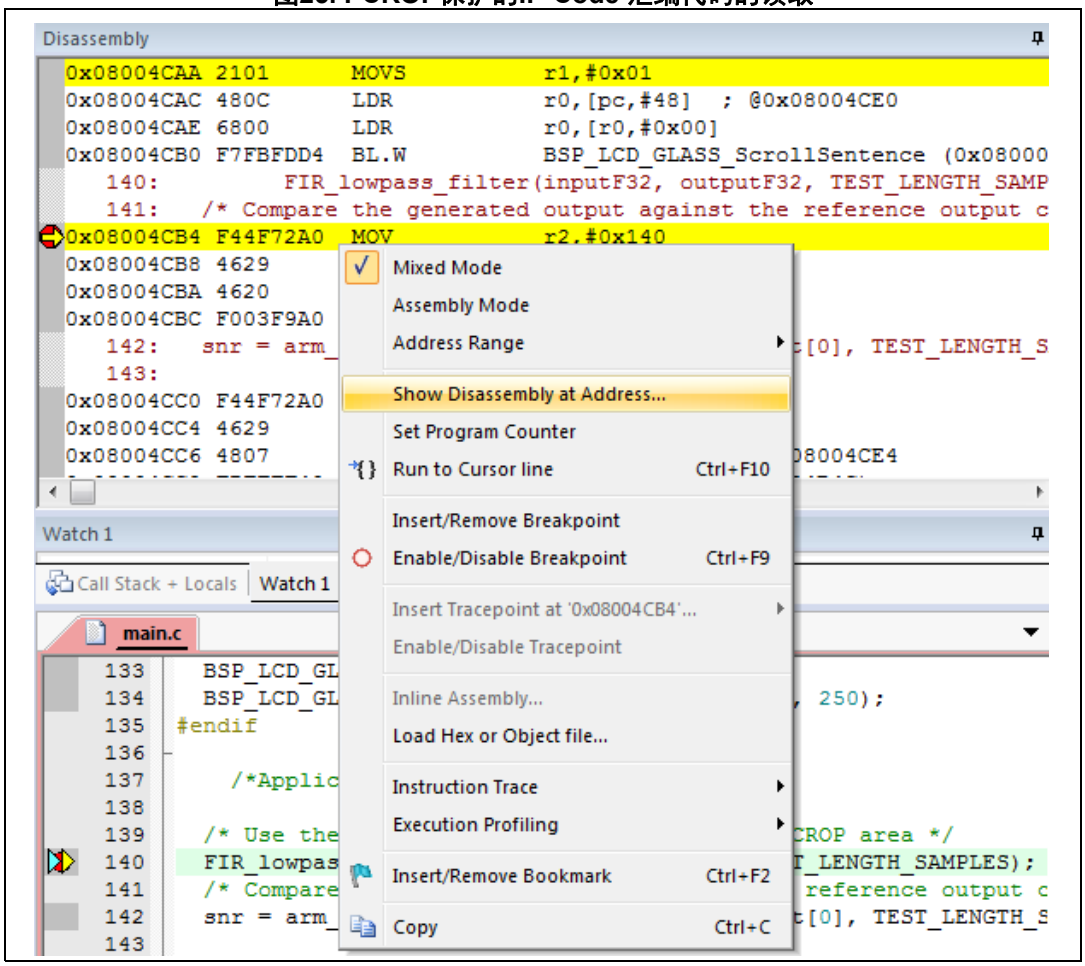


图26. 填写PCROP保护区域起始地址

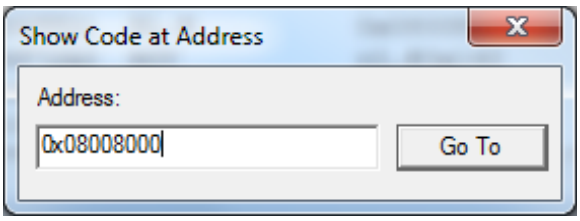


图27. PCROP保护的IP-Code 汇编代码的读取

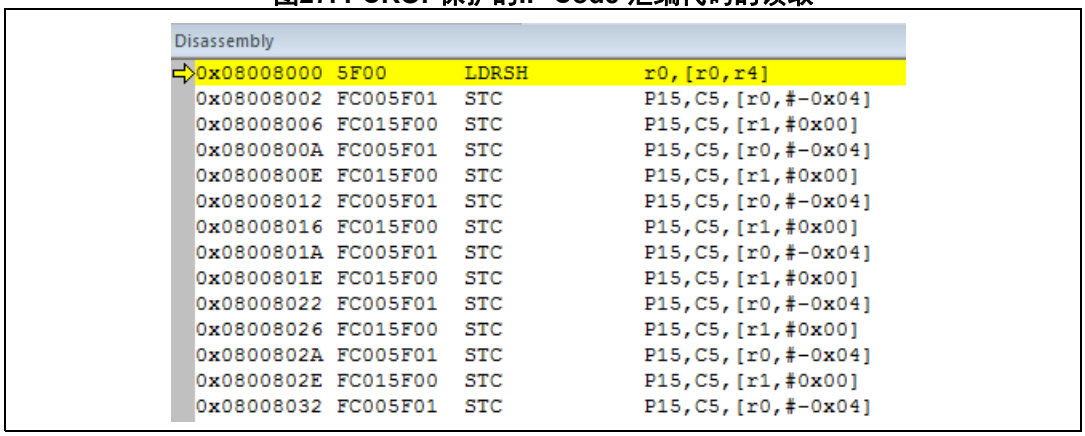
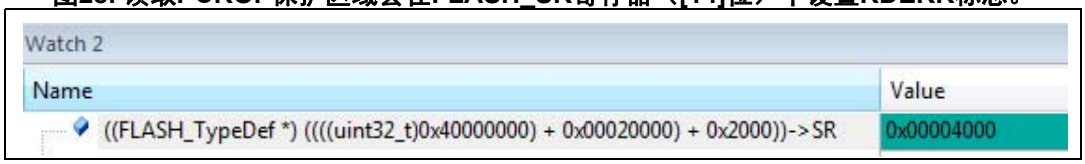


图28. 读取PCROP保护区域会在FLASH_SR寄存器（[14]位）中设置RDERR标志。



3 结论

STM32L4、STM32L4+和STM32G4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读、写和PCROP保护功能。

4 版本历史

表3. 文档版本历史

日期	版本	变更
2015年10月26日	1	初始版本。
2017年9月8日	2	在封面上添加了对RM0392和RM0394的引用。
2019年9月5日	3	引入了STM32G4系列。 更新了文档标题、*引言*、*第 1节：存储器保护说明*、*第 1.1节：读取保护（RDP）*、*第 1.1.2节：读保护级别1*、*第 1.2.1节：闪存写保护*、*第 1.2.2节：SRAM2 CCM-SRAM写保护*、*第 1.3.2节：如何启用PCROP保护？*、*第 1.3.3节：如何禁用PCROP保护？*、*第 1.4.1节：防火墙*、*第 2节：PCROP示例*和*第 3节：结论*。 增加了*第 1.4.2节：STM32G4系列的安全存储区* 更新了*表 1：访问状态 vs 保护级别和执行模式*。
2019年10月18日	4	引入了STM32L4+系列。 更新了文档标题、*引言*、*第 1节：存储器保护说明*、*第 1.1节：读取保护（RDP）*、*第 1.1.2节：读保护级别1*、*第 1.2.1节：闪存写保护*、*第 1.2.2节：SRAM2 CCM-SRAM写保护*、*第 1.3.2节：如何启用PCROP保护？*、、*第 1.4.1节：防火墙第 2节：PCROP示例*和*第 3节：结论*。

表4. 中文文档版本历史

日期	版本	变更
2022年11月11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。若需 ST 商标的更多信息，请参考 *www.st.com/trademarks*。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。