AN0027
应用笔记

Setup SLIB with Eclipse and GCC

前言

本篇应用指南主要描述怎样用现成的Eclipse插件来调试AT32系列芯片以及SLIB的配置范例。
本文档仅以AT32F403A为例进行说明，关于AT32F403A SLIB的详细说明，请详阅《AT32F403A Security Library Application Note》。

支持型号列表：

支持型号	AT32 全系列

目录

表目录

图目录

1 概述

本文档仅介绍如何通过使用Eclipse、ARM-GCC编译工具、GNU-ARM插件、J-Link或AT-Link等资源来配置及调试AT32F403A的SLIB，并提供SLIB方案商开发算法范例及终端用户应用范例。关于AT32F403A的SLIB详细介绍及说明，请详阅《AN0040_AT32F403A_407_Security_Library_Application_Note》。

环境说明：

本文档安装说明基于WINDOWS 7 x64系统下实现，开发板使用AT-START-F403A。

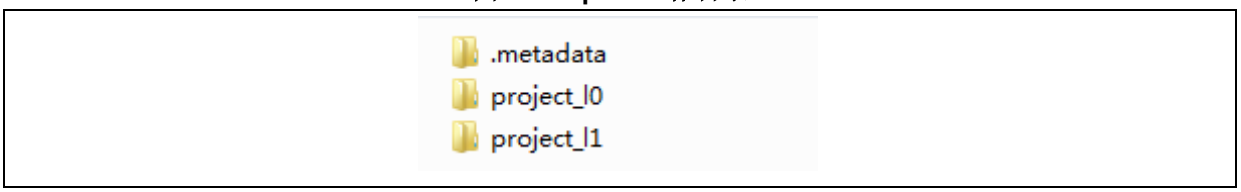
关于Eclips调试环境的安装及Eclipse工程的建立，请参阅《AN0033_Eclipse_with_GCC》。

本文档所使用的软件都打包到Setup_SLIB_with_Eclipse_and_GCC_V2.0.0.zip，直接解压重新编译后即可运行。

解压后Eclipse的workspace位于目录Setup_SLIB_with_Eclipse_and_GCC_V2.0.0\utilities\slib_with_eclipse_and_gcc_demo。

其中包含文件：

图 1. Eclipse 工作目录



.metadata：此workspace的环境设定

project_l0：方案商开发算法范例

project_l1：终端用户应用范例

2 Project_L0 方案商范例

在此阶段的范例程序，将完成下列几个项目：

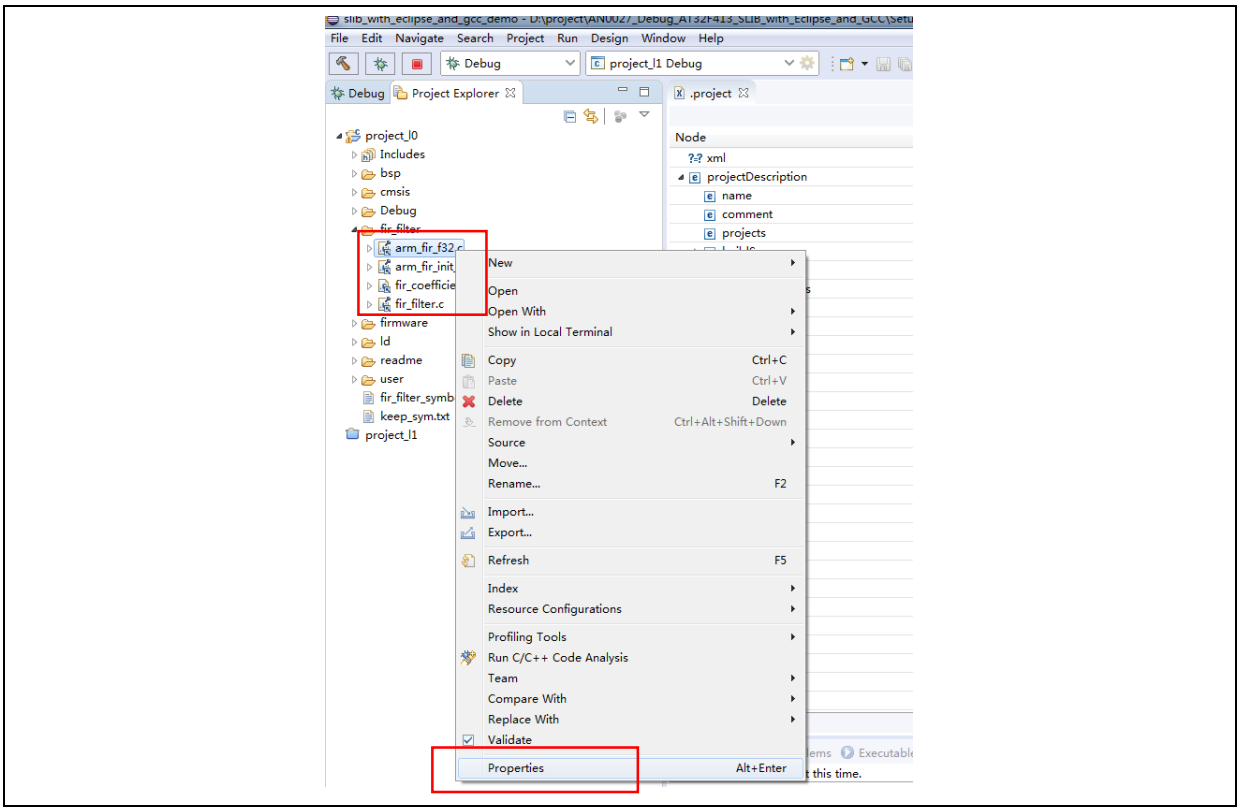
- 将低通滤波器函数编译成可执行(executr-only)的代码；
- 将低通滤波器函数的代码编排放置到主闪存区地址0x08004000 ~0x08004FFF(sector 8~9)；
- 将低通滤波器函数的系数编排放置到主闪存区地址0x08005000 ~ 0x08005FFF(sector 10~11)；
- 验证成功后，将sector 8~9设置为指令安全库区，将sector 10~11设置为数据安全库区，此部分可在范例的主程序中以调用slib_enable()函数来完成，或使用Artery ICP Programmer来完成(建议使用ICP工具完成设置)；
- 产出终端用户程序调用低通滤波函数时，需用到的头文件及符号定义文件。

2.1 产生只执行(Exacute-only)代码

设置方式如下：

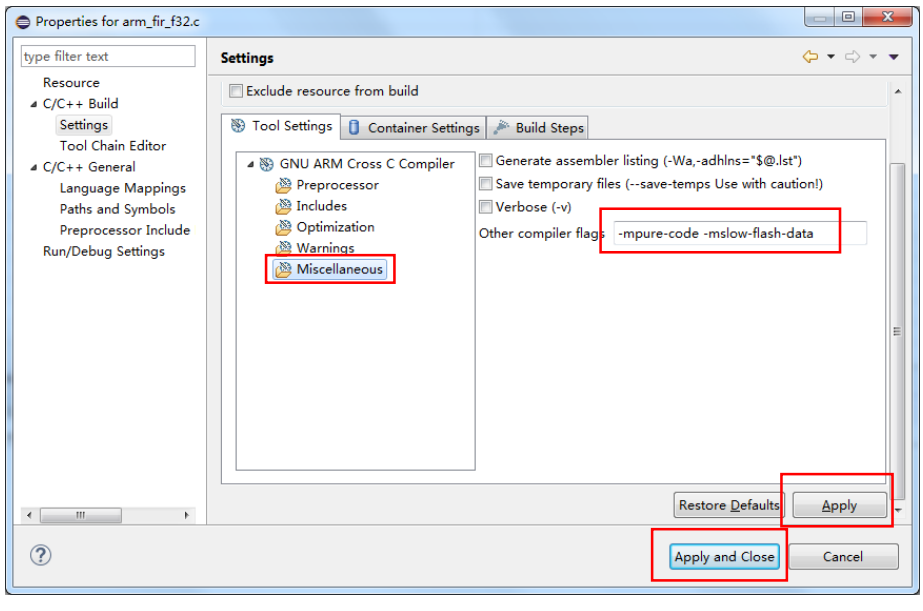
1) 选择C文件群组或个别的C文件，范例中是把要保护的相关C文件都放在FIR_Filter群组，点击FIR_Filter群组内需要设置只执行的文件，按鼠标右键选择 “Properties”

图 2. 设置要保护的 C 文件



2) 点选C/C++ Build->Settings->GNU ARM Cross C Complier->Miscellaneous，在”Other compiler flags”填入-mpure-code 以及-mslow-flash-data 这两个关键字，然后按 Apply 使设定生效

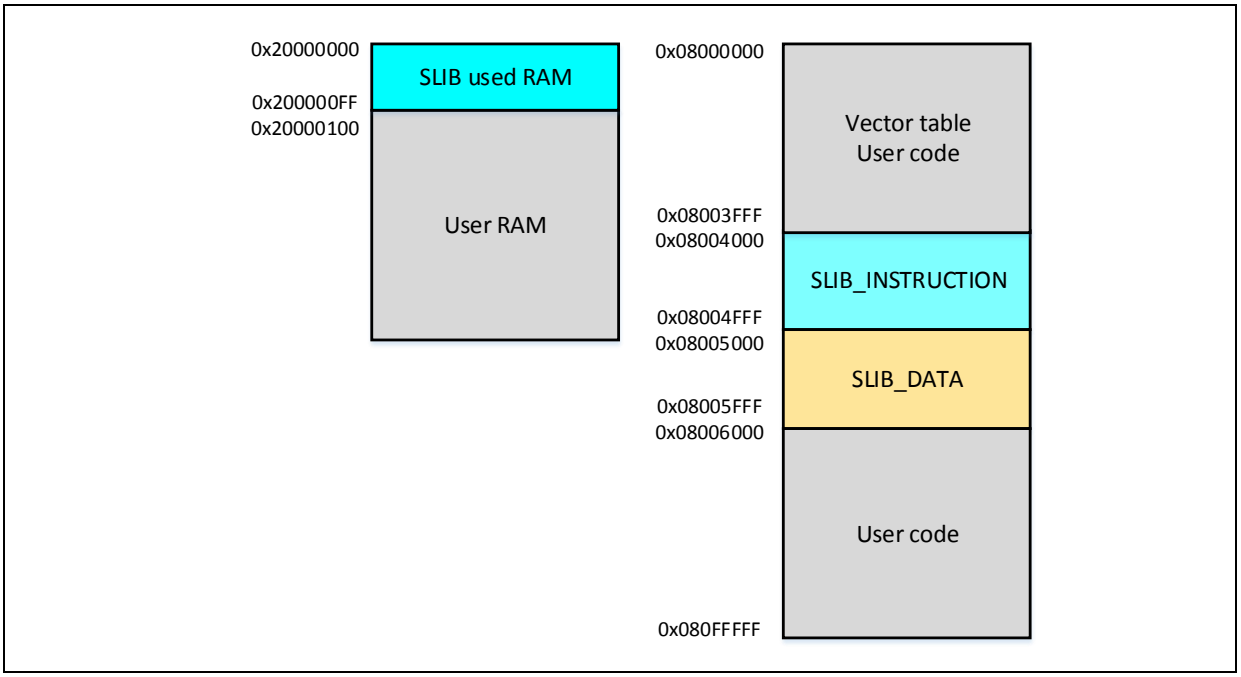
图 3. 设置 Miscellaneous



2.2 编排安全库区的地址

Project_L0范例的主闪存映像及RAM的使用分区如下图，RAM的分区主要是为了避免 SLIB保护区的代码与终端用户的代码用到相同的RAM而产生的冲突问题。

图 4. 主闪存映像及 RAM 的使用分区



其中滤波器函数的代码编排放置到主闪存区的地址0x08004000 ~ 0x08004FFF(sector 8 ~ 9)，并将滤波器函数的系数编排放置到主闪存区的地址0x08005000 ~ 0x08005FFF(sector 10 ~ 11)。RAM的部分则是将0x20000000 到0x200000FF共256个字节保留给SLIB保护区的代码使用。

步骤如下：

1) 依据”AT32F403AxC_FLASH.ld” linker descriptor文件做修改，编写一个ld文件，如project_l0\eclipse_gcc\ld目录下的 slib.ld。

2) 在slib.ld当中，将主闪存及RAM分区划分如下：

```
/* Specify the memory areas */
MEMORY
{
FLASH_1 (rx)     : ORIGIN = 0x08000000, LENGTH = 16K
SLIB_INST (x)    : ORIGIN = 0x08004000, LENGTH = 4K
SLIB_DATA (r)    : ORIGIN = 0x08005000, LENGTH = 4K
FLASH_2 (rx)     : ORIGIN = 0x08006000, LENGTH = 1000K
SLIB_RAM (xrw)   : ORIGIN = 0x20000000, LENGTH = 0x100  /* used for SLIB code */
RAM (xrw)        : ORIGIN = 0x20000100, LENGTH = 96K - 0x100
}
```

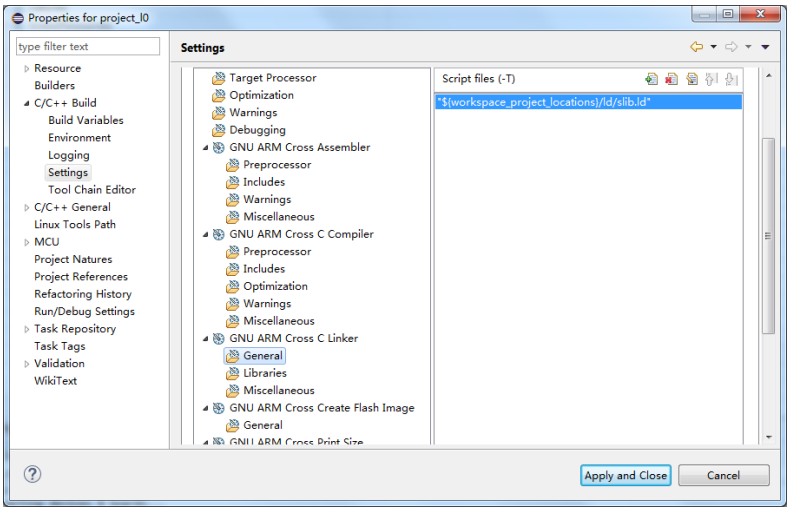
3) 将算法代码放到 .slib_inst section，低通滤波器的系数放到 .slib_data section，并将算法使用到的全局变量指定到 .slib_ram section，如下图：

图 5. 配置 code、data、ram section

```
/* Define output sections */
SECTIONS
{
  /* The startup code goes first into FLASH */
  .isr_vector :
  {
    . = ALIGN(4);
    KEEP(*(.isr_vector)) /* Startup code */
    . = ALIGN(4);
  } >FLASH_1

  .slib_inst :
  {
    . = ALIGN(4);
    *fir_filter.o (.text .text*);
    *arm_fir_f32.o (.text .text*);
    *arm_fir_init_f32.o (.text .text*);
    . = ALIGN(4);
  } > SLIB_INST

  .slib_data :        /* SLIB_DATA arae */
  {
    . = ALIGN(4);
    *fir_coefficient.o (.rodata .rodata*);
    . = ALIGN(4);
  } > SLIB_DATA

  .slib_ram :          /* Used for SLIB */
  {
    . = ALIGN(4);
    *fir_filter.o (.data .data*);
    *fir_filter.o (.bss .bss*);
    . = ALIGN(4);
  } > SLIB_RAM
```

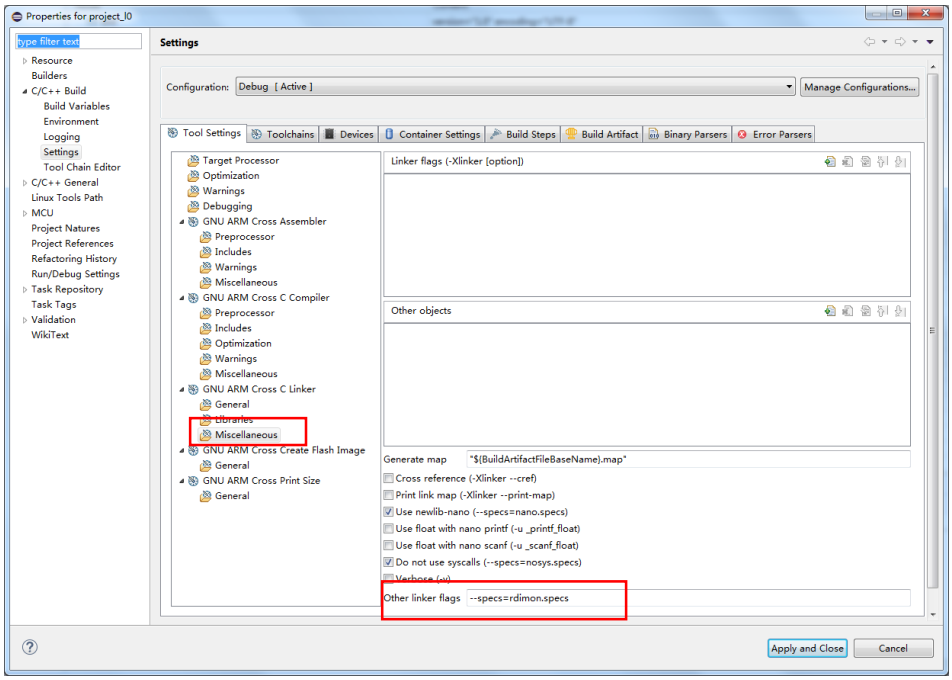
4) 在Project->Properties->C/C++ Build->Setting->GNU ARM Cross C Linker->General设定中的Script files，加入slib.ld。

图 6. 设置 Script files



5) 本范例会使用到 gcc 的数学运算函数库 libm.a，在Properties->GNU ARM Cross C Linker->Miscellaneous设定中的 Other linker flags填入--specs=rdimon.specs，linker才不会出现错误讯息。

图 7. 设置 Other linker flags



关于ld文件的语法，可参考 GNU linker 的相关名文件。

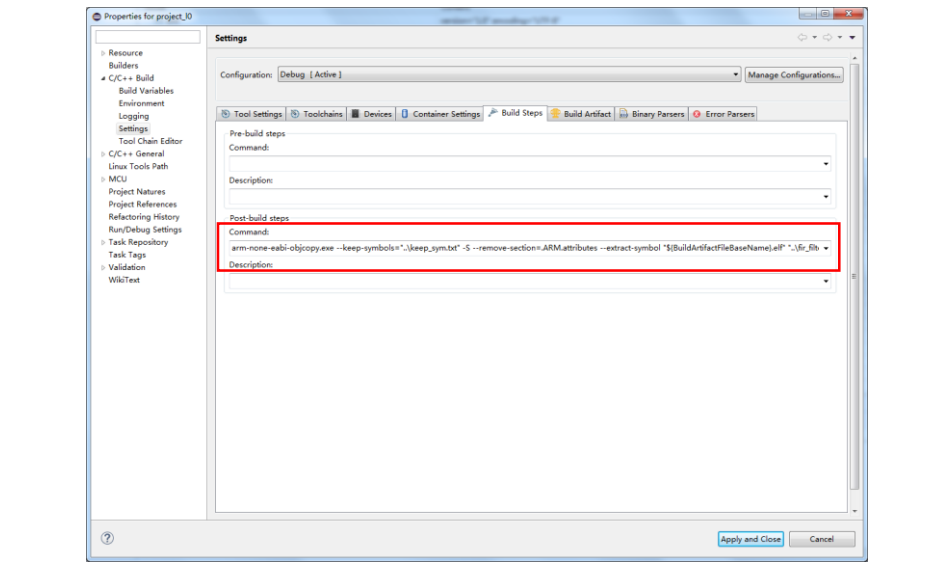
2.3 产生头文件及符号定义文件

头文件(header file)跟符号定义文件(symbol definition file)是终端客户应用范例Project_L1在调用FIR低通滤波函数时需要用到。在范例中，就是main.c中包含的fir_filter.h 文件。符号定义文件则是定义低通滤波函数的实际地址。

产生符号定义文件的方法：

1) 进入Project->Properties->C/C++ Build->Setting的Build Steps设定画面。

图 8. 设置 Build Steps



在 Post-build steps的命令行中输入以下命令：

arm-none-eabi-objcopy.exe --keep-symbols="..\keep_sym.txt" -S --remove-section=.ARM.attributes

--extract-symbol "${BuildArtifactFileBaseName}.elf" "..\fir_filter_symbol.sym"

2) 此处 fir_filter_symbol.sym是要产出的符号定义文件，keep_sym.txt放在project_l0\eclipse_gcc目录下，是用来选择要产生哪些函数的符号，内容如下：

```
FIR_lowpass_filter
```

3) 重新编译整个项目后，在project_l0\eclipse_gcc目录下，就会产生一个名为fir_filter_symbol.sym的符号定义文件。

2.4 启用安全库区保护

要启用安全库区的保护功能，有以下两种方式：

1) 使用ICP刻录工具 Artery ICP Programmer(建议用此方式)。

使用ICP Programmer 启用SLIB的方法，请参阅《AT32F403A Security Library Application Note》。

2) 使用范例程序main.c之中的slib_enable()函数。

在低通滤波函数测试正确后执行过一次此函数，就可以启用安全库区的保护功能。要执行此函数，只要在main.c中使能 #define USE_SLIB_FUNCTION 即可。

3 Project_L1 终端用户范例

Project_L1范例会使用到在Project_L0中调试好，并已经被刻录到AT32F403A芯片的主闪存中且被SLIB保护的FIR低通滤波器函数。根据 Project_L0提供的头文件、符号定义文件以及主闪存区块映像，终端用户就可以参照Project_L1做到

- 建立一个应用项目；
- 引用Project_L0提供的头文件及符号定义文件到项目里；
- 调用FIR低通滤波器函数；
- 开发并调试用户自己的应用程序。

注： *Project_L1必须使用跟Project_L0开发时一样的工具链及相同版本的编译程序，不然有可能会因为版本差异的兼容性问题，而无法使用Project_L0提供的代码。*

3.1 建立用户的应用项目

因为Project_L0启用的安全库区已经占用了一些特定的主闪存页面，Project_L1的代码必须参照Project_L0提供的主闪存区块映像来编排放置的地址。其中sector 8~11为安全库区所占用，终端用户需使用ld文件将这个区域隔离起来，避免代码在编译时被编排到这个区域内，方式如下：

参照project_l1\eclipse_gcc\ld目录下的end_user_code.ld文件，将主闪存空间切成两个区块FLASH_1及FLASH_2，中间空出来的区域就是SLIB保护区。此外，RAM的区域也要保留0x20000000到0x200000FF的区域。如下图：

图 9. end-user-code.ld 配置

```
/* Specify the memory areas */
MEMORY
{
FLASH_1 (rx)     : ORIGIN = 0x08000000, LENGTH = 16K
SLIB_INST (x)    : ORIGIN = 0x08004000, LENGTH = 4K
SLIB_DATA (r)    : ORIGIN = 0x08005000, LENGTH = 4K
FLASH_2 (rx)     : ORIGIN = 0x08006000, LENGTH = 1000K
SLIB_RAM (xrw)   : ORIGIN = 0x20000000, LENGTH = 0x100  /* used for SLIB code */
RAM (xrw)        : ORIGIN = 0x20000100, LENGTH = 96K - 0x100
}

/* Define output sections */
SECTIONS
{
  /* The startup code goes first into FLASH */
  .isr_vector :
  {
    . = ALIGN(4);
    KEEP(*(.isr_vector)) /* Startup code */
    . = ALIGN(4);
  } >FLASH_1

  .slib_inst (NOLOAD):
  {
    *(.slib_code)
  } > SLIB_INST

  .slib_data (NOLOAD) :
  {
    KEEP(*(.slib_data))
  } > SLIB_DATA

  .slib_ram :
  {
    KEEP(*(.slib_ram))
  } > SLIB_RAM
```

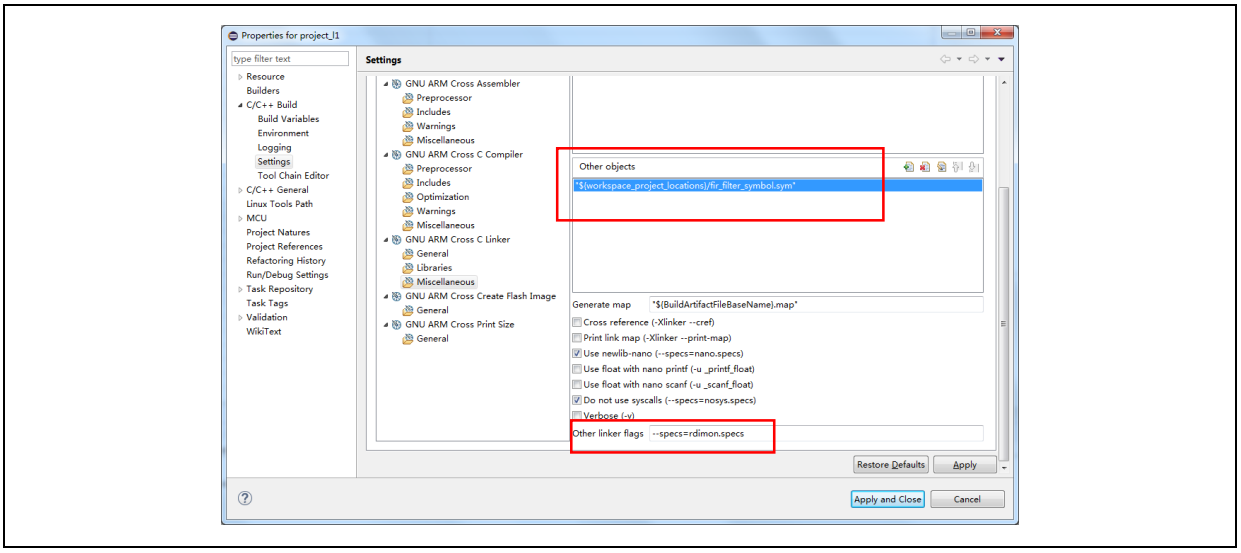
其中 SLIB_CODE及SLIB_DATA两个区域，方案商已事先刻录代码，所以设定为NOLOAD，在下载Project_L1代码到主闪存时，就不会再次被下载。

3.2 在项目中加入符号定义文件

Project_L0所产生的符号定义文件fir_filter_symbol.sym必须被添加到Project_L1项目中，才能被正确的编译并链结到SLIB保护区的代码。方法如下：

1) 将fir_filter_symbol.sym 这个文件加到FIR_Filter群组；
2) 打开Project->Properties->C/C++ Build->Settings->Tool Setting->GNU ARM Cross C Linker->Miscellaneous设定画面，在Other objects选单中加入此文件，在编译项目时就会可以被引用到。

图 10. 设置 Other objects



4 版本历史

表 1. 文档版本历史

日期	版本	变更
2022.01.06	2.0.0	最初版本

重要通知 - 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途(及其依据任何司法管辖区的法律的对应情况)，或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力的产品不得应用于武器。此外，雅特力产品也不是为下列用途而设计并不得应用于下列用途：(A) 对安全性有特别要求的应用，例如：生命支持、主动植入设备或对产品功能安全有要求的系统；(B) 航空应用；(C) 汽车应用或汽车环境，且/或(D) 航天应用或航天环境。如果雅特力产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，采购商仍将独自承担因此而导致的任何风险，雅特力的产品设计规格明确指定的汽车、汽车安全或医疗工业领域专用产品除外。根据相关政府主管部门的规定，ESCC、QML 或 JAN 正式认证产品适用于航天应用。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。