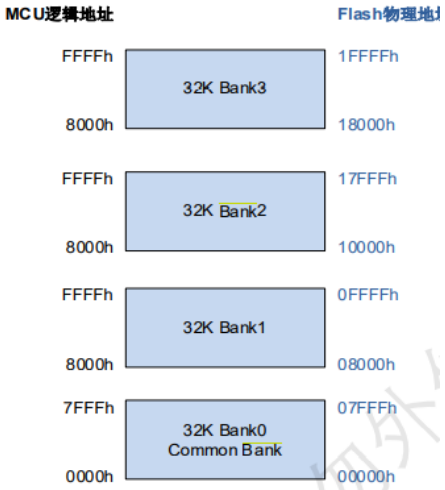
目录

1 BANK 技术简介

标准的 8051 单片机能寻址 64KB 的代码空间，在对超过 64KB 的代码，单片机采用 Code Bank 的方式来扩展程序空间。Code Bank 的机理就是将地址空间分成小于或等于 64KB 的代码段，通过一个特殊功能寄存器实现程序在不同的代码空间跳转。而 Keil C51 中提供了 Code Bank 的支持，最多可管理 32 个每个最大达 64K 的 Bank，从而达到总共 2MB 的代码空间。

2 赛元 128K 单片机 BANK 相关说明

赛元 128K 单片机的 128K Flash ROM 分为 4 个 Bank,Bank 大小为 32k，结构如下：



赛元 128K 单片机 Flash 结构

赛元 128K 单片机最大寻址区间为 64K，既两个 Bank，其中 Bank0 固定,BANK1~3 可切换。Bank1~3 跳转方式为：用户程序中关闭中断->跳转至 Bnak0（切换 Bank 的代码存放在 Bnak0）->修改 ROMBANK、DATABANK。

Bank 相关寄存器：ROMBNK(DFH)程序 Bank 切换寄存器(读/写)：

位编号	7	6	5	4	3	2	1	0
符号	-	-	DATABNK1	DATABNK0	-	-	ROMBNK1	ROMBNK0
读/写	-	-	读/写	读/写	-	-	读/写	读/写
上电初始值	x	x	0	1	x	x	0	1

位编号	位符号	说明
5~4	**DATABNK [1:0]**	**目标数据地址 Bank 切换位，控制 ROM 高 32K 地址目标 MOVC 和 IAP 指向的区域：** 00：MOVC 和 IAP 烧写都针对 Bank0，此时 IAP 寻址范围只是 32K 01：MOVC 和 IAP 烧写都针对 Bank0 和 Bank1 10：MOVC 和 IAP 烧写都针对 Bank0 和 Bank2 11：MOVC 和 IAP 烧写都针对 Bank0 和 Bank3 **注意：该控制位仅在 IAPADE=0x00 时有效**
1~0	**ROMBANK[1:0]**	**取指操作 Bank 切换位:** 00：指令之于 Bank0； 01：指令之于 Bank0 和 Bank1； 10：指令之于 Bank0 和 Bank2； 11：指令之于 Bank0 和 Bank3；
7~6,3~2	-	保留

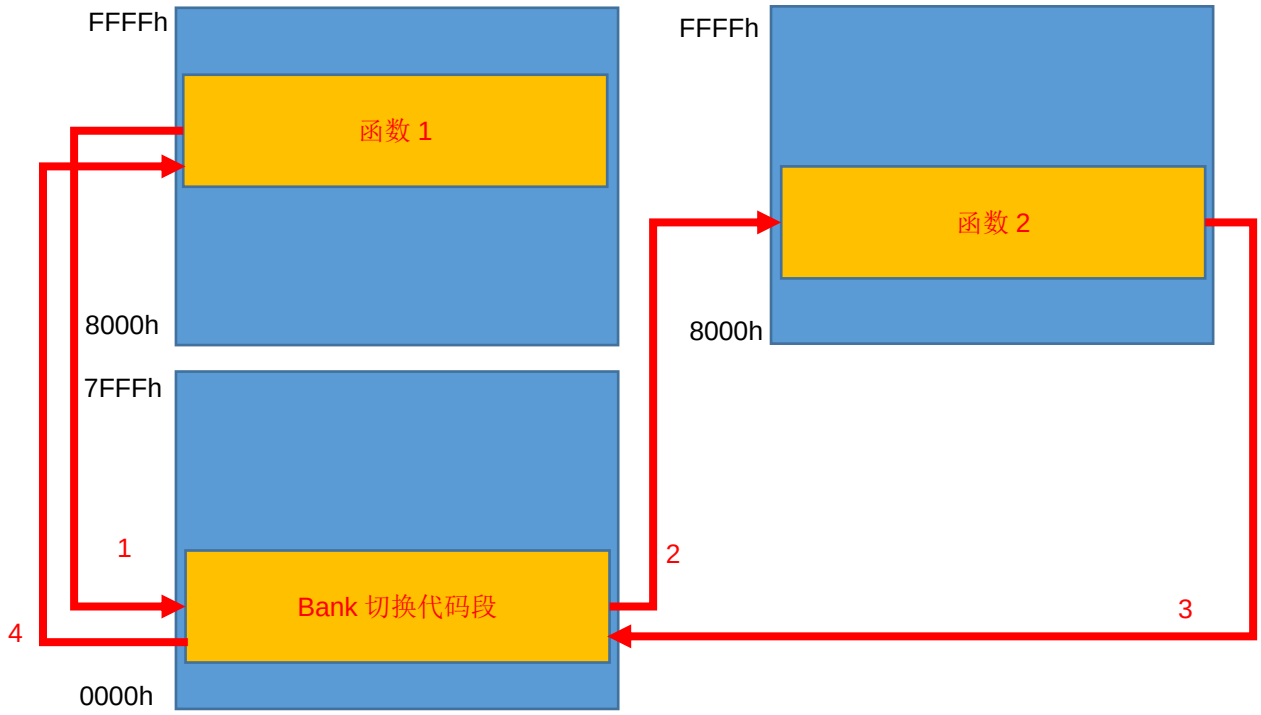
注意事项：在 ROMBNK 进行读值操作时，建议定义一个 xdata 变量。

3 分 BANK 程序的软件结构解析

Bank 的硬件存储结构决定其在软件设计过程中需要软件上做相应的软件结构设计，虽然代码通过扩大 Flash 空间存储下来了，但是 51 内核程序最大寻址依然是 64K，Bank1~Bank3 的内容是共用 MCU 的高 32K 逻辑地址的，因此进行跨 Bank 操作时，需要通过设置特殊功能寄存器重映射高 32K 地址的内容，而这个动作我们在程序设计上将其称为 Bank 切换，每个 Bank 都对应一个 Bank 切换，我们把这些 Bank 切换的程序集中在一起成为一个程序块，将其称为“Bank 切换代码段”。由于这段代码服务于各个 Bank,因此此代码段需要存储在 Common 区中，同理，中断服务函数应该随时能够准确响应，因此中断服务函数也需要存储在 Common 中，而被 Code 声明的全局变量需要被所有的 Bank 访问所以也需要放在 Common 中，这样 Bank 结构下的程序结构就如下图所示：

赛元 128K 单片机 存储结构

当出现跨区域函数调用时就需要调用一次 Bank 切换再 LCALL 到对应的函数中，而同一 Bank 中进行函数调用则不需要经过 Bank 切换代码段处理，程序的执行走向如下图所示。



跨 Bank 函数调用示意图

在实际应用中我们不用关心 Bank 的切换，Bank 切换动作编译器会处理。

函数切换的时候会改动 ROMBNK 寄存器进行切 Bank 操作，实际应用时需要注意，避免 ROMBNK 更改值不在预期。

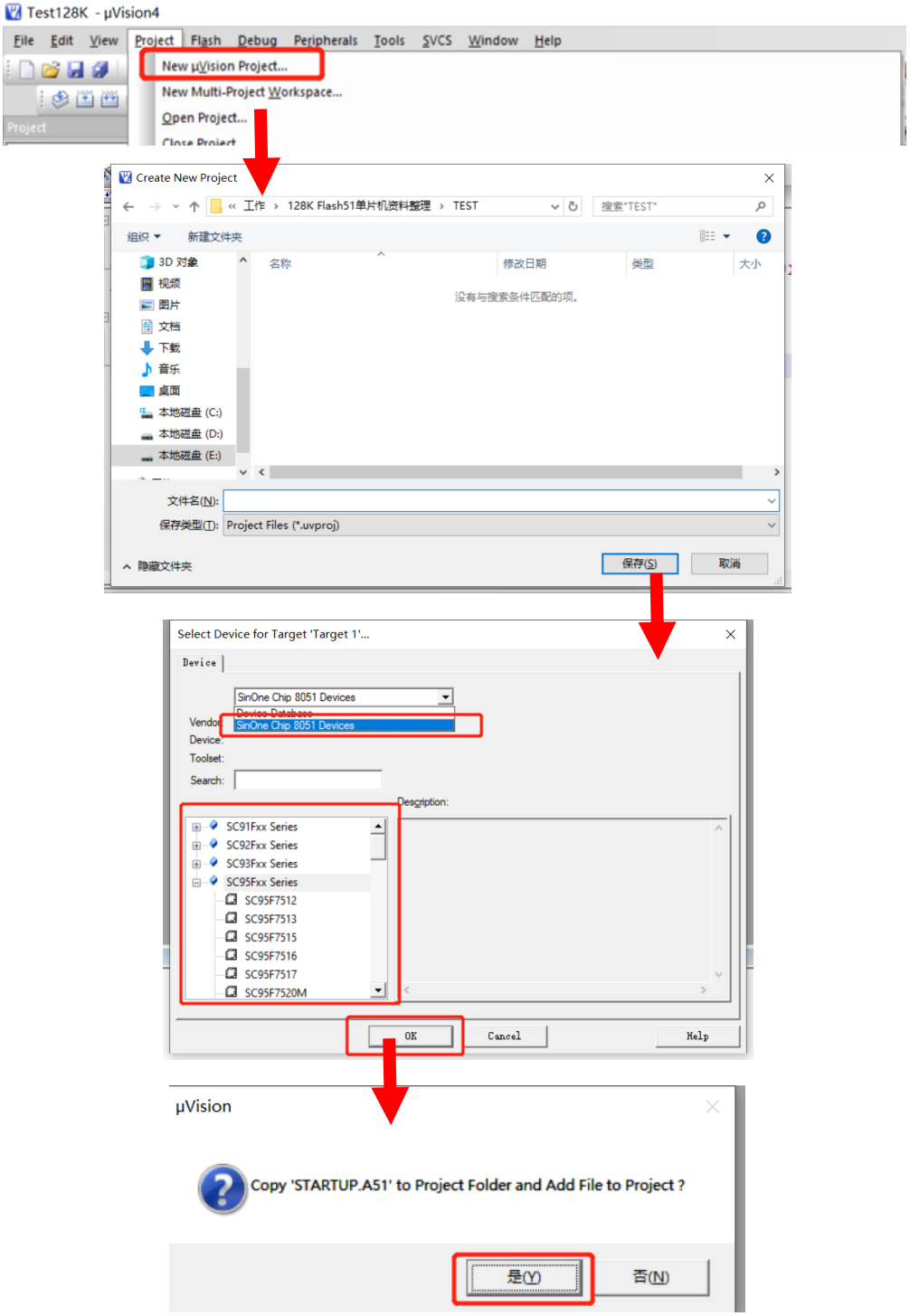
4 分 BNAK 的 KEIL 工程搭建

Keil 软件本身就支持基于 Bank 结构的 51 单片机的应用程序开发，使用 Keil 实现代码项目主要有三个操作：

1. 设置 Keil 用于 bank 应用的相关软件配置 。
2. L51_BANK.A51 和 STARTUP.A51 源代码文件配置，以适应芯片 bank 存储结构。
3. 设置每个源文件的 Bank 编号。

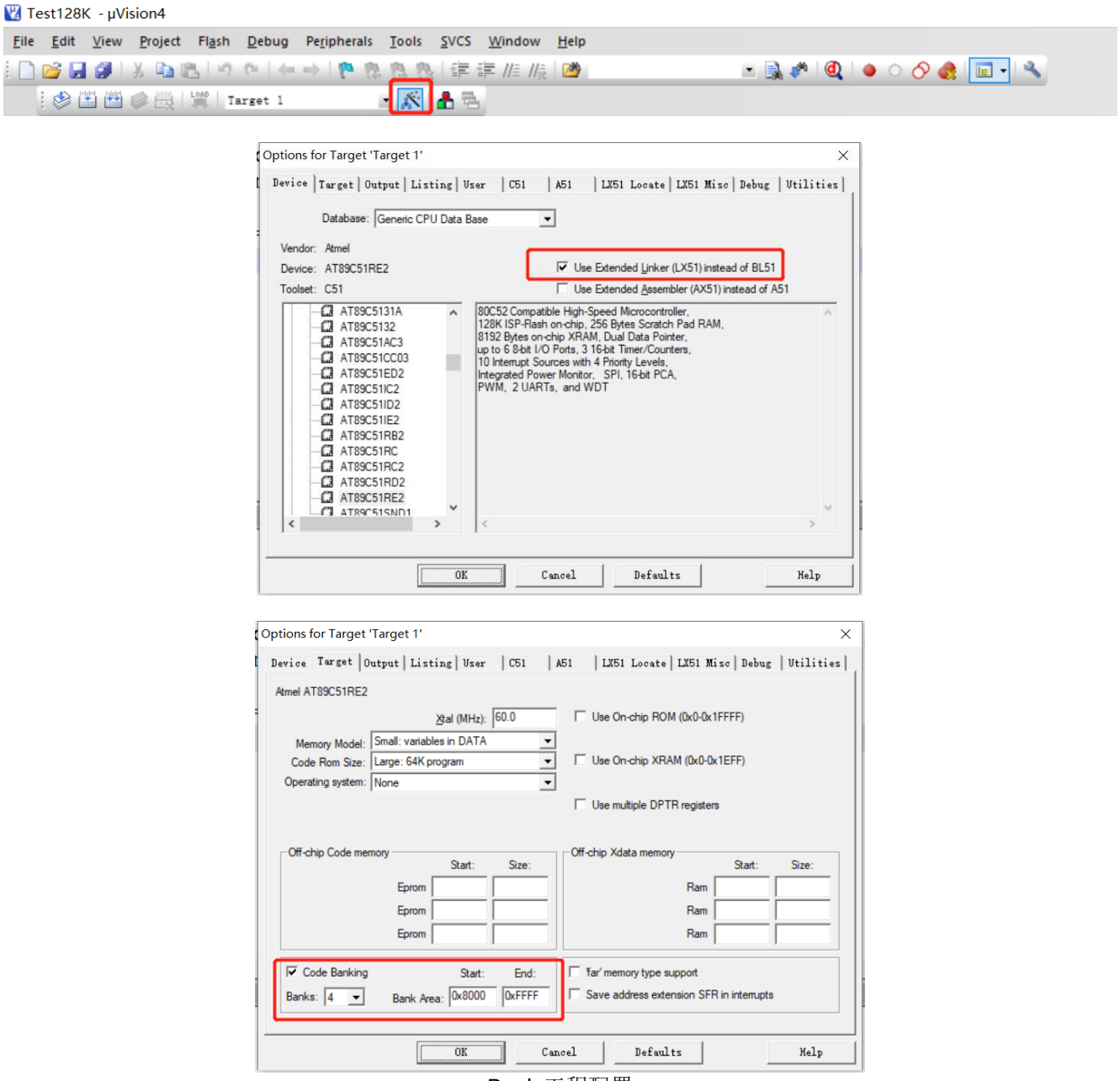
在 keil 上搭建一个 Bank 项目开发环境的过程如下：

1. 新建一个 Keil 工程，选择 Project->New uVision Project，选择芯片，确认后点是导入 StartUp 文件

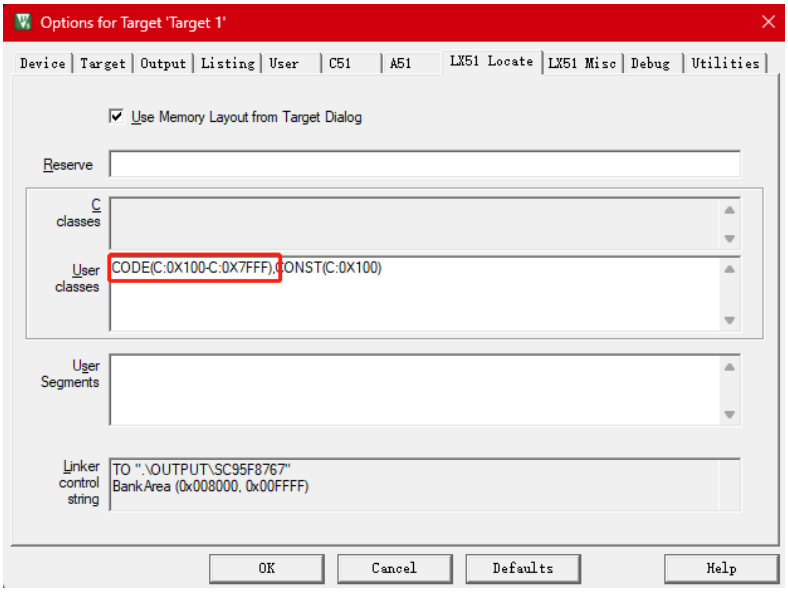


新工程创建流程

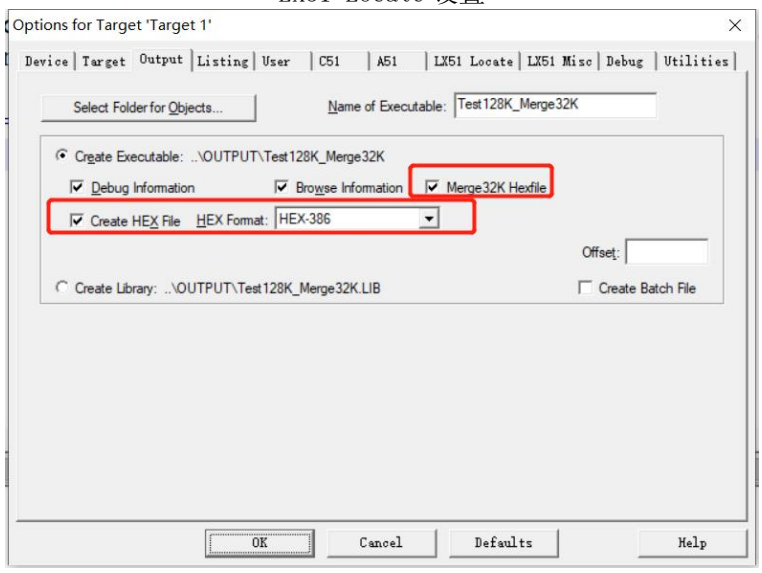
2. 点击魔术棒进行 KEIL 上 Bank 相关的软件设置：
   1) 选择使用 LX51 编译；
   2) 勾选 Code Banking 设置 Banks 数量以及地址(0x8000、0XFFFF)；
   3) 配置 LX51 Locate，配置 Common 区域为 0x100 到 0x7FFF（设置 COMMON 区域代码小于等于 32K）
   4) 设置 output 输出 Hex Fomat 为 Hex386（默认为 Hex80，该格式无法输出大于 64K 的程序）；
   5) 设置烧录和仿真工具；



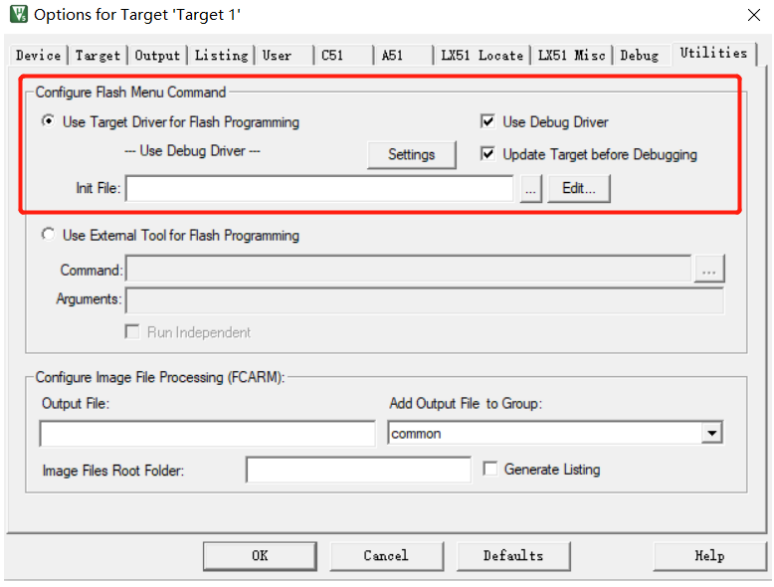
Bank 工程配置



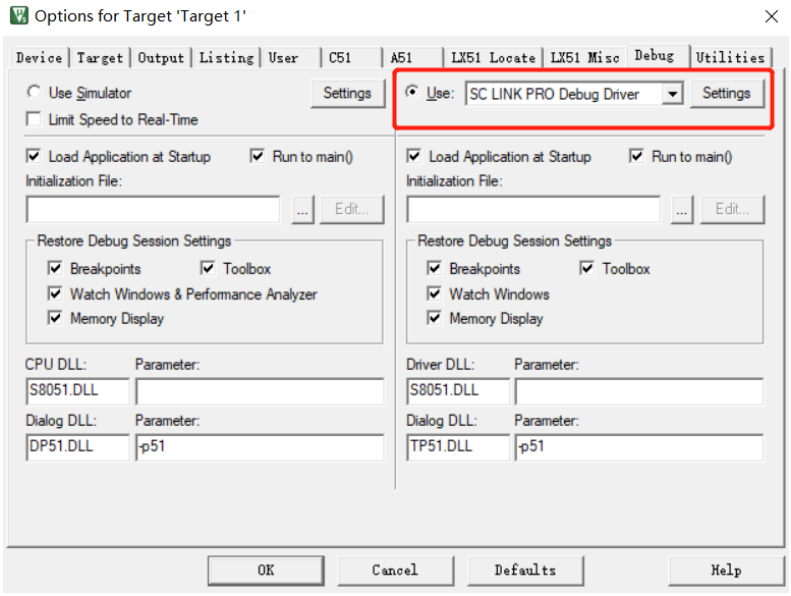
LX51 Locate 设置



Hex 文件生成设置



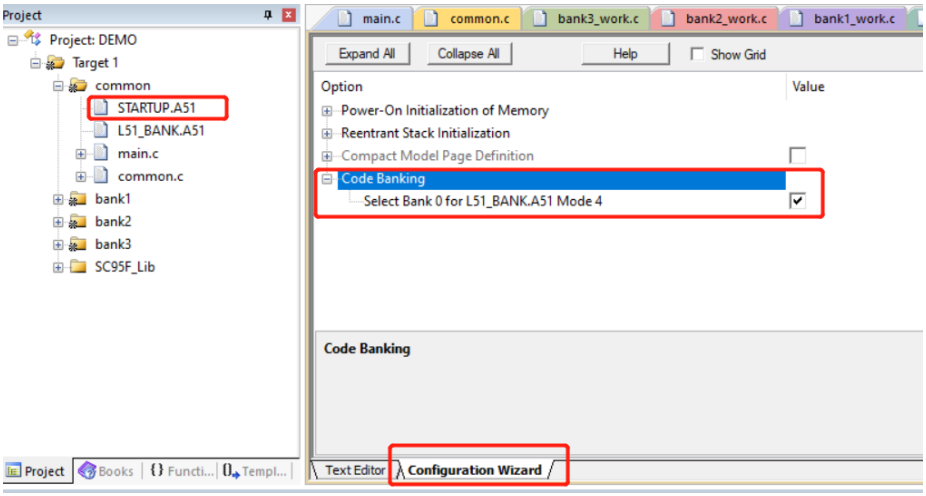
仿真烧录设置



仿真烧录设置

3. 从 Keil\C51\LIB 文件夹中把 L51_BANK.A51 文件导入工程并对 STARTUP.A51 和 L51_BANK.A51 中的相关宏设置进行修改.

STARTUP.A51 设置:



Code Banking 设置

L51_BANK.A51 文件设置:

设置 Bank 数量、选择使用 Bank 开关切换 Code：

```
$NOMOD51 NOLINES
$NOCOND
;------------------------------------------------------------------------------
;  This file is part of the BL51 / LX51 Banked Linker/Locater package
;  Copyright (c) 1988 - 2005 Keil Elektronik GmbH and Keil Software, Inc.
;  Version 2.21 (Code and Variable Banking for Classic 8051 Derivatives)
;------------------------------------------------------------------------------
;************************ Configuration Section ********************************
?B_NBANKS       EQU  4    ; Define maximum Number of Banks                    *
;                         ; following values are allowed: 2, 4, 8, 16, 32, 64 *
;                         ; for BL51 the maximum value for ?B_BANKS is 32     *
;                         ; for LX51 the maximum value for ?B_BANKS is 64     *
;                                                                             *
?B_MODE         EQU  4    ; 0 for Bank-Switching via 8051 Port                *
;                         ; 1 for Bank-Switching via XDATA Port               *
;                         ; 4 for user-provided bank switch code              *
;                                                                             *
?B_RTX          EQU  0    ; 0 for applications without real-time OS           *
;                         ; 1 for applications using the RTX-51 real-time OS  *
;                                                                             *
?B_VAR_BANKING  EQU  0    ; Variable Banking via L51_BANK (far memory support)*
;                         ; 0 Variable Banking does not use L51_BANK.A51      *
;                         ; 1 Variable Banking uses this L51_BANK.A51 module  *
; Notes: ?B_VAR_BANKING uses the 'far' and 'far const' C51 memory types to    *
;        extent the space for variables in RAM and/or ROM of classic 8051     *
;        device.  The same hardware as for code banking is used.  Program     *
;        code banking and variable banking share the same hardware I/O pins.  *
;        The C51 Compiler must be used with the VARBANKING directive.         *
;        Variable Banking is only supported with the LX51 linker/locater.     *
;                                                                             *
?B_RST_BANK     EQU  0xFF ; specifies the active code bank number after CPU   *
;                         ; Reset.  Used to reduce the entries in the         *
;                         ; INTERBANK CALL TABLE.  The value 0xFF disables    *
```

L51_BANK.A51 文件截图

这些区域对应的是 Bank 切换程序，编译时是需要切换 Bank 调用时 Keil 会自动调用。

```
;    saving are:  DPTR and ACC.                                              *
;                                                                            *
;                                                                            *
LONG_MACRO      EQU  0    ; 0 default, for normal macros and up to 8 banks    *
;                         ; 1 big macro code or many banks                   *

ROMBANK         DATA    0DFH          ; Select ROMBANK Address               *

SWITCH0         MACRO                 ; Switch to Memory Bank #0             *

                MOV     ROMBANK,#00h  ; Set ROMBANK                          *

                ENDM                  ;                                      *
;                                                                            *
SWITCH1         MACRO                 ; Switch to Memory Bank #1             *

                MOV     ROMBANK,#11h  ; Set ROMBANK                          *

                ENDM                  ;                                      *
;                                                                            *
SWITCH2         MACRO                 ; Switch to Memory Bank #2             *

                MOV     ROMBANK,#22h  ; Set ROMBANK                          *

                ENDM                  ;                                      *
;                                                                            *
SWITCH3         MACRO                 ; Switch to Memory Bank #3             *

                MOV     ROMBANK,#33h  ; Set ROMBANK                          *

                ENDM                  ;                                      *
;                                                                            *
;----------------------------------------------------------------------------*
ENDIF;                                                                       *
```

设置ROMBANK寄存器的地址

切换到Bnak0

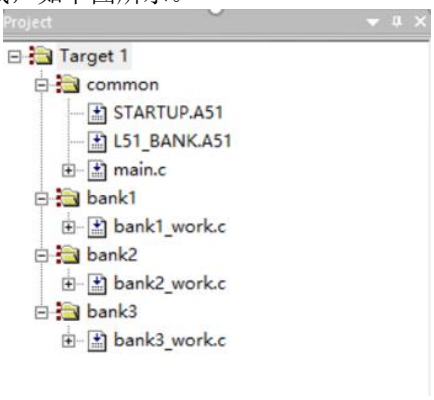
切换到Bank1

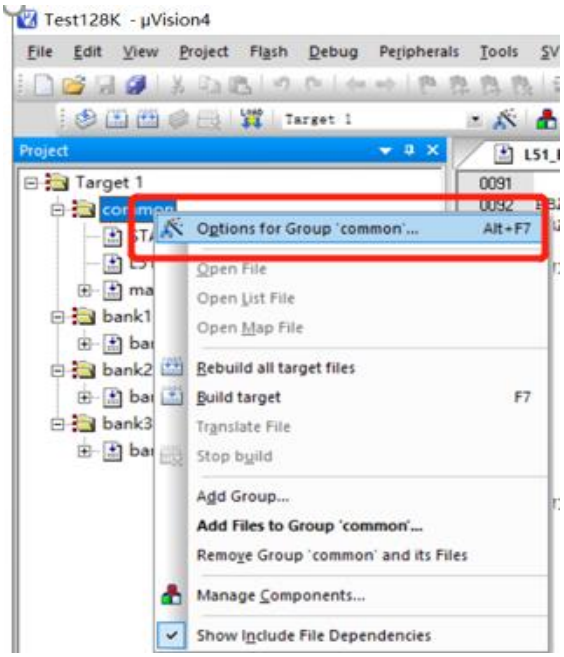
切换到Bank2

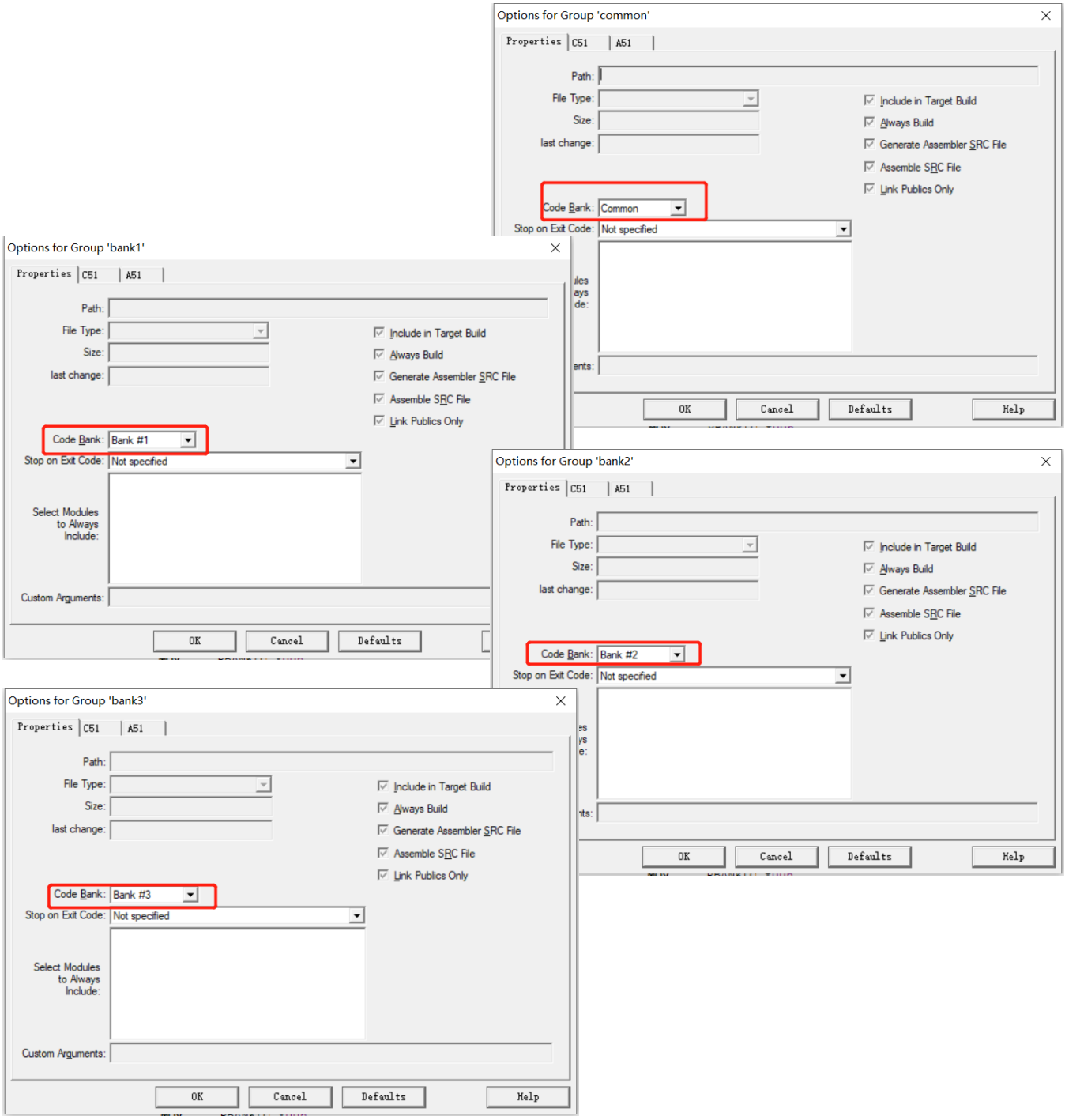
切换到Bank3

L51_BANK.A51 文件切 Bank 代码截图

4、创建四个文件夹，分别命名为 common、bank1、bank2、bank3，并将其文件设置下的 Code Bank 分别设置为 Common、Bank #1、Bank #2、Bank #3，之后对应的源文件只要导入到对应的文件夹下其代码就会被编译到对应的 Bank 区域，如下图所示。





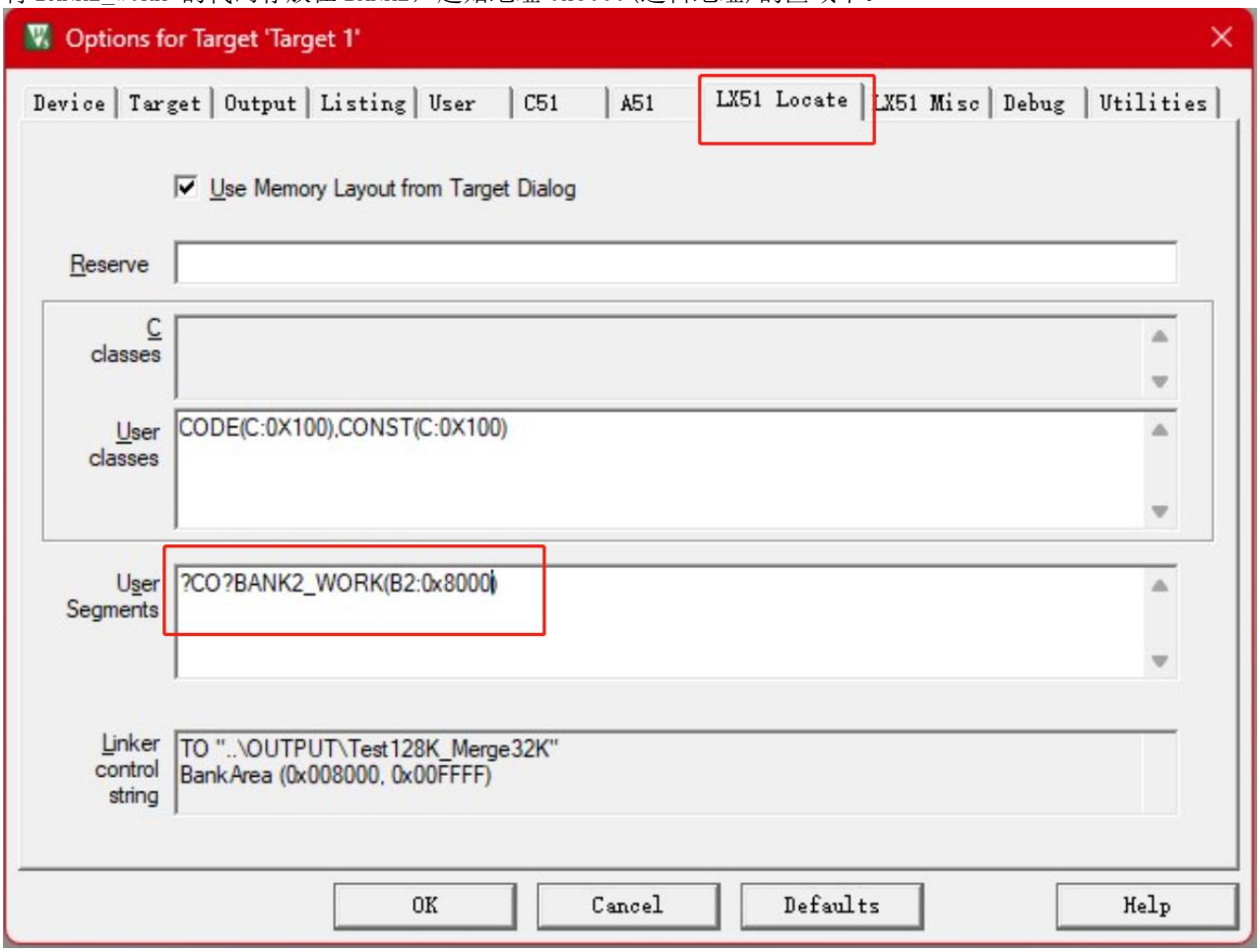


文件所在 Bank 设置

通过以上一系列的设置后，用户在编写程序的时候就无需再关心何时需要进行 Bank 切换动作，Keil 编译器会自动识别出当前程序是否需要进行 Bank 切换，从而减弱了 C 语言编程者对底层寄存器工作情况的关注，减弱了 Bank 模式下开发 51 单片机与普通 64K 的 51 单片机应用开发的区别。

5 使用 KEIL 开发 BANK51 项目的注意事项

1. 虽然 KEIL 编译器在一定程度上免除了大部分时候用户在 Bank 切换环节的参与，但如果想让芯片效率最优用户是需要根据函数调用情况安放程序到合理的 Bank 中以减少 Bank 切换频率。
2. 每个 Bank 都是按 32K 的空间进行划分的，因此用户的任意一个 C 程序模块都不允许超过 32K。
3. KEIL 只会为 C 语言区域的跨 Bank 程序调用增加 Bank 切换动作，对于汇编代码如果存在跨区域操作则需要用户手动添加。
4. 指定 Code 存储的全局变量无论在哪个 C 文件下都会被编译到 Common 区域。当用户需要将数据量大的常量数据，则可以将该数据单独放在一个.C 文件中，并在 user segments 中定义相应的地址，使对应 C 文件中的 CODE 定义的数据放在相应的地址，Bx 为所对应的 bank。
   例如，将常量数组 char Code Chr[65530]定义在 BANK2_WORK.C 文件中，并在 user segments 中设置，将 BANK2_WORK 的代码存放在 BANK2，起始地址 0x8000(逻辑地址)的区域中。



5. 在访问该常量数组时，系统是不知道存放的 bank，所以需要手动切换 MOVC 所指向的 BANK，注意在手动切换 BANK 读取完数据后需要将 ROMBNK 回复为原来的值，以防止 MOVC 指向错误。
6. 被说明为 interrupt 的中断服务函数必须放在 Common 区域的 C 文件中。
7. 普通 64K 的 51 单片机产生的 HEX 文件是 HEX-80 格式的，仅支持 64K 地址信息的存储，Bank 模式下开发输出的 HEX 文件通常都选为 HEX-386 格式，这样 1 个 HEX 文件就可以存储大于 64K 的代码信息。

6 版本记录

版本	记录	日期
V1.0	初版	2021 年 5 月
V1.1	修正使用范围（现为赛元 128K 单片机），纠正 L51_BANK.A51 切 Bank 代码截图	2022 年 1 月
V1.2	修正 L51_BANK.A51 bug	2022 年 5 月
V1.3	定义在 ROM 的常量数据的注意事项	2022 年 5 月
V1.4	新增配置 Common 区域范围小于等于 32K	2023 年 3 月
V1.5	新增 ROMBNK 读值操作的注意事项	2023 年 8 月

7 声明