

概述

适用范围	
SPC1125 系列	SPC1125, SPC1128, SPD1121
SPC1168 系列	SPC1155, SPC1156, SPC1158, SPC1168, SPD1148, SPD1178, SPD1188, SPD1163, SPM1173
SPC2168 系列	SPC2168, SPC2165, SPC2166, SPC1198
SPC1169 系列	SPC1169, SPD1179, SPD1176, SPD1177, SPD1179B
SPC2188 系列	SPC1185, SPC2188
SPC1198B 系列	SPC1198B

目录

1	FPU 启用	5
2	IDE 设置 FPU	7
2.1	Keil IDE 设置编译器启用 FPU	7
2.2	IAR IDE 设置编译器启用 FPU	7
2.3	GCC 设置编译器启用 FPU	8
3	FPU 使能检测	9
4	注意事项	10
4.1	打印问题	10
4.2	浮点函数库问题	10

图片列表

图 2-1: Keil 配置界面	7
图 2-2: IAR 配置界面.....	7

SPIN TROL

版本历史

版本	日期	作者	状态	变更
C/0	2024-05-31	HangSu	已过期	1. 首次发布。
C/1	2024-07-30	LemengZhou	已过期	1. 修改为全系列通用文档。
C/2	2025-03-31	CanChai	已发布	1. 适用范围增加。 2. 添加对 SPC1198B 系列的描述。

1 FPU 启用

想要正确启用 FPU 用以加速计算，需要满足以下三要素：

- IDE 界面上开启 FPU，从而给编译器传递正确的编译参数；
- 因为 CM4_FP 只具有 float 类型浮点运算指令，代码常量后必须加 f，例如 0.5f。否则，常量默认被编译器识别为 double 类型（Cortex-M4 对于 Double 类型采用软件模拟），从而不产生 FPU 指令；
- 启用 FPU 单元。

注意：需要以上三点都满足，才能生成浮点指令并启用 FPU 单元。

影响 FPU 正常启用需要三个宏定义的正确配置：

1. `__TARGET_FPU_VFP`：此宏在 KEIL IDE 界面上开启 FPU 后（不同的 IDE 设置方式见[章节 2](#)），编译器会自动定义；
2. `__FPU_PRESENT`：此宏在 Spintrol SDK 中一直被定义为“1”；
3. `__FPU_USED`：当 `__TARGET_FPU_VFP` 被定义，且 `__FPU_PRESENT` 为“1”时，`__FPU_USED` 将被定义为“1”。

所以，只需要在 IDE 界面上开启 FPU 后，就可以确保 FPU 能够正常工作，不需要人为对三个宏进行配置。在 Spintrol SDK 中的 `SystemInit()` 函数会根据 `__FPU_USED` 是否为“1”，而决定是否开启 FPU 单元。

core_cm4.h

```
/** __FPU_USED indicates whether an FPU is used or not.
    For this, __FPU_PRESENT has to be checked prior to making use of FPU
    specific registers and functions.
*/
#if defined ( __CC_ARM )
    #if defined __TARGET_FPU_VFP
        #if ( __FPU_PRESENT == 1)
            #define __FPU_USED      1
        #else
            #warning "Compiler generates FPU instructions for a device without an FPU
(check __FPU_PRESENT)"
            #define __FPU_USED      0
        #endif
    #else
        #define __FPU_USED      0
    #endif
#elif defined ( __GNUC__ )
    #if defined ( __VFP_FP__ ) && !defined( __SOFTFP__ )
        #if ( __FPU_PRESENT == 1)
            #define __FPU_USED      1
        #else
            #warning "Compiler generates FPU instructions for a device without an FPU
(check __FPU_PRESENT)"
        #endif
    #endif
#endif
```

core_cm4.h

```
#define __FPU_USED 0
#endif
#else
#define __FPU_USED 0
#endif

#elif defined ( __ICCARM__ )
#if defined __ARMVFP__
#if ( __FPU_PRESENT == 1 )
#define __FPU_USED 1
#else
#warning "Compiler generates FPU instructions for a device without an FPU
(check __FPU_PRESENT)"
#define __FPU_USED 0
#endif
#else
#define __FPU_USED 0
#endif
#endif
```

Code

```
void SystemInit (void)
{
    #if ( __FPU_USED == 1 )
        SCB->CPACR |= ((3UL << 10*2) |           /* set CP10 Full Access */
                      (3UL << 11*2) );          /* set CP11 Full Access */
    #endif

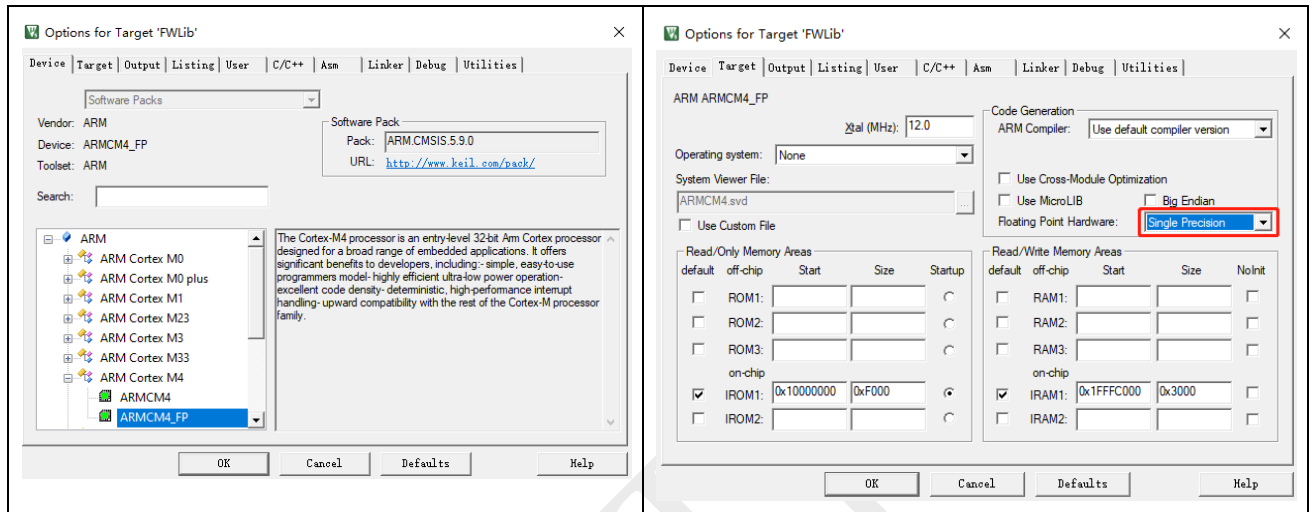
    #ifdef UNALIGNED_SUPPORT_DISABLE
        SCB->CCR |= SCB_CCR_UNALIGN_TRP_Msk;
    #endif
}
```

2 IDE 设置 FPU

2.1 Keil IDE 设置编译器启用 FPU

开启 FPU 指令，如图 2-1 所示。

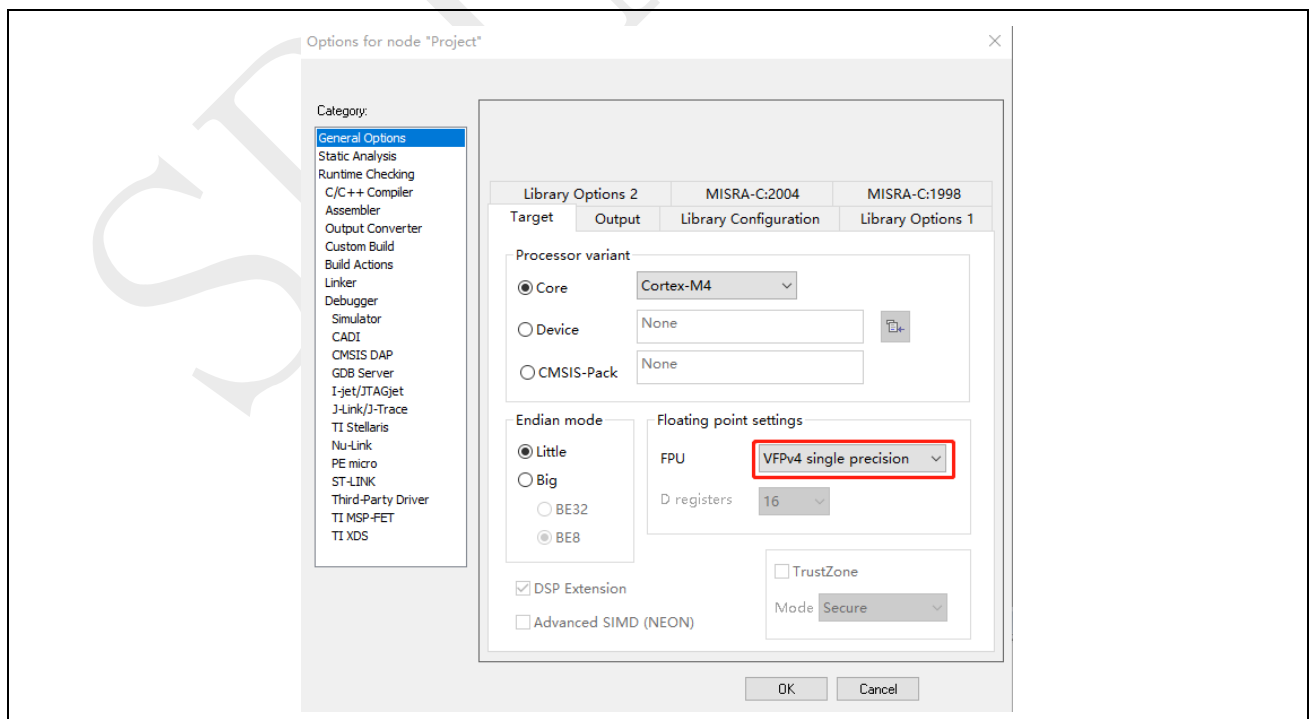
图 2-1: Keil 配置界面



2.2 IAR IDE 设置编译器启用 FPU

开启 FPU 指令，如图 2-2 所示。

图 2-2: IAR 配置界面



2.3 GCC 设置编译器启用 FPU

CMakeLists.txt

```
# Compiler options
target_compile_options(${EXECUTABLE} PRIVATE
    -mcpu=cortex-m4
    -mthumb
    -mfpv4-sp-d16
    -mfloat-abi=hard
    -fdata-sections
    -ffunction-sections
    -Wall
    -O0
    -g3
    -DSPC1169
)

# Linker options
target_link_options(${EXECUTABLE} PRIVATE
    -T${LINKER_FILE}
    -mcpu=cortex-m4
    -mthumb
    -mfpv4-sp-d16
    -mfloat-abi=hard
    -specs=nano.specs
    -specs=nosys.specs
    -lc
    -lm
    -Wl,-Map=${PROJECT_NAME}.map,--cref
    -Wl,--gc-sections
    -Wl,--no-warn-rwx-segments
    -Xlinker -print-memory-usage -Xlinker
)
```


3 FPU 使能检测

可以在 main.c 任意位置加入检测代码，从而检测 FPU 在编译阶段有无使能。

```
main.c
/* If the FPU is not enabled, the compilation is terminated */
# if __FPU_USED == 0
    # error "__FPU_USED == 0"
# endif
```

更进一步需要直接查看汇编码，只有常量后加 f，才能编译出 FPU 指令，从而加速运行。

```
main.c
    a = b * 0.7f;
```

汇编代码

```
0x10000a4c:    ee200a20    . .    VMUL.F32 s0,s0,s1
```

汇编代码中 VMUL.F32 为单精度浮点数乘法指令，S0、S1 为浮点数寄存器，证明开启了 FPU 功能。

4 注意事项

4.1 打印问题

GCC 下浮点数是打印不出来的。如果使用 `-lrdimon -u _printf_float` 实现浮点数打印，编译出代码过大，不符合嵌入式要求。

如需查看结果，可将浮点数放大 10 倍，或 100 倍后转换成整数打印。

1_Application/ Math_Lib/main.c

```

/*****
 * @brief      GccPrint10xFloat
 *
 * @param[in]  f32Value : Float precision is 7 to 8 decimal places
 *                  Integer of Value from -2147483648 to 2147483647
 *
 * @return     none
 *
 *****/
void GccPrint10xFloat(float f32Value)
{
    i32 i32Integer;

    i32Integer = (int32_t)(10 * f32Value);

    printf("10xFloat %d\n", i32Integer);
}

```

4.2 浮点函数库问题

参考《Arm Cortex-M4 Processor Technical Reference Manual》第 68 页可知，CM4 内核用 VSQRT.F32 指令来计算单精度浮点开根号，在 14 个 cycle 计算完毕。在 keil 如果调用 ARM 提供的数学库中的 `__sqrt(x)` 编译时会有 VSQRT.F32 指令，但如果使用 `math.h` 库函数中的 `sqrt(x)` 不会产生 VSQRT.F32 指令，这一点需要额外注意。