STM32F4xx FPU和DSP库的使用

STM32F4xx属于Cortex M4F架构，带有32位的单精度硬件FPU(Float Point Unit)，支持浮点指令集，相对比M0和M3架构，浮点运算性能高出数十倍甚至上百倍。Cortex™M4 FPU是ARM™FPv4-SP单精度FPU一种实现形式。

1、硬件FPU的开启

（1）通过修改代码实现

默认情况下，STM32F4xx的FPU是禁用的，可以通过设置协处理器控制寄存器（CPACR）来开启硬件FPU。在keil编程环境下，可以通过定义全局宏定义标识符**_FPU_PRESENT**和**_FPU_USED**都为1来开启硬件FPU。其中宏定义标识符_FPU_PRESENT用来确认处理是否带有FPU功能，标识符_FPU_USED用来确定是否开启FPU功能。实际上，因为STM32F4是带有FPU功能的，所以在stm32f4xx.h头文件中，默认定义_FPU_PRESENT为1。可以在文件stm32f4xx.h中找到如下定义。

```
159 /**
160   * @brief Configuration of the Cortex-M4 Processor and Core Peripherals
161   */
162 #define __CM4_REV                 0x0001  /*!< Core revision r0p1                          */
163 #define __MPU_PRESENT             1       /*!< STM32F4XX provides an MPU                   */
164 #define __NVIC_PRIO_BITS          4       /*!< STM32F4XX uses 4 Bits for the Priority Levels */
165 #define __Vendor_SysTickConfig    0       /*!< Set to 1 if different SysTick Config is used  */
166 #define __FPU_PRESENT             1       /*!< FPU present                                 */
```

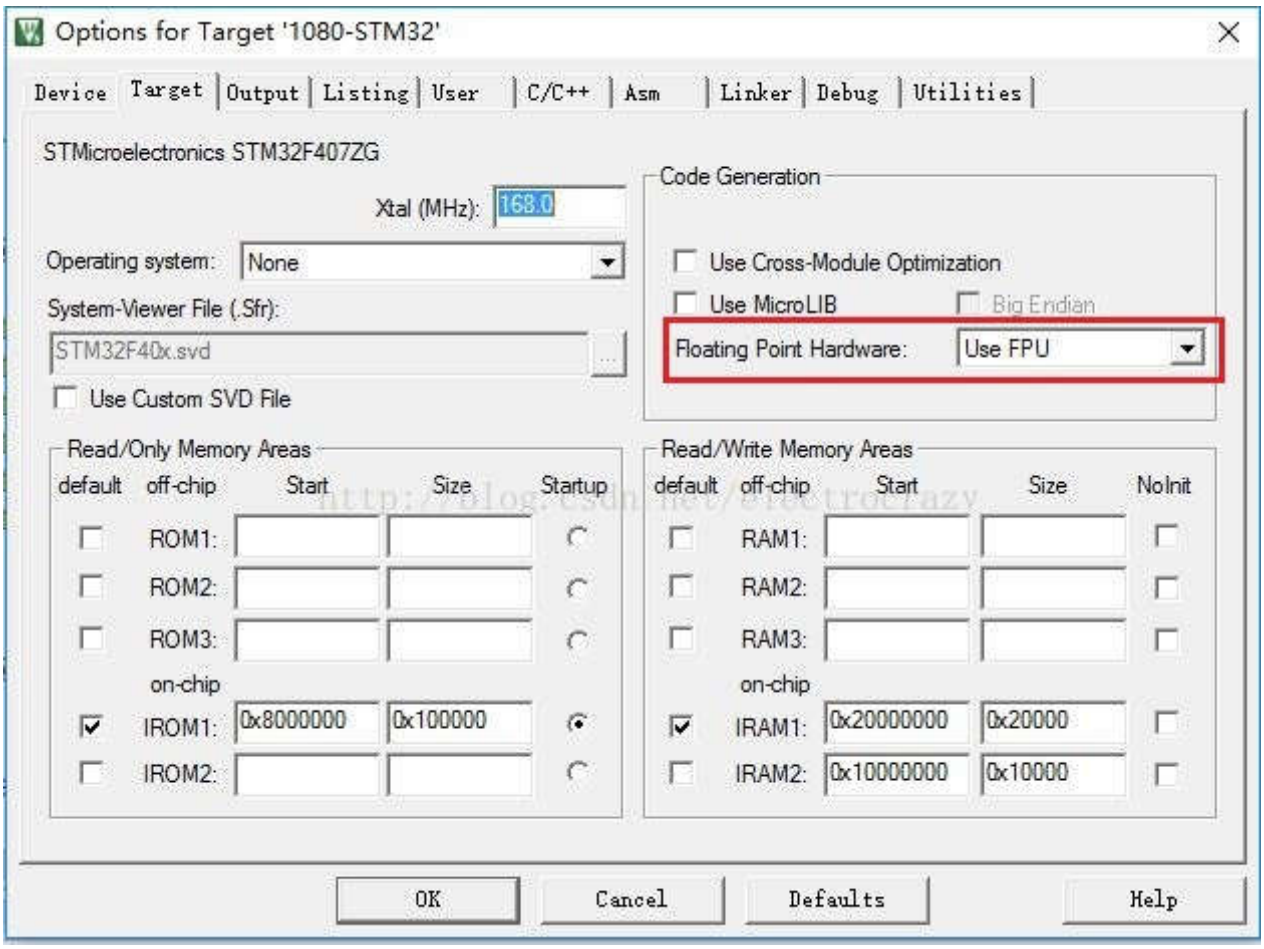
若要开启FPU还需要在头文件stm32f4xx.h中定义标示符_FPU_USED的值为1。即在刚才的宏定义下边添加一个宏定义。

```
159 /**
160   * @brief Configuration of the Cortex-M4 Processor and Core Peripherals
161   */
162 #define __CM4_REV                 0x0001  /*!< Core revision r0p1                          */
163 #define __MPU_PRESENT             1       /*!< STM32F4XX provides an MPU                   */
164 #define __NVIC_PRIO_BITS          4       /*!< STM32F4XX uses 4 Bits for the Priority Levels */
165 #define __Vendor_SysTickConfig    0       /*!< Set to 1 if different SysTick Config is used  */
166 #define __FPU_PRESENT             1       /*!< FPU present                                 */
167 #define __FPU_USED                1       /*!< FPU Enable                                  */
168
```

（2）通过软件设置实现

如果使用的keil是5.0以上的版本，也可以直接在keil里设置开启FPU。点击图标

（Options forTarget...），在”Target”选项卡里的“Code Generation”下的“Floating Point Hardware”下拉菜单中选择”Use FPU”。点击”OK“完成设置。经过设置之后，在程序编译时编译器会自动加入宏定义标识符_FPU_USED为1。



至此则完成硬件FPU的使能。在程序中如果遇到浮点运算就会使用硬件FPU相关指令，执行浮点运算，大大提升系统浮点运算速度。

2、DSP库的使用

STM32F4的Cortex-M4内核不仅内置硬件FPU单元，还支持DSP多种指令集，比如支持单周期乘加指令（MAC）、优化的单指令多数据指令（SIMD）等。因此Cortex-M4执行所有的DSP指令集都可以在单周期内完成，而Cortex-M3和M0需要多个指令和多个周期才能完成同样的功能。比如开方运算，M3和M0只能通过迭代法（标准数学函数库）计算，而M4F直接调用VSQRT指令完成。

（1）获取DSP库

ST官方提供了一整套的DSP库方便我们开发使用。在ST提供的标准库：stm32f4_dsp_stdperiph_lib.zip里面就有（该文件可以从ST官网上下载：http://www.st.com/web/en/catalog/tools/FM147/CL1794/SC961/SS1743/PF257901下载，文件名：STSW-STM32065）。下载解压缩之后，在目录STM32F4xx_DSP_StdPeriph_Lib_V1.4.0→Libraries→CMSIS→DSP_Lib下可以找到DSP库文件和测试实例。Sourse中是所有DSP库文件源代码，Examples文件夹下是一些测试实例。

（2）DSP库简介

DSP库主要包含以下几个分库：

BasicMathFunctions

基本数学函数：提供浮点数的各种基本运算函数，如向量加减乘除等运算。

CommonTables

arm_common_tables.c文件提供位翻转或相关参数表。

ComplexMathFunctions

复杂数学功能，如向量处理，求模运算的。

ControllerFunctions

控制功能函数。包括正弦余弦，PID电机控制，矢量Clarke变换，矢量Clarke逆变换等。

FastMathFunctions

快速数学功能函数。提供了一种快速的近似正弦，余弦和平方根等相比CMSIS计算库要快的数学函数。

FilteringFunctions

滤波函数功能，主要为FIR和LMS（最小均方根）等滤波函数。MatrixFunctions

矩阵处理函数。包括矩阵加法、矩阵初始化、矩阵反、矩阵乘法、矩阵规模、矩阵减法、矩阵转置等函数。

StatisticsFunctions

统计功能函数。如求平均值、最大值、最小值、计算均方根RMS、计算方差/标准差等。

SupportFunctions

支持功能函数，如数据拷贝，Q格式和浮点格式相互转换，Q任意格式相互转换。

TransformFunctions

变换功能。包括复数FFT（CFFT）/复数FFT逆运算（CIFFT）、实数FFT（RFFT）/实数FFT逆运算（RIFFT）、和DCT（离散余弦变换）和配套的初始化函数。

ST提供了.lib格式的文件，方便使用这些库。这些.lib文件就是由Source文件夹下的源码编译生成的，如果想看某个函数的源码，可以在Source文件夹下面查找。.lib格式文件路径：STM32F4xx_DSP_StdPeriph_Lib_V1.4.0→Libraries→CMSIS→Lib→ARM，总共有8个.lib文件，和M4F相关的有两个：

arm_cortexM4bf_math.lib(浮点Cortex-M4大端模式)

arm_cortexM4lf_math.lib(浮点Cortex-M4小端模式)

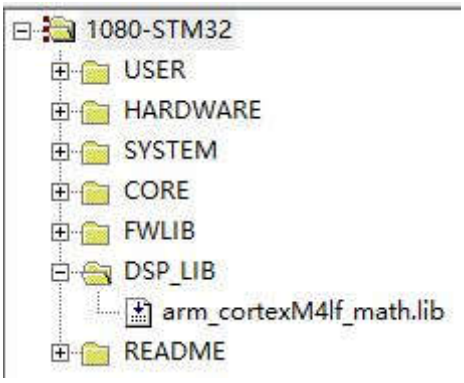
STM32F4的内核CortexM4F采用小端模式，所以选择：arm_cortexM4lf_math.lib(浮点Cortex-M4小端模式)。

（3）DSP库编程环境搭建

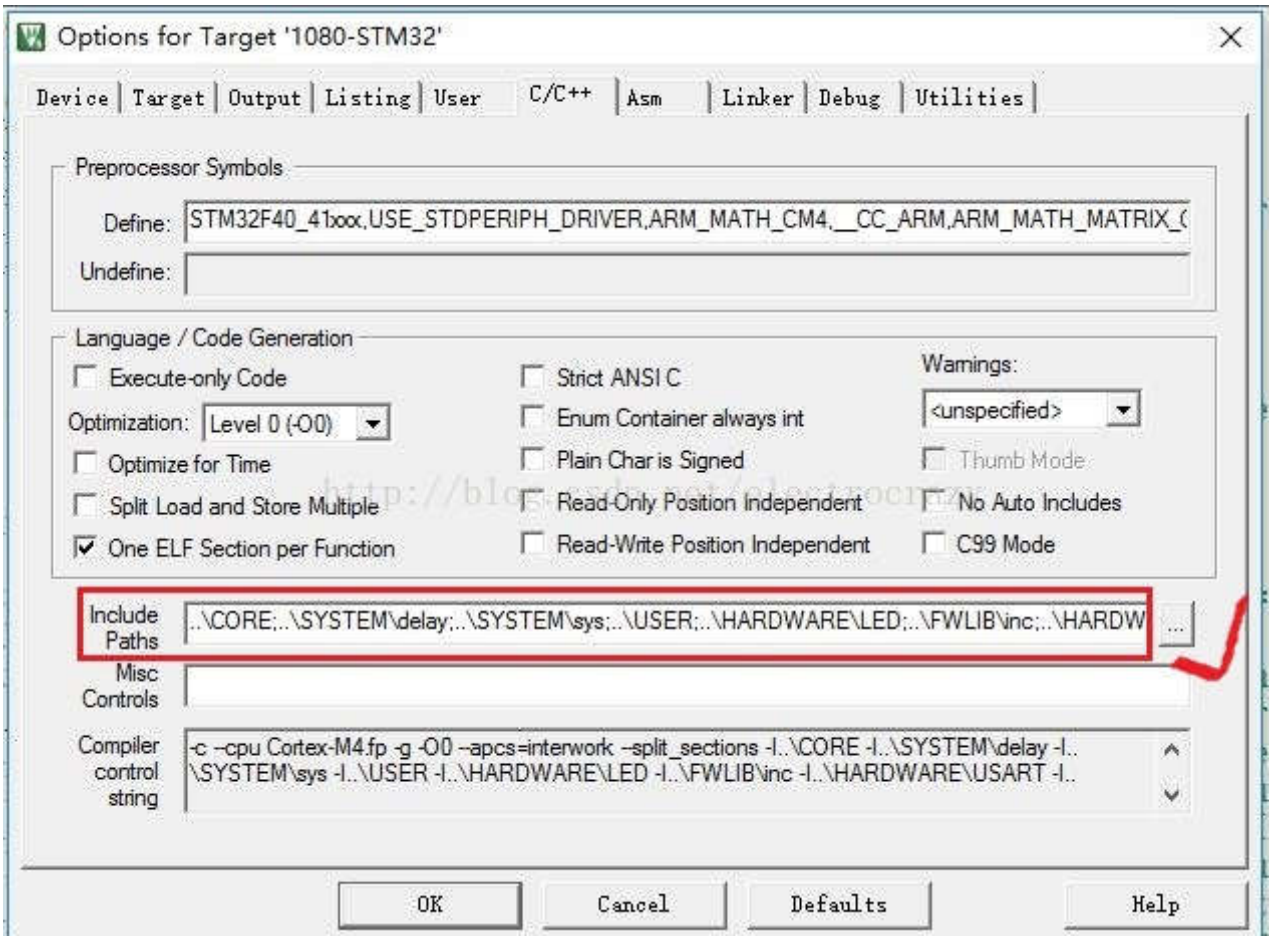
在设置使用DSP库之前首先要先开启硬件FPU，然后按照如下步骤搭建DSP库运行环境。

首先添加库文件。在工程目录下新建DSP_LIB文件夹用于存放库文件。然后把arm_cortexM4lf_math.lib和相关头文件（路径STM32F4xx_DSP_StdPeriph_Lib_V1.4.0à LibrariesàCMSISà Include 里的文件）拷贝到DSP_LIB文件夹中。

然后打开工程，新建DSP_LIB分组，并将arm_cortexM4lf_math.lib添加到工程里面。



添加好文件之后，需要添加头文件包含路径，将第一步拷贝的Include文件夹和DSP_LIB文件夹，加入头文件包含路径。打开工程属性设置面板，然后点击”C/C++“选项卡，点击对号处，弹出include path设置面板。添加”..\DSP_LIB“和”..\DSP_LIB\Include“两个路径。



最后，为了能够使用DSP库的所有功能，还需要添加以下几个全局宏定义：

1、**__FPU_USED**

2、**__FPU_PRESENT**

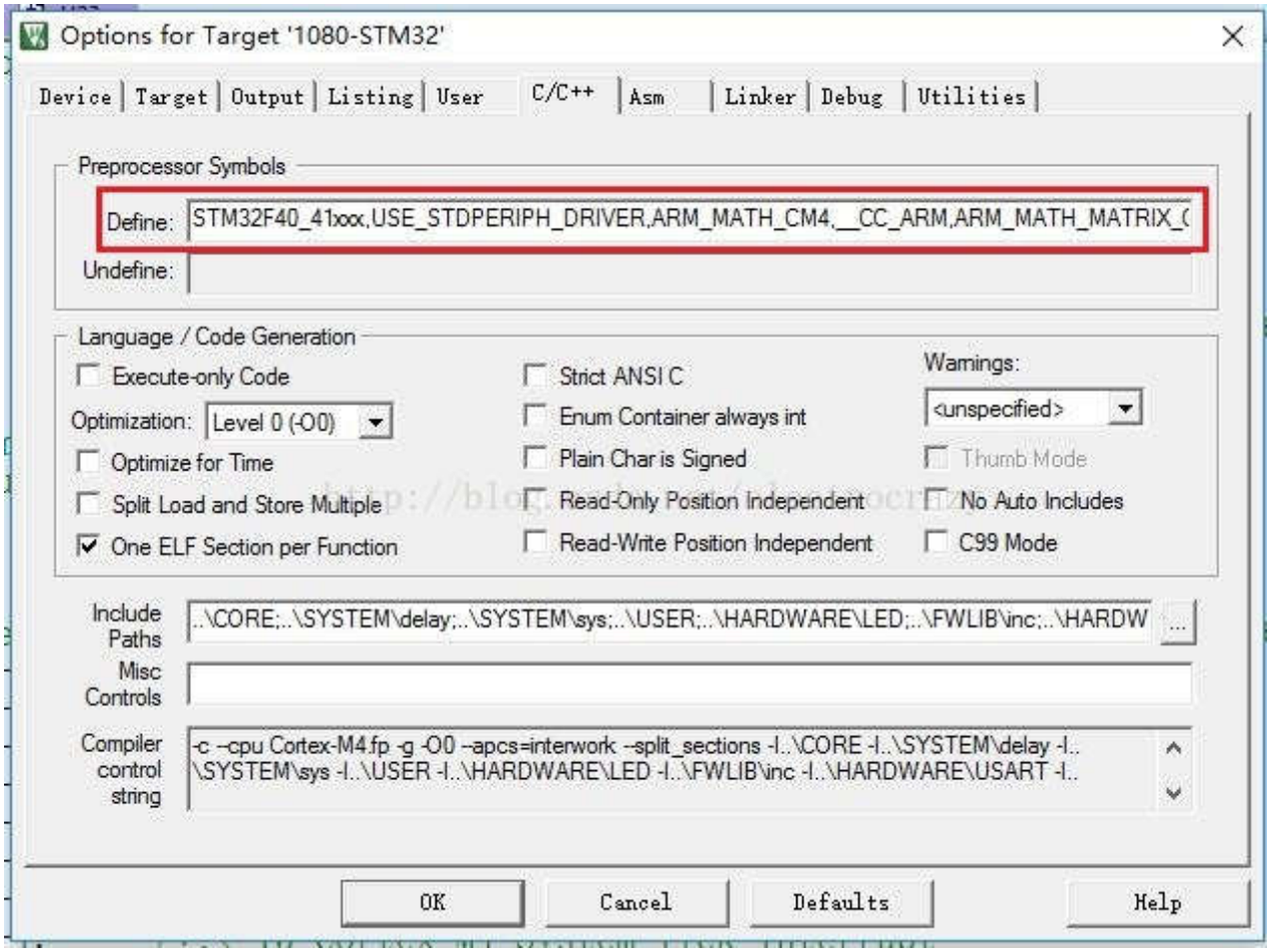
3、**ARM_MATH_CM4**

4、**__CC_ARM**

5、**ARM_MATH_MATRIX_CHECK**

6、**ARM_MATH_ROUNDING**

添加方法是打开工程属性设置面板，然后点击”C/C++“选项卡，在“Preprocessor Symbols“下的”Define：”文本框中进行添加。两个宏之间用“，”隔开。如果已经开启了硬件FPU，则无需添加__FPU_USED和__FPU_PRESENT这两个宏。



至此，STM32F4的DSP库运行环境已经搭建好了。可以使用DSP库的相关函数了。

参考：正点原子——ALIENTEK探索者STM32F407开发板相关资料