* 每个版本的 Cube 软件包都会携带 CMSIS 文件夹，只是版本比较老，不推荐
* 不使用MDK的math.h库，只使用arm\_math.h的库，即ARM的数学库，从F0到F4等的HAL包里的CMSIS都有相应的数学库，只不过针对有没有DSP的MCU的区别了，记着要加的六个宏定义：ARM\_MATH\_CM4可以改成M0~M4

USE\_HAL\_DRIVER,STM32F407xx,

ARM\_MATH\_CM4,\_\_CC\_ARM,ARM\_MATH\_MATRIX\_CHECK,ARM\_MATH\_ROUNDING

ARM\_MATH\_LOOPUNROLL

PS,在CMSIS\DSP下的编译生成.lib的MDK工程中，看见已经定义了宏：

（是不是工程就不用添加了）

ARM\_MATH\_MATRIX\_CHECK, ARM\_MATH\_ROUNDING, ARM\_MATH\_LOOPUNROLL

* 正确使用DSP库才能发挥100倍的效果：

1. 合理选用编译器的优化等级，提高代码的执行效率。选择优化等级-O3
2. 在计算表达式中，强制常量为单精度浮点数，以避免引入双精度浮点数动算。即常数浮点数后面加f，运算式使用单精度浮点的（float），不要使用双精度浮点的（double），math.h库里的sin等函数都是双精度浮点的，所以不要用
3. 选用 ARM 公司提供的，采用单精度浮点数优化的三角函数库，以避免由普通的数学函数库引入的双精度浮点数运算；
4. 在 STM32F4 中，将数据放在 CCM 存贮器中，要比放在 SRAM1 中更能保证 CPU 对数据的存取速率，因为 CCM 存贮器是 CPU独享的，而 SRAM1 还可能被 DMA 访问。
5. 在 CPU 的指令实现中，乘法和除法的计算速度上是有很大差别的，通常乘法的计算速度远高于除法的计算速度。所以，有必要在运算中，使用乘法来代替除法。比如： a÷b÷c可以使用a÷(b×c)来代替，或者都用乘法，变成乘以倒数。
6. 找出算法中的重复计算，将其合并，只计算一次。对这样的计算，完全可以在第一次计算之后，将结果放在中间变量中，而在后面的计算中直接引用。
7. 对于复杂的计算，考虑是否能用查表来代替。查表是一种快速得出结果的好方法，它以牺牲存贮器空间来换取速度。在存贮器空间不是很紧张的情况下，用查表代替计算还是很划算的。
8. 常规乘法加法或单精度浮点的算式尽量直接写；特殊计算如三角函数、开方或者矩阵运算、FFT等就要用DSP函数了！

比如： 计算两向量点乘

for(ii = 0;ii < MAX\_BLOCKSIZE;ii++)

result += srcA\_buf\_f32[ii] \* srcB\_buf\_f32[ii]; //直接写

比：

arm\_mult\_f32(srcA\_buf\_f32, srcB\_buf\_f32, multOutput, MAX\_BLOCKSIZE);

for(ii=0; ii< MAX\_BLOCKSIZE; ii++) //用DSP库实现

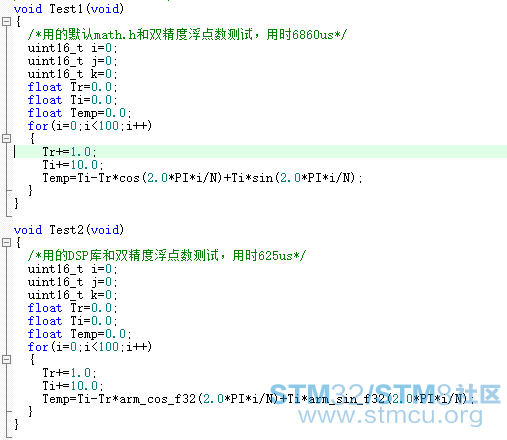
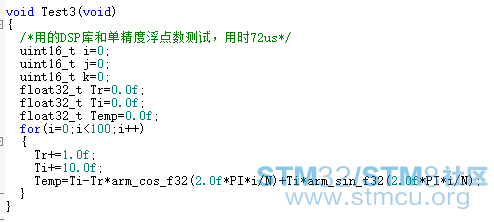
arm\_add\_f32(&testOutput, &multOutput[ii], &testOutput, 1);

更快！结果一样，速度更慢，所以：

向量点乘用专用ARM DSP函数：arm\_dot\_prod\_f32

尽量找ARM DSP提供的最高层的函数，不要用底层基础的ARM DSP函数组合，还不如直接写算式快！！

测试了arm\_sqrt\_f32、三角函数，这些都要比math.h库快10倍左右！

例子：

* 以\_f32结尾的函数是浮点运算，以\_q8, \_q15, \_q31,结尾的函数是定点运算：

各种Qn的数据类型的范围按照下面类型重定义来就行了…

typedef int8\_t q7\_t;

typedef int16\_t q15\_t;

typedef int32\_t q31\_t;

typedef int64\_t q63\_t;

typedef float float32\_t;

typedef double float64\_t;

~~首先定点数就是用整数来表示浮点数。比如Q15定点数，我们这里采用Q1.15格式，也就是1个符号位，剩下15位是数据位作为整数，他能表示的数据范围是-32768 到 32767,   对应到浮点数能表示的范围就是 -32768 / 32768 到 32767 / 32768 ，即 -1 到 0.9999695！！~~

在 CMSIS-DSP 常用的是 Q15，Q31等等，分别表示除以 2^15(32768)，除以 2^31(2147483648) 就可以得到实际的浮点数。

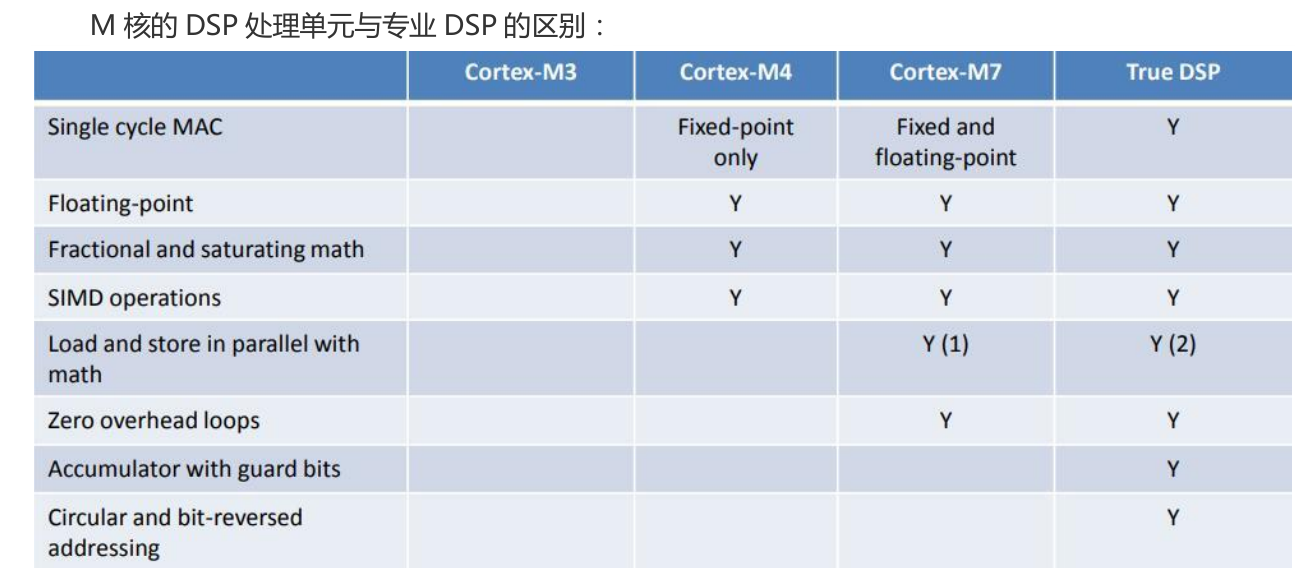
~~Q 表示 十进制数表示范围~~

~~Q31 -1≤x≤0.9999~~

~~Q15 －65,536≤x≤65,535.9999~~

~~Q7 -16,777,216≤x≤16,777,215.992~~

* STM32F4 采用 Cortex-M4 内核，相比 Cortex-M3 系列除了内置硬件 FPU 单元，在数字信号处理方面还增加了 DSP 指令集，支持诸如单周期乘加指令（MAC），优化的单指令多数据指令（SIMD），饱和算数等多种数字信号处理指令集。

Cortex-M4 执行所有的 DSP 指令集都可以在单周期内完成，而Cortex-M3 需要多个指令和多个周期才能完成同样的功能。

* 源码在HAL库的在CMSIS的DSP文件夹中（有丰富例程），方便添加到工程的.lib文件在CMSIS下的LIB文件夹中：

⑤ arm\_cortexM4b\_math.lib (Cortex-M4 大端模式)

⑥ arm\_cortexM4l\_math.lib (Cortex-M4 小端模式)

⑦ arm\_cortexM4bf\_math.lib (浮点 Cortex-M4 大端模式)

⑧ arm\_cortexM4lf\_math.lib (浮点 Cortex-M4 小端模式)

STM32F4 属于 CortexM4F 内核，小端模式，应选择：

arm\_cortexM4lf\_math.lib(浮点 Cortex-M4小端模式)。

* 源码文件介绍如下：

（函数的用法，常见的看安富莱的DSP的手册，全面的看CMSIS Docs）

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

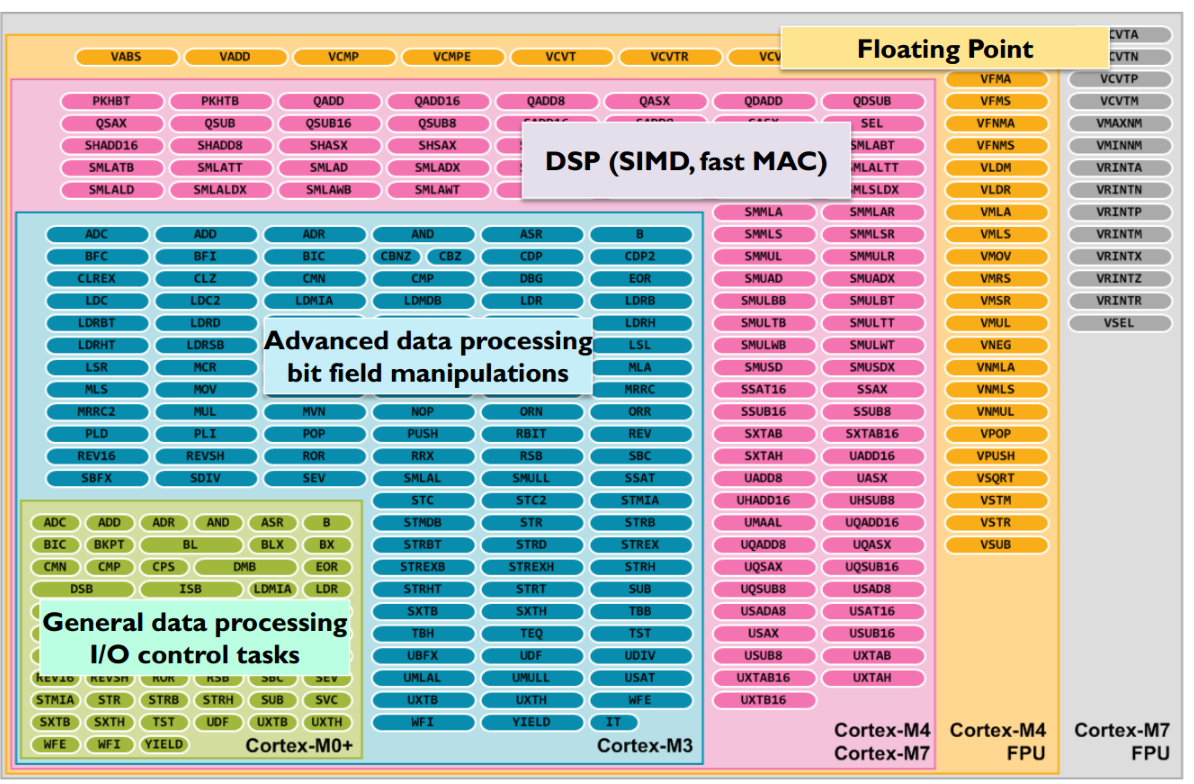
统计功能函数。如求平均值、最大值、最小值、计算均方根 RMS、计算方差/标准差等。

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。

* 指令集图

浮点指令都是以字符 V 开头的

* 最后，详细每个函数的使用用法看 CMSIS Docs 手册！