**AN0036**
应用笔记

DSP Instruction and Library on AT32

# 前言

这篇应用笔记主要介绍ARM Cortex-M4F的DSP指令、DSP相关库函数以及移植DSP库到AT32 MCU的方法，并在最后以具体示例介绍一些库函数在应用中的使用方法。

为了充分利用本应用笔记中的信息，用户应熟悉 AT32 微控制器系列。可以参考以下资料：

- 《Cortex--M4 权威指南》
- 《数字信号与处理》（作者：赵健、王宾等）清华大学
- 《信号与系统第二版》（作者：奥本海姆）
- CMSIS package 下载地址
  https://github.com/ARM-software/CMSIS_5/releases/tag/5.6.0
- CMSIS DSP 官方教程
  http://www.keil.com/pack/doc/CMSIS/DSP/html/index.html
- CMSIS NN 官网教程
  http://www.keil.com/pack/doc/CMSIS/NN/html/index.html
- 安富莱_STM32-V5 开发板_DSP 数字信号处理教程（**笔者强烈推荐**）
  http://www.armbbs.cn/forum.php?mod=viewthread&tid=94547&highlight=CMSIS-NN%C9%F1%BE%AD%CD%F8%C2%E7%BD%CC%B3%CC

*注：本应用笔记对应的代码是基于雅特力提供的V2.x.x 板级支持包（BSP）而开发，对于其他版本BSP，需要注意使用上的区别。*

支持型号列表：

| 支持型号 | AT32F4 系列 |
| --- | --- |

# 目录

# 表目录

# 图目录

# 1 概述

AT32F4xx 使用的是 ARM Cortex®-M4F 内核。ARM Cortex®-M4F 是带有 FPU 内核处理器是一款 32 位的 RISC 处理器，具有优异的代码效率，采用通常 8 位和 16 位器件的存储器空间即可发挥 ARM 内核的高性能。该处理器支持一组 DSP 指令，能够实现有效的信号处理和复杂的算法执行。其单精度 FPU(浮点单元)通过使用元语言开发工具，可加速开发，防止饱和。

本文重点介绍基于 AT32 MCU 的 DSP 指令相关库函数及其简单应用示例，主要内容有：

- ARM Cortex®-M4F 内核
- ARM 官方 CMSIS DSP 库概述
- CMSIS DSP 库移植到 AT32
- 常用示例展示
- CMSIS NN with DSP

*注意：　本文是基于 AT32F403A 的硬件条件，若使用者需要在 AT32 其他型号上使用，请修改相应配置即可。*

# 2 AT32 MCU 与 M4F 内核

AT32F403A系列与所有的ARM工具和软件兼容。

这些丰富的外设配置，使得AT32系列微控制器适合于多种应用场合：

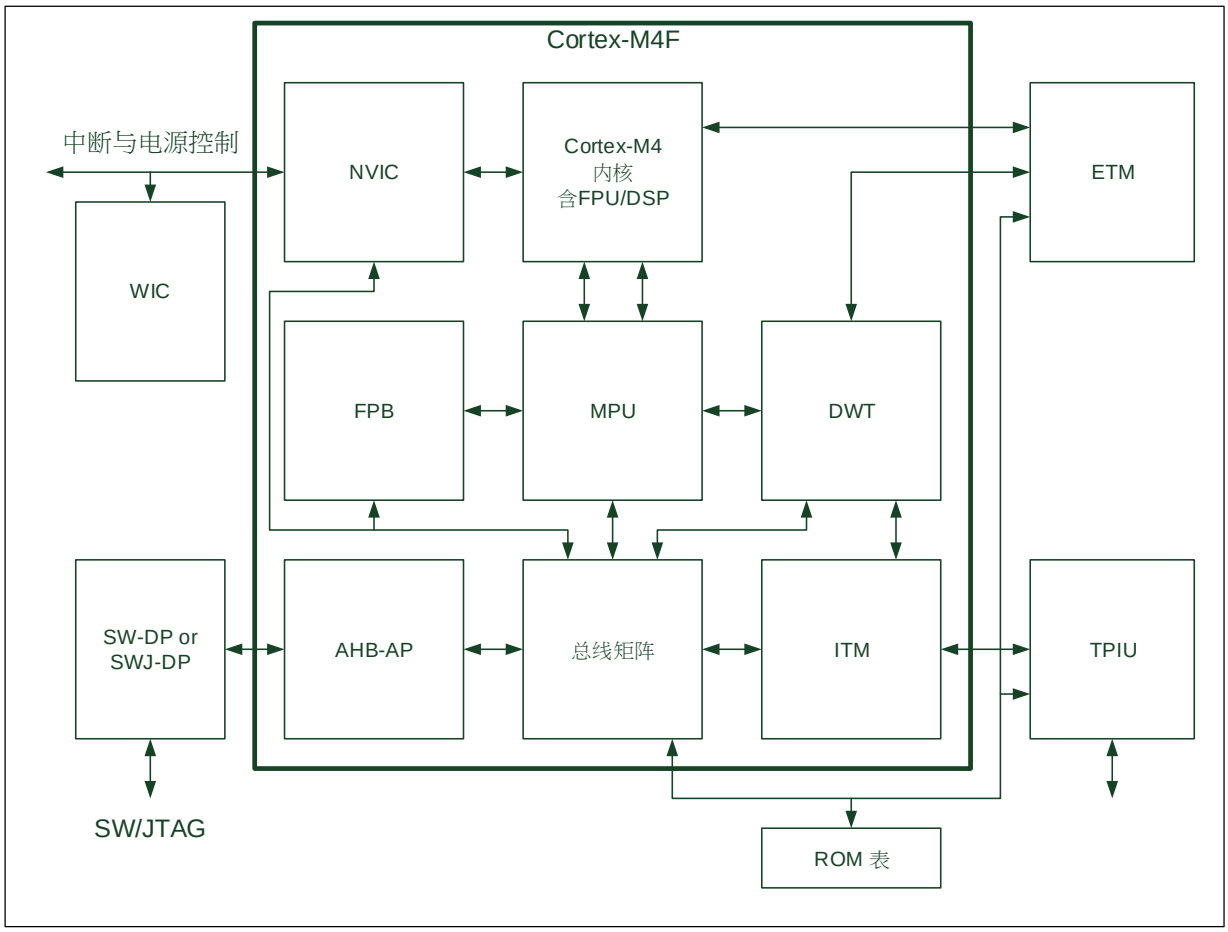
- 消费类产品
  - 手持云台
  - 微型打印机
  - 条形码扫描枪
  - 读卡器
  - 灯光控制
- 物联网应用
  - 智能家居应用
  - 物联网传感器节点
- 工业应用
  - 双CAN应用(OBD-II)
  - 光电编码器
  - 充电桩/BMS
  - 机器人控制
  - 电力控制
- 电机控制
  - BLDC/PMSM电机控制
  - 变频器
  - 伺服电机控制

## 2.1 系统架构

AT32F403A 系列微控制器包括 ARM® Cortex™-M4F 处理器内核、总线架构、外设以及存储器构成。Cortex™-M4F 处理器是一种新时代的内核，拥有许多先进功能。对比于 Cortex™-M3，Cortex™-M4F 处理器支持增强的高效 DSP 指令集，包含扩展的单周期 16/32 位乘法累加器 MAC、双 16 位 MAC 指令、优化的 8/16 位 SIMD 运算及饱和运算指令，并且具有单精度 IEEE-754 浮点运算单元 FPU。当设计中使用带 DSP 功能的 Cortex™-M4F 时就能格外节能，比软件解决方案更快，使 Cortex™-M4F 适用于那些要求微控制器提供高效能与低功耗的产品市场。

1) Cortex-M4 内核架构

**图 1. Cortex-M4 内核架构**




2) Cortex-M4 与 Cortex-M3 的区别

表 1. Cortex-M4 与 Cortex-M3 的区别

| ITEM | Cortex-M3 | Cortex-M4 |
|---|---|---|
| architecture Version | v7M | v7ME |
| Instruction set architecture | Thumb + Thumb-2 | Thumb + Thumb-2, DSP, SIMD, FP |
| DMIPS/MHz | 1.25 | 1.25 |
| Integrated NVIC | Yes | Yes |
| Number interrupts | 1-240 + NMI | 1-240 + NMI |
| Interrupt priorities | 8-256 | 8-256 |
| Single Cycle Multiply | Yes | Yes |
| Hardware Divide | Yes | Yes |
| Single cycle DSP/SIMD | No | Yes |
| Floating point hardware | No | Yes |
| Bus protocol | AHB Lite, APB | AHB Lite, APB |

3) 部分 DSP 指令的介绍

表 2. 单周期 MAC 指令介绍

| OPERATION | INSTRUCTIONS | CM3 | CM4 |
|---|---|---|---|
| 16X16=32 | SUBLBB，MULBT，SMULTB，SMULTT | n/a | 1 |
| 16X16+32=32 | SMLABB，SMLABT，SMLATB，SMLATT | n/a | 1 |
| 16X16+64=64 | SMLALBB，SMLALBT，SMLALTB，SMLALTT | n/a | 1 |
| 16X32=32 | SMULWB，SMULWT | n/a | 1 |
| 16X32+32=32 | SMLAWB，SMLAWT | n/a | 1 |
| (16X16)±(16X16)=32 | SMUAD，SMUADX，SMUSD，SMUSDX | n/a | 1 |
| (16X16)±(16X16)+32=32 | SMLAD，SMLADX，SMLSD，SMLSDX | n/a | 1 |
| (16X16)±(16X16)+64=64 | SMLALD，SMLALDX，SMLSLD，SMLSLDX | n/a | 1 |
| 32X32=32 | MUL | 1 | 1 |
| 32±32X32=32 | MLA，MLS | 2 | 1 |
| 32X32=64 | SMULL，UMULL | 5-7 | 1 |
| 32X32+64=64 | SMLAL，UMLAL | 5-7 | 1 |
| 32X32+32+32=64 | UMAAL | n/a | 1 |
| 32±32X32=32(upper) | SMMLA，SMMLAR，SMMLS，SMMLSR | n/a | 1 |
| 32X32=32(upper) | SMMUL，SMMULR | n/a | 1 |

*注意:* *上面所有的指令操作在 CM4 处理器上都只需一个指令周期。*

4) Cortex-M4 DSP 指令比较

表 3. Cortex-M4 DSP 指令比较

| CLAS | INSTRUCTIN | Cycle counts | |
|---|---|---|---|
| | | CORTEX-M3 | CORTEX-M4 |
| Arithmetic | ALU operation (not PC) | 1 | 1 |
| | ALU operation to PC | 3 | 3 |
| | CLZ | 1 | 1 |
| | QADD, QDADD, QSUB, QDSUB | n/a | 1 |
| | QADD8, QADD16, QSUB8, QSUB16 | n/a | 1 |

| CLAS | INSTRUCTIN | Cycle counts | |
|---|---|---|---|
| | | CORTEX-M3 | CORTEX-M4 |
| | QDADD, QDSUB | n/a | 1 |
| | QASX, QSAX, SASX, SSAX | n/a | 1 |
| | SHASX, SHSAX, UHASX, UHSAX | n/a | 1 |
| | SADD8, SADD16, SSUB8, SSUB16 | n/a | 1 |
| | SHADD8, SHADD16, SHSUB8, SHSUB16 | n/a | 1 |
| | UQADD8, UQADD16, UQSUB8, UQSUB16 | n/a | 1 |
| | UHADD8, UHADD16, UHSUB8, UHSUB16 | n/a | 1 |
| | UADD8, UADD16, USUB8, USUB16 | n/a | 1 |
| | UQASX, UQSAX, USAX, UASX | n/a | 1 |
| | UXTAB, UXTAB16, UXTAH | n/a | 1 |
| | USAD8, USADA8 | n/a | 1 |
| ultiplication | MUL, MLA | 1-2 | 1<br>Single cycle MAC |
| | MULS, MLAS | 1-2 | 1<br>Single cycle MAC |
| | SMULL, UMULL, SMLAL, UMLAL | 5-7 | 1<br>Single cycle MAC |
| | SMULBB, SMULBT, SMULTB, SMULTT | n/a | 1 |
| | SMLABB, SMLBT, SMLATB, SMLATT | n/a | 1 |
| | SMULWB, SMULWT, SMLAWB, SMLAWT | n/a | 1 |
| | SMLALBB, SMLALBT, SMLALTB, SMLALTT | n/a | 1 |
| | SMLAD, SMLADX, SMLALD, SMLALDX | n/a | 1 |
| | SMLSD, SMLSDX | n/a | 1 |
| | SMLSLD, SMLSLD | n/a | 1 |
| | SMMLA, SMMLAR, SMMLS, SMMLSR | n/a | 1 |
| | SMMUL, SMMULR | n/a | 1 |
| | SMUAD, SMUADX, SMUSD, SMUSDX | n/a | 1 |
| | UMAAL | n/a | 1 |
| Division | SDIV, UDIV | 2-12 | 2-12 |

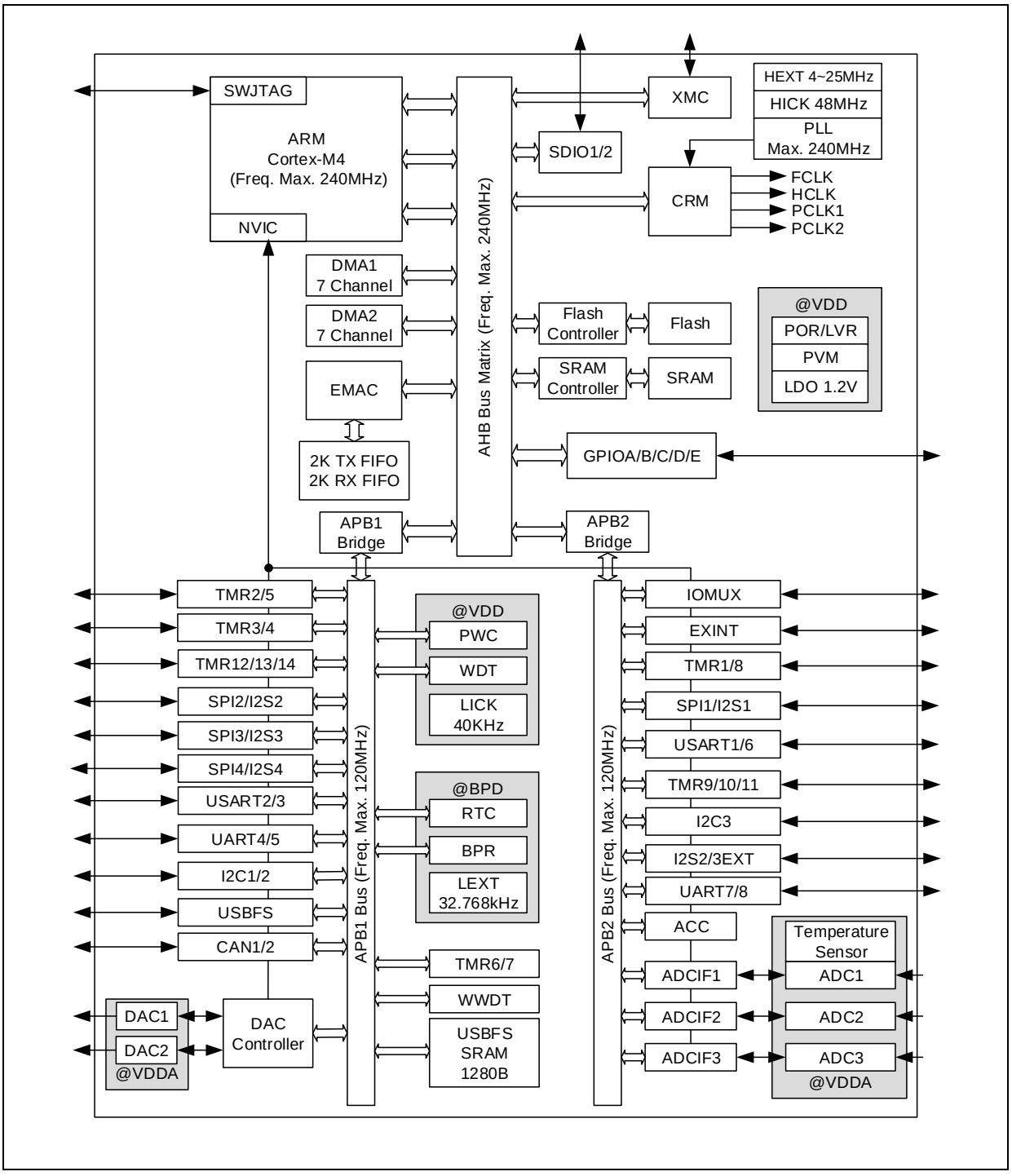
5) 编译器对 DSP 指令的支持

**表 4. 编译器对 DSP 指令的支持**

| Feature | IAR Compiler | Keil compiler | GCC from web 02/11 | ARM DSP Library |
|---|---|---|---|---|
| SIMD instruction | No native support *Confirmed by tests (IAR6.21)* | To check | Not supported Under analysis | API for each SIMD instruction *Used efficiently by IAR 6.21* |
| Saturated arithmetic instructions | Some pattern detection | To check | Not supported Under analysis | |
| Floating point HW | Native support | Native support | Supported | Used in CMSIS DSP Library |
| Floating point software library | IEEE754 To check the subnormal support | Fully IEEE754 compliant | To check | No implementation for the IEEE754 not supported by the FPU |

6) AT32F403A 系统架构

图 2. AT32F403A 系统架构图




*注意：AT32F403A不支持EMAC，AT32F407/407A支持EMAC*

# 3 ARM 官方 CMSIS DSP 库概述

## 3.1 CMSIS DSP 库说明

CMSIS DSP 软件库,是针对使用 Cortex-M 内核芯片提供一套数字信号处理函数。CMSIS DSP 库大部分函数都是支持 f32，Q31，Q15 和 Q7 四种格式的。该库分为以下几个功能：

- 基本数学函数 Basic math functions
- 快速数学函数 Fast math functions
- 复数型数学函数 Complex math functions
- 滤波器函数 Filters
- 矩阵型函数 Matrix functions
- 数学变换型函数 Transform functions
- 电机控制函数 Motor control functions
- 统计型数学函数 Statistical functions
- 支持型数学 Support functions
- 插补型数学函数 Interpolation functions

针对以上每一种类型的库函数，下文会有详细介绍其使用方法和使用示例。

## 3.2 CMSIS DSP 库文件

考虑到方便用户使用，ARM 官方已编译好 Cortex-M 各型号的.lib 库文件，并放置于 Lib 文件夹。与 AT32F4xx 相关的.Lib 库文件主要有以下两种

- arm_cortexM4lf_math.lib (Cortex-M4, Little endian, Floating Point Unit) for AT32F403 and AT32F413
- arm_cortexM4l_math.lib (Cortex-M4, Little endian) for AT32F415

DSP 库函数的声明位域头文件 arm_math.h 中，用户只要简单地将该头文件和.lib 文件添加到自己的工程中，即可呼叫 DSP 库函数。该头文件对于浮点运算单元（FPU）的变量同样适用。

## 3.3 CMSIS DSP 库示例

该 CMSIS DSP 库中的多个示例可以很好地展现 DSP 库函数的使用。

## 3.4 CMSIS DSP 库的工具链支持

该DSP库已经可以在5.14版本MDK上开发和测试过。另外针对GCC编译器和IAR IDE，已经支持。

## 3.5 编译生成 DSP 的.lib 库文件

该DSP安装包中已包含一个基于MDK的工程，通过编译该工程可生成需要的.lib库文件。该MDK工程位于CMSIS\DSP\Projects\ARM文件夹中。工程名为

- arm_cortexM_math.uvprojx

通过打开并编译该arm_cortexM_math.uvprojx MDK工程，可以生成该DSP的.lib库文件。这样用户就可以根据特定的内核，特定的优化选择去编译特定DSP的.lib库文件。同时，通过该MDK工程，用户也可以查看与修改指定的库函数原型，便于了解库函数的实现原理。

## 3.6 CMSIS-DSP 文件夹结构

以下表格展现了CMSIS-DSP 文件夹结构

**表 5. CMSIS-DSP 文件夹结构**

| File/Folder | Content |
|---|---|
| CMSIS\Documentation\DSP | CMSIS DSP 文档 |
| CMSIS\DSP\DSP_Lib_TestSuite | DSP_Lib 测试套件 |
| CMSIS\DSP\Examples | 展现 DSP 库函数的示例 |
| CMSIS\DSP\Include | DSP_Lib 头文件 |
| CMSIS\DSP\Lib | DSP_Lib .lib 库文件 |
| CMSIS\DSP\Projects | 用于再编译.lib 的工程 |
| CMSIS\DSP\Source | DSP_Lib 源文件 |

# 4 CMSIS DSP 库移植到 AT32

本文主要介绍DSP库在MDK上的移植方法。

## 4.1 ARM 官方 CMSIS DSP 函数详解

- 基本数学函数Basic math functions
- 快速数学函数Fast math functions
- 复数型数学函数Complex math functions
- 滤波器函数Filters
- 矩阵型函数Matrix functions
- 数学变换型函数Transform functions
- 电机控制函数Motor control functions
- 统计型数学函数Statistical functions
- 支持型数学Support functions
- 插补型数学函数Interpolation functions

详细使用方法和使用案例请参考

1) ARM官网DSP培训资料

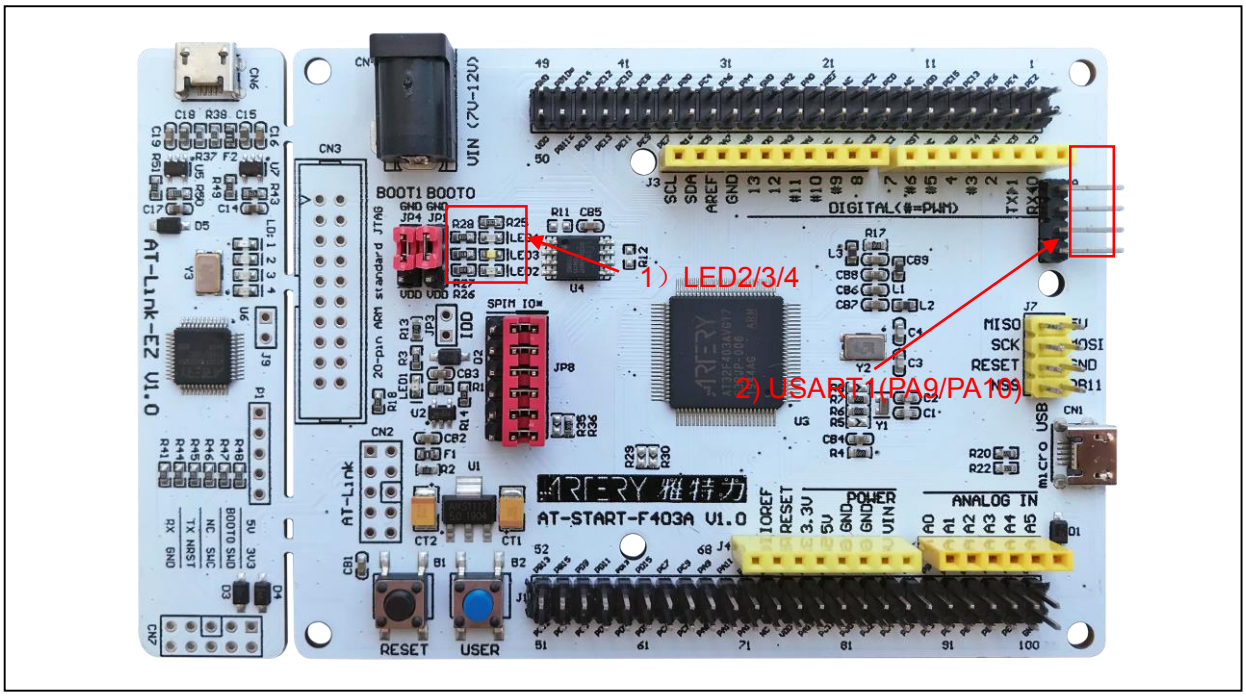
地址：http://www.keil.com/pack/doc/CMSIS_Dev/DSP/html/index.html

## 4.2 AT32 DSP 库快速使用

### 4.2.1 硬件资源

1) 指示灯LED2/LED3/LED4
2) USART1(PA9/PA10)
3) AT-START-F403A V1.0 实验板

图 3. AT-START-F403A V1.0 实验板




*注: 该DSP demo 是基于AT32F403A 的硬件条件，若使用者需要在AT32 其他型号上使用，请修改相应配置即可。*

## 4.2.2 软件资源

1) Libraries
   - drivers AT32底层驱动库
   - cmsis CMSIS DSP库和CMSIS NN库
2) Project\AT_START_F403A
   - examples ，本文使用到的示例，如5_1_arm_class_marks_example，“5_1”表示章节，“arm_class_marks_example”表示示例名称
   - templates，基于.lib建立的DSP template工程
3) Doc
   a) AN0036_DSP_Instruction_and_Library_on_AT32_ZH_V2.x.x.pdf

## 4.2.3 DSP demo 使用

1) 打开AT32_DSP_DEMO_2.x.x\project\at_start_xxx\templates，编译后下载到实验板
2) 观察LED2/LED3/LED4，若依次翻转则表明程序有正确执行DSP函数。

# 5 常用示例展示

本节主要通过使用前面介绍的 DSP 库函数进行案列展示，展示的示例如下：

- 班级成绩统计示例
- 卷积示例
- 点积示例
- 频率仓示例
- 低通滤波示例
- 图形音频均衡器示例
- 线性插值示例
- 矩阵示例
- 信号收敛示例
- 正弦余弦示例
- 方差示例
- 卷积网络神经示例

## 5.1 班级成绩统计示例

描述:

演示使用最大，最小，均值，标准差，方差和矩阵函数来统计一个班级的成绩。

注意：

此示例还演示了静态初始化的用法。

变量说明：

- testMarks_f32 ： 指向 20 名学生在 4 门学科中获得的分数
- max_marks ： 最高分成绩
- min_marks ： 最低分成绩
- mean ： 所有成绩的平均分
- var ： 所有成绩的方差
- std ： 标准差
- numStudents ： 学生总数

使用到 DSP 软件库的函数有：

- arm_mat_init_f32()
- arm_mat_mult_f32()
- arm_max_f32()
- arm_min_f32()
- arm_mean_f32()
- arm_std_f32()
- arm_var_f32()

参考

AT32_DSP_DEMO\project\at_start_f403a\examples\5_1_arm_class_marks_example

## 5.2 卷积示例

描述:

本示例主要展示基于复数 FFT、复数乘法与支持函数的卷积理论。

算法:

卷积理论指出，时域中的卷积对应频域中的乘法。因此，两个信号的卷积后的傅里叶变换等于他们各自的傅里叶变换的乘积。使用快速傅里叶变换（FFT）可以有效的评估信号的傅里叶变换。

两个输入信号 a[n]和 b[n]填充为零，n1 和 n2 分别对应其信号长度。因此他们的长度将变为 N，N 大于或等于 n1+n2-1。由于采用基 4 变换，因此基数为 4。a[n]和 b[n]的卷积是通过对输入信号进行 FFT 变换，对联更新好进行傅里叶变换。并对相乘后的结果进行逆 FFT 变换来获得的。

由以下公式表示：

A[k] = FFT(a[n],N)

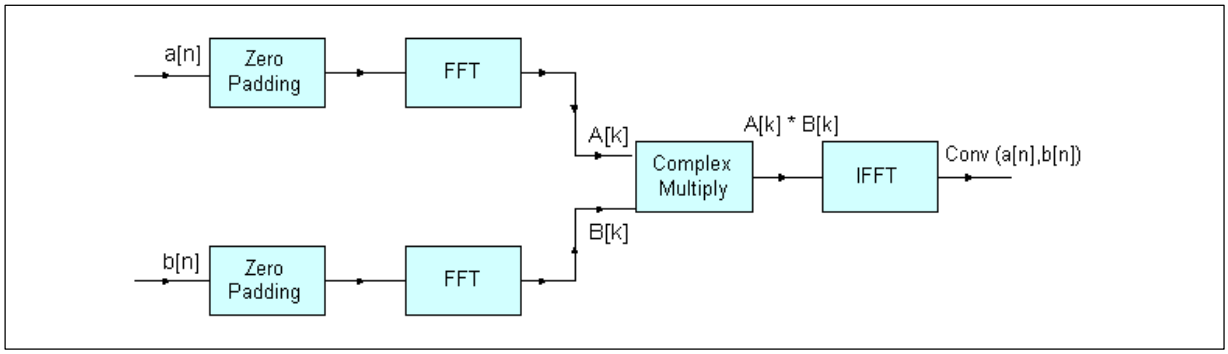
B[k] = FFT(b[n],N)

conv(a[n], b[n]) = IFFT(A[k] * B[k], N)

其中 A[k]和 B[k]分别是信号 a[n]和 b[n]的 N 点 FFT。卷积长度为 n1+n2-1

框图：

**图 4. 卷积算法框图**




变量说明

- testInputA_f32：指向第一个输入序列
- srcALen： 第一个输入时序的长度
- testInputB_f32：指向第二个输入序列
- srcBLen：第二个输入时序的长度
- outLen： 卷积输出序列的长度, (srcALen + srcBLen - 1)
- AxB： 指向 FFT 乘积后输出数组地址

使用到 DSP 软件库的函数有：

- arm_fill_f32()
- arm_copy_f32()
- arm_cfft_radix4_init_f32()
- arm_cfft_radix4_f32()
- arm_cmplx_mult_cmplx_f32()

参考

AT32_DSP_DEMO\project\at_start_f403a\examples\5_2_arm_convolution_example

## 5.3 点积示例

描述:

本示例主要展示如何使用相乘和相加来实现点积。两个向量的点积是通过将对应元素相乘并相加来获得的。

算法:

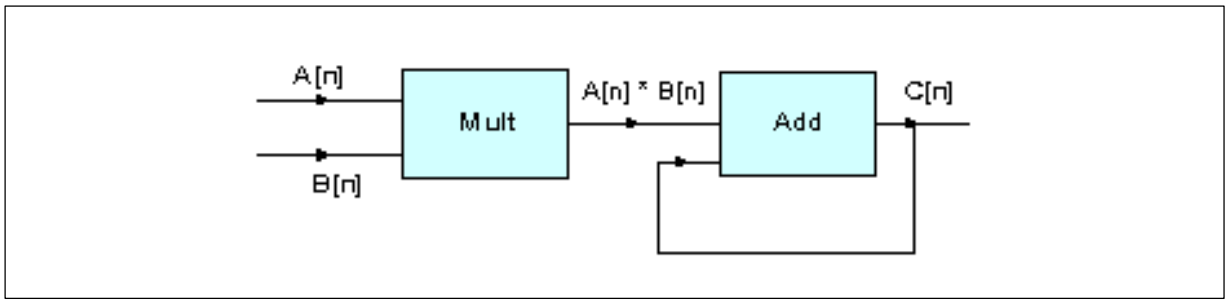
将长度为 n 的两个输入向量 A 和 B 诸个元素相乘，然后相加以获得点积。

由以下公式表示：

dotProduct = A[0] * B[0] + A[1] * B[1] + ... + A[n-1] * B[n-1]

框图:

**图 5. 点积算法框图**




变量描述:

- srcA_buf_f32：指向第一个输入向量
- srcB_buf_f32：指向第二个输入向量
- testOutput：存储两个向量的点积

使用到 DSP 软件库的函数有：

- arm_mult_f32()
- arm_add_f32()

参考

AT32_DSP_DEMO\project\at_start_f403a\examples\5_3_arm_dotproduct_example

## 5.4 频率仓示例

描述：

该示例主要展示使用复数 FFT，复数幅值和最大值函数在输入信号的频域中计算最大能量仓。

算法:

输入测试信号为一个 10 kHz 信号，该信号具有均匀分布的白噪声。通过计算输入信号的 FFT 计算可以得到与 10 kHz 输入频率相对应的最大能量仓。

框图:

**图 6. 频率仓算法框图**

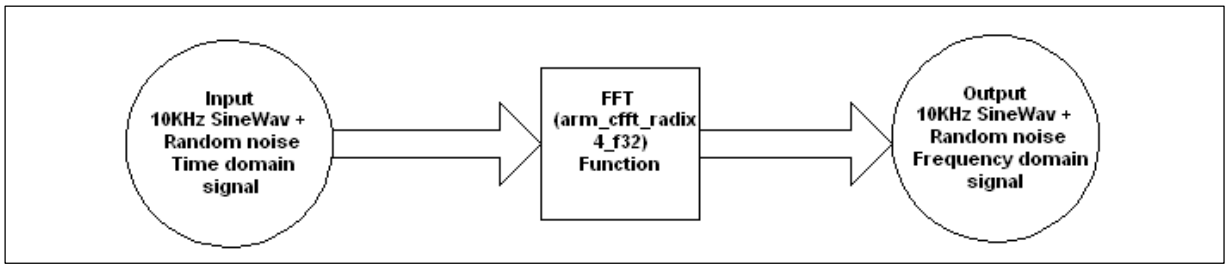



图 8 展示了具有均匀分布白噪声的 10 kHz 信号的时域信号， 图 9 展示了这个输入信号的对应的频域信号，其中出现最高点的数对应的频率即为 10 kHz 信号能量仓。

图 7. 输入信号的时域

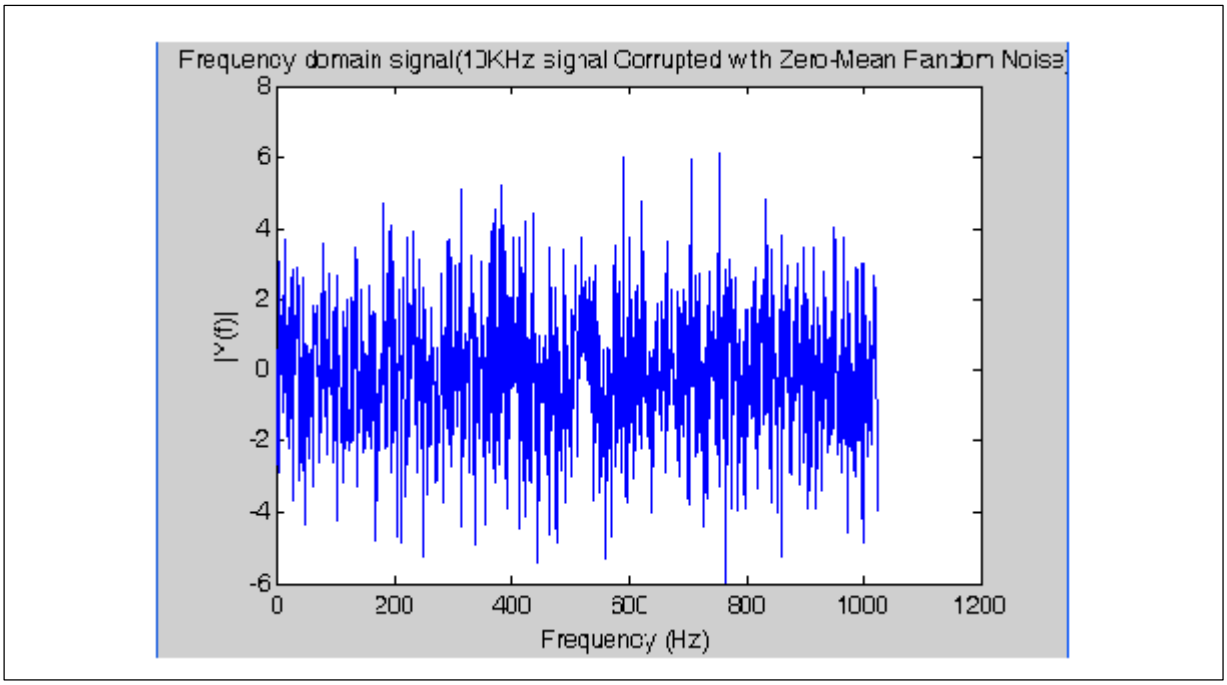
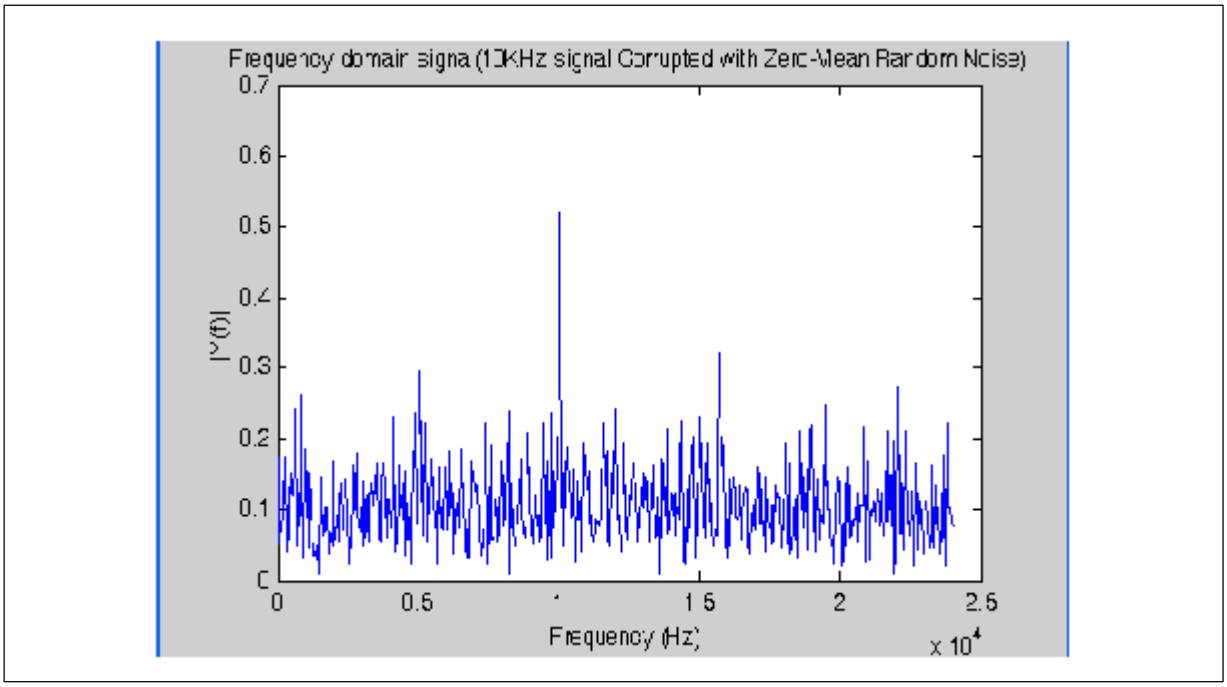



图 8. 输入信号的频域




输入信号的频域

变量描述

- testInput_f32_10khz ： 指向输入数据
- testOutput ： 指向输出数据
- fftSize l：FFT 的长度
- ifftFlag flag ： 用于选择 CFFT/CIFFT
- doBitReverse Flag ： 用于选择是顺序还是逆序

- refIndex ：参考索引值，在该值处能量最大
- testIndex ：计算出的索引值，在该值处能量最大

使用到 DSP 软件库的函数有：

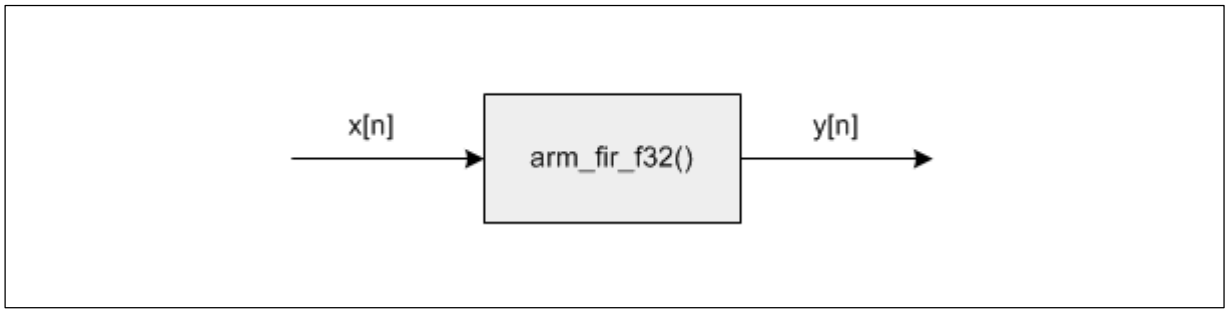
- arm_cfft_f32()
- arm_cmplx_mag_f32()
- arm_max_f32()

参考
AT32_DSP_DEMO\project\at_start_f403a\examples\5_4_arm_fft_bin_example

## 5.5 FIR 低通滤波示例

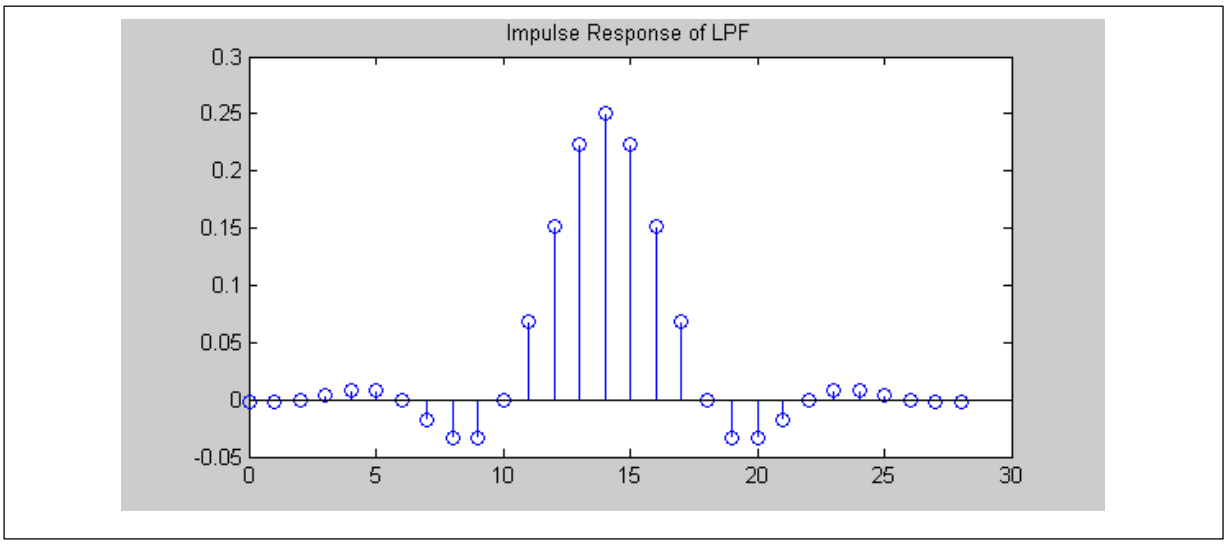
描述:
使用 FIR 低通滤波器从输入中去除高频信号部分。本示例展示了如何配置 FIR 滤波，然后以块方式传递数据。

**图 9. FIR 低通滤波算法框图**



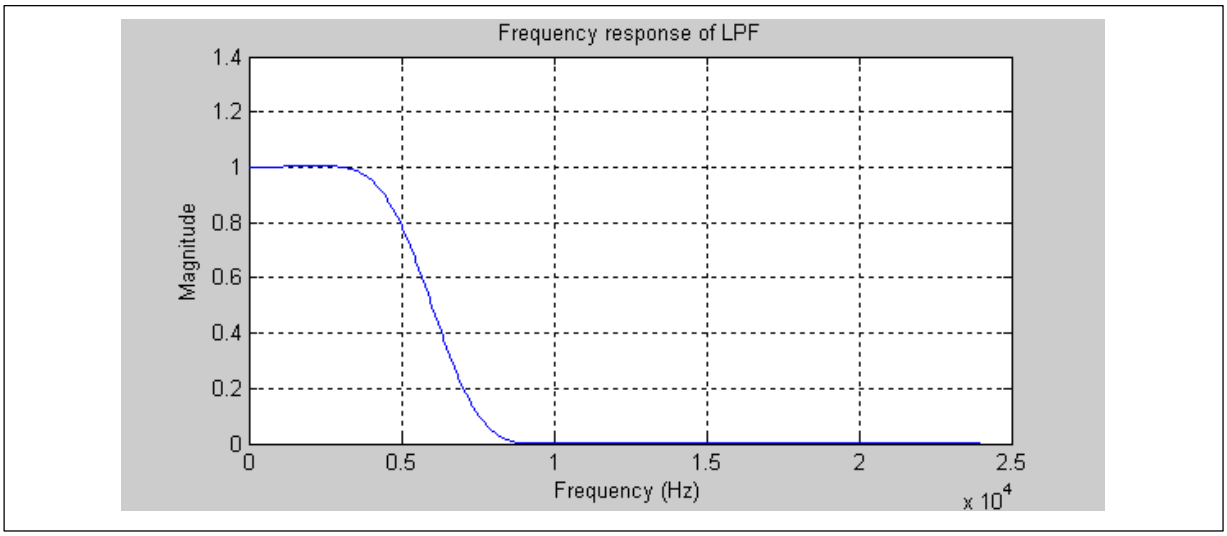

算法:
输入信号时两个正弦波的叠加：1 kHz and 15 kHz.该信号将被截止频率为 6 kHz 的进行低通滤波。低通滤波器滤掉了 15 kHz 信号，仅留下 1 kHz 信号输出。
低通滤波器采用 MATLAB 设计，采样率为 48 kHz，长度为 29 点。生成滤波器的 MATLAB 代码如下：
h = fir1(28, 6/24);
第一个参数是过滤器的“顺序”，并且总是比所需长度小 1,。第二个参数是归一化截止频率。范围是 0（DC）到 1.0（Nyquist）。24 kHz 奈奎斯特频率的 6kHz 截止频率为 6/24=0.25 归一化频率。
CMSIS FIR 滤波器函数要求系数按时间倒序排列。
所得滤波器系数如下图所示。需要注意的是，该滤波器是对称的（线性相位 FIR 滤波器的属性）。对称点是样本 14，对于所有频率，该滤波器具有 14 个样本的延迟。

图 10. 低通滤波时域响应



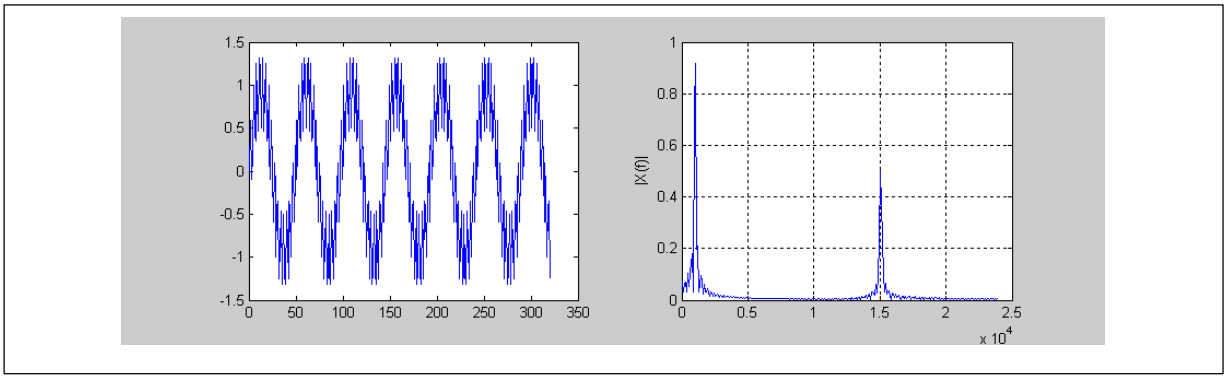

接下来显示滤波器的响应。滤波器的带通增益为 1.0，截止频率为 6kHz 时达到 0.5。

图 11. 低通滤波频域响应



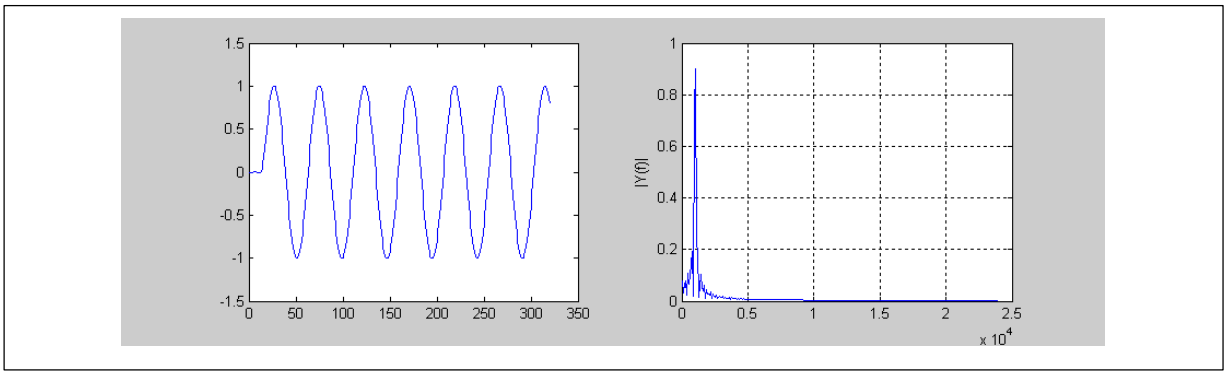

输入信号如下所示。左侧显示时域信号，右侧显示频域。可以清楚的看到两个正弦波分量。

图 12. 输入信号的时域信号和频域信号




滤波器输出如下所示，15kHz 分量已被消除。

图 13. 输出信号的时域信号和频域信号




变量描述：

- testInput_f32_1kHz_15kHz ：指向输入数据
- refOutput points to the reference output data：指向参考输出数据
- testOutput points to the test output data：指向测试输出数据
- firStateF32 points to state buffer：指向状态缓冲区
- firCoeffs32 points to coefficient buffer：指向系数缓冲区
- blockSize number of samples processed at a time：一次处理的样本数
- numBlocks number of frames：帧数

使用到 DSP 软件库的函数有：

- arm_fir_init_f32()
- arm_fir_f32()

参考

AT32_DSP_DEMO\project\at_start_f403a\examples\5_5_arm_fir_example

## 5.6 图形音频均衡器示例

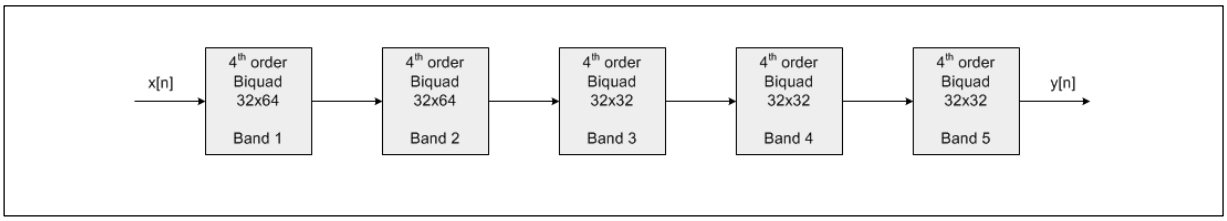
描述:：

本示例展示了如何使用 Biquad 级联函数构造 5 频段图形均衡器。在音频应用中使用图形均衡器来改变音频的音质。

框图：

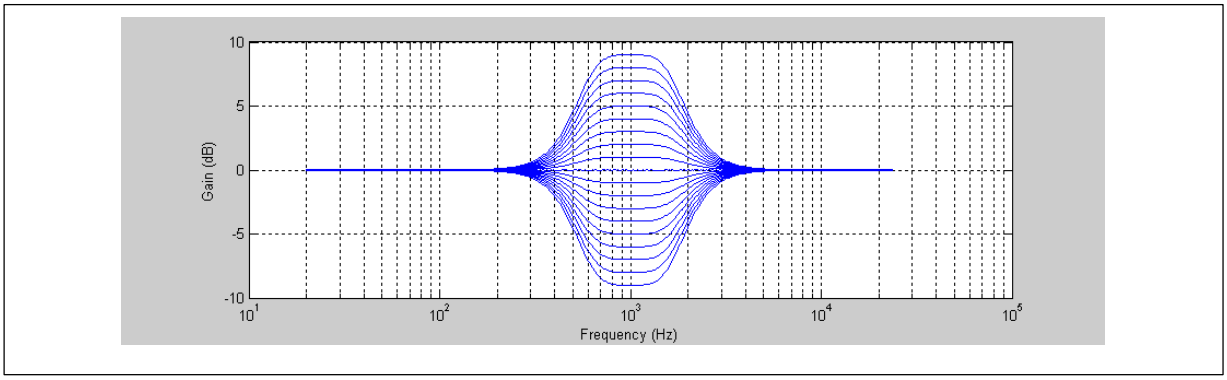
该设计是基于五级滤波器的级联

**图 14. 五级滤波器联算法框图**



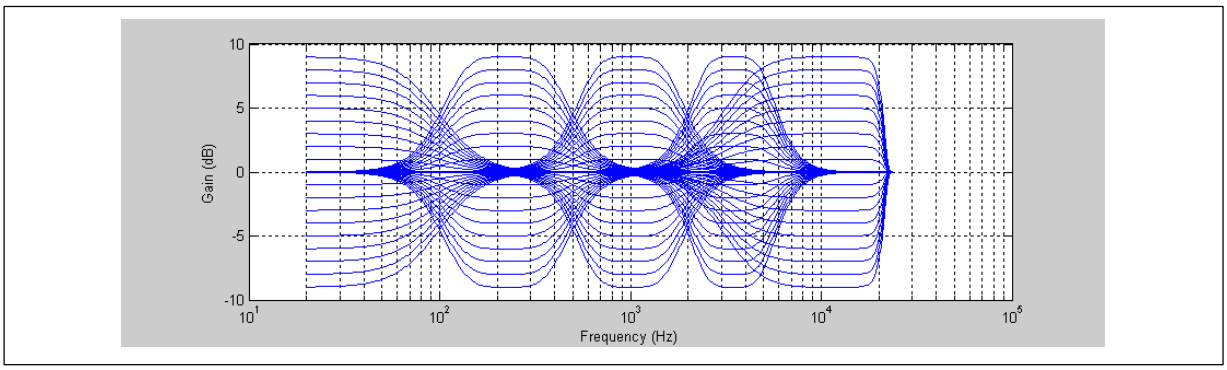

每个滤波器部分均为 40 阶，由两个 Biquad 级联组成。每个滤波器的标称为 0 dB（线性单位为 1.0）并对特定频率范围内的信号进行增强或截止。5 个频率段之间的边缘频率为 100、500、2000 和 6000 Hz。每个频段都有一个可调的增强或消减范围，范围为+/- 9 dB。列如，从 500 到 2000 Hz 的频宽具有如下所示响应：

**图 15. 从 200Hz 到 2KHz 的频宽响应**




以 1 dB 为步长，每个滤波器共有 19 种不同的设置。在 MATLAB 中预先计算了所有 19 中可能设置的频率器系数，并将其存储在表格中。使用 5 个不同表格，总共有 5 x 19 =95 个不同的 4 阶过滤器。所有 95 个响应如下所示：

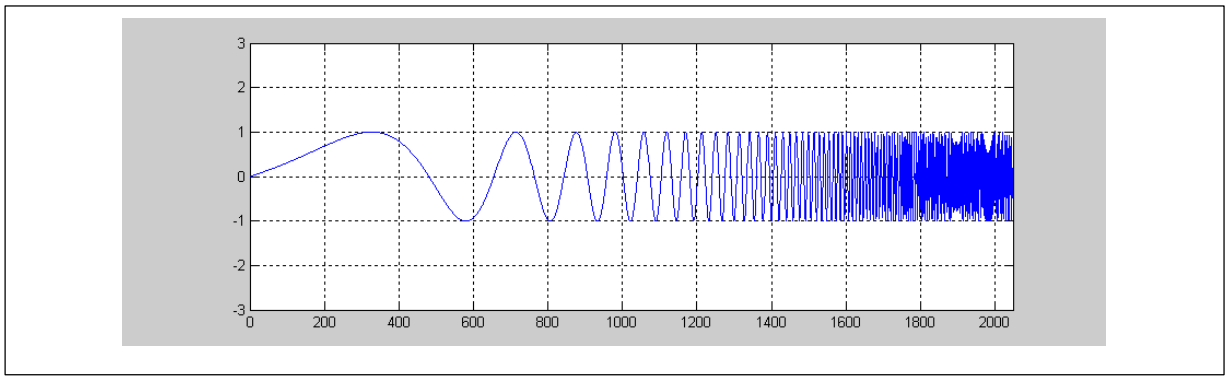
**图 16. 19X5 频率系数的过滤器响应**




每个 4 阶滤波器具有 10 个系数，意味着排列成 950 个不同滤波器系数。输入和输出数据为 Q31 模

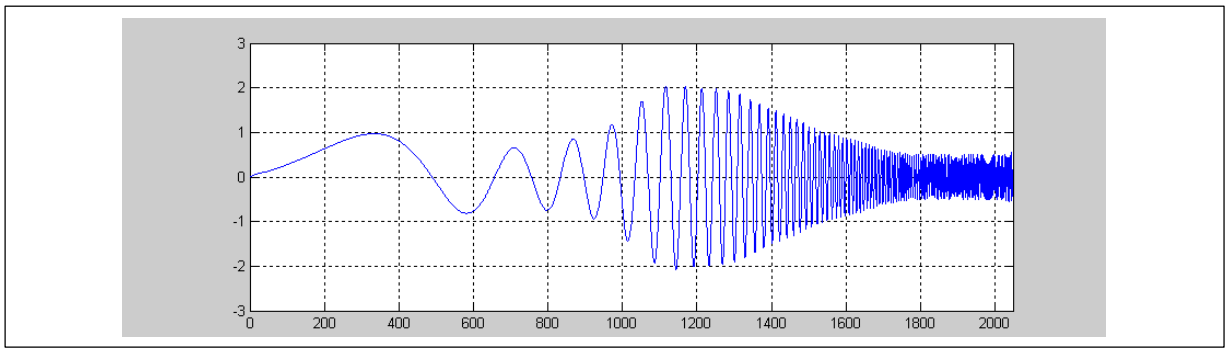
式。为了获得更好的噪声性能,两个低频算使用高精度 32x64 位双二阶滤波器。本示例中的输入信号使用对数线性调频。

**图 17. 输入信号对数线性调频**




数组 bandGains 指定以 dB 为单位的增益应用于每个带宽。例如，如果 bandGains={0, -3, 6, 4, -6}；那么输出信号将是：

**图 18. bandGains 调频输出信号**




注意：

输出线性调频信号跟随着每个带宽的增益或增强而变化。

变量描述:

- testInput_f32 ： 指向输入数据
- testRefOutput_f32 ： 指向参考输出数据
- testOutput ： 指向测试输出数据
- inputQ31 ： 临时输入缓冲区
- outputQ31 ： 临时输出缓冲区
- biquadStateBand1Q31 ： 指向 band1 的状态缓冲区
- biquadStateBand2Q31 ： 指向 band2 的状态缓冲区
- biquadStateBand3Q31 ： 指向 band3 的状态缓冲区
- biquadStateBand4Q31 ： 指向 band4 的状态缓冲区
- biquadStateBand5Q31 ： 指向 band5 的状态缓冲区
- coeffTable ： 指向所有频段的系数缓冲区
- gainDB ： 增益缓冲器，其增益适用于所有频段

使用到 DSP 软件库的函数有：

- arm_biquad_cas_df1_32x64_init_q31()
- arm_biquad_cas_df1_32x64_q31()
- arm_biquad_cascade_df1_init_q31()

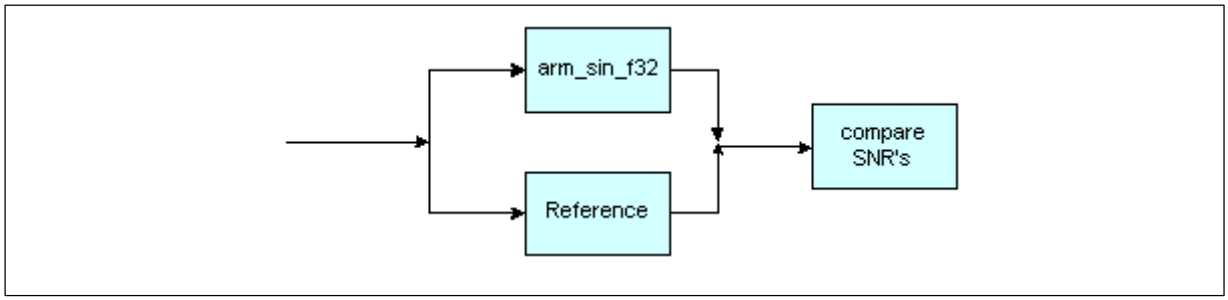
- arm_biquad_cascade_df1_q31()
- arm_scale_q31()
- arm_scale_f32()
- arm_float_to_q31()
- arm_q31_to_float()

参考
AT32_DSP_DEMO\project\at_start_f403a\examples\5_6_arm_graphic_equalizer_example

## 5.7 线性插值示例

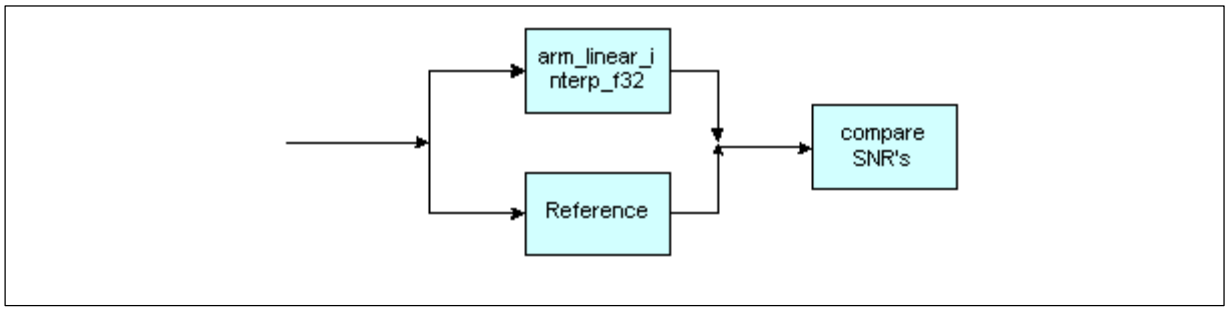
描述：
本案示例展示了线性插值模型和快速数学模型的用法。方法 1 使用快速数学正弦函数通过三次插值计算正弦值。方法 2 使用线性插值函数并将结果与参考输出进行比较。示例显示，与快速数学正弦计算相比，线性插值函数可用于获得更高的精度。
算法 1：使用快速数学函数进行正弦计算

**图 19. 快速数学函数算法框图**




算法 2：使用插值函数进行正弦计算

**图 20. 插值函数算法框图**




变量描述:

- testInputSin_f32 指向用于正弦计算的输入值
- testRefSinOutput32_f32 指向由 matlab 计算得到输出参考值 p
- testOutput 指向由三次插值计算得到的输出缓冲
- testLinIntOutput 指向由线性插值计算得到的输出缓冲
- snr1 参考输出和三次插值输出的信噪比
- snr2 参考输出和线性插值输出的信噪比

使用到 DSP 软件库的函数有：

- arm_sin_f32()

- arm_linear_interp_f32()

参考

AT32_DSP_DEMO\project\at_start_f403a\examples\5_7_arm_linear_interp_example

## 5.8 矩阵示例

描述：：

该示例展示了使用矩阵转置、矩阵乘法和矩阵求逆函数应用于最小二乘法处理的输入数据。最小二乘法是用于查找最佳拟合曲线，该曲线可使给定数据及的偏移平方和（最小方差）最小化。

算法：：

做考虑参数的线性组合如下：

The linear combination of parameters considered is as follows:

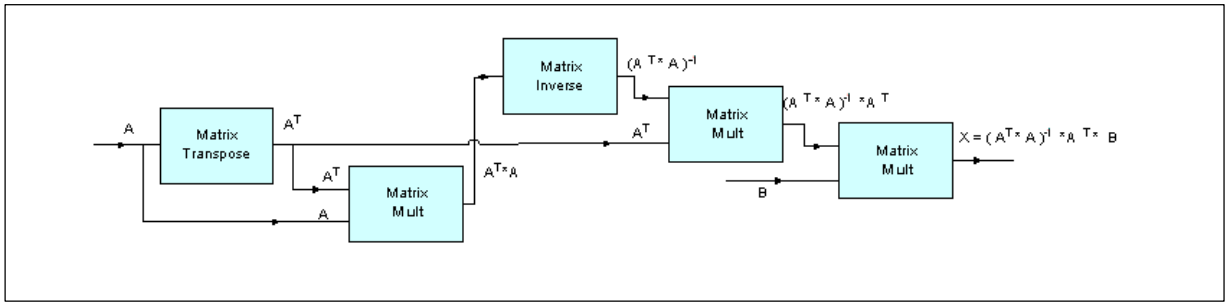
A * X = B, where X is the unknown value and can be estimated from A & B.其中 X 表示未知值，可以根据 A 和 B 进行估算。

最小二乘法估算值 X 由以下公式算出

X = Inverse(AT * A) * AT * B

框图：

**图 21. 矩阵算法框图**




变量描述:

A_f32 input matrix ： 线性组合方程的输入矩阵

B_f32 output matrix ： 线性组合方程的输出矩阵

X_f32 unknown matrix ： 矩阵 A_f32 和 B_f32 估计而得到的未知矩阵

使用到 DSP 软件库的函数有：

arm_mat_init_f32()

arm_mat_trans_f32()

arm_mat_mult_f32()

arm_mat_inverse_f32()

参考

AT32_DSP_DEMO\project\at_start_f403a\examples\5_8_arm_matrix_example

## 5.9 信号收敛示例

描述:

演示了展示了 FIR 低通滤波传递函数的自适应滤波器“学习”能力，使用到的函数有归一化 LMS 滤波器，有限冲击相应（FIR）滤波器和基本数学函数来。

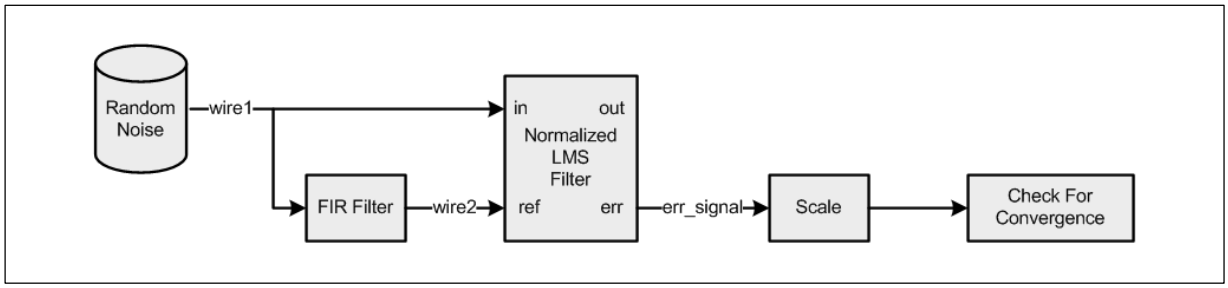
算法：

下图说明了此示例的信号流。均匀分布的白噪声通过 FIR 低通滤波器进行滤波。FIR 滤波器的输出为自适应滤波器（标准化 LMS 滤波器）的提供参考输入。白噪声是自适应滤波器的输入。自适应滤波器学习 FIR 滤波器的传递函数。该滤波器输出两个信号：（1）内部自适应 FIR 滤波器的输出 （2）自适应滤波器与 FIR 的参考输出之间的误差信号。随着自适应的滤波器不断学习学习 FIR 滤波器的传递函数，第一个输出将会接近于 FIR 滤波器的参考输出，误差信号也会不断接近于零。

即使输入信号具有大的变化范围（即，从小到大变化），自适应滤波器也能正确收敛。自适应滤波器的系数初始化为零，在 1536 个样本上，内部函数 test_signal_converge()找到停止条件。该功能检查误差信息的所有值是否都低于阈值 DELTA 的幅度。

框图：

**图 22. 信号收敛算法框图**




变量描述:

testInput_f32：指向输入数据

firStateF32：指向 FIR 状态缓冲区

lmsStateF32：指向归一化最小方差 FIR

FIRCoeff_f32：指向系数缓冲区

lmsNormCoeff_f32：指向归一化最小方差 FIR 滤波器系数缓冲区

wire1, wir2, wire3 ：临时缓冲区

errOutput, err_signal：临时错误缓冲区

使用到 DSP 软件库的函数有：

arm_lms_norm_init_f32()

arm_fir_init_f32()

arm_fir_f32()

arm_lms_norm_f32()

arm_scale_f32()

arm_abs_f32()

arm_sub_f32()

arm_min_f32()

arm_copy_f32()

参考

AT32_DSP_DEMO\project\at_start_f403a\examples\5_9_arm_signal_converge_example

## 5.10 正弦余弦示例

描述:

Demonstrates the Pythagorean trignometric identity with the use of Cosine, Sine, Vector Multiplication, and Vector Addition functions.

通过使用正弦，余弦，向量乘法和向量加法函数演示三角学的勾股定理

算法:

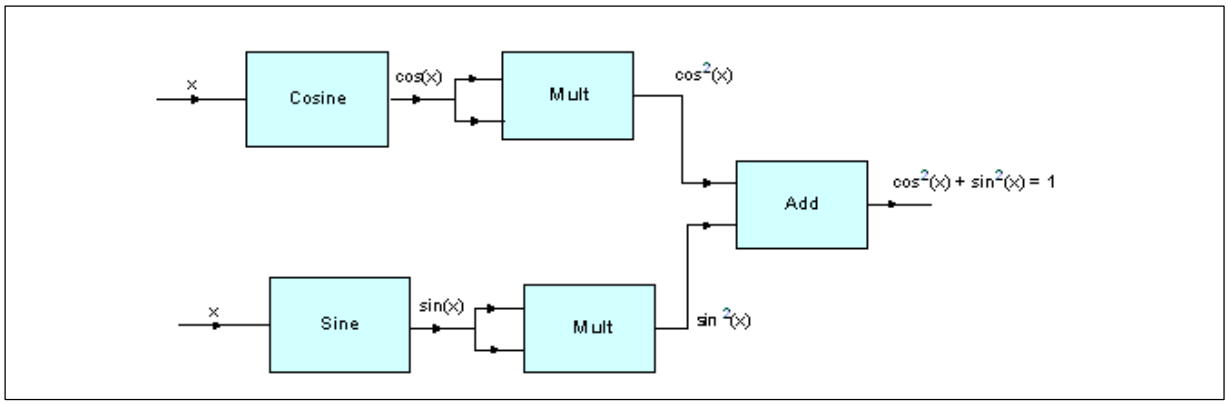
数学上，勾股三角学恒等式由以下方程式定义：

sin(x) * sin(x) + cos(x) * cos(x) = 1

其中 x 为弧度值

框图：

**图 23. 使用正弦余弦演示勾股定理算法框图**




变量描述:

testInput_f32：以弧度为单位的角度输入数组

testOutput stores：正弦值和余弦值的平方和

使用到 DSP 软件库的函数有：

arm_cos_f32()

arm_sin_f32()

arm_mult_f32()

arm_add_f32()

参考

AT32_DSP_DEMO\project\at_start_f403a\examples\5_10_arm_sin_cos_example

## 5.11 方差示例

描述:

演示如何使用基本函数和支持函数来计算 N 个样本的输入序列的方差，将均匀分布的白噪声作为输入。

算法:

序列的方差是各序列与序列平均值的平方差的平均值。

这有以下等式表示：

variance = ((x[0] - x') * (x[0] - x') + (x[1] - x') * (x[1] - x') + ... + * (x[n-1] - x') * (x[n-1] - x')) / (N-1)

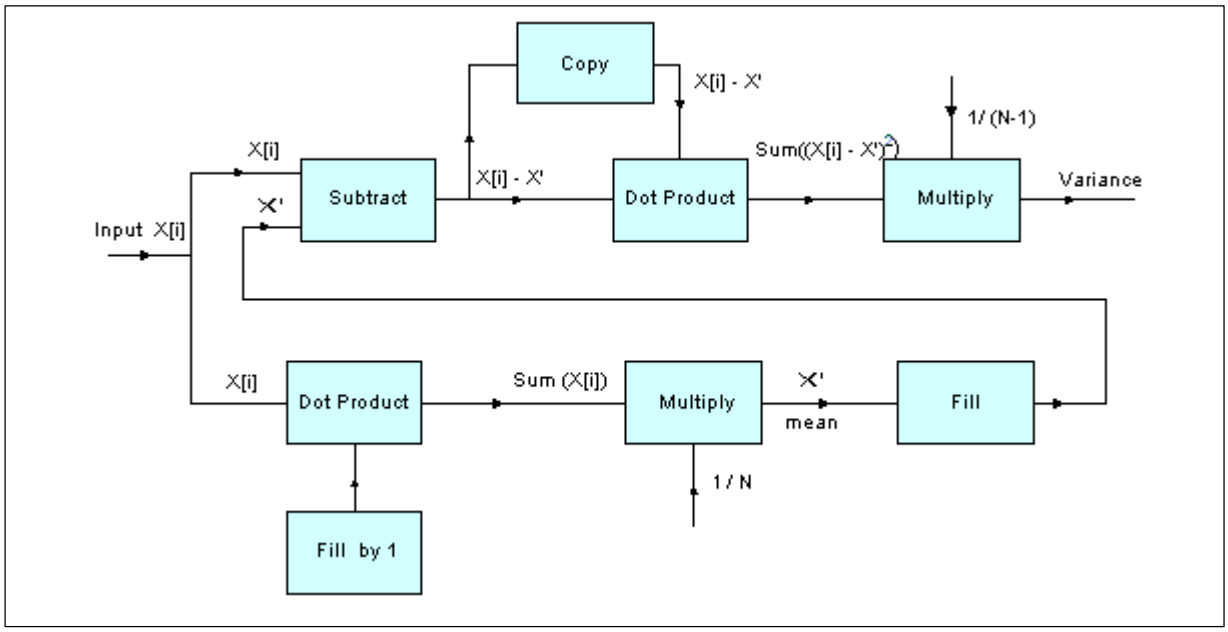
其中，x[n]是输入序列，N 输入样本数，x 是输入序列 x[n]的平均值。

平均值 x 的定义如下：

x' = (x[0] + x[1] + ... + x[n-1]) / N

框图:

**图 24. 方差算法框图**




变量描述：

testInput_f32 ： 指向输入数据

wire1, wir2, wire3 ： 临时数据缓冲区

blockSize ： 一次处理的样本数

refVarianceOut ： 参考方差值

使用到 DSP 软件库的函数有：

arm_dot_prod_f32()

arm_mult_f32()

arm_sub_f32()

arm_fill_f32()

arm_copy_f32()

参考

AT32_DSP_DEMO\project\at_start_f403a\examples\5_11_arm_variance_example

# 6 CMSIS NN with DSP

介绍

本用户手册介绍了 CMSIS NN 软件库，这是一个有效的神经网络内核的集合，这些内核的开发旨在最大程度地提高性能，并最大程度地减少神经网络在 Cortex-M 处理器内核上的存储空间。

该库分为多个函数，每个函数涵盖特定类别：

神经网络卷积函数

神经网络激活功能

全连接层功能

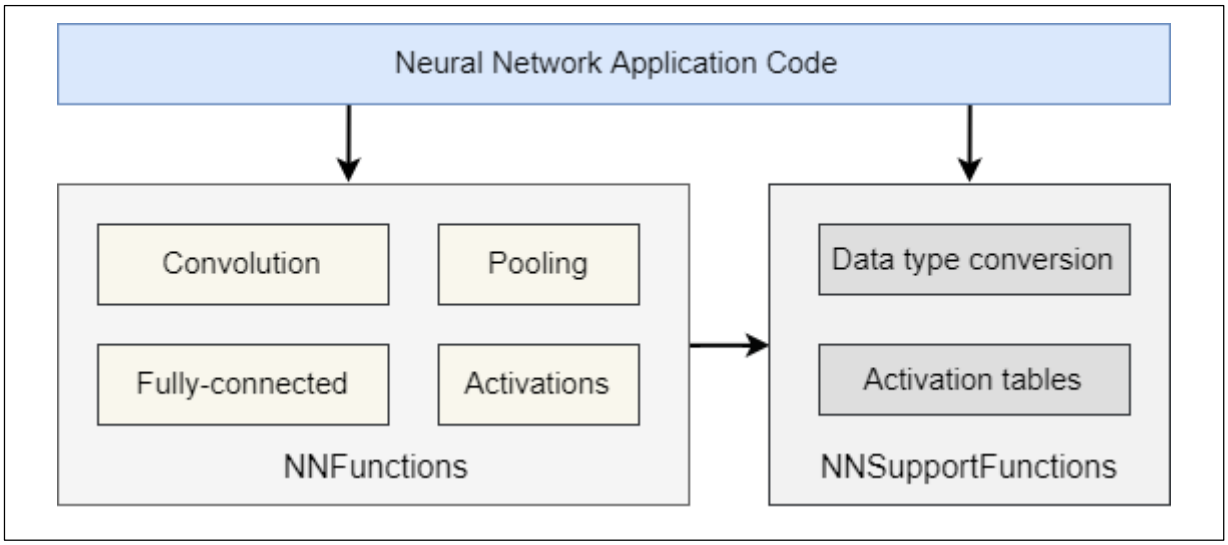
神经网络池功能

Softmax 函数

神经网络支持功能

该库具有用于对不同的权重和激活数据类型进行操作的单独函数，包括 8 位整数（q7_t）和 16 位整数（q15_t）。功能说明中包含内核的描述。本文[1]中也描述了实现细节。

**图 25. CMSIS NN 程序架构**




例子

该库附带了许多示例，这些示例演示了如何使用库函数。

预处理器宏

每个库项目都有不同的预处理器宏。

ARM_MATH_DSP：

如果芯片支持 DSP 指令，则定义宏 ARM_MATH_DSP。

ARM_MATH_BIG_ENDIAN：

定义宏 ARM_MATH_BIG_ENDIAN 来为大型字节序目标构建库。默认情况下，为小端目标建立库。

ARM_NN_TRUNCATE：

定义宏 ARM_NN_TRUNCATE 以使用 floor 而不是 round-to-the-nearest-int 进行计算

## 6.1 卷积神经网络示例

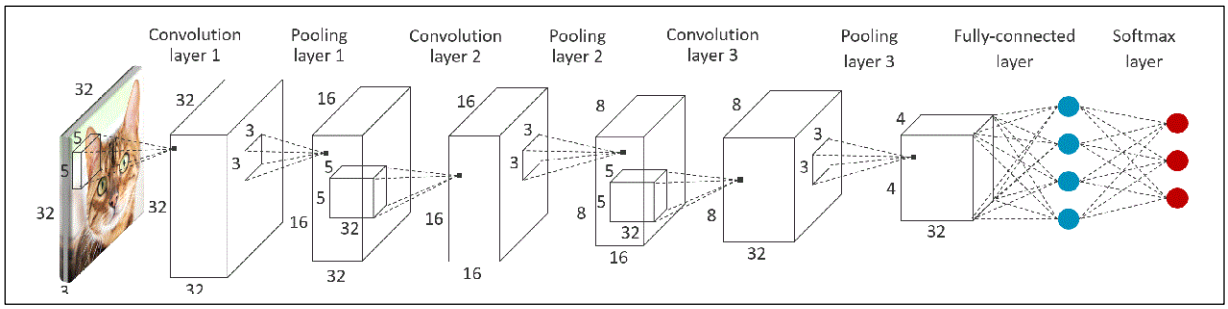
描述：

演示了使用卷积，ReLU激活，池化和全连接功能的卷积神经网络（CNN）示例。

型号定义：

本示例中使用的CNN基于Caffe [1]的CIFAR-10示例。该神经网络由3个卷积层组成，其中散布有ReLU激活层和最大池化层，最后是一个完全连接的层。网络的输入是32x32像素的彩色图像，它将被分类为10个输出类别之一。此示例模型实现需要32.3 KB的存储权重，40 KB的激活权和3.1 KB的存储im2col数据。

**图 26. CIFAR10 CN 算法框图**




神经网络模型定义

变量说明：

conv1_wt，conv2_wt，conv3_wt是卷积层权重矩阵

conv1_bias，conv2_bias，conv3_bias是卷积层偏置数组

ip1_wt，ip1_bias指向完全连接的图层权重和偏差

input_data指向输入图像数据

output_data指向分类输出

col_buffer是用于存储im2col输出的缓冲区

scratch_buffer用于存储激活数据（中间层输出）

CMSIS DSP软件库使用的功能：

arm_convolve_HWC_q7_RGB（）

arm_convolve_HWC_q7_fast（）

arm_relu_q7（）

arm_maxpool_q7_HWC（）

arm_avepool_q7_HWC（）

arm_fully_connected_q7_opt（）

arm_fully_connected_q7（）

请参阅

AT32_DSP_DEMO\project\at_start_f403a\examples\6_1_arm_nnexamples_cifar10

## 6.2 门控循环单元示例

描述：

使用完全连接的 Tanh / Sigmoid 激活功能演示门控循环单元（GRU）示例。

型号定义：

GRU 是一种递归神经网络（RNN）。它包含两个 S 型门和一个隐藏状态。

计算可以总结为：

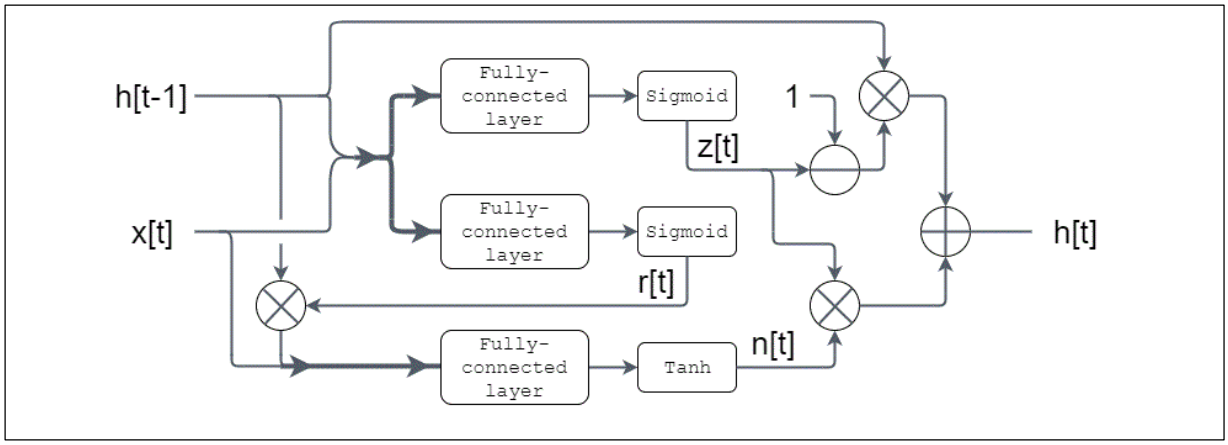
z [t] = Sigmoid（W_z·{h [t-1]，x [t]}）

r [t] = sigmoid（W_r·{h [t-1]，x [t]}）

n [t] = tanh（W_n·[r [t]×{h [t-1]，x [t]}）

h [t] =（1-z [t]）×h [t-1] + z [t]×n [t]

**图 27. 门极递归单元图**




变量说明：

update_gate_weights，reset_gate_weights，hidden_state_weights 是与更新门（W_z），重置门（W_r）和隐藏状态（W_n）对应的权重。

update_gate_bias，reset_gate_bias，hidden_state_bias 是图层偏置数组

test_input1，test_input2，test_history 是输入和初始历史记录

缓冲区分配为：

|重置|输入|历史|更新| hidden_state |

这样，由于（复位，输入）和（输入，历史记录）在存储器中被物理地隐含，所以自动完成隐含。

权重矩阵的顺序应相应调整。

CMSIS DSP 软件库使用的功能：

arm_fully_connected_mat_q7_vec_q15_opt（）

arm_nn_activations_direct_q15（）

arm_mult_q15（）

arm_offset_q15（）

arm_sub_q15（）

arm_copy_q15（）

请参阅

AT32_DSP_DEMO\project\at_start_f403a\examples\6_2_arm_nnexamples_gru

# 7 DSP Lib 的生成和使用

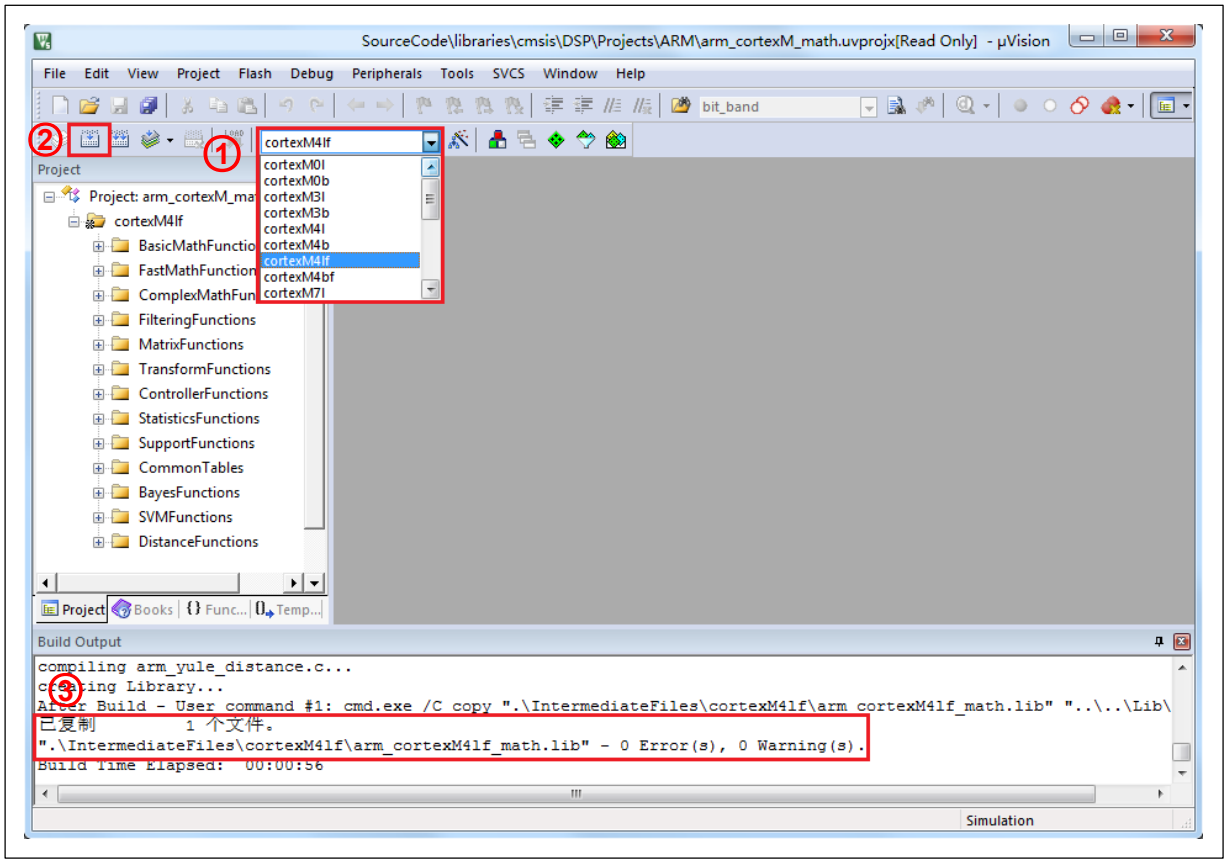
本节主要讲解如何将 DSP 源码打包为不同内核 MCU 所使用的 lib 文件。在 Artery 所提供的 DSP 包中没有包含官方所提供的 lib 文件，但包含了可生成 lib 文件的 ARM、GCC、IAR 三种编译环境的工程，用户可根据自己的需要选择适用的 lib 文件来进行生成。亦可将生成的 lib 文件替换掉工程中的 DSP 源码。下面分为两个部分来讲解 lib 文件的生成和使用。

## 7.1 DSP Lib 生成

下面以 ARM 编译环境为例，展示如何生成所需的 lib 文件：

1) 打开 SourceCode\libraries\cmsis\DSP\Projects\ARM 中的 Keil 工程；
2) 在①处 select target 下拉框选择所需生成的 lib 文件；
3) 点击②处进行编译；
4) 待③处显示 lib 文件生成信息；
5) 在 SourceCode\libraries\cmsis\DSP\Lib\ARM 中查看生成的 lib 文件。

**图 28. DSP Lib 生成**

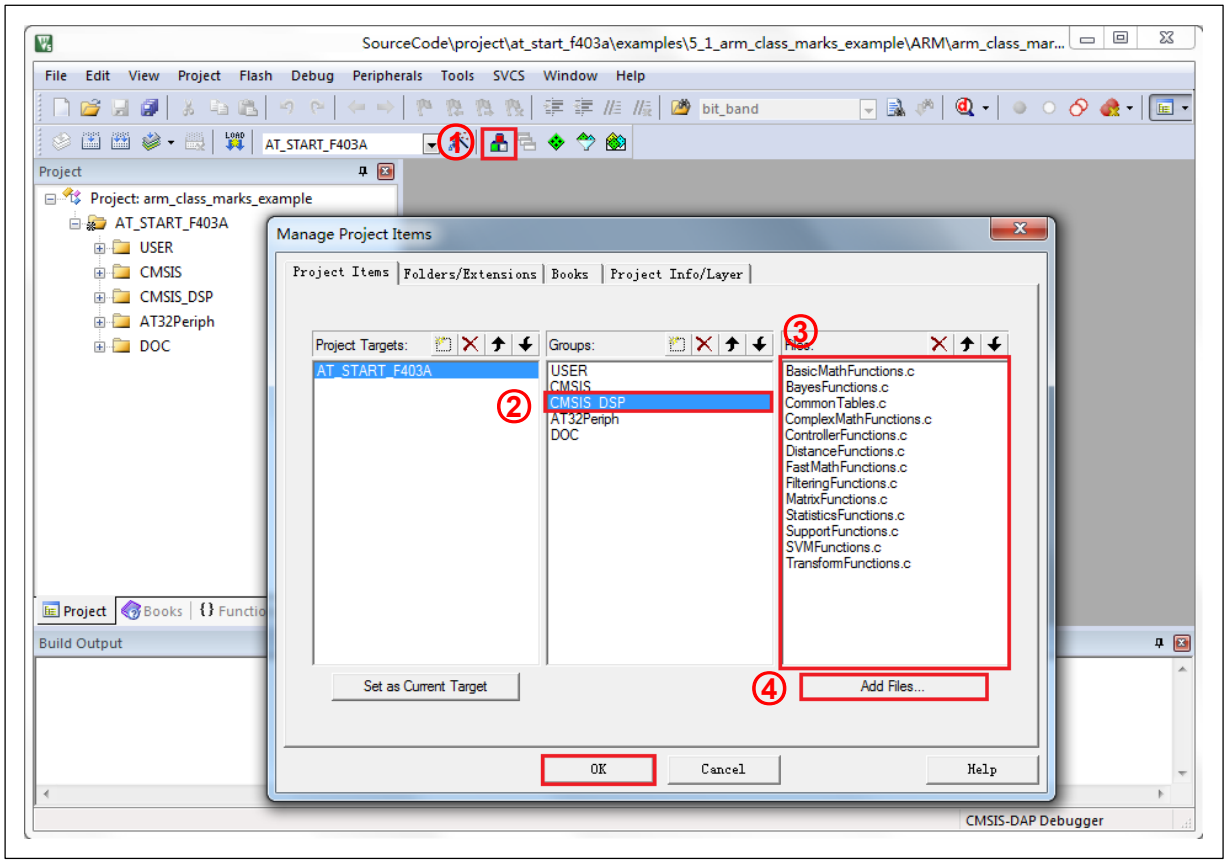



## 7.2 DSP Lib 使用

下面以 5_1_arm_class_marks_example 为例，展示如何使用 lib 文件：

1) 点击①处打开 manage project items 界面；
2) 点击②处，将③处内容全部删除；
3) 点击④处找到 SourceCode\libraries\cmsis\DSP\Lib\ARM 路径下的 lib 文件进行添加；
4) 点击 OK，编译工程。

图 29. DSP Lib 使用



# 8 版本历史

表 6. 文档版本历史

| 日期 | 版本 | 变更 |
| --- | --- | --- |
| 2021.09.10 | 2.0.0 | 最初版本 |
| 2022.03.24 | 2.0.1 | CMSIS进版为5.7.0，并新增DSP Lib的生成和使用章节 |

**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。

雅特力产品并非设计或专门用于下列用途的产品：(A) 对安全性有特别要求的应用，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；(D) 航天应用或航天环境，且/或(E) 武器。因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，风险由购买者单独承担，并且独力负责在此类相关使用中满足所有法律和法规要求。

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。