AN13498

使用恩智浦 LPC553x/LPC55S3x 中的 PowerQuad 进行数字信号处理

第 1 版 — 2022 年 5 月 25 日

应用笔记

目录

1 PowerQuad 介绍

随着移动物联网和情境感知技术的迅速发展，需要进行更多的本地数字信号处理。对基于 Cortex M33 的 MCU 来说，低功耗不间断运行系统是不错的选择（对于有限的计算量，其漏电流较低，总体低功耗也很少）。

Arm® Cortex-M33 架构朝着能效控制应用的方向发展。

但其信号处理落后于传统的 DSP 架构，有时在性能上的差距可达 10 倍至 20 倍，主要的影响因素是：

- 内存带宽较窄（单个 32 位数据总线）– DSP 通常有至少两个数据总线以及本地内存块。
- 有限的并发计算能力（例如，每个周期只有一个乘法+加法的运算）。
- 没有足够的寄存器用来保存必要的中间数据。
- 没有专用的内置加速器来实现诸如 FFT（快速傅里叶变换）所需的大量加/减法运算，以及双二阶（Biquad）滤波器之类的功能。

尽管 Arm 并没有为 Cortex-M 系列内核带来大规模的 DSP 改进，但它已经标准化了 DSP 库（CMSIS DSP Lib）。当把通用的标准接口用于 DSP 功能时，芯片厂商就有机会提供其优化的方案。虽然用户的代码仍然使用 CMSIS DSP，但是恩智浦能够从底层进行修改。加速计算可以削减功耗。MCU 可进入睡眠状态，在较慢的频率和较低的电压下低速运行（从而进一步降低能耗）。这样，PowerQuad 就应运而生了。

以下是 DSP 应用中的数学计算需求：

- 运动情境
  - 矩阵运算，三角函数的旋转，FFT（快速傅里叶变换），用于校准的滤波器（FIR/IIR）。
  - 用于运动特征提取和匹配的卷积和相关计算。
- 语音识别
  - 用于频谱分析的 FFT（快速傅里叶变换），对数，梅尔频率（Mel-Frequency）及其他窗函数（矩阵乘法），滤波器（FIR/IIR），用于提取倒谱的 DCT（离散余弦变换）。
  - 用于特征提取和比较的统计建模。
- 神经网络架构的特定功能
  - 矩阵 MAC
  - 用于感知评估的 Logistic/Sigmoid 函数（采用指数运算），对于统计分布分析也非常适用。
- 生物识别
  - 用于心率监测的 FFT（快速傅里叶变换），用于指纹识别的反正切/其他三角函数。

如今，PowerQuad 可在硬件上支持大部分数学计算需求。它能够累积进程，同时可节省出 CPU 时间，用于其他的并发线程。

2 PowerQuad 硬件

2.1 PowerQuad 的计算功能

作为一个集成在芯片中的硬件模块，PowerQuad 完全在硬件上执行计算任务。它涉及多种计算引擎：

- 变换引擎
- 超越函数引擎
- 三角函数引擎
- 双二阶 IIR 滤波器引擎
- 矩阵加速器引擎
- FIR 滤波器引擎
- CORDIC 引擎

表 1 列出了 PowerQuad 直接支持的计算功能。

表 1. PowerQuad 的硬件功能

类别	功能	注释
函数	1/x, ln(x), sqrt(x), 1/sqrt(x), e"(x), e"(-x), (x1) / (x2), sin(x), cos(x)	协处理器指令
	arctan(x), arctanh(x)	
滤波器	二阶 IIR 滤波器	协处理器指令
	• FIR 滤波器 • FIR 滤波器增量 • 相关 • 卷积	
矩阵	• 缩放 • 加法 • 减法 • 求逆 • 乘积 • 哈达玛积（逐元素乘积） • 转置 • 点积	—
变换	• 复数快速傅里叶变换（复数值输入序列） • 实数快速傅里叶变换（实数值输入序列）	—

表格续下页.....

表 1. PowerQuad 的硬件功能（续）

类别	功能	注释
	• 逆快速傅里叶变换 • 复数离散余弦变换（复数值输入序列） • 实数离散余弦变换（实数值输入序列） • 逆离散余弦变换	

这些功能构成了实现高级算法的基础。

2.2 PowerQuad 的总线接口

PowerQuad 与 Arm Cortex-M33 协处理器的接口是集成在一起的，可以通过协处理器指令（MCR 和 MRC）访问。另外，PowerQuad 内部设计有可编程的寄存器来连接 AHB 总线。在 Cortex-M33 内核上运行的用户代码可以像读写其他普通可编程模块一样读写其寄存器。请参见图 1。

但是，特定的访问方式只适用于特定的用途。通常，对于 PowerQuad，Arm Cortex-M 协处理器接口和 AHB 从机接口用于传递命令/配置，而 AHB 主机接口和专用的 RAM 主机接口则用于操作内存。

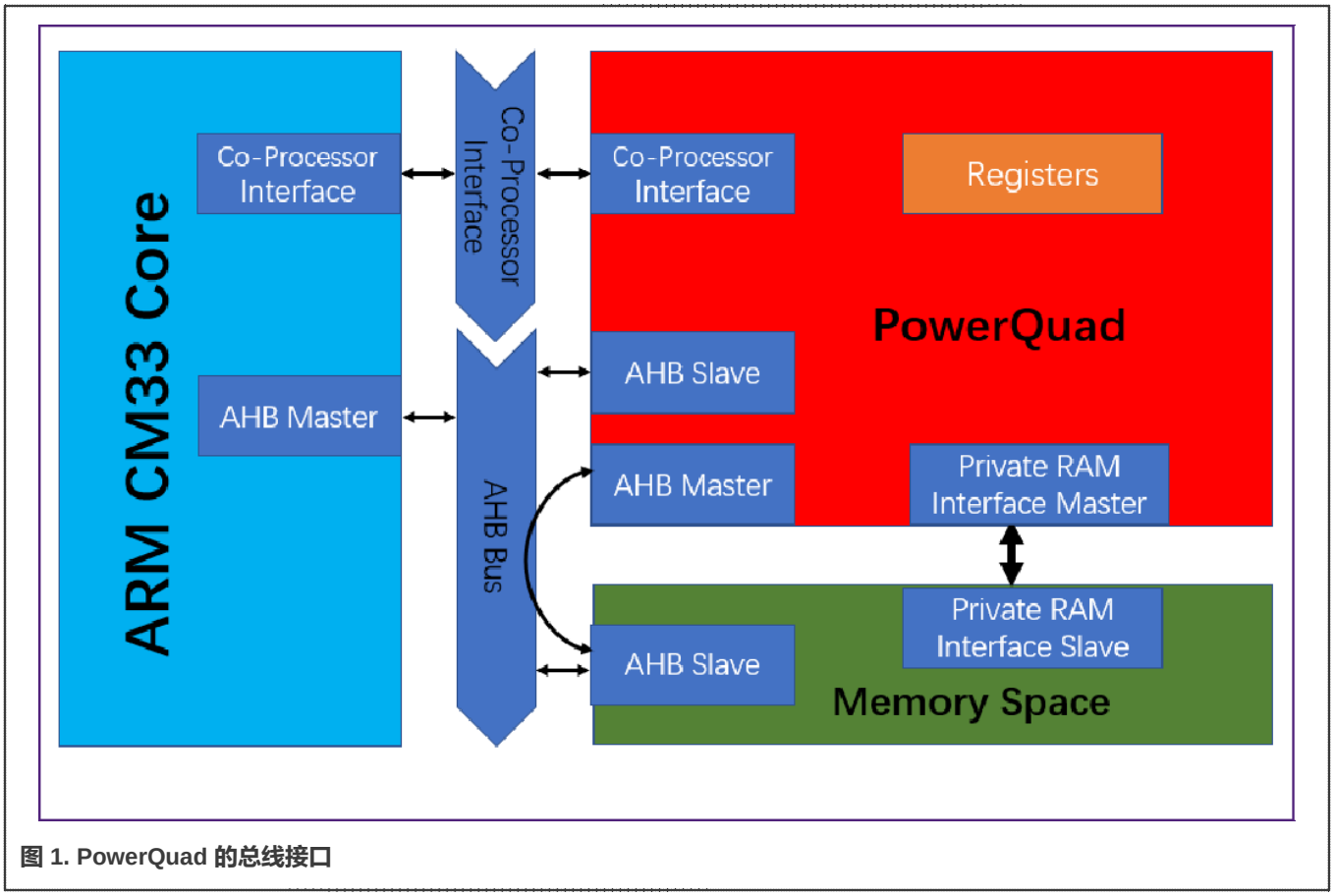

图 1. PowerQuad 的总线接口

- 协处理器功能

  在接受一个数字作为输入参数并返回一个数字作为输出结果进行计算时，通常会使用 Cortex-M 协处理器接口来传递输入参数并返回结果。例如，大多数数学计算功能都是以这种方式实现的。这些功能简单且运行得很快。

- 数据流/DMA 功能

  在对一组数据进行计算，但结果是另一组数据时，PowerQuad 使用类似于 DMA 的方式来处理输入和输出数据。AHB 访问功能的示例包括变换功能、矩阵功能和大多数滤波器功能。当把 PowerQuad 用于这些功能时，需要设置某些 PowerQuad 的基址寄存器，类似于 DMA 的使用。随后，在计算启动时，PowerQuad 硬件会自动使用这些地址指示的内存空间。

恩智浦 MCUXpresso SDK 已经为 PowerQuad 提供了驱动程序。该驱动程序打包了协处理器接口操作（协处理器指令）和 AHB 总线操作（功能寄存器）。因此，如果用户使用 SDK API 开发应用程序，则无需关心如何选择指令或寄存器的设置。

2.3 PowerQuad 的内存处理接口

当 PowerQuad 被视为一个嵌入式数学计算器时，它需要处理和产生大量数据。除了功能强大的计算引擎外，还有四组内存处理接口，分别指向四个内存区域，支持 PowerQuad 功能的数据管理需求。

- 输入 A （Input A），指向输入数据数组 1 的指针。
- 输入 B （Input B），必要时指向输入数据数组 2 的指针。例如，当进行矩阵加法运算时，将由 Input B 处理接口指向另一个矩阵。
- 临时（Temp）指针，指向临时内存，该临时内存在必要时会保留中间计算结果（用于快速傅里叶变换和矩阵求逆）。该内存应在当前计算之前初始化，并在之后清除。PowerQuad 会在计算过程中自动对它们写入和读取数值。

将每个内存区域配置为自定义的格式：

- 原始数据的格式（32 位定点，16 位定点或 32 位浮点）
- PowerQuad 所需的数据格式（除了快速傅里叶变换是定点引擎以外，其余的均为浮点）
- 结果的缩放（PowerQuad 能够在输出的过程中按 2 的幂进行缩放。）

用户可以将准备好的内存地址填充到 PowerQuad 模块相应的寄存器中。如表 2 所示。

表 2. Power Quad 的内存处理接口寄存器

地址	名称	说明	访问	复位值
0x000	OUTBASE	输出区域的基址寄存器	RW	0
0x004	OUTFORMAT	输出区域的数据格式	RW	0
0x008	TMPBASE	临时区域的基址寄存器	RW	0
0x00C	TMPFORMAT	临时区域的数据格式	RW	0
0x010	INABASE	输入 A 区域的基址寄存器	RW	0
0x014	INAFORMAT	输入 A 区域的数据格式	RW	0

表格续下页.....

表 2. PowerQuad 的内存处理接口寄存器（续）

地址	名称	说明	访问	重置值
0x018	INBBASE	输入 B 区域的基址寄存器	RW	0
0x01C	INBFORMAT	输入 B 区域的数据格式	RW	0

PowerQuad 可以处理常规 RAM 内存（与 Cortex-M 内核等其他 AHB 主设备共享）和专用 RAM 内存（从 0xE000_0000 开始，16 KB）。特别是对于专用 RAM 内存，由于它只是保留给了 PowerQuad，因此 PowerQuad 对它的访问没有任何仲裁延迟，这就节省了 PowerQuad 获取数据的时间。而且，PowerQuad 还能够并行地访问这四块专用 RAM 内存，从而达到 128 位的带宽。因此，它甚至能以更快的速度执行某些功能，例如 FFT（快速傅里叶变换）、FIR、卷积和矩阵等。

使用该专用 RAM 时，请注意：

- 快速傅里叶变换引擎只能将专用内存用作临时内存（不能用作输入或输出内存）。
- 专用内存中的所有数据必须为浮点数。（可以通过以专用内存为对象的矩阵缩放操作将数据移入和移出专用内存）。
- 专用内存不提供任何缩放。缩放仅适用于正在读取/写入系统内存的数据。

3 PowerQuad DSP 示例

本节将在讲解示例的过程中，介绍 PowerQaud 在应用中的基本用法和 PowerQuad 的 API。

该示例在 LPCXpresso55S36 板上运行，板上带有 LCD 屏模块以显示 GUI。在示例工程中，一个简单的框架能将一个单独的进程切换为调度器，逐一执行简单的任务，包括快速傅里叶变换、矩阵和 FIR。再结合 LCD 屏模块，将显示功能也集成到该框架中。

本示例选择了 PowerQuad 的快速傅里叶变换、矩阵和 FIR 滤波器计算。这些计算在大多数 DSP 应用中都很常用，但在用纯软件（Arm CMSIS-DSP Lib）实现时，通常要花费大量时间。PowerQuad 与 Arm CMSIS-DSP 的性能对比提供了 PowerQuad 的 API 与 Arm CMSIS-DSP API 的性能比较。

有关计算过程的细节将不在本应用笔记中讨论。有关更多信息，请参见 PowerQuad UM 和 SDK 驱动程序代码。

本节以快速傅里叶变换为例，详细介绍了使用 PowerQuad 的 API 的方法。同样的思路也适用于其他情况。

3.1 硬件环境设置

在运行 DSP 示例之前，需设置硬件环境。

- 准备一个 LPCXpresso55S36。
- 准备一个 LCD 模块（2.8 英寸的 TFT 屏）
- 将 LCD 连接到 LPCXpresso55S36（J102，J132，J92，J122）
- 将 JP64 的引脚 3 连接到 J9 的 D13（由于 `flexspi` 使用 JP64 的引脚 1 和引脚 2，LCD 的 BK 必须跳接到 J9 的 D13。）
- 使用 `blhost.exe` 将位于.`\docs\images` 的映像 bin 文件下载到 `flexspi` 闪存中。要完成下载，请按照《LPC553x 和 LPC55S3x 参考手册》中描述的，通过 blhost 工具进行 NOR FLASH 配置、擦除和编程的步骤进行操作。表 3 列出了外部闪存中映像 bin 文件的目标地址。

表 3. 映像 bin 文件的目标地址

映像 bin 文件	目标地址
Wel24b.bin	0x08000000
MAdd24b.bin	0x08030000
MInv24b.bin	0x08060000
MMul24b.bin	0x08090000
Tab24b.bin	0x080C0000

- 将代码下载到 LPC5536/LPC55S36 芯片的内部闪存中并运行。
- 要逐个切换不同的 DSP 实例，请按 sw3（USR）按钮。

3.2 带显示 GUI 的任务调度

为了将单独的实例合并到一个工程中，在演示工程中实现了一个调度器。每个实例都在一个子程序内实现，并将该程序作为任务的入口。所有的任务入口都集中到任务数组 `cAppLcdDisplayPageFunc[]`中。同时，还启动了一个用于捕获按钮的硬件线程。

接着，MCU 进入睡眠模式，直到被按键中断唤醒。键值在按键中断的 ISR 中被改变。主循环检查键值的变化，并使用任务列表中的索引（使用键值）切换到相应的任务。

```
/* List of lcd display with tasks. */
void (*cAppLcdDisplayPageFunc[])(void) =
   {
      task_pq_fft_128,
      task_pq_fft_256,
      task_pq_fft_512,
      task_pq_mat_add,
      task_pq_mat_inv,
      task_pq_mat_mul,
      task_pq_fir_lowpass,
      task_pq_fir_highpass,
      task_pq_records
    };
    int main(void)
    {
        ...
        while (1)
        {
            keyValue = App_GetUserKeyValue(); /* keyvalue is used as the index of task. */
            if (keyValue != keyValuePre) /* only switch task when keyvalue is changed. */
            {
            App_DeinitUserKey(); /* disable detecting key when changing lcd display. */
              (*cAppLcdDisplayPageFunc[keyValue])(); /* switch to new page with new task. */
              keyValuePre = keyValue;
              App_InitUserKey(); /* enable detecting key for next event. */
            }
             _ WFI(); /* sleep when in idle. would wake up when the key interrupt happens caused by
the touch screen. */
          }
      }
```

在每个任务中，都会执行 PowerQuad 计算以完成一个简单的任务，并测量关键操作所花的时间。然后将测量结果显示到 LCD 屏模块。

3.3 时间测量功能

考虑到这些函数通常运行得很快，因此在本示例中不适合使用基于中断的计时方法。但是，在一些专门用于测量的测试项目中，基于中断的时间测量方法仍然适用。该方法通过测量目标函数的多次执行时间而获得单次执行的平均时间。

在此示例中，选择 `SysTick` 定时器作为计时器，此代码可以移植到其他 Arm Cortex-M MCU 中。直接使用 24 位计数器值进行计时。对于 LPC5536/LPC55S36 芯片，其 SysTick 计时器的时钟源运行在 98 MHz，最大计时周期为 171 ms。

```
/* Systick Start */
#define TimerCount_Start()    do {                  \
   SysTick->LOAD   =     0xFFFFFF    ;    /*    Set reload register    */\
   SysTick->VAL    =    0    ;         /*    Clear Counter */    \
   SysTick->CTRL    =    0x5 ;         /*    Enable Counting*/    \
} while(0)

/* Systick Stop and retrieve CPU Clocks count */
#define TimerCount_Stop(Value) do {    \
SysTick->CTRL =0; /* Disable Counting */    \ Value = SysTick->VAL;/* Load the SysTick Counter Value
*/ \ Value = 0xFFFFFF - Value;/* Capture Counts in CPU Cycles*/\ } while(0)
```

使用方法是：

```
uint32_t calcTime;

TimerCount_Start();
arm_cfft_q31(&instance, gPQFftQ31InOut, 0, 1); /* Calculation. */
TimerCount_Stop(calcTime);

printf("calcTime: %d", calcTime);
```

3.4 快速傅里叶变换演示实例

该演示中有三种快速傅里叶变换实例：128 点，256 点和 512 点。下面是使用 PowerQuad 快速傅里叶变换引擎的提示：

- PowerQuad 可以在硬件上支持 16/32/64/128/256/512 点的快速傅里叶变换计算引擎。
- 当 PowerQuad 快速傅里叶变换引擎在计算快速傅里叶变换（以及扩展离散余弦变换）时，它按 1/N 的比例缩放输入数据。如果需要非缩放的结果，则需要手动将输入数据（在输入 A 区域）乘以 N，并将逆快速傅里叶变换的缩放设置为 1/N。根据 iDFT 的公式，这样就可以了，不再需要另外的缩放处理。
- 快速傅里叶变换引擎仅查看输入字的底部 27 位，所以预缩放不能过大以避免饱和。
- 纯实数函数（在 API 名称中以 ‘r’为前缀）和复数函数（在 API 名称中以‘c’为前缀）要求输入数据序列在内存中按如下方式排列。
- 如果序列 x = x0，x1 ... xN-1 是实数，则内存中的输入数组将以 x[N] = {x0，x1，... xN-1}的形式存放。
- 如果序列 x = x0，x1 ... xN-1 是（x0_real + i * x0_im），（x1_real + i * x1_im），...（xN-1_real + i * xN-1_im）形式的复数，则内存中的输入数组将以 x[N] = {x0_real，x0_im，x1_real，x1_im，... xN-1_real，xN-1_im}的形式存放。
- 输出序列以复数数组的形式存储在内存中，其中实数输出数据的虚部为零。

当运行 PowerQuad 变换引擎（包括快速傅里叶变换）时，只有 INPUT A 内存处理接口用于输入，而 OUT 内存处理接口则用于输出。有关变换引擎内存处理接口使用方法的完整信息，请参见表 4。

表 4. 快速傅里叶变换引擎的内存处理接口的用法

操作方式	驱动功能	访问类型	输入/输出数据格式	Input A 区域用法	Input B 区域用法	输出区域用法	临时区域用法	定点输入/输出缩放器	引擎	是否使用 GPREG/ COMPREG?
复数快速傅里叶变换	`Pq_cfft`	AHB	Fix-16, Fix-32	输入数据	N.A.	输出数据	N.A	`Ina_scaler/ Inb_scaler/ Out_scaler`	X_{form}	是
实数快速傅里叶变换	`Pq_rfft`	AHB	Fix-16, Fix-32	输入数据	N.A.	输出数据	N.A	`Ina_scaler/ Inb_scaler/ Out_scaler`	X_{form}	是
逆快速傅里叶变换	`Pq_ifft`	AHB	Fix-16, Fix-32	输入数据	N.A.	输出数据	N.A.	`Ina_scaler/ Inb_scaler/ Out_scaler`	X_{form}	是
复数离散余弦变换	`Pq_cdct`	AHB	Fix-16, Fix-32	输入数据	N.A.	输出数据	N.A.	`Ina_scaler/ Inb_scaler/ Out_scaler`	X_{form}	是
实数离散余弦变换	`Pq_rdct`	AHB	Fix-16, Fix-32	输入数据	N.A.	输出数据	N.A.	`Ina_scaler/ Inb_scaler/ Out_scaler`	X_{form}	是
逆离散余弦变换	`Pq_idct`	AHB	Fix-16, Fix-32	输入数据	N.A.	输出数据	N.A.	`Ina_scaler/ Inb_scaler/ Out_scaler`	X_{form}	是

演示中使用的 PowerQuad API 与 CMSIS-DSP API 兼容。CMSIS-DSP 用户不需要更改现有代码，就可以通过 PowerQuad 实现更快的运行。

以 128 点的快速傅里叶变换为例：

```
extern q31_t    gPQFftQ31In[APP_PQ_FFT_SAMPLE_COUNT_MAX*2u];
extern q31_t    gPQFftQ31Out[APP_PQ_FFT_SAMPLE_COUNT_MAX*2u];
extern q31_t    gPQFftQ31InOut[APP_PQ_FFT_SAMPLE_COUNT_MAX*2u];
extern float32_t gPQFftF32In[APP_PQ_FFT_SAMPLE_COUNT_MAX*2u];
extern float32_t gPQFftF32Out[APP_PQ_FFT_SAMPLE_COUNT_MAX*2u];

void task_pq_fft_128(void)
{
     arm_cfft_instance_q31 instance;
     uint32_t i;
     uint32_t calcTime;

     /* Create the input signal. */
     for (i = 0; i < APP_PQ_FFT_SAMPLE_COUNT_128; i++)
     {
         /* real part. */
         gPQFftF32In[i*2] = 1.5f /* direct current. */
                            + 1.0f * arm_cos_f32( (    2.0f * PI / APP_PQ_FFT_PERIOD_BASE) *
i ) /* low frequence */
                            + 0.5f * arm_cos_f32( (4.0f * 2.0f * PI / APP_PQ_FFT_PERIOD_BASE) *
i ) /* high frequence */
                             ;
            gPQFftF32In[i*2] /= 3.0f; /* make sure the value in (0, 1) */
            /* imaginary part */
            gPQFftF32In[i*2+1] = 0.0f;

         }
    /* PowerQuad FFT can only operate fix-point number. */
         arm_float_to_q31(gPQFftF32In, gPQFftQ31In, APP_PQ_FFT_SAMPLE_COUNT_128*2u);
         for (i = 0u; i < APP_PQ_FFT_SAMPLE_COUNT_128 * 2u; i++)
         {
            gPQFftQ31InOut[i] = gPQFftQ31In[i] >> 5u; /* powerquad fft engine can only accept 27-bit
input data. */
          }

          instance.fftLen = APP_PQ_FFT_SAMPLE_COUNT_128;
          TimerCount_Start(); /* start timing. */
          arm_cfft_q31(&instance, gPQFftQ31InOut, 0, 1); /* computing. */
          TimerCount_Stop(calcTime);

          for (i = 0u; i < APP_PQ_FFT_SAMPLE_COUNT_128 * 2u; i++)
          {
              gPQFftQ31Out[i] = gPQFftQ31InOut[i] « 5u; /* restore the data from 27-bit to 32-bit. */
          }

          arm_q31_to_float(gPQFftQ31Out, gPQFftF32Out, APP_PQ_FFT_SAMPLE_COUNT_128*2u);
arm_cmplx_mag_f32( gPQFftF32Out, gPQFftF32In, APP_PQ_FFT_SAMPLE_COUNT_128);
          /* Todo ...
          * - Record the time.
          * - Display the waveform.
          */
   }
```

`arm_cfft_q31()`调用 PowerQuad 驱动程序 `PQ_TransformCFFT()`/`PQ_TransformIFFT()`。

```
  void arm_cfft_q31(const arm_cfft_instance_q31 *S, q31_t *p1, uint8_t ifftFlag, uint8_t
bitReverseFlag)
```

```
  {
       assert(bitReverseFlag == 1);

       q31_t *pIn = p1;
       q31_t *pOut = p1;
       uint32_t length = S->fftLen;

       PQ_DECLARE_CONFIG;
       PQ_BACKUP_CONFIG;
       PQ_SET_FFT_Q31_CONFIG;

       if (ifftFlag == 1U)
       {
           PQ_TransformIFFT(POWERQUAD_NS, length, pIn, pOut);
        }
        else
        {
            PQ_TransformCFFT(POWERQUAD_NS, length, pIn, pOut);
         }

            PQ_WaitDone(POWERQUAD_NS);

            PQ_RESTORE_CONFIG;
  }
```

然后，`PQ_TransformCFFT()`函数配置 PowerQuad 寄存器以设置输入/输出和内存长度；之后启用 PowerQuad 作为 CFFT 引擎来启动计算。完成这些操作后，PowerQuad 开始工作。

```
void PQ_TransformCFFT(POWERQUAD_Type *base, uint32_t length, void *pData, void *pResult)
{
     assert(pData);
     assert(pResult);

     base->OUTBASE = (int32_t)pResult;
     base->INABASE = (int32_t)pData;
     base->LENGTH = length;
     base->CONTROL = (CP_FFT « 4) | PQ_TRANS_CFFT; /* Launch the computing task. */
}
```

当计算完成后，`INST_BUSY`生效。用户可以使用`PQ_WaitDone()`函数来等待 PowerQuad 完成计算。

```
void PQ_WaitDone(POWERQUAD_Type *base)
{
    /* wait for the completion */
    while ((base->CONTROL & INST_BUSY) == INST_BUSY)
    {
           _ WFE(); /* Enter to low power. */
    }
}
```

当运行演示工程时，每种快速傅里叶变换实例在 LCD 屏模块上都有显示，如图2 所示。

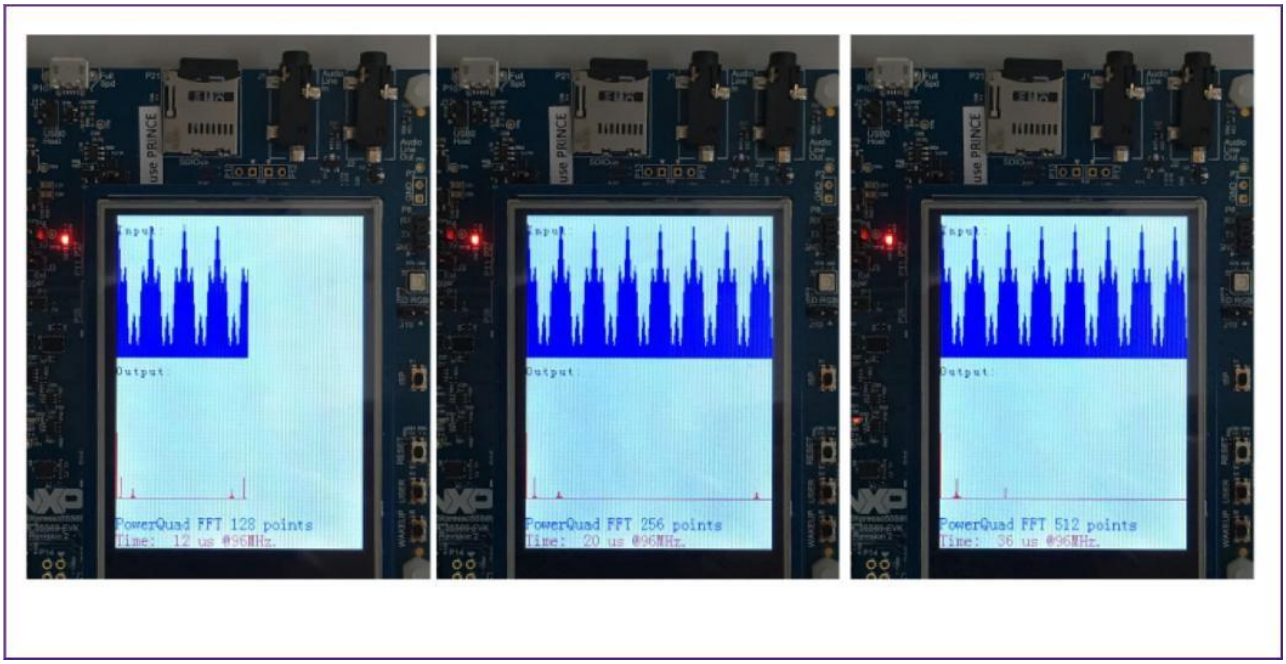

图 2. PowerQuad 快速傅里叶变换 128/256/512 点

3.5 矩阵演示实例

矩阵加速器引擎支持八种操作。表 5 列出了这些操作和它们的最大支持维度。

表 5. PowerQuad 矩阵长度范围

PowerQuad 引擎	操作方式	最大行
矩阵	加法	16 × 16
	减法	16 × 16
	哈达玛积	16 × 16
	矩阵乘法	16 × 16
	向量点积	256 元素
	矩阵求逆	9 × 9
	转置	16 × 16
	缩放	16 × 16

矩阵数据按标准 C / C ++阵列逐行存储在内存中。 因此，如果两个 2×2 整数矩阵 A 和 B 为：

```
A =    [1    2]    B =    [5    6]
       [3    4]           [7    8]
```

那么输入数据被存储在内存阵列中，如下所示：

```
int    MatA[4]    =    {1,    2,    3,    4};
int    MatB[4]    =    {5,    6,    7,    8};
```

有关 PowerQuad 的矩阵引擎的内存处理接口的用法，请参见表 6。

表 6. 矩阵引擎的内存处理接口的用法

操作方式	驱动功能	访问类型	输入/输出数据格式	Input A 区域用法	Input B 区域用法	输出区域用法	临时区域用法	引擎
矩阵加法	Pq_mtx_add	AHB	FP, Fix-16, Fix-32	矩阵 M1	矩阵 M2	结果矩阵	N.A.	矩阵（Matrix）
矩阵减法	Pq_mtx_sub	AHB	FP, Fix-16, Fix-32	矩阵 M1	矩阵 M2	结果矩阵	N.A.	矩阵（Matrix）
矩阵哈达玛积	Pq_mtx_hadamard	AHB	FP, Fix-16, Fix-32	矩阵 M1	矩阵 M2	结果矩阵	N.A.	矩阵（Matrix）
矩阵积	Pq_mtx_pro d	AHB	FP, Fix-16, Fix-32	矩阵 M1	矩阵 M2	结果矩阵	N.A.	矩阵（Matrix）
矩阵逆	Pq_mtx_inv	AHB	FP, Fix-16, Fix-32	矩阵 M1	N.A.	结果矩阵	最多 1024 个字	矩阵（Matrix）
矩阵转置	Pq_mtx_tra n	AHB	FP, Fix-16, Fix-32	矩阵 M1	N.A.	结果矩阵	N.A.	矩阵（Matrix）
矩阵缩放	Pq_mtx_scale	AHB	FP, Fix-16, Fix-32	矩阵 M1	N.A.（缩放因子在 MISC 寄存器中）	结果矩阵	N.A.	矩阵（Matrix）
向量点积	Pq_vec_dot p	AHB	FP, Fix-16, Fix-32	向量 A	向量 B	缩放结果	N.A.	矩阵（Matrix）

在示例中，每个任务使用了三种计算：

- `task_pq_mat_add()`用于矩阵加法
- `task_pq_mat_mul()`用于矩阵乘法
- `task_pq_mat_inv()`用于矩阵求逆

就像快速傅里叶变换一样，PowerQuad 驱动程序也实现了 CMSIS-DSP API。以 `task_pq_mat_add()`为例，它的用法与 CMSIS-DSP API 相同。

```
#define PQ_MAT_ROW_COUNT_MAX    16u
  #define PQ_MAT_COL_COUNT_MAX    16u
  /* A + B = C. */
  void task_pq_mat_add(void)
  {
  arm_matrix_instance_f32 matrixA;
  arm_matrix_instance_f32 matrixB;
  arm_matrix_instance_f32 matrixC;
```

```
  float32_t mDataA[PQ_MAT_ROW_COUNT_MAX][PQ_MAT_COL_COUNT_MAX];
  float32_t mDataB[PQ_MAT_ROW_COUNT_MAX][PQ_MAT_COL_COUNT_MAX];
  float32_t mDataC[PQ_MAT_ROW_COUNT_MAX][PQ_MAT_COL_COUNT_MAX];
  uint32_t i, j;
  uint32_t calcTime;

   /* Initialize the matrix. */
   for (i = 0u; i < PQ_MAT_ROW_COUNT_MAX; i++)
   {
   for (j = 0u; j < PQ_MAT_COL_COUNT_MAX; j++)
   {
    mDataA[i][j] = 1.0f * i * PQ_MAT_ROW_COUNT_MAX + j;
    mDataB[i][j] = 1.0f * i * PQ_MAT_ROW_COUNT_MAX + j;
    }
    }
    matrixA.numRows = PQ_MAT_ROW_COUNT_MAX; matrixA.numCols = PQ_MAT_COL_COUNT_MAX; matrixA.pData =
(float32_t *)mDataA; matrixB.numRows = PQ_MAT_ROW_COUNT_MAX; matrixB.numCols = PQ_MAT_COL_COUNT_MAX;
matrixB.pData = (float32_t *)mDataB; matrixC.numRows = PQ_MAT_ROW_COUNT_MAX; matrixC.numCols
= PQ_MAT_COL_COUNT_MAX;
    matrixC.pData    = (float32_t *)mDataC;
    /* Calc & Measure. */
    TimerCount_Start();
    arm_mat_add_f32(&matrixA, &matrixB, &matrixC);
    TimerCount_Stop(calcTime);

    /* Todo ...
    * - Record the time.
    * - Display the waveform.
    */
    }
```

当运行示例工程时，每个矩阵演示实例在 LCD 屏模块上都有显示，如图3 所示。

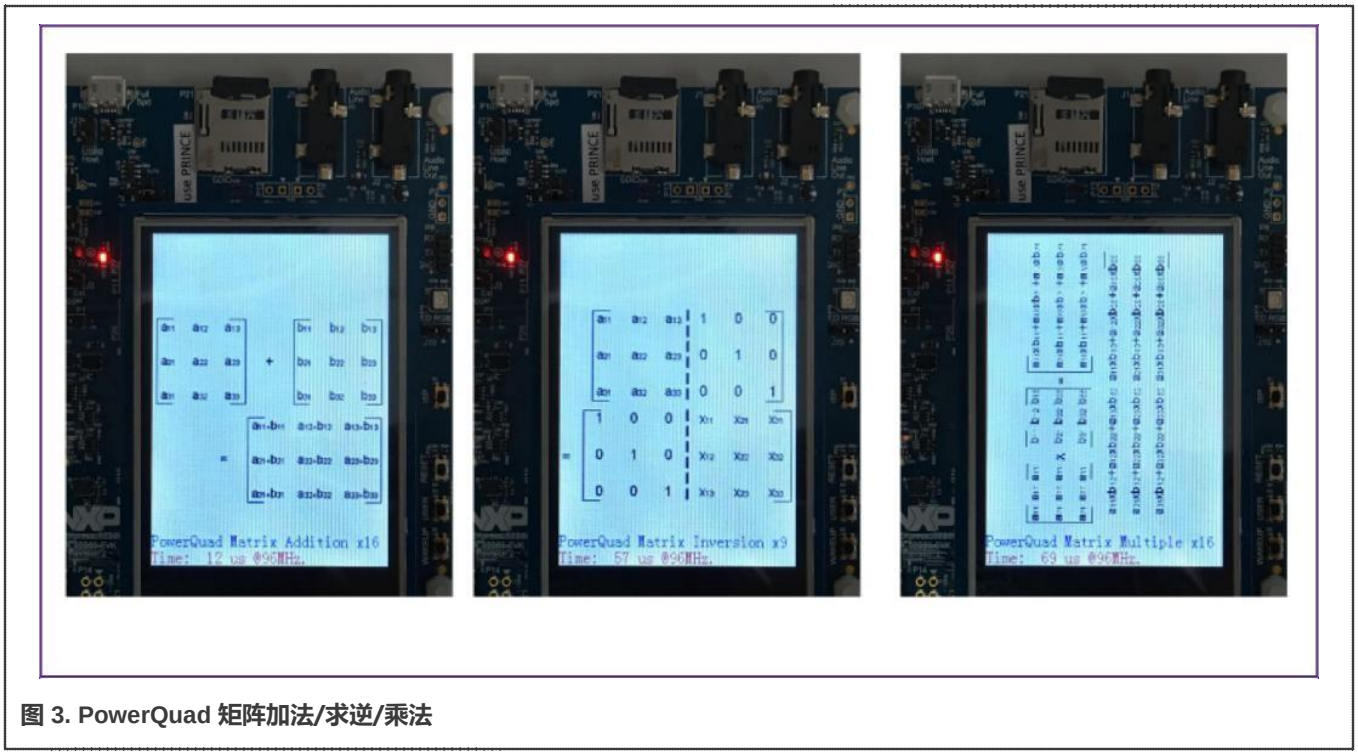

图 3. PowerQuad 矩阵加法/求逆/乘法

3.6 FIR 演示实例

该示例的目的是创建一个高通/低通 FIR 滤波器。

用两个示例来创建不同的滤波器：

- `task_pq_fir_lowpass()`用于低通滤波器，以去除混合信号中的高频信号并获取低频信号。
- `task_pq_fir_highpass()`用于高通滤波器，以去除混合信号中的低频信号并获取高频信号。

在示例中，Matlab 软件先计算滤波器的抽头（系数）。然后进入 PowerQuad，硬件会帮助自动完成对信号的过滤处理，避免了耗时的数学计算。

原始信号是低频信号（1 kHz 的正弦波）和高频信号（15 kHz 的正弦波）的混合信号。图 4 是波形，图 5 是频谱。

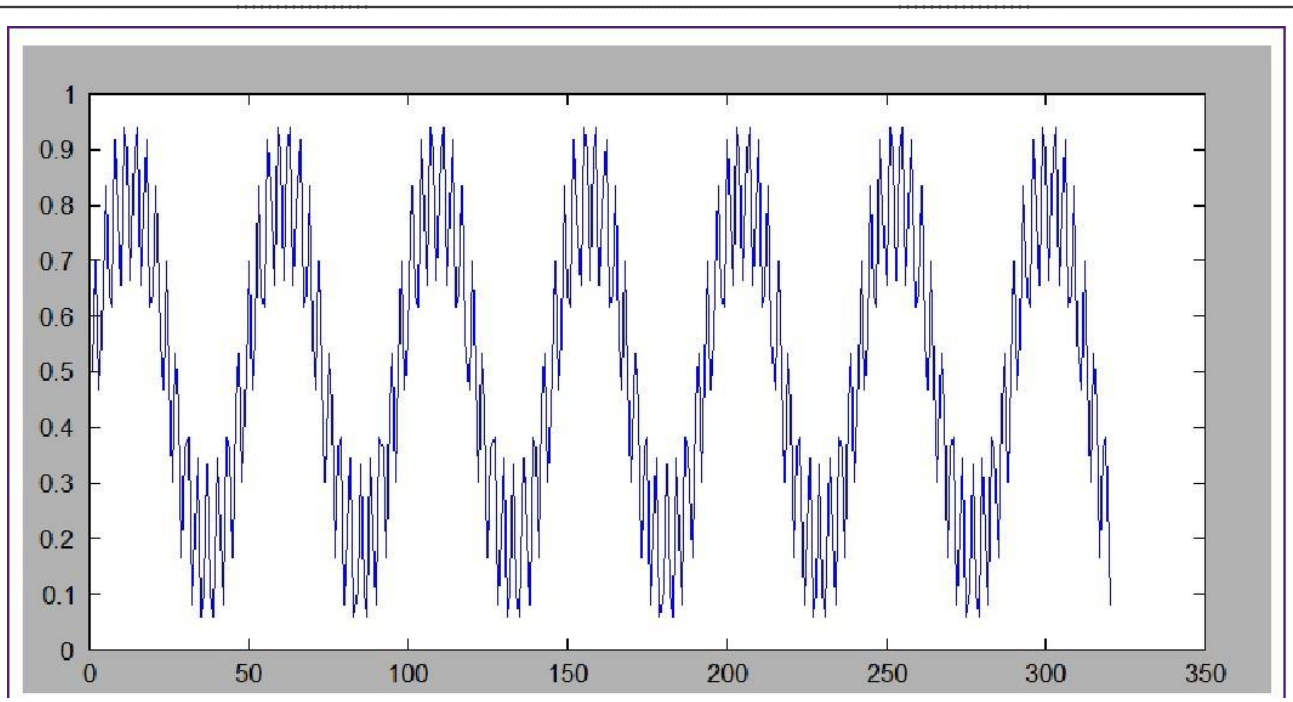

图 4. 混合信号的波形

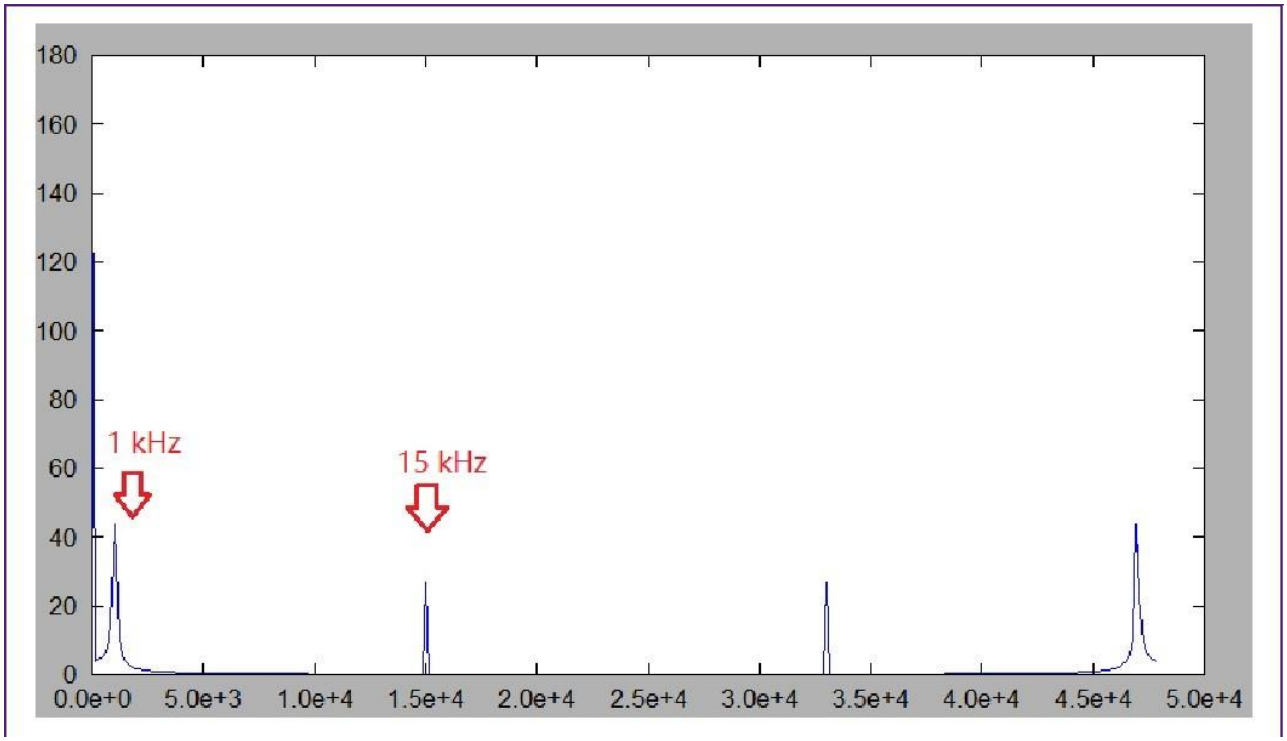

图 5. 混合信号的频谱

在 MatLab 中运行以下代码以创建系数。

```
clear all
close all
Fs=48000;
T=1/Fs;
Lenght=320;
t=(0:Lenght-1)*T;
Input_signal=(sin(2*pi*1000*t)+0.5*sin(2*pi*15000*t)+1.5)/3;
figure;
plot(Input_signal);
res=fft(Input_signal,Lenght);
figure;
f=((0:Lenght-1)/320*Fs);
plot(f,abs(res));
Cutoff_Freq=6000;
Nyq_Freq=Fs/2;
cutoff_norm=Cutoff_Freq/Nyq_Freq;
order=31;
FIR_Coeff=fir1(order,cutoff_norm,'high'); % for high-pass
%FIR_Coeff=fir1(order,cutoff_norm); % for low-pass
Filterd_signal=filter(FIR_Coeff,1,Input_signal);
figure;
plot(Filterd_signal);

fvtool(FIR_Coeff,'Fs',Fs); % generate the coeff and display the diagram
```

滤波器的特性是：

- 类型：高通/低通
- 阶数：32

- kHz 采样频率：48 kHz
- 截止频率：6 kHz

响应报告如图 6、图 7、图 8 和图 9 所示。

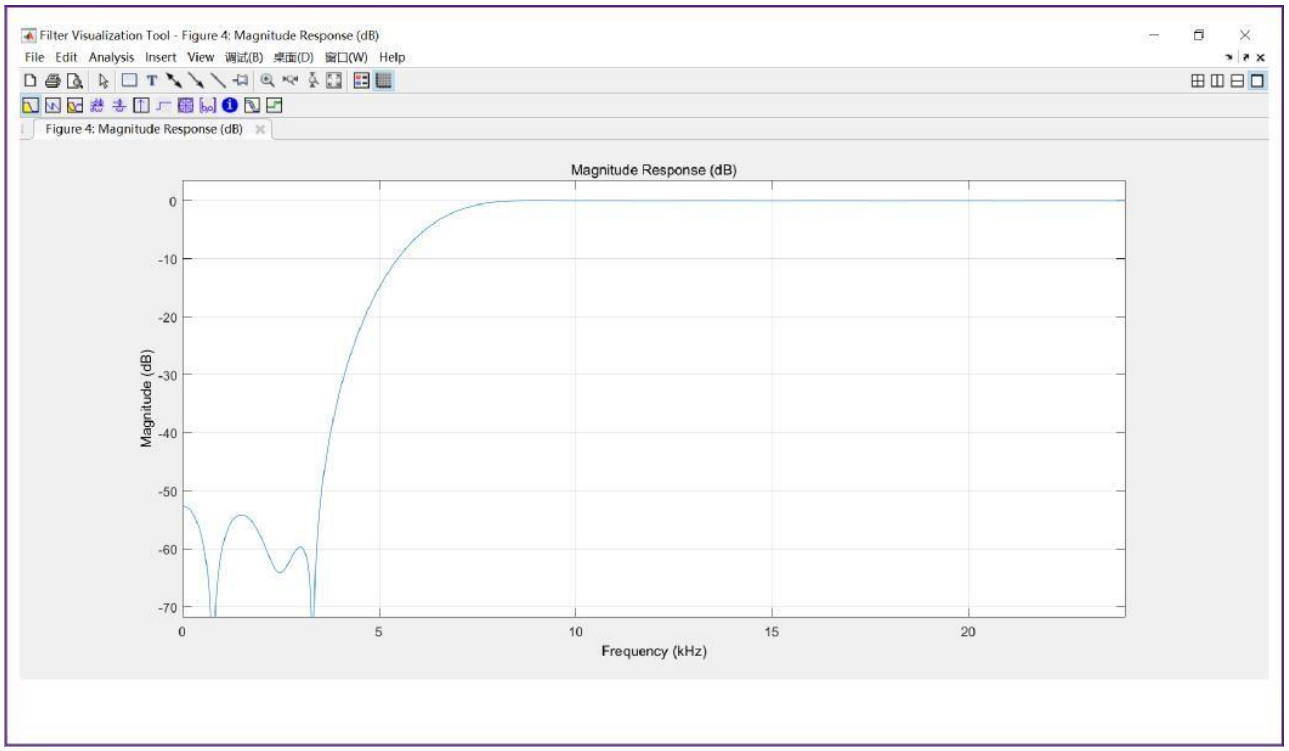

图 6. FIR 滤波器的幅值响应

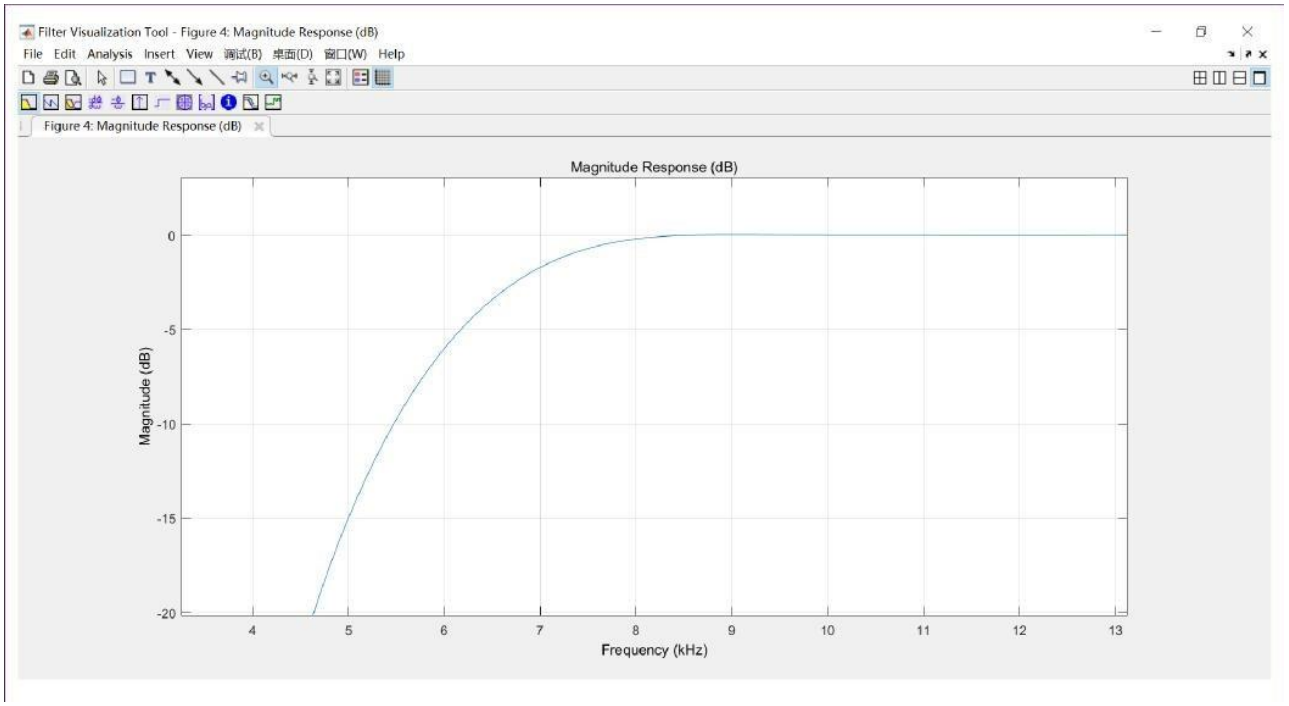

图 7. FIR 滤波器的幅值响应

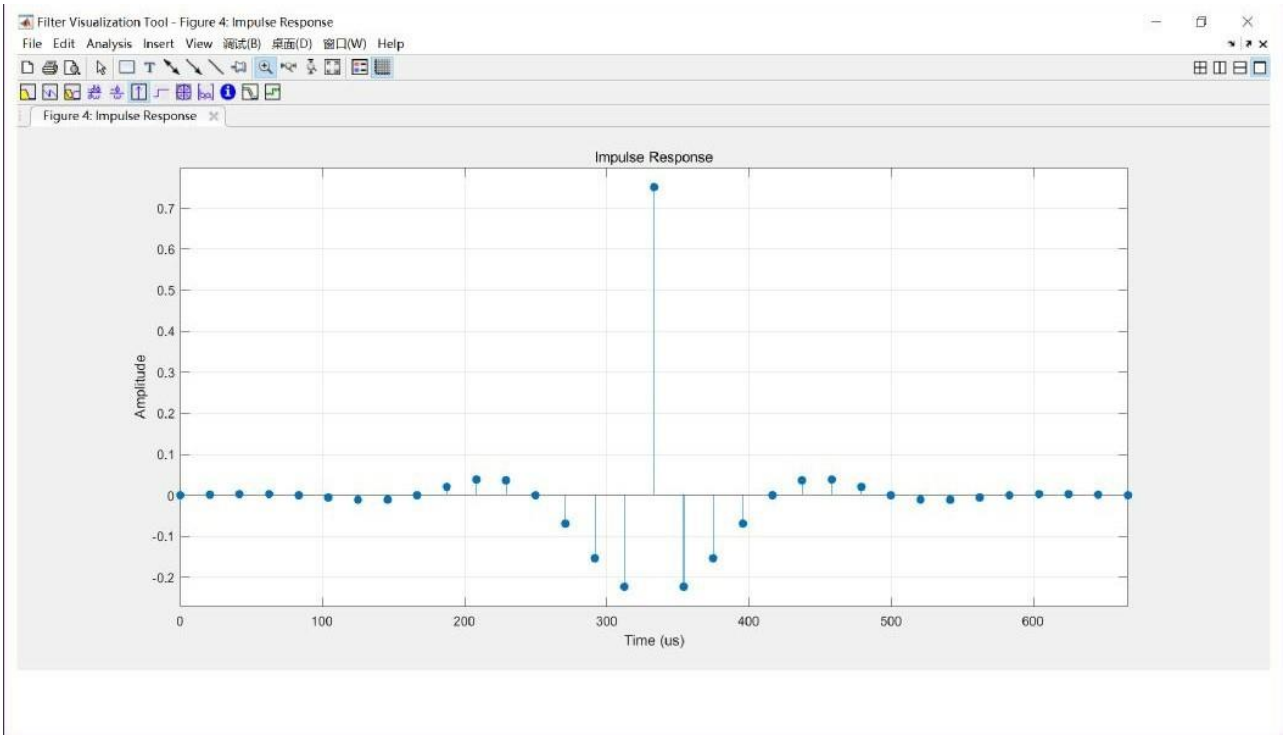

图 8. FIR 滤波器的脉冲响应

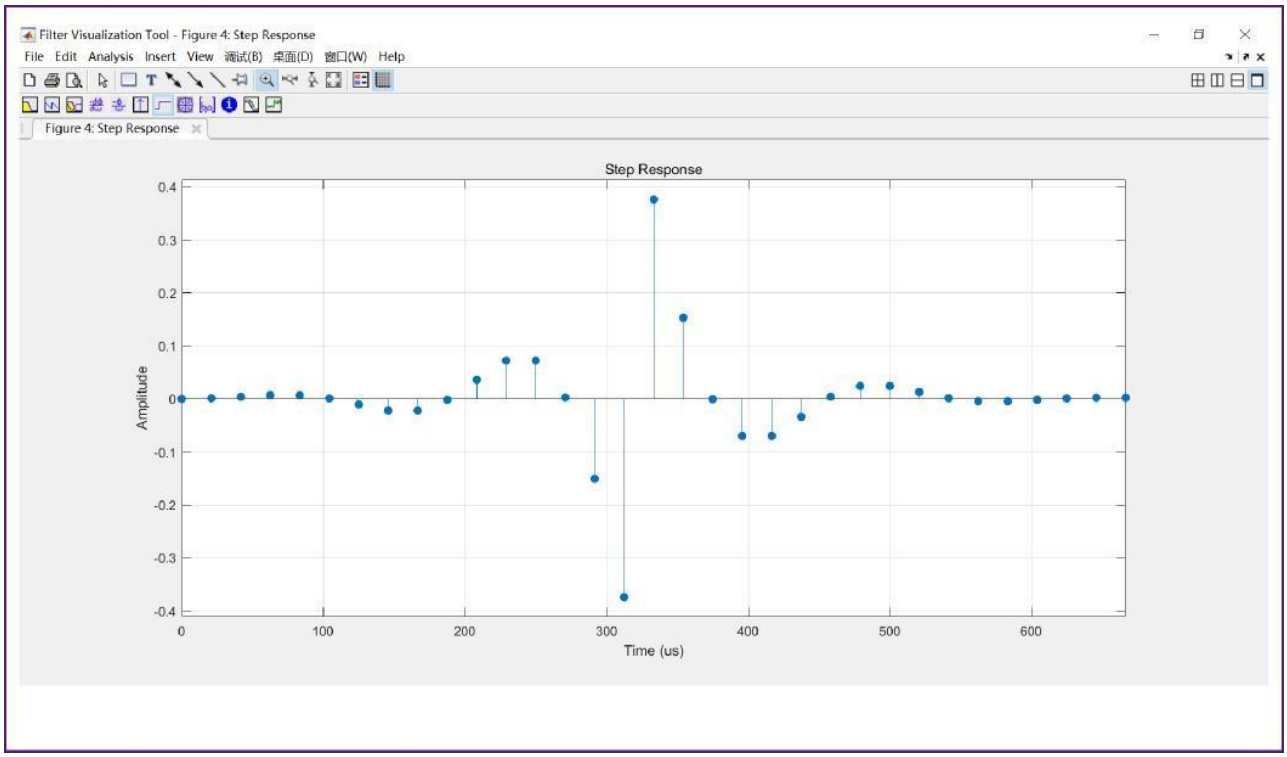

图 9. FIR 滤波器的阶跃响应

设置 PowerQuad 以在 MCU 上执行滤波过程，以高通滤波器为例。

```
void task_pq_fir_highpass(void)
{
     uint32_t i;
     uint32_t Fs=48000;

     arm_fir_instance_f32 S;
     float32_t *inputF32, *outputF32;
     uint32_t calcTime;

     inputF32 = &gPQFirF32In[0];
     outputF32 = &gPQFirF32Out[0];

     /* Generate the wave. */
     for (i = 0; i < FIR_INPUT_LEN; i++)
     {
         gPQFirF32In[i] = 1.5
                         + 0.5 * arm_sin_f32(2*PI*15000*i/Fs)
                         +    arm_sin_f32(2*PI*1000*i/Fs) ;
         gPQFirF32In[i] /= 3.0f;
      }
      // ...

      /* Call FIR init function to initialize the instance structure. */
      arm_fir_init_f32(    &S,
                               NUM_TAPS,
                               (float32_t *)&firCoeffs32_highpass[0],
                               &firStateF32[0],
                               FIR_INPUT_LEN  );
      PQ_Init(POWERQUAD_NS);
      pq_config_t pqConfig;

      pqConfig.inputAFormat = kPQ_Float;
      pqConfig.inputAPrescale = 0;
      pqConfig.inputBFormat = kPQ_Float;
      pqConfig.inputBPrescale = 0;
      pqConfig.outputFormat = kPQ_Float;
      pqConfig.outputPrescale = 0;
      pqConfig.tmpFormat = kPQ_Float;
      pqConfig.tmpPrescale = 0;
      pqConfig.machineFormat = kPQ_Float;
      pqConfig.tmpBase = (uint32_t *)0xE0000000;
      PQ_SetConfig(POWERQUAD_NS, &pqConfig);

      /* move the taps into private RAM to improve the performance of operating memory. */
      PQ_MatrixScale( POWERQUAD_NS,
                      POWERQUAD_MAKE_MATRIX_LEN(16, NUM_TAPS / 16, 0),
                      1.0,
                      firCoeffs32_highpass,
                      EXAMPLE_PRIVATE_RAM );
      PQ_WaitDone(POWERQUAD_NS);

      /* In the next calculation, data in private ram is used. */
      pqConfig.inputBFormat = kPQ_Float;
      pqConfig.outputFormat = kPQ_Float;
      PQ_SetConfig(POWERQUAD_NS, &pqConfig);

      TimerCount_Start();
      PQ_FIR(POWERQUAD_NS, inputF32, APP_PQ_FIR_SAMPLE_COUNT_240, EXAMPLE_PRIVATE_RAM, NUM_TAPS,
```

```
outputF32,PQ_FIR_FIR);
      PQ_WaitDone(POWERQUAD_NS);
      //arm_fir_f32(&S, inputF32, outputF32, FIR_INPUT_LEN);
      TimerCount_Stop(calcTime);

      /* Todo ...
      * - Record the time.
      * - Display the waveform.
      */
}
```

当运行由 PowerQuad 硬件执行的滤波器演示实例时，结果将显示在 LCD 屏上，如图 10 所示。

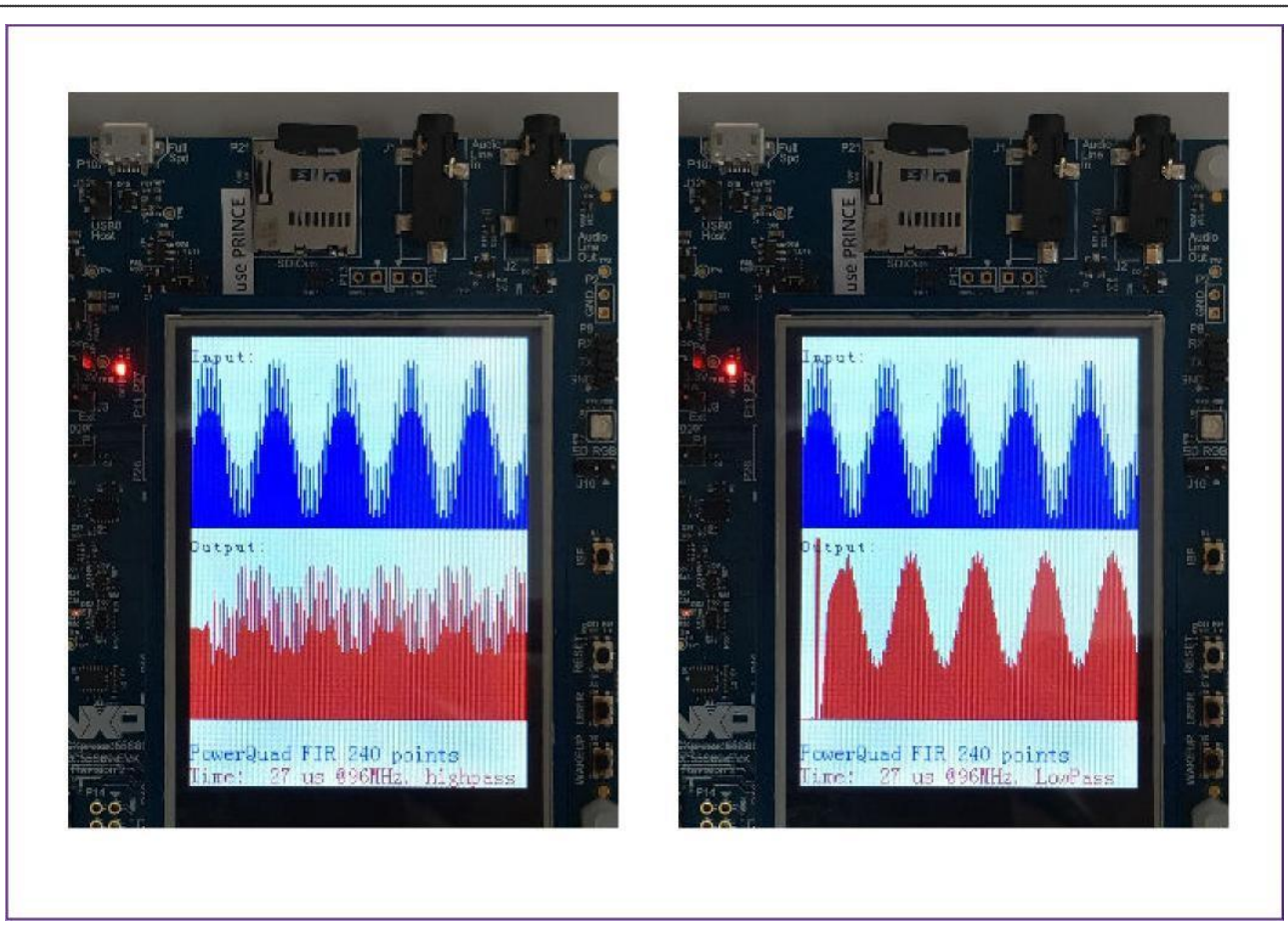

图 10. PowerQuad FIR 高通/低通滤波器

4 PowerQuad 与 Arm CMSIS-DSP 的性能对比

在演示工程中，设置了一个页面，用于在 PowerQuad 和 Arm CMSIS-DSP 运行相同任务时进行比较。为了比较的公平性，在运行 DSP 任务时，将在 RAM 中运行 Arm CMSIS-DSP 代码，而 PowerQaud 将使用专用 RAM，保证它们都达到最高性能。

图 11 显示了截屏图。

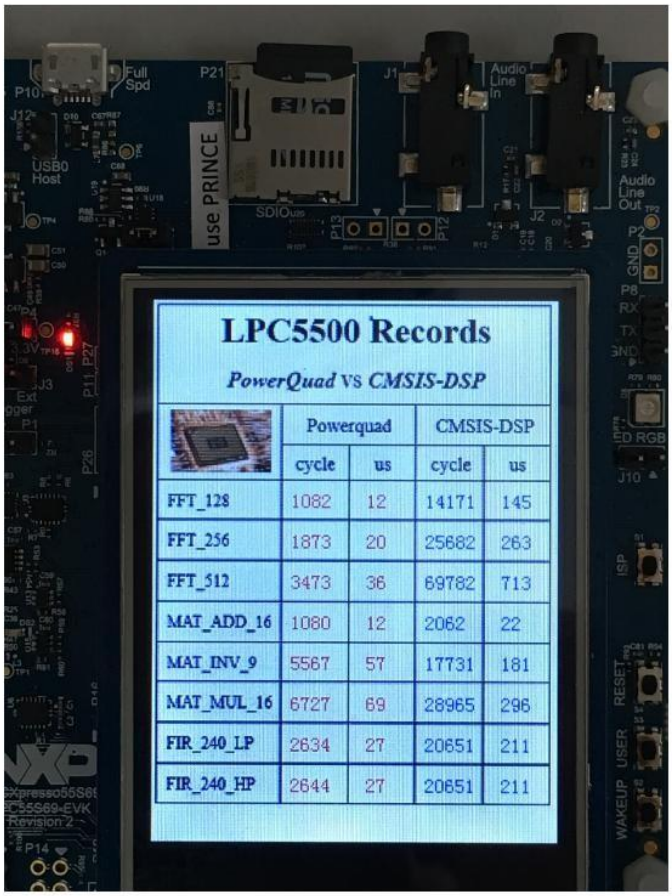

图 11. PowerQuad 与 Arm CMSIS-DSP 的运行时间对比

图 12 汇总了数据。

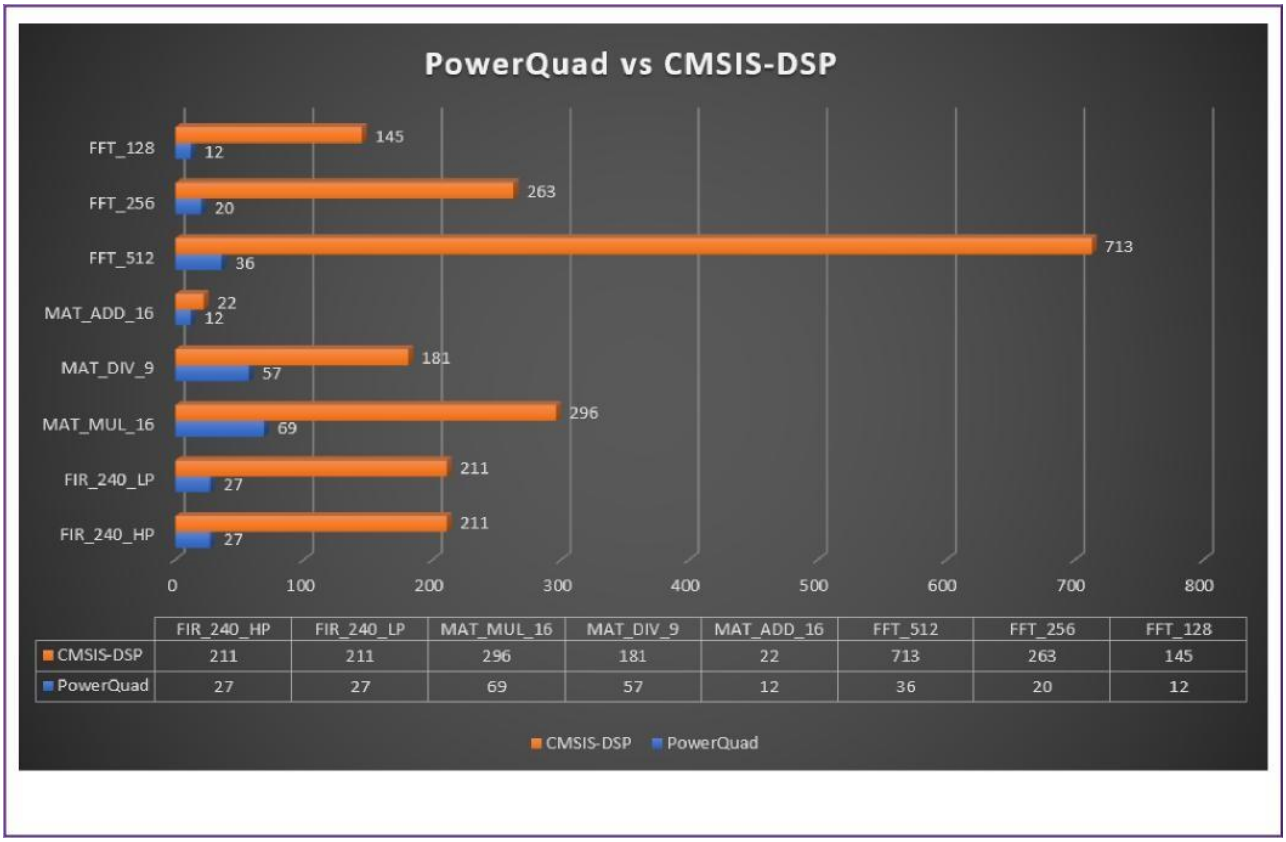

图 12. PowerQuad 与 Arm CMSIS-DSP 的运行时间对比数据表

5 修订历史

版本号	日期	说明
第 0 版	2021 年 12 月 25 日	初版发布
第 1 版	2022 年 5 月 25 日	用 LPC553x/LPC55S3x 替代 LPC55S36

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com.cn/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Airfast — is a trademark of NXP B.V.

Bluetooth — the Bluetooth wordmark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NXP Semiconductors is under license.

Cadence — the Cadence logo, and the other Cadence marks found at www.cadence.com/go/trademarks are trademarks or registered trademarks of Cadence Design Systems, Inc. All rights reserved worldwide.

CodeWarrior — is a trademark of NXP B.V.

ColdFire — is a trademark of NXP B.V.

ColdFire+ — is a trademark of NXP B.V.

EdgeLock — is a trademark of NXP B.V.

EdgeScale — is a trademark of NXP B.V.

EdgeVerse — is a trademark of NXP B.V.

eIQ — is a trademark of NXP B.V.

FeliCa — is a trademark of Sony Corporation.

Freescale — is a trademark of NXP B.V.

HITAG — is a trademark of NXP B.V.

ICODE and I-CODE — are trademarks of NXP B.V.

Immersiv3D — is a trademark of NXP B.V.

I2C-bus — logo is a trademark of NXP B.V.

Kinetis — is a trademark of NXP B.V.

Layerscape — is a trademark of NXP B.V.

Mantis — is a trademark of NXP B.V.

MIFARE — is a trademark of NXP B.V.

MOBILEGT — is a trademark of NXP B.V.

NTAG — is a trademark of NXP B.V.

Processor Expert — is a trademark of NXP B.V.

QorIQ — is a trademark of NXP B.V.

SafeAssure — is a trademark of NXP B.V.

SafeAssure — logo is a trademark of NXP B.V.

StarCore — is a trademark of NXP B.V.

Synopsys — Portions Copyright © 2021 Synopsys, Inc. Used with permission. All rights reserved.

Tower — is a trademark of NXP B.V.

UCODE — is a trademark of NXP B.V.

VortiQa — is a trademark of NXP B.V.

For more information, please visit: http://www.nxp.com.cn
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 25 May 2022
Document identifier: AN13498