Freescale Semiconductor, Inc.
应用笔记

Document Number: AN4255
Rev. 4，07/2015

面向计量应用的基于 FFT 的算法

作者：Ludek Slosarcik

1 简介

快速傅里叶变换 (FFT) 是一种数学方法，用于将信号的表示方法由时域转换为频域。在频域中，信号会转换为一系列不同振幅，不同频率的信号叠加。FFT 是一种可高效完成此过程的方法。它可以通过一短截信号计算。

FFT 是数字信号处理中最重要的主题之一。它对于频率（频谱）分析尤为重要；例如，语音识别、声音信号的数字编码 （以便在数字传输时减少数据流）、检测机械震动、信号滤波、求解偏微分方程式等。

本应用笔记介绍了如何在计量应用中使用 FFT，特别是电表中的功率和能量计算。计量应用的首要任务是准确计算能量 （也就是计费电量）。其计算必须符合国际电表标准。剩余数量的计算仅用于提供信息，它们被称为不计费电量。

内容

2 DFT 基础知识

为了正确理解下一章节，必须先弄清楚什么是离散傅里叶变换 (DFT)。DFT 是傅里叶分析中使用的一种特殊的离散变换。它将一个函数变换为另一个，这被称为原函数 （时间域中的一个函数）的频率域表示。 DFT 输入为有限序列的实数或复数，因而 DFT 非常适合处理计算机内存储的信息。DFT 与 FFT 之间的关系如下所示：DFT 是指数学变换或函数，是不涉及计算方法的；而 FFT 是指计算 DFT 的一种特定算法。

有限长 *N* 序列 DFT 定义如下：

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi nk}{N}} = \sum_{n=0}^{N-1} \left[x(n) \cdot \cos\left(\frac{2\pi nk}{N}\right) - j \cdot x(n) \cdot \sin\left(\frac{2\pi nk}{N}\right) \right] \qquad \textit{公式 1}$$

$$0 \le k < N$$

其中：

- *X (k)* 为变换后的输出
- *x (n)* 为变换前的输入 （即采样的输入信号）
- *j* 为虚数单位

公式 1 中的每一项定义复数格式中的部分正弦项，其中频率为 kF_0、相位为 $(2\pi nk/N)$、幅度为 *x (n)*。*n*=0,1,...,*N*-1（请参见公式 1）和选定 *k* 项的向量求和代表 kF_0 频率的复数格式频谱 *X(k)* 的总正弦项。请注意，F_0 为输入周期信号频率。如果是非周期信号，则 F_0 表示该信号用于 DFT 计算所选择的基本周期。

逆离散傅里叶变换 (IDFT) 由下式得出：

$$x(n) = \frac{1}{N} \cdot \sum_{k=0}^{N-1} X(k) \cdot e^{j\frac{2\pi nk}{N}} \qquad \textit{公式 2}$$

$$0 \le n < N$$

借助于公式 2，可以通过 *X (k)* 的频谱项回溯计算 *x (n)* 的离散值。

在这两个方程式中，*X (k)* 和 *x (n)* 都可以为复数，因此，如果我们直接使用公式 1，则需要使用 *N* 次复数乘法和 (*N*-1) 次复数加法来计算每个 DFT 值。计算频率成分的所有 *N* 值需要使用 N^2 次复数乘法和 *N*·(*N*-1) 次复数加法。

3 FFT 的实现

对于 2 节， “DFT 基础知识 ” 中导出的方程式，最好先引入下面的替代式：

$$W_N^{nk} = e^{-j\frac{2\pi nk}{N}} \quad \textbf{公式 3}$$

此替代式中的 W_N^{nk} 项也称为 “ 旋转因子 ”。通过此替代式，我们可以将计算 DFT 和 IDFT 的方程式改写为这些格式：

$$DFT[x(n)] = X(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{nk} \quad \textbf{公式 4}$$

$$IDFT[X(k)] = x(n) = \frac{1}{N} \cdot \sum_{k=0}^{N-1} X(k) \cdot W_N^{-nk} \quad \textbf{公式 5}$$

为了提高计算 DFT 的效率，可以充分利用 W_N^{nk} 的部分属性。如下所示：

对称属性：

$$W_N^{nk+N/2} = -W_N^{nk} \quad \textbf{公式 6}$$

周期属性：

$$W_N^{nk} = W_N^{nk+N} = W_N^{nk+2N} = \ldots \quad \textbf{公式 7}$$

递归属性：

$$W_{N/2}^{nk} = W_N^{2nk} \quad \textbf{公式 8}$$

这些属性源自每个 *nk* 值的旋转向量带来的旋转因子 （公式 4）的图形表示。

3.1 radix-2 基二时间抽取 FFT 说明

FFT 的基本思想是将时间域序列长度 *N* 的 DFT 分解成几个较短的连续 DFT，其计算需要的算数运算较少。这就是所谓的分治策略，可以通过使用上一章节中所述的属性实现。分解成较短 DFT 的执行方法如下：分别根据 *x (n)* 的偶数和奇数样本，将 *N* 点输入数据序列 *x (n)* 拆分成两个 *N*/2 点数据序列 *a (m)* 和 *b (m)*，即：

- *a (m) = x (2m)*，即， *x (n)* 样本 *(n = 2m)*
- *b (m) = x (2m + 1)*，即， *x (n)* 样本 *(n = 2m + 1)*

其中， *m* 为 $0 \le m < N/2$ 范围内的整数。

将时间域序列拆分成奇数和偶数样本的过程的算法被称作时间抽取算法 (DIT)。因此， *a (m)* and *b (m)* 可通过将 *x (n)* 以因数 2 抽取获得，因此，获得的 FFT 算法也称为 “ 基 -2”。它是 Cooley- Tukey 算法 [1] 最简单、最常见的形式。

现在，通过抽取序列的 DFT 的方式， *N* 点的 DFT （请参见公式 1）可以以如下方式表示：

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{nk} = \sum_{m=0}^{N/2-1} x(2m) \cdot W_N^{2mk} + \sum_{m=0}^{N/2-1} x(2m+1) \cdot W_N^{(2m+1)k} =$$

公式 9

$$= \sum_{m=0}^{N/2-1} x(2m) \cdot W_N^{2mk} + W_N^{k} \sum_{m=0}^{N/2-1} x(2m+1) \cdot W_N^{2mk}$$

借助公式 8 的替代式， 公式 9 可表示为：

$$X(k) = \sum_{m=0}^{N/2-1} a(m) \cdot W_{N/2}^{mk} + W_N^{k} \sum_{m=0}^{N/2-1} b(m) \cdot W_{N/2}^{mk} = A(k) + W_N^{k} B(k)$$

公式 10

$$0 \le k < N$$

这两个求和分别表示序列 *a (m)* 和 *b (m)* 的 *N*/2 点 DFT。因此，偶数样本的 DFT [*a (m)*] = *A(k)*，奇数样本的 DFT [*b (m)*] = *B (k)*。利用 DFT (公式 7) 的周期属性，长度 *N*/2 的 DFT 对于 $N/2 \le k < N$ 的输出等于 $0 \le k < N/2$ 的输出。也就是说， $A(k+N/2) = A(k)$ ， $B(k+N/2) = B(k)$ （对于 $0 \le k < N/2$）。此外，利用对称属性 (公式 6)，因数 $W_N^{k+N/2} = -W_N^{k}$。因此，整个 DFT 计算如下：

$$X(k) = A(k) + W_N^{k} B(k)$$

$$X(k + N/2) = A(k) - W_N^{k} B(k)$$

公式 11

$$0 \le k < N/2$$

此结果（以两个 *N*/2 的 DFT 递归表示长度 *N* 的 DFT）为基 -2 DIT FFT 的核心。请注意， *X (k)* 的最终输出可通过 +/– 组合 *A (k)* 和 $B(k)W_N^{k}$（简单地说为大小为 2 的 DFT）获得。这些组合可以通过简单图形表示，有时称为 “ 蝶形 ”（请参见图 1）。

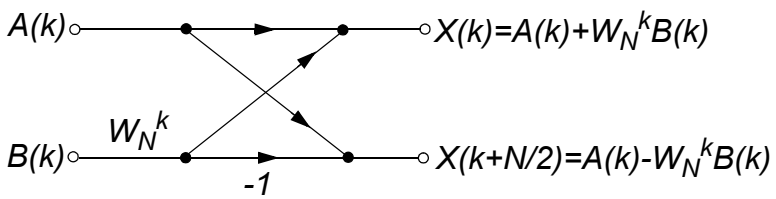

图 1. DIT FFT 算法中的基本蝶形计算

将 *N* 点 DFT 离散序列计算为两个 *N*/2 点 DFT 的过程可用于通过 *N*/4 点 DFT 项计算 *N*/2 点 DFT 序列。为此，应将每个 *N*/2 点序列分成奇数和偶数项两个子序列，然后再连续计算其 DFT。可以反复执行数据序列抽取，直到获得的序列降至一个基本 DFT。

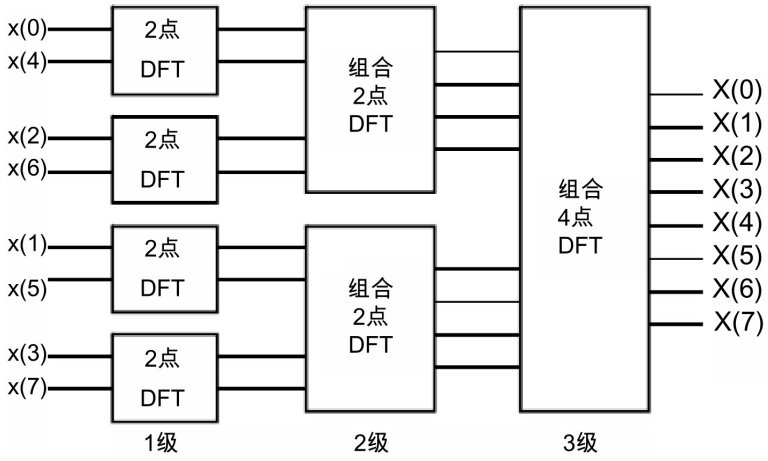

图 2. 分解 8 点 DFT

为了便于说明，图2描述了 $N=8$ 点DFT的计算过程。我们注意到此计算分三个阶段进行 ($3=\log_2 8$)，从计算四个 2 点 DFT 开始，然后是两个 4 点 DFT，最后是一个 8 点 DFT。一般来说，对于 *N* 点 FFT， FFT 算法会将 DFT 分解为 $\log_2 N$ 阶，每个包含 *N*/2 个蝶形计算。较短 DFT 组合构成较大 DFT (*N*=8) 如图 3 所示。

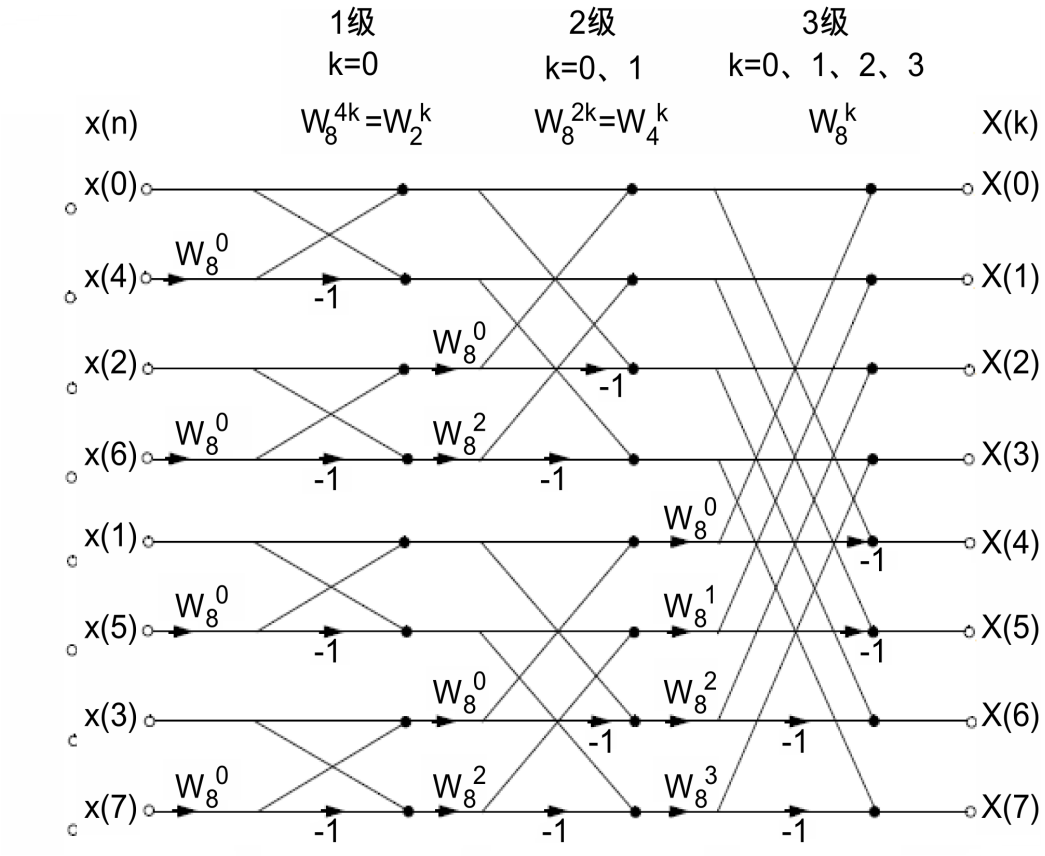

图 3. 8 点基 2 DIT FFT 算法数据流

每个点代表一个复数加法，每个箭头代表一个复数乘法，如图 3 所示。图 3 中的 W_N^k 因子可表示为二的幂 (W_2)（第一阶）、四的幂 (W_4)（第二阶）和八的幂 (W_8)（第三阶），以此类推。它还可以统一表示为 N (W_N) *的幂，*其中 N 为输入序列 x *(n) 的大小*。两个表达式之间的关系如公式 8 所示。

3.2 基二时间抽取 FFT 要求

为了高效、最优地将输入数据序列分解为奇数和偶数子序列，最好具有二的幂个输入数据样本（...、 64、 128 等）。

计算基 -2 FFT 之前先将输入数据序列重新排序（另请参见图 2 和图 3 的左侧）。这意味着此算法需要对数据进行位反转排序；也就是说，MSB 变为 LSB，反之亦然。表 1 所示为 8 点输入序列的位反转示例。

表 1. 按照 8 点输入序列进行位反转

十进制数	0	1	2	3	4	5	6	7
等效二进制数	000	001	010	011	100	101	110	111
位反转，二进制	000	100	010	110	001	101	011	111
等效十进制数	0	4	2	6	1	5	3	7

我们必须注意到，此算法为“同址计算”类型，这意味着计算中的每个蝶形吞吐量输出可置于从其中提取输入的相同内存位置，从而使“同址”算法无需占用其他内存即可执行 FFT。

3.2.1 窗口选择

FFT 计算假定每个数据块中的信号是周期性的；也就是说，它反复出现。大多数信号并非周期性的，即便为周期性的，但也可能具有未知周期。计算非周期性信号的 FFT 时，所得的频谱可能存在泄漏。为了解决这一问题，最好采用 *N* 个输入信号样本并使其具有一定的周期。这通常可以通过窗口功能（Barlett、Blackman、Kaiser-Bessel 等）实现。考虑到所得的频谱形状可能在应用窗口功能之后稍有不同（与未应用窗口功能的纯周期性信号频谱相比），最好也不要在计量应用中使用特殊的窗口功能，或者使用简单的矩形窗口（一个相干增益为 1.0 的函数）。这需要输入信号的频率为已知。在计量应用中，这可通过测量线路电压周期来实现。

可以利用过零检测 (ZCD) 技术来执行信号（市电）周期检测。过零点是一个瞬时点，在该点不存在电压（请参见图 4a）。在线路电压波或其他简单波形中，这在一个周期内通常发生两次。过零点计数是一种用于测量输入信号（线路电压）频率的方法。例如，ZCD 电路可以使用 MCU 中的模拟比较器实现，在此比较器中，第一条通道连接至参考电压，第二条通道通过简单的分压器连接至线路。最后，此比较器中逻辑电平的变化将被软件解读为市电过零。过零点之间的时间可利用软件中的计时器测量。过零点还定义了简单矩形 FFT 窗口（图 4a）的开始和结束点。从技术角度来说，没必要测量过零点的输入信号频率，但可使用任何其他两个输入信号点，这两个点可能会被简单地视为 – 峰值点，例如（参见图 4b），– 具有相似的结果（幅度相同并且统一移相）。

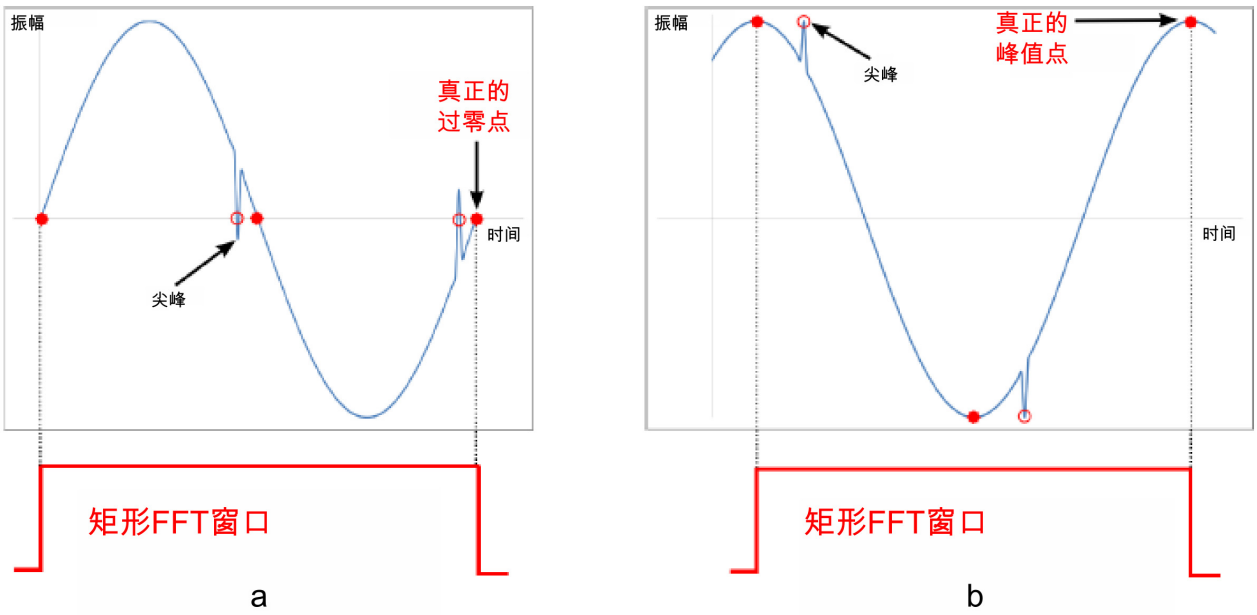

图 4. 过零点与峰值点检测

此外，还必须了解用于测量信号频率的软件技术必须包含某种用于消除可能电压尖峰的复杂算法（请参见图 4）。这些尖峰可能作为负载（电机、接触器等）干扰出现在线路中，并且可能导致出现错误的过零或峰值检测。

在实际运用中，最好测量几个真实过零或峰值点之间的时间。最后，必须执行算术平均以计算正确的信号频率。然后对每个输入信号（电压和电流）的周期进行频率采样，它比测得的线路电压频率高 *N* 倍，其中 *N* 为样本数。如果采样频率与此不同，则所得的频谱可能存在泄漏。

3.3 基二时间抽取 FFT 结论

基 -2 FFT 利用有效的算法完成与 DFT 相同的任务，但所花费的时间更短。其中 DFT 需要 N^2 复数乘法（请参见 2 节，“DFT 基础知识 ”），FFT 仅进行 $N/2 \cdot \log_2 N$ 复数乘法和 $N \cdot \log_2 N$ 复数加法。因此，对于相同 *N*，DFT 计算与 FFT 计算之间的比率与 $2N / \log_2 N$ 成比例。如果 *N* 较小，则此比率不是很明显，但当 *N* 变大时，该比率也会变得非常大。因此， FFT 是一种计算 DFT 的更快捷方式。

基 -2 FFT 算法通常定义为基 -*r* FFT 算法，其中，*N* 点输入序列被拆分为 *r* 个子序列，以便使计算更加高效，例如基 -4 或基 -8。因此，基数为 FFT 分解的大小。

同样地， DIT 算法有时也用于频率抽取 (DIF) 算法 （也被称为 Sande-Tukey 算法），它将 DFT 系数 *X (k)* 序列分解成若干个连续的较小子序列。但是，本应用笔记仅介绍基 -2 DIT FFT 算法。

4 使用 FFT 进行功率计算

4.1 直角坐标与极坐标形式之间的转换

功率计中实现 FFT 需要复数计算，因为先前章节中介绍 DFT 或 FFT 的数学公式均假定这些公式（在图形格式中，为图 3 中 *X(k)*）中的每一项包含一个复数。

复数是指由实数和虚数部分组成的数。在二维直角坐标系 （复数平面）中，该数可以表示为点或位置向量。该数的实数部分一般被绘制为水平分量，而虚数部分则被绘制为垂直分量 （请参见图 5）。

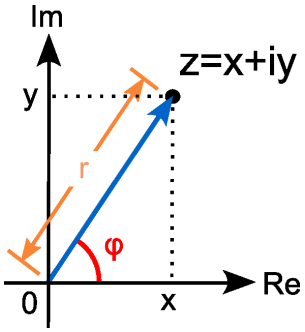

图 5. 复数的图形表示

除了使用 *x*- 和 *y*- 坐标之外，在复数平面中对点进行编码的另一种方法是使用点 *z* 至 *O* 之间的距离、坐标为 (0,0) 的点和 *z* 与 *O* 之间的线段夹角。于是产生了复数的极坐标形式。复数 *z=x+iy* 的绝对值 （或幅度）为：

$$r = |z| = \sqrt{x^2 + y^2} \qquad \textit{公式 12}$$

z 的参数或相位定义如下：

$$\varphi = \arg(z) = \operatorname{atan}\left(\frac{y}{x}\right) \qquad \textit{公式 13}$$

r 和 *φ* 一起组成了复数的另一种表示方式 - 极坐标形式，该方式结合使用模数和参数，可完全指定点在平面上的位置。

4.2 均方根计算

在电气设计中，均方根 (RMS) 是测量 AC 信号幅度的一种基本方法。分配给 AC 信号的 RMS 值为在相同负载中产生同等热量所需的 DC 量。

在复数平面中，电流 (I_{RMS}) 和电压 (U_{RMS}) 的 RMS 值与每个谐波关联的幅度求和（参见图 5 中的 r）相同。对于公式 12，频率域中的电流和电压总 RMS 值定义如下：

$$I_{RMS} = \sqrt{\sum_{k=0}^{\frac{N}{2}-1} (I_{RE}^2(k) + I_{IM}^2(k))} \quad \text{公式 14}$$

$$U_{RMS} = \sqrt{\sum_{k=1}^{\frac{N}{2}-1} (U_{RE}^2(k) + U_{IM}^2(k))} \quad \text{公式 15}$$

其中：

- $I_{RE}(k)$、$U_{RE}(k)$ 为电流和电压 k^{th} 谐波的实数部分
- $I_{IM}(k)$、$U_{IM}(k)$ 为电流和电压 k^{th} 谐波的虚数部分

附注

电压偏移量（零谐波）未包括在 U_{RMS} 计算中。由于在市电中为零偏移量，因此可使用此简化方法。

4.3 复数功率计算

AC 功率流具有三个组成部分：实际或真实测得的功率 (P)(W)、测得的视在功率 (S)(VA) 和测得的无功功率 (Q)(VAr)。这三种类型的功率（– 有功、无功和视在 –）以三角形式相互关联。这称之为功率三角形 （请参见图 6）。

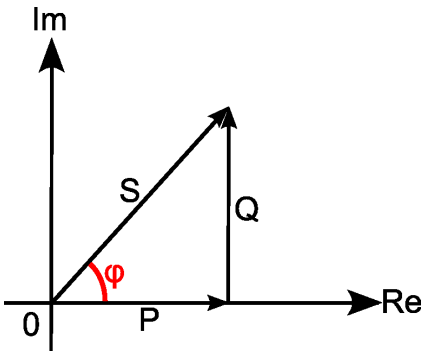

图 6. 功率三角形

此图中的夹角 φ 为电压相对于电流的相位。复数功率定义如下：

$$\bar{S} = P + jQ = \bar{U} \cdot \overline{I^*} \quad \text{公式 16}$$

其中 $\overline{U}$ 为电压向量 ($\overline{U} = U_{RE} + jU_{IM}$)，$\overline{I^*}$ 为共轭复数电流向量 ($\overline{I^*} = I_{RE} - jI_{IM}$)，两者分别针对每个谐波。

对于公式 12，复数功率的长度 (|S|) 实际上为视在功率 (VA)。就电流和电压相量 （FFT 输出）和公式 16 而言，直角坐标形式中的复数功率可最终表示为：

$$\bar{S} = P + jQ = \bar{U} \cdot \overline{I^*} = \sum_{k=1}^{\frac{N}{2}-1} (U_{RE}(k) + jU_{IM}(k)) \cdot (I_{RE}(k) - jI_{IM}(k)) =$$

$$= \sum_{k=1}^{\frac{N}{2}-1} (\underbrace{(I_{RE}(k) \cdot U_{RE}(k) + I_{IM}(k) \cdot U_{IM}(k))}_{\text{复数功率的实数部分 = 有功功率 (P)}} + \underbrace{j(U_{IM}(k) \cdot I_{RE}(k) - U_{RE}(k) \cdot I_{IM}(k))}_{\text{复数功率的虚数部分 = 无功功率 (Q)}}) \quad \textit{公式 17}$$

其中：

- $I_{RE}(k)$、 $U_{RE}(k)$ 为电流和电压 k^{th} 谐波的实数部分
- $I_{IM}(k)$、 $U_{IM}(k)$ 为电流和电压 k^{th} 谐波的虚数部分

就公式 12 和公式 13 而言，整个实数功率的两个部分 （P 和 Q）在极坐标中还可以表示为：

$$\bar{S} = \sum_{k=1}^{\frac{N}{2}-1} (\underbrace{(|I(k)| \cdot |U(k)| \cdot \cos(U_{\varphi}(k) - I_{\varphi}(k)))}_{\text{复数功率的实数部分 = 有功功率 (P)}} + j \underbrace{(|I(k)| \cdot |U(k)| \cdot \sin(U_{\varphi}(k) - I_{\varphi}(k)))}_{\text{复数功率的虚数部分 = 无功功率 (Q)}}) \quad \textit{公式 18}$$

其中：

- $|I(k)|$、 $|U(k)|$ 为电流和电压 k^{th} 谐波的幅度
- $I\varphi(k)$、 $U\varphi(k)$ 为电流和电压 k^{th} 谐波的相移 （对于 FFT 窗口圆点）

请注意，这些方程式中的输入为电流和电压的傅里叶项 （在直角坐标或极坐标形式中）。有关这些项的图形表示，请参见图 3 中的 $X(k)$。

先前公式中使用了两种基本的简化：

- 由于 FFT 频谱的对称性，仅 $N/2$ 个项用于复数功率计算。
- 预期市电中的电压无 DC 偏移量。因此，两个公式中均缺少 *0* 谐波，因为电流值（$I_{RE}(0)$、$I_{IM}(0)$、$|I(0)|$）均乘以零。

复数功率的幅度 (|S|) 为视在功率 (VA)。在无更高谐波的纯正弦系统中，视在功率计算将得出正确的结果。遇到系统中的谐波之后，视在功率计算的准确性将会受到影响。在此情况下，最好使用总视在功率 (VA)。总视在功率定义为电压和电流 RMS 值的乘积：

$$S_{tot} = U_{RMS} \cdot I_{RMS}$$ 公式 19

其中：

- U_{RMS} 为线路电压 [V] 的 RMS 值
- I_{RMS} 为线路电流 [A] 的 RMS 值

4.4 电能计算

可以累积每个时间单元（通常为每个小时）内的功率，通过功率（有功、无功）计算有功和无功电能。计算公式表示如下：

$$\Delta Energy = \frac{Power}{(Frequency \cdot 3600)}$$ 公式 20

其中：

- *Δ 电能*为每个计算周期的有功或无功电能增量 [Wh/VARh]
- *功率*为一个周期内测得的瞬时有功或无功功率 [W/VAR]
- *频率*为线路频率 [Hz]
- 3600 为“ 小时 ”系数

4.5 功率系数计算

在电气设计中， AC 电功率系统的功率因数定义为流入负载的实际（有功）功率与电路中视在功率的比率，它是一个无量纲数，范围在 –1 与 1 之间。

$$PF = \frac{P}{S}$$ 公式 21

其中：

- *P* 为瞬时有功（实际）功率 [W]
- *S* 为瞬时视在功率 [VA]

实际功率为在特定时间内执行工作的电路电容。视在功率为电路的电流和电压的乘积。由于能量存储于负载中并返回至源，或者由于非线性负载使从源中得到的电流波形失真，视在功率将大于实际功率。当设备（通常为负载）产生功率，然后回流至设备（通常被视为发生器）时，将会出现负功率因数（请参见表 2）。

表 2. 功率因数范围与能流方向

象限	功率因数范围	功率符号	负载模式	I 至 U 相移
I	0...1	+P、 +Q	带电感负载的电机模式	滞后电流
II	-1...0	-P、 +Q	电感作用发生器模式	超前电流
III	-1...0	-P、 -Q	电容作用发生器模式	滞后电流
IV	0...1	+P、 -Q	带电容负载的电机模式	超前电流

4.6 总谐波失真计算

在电气设计中，总谐波失真 (THD) 是指一组较高谐波频率的 RMS 幅度与第一个谐波（或基本频率）的 RMS 幅度之间的比率。因此， THD 是一个信号失真指标。对于电压和电流信号，使用频率分量的 THD 计算公式如下所示：

$$THD_U = \sqrt{\sum_{k=2}^{\frac{N}{2}-1} \frac{(U_{RE}^2(k) + U_{IM}^2(k))}{(U_{RE}^2(1) + U_{IM}^2(1))}} \cdot 100 \qquad \text{公式 22}$$

$$THD_I = \sqrt{\sum_{k=2}^{\frac{N}{2}-1} \frac{(I_{RE}^2(k) + I_{IM}^2(k))}{(I_{RE}^2(1) + I_{IM}^2(1))}} \cdot 100 \qquad \text{公式 23}$$

其中：

- $I_{RE}(k)$、 $U_{RE}(k)$ 为电流和电压 k^{th} 谐波的实数部分
- $I_{IM}(k)$、 $U_{IM}(k)$ 为电流和电压 k^{th} 谐波的虚数部分

先前方程式的最终结果为百分比。百分比越高，则信号失真越高。

5 计量库

本节介绍基于 FFT 计量算法的计量库实现方法。本应用笔记随计量库和测试应用一起提供。该库包含若干适合最常用功率计拓扑的独特应用编程接口 (API) 的函数；即，单相、双相 (Form-12S) 和三相。该库以对象格式 （*.*a* 和 / 或 *.*lib* 文件）提供，测试应用以 C 源代码提供。函数原型以及内部数据结构在 *meterlibfft.h* 头文件中声明。典型单相功率计应用的整个 FFT 计算过程的简单框图如图 7 所示。关于此计算过程在实际功率计中的具体实现方法，请参见 7 节， “ 参考文献 ” 中的 Freescale 参考设计 [4]、 [5] 或 [6]。

附注

IAR Embedded Workbench® for ARM® （版本 7.40.1）工具用于获取所有库函数的性能数据 （请参见以下性能表）。代码经过编译，执行速度针对 MKM34Z128 目标 （ARM Cortex®-M0+ 内核）和 MKM34Z256 目标 （带 MMAU 的 ARM Cortex-M0+ 内核）进行了全面优化。设备由外部 32.768 kHz 晶振驱动，利用 FFL 外部启用 (FEE) 模式下工作的锁频环 (FLL) 模块提供 24 MHz 时钟。测得的执行时间将重新计算到内核时钟周期中。

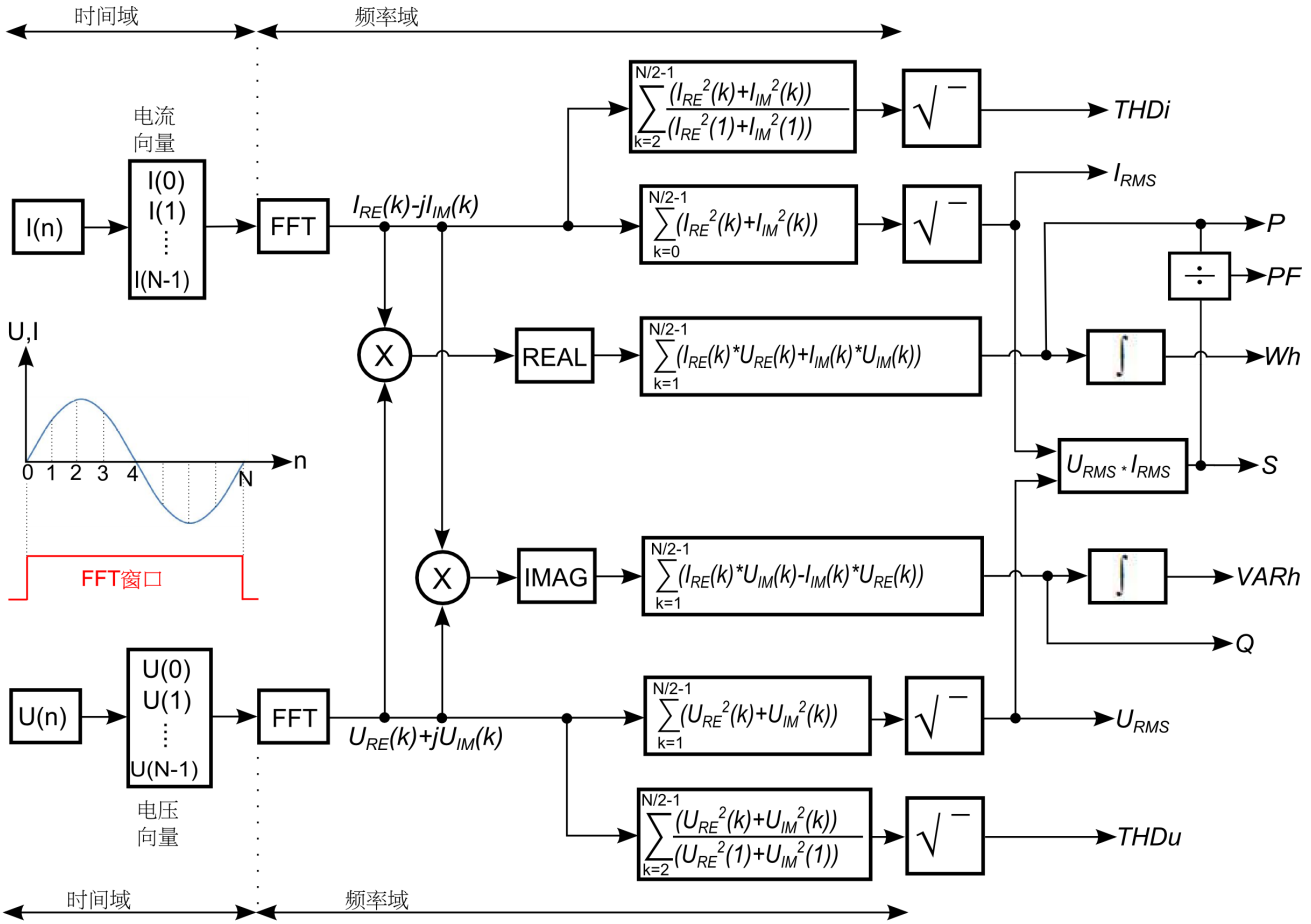

图 7. 基于 FFT 的单相功率计计算流程框图

5.1 内核架构和编译器支持

这款基于 FFT 的高性能计量库支持 ARM Cortex-M0+ 和 Cortex-M4 内核。除了标准内核之外，该库还支持内存映射运算单元 (MMAU)，它是 Freescale 设计的一款用于提高特定计量算法执行速度的硬件数学模块。

计量库的默认安装文件夹为 *C:\Freescale\METERLIBFFT_R4_0_0*。表 3 列出了所需的所有头文件、库文件及其位置 （相对默认安装文件夹）。将这些文件夹和路径添加到您的项目工作空间，便可将此计量库成功集成到应用中。

表 3. 基于 FFT 的计量库集成

<table>
<tr><th colspan="3" rowspan="2">包括文件和库</th><th colspan="3">METERLIBFFT</th></tr>
<tr><th>Cortex-M0+ w/o MMAU</th><th>Cortex-M0+ w/ MMAU</th><th>Cortex-M4</th></tr>
<tr><td rowspan="7">包括</td><td rowspan="3">文件</td><td>iar</td><td colspan="3" rowspan="3">fraclib.h
meterlibfft.h</td></tr>
<tr><td>armcc</td></tr>
<tr><td>gcc</td></tr>
<tr><td rowspan="3">路径</td><td>iar</td><td rowspan="3">..\lib\fraclib\inc
..\lib\fraclib\inc\cm0p
..\lib\meterlibFFT\inc</td><td>..\lib\fraclib\inc
..\lib\fraclib\inc\cm0p_mmau
..\lib\fraclib\inc\cm0p_mmau\iar
..\lib\meterlibFFT\inc</td><td rowspan="3">..\lib\fraclib\inc
..\lib\fraclib\inc\cm4
..\lib\meterlibFFT\inc</td></tr>
<tr><td>armcc</td><td>..\lib\fraclib\inc
..\lib\fraclib\inc\cm0p_mmau
..\lib\fraclib\inc\cm0p_mmau\armcc
..\lib\meterlibFFT\inc</td></tr>
<tr><td>gcc</td><td>..\lib\fraclib\inc
..\lib\fraclib\inc\cm0p_mmau
..\lib\fraclib\inc\cm0p_mmau\gcc
..\lib\meterlibFFT\inc</td></tr>
<tr><td rowspan="6">库</td><td rowspan="3">文件</td><td>iar</td><td>fraclib_cm0p_iar.a
meterlibFFT_cm0p_iar.a</td><td>fraclib_cm0p_mmau_iar.a
meterlibFFT_cm0p_mmau_iar.a</td><td>fraclib_cm4_iar.a
meterlibFFT_cm4_iar.a</td></tr>
<tr><td>armcc</td><td>fraclib_cm0p_armcc.lib
meterlibFFT_cm0p_armcc.lib</td><td>fraclib_cm0p_mmau_armcc.lib
meterlibFFT_cm0p_mmau_armcc.lib</td><td>fraclib_cm4_armcc.lib
meterlibFFT_cm4_armcc.lib</td></tr>
<tr><td>gcc</td><td>fraclib_cm0p_gcc.a
meterlibFFT_cm0p_gcc.a</td><td>fraclib_cm0p_mmau_gcc.a
meterlibFFT_cm0p_mmau_gcc.a</td><td>fraclib_cm4_gcc.a
meterlibFFT_cm4_gcc.a</td></tr>
<tr><td rowspan="3">路径</td><td>iar</td><td colspan="3" rowspan="3">..\lib\fraclib
..\lib\meterlibFFT</td></tr>
<tr><td>armcc</td></tr>
<tr><td>gcc</td></tr>
</table>

5.2 函数 API 汇总

以下子章节介绍基于FFT计量库中定义的函数API。所有函数和内部数据结构的原型如*meterlibfft.h*头文件所声明。

5.2.1 单相功率计量 API

- void **METERLIBFFT1PH_CalcMain** (tMETERLIBFFT1PH_DATA *p)
  FFT 计算和信号调节处理函数
- long **METERLIBFFT1PH_CalcVarHours** (tMETERLIBFFT1PH_DATA *p, unsigned long *varh_i, unsigned long *varh_e, unsigned long frequency)
  无功功率计算函数
- long **METERLIBFFT1PH_CalcWattHours** (tMETERLIBFFT1PH_DATA *p, unsigned long *wh_i, unsigned long *wh_e, unsigned long frequency)
  有功功率计算函数

- void **METERLIBFFT1PH_GetAvrgValues** (tMETERLIBFFT1PH_DATA *p, double *urms, double *irms, double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

  不计费的(U、I、P、Q、S、PF、THD) 平均变量读取函数
- void **METERLIBFFT1PH_GetInstValues** (tMETERLIBFFT1PH_DATA *p, double *urms, double *irms, double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

  不计费的(U、I、P、Q、S、PF、THD) 瞬时变量读取函数
- void **METERLIBFFT1PH_GetMagnitudes** (tMETERLIBFFT1PH_DATA *p, unsigned long magn_fft)

  谐波幅度计算函数
- void **METERLIBFFT1PH_GetPhases** (tMETERLIBFFT1PH_DATA *p, unsigned long ph_fft)

  谐波相移计算函数
- void **METERLIBFFT1PH_InitAuxBuff** (tMETERLIBFFT1PH_DATA *p, Frac24 *mag_u, Frac24 *mag_i, long *ph_u, long *ph_i)

  辅助缓冲区初始化函数
- void **METERLIBFFT1PH_InitMainBuff** (tMETERLIBFFT1PH_DATA *p, Frac24 *u_re, Frac24 *i_re, Frac24 *u_im, Frac24 *i_im, long *shift)

  主缓冲区初始化函数
- long **METERLIBFFT1PH_InitParam** (tMETERLIBFFT1PH_DATA *p, unsigned long samples, unsigned long sensor, unsigned long kwh_cnt, unsigned long kvarh_cnt, unsigned long en_res)

  参数初始化函数
- long **METERLIBFFT1PH_Interpolation** (tMETERLIBFFT1PH_DATA *p, unsigned long u_ord, unsigned long i_ord, unsigned long samples_inp)

  插值函数
- long **METERLIBFFT1PH_SetCalibCoeff** (tMETERLIBFFT1PH_DATA *p, double u_max, double i_max, Frac24 *i_offs, double p_offs, double q_offs)

  设置校准系数函数

5.2.2 双相功率计量 API

- void **METERLIBFFT2PH_CalcMain** (tMETERLIBFFT2PH_DATA *p)

  FFT 计算和信号调节处理函数
- long **METERLIBFFT2PH_CalcVarHours** (tMETERLIBFFT2PH_DATA *p, unsigned long *varh_i, unsigned long *varh_e, unsigned long frequency)

  无功功率计算函数
- long **METERLIBFFT2PH_CalcWattHours** (tMETERLIBFFT2PH_DATA *p, unsigned long *wh_i, unsigned long *wh_e, unsigned long frequency)

  有功功率计算函数

- void **METERLIBFFT2PH_GetAvrgValuesPh1** (tMETERLIBFFT2PH_DATA *p, double *urms, double *irms, double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

  不计费的(U、I、P、Q、S、PF、THD) 平均变量相 1 读取函数
- void **METERLIBFFT2PH_GetAvrgValuesPh2** (tMETERLIBFFT2PH_DATA *p, double *urms, double *irms, double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

  不计费的(U、I、P、Q、S、PF、THD) 平均变量相 2 读取函数
- void **METERLIBFFT2PH_GetInstValuesPh1** (tMETERLIBFFT2PH_DATA *p, double *urms, double *irms, double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

  不计费的(U、I、P、Q、S、PF、THD) 瞬时变量相 1 读取函数
- void **METERLIBFFT2PH_GetInstValuesPh2** (tMETERLIBFFT2PH_DATA *p, double *urms, double *irms, double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

  不计费的(U、I、P、Q、S、PF、THD) 瞬时变量相 2 读取函数
- void **METERLIBFFT2PH_GetMagnitudesPh1** (tMETERLIBFFT2PH_DATA *p, unsigned long magn_fft)

  相 1 谐波幅度计算函数
- void **METERLIBFFT2PH_GetMagnitudesPh2** (tMETERLIBFFT2PH_DATA *p, unsigned long magn_fft)

  相 2 谐波幅度计算函数
- void **METERLIBFFT2PH_GetPhasesPh1** (tMETERLIBFFT2PH_DATA *p, unsigned long ph_fft)

  相 1 谐波相移计算函数
- void **METERLIBFFT2PH_GetPhasesPh2** (tMETERLIBFFT2PH_DATA *p, unsigned long ph_fft)

  相 2 谐波相移计算函数
- void **METERLIBFFT2PH_InitAuxBuffPh1** (tMETERLIBFFT2PH_DATA *p, Frac24 *mag_u, Frac24 *mag_i, long *ph_u, long *ph_i)

  相 1 辅助缓冲区初始化函数
- void **METERLIBFFT2PH_InitAuxBuffPh2** (tMETERLIBFFT2PH_DATA *p, Frac24 *mag_u, Frac24 *mag_i, long *ph_u, long *ph_i)

  相 2 辅助缓冲区初始化函数
- void **METERLIBFFT2PH_InitMainBuffPh1** (tMETERLIBFFT2PH_DATA *p, Frac24 *u_re, Frac24 *i_re, Frac24 *u_im, Frac24 *i_im, long *shift)

  相 1 主缓冲区初始化函数
- void **METERLIBFFT2PH_InitMainBuffPh2** (tMETERLIBFFT2PH_DATA *p, Frac24 *u_re, Frac24 *i_re, Frac24 *u_im, Frac24 *i_im, long *shift)

  相 2 主缓冲区初始化函数
- long **METERLIBFFT2PH_InitParam** (tMETERLIBFFT2PH_DATA *p, unsigned long samples, unsigned long sensor, unsigned long kwh_cnt, unsigned long kvarh_cnt, unsigned long en_res)

  参数初始化函数

- long **METERLIBFFT2PH_Interpolation** (tMETERLIBFFT2PH_DATA *p, unsigned long u_ord, unsigned long i_ord, unsigned long samples_inp)

  插值函数
- long **METERLIBFFT2PH_SetCalibCoeffPh1** (tMETERLIBFFT2PH_DATA *p, double u_max, double i_max, Frac24 *i_offs, double p_offs, double q_offs)

  相 1 设置校准系数函数
- long **METERLIBFFT2PH_SetCalibCoeffPh2** (tMETERLIBFFT2PH_DATA *p, double u_max, double i_max, Frac24 *i_offs, double p_offs, double q_offs)

  相 2 设置校准系数函数

5.2.3 三相功率计量 API

- void **METERLIBFFT3PH_CalcMain** (tMETERLIBFFT3PH_DATA *p)

  FFT 计算和信号调节处理函数
- long **METERLIBFFT3PH_CalcVarHours** (tMETERLIBFFT3PH_DATA *p, unsigned long *varh_i, unsigned long *varh_e, unsigned long frequency)

  无功功率计算函数
- long **METERLIBFFT3PH_CalcWattHours** (tMETERLIBFFT3PH_DATA *p, unsigned long *wh_i, unsigned long *wh_e, unsigned long frequency)

  有功功率计算函数
- void **METERLIBFFT3PH_GetAvrgValuesPh1** (tMETERLIBFFT3PH_DATA *p, double *urms, double *irms, double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

  不计费的 (U、I、P、Q、S、PF、THD) 平均变量相 1 读取函数
- void **METERLIBFFT3PH_GetAvrgValuesPh2** (tMETERLIBFFT3PH_DATA *p, double *urms, double *irms, double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

  不计费的 (U、I、P、Q、S、PF、THD) 平均变量相 2 读取函数
- void **METERLIBFFT3PH_GetAvrgValuesPh3** (tMETERLIBFFT3PH_DATA *p, double *urms, double *irms, double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

  不计费的 (U、I、P、Q、S、PF、THD) 平均变量相 3 读取函数
- void **METERLIBFFT3PH_GetInstValuesPh1** (tMETERLIBFFT3PH_DATA *p, double *urms, double *irms, double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

  不计费的 (U、I、P、Q、S、PF、THD) 瞬时变量相 1 读取函数
- void **METERLIBFFT3PH_GetInstValuesPh2** (tMETERLIBFFT3PH_DATA *p, double *urms, double *irms, double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

  不计费的 (U、I、P、Q、S、PF、THD) 瞬时变量相 2 读取函数
- void **METERLIBFFT3PH_GetInstValuesPh3** (tMETERLIBFFT3PH_DATA *p, double *urms, double *irms, double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

  不计费的 (U、I、P、Q、S、PF、THD) 瞬时变量相 3 读取函数

- void **METERLIBFFT3PH_GetMagnitudesPh1** (tMETERLIBFFT3PH_DATA *p, unsigned long magn_fft)

  相 1 谐波幅度计算函数
- void **METERLIBFFT3PH_GetMagnitudesPh2** (tMETERLIBFFT3PH_DATA *p, unsigned long magn_fft)

  相 2 谐波幅度计算函数
- void **METERLIBFFT3PH_GetMagnitudesPh3** (tMETERLIBFFT3PH_DATA *p, unsigned long magn_fft)

  相 3 谐波幅度计算函数
- void **METERLIBFFT3PH_GetPhasesPh1** (tMETERLIBFFT3PH_DATA *p, unsigned long ph_fft)

  相 1 谐波相移计算函数
- void **METERLIBFFT3PH_GetPhasesPh2** (tMETERLIBFFT3PH_DATA *p, unsigned long ph_fft)

  相 2 谐波相移计算函数
- void **METERLIBFFT3PH_GetPhasesPh3** (tMETERLIBFFT3PH_DATA *p, unsigned long ph_fft)

  相 3 谐波相移计算函数
- void **METERLIBFFT3PH_InitAuxBuffPh1** (tMETERLIBFFT3PH_DATA *p, Frac24 *mag_u, Frac24 *mag_i, long *ph_u, long *ph_i)

  相 1 辅助缓冲区初始化函数
- void **METERLIBFFT3PH_InitAuxBuffPh2** (tMETERLIBFFT3PH_DATA *p, Frac24 *mag_u, Frac24 *mag_i, long *ph_u, long *ph_i)

  相 2 辅助缓冲区初始化函数
- void **METERLIBFFT3PH_InitAuxBuffPh3** (tMETERLIBFFT3PH_DATA *p, Frac24 *mag_u, Frac24 *mag_i, long *ph_u, long *ph_i)

  相 3 辅助缓冲区初始化函数
- void **METERLIBFFT3PH_InitMainBuffPh1** (tMETERLIBFFT3PH_DATA *p, Frac24 *u_re, Frac24 *i_re, Frac24 *u_im, Frac24 *i_im, long *shift)

  相 1 主缓冲区初始化函数
- void **METERLIBFFT3PH_InitMainBuffPh2** (tMETERLIBFFT3PH_DATA *p, Frac24 *u_re, Frac24 *i_re, Frac24 *u_im, Frac24 *i_im, long *shift)

  相 2 主缓冲区初始化函数
- void **METERLIBFFT3PH_InitMainBuffPh3** (tMETERLIBFFT3PH_DATA *p, Frac24 *u_re, Frac24 *i_re, Frac24 *u_im, Frac24 *i_im, long *shift)

  相 3 主缓冲区初始化函数
- long **METERLIBFFT3PH_InitParam** (tMETERLIBFFT3PH_DATA *p, unsigned long samples, unsigned long sensor, unsigned long kwh_cnt, unsigned long kvarh_cnt, unsigned long en_res)

  参数初始化函数

- long **METERLIBFFT3PH_Interpolation** (tMETERLIBFFT3PH_DATA *p, unsigned long u_ord, unsigned long i_ord, unsigned long samples_inp)

  插值函数
- long **METERLIBFFT3PH_SetCalibCoeffPh1** (tMETERLIBFFT3PH_DATA *p, double u_max, double i_max, Frac24 *i_offs, double p_offs, double q_offs)

  相 1 设置校准系数函数
- long **METERLIBFFT3PH_SetCalibCoeffPh2** (tMETERLIBFFT3PH_DATA *p, double u_max, double i_max, Frac24 *i_offs, double p_offs, double q_offs)

  相 2 设置校准系数函数
- long **METERLIBFFT3PH_SetCalibCoeffPh3** (tMETERLIBFFT3PH_DATA *p, double u_max, double i_max, Frac24 *i_offs, double p_offs, double q_offs)

  相 3 设置校准系数函数
- long **METERLIBFFT3PH_GetRotation**(tMETERLIBFFT3PH_DATA *p, double *u12_ph, double *u13_ph, double *u23_ph)

  角度和旋转计算函数

5.2.4 常见功率计 API

- void **METERLIBFFT_SetEnergy** (p, whi, whe, varhi, varhe)

  常见置位 / 清零能量计数器定义

5.3 METERLIBFFT_CalcMain

首先，这些函数可对电压和电流信号执行主要的 FFT 计算过程；即，它使用基 -2 DIT 算法将来自时间域的输入数据转换为频率域（请参见 3.1 节，“radix-2 基二时间抽取 FFT 说明 ”）。数据将以直角坐标数据格式进行内部计算。

其次，这些函数执行额外的电流信号调节处理，包括软件相移校正（如需要）和对衍生类型的电流传感器进行信号集成（如需要）。所有这些额外的处理均可以消除电流传感器误差和在频率域中使用软件计算的传感器特性。

最后，这些函数可执行额外的后处理，例如按照设计单元进行缩放以及取所有不计费的值的平均值，并且可将其保存至内部数据结构。可以针对每相单独执行以上所有操作。

为了正确计算，这些函数需要周期性测量瞬时电压和电流样本，并在先前信号周期内将其保存至 “ 时间域 ” 缓冲区中。计算之后，这些缓冲区（由 *u_re* 和 *i_re* 指针寻址）将由频率域的实数值改写，而结果的虚数频率域部分将保存至 *u_im* 和 *i_im* 指针所设置的单独缓冲区中。必须通过 5.12 节，“METERLIBFFT_InitParam” 中描述的函数初始化所有这些缓冲区的指针。第一个 FFT 缓冲区位置与零谐波匹配，以此类推。

5.3.1 句法

```
#include"meterlibfft.h"
void METERLIBFFT1PH_CalcMain (tMETERLIBFFT1PH_DATA *p);
void METERLIBFFT2PH_CalcMain (tMETERLIBFFT2PH_DATA *p);
void METERLIBFFT3PH_CalcMain (tMETERLIBFFT3PH_DATA *p);
```

5.3.2 参数

表 4. METERLIBFFT_CalcMain 函数参数

类型	名称	方向	说明
tMETERLIBFFT1PH_DATA	p	输入	指向单相计量库数据结构的指针
tMETERLIBFFT2PH_DATA	p	输入	指向双相计量库数据结构的指针
tMETERLIBFFT3PH_DATA	p	输入	指向三相计量库数据结构的指针

5.3.3 返回

这些函数不会返回任何参数。

5.3.4 调用顺序

系统将以定义的间隔周期性调用所有这些函数，具体取决于线路频率和 / 或多个 ADC 采样率（同步或异步处理）。必须在调用这些函数之前执行单次强制参数初始化。必须在调用这些函数之前完成周期性插值过程。应在调用这些函数之后立即执行能量计算过程。

5.3.5 性能

表 5. METERLIBFFT_CalcMain CM0+ 内核的函数性能

函数名称	代码大小[1] [B]	时钟周期[2]		
		基本模式	集成模式	相移校正模式
METERLIBFFT1PH_CalcMain	2524	95465	98343	95465+2880*harm
METERLIBFFT2PH_CalcMain	2938	190450	196207	190450+6230*harm
METERLIBFFT3PH_CalcMain	3376	285435	295030	285435+9600*harm
附注： *harm* 为已相移谐波的总数				

1 正弦查找表的代码大小 (4 KB) 不包括在内

2 对于 64 个输入样本、 32 个输出谐波、包含第 1 和第 5 谐波的 U 和 I 信号的测试向量有效

表 6. METERLIBFFT_CalcMain 带 MMAU 的 CM0+ 内核的函数性能

函数名称	代码大小[1] [B]	时钟周期[2]		
		基本模式	集成模式	相移校正模式
METERLIBFFT1PH_CalcMain	3094	49891	52770	49891+840*harm
METERLIBFFT2PH_CalcMain	3508	99782	105539	99782+1920*harm
METERLIBFFT3PH_CalcMain	3946	149674	158309	149674+2640*harm
附注： *harm* 为相移谐波的总数				

1 正弦查找表的代码大小 (4 KB) 不包括在内

2 对于 64 个输入样本、 32 个输出谐波、包含第 1 和第 5 谐波的 U 和 I 信号的测试向量有效

5.4 METERLIBFFT_CalcVarHours

这些函数对所有相同时执行无功能量（导入、导出）计算。两个能量通过每个时间单元累积的功率，采用每个相的瞬时无功功率增量来计算。当总导入无功能量根据正无功功率增量计算得出时，总导出无功能量则根据负无功功率增量计算得出。输出能量分辨率 （*varh_i* 和 *varh_e*）取决于 5.12 节， “METERLIBFFT_InitParam” 中所述函数设定的 *en_res* 参数。

5.4.1 句法

```
#include"meterlibfft.h"

long METERLIBFFT1PH_CalcVarHours (tMETERLIBFFT1PH_DATA *p, unsigned long *varh_i, unsigned long
*varh_e, unsigned long frequency);

long METERLIBFFT2PH_CalcVarHours (tMETERLIBFFT2PH_DATA *p, unsigned long *varh_i, unsigned long
*varh_e, unsigned long frequency);

long METERLIBFFT3PH_CalcVarHours (tMETERLIBFFT3PH_DATA *p, unsigned long *varh_i, unsigned long
*varh_e, unsigned long frequency);
```

5.4.2 参数

表 7. METERLIBFFT_CalcVarHours 函数参数

类型	名称	方向	说明
tMETERLIBFFT1PH_DATA	p	输入	指向单相计量库数据结构的指针
tMETERLIBFFT2PH_DATA	p	输入	指向双相计量库数据结构的指针
tMETERLIBFFT3PH_DATA	p	输入	指向三相计量库数据结构的指针
无符号长整型	varh_i	输出	指向 LCD 导入无功电量计数器的指针
无符号长整型	varh_e	输出	指向 LCD 导出无功电量计数器的指针
无符号长整型	频率	输入	线路频率 [mHz]，例如 50000 = 50.000 Hz

5.4.3 返回

如果为正，则函数将针对电流线路周期返回无功能量 LED 闪烁频率分辨率 (mHz)（每个周期仅允许一次 LED 闪烁）。这可用于使用软件和定时器的低抖动脉冲输出生成 （专利方法）。

如果为负，则此电流周期无需生成输出脉冲。

5.4.4 调用顺序

系统将以定义的间隔周期性调用所有这些函数，具体取决于线路频率和 / 或多个 ADC 采样率。总之，必须在主 (FFT) 计算过程完成之后立即调用这些函数。必须在调用这些函数之前完成单次强制参数初始化。

5.4.5 性能

表 8. METERLIBFFT_CalcVarHours 函数性能

函数名称	CM0+ 内核		带 MMAU 的 CM0+ 内核	
	代码大小 [B]	时钟周期	代码大小 [B]	时钟周期
METERLIBFFT1PH_CalcVarHours	292	698	292	696
METERLIBFFT2PH_CalcVarHours	372	751	372	744
METERLIBFFT3PH_CalcVarHours	490	883	490	878

5.5 METERLIBFFT_CalcWattHours

这些函数对所有相同时执行有功能量（导入、导出）计算。两个能量通过每个时间单元累积的功率，采用每个相的瞬时有功功率增量来计算。当总导入有功能量根据正有功功率增量计算得出时，总导出有功能量则根据负有功功率增量计算得出。输出能量分辨率 （*wh_i* 和 *wh_e*）取决于 5.12 节， “METERLIBFFT_InitParam” 中所述函数设定的 *en_res* 参数。

5.5.1 句法

```
#include"meterlibfft.h"

long METERLIBFFT1PH_CalcWattHours (tMETERLIBFFT1PH_DATA *p, unsigned long *wh_i, unsigned long
*wh_e, unsigned long frequency);

long METERLIBFFT2PH_CalcWattHours (tMETERLIBFFT2PH_DATA *p, unsigned long *wh_i, unsigned long
*wh_e, unsigned long frequency);

long METERLIBFFT3PH_CalcWattHours (tMETERLIBFFT3PH_DATA *p, unsigned long *wh_i, unsigned long
*wh_e, unsigned long frequency);
```

5.5.2 参数

表 9. METERLIBFFT_CalcWattHours 函数参数

类型	名称	方向	说明
tMETERLIBFFT1PH_DATA	p	输入	指向单相计量库数据结构的指针
tMETERLIBFFT2PH_DATA	p	输入	指向双相计量库数据结构的指针
tMETERLIBFFT3PH_DATA	p	输入	指向三相计量库数据结构的指针
无符号长整型	wh_i	输出	指向 LCD 导入有功电量计数器的指针
无符号长整型	wh_e	输出	指向 LCD 导出有功电量计数器的指针
无符号长整型	频率	输入	线路频率 [mHz]，例如 50000 = 50.000 Hz

5.5.3 返回

如果为正，则函数将针对电流线路周期返回有功能量 LED 闪烁频率分辨率 (MHz)（每个周期仅允许一次 LED 闪烁）。这可用于使用软件和定时器的低抖动脉冲输出生成（专利方法）。如果为负，则此电流周期无需生成输出脉冲。

5.5.4 调用顺序

系统将以定义的间隔周期性调用所有这些函数，具体取决于线路频率和 / 或多个 ADC 采样率。总之，应在主 (FFT) 计算过程完成之后立即调用这些函数。必须在调用这些函数之前完成单次强制参数初始化。

5.5.5 性能

表 10. METERLIBFFT_CalcWattHours 函数性能

函数名称	CM0+ 内核		带 MMAU 的 CM0+ 内核	
	代码大小 [B]	时钟周期	代码大小 [B]	时钟周期
METERLIBFFT1PH_CalcWattHours	292	698	292	696
METERLIBFFT2PH_CalcWattHours	372	751	372	744
METERLIBFFT3PH_CalcWattHours	490	883	490	878

5.6 METERLIBFFT_GetAvrgValues

这些函数将返回所有平均不计费的值，这些值已针对每个相按照设计单元单独调整。这些值可用于 LCD 显示、远程数据可视化等。

5.6.1 句法

```
#include"meterlibfft.h"
```

```
void METERLIBFFT1PH_GetAvrgValues (tMETERLIBFFT1PH_DATA *p, double *urms, double *irms, double
*w, double *var, double *va, double *pf, double *thd_u, double *thd_i);

void METERLIBFFT2PH_GetAvrgValuesPh1 (tMETERLIBFFT2PH_DATA *p, double *urms, double *irms,
double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i);

void METERLIBFFT2PH_GetAvrgValuesPh2 (tMETERLIBFFT2PH_DATA *p, double *urms, double *irms,
double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i);

void METERLIBFFT3PH_GetAvrgValuesPh1 (tMETERLIBFFT3PH_DATA *p, double *urms, double *irms,
double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i);

void METERLIBFFT3PH_GetAvrgValuesPh2 (tMETERLIBFFT3PH_DATA *p, double *urms, double *irms,
double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i);

void METERLIBFFT3PH_GetAvrgValuesPh3 (tMETERLIBFFT3PH_DATA *p, double *urms, double *irms,
double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i);
```

5.6.2 参数

表 11. METERLIBFFT_GetAvrgValues 函数参数

类型	名称	方向	说明
tMETERLIBFFT1PH_DATA	p	输入	指向单相计量库数据结构的指针
tMETERLIBFFT2PH_DATA	p	输入	指向双相计量库数据结构的指针
tMETERLIBFFT3PH_DATA	p	输入	指向三相计量库数据结构的指针
双精度浮点	urms	输出	指向平均 RMS 线路电压 (V) 值的指针
双精度浮点	irms	输出	指向平均 RMS 线路电流 (A) 值的指针
双精度浮点	w	输出	指向平均有功功率 (P) 值 (W) 的指针
双精度浮点	var	输出	指向平均无功功率 (Q) 值 (var) 的指针
双精度浮点	va	输出	指向平均无符号视在功率值 (S)(va) 的指针
双精度浮点	pf	输出	指向平均功率因数值 （无量纲量）的指针
双精度浮点	thd_u	输出	指向平均 THD 电压 (V) 值 (%) 的指针
双精度浮点	thd_i	输出	指向平均 THD 电流值 (%) 的指针

5.6.3 返回

这些函数不会返回任何参数。

5.6.4 调用顺序

这些函数的调用频率应该在范围 < 0.004 Hz – 线路频率 > 内。如果调用频率过低，则内部计数器可能会溢出。在此情况下，必须先进行虚拟读取，以清零所有内部计数器。如果调用频率过高，则所有输出值将等于零。所有输出值将以双精度形式根据设计单元进行调整。

5.6.5 性能

表 12. METERLIBFFT_GetAvrgValues 函数性能

函数名称	CM0+ 内核		带 MMAU 的 CM0+ 内核	
	代码大小 [B]	时钟周期[1]	代码大小 [B]	时钟周期[1]
METERLIBFFT1PH_GetAvrgValues	296	6764	296	6764
METERLIBFFT2PH_GetAvrgValuesPh1	296	6764	296	6764
METERLIBFFT2PH_GetAvrgValuesPh2	298		298	
METERLIBFFT3PH_GetAvrgValuesPh1	296	6764	296	6764
METERLIBFFT3PH_GetAvrgValuesPh2	298		298	
METERLIBFFT3PH_GetAvrgValuesPh3	298		298	

[1] 每 25 个周期 （每秒两次）计算一次平均值

5.7 METERLIBFFT_GetInstValues

这些函数将返回所有瞬时不计费的值，这些值已针对每个相按照设计单元单独调整。

5.7.1 句法

```
#include“meterlibfft.h”

void METERLIBFFT1PH_GetInstValues (tMETERLIBFFT1PH_DATA *p, double *urms, double *irms, double
*w, double *var, double *va, double *pf, double *thd_u, double *thd_i);

void METERLIBFFT2PH_GetInstValuesPh1 (tMETERLIBFFT2PH_DATA *p, double *urms, double *irms,
double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

void METERLIBFFT2PH_GetInstValuesPh2 (tMETERLIBFFT2PH_DATA *p, double *urms, double *irms,
double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

void METERLIBFFT3PH_GetInstValuesPh1 (tMETERLIBFFT3PH_DATA *p, double *urms, double *irms,
double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

void METERLIBFFT3PH_GetInstValuesPh2 (tMETERLIBFFT3PH_DATA *p, double *urms, double *irms,
double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)

void METERLIBFFT3PH_GetInstValuesPh3 (tMETERLIBFFT3PH_DATA *p, double *urms, double *irms,
double *w, double *var, double *va, double *pf, double *thd_u, double *thd_i)
```

5.7.2 参数

表 13. METERLIBFFT_GetInstValues 函数参数

类型	名称	方向	说明
tMETERLIBFFT1PH_DATA	p	输入	指向单相计量库数据结构的指针
tMETERLIBFFT2PH_DATA	p	输入	指向双相计量库数据结构的指针
tMETERLIBFFT3PH_DATA	p	输入	指向三相计量库数据结构的指针
双精度浮点	urms	输出	指向瞬时 RMS 线路电压值 (V) 的指针
双精度浮点	irms	输出	指向瞬时 RMS 线路电流值 (A) 的指针
双精度浮点	w	输出	指向瞬时有功功率 (P) 值 (W) 的指针
双精度浮点	var	输出	指向瞬时无功功率 (Q) 值 (var) 的指针
双精度浮点	va	输出	指向瞬时无符号视在功率值 (S)(va) 的指针
双精度浮点	pf	输出	指向瞬时功率因数值 （无量纲量）的指针
双精度浮点	thd_u	输出	指向瞬时 THD 电压值 (%) 的指针
双精度浮点	thd_i	输出	指向瞬时 THD 电流值 (%) 的指针

5.7.3 返回

这些函数不会返回任何参数。

5.7.4 调用顺序

可随时调用这些函数，最佳时间是在完成主 (FFT) 计算过程之后。所有输出值将以双精度形式根据设计单元进行调整。

5.7.5 性能

表 14. METERLIBFFT_GetInstValues 函数性能

函数名称	CM0+ 内核		带 MMAU 的 CM0+ 内核	
	代码大小 [B]	时钟周期	代码大小 [B]	时钟周期
METERLIBFFT1PH_GetInstValues	232	5925	232	5925
METERLIBFFT2PH_GetInstValuesPh1	232	5925	232	5925
METERLIBFFT2PH_GetInstValuesPh2	234		234	
METERLIBFFT3PH_GetInstValuesPh1	232	5949	232	5925
METERLIBFFT3PH_GetInstValuesPh2	234		234	
METERLIBFFT3PH_GetInstValuesPh3	234		234	

5.8 METERLIBFFT_GetMagnitudes

这些函数将电压和电流数据（通过 5.3 节，“METERLIBFFT_CalcMain” 中所述的函数计算所得）从直角坐标数据形式转换为极坐标形式，并单独返回每个相的电压和电流谐波幅度。这些值可用于其他的后处理和可视化。电压和电流幅度位于 *mag_u* 和 *mag_i* 指针（由 5.10 节，“METERLIBFFT_InitAuxBuff” 中所述的函数初始化）寻址的两个缓冲区中。第一个缓冲区位置与零谐波匹配，以此类推。

5.8.1 句法

```
#include“meterlibfft.h”
void METERLIBFFT1PH_GetMagnitudes (tMETERLIBFFT1PH_DATA *p, unsigned long magn_fft);
void METERLIBFFT2PH_GetMagnitudesPh1 (tMETERLIBFFT2PH_DATA *p, unsigned long magn_fft);
void METERLIBFFT2PH_GetMagnitudesPh2 (tMETERLIBFFT2PH_DATA *p, unsigned long magn_fft);
void METERLIBFFT3PH_GetMagnitudesPh1 (tMETERLIBFFT3PH_DATA *p, unsigned long magn_fft);
void METERLIBFFT3PH_GetMagnitudesPh2 (tMETERLIBFFT3PH_DATA *p, unsigned long magn_fft);
void METERLIBFFT3PH_GetMagnitudesPh3 (tMETERLIBFFT3PH_DATA *p, unsigned long magn_fft);
```

5.8.2 参数

表 15. METERLIBFFT_GetMagnitudes 函数参数

类型	名称	方向	说明
tMETERLIBFFT1PH_DATA	p	输入	指向单相计量库数据结构的指针
tMETERLIBFFT2PH_DATA	p	输入	指向双相计量库数据结构的指针
tMETERLIBFFT3PH_DATA	p	输入	指向三相计量库数据结构的指针
无符号长整型	magn_fft	输入	< 输入样本的 – 一半 > 范围内所需的谐波幅度数

5.8.3 返回

这些函数不会返回任何参数。

5.8.4 调用顺序

必须在主 (FFT) 计算过程完成之后调用这些函数。必须在调用这些函数之前完成单次强制和辅助参数初始化。

5.8.5 性能

表 16. METERLIBFFT_GetMagnitudes 函数性能

函数名称	CM0+ 内核		带 MMAU 的 CM0+ 内核	
	代码大小 [B]	时钟周期[1]	代码大小 [B]	时钟周期[1]
METERLIBFFT1PH_GetMagnitudes	234	17702	444	3142
METERLIBFFT2PH_GetMagnitudesPh1	234	17702	444	3142
METERLIBFFT2PH_GetMagnitudesPh2	236		446	
METERLIBFFT3PH_GetMagnitudesPh1	234	17702	444	3142
METERLIBFFT3PH_GetMagnitudesPh2	236		446	
METERLIBFFT3PH_GetMagnitudesPh3	240		450	

[1] 对于 32 个谐波幅度有效

5.9 METERLIBFFT_GetPhases

这些函数将电压和电流数据（通过 5.3 节，“METERLIBFFT_CalcMain”中所述的函数计算所得）从直角坐标数据形式转换为极坐标形式，并单独返回每个相的电压和电流谐波相移。这些值可用于其他的后处理和可视化。电压和电流相移可用于 *ph_u* 和 *ph_i* 指针（由 5.10 节，“METERLIBFFT_InitAuxBuff”中所述的函数初始化）寻址的两个缓冲区中。第一个缓冲区位置与零谐波匹配，以此类推。

5.9.1 句法

```
#include"meterlibfft.h"
void METERLIBFFT1PH_GetPhases (tMETERLIBFFT1PH_DATA *p, unsigned long ph_fft);
void METERLIBFFT2PH_GetPhasesPh1 (tMETERLIBFFT2PH_DATA *p, unsigned long ph_fft)
void METERLIBFFT2PH_GetPhasesPh2 (tMETERLIBFFT2PH_DATA *p, unsigned long ph_fft)
void METERLIBFFT3PH_GetPhasesPh1 (tMETERLIBFFT3PH_DATA *p, unsigned long ph_fft)
void METERLIBFFT3PH_GetPhasesPh2 (tMETERLIBFFT3PH_DATA *p, unsigned long ph_fft)
void METERLIBFFT3PH_GetPhasesPh3 (tMETERLIBFFT3PH_DATA *p, unsigned long ph_fft)
```

5.9.2 参数

表 17. METERLIBFFT_GetPhases 函数参数

类型	名称	方向	说明
tMETERLIBFFT1PH_DATA	p	输入	指向单相计量库数据结构的指针
tMETERLIBFFT2PH_DATA	p	输入	指向双相计量库数据结构的指针
tMETERLIBFFT3PH_DATA	p	输入	指向三相计量库数据结构的指针
无符号长整型	ph_fft	输入	< 输入样本的 – 一半 > 范围内所需的谐波相移数

5.9.3 返回

这些函数不会返回任何参数。

5.9.4 调用顺序

必须在主 (FFT) 计算过程完成之后调用此函数。必须在调用这些函数之前执行单次强制和辅助参数初始化。

5.9.5 性能

表 18. METERLIBFFT_GetPhases 函数性能

函数名称	CM0+ 内核		带 MMAU 的 CM0+ 内核	
	代码大小[1] [B]	时钟周期[2]	代码大小[1] [B]	时钟周期[2]
METERLIBFFT1PH_GetPhases	346	15591	338	6188
METERLIBFFT2PH_GetPhasesPh1	346	15591	338	6188
METERLIBFFT2PH_GetPhasesPh2	348		340	
METERLIBFFT3PH_GetPhasesPh1	346	15639	338	6188
METERLIBFFT3PH_GetPhasesPh2	348		340	
METERLIBFFT3PH_GetPhasesPh3	352		344	

1 反正切查找表的代码大小 (8 KB) 不包括在内

2 对于 32 个谐波相移有效

5.10 METERLIBFFT_InitAuxBuff

这些函数用于初始化电压和电流幅度以及相移缓冲区的指针。可以针对各特定相单独执行初始化。这些缓冲器并非主 FFT 计算所必需，因此，这些初始化只是辅助性的，应在需要额外谐波幅度和相移计算时执行。这些计算由 5.8 节，“METERLIBFFT_GetMagnitudes” 和 5.9 节，“METERLIBFFT_GetPhases” 中所述的函数执行。第一个缓冲区位置与零谐波匹配，以此类推。这些缓冲区的 l 长度可选，但其最大长度不得超过 FFT 谐波数（5.12 节，“METERLIBFFT_InitParam” 中 *样本* 参数设定的输入样本的一半）。

5.10.1 句法

```
#include“meterlibfft.h”

void METERLIBFFT1PH_InitAuxBuff (tMETERLIBFFT1PH_DATA *p, Frac24 *mag_u, Frac24 *mag_i, long
*ph_u, long *ph_i);

void METERLIBFFT2PH_InitAuxBuffPh1(tMETERLIBFFT2PH_DATA *p, Frac24 *mag_u, Frac24 *mag_i, long
*ph_u, long *ph_i);

void METERLIBFFT2PH_InitAuxBuffPh2(tMETERLIBFFT2PH_DATA *p, Frac24 *mag_u, Frac24 *mag_i, long
*ph_u, long *ph_i);

void METERLIBFFT3PH_InitAuxBuffPh1(tMETERLIBFFT3PH_DATA *p, Frac24 *mag_u, Frac24 *mag_i, long
*ph_u, long *ph_i);

void METERLIBFFT3PH_InitAuxBuffPh2(tMETERLIBFFT3PH_DATA *p, Frac24 *mag_u, Frac24 *mag_i, long
*ph_u, long *ph_i);

void METERLIBFFT3PH_InitAuxBuffPh3(tMETERLIBFFT3PH_DATA *p, Frac24 *mag_u, Frac24 *mag_i, long
*ph_u, long *ph_i);
```

5.10.2 参数

表 19. METERLIBFFT_InitAuxBuff 函数参数

类型	名称	方向	说明
tMETERLIBFFT1PH_DATA	p	输入	指向单相计量库数据结构的指针
tMETERLIBFFT2PH_DATA	p	输入	指向双相计量库数据结构的指针
tMETERLIBFFT3PH_DATA	p	输入	指向三相计量库数据结构的指针
Frac24	mag_u	输出	指向谐波幅度电压缓冲区 （Q0.23 数据格式）的指针
Frac24	mag_i	输出	指向谐波幅度电流缓冲区 （Q0.23 数据格式）的指针
长整形指针	ph_u	输出	指向谐波相移电压缓冲区 (0.001°) 的指针，例如 45000 = 45.000°
长整形指针	ph_i	输出	指向谐波相移电流缓冲区 (0.001°) 的指针，例如 45000 = 45.000°

5.10.3 返回

这些函数不会返回任何参数。

5.10.4 调用顺序

应仅在初始化部分调用这些函数。

5.10.5 性能

表 20. METERLIBFFT_InitAuxBuff 函数性能

函数名称	CM0+ 内核		带 MMAU 的 CM0+ 内核	
	代码大小 [B]	时钟周期	代码大小 [B]	时钟周期
METERLIBFFT1PH_InitAuxBuff	12	32		30
METERLIBFFT2PH_InitAuxBuffPh1	12	32	12	30
METERLIBFFT2PH_InitAuxBuffPh2	22		22	
METERLIBFFT3PH_InitAuxBuffPh1	12	32	12	30
METERLIBFFT3PH_InitAuxBuffPh2	22		22	
METERLIBFFT3PH_InitAuxBuffPh3	22		22	

5.11 METERLIBFFT_InitMainBuff

这些函数用于初始化三类缓冲区的指针：输入时间域缓冲区、输出频率域缓冲区和相移校正缓冲区。第一个缓冲区位置与零谐波匹配，以此类推。可以为各特定相单独执行所有这些初始化。

输入电压和电流时间域缓冲区 （采样后在其中保存 ADC 值）与输出频率域缓冲器 （计算后在其中保存 FFT 结果的实数部分）相结合。因此，输入时间域数据将由 FFT 结果的实数部分改写，而 FFT 结果的虚数部分则保存至单独的缓冲区。时间域缓冲区的长度取决于最大输入样本数，频率域缓冲区的长度始终为二的幂（由 5.12 节，“METERLIBFFT_InitParam” 中所述函数中的*样本*参数设定）。这些函数还可以初始化 U-I 相移校正缓冲区的指针，以便实现频率域中的软件相移校正。将会单独保存每个谐波的寄生电流传感器相移。这些相移可由 5.3 节，“METERLIBFFT_CalcMain” 中所述的函数补偿。如果无需软件相移校正，则必须将此指针设为 NULL。

5.11.1 句法

```
#include“meterlibfft.h”

void METERLIBFFT1PH_InitMainBuff (tMETERLIBFFT1PH_DATA *p, Frac24 *u_re, Frac24 *i_re, Frac24
*u_im, Frac24 *i_im, long *shift);

void METERLIBFFT2PH_InitMainBuffPh1 (tMETERLIBFFT2PH_DATA *p, Frac24 *u_re, Frac24 *i_re,
Frac24 *u_im, Frac24 *i_im, long *shift);

void METERLIBFFT2PH_InitMainBuffPh2 (tMETERLIBFFT2PH_DATA *p, Frac24 *u_re, Frac24 *i_re,
Frac24 *u_im, Frac24 *i_im, long *shift);

void METERLIBFFT3PH_InitMainBuffPh1 (tMETERLIBFFT3PH_DATA *p, Frac24 *u_re, Frac24 *i_re,
Frac24 *u_im, Frac24 *i_im, long *shift);

void METERLIBFFT3PH_InitMainBuffPh2 (tMETERLIBFFT3PH_DATA *p, Frac24 *u_re, Frac24 *i_re,
Frac24 *u_im, Frac24 *i_im, long *shift);

void METERLIBFFT3PH_InitMainBuffPh3 (tMETERLIBFFT3PH_DATA *p, Frac24 *u_re, Frac24 *i_re,
Frac24 *u_im, Frac24 *i_im, long *shift);
```

5.11.2 参数

表 21. METERLIBFFT_InitMainBuff 函数参数

类型	名称	方向	说明
tMETERLIBFFT1PH_DATA	p	输入	指向单相计量库数据结构的指针
tMETERLIBFFT2PH_DATA	p	输入	指向双相计量库数据结构的指针
tMETERLIBFFT3PH_DATA	p	输入	指向三相计量库数据结构的指针
Frac24	u_re	输入 / 输出	指向与输出频率域电压缓冲区 （实数部分）混合的输入时间域电压缓冲区 （Q0.23 数据格式）的指针
Frac24	i_re	输入 / 输出	指向与输出频率域电流缓冲区 （实数部分）混合的输入时间域电流缓冲区 （Q0.23 数据格式）的指针
Frac24	u_im	输出	指向频率域电压缓冲区 （虚数部分）（Q0.23 数据格式）的指针
Frac24	i_im	输出	指向频率域电流缓冲区 （虚数部分）（Q0.23 数据格式）的指针
长整形指针	移位	输入	指向 U–I 相移校正缓冲区 (0.001°) 的指针，例如 4500=4.500°。如无需校正，则设为 NULL。

5.11.3 返回

这些函数不会返回任何参数。

5.11.4 调用顺序

大部分缓冲区用于主 FFT 计算（由 5.3 节，“METERLIBFFT_CalcMain” 中所述的函数执行），因此，这些初始化是强制性的且必须在 i 初始化部分执行。

5.11.5 性能

表 22. METERLIBFFT_InitMainBuff 函数性能

函数名称	CM0+ 内核		带 MMAU 的 CM0+ 内核	
	代码大小 [B]	时钟周期	代码大小 [B]	时钟周期
METERLIBFFT1PH_InitMainBuff	26	57	26	55
METERLIBFFT2PH_InitMainBuffPh1	26	57	26	55
METERLIBFFT2PH_InitMainBuffPh2	28		28	
METERLIBFFT3PH_InitMainBuffPh1	26	57	26	55
METERLIBFFT3PH_InitMainBuffPh2	28		28	
METERLIBFFT3PH_InitMainBuffPh3	30		30	

5.12 METERLIBFFT_InitParam

这些函数用于参数的初始化。所有这些初始化对于所有相均有效。正确的初始化对于从其他函数接收正确的输出非常重要。

5.12.1 句法

```
#include"meterlibfft.h"

long METERLIBFFT1PH_InitParam (tMETERLIBFFT1PH_DATA *p, unsigned long samples, unsigned long sensor, unsigned long kwh_cnt, unsigned long kvarh_cnt, unsigned long en_res);

long METERLIBFFT2PH_InitParam (tMETERLIBFFT2PH_DATA *p, unsigned long samples, unsigned long sensor, unsigned long kwh_cnt, unsigned long kvarh_cnt, unsigned long en_res);

long METERLIBFFT3PH_InitParam (tMETERLIBFFT3PH_DATA *p, unsigned long samples, unsigned long sensor, unsigned long kwh_cnt, unsigned long kvarh_cnt, unsigned long en_res);
```

5.12.2 参数

表 23. METERLIBFFT_InitParam 函数参数

类型	名称	方向	说明
tMETERLIBFFT1PH_DATA	p	输入	指向单相计量库数据结构的指针
tMETERLIBFFT2PH_DATA	p	输入	指向双相计量库数据结构的指针
tMETERLIBFFT3PH_DATA	p	输入	指向三相计量库数据结构的指针
无符号长整型	样本	输入	所需 FFT 样本数 – 请参见表 24
无符号长整型	传感器	输入	电流传感器类型 – 请参见表 25
无符号长整型	kwh_cnt	输入	有功能量脉冲数 – 请参见表 26
无符号长整型	kvarh_cnt	输入	无功能量脉冲数 – 请参见表 26
无符号长整型	en_res	输入	有功 / 无功能量分辨率 – 请参见表 27

表 24. 输入样本定义数

定义名称	说明
SAMPLES8	8 个输入样本， 4 个输出谐波
SAMPLES16	16 个输入样本， 8 个输出谐波
SAMPLES32	32 个输入样本， 16 个输出谐波
SAMPLES64	64 个输入样本， 32 个输出谐波
SAMPLES128	128 个输入样本， 64 个输出谐波
SAMPLES256	256 个输入样本， 128 个输出谐波
SAMPLES512	512 个输入样本， 256 个输出谐波

表 25. 电流传感器类型定义

定义名称	说明
SENS_DERIV	衍生型电流传感器 （罗戈夫斯基线圈）
SENS_PROP	比例型电流传感器 （分流器、电流互感器）

表 26. 脉冲数定义

定义名称	说明	定义名称	说明
IMP200	200 imp / kWh 或 200 imp / kVARh	IMP5000	5000 imp / kWh 或 imp / kVARh
IMP250	250 imp / kWh 或 250 imp / kVARh	IMP10000	10000 imp / kWh 或 10000 imp / kVARh
IMP500	500 imp / kWh 或 500 imp / kVARh	IMP12500	12500 imp / kWh 或 12500 imp / kVARh
IMP1000	1000 imp / kWh 或 1000 imp / kVARh	IMP20000	20000 imp / kWh 或 20000 imp / kVARh
IMP1250	1250 imp / kWh 或 1250 imp / kVARh	IMP25000	25000 imp / kWh 或 25000 imp / kVARh
IMP2000	2000 imp / kWh 或 2000 imp / kVARh	IMP50000	50000 imp / kWh 或 50000 imp / kVARh
IMP2500	2500 imp / kWh 或 2500 imp / kVARh	IMP100000	100000 imp / kWh 或 100000 imp / kVARh

表 27. 能量分辨率定义

定义名称	说明
EN_RES1	能量分辨率为 1 Wh / VARh，最大范围为 4294.967 MWh / MVARh，支持的脉冲数 <= 1000
EN_RES10	能量分辨率为 0.1 Wh / VARh，最大范围为 429.4967 MWh / MVARh，支持的脉冲数 <= 10000
EN_RES100	能量分辨率为 0.01 Wh / VARh，最大范围为 42.94967 MWh / MVARh，支持所有脉冲数

5.12.3 返回

这些函数将返回以下错误代码之一：

- FFT_ERROR (positive) – 部分输入参数不正确，函数输出无效
- FFT_OK (zero) – 所有输入参数正确，函数输出有效

5.12.4 调用顺序

在初始化过程中或在编程时更改部分参数之后，必须首先调用这些强制函数。

5.12.5 性能

表 28. METERLIBFFT_InitParam 函数性能

函数名称	CM0+ 内核		带 MMAU 的 CM0+ 内核	
	代码大小 [B]	时钟周期	代码大小 [B]	时钟周期
METERLIBFFT1PH_InitParam	1380	787	1380	782
METERLIBFFT2PH_InitParam	1414	837	1414	830
METERLIBFFT3PH_InitParam	1440	849	1440	847

5.13 METERLIBFFT_Interpolation

这些函数用于将无符号整数样本提供的原始输入曲线插值到二的幂样本 （FFT 函数 [7] 所需）提供的曲线中。这些函数支持过采样和欠采样。

5.13.1 句法

```
#include"meterlibfft.h"

long METERLIBFFT1PH_Interpolation (tMETERLIBFFT1PH_DATA *p, unsigned long u_ord, unsigned long i_ord, unsigned long samples_inp);

long METERLIBFFT2PH_Interpolation (tMETERLIBFFT2PH_DATA *p, unsigned long u_ord, unsigned long i_ord, unsigned long samples_inp);

long METERLIBFFT3PH_Interpolation (tMETERLIBFFT3PH_DATA *p, unsigned long u_ord, unsigned long i_ord, unsigned long samples_inp);
```

5.13.2 参数

表 29. METERLIBFFT_Interpolation 函数参数

类型	名称	方向	说明
tMETERLIBFFT1PH_DATA	p	输入	指向单相计量库数据结构的指针
tMETERLIBFFT2PH_DATA	p	输入	指向双相计量库数据结构的指针
tMETERLIBFFT3PH_DATA	p	输入	指向三相计量库数据结构的指针
无符号长整型	u_ord	输入	电压插值阶 – 请参见表 30
无符号长整型	i_ord	输入	电流插值阶 – 请参见表 30
无符号长整型	samples_inp	输入	输入样本数可以高于或低于指数 FFT 样本所需的样本数

表 30. 插值阶定义

定义名称	说明
ORD1	1 阶 （一次）插值
ORD2	2 阶 （二次）插值
ORD3	3 阶 （三次）插值

5.13.3 返回

这些函数返回的以下错误代码之一仅对欠采样使用有效，对于输入样本少于 FFT 样本 （由 5.12 节， “METERLIBFFT_InitParam” 中的*样本*参数设定）：

- FFT_ERROR (positive) – FFT 样本多于输入样本，且 FFT 采样超过 128 个。
- FFT_OK (zero) – 欠采样率正确。

5.13.4 调用顺序

仅当 [7] 需要插值处理时才使用这些函数。在此情况下，系统将以既定的间隔周期性调用这些函数，具体取决于线路频率和 / 或多个 ADC 采样率。应在主 (FFT) 计算过程完成之前立即调用这些函数。必须在调用这些函数之前执行单次强制参数初始化。除了其他任务之外，此参数初始化函数还可设置所需的 FFT 点数，并初始化插值函数所用的输入缓冲区的所有指针。

附注

输入缓冲区中的原始值 （ADC 值）将在执行这些函数之后由新 （已插值）的值改写。

5.13.5 性能

表 31. METERLIBFFT_Interpolation CM0+ 内核的函数性能

函数名称	代码大小 [B]			堆栈大小 [B][1]	时钟周期[2]		
	1 阶	2 阶	3 阶		1 阶	2 阶	3 阶
METERLIBFFT1PH_Interpolation	506	842	1654	512	12521	35260	76996
METERLIBFFT2PH_Interpolation	586	922	1734		24946	70519	153991
METERLIBFFT3PH_Interpolation	682	1018	1830		37418	106019	231227

1 由于示例是欠采样 （输入样本 < FFT 样本）

2 输入样本数 = 120，所需 FFT 点数 = 64，两个通道的插值阶相同

表 32. METERLIBFFT_Interpolation 带 MMAU 的 CM0+ 内核的函数性能

函数名称	代码大小 [B]			堆栈大小 [B][1]	时钟周期[2]		
	1 阶	2 阶	3 阶		1 阶	2 阶	3 阶
METERLIBFFT1PH_Interpolation	940	1116	1560	512	7388	15543	28304
METERLIBFFT2PH_Interpolation	1020	1196	1640		14728	30942	56607
METERLIBFFT3PH_Interpolation	1116	1292	1736		22067	46533	84911

1 由于示例是欠采样 （输入样本 < FFT 样本）

2 输入样本数 = 120，所需 FFT 点数 = 64，两个通道的插值阶相同

5.14 METERLIBFFT_SetCalibCoeff

这些函数用于设置所有电压和电流校准系数（增益因数）。这些系数的绝对值取决于硬件拓扑（传感器、AFE）。因此，部分库函数最终的计算精度完全取决于这些系数的正确设置。这些系数应解读为最大 AFE 范围（24 位 AFE 范围）内有效的最大峰值电压或电流值。如果未使用电流偏移量校正，则应将此偏移量指针分配给 NULL（在此情况下无真实 I_{RMS} 计算）。

5.14.1 句法

```
#include"meterlibfft.h"

long METERLIBFFT1PH_SetCalibCoeff (tMETERLIBFFT1PH_DATA *p, double u_max, double i_max, Frac24
*i_offs, double p_offs, double q_offs);

long METERLIBFFT2PH_SetCalibCoeffPh1 (tMETERLIBFFT2PH_DATA *p, double u_max, double i_max,
Frac24 *i_offs, double p_offs, double q_offs);

long METERLIBFFT2PH_SetCalibCoeffPh2 (tMETERLIBFFT2PH_DATA *p, double u_max, double i_max,
Frac24 *i_offs, double p_offs, double q_offs);

long METERLIBFFT3PH_SetCalibCoeffPh1 (tMETERLIBFFT3PH_DATA *p, double u_max, double i_max,
Frac24 *i_offs, double p_offs, double q_offs);

long METERLIBFFT3PH_SetCalibCoeffPh2 (tMETERLIBFFT3PH_DATA *p, double u_max, double i_max,
Frac24 *i_offs, double p_offs, double q_offs);

long METERLIBFFT3PH_SetCalibCoeffPh3 (tMETERLIBFFT3PH_DATA *p, double u_max, double i_max,
Frac24 *i_offs, double p_offs, double q_offs);
```

5.14.2 参数

表 33. METERLIBFFT_SetCalibCoeff 函数参数

类型	名称	方向	说明
tMETERLIBFFT1PH_DATA	p	输入	指向单相计量库数据结构的指针。
tMETERLIBFFT2PH_DATA	p	输入	指向双相计量库数据结构的指针。
tMETERLIBFFT3PH_DATA	p	输入	指向三相计量库数据结构的指针。
双精度浮点	u_max	输入	对于 AFE 满量程范围有效的峰值线路电压 [V]。
双精度浮点	i_max	输入	对于 AFE 满量程范围有效的峰值线路电流 [A]。
Frac24	i_offs	输入	指向电流偏移量（Q0.23 数据格式）的指针如果偏移量无需包括在 I_{RMS} 计算中，则使用 NULL。
双精度浮点	p_offs	输入	有功功率 (P) 偏移量校正值 (W)。如果 P 偏移量无需包括在有功功率计算中，则使用 0。
双精度浮点	q_offs	输入	无功功率 (Q) 偏移量校正值 (var) 如果 Q 偏移量无需包括在无功功率计算中，则使用 0。

5.14.3 返回

这些函数将返回以下代码之一：

- FFT_ERROR (positive) – 部分输入参数过大，可能发生溢出；正确的系数值应保持为 $(u\_max*i\_max) < (2^{31}/10000)$
- FFT_OK (zero) – 所有输入参数均正确

5.14.4 调用顺序

应首先在初始化部分调用这些强制函数。还可以在硬件校准过程中或之后调用它们。

5.14.5 性能

表 34. METERLIBFFT_SetCalibCoeff 函数性能

函数名称	CM0+ 内核		带 MMAU 的 CM0+ 内核	
	代码大小 [B]	时钟周期	代码大小 [B]	时钟周期
METERLIBFFT1PH_SetCalibCoeff	146	2130	146	2130
METERLIBFFT2PH_SetCalibCoeffPh1	146	2130	146	2130
METERLIBFFT2PH_SetCalibCoeffPh2				
METERLIBFFT3PH_SetCalibCoeffPh1	146	2130	146	2130
METERLIBFFT3PH_SetCalibCoeffPh2				
METERLIBFFT3PH_SetCalibCoeffPh3				

5.15 METERLIBFFT_SetEnergy

此宏用于设置所有能量计数器和清除所有提醒器。对于所有类型的计量拓扑（单相、双相、三相），只有一个宏执行相同操作。能量分辨率取决于 5.12 节，“METERLIBFFT_InitParam” 中所述函数设定的 *en_res* 参数。

5.15.1 句法

```
#include“meterlibfft.h”

METERLIBFFT_SetEnergy (p, whi, whe, varhi, varhe);
```

5.15.2 参数

表 35. METERLIBFFT_SetEnergy 宏参数

类型	名称	方向	说明
tMETERLIBFFT1PH_DATA, tMETERLIBFFT2PH_DATA, tMETERLIBFFT3PH_DATA	p	输入	指向其中一个计量库数据结构的指针
无符号长整型	whi	输入	导入有功能量值 （请参见表 27 获取值分辨率）
无符号长整型	whe	输入	导出有功能量值 （请参见表 27 获取值分辨率）
无符号长整型	varhi	输入	导入无功能量值 （请参见表 27 获取值分辨率）
无符号长整型	varhe	输入	导出无功能量值 （请参见表 27 获取值分辨率）

5.15.3 返回

此宏不会返回任何参数。

5.15.4 调用顺序

此宏主要用于初始化部分。

5.16 METERLIBFFT_GetRotation

此函数用于计算各相位之间（相 2 与相 1、相 3 与相 1、相 3 与相 2）的相位角，并且可返回旋转方向 （前向、反向）。此函数专用于三相市电。

5.16.1 句法

```
#include“meterlibfft.h”

long METERLIBFFT3PH_GetRotation(tMETERLIBFFT3PH_DATA *p, double *u12_ph, double *u13_ph, double *u23_ph);
```

5.16.2 参数

表 36. METERLIBFFT3PH_GetRotation 函数参数

类型	名称	方向	说明
tMETERLIBFFT3PH_DATA	p	输入	指向三相计量库数据结构的指针。
双精度浮点	u12_ph	输出	指向 2 相与 1 相夹角 （度）的指针。
双精度浮点	u13_ph	输出	指向 3 相与 1 相夹角 （度）的指针。
双精度浮点	u23_ph	输出	指向 3 相与 2 相夹角 （度）的指针。

5.16.3 返回

这些函数将返回以下输出状态之一：

- ROT_FORWARD (positive) – 旋转方向为顺时针 （或前向），即 1-2-3、 2-3-1 或 3-1-2。
- ROT_REVERSE (negative) – 旋转方向为逆时针 （或反向），即 2-1-3、 1-3-2 或 3-2-1。
- ROT_UNKNOWN (zero) – 因相位丢失无法识别旋转方向。

5.16.4 调用顺序

必须在主(FFT)计算过程完成之后调用此函数。必须在调用此函数之前完成单次强制参数初始化。

5.16.5 性能

表 37. METERLIBFFT3PH_GetRotation 函数性能

函数名称	CM0+ 内核		带 MMAU 的 CM0+ 内核	
	代码大小[1] [B]	时钟周期	代码大小[1] [B]	时钟周期
METERLIBFFT3PH_GetRotation	532	3982	524	2926

1 反正切查找表的代码大小 (8 KB) 不包括在内。

6 总结

本应用笔记介绍了如何使用 FFT 在计量应用中计算基本计量值。所介绍的算法简单、高度准确，它可以轻松集成到电子电表中，仅需在其输入中提供瞬时相位电压和电流样本。它设计用于具有固定或可调测量采样率的 Σ-Δ 或 SAR 转换器设备。基于 FFT 计量库的性能已在若干个功率计参考设计 [5][6] 中经过测试，对于高电流动态范围具有高精度。

在计量应用中，基于 FFT 的计算方法具有以下优势和劣势：

实现优势：

- 有功和无功能量具有相同精度 （因为使用的是同一个计算公式）
- 四象限有功和无功能量测量
- 市电频率分析，可用于计算总谐波失真 (THD)。
- 消除偏移量，因为功率计算时无零谐波。

实现劣势：

- 使用固定采样率 (Σ-Δ ADC) 时需要进行额外的插值过程。
- MCU 的计算功率增大 （需要使用 32 位 MAC 单元）。

7 参考文献

1. J.W.Cooley and J.W.Tukey, An algorithm for the machine calculation of the complex Fourier series, Math. Comp., Vol. 19 (1965), pp. 297-301
2. Wikipedia articles“Cooley-Tukey FFT algorithm”,“Complex number”,“AC Power”, “Power Factor”,“Total Harmonic Distortion”,“Q (number format)”, available at en.wikipedia.org
3. Fast Fourier Transform (FFT) article, available at www.cmlab.csie.ntu.edu.tw/cml/dsp/training/coding/transform/fft.html
4. *MQX-enabled MK30X256 Single-Phase Electricity Meter Reference Design* (document DRM122)
5. *Kinetis-M Two-Phase Power Meter Reference Design* (document DRM149)
6. *MKM34Z256 One-Phase Power Meter Reference Design* (document DRM163)
7. *Using the FFT on the Sigma-Delta ADCs* (document AN4847)
8. Fractional and Integer Arithmetic – DSP56000 Family of General-Purpose Digital Signal Processors, (Motorola 1993, USA)

8 修订历史记录

表 38. 修订历史记录

修订版本号	日期	重要改动
0	11/2011	初始版本
1	10/2013	更新了 5 节，“计量库”
2	11/2014	更新了 4.2 节，“均方根计算” 更新了 4.3 节，“复数功率计算” 增加了 4.4 节，“电能计算” 增加了 4.5 节，“功率系数计算” 增加了 4.6 节，“总谐波失真计算” 更新了 5 节，“计量库”
3	02/2015	更新了 4.2 节，“均方根计算” 更新了 5.14 节，“METERLIBFFT_SetCalibCoeff” 更新了图 7 更新了表 5- 表 32 中的时钟周期值
4	07/2015	更新了表 5- 表 37 中的时钟周期值 更新了图 7 更新了 5 节，“计量库” 增加了 5.1 节，“内核架构和编译器支持” 更新了 5.12 节，“METERLIBFFT_InitParam” 更新了 5.13 节，“METERLIBFFT_Interpolation” 更新了 5.14 节，“METERLIBFFT_SetCalibCoeff” 增加了 5.16 节，“METERLIBFFT_GetRotation” 更新了 6 节，“总结” 更新了 7 节，“参考文献”

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

本文档中的信息仅供系统和软件实施方使用 Freescale 产品。本文并未明示或者暗示授予利用本文档信息进行设计或者加工集成电路的版权许可。Freescale 保留对此处任何产品进行更改的权利，恕不另行通知。

Freescale 对其产品在任何特定用途方面的适用性不做任何担保、表示或保证，也不承担因为应用程序或者使用产品或电路所产生的任何责任，明确拒绝承担包括但不局限于后果性的或附带性的损害在内的所有责任。Freescale 的数据表和 / 或规格中所提供的 “ 典型 ” 参数在不同应用中可能并且确实不同，实际性能会随时间而有所变化。 所有运行参数，包括 “ 经典值 ” 在内，必须经由客户的技术专家对每个客户的应用程序进行验证。Freescale 未转让与其专利权及其他权利相关的许可。Freescale 销售产品时遵循以下网址中包含的标准销售条款和条件：
freescale.com/SalesTermsandConditions.

Document Number: AN4255
Rev. 4, 07/2015