提升 dsPIC33AK 器件的 ADC 性能

AN5971

简介

本应用笔记指导用户如何使用校准和补偿技术提升 dsPIC33AK 系列器件中的 ADC 性能。文中讨论了 ADC 误差类型和校正方法，并概述了内置 ADC 校准功能。本文档提供了通过测量并补偿增益误差提高精度的步骤和详细信息。

1. ADC 误差类型

ADC 的性能和精度在具体器件数据手册的“电气特性”部分定义。典型的精度特性包括失调误差、增益误差、微分非线性（Differential Nonlinearity，DNL）误差和积分非线性（Integral Nonlinearity，INL）误差。基于这四项特性，可以计算出绝对误差。

1.1. 测试与测量方法

测试方法和电气特性基于线性斜坡直方图。对于 12 位 ADC，理想传递函数定义如下：

$$floor\left(\frac{VIN}{VDD} \times 4096\right)$$

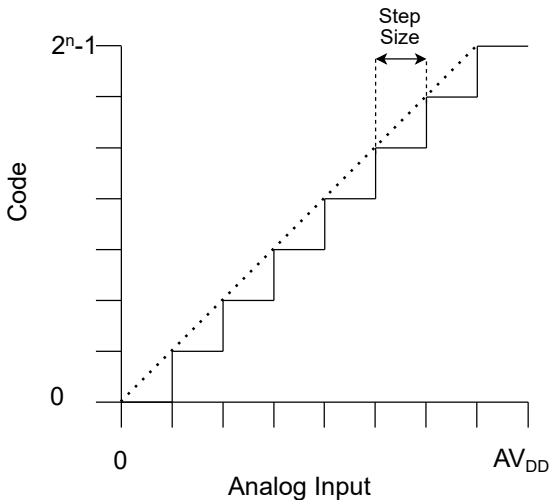
每个 ADC 计数对应输入范围的 1/4096。

当 AV_{DD} 标称值为 3.3V 时，输入范围为 0V 至 3.3V。

一个理想 ADC 计数对应的步长为$\frac{3.3V}{4096} = 0.805mV$。

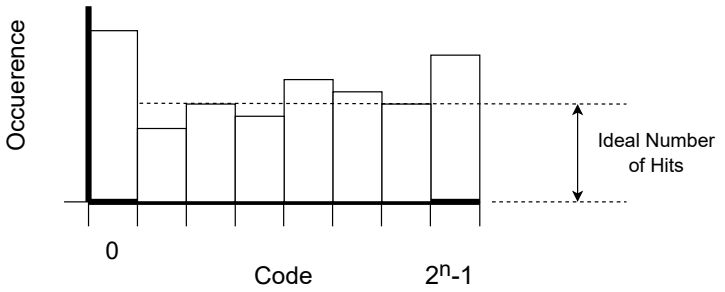
图 1-1 显示了理想传递函数。当模拟输入介于 0V 与 1 个步长之间时，理想传递函数中的 floor()生成的输出代码为 0。当模拟输入介于 AV_{DD} 减去 1 个步长与 AV_{DD} 之间时，对应的输出代码为最大值 4095。

图 1-1. 理想传递函数



线性斜坡直方图方法使用了与每个输出代码相对应的 bin。为涵盖误差，在限值之外还包含了额外的 bin。代表实际传递函数两端的两个 bin 不参与计算。低端的 bin 可表示失调误差。图 1-2 给出了相应的直方图示例。测量每个 bin 中的命中次数，并将其与理想命中次数进行比较。随后使用每个 bin 中的命中比例来映射传递函数。所有规范值的单位均为 LSB。

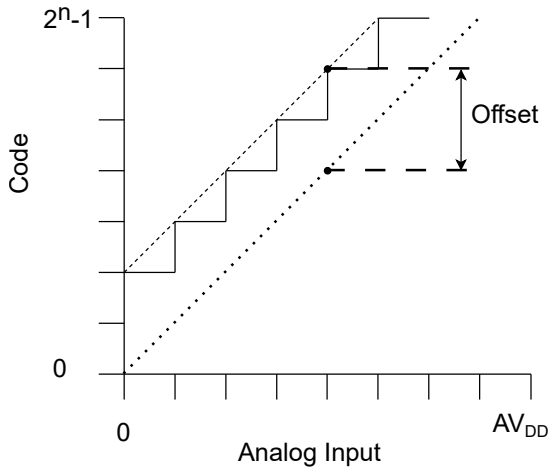
图 1-2. 直方图



1.2. 失调误差

失调误差的定义为实测传递函数中点与理想传递函数中点之间的偏差。计算失调误差时，通过直方图命中次数来确定实测传递函数的中心点。计算时需排除实测传递函数高低两端的各 256 个 bin。实测中间代码与 n/2 的差值即为失调误差。使用实测中心 bin 的命中次数可获得高于 1 LSB 的分辨率。失调误差可正亦可负。测量负失调误差需要施加低于 0V 的激励电压。失调误差通过对输出值加减失调误差值进行校正。图 1-3 给出了正失调误差的示例。

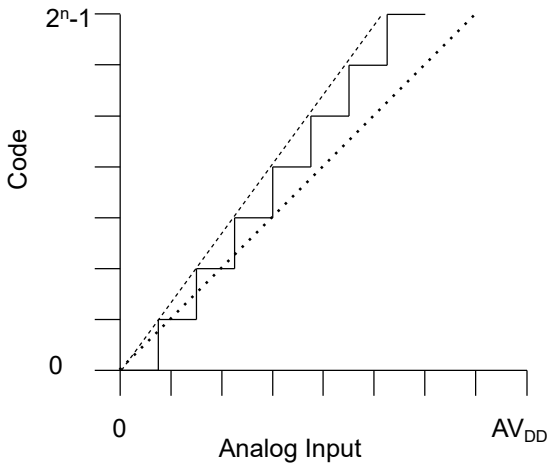
图 1-3. 正失调误差示例



1.3. 增益误差

增益误差的定义为在补偿失调误差后，实测传递函数斜率相对于理想传递函数斜率的偏差。测量方法为使用直方图的 bin 256 和 bin n-256 进行两点线性截距计算。增益误差通过缩放输出值进行校正。 图 1-4 给出了正增益误差的示例。

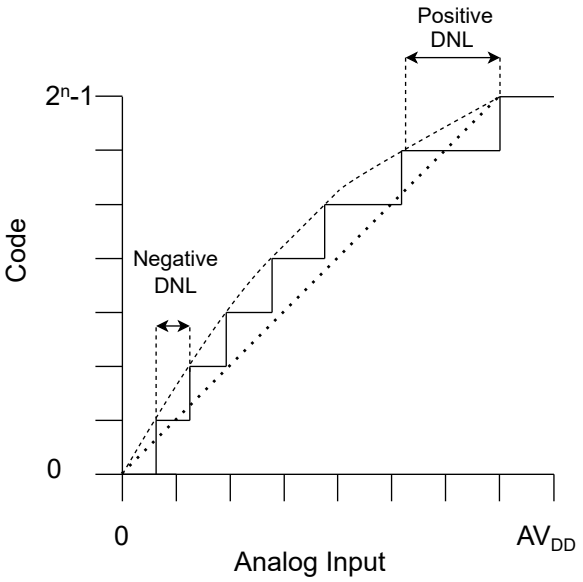
图 1-4. 正增益误差示例



1.4. 微分非线性（DNL）误差

DNL 的定义为实测传递函数的步长与理想步长 1 LSB 之间的差值。DNL 在增益误差和失调误差均校正后进行计算。非线性会导致量化步长不一致。具体器件数据手册的“电气特性”部分定义的 DNL 值代表 DNL 的限值。 图 1-5 给出了 DNL 步长误差的示例。

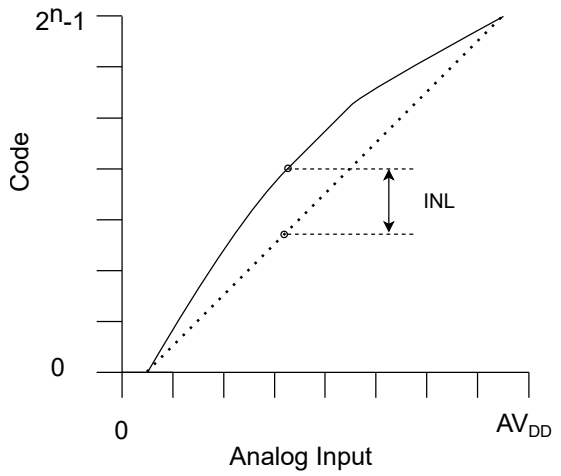
图 1-5. 传递函数 DNL 误差示例



1.5. 积分非线性（INL）误差

INL 的定义为实测传递函数与线性传递函数之间的垂直差。INL 通过对已校正增益误差和失调误差的 DNL 值进行累加求和计算得出。具体器件数据手册的“电气特性”部分定义的 INL 规范值代表 INL 曲线偏差的限值。 图 1-6 给出了 INL 误差曲线的示例。

图 1-6. 传递函数 INL 误差示例



2. dsPIC33AK ADC 内置校准功能

dsPIC33AK ADC 内置校准功能，包括可配置的失调校准。使能该功能后，ADC 将自动执行启动校准过程，其中包括增益校准和失调校准。自动应用的增益误差校正可能无法满足应用对精度的要求。ADC 性能可通过下一章所述的增益误差补偿得到进一步提升。在运行期间，可通过两种方法执行精细失调校准：应软件请求，或者利用调度功能定期执行。

有关失调校准的详细信息，请参见具体器件数据手册的 ADC 部分。

3. 增益误差补偿

ADC 在启动时完成自动校准过程后，其增益误差可能相对于上一次启动校准周期发生偏移。为提高精度，dsPIC33AK 器件内置参考电压源以支持运行时增益误差补偿。自动启动校准过程完成后，可运行一次提供的软件增益校准程序并将补偿系数应用于 ADC 结果。dsPIC33AK 系列器件具有多个 ADC 实例，每个 ADC 实例都需要独立进行校准。

3.1. 增益补偿方案

增益误差补偿基于两点线性校正方案。鉴于 ADC 已自动执行失调校准，因此将零点与接近满量程的参考电压结合使用。其中一个 ADC 输入提供了内部 15/16 * AV_{DD} 校准参考电压（CalrefH）。建议在测量 CalrefH 时使用过采样模式以提高精度。一次只能将一个 ADC 实例连接到 CalrefH。CalrefH 源的采样时间要求在具体器件数据手册的“电气特性”部分定义。转换完成后，即可计算补偿系数并将其存入 RAM。随后将补偿系数应用于应用的 ADC 结果。下面总结了操作步骤：

对于每个 ADC 实例：

1. 通过读取 ADRDY 位验证自动校准是否完成。
2. 选择任一 ADC 通道并将输入配置为 CalrefH 源。
3. 使用过采样模式测量 CalrefH 电压。
4. 计算补偿系数并将其存入 RAM。
5. 将关联的 ADC 补偿系数应用于结果数据。

如果后续执行失调校准，增益误差校准仍然有效，无需重新执行。但是，如果器件复位或 ADC 复位，则需再次执行增益误差校准程序。

3.2. 补偿系数计算

公式 3-1 为补偿系数公式。

公式 3-1. 增益误差补偿系数计算

$$\text{增益误差补偿系数} = \frac{\frac{15}{16} \times 4096}{\frac{15}{16} \times CalrefH\_result} = \frac{3840}{CalrefH\_result}$$

3.3. 补偿系数的应用

为补偿增益误差，需将每个 ADC 结果乘以计算出的系数。可使用定点或浮点乘法来执行补偿。浮点计算如例 3-1 所示，耗时 21 个指令周期。因此，当 CPU 时钟频率为 200 MHz 时，需要长达 105 ns 的时间。

例 3-1. 使用浮点计算校正 ADC 转换结果

```
float coefficient = 3840.0/CalrefH_result; // 每个 ADC 实例只需执行一次
……
// 耗时 21 个指令周期，即 CPU 时钟为 200 MHz 时耗时 105 ns
float corrected_result = (coefficient*((float)AD1CH0DATA));
```

使用定点计算时，乘法时间可缩短至 6 个周期（30 ns），如例 3-2 所示。请注意，左移 18 位是为了将系数放大为整数值并最大限度地减小舍入误差。校正后的结果有两种处理方式：重新归一化（右移）为 12 位值，或者保留较大数值格式带来的额外 LSB 精度。移位（截断）回 12 位会引入舍入误差，该误差将表现为结果中的 DNL 误差。为充分利用补偿方案所提升的精度，建议保留额外的精度，但必须在后续使用中对此加以考虑。为简单起见并与浮点计算保持一致，示例中显示的是截断方法。

例 3-2. 使用定点计算校正 12 位无符号 ADC 转换结果

```
int32_t coefficient = (uint32_t) ((1<<18)*3840.0/CalrefH_result; // 只需执行一次
// 耗时 6 个指令周期，即 CPU 时钟为 200 MHz 时耗时 30 ns
uint32_t corrected_result = (coefficient*AD1CH0DATA)>>18;
```

3.4. 代码示例

例 3-3 包含对一个 ADC 实例执行增益误差补偿所需的全部步骤和信息。这里以 ADC 1 的通道 0 为例，但对其他通道同样适用。

例 3-3. dsPIC33AK 器件的增益误差补偿代码示例

```
#include <xc.h>

int32_t result = 0;  // ADC 转换结果输出。
int32_t coefficient; // 增益补偿系数。

void OscillatorInitialization(); // 振荡器初始化程序。

int main(){

   OscillatorInitialization();  // 初始化振荡器。
   AD1CONbits.ON = 1; // 使能 ADC。
   while(AD1CONbits.ADRDY == 0); // 等待 ADC 就绪/校准完成

   ////////////////////////////////////////////////////////////////////
   // 获取增益误差补偿系数
   ////////////////////////////////////////////////////////////////////
   AD1CH0CONbits.MODE = 3; // 选择过采样模式
   AD1CH0CONbits.ACCNUM = 3; // 256 次转换
   AD1CH0CONbits.TRG1SRC = 1; // 软件触发信号将启动转换
   AD1CH0CONbits.TRG2SRC = 2; // 连续转换
   AD1CH0CONbits.PINSEL = 14; // 选择连接到 AVDD 的 15/16 的 AN14 输入
   AD1CH0CONbits.SAMC = 3; // 采样时间（6.5 个 TAD = 81 ns @ 40 MHz ADC 时钟）
   AD1SWTRGbits.CH0TRG = 1; // 对 256 个参考电压结果计算平均值
   while(AD1STATbits.CH0RDY == 0); // 等待结果就绪

   // 过采样结果为 16 位（有额外的 4 位）
   // 计算增益补偿系数
   // 系数为定点格式（小数点前 18 位）
   coefficient = (int32_t)(3840.0*16.0*(1<<18)/AD1CH0DATA);

   ////////////////////////////////////////////////////////////////////
   // 转换并补偿增益误差
   ////////////////////////////////////////////////////////////////////
   AD1CH0CON = 0; // 清空通道寄存器以存储新设置
   AD1CH0CONbits.MODE = 0; // 选择单次转换模式
   AD1CH0CONbits.TRG1SRC = 1; // 软件触发信号将启动转换
   AD1CH0CONbits.PINSEL = 7; // 选择 AN7 输入进行转换
   AD1CH0CONbits.SAMC = 3; // 采样时间（6.5 个 TAD = 81 ns @ 40 MHz ADC 时钟）

   // 在软件中触发通道 1 并等待结果
   while(1){
      AD1SWTRGbits.CH0TRG = 1; // 触发通道 1
      while(AD1STATbits.CH1RDY == 0); // 等待转换就绪标志
      // 读取结果。这将清零转换就绪标志
      // 校正系数为定点格式（小数点前 18 位）
      result = (coefficient*AD1CH0DATA)>>18;
      }
   return 1;
}

void OscillatorInitialization(){
   // 时钟发生器 6 应为 ADC 提供 320 MHz 时钟
   PLL1CONbits.ON = 1;
   OSCCTRLbits.PLL1EN = 1;
   while(OSCCTRLbits.PLL1RDY == 0);
   PLL1CONbits.FSCMEN = 0; // 禁止时钟故障监视器
   VCO1DIVbits.INTDIV = 1; // 1:2 = 320 MHz
   PLL1DIVbits.PLLFBDIV = 80; // VCO = 640 MHz
   PLL1DIVbits.PLLPRE = 1;
   PLL1DIVbits.POSTDIV1 = 4;
   PLL1DIVbits.POSTDIV2 = 1;
   PLL1CONbits.DIVSWEN = 1;
   while(PLL1CONbits.DIVSWEN == 1);
   PLL1CONbits.NOSC = 1; // FRC
```

```
    PLL1CONbits.OSWEN = 1;
    while(PLL1CONbits.OSWEN == 1);
    PLL1CONbits.FOUTSWEN = 1;
    while(PLL1CONbits.FOUTSWEN == 1);
    PLL1CONbits.PLLSWEN = 1;
    while(PLL1CONbits.PLLSWEN == 1);
    while(PLL1CONbits.CLKRDY == 0);
    CLK1CONbits.NOSC = 5; // PLL1
    CLK1CONbits.OSWEN = 1;
    while(CLK1CONbits.OSWEN == 1);
    while(CLK1CONbits.CLKRDY == 0);
    // ADC 高速时钟（发生器 6），应为 320 MHz 以支持 80 MHz 工作频率
    CLK6CONbits.ON = 1;
    CLK6CONbits.NOSC = 7; // PLL1 VCO 分频比
    CLK6CONbits.OSWEN = 1;
    while(CLK6CONbits.OSWEN == 1);
    while(CLK6CONbits.CLKRDY == 0);
}
```

4. 增益和失调补偿

为进一步提升性能，可再使用一个非零参考电压测量值来进行增益和失调补偿。由于可能存在无法测量的负失调误差，因此不能使用零点。为此，可将外部电阻分压器与另一个 ADC 输入结合使用。建议的参考电压为 $AV_{DD}/16$ 或更低（CalrefL）。低于 $AV_{DD}/16$ 的值可为元件容差提供额外的稳健性。但是，必须注意确保参考电压不低于最大绝对误差，以使最终 ADC 结果始终至少为一个计数。补偿系数通过公式 4-1 计算得出，其中使用 $1/16*AV_{DD}$ 作为 CalrefL。

公式 4-1. 双参考电压增益和失调误差计算

$$\text{增益误差补偿系数} = \frac{\frac{14}{16} \times 4096}{(CalrefH\_result - CalrefL\_result)}$$

$$\text{失调误差} = CalrefL\_result \frac{\frac{1}{16} \times 4096}{\text{增益误差补偿系数}}$$

5. 结果与结论

使用本文档中所述的补偿技术后，dsPIC33AK ADC 的精度得以提升，超越了其自动校准功能本身提供的性能。运行时补偿无需外部参考电压即可提供一致的增益误差性能。文中推荐的方法、公式和代码示例提供了一个经过验证的解决方案，可集成到需要更高性能的应用中。

6. 版本历史

版本历史部分汇总了文档中所做的更改，并按照出版物版本从新到旧的顺序列出。

版本	日期	说明
A	2025 年 5 月	初始版本

Microchip 信息

商标

“Microchip”的名称和徽标组合、“M”徽标及其他名称、徽标和品牌均为 Microchip Technology Incorporated 或其关联公司和/或子公司在美国和/或其他国家或地区的注册商标或商标（“Microchip 商标”）。有关 Microchip 商标的信息，可访问 https://www.microchip.com/en-us/about/legal-information/microchip-trademarks。

ISBN： 979-8-3371-2664-7

法律声明

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc.及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc.的英文原版文档。

本出版物及其提供的信息仅适用于 Microchip 产品，包括设计、测试以及将 Microchip 产品集成到您的应用中。以其他任何方式使用这些信息都将被视为违反条款。本出版物中的器件应用信息仅为您提供便利，将来可能会发生更新。您须自行确保应用符合您的规范。如需额外的支持，请联系当地的 Microchip 销售办事处，或访问 www.microchip.com/en-us/support/design-help/client-support-services。

Microchip“按原样”提供这些信息。Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对非侵权性、适销性和特定用途的适用性的暗示担保，或针对其使用情况、质量或性能的担保。

在任何情况下，对于因这些信息或使用这些信息而产生的任何间接的、特殊的、惩罚性的、偶然的或附带的损失、损害或任何类型的开销，Microchip 概不承担任何责任，即使 Microchip 已被告知可能发生损害或损害可以预见。在法律允许的最大范围内，对于因这些信息或使用这些信息而产生的所有索赔，Microchip 在任何情况下所承担的全部责任均不超出您为获得这些信息向 Microchip 直接支付的金额（如有）。如果将 Microchip 器件用于生命维持和/或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切损害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任。除非另外声明，在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

Microchip 器件代码保护功能

请注意以下有关 Microchip 产品代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术规范。
- Microchip 确信：在正常使用且符合工作规范的情况下，Microchip 系列产品非常安全。
- Microchip 注重并积极保护其知识产权。严禁任何试图破坏 Microchip 产品代码保护功能的行为，这种行为可能会违反《数字千年版权法案》（Digital Millennium Copyright Act）。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。