AN2406

应用笔记

UM32x42x Cordic 配置指南

版本：V1.0.0

UNICMICRO
广芯微电子

广芯微电子（广州）股份有限公司

http://www.unicmicro.com/

条款协议

目录

1 摘要

本篇应用笔记主要介绍 UM32x42x CORDIC 的功能以及配置方法。

本篇应用笔记主要包括:

- CORDIC 概述
- CORDIC 配置流程
- 使用参考例程

注: 具体功能及寄存器的操作等相关事项请以用户手册为准。

2 CORDIC 概述

2.1 前言

本文档主要介绍了 UM32x42x 系列器件中三角数学单元 CORDIC 的功能配置及使用注意事项。其目的是帮助使用 UM32x42x MCU 开发者正确、快速地使用 CORDIC，从而缩短开发周期。

CORDIC（Coordinate Rotation Digital Computer）硬件加速运算协处理器，是一个完全可配置的模块，可执行常见的三角运算和算术运算操作。CORDIC 可以减轻 CPU 的负担，通常应用于电机控制、信号处理及其他应用场景。

CORDIC 可以计算多种函数，输入和输出数据支持 q1.31、q1.15 定点格式。

主要特性:

1. 内部使用 24 位精度计算。
2. AHB 接口可以输入输出 32 或 16 位数据。
3. 支持 DMA 方式传输。
4. 输入输出数据地址可配置。

2.2 CORDIC 介绍

UM32x42x 中的 CORDIC 通过坐标旋转数字计算的算法来求解三角函数及其他复杂的数学函数。CORDIC 算法通过迭代方式近似向量旋转，将旋转操作分解为一系列小的旋转步骤，每一步仅涉及简单的移位、加法或减法运算，因此非常适合硬件实现。

CORDIC_CTRL 寄存器中的[7:4] 位域用于配置 CORDIC 算法核心的迭代次数。迭代次数越多，计算精度越高。

2.3 模式配置

CORDIC_CTRL 寄存器中的 MODE[2:0]位域用于配置 CORDIC 算法核心的工作模式，支持 8 种计算模式。详细配置请参见**表 2-1：模式配置**。

表 2-1：模式配置

模式	输入数据 1	输入数据 2	输出数据 1	输出数据 2
模式 0	θ	m	$m \cdot \sin\theta$	$m \cdot \cos\theta$
模式 1	y	x	atan2(y, x)	$\sqrt{x^2 + y^2}$
模式 2	y	x	$y \cdot x$	-
模式 3	y	x	y/x	-
模式 4	$2^{-1} \cdot w$	-	$2^{-1} \cdot \sinh w$	$2^{-1} \cdot \cosh w$
模式 5	y	x	$2^{-1} \cdot \theta$	-
模式 6	x $x \in [0.1069, 1)$	-	$2^{-2} \cdot \ln x$	-
	$2^{-1} \cdot x$ $x \in (1, 3)$	-	$2^{-2} \cdot \ln x$	-
	$2^{-2} \cdot x$ $x \in [3,7)$	-	$2^{-2} \cdot \ln x$	-
	$2^{-3} \cdot x$ $x \in [7, 9.35]$	-	$2^{-2} \cdot \ln x$	-
模式 7	x $x \in [0.1069, 1)$	-	$\sqrt{x}$	-
	$2^{-1} \cdot x$ $x \in (1, 3)$	-	$2^{-1} \cdot \sqrt{x}$	-
	$2^{-2} \cdot x$ $x \in [3,7)$	-	$2^{-2} \cdot \sqrt{x}$	-
	$2^{-3} \cdot x$ $x \in [7, 9.35]$	-	$2^{-3} \cdot \sqrt{x}$	-

尽管 CORDIC 算法只能直接计算少量函数，但可以通过间接方法获得更多函数。例如，$\tan\theta = \sin\theta \div \cos\theta$。

2.4 数据格式与配置

UM32x42x 中的 CORDIC 的输入和输出数据可配置为 q1.31、q1.15 定点格式。

每种模式需要不同数量的输入数据，如模式 0 需要两个输入数据，模式 4 只需要一个输入数据。

输入数据的模式配置，根据 CORDIC_CTRL 寄存器的 WIDTH_IN 位确定 q1.15、q1.31 定点格

式，MERGE_IN 来确定 q1.15 的输入数据方式，ADDR_IN 位来确定 q1.31 的输入数据方式。

输入数据方式为两种:

1) 两个输入数据分别写入 CORDIC_DIN1、CORDIC_DIN2 寄存器。

2) 两个输入数据按顺序写入 CORDIC_DIN1。

详细请见表 2-2：输入数据配置。

注意：当输入数据配置为 q1.15 格式，输入两个参数的模式时，只写一次 CORDIC_DIN1 寄存器，CORDIC_DIN1 的第一个输入数据位于低半字，第二个输入数据位于高半字。如果模式仅需一个输入数据，则仅使用低半字，高半字不使用。

表 2-2：输入数据配置

WIDTH_IN 位	MERGE_IN 位	ADDR_IN 位	数据格式	输入寄存器的写操作
1	0	0	q1.15 定点	写 CORDIC_DIN1 写 CORDIC_DIN2
		1	q1.15 定点	写 CORDIC_DIN1 写 CORDIC_DIN1
1	1	-	q1.15 定点	写 CORDIC_DIN1
0	-	0	q1.31 定点	写 CORDIC_DIN1 写 CORDIC_DIN2
		1	q1.31 定点	写 CORDIC_DIN1 写 CORDIC_DIN1

每种模式输出数据的数据数量不同，如模式 0 输出两个输出数据，模式 2 输出一个输出数据。

输出数据的模式配置，根据 CORDIC_CTRL 寄存器的 WIDTH_OUT 位确定 q1.15、q1.31 定点格式，MERGE_OUT 来确定 q1.15 的输出数据方式，ADDR_OUT 位来确定 q1.31 的输出数据方式。输出数据方式为两种:

1) 两个输出数据 CORDIC_DOUT1、CORDIC_DOUT2 寄存器分别读出数据;

2) 两个输出数据按顺序从 CORDIC_DOUT1 读出。

详细请见表 2-3：输出数据配置。

注意：当输出数据配置为 q1.15 格式，输出两个数据的模式时，只读取一次 CORDIC_DOUT1 寄存器，CORDIC_DOUT1 寄存器的第一个输出数据位于低半字，第二个输入数据位于高半字。如

果模式仅需一个输入数据，则仅使用低半字，高半字不使用。

表 2-3：输出数据配置

WIDTH_OUT 位	MERGE_OUT 位	ADDR_OUT 位	数据格式	输出寄存器的读操作
1	0	0	q1.15 定点	读 CORDIC_DOUT1 读 CORDIC_DOUT2
		1	q1.15 定点	读 CORDIC_DOUT1 读 CORDIC_DOUT1
1	1	-	q1.15 定点	读 CORDIC_DOUT1
0	-	0	q1.31 定点	读 CORDIC_DOUT1 读 CORDIC_DOUT2
		1	q1.31 定点	读 CORDIC_DOUT1 读 CORDIC_DOUT1

3 CORDIC 配置流程

3.1 CORDIC 算法使用流程

1. 使能 CORDIC 模块时钟和释放复位。
2. 配置输入输出地址格式、数据格式等。
3. 配置运算次数（迭代次数越大运算结果越精确）。
4. 配置运算方式、输入输出数据传输方式。
5. 根据配置的运算方式输入合适的数据到 CORDIC_DIN1 和 CORDIC_DIN2。
6. 等待 CORDIC_CTRL[31] DATA_READY 位置位。
7. 读取两个输出数据 CORDIC_DOUT1 和 CORDIC_DOUT2。

4 使用参考例程

本例程 CORDIC 在 UM32x42x 平台上实现，具体可参考 UM32x42x_SDK Examples 中的 CORDIC 应用例程。

4.1 CORDIC 的模式、输入输出格式配置

以下简单介绍如何使用 hal 库对 CORDIC 的算法配置及使用过程。

配置 CORDIC 工作在某种模式，输入输出数据格式及迭代次数。

```
void Cordic_Calculate(void)
{
    float cordic_result;
    hcordic.Instance = CORDIC;

    hcordic.Init.Function = CORDIC_FUNCTION_MODE0;
    hcordic.Init.Iteration = CORDIC_PRECISION_12CYCLES;
    hcordic.Init.Scale = CORDIC_SCALE_0;

    hcordic.Init.WIDTH_IN = CORDIC_INSIZE_32BITS;
    hcordic.Init.WIDTH_OUT = CORDIC_OUTSIZE_32BITS;

    hcordic.Init.ADDR_IN = CORDIC_ADDR_IN_2;
    hcordic.Init.ADDR_OUT = CORDIC_ADDR_OUT_2;

    hcordic.Init.MERGE_IN = CORDIC_MERGE_IN_2;
    hcordic.Init.MERGE_OUT = CORDIC_MERGE_OUT_2;

    if(HAL_CORDIC_Init(&hcordic) != HAL_OK)
```

模式
迭代次数
输入q1.31
输出q1.31
两次写入寄存器
两次读出数据

图 4-1：CORDIC 模式、输入输出格式配置

4.2 CORDIC 的数据输入输出

输入数据进行计算并读出数据，获得算法结果。

```
/* Write of input data in Write Data register, and increment input buffer pointer */
CORDIC_WriteInDataIncrementPtr(hcordic, &p_tmp_in_buff);

while (HAL_IS_BIT_CLR(hcordic->Instance->CTRL, CORDIC_CTRL_DATA_READY))
{
    /* Check for the Timeout */
    if (Timeout != HAL_MAX_DELAY)
    {
        if ((HAL_GetTick() - tickstart) > Timeout)
        {
            /* Set CORDIC error code */
            hcordic->ErrorCode = HAL_CORDIC_ERROR_TIMEOUT;

            /* Change the CORDIC state */
            hcordic->State = HAL_CORDIC_STATE_READY;

            /* Return function status */
            return HAL_ERROR;
        }
    }
}

/* Read output data from Read Data register, and increment output buffer pointer */
CORDIC_ReadOutDataIncrementPtr(hcordic, &p_tmp_out_buff);
```

写入数据

查询ready状态

读出数据

图 4-2：CORDIC 数据输入输出示例

5 注意事项

1. 功能模式 0：$m \cdot \sin\theta$ / $m \cdot \cos\theta$的注意点

$\sin\theta$、$\cos\theta$在坐标轴附近约 4.4°范围转换值不正确，可以对输入数据先处理一下再输入计算，如果输出的结果精度不够，可以修补一下输出结果。详细见下图，对 0°-90°区间内靠近坐标轴的数据进行输入预处理及输出精度补偿。

```
for(i=0; i<90; i=i+0.001)
{
  if(i<4.398)
  {
      tmpIn2 = i*11930464.7f;                                    //输入数据的处理
      Cordic_two_q31_write((int32_t)(tmpIn2), 0x7fffffff);

      while(!((CORDIC->CTRL>>31)&0x1));

      Cordic_two_q31_read();
      f32_out_data[0] = (int32_t)q31_out_data[0]/2147483648.0f;   //sin
      f32_out_data[1] = (int32_t)q31_out_data[1]/2147483648.0f;  //cos

      if((i>0) && (i<4.398))
          f32_out_data[1] = f32_out_data[1] + 0.00645;           //cos补偿精度
  }
  else
  {
      tmpIn1 = (180.0f-i)*11930464.7f;                          //输入数据的处理
      Cordic_two_q31_write((int32_t)(tmpIn1), 0x7fffffff);

      while(!((CORDIC->CTRL>>31)&0x1));

      Cordic_two_q31_read();

      f32_out_data[0] = (int32_t)q31_out_data[0]/2147483648.0f;   //sin
      f32_out_data[1] = -(int32_t)q31_out_data[1]/2147483648.0f;  //cos

      if(((i>85.601) && (i<90)))
          f32_out_data[0] = f32_out_data[0] + 0.00593;  //sin补偿精度
```

图 5-1：0°-90°区间内靠近坐标轴处数据的输入处理及输出精度补偿

2. 功能模式 2：$y \cdot x$的注意点

当 y=-1，x=-1 时，可以处理输入，也可以处理输出，正确值为 1，目前 CORDIC 计算出来的值为-1。

3. 功能模式 6：ln(x)、功能模式 7：Sqr(x)，输入输出数据格式根据输入数据的范围变化，需要在控制寄存器中输入数值范围 SCALE，SCALE 的缩放与范围使用注意点

1) 当$x \in (1,3)$时，配置 SCALE=1：

   a) 当$x \in (1,2]$时，缩放正常。

b) 当$x \in (2,3)$时，缩放后的数据转换成 q31 数据后越界了，这个时候需要对这个范围的数据进行输入处理后再转换成 q31 数据。

2) 当$x \in [3,7)$时，配置 SCALE=2：

a) 当$x \in [3,4]$时，缩放正常。

b) 当$x \in (4,7)$时，缩放后的数据转换成 q31 数据后越界了，这个时候需要对这个范围的数据进行输入处理后再转换成 q31 数据。

3) 当$x \in [7,9.35]$时，配置 SCALE=3：

a) 当$x \in [7,8]$时，缩放正常。

b) 当$x \in (8,9.35)$时，缩放后的数据转换成 q31 数据后越界了，这个时候需要对这个范围的数据进行输入处理后再转换成 q31 数据。

表 5-1：不同输入数据范围对应的 SCALE 缩放情况

X 取值范围		**缩放因子**	**备注**
$x \in [0.1069,1)$	-	SCALE=0	正常
$x \in (1,3)$	$x \in (1,2]$	SCALE=1	正常
	$x \in (2,3)$		需对输入数据进行处理
$x \in [3,7)$	$x \in [3,4]$	SCALE=2	正常
	$x \in (4,7)$		需对输入数据进行处理
$x \in [7,9.35]$	$x \in [7,8]$	SCALE=3	正常
	$x \in (8,9.35)$		需对输入数据进行处理

6 版本维护

版本	日期	描述
V1.0	2026.03.03	初始版