AN-897
应用笔记

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

超级时序控制器的ADC回读

作者：Enrico Del Mastro和Michael Bradley

简介

ADM1062、ADM1063、ADM1064、ADM1066、ADM1069、ADM1166和ADM1169系列是完全可编程的电源时序控制器和监控器，可以为采用多个电压源的系统提供完整的电源管理解决方案。

这些器件均集成一个片内12位ADC。ADC可设置以读取一个结果或连续读取选定通道。每个通道均提供均值功能并可任意启用(开启)或停用(关闭)。

本应用笔记介绍如何逐步设置并从ADC读取一个结果(均值功能开启或关闭)。此外还介绍如何逐步从ADC连续读取(均值功能开启或关闭)。本文在所提供指令集中以VH通道为例。

有关ADM1062、ADM1063、ADM1064、ADM1066、ADM1069、ADM1166和ADM1169器件的特性和功能详情，请参阅相关数据手册以及应用笔记AN-698和应用笔记AN-721，了解配置寄存器的详细信息。

目录

修订历史

2014年8月 — 修订版0至修订版A

增加ADM1166和ADM1169 ..... 通篇
RRCTL更改为RRCTRL ..... 通篇
更改标题、作者和简介 1
增加“ADC轮询使能指南”部分、“特定器件的考虑因素”部分、“全零值回读”部分和“静态值回读”部分 3

2007年6月—修订版0：初始版

ADC轮询使能指南

在Super Sequencer®超级时序控制器中，有三种方法可以使能ADC轮询(RR)。

- RRCTRL寄存器(寄存器0x82[0])中的GO位置1，触发单周期RR。
- RRCTRL寄存器(寄存器0x82[1])中的使能位置1，执行连续RR操作。
- RRCTRL寄存器(寄存器0x82[1])中的使能位在时序控制引擎状态中置位。

可以同时采用以上多种方法来使能ADC RR。然而，通常建议一次仅采用这些方法中的某一种来使能ADC RR。这三种使能方法以逻辑OR方式进行组合；因此，如果ADC RR在有效时序控制引擎状态中使能，则置位或清零使能位(寄存器0x82[1])对ADC RR的状态无影响。

若只使用一种方法来使能ADC RR，那么用户便可清楚地了解采样如何开始和停止。对于某些超级时序控制器而言，同样建议仅采用一种方法来使能ADC RR，以保证工作正常；详情参见"特定器件的考虑因素"部分。

特定器件的考虑因素

注意，这些特定器件考虑因素仅适用于ADM1062、ADM1063、ADM1064、ADM1066和ADM1069。它们不适用于ADM1166或ADM1169。

这些器件正常工作期间，某些条件下可观察到下列行为。这些行为的发生频率相对较低，与I²C总线处理的次数与频率有关。

全零值回读

回读不同通道的ADC RR值时，软件可能会报告所有通道从单次ADC RR迭代获取的零值或0x0000值。

发生这种回读时，软件必须等待下一个ADC RR周期完成(如果连续运行)，或者按需触发另一个单次ADC RR周期。下一个ADC RR周期会产生一组新值供使用。

只有通过I²C总线对ADC RR通道值的回读会受到影响；这一点很重要。用于警报和引导时序控制引擎的内部ADC RR寄存器将不受影响。

静态值回读

在某些配置下，ADC RR输出寄存器可能会报告静态值或不变的值，哪怕输入电源正在发生改变。发生这种情况说明ADC RR已停止周期操作，必须停止ADC RR并重启。

根据系统特性以及ADC RR的使能方式，ADC RR自动停止与重启操作也许可以作为时序控制引擎正常工作的一部分，或者用户可以采用外部处理器来实现软件临时解决方案。

如果ADC RR作为时序控制引擎状态的一部分来使能(比如电源良好状态的一部分)，则略微修改时序控制引擎的配置即可避免回读静态值。建议创建一种与初始状态完全一致的第二个电源良好状态，而不是只用一种电源良好状态来使能ADC RR。在第二个状态中，禁用ADC RR。采用较长的(400 ms)超时条件来实现从初始电源良好状态到新的电源良好状态的转换；采用较短的(0.1 ms)超时条件来回到初始电源良好状态。通过这种方式，ADC RR可以被连续使能和禁用，且读取的ADC值将永远不会是静态值。

如果使用单次模式ADC RR，则应在每次轮询之后检查GO位(寄存器0x82[0])，确定该位是否为1。如果为1，就必须向该位写入0。

如果使用连续ADC RR模式，软件必须检测回读值是否是静态的，如果是则停止和重启ADC RR。可通过将使能位(寄存器0x82[1])先设为0，然后再设为1来实现。

如果同时使用了超过一种方法来使能ADC RR，且输出值是静态的，则必须禁用所有已经使能的方法，以便ADC RR停止并重启。

单次读取(均值关闭)

1. 设置寄存器0x80和0x81(分别为RRSEL1和RRSEL2)。这些寄存器选择监控通道。
   0 = 选择通道
   1 = 未选择通道
   例如，若要选择VH通道，可向寄存器0x80写入0xEF，向寄存器0x81写入0x1F。
2. 将0x01写入寄存器0x82(RRCTRL)，设置GO位。
   Go = 1
   Enable = 0
   Average = 0
   STOPWRITE = 0
   CLEARLIM = 0
3. 超时进入环路并从寄存器0x82(RRCTRL)连续读取数据。
   - 如果寄存器0x82 = 0x00，则退出环路(GO位复位)。
   - 如果寄存器0x82 ≠ 0x00，则继续环路。
4. 如果GO位已复位，则向寄存器0x82 (RRCTRL)写入0x08。
   Go = 0
   Enable = 0
   Average = 0
   STOPWRITE = 1
   CLEARLIM = 0
5. 读取VH相关的寄存器。
   0xA8 (ADCHVH)
   0xA9 (ADCLVH)
6. 复位STOPWRITE位。将0x00写入寄存器0x82。
   Go = 0
   Enable = 0
   Average = 0
   STOPWRITE = 0
   CLEARLIM = 0

单次读取(均值开启)

1. 设置寄存器0x80和0x81(分别为RRSEL1和RRSEL2)。这些寄存器选择监控通道。
   0 = 选择通道
   1 = 未选择通道
   例如，若要选择VH通道，可向寄存器0x80写入0xEF，向寄存器0x81写入0x1F。
2. 将寄存器0x82 (RRCTRL)设置为0x05。
   Go = 1
   Enable = 0
   Average = 1
   STOPWRITE = 0
   CLEARLIM = 0
3. 超时进入环路并从寄存器0x82(RRCTRL)连续读取数据。
   - 如果寄存器0x82 = 0x04，则退出环路(GO位复位)。
   - 如果寄存器0x82 ≠ 0x04，则继续环路。
4. 如果GO位已复位，则向寄存器0x82 (RRCTRL)写入0x0C。
   Go = 0
   Enable = 0
   Average = 1
   STOPWRITE = 1
   CLEARLIM = 0
5. 读取VH相关的寄存器。
   0xA8 (ADCHVH)
   0xA9 (ADCLVH)
6. 复位STOPWRITE位。将0x04写入寄存器0x82。
   Go = 0
   Enable = 0
   Average = 1
   STOPWRITE = 0
   CLEARLIM = 0

连续读取(均值关闭)

1. 设置寄存器0x80和0x81(分别为RRSEL1和RRSEL2)。这些寄存器选择监控通道。
   0 = 选择通道
   1 = 未选择通道
   例如，若要选择VH通道，可向寄存器0x80写入0xEF，向寄存器0x81写入0x1F。
2. 将0x02写入寄存器0x82(RRCTRL)。
   Go = 0
   Enable = 1
   Average = 0
   STOPWRITE = 0
   CLEARLIM = 0
3. 将0x0A写入寄存器0x82(RRCTRL)。
   Go = 0
   Enable = 1
   Average = 0
   STOPWRITE = 1
   CLEARLIM = 0
4. 读取VH相关的寄存器。
   寄存器0xA8 (ADCHVH)
   寄存器0xA9 (ADCLVH)
5. 将0x02写入寄存器0x82(RRCTRL)。
   Go = 0
   Enable = 1
   Average = 0
   STOPWRITE = 0
   CLEARLIM = 0

连续读取(均值开启)

1. 设置寄存器0x80和0x81(分别为RRSEL1和RRSEL2)。这些寄存器选择监控通道。
   0 = 选择通道
   1 = 未选择通道
   例如，若要选择VH通道，可向寄存器0x80写入0xEF，向寄存器0x81写入0x1F。
2. 将0x06写入寄存器0x82(RRCTRL)。
   Go = 0
   Enable = 1
   Average = 1
   STOPWRITE = 0
   CLEARLIM = 0
3. 将0x0E写入寄存器0x82(RRCTRL)。
   Go = 0
   Enable = 1
   Average = 1
   STOPWRITE = 1
   CLEARLIM = 0
4. 读取VH相关的寄存器。
   0xA8 (ADCHVH)
   0xA9 (ADCLVH)
5. 将0x06写入寄存器0x82(RRCTRL)。
   Go = 0
   Enable = 1
   Average = 1
   STOPWRITE = 0
   CLEARLIM = 0

I^2C指最初由Philips Semiconductors(现为NXP Semiconductors)开发的一种通信协议。