TMC4361A 数据手册

TMC4361A 版本 1.24 • 2018 年 9 月 20 号

步进电机运动控制器，支持 S 型斜坡和 sixPoint 六点式斜坡，进行了高速优化，支持动态修改运动参数。TMC4361A 包含 SPI 接口、Step/Dir 接口及闭环所需的编码器接口。

注意:

→ *TMC4361A 是 TMC4361 的升级更新芯片。*

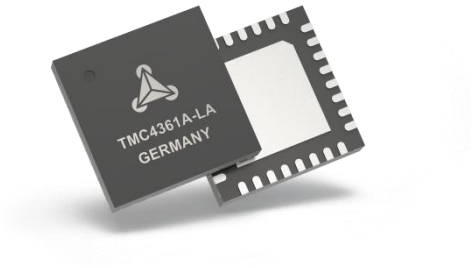


图 1: 芯片样品图
TMC4361A 闭环驱动
** 第 224 页解释了标记细节。*

特征

- 简单易用的与微处理器通讯的 SPI 接口。
- 与 SPI 步进电机驱动器通讯的 SPI 接口。
- 增量或串行编码器的 编码器接口。
- 脉冲及 SPI 步进驱动器的闭环控制。
- 集成的 ChopSync™ 及 dcStep™ 。
- 内部斜坡发生器产生可动态修改的 S 形斜坡或者 sixPoint™六点斜坡。
- 可控的 PWM 输出。
- 参考开关处理。
- 硬件和虚拟停止切换开关。
- 支持 TMC 步进电机驱动器 。

应用

- 纺织、缝纫机
- CCTV, 安防
- 打印机、扫描仪
- ATM, 现金回收机
- 办公自动化
- POS
- 工厂自动化
- 实验室自动化
- 泵和阀
- 定日镜控制器
- 数控机器
- 机器人

框图:TMC4361A 接口和特性

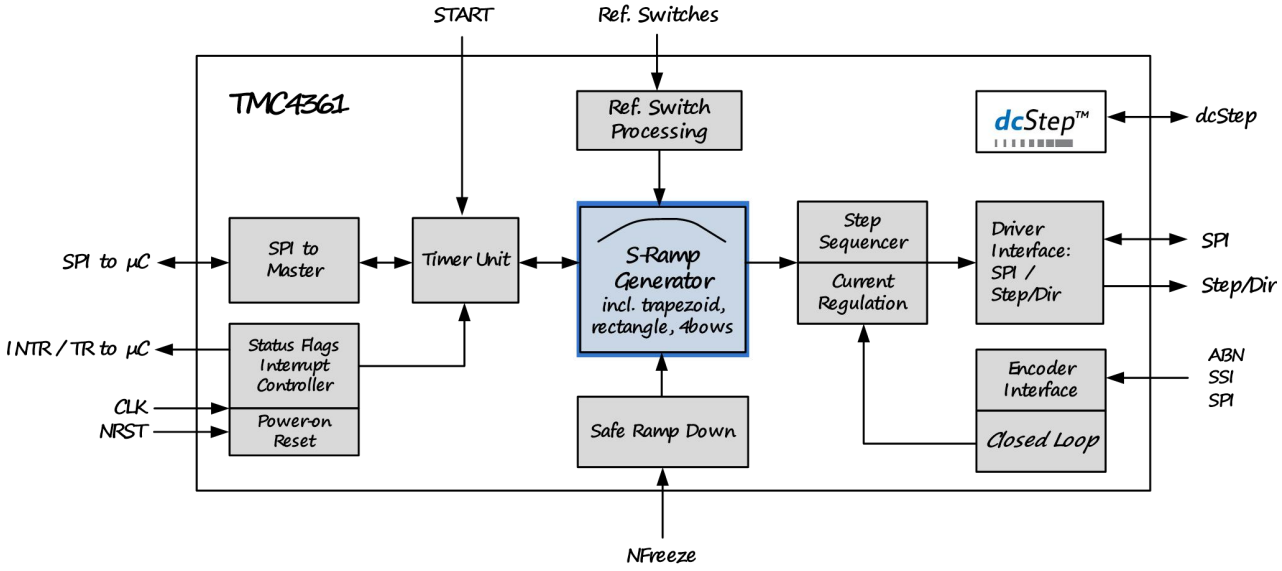


图 2: 功能图

TMC4361A 的功能范围

TMC4361A 是一款小型化、高性能的驱动步进电机的运动控制器。实用于很多的斜坡轮廓的应用，特别是速度快、限制过冲的运动场合。用户根据自己的要求实现 S 形或 sixPoint™六点式速度轮廓配置及闭环或开环的操作，如下所示:

S 形速度轮廓 由七个斜坡段构成的 S 形斜坡轮廓可以实现速度无跳变。用户根据需求优化调整。如本手册所述，通过校准斜坡弓形参数可实现高速高扭矩。

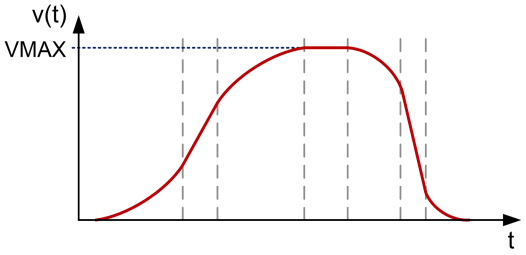


图 3: S 形速度轮廓

i 斜坡配置和其他速度轮廓，如 sixPoint™六点斜坡，的更多信息请参加第 6 章(第 28 页)。

闭环操作特性 TMC4361A 连接步进电机栅极驱动芯片 TMC262 实现闭环操作的典型框图如下。如果需要内部功率管 MOSFETs，则 TMC4361A 与 TMC2620、TMC261 或 TMC2660 配合使用。

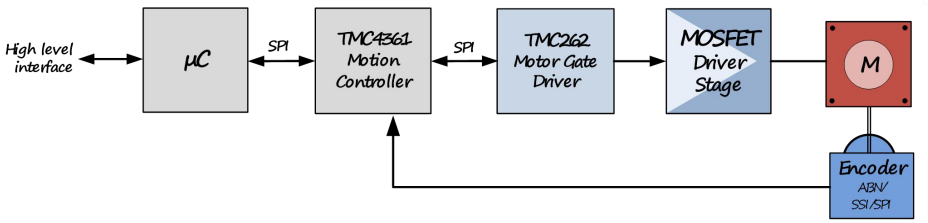


图 4: TMC262 闭环操作的硬件设置

带 dcStep™功能的开环操作 TMC4361A 连接步进电机驱动芯片 TMC2130 实现 dcStep 操作的典型框图如下。同样也适用于 TMC2160 和 TMC26x 步进电机驱动芯片。

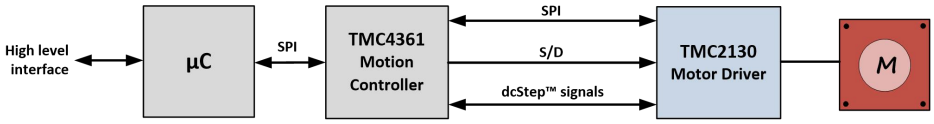


图 5: TMC2160 或 TMC2130 开环操作的硬件设置

订购代码

订购代码	描述	尺寸
TMC4361A-LA	*带闭环和 dcStep 功能的运动控制器, QFN40, 盘*	*6 x 6 mm²*
TMC4361A-LA-T	*带闭环和 dcStep 功能的运动控制器, QFN40, 卷带*	*6 x 6 mm²*

表 1: TMC4361A 订购代码

目 录

主手册

1. 引脚和设计过程信息

本章介绍所有引脚的命名并描述对应的功能。

1.1. 引脚分配：顶视图

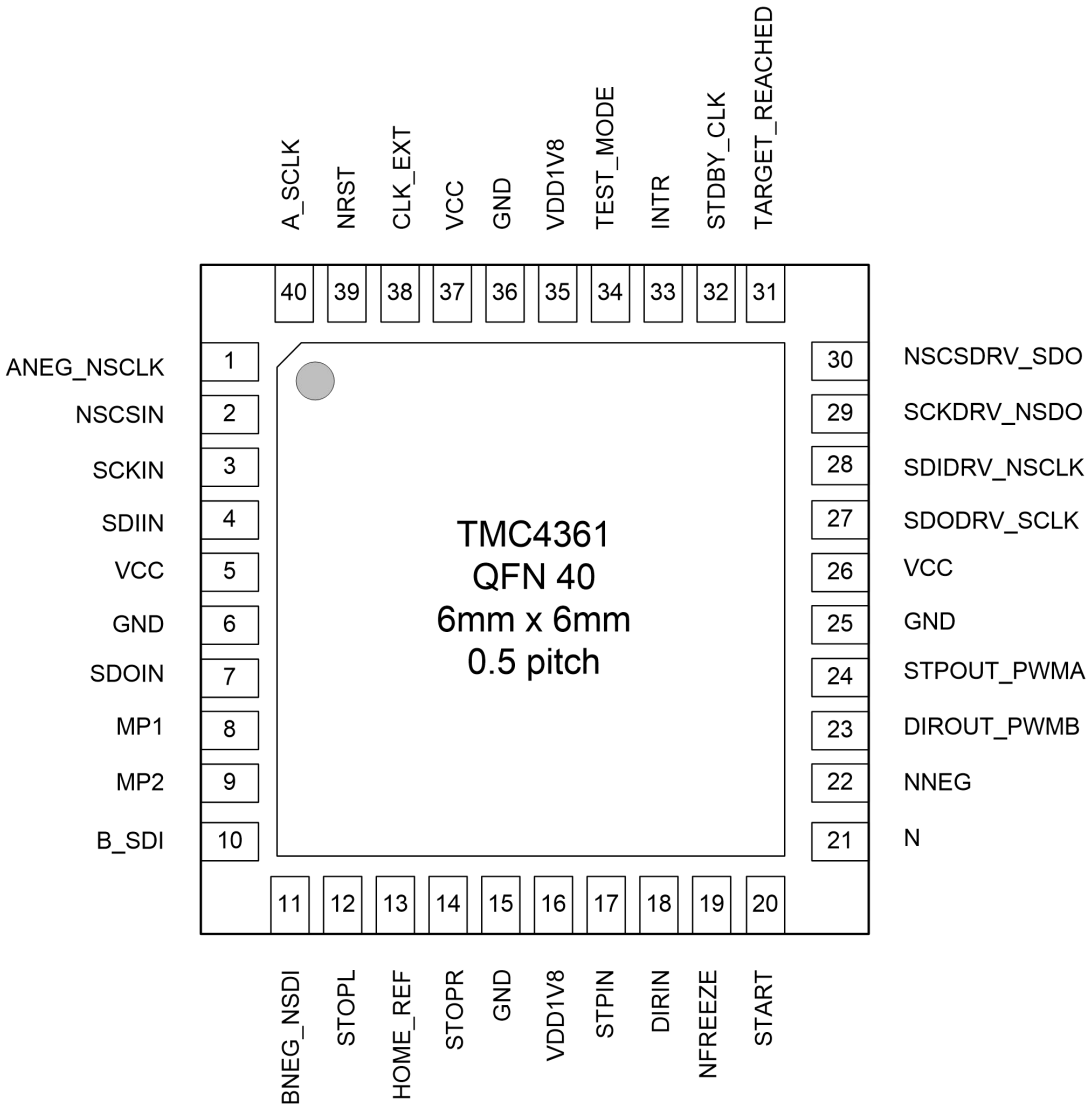


图 6: 封装： 引脚分配顶视图

1.2. 引脚定义

引脚命名和定义			
引脚	**引脚号**	**类型**	**功能**
电源引脚			
GND	6, 15, 25, 36	GND	输入输出口和数字电路的数字地引脚。
VCC	5, 26, 37	VCC	输入输出口和数字电路的数字电源 (3.3V… 5V)。
VDD1V8	16, 35	VDD	内部产生的内核 1.8V 电压的连接信号。
CLK_EXT	38	输入	时钟输入，提供频率为 f_{CLK} 的时钟供内部操作。
NRST	39	输入 (上拉)	低有效复位信号。 **如果未连接，上电复位**(VCC 上升期间有下拉电流)，之后内部上拉电阻**有效**。 下拉和上拉电流低(约 30 微安)。
TEST_MODE	34	输入	测试模式输入。正常连低电平。
NFREEZE	19	输入 (上拉)	低有效安全引脚，可立即冻结输出。 如果不连，内部上拉电阻有效。
微处理器接口引脚			
NSCSIN	2	输入	SPI 接口到微处理器的片选输入，低有效。
SCKIN	3	输入	SPI 接口到微处理器的串行时钟。
SDIIN	4	输入	SPI 接口到微处理器的串行数据输入。
SDOIN	7	输出	SPI 接口到微处理器的串行数据输出（如果 NSCSIN=1 则为高阻态）。
INTR	33	输出	中断输出，可配置为上拉/下拉，对应线与/或。
TARGET_REACHED	31	输出	目标达到输出，可配置为上拉/下拉，对应线与/或。
参考引脚			
STOPL	12	输入 (下拉)	左停止开关。外部信号停止斜坡运动。 如果不连，内部下拉电阻有效。
HOME_REF	13	输入 (下拉)	回零参考信号输入。外部信号用于搜索参考位置。 如果不连，内部下拉电阻有效。
STOPR	14	输入 (下拉)	右停止开关。外部信号停止斜坡。 如果不连，内部下拉电阻有效。
STPIN	17	输入 (下拉)	外部步进控制的步进输入引脚。 如果不连，内部下拉电阻有效。
DIRIN	18	输入 (下拉)	外部步进控制的方向输入引脚。 如果不连，内部下拉电阻有效。
START	20	输入**输出**	启动信号输入/输出引脚。缺省:输出。
步进/方向输出引脚			
STPOUT PWMA DACA	24	输出	步进输出信号。 第一个 PWM 信号(正弦)。 第一个 DAC 输出信号 (正弦)。.
DIROUT PWMB DACB	23	输出	方向输出信号。 第二个 PWM 信号(余弦)。 第二个 DAC 输出信号(余弦)。
接下一页!			

引脚命名和定义			
引脚	**引脚号**	**类型**	**功能**
步进电机驱动器的接口引脚			
NSCSDRV PWMB SDO	30	输出	连接电机驱动器的 SPI 接口的片选输出引脚，低有效。 第二个 PWM 信号(余弦)与 PHB (TMC23x/24x)连接的信号。 串行编码器输出接口的串行数据输出信号。
SCKDRV MDBN NSDO	29	输出	连接电机驱动器的 SPI 接口的串行时钟输出信号。 TMC23x/24x 的 MDBN 引脚的 MDBN 输出信号。 串行编码器输出接口的串行数据输出反相信号。
SDODRV PWMA SCLK	27	输入 **输出**	连接电机驱动器的 SPI 接口的串行数据输出信号（**缺省**）。 第一个 PWM 信号(正弦)与 PHA(TMC23x/24x)连接的信号。 串行编码器输出接口的时钟输入信号。
SDIDRV ERR NSCLK	28	输入 (下拉)	连接电机驱动器的 SPI 接口的串行数据输入信号。 TMC23x/24x 的 ERR 引脚的错误输入信号。 串行编码器输出接口的时钟反相输入。 如果不连，内部下拉电阻有效。
MP1	8	输入 (下拉)	DC_IN，作为外部 dcStep 输入控制信号。 如果不连，内部下拉电阻有效。
MP2	9	输入 **输出**	DCSTEP_ENABLE，作为 dcStep 输出控制信号 (缺省)。 SPE_OUT 输出信号，连接 TMC23x/24x 的 SPE 引脚。
STDBY_CLK	32	输出	StandBy 信号或内部 CLK 输出或 ChopSync 输出。
编码器接口引脚			
N	21	输入 (下拉)	增量编码器输入接口的 N 信号。 如果不连，内部下拉电阻有效。
NNEG	22	输入 (下拉)	增量编码器输入接口的 N 反相信号。 如果不连，内部下拉电阻有效。
B SDI	10	输入 (下拉)	增量编码器输入接口的 B 相信号。 串行编码器接口（SSI/SPI）的串行数据信号。 如果不连，内部下拉电阻有效。
BNEG NSDI SDO_ENC	11	**输入** 输出	增量编码器接口的 B 相输入反相信号(**缺省**)。 SSI 编码器输入接口的串行数据输入反相信号。 SPI 编码器输入接口的串行数据输出信号。
A SCLK	40	**输入** 输出	增量编码器接口的 A 相输入信号(**缺省**)。 串行编码器接口（SSI/SPI）的串行时钟输出信号。
ANEG NSCLK NSCS_ENC	1	**输入** 输出	增量编码器接口的 A 相输入反相信号(**缺省**)。 串行编码器接口的串行时钟输出反相信号。 SPI 编码器输入接口的低有效片选信号输出。

表 2: 引脚名称和描述

1.3. 系统概述

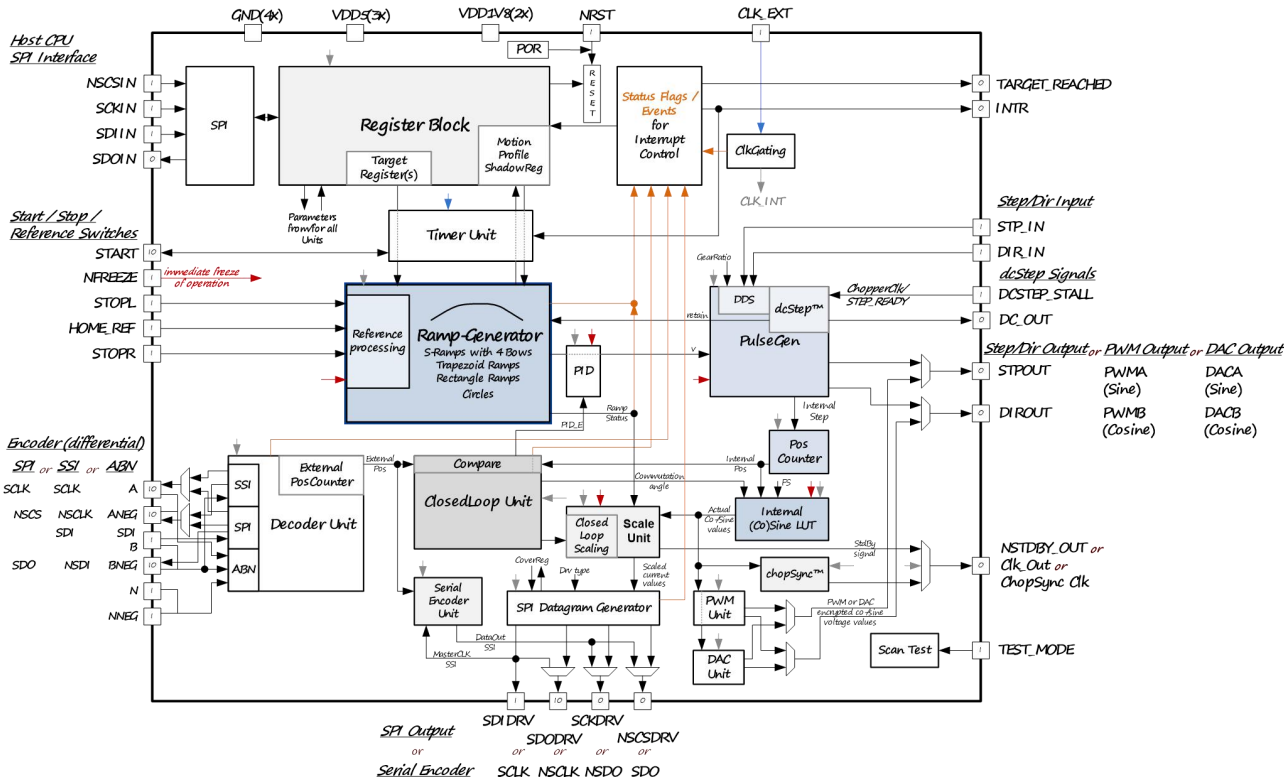


图 7:系统概述

2. 应用电路

本章介绍了应用电路及外部连接的元器件。

2.1. TMC4361A 标准连接: VCC=3.3V

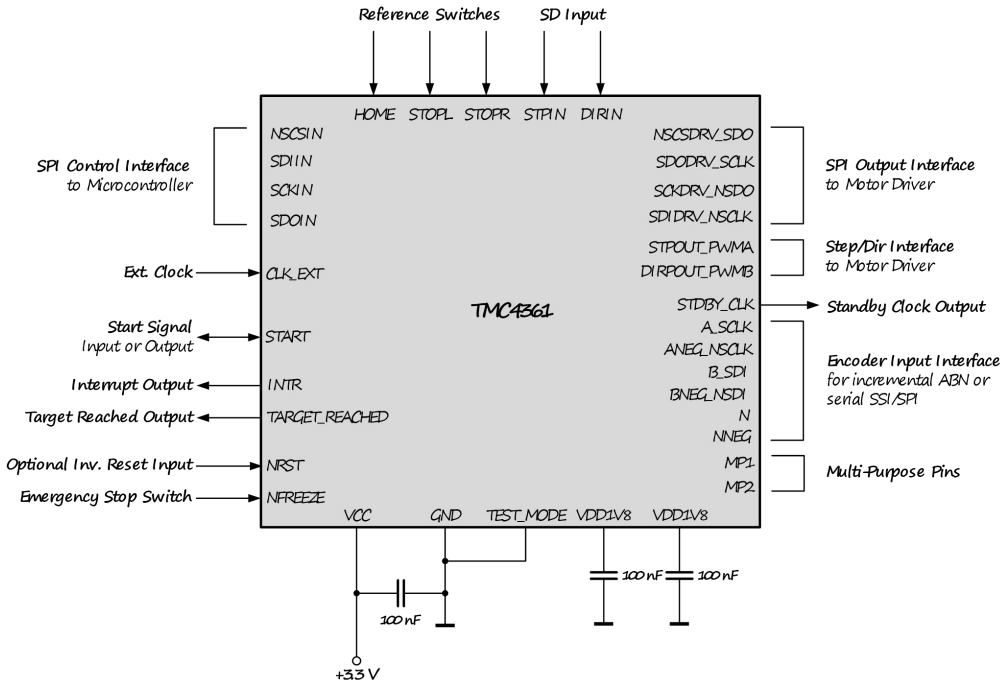


图 8: TMC4361A 连接: VCC=3.3V

2.2. TMC4361A 连接 TMC26x 驱动芯片

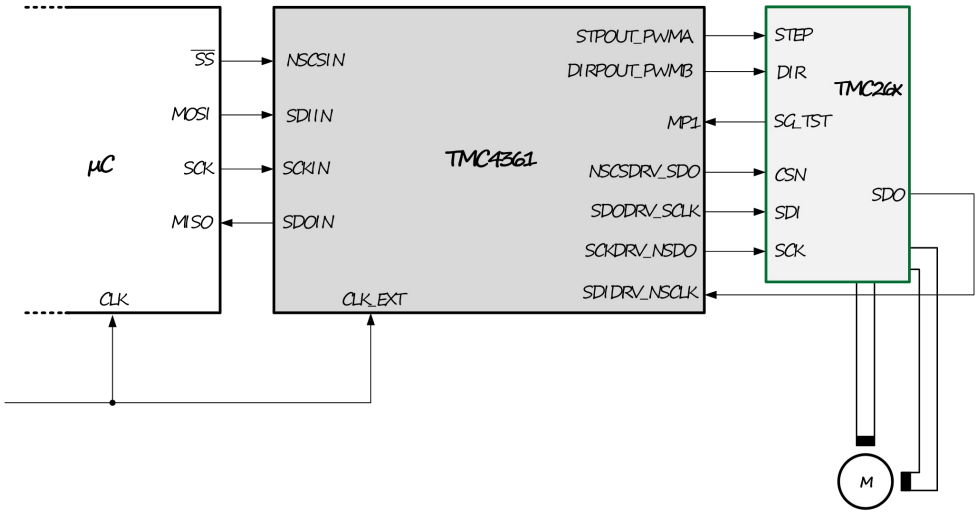


图 9: TMC4361A 以 SPI 模式或者 S/D 模式连接 TMC26x 步进驱动芯片

2.3. TMC4361A 连接 TMC2130 或 TMC2160 驱动芯片

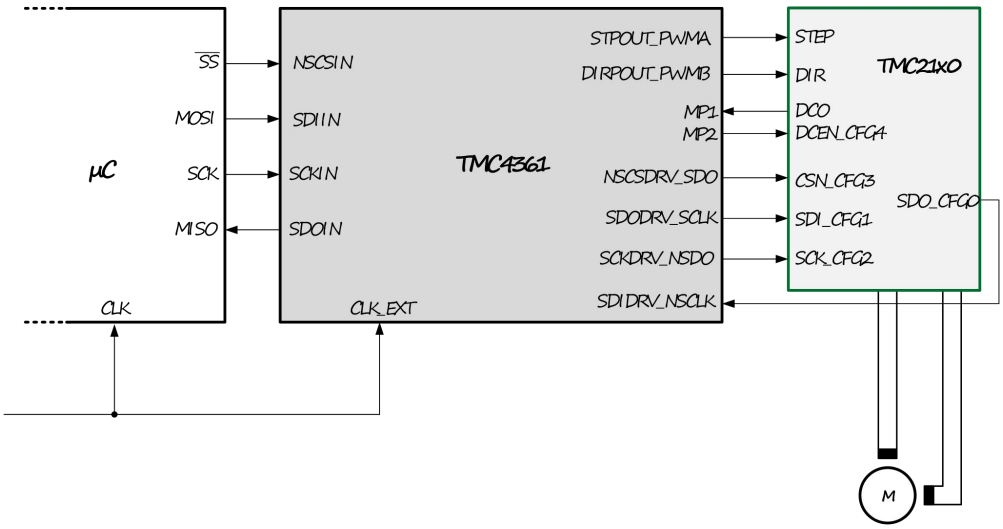


图 11: TMC4361A 以 SPI 模式或者 S/D 模式连接 TMC2130 或 TMC2160 步进驱动芯片

2.4. TMC4361A 连接 TMC5130A 或 TMC5160

TMC5130A 和 TMC5160 集成了运动控制器及两相步进电机驱动功能。
对于某些应用，建议将 TMC4361A 与 TMC5130A 或 TMC5160 结合使用。
如果需要这些组合，之前描述的有关 TMC2130 或者 TMC2160 的所有信息和配置程序，同样适用于 TMC5130A 或 TMC5160，因为从 TMC4361A 的角度来看，这些芯片都是软件兼容的。

i 请参考 TMC5130A 或 TMC5160 手册获取更多信息。

阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

3. SPI 接口

TMC4361A 采用 40 位 SPI 数据报与微控制器通信。位串行接口与总线时钟同步。当总线主设备向从设备发送每一位时，从设备同时向主设备返回一位。下一章介绍了 SPI 控制接口、SPI 数据报结构和 SPI 处理。

SPI 输入控制接口引脚		
引脚名称	**类型**	**备注**
NSCSIN	输入	SPI 到微控制器的片选信号(低电效)
SCKIN	输入	SPI 到微控制器的串行时钟
SDIIN	输入	SPI 到微控制器的串行输入数据
SDOIN	输出	SPI 到微控制器的串行输出数据

表 3: SPI 输入控制接口引脚

3.1. SPI 数据报结构

- 带硬件 SPI 的微控制器通常按照一个字节 8 位的整数倍进行通信。
- TMC4361A 的 NSCSIN 信号必须在数据报传输的整个持续时间内保持有效(低电平)。
- 每个发送到 TMC4361A 的数据报由一个地址字节和四个数据字节组成。TMC4361A 寄存器组直接进行 32 位数据通信和处理。每个寄存器按照 32 个数据位处理，即使少于 32 个数据位。

i 每个寄存器由一个字节的地址:

- *读访问，地址字节的最高有效位是 0。*
- *写访问，地址字节的最高有效位是 1。*

注意:

→ *有些寄存器是只写寄存器。大多数寄存器也可读；还有一些只读寄存器。.*

<table>
<tr><th colspan="40">TMC4361A SPI 数据报结构</th></tr>
<tr><th colspan="13">最高位 (先传送)</th><th colspan="14">40 位</th><th colspan="13">最低位(后传送)</th></tr>
<tr><td colspan="13">39</td><td colspan="14">...</td><td colspan="13">0</td></tr>
<tr><td colspan="8">→8 位地址
←8 位 SPI 状态</td><td colspan="32">←→32 位数据</td></tr>
<tr><td colspan="8">39 ... 32</td><td colspan="32">31 ... 0</td></tr>
<tr><td colspan="8">→发送到 TMC4361:
读写+7 位地址
←TMC4361 返回:
8 位 SPI 状态</td><td colspan="8">8 位数据</td><td colspan="8">8 位数据</td><td colspan="8">8 位数据</td><td colspan="8">8 位数据</td></tr>
<tr><td colspan="8">39 / 38 ... 32</td><td colspan="8">31 ... 24</td><td colspan="8">23 ... 16</td><td colspan="8">15 ... 8</td><td colspan="8">7 ... 0</td></tr>
<tr><td>W</td><td colspan="7">38...32</td><td colspan="4">31...28</td><td colspan="4">27...24</td><td colspan="4">23...20</td><td colspan="4">19...16</td><td colspan="4">15...12</td><td colspan="4">11...8</td><td colspan="4">7...4</td><td colspan="4">3...0</td></tr>
<tr><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td><td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

图 12: TMC4361A SPI 数据报结构

阅读整个文档；是第 22 章(第 174 页)中的补充指令。

读/写选择原则和处理

读写选择由地址字节的最高位(串行接口数据报的第 39 位)控制。读访问该位为 0，写访问为 1。因此，名为 W 的位是 WRITE_notREAD 控制位。

地址字节的最高位是有效的高电平，对应写入操作。
因此，0x80 必须加入写地址。

SPI 接口始终将数据传送回主机，与 W 写位无关。

读写访问的区别	
如果...	**则...**
以前的访问是读访问。	传回的数据是之前一个读数据报地址对应的数据值。
先前的访问是写访问	数据回读反映了先前接收的写数据。

图 13:读写访问的区别

结果:
因此，读访问和写访问之间的区别在于：读访问不将数据传输到寻址寄存器，而是仅传输地址；它的 32 个数据位都是无用的。

注意:

→ *请注意，接下来读访问会将上一个读周期中传输的地址对应的读数据传回。读请求后，数据立即锁存。*

特别注意:

读访问请求数据报的数据为空。
读取的数据通过后续的读或写访问传输回主机。

使用空的写数据

`i` 读多个寄存器可以流水线方式完成。发送的数据在启动数据传输后立即锁存。

读写访问例程

读地址为 0x21 的寄存器 *XACTUAL*，地址字节必须设置为 0x21。
写寄存器 *VACTUAL*，地址字节必须设置为 0x 80+0x 22 = 0xA2。读访问的数据部分可以具有任何值，例如 0。

读写访问示例		
操作步骤	**数据发送到 TMC**	**从 TMC 接收的数据**
读 *XACTUAL*	→ 0x2100000000	← 0xSS[1] & 无用的数据
读 *XACTUAL*	→ 0x2100000000	← 0xSS & *XACTUAL*
写 *VACTUAL*:= 0x00ABCDEF	→ 0xA200ABCDEF	← 0xSS & *XACTUAL*
写 *VACTUAL*:= 0x00123456	→ 0xA200123456	← 0xSS00ABCDEF

表 4:读写访问示例

[1] SS 是 SPI_STATUS.状态位 SPI_STATUS 的占位符

 阅读整个文档；是第 22 章(第 174 页)中的补充指令。

数据排列

所有数据都是右对齐的。一些寄存器是无符号(正)值；另一些将整数值(有符号)表示为二进制补码。

有些寄存器由表示为位或位向量的开关组成。

SPI 传送处理

SPI 传送处理过程如下:

- 拉低片选输入信号 NSCSIN，使能从机 SPI 传送处理。
- 位传输与总线时钟 SCKIN 同步，从机在 SCKIN 的上升沿锁存来自 SDIN 的数据，并在下降沿之后将数据驱动至 SDOIN。
- 首先发送最高有效位。

i 与 TMC4361A 的 SPI 总线传输至少需要 40 个 SCKIN 时钟周期。

特别注意事项

请考虑以下方面:

系统行为说明

- **无论从 TMC4361A 读数据或向其写数据时**，返回的前八位是由用户选择的八个事件位组成的 SPI 状态 *SPI_STATUS*。第 5.2. 章第 21 页解释了这些位。
- **如果传输的时钟周期少于 40 个**，则传输无效；即使对于读访问。但是，仅发送八个时钟周期对于获得 SPI 状态可能是有用的，因为它首先将状态信息发回。
- **如果传输超过 40 个时钟周期**，则内部移位寄存器经过 40 个时钟延迟后，移入 SDOIN 的额外位会在 SDOIN 上移出。可用于多个芯片的菊花链。
- **在整个总线传输期间，NSCSIN 必须为低电平**。当 NSCSIN 变为高电平时，内部移位寄存器的内容被锁存到内部控制寄存器中，并被识别为主机到从机的指令。如果发送了超过 40 位，则仅接收到的最后 40 位(NSCSIN 上升沿之前)为有效指令。

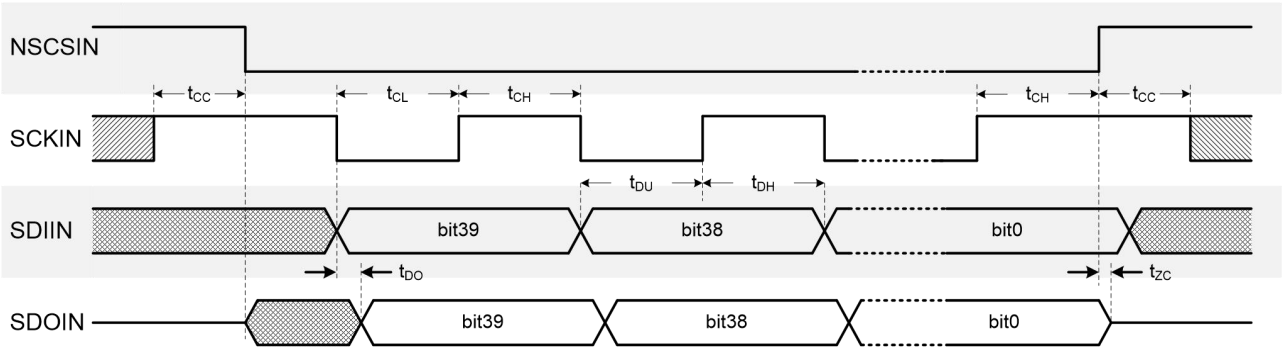


图 14: SPI 数据报时序

 阅读整个文档；是第 22 章(第 174 页)中的补充指令。

3.1.1. SPI 时序描述

SPI 与内部系统时钟同步，SPI 总线时钟 SCKIN 应小于在系统时钟频率的四分之一。内部施密特触发器支持 SPI 输入的信号处理，但不带 RC 滤波。

注意:

→ *为了避免微处理器和 TMC4361A 之间的 SPI 接口输入端出现毛刺，必须提供外部 RC 元件。*

图 14 是 SPI 总线传输的时序参数，下表指定了参数值。

SPI 接口时序						
SPI 接口时序	**交流特性:**		**外部时钟周期: t_{CLK}**			
参数	**符号**	**条件**	**最小**	**典型**	**最大**	**单位**
NSCSIN 变化前或者后的 SCKIN 有效时间	t_{CC}		10			ns
NSCSIN 高电平时间	t_{CSH}	最短时间是同步 CLK 与 SCKIN 高电平，仅在 SCHIN 高电平之前一个 t_{CH}。	t_{CLK}	>2·t_{CLK}+10		ns
SCKIN 低电平时间	t_{CL}	最短时间仅同步 CLK。	t_{CLK}	>t_{CLK}+10		ns
SCKIN 高电平时间	t_{CH}	最短时间仅同步 CLK。	t_{CLK}	>t_{CLK}+10		ns
外部时钟（如：f_{CLK} = 16 MHz） 下的 SCKIN 频率	f_{SCK}	假设同步时钟			f_{CLK} / 4 (4)	MHz
上升沿之前的 SDIIN 建立时间	t_{DU}		10			ns
SCKIN 上升沿后的 SDIIN 保持时间	t_{DH}		10			ns
SCKIN 时钟沿下降后的数据输出有效时间	t_{DO}	SDOIN 上没有容性负载.			t_{FILT}+5	ns

表 5: SPI 接口时序

ℹ $t_{CLK} = 1 / f_{CLK}$

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

4. 状态标志和事件

TMC4361A 包含 32 个状态标志和 32 个状态事件，以获取内部或电机驱动器状态信息。这些标志和事件可以从专用寄存器中读取。在接下来的章节中，您将了解到状态事件的中断生成。状态事件也可以分配给前八个 SPI 状态位，它们在每个 SPI 数据报中被发送。

引脚名称: 状态事件		
引脚名称	**类型**	**备注**
INTR	输出	中断输出以指示状态事件。

表 10: 引脚名称: 状态事件

寄存器名称:状态标志和事件			
寄存器名称	**寄存器地址**		**备注**
GENERAL_CONF	0X00	RW	位: 15, 29, 30.
STATUS_FLAGS	0X0F	R	TMC4361A 和连接的 TMC 电机驱动芯片的 32 个状态标志。
EVENTS	0X0E	R+C W	TMC4361A 状态位变化触发的 32 个事件。
SPI_STATUS_SELECTION	0X0B	RW	从 32 个事件中选择 8 个事件作为 SPI 状态位。
EVENT_CLEAR_CONF	0X0C	RW	清除事件位。
INTR_CONF	0X0D	RW	选择 32 个事件作为 INTR 输出。

表 11: 寄存器名称:状态标志和事件

阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

4.1. 状态事件描述

状态事件基于状态位。如果状态位发生变化，相关事件将从无效触发到有效。必须手动才能将事件重置到无效状态。

状态位的关联 状态位和状态事件有不同的关联方式:

- 状态标志反映现状，而状态事件表示自 *EVENTS* 寄存器的上次读取请求以来，专用的信息已发生变化。几个状态事件可以与一个状态位相关联。
- 一些状态事件显示了一个或多个的状态发生了转变。例如，在所选的电机驱动器状态标志中，只要其中一个有效，则都会触发电机驱动器事件 *MOTOR_EV*。
- 如果一个标志包含多位，则相关事件触发可对应有效的标志位组合。例如，*VEL_STATE* 标志有两个比特位但对应三个相关联的速度状态事件(b'00/b'01/b'10)。如果相关的组合从无效切换到有效，则触发对应的事件。

注意:

→ *STATUS_FLAGS 寄存器 0x0F 中的某些标志没有对应的事件(例如，*COVER_DONE *表示完成电机驱动器芯片的数据覆盖)。*

事件自动清除 读 *EVENTS* 寄存器 0x0E 对应事件自动清零。事件对于中断生成和 SPI 状态监控非常重要。

注意:

→ *建议在正常操作前通过读请求清除 EVENTS 寄存器 0x0E。*

特别注意 **如果状态事件正好在 *EVENTS* 寄存器 *0x0E* 清零时或者之前触发,可能会无法识别状态事件。**

如何避免信息缺失 为了防止事件被清除，请根据事件寄存器中的特定事件分配 *EVENT_CLEAR_CONF* 寄存器 0x0C:

操作步骤:

➢ 设置 EVENT_*CLEAR_CONF* 寄存器位置位为 1。

结果:
读 *EVENTS* 寄存器不清除相关事件。

为了清除这些事件，如有必要，请执行以下操作:

操作步骤:

➢ 设置相关 *EVENTS* 寄存器 0x0E 位置位为 1。

结果:
写 *EVENTS* 寄存器清除相关事件。

4.2. SPI 状态位传输

最多可以选择八个事件作为 SPI 状态返回。这些事件总在每个 TMC4361A SPI 响应中的最高有效传输位传输。

将事件分配给状态位

给 SPI 状态位分配事件，请根据 *EVENTS* 事件寄存器中的特定事件设置 *SPI_STATUS_SELECTION* 寄存器 0x0B:

操作步骤:

➢ 将相关的 *SPI_STATUS_SELECTION* 寄存器对应的比特位置 1。

结果:

每个 SPI 数据报中返回的 *SPI_STATUS* 对应了所选的事件。

注意:

→ *比特位的位置根据 EVENTS 寄存器 0x0E 中的事件的比特位的位置进行排序。如果选择了八个以上的事件，前八位(从索引 0 = LSB 最低位开始)将作为 SPI_STATUS 传送。*

4.3. 中断的产生

类似于 *EVENT_CLEAR_CONF* 寄存器和 *SPI_STATUS_SELECTION* 寄存器，可选择事件作为 INTR 输出。所选事件“或”运算输出信号，这意味着一旦其中一个所选事件触发，INTR 输出开关就会激活。

产生中断

请根据 EVENTS 事件寄存器中的特定事件分配 *INTR_CONF* 寄存器 0x0D，选择 INTR 输出引脚的事件:

操作步骤:

➢ 设置相关 *INTR_CONF* 寄存器的比特位为 1。

结果:

相关事件在 INTR 输出端输出。如果请求了多个事件，一旦选定的事件之一激活，INTR 就有效。

INTR 输出极性

缺省，INTR 是低有效。

要将 INTR 极性变为高有效，请执行以下操作:

操作步骤:

➢ 设置 *intr_pol* =1 (*GENERAL_CONF* 寄存器 0x00)。

结果:

INTR 现在为高有效。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

4.4. 连接多个 INTR

INTR 引脚可以配置为多个 TMC4361A 连接微控制器的中断共享信号线。

连接多个中断引脚 配置成“线或”或者“线与”行为，须采取以下操作:

操作步骤:

- **步骤 1:** 设置 *intr_tr_pu_pd_en* = 1 (*GENERAL_CONF* 寄存器 0x00)。

选项 1: 线或

操作步骤:

- **步骤 2:** 设置 *intr_as_wired_and* = 0 (*GENERAL_CONF* 寄存器 0x00)。

结果:

INTR 引脚工作在线或(默认配置)模式。

i 如果 INTR 引脚无效，则引脚为非有效极性的弱输出。如果其中一个连接引脚有效，整条线路将被设置为有效极性。

选项 2: 线与

操作步骤:

- **步骤 2:** 设置 *intr_as_wired_and* = 1 （*GENERAL_CONF* 寄存器 0x00）。

结果:

在没有中断的情况下，INTR 引脚为非有效极性的强输出。在有效态引脚为弱输出。因此，在所有引脚都在有效极性的情况下，整个信号线才有效。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

5. 不同运动轮廓的斜坡配置

产生步进信号是步进电机运动控制器的主要任务之一。TMC4361A 的内部斜坡发生器提供了几种具有不同运动轮廓的步进发生配置，可结合速度或定位模式。

引脚名称:斜坡发生器		
引脚名称	**类型**	**备注**
STPOUT_PWMA	输出	步进输出信号。
DIROUT_PWMB	输出	方向输出信号。

表 12:引脚名称:斜坡发生器

寄存器名称:斜坡发生器			
寄存器名称	**寄存器地址**		**备注**
GENERAL_CONF	0x00	RW	斜坡发生器相关位 5:0。
STP_LENGTH_ADD	0x10	RW	延长步进信号有效电平的时间，以时钟周期为单位; 16 位。
DIR_SETUP_TIME			方向信号电平改变后，不产生步进信号的时间，以时钟周期为单位; 16 位。
RAMPMODE	0x20	RW	请求的运动轮廓和操作模式; 3 位。
XACTUAL	0x21	RW	当前内部微步位置；有符号；32 位。
VACTUAL	0x22	R	当前步进速度；24 位；有符号；没有小数。
AACTUAL	0x23	R	当前步进加速度；24 位；有符号；没有小数。
VMAX	0x24	RW	最大允许速度或目标速度；有符号；32 位= 24+8 (24 位整数部分，8 位小数)。
VSTART	0x25	RW	斜坡开始时的速度；无符号；31 位=23+8。
VSTOP	0x26	RW	斜坡结束的速度；无符号；31 位=23+8。
VBREAK	0x27	RW	梯形斜坡模式下，在此速度值，加速/减速将发生变化；无符号；31 位=23+8。
AMAX	0x28	RW	最大允许加速度或目标加速度；无符号；24 位=22+2 (22 位整数部分，2 位小数)。
DMAX	0x29	RW	最大允许减速或目标减速；无符号；24 位=22+2。
ASTART	0x2A	RW	斜坡开始时或低于 VBREAK 时的加速度；无符号；24 位=22+2。
DFINAL	0x2B	RW	斜坡末端或低于 VBREAK 时的减速度，无符号；24 位=22+2。
BOW1	0x2D	RW	完整速度斜坡的第一个弓形值；无符号；24 位=24+0 (24 位整数部分，无小数位数)。
BOW2	0x2E	RW	完整速度斜坡的第二个弓形值；无符号；24 位=24+0。
BOW3	0x2F	RW	完整速度斜坡的第三个弓形值；无符号；24 位=24+0。
BOW4	0x30	RW	完整速度斜坡的第四个弓形值；无符号；24 位=24+0。
CLK_FREQ	0x31	RW	外部时钟频率 f_{CLK}，无符号；25 位。
XTARGET	0x37	RW	目标位置; 无符号; 32 位。

表 13:寄存器名称:斜坡发生器

5.1. Step/Dir 输出配置

本节重点介绍 Step/Dir 输出配置。.

5.1.1. Step/Dir 输出配置步骤

可为连接驱动器配置 Step/Dir 输出信号。

如果步进信号需要大于一个时钟周期，请执行以下操作:

➢ 设置正确 *STP_LENGTH_ADD* 寄存器 0x10 (位 15:0)。

操作步骤:

结果:
最终步长等于 *STP_LENGTH_ADD*+1 个时钟周期。因此步进长度范围为 1~2^{16} 个时钟周期。

操作步骤:
➢ 设置正确 *DIR_SETUP_TIME* 寄存器 0x10 (位 31:16)。

结果:
DIR_SETUP_TIME 时钟周期为 Dir 信号的建立时间，对应 DIROUT 信号和 STPOUT 信号电压变化之间的延迟周期。在 DIROUT 电平改变后的 *DIR_SETUP_TIME* 时间内， STPOUT 不发送任何步进信号。

原则:

DIROUT 在以下条件下不会改变电平

- 在步进脉冲信号有效电平期间
- 步进信号返回非有效电平后的(*STP_LENGTH_ADD*+1)时钟周期内

5.1.2. STPOUT: 改变极性

可设置 STPOUT 特性，如下所示:

默认情况下，步进输出为高电平有效，因此 STPOUT 的上升沿表示有效步进输入。

要改变极性，请执行以下操作:

操作步骤:
➢ 设置 *step_inactive_pol* =1 (*GENERAL_CONF* 寄存器 0x00 的第 3 位)。

结果:
每个下降沿代表有效的步进输入。

如何在电平更改时提示一个步进信号

要在每次电平更改时对应产生一个有效的步进信号，请执行以下操作:

操作步骤:
➢ 设置 *toggle_step* =1 (*GENERAL_CONF* 寄存器 0x00 的第 4 位)。

结果:
每一次电平的变化都对应着一个步进输入。

DIROUT: 改变极性

默认情况下，DIROUT 的电压电平 1 表示步进的运动方向位反方向。
可以设置不同的 DIROUT 特性，如下所示。

要改变极性，请执行以下操作:

操作步骤:
➢ 设置 *pol_dir_out* =0 (*GENERAL_CONF* 寄存器 0x00 的第 5 位)。

结果:
DIROUT 高电平表示步进正方向。

注意:

→ *DIROUT 对应内部微步位置 MSCNT 寄存器，因此整个过程基于内部 SinLUT，参见第 10.2. 节第错误！未定义书签。页。*

5.2. 改变内部运动方向

缺省情况下，正的内部速度 *VACTUAL* 对应内部 SinLUT 表的向前运动。因此，如果 VACTUAL <0，则 SinLUT 值向后运动。

如何改变运动方向 要更改内部运动方向，请执行以下操作:

操作步骤:

- 设置 *reverse_motor_dir* =1 (*GENERAL_CONF* 寄存器 0x00 的第 28 位)。

结果:

内部实际速度为正对应内部 SinLUT 表的向后运动。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

5.3. 操作模式和运动轮廓的详细配置信息

本节提供了两种操作模式(速度模式和定位模式)以及四种可能的运动模式(无斜坡、梯形斜坡(包括六点斜坡)和 S 形斜坡)的信息，可以组合。他们每个都有各自的优势。请用户根据系统选择适合的配置。

内部斜坡发生器的描述 配置产生 TMC4361A 的内部斜坡发生器的各种斜坡，并为 STPOUT 提供相关的步进输出。

为了成功配置内部斜坡发生器，使其尽可能适合您的特定应用，请参考下表和以下页面介绍的每个可能的组合信息。

斜坡发生器配置选项			
操作模式	**运动轮廓**	**RAMPMODE(2:0)**	**描述**
速度模式	无斜坡	b'000	仅遵循 *VMAX* 请求。
	梯形斜坡	b'001	遵循 *VMAX* 请求，并考虑加速和减速值。
	sixPoint 六点斜坡	b'001	遵循 *VMAX* 请求，并考虑加速/减速值以及启动和停止速度值。
	S 形斜坡	b'010	遵循 *VMAX* 要求，考虑最大加速度/减速度值，并用 4 个不同的弓形值参数调整这些值。
定位模式	无斜坡	b'100	仅遵循 *XTARGET* 和 *VMAX* 请求.
	梯形斜坡	b'101	遵循 *XTARGET* 请求和最大速度 *VMAX* 请求，并考虑加速和减速值.
	sixPoint 六点斜坡	b'101	遵循 *XTARGET* 请求和最大速度 *VMAX* 请求，并考虑加速/减速值以及开始和停止速度值。
	S 形斜坡	b'110	遵循 *XTARGET* 请求和最大速度 *VMAX* 请求，考虑最大加速/减速值，并用 4 个不同的弓形值参数调整这些值。

表 14: 通用和基本斜坡配置选项概述

 阅读整个文档；是第 22 章(第 174 页)中的补充指令。

5.3.1. 开始:选择操作模式

有两种操作模式:速度模式和定位模式。

开始之前 **设置这些参数之前:**

先选择:

- **操作模式和**
- **运动轮廓**

不建议在运动过程中改变操作模式或运动轮廓。

操作模式:速度模式 **RAMPMODE** 寄存器提供速度模式或定位模式两种工作模式的选择。

设置速度模式，请执行以下操作:

操作步骤:

➢ 设置 *RAMPMODE*(2) =0 (*RAMPMODE* 寄存器 0x20)。

结果:

选择速度模式。选定的运动轮廓达到目标速度 *VMAX*。

操作模式:定位模式 **设置定位模式，请执行以下操作:**

➢ 设置 *RAMPMODE*(2)=1 (*RAMPMODE* 寄存器 0x20)。

操作步骤:

结果:

选择定位模式。*VMAX* 为运动过程中的最大速度值，目标为停止在目标位置 *XTARGET*。

注意:

→ *定位过程与* VMAX *的正负不相关。运动方向取决于* XACTUAL、XTARGET *和当前斜坡运动轮廓状态。*

注意:

→ *定位模式需要满足* VMAX ≤ ¼ f_{CLK}

5.3.2. 运动中停止

定位过程中停止运动，请执行以下操作:

➢ 设置 *VMAX* = 0 (寄存器 0x24)。

操作步骤:

结果:

速度斜坡调用实际斜坡参数指向 *VACTUAL* = 0。

i *VMAX* ≠ 0 运动继续进行.

 阅读整个文档;是第 22 章(第 174 页)中的补充指令。

5.3.3. 运动轮廓配置

提供了三种基本的运动轮廓。在过程中，每一个都有不同的速度值。见下表。

对于运动轮廓的配置，请执行以下操作:

操作步骤:

➢ 配置 RAMPMODE 寄存器 0x20 的位 1 和 0。

结果:

如下表所示.

您可以从下面的列表中选择不同的配置选项:

- 无斜坡运动轮廓
- 梯形斜坡运动轮廓(包括 SixPoint 六点斜坡)
- S 形斜坡运动轮廓

TMC4361A 运动轮廓		
***RAMPMODE* (1:0)**	**运动轮廓**	**功能**
b'00	无斜坡	仅遵循 VMAX(矩形速度形状).
b'01	梯形斜坡	考虑加速度和减速度值，而不调整这些加速度值。
	sixPoint 六点斜坡	考虑加速度和减速度值，而不调整这些加速度值。 **采用开始和停止速度值。** **(见章节** 6.5. 第 40 页**)**
b'10	S 形斜坡	使用所有斜坡值(包括弓形参数值)。

表 15: TMC 4361 A 运动轮廓的描述

 阅读整个文档; 是第 22 章(第 174 页)中的补充指令。

5.3.4. 无转折点的四点梯形斜坡

无转折速度 VREAK 的四点梯形斜坡运动轮廓，请执行以下操作:

操作步骤:

- 设置 RAMPMODE*(1:0) =b'01 (*寄存器 *0x20)*。
- 设置 *VBREAK* =0 (寄存器 0x27)。
- 设置合适的 *AMAX* 寄存器 0x28 和 *DMAX* 寄存器 0x29。
- 设置合适的 *VMAX* 寄存器 0x24。

结果:

内部速度 *VACTUAL* 线性上升到 *VMAX*。只有 *AMAX* 和 DMAX 定义加速/减速斜率。

注意:

→ *AMAX 为从绝对低速到绝对高速的上升斜率，而 DMAX 为从绝对高速到绝对低速的下降斜率。*

→ *加速斜率和减速斜率各只有一个加速和减速值对应。*

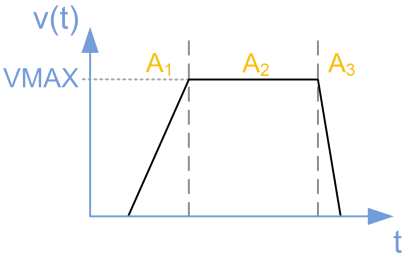


图 20: 不带 Break 点的梯形斜坡

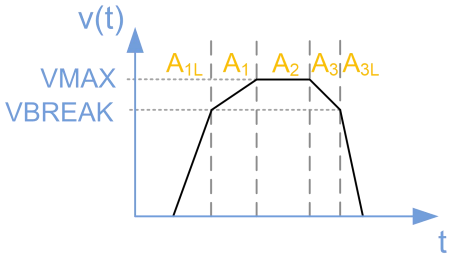


图 21: 带 Break 点的梯形斜坡

5.3.5. 带转折点的梯形斜坡

带速度转折点 VREAK 的梯形斜坡运动轮廓，请执行以下操作:

操作步骤:

- 设置 *RAMPMODE*(1:0)=b'01 (寄存器 0x20)。
- 设置适合的 *VBREAK* 寄存器 0x27。
- 设置适合的 *AMAX* 寄存器 0x28 和 *DMAX* 寄存器 0x29。
- 设置适合的 *ASTART* 寄存器 0x2A 和 *DFINAL* 寄存器 0x2B。
- 设置适合的 *VMAX* 寄存器 0x24。

结果:

内部速度 *VACTUAL* 线性改变为 *VMAX*。除了 *AMAX* 和 DMAX 之外，增加 *ASTART* 和 *DFINAL* 控制加速或减速斜率(见上图)。

注意:

→ *AMAX 和 ASTART 为从绝对低速到绝对高速的上升斜率。*

→ *DMAX 和 DFINAL 为从绝对高速到绝对低速的下降斜率。*

→ *加速/减速系数在 VBREAK 处改变。在 VBREAK 以下的加减速值为 ASTART 和 DFINAL，在 VBREAK 以上的加减速值为 AMAX 和 DMAX。*

5.3.6. 结合梯形斜坡的定位模式

运动方向取决于 *XTARGET*。

要在定位模式下设置 4 点或 6 点斜坡，请执行以下操作:

操作步骤:

- 设置 *RAMPMODE*(2:0) =b'101 (寄存器 0x20)。
- 如上所述，相应地设置梯形斜坡类型。
- 设置合适的 *XTARGET* 寄存器 0x37。

结果:

尽可能长时间保持|VACTUAL| = *VMAX*，斜坡精确地在目标位置 *XTARGET* 处结束。

梯形斜坡的 *AACTUAL* 分配

***AACTUAL* 设置分配适用于 4 点和 6 点斜坡.**

加速/减速系数 *AACTUAL* 寄存器取决于当前斜坡阶段和需要达到的速度。下表给出了不同斜坡阶段的相关符号分配:

梯形斜坡的 *AACTUAL* 分配					
斜坡相位	**A_{1L}**	**A_1**	**A_2**	**A_3**	**A_{3L}**
v>0: AACTUAL=	ASTART	AMAX	*0*	−DMAX	−DFINAL
v<0: AACTUAL=	−ASTART	−AMAX	*0*	DMAX	DFINAL

表 16: 梯形斜坡: 运动过程中的 AACTUAL 分配

阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

5.3.7. S 形斜坡的配置

S 形斜坡，请执行以下操作:

操作步骤:

- 设置 *RAMPMODE*(1:0)=b'10 (寄存器 0x20)。
- 设置适合的 *BOW1* … *BOW4* 寄存器 0x2C…0x30。
- 设置适合的 *AMAX* 寄存器 0x28 和 *DMAX* 寄存器 0x29。
- 设 *ASTART* = 0 (寄存器 0x2A)。
- 设置 *DFINAL* = 0 (寄存器 0x2B)。
- 设置适合的 *VMAX* 寄存器 0x24。

结果:

内部速度 *VACTUAL* 按照"S"形斜坡上升为 *VMAX*。加速度/减速度值根据弓形参数值而改变

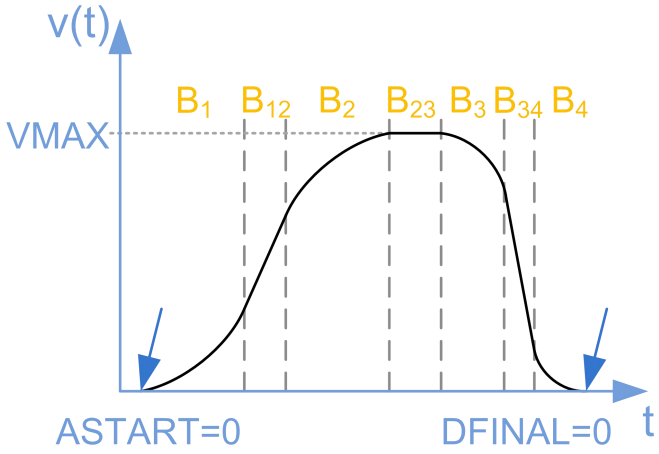


图 22: 没有初始和最终加速/减速值的 s 形斜坡

S 形坡道上升的定义

上升坡度(绝对较低速度到绝对较高速度):

- *BOW1* 定义了增加绝对加速度值的值。
- *BOW2* 定义了减少绝对加速度值的值。
- *AMAX* 定义了最大绝对加速度值的值。

S 形坡道下降坡度的定义

下降坡度(绝对较高速度至绝对较低速度):

- *BOW3* 定义了增加绝对减速度值的值。
- *BOW4* 定义了减少绝对减速度值的值。
- *DMAX* 定了最大绝对减速度的值。

∞∀接下一页。

不建议在运动期间更改斜坡参数[1]和/或操作模式。但是，如果有必要，请注意:

注意

避免在定位模式期间出现不期望的系统动作！
斜坡过程中斜坡参数值的变化可能导致:

- *XTARGET* 或机械停止位置的过冲。
- 由于在斜坡过程中保持了 B1、B2、B3 和 B4 的弓形，因此会出现 *VACTUAL* 瞬间过冲超过 VMAX。

这将确保定位模式下的平稳操作。

[1] *XTARGET 和 VMAX 除外。这些参数可以在运动过程中改变。*

5.3.8. S 形斜坡: 在运动过程中改变斜坡参数或切换到定位模式

但是，如果有必要在运动过程中改变“S”形斜坡的斜坡参数或从速度模式切换到定位模式，请执行以下操作:

操作步骤:

➢ 设置或再次设置正确的 BOW3 寄存器 0x2F，无论值是否改变。

i 设置完所有其他参数后，设置此参数。

结果:
重新计算内部斜坡，在该过程速度斜坡以安全模式运行。在此模式下，目标速度设置为 0。如果内部斜坡计算是最新的，由实际斜坡参数配置的斜坡将继续。

5.3.9. *带 ASTART 和 DFINAL 的“S”形斜坡配置*

配置 S 形斜坡的加减速的起始值和结束值，请执行以下操作:

➢ 设置 *RAMPMODE*(1:0)=b’10 (寄存器 0x20)。
➢ 设置如上所述的 S 形斜坡 (*BOW1 … BOW4, AMAX, DMAX)*。

操作步骤:

➢ 设置适合的 *ASTART* 寄存器 0x2A。
➢ 设置适合的 *DFINAL* 寄存器 0x2B。
➢ 设置适合的 *VMAX* 寄存器 0x24。

结果:
内部速度 *VACTUAL* 按照“S”形斜坡上升为 *VMAX*。

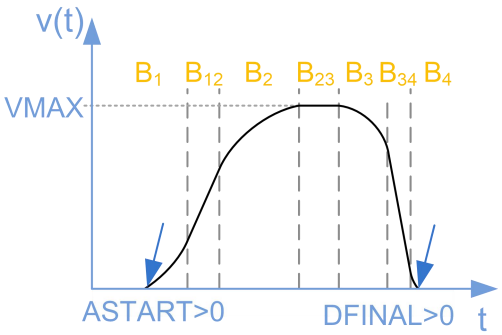


图 23: 带有初始和最终加速/减速值的 S 形斜坡

∞∀接下一页。

S 形斜坡的定义

- 加速/减速值根据弓形值而改变。
- S 形斜坡的开始阶段和结束阶段由 ASTART 和 DFINAL 控制加速/减速。
- 斜坡始于 ASTART，止于 *DFINAL*。
- 当 *AACTUAL* 达到选定的 *DFINAL* 值时，*DFINAL* 有效。

i 定位模式期间不考虑参数 DFINAL。

S 形斜坡的 *AACTUAL* 分配

S 形斜坡的 *AACTUAL* 分配及弓形值。
加速/减速系数取决于当前斜坡阶段且在 B1、B2、B3 和 B4 阶段每 64 个时钟周期更新一次。
详情见下表：

S 形斜坡:AACTUAL 和内部弓形值的分配								
斜坡阶段		**B_1**	**B_{12}**	**B_2**	**B_{23}**	**B_3**	**B_{34}**	**B_4**
v>0:	*AACTUAL*=	*ASTART→AMAX*	*AMAX*	*AMAX→0*	0	*0→-DMAX*	*-DMAX*	*-DMAX→-DFINAL*
	BOW_{ACTUAL}=	*BOW1*	0	*-BOW2*	0	*-BOW3*	0	*BOW4*
v<0:	*AACTUAL*=	*-ASTART→-AMAX*	*-AMAX*	*-AMAX→0*	0	*0→DMAX*	*DMAX*	*DMAX→DFINAL*
	BOW_{ACTUAL}=	*-BOW1*	0	*BOW2*	0	*BOW3*	0	*-BOW4*

表 17: S 形斜坡的参数分配

5.3.10. S 形斜坡和定位模式:快速运动

***RAMPMODE*(2:0) =b'110**

- 斜坡正好在目标位置结束；保持| *VACTUAL* | = *VMAX* 尽可能长的时间，直到斜坡下降恰好达到 *XTARGET*。
- 可以忽略 B12、B23 和 B34 的设置，因此最大速度定位斜坡可实现最高速度性能。
- 通常最快的斜坡过程不包含 S 形斜坡上升和/或下降期间的 B12 和/或 B34 阶段。
- 斜坡在定位模式下尽可能长时间地保持最大速度 *VMAX*，直到下降的斜坡完全达到 *XTARGET*。这是以最快时间完成定位过程的斜坡模式。

 阅读整个文档：是第 22 章(第 174 页)中的补充指令。

5.4. 起始速度 *VSTART* 和停止速度 *VSTOP*

S 形和梯形速度斜坡可以配置无符号的起始和停止速度值:*VSTART* 或 *VSTOP*。

默认情况下，*VSTART* 和 *VSTOP* 设置为 0。符号取决于当前斜坡状态和目标速度或目标位置。本节说明如何正确设置相应的值。

以初始速度启动斜坡

如果您将 *VSTART* 值设置为大于零(见下图)，S 形和梯形速度斜坡可以从初始速度值开始。

实现具有初始起始速度的梯形斜坡，请执行以下操作:

操作步骤:

- 设置 *RAMPMODE*(1:0)=b'01 (寄存器 0x20)。
- 如上所述，相应地设置梯形斜坡类型。
- 设置适合的 *VSTART* > 0 (寄存器 0x25)。
- 设置 *VSTOP* = 0 (寄存器 0x26)。

结果:

梯形斜坡从非零的初始速度开始运行。

注意:

→ *如果 VBREAK <VSTART，初始加速度值为 AMAX，否则初始加速度值为 ASTART。*

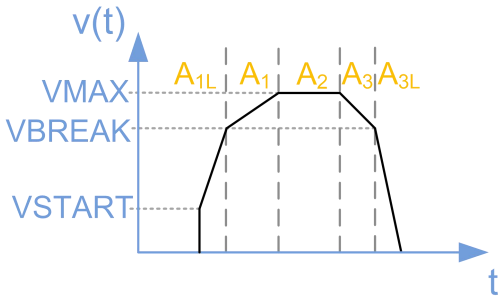


图 24: 具有初始速度的梯形斜坡

初始速度为 *VSTART* 的梯形斜坡:

注意

避免在定位模式期间出现不期望的系统动作！

- 仅在当前位置 *XACTUAL* 和目标位置 *XTARGET* 之间有足够的距离的定位模式下可以设置 VSTART，而不要求 *VSTOP* >*VSTART*。

这将确保定位模式下的平稳操作。

翻页了解配置带初始起始速度 S 形斜坡的信息。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

具有初始起始速度的 S 形斜坡

实现具有初始启动速度的 S 形斜坡，请执行以下操作:

操作步骤:

- 设置 *RAMPMODE*(1:0)=b'10 (寄存器 0x20)。
- 如上所述，相应地设置 S 形斜坡类型。
- 设置适合的 *VSTART* > 0 (寄存器 0x25)。
- 设置 *VSTOP* = 0 (寄存器 0x26).

结果:

S 形斜坡从非零的初始速度开始运行。

原则:

→ *初始加速度值等于 AMAX。不考虑参数 ASTART。因此，不执行斜坡阶段 B1。*

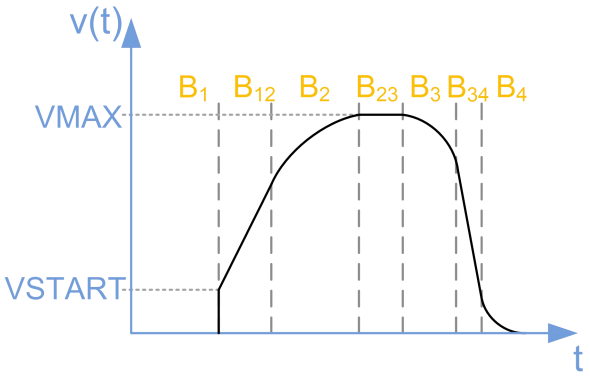


图 25:具有初始起始速度的 S 形斜坡

如果选择初始速度为 VSTART 的 S 形斜坡:

注意

避免在定位模式期间出现不期望的系统动作！

- 请记住，曲线的"S"形特征保持不变。因为 *AMAX* 是初始加速度值，斜坡将始终执行 B2 阶段，这可能产生位置位置过冲。
- 仅当当前位置 *XACTUAL* 和目标位置 *XTARGET* 之间有足够的距离的定位模式下使用 VSTART。

这将确保定位模式下的平稳操作。

ꕥ翻页了解如何配置用停止速度的结束 S 波形。

用停止速度结束斜坡

如果 *VSTOP* 设置大于零，S 形和梯形速度斜坡可以以停止速度结束(见下图)。

实现带停止速度的梯形斜坡，请执行以下操作:

操作步骤:

- 设置 *RAMPMODE*(1:0)=b'01 (寄存器 0x20)。
- 如上所述，相应地设置梯形斜坡类型。
- 设置 *VSTART* = 0 (寄存器 0x25)。
- 设置适合的 *VSTOP* > 0 (寄存器 0x26)。

结果:

梯形斜坡在定义的停止速度点停止斜坡运动。

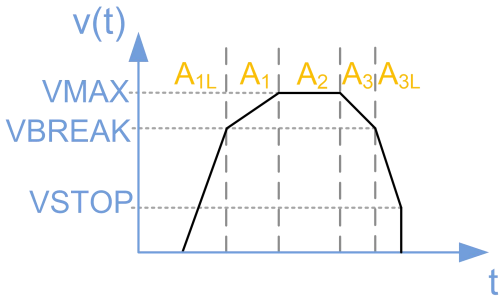


图 20: 带停止速度的梯形斜坡

如果选择梯形斜坡 (*VBREAK* > 0):

注意

避免在定位模式期间出现不期望的系统动作！

- 设置 *VBREAK* > *VSTOP*.
- 设置 *VSTART* < *VSTOP*.

这将确保定位模式下的平稳操作

ꕥ翻页查看带有停止速度的 S 形斜坡的配置信息。

具有停止速度的 S 形斜坡

实现带停止速度的 S 形斜坡，请执行以下操作:

操作步骤:

- 设置 *RAMPMODE*(1:0)=b'10 (寄存器 0x20)。
- 如上所述，相应地设置 S 形斜坡类型。
- 设置 *VSTART* = 0 (寄存器 0x25)。
- 设置正确的 *VSTOP* > 0 (寄存器 0x26)。

结果:

S 形斜坡以停止速度结束。

注意:

→ *最终减速度值等于 DMAX。不考虑参数 DFINAL，因此不执行斜坡阶段 B4。*

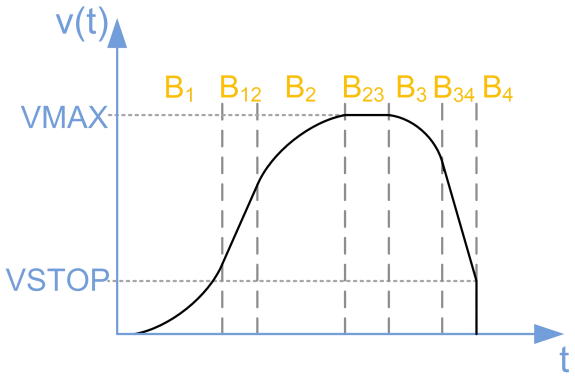


图 26: 具有停止速度的 S 形斜坡

VSTART, VSTOP, VACTUAL 和 VMAX 关联:

- 在定位模式下，当达到目标位置的时候可使用 *VSTOP*。在速度模式下，如果 *VACTUAL*≠ 0 并且目标速度 *VMAX* 为 0，也使用 *VSTOP*。
- *VSTART* 和 *VSTOP* 不仅用于开始或结束速度斜坡。如果在速度斜坡运行的过程中，重新配置寄存器分配引起电机运动方向的改变时，系统将根据当前速度斜坡类型，使用 *VSTART* 或 *VSTOP* 来控制。
- 无符号的 *VSTART* 和 *VSTOP* 寄存器对不同的速度方向都有效。
- 每个寄存器值的更新都是及时生效。

翻页了解如何配置具有开始和停止速度的 S 形斜坡。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

5.4.1. 具有初始启动和停止速度的 S 形斜坡

S 形斜坡可以配置 *VSTART* 和 *VSTOP* 的组合。
可以在一个 S 形斜坡中同时包括两个过程，以减少斜坡开始和停止的时间。

实现带初始启动速度和停止速度的“S”形斜坡，请执行以下操作

操作步骤:

- 设置 *RAMPMODE*(1:0)=b’10。
- 如上所述，相应地设置 S 形斜坡类型,但 BOW2 ≠ BOW4。
- 设置 适合的 *VSTART* > 0 (寄存器 0x25)。
- 设置 适合的 *VSTOP* > 0 (寄存器 0x26)。

结果:
S 形斜坡以初始启动速度 *VSTART* 开始，以停止速度 *VSTOP* 停止。

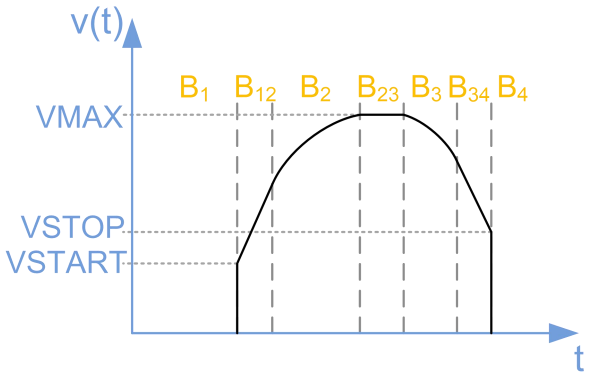


图 27: 具有开始和停止速度的 S 形斜坡

如果选择带初始速度 *VSTART* 和停止速度 *VSTOP* 的 S 形斜坡:

注意

避免在定位模式期间出现不期望的系统动作！

- 请记住，曲线的“S”形特征保持不变。因为 *AMAX* 是起始加速度值，斜坡将始终执行 B2 阶段，这可能导致位置过冲。
- 仅在当前位置 *XACTUAL* 和目标位置 *XTARGET* 之间足够远的定位模式下才使用 *VSTART*。

这将确保电机在定位模式下的平稳运行。

翻页了解如何在 S 形斜坡上使用 VSTART 和 ASTART。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

5.4.2. S 型斜坡的 VSTART 和 ASTART 的组合

对于某些 S 形斜坡应用，需要一个定义的初始启动速度值(*VSTART*>0)但不需要最大加速度值 *AMAX*。

实现电机从定义的速度值开始启动，请执行以下操作:

操作步骤:

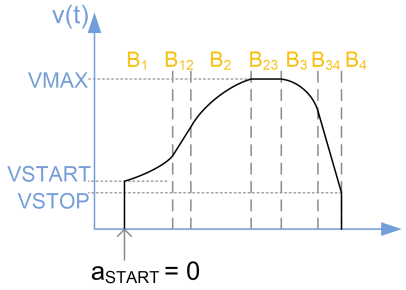
- 设置 *RAMPMODE*(1:0) =b'10 (寄存器 0x20)。
- 如上所述，相应地设置 S 形斜坡类型。
- 设置 适合的 *VSTART* > 0 (寄存器 0x25)。
- 设置 适合的 *VSTOP* > 0 (寄存器 0x26)。
- 设置 *use_astart_and_vstart* =1 (*GENERAL_CONF* 寄存器 0x00 第 0 位)。

结果:

i 尽管使用了 *VSTART*，B1 阶段还是有效。

通过这种方式生成以下特殊斜坡类型，如下所示。

带 VSTART 及启动加速度为 0 的 S 形斜坡



带 VSTART 及启动加速度小于 AMAX 的 S 形斜坡

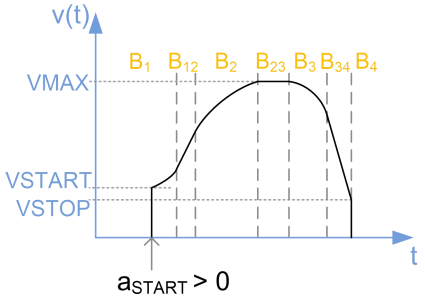


图 28: 带 VSTART 和 ASTART 参数的 S 形斜坡

如果配置了 *VSTART, ASTART, 和 VSTOP* 参数的 S 型斜坡:

注意

避免在定位模式期间出现不期望的系统动作！

- 请记住，曲线的"S"形特征保持不变。因为 *ASTART* 是起始加速度值，斜坡将始终执行 B2 阶段，这可能导致位置过冲。
- 仅在当前位置 *XACTUAL* 和目标位置 *XTARGET* 之间足够远的定位模式下可以不要求在 *VSTOP* > *VSTART* 的情况下设置 *VSTART* 和 *ASTART* > 0。

这将确保电机在定位模式下的平稳运行。

阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

5.5. sixPoint 六点斜坡

sixPoint 六点斜坡是具有初始和停止速度的梯形斜坡，同时有两个加速度和两个减速度值。

sixPoint 六点斜坡配置

sixPoint 六点斜坡是梯形速度斜坡，可以结合 *VSTART* 和 *VSTOP* 进行配置。

实现带初始启动速度和停止速度的梯形斜坡，请执行以下操作:

操作步骤:

- 设置 *RAMPMODE*(1:0)=b'01 (寄存器 0x20)。
- 根据章节 6.3.6，第 35 页设置梯形斜坡 。
- 设置 适合的 *VSTART* > 0 (寄存器 0x25)。
- 设置 适合的 *VSTOP* > 0 (寄存器 0x26)。
- 设置 适合的 *VBREAK* > 0 (寄存器 0x27)。

结果:

sixPoint **六点斜坡以初始启动速度开始，以规定的停止速度停止。**

sixPoint 斜坡框图

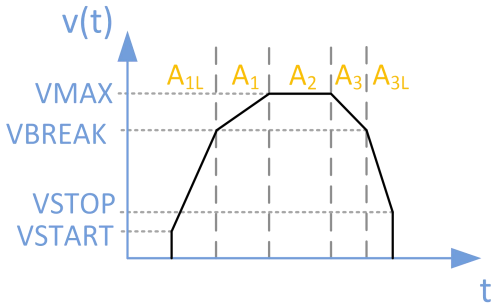


图 29: 六点斜坡:带初始启动和停止速度的梯形斜坡

如果选择 sixPoint 六点斜坡模式

注意

避免在定位模式期间出现不期望的系统动作！

- 设置 *VBREAK* > *VSTOP*。
- 设置 *VSTART* < *VSTOP*。

这将确保电机在定位模式下的平稳运行。

5.6. U-Turn 行为

下面介绍了运动过程中运动方向发生变化的过程，适用于所有斜坡类型。

U-Turn 行为

如果在速度模式(设置 *VMAX*)或定位模式(重新设置 *XTARGET*)的运动过程中运动方向发生变化，将触发以下过程:

1. 运动朝着 *VACTUAL* = 0 的方向运动 。
   - **i** 如果设置了 *VSTOP* (≠ 0)，运动停止在 *VSTOP*。
   - **i** 如果使用 *VSTOP* 和/或梯形斜坡类型，建议将 *TZEROWAIT* 设置为大于 0，否则可能会发生电机振荡，必须消除振荡。
2. 等待 *TZEROWAIT* 时钟周期(寄存器 0x7B)后进入静止阶段。
3. 继续运动到实际的 *XTARGET*(定位模式)或新分配的 *VMAX*(速度模式)。
   - **i** 如果设置了 *VSTART*(≠0)， *TZEROWAIT* > 0，运动从 *VSTART* 开始。

示例:
六点斜坡的 U-Turn

到达 *VSTOP* 后，等待 *TZEROWAIT* 时钟周期，直到消除电机振荡运动继续。

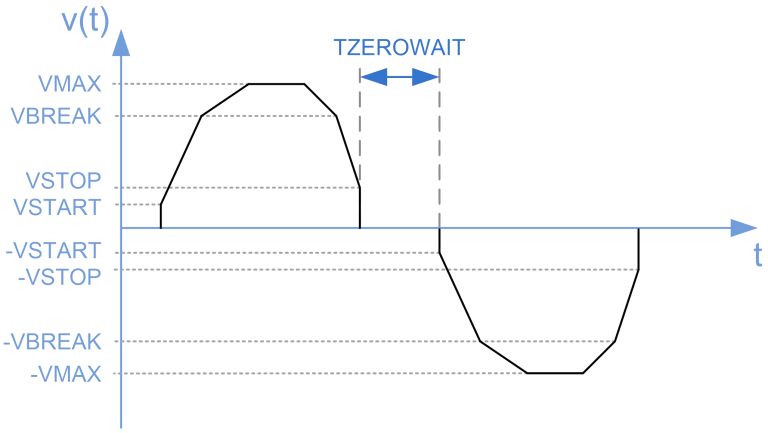


图 30: 六点斜坡的 U-Turn 行为示例

∞∀翻页查看关于"S"形斜坡 U-TURN 信息。

示例:
S 型斜坡的 U-Turn

当速度达到 *VACTUAL* = 0 时，立刻继续运动。大多数不使用 *VSTOP* 的 S 形斜坡应用不需要静止阶段。
如果设置了 *ASTART* > 0 和/或 *DFINAL* >0，在 U-Turn 实现过程也会调用和遵从这些参数。

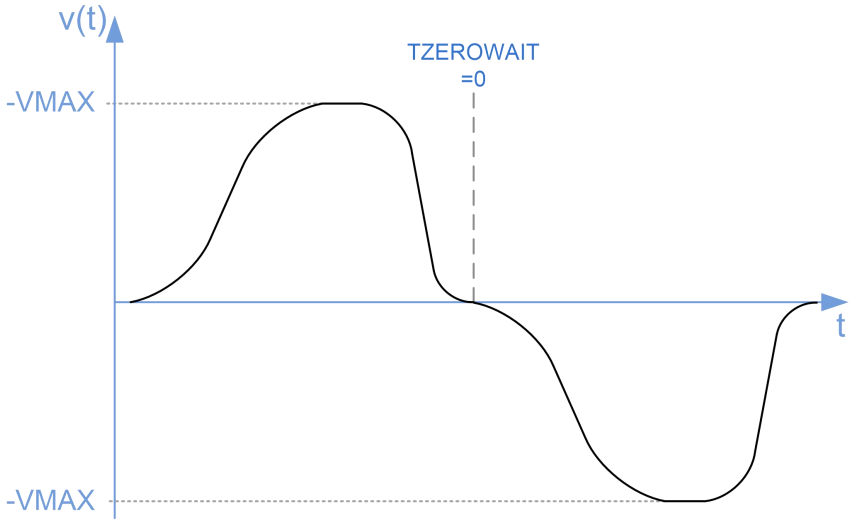


图 31: S 型斜坡的 U-Turn 斜坡

5.6.1. S 形斜坡的连续速度运动轮廓

上述 U-TURN 过程有一个例外:
如果 ***BOW2* 等于 *BOW4***，在 *VACTUAL* = 0 时，S 形斜坡不会停止。经过 *VACTUAL* = 0 时，运动加速度不等于 0。因此，实现了最快的斜坡 U-Turn 行为。

下图，该速度斜坡行为被描绘为粗黑线，而上述的速度斜坡行为被描绘为灰色线:

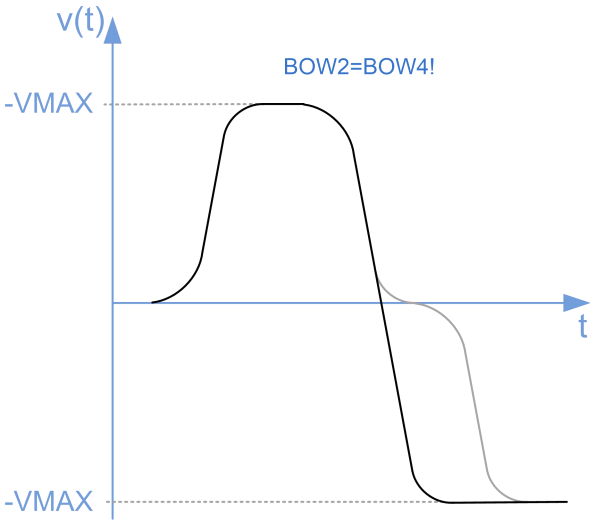


图 32: S 型斜坡穿过 VACTUAL=0 直接运行

5.7. 内部斜坡发生器单元

本节提供斜坡参数的算术相关的单位信息。

5.7.1. 时钟频率

所有参数单位都是算术单位。
因此，有必要设置 *CLK_FREQ* 寄存器 0x31 为合适的外部时钟频率 f_{CLK} [Hz]值。f_{CLK} 频率范围为 4.2 MHz 到 32 MHz。

默认为 16 MHz.

5.7.2. 速度值单元

速度值单位始终定义为每秒脉冲数[pps]。
VACTUAL 是 32 位有符号数，没有小数位。
速度值 *VSTART*、*VSTOP* 和 *VBREAK* 由 23 位整数和 8 位小数的无符号数组成。
VMAX 是 24 位整数和 8 位小数的有符号数。

最大速度 *VMAX* 如下:

速度模式: $|VMAX| \leq \frac{1}{2} f_{CLK}$

定位模式: $|VMAX| \leq \frac{1}{4} f_{CLK}$

注意:

→ *如果 VACTUAL 超过此限值，STPOUT 输出端会出现不正确的步进脉冲和/或定位不正确。*

此外，*VMAX* 必须大于所有速度值中的最高值:

$|VMAX| > \max(VSTART;VSTOP;VBREAK)$

5.7.3. 加速度值单位

无符号值 *AMAX*、*DMAX*、*ASTART*、*DFINAL* 和 *DSTOP* 由 22 位整数和两位小数组成。
AACTUAL 为 32 位有符号非整数值。加速和减速单位默认定义为每平方秒的脉冲[pps^2]。

如果短斜坡和陡坡需要更高的加速/减速值，请执行以下操作:

操作步骤:

➢ 设置 *direct_acc_val_en* =1 (*GENERAL_CONF* 寄存器 0x00)。

结果:

参数定义为每个时钟周期的速度值变化，小数位数为 24 位无符号(MSB =2^{-14})。数值计算如下:

$AMAX\ [pps^2] = AMAX / 2^{37} \cdot f_{CLK}^2$

$DMAX\ [pps^2] = DMAX / 2^{37} \cdot f_{CLK}^2$

$ASTART\ [pps^2] = ASTART / 2^{37} \cdot f_{CLK}^2$

$DFINAL\ [pps^2] = DFINAL / 2^{37} \cdot f_{CLK}^2$

$DSTOP\ [pps^2] = DSTOP / 2^{37} \cdot f_{CLK}^2$

最大加速度或减速度值如下:

$\max(AMAX;DMAX;ASTART;DFINAL;DSTOP)\ [pps^2] \leq VMAX \cdot f_{CLK} / 1024$

如果 *direct_acc_val_en* = 1, 最大值也受限于:

$\max(AMAX;DMAX;ASTART;DFINAL;DSTOP) \leq 2^{20}$

∞∀接下一页。

5.7.4. 弓形值

弓形值 BOW1…BOW4:

弓形值是不带小数的 24 位无符号数。单位默认为每立方秒的脉冲[pps³]。

如果短斜坡和陡坡需要更高的弓形值，请执行以下操作:

操作步骤:

➢ 设置 *direct_bow_val_en* =1 (*GENERAL_CONF* 寄存器 0x00)

结果:

参数定义为每个时钟周期的加速度值变化，为 24 位无符号小数，MSB 定义为 2^{-29}。

特定的弓值 *BOW1*, *BOW2*, *BOW3*, *BOW4* 计算如下:

$$BOWx\ [pps^3] = BOWx / 2^{53} \cdot f_{CLK}^3$$

最大弓形值如下:

$$\max(BOW1…4)\ [pps^2] \le \max(AMAX;DMAX)\ [pps^2] \cdot f_{CLK} / 1024$$

如果 *direct_bow_val_en* = 1, 最大值也限于:

$$\max(BOW1…4) \le 2^{20}$$

5.7.5. 最小值和最大值概述:

最小值和最大值(频率模式 及 一般情况)				
参数类别	**速度**	**加速度**	**弓形**	**时钟**
受影响的寄存器	*VMAX, VSTART, VSTOP, VBREAK*	*AMAX, DMAX, ASTART, DFINAL*	*BOW1, BOW2, BOW3, BOW4*	*CLK_FREQ* (f_{CLK})
最小标称值	3.906 mpps	0.25 $mpps^2$	1 $mpps^3$	4.194 MHz
最大标称值	8.388 Mpps	4.194 $Mpps^2$	16.777 $Mpps^3$	32 MHz
最大相关值	速度模式: ½ · f_{CLK}	*VMAX* · f_{CLK} / 1024	max(*AMAX;DMAX*) · f_{CLK} / 1024	
	定位模式: ¼ · f_{CLK}			
	|*VMAX*| > max(*VSTART;VSTOP;VBREAK*)			

表 18: 选择现实物理单位时的最小值和最大值

陡坡的最小值和最大值(直接模式，例如 f_{CLK} = 16MHz)		
参数类别	**加速度(*direct_acc_val_en* =1)**	**弓形值(*direct_bow_val_en* =1)**
受影响的寄存器	*AMAX, DMAX, ASTART, DFINAL, DSTOP*	*BOW1, BOW2, BOW3, BOW4*
计算	a[pps^2] = (Δv/clk_cycle) / $2^{37} \cdot f_{CLK}^2$	bow[pps^3] = (Δa/clk_cycle) / $2^{53} \cdot f_{CLK}^3$
最小标称值	~1.86 $kpps^2$	~454.75 $kpps^3$
最大标称值	~1.95 $Gpps^2$	~476.837 $Gpps^3$
最大相关值	*VMAX* · 15625 Hz	max(*AMAX;DMAX*) · 15625 Hz

表 19: f_{CLK} =16MHz 陡坡情况下的最小值和最大值

6. 参考开关

TMC4361A 的参考输入信号部分用作安全功能。TMC4361A 提供一系列参考开关设置，可针对许多不同应用进行配置。TMC4361A 包含两个硬件开关(STOPL, STOPR)、两个额外的虚拟停止开关(*VIRT_STOP_LEFT*, *VIRT_STOP_RIGHT*)及一个回零开关 HOME_REF。

参考开关的引脚		
引脚名称	**类型**	**备注**
STOPL	输入	左参考开关。
STOPR	输入	右参考开关。
HOME_REF	输入	回零开关。
TARGET_REACHED	输出	参考开关表明 *XACTUAL=XTARGET*。

表 22:参考开关的引脚

参考开关专用寄存器			
寄存器名称	**寄存器地址**		**备注**
REFERENCE_CONF	0x01	RW	与参考引脚相关的配置。
HOME_SAFETY_MARGIN	0x1E	RW	*X_HOME* 周围的不确定区域。
DSTOP	0x2C	RW	如果停止开关 STOPL / STOPR 或虚拟停止设置了软停功能，则 DSTOP 对应减速值。该值支持自动线性停止斜坡。
POS_COMP	0x32	RW	自由配置的比较位置；有符号；32 位。
VIRT_STOP_LEFT	0x33	RW	当 *XACTUAL* ≤ *VIRT_STOP_LEFT* 时触发虚拟左停止的停止事件; 有符号；32 位。
VIRT_STOP_RIGHT	0x34	RW	当 *XACTUAL* ≥ *VIRT_STOP_RIGHT* 时触发虚拟右停止的停止事件; 有符号；32 位。
X_HOME	0x35	RW	回零参考位置; 有符号；32 位。
X_LATCH	0x36	RW	在不同条件下锁存当前位置 *XACTUAL*; 有符号；32 位。

表 23: 参考开关专用寄存器

6.1. 硬件开关支持

TMC4361A 有两个硬件开关，可根据您的设计进行配置。

STOPL 和 STOPR 硬件有左和右停止开关，以便在其中一个被触发时立即停止驱动，对应运动控制芯片的引脚 12 和引脚 14。

注意:

→ *运动发生前，必须使能。*

正确使能 STOPL，请执行以下操作：

操作步骤:

- 确定 STOPL 的有效电压极性并设置对应的 *pol_stop_left* (*REFERENCE_CONF* 寄存器 0x01)。
- 设置 *stop_left_en* =1 (*REFERENCE_CONF* 寄存器 0x01)。

结果:

如果 STOPL 电压电平与 *pol_stop_left* 相符且满足 *VACTUAL <0* 匹配，则当前速度斜坡停止。

正确使能 STOPR，请执行以下操作:

操作步骤:

- 确定 STOPR 的有效电压极性并设置对应的 *pol_stop_right* (*REFERENCE_CONF* 寄存器 0x01)。
- 设置 *stop_right_en* =1 (*REFERENCE_CONF* 寄存器 0x01)。

结果:

如果 STOPR 电压电平与 *pol_stop_right* 相符且满足 *VACTUAL* > 0，则当前速度斜坡停止。

6.1.1. 硬或线性停止的停止坡配置

停止斜坡可以配置为硬停或线性停。默认情况下，选择硬停。

如果需要硬停止，请执行以下操作:

选项 1:硬停止斜坡

操作步骤:

- 设置 *soft_stop_en* =0 (*REFERENCE_CONF* 寄存器 0x01)。

结果:

如果其中一个停止开关有效并使能停止功能，*VACTUAL* = 0。

选项 2:线性停止斜坡

如果需要线性停止，请执行以下操作:

操作步骤:

- 设置正确的 *DSTOP* > max(*DMAX*; *DFINAL*) (寄存器 0x2C)。
- 设置 *soft_stop_en* =1 (*REFERENCE_CONF* 寄存器 0x01)。

结果:

如果其中一个停止开关有效并使能停止功能，速度斜坡将以线性减速斜率停止，直到达到 *VACTUAL* = 0。此时减速系数由 *DSTOP* 决定。减速停止的斜坡期间不考虑 *VSTOP*。

6.1.2. 如何指示有效的停止并将其重置为自由运动

当使能的停止开关变为有效状态时，相关状态标志在状态标志寄存器 *STATUS* 0x0F 中设置。只要停止开关保持在有效状态不变，标志就保持不变。
特定事件也会在 *EVENTS* 寄存器 0x0E 中有效并保持，直到事件位被手动复位。当停止事件达到 *VACTUAL* = 0 时，不会朝该方向运动。

向锁定方向移动，需要以下条件:

前提条件 1:

相应的停止开关不再有效。

与/或

前提条件 2:

停止开关被禁用(*stop_left/right_en* = 0)。

操作步骤:

➢ 读 *EVENTS* 寄存器 0x0E 清事件标志。

i 参见第 5.1 节第 25 页中关于清除事件的信息。

结果:

有效的停止事件复位，电机可以向锁定方向自由运动。

6.1.3. 如何锁定开关事件的内部位置

可选择四种不同的事件将当前内部位置 *XACTUAL* 存储到寄存器 *X_LATCH* 中。

下表显示了参考信号和 *X_LATCH* 的关系。对于每个转换过程，必须相应地设置 *REFERENCE_CONF* 寄存器 0x01 中的指定参考配置。

参考配置	*pol_stop_left*=0	*pol_stop_left*=1	*pol_stop_right*=0	*pol_stop_right*=1
latch_x_on_inactive_l=1	STOPL=0 → 1	STOPL=1 → 0	---	---
latch_x_on_active_l=1	STOPL=1 → 0	STOPL=0 → 1	---	---
latch_x_on_inactive_r=1	---	---	STOPR=0 → 1	STOPR = 1→0
latch_x_on_active_r=1	---	---	STOPR=1 → 0	STOPR = 0→1

表 24: 参考配置与相应参考开关之间的转换

无需物理重新连接即可交换参考交换开关

需要更改参考开关的方向，请执行以下操作:

操作步骤:

➢ 设置 *invert_stop_direction*=1 (*REFERENCE_CONF* 寄存器 0x01)。

结果:

STOPL 现在是右参考开关，STOPR 现在是左参考开关。因此，STOPL 的所有配置参数对 STOPR 都有效，反之亦然。

阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

6.2. 虚拟限位开关

TMC4361A 提供额外的虚拟限位；当位置达到对应的虚拟停止开关微步位置时触发停止。虚拟停止位置包含 *VIRTUAL_STOP_LEFT* 寄存器 0x33 和 *VIRTUAL_STOP_RIGHT* 0x34 寄存器。本节介绍了虚拟停止开关的配置细节以满足不同的设计要求。

注意:

→ *虚拟停止开关必须按照非虚拟参考开关相同的方式使能。与 STOPL 或 STOPR 过程相同，按下虚拟限位开关——通过接收指定的位置——触发。*

6.2.1. 使能虚拟停止开关

使能虚拟停止开关，请执行以下操作:

操作步骤:

- 根据左停止位置设置 *VIRTUAL_STOP_LEFT* 寄存器 0x33。
- 设置 *virtual_left_limit_en* =1 (*REFERENCE_CONF* 寄存器 0x01)。

结果:
当 *XACTUAL* ≤ *VIRT_STOP_LEFT* 时，实际速度斜坡根据所选斜坡类型停止斜坡。

使能右虚拟停止开关，，请执行以下操作:

操作步骤:

- 根据右停止位置设置 *VIRTUAL_STOP_RIGHT* 寄存器 0x34 。
- 设置 *virtual_right_limit_en* =1 (*REFERENCE_CONF* 寄存器 0x01)。

结果:
当 *XACTUAL* ≥ *VIRT_STOP_RIGHT* 时，实际速度斜坡根据所选斜坡类型停止斜坡。

6.2.2. 虚拟停止斜坡配置

虚拟停止斜坡也可以配置为硬停或线性停。

实现虚拟硬停，请执行以下操作:

操作步骤:

- 设置 *virt_stop_mode* = b'01 (*REFERENCE_CONF* 寄存器 0x01)。

结果:
如果其中一个虚拟停止开关有效并使能停止功能，速度斜坡将立即设置为 *VACTUAL* = 0。

实现虚拟线性停止，请执行以下操作:

操作步骤:

- 设置适合的 *DSTOP* > max(*DMAX*; *DFINAL*) (寄存器 0x2C)。
- 设置 *virt_stop_mode* = b'10 (*REFERENCE_CONF* 寄存器 0x01)。

结果:
如果其中一个虚拟停止开关有效并使能停止功能，速度斜坡将以线性减速斜率停止，直到达到 *VACTUAL* = 0。此时减速系数由 *DSTOP* 决定。减速停止的斜坡期间不考虑 *VSTOP*。

∀接下一页。

6.2.3. 如何指示有效的虚拟停止并将其重置为自由运动

同时，当使能的虚拟停止开关变为有效状态时，状态标志寄存器 *STATUS* 0x0F 中的相关状态标志被激活。只要停止开关保持有效状态，标志就保持不变。

特定事件也会在 *EVENTS* 寄存器 0x0E 中有效并保持，直到事件被手动清除。当停止事件达到 *VACTUAL* = 0 时，不会朝该方向运动。

向锁定方向运动，需要以下条件:

前提条件 1:

因为实际位置没有超过规定的极限，因此相应的停止开关不再有效。

与/或

前提条件 2:

虚拟停止开关未使能(*virtual_left/right_limit_en* = 0)。

操作步骤:

➢ 读 *EVENTS* 寄存器 0x0E 清时间标志。

i 参见第 5.1.节第 25 页中关于清除事件的信息描述。

结果:

有效的虚拟停止事件复位，电机可以向锁定方向自由运动。

i *invert_stop_direction* 对 *VIRTUAL_STOP_LEFT* 及 *VIRTUAL_STOP_RIGHT* 没有影响。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

6.3. 回零参考配置

本节解释回零参考开关的处理、回零跟踪模式、回零事件配置及回零事件监控相关的信息。开关参考输入的引脚为 HOME_REF。

参考输入开关 HOME_REF

执行以下操作初始化回零过程:

- 根据回零过程的需要分配斜坡模式。

操作步骤:

- *start_home_tracking* = 1 使能回零跟踪模式(*REFERENCE_CONF* 寄存器 0x01)。
- 根据 HOME_REF 输入引脚(见下表)设置正确的 *home_event* (*REFERENCE_CONF* 寄存器 0x01)。
- 向回零开关 HOME_REF 方向开始斜坡运动。

结果:

- 当下一个回零事件发生时，*XACTUAL* 被锁存到 *X_HOME*。
- 同时，如果清除 *XLATCH_DONE* 事件，则自动禁用 *start_home_tracking* 开关。
- *XLATCH_DONE* 事件在事件寄存器 0x0E 有效。此事件可用于回零过程的中断信号替换查询操作。

i 如果运动中有增量编码器，则可以使用 N 通道来微调回零位置(*home_event* = b'0000)。如前所述，在执行回零过程之后，可以使用 N 信号事件来获得更精确的回零位置。

i *X_HOME* 可被手动重置。

6.3.1. 回零事件选择

支持九种不同的回零事件。

i 除了 *home_event* = b'0000 对应增量编码器的 N 索引通道外，回零事件与 HOME_REF 输入引脚相关:

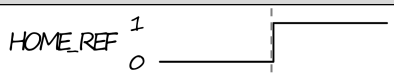
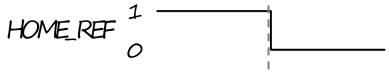
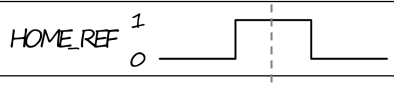
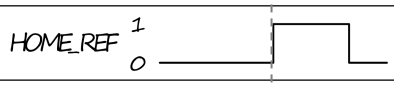
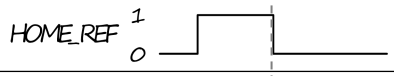
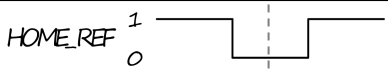
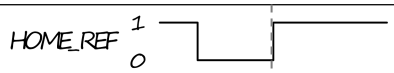
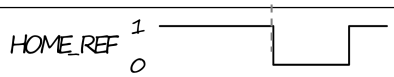
回零事件选择表			
home_event	**描述**		**X_HOME (方向: 负/正)**
b'0011	*HOME_REF* = 0 指示低电平对应 X_HOME 的反方向		
b'1100	*HOME_REF* = 0 指示低电平对应 X_HOME 的正方向		
b'0110	*HOME_REF* = 1 指示回零位置	X_HOME 在中间	
b'0010		X_HOME 在左边	
b'0100		X_HOME 在右边	
b'1001	*HOME_REF* = 0 指示回零位置	X_HOME 在中间	
b'1011		X_HOME 在右边	
b'1101		X_HOME 在左边	

表 25: 不同 home_event 设置概述

6.3.2. HOME_REF 监测

芯片将一直监测错误标志 *HOME_ERROR_F*。该错误标志指示 *HOME_REF* 参考输入的当前电压电平对于 *X_HOME* 和选定的 home_event 是否有效。

定义 HOME_REF 的回零范围

为了避免由于机械不确定性造成的错误标志(*HOME_ERROR_F*)，可以设置 *X_HOME* 周围的不确定回零范围寄存器。在此范围内，不会监测错误标志。

定义 X_HOME 的不确定区域，请执行以下操作:

操作步骤:

➢ 根据所需的范围[ustep]设置 *HOME_SAFETY_MARGIN* 寄存器 0x1E 。

结果:

正在进行的运动应考虑与应用环境相关的不确定性，因此错误标志不在以下范围内有效:

X_HOME − HOME_SAFETY_MARGIN ≤ XACTUAL ≤ X_HOME + HOME_SAFETY_MARGIN

注意:

→ *当 home_events 为 b'0110、b'0010、b'0100、b'1001、b'1011 和 b'1101 时，建议设置较高的 HOME_SECURITY_MARGIN 大于 HOME_REF 有效长度。避免出现错误的 HOME_ERROR_Flags 标志。*

→ *当用索引通道(home_event = b'0000) 精确设置X_HOME 完成回零后，必须正确设置 home_event，以便激活HOME_ERROR_Flags 的生成。请注意，home_event = b'0000 会一直使HOME_ERROR_Flag=0。*

→ *以下示例说明了HOME_REFE 引脚根据所选home_event 出现错误标志的点，此处为 home_event =b'0011 (*), b'1100(**), b'0110 (***), b'0010 (***), b'0100 (***), b'1001 (****), b'1011 (****),和b'1101 (****)。*

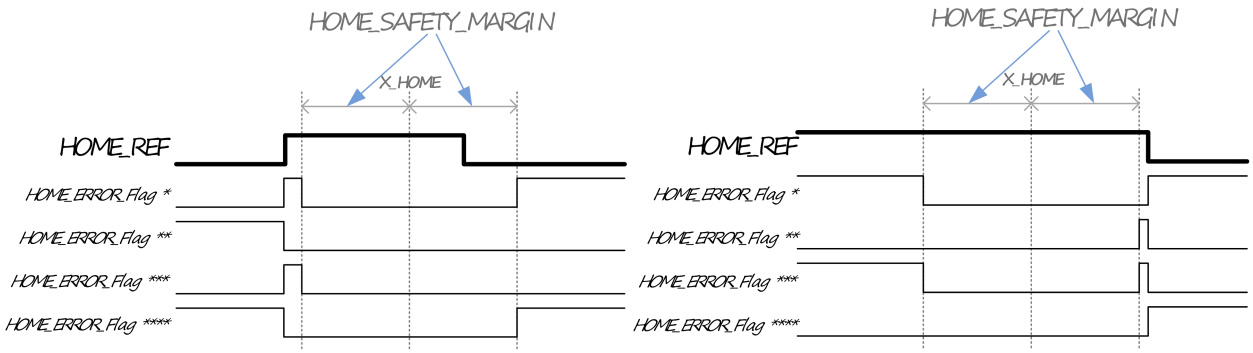


图 33: HOME_REF 监测和 HOME_ERROR_FLAG

阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

6.3.3. 用 STOPL 或 STOPR 做回零

STOPL 和 STOPR 输入可以作为 HOME_REF 输入。

选项 1: STOPL 是回零开关

操作步骤:

- 设置 *stop_left_is_home* = 1 (*REFERENCE_CONF* 寄存器 0x01)。

结果:

STOPL 处的停止事件仅在 STOPL 输入有效后越过回零范围时发生。*X_HOME* 和 *HOME_SAFETY_MARGIN* 设置了回零范围。

OPTION 2: STOPR 是回零开关

操作步骤:

- 设置 *stop_right_is_home* = 1 (*REFERENCE_CONF* 寄存器 0x01)。

结果:

STOPR 处的停止事件仅在 STOPR 输入有效后越过回零范围时发生。*X_HOME* 和 *HOME_SAFETY_MARGIN* 设置了回零范围。

阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

6.4. 目标到达/位置比较

本节将解释 TARGET_REACHED 输出引脚配置，以及如何实现内部比较的不同方法。

目标已达到输出引脚

TARGET_READED 输出引脚对应 *TARGET_REACHED_Flag*。一旦 *XACTUAL* 等于 *XTARGET*，TARGET_REACHED 就一直有效。默认情况下，TARGET _ REACHED 引脚为高电平有效。

更改 TARGET _ REACHED 输出极性，请执行以下操作:

操作步骤:

- 设置 *invert_pol_target_reached* = 1 (*GENERAL_CONF* 寄存器 0x00 的第 16 位)。

结果:

TARGET_REACHED 引脚低有效。

6.4.1. 连接几个 TARGET_REACHED 引脚

TARGET_REACHED 引脚也可以配置为共享信号线，就像几个 INTR 引脚可以配置成一个中断信号一样(参见第 5.4 节第 27 页)

实现“线或”或者“线与”，必须执行以下操作:

操作步骤:

- **步骤 1:** 设置 *intr_tr_pu_pd_en* = 1 (*GENERAL_CONF* 寄存器 0x00)。

选项 1: 线或

操作步骤:

- **步骤 2:** 设置 *intr_as_wired_and* = 0 (*GENERAL_CONF* 寄存器 0x00)。

结果:

TARGET_REACHED 引脚配置成线或(默认配置)。

i 如果 TARGET_REACHED 无效，引脚为弱输出。有效时，引脚为强输出。因此，如果其中一个连接引脚被激活，整条线路将被设置为有效极性。

选项 2: 线与

操作步骤:

- **步骤 2:** 设置 *intr_as_wired_and* = 1 (*GENERAL_CONF* 寄存器 0x00)。

结果:

只要没有到达目标位置，TARGET_REACHED 为非有效极性的强输出。到达目标位置，引脚为有效极性的弱输出。因此，只有所有连接的引脚都产生有效极性，则整个信号线有效。

6.4.2. TARGET_REACHED 输出的使用

默认情况下，TARGET _ READED 引脚对应 *TARGET_REACHED_Flag*，表示 *XACTUAL = XTARGET*。该引脚还可用于对应其他三个标志:*VELOCITY_REACHED _Flag*、*ENC_FAIL_Flag*、*POS _COMP_REACHED_Flag*。

注意:

→ *只能选择一个选项*

四个 TARGET_REACHED 选项

REFERENCE_CONF 寄存器 0x01 配置了 TARGET_REACHED 输出引脚。

可选择的选项为:

TARGET_REACHED 输出引脚配置	
如果 *pos_comp_output*…	**则 TARGET_REACHED 输出…**
b’00	*TARGET_REACHED_*Flag
b’01	*VELOCITY_REACHED_*Flag
b’10	*ENC_FAIL_*Flag
b’11	*POS_COMP_REACHED_*Flag

表 26: TARGET_REACHED 输出引脚配置

6.4.3. 内部值的位置比较

TMC4361A 提供了几种内部比较的方法。位置比较一直有效，对应一个标志位和一个事件。如章节 8.4.2 第 54 页所述，可以通过 INTR 引脚使用中断源的“*POS_COMP_REACHED*”事件，或者使用“TARGET_REACHED”引脚表示明确的比较结果。

基本比较设置

如何比较内部位置与任意值:

操作步骤:

- 在 *POS_COMP* 寄存器 0x32 设置一个适合的数值。
- 选择 *pos_comp_source* = 0 (*REFERENCE_CONF* 寄存器 0x01)。

结果:

XACTUAL 与 *POS_COMP* 比较。当 *POS_COMP* 等于 *XACTUAL* 时，*POS_COMP_REACHED_Flag* 有效，*POS_COMP_REACHED* 事件有效。

选择外部位置作为比较源

如何比较外部位置与任意值:

操作步骤:

- 在 *POS_COMP* 寄存器 0x32 设置一个适合的数值。
- 选择 *pos_comp_source* = 1 (*REFERENCE _CONF* 寄存器 0x01)。

结果:

ENC_POS 与 *POS_COMP* 比较。当 *POS_COMP* 等于 *ENC_POS* 时，*POS_COMP_REACHED_Flag* 有效，*POS_COMP_REACHED* 事件有效。

注意:

→ *因为 ENC_POS 是电机微步数而不是编码器的数值，因此与外部位置比较过程的 POS_COMP 也是微步数。*

→ *如果 ENC_POS 运动经过 POS_COMP，而 ENC_POS 没有与 POS_COMP 一样的数值，则不会标记 POS_COMP_REACHED 事件，但会将其列在 EVENTS 寄存器中，以指示它已经经过。*

比较选择网格

除了将 *XACTUAL* / *ENC_POS* 与 *POS_COMP* 进行比较之外，还可以将两个参数中的一个分别与 *X_HOME* 或 *X_LATCH*/*ENC_LATCH* 进行比较。TMC4361A 还允许将转数计数器 *REV_CNT* 与 *POS_COMP* 进行比较。

设置注意事项

只有选定的组合会生成 *POS_COMP_REACHED*_Flag 和相应的事件。因此，在 *REFERENCE_CONF* 寄存器 0x01 中选择 *modified_pos_compare*，如下表所示:

比较选择网格		
modified_pos_compare \ *pos_comp_source*	**‘0’**	**‘1’**
‘00’	*XACTUAL* vs. *POS_COMP*	*ENC_POS* vs. *POS_COMP*
‘01’	*XACTUAL* vs. *X_HOME*	*ENC_POS* vs. *X_HOME*
‘10’	*XACTUAL* vs. *X_LATCH*	*ENC_POS* vs. *ENC_LATCH*
‘11’	*REV_CNT* vs. *POS_COMP*	

表 27: POS_COMP_REACHED_Flag 的比较选择表格

6.5. 重复和圆周运动

TMC4361A 还提供自动重复或自动圆周运动选项。本节将解释配置选项。

6.5.1. 到 XTARGET 的重复运动

默认情况下，在定位模式下运动到 *XTARGET* 则完成定位斜坡。

实现连续重复指定的斜坡，请执行以下操作:

前提条件:

- 设置 *RAMPMODE*(2) = 1 (定位模式有效)。
- 根据您的要求配置速度斜坡。

操作步骤:

- 设置 *clr_pos_at_target* =1 (*REFERENCE_CONF* 寄存器 0x01)。

结果:

达到 *XTARGET* 后(*TARGET_REACHED_*Flag 有效)，*XACTUAL* 设置为 0。只要 *XTARGET* 不为 0，斜坡重新启动，以便再次到达 *XTARGET*。实现从 0 到 *XTARGET* 的重复定位斜坡。

注意:

→ *在重复运动过程中可以改变 XTARGET。一旦 XCATUAL 等于 XTARGET 时，总是将 XACTUAL 重置为 0。*

6.5.2. 使能圆周运动

TMC4361A 可以对 *XACTUAL* 的位置限制范围做自动溢出处理实现某些圆周运动。一旦 *XACTUAL* 达到两个位置范围限制之一(正/负)，*XACTUAL* 将自动设置为相反的位置限制值。

实现圆周运动，请执行以下操作:

前提条件:

圆周运动要求 *XACTUAL* 必须位于定义的位置范围内。

处理:

操作步骤:

- 设置 *X_RANGE* ≠ 0 (寄存器 0x36, 只写访问！)。
- 设置 *circular_motion* = 1 (*REFERENCE_CONF* 寄存器 0x01)。

结果:

XACTUAL 的定位范围限于: *–X_RANGE* ≤ *XACTUAL* < *X_RANGE*。

当 *XACTUAL* 到达正位置(*X_ RANGE*–1)并且运动朝正的方向进行时；下一个 *XACTUAL* 值设置为*-X_RANGE*。同样适用于反方向的处理；其中(*X_RANGE*–1)是*-X_RANGE* 之后的位置。

i 定位模式下，运动方向取决于到达目标位置 *XTARGET* 的最短路径。例如，如果 *XACTUAL* = 200，*X_RANGE* = 300，*XTARGET* = -200，定位斜坡将根据表 27 (第**错误！未定义书签。**页)中的朝着溢出位置运行(299→ -300)(见图 A)。

 阅读整个文档：是第 22 章(第 174 页)中的补充指令。

7. 斜坡时序和同步

TMC4361A 提供各种选项来初始化新斜坡。默认情况下，每个外部寄存器更改都会通过 SPI 输入立即分配给内部寄存器。通过适当的初始配置，斜坡序列可以在无需任何干预的情况下工作。

同步 有三个级别复杂度的斜坡。预定义的斜坡与后续解释的 SPI 数据传输无关，详情见章节 (第 70 页)。

可以配置两个可选功能，既可以单独使用，也可以组合使用，如下所示:

阴影寄存器集 一套完整的阴影运动寄存器组可以加载到实际的运动寄存器，更改运动轮廓开始下一个斜坡。

目标寄存器流水线 可以预定义不同的目标位置，然后依次激活。该流水线可以配置为循环；和/或不同的参数排序。

无主机情况下的同步 此外，在没有主机的情况下，可以使能启动忙状态，用于同步几个运动控制器。

专用斜坡时序引脚		
引脚名称	**类型**	**备注**
START	输入和输出	外部启动输入引脚产生内部启动状态或 内部启动事件产生外部启动输出

表 29: 专用斜坡时序引脚

专用斜坡时序寄存器			
寄存器名称	**寄存器地址**		**备注**
START_CONF	0x02	RW	同步单元的配置寄存器。
START_OUT_ADD	0x11	RW	外部启动信号的有效输出长度。
START_DELAY	0x13	RW	启动触发器和启动信号之间的延迟时间。
X_PIPE0… 7	0x38…0x3F	RW	目标位置流水线和/或参数流水线。
SH_REG0…12	0x40…0x4C	RW	阴影寄存器组

表 30: 专用斜坡时序寄存器

7.1. 基本同步设置

通常，斜坡可以由内部或外部触发开始。请注意，启动触发不是启动信号本身，而是启动向有效启动状态过渡的斜率。经过定义的延迟事件后，产生内部启动信号。

7.1.1. 启动信号触发选择

斜坡启动配置，请考虑以下步骤:

操作步骤:

- 选择内部或外部启动触发源。

i 所有触发可以单独使用或组合使用。

- 根据下表设置触发信号。

启动触发配置表	
***trigger_events* = *START_CONF*(8:5)**	**结果**
b'0000	不会产生或进一步处理启动信号。
b'xxx0	内部触发产生，需设置 *trigger_events*(0) = 0。内部产生的启动信号映射到**输出**的 **START** 引脚。
b'xxx1	外部 Start 引脚启动触发，需设置 *trigger_events*(0) = 1 。**START 引脚为输入脚**。对于 Start 启动输入，请考虑滤波设置。见章节 第 18 页。
b'xx1x	TARGET_REACHED 事件为斜坡的启动触发信号。
b'x1xx	VELOCITY_REACHED 事件为斜坡的启动触发信号。
b'1xxx	POSCOMP_REACHED 事件为斜坡的启动触发信号。.

表 31: 启动触发配置

7.1.2. 用户定义的时序配置

默认情况下，TMC4361 会及时处理每个串行接口数据报并分配给对应的寄存器。可以设置以下的开关选择，将 SPI 请求与 SPI 传输断开。只有在内部产生启动信号后，才处理寄存器 *XTARGET*、*VMAX*、*RAMP_MODE* 和 *GEAR_RATIO* 的更新。

执行启动信号对内部参数分配的影响，请执行以下操作:

操作步骤:

- 如下表所示，选择以下选项。

启动使能切换配置表 *(所有的切换能单独或者组合使用)*	
start_en* = START_CONF*(4:0)	**结果**
b'xxxx1	*XTARGET* 仅在内部产生启动信号后被更新。
b'xxx1x	*VMAX* 仅在内部产生启动信号后被更新。
b'xx1xx	*RAMPMODE* 仅在内部产生启动信号后被更新。
b'x1xxx	*GEAR_RATIO* 仅在内部产生启动信号后被更新。
b'1xxxx	在内部产生启动信号后，阴影寄存器被指定为有效斜坡参数。第 9.2 节将对其进行更详细的解释。(第 75 页)。

表 32: 启动开关配置

 阅读整个文档：是第 22 章(第 174 页)中的补充指令。

7.1.3. 触发和内部产生的启动信号之间的延迟定义

默认情况下，触发后面紧跟内部启动信号。

实现延迟产生内部启动信号，请执行以下操作:

- 根据要求设置 *START_DELAY* 寄存器 0x13。

操作步骤:

结果:
当识别到启动触发时，内部启动信号在 *START_DELAY* 时钟周期后产生。

优先考虑外部输入 默认情况下，内部启动信号产生也会延迟外部触发。

为了立即提示外部启动并触发产生内部启动信号(无论定义的延迟如何)，请执行以下操作:

操作步骤:
- 设置 *immediate_start_in* = 1 (*START_CONF* 寄存器 0x02)。

结果:
当识别到外部启动触发器时，即使内部启动触发器已经启动了延时定时过程，也会立即生成内部启动信号。

START 引脚极性 START 引脚既可以用作输入引脚，也可以用作输出引脚。可配置 *START_CONF* 寄存器 0x02 选择 START 引脚的有效电平极性。

默认情况下，电压电平从高到低的跳变会触发启动信号(START 是输入信号)，或者 START 输出从高到低的转变来指示有效的 START 事件。

改变有效启动极性，请执行以下操作:

操作步骤:
- 设置 *pol_start_signal* = 1 (*START_CONF* 寄存器 0x02)。

结果:
START 引脚为高电平有效。电平从低到高的转换触发启动信号(START 是一个输入)，或者 START 输出从低到高的转换来指示有效的 START 事件。

7.1.4. 使能 START 引脚输出配置

默认情况下，START 引脚的有效输出电平持续一个时钟周期。

延长此时间长度，请执行以下操作:

条件:
- START 引脚被分配为输出: *trigger_events*(0) = 1。

操作步骤:
- 根据需求设置 *START_OUT_ADD* 寄存器 0x11 。

结果:
有效电平持续(*START_OUT_ADD* + 1)时钟周期。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

7.1.5. 斜坡时序例程

以下三个示例描述了 SPI 数据报、内部和外部信号电平、相应的速度斜坡和其它。SPI 数据在每个数据报的末尾传输。

斜坡时序例程 1

在本例中，速度值变化立即执行。

过程描述

- 新的 *XTARGET* 值是在 TARGET_REACHED 有效且 *START_DELAY* 已过去后更新的。
- 因为没有分配新的 XTARGET 值，所以新的斜坡不会从第二个斜坡的末尾开始。
- START 是输出信号。
- 内部启动信号的步长是(*START_OUT_ADD* + 1)时钟周期。

外部设备可以这样同步:

参数设置时序示例 1	
参数	**设置**
RAMPMODE	b'101
start_en	b'00001
trigger_events	b'0010
START_DELAY	>0
START_OUT_ADD	>0
pol_start_signal	1

表 33: 参数设置示例 1

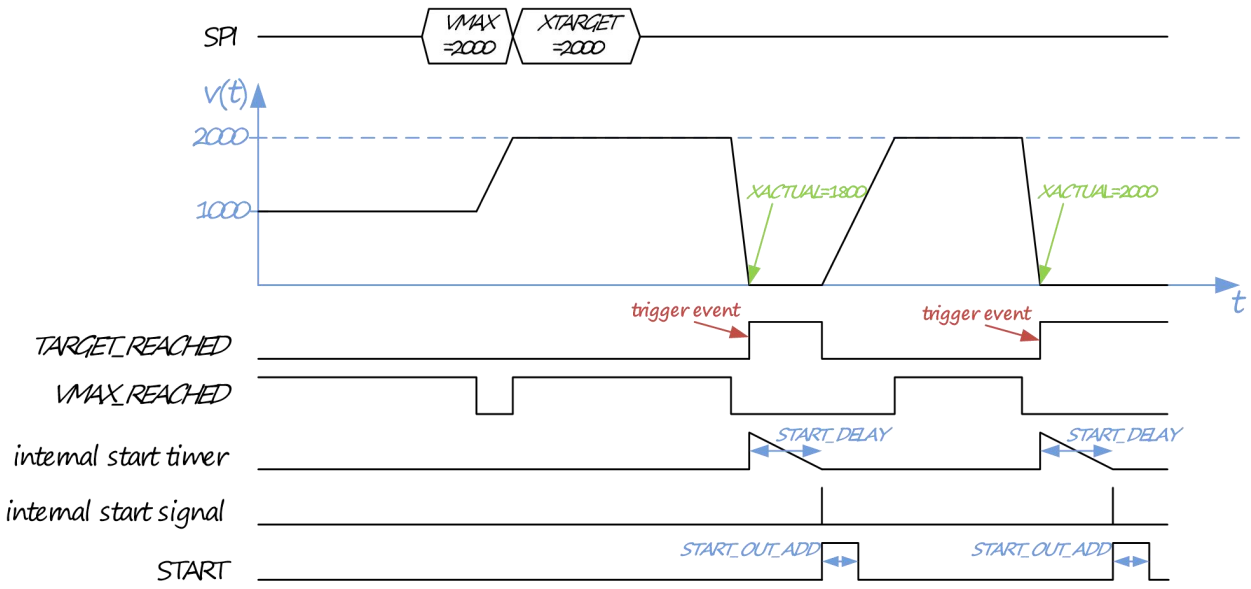


图 34: 斜率时序例程 1

斜坡时序 2

在该示例中，速度值和斜坡模式值变化在第一个启动信号之后执行。

过程描述

- 新的斜坡模式变成 S 形斜坡的定位模式。
- 然后，由于斜坡模式的改变，斜坡停止在目标位置 *XTARGET*。
- *XTARGET* 的进一步改变再次启动斜坡。
- 当第一个 TARGET _ REACHED 事件触发的启动延迟完成后，斜坡立即启动。
- 有效 START 启动输出信号仅持续一个时钟周期。

参数设置时序示例 2	
参数	**设置**
RAMPMODE	b'001 → b'110
start_en	b'00111
trigger_events	b'0110
START_DELAY	>0
START_OUT_ADD	0
pol_start_signal	0

表 34: 参数设置时序示例 2

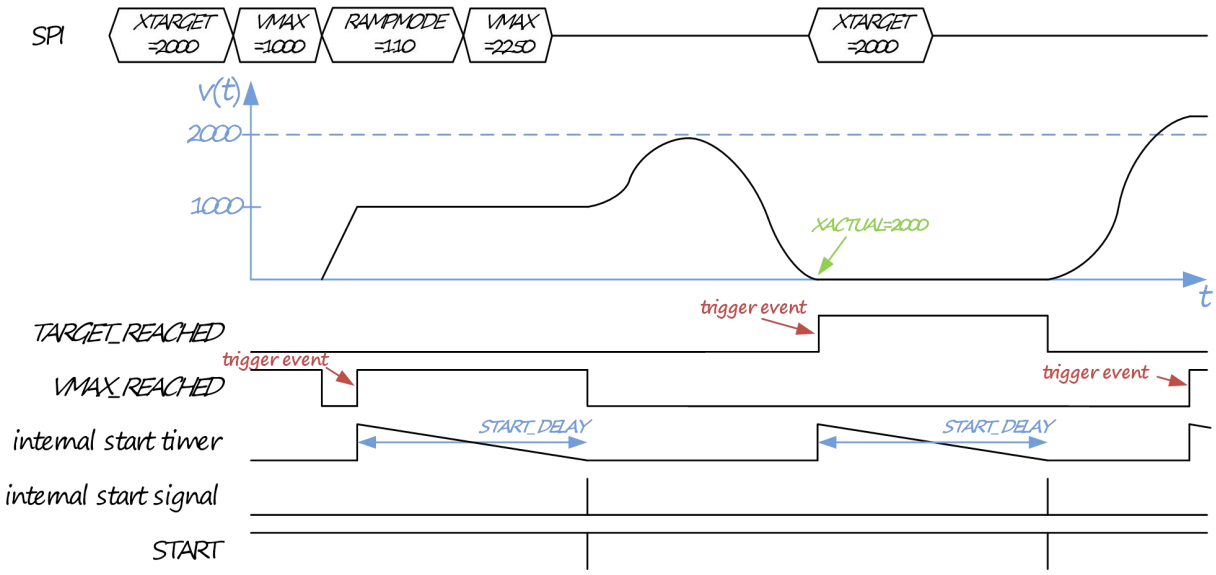


图 35: 斜坡时序示例 2

斜坡时序示例 3

在本例中， *START_DELAY>0* 并将 *immediate_start_in* 同时设置为 1，外部启动信号触发器被优先排序。

过程描述

- 当 *XACTUAL* 等于 *POSCOMP* 时使能启动定时器，忽略其间的外部启动信号。
- 第二个启动事件由外部启动信号触发。忽略 *POSCOMP_REACHED* 事件。
- 由于设置:*immediate_start_in* = 1，第三个启动定时器进程被外部 START 信号中断，立刻有效。

参数设置时序示例 3	
参数	**设置**
RAMPMODE	b'000
start_en	b'00010
trigger_events	b'1001
immediate_start_in	0 → 1
START_DELAY	>0
pol_start_signal	1

表 35: 斜坡时序示例 3

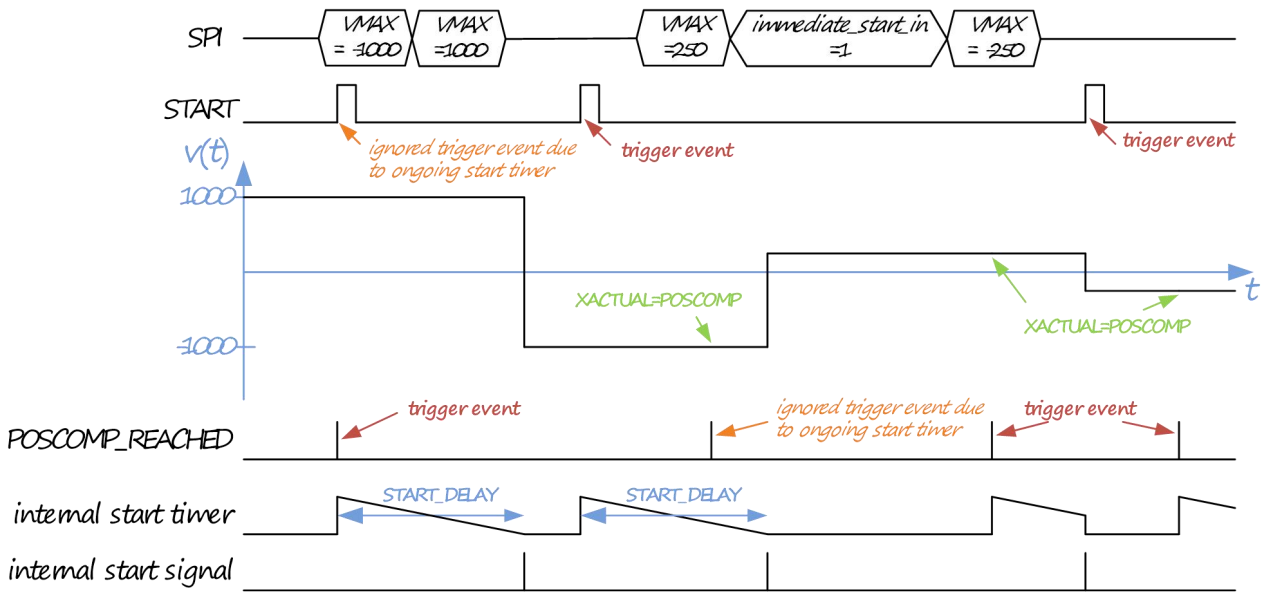


图 36: 斜坡时序示例 3

7.2. 阴影寄存器设置

某些应用需要针对特定斜坡情况/时间点设置一套完整的新斜坡参数。TMC4361A 最多提供 14 个阴影寄存器，在内部启动信号产生后，这些阴影寄存器被装载到相应的斜坡参数寄存器中。

使能阴影寄存器

使能阴影寄存器，请执行以下操作:

- 设置 *start_en*(4) = 1 并选择一个或多个 *trigger_events* (*START_CONF* 寄存器 0x02), 见章节 9.1.2 (第 70 页)。

操作步骤

结果:
当每个连续的内部启动信号产生时，阴影寄存器被加载到相应的活动斜坡寄存器中。

使能循环阴影寄存器

也可以将当前运动轮廓写回到阴影运动寄存器中，以连续交换斜坡运动轮廓。

使能循环阴影寄存器，请执行以下操作:

操作步骤

- 设置 *start_en*(4) = 1 并选择一个或多个 *trigger_events* (*START_CONF* 寄存器 0x02) , 见章节 9.1.2 (第 70 页)。
- 设置 *cyclic_shadow_regs* = 1 (*START_CONF* 寄存器 0x02)。

结果:
当每个连续的内部启动信号产生时，阴影寄存器被加载到相应的斜坡寄存器中，而运动斜坡寄存器加载到阴影寄存器中。

∞∀接下一页。

7.2.1. 阴影寄存器配置选项

根据您选择的斜坡类型，有四种不同的阴影寄存器分配选择用于匹配阴影寄存器集。下几页介绍对应的选项.

i 请注意，阴影选项 3 和 4 的配置之间的唯一区别是 *VSTART* 由 *VSTOP* 替换。

选项 1: 阴影默认配置

如果需要将整个斜坡寄存器设置在单级堆栈中，请执行以下操作:

操作步骤:

- 设置 *shadow_option* = b'00 (*START_CONF* 寄存器 0x02)。
- 设置 *start_en*(4) = 1 并选择一个或者多个 *trigger_events* (*START_CONF* 寄存器 0x02)。

操作步骤:

- **缺省配置:** 设置 *cyclic_shadow_regs* = 0 (*START_CONF* 寄存器 0x02)
- **可选配置:** 设置 *cyclic_shadow_regs* = 1 (*START_CONF* 寄存器 0x02)

结果:

每个相关的运动参数在下一个内部启动信号时被相应的阴影寄存器参数改变。如果使能循环阴影寄存器，阴影寄存器集被当前运动轮廓寄存器集改变。

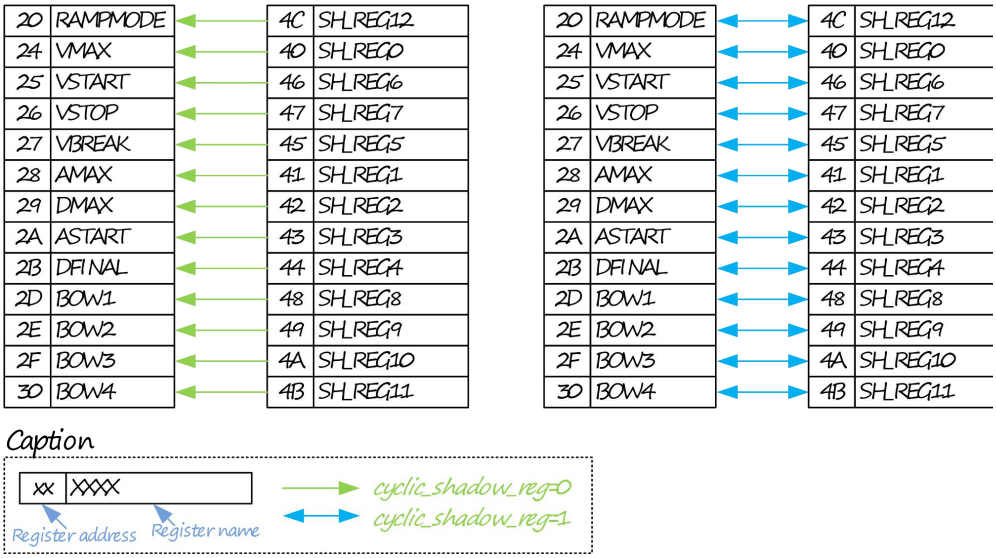


图 37: 替换完整斜坡运动轮廓的单级阴影寄存器选项

i 绿色箭头显示默认设置

i 蓝色箭头显示可选设置。

特别注意事项

如果选择了 S 形斜坡类型且操作模式从速度切换到定位模式(由阴影寄存器传输触发)时，SH_REG10 不能等于 BOW3，以确保安全操作模式切换。

ꕥ在接下来的几页中，我们将解释更多选项。请翻页。

选项 2: S 形斜坡的两级阴影寄存器组

如果配置了 S 形斜坡，可以使用两级阴影寄存器组。阴影寄存器激活时，S 形斜坡的七个相关运动参数会受到影响。

实现 S 形斜坡的两级阴影寄存器流水线，请执行以下操作：
操作步骤:

- 设置 *shadow_option* = b'01 (*START_CONF* 寄存器 0x02)。
- 设置 *start_en*(4) = 1 并设置一个或者多个 *trigger_events* (*START_CONF* 寄存器 0x02)。

操作步骤:

- **缺省配置:** 设置 *cyclic_shadow_regs* = 0 (*START_CONF* 寄存器 0x02)。
- **可选配置:** 设置 *cyclic_shadow_regs* =1 (*START_CONF* 寄存器 0x02)。

结果:
七个运动参数(*VMAX*、*AMAX*、*DMAX*、*BOW1...4*)在下一个内部启动信号时被相应的阴影寄存器参数(SH_REG0...6) 改变。同时，这些阴影寄存器与第二阴影级的参数交换。
如果使能循环阴影寄存器，第二个阴影寄存器组(SH_REG7...13)被当前运动轮廓组改变，例如 0x 28(AMEX)被写回到 0x48 (SH_REG8)。
其他斜坡寄存器保持不变。

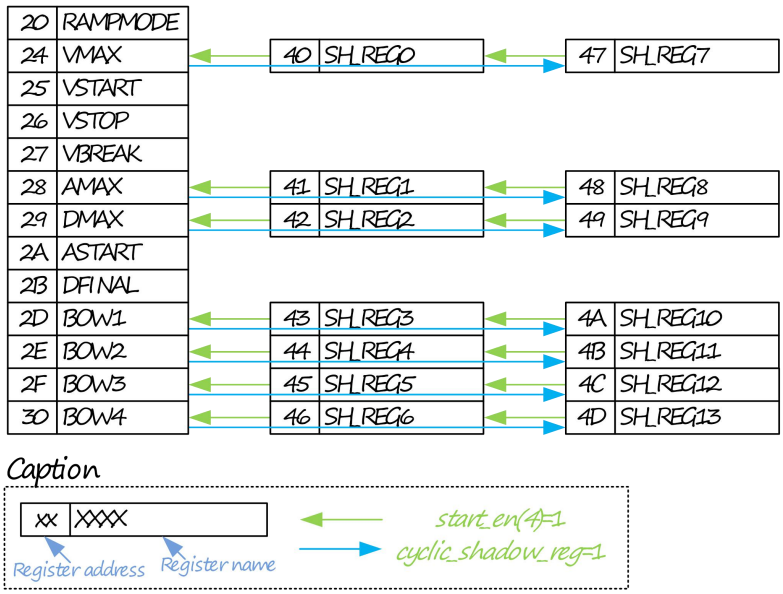


图 38: 两级阴影寄存器选项 1, 适用于 S 型斜坡.

i 绿色箭头显示默认设置

i 蓝色箭头显示可选设置。

∞∀接下一页。

选项 3: 梯形斜坡双级阴影寄存器组)

梯形斜坡可以配置两级阴影寄存器组。阴影寄存器激活时，梯形斜坡的七个相关运动参数会受到影响。

实现梯形斜坡的两级阴影寄存器流水线，请执行以下操作:

操作步骤:

- 设置 *shadow_option* = b'10 (*START_CONF* 寄存器 0x02)。
- 设置 *start_en*(4) = 1 及设置一个或者多个 *trigger_events* (*START_CONF* 寄存器 0x02)。

操作步骤:

- **缺省配置:** 设置 *cyclic_shadow_regs* = 0 (*START_CONF* 寄存器 0x02)。
- **可选配置:** 设置 *cyclic_shadow_regs* = 1 (*START_CONF* 寄存器 0x02)。

结果:

七个运动参数(*VMAX*、*AMAX*、*DMAX*、*ASTART*、*DFINAL*、*VBREAK* 和 *VSTART*)在下一个内部起始信号时刻被相应的阴影寄存器参数(*SH_REG0...6*) 改变。同时，这些阴影寄存器与第二阴影级的参数交换。

如果使用循环阴影寄存器，第二个阴影寄存器组(SH_REG7...13)会被当前运动轮廓组改变，例如 0x 27(*VBREAK*)被写回 0x4C (*SH_REG12*)。其他斜坡寄存器保持不变。

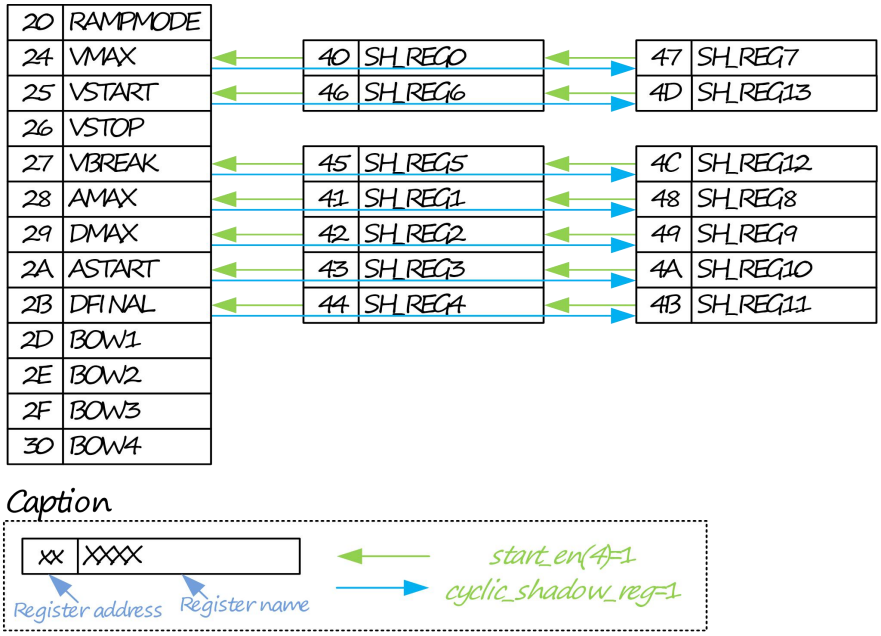


图 39: 两级阴影寄存器选项 2，适用于梯形斜坡。

i 绿色箭头显示默认设置

i 蓝色箭头显示可选设置。

∀下一页继续

选项 4: 梯形斜坡双级阴影寄存器组

梯形斜坡可以配置两级阴影寄存器组。阴影寄存器激活时，梯形斜坡的七个相关运动参数会受到影响。

配置梯形斜坡的双级阴影寄存器流水线，请执行以下操作:

操作步骤:

- 设置 *shadow_option* = b'10 (*START_CONF* 寄存器 0x02)。
- 设置 *start_en*(4) = 1 或者设置一个或者多个 *trigger_events* (*START_CONF* 寄存器 0x02)。

操作步骤:

- **缺省配置:** 设置 *cyclic_shadow_regs* = 0 (*START_CONF* 寄存器 0x02)
- **可选配置:** 设置 *cyclic_shadow_regs* = 1 (*START_CONF* 寄存器 0x02)

结果:

七个运动参数(*VMAX*、*AMAX*、*DMAX*、*ASTART*、*DFINAL*、*VBREAK* 和 *VSTOP*)在下一个内部起始信号处被相应的阴影寄存器参数(SH_REG0...6)改变。同时，这些阴影寄存器与第二级阴影的参数交换。

如果使用循环阴影寄存器，第二个阴影寄存器组(SH_REG7...13)会被当前运动轮廓组改变，例如 0x26 (VSTOP)被写回 0XD(SH _ REG 13)。其他斜坡寄存器保持不变。

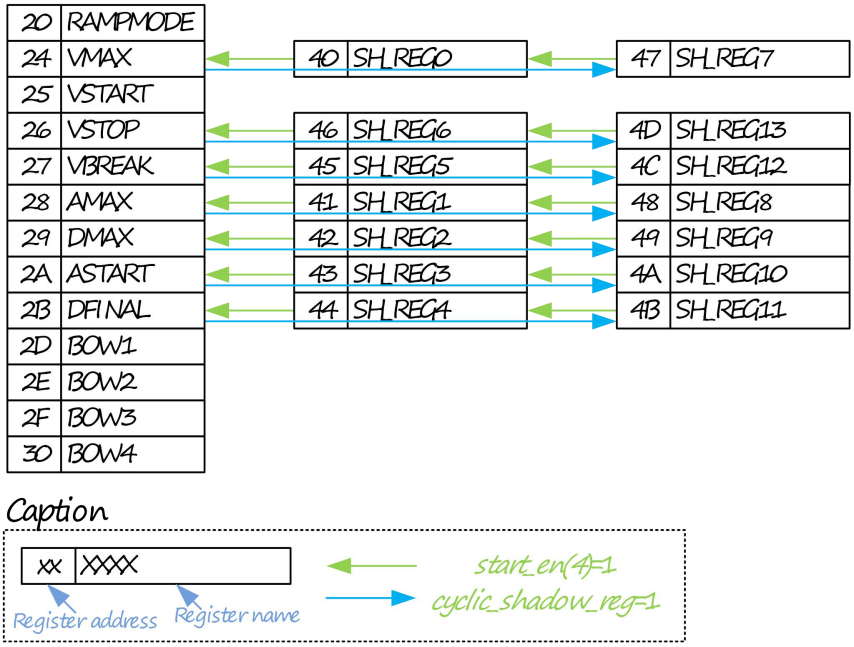


图 40: 两级阴影寄存器选项 3，适用于梯形斜坡

i 绿色箭头显示默认设置

i 蓝色箭头显示可选设置。

翻页查看与本节相关的特别注意。

特别注意

没有被四个阴影选项选择的斜坡参数值一直保持最初配置，直到被 SPI 写请求更改。

此外，如果没有选择循环阴影寄存器，阴影寄存器流水线的最后一级将保留这些值，直到它们被串行接口写请求覆盖。

7.2.2. 阴影寄存器传送延时

最多可以跳过 15 个内部启动信号，再执行阴影寄存器传输。

设置要跳过阴影传输的内部启动信号的个数，请执行以下操作:

操作步骤:

- 根据你的要求设置 *shadow_option*。
- 设置 *start_en*(4) = 1 并选择一个或者多个 *trigger_events* (*START_CONF* 寄存器 0x02)。
- *可选配置:* 设置 *cyclic_shadow_regs* = 1。
- 根据要跳过阴影传输的内部启动信号的个数来设置 *SHADOW_MISS_CNT* ≠ 0 (*START_CONF* 寄存器 0x02)。

结果:

阴影寄存器传输不是在每个内部启动信号下执行的。相反，指定数量的启动信号被忽略，直到第(*SHADOW_MISS_CNT*+1)个启动信号才执行阴影传输。

下图为 *SHADOW_MISS_CNT* 的示例，其中 SHADOW 寄存器传输由内部信号 sh_reg_transfer 来说明。寄存器 *START_CONF* CONF (23:20) 读出信号丢失计数器 *CURRENT_MISS_CNT*:

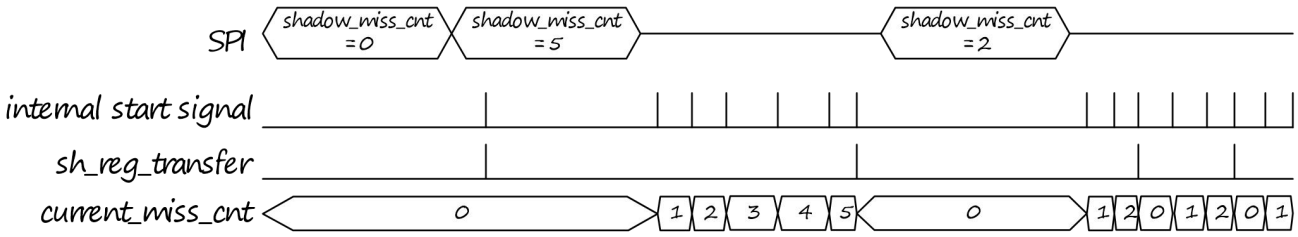


图 41: 几个内部 start 启动信号的 SHADOW_MISS_CNT 参数

特别注意

将请求的阴影 BOW 值传递到内部结构并完成内部计算最多需要(320/f_{CLK})[秒]。因此在提示任何阴影寄存器传输之前，必须等待弓形值阴影参数的所有内部计算完成。

为了更好地理解这一点，下面提供了一个 S 形坡道的两级阴影流水线的示例:

前提条件:

S 形斜坡(*shadow_option*= b'01)的阴影寄存器传输有效(*start_en*(1) = 1，并选择一个或多个 *trigger_events*)

操作步骤

- 设置 *SH_REG0*, *SH_REG1*, *SH_REG2* (*VMAX*, *AMAX*, *DMAX* *的阴影寄存器*)。
- 设置 *SH_REG3*, *SH_REG4*, *SH_REG5*, *SH_REG6* (*BOW1…4* *的阴影寄存器*)。
- 确保接下来的 320 / f_{CLK} [s]时间内没有阴影寄存器传输。

结果:

阴影寄存器传输可以在这个时间跨度之后启动.

7.3. 内部参数流水线

TMC4361A 包含了一个目标流水线，用于对子目标进行排序，以便轻松安排复杂的目标结构。

7.3.1. 目标流水线的配置和激活

为 *X_PIPE0…7* 寄存器分配不同的目标值。如果使能目标流水线，一旦产生内部开始信号，即刻使能新的分配周期；如上所述，同时移动这些值。

处理描述:

- *X_PIPE0 赋给了新的 XTARGET 值。*
- 每个 *X_PIPEn* 寄存器接收其后继寄存器的值:
  X_PIPEn = X_PIPEn+1

激活目标流水线，请执行以下操作:

操作步骤:

➢ 设置 *pipeline_en* = b'0001 (*START_CONF* 寄存器 0x02)。

结果:

每个新的内部启动信号产生时，执行上述的过程。

循环目标流水线的配置

也可以将 *XTARGET* 的值重新分配给一个(或多个)流水线寄存器 *X_PIPE0…7*。由此，创建目标流水线循环。

使能目标流水线循环，请执行以下操作:

操作步骤:

➢ 设置 *pipeline_en* = b'0001 (*START_CONF* 寄存器 0x02)。

➢ 将 *XPIPE_REWRITE_REG* 设置为 *XTARGET* 回写的流水线寄存器 (例如 *XPIPE_REWRITE_REG* = b'00010000)。

结果:

每个新的内部启动信号产生时，执行上述的过程，*XTARGET* 被写回到所选的 X_PIPEx 寄存器(例如 *XPIPE _ REWRITE _ REG* = 0x 10， *XTARGET* 被写回到 *X_PIPE4*)。

上一页描述的过程和操作如下图所示，内部启动信号出现时使能对应的分配。

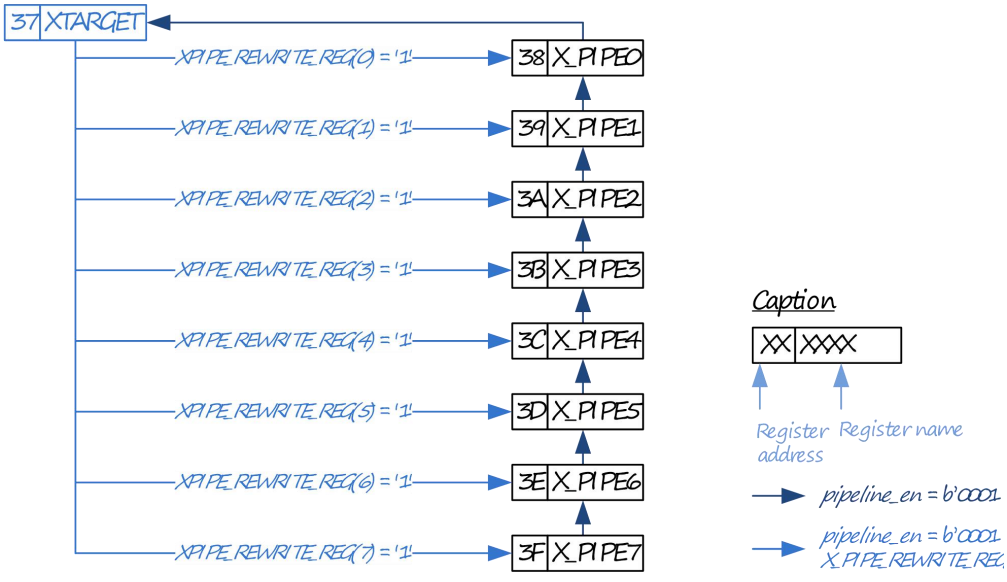


图 42: 具有配置选项的目标流水线

 阅读整个文档：是第 22 章(第 174 页)中的补充指令。

7.3.2. 流水线用于其它内部寄存器

TMC4361A 流水线(寄存器 0x38…0x3F)可以分段配置，最多分成四段。这些段可用于以下内部参数:

- *XTARGET* 寄存器 0x37
- *POS_COMP* 寄存器 0x32
- *GEAR_RATIO* 寄存器 0x12
- *GENERAL_CONF* 0x00

因此，这些参数值变化对于连续斜坡运动和/或对于减少几个运动控制器的同步开销可能是重要的。

POS_COMP 值可为运动过程中产生启动信号。因此，通过流水线传输该参数可能会很有用，避免依赖于 SPI 传输速度。
例如，如果两个 *POS_COMP* 值之间的距离非常近且速度足够快会造成 SPI 传输完成前错过第二个值。为避免错过，建议在启动信号后立即更改 POS_COMP。

GEAR_RATIO 参数也是如此，它定义了输入步进脉冲的步进响应。一些应用需要非常快速地改变从控制器的齿轮系数。当直接提示启动信号时，立即更改参数比通过 SPI 传输来更改可能非常有用。

同样，在定义的时间点更改一般配置参数可能(但不一定)是至关重要的。一些合适的应用需要明确定义从直接外部控制(*sd_in_mode* = b'01)到内部斜坡(*sd_in_mode* = b'00)的转换，或者反之从内部斜坡模式到直接外部控制，因为在这种情况下主/从关系被互换的。

以下为流水线选项:

流水线使能选项	
***pipeline_en*(3:0)**	**流水线**
b'xxx1	*XTARGET* 流水线使能
b'xx1x	*POS_COMP* 流水线使能
b'x1xx	*GEAR_RATIO* 流水线使能
b'1xxx	*GENERAL_CONF* 流水线使能

表 36: 流水线使能选项

7.3.3. 流水线映射概述

pipeline_en 参数为分离流水线提供了 16 种不同组合配置。因此，流水线数量从 0 到 4 不等，流水线深度也受影响。可能的选择如下:八级、四级、三级和两级。

下面的“流水线映射”表介绍了流水线设置，如何分配流水线和对应的深度。描述了流水线寄存器的最终目的寄存器，说明了最终目标寄存器(*XTARGET*、*POS_COMP*、*GEAR_RATIO*、*GENERAL_CONF*)是从哪个流水线寄存器(*X_PIPE0…7*)传送来的。

例如，如果选择 *POS_COMP* 和 *GEAR_RATIO* 作为流水线输送的参数，将创建两个 4 级流水线。当产生内部启动信号时，*POS_COMP* 取值为 *X_PIPE0*，*GEAR_RATIO* 寄存器取值为 *X_PIPE4*。

但是如果选择 *POS_COMP, GEAR_RATIO* 和 *XTARGET* 作为参数，则会创建两个三级流水线和一个双级流水线。当产生内部启动信号时，XTARGET 取值为 *X_PIPE0*，*POS_COMP* 取值为 X_PIPE3，*GEAR_RATIO* 取值为 *X_PIPE6*。

流水线映射表

更多示例将在下表中详细描述，并解释了一些可能的配置和参考示例。

流水线映射						
示例.	***pipeline_en* (3:0)**	**分配**	*为以下寄存器最后传送的流水线*			
			GENERAL_CONF →pipeline_en(3)	*GEAR_RATIO* →pipeline_en(2)	*POS_COMP* →pipeline_en(1)	*XTARGET* →pipeline_en(0)
-	*b'0000*	*五流水线*	-	-	-	-
-	*b'0001*	*一个 8 级流水线*	-	-	-	X_PIPE0
A	*b'0010*		-	-	X_PIPE0	-
B	*b'0100*		-	X_PIPE0	-	-
-	*b'1000*		X_PIPE0	-	-	-
C	*b'0011*	*两个 4 级流水线*	-	-	X_PIPE4	X_PIPE0
-	*b'0101*		-	X_PIPE4	-	X_PIPE0
-	*b'1001*		X_PIPE4	-	-	X_PIPE0
-	*b'0110*		-	X_PIPE4	X_PIPE0	-
-	*b'1010*		X_PIPE4	-	X_PIPE0	-
D	*b'1100*		X_PIPE4	X_PIPE0	-	-
F	*b'0111*	*两个 3 级流水线和一个两级流水线*	-	X_PIPE6	X_PIPE3	X_PIPE0
-	*b'1011*		X_PIPE6	-	X_PIPE3	X_PIPE0
E	*b'1101*		X_PIPE6	X_PIPE3	-	X_PIPE0
-	*b'1110*		X_PIPE6	X_PIPE3	X_PIPE0	-
G/H	*b'1111*	*四个 2 级流水线*	X_PIPE6	X_PIPE4	X_PIPE2	X_PIPE0

表 37:不同流水线配置的流水线映射

7.3.4. 循环流水线操作

以上所有配置示例都可将所选寄存器(*XTARGET*、*POS_COMP*、*GEAR_RATIO* 和/或 *GENERAL_CONF*)的当前值写回到其分配的流水线中的任何流水线寄存器，形成循环流水线。设置 *XPIPE_REWRITE_REG* 为写回的流水线寄存器完成流水线映射。

7.3.5. 流水线示例

下面列出了几个流水线映射的配置示例。

示例 A+B: 用一个流水线

示例 A: *POS_COMP* 八级循环流水线。

示例 B: *GEAR_RATIO* 六级循环流水线。

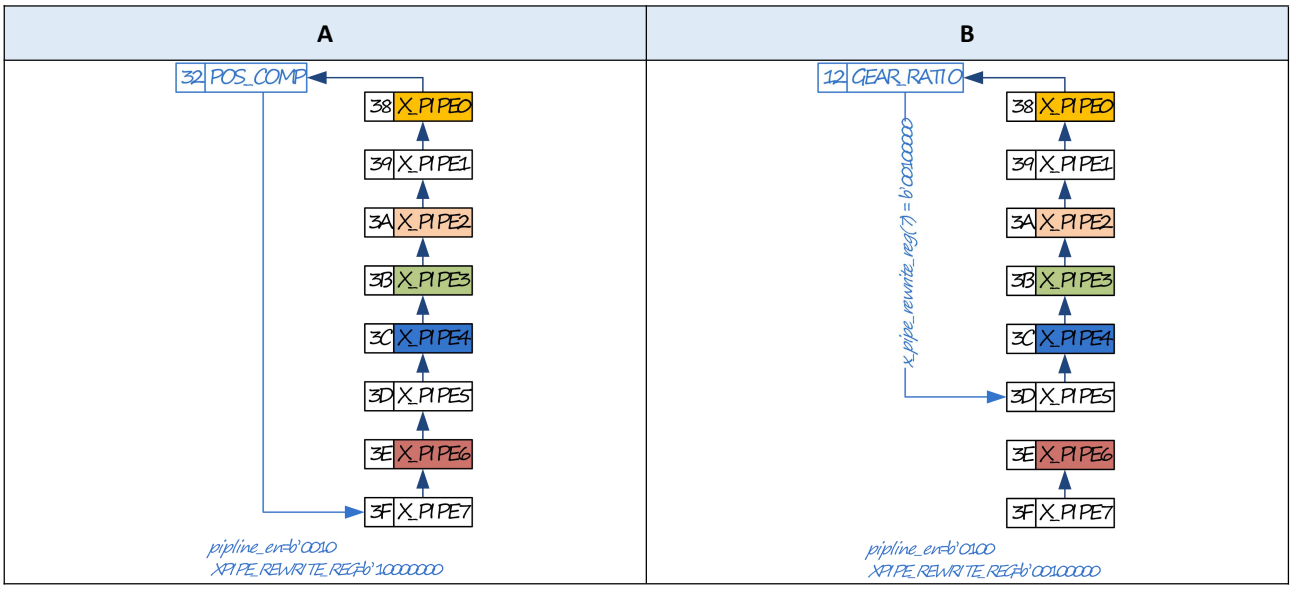


图 43: 流水线示例 A

图 44: 流水线示例 B

示例 C+D: 用两个流水线

示例 C: *XTARGET* 和 *POS_COMP* 的循环流水线，每个流水线有四级。

示例 D: *GEAR_RATIO* 三级循环流水线，*GENERAL_CONF* 两级循环流水线。

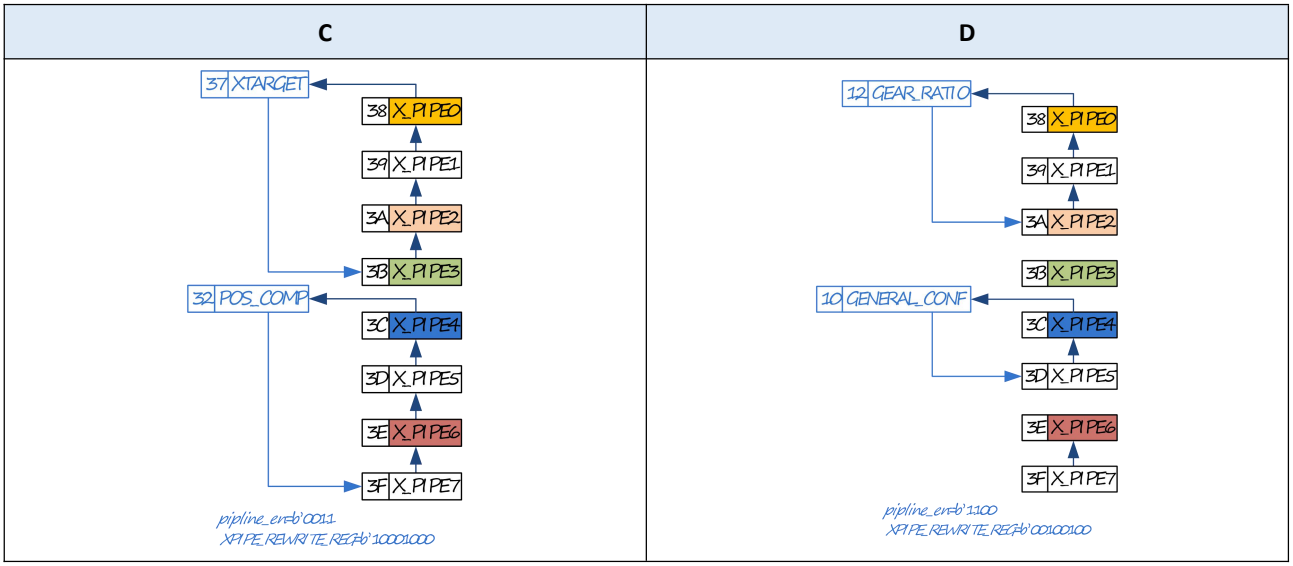


图 45: 流水线示例 C

图 46: 流水线示例 D

示例 E+F: 三个流水线

示例 E: *XTARGET* 和 *GEAR_RATIO* 的三级循环流水线，*GENERAL_CONF* 的两级循环流水线。

示例 F: *XTARGET* 和 *GEAR_RATIO* 的两个两级循环流水线 *GEAR_RATIO* 的非循环流水线有三级。

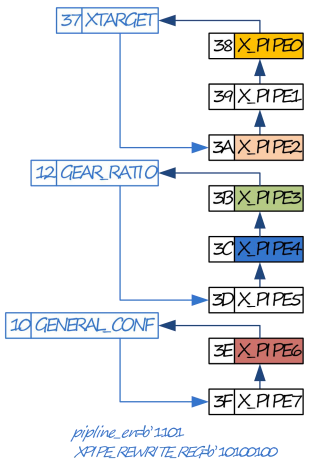
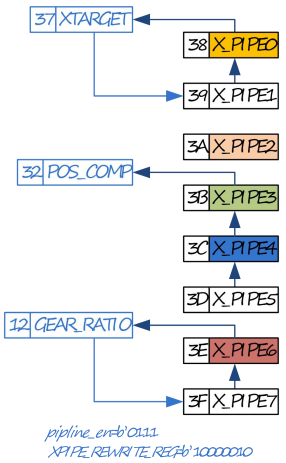
E	F
	

图 47: 流水线示例 E

图 48: 流水线示例 F

示例 G+H:三个流水线

示例 G: *XTARGET*、*POS_COMP*、*GEAR_RATIO* 和 *GENERAL_CONF* 的循环流水线，每个流水线有两级。

示例 H: *XTARGET*、*POS_COMP*、*GEAR_RATIO* 和 *GENERAL_CONF* 的四个非循环流水线，每个流水线有两级。

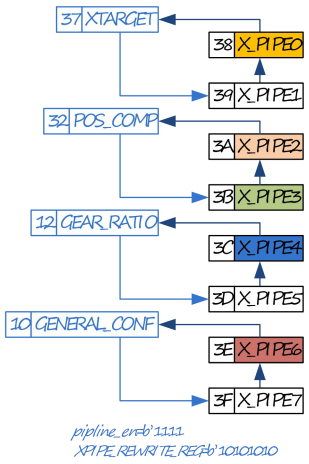
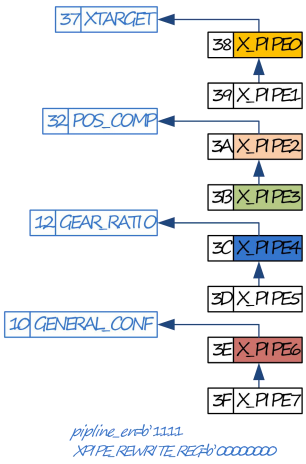
G	H
	

图 49: 流水线示例 G

图 50: 流水线示例 H

7.4. 在没有主机情况下通过 START 引脚实现几个运动控制器的控制同步

START 引脚可以配置成三态输入，以便在没有主机的情况下实现几个控制器的同步。

使能三态 START 引脚

此时 START 脚配置为三态。忙状态使能。在忙状态下，START 为非有效极性的强输出。如果在内部启动定时器溢出后产生内部启动信号，则 START 引脚内部设置为输入。此外，START 脚此时为有效极性的弱输出。

如果 START 输入引脚的信号被设置为有效，因为所有信号线都准备好了，则 START 输出在 *START_OUT_ADD* 时钟周期内保持有效(强驱动)。

然后，再次激活忙状态，直到下一个开始信号出现。

使能三态 START 引脚，请执行以下操作:

操作步骤:

➢ 设置 *busy_en* = 1 (*START_CONF* 寄存器 0x02)。

结果:

执行上述过程。

START 引脚连接

如果 START 引脚与其他 TMC4361A 器件的 START 引脚相连，建议在器件之间加一个串联电阻(例如 220 欧姆)，以避免在配置阶段，由于不同器件的 START 引脚在电压不同时产生短路电流。

注意:

→ *避免短路持续太久。*

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

8. 串行数据输出

TMC4361A 包含一个 SPI 接口，在电机运动之前给电机驱动器(除 Step/Dir 输出之外)初始化，并配置运动过程中的参数。可通过 SPI 电机驱动控制 TMC 步进驱动器。

SPI 接口配置

SPI 接口的主要任务:

- TMC4361A 集成了覆盖寄存器，以便轻松设置 TMC 电机驱动芯片和第三方芯片相关参数。
- 集成微步正弦波查找表(*MSLUT*)可产生两组电流值，分别为正弦和余弦值。
- 可以一次传输两个电流值到 TMC 电机驱动器芯片，同时给电机线圈通电。每次串行接口数据报通过传输一系列电流值来控制电机运动。速度斜坡相关电流调节值可调整 MSLUT 输出，这些电流调节值将电流的最大幅度值与特定速度斜坡的要求相结合。

SPI 电机驱动器的引脚名称		
引脚名称	**类型**	**备注**
NSCSDRV_SDO	输出	发送到电机驱动器的片选信号输出，低有效。
SCKDRV_NSDO	输出	发送到电机驱动器的串行时钟输出。
SDODRV_SCLK	输入输出口用于输出	发送到电机驱动器的串行数据输出。
SDIDRV_NSCLK	输入	来自电机驱动器的串行数据输入。
STDBY_CLK	输出	时钟输出、待机输出或 ChopSync 时钟输出。

表 38: SPI 电机驱动的引脚名称

SPI 输出寄存器的寄存器名称			
寄存器名称	**寄存器地址**		**备注**
GENERAL_CONF	0x00	RW	相关位: 位 14:13, 位 19, 位 20, 位 28。
REFERENCE_CONF	0x01	RW	相关位: 位 26, 位 27, 位 30。
SPIOUT_CONF	0x04	RW	配置 SPI 输出通信的寄存器。
STEP_CONF	0x0A	RW	微步数、每圈的全步数和电机状态位事件选择。
DAC_ADDR	0x1D	RW	在 DAC 数值之前的串行接口地址/命令: 线圈 A: DAC_ADDR(15:0), 线圈 B: DAC_ADDR(31:16)
SPI_SWITCH_VEL			使能自动发送覆盖数据报的速度阈值。
CHOPSYNC_DIV	0x1F	RW	斩波时钟分频器(位 11:0)。
FS_VEL	0x60	W	使能全步驱动的速度阈值。
COVER_LOW	0x6C	W	覆盖寄存器的低 32 位(微控制器到电机驱动)。
COVER_HIGH	0x6D	W	覆盖寄存器的高 32 位(微控制器到电机驱动)。
COVER_DRV_LOW	0x6E	R	覆盖响应寄存器的低 32 位 (电机驱动到微控制器)。
COVER_DRV_HIGH	0x6F	R	覆盖响应寄存器的高 32 位 (电机驱动到微控制器)。
CURRENT_CONF	0x05	RW	电流调节配置。
SCALE_VALUES	0x06	RW	电流调节值
STDBY_DELAY	0x15	RW	斜坡停止到待机标志有效的延迟时间。
FREEWHEEL_DELAY	0x16	RW	从待机标志有效到飞轮状态的延迟时间。
VDRV_SCALE_LIMIT	0x17	RW	更改电流驱动调节值的速度设置。

阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

SPI 输出寄存器的寄存器名称			
寄存器名称	**寄存器地址**		**备注**
UP_SCALE_DELAY	0x18	RW	增加到更高的电流调节值的增加延时；24 位。
HOLD_SCALE_DELAY	0x19	RW	减小到静态保持电流调节值的递减延时；24 位。
DRV_SCALE_DELAY	0x1A	RW	减小到驱动电流调节值的递减延时。
BOOST_TIME	0x1B	RW	斜坡开始后上升调节值有效的时间。
SCALE_PARAM	0x7C	R	实际电流调节参数；8 位。
CURRENTA *CURRENTB*	0x7A	R	MSLUT 的实际电流值: SIN (线圈 A)和 SIN90_120 (线圈 B)；每个 9 位。
CURRENTA_SPI *CURRENTB_SPI*	0x7B	R	MSLUT 的实际电流调节值: SIN (线圈 A) 和 SIN90_120 (线圈 B)；每个 9 位。
MSLUT registers	0x70…78	W	MSLUT 值定义。
MSCNT	0x79	R	MSLUT 的实际微步位置。
START_SIN *START_SIN90_120* *DAC_OFFSET*	0x7E	RW	MSLUT 正弦起始值 (位 7:0)。 MSLUT 余弦起始值 (位 23:16)。 DAC 输出值的偏移值 (位 31:24)。

表 39: 专用 SPI 输出寄存器

8.1. 使用 TMC 电机驱动器入门

本章介绍如何轻松连接 TMC 电机驱动器。

设置 *SPIOUT_CONF* 正确设置 *SPIOUT_CONF* 寄存器 0x04 对连接的 TMC 电机步进驱动器很重要。如果正确选择了 TMC 电机驱动器，TMC4361A 发送电流及自动配置电机驱动器。TMC 电机驱动器将相应的状态返回到运动控制器的状态寄存器。

TMC4361A 内含一个可编程的电流查找表。默认情况下，预先编程的查找表为正弦波，适用于大多数步进电机。

i TMC2130、TMC2160 和 TMC26x 的初始设置请见第 22 章第 225 页中。

8.2. SPI 输出接口配置参数

TMC4361A 内含一个 SPI 输出接口。下节将详细解释接口参数的配置。

8.2.1. 如何使能 SPI 输出通信

使能 SPI 输出通信，请执行以下操作:

操作步骤:

- 设置 *serial_enc_out_enable* = 0 (*GENERAL_CONF* 寄存器 0x00 的第 24 位)。

结果:

SPI 输出使能。

i SPI 按缺省的配置输出。

串行接口输出通信的专用引脚 下表列出了专用于 SPI 输出通信的引脚:

SPI 输出通讯引脚	
引脚	**描述**
NSCSDRV_SDO	低有效片选信号。
SCKDRV_NSDO	SPI 输出时钟。
SDODRV_SCLK	MOSI –输出数据给电机驱动器。
SDIDRV_NSCLK	MISO –接收电机驱动器响应的输入引脚。在向电机驱动器传输数据的过程中，对响应进行采样。

表 42: SPI 输出通讯引脚

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

8.2.2. SPI 输出时序配置

TMC4361A 是与电机驱动器(从机)SPI 通信的主设备，因此必须为 SPI 输出设置时序配置。TMC4361A 的 SCKDRV_NSDO 输出引脚对应 SPI 时钟输出。

按如下方式设置 *SPIOUT_CONF* 寄存器 0x04 配置 SPI 时钟:

操作步骤:

- 设置 *SPI_OUT_LOW_TIME* = *SPIOUT_CONF* (23:20) 为串行时钟低电平的时间长度。
- 设置 SPI_OUT_HIGH_TIME = SPIOUT_CONF(27:24) 为串行时钟高电平的时间长度。
- 此外，设置 *SPI_OUT_BLOCK_TIME* = *SPIOUT_CONF*(31:28)为最后一个 SPI 输出数据报之后不发送新数据报的最短时间。

结果:

设置完 SPI 输出通信方案。在 SPI 数据报之间的非有效阶段-时间长度至少为 *SPI_OUT_BLOCK_TIME* 时钟周期- SCKDRV_NSDO 和 NSCSDRV_SDO 输出引脚需保持高电平。SPI 输出通信的时序如下图所示。

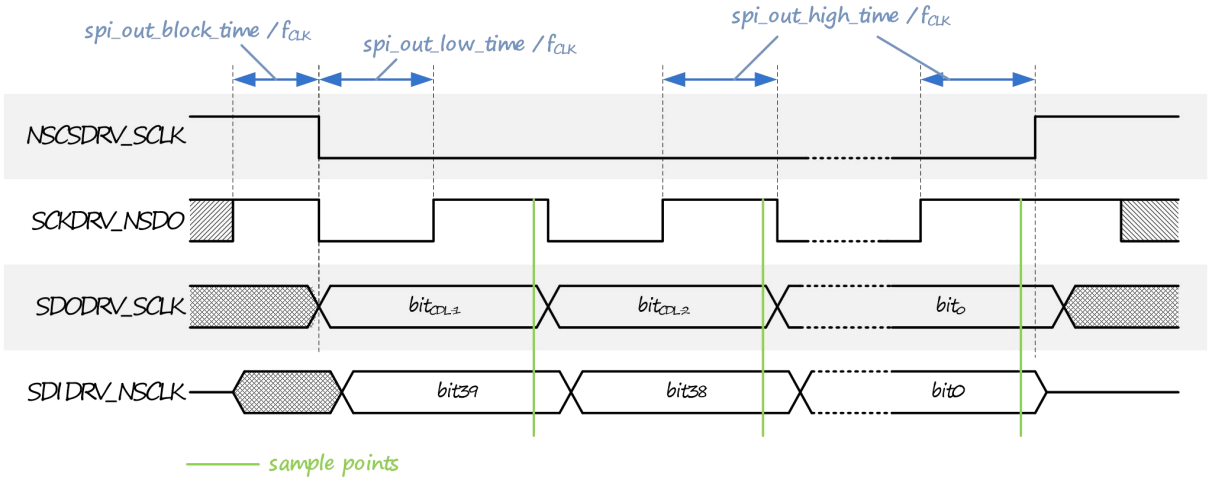


图 53: SPI 输出数据报时序

最小和最大时间周期

三个参数的最小时间周期均为 $2/f_{CLK}$。如果 SPI 输出参数设置为 0，则为 2 个时钟周期。三个参数的最大时间周期为 $15/f_{CLK}$。

因此，SPI 时钟频率 f_{SPI_CLK} 涵盖以下范围:

$$f_{CLK} / 30 \le f_{SPI\_CLK} \le f_{CLK} / 2.$$

阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

8.2.3. 电流数据报

一般 SPI 自动发送电流数据报给相连电机驱动器。TMC4361A 内部微步查找表(MSLUT)对应电机驱动器的实际电流数据。

过程描述

- 斜坡发生器每产生一步，*MSCNT* 值都会根据斜坡方向而增加或减少。
- *MSCNT* 寄存器 0x79(可读值)包含正弦表的当前微步位置。
- 因此，电流值 *CURRENTA* (0x7A)和 *CURRENTB* (0x7B)根据位置改变。
- 如果 TMC4361A 使能自动电流传输，那么一旦电流值更新就会产生新的数据报。
- 因此电机驱动总是收到最新的数据。TMC4361A 可以自动设置电流数据报的长度并将新值转换成所选的数据报格式，通常分为 TMC 电机驱动器的振幅和极性。

8.2.4. 改变细分

微步分辨率从 256 (*MSTEP_PER_FS* = b'0000)更改到更低的值，则 *MSLUT* 内部步长大于 1。例如，如果微步分辨率设置为 64 (*MSTEP_PER_FS* = b'0010)，则每一个内部步长对应的 *MSCNT* 要么增加，要么减少 4。因此，每步跳过 MSLUT 内部的三个电流值，以匹配新的微步长分辨率。

8.2.5. 微控制器和驱动器之间的覆盖数据报通信

除了自动传输电流数据报之外，微控制器还可以通过 TMC4361A 的覆盖数据报直接与电机驱动器通信。该功能对配置很有用，因此微控制器和电机驱动器之间不需要额外的 SPI 通信通道。

一个覆盖数据报最多包含 64 位。该 64 位 SPI 覆盖寄存器分为两个 32 位寄存器——*COVER_HIGH* 寄存器 0x6D 和 *COVER_LOW* 寄存器 0x6C。仅当必须发送一次超过 32 位时，才需要 *COVER_HIGH* 寄存器。

如何定义覆盖数据报长度

覆盖数据报发送的位数由覆盖数据报长度 *COVER_DATA_LENGTH* 定义。

定义覆盖数据报长度，请执行以下操作:

操作步骤:

➢ 设置 *COVER_DATA_LENGTH* = *SPIOUT_CONF* (19:13)配置覆盖数据报位数。

结果:

覆盖数据报长度设置为 *COVER_DATA_LENGTH*。如果该参数设置为高于 64，则覆盖寄

i 对于 TMC 电机驱动器，可以将覆盖数据长度设置为 0。在这种情况下，覆盖数据长度会根据所选电机驱动器自动选择。更多细节将在后续页面中描述。

存器数据长度仍为最大 64 位。

 阅读整个文档；是第 22 章(第 174 页)中的补充指令。

8.2.6. 发送覆盖数据报

整个覆盖数据报寄存器的 LSB(最后一个有效位)位于 *COVER_LOW*(0)。只要 *COVER_DATA_LENGTH* < 33，就仅需 *COVER_LOW* 或该寄存器的一部分。

如果需要 32 位以上，则需要完整的 *COVER_LOW* 和 *COVER_HIGH* 寄存器(的一部分)才能完成 SPI 覆盖数据传输。

注意:

→ *每个串行接口通信都从最高有效位开始。*

选项 1: 覆盖数据报长度 < 33 位

发送小于 33 位的覆盖数据报，请执行以下操作:

操作步骤:

- 设置 *COVER_LOW* (*COVER_DATA_LENGTH*-1:0) 寄存器 0x6C = cover_data。

结果:

COVER_LOW 寄存器的有效请求后，SPI 输出 *COVER_DATA_LENGTH* 位 *COVER_LOW* 寄存器的。

32 位覆盖数据报

选项 2：覆盖数据报长度> 32 位

发送大于 32 位的覆盖数据报，请执行以下操作:

操作步骤:

- 将覆盖数据分成两段:
- cover_data_low = cover_data(31:0)。
- cover_data_high = cover_data >> 32。
- cover_data_high = cover_data(31:0)。
- 设置 *COVER_HIGH*(*COVER_DATA_LENGTH*−32:0) 寄存器 0x6D=cover_data_high。
- 设置 *COVER_LOW* 寄存器 0x6C = cover_data_low。

结果:

COVER_LOW 寄存器的有效请求后，SPI 输出 *COVER_DATA_LENGTH* 位数据，包含 *COVER_HIGH* 和 *COVER_LOW* 寄存器值。

覆盖寄存器和数据报结构如下图所示:

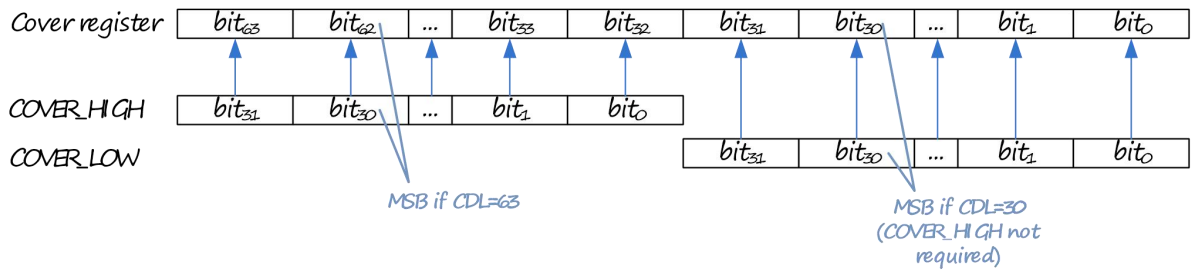


图 54: 覆盖数据寄存器组成 (CDL – COVER_DATA_LENGTH)

∞∀接下一页。

接收覆盖数据报的响应

发送覆盖数据报通常同时返回电机驱动器的数据或状态，返回响应存储在覆盖响寄存器中，微控制器可随时查询；覆盖响应寄存器由 *COVER_DRV_HIGH* 寄存器 0x6F 和 *COVER_DRV_LOW* 寄存器 0x6E 构成，最高可包含 64 位。

与 *COVER_LOW* 和 *COVER_DRV_HIGH* 类似，电机驱动器的响应分为 *COVER_DRV_LOW* 和 *COVER_DRV_HIGH*。响应覆盖寄存器的组成和最高位的位置与覆盖寄存器遵循相同的结构。

COVER_DONE 事件

成功传输完数据时，事件 *COVER_DONE* 有效。这表明覆盖寄存器数据发送到电机驱动器，同时 *COVER_DRV_HIGH* 寄存器 0x6F 和 *COVER_DRV_LOW* 寄存器 0x6E 接收返回响应。

8.2.7. 自动生成覆盖数据报的配置

自动发送斜坡速度相关的覆盖数据报对于某些应用非常有用，例如，在运动期间更改斩波器设置。

注意:

→ *仅当覆盖数据报长度不超过 32 位时，此功能才可用*

使能自动传输斜坡速度相关的覆盖数据，请执行以下操作:

操作步骤:

- 定义使能自动覆盖数据报传输的触发速度。
- 设置 *SPI_SWITCH_VEL* 寄存器 0x1D 为这个绝对速度 [pps]。
- 设置 *COVER_LOW* 寄存器 0x6C 为 cover_data，较低的速度下该值有效。
- 设置 *COVER_HIGH* 寄存器 0x6D 为 cover_data，较高的速度下该值有效。
- 设置 *automatic_cover* = 1 (*REFERENCE_CONF* 寄存器 0x01)。

结果:

每当绝对内部斜坡速度| *VACTUAL* |超过 *SPI_SWITCH_VEL* 值时，发送相应的覆盖数据到电机驱动器，

当|*VACTUAL*| < *SPI_SWITCH_VEL*，发送 *COVER_LOW*。

当|*VACTUAL*| ≥ *SPI_SWITCH_VEL*，发送 *COVER_HIGH*。

8.3. 概述:TMC 电机驱动器连接

如上所述，TMC4361A 能够自动设置 TMC 电机驱动器的覆盖寄存器长度。此外，选择了 TMC 电机驱动器，还可以实现 SPI 自动通信功能。

8.3.1. TMC 电机驱动器连接设置

TMC 电机可用的 SPI 和步进/方向接口通信方案

以下产品系列支持 SPI 和步进/方向通讯接口，下面是详细的解释:

- TMC236, TMC239
- TMC246, TMC248, TMC249
- TMC260, TMC261, TMC262, TMC2660
- TMC389
- TMC2130, TMC2160

如何根据 TMC 步进电机驱动器使能 SPI 输出设置

根据 TMC 步进电机驱动器使能 SPI 输出设置，请按以下步骤操作:

操作步骤:

- 如前所述，根据 TMC 电机驱动器规格设置 *SPI_OUT_LOW_TIME, SPI_OUT_HIGH_TIME 和 SPI_OUT_BLOCK_TIME* 。
- 设置 *COVER_DATA_LENGTH* = 0 (*SPIOUT_CONF* 寄存器 0x04 的第 19:13 位)。
- 根据连接的 SPI 电机驱动器，如下表所示，设置 *spi_output_format* = *SPI_OUT_CONF* (3:0) .

结果:

现在完成 TMC 电机驱动器的通信配置。

TMC 步进电机驱动器选项				
TMC 电机驱动	***spi_output_format*** *=SPI_OUT_CONF* (3:0)	**覆盖寄存器数据长度** *COVER_DATA_LENTGH=0*	**自动电流数据报传输**	**覆盖寄存器数据报传输**
SPI 输出关断	b'0000	0	--	--
TMC23x	b'1000	12	⚑	⚑
TMC24x	b'1001	12	⚑	⚑
TMC26x/389	b'1010 b'1011	20 20	⚑ S/D 输出	⚑
TMC2130 / TMC2160	b'1101 b'1100	40 40	⚑ S/D 输出	⚑

表 43: TMC 步进电机驱动器选项

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

8.3.2. 电机驱动器数据报的响应和状态位

当 TMC 电机驱动器接收到通过 TMC4361A 的 SPI 输出传输的电流数据报或覆盖数据报时，状态数据立即被发送回 TMC4361A 控制器。响应存储在 *COVER_DRV_LOW* 0x6E 和 *COVER_DRV_HIGH* 0x6F 寄存器中，就像所有其他覆盖请求一样。

返回的状态位的类型和顺序取决于所选电机驱动器。接下来的章节介绍每个电机驱动器的详细列表及通信细节。

TMC 步进电机驱动器的可用状态位与 TMC4361A 状态寄存器的映射是相似的。最后八位-状态(31:24)-等于传输的电机状态位。第 19.15.章给出了寄存器详细概述 (第 148 页)。

8.3.3. 电机驱动器状态位事件和中断

TMC4361A 的 ***EVENTS*** (30)是一个与电机驱动器状态位相关的事件。这里，电机驱动器相关状态位变化会影响该事件。

根据电机驱动器状态位使能电机事件的 *EVENTS* (30), 按照以下步骤操作:

操作步骤:

- 选择一个或多个电机驱动器状态作为电机事件设置 *MSTATUS_SELECTION* = *STEP_CONF* (23:16)寄存器 0x0A。

结果:

如果所选电机状态位之一有效(线或)，电机事件 EVENTS(30)产生一个事件。

配置相应的 INTR 输出可产生对此电机事件的中断，如章节 5.3 第 26 页所述。

 阅读整个文档：是第 22 章(第 174 页)中的补充指令。

8.3.4. 堵转检测和堵转停止

stallGuard 和 stallGuard2 功能

TMC 步进电机驱动芯片的 stallGuard 和 stallGuard2 可在不需要位置传感器的情况下根据电机的反电动势检测堵转和过载情况。SPI 返回堵转的检测状态。

有关更多信息，请参考 www.trinamic.com 在线提供的应用笔记“Parameterization of stallGuard2 & coolStep”。

电机堵转状态的表示

除了 TMC23x 和 TMC24x 有三个负载检测位之外，其它 TMC 驱动芯片的堵转状态由一个状态位表示。一旦发现堵转，TMC4361A 能够停止内部斜坡。因为低速运动期间可能会激活不希望的堵转标志，所以建议为堵转停止动作设置速度阈值。

激活内部速度斜坡堵转停止

使能内部速度斜坡的堵转停止，请执行以下操作:

- 根据能正确识别堵转的最小速度的绝对值设置 *VSTALL_LIMIT* 寄存器 0x67 [pps]。

操作步骤:

- 设置 *stop_on_stall* = 1 (*REFERENCE_CONF* 寄存器 0x01 的 26 位)。
- 设置 *drive_after_stall* = 0 (*REFERENCE_CONF* 寄存器 0x01 的 27 位)。

结果:

每当检测到堵转且下列情况成立时，内部斜坡速度立即设置为 0:
|*VACTUAL*| > *VSTALL_LIMIT*。
因此，*STOP_ON_STALL* 事件产生。

i 步进电机驱动器直接映射 stallGuard 状态位到寄存器 *STATUS* (24)中。一旦电机驱动器产生堵转状态位，该标志总是有效。

i 一旦检测到堵转且满足 |*VACTUAL*| > *VSTALL_LIMIT*，则 *STATUS* (11) 的 *ACTIVE_STALL* 转态位有效。

堵转停止后使能内部速度斜坡

实现 Stop-on-Stall 堵转停止后使能内部速度斜坡，请执行以下操作:

操作步骤:

- 读 *EVENTS* 寄存器 0x0E 清除 *STOP_ON_STALL*。
- 设置 *drive_after_stall* = 1 (*REFERENCE_CONF* 寄存器 0x01 的 27 位)。

结果:

i 再次使能 Stop-on-Stall 堵转停止，需手动复位 *drive_after_stall* 为 *0*。

Stop-on-Stall 堵转停止事件不再阻止内部斜坡运动。

8.4. TMC26x 步进电机驱动器

支持 TMC26x 步进电机驱动器

TMC4361A 提供以下特性，以便更好地支持 TMC26x 电机步进驱动器系列:

- 直接设置电流值的 SPI 模式。
- TMC26x 处理 TMC4361A S/D 输出的 S/D 模式。
- 两种模式下都支持自动切换微步和全步。
- 两个模式下都支持堵转检测和堵转停止。
- 仅 S/D（步进/脉冲）模式: TMC4361A 向 TMC26x 传输自动电流调节值。
- 仅 S/D（步进/脉冲）模式: TMC4361A 发送自动生成的查询数据报，接收 TMC26x 的状态数据和微步位置。

下节将更详细地解释这些功能.

i 有关更多信息，请参考连接的步进电机驱动器手册。.

8.4.1. TMC26x 设置 (SPI 模式)

使能已连接 TMC26x 步进电机驱动器的 SPI 数据传输模式和功能集，请执行以下操作:

操作步骤:

- 设置 *spi_output_format* = b'1010 (*SPI_OUT_CONF* 寄存器 0x04)。
- 设置 *COVER_DATA_LENGTH* = 0 (*SPI_OUT_CONF* 寄存器 0x04)。

结果:

连接的步进电机驱动器设置为 SPI 模式 TMC26x。TMC4361A 的 SPI 输出相应的覆盖数据报和电流数据报。

8.4.2. TMC26x 设置 (S/D 模式)

使能 TMC26x 步进电机驱动器的 S/D 模式和功能集，请执行以下操作:

操作步骤:

- 连接 TMC26x 步进电机驱动器的 SPI 输出引脚和 S/D 输出。
- 设置 *spi_output_format* = b'1011 (*SPI_OUT_CONF* 寄存器 0x04)。
- 设置 *COVER_DATA_LENGTH* = 0 (*SPI_OUT_CONF* 寄存器 0x04)。
- 根据硬件设置 *DIR_SETUP_TIME* 和 *STP_LENGTH_ADD* (寄存器 0x10)。
- 设置适合的 *POLL_BLOCK_EXP* (*SPIOUT_CONF* 寄存器 0x04 的位 11:8)。

结果:

连接的步进电机驱动器设置为 S/D 模式 TMC26x。SPI 输出引脚仅传输覆盖数据报和自动配置数据报，因为运动是通过处理 TMC4361A 的 STPOUT/DIREOUT 输出信号产生的。下一个查询数据报在上一个查询数据报后的 2^POLL_BLOCK_EXP* SPI_BLOCK_TIME 时钟周期后发送。

i 更高的微步频率需要更短的 SPI 数据报查询时间。

∞∀接下一页。

8.4.3. 向 TMC26x 发送覆盖数据报

根据上述的 TMC26x 设置，TMC4361A 现在自动发送 20 位数据报。

实现发送覆盖数据报到 TMC26x 电机步进驱动器，请执行以下操作:

操作步骤:

➢ 设置需配置的寄存器值到寄存器 *COVER_LOW* (19:0)。

结果:

发送覆盖数据报到连接的驱动器，数据传输后 *COVER_DONE* 有效。TMC26x 的响应存储在 *COVER_DRV_LOW* (19:0)中。

如果 TMC26x 驱动器工作在 SPI 模式，当完成传输电流数据报时， *COVER_DONE* 也会有效。

实现只对覆盖数据报使能 *COVER_DONE*，请执行以下操作:

操作步骤:

➢ 设置 *cover_done_only_for_covers* = 1 (SPI_OUT_CONF 寄存器 0x04 的位 12)。

结果:

COVER_DONE 事件仅在发送覆盖数据报时有效，电流数据报不生成对应该标志。

8.4.4. 自动连续的 TMC26x 覆盖数据流

如果 TMC26x 的 VS 引脚上出现的电压降，通常内部寄存器全部复位，因此微控制器需要连续重写 TMC26x 的寄存器值。

TMC4361A 支持连续重写 TMC26x 的五个配置寄存器，从而减轻微控制器的工作负荷。

使能自动连续发送 TMC26x 覆盖数据报，请执行以下操作:

操作步骤:

➢ *设置 autorepeat_cover_en* = 1 (*SPI_OUT_CONF* 寄存器 0x04 的位 7)。

结果:

如果 *autorepeat_cover_en*=1，则 TMC4361A 会每 2^{20} 个时钟周期发送一个覆盖数据报到 TMC26x。每次寻址另一个寄存器后，这五个覆盖数据报都会一个接一个地被连续重传；即循环。

i 然而，传输速率保持在每 2^{20} 个时钟周期一个数据报.

注意:

→ *当 TMC26x 以 SPI 模式运行时，如果值不变，在一个传输间隔周期内电流数据报也会重复出现。*

→ *如果 TMC26x 寄存器被覆盖数据报手动重写，根据定义，最新定义的寄存器值将被重复。*

→ *TMC4361A 执行的自动寄存器更改-例如电流自动调节值传输-按照重复的覆盖数据报一样考虑。*

在自动连续传输覆盖数据报期间手动发送覆盖数据报

注意

如果在自动覆盖流期间使能手动覆盖数据报传输，则存在无法将手动修改的覆盖数据报传输到 TMC26x 的风险！

- **您必须在 2^{20} 个时钟周期内发送两次相同的覆盖数据报。**

这将确保成功的手动发送覆盖数据报。

8.4.5. TMC26x SPI 模式：自动切换到全步

因为 SPI 模式下电机电流是通过 SPI 传输更新的，所以微步和全步之间的自动切换完全取决于内部斜坡速度。

使能自动切换微步和全步，请执行以下操作:

操作步骤:

- 根据切换速度[pps]的绝对值设置 *FS_VEL* 寄存器 0x60。
- 设置 *fs_en* = 1 (*GENERAL_CONF* 寄存器 0x00 的位 19)。

结果:

现在，如果|*VACTUAL*|≥*FS_VEL*，切换到全步；
如果|*VACTUAL*|<*FS_VEL*，从全步切换到细分。

只要使能并全步模式有效，状态位 *FS_ACTIVE* 则相应有效。

8.4.6. TMC26x S/D 模式:自动切换到全步

在 S/D 模式下，微步和全步之间的自动切换不仅取决于内部斜坡速度，还取决于 TMC26x 查找表 MSLUT 的微步位置；切换到较低的分辨率必须小心处理以捕捉正确的微步位置。必须正确设置 TMC26x 步进驱动器的读取选择位才能完成 TMC4361A 自动切换。

实现 TMC26x S/D 模式下微步和全步操作之间的自动切换，请执行以下操作:

前提条件:

TMC26x 配置必须通过覆盖数据报:

- 设置 *RDSEL1* = 0 和 *RDSEL0* = 0 @TMC26x。

操作步骤:

- 设置 *disable_polling* = 0 (*SPI_OUT_CONF* 寄存器 0x04 的位 6)。
- 根据切换速度[pps]的绝对值设置 *FS_VEL* register 0x60。
- 设置 *fs_en* = 1 (*GENERAL_CONF* 寄存器 0x00 的位 19)。
- 设置 *fs_sdout* = 0 (*GENERAL_CONF* 寄存器 0x00 的位 20)。

结果:

如果| *VACTUAL* | ≥ FS_VEL，则 TMC26x 的步进分辨率设置为全步。
如果| *VACTUAL* | < FS_VEL，从全步切换到微步。
只要使能并且全步有效，则 *FS_ACTIVE* 有效。
每当相应的寄存器被新分配的微步分辨率覆盖时，都应预先设置 TMC26x DRVCTRL 寄存器(执行覆盖数据报之前)。

ꕤ翻页了解在 S/D 模式下更改 TMC26x 电流调节参数。

 阅读整个文档：是第 22 章(第 174 页)中的补充指令。

8.4.7. TMC 26x S/D 模式：改变电流调节参数

SPI 模式的 TMC26x 驱动器通过电流数据报传输自动调节电流。S/D 模式的 TMC26x 步进电机驱动器的电流通过 TM4361A 发送生成的覆盖数据报直接更改 TMC26x SGCSCONF 寄存器的 CS 值实现。

TMC4361A 可以自动调节电流值，这将在第 11 章 120 页中解释。

使能 TMC26x 的 S/D 模式的自动电流调节，请执行以下操作：

操作步骤：

- 设置 *scale_val_transfer_en* = 1 (*SPI_OUT_CONF* 寄存器 0x04 的位 5)。
- 根据你的要求设置电流调节幅值寄存器 0x06 和电流调节配置寄存器 0x05 (见 11 章 120 页)。

结果：

如果内部调整了电流幅值，TMC4361A 自动向 TMC26x 发送覆盖数据报直接更改 CS 位。每当相应的寄存器被新分配的电流调节值覆盖时，都应预先设置 TMC26x SGCSCONF 寄存器 (执行覆盖数据报之前)。

注意：

→ *请考虑 CS 值仅由 5 位组成。因此，寄存器 0x06 中的电流调节值也必须设置为 5 位。*

8.4.8. TMC26x 状态位

TMC26x 步进驱动器的以下状态位对应 TMC4361A 状态寄存器 0x0F，每次 SPI 响应返回这些状态位：

TMC26x 的状态寄存器映射		
状态位@TMC4361A	**状态标志 @TMC26x**	**描述**
STATUS(24)	SG	stallGuard2™ 状态标志
STATUS(25)	OT	过温标志
STATUS(26)	OTPW	温度预警标志
STATUS(27)	S2GA	线圈 A 高端 MOSFET 对地短路检测标志
STATUS(28)	S2GB	线圈 B 高端 MOSFET 对地短路检测标志
STATUS(29)	OLA	桥臂 A 的开路标志
STATUS(30)	OLB	桥臂 B 的开路标志
STATUS(31)	STST	静止标志

表 45: TMC26x 状态标志的映射

ℹ 如果未禁用查询，TMC26x 的状态数据在 S/D（步进/脉冲）模式下也可用.

8.4.9. TMC26x 状态响应

TMC4361A 传输任何数据报总是返回 TMC26x 的 DRV_STATUS 寄存器。

要存储 TMC26x 的 DRV_STATUS 响应，请执行以下操作：

操作步骤：

- 设置 *disbale_polling* = 0 (*SPI_OUT_CONF* 寄存器 0x04 的第五位)。

结果：

TMC4361A 将响应的值存储在可读的 *POLLING_STATUS* 寄存器 0x6C 中。

8.5. TMC2130 / TMC2160 步进电机驱动器

TMC4361A 支持 TMC2130 和 TMC2160 电机步进驱动器，以下称之为 TMC21x0。
由于 TMC5130 和 TMC5160 的串行接口数据传输和寄存器映射一致。也可以采用以下配置过程。

支持 TMC2130 及 TMC2160

TMC4361A 提供以下功能，以支持 TMC21x0 电机步进驱动器:

- 直接设置电流值的 SPI 模式。
- TMC21x0 处理 TMC4361A S/D 输出的 S/D 模式。
- 两种模式下都支持自动切换微步和全步。
- 两种模式下都支持堵转检测和堵转停止。
- 仅 S/D 模式: TMC4361A 向 TMC21x0 传输电流自动调节值。
- 仅 S/D 模式: TMC4361A 发送自动生成的查询数据报，接收 TMC21x0 的状态数据和微步位置。

下节将更详细地解释这些功能。

i 有关更多信息，请参考 TMC21x0 步进驱动电机手册。.

8.5.1. 设置 TMC21x0 (SPI 模式)

使能 SPI 数据传输模式和功能集(针对已连接的 TMC21x0 步进电机驱动器)，请执行以下操作:

操作步骤:

- ➢ 设置 *spi_output_format* = b'1101 (*SPI_OUT_CONF* 寄存器 0x04)。
- ➢ 设置 *COVER_DATA_LENGTH* = 0 (*SPI_OUT_CONF* 寄存器 0x04)。

结果:
配置 SPI 模式下的 TMC21x0 作为连接的步进电机驱动器。覆盖数据报和电流数据报通过 SPI 输出引脚发送.

SPI 模式下 STPOUT / DIROUT 的连接

为了利用 TMC21x0 的 stealthChop 特性，请连接 TMC4361A 的 STPOUT 及 DIROUT。否则，该功能将不起作用。

spreadCycle 操作，也建议使用这种连接，但不是必需的。如果无连接，只有 IHOLD 用于运动。请参考第 14 页图 11。

8.5.2. 设置 TMC21x0 (S/D 模式)

使能 S/D 模式和功能集(针对已连接的 TMC21x0 步进电机驱动器)，请执行以下操作:

- ➢ 将 SPI 输出引脚和 S/D 输出连接到 TMC21x0 步进电机驱动。

操作步骤:

- ➢ 设置 *spi_output_format* = b'1100 (*SPI_OUT_CONF* 寄存器 0x04)。
- ➢ 设置 *COVER_DATA_LENGTH* = 0 (*SPI_OUT_CONF* 寄存器 0x04)。
- ➢ 根据硬件设置 *DIR_SETUP_TIME* 和 *STP_LENGTH_ADD* (寄存器 0x10)。
- ➢ 设置适合的 *POLL_BLOCK_EXP* (*SPIOUT_CONF* 寄存器 0x04 的位 11：8)。

结果:
设置 S/D 模式的 TMC21x0 作为连接的步进电机驱动器。SPI 输出引脚仅传输覆盖数据报和自动配置数据报，因为运动是通过处理 TMC4361A 的 STPOUT/DIREOUT 输出信号产生的。
下一个查询数据报在上一个查询数据报后的 *2^POLL_BLOCK_EXP · SPI_BLOCK_TIME* 时钟周期后发送。

i 更高的微步频率要求更短的 SPI 数据报查询时间。

8.5.3. 向 TMC21x0 发送覆盖数据报

根据上述 TMC21x0 设置，TMC4361A 现在自动发送 40 位数据报。

实现向 TMC21x0 驱动器发送覆盖数据报，请执行以下操作:

操作步骤:

- 设置 *COVER_HIGH* (7:0) 寄存器 0x6D 为需要发送的地址值。
- 设置 *COVER_LOW* (31:0) 寄存器 0x6C 为需要发送的数据值。

结果:

发送覆盖数据报到连接的驱动器。数据传输完成后 *COVER_DONE* 有效。*COVER_DRV_HIGH* (7:0)和 *COVER_DRV_LOW* (31:0)为 TMC21x0 的响应数据。

如果 TMC21x0 驱动器工作在 SPI 模式，传输电流数据报完成时 *COVER_DONE* 也会有效。同样适用于查询数据报，详见第 10.8.8 节第 15 页。

设置仅对覆盖数据报使能 *COVER_DONE*，请执行以下操作:

操作步骤:

- 设置 *cover_done_only_for_covers* = 1 (*SPI_OUT_CONF* 寄存器 0x04 的第 12 位)。

结果:

COVER_DONE 事件仅在发送覆盖数据报时有效，而发送电流数据不产生。

8.5.4. 自动连续的 TMC21x0 覆盖数据流

如果 TMC21x0 的 VS 引脚上出现的电压降，通常内部寄存器全部复位，因此微控制器需要连续重写 TMC21x0 的寄存器值。

TMC4361A 支持连续重写 TMC21x0 的五个配置寄存器，从而减轻微控制器的工作负荷。这些寄存器是: **GCONF 0x00, IHOLD_IRUN 0x10, CHOPCONF 0x6C, COOLCONF 0x6D 和 DCCTRL 0x6E**。

使能自动连续传输 TMC21x0 覆盖数据报，请执行以下操作:

操作步骤:

- 设置 *autorepeat_cover_en* = 1 (*SPI_OUT_CONF* 寄存器 0x04 的第 7 位)。

结果:

如果 *autorepeat_cover_en* = 1，则 TMC4361A 每 2^{20} 个时钟周期传输一个覆盖数据报 TMC21x0 寄存器(如上所述)。每次寻址另一个寄存器后，这五个覆盖数据报都会一个接一个地被连续重传；即循环。

i 然而，传输速率保持在每 2^{20} 个时钟周期一个数据报。

注意:

→ *当 TMC21x0 在 SPI 模式下工作时，如果值不变，在一个传输间隔周期内电流数据报也会重复出现。*

→ *如果上面提到的五个 TMC21x0 寄存器中的一个被覆盖数据报手动重写，根据定义，最新的寄存器值被重复。*

→ *TMC4361A 执行的自动寄存器更改——例如电流自动调节值传输——也按照重复的覆盖数据报一样处理考虑。*

在自动传输覆盖数据报期间手动发送覆盖数据报

注意

如果在自动传输覆盖数据报期间使能手动传输覆盖数据报，则存在无法将手动发送的覆盖数据报传输到 TMC21x0 的风险！

- **您必须在 2^{20} 个时钟周期内发送两次相同的覆盖数据报。**

这将确保成功的手动传输覆盖数据报。

8.5.5. TMC21x0 SPI 模式: 自动切换全步模式

因为 SPI 模式下电机电流是通过 SPI 传输更新的，所以微步和全步之间的自动切换完全取决于内部斜坡速度。

实现自动切换微步和全步，请执行以下操作:

操作步骤:

- 根据切换速度[pps]的绝对值设置 *FS_VEL* 寄存器 0x60 。
- 设置 *fs_en* = 1 (*GENERAL_CONF* 寄存器 0x00 的第 19 位)。

结果:

现在，在| *VACTUAL* | ≥ FS_VEL 的情况下，则切换到全步控制。
在| *VACTUAL* | < *FS_VEL* 的情况下，全步切换到细分。
只要使能并且全步有效，状态位 *FS_ACTIVE* 有效。

8.5.6. TMC21x0 S/D 模式: 自动切换全步模式

在 S/D 模式下，TMC21x0 直接执行微步和全步之间切换(反之亦然)。因此，只需在 TMC21x0 中定义全步速度。无论 TMC4361A 是在全步模式还是微步模式下运行，都会输出微步脉冲。

8.5.7. TMC 21x0 S/D 模式: 更改电流调节参数

TMC4361A 提供了自动改变电流的功能，这将在第 11 章第 120 页进行解释。支持 SPI 电流数据报的步进电机驱动器通过电流数据报自动调节其电流值。TMC4361A 发送覆盖数据报更改 TMC21x0 IHOLD_IRUN 寄存器 CS 值实现 S/D 模式的 TMC21x0 电机步进驱动电流的调节。

实现 TMC21x0 的 S/D 模式的自动电流调节:

操作步骤:

- 设置 *scale_val_transfer_en* = 1 (*SPI_OUT_CONF* 寄存器 0x04 的第五位)。
- 根据需求设置调节寄存器 0x06 和调节配置寄存器 0x05 (见第 11 章第 120 页)。

结果:

当内部调整电流值时，TMC4361A 自动向 TMC21x0 发送更改 CS 位的覆盖数据报。
每当特定寄存器被新分配的电流调节值覆盖时，都应预先设置 TMC21x0 的 IHOLD_IRUN 寄存器(执行覆盖数据报之前)。

i 请考虑 IRUN 和 IHOLD 值仅由 5 位组成。因此，寄存器 0x06 中的电流调节参数值也应为 5 位。

TMC21x0 状态位 TMC4361A 将 TMC21x0 步进驱动器的以下状态位映射到 *STATUS* 寄存器 0x0F，每次 SPI 响应返回这些状态位:

TMC2130 和 TMC2160 的状态寄存器映射		
状态位 @TMC4361A	**状态标志 @TMC21x0**	**描述**
STATUS (24)	SG	stallGuard2™ 状态标志
STATUS (25)	OT	过温标志
STATUS (26)	OTPW	温度预警标志
STATUS (27)	S2GA	线圈 A 高端 MOSFET 对地短路检测标志
STATUS (28)	S2GB	线圈 B 高端 MOSFET 对地短路检测标志
STATUS (29)	OLA	桥臂 A 的开路标志
STATUS (30)	OLB	桥臂 B 的开路标志
STATUS (31)	STST	静止标志

i 如果查询功能未禁用(disable_polling= 0)，TMC21x0 的 S/D 模式也可以查询状态数据。

表 46: TMC21x0 状态标志的映射

8.5.8. TMC21x0 状态响应

TMC4361A 连续获得 TMC21x0 的五个状态寄存器(如果 disable_polling 未禁用)。这些寄存器是 GSTAT 0x01, PWM_SCALE 0x71, LOST_STEPS 0x73 和 DRV_STATUS 0x6F。

存储查询的 TMC21x0 的状态寄存器值，请执行以下操作:

操作步骤:

➢ 设置 *disbale_polling* = 0 (*SPI_OUT_CONF* 寄存器 0x04 第五位)。

结果:

TMC4361A 的 *POLLING_STATUS* 寄存器 0x6C 存储 *DRV_STATUS* 状态。
GSTAT、PWM_SCALE 和 LOST_STEPS 响应值被合并到可读的 *POLLING_REG* 寄存器 0x6D 中。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

9. 电流调节

TMC4361A 微步查找表(MSLUT)寄存器 0x7A–CURRENTA 和 CURRENTB–的电流数据格式为 9 位带符号数据，可通过 SPIOUT 输出接口发送。在速度斜坡的大多数阶段，不需要以全电流幅度驱动电机。可以配置允许的 MSLUT 电流值适应所需要的斜坡状态。调节参数有上升电流、保持电流和驱动电流。

这些参数可与 SCALE_VALUES 寄存器 0x06 中独立。如果使能，在速度斜坡的不同阶段这些参数有效，如下所述。在描述各种可行的调节情况之前，先简要说明调节参数的计算。

电流输出值的计算 使能在当前斜坡阶段调节电流时，MSLUT 的实际电流值与 MULT_SCALE 参数相乘，MULT_SCALE 参数是四个 *SCALE_VALUES* 中的一个推导计算出来的:

计算电流调节的描述

MULT_SCALE = (actual_SCALE_VAL + 1) / 256
with actual_SCALE_VAL = {HOLD, BOOST, DRV1, DRV2}.

因此，这个 MULT_SCALE 的范围是从 0 到 1: $0 < MULT\_SCALE \le 1$.

再将 MULT_SCALE 乘以 MSLUT 对应的实际电流值 CURRENTA 和 CURRENTB:

$CURRENTA\_SPI = CURRENTA \cdot MULT\_SCALE$	(0x7B 的位 8:0)
$CURRENTB\_SPI = CURRENTB \cdot MULT\_SCALE$	(0x7B 的位 24:16)

SPI 输出接口传输计算后的参数值。如果没使能电流调节，输出值 *CURRENTA_SPI* 和 *CURRENTB_SPI* 等于 MSLUT 值 *CURRENTA* 和 *CURRENTB*，因为默认的电流调节值等于最大值 255。因此，电流调节只会降低原始 MSLUT 值。
此外，因为 TMC4361A 可能从一个调节值平滑转换到另一个调节值。可通过中间变量 *SCALE_PARAM* 寄存器 0x7C 中读出实际电流调节参数。数值范围与四个 *SCALE_VALUES* 一致。

特别注意

在 S/D 模式下使用 TMC26x 和 TMC21x0 步进电机驱动器:
S/D 模式的 TMC 电机步进驱动器，因为 TMC26x 的 CS 和 TMC21x0 电机步进驱动器的 IHOLD、IRUN 的电流参数只有 5 位，可以直接修改。因此，MULT_SCALE 的计算略有不同

MULT_SCALE = (actual_SCALE_VAL + 1) / 32

 阅读整个文档; 是第 22 章(第 174 页)中的补充指令。

9.1. 保持电流调节

大多数应用的电机静止状态的电流可以显著减小，因为能量需求低于运动期间。除了调节电流值之外，还必须配置待机延迟时间。延迟定义了斜坡停止和使能静态保持电流之间的时间。若设置延迟时间为 0，则速度斜坡结束时静止保持电流立即有效。然而大多数应用要求在斜坡停止后等待系统振荡，因此通常必须设置此延迟。

设置及使能静态保持电流调节，请执行以下操作:

操作步骤:
- 根据斜坡停止后和待机阶段开始前的的时间范围设置 *STDBY_DEALY* 寄存器 0x15。
- 根据电机停止期间的最大电流设置 *HOLD_SCALE_VAL* = *SCALE_VALUES* (31:24)。
- 设置 *hold_current_scale_en* = 1 (*CURRENT_CONF* 寄存器 0x05)。
- 设置 *closed_loop_scale_en* = 0 (*CURRENT_CONF* 寄存器 0x05) 。

结果:
一旦 *VACTUAL* 达到 0，待机计时器就会启动。*STDBY_DELAY* 时钟周期后，待机定时器定时到，保持电流有效。

待机状态

STDBY_CLK 输出引脚可输出待机状态。

输出待机信号，请执行以下操作:

操作步骤:
- 设置 *stdby_clk_pin_assignment* (1) = 0 (*GENERAL_CONF* 寄存器 0x00 的第 14 位)。
- 根据输出引脚的有效电压电平设置 *stdby_clk_pin_assignment* (0) (*GENERAL_CONF* 寄存器 0x00 的第 13 位)。

结果:
STDBY_CLK 引脚输出内部产生的待机状态。stdby_clk_pin_assignment (0)设置引脚的有效输出电平。

9.1.1. Boost 电流

在速度斜坡的某些阶段，提高电流是有用的。上升电流可以暂时分配给斜坡开始后或整个加速/减速阶段。所有选项可以单独设置，也可以组合设置。

i 所有三个选项都使用相同的调节值 *BOOST_SCALE_VAL*。

选项 1:斜坡开始时的电流上升

设置和使能在速度斜坡开始后定义的时间内调节上升电流，请执行以下操作:

操作步骤:
- 根据速度斜坡开始后电流上升调节的时间，设置 *BOOST_TIME* 寄存器 0x18。
- 根据最大上升电流设置 *BOOST_SCALE_VAL* = *SCALE_VALUES* (7:0) 。
- 设置 *boost_current_after_start_en* = 1 (*CURRENT_CONF* 寄存器 0x05)。
- 设置 *closed_loop_scale_en* = 0 (*CURRENT_CONF* 寄存器 0x05)。

结果:
速度斜坡开始后(*VACTUAL* = 0)，根据 *BOOST_SCALE_VAL* 设置立刻调节电流上升。上升电流定时器在 *BOOST_TIME* 时钟周期后结束。之后，如果使能并选择其它调节，则使用其他选定的调节值。

选项 2:加速斜坡上的电流上升

设置和使能在速度斜坡加速阶段调节上升电流，请执行以下操作:

操作步骤:
- 根据最大上升电流设置 *BOOST_SCALE_VAL* = *SCALE_VALUES* (7:0) 。
- 设置 *boost_current_on_acc_en* = 1 (*CURRENT_CONF* 寄存器 0x05)。
- 设置 *closed_loop_scale_en* = 0 (*CURRENT_CONF* 寄存器 0x05)。

结果:
只要内部绝对速度| *VACTUAL* |增加，根据 *BOOST_SCALE_VAL* 设置立刻调节电流上升。*RAMP_STATE* 反应当前斜坡状态。*RAMP_STATE* = b'01 为加速阶段。

选项 3:在减速斜坡上提升比例

设置和使能在斜坡减速阶段调节电流上升，请执行以下操作:

操作步骤:

- 根据最大上升电流设置 *BOOST_SCALE_VAL* = *SCALE_VALUES*(7:0) 。
- 设置 *boost_current_on_dec_en* = 1 (*CURRENT_CONF* 寄存器 0x05)。
- 设置 *closed_loop_scale_en* = 0 (*CURRENT_CONF* 寄存器 0x05)。

结果:

只要内部绝对速度| *VACTUAL* |降低，根据 *BOOST_SCALE_VAL* 设置立刻调节上升电流。*RAMP_STATE* 反应当前斜坡状态。*RAMP_STATE* = b'10 为减速阶段。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

10. NFREEZE 和紧急停止

如果电路板级出现功能障碍，某些应用需要对策来立即结束当前操作。因此，TMC4361A 提供低有效安全引脚 NFREEZE 满足该需求。

NFREEZE 操作原则 NFREEZED 低有效，NFREEZE 输入信号从高到低的跳变立即停止电流斜坡。

当 NFREEZE 切换到低电平时，事件(10)的触发事件 *FROZEN* 有效。在 TMC4361A 复位之前，“*FROZEN*”保持激活状态。

特别注意

输入滤波器需要连续采样三个点，因此 NFREEZE 低电平时间至少保持三个时钟周期。

引脚描述: NFREEZE		
引脚名称	**类型**	**备注**
NFREEZE	输入	外部使能引脚；低有效。

表 49: 引脚描述: NFREEZE

寄存器描述: DFREEZE 和 IFREEZE			
寄存器名称	**寄存器地址**		**备注**
DFREEZE	0x4E (23:0)	RW	有效 FREEZE 事件中的减速度值。
IFREEZE	0x4E (31:24)	RW	有效 FREEZE 事件中的电流调节值。

表 50: 寄存器 DFREEZE 和 IFREEZE

10.1.1. FREEZE 功能配置

冻结寄存器的两个参数(*DFREEZE* and *IFREEZE*)是使能冻结功能的必要参数，**在有效复位后，且没有斜坡开始情况，只能写入一次**。因此，应在操作前直接设置冻结参数。

注意:

→ *在下一次复位之前，不能更改选定的值。这些限制是必要的，以保护 TMC4361A 冻结配置，防止微控制器在出错时发送不正确的 SPI 接口数据。*

特别注意

请牢记:

- 不能改变 NFREEZE 输入有效极性。
- 冻结 freeze 寄存器可读。
- 冻结 freeze 状态中，斜坡寄存器可读。

10.1.2. 配置 DFREEZE 自动斜坡停止

***DFREEZE* 可用于自动斜坡停止配置。有两种选择:**

- **选项 1:** *DFREEZE* = 0 用于硬停。
- **选项 2:** *DFREEZE* ≠ 0 用于线性减速斜坡。

原则:

TMC4361A 的内部寄存器物理单位可以设置成 *direct_acc_val_en* 或按照给定的时钟频率 f_{CLK}(可能被错误的 SPI 信号改变)，而减速值 *DFREEZE* 独立于内部寄存器值，总是以每个时钟周期的速度值变化给出。因此，DFREEZE 值计算如下:

$$d\_freeze\ [pps^2] = DFREEZE / 2^{37} \cdot f_{CLK}^2$$

电机的运行效果，等同于正常运行期间将其它加速度值的 *direct_acc_val_en* 设置为 1 一样。

配置 IFREEZE 的电流调节值

***IFREEZE* 设置冻结事件期间的电流值。有两种选择:**

- **选项 1:** *IFREEZE* = 0 冻结事件的电流值为冻结前最后指定的电流值。
- **选项 2:** *IFREEZE* ≠ 0 冻结事件的电流值为 *IFREEZE* 的值。

原则:

IFREEZE 是电流调节值，当 *NFREEZE* 变低并且相关事件(*FROZEN*)有效的情况下，该值有效。
如果 *IFREEZE* 设置为 0，则冻结事件的电流值是紧急事件前的最新电流调节值。
电流调节值 *IFREEZE* 按第 11 章第 120 页中解释的方式控制电流值。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

11. 解码器单元:正确连接 ABN、SSI 或 SPI 编码器

TMC4361A 支持增量式 ABN 编码器、绝对 SSI 或 SPI 编码器。本章解释了如何连接编码器信号和相关配置。

解码器引脚		
引脚名称	**类型**	**备注**
A_SCLK	输入或输出	ABN 编码器的 A 相信号或 绝对 SSI 或 SPI 编码器的串行时钟输出
ANEG_NSCLK	输入或输出	ABN 编码器的 A 相反相信号或 SSI 编码器的串行时钟反相信号输出或 SPI 编码器的低有效片选信号。
B_SDI	输入	ABN 编码器的 B 相信号或 SSI 或 SPI 编码器的串行数据输入。
BNEG_NSDI	输入或输出	ABN 编码器的反相 B 信号或 SSI 编码器的反相串行数据输入或 SPI 编码器的串行数据输出。
N	输入	ABN 编码器的 N 信号。
NNEG	输入	ABN 编码器的 N 反相信号。

表 55: 专用解码器单元引脚

解码器单元寄存器			
寄存器名	**寄存器地址**		**备注**
GENERAL_CONF	0x00	RW	位 11:10: serial_enc_in_mode, 位 12: diff_enc_in_disable
INPUT_FILT_CONF	0x03	RW	输入滤波器配置(SR_ENC_IN, FILT_L_ENC_IN)。
ENC_IN_CONF	0x07	RW	编码器配置寄存器。
ENC_IN_DATA	0x08	RW	串行编码器输入数据结构。
STEP_CONF	0x0A	RW	电机配置。
ENC_POS	0x50	RW	当前绝对编码器位置，单位为微步。
ENC_LATCH	0x51	R	绝对编码器的锁存位置。
ENC_POS_DEV	0x52	R	*XACTUAL* 和 *ENC_POS* 之间的偏差。
ENC_CONST	0x54	R	内部计算的编码器常数。
编码器寄存器指令集	0x51…58 0x62…63	W	编码器配置寄存器参数。
编码器速度	0x65 0x66	R	当前编码器速度(带符号)。 当前滤波编码器速度(带符号)
ADDR_TO_ENC *DATA_TO_ENC*	0x68 0x69	W	串行编码器请求数据。
ADDR_FROM_ENC *DATA_FROM_ENC*	0x6A 0x6B	R	串行编码器请求数据响应。
编码器补偿	0x7D	W	编码器补偿寄存器组。

表 56: 专用解码器单元寄存器

11.1.1. 选择正确的编码器

编码器接口有六个引脚，可连接不同类型的编码器。根据编码器类型，引脚用作输入或输出。如果配置为输入引脚，还可配置滤波处理，如第 20 页第 4 章所述。因此，须相应地设置 *SR_ENC_IN* 和 *FILT_L_ENC_IN*。下面给出正确连接编码器的三种方法。

选项 1:增量 ABN 编码器

选择增量 ABN 编码器，请执行以下操作:

操作步骤:

- 设置 *serial_enc_in_mode* = b'00 (*GENERAL_CONF* 寄存器 0x00)。

结果:

选择增量 ABN 编码器。

选项 2:绝对 SSI 编码器

选择绝对 SSI 编码器，请执行以下操作:

操作步骤:

- 设置 *serial_enc_in_mode* = b'01 (*GENERAL_CONF* 寄存器 0x00)。

结果:

i 为了避免连接的绝对 SSI 编码器出现错误，在使能之前需要适当的配置；后续页面会详细描述：参见第 15.4 节，在第 149 页。

选择绝对 SSI 编码器。

选项 3:绝对 SPI 编码器

为了设置连接的绝对 SPI 编码器:

操作步骤:

- 设置 *serial_enc_in_mode* = b'11 (*GENERAL_CONF* 寄存器 0x00)。

结果:

选择绝对 SPI 编码器。

i 为了避免绝对 SPI 编码器出现错误，在使能之前需要适当的配置；后续页面会详细描述:参见第 15.4 节，在第 149 页。

翻页查看编码器引脚分配概述。

11.1.2. 禁用数字编码器差分信号

如果选择增量 ABN 或绝对 SSI 编码器，编码器信号默认为数字差分信号。如果编码器只有单端信号，则必须进行相应设置产生有效的电平。

i 无模拟差分电路可用。.

禁用数字差分输入信号，请执行以下操作:

操作步骤:

➢ 设置 *diff_enc_in_disable* = 1 (*GENERAL_CONF* 寄存器 0x00)。

结果:

编码器信号为单端信号，忽略负端的引脚。

i 如果选择绝对 SPI 编码器，系统默认编码器输入信号为单端输入信号。

编码器设置的引脚分配						
Pin No.	Pin Name	增量 ABN		绝对 SSI		绝对 SPI
		差分	单端	差分	单端	单端
40	A_SCLK	A	A	SCLK	SCLK	SCLK
1	ANEG_NSCLK	¬A	-	¬SCLK	-	CS
10	B_SDI	B	B	SDI	SDI	SDI
11	BNEG_NSDI	¬B	-	¬SDI	-	SDO
21	N	N	N	-	-	-
22	NNEG	¬N	-	-	-	-

表 57:编码器设置对应的引脚分配

11.1.3. 编码器方向的反转

可以设置开关量来反转编码器方向便于实现编码器方向与电机方向对齐。

反转编码器方向，请执行以下操作:

操作步骤:

➢ 设置 *invert_direction* = 1 (*ENC_CONF* 寄存器 0x07)。

结果:

外部位置 *ENC_POS* 的计算被反转，将增量变为减量，反之亦然。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

11.2. 增量 ABN 编码器设置

增量 ABN 编码器根据 A 和 B 信号电平变化增加或减少外部位置计数器寄存器 *ENC_POS* 0x50。

11.2.1. 增量式 ABN 编码器的自动常数配置

外部位置寄存器 ENC_POS 0x50 以内部微步为计量单位。因此，每个 AB 信号跳变都通过固定的常数值转换到微步。TMC4361A 可以自动计算该常数。

自动配置增量 ABN 编码器常数，请执行以下操作:

操作步骤:

- 根据电机的全步分辨率设置 *FS_PER_REV*(*STEP* _CONF 寄存器 0x0A)。
- 根据微步分辨率设置 *MSTEP_PER_FS* (*STEP_CONF 寄存器* 0x0A)。
- 根据编码器分辨率即电机旋转一圈期间 AB 转换的次数设置寄存器 *ENC_IN_RES* 0x54(写访问)。

结果:

编码器常数值 *ENC_CONST*(寄存器 0x54 可读)计算如下:

$$ENC\_CONST = MSTEP\_PER\_FS \cdot FS\_PER\_REV / ENC\_IN\_RES$$

该常数是 *AB 变化后*增加或减少一个 *ENC_POS* 的微步数。

i *ENC_CONST* 由 15 位数字和 16 位小数组成。

i 如果不能用二进制准确表示 16 位小数的数值，TMC4361A 会尝试将这 16 位小数乘以 10000 与十进制匹配，如果十进制可以准确表达，则十进制表示优于二进制。

i 如果十进制表示也不完全适合，可设置 *ENC_IN_CONF* (0)手动选择 *ENC_CONST* 的小数位数类型。ENC_IN_CONF (0)为 0 对应二进制；或 1 对应十进制。请注意，使用这种方法，ENC_POS 可能与实际位置略有不同；尤其是离 0 越远的位置。

11.2.2. 增量 ABN 编码器的手动常数配置

对于某些应用，例如如果编码器没有直接安装在电机上，定义编码器常数值是有用的，在这种情况下，该常数值不对应于每转的微步数；

手动配置增量 ABN 编码器常数，请执行以下操作:

操作步骤:

- 设置 *ENC_IN_RES*(31) =1。
- 如前一节所述，设置 *ENC_IN_CONF*(0)为 0 对应二进制；或者设置为 1 对应十进制。
- 根据编码器分辨率设置 *ENC_IN_RES* (30:0)寄存器 0x54。

结果:

ENC_CONST 由 15 位整数和 16 位小数组成。常数是每次 AB 转换对应 *ENC_POS* 递增或递减的微步数。

 阅读整个文档；是第 22 章(第 174 页)中的补充指令。

11.3. 增量编码器:索引信号: N 或者 Z

索引信号(N 或 Z 通道)在每次电机编码器旋转的相同位置重复出现。TMC4361A 利用该信号清除外部位置计数器，或者捕获外部或内部位置，可用于更精确地调整初始位置用于回零操作。

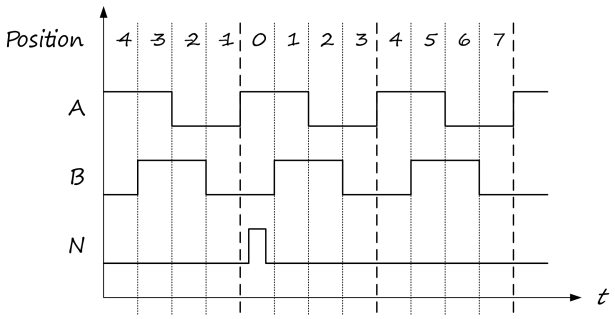


图 62: 增量编码器 ABN 信号概述

11.3.1. 设置索引通道的有效极性

默认情况下，索引通道为低有效状态.

设置索引通道有效极性为高，请执行以下操作:

操作步骤:

- 设置 *pol_n* =1 (寄存器 *ENC_CONF* 0x07)。

结果:
索引通道 高有效。

翻页查看 N 事件配置选项。

11.3.2. N 事件的配置

索引通道的有效极性可用于清除外部位置计数器或捕获外部或内部位置。因此，N 事件在内部创建。N 事件根据索引通道的有效极性产生。此外，还可设置与 A 和 B 通道的极性相关。

索引通道敏感度

索引通道有四种有效极性配置选项，如下所示。配置选择取决于客户的设计。

要根据有效极性设置索引通道灵敏度，请执行以下操作:

操作步骤:

➢ 设置 *n_chan_sensitivity* (寄存器 *ENC_CONF* 0x07):

索引通道灵敏度	
n_chan_sensitivity	**结果**
b'00	如果索引电压电平符合 *pol_n*，则 N 事件有效。
b'01	当索引通道切换到有效极性时，N 事件有效。
b'10	当索引通道切换到非有效极性时，N 事件有效。
b'11	当索引通道切换到有效或非有效极性时，在两个边沿触发 N 事件有效。

表 58: 索引通道灵敏度

N 事件的 A 和 B 通道信号极性

为 N 事件指定 A 和 B 通道极性可能很有用。默认情况下，两条信号线的极性都设置为 0(低有效)。

设置 A 通道有效极性为高电平激活 N 事件，请执行以下操作:

操作步骤:

➢ 设置 *pol_a_for_n* = 1 (*ENC_CONF* 寄存器 0x07)。

结果:

现在，产生 N 事件的 A 通道信号的有效极性是高电平。

设置 B 通道极性为高电平激活 N 事件，请执行以下操作:

操作步骤:

➢ 设置 *pol_b_for_n* = 1 (*ENC_CONF* 寄存器 0x07)。

结果:

现在，产生 N 事件的 B 通道信号有效极性是高电平。

如果 A 和 B 通道极性对 N 事件没有影响，A 和 B 通道极性信号都可以忽略。

忽略 A 和 B 通道极性，请执行以下操作:

操作步骤:

➢ 设置 *ignore_ab* = 1 (*ENC_CONF* 寄存器 0x07)。

结果:

A 和 B 通道极性对 N 事件无影响。

11.3.3. 检测 N 事件

如果按照前面章节中的说明正确配置了 N 事件，有两种检测 N 事件的选择:连续检测和单次检测。

连续检测 **实现连续检测 N 事件，请执行以下操作:**

操作步骤:

- 正确配置 N 事件。
- 设置 *clr_latch_cont_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。

结果:

每发生一次事件，*N_ACTIVE_*Flag 标志(*STATUS* 寄存器位 15)有效。此外，*N_ACTIVE* 事件(*EVENTS* 寄存器位 19)将被置位，直到它被清零(参见第 5.1 章第 25 页)。

单次检测 **只检测下一个 N 事件，请执行以下操作:**

操作步骤:

- 正确配置 N 事件。
- 设置 *clr_latch_cont_on_n* = 0 (*ENC_CONF* 寄存器 0x07)。
- 设置 *clr_latch_once_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。

结果:

当下一个 N 事件发生时，*N_ACTIVE_Flag*(*STATUS* 寄存器的第 15 位)有效。此外，*N_ACTIVE* 事件(*EVENTS* 寄存器位 19)将被置位，直到它被清零(参见第 5.1 章第 25 页)。在对应的 N 事件之后，*clr_latch_once_on_n* 会自动复位为 0。

11.3.4. 外部位置计数器 *ENC_POS* 清零

N 事件可用于清除外部位置寄存器 *ENC_POS* 0x50。也有两种选择:连续清零和单次清零。

i 通常的做法是清零。但 TMC4361A 可以将外部寄存器 *ENC_POS* 清除为任何微步值。

ENC_POS 连续清零 **设置 N 事件连续清除 *ENC_POS* 位置，请执行以下操作:**

操作步骤:

- 根据复位需要的微步位置设置 *ENC_RESET_VAL* 寄存器 0x51。
- 设置 *clr_latch_cont_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。
- 设置 *clear_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。

结果:

在每一个 N 事件中，*ENC_POS* 被设置为 *ENC_RESET_VAL*。

ENC_POS 单次清零 **只清除下一个 N 事件的 *ENC_POS* 位置，请执行以下操作:**

操作步骤:

- 根据复位需要的微步位置设置 *ENC_RESET_VAL* 寄存器 0x51。
- 设置 *clr_latch_cont_on_n* = 0 (*ENC_CONF* 寄存器 0x07)。
- 设置 *clr_latch_once_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。
- 设置 *clear_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。

结果:

当下一个 N 事件发生时，*ENC_POS* 被设置为 *ENC_RESET_VAL*。
在对应的 N 事件之后，*clr_latch_once_on_n* 会自动复位为 0。

11.3.5. 锁存外部位置

N 事件可将外部位置寄存器 *ENC_POS* 0x50 锁存到 *ENC_LATCH* 寄存器 0x51(读取访问)。有两种操作:连续锁存和单次锁存。

连续编码器锁存

实现当 N 事件发生时持续锁存 *ENC_POS* 到 *ENC_LATCH*，请执行以下操作:

操作步骤:

- 设置 *clr_latch_cont_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。
- 设置 *latch_enc_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。

结果:

每发生一次 N 事件，*ENC_LATCH* 寄存器 0x51 锁存 *ENC_POS* 寄存器 0x50。

单次编码器锁存

实现只在下一个 N 事件中将 *ENC_POS* 锁存到 *ENC_LATCH*，请执行以下操作:

操作步骤:

- 设置 *clr_latch_cont_on_n* = 0 (*ENC_CONF* 寄存器 0x07)。
- 设置 *clr_latch_once_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。
- 设置 *latch_enc_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。

结果:

当下一个 N 事件发生时，*ENC_LATCH* 寄存器 0x51 锁存 *ENC_POS* 寄存器 0x50。在相应的 N 事件之后，*clr_latch_once_on_n* 会自动复位为 0。

11.3.6. 锁定内部位置

N 事件可将内部位置寄存器 *X_ACTUAL* 0x21 锁存到寄存器 *X_LATCH* 0x36(读取访问)。有两种选择:连续锁存和单一锁存。

连续锁存

配置 N 事件连续锁存 *X_ACTUAL* 到 *X_LATCH*，请执行以下操作:

操作步骤:

- 设置 *clr_latch_cont_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。
- 设置 *latch_enc_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。
- 设置 *latch_x_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。

结果:

在每一个 N 事件中，*X_LATCH* 寄存器 0x36 锁存 *X_ACTUAL* 寄存器 0x21。

单次锁存

配置只在下一个 N 事件中锁存 *X_ACTUAL* 到 *X_LATCH*，请执行以下操作:

操作步骤:

- 设置 *clr_latch_cont_on_n* = 0 (*ENC_CONF* 寄存器 0x07)。.
- 设置 *clr_latch_once_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。
- 设置 *latch_enc_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。
- 设置 *latch_x_on_n* = 1 (*ENC_CONF* 寄存器 0x07)。

结果:

当下一个 N 事件发生时，*X_LATCH* 寄存器 0x36 锁存 *X_ACTUA*L 寄存器 0x21。在相应的 N 事件之后，*clr_latch_once_on_n* 会自动复位为 0。

11.4. 绝对编码器设置

串行编码器输出绝对编码器角度数据，这与增量编码器提供的信号脉冲转换不同。
TMC4361A 为编码器提供外部时钟，以便触发串行数据输入。

11.4.1. 单圈或多圈数据

TMC4361A 根据设置的单圈和多圈选项解析串行数据流。缺省不使能多圈数据。在使能多圈数据的情况下，默认数据格式为无符号。

如果传输多圈编码器数据，请执行以下操作:

操作步骤:
- 设置 *multi_turn_in_en* = 1 (*ENC_CONF* 寄存器 0x07)。
- *可选配置:* 设置 *multi_turn_in_signed* = 1。
  在这种情况下，多圈数据为有符号的编码器转数。

结果:
编码器的输入数据按照多圈数据格式处理。

如果仅传输单圈数据，则 TMC4361A 一直计算内部编码器转数，就好像传输多圈数据一样。

如果编码器传输单圈数据，但需要内部多圈数据，请执行以下操作:

操作步骤:
- 设置 *multi_turn_in_en* = 0 (*ENC_CONF* 寄存器 0x07)。
- 设置 *calc_multi_turn_behav* = 1 (*ENC_CONF* 寄存器 0x07)。

结果:
来自单圈编码器的数据在内部被计算成多圈数据。

注意:

→ *多圈计算仅在两个连续单圈数据值相差小于半圈或更小一步的情况下才是正确的。*

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

11.4.2. 绝对编码器的自动常数配置

外部位置寄存器 *ENC_POS* 0x50 单位为内部微步。因此，每个输入角度数据根据设定的常数转换为微步。TMC4361A 能够自动计算该常数。

要自动配置绝对编码器常数，请执行以下操作:

操作步骤:

- 根据电机的全步分辨率设置 *FS_PER_REV*(*STEP* _CONF 寄存器 0x0A)。
- 根据细分分辨率设置 *MSTEP_PER_FS* (*STEP_CONF* 寄存器 0x0A)。
- 根据编码器分辨率设置寄存器 *ENC_IN_RES* 0x54 (写操作)。

结果:

编码器常数值 *ENC_CONST*(寄存器 0x54 可读)计算如下:

$$ENC\_CONST = MSTEP\_PER\_FS \cdot FS\_PER\_REV / ENC\_IN\_RES$$

外部位置 *ENC_POS* 0x50 等于编码器常数乘以输入角度。

i *ENC_CONST* 由 15 位整数和 16 位小数组成。

i 与增量 ABN 编码器不同，*ENC_CONST* 总是二进制格式。

11.4.3. 增量式 ABN 编码器常量的手动配置

某些应用，例如编码器没有直接安装在电机上，需要手动定义编码器常数值，在这种情况下，该常数值不对应于每转的微步数;

实现手动配置绝对编码器常数，请执行以下操作:

操作步骤:

- 设置 *ENC_IN_RES* (31) =1。
- 根据所需的编码器分辨率设置 *ENC_IN_RES* (30:0) 寄存器 0x54。

结果:

ENC_CONST 由 15 位数字和 16 位小数组成。外部位置 *ENC_POS* 0x50 等于编码器常数乘以输入角度。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

11.4.4. 设置绝对编码器数据

需要配置某些寄存器，以便 TMC4361A 按照规定正确显示编码器数据。

配置绝对编码器数据，请执行以下操作:

操作步骤:

- 根据（单圈数据位数-1）设置 *SINGLE_TURN_RES* (*ENC_IN_DATA* 寄存器 0x08) 位宽。

  选项 A1:如果传输多圈数据

  - 根据（多圈数据位数-1）设置 *MULTI_TURN_RES* (*ENC_IN_DATA* 寄存器 0x08)位宽。

  - ***或者选项 A2:如果不传输多圈数据***

  - 设置 *MULTI_TURN_RES* = 0 (*ENC_IN_DATA* 寄存器 0x08)。

- 根据状态位个数设置 *STATUS_BIT_CNT* (寄存器 0x08)。

  - ***选项 B1:如果状态标志排列在前面***

  - 设置 *left_aligned_data* = 0 (*ENC_IN_CONF* 寄存器 0x07)。

  或选项 B2:如果数据排列在前面

  - 设置 *left_aligned_data* = 1 (*ENC_IN_CONF* 寄存器 0x07)。

结果:

SINGLE_TURN_RES 定义角度数据位的最高有效位，而 *MULTI_TURN_RES* 定义旋转圈数的最高有效位。最多可以接收三个状态位。发送到编码器的传输时钟位数计算如下:

#SCLK Cycles= (*SINGLE_TURN_RES*+1) + (*MULTI_TURN_RES*+1) + *STATUS_BIT_CNT*

此外，可以配置状态位在编码器数据流中的顺序。图 63 为示例设置。

注意:

→ *如果编码器发送三个以上的状态位或额外的填充位，可能会出现时钟错误，因为与传输的时钟位数不匹配*

→ *为了防止时钟故障，可以将 MULTI_TURN_RES 设置为比所需更高的值；即使编码器不提供多圈数据。这可能会导致错误的多圈数据，可以通过设置 multi_turn_in_en=0 来纠正，以便自动跳过多圈数据。*

→ *设置 calc_multi_turn_behav 有效可补偿不可用的多圈数据，如章节 15.4.1 第 149 页所述。*

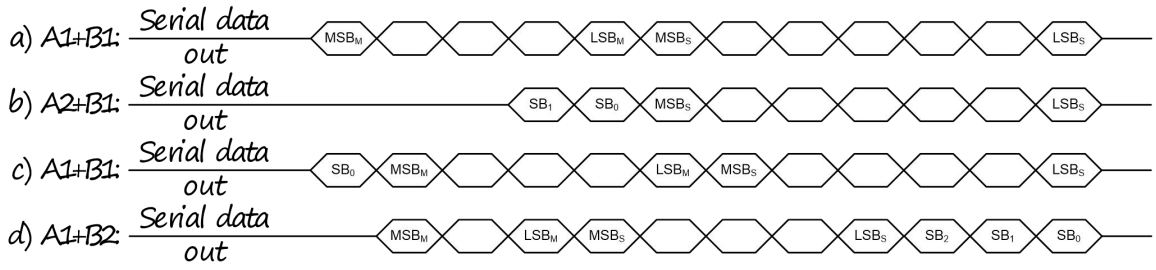


图 63: 串行数据输出:四个例子

Key:

a) SINGLE_TURN_RES=6; MULTI_TURN_RES=4; STATUS_BIT_CNT=0; left_aligned_data=0

b) SINGLE_TURN_RES=6; MULTI_TURN_RES=0; STATUS_BIT_CNT=2; left_aligned_data=0

c) SINGLE_TURN_RES=5; MULTI_TURN_RES=4; STATUS_BIT_CNT=1; left_aligned_data=0

d) SINGLE_TURN_RES=4; MULTI_TURN_RES=2; STATUS_BIT_CNT=3; left_aligned_data=1

11.4.5. 消除编码器噪音

限制两个连续编码器数据值的差值对某些应用可能是有用的，比如编码器数据线受到太多噪声的影响。

如果使能编码器差值限制，两次连续编码器数据的最大数值差默认设置为编码器分辨率的1/8。如有必要，可以将差值配置为较小的值。

使能编码器数据差值限制，请执行以下操作:

操作步骤:

- *可选:* 设置正确的 *SER_ENC_VARIATION* 寄存器 0x63 (7:0)。
- 设置 *serial_enc_variation_limit* =1 (*ENC_IN_CONF* 寄存器 0x07)。

结果:

随后接收的编码器数据值与先前的数据的相差不得多于:

最大容许偏差= *SER_ENC_VARIATION* / 256 · 1/8 · *ENC_IN_RES*。

若差值超过上述限制，内部拒绝新得到的数据值，状态标志 *SER_ENC_DATA_FAIL* 有效。

i 如果 *SER_ENC_VARIATION* = 0, 限幅为 1/8 · *ENC_IN_RES*。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

11.4.6. SSI 时钟生成

TMC4361A 根据 SSI 标准生成时钟接收绝对编码器编码器数据。时钟线 SCLK 从高电平到低电平的切换启动绝对编码器的数据传输，从 SCLK 的下一个上升沿开始数据传输。产生的时钟周期数量取决于预期的数据宽度，如 15.4.4 所述。

配置细节

时钟周期分高低电平，可以根据内部时钟周期分别定义。默认情况下，串行数据的采样点设置在 SCLK 的下降沿。一些编码器需要更多的低电平时钟周期来准备数据传输。此外，较长的线缆可能需要更多时间的传输数据。为了解决上述问题，可以设置补偿的延迟时间 *SSI_IN_CLK_DELAY*(默认值等于 0)，以延长采样开始时间。因此，这种延迟配置可以自动生成更多时钟周期。

数据请求后-当所有时钟周期都发出后-串行时钟必须保持空闲一段时间，下一个请求

i 根据 SSI 标准，时间间隔需大于 21µs。

才会自动启动。该间隔 *SER_PTIME* 也可以根据内部时钟周期配置。

生成 SSI 时钟，请执行以下操作:

操作步骤:

- 设置 *SINGLE_TURN_RES* (*ENC_IN_DATA* 寄存器 0x08) 位宽为（单圈数据位数 -1）。
- 如果使能和使用多圈数据，设置 *MULTI_TURN_RES* (*ENC_IN_DATA* 寄存器 0x08) 位宽为（多圈数据位数-1）。
- 设置 *STATUS_BIT_CNT* (*ENC_IN_DATA* 寄存器 0x08) 为状态位位宽。
- 设置正确的对齐方式 *l*eft_aligned_data (*ENC_IN_CONF* 寄存器 0x07)。
- 设置适合的以内部时钟周期为单位的 *SER_CLK_IN_LOW* (寄存器 0x56)。
- 设置适合的以内部时钟周期为单位的 *SER_CLK_IN_HIGH* (寄存器 0x56)。
- ***可选的配置:*** 设置正确的以内部时钟周期为单位的 *SSI_IN_CLK_DELAY* (寄存器 0x57) 。
- ***可选的配置:*** 设置正确的以内部时钟周期为单位的 *SER_PTIME* (寄存器 0x58)。
- **最后，设置 *serial_enc_in_mode* = b'01**。

结果:

TMC4361A 产生 SCLK 串行时钟流， SDI 信号接收绝对编码器数据。如果 *SSI_IN_CLK_DELAY* > 0，则延后采样 SDI(见下图)。*SER_PTIME* 定义了两个连续数据请求之间的间隔。

i 如果选择差分编码器， SCLK 和 SDI 的反向信号有效。

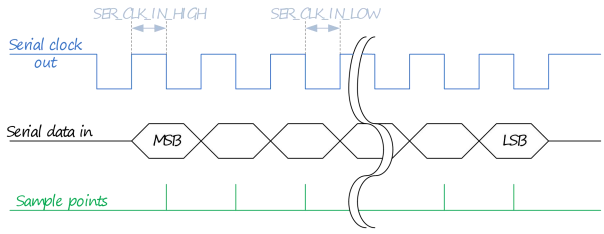


图 64: SSI: SSI_IN_CLK_DELAY=0

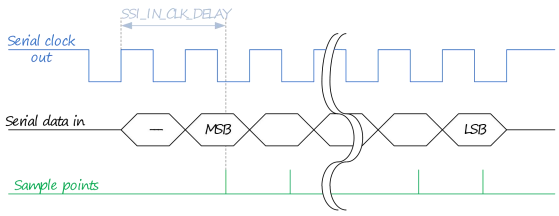


图 65: SSI: SSI_IN_CLK_DELAY>SER_CLK_IN_HIGH

11.4.7. 使能多圈 SSI 请求

可以发送第二个请求让编码器重复相同的编码器数据，确保安全传输。因此，需定义第二个时间间隔 *SSI_WTIME*。

i 根据 SSI 标准，时间间隔需小于 19µs。

使能多圈请求，请执行以下操作:

操作步骤:

- 设置 *ssi_multi_cycle_data* =1 (*ENC_IN_CONF* 寄存器 0x07)。
- 设置正确以内部时钟周期为单位的 *SSI_WTIME* (寄存器 0x57) 。

结果:

数据请求后-当所有时钟周期都发出后-串行时钟在 *SSI_WTIME* 时钟周期内保持空闲。之后，自动使能第二个请求以接收相同的编码器数据。如果第二个编码器数据与第一个不同，则生成错误标志 *MULTI_CYCLE_FAIL*(寄存器 0x0F) 和错误事件 *SER_ENC_DATA_FAIL*(寄存器 0x0E)。
在第二次数据请求之后，间隔持续 *SER_PTIME* 时钟周期后请求新的编码器数据。

11.4.8. 格雷编码数据

某些但不是所有的 SSI 编码器传送格雷编码的角度数据。TMC4361A 能够解码这些数据。

使能格雷编码角度数据，请执行以下操作:

操作步骤:

- 设置 *ssi_gray_code_en* =1 (*ENC_IN_CONF* 寄存器 0x07)。

结果:

编码器数据按格雷码处理。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

11.4.9. SPI 编码器数据

SPI 编码器接口通常由四条信号线组成。除了 SSI 编码器信号线(SCLK，MISO)，还提供了片选信号线(CS)和主机的数据输入(MOSI)。

SPI 编码器通讯处理

每次传输的位数是根据 *multi_turn_in_en*, *SINGLE_TURN_RES*, *MULTI_TURN_RES* 和 *STATUS_BIT_CNT* 的设置自动计算的；如章节 15.4.1 (第 149 页)和章节 15.4.4 (第 151 页)所述。

下一次传输时，SPI 接口会响应任何 SPI 接口数据传输请求。当 TMC4361A 收到编码器的应答时，它会立即计算 *ENC_POS*。编码器从机先接收请求再发送数据。
因此，TMC4361A 总是发送 *ADDR_TO_ENC* 值，向 SPI 编码器从设备请求编码器数据。
串行数据输出的最低位是 *ADDR_TO_ENC* (0)。

i 时钟生成的工作原理类似于 SSI 时钟生成，根据设置的 *SER_CLK_IN_HIGH*, *SER_PTIME* 和 *SER_CLK_IN_LOW* 产生时钟，如章节 15.4.6 第 153 页所述。

ADDR_FROM_ENC 存储接收到的编码器数据，可与微控制器数据进行比较并验证。

配置 SPI 接口通信过程，请执行以下操作:

操作步骤:

- 设置 *SINGLE_TURN_RES* (*ENC_IN_DATA* register 0x08) 位数为（单圈数据位数-1）。
- 如果使能多圈数据，则设置 *MULTI_TURN_RES* (*ENC_IN_DATA* 寄存器 0x08) 位数为（多圈数据位数-1）。
- 设置 *STATUS_BIT_CNT* (*ENC_IN_DATA* 寄存器 0x08)为状态位数。
- 设置正确的对齐方式 *left_aligned_data* (*ENC_IN_CONF* 寄存器 0x07)。
- **设置正确的 SPI 传输模式，见下一章节。**
- 设置 *ADDR_TO_ENC* 寄存器 0x68 为包含角度数据的 SPI 编码器地址。
- 设置适合的以内部时钟周期为单位的 *SER_CLK_IN_LOW* (寄存器 0x56)。
- 设置适合的以内部时钟周期为单位的 *SER_CLK_IN_HIGH* (寄存器 0x56)。
- ***可选配置:***设置适合的以内部时钟周期为单位的 *SER_PTIME* (寄存器 0x58)。
- **最后,设置 *serial_enc_in_mode* = b'11**。

结果:

SCLK 产生串行时钟流，接收 SDI 引脚绝对编码器数据。产生的时钟个数取决于 *SINGLE_TURN_RES*, *MULTI_TURN_RES* 和 *STATUS_BIT_CNT* 的设置。
引脚 ANEG_NSCLK 是根据串行时钟和所选串行接口模式生成的 SPI 编码器的反相片选信号线；这将在下一节中描述。
引脚 BNEG_NSDI 是将串行接口数据报传输到串行接口编码器的 MOSI 线。
接收角度数据的数据报由 *ADDR_TO_ENC* 数据组成。
SER_PTIME 定义了两个连续数据请求之间的间隔。

ꕥ翻页了解 SPI 模式信息。

11.4.10. SPI 编码器模式选择

默认情况下，SPI 编码器数据传输方式与微控制器和 TMC4361A 之间的通信方式相同。TMC4361A 设置 *spi_low_before_cs* 和 *spi_data_on_cs* 支持所有四种 SPI 模式。

过程如下:

设置 ***spi_low_before_cs* = 0**，ANEG_NSCLK 引脚上的反相片选信号线在串行时钟线 SCLK 切换**之前**被切换为有效的低电平。

设置 ***spi_low_before_cs* = 1**，ANEG_NSCLK 引脚上的的反相片选信号线在串行时钟线 SCLK 切换之**后**被切换为有效的低电平。

设置 ***spi_data_on_cs* = 0**，BNEG_NSDI 的第一个数据位将与串行时钟 SCLK 的第一个斜率同时改变。

设置 ***spi_data_on_cs* = 1**，在 BNEG_NSDI 的反相片选信号切换到有效电平的同时，BNEG_NSDI 的第一个数据位被改变。

下表显示了所有四种串行接口模式。

默认情况下，串行时钟线和反相片选信号线之间的延迟时间为 *SER_CLK_IN_HIGH* 或 *SER_CLK_IN_LOW* 设置的时钟周期，这取决于串行时钟的实际电压电平。
这个相应的间隔并不总是与编码器行为完全匹配。因此，串行时钟线和反相片选信号线之间的第一个和最后一个间隔时间可以通过 *SSI_IN_CLK_DELAY* 寄存器 0x57 设置延时的时钟周期。

下面，在所有四个图表中，*SSI_IN_CLK_DELAY* 间隔用红色显示。

支持的 SPI 编码器数据传输模式

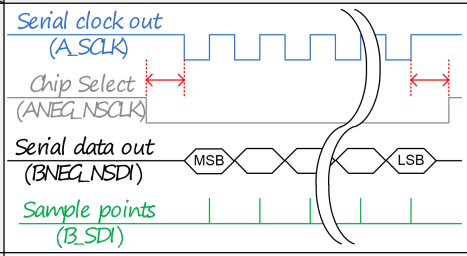
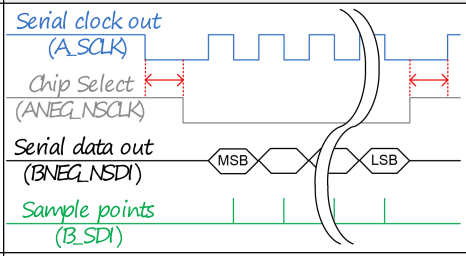
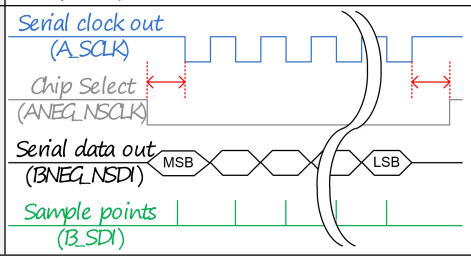
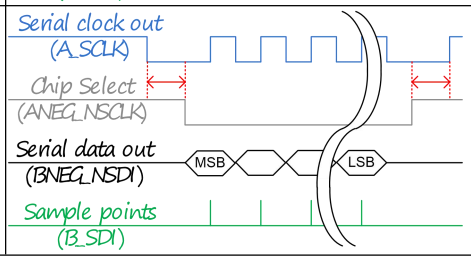
spi_low_before_cs: / spi_data_on_cs	0	1
0		
1		

表 59: 支持的串行接口编码器数据传输模式

11.4.11. 通过 TMC4361A 配置 SPI 编码器

SPI 编码器可以通过 TMC4361A 配置。省去微控制器和编码器之间的连接。

配置 SPI 编码器通信过程

根据设置的传输位数 *SERIAL_ADDR_BITS* 和 *SERIAL_DATA_BITS* 发送请求。

实现 SPI 编码器配置过程，请执行以下操作:

操作步骤:

- 根据 SPI 编码器配置数据报地址位的位数设置 *SERIAL_ADDR_BITS* (*ENC_IN_DATA* 寄存器 0x08)。
- 根据 SPI 编码器配置数据报数据位的位数设置 *SERIAL_DATA_BITS* (*ENC_IN_DATA* 寄存器 0x08)。

结果:

如果发送配置数据到 SPI 编码器，则 *SERIAL_ADDR_BITS* 和 *SERIAL_DATA_BITS* 将按照这个顺序在两个串行接口配置数据报中发送。

因为一直请求编码器数据，所以当配置请求出现时，有必要中断编码器数据请求。需握手保证配置成功。

发送配置数据给串行接口编码器，请执行以下操作:

操作步骤:

- 设置 *DATA_TO_ENC* 寄存器 0x69 为任何值。
- 根据要配置的 SPI 编码器地址设置 *ADDR_TO_ENC* 寄存器 0x68。
- 根据 SPI 编码器数据设置 *DATA_TO_ENC* 寄存器 0x69。

结果:

第一次 *DATA_TO_ENC* 停止不断重复的编码器数据请求。

第二次 *DATA_TO_ENC* 数据访问之后，三个数据报被发送到 SPI 串行接口编码器:

1. 发送了 *ADDR_TO_ENC* 地址数据报。不保存在发送请求时同时接收到的数据。
2. 发送了 *DATA_TO_ENC* 数据数据报。*ADDR_FROM_ENC* 寄存器 0x6A 保存了在发送请求时同时接收到的数据，这是 *ADDR_TO_ENC* 请求的响应。
3. 发送了 NOP 数据报。*DATA_FROM_ENC* 寄存器 0x6B 保存了在发送请求时同时接收到的数据，这是 *DATA_TO_ENC* 请求的响应。

完成配置过程并继续编码器数据请求，请执行以下操作:

- 先读 *ADDR_FROM_ENC* 寄存器 0x6A。
- 将 *ADDR_TO_ENC* 寄存器 0x68 设置为包含角度数据的指定 SPI 接口编码器地址。
- **结束时必须执**行:读 *DATA_FROM_ENC* 寄存器 0x6B。

结果:

读配置请求数据。读 *DATA_FROM_ENC* 数据寄存器后，继续传输编码器数据角度请求数据流。

阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

12. 编码器反馈的可能调节选项

除了简单的反馈监控之外，编码器反馈还可以用于控制运动控制器的输出，使内部实际位置匹配或跟随实际位置 *ENC_POS*。提供了两种选择:PID 控制和闭环操作。

如果编码器直接安装在电机背面精确反应位置数据，则闭环操作是首选的。如果编码器位于驱动侧，电机和驱动侧之间没有固定连接，则最好采用 PID 控制；例如皮带传动。

闭环和 PID 控制			
寄存器名称	**寄存器地址**		**备注**
ENC_IN_CONF	0x07	RW	编码器配置寄存器:闭环配置开关。
CL_TR_TOLERANCE	0x51	R	调节期间触发 TARGET_REACHED 的绝对容许偏差。
ENC_POS_DEV	0x52	R	*XACTUAL* 和 *ENC_POS* 之间的偏差。
闭环和 PID 寄存器集	0x59…5F 0x60…61	W	闭环和 PID 配置参数。
编码器速度配置	0x63	W	编码器速度滤波器配置参数。
编码器速度	0x65 0x66	R	当前编码器速度(带符号)。 当前编码器滤波速度(带符号)。

表 60: 专用的 Closed-Loop 和 PID 寄存器

12.1. 闭环操作

TMC4361A 的闭环单元根据反馈数据直接修改内部步进发生器的输出电流和 S/D 输出。TMC4361 的两相闭环控制采用了不同于磁场定向控制(FOC)的控制方法；FOC 类似于 PID 控制级联。斜坡发生器分配目标和速度，独立于位置控制(换向角度控制)；其也独立于电流控制。闭环操作只能与 256 微步结合使用。

12.1.1. 基本闭环参数

闭环不通过内部步进发生器控制电流。通过内部位置 *XACTUAL* 和外部位置 *ENC_POS* 之间的差值并结合校准的偏移参数 *CL_OFFSET* 来验证 SPI 输出和 Step/Dir 输出处的电流值。

为了正确设置闭环调节的参数和限幅值，请考虑以下细节:

CL_OFFSET 0x59 该寄存器包含校准过程中内部和外部位置之间的基本偏移值，也是闭环操作所必需的，可读可写。如果有预先验证的固定偏移值。则可以直接写该参数。

ENC_POS_DEV 0x52 不断更新的参数 *ENC_POS_DEV* 显示 *XACTUAL* 和 *ENC_POS* 之间的偏差；需结合 *CL_OFFSET*。

CL_BETA 0x1C (8:0) *CL_BETA* 是用来补偿偏差 *ENC_POS_DEV* 的最大换向角。如果偏差达 *CL_BETA* 值，换向角保持稳定在该值，以跟随过载。*CL_MAX* 事件在此时触发。

CL_TOLERANCE 0x5F (7:0) 该参数设置为位置偏差的容差范围。如果| *ENC_POS_DEV* |≤*CL_TOLERANCE*，*CL_FIT_F* lag 标志有效。

如果发生了位置偏差，则触发的 *CL_FIT* 事件表示消除了这种偏差。

CL_DELTA_P 0x5C *CL_DELTA_P* 是比例控制器，用于补偿内部和检测到的外部位置之间的位置偏差。另请参见第 117 页的图 66。

如果|*ENC_POS_DEV*| ≤ *CL_TOLERANCE*, *CL_DELTA_P* 自动设置为 1.0。

如果|*ENC_POS_DEV*| > *CL_TOLERANCE*， TMC4361A 闭环单元将 *CL_DELTA_P* 乘以 *ENC_POS_DEV* 的结果添加到 *ENC_POS*。由此产生维持更高刚度位置的电流换向角，该换相角的最大限幅为 *CL_BETA*。

i *CL_DELTA_P* 由 24 位组成,最后 16 位为小数数。因此，最终的比例项由下式计算: p_{PID} = *CL_DELTA_P* / 65536。

i 因此，p_{PID} 越高，对位置偏差的反应越快

注意:

→ *高 p_{PID} 会导致振荡，需要避免。*

CL_CYCLE 0x63 (31:16) 该参数代表绝对编码器的两个连续调节周期之间的以时钟周期数为单位的延迟时间。建议根据调节周期调整该值；该速率等于或慢于编码器请求速率。如果选择增量 ABN 编码器，该值将自动设置为获取最快的可能调节速率；在大多数情况下是五个时钟周期。

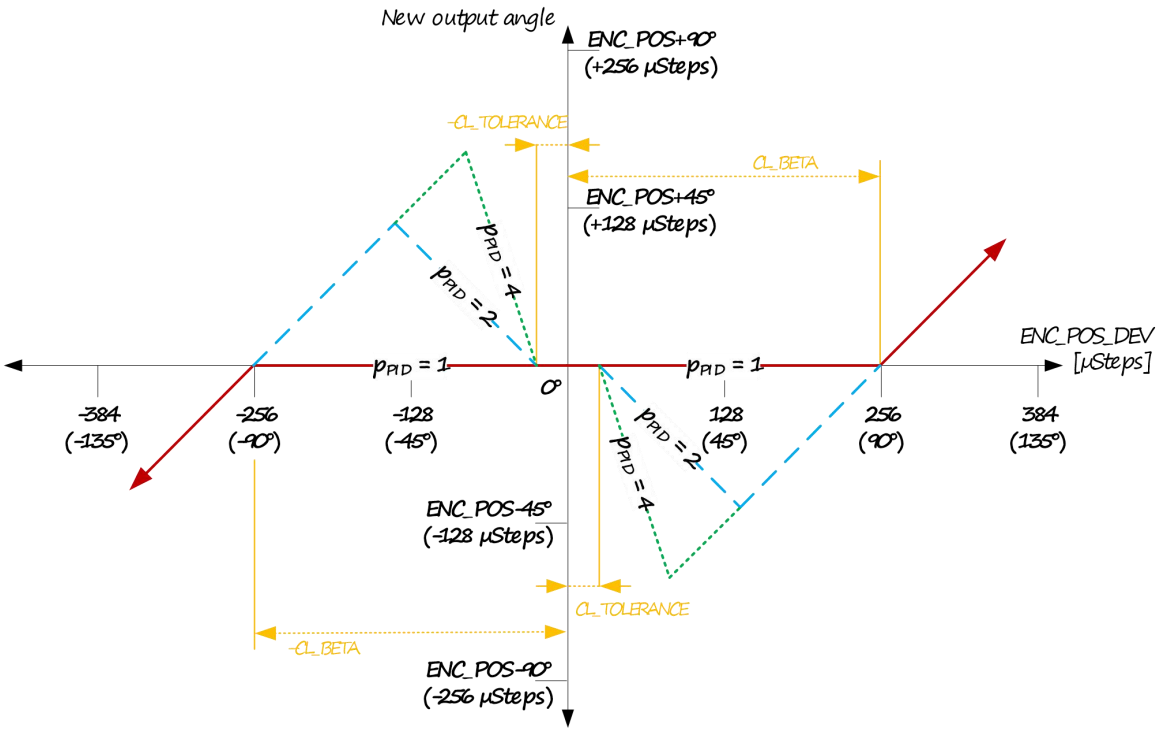


图 66: 用合适的 CL_DELTA_P 计算输出角

12.1.2. 使能并计算闭环操作

i 这里提出基本的校准过程。有关校准过程的详细信息，请参考闭环应用笔记。

如上所述，配置基本闭环控制参数后，就可以使能闭环调节.

为了使能和校准闭环控制，请执行以下操作:

前提条件: 电流设置为最大调节电流值

继续:选项 1A: CL_OFFSET 在使能校准有效位的校准过程中产生

操作步骤:

- 设置 *MSTEPS_PER_FS* = 0 (*STEP_CONF* 寄存器 0x0A) 为 256 细分。
- 运动到全步位置 (*MSCNT* 寄存器 0x79 ϵ {128, 384, 640, 896})。
- 设置 *cl_calibration_en* =1 (*ENC_IN_CONF* 寄存器 0x07)。
- 设置 *regulation_modus* = b'01 (*ENC_IN_CONF* 寄存器 0x07)。
- 等待规定的时间 (系统稳定下来)。
- 设置 *cl_calibration_en* =0 (*ENC_IN_CONF* 寄存器 0x07)。

结果:
基本校准过程使能闭环操作。*CL_OFFSET* 设置为校准过程中的位置偏差。

或者继续:选项 1B: CL_OFFSET 在校准过程中产生，不使能校准开关有效位

操作步骤:

- 设置 *MSTEPS_PER_FS* = 0 (*STEP_CONF* 寄存器 0x0A) - 256 细分。
- 运动到全步位置 (*MSCNT* 寄存器 0x79 ϵ {128, 384, 640, 896})。
- *设置 XACTUAL* 寄存 0x21 等于 *MSCNT*。
- 设置 *regulation_modus* = b'01 (*ENC_IN_CONF* 寄存器 0x07)。

结果:
基本校准过程使能闭环操作。*CL_OFFSET* 设置为校准过程中的位置偏差。

或者继续选项 3: CL_OFFSET 用于校准

如果保存了 *CL_OFFSET*，并且 *ENC_POS* 寄存器 0x50 和 *XACTUAL* 寄存器 0x21 之间的差值没有改变为最初校准值.。

操作步骤:

- 设置 *MSTEPS_PER_FS* = 0 (*STEP_CONF* 寄存器 0x0A) 为 256 细分。
- 设置 *regulation_modus* = b'01 (*ENC_IN_CONF* 寄存器 0x07)。
- 设置 *CL_OFFSET* 为所要的细分值。

结果:
使能闭环操作。

i 如果已经明确回零的位置，比如通过检测 N 事件知道内部和外部位置的情况下，该方法有效。

阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

12.1.3. 限制闭环追赶速度

配置以下参数用于限制补偿电机常规运动被干扰后的追赶速度:

i 参见章节 16.2 第**错误！未定义书签。**页，了解关于最大速度的 PI 调节器的更多信息，因为它使用与 PID 调节器相同的 PI 调节器。基本速度是实际的斜坡速度 *VACTUAL*。

CL_VMAX_CALC_P 0x5A 控制最大速度的 PI 调节器的 p 参数。

CL_VMAX_CALC_I 0x5B 控制最大速度的 PI 调节器的 I 参数.

PID_DV_CLIP 0x5E 设置 *PID_DV_CLIP* 可以避免大的速度变化；也是与最大速度 *VMAX* 的速度偏差限制。

PID_I_CLIP 0x5D 该参数与 *PID_DV_CLIP* 一起使用，以限制误差补偿的速度。误差积分 *PID_ISUM* 由积分项产生。可设置 *PID_I_CLIP* 限制该误差积分。

建议设置 *PID_I_CLIP* 最大值为:

PID_I_CLIP ≤ *PID_DV_CLIP* / *PID_I*。

i 如果没有限制误差积分 *PID_ISUM*，每次计算周期会增加 *PID_I* • *PID_E*，只要电机不跟随，该误差一直存在。

12.1.4. 允许追赶速度限制

如上所述，配置 PI 控制参数和限幅值后可使能追赶速度。

限制闭环追赶速度，请执行以下操作:

操作步骤:

➢ 设置 *cl_vlimit_en* = 1 (*ENC_IN_CONF* 寄存器 0x07)。

结果:

根据设置的参数限制闭环追赶速度。

注意:

→ *如果满足以下条件，电机速度可能高于规定的 VMAX (对于负速度:- VMAX):*

- *使能闭环操作*
- *未使能闭环追赶速度，或在 PID_DV_CLIP> 0 时使能；及 CL_VMAX_CALC_P 和 CL_VMAX_CALC_I 高于 0。*
- *ENC_POS_DEV > CL_TOLERANCE 或 ENC_POS_DEV < CL_TOLERANCE。*

特别注意

如果内部斜坡已经停止，而位置不匹配仍然需要校正，则追赶速度限制的速度基准为零。

校正位置不匹配斜坡是一个线性减速斜坡，独立于指定的斜坡斜率。如上所述出现这种情况是因为调节追赶速度是通过 PI 方法实现。

因此，误差补偿的最终斜坡是由 *ENC_POS_DEV* 和 PI 控制参数共同决定。

翻页了解闭环速度模式的信息。

12.1.5. 启用闭环速度模式

一些应用只需要在闭环行为期间维持所需的速度值，而不考虑位置不匹配。TMC4361A 支持该功能。

注意:

→ *闭环速度模式的设置独立于内部斜坡运行模式(速度或定位模式)。*

为了使能和校准闭环控制，请执行以下操作:

操作步骤:

- 按照第 119 页章节 16.3.3 的详细说明，设置追赶速度参数。
- 设置 *cl_vlimit_en* = 1 (*ENC_IN_CONF* 寄存器 0x07)。
- 设置 *cl_velocity_mode_en* = 1 (*ENC_IN_CONF* 寄存器 0x07)。

结果:

使能闭环运行速度模式。

如果位置不匹配|*ENC_POS_DEV*|超过 768 微步，内部位置计数器 *XACTUA*L 将自动设置为 ENC _ POS ± 768，以限制位置不匹配。

因此，闭环操作保持指定的速度值 *VMAX*。

i 如果 *PID_DV_CLIP* > 0，则速度可能比指定的 *VMAX* 高(负速度:-VMAX)。

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

12.1.6. 闭环电流调节

在闭环运行期间，电流比例可以根据实际负载进行调整节约能源。

使能和配置闭环电流调节

闭环电流调节略微改变了电流调节寄存器的使用，但保持了内部电流调节和与步进驱动器传输的一致性：

1. 闭环电流调节使用与开环配置相同的电流调节寄存器，如章节 11 第 120 页所述。但是对应的参数值和命名不同。
2. MSLUT 电流值的内部调节过程和发送到电机步进驱动器的过程与章节 10 第 87 页解释的完全相同。

配置和使能闭环缩放，请执行以下操作：

操作步骤：

- 设置适合的 *CL_IMIN* (*SCALE_VALUES* 寄存器 0x06)。
- 设置适合的 *CL_IMAX* (*SCALE_VALUES* 寄存器 0x06)。
- 设置适合的 *CL_START_UP* (*SCALE_VALUES* 寄存器 0x06)。
- 设置 *SCALE_VALUES* (31:24)=0。
- 设置 *closed_loop_scale_en* = 1 (*CURRENT_CONF* 寄存器 0x05)。

结果：

一旦使能闭环电流调节，所有其他开环电流缩放选项都会自动禁用。以下电流调节情况是可能的：

1. 如果|*ENC_POS_DEV*| ≤ *CL_START_UP*, 电流值调节值是 *CL_IMIN*。
2. 如果 |*ENC_POS_DEV*| > *CL_START_UP* 及 |*ENC_POS_DEV*| ≤ *CL_BETA*, 电流值从 *CL_IMIN* 到 *CL_IMAX* 线性缩放。
3. 如果 |*ENC_POS_DEV*| > *CL_BETA*, 电流值调节值是 *CL_IMAX*。

下表给出实际的电流调节参数 *SCALE_PARAM*，它取决于上述情况：

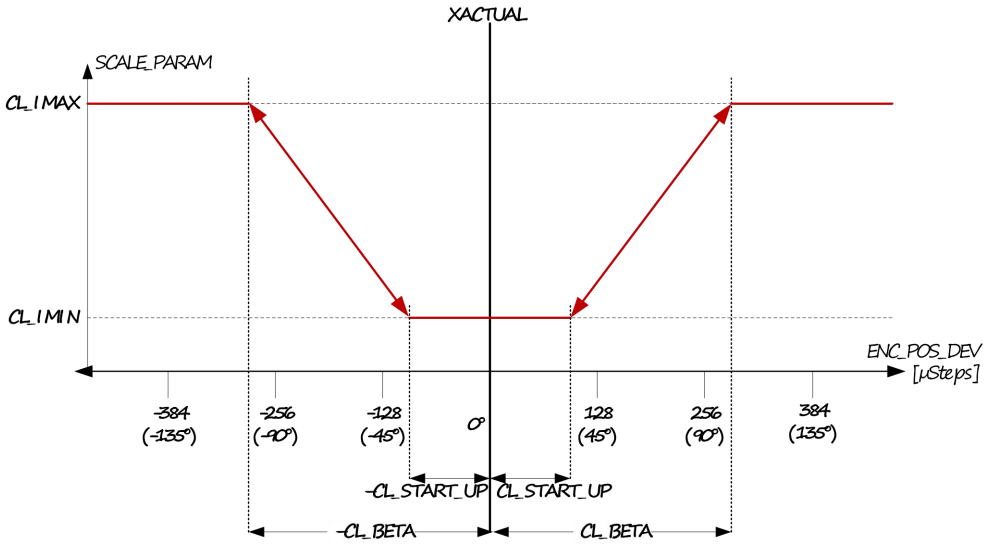


图 67: 闭环电流调节

翻页了解闭环操作期间电流调节过程的信息。

12.1.7. 闭环比例转换过程控制

可配置实现两个电流调节值的平稳切换。两套不同的参数控制闭环电流向更高或更低的调节值调节，如下图所示。

配置从较低运动电流调节值到较高运动电流调节值的平滑过渡，请执行以下操作:

操作步骤:

- 根据延迟周期设置 *CL_UPSCALE_DELAY* 寄存器 0x18，在该延迟周期之后，实际电流调节参数向更高的电流调节值靠近一步。

结果:

每当内部分配一个较高的电流调节值时，实际电流调节值每 *CL_PROCEPTION_DELAY* 时钟周期后增加一步，直到达到分配的参数。

i 如果 *CL_UPSCALE_DELAY*=0, 每当相应的电流调节阶段有效时，立即分配较高的电流调节值。

配置从较高运动电流调节值到较低运动电流调节值的平滑过渡，请执行以下操作:

操作步骤:

- 根据延迟周期设置 *CL_DNSCALE_DELAY* 寄存器 0x19，在该延迟周期之后，实际电流参数向较低的电流调节值靠近一步 。

结果:

每当内部分配一个较低的电流调节标度值时，实际标度参数在每个 *CL_DNSCALE_DELAY* 时钟周期下降一步，直到达到分配的参数。

i 如果 *CL_DNSCALE_DELAY*=0, 每当相应的电流调节阶段有效时，立即分配较低的电流调节值。

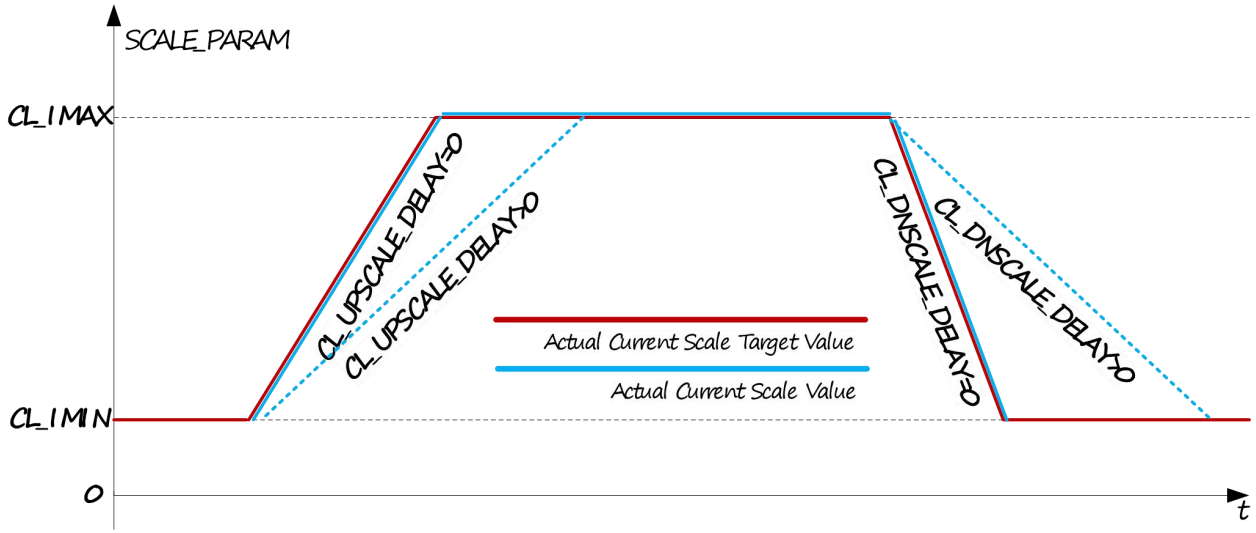


图 68: 闭环电流缩放时序行为

12.1.8. 闭环运行期间的反电动势补偿

达到高速时，电机线圈的电流和电压之间的会出现相移。电流控制可转化为电压控制。

由于线圈电流控制电机，因此必须补偿这种与电机及相关系统设置相关的影响。TMC4361A 通过γ校正实现补偿过程，补偿过程将速度相关角度按照运动方向叠加到当前换向角度。

负载角计算

Gamma 校正在实际换向角度上不断增加一个补偿角度 GAMMA；因为反电动势影响的速度相关量，GAMMA 也是速度相关的。根据实际电机速度 V_ENC(寄存器 0x65)设置速度限制。电机速度是内部计算并经过（*V_ENC_MEAN* 寄存器 0x66) 滤波处理，用来平滑γ校正，将在下一节中解释。

配置和使能闭环操作期间的反电动势补偿，请执行以下操作:

操作步骤:

- 设置合适的 *CL_GAMMA* 寄存器 0x1C。
- 设置合适的 *CL_VMIN_EMF* 寄存器 0x60。
- 设置合适的 *CL_VADD_EMF* 寄存器 0x61。
- 设置 *cl_emf_en* = 1 (*ENC_IN_CONF* 寄存器 0x07)。

结果:

使能闭环操作期间的反电动势补偿。CL_GAMMA 代表 GAMMA 的最大值。默认情况下，CL_GAMMA 设置为其最大可能值 255，代表 90 度角。

以下补偿情况是可能的:

1. 如果|*V_ENC_MEAN*| ≤ *CL_VMIN_EMF*, GAMMA 设置为 0。
2. 如果|*V_ENC_MEAN*| > *CL_VMIN_EMF* 及 |*V_ENC_MEAN*| ≤ (*CL_VMIN_EMF* + *CL_VADD_EMF*), GAMMA 在 0 和它的最大值之间线性缩放。
3. 如果|*V_ENC_MEAN*| > (*CL_VMIN_EMF* + *CL_VADD_EMF*), GAMMA = *CL_GAMMA*。

下表确定了参数 GAMMA，这取决于上述情况:

特别注意

如果使能γ校正，最大可能换向为(CL_BETA + CL_GAMMA)。
该值不得超过 180°(每全步 256 微步下的 511 微步)，因为 180°或更大的角度将导致不必要的运动方向变化。

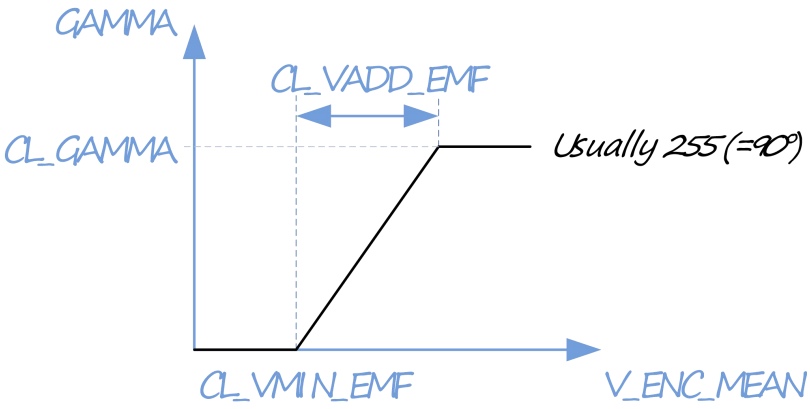


图 69: 负载角的 GAMMA 计算

12.1.9. 编码器速度读出参数

电机实际速度可通过编码器读回，实际编码器速度会在抖动，这是系统固有的。TMC4361A 含有滤波器用于计算反电动势补偿。可以读出以下速度参数。

V_ENC 0x65 实际编码器速度，单位为每秒脉冲(微步)[pps]。

V_ENC_MEAN 0x66 实际编码器滤波后的速度，单位为每秒钟脉冲(微步)的 [pps]。

12.1.10. 编码器速度滤波器配置

为了正确设置滤波器参数，请考虑以下细节:

ENC_VMEAN_WAIT 0x63 (7:0) *ENC_VMEAN_WAIT* 是两个连续 *V_ENC* 的以时钟周期数为单位的采样延迟周期，用于编码器滤波器速度计算。该值越低，*V_ENC_MEAN* 的调节过程越快。相应地: *V_ENC_MEAN* 的梯度越高。

如果连接了增量 ABN 编码器，则 *ENC_VMEAN_WAIT* 必须将设置大于 32。

如果连接了绝对编码器，*ENC_VMEAN_WAIT* 将自动设置为 *SER_PTIME*。

ENC_VMEAN_FILTER 0x63 (11:8) 该滤波器指数用于滤波计算。值越低，*V_ENC_MEAN* 的调节过程越快。相应地: *V_ENC_MEAN* 的梯度越高。每 *ENC_VMEAN_WAIT* 时钟周期，编码器滤波速度按公式更新

$$V_{ENC_{MEAN}} = \mathrm{V_{ENC_{MEAN}}} - \frac{\mathrm{V_{ENC_{MEAN}}}}{2^{ENC\_VMEAN\_FILTER}} + \frac{V_{ENC}}{2^{ENC\_VMEAN\_FILTER}}$$

ENC_VMEAN_INT 0x63 (31:16) 较高速度编码器 *V_ENC* 的刷新频率由该编码器速度更新周期决定。

如果连接了增量 ABN 编码器，*ENC_VMEAN_INT* 的最小值将自动设置为 256。

如果连接了绝对编码器，*ENC_VMEAN_INT* 将自动调节到编码器数据请求速率。

12.1.11. 编码器速度等于 0 事件

V_ENC 低速的计算是由 AB 信号变化触发的，而不是由 *ENC_VMEAN_INT* 定义的刷新频率触发的，所以编码器的任何空闲状态都不会被识别。

可以设置时钟周期数确定 *V_ENC* = 0，在该时间长度内不发生 AB 信号变化则编码器为空闲状态。

要使能编码器空闲状态，请执行以下操作:

操作步骤:

- 设置适合的 *ENC_VEL_ZERO* 寄存器 0x62。

结果:

如果在 *ENC_VEL_ZERO* 时钟周期内没有出现 AB 信号变化，则 *ENC_VEL_Zero* 事件被触发，指示编码器为空闲状态。

13. 复位和时钟门控

除了硬件复位引脚 NRST 和内部自动上电复位处理，TMC4361A 还提供软件复位选项。如果芯片不工作，时钟门控可以用来降低功耗。

复位和时钟引脚		
复位名称	**类型**	**备注**
NRST	输入	硬件复位信号，低有效。
STPIN	输入	唤醒激活信号，高有效。
CLK_EXT	输入	连接的外部时钟信号。

表 61: 专用复位和时钟引脚

复位和时钟门控寄存器			
寄存器 名称	**寄存器 地址**		**备注**
GENERAL_CONF	0x00	RW	位 18:17
CLK_GATING_DELAY	0x14	RW	时钟门控使能前的延迟时间。
CLK_GATING_REG	0x4F (2:0)	RW	时钟门控触发器.
RESET_REG	0x4F (31:8)	RW	SW-Reset 软件复位触发器

表 62: 专用复位和时钟门控寄存器

技术规格书

14. 完整的寄存器和开关列表

14.1. 通用配置寄存器 GENERAL_CONF 0x00

GENERAL_CONF 0x00 *(缺省: 0x00006020)*			
R/W	**位**	**值**	**备注**
RW	0	*use_astart_and_vstart*	***(仅适用于 S 形斜坡)***
		0	如果 *VSTART* ≠ 0，斜坡开始时设置 *AACTUAL* = *AMAX* 或–*AMAX*。
		1	如果 *VSTART* ≠ 0，斜坡开始时设置 *AACTUAL* = *ASTART* 或 –*ASTART*。
	1	*direct_acc_val_en*	
		0	加速度寄存器参数除以 *CLK_FREQ* 得到加速度值。
		1	加速度值直接设置为每个时钟周期的步数。
	2	*direct_bow_val_en*	
		0	弓行值寄存器参数除以 *CLK_FREQ* 得到弓形值。
		1	弓值直接设置为每个时钟周期的步数。
	3	*step_inactive_pol*	
		0	STPOUT = 1 对应有效的步进信号。
		1	STPOUT = 0 对应有效的步进信号。
	4	*toggle_step*	
		0	STPOUT 从非有效极性到有效极性的转换对应一个步进信号。
		1	STPOUT 的每次电平变化都表示一个步进信号。
	5	*pol_dir_out*	

GENERAL_CONF 0x00 *(缺省: 0x00006020)*			
R/W	**位**	**值**	**备注**
		0	DIROUT = 0 表示电机反向。
		1	DIROUT = 1 表示电机反向。
	7:6	*sdin_mode*	
		0	内部步进控制(将使用内部斜坡发生器)。
		1	STPIN / DIRIN 接口控制步进和方向，**S**TPIN 高电平对应有效的步进信号。
		2	STPIN / DIRIN 接口控制步进和方向，STPIN 低电平对应有效的步进信号。
		3	STPIN / DIRIN 接口控制步进和方向， STPIN 电平翻转对应有效的步进信号。
	8	*pol_dir_in*	
		0	DIRIN = 0 表示电机反向。
		1	DIRIN = 1 表示电机反向。
	9	*sd_indirect_control*	
		0	STPIN/DIRIN 输入信号将直接影响 *XACTUAL*。
		1	STPIN/DIRIN 输入信号将影响 *XTARGET* 寄存器值，内部斜坡发生器工作。
	ꕥ下页继续.		
RW	11:10	*serial_enc_in_mode*	
		0	编码器接口连接增量编码器。
		1	编码器接口连接绝对 SSI 编码器。
		2	保留
		3	编码器接口连接绝对 SPI 编码器。
	12	*diff_enc_in_disable*	
		0	使能编码器差分输入接口。
		1	禁用编码器差分输入接口(SPI 编码器自动设置为差分接口禁用)。
	14:13	*stdby_clk_pin_assignment*	
		0	STDBY_CLK 引脚输出 Standby 信号，低有效。
		1	STDBY_CLK 引脚输出 Standby 信号，高有效。
		2	STDBY_CLK 输出 ChopSync 时钟。 (仅 TMC23x, TMC24x 步进电机驱动器)。
		3	STDBY_CLK 输出内部时钟。
	15	*intr_pol*	
		0	INTR=0 表示中断发生。
		1	INTR=1 表示中断发生。
	16	*invert_pol_target_reached*	
		0	TARGET_REACHED 信号为 1 表示目标位置到达。
		1	TARGET_REACHED 信号为 0 表示目标位置到达。
	17	*clk_gating_en*	

GENERAL_CONF 0x00 *(缺省: 0x00006020)*			
R/W	**位**	**值**	**备注**
		0	禁用时钟门控。
		1	内部时钟使能。
	18	*clk_gating_stdby_en*	
		0	待机阶段不产生时钟信号。
		1	待机阶段的内部时钟使能。
	19	*fs_en*	
		0	禁用全步切换。
		1	如果 |*VACTUAL*| > *FS_VEL*，则 SPI 输出切换到全步。
	20	*fs_sdout*	
		0	不使能 Step/Dir 的全步切换输出。
		1	如果全步有效，Step/Dir 输出全步信号 。
∀下页继续			
RW	22:21	*dcstep_mode*	
		0	不使能 dcStep。
		1	自动选择生成 dcStep 信号。
		2	外部产生 STEP_READY 信号的 dcSTEP(TMC 21x 0)。
		3	内部产生 STEP_READY 信号的 dcSTEP(TMC 26x)。 **i TMC26x 配置: 设置 const_toff-Chopper (CHM = 1); 仅慢衰减 (HSTRRT = 0); TST = 1 和 SGT0=SGT1=1 (on_state_xy).**
	23	*pwm_out_en*	
		0	禁用 PWM 输出。STPOUT/DIROUT 引脚为 Step/Dir 输出。
		1	STPOUT/DIROUT 引脚为 PWM 输出(PWMA/PWMB)。
	24	*serial_enc_out_enable*	
		0	SPI 输出没有连接编码器。
		1	SPI 输出连接绝对 SSI 编码器， 传输 SSI 编码器接口数据的。
	25	*serial_enc_out_diff_disable*	
		0	使能差分串行编码器输出。
		1	禁用差分串行编码器输出。
	26	*automatic_direct_sdin_switch_off*	
		0	关闭外部步进直接控制后，*VACTUAL*=0 & *AACTUAL*=0 。
		1	关闭外部步进直接控制后 ， *VACTUAL* = *VSTART* 和 *AACTUAL* = *ASTART* 。
	27	*circular_cnt_as_xlatch*	

GENERAL_CONF 0x00 *(缺省: 0x00006020)*			
R/W	**位**	**值**	**备注**
		0	寄存器 0x36 对应寄存器 *X_LATCH*。
		1	寄存器 0x36 对应寄存器 *REV_CNT* (内部分辨率)。
	28	*reverse_motor_dir*	
		0	内部 SinLUT 方向为正。
		1	内部 SinLUT 方向为负。
	29	*intr_tr_pu_pd_en*	
		0	INTR 和 TARGET_REACHED 强驱输出。.
		1	INTR 和 TARGET_REACHED 为具有门控上拉和/或下拉功能的输出。
	30	*intr_as_wired_and*	
		0	如果 *intr_tr_pu_pd_en* = 1，INTR 输出线或。
		1	如果 *intr_tr_pu_pd_en* = 1，INTR 输出线与。
	31	*tr_as_wired_and*	
		0	如果 *intr_tr_pu_pd_en* = 1，TARGET _ REACHED 输出线或。
		1	如果 *intr_tr_pu_pd_en* = 1，TARGET _ REACHED 输出线与。

表 65: 通用配置 0x00

 阅读整个文档：是第 22 章(第 174 页)中的补充指令。

14.2. 参考开关配置寄存器 REFERENCE_CONF 0x01

REFERENCE_CONF 0x01 *(缺省: 0x00000000)*			
R/W	**位**	**值**	**备注**
RW	0	*stop_left_en*	
		0	禁用 STOPL 信号。
		1	使能 STOPL 信号。
	1	*stop_right_en*	
		0	禁用 STOPR 信号。
		1	使能 STOPR 信号。
	2	*pol_stop_left*	
		0	STOPL 输入信号低有效。
		1	STOPL 输入信号高有效。
	3	*pol_stop_right*	
		0	STOPR 输入信号低有效。
		1	STOPR 输入信号高有效。
	4	*invert_stop_direction*	
		0	STOPL/STOPR 对应反/正方向停止电机。
		1	STOPL/STOPR 对应正/反方向停止电机。
	5	*soft_stop_en*	
		0	硬停使能。在任何外部停止事件中，*VACTUAL* 立即设置为 0。
		1	软停使能。*VACTUAL* 降至 v = 0 过程为线性速度斜坡。
	6	*virtual_left_limit_en*	
		0	禁用位置限制 VIRT_STOP_LEFT。
		1	使能位置限制 VIRT_STOP_LEFT。
	7	*virtual_right_limit_en*	
		0	禁用位置限制 VIRT_STOP_RIGHT。
		1	使能位置限制 VIRT_STOP_RIGHT。
	9:8	*virt_stop_mode*	
		0	保留.
		1	硬停:在虚拟停止事件中，*VACTUAL* 设置为 0。
		2	软停使能线性速度斜坡(从 *VACTUAL* 到 v = 0)。
		3	保留.
	10	*latch_x_on_inactive_l*	
		0	如果 STOPL 变为非有效电平，*XACTUAL* 不锁存。
		1	如果 STOPL 变为非有效电平，*X_LATCH = XACTUAL*。
	11	*latch_x_on_active_l*	
		0	如果 STOPL 变为有效电平，*XACTUAL* 不锁存。
		1	如果 STOPL 变为有效电平，*X_LATCH = XACTUAL* 。

∀下一页继续。

REFERENCE_CONF 0x01 *(缺省: 0x00000000)*			
R/W	**位**	**值**	**备注**
RW	12	*latch_x_on_inactive_r*	
		0	如果 STOPR 变为非有效电平，XACTUAL 不锁存。
		1	如果 STOPR 变为非有效电平，X_LATCH = XACTUAL 。
	13	*latch_x_on_active_r*	
		0	如果 STOPR 变为有效电平，XACTUAL 不锁存。
		1	如果 STOPR 变为有效电平，X_LATCH = XACTUAL 。
	14	*stop_left_is_home*	
		0	STOPL 输入信号不是 HOME 回零信号。
		1	STOPL 输入信号也是 HOME 回零信号。
	15	*stop_right_is_home*	
		0	STOPR 输入信号不是 HOME 回零信号。
		1	STOPR 输入信号也是 HOME 回零信号。
	19:16	*home_event*	
		0	ABN 编码器信号的下一个有效 N 事件对应 HOME 回零信号。
		2	HOME_REF = 1 表示有效的 HOME 回零事件。 *X_HOME* 位于有效活动范围的上升沿。
		3	HOME_REF = 0 表示 HOME 回零位置的负区域/位置。
		4	HOME_REF = 1 表示有效的 HOME 回零事件。 *X_HOME* 位于有效活动范围的下降沿。
		6	HOME_REF = 1 表示有效的 HOME 回零事件。 *X_HOME* 位于有效活动范围的中间。
		9	HOME_REF = 0 表示有效的 HOME 回零事件。 *X_HOME* 位于有效活动范围的中间。
		11	HOME_REF = 0 表示有效的 HOME 回零事件。 *X_HOME* 位于有效活动范围的上升沿。
		12	HOME_REF = 1 指示 HOME 回零位置的负区域/位置。
		13	HOME_REF = 0 表示有效的 HOME 回零事件。 *X_HOME* 位于有效活动范围的下降沿。
	20	*start_home_tracking*	
		0	经过 HOME 回零位置，不保存 *X_HOME*。
		1	当下一个回零 HOME 事件发生时， *X_HOME= XACTUA*L。 *XLATCH_DONE* 事件有效。 如果事件被清除，*start_home_tracking* 自动复位。
	21	*clr_pos_at_target*	
		0	如果定位模式激活，斜坡在 *XTARGET* 处停止。
		1	达到 *XTARGET* 后，将 *XACTUAL* 设置为 0。 下一个坡道立即开始。
	22	*circular_movement_en*	
		0	*XACTUAL* 的范围不受限制: $-2^{31} \le XACTUAL \le 2^{31}-1$
		1	*XACTUAL* 的范围受 X_RANGE 限制: $-X\_RANGE \le XACTUAL \le X\_RANGE - 1$
			下一页继续。

REFERENCE_CONF 0x01 *(缺省: 0x00000000)*			
R/W	**位**	**值**	**备注**
RW	24:23	*pos_comp_output*	
		0	*TARGET_REACHED_Flag* 有效时 *TARGET_REACHED* 有效。
		1	*VELOCITY_REACHED_*Flag 有效时 *TARGET_REACHED* 有效。
		2	*ENC_FAIL* 有效时 *TARGET_REACHED* 有效。
		3	*POSCOMP_REACHED_*Flag 有效时触发 *TARGET_REACHED*。
	25	*pos_comp_source*	
		0	*POS_COMP* 与 *XACTUAL* 比较。
		1	*POS_COMP* 与外部位置 *ENC_POS* 比较。
	26	*stop_on_stall*	
		0	如果发生堵转事件，SPI 和 S/D 输出接口将保持活动状态。
		1	如果发生堵转事件（硬停），SPI 和 S/D 输出控制电机运动停止。
	27	*drv_after_stall*	
		0	在发生有效堵转停止事件的情况下，电机不再继续动作。
		1	在发生堵转停止事件且 stop-on-stall 复位后，电机可以继续运动。
	29:28	*modified_pos_compare:* *POS_COMP_REACHED_F* 及 事件根据比较 *XACTUAL* 或 *ENC_POS* 与以下的比较结果	
		0	*POS_COMP*
		1	*X_HOME*
		2	*X_LATCH* 或 *ENC_LATCH*
		3	*REV_CNT*
	30	*automatic_cover*	
		0	SPI 输出接口不会自动传输任何覆盖数据报。
		1	当 *VACTUAL* 经过 *SPI_SWITCH_VEL* 开关时，SPI 输出接口自动发送覆盖数据报。
	31	*circular_enc_en*	
		0	*ENC_POS* 不受限: $-2^{31} \le ENC\_POS \le 2^{31}-1$
		1	*ENC_POS* 范围受 *X_RANGE* 限制: $-X\_RANGE \le ENC\_POS \le X\_RANGE -1$

表 66: 参考开关配置 0x01

14.3. START 启动开关配置寄存器 START_CONF 0x02

START_CONF 0x02 *(缺省值: 0x00000000)*			
R/W	**位**	**值**	**备注**
RW	4:0	*start_en*	
		xxxx1	XTARGET 值的更改需要启动信号。
		xxx1x	*VMAX* 值的更改需要启动信号。
		xx1xx	*RAMPMODE* 值的更改需要启动信号。
		x1xxx	*GEAR_RATIO* 值的更改需要启动信号。
		1xxxx	阴影寄存器功能集已启用。
	8:5	*trigger_events*	
		0000	因为不使能启动信号，因此相关触发时序无效。
		xxx0	配置 START 为输出引脚。
		xxx1	外部 START 启动信号触发内部定时器。START 为输入引脚。
		xx1x	*TARGET_REACHED* 事件为启动信号的触发源。
		x1xx	*VELOCITY_REACHED* 事件为启动信号的触发源。
		1xxx	*POSCOMP_REACHED* 事件为启动信号的触发源。
	9	*pol_start_signal*	
		0	START 引脚低有效 (输入或输出)。
		1	START 引脚高有效 （输入或输出)。
	10	*immediate_start_in*	
		0	有效的 START 输入信号启动内部启动定时器。
		1	有效的 START 输入信号立即执行。
	11	*busy_state_en*	
		0	START 为输入或者输出
		1	使能 START 信号的忙碌态。如果 START 引脚为输入信号，为弱有效极性信号；如果 START 引脚为输出信号，为无效极性的强驱输出。
	15:12	*pipeline_en*	
		0000	不使能流水线。
		xxx1	*X_TARGET* 的流水线操作。
		xx1x	*POS_COMP* 的流水线操作。
		x1xx	*GEAR_RATIO* 的流水线操作。
		1xxx	*GENERAL_CONF* 的流水线操作。
	17:16	*shadow_option*	
		0	13 个相关斜坡参数的单级阴影寄存器。
		1	S 形斜坡的两级阴影寄存器。
		2	梯形斜坡的两级阴影寄存器(除了 *VSTOP*)。
		3	梯形斜坡的两级阴影寄存器(除了 *VSTART*)。

∀下一页继续。

START_CONF 0x02 *(缺省值: 0x00000000)*			
R/W	**位**	**值**	**备注**
RW	18	*cyclic_shadow_regs*	
		0	当前斜坡参数不写回阴影寄存器。
		1	当前斜坡参数写回阴影寄存器。
	19	保留，设置为 0	
	23:20	*SHADOW_MISS_CNT*	
		U	两次连续的阴影寄存器传输之间忽略的内部启动信号的数量。
	31:24	*XPIPE_REWRITE_REG*	
			如果寄存器位为“1”，则当前 *START_CONF*(15:12)寄存器中被分配的流水线寄存器在内部启动信号产生时将当前的流水线寄存器的参数写回 *X_PIPEx*: *XPIPE_REWRITE_REG*(0) → *X_PIPE0* *XPIPE_REWRITE_REG*(1) → *X_PIPE1* *XPIPE_REWRITE_REG*(2) → *X_PIPE2* *XPIPE_REWRITE_REG*(3) → *X_PIPE3* *XPIPE_REWRITE_REG*(4) → *X_PIPE4* *XPIPE_REWRITE_REG*(5) → *X_PIPE5* *XPIPE_REWRITE_REG*(6) → *X_PIPE6* *XPIPE_REWRITE_REG*(7) → *X_PIPE7* 示例： *START_CONF*(15:12) = b’0011。 *START_CONF*(31:24) = b’01000010。 如果产生内部启动信号，则将 *X_TARGET* 的值写回 *X_PIPE1*，而将 *POS_COMP* 的值写回 *X_PIPE6*。

表 67: Start 启动切换配置 START_CONF 0x02

14.4. 输入滤波器配置寄存器 INPUT_FILT_CONF 0x03

INPUT_FILT_CONF 0x03 *(Default value: 0x00000000)*			
R/W	**位**	**值**	**备注**
RW	2:0	*SR_ENC_IN*	
		U	以下引脚输入频率 = f_{clk} / $2^{SR\_ENC\_IN}$: A_SCLK, ANEG_NSCLK, B_SDI, BNEG_NSDI, N, NNEG
	3	保留，设置为 0	
	6:4	*FILT_L_ENC_IN*	
		U	以下引脚的滤波长度: A_SCLK, ANEG_NSCLK, B_SDI, BNEG_NSDI, N, NNEG。在采样输入点采样 FILT_L_ENC_IN 次，电平必须相等才能提供有效输入位。
	7	*SD_FILT0*	
		0	S/D 输入引脚(STPIN/DIRIN) 未被分配到 ENC_IN 输入滤波组。
		1	S/D 输入引脚(STPIN/DIRIN) 被分配到 ENC_IN 输入滤波组。
	10:8	*SR_REF*	
		U	STOPL, HOME_REF, STOPL 引脚输入频率= f_{clk} / 2^{REF}
	11	保留，设置为 0	
	14:12	*FILT_L_REF*	
		U	STOPL, HOME_REF, STOPL 引脚的滤波长度；在采样输入点采样 FILT_L_REF 次，电平必须相等才能提供有效输入位。
	15	*SD_FILT1*	
		0	S/D 输入引脚 (STPIN/DIRIN) 没有被分配到 REF 输入滤波组。
		1	S/D 输入引脚 (STPIN/DIRIN) 被分配到 REF 输入滤波组。
	18:16	*SR_S*	
		U	START 脚的输入采样频率= f_{clk} / 2^{S} 。
	19	保留，设置为 0	
	22:20	*FILT_L_S*	
		U	START 引脚的滤波器长度。在采样输入点采样 FILT_L_S 次，电平必须相等才能提供有效输入位。
	23	*SD_FILT2*	
		0	S/D 输入引脚(STPIN/DIRIN) 没有被分配到 S 输入滤波组。
		1	S/D 输入引脚(STPIN/DIRIN) 被分配到 S 输入滤波组。
	26:24	*SR_ENC_OUT*	
		U	SDODRV_SCLK, SDIDRV_NSCLK 引脚的输入采样频率= f_{clk} / $2^{SR\_ENC\_OUT}$
	27	保留，设置为 0	
	30:28	*FILT_L_ENC_OUT*	
		U	SDODRV_SCLK, SDIDRV_NSCLK 引脚的滤波器长度，在采样输入点采样 FILT_L_ENC_OUT 次，电平必须相等才能提供有效输入位。
	31	*SD_FILT3*	
		0	S/D 输入引脚(STPIN/DIRIN) 没有被分配到 ENC_OUT 输入滤波组。.
		1	S/D 输入引脚(STPIN/DIRIN) 被分配到 ENC_OUT 输入滤波组。

表 68: 输入滤波配置寄存器 INPUT_FILT_CONF 0x03

14.5. SPI 输出配置寄存器 SPI_OUT_CONF 0x04

SPI_OUT_CONF 0x04 *(缺省: 0x00000000)*			
R/W	**位**	**值**	**备注**
RW	3:0	*spi_output_format*	
		0	**不使能** SPI 输出接口。
		1	SPI 输出接口连 **SPI-DAC**，SPI 输出值与电压匹配: Current=0 → VCC/2 Current=-max → 0 Current=max → VCC
		2	SPI 输出接口连 **SPI-DAC**。SPI 输出值是**绝对值**。 STPOUT 输出线圈 A 的相位，DIROUT 输出线圈 B 的相位。 相位= 0:正值。
		3	SPI 输出接口连 **SPI-DAC**。SPI 输出值是**绝对值**。 STPOUT 输出线圈 A 的相位，DIROUT 输出线圈 B 的相位。 相位= 0:负值。
		4	SPI 输出接口发送**实际无符号电流调节因子**。
		5	SPI 输出接口发送实际有符号的电流值 CURRENTA 和 CURRENTB。
		6	SPI 输出接口与 **SPI-DAC** 相连。 实际无符号电流调节因子与 DAC_ADDR_A 值合并为输出数据报。
		8	SPI 输出接口连 **TMC23x** 步进电机驱动器。
		9	SPI 输出接口连 **TMC24x** 步进电机驱动器。
		10	SPI 输出接口连 **TMC26x/389** 步进电机驱动器。 发送配置和传输电流数据到步进电机驱动器。
		11	SPI 输出接口连 **TMC26x** 步进电机驱动器。 给步进电机驱动器只发送配置数据。S/D 输出接口产生步进信号。
		12	SPI 输出接口连 **TMC2130 / TMC2160** 步进电机驱动器。 给步进电机驱动器只发送配置数据。S/D 输出接口产生步进信号。
		13	SPI 输出接口连 **TMC2130 / TMC2160** 步进电机驱动器。 发送配置和电流数据到步进电机驱动器。
		15	SPI 输出接口仅发送**覆盖**数据报。
	19:13	*COVER_DATA_LENGTH*	
		U	完整覆盖数据报长度的位数。最大值= 64。 如果选择 TMC 步进电机驱动器，设置为 0。芯片自动选择数据报长度。
	23:20	*SPI_OUT_LOW_TIME*	
		U	SPI 输出时钟低电平的时钟周期数。
	27:24	*SPI_OUT_HIGH_TIME*	
		U	SPI 输出时钟高电平的时钟周期数。
	31:28	*SPI_OUT_BLOCK_TIME*	
		U	SPI 串行接口输出传输后，NSCSDRV 输出保持高电平(无效)的时钟周期数。

ꕥ下一页继续。

SPI_OUT_CONF 0x04 *(缺省: 0x00000000)*			
R/W	**位**	**值**	**备注**
	4	*three_phase_stepper_en*	***(仅 TMC389)***
		0	SPI 输出连两相步进电机驱动器(TMC26x)。
		1	SPI 输出连三相步进电机驱动器(TMC389)。
	5	*scale_val_transfer_en*	***(仅 TMC26x/2130 的 SD 模式)***
		0	不传输电流调节值。
		1	将电流调节值传输到适当的驱动器寄存器。
	6	*disable_polling*	***(仅 TMC26x/2130 的 SD 模式)***
		0	一直查询驱动器状态。
		1	不查询数据。
	7	*autorepeat_cover_en*	***(仅 TMC26x/2130)***
		0	不支持自动连续传输覆盖数据报流。
		1	支持自动连续传输覆盖数据报流。
	11:8	*POLL_BLOCK_EXP*	***(仅 TMC26x 的 SD 模式, 仅 TMC21x0)***
		U	乘法因子用于计算连续查询两个数据报的时间间隔: t_{POLL} = 2^*POLL_BLOCK_EXP* · *SPI_OUT_BLOCK_TIME* / f_{CLK}
	12	*cover_done_only_for_cover*	***(仅 TMC26x/21x0)***
		0	为每个发送到电机驱动器的数据报设置 *COVER_DONE* 事件。
		1	*COVER_DONE* 事件仅针对发送到电机驱动器的覆盖数据报。
		∞∀接下一页	
RW	4	*sck_low_before_csn*	***(不是 TMC 驱动器)***
		0	在 SCKDRV_NSDO 之前 NSCSDRV_SDO 被拉低，开始新的数据传输。
		1	在 NSCSDRV_SDO 之前 SCKDRV_NSDO 被拉低，开始新的数据传输
	5	*new_out_bit_at_rise*	***(不是 TMC 驱动器)***
		0	在 SDODRV_SCLK 的下降沿，采样 SCKDRV_NSDO 数据。
		1	在 SDODRV_SCLK 的上升沿，采样 SCKDRV_NSDO 数据。
	11:7	*DAC_CMD_LENGTH*	***(仅 SPI-DAC)***
		U	命令地址的位数。
	12	保留，设置为 0	
	23:4	*SSI_OUT_MTIME*	***(仅串行编码器输出)***
		U	SSI 输出接口的单稳态时间: 在最后一次主请求后编码器数据保持稳定的延迟时间[时钟周期]。

表 69: SPI 输出配置寄存器 SPI_OUT_CONF 0x04

14.6. 电流调节配置寄存器 CURRENT_CONF 0x05

CURRENT_CONF 0x05 *(缺省: 0x00000000)*			
R/W	**位**	**值**	备注
RW	0	*hold_current_scale_en*	
		0	静止阶段不调节保持电流。
		1	静止阶段使能保持电流的调节。
	1	*drive_current_scale_en*	
		0	运动期间不调节驱动电流。
		1	运动期间使能驱动电流的调节。
	2	*boost_current_on_acc_en*	
		0	减速斜坡不调节上升电流。
		1	如果 *RAMP_STATE* = b'01 (加速度斜坡)，使能上升电流的调节。
	3	*boost_current_on_dec_en*	
		0	减速斜坡不调节上升电流。
		1	如果 *RAMP_STATE* = b'10 (减速度斜坡)，使能上升电流的调节。
	4	*boost_current_after_start_en*	
		0	斜坡开始时无上升电流。
		1	如果 *VACTUAL* = 0 且新斜坡开始，则为临时上升电流。
	5	*sec_drive_current_scale_en*	
		0	整个运动斜坡只采用一个驱动电流值。
		1	当 *VACTUAL* > *VDRV_SCALE_LIMIT* 时，使能第二个驱动电流的调节。
	6	*freewheeling_en*	
		0	不使能飞轮。
		1	待机阶段后进入飞轮阶段。
	7	*closed_loop_scale_en*	
		0	不调节闭环电流。
		1	使能闭环电流 *CURRENT_CONF*(6:0) = 0 的自动调节 **！ 闭环校准中禁用该调节！**
	8	*pwm_scale_en*	
		0	不使能 PWM 调节。
		1	使能 PWM 调节。
	15:9	保留，设置为 0	
	31:16	*PWM_AMPL*	
		U	电压 PWM 模式下，*VACTUAL* = 0 时 PWM 幅度。 **i 最大占空比 = (0.5 + (*PWM_AMPL* + 1) / 2^{17})** **最小占空比 = (0.5 – (*PWM_AMPL* + 1) / 2^{17})** **在 *VACTUAL* = *PWM_VMAX* 时，PWM_AMPL = 2^{16} – 1**

表 70: 电流缩放配置 (0x05)

14.7. 电流调节寄存器 SCALE_VALUES 0x06

SCALE_VALUES 0x06 *(缺省: 0xFFFFFFFF)*				
R/W	**位**	**值**	**电流调节值**	**Remarks**
RW	7:0	U	*BOOST_SCALE_VAL*	**开环上升电流调节值**.
			CL_IMIN	**闭环**最小电流调节值。
	15:8	U	*DRV1_SCALE_VAL*	**开环**第一驱动电流调节值。
			CL_IMAX	**闭环**最大电流调节值。
	23:16	U	*DRV2_SCALE_VAL*	**开环**第二驱动电流调节值。
			CL_START_UP	|*ENC_POS_DEV*|大于该参数时，闭环从 *CL_IMIN* 开始增加电流值的。
	31:24	U	*HOLD_SCALE_VAL*	**开环**待机电流调节值。
			CL_START_DOWN	|*ENC_POS_DEV*|小于该参数时，闭环从 *CL_IMAX* 开始减小电流值。 **i 推荐:设置为 0 将 CL_BETA 自动分配给该参数。**

表 71: 电流调节值 (0x06)

注意:

→ *BOOST_SCALE_VAL, DRV1/DRV2_SCALE_VAL, HOLD_SCALE_VAL, CL_IMIN, CL_IMAX。*

→ *如果 spi_output_format = b'1011 或 b'1100，则实际调节值= (x+1) / 32*

→ *其它 spi_output_format 设置 则实际调节值= (x+1) / 256。*

14.8. 各种调节配置寄存器

各种电流调节配置寄存器				
R/W	**地址**	**位**	**值**	**描述**
RW	0x15	31:0	*STDBY_DELAY (缺省:0x00000000)*	
			U	斜坡停止到待机有效之间的延迟时间[时钟周期数]。
	0x16	31:0	*FREEWHEEL_DELAY (缺省:0x00000000)*	
			U	待机阶段有效到飞轮有效之间的延迟时间[时钟周期数]。
	0x17	23:0	*VDRV_SCALE_LIMIT (缺省:0x000000)*	***(非电压 PWM 模式)***
			U	调节驱动电流的速度阈值: 如果 *VACTUAL > VDRV_SCALE_LIMIT，则 DRV2_SCALE_VAL 有效* 如果 *VACTUAL ≤ VDRV_SCALE_LIMIT，则 RV1_SCALE_VAL 有效*
			第二个分配:如果使能电压 PWM,也用作 *PWM_VMAX* (见 0)	
	0x18	23:0	*UP_SCALE_DELAY (缺省:0x000000)*	***(开环操作)***
			U	电流递增延迟[时钟周期数]。对应电流朝着更高的值增加一步的时钟周期间隔。
			CL_UPSCALE_DELAY (缺省:0x000000)	***(闭环操作)***
			U	电流递减延迟[时钟周期数]。对应闭环时电流朝着更高的值增加一步的时钟周期间隔。
	0x19	23:0	*HOLD_SCALE_DELAY (缺省:0x000000)*	***(开环操作)***
			U	电流递减延迟[时钟周期数]。对应电流朝着保持电流降低一步的以时钟周期为单位的时间间隔。
			CL_DNSCALE_DELAY(缺省: 0x000000)	***(闭环操作)***
			U	电流递减延迟[时钟周期数]。对应闭环时电流朝着保持电流降低一步的以时钟周期为单位的时间间隔。
	0x1A	23:0	*DRV_SCALE_DELAY (缺省:0x000000)*	
			U	递减延迟[时钟周期数]，对应驱动电流调节值向较低值降低一步的以时钟周期为单位的时间间隔。
	0x1B	31:0	*BOOST_TIME (缺省:0x00000000)*	
			U	斜坡开始后到调节上升电流过程中的时间[时钟周期数]。
R	0x7C	8:0	*SCALE_PARAM*	
			U	实际使用的电流调节参数。
W			第二次赋值:也用作写访问的 *CIRCULAR_DEC* (参见第 19.16. 节)。	

表 72: 各种缩放寄存器配置(0x15…0x1B)

14.9. 编码器信号配置(0x07)

***ENC_IN_CONF* 0x07** *(缺省 0x00000400)*			
R/W	**位**	**值**	**描述**
RW	0	*enc_sel_decimal*	
		0	编码器常数为二进制数。
		1	编码器常数为十进制数(仅适用于 ABN)。
	1	*clear_on_n*	
		0	*ENC_POS* 未设置为 *ENC_RESET_VAL* 值。
		1	*ENC_POS* 设置为 *ENC_RESET_VAL* 如果 *clr_latch_cont_on_n*=1, 在每个 N 事件 或者 如果 *clr_latch_once_on_n*=1，在下一个 N 事件 **!** **不要在闭环操作中设置该位。**
	2	*clr_latch_cont_on_n*	
		0	每发生一次 N 事件，*ENC_POS* 信息的值不会被清除和/或锁存。
		1	每发生一次 N 事件，*ENC_POS* 信息的值被清除和/或锁存。
	3	*clr_latch_once_on_n*	
		0	下一次 N 事件，*ENC_POS* 信息的值不会被清除和/或锁存。
		1	下一次 N 事件，*ENC_POS* 信息的值会被清除和/或锁存。 **i** **锁存/清零一次后，该位置 0.**
	4	*pol_n*	
		0	N 事件的有效极性为低电平。
		1	N 事件的有效极性为高电平。
	6:5	*n_chan_sensitivity*	
		0	只要 N 信号等于 N 事件的有效极性，N 事件有效的。
		1	当 N 信号切换到 N 事件的有效极性时，N 事件触发。
		2	当 N 信号切换到 N 事件的非有效极性时，N 事件触发。
		3	当 N 信号切换到 N 事件的有效/非有效极性时，N 事件触发(双边沿)。
	7	*pol_a_for_n*	
		0	对于有效的 N 事件，A 信号极性必须为低电平。
		1	对于有效的 N 事件，A 信号极性必须为高电平。
	8	*pol_b_for_n*	
		0	对于有效的 N 事件，B 信号极性必须为低电平。
		1	对于有效的 N 事件，B 信号极性必须为高电平。
	9	*ignore_ab*	
		0	TMC4361A 根据 AB 信号的有效电平产生有效的 N 事件。
		1	N 事件不考虑 AB 信号的有效极性。

ꕥ∀ *下一页继续。*

***ENC_IN_CONF* 0x07** *(缺省 0x00000400)*			
R/W	**位**	**值**	**描述**
RW	10	*latch_enc_on_n*	
		0	*ENC_POS* 没有被锁存。
		1	使能 *ENC_LATCH* 锁存 *ENC_POS* 如果 *clr_latch_cont_on_n*=1, 则在每个 N 事件锁存 或 如果 *clr_latch_once_on_n*=1，则在下一下 N 事件 锁存
	11	*latch_x_on_n*	
		0	*XACTUAL* 没有被锁存。
		1	使能 *X_LATCH* 锁存 *XACTUAL* 如果 *clr_latch_cont_on_n*=1, 在每个 N 事件锁存 或 如果 *clr_latch_once_on_n*=1，则在下一下 N 事件锁存
	12	*multi_turn_in_en*	***(仅绝对编码器)***
		0	所连 的串行编码器传输单圈数据值。
		1	所连 的串行编码器传输单圈和多圈数据值。
	13	*multi_turn_in_signed*	***(仅绝对编码器)***
		0	串行编码器输入的多圈值是无符号数。
		1	串行编码器输入的多圈值是有符号数。
	14	*multi_turn_out_en*	***(仅串行编码器)***
		0	串行编码器输出发送单圈数据值。
		1	串行编码器输出发送单圈和多圈数据值。
	15	*use_usteps_instead_of_xrange*	
		0	如果编码器使能圆周运动，则 *X_RANGE* 有效。
		1	如果编码器使能圆周运动，则 *USTEPS_PER_REV* 有效。
	16	*calc_multi_turn_behav*	***(仅绝对编码器)***
		0	无多圈计算。
		1	TMC4361A 为单圈编码器计算多圈数据。
	17	*ssi_multi_cycle_data*	***(仅绝对编码器)***
		0	每个 SSI 值请求执行一次。
		1	每个 SSI 值请求执行两次。
	18	*ssi_gray_code_en*	***(仅绝对编码器)***
		0	SSI 输入数据是二进制编码。
		1	SSI 输入数据是格雷编码的。
	19	*left_aligned_data*	***(仅绝对编码器)***
		0	串行输入数据右对齐(先标志，后数据)。
		1	串行输入数据左对齐(先数据，后标志)。
		∀下一页继续。	

***ENC_IN_CONF* 0x07** *(缺省 0x00000400)*			
R/W	**位**	**值**	**描述**
RW	20	*spi_data_on_cs*	***(仅 SPI 编码器)***
		0	BNEG_NSDI 在下一个串行时钟线(A_SCLK)转换时输出串行数据。
		1	NEG_NSDI 在片选信号线 ANEG_NSCLK 切换到低电平的时候立即输出串行数据。
	21	*spi_low_before_cs*	***(仅 SPI 编码器)***
		0	串行时钟线 A_SCLK 在反相片选信号线 ANEG_NSCLK 切换到低电平之后切换到低电平。
		1	串行时钟线 A_SCLK 在反相片选信号线 ANEG_NSCLK 切换到低电平之前切换到低电平。
	23:22	*regulation_modus*	
		0	编码器反馈数据没有内部调节。
		1	闭环操作使能。 **！ 仅使用最大微步分辨率！(256 微步/全步=0 → MSTEPS_PER_FS=0)！**
		2	使能 PID 调节。脉冲发生器速度基准为 0。
		3	使能 PID 调节。脉冲发生器速度基准为 VACTUAL。
	24	*cl_calibration_en*	***(仅闭环操作)***
		0	不使能闭环校准。
		1	使能闭环校准。 **！ 校准期间使用最大电流而不调节电流。** **！ 在校准过程中，建议将电机驱动器保持在全步位置，不发生运动。**
	25	*cl_emf_en*	***(仅闭环操作)***
		0	闭环操作期间不使能反电动势补偿。
		1	在闭环操作期间使能反电动势补偿。 在闭环操作期间，当 *|VACTUAL| > CL_VMIN* 的情况下补偿反电动势。
	26	*cl_clr_xact*	***(仅闭环操作)***
		0	*闭环操作期间，XACTUAL 不复位至 ENC_POS。*
		1	在闭环操作期间，如果 *|ENC_POS_DEV|> ENC_POS_DEV_TOL*，则 *XACTUAL* 设置为 ENC_POS。 **！ 必须完全理解该功能，才能使用**
	27	*cl_vlimit_en*	***(仅闭环操作)***
		0	闭环调节期间追赶速度没有限制。
		1	闭环运行期间的追赶速度由内部 PI 调节器调节。
	28	*cl_velocity_mode_en*	***(仅闭环操作)***
		0	不使能闭环速度模式。
		1	不使能闭环速度模式。 如果 *|ENC_POS_DEV| > 768, XACTUAL* 相应调整。
	29	*invert_enc_dir*	
		0	编码器方向在内部不反转。
		1	编码器方向在内部反转。

∞∀下一页继续。

***ENC_IN_CONF* 0x07** *(缺省 0x00000400)*			
R/W	**位**	**值**	**描述**
RW	30	*enc_out_gray*	***(仅串行编码器输出)***
		0	SSI 输入数据为二进制编码。
		1	SSI 输入数据为格雷编码。
	31	*no_enc_vel_preproc*	***(增量 ABN 编码器)***
		0	先对 AB 信号进行预处理用于内部编码器速度计算。
		1	无 AB 信号预处理。 **！ 建议设置 AB 预处理，以便过滤编码器信号，减小谐振。**
		serial_enc_variation_limit	***(绝对编码器)***
		0	绝对编码器数据没有变化限制。
		1	两个连续的串行编码器值不偏离指定的限制才有效。如果 $\lvert ENC\_POS_X – ENC\_POS_{X-1} \rvert > 1/8 * SER\_ENC\_VARIANCE * ENC\_IN\_RES$，则 ENC_POS_X 无效且不赋值给 *ENC_POS*。

表 73: 编码器串行配置 ENC_IN_CONF (0x07)

14.10. 串行编码器数据输入配置(0x08)

ENC_IN_DATA 0x08 *(缺省: 0x00000000)*			
R/W	**位**	**值**	**备注**
RW	4:0	*SINGLE_TURN_RES (缺省: 0x00)*	
		U	一圈内角度数据的位宽= *SINGLE_TURN_RES* + 1。 ! 设置 ***SINGLE_TURN_RES*** **< 31.**
	9:5	*MULTI_TURN_RES (缺省: 0x00)*	
		U	圈数计数器的位宽= *MULTI_TURN_RES* + 1
	11:10	*STATUS_BIT_CNT (缺省: 0x0)*	
		U	状态数据位的位宽。
	15:12	保留，设置为 0x0.	
	23:16	*SERIAL_ADDR_BITS (缺省: 0x00)*	***(仅 SPI 编码器)***
		U	SPI 配置数据报中的 SPI 编码器地址位位宽。
	31:24	*SERIAL_DATA_BITS (缺省: 0x00)*	***(仅 SPI 编码器)***
		U	SPI 配置数据报中的 SPI 编码器数据位位宽。

表 74: 串行编码器数据输入配置 ENC_IN_DATA (0x08)

14.11. 串行编码器数据输出配置(0x09)

ENC_OUT_DATA 0x09 *(缺省: 0x00000000)*			
R/W	**位**	**值**	**备注**
RW	4:0	*SINGLE_TURN_RES_OUT (缺省: 0x00)*	
		U	一圈内角度数据的位宽= *SINGLE_TURN_RES_OUT* + 1
	9:5	*MULTI_TURN_RES_OUT (缺省: 0x00)*	
		U	圈数计数器的位宽= *MULTI_TURN_RES_OUT* + 1
	31:12	保留，设置为 0x00000.	

表 75: 串行编码器数据输出配置 ENC_OUT_DATA (0x09)

 阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

14.12. 电机驱动器设置寄存器 STEP_CONF 0x0A

STEP_CONF 0x0A *(Default: 0x00FB0C80)*			
R/W	**位**	**值**	**备注**
RW	3:0	*MSTEP_PER_FS (缺省: 0x0)*	
		0	最高微步: 256 微步 **i 闭环操作设置为 256。** **i 当使用 Step/Dir 驱动器时，为了获得最佳性能，Step/Dir 输入细分为 256 分辨率 (但也可以使用分辨率较低的 Step/Dir 驱动器)**
		1	128 微步。
		2	64 微步。
		3	32 微步。
		4	16 微步。
		5	8 微步。
		6	4 微步。
		7	半步: 2 微步。
		8	全步（最大）。
	15:4	*FS_PER_REV (缺省: 0x0C8)*	
		U	电机每旋转一圈的全步值。
	23:16	*MSTATUS_SELECTION (缺省: 0xFB)*	
			选择电机驱动器状态位作为 SPI 响应数据报: 与 电机驱动器状态寄存器组 (7:0)进行或运算: 如果设置了选择位且电机步进驱动器对应的选择位标志有效，则 EVENTS(30)产生事件。
	31:24	保留，设为 0x00。	

表 76: 电机驱动设置 (0x0A)

14.13. 事件选择寄存器 0x0B..0X0D

事件选择寄存器			
R/W	**地址**	**位**	**备注**
RW	0x0B	*SPI_STATUS_SELECTION (缺省: 0x82029805)*	
		31:0	SPI 数据报的事件选择: 该寄存器选择了事件寄存器 0x0E 中的事件 (=1)，对应的事件位会和每个 SPI 数据报的八个状态位(LSB 的前八位是有效的！)一起传输。
	0x0C	*EVENT_CLEAR_CONF (缺省: 0x00000000)*	
		31:0	事件保护配置: 读事件寄存器 0x0E，该寄存器中选择(=1)的事件位不会被清零。
	0x0D	*INTR_CONF (缺省: 0x00000000)*	
		31:0	INTR 输出的事件选择: 此处选择(=1)的 *EVENTS* 事件寄存器 0x0E 的事件位与中断事件寄存器组进行“或”运算: 如果所选事件中有任何一个处于活动状态，则会在 INTR 产生中断。

表 77: 事件选择寄存器 0x0B...0x0D

14.14. 状态事件寄存器(0x0E)

状态事件寄存器 *EVENTS 0x0E*		
R/W	**位**	**描述**
R+CW	0	*TARGET_REACHED* 被触发。
	1	*POS_COMP_REACHED* 被触发。
	2	*VEL_REACHED* 被触发。
	3	*VEL_STATE* = b'00 被触发 (*VACTUAL* = 0)。
	4	*VEL_STATE* = b'01 被触发 (*VACTUAL* > 0)。
	5	*VEL_STATE* = b'10 被触发(*VACTUAL* < 0)。
	6	*RAMP_STATE* = b'00 被触发(*AACTUAL* = 0, *VACTUAL* 是常数)。
	7	*RAMP_STATE* = b'01 被触发(|*VACTUAL*| 增加)。
	8	*RAMP_STATE* = b'10 被触发(|*VACTUAL*| 增加)。
	9	*MAX_PHASE_TRAP*: 梯形斜坡已经达到其极限速度，调用 *AMAX* 或 *DMAX* (|*VACTUAL*| > *VBREAK*; *VBREAK*≠0) 的最大值。
	10	*FROZEN:* NFREEZE 切换到低电平 **i 重置 TMC4361A，以便进一步运动。**
	11	STOPL 被触发。直到该事件被清除并且(STOPL 不再有效或者 *stop_left_en* 设置为 0)后，才会执行反向运动
	12	STOPR 被触发。直到该事件被清除并且(STOPR 不再有效或者 *stop_right_en* 设置为 0)后，才会执行正向运动。
	13	*VSTOPL_ACTIVE: VSTOPL* 有效。在此事件被清除并且(设置一个新的 *VSTOPL* 速度或 *virtual_left_limit_en* 被设置为 0) 之前，没有反向运动。
	14	*VSTOPR_ACTIVE: VSTOPR* 有效。在此事件被清除并且(设置一个新的 *VSTOPR* 速度或 *virtual_right_limit_en* 被设置为 0) 之前，没有正向运动。
	15	*HOME_ERROR:* 无法匹配 *HOME_REF* 极性，HOME 超出安全范围。
	16	*XLATCH_DONE* 指示是否重置 *X_LATCH* 或回零过程已经完成。
	17	*FS_ACTIVE*: 全步模式
	18	*ENC_FAIL*: *XACTUAL* 和 *ENC_POS* 之间的不匹配已超过指定的限制。
	19	*N_ACTIVE:* N 事件有效 。
	20	*ENC_DONE* 指示 *ENC_LATCH* 是否被重置。
	21	*SER_ENC_DATA_FAIL:* 多周期数据期间或两次连续数据请求之间出现故障。
	22	保留
	23	*SER_DATA_DONE:*从 SPI 串行接口编码器接收到配置数据。
	24	*SERIAL_ENC_*Flags 中的一个被置位 。
	25	*COVER_DONE:*给电机驱动器的 SPI 数据报发送完成。
	26	*ENC_VEL0:*编码器速度到 0。
	27	*CL_MAX:* 闭环换向角度已达到最大值。
	28	*CL_FIT:* 闭环偏差已达到内部极限。
	29	*STOP_ON_STALL:* 检测到电机堵转。电机斜坡已经停止。
	30	*MOTOR_EV:*选定的 TMC 电机驱动器标志之一被触发。
	31	*RST_EV*: 复位被触发。

表 78: 状态事件寄存器 EVENTS (0x0E)

14.15. 状态标志寄存器(0x0F)

状态标志寄存器 *STATUS 0x0F*		
R/W	**位**	**描述**
R	0	如果 *XACTUAL = XTARGET*，则 *TARGET_REACHED_F* 置 *1*
	1	如果 *XACTUAL = POS_COMP*，则 *POS_COMP_REACHED_F* 置 *1*
	2	如果 *VACTUAL* = |*VMAX*|，则 *VEL_REACHED_F* 置 *1*
	4:3	*VEL_STATE_F*: 当前速度状态: 0 → *VACTUAL* = 0; 1 → *VACTUAL* > 0; 2 → *VACTUAL* < 0
	6:5	*RAMP_STATE_F*: 当前斜坡状态: 0 → *AACTUAL* = 0; 1 → *AACTUAL* 增加(加速); 2 → *AACTUAL* 减小(减速)
	7	*STOPL_ACTIVE_F*: 左停止开关有效。
	8	*STOPR_ACTIVE_F*: 右停止开关有效。
	9	*VSTOPL_ACTIVE_F*: 左虚拟停止开关有效。
	10	*VSTOPR_ACTIVE_F*: 右虚拟停止开关有效。
	11	*ACTIVE_STALL_F*: 电机堵转检测且 *VACTUAL* > *VSTALL_LIMIT*。
	12	*HOME_ERROR_F*: HOME_REF 输入信号电平不等于预期的回零电平。
	13	*FS_ACTIVE_F*: 全步有效。
	14	*ENC_FAIL_F*: *XACTUAL* 和 *ENC_POS* 中间的差值超出容差。
	15	*N_ACTIVE_F*: N 事件有效
	16	*ENC_LATCH_F*: *ENC_LATCH* 被重置。
	17	仅适用于绝对编码器: *MULTI_CYCLE_FAIL_F* 表示上一个多圈数据出现故障。
		仅适用于绝对编码器: *SER_ENC_VAR_F* 表示由于两个连续串行数据值之间的重大偏差，因此在上次串行数据出现故障。
	18	保留。
	19	*CL_FIT_F*:. 如果 *ENC_POS_DEV* < *CL_COMPLATION*，则有效。*XACTUAL* 和 *ENC_POS* 之间的差值在容许范围内
	23:20	仅适用于绝对编码器:从编码器接收的 *SERIAL_ENC_FLAGS*。新的编码器传输请求是会重置这些标志。
	24	**仅 TMC26x / TMC21x0:** *SG*: StallGuard2 状态。 **仅 TMC24x 的可选:** 计算的 stallGuard 状态 status。
		仅 TMC23x / TMC24x: *UV_SF*: 欠压标志。
	25	**所有 TMC 电机驱动器**: *OT*: 过温关断。
	26	**所有 TMC 电机驱动器**: *OTPW*: 过温预警。
	27	**仅 TMC26x / TMC21x0:** *S2GA*: 线圈 A 高端 MOSFET 管对地短路检测位。
		仅 TMC23x / TMC24x: *OCA*: 线圈 A 过流。
	28	**仅 TMC26x / TMC21x0:** *S2GB*: 线圈 B 高端 MOSFET 管对地短路检测位。
		仅 TMC23x / TMC24x: *OCB*: 线圈 B 过流。
	29	**所有 TMC 电机驱动器**: *OLA*: 线圈 A 开路。
	30	**所有 TMC 电机驱动器**: *OLB*: 线圈 B 开路。
	31	**仅 TMC26x / TMC21x0:** *STST*: 静止状态。
		仅 TMC23x / TMC24x: *OCHS*: 高侧过流。

表 79: S 状态标志寄存器 STATUS (0x0F)

14.16. 各种配置寄存器: S/D, 同步, 等

各种配置寄存器: 闭环, 开关...				
R/W	**地址**	**位**	**值**	**描述**
RW	0x10	15:0	*STP_LENGTH_ADD (缺省: 0x0000)*	
			U	延长 STPOUT 步进信号有效步进极性的时间长度[时钟周期数]。
		31:16	*DIR_SETUP_TIME (缺省: 0x0000)*	
			U	DIROUT 和 STPOUT 之间电平变化的延迟时间[时钟周期数]。
	0x11	31:0	*START_OUT_ADD (缺省:0x00000000)*	
			U	延长启动信号的有效时间[时钟周期数]。 有效启动信号 = 1+START_OUT_ADD
	0x12	31:0	*GEAR_RATIO (缺省:0x01000000)*	
			S	STPIN 端一个有效的步进输入信号对应的内部位置计数器累加常数值。 数值表示:8 位数字和 24 位小数。
	0x13	31:0	*START_DELAY (缺省:0x00000000)*	
			U	开始启动触发到内部启动信号有效之间的延迟时间[时钟周期数]。
	0x14	31:0	*CLK_GATING_DELAY (缺省:0x00000000)*	
			U	时钟门控开始触发到时钟门控开始有效之间的延迟时间[时钟周期数]。
	0x1D	23:0	*SPI_SWITCH_VEL*	
			U	发送自动覆盖数据报的速度绝对值[pps]。
		31:0	第二次分配: 如果 SPI-DAC 模式使能，也用作 *DAC_ADDR_A/B* (见 19.30.)。	
	0x1E	15:0	*HOME_SAFETY_MARGIN (缺省: 0x0000)*	
			U	在 *X_HOME* ± *HOME_SAFETY_MARGIN* 范围内，HOME_REF 极性无效，无错误标志。
	0x1F	11:0	*CHOPSYNC_DIV (缺省: 0x0280)*	***(TMC23x/24x 的 ChopSync 使能)***
			U	定义斩波频率的斩波时钟分频器 f_{OSC}: $f_{OSC} = f_{CLK}/CHOPSYNC\_DIV$ 条件 96 ≤ *CHOPSYNC_DIV* ≤ 818
		15:0	第二次分配: 如果启用电压 PWM，也用作 *PWM_FREQ*(见 0)	
W	0x60	31:0	*FS_VEL(缺省:0x000000)*	***(Closed-loop 和 dcStep 禁用)***
			U	最小全步速度 [pps]。如果使能，则|*VACTUAL*| > *FS_VEL* 时全步是有效的，。
			第二次分配: 如果启用 dcStep，也用作 *DC_VEL*(参见章节 19.27.) 第三次分配: 如果闭环使能，也用作 CL_VMIN_EMF(参见章节 19.26)	
	0x64	31:0	保留，设置为 0x00000000.	
	0x67	23:0	*VSTALL_LIMIT (缺省:0x00000000)*	
			U	堵转停止速度极限[pps]: 如果使能，只有超过此限制，有效的堵转才会导致堵转停止。
	0x7B	31:0	*TZEROWAIT (Default:0x00000000)*	
			U	达到 *VACTUAL* = 0 后的静止阶段。
R			第二次分配: 也用作读 *CURRENTA/B_SPI* (见章节 19.29.)	
W	0x7C	31:0	*CIRCULAR_DEC (Default:0x00000000)*	
			U	如果定义的圆周运动一圈不是微步的偶数，则需设置该小数。 数值表示:1 位整数，31 位小数。
R		8:0	第二次分配: 也用作读 *SCALE_PARAM* (见章节。)	

表 80: 各种配置寄存器: S/D, 同步, 等

14.17. PWM 配置寄存器

PWM 配置寄存器				
R/W	**地址**	**位**	**值**	**描述**
RW	0x17	23:0	*PWM_VMAX (缺省:0x00000000)*	***(电压 PWM 使能)***
			U	达到最大调节参数值 1.0 时的 PWM 速度值。
			第二次分配: 如果没有使能电压 PWM，也用作 *VDRV_SCALE_LIMIT* (19.29.)	
	0x1F	15:0	*PWM_FREQ (缺省: 0x0280)*	***(电压 PWM 使能)***
			U	PWM 周期对应的时钟数 。
		11:0	第二次分配: 如果没有使能电压 PWM，也用作 *CHOPSYNC_DIV* (见 19.16.)	
W	0x79	9:0	*MSOFFSET (缺省:0x000)*	***(TMC23x/24x only)***
			U	PWM 模式中的微步偏移量。
			第二次分配: 也用作可读的 *MSCNT* (见 19.29.)	

表 81: PWM 配置寄存器.

14.18. 斜坡生成器寄存器

斜坡生成器寄存器				
R/W	**地址**	**位**	**值**	**描述**
RW	0x20	*RAMPMODE* (缺省:0x0)		
		2	操作模式:	
			1	**定位模式:** *XTARGET* 是速度斜坡的最佳目标。
			0	**速度模式**: *VMAX* 是速度斜坡的最佳目标。
		1:0	运动轮廓:	
			0	**无斜坡:** *VACTUAL* 仅遵循 VMAX(矩形速度形状)。
			1	**梯形斜坡(包括六点斜坡):** 在不调整加速度值的情况下，考虑产生 *VACTUAL* 的加速度和减速度值。
			2	**S 形斜坡:**考虑产生 *VACTUAL* 的所有斜坡参数(包括弓形值)。
RW	0x21	31:0	*XACTUAL (缺省: 0x00000000)*	
			S	实际内部电机位置[脉冲]: $-2^{31} \le XACTUAL \le 2^{31} - 1$
R	0x22	31:0	*VACTUAL (缺省: 0x00000000)*	
			S	实际斜坡发生器的速度[每秒脉冲数]: 1 pps ≤ |*VACTUAL*| ≤ *CLK_FREQ* · ½ 脉冲 (f_{CLK} = 16 MHz → 8 Mpps)
R	0x23	31:0	*AACTUAL (缺省: 0x00000000)*	
			S	实际加速/减速值 [脉冲/秒 ²]: -2^{31} pps² ≤ *AACTUAL* ≤ $2^{31} - 1$ 1 pps² ≤ |*AACTUAL*|
RW	0x24	31:0	*VMAX (缺省: 0x00000000)*	
			S	定位模式下的最大斜坡发生器速度 或
				速度模式下的斜坡发生器的目标速度，无斜坡运动轮廓。
				数值表示:23 位数字和 8 位小数。 ! 考虑第 44 页第 6.7.5 节中的最大值
RW	0x25	30:0	*VSTART (缺省: 0x00000000)*	
			U	定位模式和速度模式下的起始速度的绝对值。 如果使用 *VSTART*: S 形斜坡没有第一个弓形阶段 B1。
				定位模式下的 *VSTART:* *如果 VACTUAL* = 0 和 *XTARGET* ≠ *XACTUAL*: *VACTUAL* = 0 → *VSTART* 过程没有加速度阶段。
				速度模式下的 *VSTART*: 如果 *VACTUAL* = 0 和 *VACTUAL* ≠ *VMAX*: *VACTUAL* = 0 → *VSTART* 过程没有加速度阶段。
				数值表示:23 位数字和 8 位小数。 ! 考虑章节 6.7.5 第 44 页的最大值

ꕥ∀ 下一页继续

斜坡生成器寄存器				
R/W	**地址**	**位**	**值**	**描述**
RW	0x26	30:0	*VSTOP (缺省:0x00000000)*	
			U	定位模式和速度模式下的停止速度的绝对值。 如果使用 *VSTOP*: S 形斜坡则没有最后一个弓形阶段 B4。 如果 VSTOP 非常小，并且使用定位模式，斜坡有可能在达到 *XTARGET* 之前以固定的 *VACTUAL* = VSTOP 结束。
				定位模式下的 *VSTOP*: 如果 *VACTUAL*≤*VSTOP* 和 *XTARGET*=*XACTUAL*: *VACTUAL* 立即设置为 0。
				速度模式下的 *VSTOP*: 如果 *VACTUAL* ≤ *VSTOP* 和 *VMAX* = 0: *VACTUAL* 立即设置为 0。
				数值表示:23 位数字和 8 位小数。 ! 考虑章节 6.7.5 第 44 页的最大值
	0x27	30:0	*VBREAK (缺省:0x00000000)*	
			U	定位模式和速度模式下的绝对断点速度， 这仅适用于梯形斜坡运动轮廓。 如果 *VBREAK* = 0: 纯线性斜坡仅由 *AMAX/DMAX* 生成。
				如果 |*VACTUAL*| < *VBREAK*: |*AACTUAL*| = *ASTART* 或 *DFINAL* 如果|*VACTUAL*| ≥ *VBREAK*: |*AACTUAL*| = *AMAX* 或 *DMAX*
				! **总是设置 *VBREAK* > *VSTOP*! 如果 VBREAK ≠ 0.**
				数值表示:23 位数字和 8 位小数. ! 考虑章节 6.7.5 第 44 页的最大值
	0x28	23:0	*AMAX (缺省: 0x000000)*	
			U	S 形斜坡运动轮廓: 最大加速度值。
				梯形斜坡运动轮廓: 如果 |*VACTUAL*| ≥ *VBREAK* 或者 *VBREAK* = 0 情况下的加速度值
				数值表示: **频率模式**: [每平方秒的脉冲] 22 位数字和两位小数: 250 mpps² ≤ *AMAX* ≤ 4 Mpps² **直接模式:** [每时钟周期速度变化值] a[Δv 每时钟周期]= *AMAX* / 2^{37} *AMAX* [pps²] = *AMAX* / 2^{37} • f_{CLK}^2 ! 考虑章节 6.7.5 第 44 页的最大值
	0x29	23:0	*DMAX (缺省: 0x000000)*	
			U	S 形斜坡运行轮廓： 最大减速值。
				梯形斜坡运动轮廓: 如果 |*VACTUAL*| ≥ *VBREAK* 或者 *VBREAK* = 0 情况下的减速度值
				数值表示: **频率模式**: [每平方秒的脉冲] 22 位数字和两位小数: 250 mpps² ≤ *DMAX* ≤ 4 Mpps² **直接模式:** [每时钟周期速度变化值] d[Δv 每时钟周期]= *DMAX* / 2^{37} *DMAX* [pps²] = *DMAX* / 2^{37} • f_{CLK}^2 ! 考虑章节 6.7.5 第 44 页的最大值

下一页继续

斜坡生成器寄存器				
R/W	**地址**	**位**	**值**	**描述**
RW	0x2A	23:0	*ASTART (Default: 0x000000)*	
			U	S 形斜坡运行轮廓: 开始加速度值。
				梯形斜坡运动轮廓: |*VACTUAL*| < *VBREAK* 阶段的加速度。
				从外部步进控制切换到内部步进控制后的加速度值。
				数值表示: **频率模式**: [每平方秒的脉冲] 22 位数字和两位小数: 250 mpps^2 ≤ *ASTART* ≤ 4 Mpps^2 **直接模式:** [每时钟周期的速度变化值] a[每时钟周期的速度变化值]= *ASTART* / 2^{37} *ASTART* [pps^2] = *ASTART* / 2^{37} • f_{CLK}^2 ! 考虑章节 6.7.5 第 44 页的最大值
		31		从外部步进控制切换到内部步进控制后的 AACTUAL 标志。
	0x2B	23:0	*DFINAL (Default: 0x000000)*	
			U	S 形斜坡运行轮廓: 停止减速值，该值在定位模式下不使用。
				梯形斜坡运动轮廓: |*VACTUAL*| < *VBREAK* 阶段的减速度
				数值表示: **频率模式**: [每平方秒的脉冲] 22 位数字和两位小数: 250 mpps^2 ≤ *DFINAL* ≤ 4 Mpps^2 **直接模式:** [每时钟周期速度变化值] d[每时钟周期的速度变化值]= *DFINAL* / 2^{37} *DFINAL* [pps^2] = *DFINAL* / 2^{37} • f_{CLK}^2 ! 考虑章节 6.7.5 第 44 页的最大值
	0x2C	23	*DSTOP (缺省: 0x000000)*	
			U	自动线性斜坡到 *VACTUAL = 0* 停止过程中的减速度值。如果 *soft_stop_enable* 设置为 1，*DSTOP* 与有效的外部停止开关(STOPL 或 STOPR)结合使用；或者 *virt_stop_mode* 设置为 2，*DSTOP* 与有效的虚拟停止开关结合使用。
				数值表示: **频率模式**: [每平方秒的脉冲] 22 位数字和两位小数: 250 mpps^2 ≤ *DSTOP* ≤ 4 Mpps^2 **直接模式:** [每时钟周期速度变化值] d[每时钟周期速度变化值]= *DSTOP* / 2^{37} *DSTOP* [pps^2] = *DSTOP* / 2^{37} • f_{CLK}^2 ! 考虑章节 6.7.5 第 44 页的最大值

∀ 下一页继续

斜坡生成器寄存器				
R/W	**地址**	**位**	**值**	**描述**
RW	0x2D	23:0	*BOW1 (缺省: 0x000000)*	
			U	弓形值 1(加速斜坡的第一个弓形值 B1)。
				数值表示: **频率模式**: [每立方秒的脉冲] 24 位整数，无小数位: 1 pps^3 ≤ *BOW1* ≤ 16 $Mpps^3$ **直接模式:** [每时钟周期的加速度变化值] bow[av 每时钟周期]= *BOW1* / 2^{53} *BOW1* [pps^3] = *BOW1* / 2^{53} • f_{CLK}^3 ! 考虑章节 6.7.5 第 44 页的最大值
	0x2E	23:0	*BOW2 (缺省: 0x000000)*	
			U	弓形值 2(加速斜坡的第一个弓形 B2)。
				数值表示: **频率模式**: [每秒 ^{3} 的脉冲] 24 位整数，无小数位: 1 pps^3 ≤ *BOW2* ≤ 16 $Mpps^3$ **直接模式:** [每时钟周期的加速度变化值] bow[av 每时钟周期]= *BOW2* / 2^{53} *BOW2* [pps^3] = *BOW2* / 2^{53} • f_{CLK}^3 ! 考虑章节 6.7.5 第 44 页的最大值
	0x2F	23:0	*BOW3 (缺省: 0x000000)*	
			U	弓形值 3(加速斜坡的第一个弓形 B3)..
				数值表示: **频率模式**: [每立方秒的脉冲] 24 位整数，无小数位: 1 pps^3 ≤ *BOW3* ≤ 16 $Mpps^3$ **直接模式:** [每时钟周期的加速度变化值] bow[av 每时钟周期]= *BOW3* / 2^{53} *BOW3* [pps^3] = *BOW3* / 2^{53} • f_{CLK}^3 ! 考虑章节 6.7.5 第 44 页的最大值
	0x30	23:0	*BOW 4 (缺省: 0x000000)*	
			U	弓形值 4(加速斜坡的第一个弓形 B4)..
				数值表示: **频率模式**: [每立方秒的脉冲] 24 位整数，无小数位: 1 pps^3 ≤ *BOW4* ≤ 16 $Mpps^3$ **直接模式:** [每时钟周期的加速度变化值] bow[av 每时钟周期]= *BOW4* / 2^{53} *BOW4* [pps^3] = *BOW4* / 2^{53} • f_{CLK}^3 ! 考虑章节 6.7.5 第 44 页的最大值

表 82: 斜坡发生器

14.19. 外部时钟频率寄存器

外部时钟频率寄存器				
R/W	**地址**	**位**	**值**	**描述**
RW	0x31	24:0	*CLK_FREQ (缺省: 0x0F42400)*	
			U	外部时钟输入 f_{CLK} [Hz]， 满足 4.2 MHz ≤ f_{CLK} ≤ 30 MHz

表 83: 外部时钟频率寄存器

14.20. 目标寄存器和比较寄存器

目标寄存器和比较寄存器				
R/W	**地址**	**位**	**值**	**描述**
RW	0x32	31:0	*POS_COMP (缺省: 0x00000000)*	
			S	比较位置。
RW	0x33	31:0	*VIRT_STOP_LEFT (缺省: 0x00000000)*	
			S	虚拟左停止位置。
RW	0x34	31:0	*VIRT_STOP_RIGHT (缺省: 0x00000000)*	
			S	虚拟右停止位置。
RW	0x35	31:0	*X_HOME (缺省: 0x00000000)*	
			S	实际回零位置。
R	0x36	31:0	*X_LATCH (缺省: 0x00000000)*	***(如果 circular_cnt_as_xlatch = 0)***
			S	某些触发器有效时的存储位置.
			REV_CNT (缺省: 0x00000000)	***(如果 circular_cnt_as_xlatch = 1)***
			S	圆周运动中的转数。
W		30:0	*X_RANGE (缺省: 0x00000000)*	
			U	圆周运动期间的 X_ACTUAL 限制: *-X_RANGE ≤ X_ACTUAL ≤ X_RANGE* - 1
RW	0x37	31:0	*X_TARGET (缺省: 0x00000000)*	
			U	定位模式下的目标电机位置。 ! **请事先设置运动轮廓中所有其他参数！**

表 84: 目标和比较基础寄存器

14.21. 流水线寄存器

流水线寄存器				
R/W	**地址**	**位**	**值**	**描述**
RW	0x38	31:0	S	*X_PIPE0 (缺省: 0x00000000): 1st 流水线寄存器*
	0x39	31:0	S	*X_PIPE1 (缺省: 0x00000000): 2nd 流水线寄存器*
	0x3A	31:0	S	*X_PIPE2 (缺省: 0x00000000): 3rd 流水线寄存器*
	0x3B	31:0	S	*X_PIPE3 (缺省: 0x00000000): 4th 流水线寄存器*
	0x3C	31:0	S	*X_PIPE4 (缺省: 0x00000000): 5th 流水线寄存器*
	0x3D	31:0	S	*X_PIPE5 (缺省: 0x00000000): 6th 流水线寄存器*
	0x3E	31:0	S	*X_PIPE6 (缺省: 0x00000000): 7th 流水线寄存器*
	0x3F	31:0	S	*X_PIPE7 (缺省: 0x00000000): 8th 流水线寄存器*

表 85:流水线寄存器

14.22. 阴影寄存器

Shadow Register				
R/W	**地址**	**位**	**值**	**描述**
RW	0x40	31:0	S	*SH_REG0 (缺省: 0x00000000) : 第 1 个阴影寄存器*
	0x41	31:0	U	*SH_REG1 (缺省: 0x00000000) : 第 2 个 阴影寄存器.*
	0x42	31:0	U	*SH_REG2 (缺省: 0x00000000) : 第 3 个阴影寄存器*
	0x43	31:0	U	*SH_REG3 (缺省: 0x00000000) : 第 4 个阴影寄存器.*
	0x44	31:0	U	*SH_REG4 (缺省: 0x00000000) : 第 5 个 阴影寄存器*
	0x45	31:0	U	*SH_REG5 (缺省: 0x00000000) : 第 6 个阴影寄存器*
	0x46	31:0	U	*SH_REG6 (缺省: 0x00000000) : 第 7 个 阴影寄存器*
	0x47	31:0	S/U	*SH_REG7 (缺省: 0x00000000) : 第 8 个阴影寄存器*
	0x48	31:0	U	*SH_REG8 (缺省: 0x00000000) : 第 9 个阴影寄存器*
	0x49	31:0	U	*SH_REG9 (缺省: 0x00000000) : 第 10 个 阴影寄存器*
	0x4A	31:0	U	*SH_REG10 (缺省: 0x00000000) : 第 11 个 阴影寄存器*
	0x4B	31:0	U	*SH_REG11 (缺省: 0x00000000) : 第 12 个阴影寄存器*
	0x4C	31:0	U	*SH_REG12 (缺省: 0x00000000) : 第 13 个阴影寄存器*
	0x4D	31:0	U	*SH_REG13 (缺省: 0x00000000) : 第 14 个 阴影寄存器*

表 86: 阴影寄存器

14.23. 冻结寄存器

冻结寄存器仅在有效复位之后及运动之前写一次。可读。

冻结寄存器				
R/W	**地址**	**位**	**值**	**描述**
RW	0x4E	23:0	*DFREEZE (缺省: 0x000000)*	
			U	冻结事件减速度值。 如果 NFREEZE 翻转至低电平，该参数用于自动线性斜坡停止。 将 *DFREEZE* 设置为 0 对应硬停。
			U	数值表示: **频率模式**:不支持。 **直接模式:** [每时钟周期的速度变化值] a[每时钟周期的速度变化值]= *DFREEZE* / 2^{37} *DFREEZE* [pps^2] = *DFREEZE* / 2^{37} • f_{CLK}^2 **！ 设置 *DFREEZE* ≤ 2^{20}。**
		31:24	*IFREEZE (缺省: 0x00)*	
			U	NFREEZE 被拉低时的电流调节值。 如果 *IFREEZE*=0，冻结事件时实际电流调节值有效

表 87: Freeze 冻结寄存器

14.24. 复位和时钟门控寄存器

复位和时钟门控寄存器				
R/W	**地址**	**位**	**值**	**描述**
RW	0x4F	2:0	*CLK_GATING_REG (缺省: 0x0)*	
			0	时钟门控无效。
			7	时钟门控有效。
		31:8	*RESET_REG (缺省: 0x000000)*	
			0	内部没有复位。
			0x525354	内部复位有效。

表 88: 复位和时钟门控寄存器

阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

14.25. 编码器寄存器

编码器寄存器				
R/W	**地址**	**位**	**值**	**描述**
RW	0x50	31:0	*ENC_POS (缺省: 0x00000000)*	
			S	实际编码器位置 [微步]。
R	0x51	31:0	*ENC_LATCH (缺省: 0x00000000)*	
			S	锁存的编码器位置。
W			*ENC_RESET_VAL(缺省: 0x00000000)*	
			S	如果编码器位置必须清零至 0 以外的另一个值时，则设置该值为 *ENC_POS* 的复位值。
R	0x52	31:0	*ENC_POS_DEV (缺省: 0x00000000)*	
			S	*XACTUAL* 和 *ENC_POS*.之间的偏差 **i 根据不同的调节选项计算** *ENC_POS_DEV = ENC_POS - XACTUAL* (开环) *ENC_POS_DEV = ENC_POS - XACTUAL - CL_OFFSET* (闭环或者 PID)
W			*CL_TR_TOLERANCE (缺省: 0x00000000)*	***(闭环)***
			S	触发 *TARGET_ READED*(包括 *TARGET_REACHED_*Flag 和事件)的 *XACTUAL* 和 *ENC_POS* 之间的绝对容差。
W	0x53	31:0	*ENC_POS_DEV_TOL (缺省: 0xFFFFFFFF)*	
			U	*ENC_POS_DEV* 的最大容差值，位置偏差在范围内的不标记为错误。
W	0x54	30:0	*ENC_IN_RES (缺省: 0x00000000)*	
			U	连接到编码器输入端的编码器分辨率[编码器每圈的步进数]。
R			*ENC_CONST (缺省: 0x00000000)*	
			U	编码器常数. **i 数据表示: 15 整数和 16 位小数**
W		31	*manual_enc_const (缺省: 0)*	
				0 自动计算 *ENC_CONST* **1 手动定义 ENC_CONST = ENC_IN_RES**
W	0x55	31:0	*ENC_OUT_RES (缺省: 0x00000000)*	
			U	串行编码器输出接口的分辨率[编码器每圈的步进数]。

下页继续

 阅读整个文档; 是第 22 章(第 174 页)中的补充指令。

编码器寄存器				
R/W	**地址**	**位**	**值**	**描述**
W	0x56	15:0	*SER_CLK_IN_HIGH (缺省: 0x00A0)*	
			U	串行时钟输出的高电平时间[时钟周期数]。
		31:16	*SER_CLK_IN_LOW (缺省: 0x00A0)*	
			U	串行时钟输出的低电平时间[时钟周期数]。
	0x57	15:0	*SSI_IN_CLK_DELAY (缺省: 0x0000)*	
			U	**SSI 编码器:** 串行时钟输出上升沿与下一次数据传输之间的延迟时间[时钟周期数]。 **i 如果 *SSI_IN_CLK_DELAY* = 0:** ***SSI_IN_CLK_DELAY = SER_CLK_IN_HIGH***
			U	**SPI 编码器:** 在串行时钟输出和反相片选信号之间数据传输开始和结束的延迟[时钟周期数]。 **i 如果 *SSI_IN_CLK_DELAY* = 0:** ***SSI_IN_CLK_DELAY = SER_CLK_IN_HIGH***
		31:16	*SSI_IN_WTIME (缺省: 0x0F0)*	
			U	(相同数据的)多数据传输的两个时钟序列之间的延迟时间间隔 tw [时钟周期数]。 **i SSI 推荐: tw < 19 µs。**
	0x58	19:0	*SER_PTIME (缺省: 0x00190)*	
			U	**SSI 和 SPI 编码器:**请求新数据的两个连续时钟序列之间的延迟时间间隔 tp [时钟周期数]。 **i SSI 推荐: tp > 21 µs。**
	0x7D	15:0	*ENC_COMP_XOFFSET (缺省:0x0000)*	
			U	三角形补偿的水平方向起始偏移。 $0 \le ENC\_COMP\_XOFFSET < 2^{16}$
		23:16	*ENC_COMP_YOFFSET (缺省:0x00)*	
			S	三角形补偿的垂直方向起始偏移. $-128 \le ENC\_COMP\_YOFFSET \le 127$
		31:24	*ENC_COMP_AMPL (缺省:0x00)*	
			U	编码器补偿的最大幅度。

表 89: 编码器寄存器

14.26. PID &闭环寄存器

PID 和闭环寄存器				
R/W	**地址**	**位**	**值**	**描述**
RW	0x1C	8:0	*CL_BETA (0x0FF)*	
			U	闭环调节的最大换向角。 **i 仔细设置 CL_BETA > 255 (尤其当 *cl_vlimit_en* = 1 时)。** **i 为了获得最佳性能，建议设置到 255。**
		23:16	*CL_GAMMA (缺省:0xFF)*	
			U	闭环调节期间最大平衡角度，用于较高速度下补偿反电动势。
RW	0x59	31:0	*CL_OFFSET (缺省: 0x00000000)*	***(闭环操作)***
			S	闭环校准后 *ENC_POS* 和 *XACTUAL* 之间的偏移。可以通过闭环校准过程设置。也可以手动设置。
W	0x5A	23:0	*PID_P (缺省: 0x000000)*	***(PID 调节)***
			U	PID 调节器的比例 P 参数。 比例 = PID_E · *PID_P* / 256 。
W			*CL_VMAX_CALC_P (缺省: 0x000000)*	***(闭环操作)***
			U	用于控制最大追赶速度的 PI 调节器中的比例 P 参数。
R		31:0	*PID_VEL (缺省: 0x00000000)*	***(PID 调节)***
			S	实际 PID 输出速度。
W	0x5B	23:0	*PID_I (缺省: 0x000000)*	***(PID 调节)***
			U	PID 调节器的积分 I 参数。 积分 = PID_ISUM / 256 · PID_I / 256
W			*CL_VMAX_CALC_I (缺省: 0x000000)*	***(闭环操作)***
			U	用于控制最大追赶速度的 PI 调节器中的积分参数 I 参数。
R		31:0	*PID_ISUM_RD (缺省: 0x00000000)*	***(PID 调节)***
			S	实际 PID 积分器总和。更新频率 = f_{CLK}/128 。
W	0x5C	23:0	*PID_D (缺省: 0x000000)*	***(PID 调节)***
			U	PID 调节器的 **D** 参数。PID_E 采样率为 f_{CLK} / 128 / *PID_D_CLKDIV*。 微分 = (PID_E_{LAST} – PID_E_{ACTUAL}) · *PID_D*
W			*CL_DELTA_P (缺省: 0x000000)*	***(闭环)***
			U	该增益参数乘以实际位置差，用以计算保持刚度位置的实际换向角度。限幅值为 CL_BETA。实际值= CL _ DELTA _ P/2^{16}；例如:65536→ 1.0(增益=1) 数值表示:8 位数字和 16 位小数。
W	0x5D	14:0	*PID_I_CLIP (缺省: 0x0000)*	***(PID 调节) (闭环)***
			U	*PID_ISUM* 的限幅值。 实际值 = *PID_ISUM* • 2^{16} • *PID_ICLIP*
W		23:16	*PID_D_CLKDIV (缺省:0x00)*	***(PID 调节)***
			U	计算 D 部分的时钟分频器。
R		31:0	*PID_E (缺省:0x00000000)*	***(PID 调节)***
			S	实际位置偏差。
W	0x5E	30:0	*PID_DV_CLIP* (缺省:0x00000000)	***(PID 调节) (闭环)***
			U	*PID_VEL* 的限幅值
下页继续				
W	0x5F	19:0	*PID_TOLERANCE (缺省:0x00000)*	***(PID 调节)***

 阅读整个文档；是第 22 章(第 174 页)中的补充指令。

			U	位置偏差容差:如果 |*PID_E*| < *PID_TOLERANCE*， 则 *PID_E* = 0
W		7:0	*CL_TOLERANCE (缺省:0x00)*	***(闭环操作)***
			U	位置偏差容差: 如果 |*ENC_POS_DEV*| < *CL_TOLERANCE* ， 则 *CL_DELTA_P* = 65536 (增益=1)
W	0x60	23:0	*CL_VMIN_EMF (缺省:0x000000)*	***(闭环操作)***
			U	编码器速度达到该值反电动势开始补偿。
			2nd 分配: 如果启用了 dcStep，也用作 *DC_VEL* (参见 19.27.) 3rd 分配: 如果没有启用 dcStep 或闭环，也用作 *FS_VEL* (参见 19.16.)	
W	0x61	23:0	*CL_VADD_EMF (缺省:0x000000)*	
			U	速度阈值，用于计算反电动势补偿达到最大角度 *CL_GAMMA* 时的编码器速度。
		31:0	2nd 分配: 也用作 dcStep 配置寄存器 (见章节 19.27.)	
W	0x62	31:0	*ENC_VEL_ZERO (缺省:0xFFFFFF)*	
			U	最后一个增量编码器变化的延迟时间[时钟周期数]，在这个延时时间内，编码器没有变化，则设置 *V_ENC_MEAN* = 0。
W	0x63	7:0	*ENC_VMEAN_WAIT (缺省:0x00)*	***(仅增量编码器)***
			U	两个连续编码器速度值之间的延迟周期[时钟周期数]，用于计算编码器平均速度。 **！ 设置 *ENC_VMEAN_WAIT* > 32。** **i 绝对 SSI/SPI 编码器，自动设置为 SER_PTIME。**
		7:0	*SER_ENC_VARIATION (缺省:0x00)*	***(绝对编码器)***
			U	乘数因子用于计算两个连续请求的绝对编码器返回数值所允许的最大差值。 **！ 最大允许的速度= *ENC_VARIATION* / 256 • 1/8 • *ENC_IN_RES*。** **！ 如果 *ENC_VARIATION* = 0: 最大允许的值 = 1/8 • *ENC_IN_RES*。**
		11:8	*ENC_VMEAN_FILTER (缺省:0x0)*	
			U	计算编码器平均速度的滤波指数。
		31:16	*ENC_VMEAN_INT (缺省:0x0000)*	***(仅增量编码器)***
			U	编码器速度更新时间[时钟周期数]。 **i 最小速度自动设置为 256。**
		31:16	*CL_CYCLE (缺省:0x0000)*	***(绝对编码器)***
			U	闭环控制周期[时钟周期数]。 **i ABN 编码器需设置为尽可能快的周期。**
R	0x65	31:0	*V_ENC (缺省:0x00000000)*	
			S	实际编码器速度 [pps]。
	0x66	31:0	*V_ENC_MEAN (缺省:0x00000000)*	
			S	编码器滤波后的速度 [pps]。

表 90: PID 和闭环寄存器

14.27. dcStep 寄存器

Micellaneous 寄存器				
R/W	**地址**	**位**	**值**	**描述**
W	0x60	23:0	*DC_VEL (缺省:0x000000)*	***(仅 dcStep)***
			U	最小 dcStep 速度[pps]。 如果|*VACTUAL*| > *DC_VEL* 则 dcStep 有效(如果使能)。
			第二次 分配: 如果使能闭环，也用作 *CL_VMIN_EMF* (见 19.26.) 第三次分配: 如果没有使能 dcStep 或闭环，也用作 *FS_VEL* (见 19.16.)	
	0x61	7:0	*DC_TIME (缺省:0x00)*	***(仅 TMC26x 和 dcStep)***
			U	换相的 PWM 导通时间上限。 **i 设置稍微高于驱动器的有效空白时间 TBL。**
		15:8	*DC_SG (缺省:0x0000)*	***(仅 TMC26x 和 dcStep)***
			U	失步检测的最大 PWM 导通时间[时钟周期数*16]。如果检测到失步(dcStep 输入信号在消影时间后的步长小于 DC_SG)，将产生堵转事件。
		31:16	*DC_BLKTIME (缺省:0x0000)*	***(仅 TMC26x 和 dcStep)***
			U	全步有效后的消影时间[时钟周期数]，此期间不比较信号。
		23:0	2nd 分配: 如果使能闭环，也用作 *CL_ VADD _EMF*(见 19.26.)	
	0x62	31:0	*DC_LSPTM (缺省:0x00FFFFFF)*	***(仅 dcStep)***
			U	dcStep 低速定时器 [时钟周期数]
		23:0	2nd 分配: 如果禁用 dcStep，也用作 *ENC_ VEL _ZERO*(见 19.26.)	

表 91: Miscellaneous 寄存器

14.28. 转移寄存器

Transfer 寄存器				
R/W	**地址**	**位**	**值**	**描述**
W	0x68	31:0	*ADDR_TO_ENC (缺省:0x00000000)*	***(仅 SPI 编码器)***
			-	发送地址数据，以便从串行接口编码器从设备获取编码器角度数据。
				从 TMC4361A 发送到 SPI 编码器进行单次数据传输的地址数据。
W	0x69	31:0	*DATA_TO_ENC (缺省:0x00000000)*	***(SPI 编码器)***
			-	从 TMC4361A 发送到 SPI 编码器进行单次数据传输的配置数据。
R	0x6A	31:0	*ADDR_FROM_ENC (缺省:0x00000000)*	***(SPI 编码器***
			-	保存反复请求的数据。
				从 SPI 编码器接收的单次数据传输的地址响应。
R	0x6B	31:0	*DATA_FROM_ENC (缺省:0x00000000)*	***(SPI 编码器)***
			-	从 SPI 编码器接收的单次数据传输的数据响应。
W	0x6C	31:0	*COVER_LOW (缺省:0x00000000)*	
			-	通过 SPI 输出从 TMC4361A 发送至电机驱动器的 SPI 命令序列的配置位低位。
				自动覆盖数据传输(*automatic_cover* = 1): 如| *VACTUAL* |向下经过 *SPI_SWITCH_VEL* 时，*COVER_LOW* 中的值发送。 **! 设置 *COVER_DATA_LENGTH* ≤ 32.** **! 如果 *COVER_DATA_LENGTH* = 0, 不得选择 TMC21x0**
R			*POLLING_STATUS (缺省:0x00000000)*	***(TMC26x / TMC21x0 only)***
			-	TMC26x / TMC21x0 的 DRV_STATUS 响应
W	0x6D	31:0	*COVER_HIGH (缺省:0x00000000)*	
			-	通过 SPI 输出从 TMC4361A 发送至电机驱动器的 SPI 命令序列的配置位高位。
				自动覆盖数据传输(*automatic_cover* = 1): 如| *VACTUAL* |向上经过 *SPI_SWITCH_VEL* 时，*COVER_LOW* 中的值发送。 **! 设置 *COVER_DATA_LENGTH* ≤ 32.** **! 如果 *COVER_DATA_LENGTH* = 0, 不得选择 TMC21x0.**
R			*POLLING_REG (缺省:0x00000000)*	***(仅 TMC21x0)***
		19:0	-	TMC21x0 的 LOST_STEPS 响应。
		27:20	-	TMC21x0 的 PWM_SCALE 响应。
		31:28	-	TMC21x0 的 GSTAT 响应。
R	0x6E	31:0	*COVER_DRV_LOW (缺省:0x00000000)*	
			-	连接到 SPI 输出的电机驱动器返回的 SPI 响应的配置位低位。
R	0x6F	31:0	*COVER_DRV_HIGH (缺省:0x00000000)*	
			-	连接到 SPI 输出的电机驱动器返回的 SPI 响应的配置位高位。

表 92: Transfer 寄存器

14.29. SinLUT 寄存器

SinLUT 寄存器				
R/W	**地址**	**位**	**值**	**描述**
W	0x70	31:0	*MSLUT[0] (缺省:0xAAAAB554)*	
	0x71		*MSLUT[1] (缺省:0x4A9554AA)*	
	0x72		*MSLUT[2] (缺省:0x24492929)*	
	0x73		*MSLUT[3] (缺省:0x10104222)*	
	0x74		*MSLUT[4] (缺省:0xFBFFFFFF)*	
	0x75		*MSLUT[5] (缺省:0xB5BB777D)*	
	0x76		*MSLUT[6] (缺省:0x49295556)*	
	0x 77		*MSLUT[7] (缺省:0x00404222)*	
			-	**! 每个位定义微步查找表 MSLUT(与 MSLUTSEL 结合)里两个连续值之间的差值。**
W	0x78	31:0	*MSLUTSEL (缺省:0xFFFF8056)*	
			-	每四分之一 MSLUT 波内的四个分段定义。
R	0x79	9:0	*MSCNT (缺省:0x000)*	
			U	正弦波的实际微步位置。
W			2nd 分配: 如果使能电压 PWM，也用作 *MS_OFFSET* (见 0)	
R	0x7A	8:0	*CURRENTA (缺省:0x000)*	
			S	线圈 A 的实际电流值(正弦值)。
		24:16	*CURRENTB (缺省:0x0F7)*	
			S	线圈 B 的实际电流值(sine90_120 值)。
R	0x7B	8:0	*CURRENTA_SPI*	
			S	发送给电机驱动的线圈 A 的实际电流值(正弦值)。
		24:16	*CURRENTB_SPI*	
			S	发送给电机驱动的线圈 B 的实际电流值(sine90_120 值)。
W		31:0	第二次分配: 也用于写访问的 *TZERO_WAIT* (见章节 19.16.) (缺省:0x000)	
W	0x7E	7:0	*START_SIN (缺省:0x00)*	
			U	正弦波的起始值。
		23:16	*START_SIN90_120 (缺省:0xF7)*	
			U	余弦波的起始值。
		31:24	2nd 分配:也用作写访问的 *DAC_OFFSET* (见章节 19.30.)	

表 93: SinLUT 寄存器

14.30. SPI-DAC 配置寄存器

SPI-DAC 配置寄存器				
R/W	**地址**	**位**	**值**	**描述**
RW	0x1D	15:0	*DAC_ADDR_A (缺省:0x0000)*	
			U	固定命令/地址，在发送 CURRENTA_SPI 值之前通过 SPI 输出发送。
		31:16	*DAC_ADDR_B (缺省: 0x0000)*	
			U	固定命令/地址，在发送 CURRENTB_SPI 值之前通过 SPI 输出发送。
		23:0	2nd 分配:如果没使能 SPI-DAC，也用作 *SPI_SWITCH_VEL* (19.16.)	
W	0x7E	31:24	*DAC_OFFSET (缺省:0x00)*	
			U	偏移量(正弦和余弦 DAC 绝对 值)。
			S	偏移量(映射 DAC 值).
		23:0	2nd 分配:也用作可读的 *START_SIN/90_120* (see section 19.29.)	

表 94: SPI-DAC 配置寄存器.

14.31. TMC 版本寄存器

版本寄存器				
R/W	**地址**	**位**	**值**	**描述**
R	0x7F	15:0	*Version No (缺省:0x0002)*	
			U	TMC4361 版本号

表 95: 版本寄存器

15. 最大额定绝对值

任何情况下都不得超过最大额定值。应用设计应避免长时间超过或接近一个以上的最大额定值运行。

最大绝对值: 3.3V 供电				
参数(VCC = 3.3V nominal → TEST_MODE = 0V)	**符号**	**最小值**	**最大值**	**单位**
供电电压	V_{CC}	3.0	3.6	V
IO 输入电压	V_{IN}	−0.3	3.6	V

*表 96: **最大绝对值**: 3.3V 供电*

最大绝对值: 5.0V 供电				
参数(VCC = 5V nominal → TEST_MODE = 0V)	**符号**	**最小值**	**最大值**	**单位**
供电电压	V_{CC}	4.8	5.2	V
IO 输入电压	V_{IN}	−0.3	5.2	V

*表 97: **最大绝对值**: 5.0V 供电*

最大绝对值: 温度				
参数	**符号**	**最小值**	**最大值**	**单位**
温度	T	−40	125	°C

*表 98: **最大绝对值**: 温度*

16. 电气特性

除非另有规定，直流特性包含在规定电源电压范围内保证的值分布。典型值为在+25℃下测量的所有部件的平均值。温度变化也会导致偏离某些值。具有典型值的设备需工作在全温度范围的最小/最大值范围内。

直流特性						
参数	**符号**	**条件**	**最小值**	**典型值**	**最大值**	**单位**
温度范围	T_{COM}		−40°C		125	°C
内核电压	V_{DD}			1.8		V
输入输出电压	V_{DD}			3.3 / 5.0		V
输入电压	V_{IN}		0.0		3.3 / 5.0	V
输入电压低电平	V_{INL}	V_{DD} = 3.3V / 5V	−0.3		0.8 / 1.2	V
输入电压高电平	V_{INH}	V_{DD} = 3.3V / 5V	2.3 / 3.5		3.6 / 5.2	V
下拉输入		V_{IN} = V_{DD}	5	30	110	µA
上拉输入		V_{IN} = 0V	−110	−30	−5	µA
上电复位期间的 NRST 输入下拉电流		V_{IN} = V_{DD}	5	30	110	µA
输入低电流		V_{IN} = 0V	−10		10	µA
输入高电流		V_{IN} = V_{DD}	−10		10	µA
输出电压低电平	V_{OUTL}	V_{DD} = 3.3V / 5V			0.4	V
输出电压高电平	V_{OUTH}	V_{DD} = 3.3V / 5V	2.64 / 4.0			V
输出驱动强度	I_{OUT_DRV}	V_{DD} = 3.3V / 5V		4.0		mA

表 99: 直流特性

16.1. 功耗

功耗						
参数	**符号**	**条件**	**最小值**	**典型值**	**最大值**	**单位**
静态功耗	PD_{STAT}	所有输入为 VDD 或 GND V_{DD} = 3.3V / 5V			1.1 / 1.7	mW
动态功耗	PD_{DYN}	所有输入为 VDD 或 GND 动态 f_{CLK} V_{DD} = 3.3V / 5V			2.7 / 4.0	mW / MHz
总功耗	PD	f_{CLK} = 16 MHz V_{DD} = 3.3V / 5V			44.3 / 65.7	mW

表 100: 功耗

16.2. 通用 IO 时序参数

通用 IO 时序参数						
参数	**符号**	**条件**	**最小值**	**典型值**	**最大值**	**单位**
操作频率	f_{CLK}	$f_{CLK} = 1 / t_{CLK}$	4.2[1)]	16	30	MHz
时钟周期	t_{CLK}	上升沿至上升沿	33.5	62.5		ns
时钟低电平时间			16.5			ns
时钟高电平时间			16.5			ns
CLK 输入信号上升时间	t_{RISE_IN}	20 % 到 80 %			20	ns
CLK 输入信号下降时间	t_{FALL_IN}	80 %到 20 %			20	ns
输出信号上升时间	t_{RISE_OUT}	20 %到 80 % 负载 32 pF		3.5		ns
输出信号下降时间	t_{FALL_OUT}	80 %到 o 20 % 负载 32 pF		3.5		ns
同步设计中 SPI 输入信号的建立时间	t_{SU}	相对于 clk 上升沿	5			ns
保持时间	t_{HD}	相对于 clk 上升沿	5			ns
外部时钟下的 SCKIN 频率	t_{SCK}	假设同步 CLK			f_{CLK} / 4	MHz
外部时钟下的编码器接口引脚频率(A_SCLK, ANEG_NSCLK, B_SDI, BNEG_NSDI, N, NNEG)	t_{ENC}				f_{CLK} / 4	MHz
外部时钟下的参考输入引脚频率 (STOPL, HOME_REF, STOPR, STPIN, DIRIN, START)	t_{REF}				f_{CLK} / 4	MHz

表 101: 通用 IO 时序参数

[1)] f_{CLK} 的下限是指内部单位换算对物理单位的限制。芯片也将在较低的频率下工作。

16.3. 布线示例

16.3.1. 示例的内部电路图

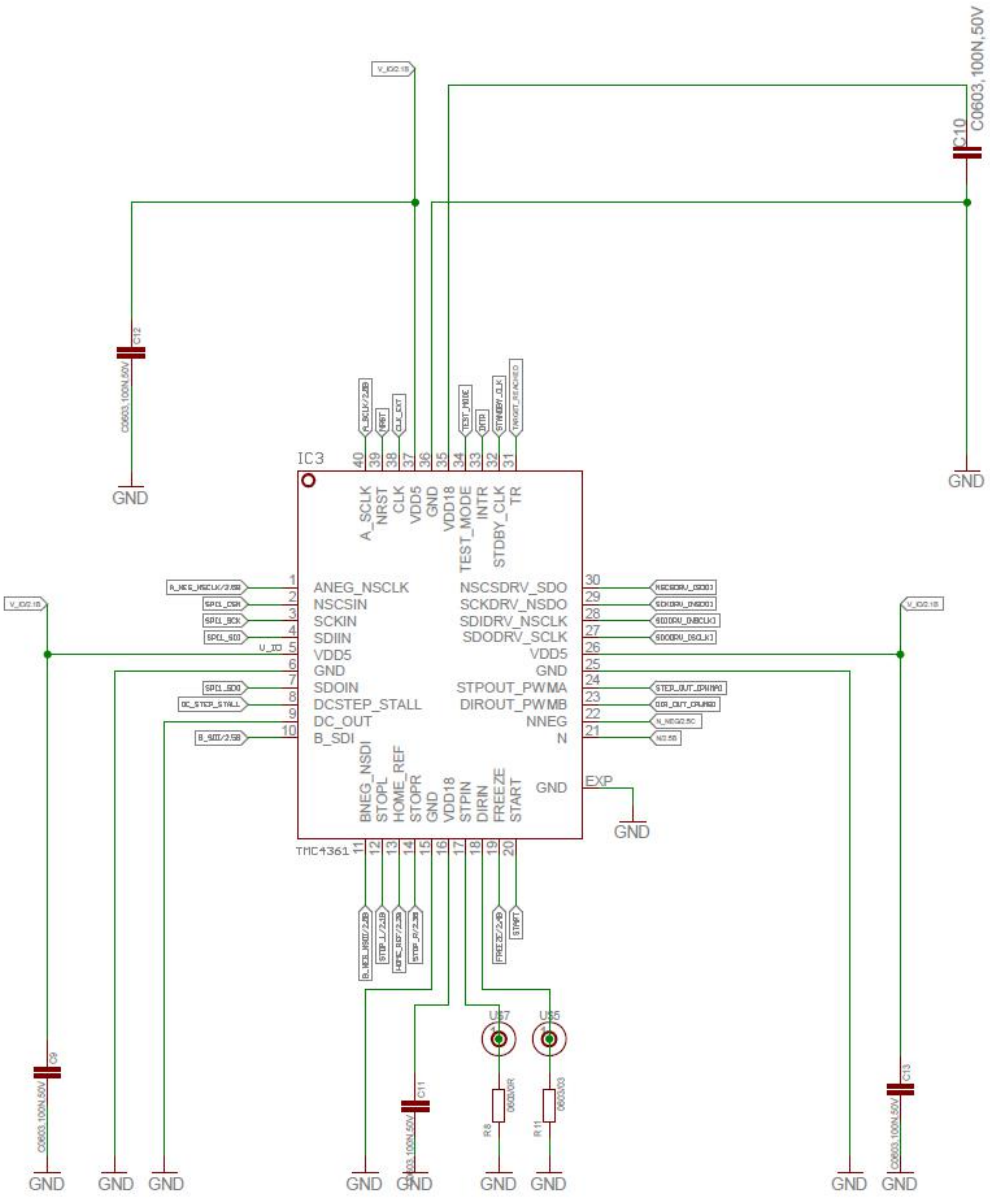


图 73: 布局示例的内部电路图

16.3.2. 带编码器应用的元器件焊接

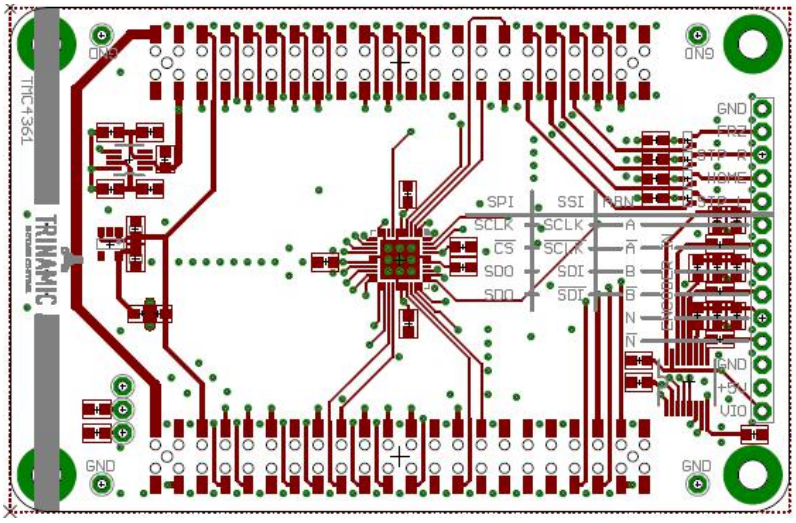


图 74: 用于编码器应用的组件装配

16.3.3. 顶层:焊接层

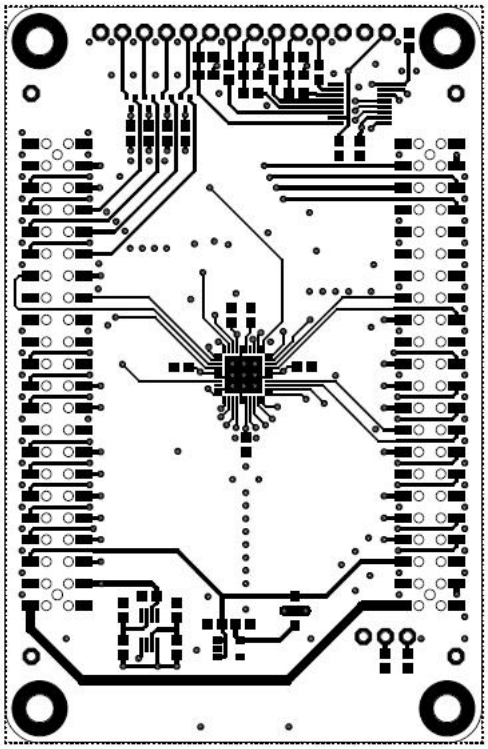


图 75: 顶层:焊接层

16.3.4. 中间层(GND)

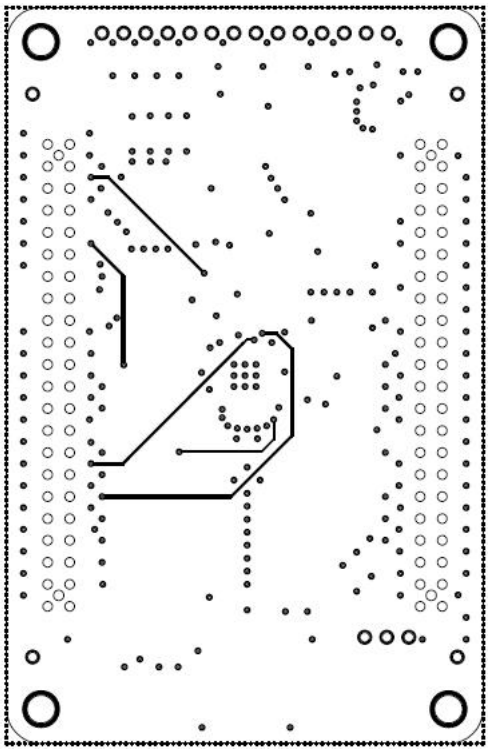


图 76: 中间层(GND)

16.3.5. 中间层(VS 供电)

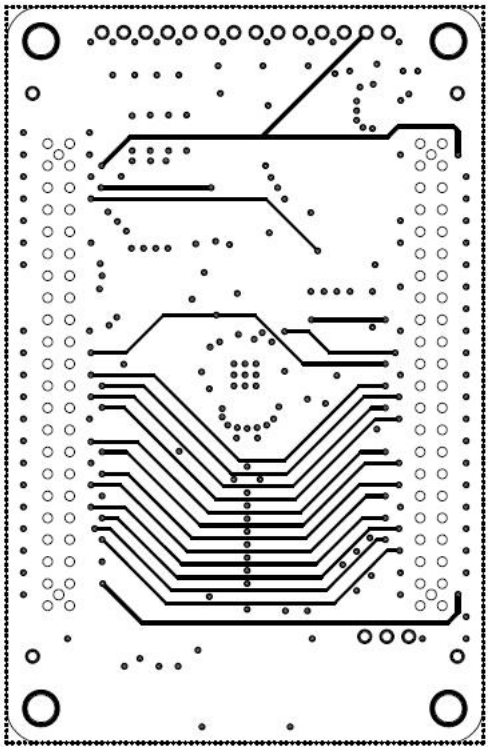
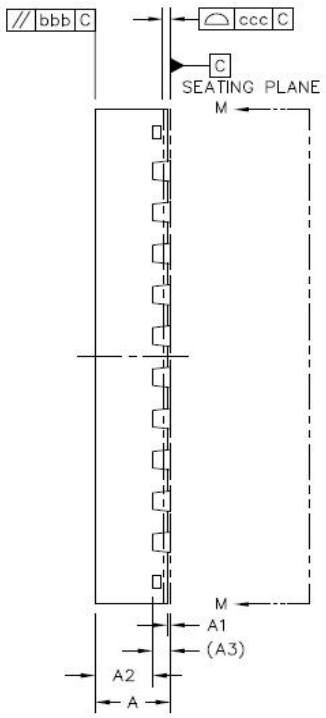


图 77: 中间层(VS 供电)

16.4. 封装尺寸



封装尺寸				
参数	Ref	最小	标称	正常
总厚度	A	0.8	0.85	0.9
基准	A1	0	0.035	0.05
Mold 厚度	A2	-	0.65	0.67
引脚框厚度	A3	0.203 REF		
引脚宽度	b	0.2	0.25	0.3
主体尺寸 X	D	6 BSC		
主体尺寸 Y	E	6 BSC		
引脚间距	e	0.5 BSC		
芯片底部裸露焊盘尺寸 X	J	4.52	4.62	4.72
芯片底部裸露焊盘尺寸 Y	K	4.52	4.62	4.72
引脚长度	L	0.35	0.4	0.45
封装边缘容差	aaa	0.1		
Mold 平整度	bbb	0.1		
同面性	ccc	0.08		
引脚偏移量	ddd	0.1		
芯片底部裸露焊盘偏移量	eee	0.1		

表 102: 封装尺寸

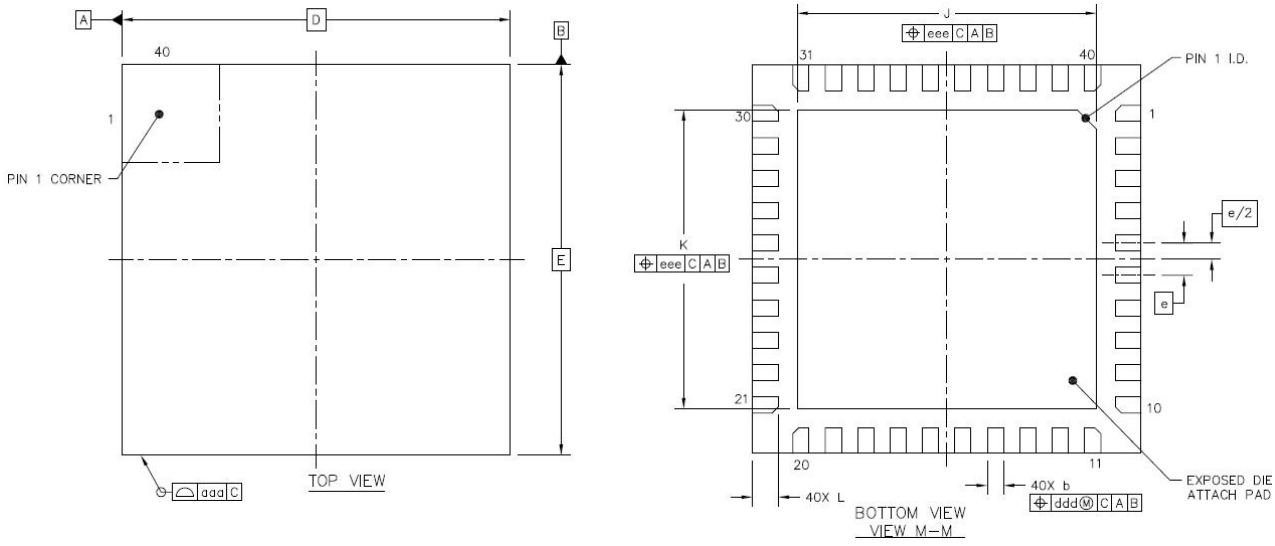


图 78: 尺寸图

16.5. 包装材料信息

请参考相关文件"TMC43xx *Package Material Information,* 1.00 版"，了解可用包装尺寸以及各种托盘和卷轴包装选项。本文档告知您每个托盘和/或卷轴的外部尺寸以及每个托盘/卷轴的集成电路数量。它还提供可用包装单位及其重量的信息，以及外包装的箱尺寸和重量细节。

该文档可在 TMC4361A 产品页面下载，网址为 www.trinamic.com。

i 如果您需要定制的组件包装解决方案或不同的外包装解决方案，或者对组件包装选择有疑问，请联系我们的客户服务。

注意:

→ *我们的托盘和卷轴是 JEDEC-兼容*

16.6. 芯片上的标记细节

单芯片上提供的标记现实:

1. Trinamic
2. 产品代码
3. 日期代码
4. 版权所有者的位置，
   TRINAMIC 德国汉堡。
5. 批号

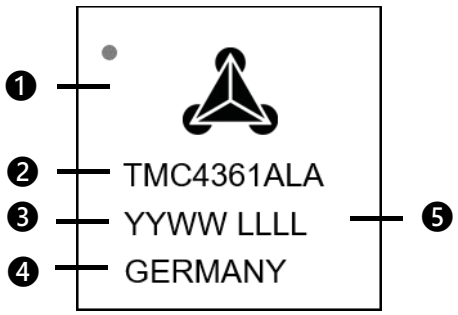


图 79: 芯片上标记 [1]

[1] 所提供的图像不是原始产品的精确再现，而只是作为例证。

附录

17. 入门指南

TMC4361A 设计了很好的功能实现运动控制及驱动 TMC 步进电机驱动器，下面的解释如何设置运动控制器及相关的 TMC 步进驱动器。

另请参考本章中提供的驱动芯片的数据手册和评估板设置文档。

i 建议根据应用功能优化调整电机驱动器参数达到优化应用性能的目的。尤其是每一个 CS 值或 IHOLD/IRUN！

17.1. TMC4361A 连接 TMC2130 或 TMC2160

下图概述了 TMC4361A 和 TMC21x0 之间的芯片间连接及连接微控制器的其他数字引脚。在接下来的章节中，将简要介绍从微控制器到 TMC4361A 的 SPI 数据报，以初始化第一个操作。更多详情，请参考章节 10.8. 的第 89 页。

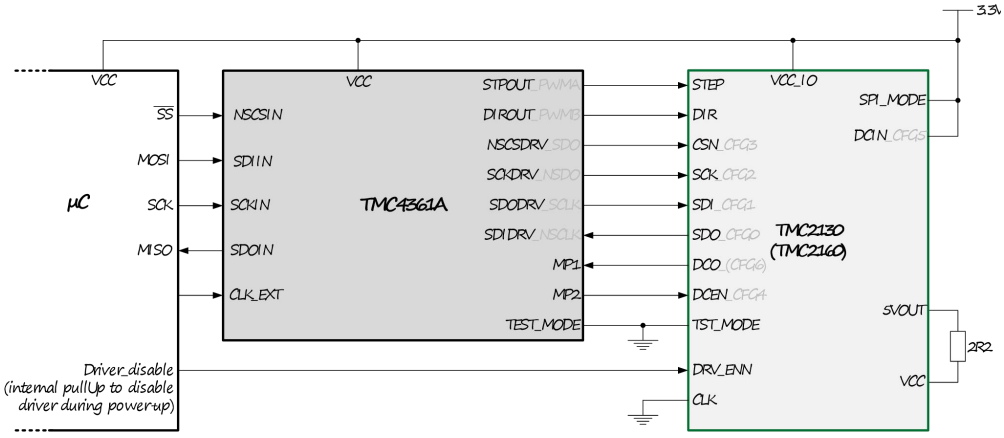


图 80: TMC4361A 的 S/D 模式连接 TMC2130 或者. TMC2160

17.1.1. SPI 模式和 spreadCycle 的初始设置

建议 TMC2130 或 TMC2160 工作在 S/D 模式，闭环操作期间同样如此。无论如何，为了在 S/D 模式下使用 8 位电流调节值而不是 5 位，必须打开从 TMC4361A 传输的 SPI 数据，电流数据报分别直接传输到 TMC2130 或者 TMC2160。

i 在这种模式下，不能用 TMC2130 的 stealthChop2 和 TMC2160 的 stealthChop2 的自动调节功能。

使能已连接的 TMC21x0 步进驱动器的 SPI 接口数据传输模式，并设置 spreadCycle 斩波器算法，请执行以下操作：

操作步骤:

- 将 0x4440128D 发送至 *SPI_OUT_CONF* 寄存器 0x04。
- 发送 0x80 至 *COVER_HIGH* 寄存器 0x6D，发送 0x00010000 至 *COVER_LOW* 寄存器 0x6C。
- 发送 0xEC 至 *COVER_HIGH* 寄存器 0x6D，发送 0x000100C3 至 *COVER_LOW* 寄存器 0x6C。
- 发送 0x90 至 *COVER_HIGH* 寄存器 0x6D，发送 0x00000A0A 至 *COVER_LOW* 寄存器 0x6C。
- 根据章节 11 第 120 页设置当前开环电流调节值，或使用第 16.3 节的第 161 页的闭环功能，包括闭环电流调节。

结果:

选择 TMC21x0 的 SPI 模式下 (包括重复的覆盖数据报；POLL_BLOCK_EXP = 2；使能 *cover_done_oly_for_cover*)，它将电流数据报直接写入 TMC21x0 的 XDIRECT 寄存器 0x2D。TMC21x0 的 CHOPCONF 寄存器是 spreadCycle 的斩波设置:TOFF=3；HSTRT=4；HEND=1；TBL=2；CHM=0。因为电流调节本身由 TMC4361A 处理，所以设置 TMC21x0 的电流值相等（**IRUN** = **IHOLD** = 10)，这标志着最大可访问值。请根据**您的要求进行调整**。

17.1.2. 切换斩波算法的初始设置

TMC2130 或 TMC2160 除了分别提供的高精度斩波算法 spreadCycle 之外，这些步进驱动器包含无噪声、高精度斩波算法 stealthChop 或者 stealthChop2。下一个设置是根据 TMC21x0 步进输入端或者 TMC4361A 的 STPOUT 引脚的实际步进频率，动态切换两种斩波算法。

使能已连接的 TMC21x0 步进驱动器的 S/D 传输模式，并分别设置 stealthChop 或 stealthChop2 到 spreadCycle 的切换，执行以下操作:

操作步骤:

- 发送 0x4440128C to *SPI_OUT_CONF* 寄存器 0x04。
- 发送 0xEC 到 *COVER_HIGH* 寄存器 0x6D 及 0x000100C3 到 *COVER_LOW* 寄存器 0x6C。
- 发送 0x90 到 *COVER_HIGH* 寄存器 0x6D 及 0x00061F0A 到 *COVER_LOW* 寄存器 0x6C。
- 发送 0x91 到 *COVER_HIGH* 寄存器 0x6D 及 0x0000000A 到 *COVER_LOW* 寄存器 0x6C。
- 发送 0x80 到 *COVER_HIGH* 寄存器 0x6D 及 0x00000004 到 *COVER_LOW* 寄存器 0x6C。
- 发送 0x93 到 *COVER_HIGH* 寄存器 0x6D 及 0x000001F4 到 *COVER_LOW* 寄存器 0x6C。
- **仅 TMC2130:**
  发送 0xF0 至 *COVER_HIGH* 寄存器 0x6D，发送 0x000401C8 至 *COVER_LOW* 寄存器 0x6C.

结果:

选择 TMC21x0 的 S/D 模式(包括重复的覆盖数据报；*POLL_BLOCK_EXP* = 2；使能 *cover_done_only_for_cover*)。

然后，TMC21x0 的 CHOPCONF 寄存器将 spreadCycle 初始化如下:TOFF = 3；HSTRT = 4；HEND = 1；TBL = 2；CHM = 0。

运行电流设置为最大值(**IRUN** = 31)， **IHOLD 设置为** 10。IHOLDDELAY 设置为 6。请**根据您的应用进行调整**

TMC21x0 设置从停止到电流下降的延迟 TPOWERDOWN 为 10。对应约 0.22 秒的延迟，因为 TMC21x0 的 CLK 输入连接到第 174 页图 80 中的 GND，对应约 12Mhz 的内部时钟。

最后，设置 TMC21x0 的 TPWMTHRS 阈值寄存器(= 500)使能 stealthChop，这产生约 30 rpm 的步进输入频率阈值。低于此频率时， stealthChop (TMC2130)或 stealthChop2 (TMC2160)斩波器使能。高于此频率时，spreadCycle 斩波器有效。

此外，TMC2130 的 PWMCONF 必须设置为 stealthChop:例如，AUTO = 1, 2/1024 fclk，开关幅度限值 limit = 200，梯度 Grad = 1。对于 TMC2160，PWMCONF 寄存器的默认值即可。

 阅读整个文档；是第 22 章(第 174 页)中的补充指令。

17.2. TMC4361A 连接 TMC26x

下图显示了 TMC4361A 和 TMC26x 之间的芯片间连接及连接微控制器的其他数字引脚。在下一节中，简要介绍了从微控制器到 TMC4361A 的 SPI 数据报，以初始化第一个操作。更多详情，请参见章节 10.6.第 107 页。

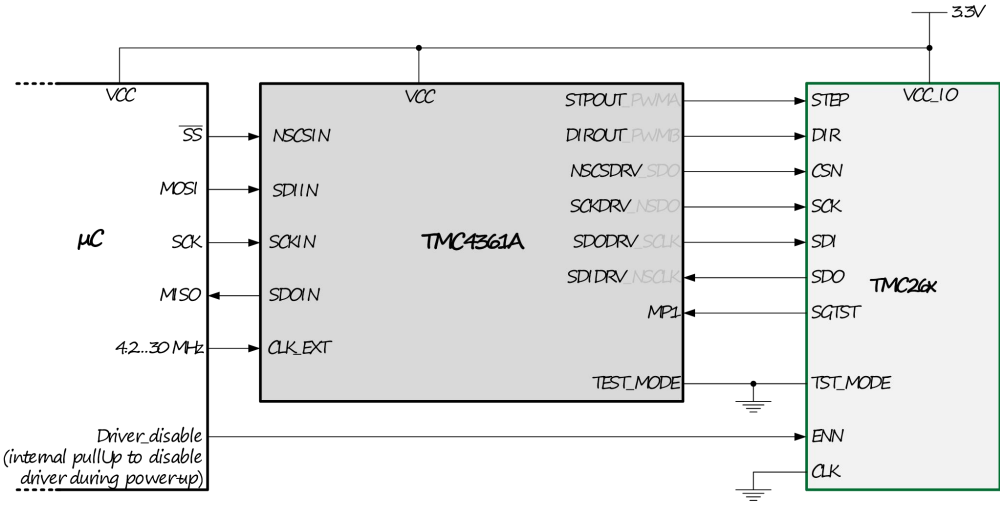


图 81: TMC4361A SPI 模式连接 TMC26x 驱动器

17.2.1. SPI 模式和扩 spreadCycle 斩波器的初始设置

建议 TMC26x 在 SPI 模式下工作，尤其是在闭环工作期间，利用 8 位电流比例，而不是 S/D 模式下的 5 位电流比例。

使能已连接 TMC26x 步进驱动器的串行接口数据传输模式，并设置 spreadCycle 斩波器算法，请执行以下操作:

操作步骤:

- 将 0x4440108A 发送至 *SPI_OUT_CONF* 寄存器 0x04。
- 将 0x000900C3 发送至 *COVER_LOW* 寄存器 0x6C。
- 将 0x000A0000 发送至 *COVER_LOW* 寄存器 0x6C。
- 将 0x000C000A 发送至 *COVER_LOW* 寄存器 0x6C。
- 将 0x000E00A0 发送到 *COVER_LOW* 寄存器 0x6C
- 根据章节 11 第 120 页设置当前开环电流调节值，或使用章节 16.3 第 161 页的闭环功能，包括闭环电流调节值。

结果:

选择 TMC26x 的 SPI 模式(包括重复的覆盖数据报；*使能 cover_done_only_for_cover*)，将电流数据报直接写入 TMC26x 的 DRVCTRL 寄存器。

TMC26x 的 CHOPCONF 寄存器设置 spreadCycle 斩波:TOFF = 3；HSTRT = 4；HEND = 1;TBL = 2；CHM = 0。未使能 TMC26x 的 coolStep 功能。

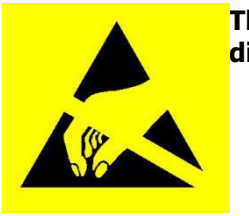
因为电流调节过程由 TMC4361A 处理，所以 TMC260 的电流调节值 CS 在这里被设置为标记最大可访问值的 CS = 10。**请根据您的应用进行调整。VSENS**E 也应进行调整，因为该值直接影响电机电流。它在最后一个覆盖数据报中被设置为 1。

最后，SDOFF 设为 1，使能 TMC26x 的 SPI 模式。。

阅读整个文档: 是第 22 章(第 174 页)中的补充指令。

18. Supplemental Directives

ESD-DEVICE INSTRUCTIONS



This product is an ESD-sensitive CMOS device. It is sensitive to electrostatic discharge.

- Provide effective grounding to protect personnel and machines.
- Ensure work is performed in a nonstatic environment.
- Use personal ESD control footwear and ESD wrist straps, if necessary.

Failure to do so can result in defects, damages and decreased reliability.

Producer Information

The producer of the product TMC4361A is TRINAMIC GmbH & Co. KG in Hamburg, Germany; hereafter referred to as TRINAMIC. TRINAMIC is the supplier; and in this function provides the product and the production documentation to its customers.

Copyright

TRINAMIC owns the content of this user manual in its entirety, including but not limited to pictures, logos, trademarks, and resources.

Trademark Designations and Symbols

Trademark designations and symbols used in this documentation indicate that a product or feature is owned and registered as trademark and,'or patent either by TRINAMIC or by ather manufacturers, whose products are used or referred to in combination With TRINAMlC's products and TRINAMlC's product documentation. This documentation is a noncommercial publication that seeks to provide concise scientific and technical user information to the target user. Thus, we only enter trademark designations and symbols in the Short Spec of the documentation that introduces the product at a quick glance. We also enter the trademark designation 'symbol when the product or feature name occurs for the first time in the document. All trademarks used are property of their respective owners.

Target User

The documentation provided here, is for programmers and engineers only, who are equipped with the necessary skills and have been trained to work with this type of product.

The ***Target User*** knows how to responsibly make use of this product without causing harm to himself or others, and without causing damage to systems or devices, in which the user incorporates the product.

Disclaimer: Life Support Systems

TRINAMIC Motion Control GmbH & Co. KG does not authorize or warrant any of its products for use in life support systems, without the specific written consent of TRINAMIC Motion Control GmbH & Co. KG.

Life support systems are equipment intended to support or sustain life, and whose failure to perform, when properly used in accordance with instructions provided, can be reasonably expected to result in personal injury or death.

Information given in this document is believed to be accurate and reliable. However, no responsibility is assumed for the consequences of its use nor for any infringement of patents or other rights of third parties which may result from its use. Specifications are subject to change without notice.

Disclaimer: Intended Use

The data specified in this user manual is intended solely for the purpose of product description. No representations or warranties, either express or implied, of

merchantability, fitness for a particular purpose or of any other nature are made hereunder with respect to information/specification or the products to which information refers and no guarantee with respect to compliance to the intended use is given.

In particular, this also applies to the stated possible applications or areas of applications of the product. TRINAMIC products are not designed for and must not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death (Safety-Critical Applications) without TRINAMIC's specific written consent.

TRINAMIC products are not designed nor intended for use in military or aerospace applications or environments or in automotive applications unless specifically designated for such use by TRINAMIC. TRINAMIC conveys no patent, copyright, mask work right or other trade mark right to this product. TRINAMIC assumes no liability for any patent and/or other trade mark rights of a third party resulting from processing or handling of the product and/or any other use of the product.

Product Documentation Details

This document ***Datasheet User Manual*** contains the ***User Information*** for the ***Target User***.

The ***Short Spec*** forms the preface of the document and is aimed at providing a general product overview. The Main Manual contains detailed product information pertaining to functions, and configuration settings. It contains all other pages of this document.

Collateral Documents & Tools

This product documentation is related and/or associated with additional tool kits, firmware and other items, as provided on the product page at: www.trinamic.com .

19. 表格目录

20. 框图目录

21. Revision History

Document Revision History			
Version	**Date**	**Author**	**Description**
1.00	2014-APR-11	HS, SD	First complete version. New release variant, which is a product upgrade of TMC4361.
1.10	2016-JUL-20	HS, SV	New chapter organization with additional information. Specifically for: • Chapter 17, page 170. • Chapter 18: page 错误！未定义书签。 . New Layout, ANSI-compliant safety notices.
1.20	2016-NOV-10	HS	Repair of references • **Maximum velocity, acceleration and bow values changed! Section 6.7.5, page 44**
1.21	2016-NOV-25	HS	Adaption of register overview that is now more arranged according to features.
1.22	2017-JAN-12	HS	1) Section 2.5. , page 14, added: TMC5130A and TMC5160 are software compatible from TMC4361A point of view. 2) Default settings for IO ports added.
1.23	2018-JUN-22	HS	1) 图 7, page 12, corrected: SDO and NSDO of serial encoder output have been changed. 2) Default settings for IO ports added. 3) Added a NOTICE in section 10.6.4, page 108, to transfer manual cover datagrams safely in combination with automatically repeated cover datagrams for TMC26x. 4) Added a "Special Area of concern" for the TMC2130 SPI mode (section 10.8.1, page 89) to use the stealthChop and spreadCycle feature of TMC2130. 5) Added a NOTICE in section **10.8.4**, page 113, to transfer manual cover datagrams safely in combination with automatically repeated cover datagrams for TMC2130. 6) Section 19.29. , page 164: Added default value for *TZERO_WAIT* register (default = 0) 7) Table 99: Information about NRST pull-down current during Power-On-Reset added 8) Table101: Information for maximum input pins frequency added 9) Calculations for *ENC_POS_DEV* in section 错误！未找到引用源。added
1.24	2018-SEP-20	HS	1) TMC2160 included in the datasheet. Interfacing and controlling is similar to TMC2130! If both drivers should be addressed, the abbreviation TMC21x0 is used now. 2) Added section 15.3.3,page 147, for detection of N events. 3) Added "Getting started" chapter 22, page 225, to faciliate first operational steps. 4) Updated section 16.3.2, page 118. closed loop operation enabling resp. calibration routines clarified and completed. 5) Order code update on page 2: Tape option added. 6) Added section 7.2. , page 错误！未定义书签。, to point out options to manipulate *VACTUAL* during external step control

Table 103: Document Revision History

完整 TMC4361 资料请参阅英文手册：https://www.trinamic.com/products/integrated-circuits/details/tmc4361a-la/

关注 TRINAMIC 微信公众号了解更多信息