6609具有下述的重要特征，步进电机的运动会更加精密，高效，可靠，平滑以及节能。

静音模式：运动与静止时无噪声，高精度。运行更快的电机加速与减速，并静态电机电流更低。

快速模式：高精度，闭环电流控制方式，可以达到最高的动态性能。

微细分控制：微步插值法得到256细分的内部微步进，外部只需要低精度步进控制，从整步起。另外出来这些功能，6609还集成了输出短接保护，过温保护，欠压保护等来增强芯片的高可靠性。

1.1 控制接口

基础模式采用离散控制线，UART模式信号线接口带有CRC校验功能。当发送正确的UART数据时，UART接口自动变为有效。

1.1.1 UART接口

UART接口允许波特率从9600波特到500k波特，甚至更高（使用外部时钟）。6609能够自动适应主机波特率。芯片有固定的从机地址，在没有读寄存器操作的情况下，多个从机可以并联在一起。模拟多路复用器可以提供任意的从机地址，比如74HC4066。

1.2 电机的运动与控制

1.2.1 STEP/DIR接口

电机运动通过步进和方向输入控制。一个特殊位（DEDGE）控制步进的有效沿是上升沿或者双沿。双沿触发用在通信过慢的接口，比如光隔离接口。每个STEP可以是全步进和微步进，一个全步进可以等于2,4,8,16,32,64,128,256个微步进。内部表格转化为正弦和余弦的值，控制电机电流。

1.2.2 内部STEP脉冲发生器

在一些不要求精确的坐标运动，只要求达到精确的时间和速度，6609有一个内部STEP脉冲发生器来满足这个要求：只要通过UART接口配置速度就可以使电机运动。速度信号自动控制运　　动方向，只是它没有斜坡修整功能。在较高速度下的运动将需要通过软件来提高和降低速度值。

STEP/DIR模式和内部脉冲发生器模式可以在一个应用中混合使用

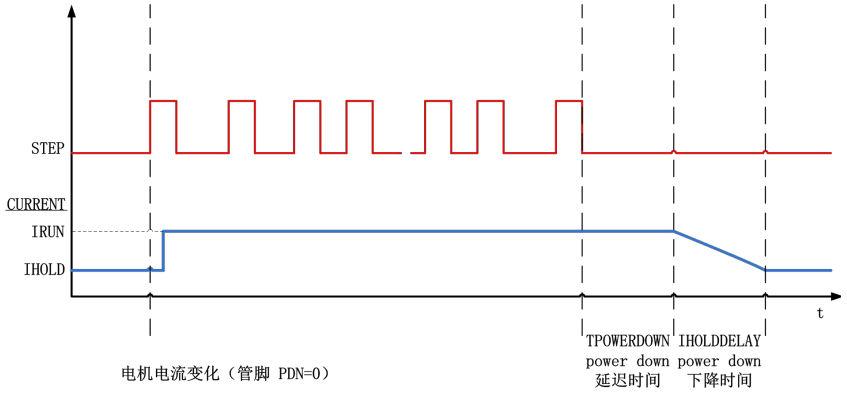
1.3 静音模式与快速模式

静音模式基于电压控制原理，保证电机在静止与低速时绝对安静，当然电机的轴承噪声除外。与其他的电压模式不同的是，它在电源启动的第一个动作后，学习了最好的设置，并且在后续运动中使用这个设置。初始的参数可以写入并且存贮在OTP寄存器中。静音模式通过对电机速度的变化立即作出反应，从而实现了电机的高动态特性

高速应用中，快速模式比静音模式会更加合适，可以通过UART或者OTP来切换模式。这两种模式还可以组合应用。快速模式是一种增强型的电流反馈模式，它可以提供平滑的操控以及共鸣　　抑制，包含很宽的速度与负载范围。快速模式斩波器可以自动调整快衰减周期来保证平滑的过零　　点。

1.4 自动停转电流降低

自动电流衰减大大减小了功耗。非UART模式下，通过下拉PDN_UART管脚使能停转电流衰减　　功能。它在运行电流为50%左右的时候可以把功耗降到33%　。



电机电流变化（管脚 PDN=0）

1.5 精准时钟产生器与时钟输入

6609提供一个精准的内部时钟发生器确保斩波器频率和性能稳定。然而，当要求更高的　　晶振精密度或者更高和更低的频率时，则需要用外部时钟。为安全起见，时钟输入具有超时检测　　功能，并在外部源故障时切换回内部时钟。

1.6 FLAG2输出

FLAG2　转一圈给出一个脉冲，即每四个整步一个脉冲。它显示内部定序器微步 0　位置（MSTEP接近 0）。再结合机械式原点开关，可以实现更精确的归位。

二. Uart 单总线接口

UART 单线接口允许 6609 被任何微控制器控制。它像 RS485 基本接口一样共用发送线和接收线。使用 CRC 冗余校验使数据传输更安全可靠，因此增加接口距离（两个 PCB 板之间的线），可以避免出现错误，甚至由于磁干扰发生失控。自动波特率检测使接口使用更加方便。

2.1 数据结构

2.1.1 写寄存器

<table>
<tr><th colspan="24">UART 写寄存器数据结构</th></tr>
<tr><td colspan="24">每个字节从 LSB...MSB,高字节先传输</td></tr>
<tr><td colspan="24">0...63</td></tr>
<tr><td colspan="8">同步+预留</td><td colspan="4">8 位从机地址</td><td colspan="5">读写位+7 位寄存器地址</td><td colspan="6">32 位数据</td><td colspan="3">CRC</td></tr>
<tr><td colspan="8">0...7</td><td colspan="4">8...15</td><td colspan="5">16...23</td><td colspan="6">24...55</td><td colspan="3">56...63</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td colspan="4">预留（无需注意）</td><td colspan="4">从机地址=0</td><td colspan="4">寄存器地址</td><td>1</td><td colspan="6">数据字节 3,2,1,0（高到低）</td><td colspan="3">CRC</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>...</td><td>15</td><td>16</td><td>...</td><td>23</td><td>24</td><td>...</td><td>55</td><td>56</td><td>...</td><td>63</td></tr>
</table>

一个同步半字节在6609的每次传输最前面并且被装入第一个字节，接着跟着一个从机地址字节（6609为0）。每次传输允许内部波特率同步器同步到主机时钟频率。实际波特率可以适应内部时钟频率的变化，因此波特率可以在有效范围内自由选择。每个传输字节开始于一个起始位（逻辑 0），结束于与一个结束位（逻辑 1）。位时间通过测量起始位的开始（1到0的转变）到同步桢的结束（第二位到第三位1到0的转变）来计算。所有数据按字节被传输，32位数据从高字节开始传输。在20M系统时钟下，允许最小波特率为 9000，最大波特率为 fclk/1。6609 的从机地址 SLAVE 地址总为 0。

如果两个连续起始位之间的暂停时间超过63位的时间，通信将被复位。这个时序条件是上一次传输数据是正确的，在这种情况下，传输再开始需要至少12位的故障恢复时间。这个电路允许主机复位通信，一旦传输出现错误，任何小于16个时钟的脉冲会被视为干扰信号并导致一个12位的暂停时间。其他错误，比如 CRC错误也做同样处理。在任何传输错误之后允许再次同步。

UART 线在总线空闲状态下必须置高，因此待机功能在数据传输时不能通过 PDN_UART 管脚使能。在 UART 接口的应用中，通过寄存器设置 pdn_disable 位来禁止 PDN_UART 管脚的功能。

2.1.2 读寄存器

UART 读寄存器数据结构																			
每个字节从 LSB...MSB,高字节先传输																			
0...63																			
同步+预留								8 位从机地址			读写位+7 位寄存器地址					CRC			
0...7								8...15			16...23					24...31			
1	0	1	0	预留（无需注意）				从机地址=0			寄存器地址				0	CRC			
0	1	2	3	4	5	6	7	8	…	15	16	…			23	24	…		31

读寄存器要求的数据结构和写寄存器一样，但是需要的位数更少。它的功能是寻址从机地址和寄存器地址来实现读取功能。6609 用与主机相同的波特率作出应答。

为了确保主机到从机的总线传输不受干扰，6609不能马上发送应答。但是它在第一个应答字节发送以后使用一个可编程的延时时间，这个延时时间根据主机的需要通过设置寄存器SENDDELAY的值，时间为 8 位的倍数。

UART 读应答数据结构																			
每个字节从 LSB...MSB,高字节先传输																			
0...63																			
同步+预留								8 位主机地址			读写位+7 位寄存器地址			32 位数据			CRC		
0...7								8...15			16...23			24...55			56...63		
1	0	1	0	预留（无需注意）				主机地址=0xFF			寄存器地址		0	数据字节 3,2,1,0（高到低）			CRC		
0	1	2	3	4	5	6	7	8	…	15	16	…	23	24	…	55	56	…	63

读响应发送地址代码%11111111 给主机，传输在最后一位发送完以后四位的时间转换为无效。

2.2 CRC 运算

8位的CRC多项式用来检查读写寄存器。CRC8-ATM多项式初始值为零，应用中从低位到高位，包括同步和地址字节。同步半字节被假设为总是正确的，6609只响应正确的数据电报， 内部还加了一个计数器用来计数正确的写寄存器次数。

$$CRC=x^8+x^2+x^1+x^0$$

连续运算例子

CRC =（CRC<<1）OR (CRC.7 XOR CRC.1 XOR CRC.0 XOR [new incoming bit])

CRC 计算 C 语言例子:

For example:

```
//6609 Serial communication verification algorithm
//CRC calculation
//CRC = (CRC << 1) OR (CRC.7 XOR CRC.1 XOR CRC.0 XOR [new incoming bit])
//CRC = x8 + x2 + x1 + x0
void swuart_calcCRC(uint8_t* datagram, uint8_t datagramLength)
{
    int i,j;
    uint8_t* crc = datagram + (datagramLength-1); // CRC located in last byte of message
    uint8_t currentByte;
    *crc = 0;

    for (i=0; i<(datagramLength-1); i++)
    { // Execute for all bytes of a message
        currentByte = datagram[i]; // Retrieve a byte to be sent from Array
        for (j=0; j<8; j++)
        {
            if ((*crc >> 7) ^ (currentByte&0x01)) // update CRC based result of XOR
operation
            {
                *crc = (*crc << 1) ^ 0x07;
            }
            else
            {
                *crc = (*crc << 1);
            }
            currentByte = currentByte >> 1;
        } // for CRC bit
    } // for message byte
}   // END
```

2.3 UART 信号

6609 中采用了一个双向引脚

UART 接口信号	
HOLDEN(PIN14)	非反向输入和输出。

芯片检查PDN_UART(PIN14)引脚的输入，确保接收正确的数据报文。它可以依靠同步半字节适应输入数据波特率。读寄存器时，引脚变为输出状态，用相同的波特率发送响应，最后一位传输完以后 延迟四位时间，引脚状态变为输出关闭。

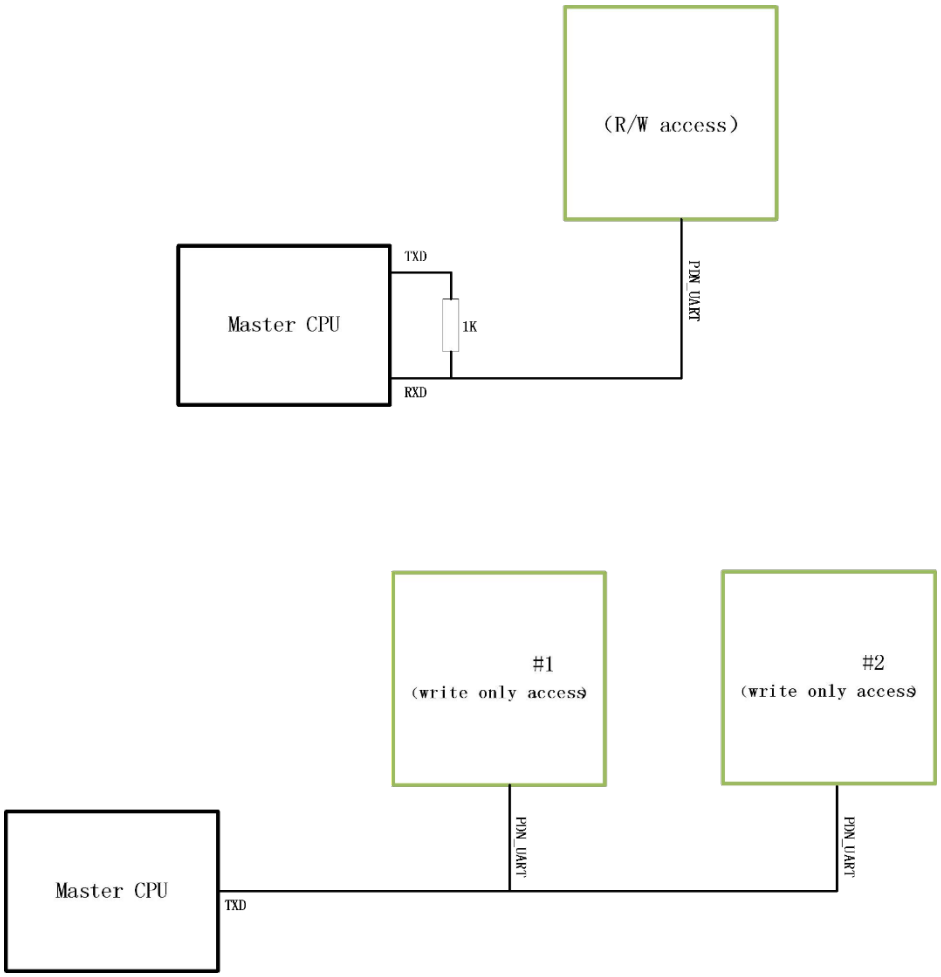


图 2.1 微控制器控制 6609

2.4 多从机地址

只写寄存器：

如果没有用到读寄存器，并且所有从机被赋予相同的初始值，没有地址要求。所有从机并联在一起。（图2.1）

多从机地址：

6609使用一个固定的从机地址，原则上每个UART接口通道只能连接一个芯片。通过添

加模拟开关可以让多个芯片独立工作。（图 2.2）

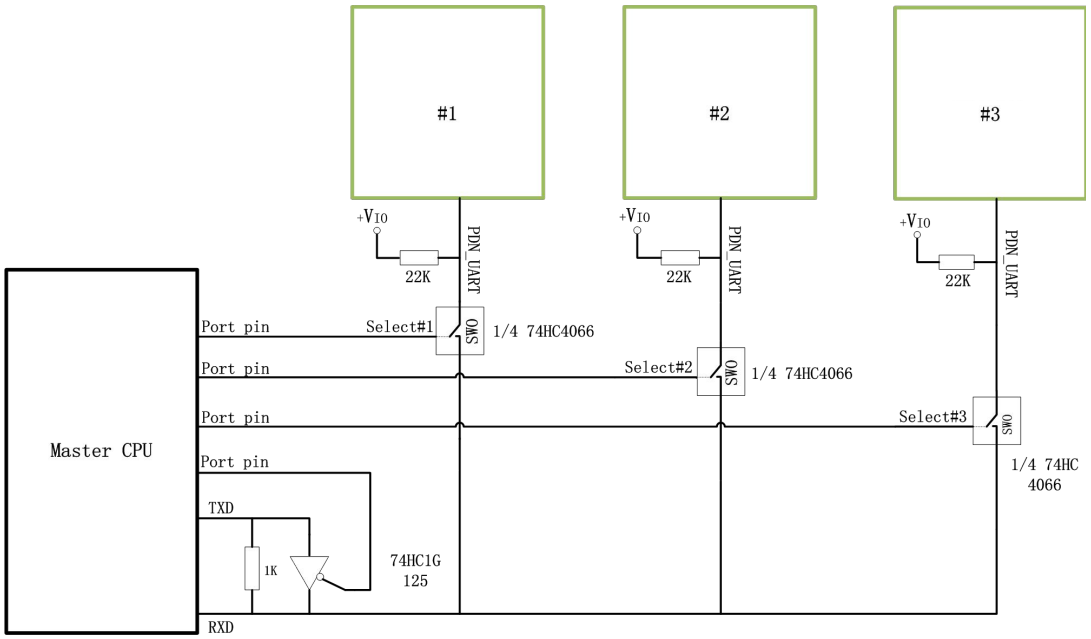


图 2.2 模拟开关控制多地址从机

- 在有效范围内选择一个可靠的波特率，比如 250K，写一次寄存器需要 320us（=8 Bytes*（8+2）bits*4us）。
- 传输开始之前，通过设置 port pin为高让模拟开关打开，其他所有从机不工作，除非有请求。
- 当使用可选择缓冲器的时候，允许6609在接收到应答数据之后设置一个适当的发送延迟时间。
- 开始传输时，通过 port pin 置低激活 TXD 线。
- 当发送读寄存器请求时，在传输的最后一位结束位完成时，关闭可选择缓冲器。
- 注意，所以传输数据都是 RXD 线接收。

三．寄存器列表

本章介绍了全部的寄存器设置，实际应用功能详细的在下面的章节介绍。

3.1 寄存器总表

全部寄存器配置（0x00...0x0F）					
R/W	**地址**	**n**	**寄存器**	**描述/位名**	
RW	0x00	10	GCONF	**Bit**	**GCONF-全局配置**
				0	I_scale_analog (默认=1) 0:用内部基准电压（5VOUT 分压） 1:用 VREF 脚输入做电流参考
				1	Internal_Rsense (默认:OTP) 0: 外部感应电阻模式 1:内部感应电阻模式 内部感应电阻模式采用 VREF 电流输入作为基准，此时 VREF 脚近似接地。
				2	en_Quick(默认:OTP) 0:静音模式使能 1:快速模式使能
				3	Shaft 1:改变电机方向
				4	index_OTPw 0:INDEX 显示定序器第一个微步的位置 1:INDEX 显示过热预警信号
				5	index_step 0:INDEX 当做 index_OTPw 来用 1:INDEX 显示内部发生器产生的 step 信号
				6	pdn_disable 0:PDN_UART 管脚控制停止电流模式 1:PDN_UART 输入被禁止，使用 UART 接口模式时设置此位为高。
				7	mstep_reg_select 0:微步分辨率通过 MS1，MS2 管脚设置。 1:微步分辨率通过寄存器 MSTEP 设置。
				8	multistep_filt(默认=1) 0:STEP 脉冲不做滤波 1:当全步频率大于 750HZ 时滤波，启用时 TSTEP 会显示滤波步长时间值
				9	test_mode 0:正常操作 1:测试模式，模式测试信号从 ENN 输出。

全部寄存器配置（0x00...0x0F）					
R/W	**地址**	**n**	**寄存器**	**描述/位名**	
R+ WC	0x01	3	GSTAT	**Bit**	**GSTAT-全局状态**
				0	reset 1:指示芯片已经复位，所有寄存器回到复位值。
				1	drv_err 1:指示芯片由于过热或者短路检测停止驱动，详情读 DRV_STATUS 寄存器，这个标志只能在所有错误都被消除以后被清除。
				2	uv_cp 1:指示电荷泵欠压，这种情况下驱动停止。这个标志不会被锁存所以不需要清除。
R	0x02	8	IFCNT		接口传输计数器。这个寄存器值随着成功写寄存器次数增加而增加。读这个寄存器可以检查连续传输丢失的数据。读寄存器不会改变这个值，这个值从 255 到 0 循环。
W	0X03	4	SLAVECONF	**Bit**	**SLAVECONF**
				11..8	读寄存器发送延时（应答被发送以后） 0,1: 8 bit times 2,3: 3*8 bit times 4,5: 5*8 bit times 6,7: 7*8 bit times 8,9: 9*8 bit times 10,11: 11*8 bit times 12,13: 13*8 bit times 14,15: 15*8 bit times
W	0x04	16	OTP_PROG	**Bit**	**OTP_PROGRAM-可编程 OTP**
				2..0	OTPBIT 选择要编程的 OTP 位
				5..4	OTPBYTE 选择要编程的 OTP 字节
				15..8	OTPMAGIC 设置该寄存器值为 0xbd 来使能 OTP 编程，推荐每次编程时间最少为 10ms（通过读寄存器 OTP_READ 检查）。
R	0x05	24	OTP_READ	**Bit**	**OTP_READ（读 OTP 存储器的值）**
				7..0	OTP0 读数据
				15..8	OTP1 读数据
				23..16	OTP2 读数据

全部寄存器配置（0x00...0x0F）					
R/W	**地址**	**n**	**寄存器**	**描述/位名**	
R	0x06	10+8	IOIN	**Bit**	**INPUT（读所有输入管脚的状态）**
				0	ENN
				1	-
				2	MS1
				3	MS2
				4	DIAG
				5	STEP
				6	PDN_UART
				7	STEP
				8	-
				9	DIR
				31..24	-
RW	0x07	5+2	FACTORY_CONF	4..0	FCLKTRIM (默认:OTP 值) 0..31:最低到最高频率。检查电荷泵输出，频率范围不是固定的，但是可以测试，内部时钟有可能调到 12MHz，12MHz 的时钟频率可以通过 OTP 编程预设。
				9..8	OTTRIM （默认:OTP 值） 00： OTP=143℃，OTPW=120℃ 01： OTP=150℃，OTPW=120℃ 11： OTP=150℃，OTPW=143℃ 10： OTP=157℃，OTPW=143℃

3.1.1 OTP_READ-OTP 配置存储器

所有 OTP 存储器的默认值为 0，每次编程只能设置 1 位，设置过的位无法清除。工厂通过设置OTP0.0-OTP0.4 调节时钟频率，因此各个芯片之间这些位的值可能不一样。

<table>
<tr><th colspan="5">0x05:OTP_READ-OTP 存储器表格</th></tr>
<tr><th>Bit</th><th>名称</th><th>功能</th><th colspan="2">描述</th></tr>
<tr><td rowspan="3">23</td><td rowspan="3">OTP2.7</td><td rowspan="3">OTP_en_Quick</td><td colspan="2">设定默认工作模式：快速模式或者静音模式。</td></tr>
<tr><td>0</td><td>默认:静音模式(en_Quick=0)
OTP1.0 到 1.7 和 2.0 用在静音模式
快速模式设置：HEND=0；HSTART=5；
TOFF=3</td></tr>
<tr><td>1</td><td>默认:快速模式(en_Quick=1) OTP1.0
到 1.7 和 2.0 用在快速模式静音
模式设置：
PWM_GRAD=0;TPWM_THRS=0;
PWM_OFS=36;pwm_autograd=1</td></tr>
<tr><td>22</td><td>OTP2.6</td><td rowspan="2">OTP_IHOLD</td><td colspan="2" rowspan="2">默认待机电流由 IHOLD 决定(只在待机功能开启的时候起作用，管脚 PDN_UART 置0)。
00： IHOLD=16 (53% 的 IRUN)
01： IHOLD=2 (9% 的IRUN)
10： IHOLD=8 (28% 的 IRUN)
11： IHOLD=24 (78% 的 IRUN)
（默认运行电流 IRUN=31）</td></tr>
<tr><td>21</td><td>OTP2.5</td></tr>
<tr><td>20</td><td>OTP2.4</td><td rowspan="2">OTP_IHOLDDELAY</td><td colspan="2" rowspan="2">默认 IHOLDDELAY
00: IHOLDDELAY=1
01: IHOLDDELAY=2
10: IHOLDDELAY=4
11: IHOLDDELAY=8</td></tr>
<tr><td>19</td><td>OTP2.3</td></tr>
<tr><td>18</td><td>OTP2.2</td><td>OTP_PWM_FREQ</td><td colspan="2">默认 PWM_FREQ
0: PWM_FREQ=01=2/683
1: PWM_FREQ=10=2/512</td></tr>
<tr><td>17</td><td>OTP2.1</td><td>OTP_PWM_REG</td><td colspan="2">默认 PWM_REG
0: PWM_REG=1000: 4 increment/cycle
1: PWM_REG=0010: 1 increment/cycle</td></tr>
<tr><td rowspan="3">16</td><td rowspan="3">OTP2.0</td><td rowspan="2">OTP_PWM_OFS</td><td colspan="2">依附于 OTP_en_Quick</td></tr>
<tr><td>0</td><td>0：PWM_OFS=36
1：PWM_OFS=00 （没有提前调节）
;pwm_autograd=0</td></tr>
<tr><td>OTP_CHOPCONF8</td><td>1</td><td>默认值 hend1</td></tr>
</table>

0x05:OTP_READ-OTP 存储器表格				
Bit	**名称**	**功能**	**描述**	
15	OTP1.7	OTP_TPWMTHRS	依附于 OTP_en_Quick	
14	OTP1.6		0	默认 TPWM_THRS 0:TPWM_THRS =0 1:TPWM_THRS =200 2:TPWM_THRS =300 3:TPWM_THRS =400 4:TPWM_THRS =500 5:TPWM_THRS =800 6:TPWM_THRS =1200 7:TPWM_THRS =4000
13	OTP1.5			
		OTP_CHOPCONF7..5	1	默认 hstrt1，hstrt2，hend0
12	OTP1.4	OTP_pwm_autograd	依附于 OTP_en_Quick	
			0	0: pwm_autograd=1 1: pwm_autograd=0
		OTP_CHOPCONF4	1	默认 hstrt0 (pwm_autograd=1)
11	OTP1.3	OTP_PWM_RGAD	依附于 OTP_en_Quick	
10	OTP1.2		0	默认 PWM_RGAD 0: PWM_RGAD= 14 0: PWM_RGAD= 16 0: PWM_RGAD= 18 0: PWM_RGAD= 21 0: PWM_RGAD= 24 0: PWM_RGAD= 27 0: PWM_RGAD= 31 0: PWM_RGAD= 35 0: PWM_RGAD= 40 0: PWM_RGAD= 46 0: PWM_RGAD= 52 0: PWM_RGAD= 59 0: PWM_RGAD= 67 0: PWM_RGAD= 77 0: PWM_RGAD= 88 0: PWM_RGAD= 100
9	OTP1.1			
8	OTP1.0			
		OTP_CHOPCONF3..0	1	默认 TOFF

0x05:OTP_READ-OTP 存储器表格			
Bit	**名称**	**功能**	**描述**
7	OTP0.7	OTP_TBL	默认 TBL 0: TBL=10 1: TBL=01
6	OTP0.6	OTP_internalRsense	默认 internalRsense 0: 外部感应电阻 1: 内部感应电阻
5	OTP0.5	OTP_OTTRIM	默认 OTTRIM 0: OTTRIM=00(143℃) 1: OTTRIM=01(150℃)
4	OTP0.4	OTP_FCLKTRIM	默认 FCLKTRIM 0: 最低频率 31: 最高频率
3	OTP0.3		
2	OTP0.2		
1	OTP0.1		
0	OTP0.0		

3.2 速度控制

<table>
<tr><th colspan="5">速度控制寄存器设置（0x10...0x1F）</th></tr>
<tr><th>R/W</th><th>地址</th><th>n</th><th>寄存器</th><th colspan="2">描述/位名</th></tr>
<tr><td rowspan="4">W</td><td rowspan="4">0x10</td><td rowspan="4">5+5+4</td><td rowspan="4">IHOLD_IRUN</td><td>Bit</td><td>IHOLD_IRUN-驱动电流控制</td></tr>
<tr><td>4..0</td><td>IHOLD （默认:OTP 值）
停转电流（0=1/32...31=32/32）
在静音模式下，设置 IHOLD=0，可以在停转条件下选择自由转动或者被动制动方式。</td></tr>
<tr><td>12..8</td><td>IRUN（默认=31）
电机运行电流（0=1/32...31=32/32）
提示：在选择感应电阻的情况下，正常的 IRUN 值是 16 到31。</td></tr>
<tr><td>19..16</td><td>IHOLDDELAY（默认:OTP 值）
控制电机在停转条件（stst=1）时电流平稳降低，避免电流的跳跃。</td></tr>
<tr><td>W</td><td>0x11</td><td>8</td><td>TPOWERDOWN</td><td colspan="2">TPOWERDOWN（默认=20）
设置从检测到停转条件（stst）到电流开始减少的延迟时间。范围从 0 到5.6s。
0...((2^8)-1)*2^18 tclk
注意：静音模式下电流自动调节允许 TPOWERDOWN 最小值为 2。</td></tr>
<tr><td>R</td><td>0x12</td><td>20</td><td>TSTEP</td><td colspan="2">TSTEP=(fclk/fstep)/(256/细分数)（fstep：输入的 step 频率）
TSTEP 使用一个 1/16 的迟滞来补偿电流跳跃，TSTEP 值与
(TPWMTHRS*15/16)-1 比较。
减速切换模式切换点：TSTEP=设定值
加速切换模式切换点：TSTEP<设定值</td></tr>
<tr><td>W</td><td>0x13</td><td>20</td><td>TPWMTHRS</td><td colspan="2">设置静音模式速度上限
TSTEP>=TPWMTHRS
静音模式使能时，如果电机速度超过 TPWMTHRS 设定的极限值，切换到快速模式。
0：Disabled</td></tr>
<tr><td>W</td><td>0x22</td><td>24</td><td>VACTUAL</td><td colspan="2">VACTUAL 允许用 UART 接口驱动电机
速度范围为 0-(+-（2^23-1）*(usteps/t)）
0:正常操作，外部 STEP 起作用。
/=0：电机转动受 VACTUA 控制，内部 step 可以通过 INDEX 管脚监测。
电机方向由 VACTUAL 符号位控制。</td></tr>
</table>

3.3 定序器寄存器

微步控制寄存器设置（0x60-0x6B）					
R/W	**地址**	**n**	**寄存器**	**描述/位名**	**范围**
R	0x6A	10	MSCNT	微步计数器。指示 CUR_A 在微步表中的位置，CUR_B 通过在微步表中将 A 移位 256得到。	0...1023
R	0x6B	9+9	MSCURACT	Bit8..0: CUR_A(带符号)： 电机 A 相电流，从内部正弦波表格读出。（无电流缩小） Bit24..16: CUR_B(带符号)： 电机 B 相电流，从内部正弦波表格读出。（无电流缩小）	+/-0...255

3.4 斩波器控制寄存器

驱动寄存器设置（0x6C-0x7F）						
R/W	**地址**	**n**	**寄存器**	**描述/位名**		**范围**
RW	0x6C	32	CHOPCONF	斩波器和驱动设置		
R	0x6F	32	DRV_STATUS	驱动状态标志和电流数值读取		
RW	0x70	22	PWMCONF	静音模式斩波器配置		
R	0x71	9+8	PWM_SCALE	静音模式幅值调节结果。这个值用来监测自动 PWM 幅值缩小的计算。（255= 最大电压）		
				bit7...0	PWM_SCALE_SUM: 这个值是用来按比例缩小 CUR_A 和 CUR_B。	0...255
				bit24...16	PWM_SCALE_AUTO： 9bit 有符号数加入 PWM 占空比的计算。	有符号 -255...+255
R	0x72	8+8	PWM_AUTO	bit7...0	PWM_OFS_AUTO: 自动计算偏置值	0...255
				bit23...16	PWM_GRAD_AUTO: 自动计算梯度值	0...255

3.4.1 CHOPCONF-斩波器配置

0x6C:CHOPCONF-斩波器配置			
Bit	**名称**	**功能**	**描述**
31	diss2vs	低边短路 保护失效	0：低边短路保护打开 1：低边短路保护关闭
30	diss2g	接地短路 保护失效	0：接地短路保护打开 1：接地短路保护关闭
29	dedge	Step 双沿使能	1：使能 step 双沿，降低 step 频率要求，这个功能不能和 step 滤波功能兼容
28	intpol	256 微步内插	实际微步分辨率（MRES）外扩到 256,使电机更平稳操作。
27	mres3	MRES 微步分辨率	0000： 256 微步分辨率
26	mres2		
25	mres1		0001...1000: 128,64,32,16,8,4,2,FULLSTEP 递减的微步分辨率 微步分辨率的值表示四分之一正弦波有多少个微步。 每个 step 等于 2^MRES 个微步
24	mres0		
23 … 18		预留	设置为 0
17	vsense	电压感应电阻	0：低灵敏度，高感应电阻电压 1：高灵敏度，低感应电阻电压
16	tbl1	TBL 空白时间选择	00...11 设置比较器空白时间为 16,24,32，40 个时钟。 注意：00 和01 被推荐使用的比较多。
15	tbl0		
14 … 11		预留	设置为 0
10	hend3	HEND 迟滞低值 OFFSET 正弦波偏置	0000...1111 迟滞为-3，-2，-1,0,1...,12 (1/512 为迟滞的尺度加入电流的计算）这个值用于迟滞斩波器。
9	hend2		
8	hend1		
7	hend0		
6	hstrt2	HSTRT 迟滞开始值 加到 HEND	000...111 加 1,2，...,8 到迟滞低值 HEND （1/512 为迟滞的尺度加入电流的计算）注意：有效 HEND+HSTRT <=16 迟滞每 16 个时钟衰减完成 (默认：OTP，静音模式默认值为 0)
5	hstrt1		
4	hstrt0		

0x6C:CHOPCONF-斩波器配置			
Bit	**名称**	**功能**	**描述**
3	toff3	TOFF 关闭时间和驱动使能	Off 时间设置，控制迟滞衰减持续时间。 0000：驱动关闭 0001：只用在 TBL>=2 的时候 0010…1111: 2 …15 (默认：OTP,静音模式默认值为 3)
2	toff2		
1	toff1		
0	toff0		

3.4.2 PWMCONF-静音模式

0x70:PWMCONF-静音模式				
Bit	**名称**	**功能**	**描述**	
31 … 28	PWM_LIM	模式切换时 PWM 幅值自动限制	快速模式切换到静音模式时限 PWM_SCALE_AUTO 的值，这个值定义了自动电流控制的幅值上限。 可以减少快速模式切换到静音模式时的电流跳跃。不会限制 PWM_GRAD 和 PWM_GRAD_AUTO 的值（默认 12）	
27 … 24	PWM_REG	调节梯度	PWM 幅值改变以半波来算(pwm_autoscale=1） 1：0.5 增量 … 8: 4 增量 … 15: 7.5 增量 （每一位+0.5）	
23		预留	设置为 0	
22		预留	设置为 0	
21	freewheel1	不同停转模式	电机电流设置为零。（I_HOLD=0） 00: 正常操作 01：自由转动 10：线圈短接，使用 LS 驱动器 11：线圈短接，使用 HS 驱动器	
20	freewheel0			
19	pwm_ autograd	PWM 自动梯度适配	0	PWM_GRAD_AUTO=PWM_GRAD
			1	PWM_GRAD_AUTO 初始化后， 在运动中调整为最佳值。条件： 1.PWM_OFS_AUTO 自动初始化， 这要求满足三个条件: (a)进入停转状态，电流为 IRUN，总时间大于 130ms。 (b)检测到停转后等待 128 个斩波时钟。 (c)-1<PWM_SCALE_AUTO<1 时 调 节 PWM_OFS_AUTO 2.电机运行且 1.5*PWM_OFS_AUTO<PWM_SCALE_SUM<4*PWM_OFS_AUTO 并且 PWM_SCALE_SUM<255 PWM_GRAD_AUTO 每 8 个全步变化 1
18	pwm_ autoscale	PWM 幅值自动缩小	0	用户定义期望幅值，电流设定 IRUN 和 IHOLD 不影响。PWM 幅值结果是： PWM_OFS*((CS_ACTUAL+1)/32)+PWM_GRAD*256/TSTEP
			1	使能自动电流调整（默认）

3.4.2 PWMCONF-静音模式

0x70:PWMCONF-静音模式			
Bit	**名称**	**功能**	**描述**
17	pwm_freq1	PWM 频率选择	00：fpwm=2/1024 fclk 01： fpwm=2/683 fclk 10： fpwm=2/512 fclk 11： fpwm=2/410 fclk
16	pwm_freq0		
15	PWM_ GRAD	用户定义幅值梯度	幅值梯度对幅值的影响： PWM_GRAD*256/TSTEP 这个值加入幅值的计算补偿电机的反电动势。 在电流自动调节的过程中，这个值只是用来初始化。 设置 PWM_GRAD 和 PWM_GRAD_AUTO 的比值来加速自动调节过程。可以保存一个近似值到 OTP 里面。
14			
13			
12			
11			
10			
9			
8			
7	PWM_ OFS	用户定义幅值	用户自定义幅值，停转条件下要满足电流为全电流（CS_ACTUAL=31） （默认值=36） 在使用电流自动调节时，这个值只用来初始化。自动秤功能开始于 PWM_SCALE_AUTO=PWM_OFS。 PWM_OFS=0,会让电流自动调整失效。 PWM_OFS>0 允许自动调整到 PWM 低占空比，甚至低于阈值。
6			
5			
4			
3			
2			
1			
0			

3.4.3 DRV_STATUS-驱动器状态标志

0x6F:DRV_STATUS-驱动器状态标志和电流读取			
Bit	**名称**	**功能**	**描述**
31	stst	停转指示器	这个标志指示电机出现停转条件，这个条件出现在 step 长度大于 2^20 个时钟的时候。
30	slient	静音模式指示器	1：驱动器工作在静音模式 0：驱动器工作在快速模式
29 … 24		预留	忽略这些位
23 22 21		预留	忽略这些位
20 19 18 17 16	CS_ ACTUAL	真实电机电流	真实电流，控制电流缩小，用来监控自动电流调整。
15 … 12		预留	忽略这些位
11	t157	157℃比较器	1:超过温度阈值
10	t150	150℃比较器	1:超过温度阈值
9	t143	143℃比较器	1:超过温度阈值
8	t120	120℃比较器	1:超过温度阈值
7	olb	B 相开路指示器	A 相和 B 相开路检测。 注意：错误的检测可能会出现在快速运动和停转状态的时候，只能在低速的时候检测。
6	ola	A 相开路指示器	
5	s2vsb	B 相低边短路指示器	A 相和 B 相低边短路检测，驱动器变成关闭。错误标志保持，直到驱动器因为 TOFF=0 或者 ENN 置高而关闭会清除。这个标志与双斩波模式互相独立。
4	s2vsa	A 相低边短路指示器	
3	s2gb	B 相接地短路指示器	A 相和 B 相接地短路检测，驱动器变成关闭。错误标志保持，直到驱动器因为 TOFF=0 或者 ENN 置高而关闭会清除。这个标志与双斩波模式互相独立。
2	s2ga	A 相接地短路指示器	
1	ot	过热标志	超过选择的过温温度极限，直到 OTPw 由于芯片冷却被清除，驱动器打开。
0	OTPw	过热预警标志	超过选择的过温警告温度极限。

四．静音模式

静音模式是步进电机的一种极低噪音的控制模式。它基于 PWM 电压控制模式，当工作在静止或者低转速下几乎没有噪音，所以这种控制模式适合应用在室内或者家里。电机在低速时震动 也小。静音模式的电压模式通过驱动细分的电压来得到电流，驱动器可以自动适应环境，不需要 额外的设定。对于一些特殊情况，还提供了一些可选的配置。电机高转速下，可以考虑将快速模 式与静音模式相结合。

4.1 自动调整

静音模式集成了一个自动调整程序，可以调节最重要的参数来适应大部分的电机。这样，静音模式允许高动态特性，并且可以可以支持工作在非常低的电流。两个步骤来得到好的控制结果：电机起始于静态，启动于常规电流（AT#1）。电机在中速运行，自动调整（AT#2）。
图 4.1 为调整时序图。
下表为 AT#1 与 AT#2 的条件。

步骤	参数	条件	动作
AT#1	PWM_OFS_AUTO	电机处于静止并且电流调整值等于运行电流设定（*IRUN*） 如果静态功耗消减功能启用（PDN_UART=0), 初始的步进脉冲会使电流回到运行电流。 VS 与 VREF 管脚处于正常电平	≥2^20+2*2^18 tclk， ≥130ms （使用内部时钟）
AT#2	PWM_GRAD_AUTO	电机必须在一个速度下运转，此时有反电动势EMF，并且可以达到最大电流。条件： 1.5*PWM_OFS_AUTO＜ PWM_SCALE_SUM ＜4*PWM_OFS_AUTO PWM_SCALE_SUM ＜ 255 建议： 在60至300RPM转速下，电机效果最好	需要 8 个整步来实现＋/－1． 对于一个标准电机 PWM_GRAD_AUTO 的目标值在 64 或者更小，从 OTP 设定值 14，需要 400 个整步调整完

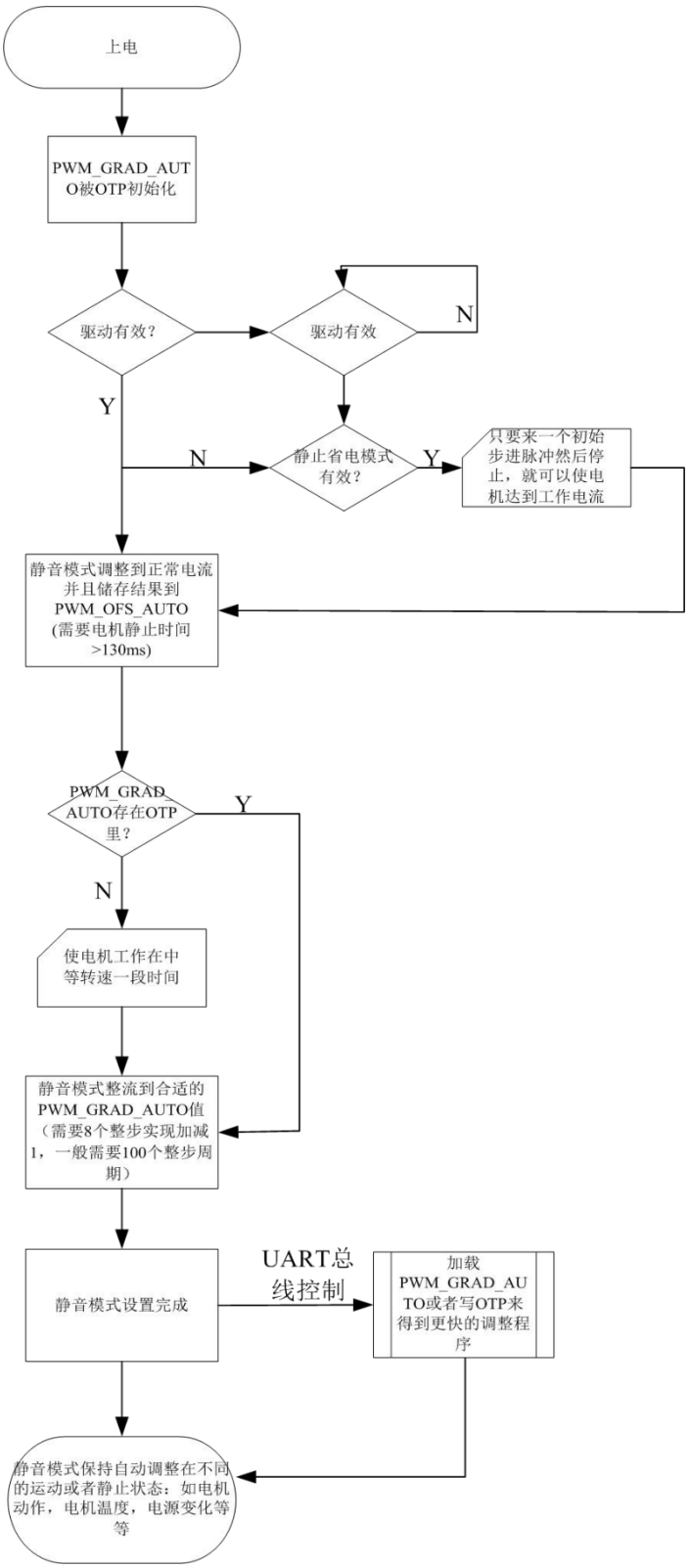


图 4.1 静音模式自动调整程序

注意事项：

改变 VREF 以及电源 VS 会使自动调整的结果无效。电机整流不能补偿这种重大的调整直到下一个 AT#1 的到来。

4.2 静音模式选项

为了匹配电流到一个固定值，有效的PWM电压需要根据电机的实际转速来调整。几个额外的条件会影响到实际的有效电压：电机阻抗，电机的反向电动势 EMF（正比于转速）。
有两种PWM调整方法：自动调整模式使用电流反馈（pwm_autoscale=1，pwm_autograd=1），以及前置速度控制模式。前置速度控制模式不受电源电压的影响，如堵转情况，但是能提供非常稳定的幅度，它不需要任何的电流检测，适合于常用的电机种类与电压环境。因此我们建议使用自动调整模式，触发电流整流不能满足给定的工作条件。

建议客户使用自动调整模式。前置速度控制模式是一种非自动调整模式（pwm_autoscale=0）只能用在常规的电机与运动条件，需要通过UART接口写寄存器去控制电机。寄存器PWM_GRAD 与PWM_OFS可以在自动控制模式的检测到。在非自动模式下，电源电流直接反应出电机的机械负载变化。

静音模式下的 PWM 频率选择由时钟频率的不同分频得到，设置在 20-50kHz 满足大部分要求，此设置主要考虑低电流纹波，高速特性与动态功耗之间的平衡。

静音模式频率选择				
时钟频率 f_{CLK}	PWM_FREG=%00 $f_{PWM}=2/1024f_{CLK}$	PWM_FREG=%01 $f_{PWM}=2/683f_{CLK}$	PWM_FREG=%10 $f_{PWM}=2/512f_{CLK}$	PWM_FREG=%11 $f_{PWM}=2/410f_{CLK}$
18MHz	35.2kHz	52.7kHz	70.3kHz	87.8kHz
16MHz	31.3kHz	46.9kHz	62.5kHz	78.0kHz
12MHz	23.4kHz	35.1kHz	46.9kHz	58.5kHz
10MHz	19.5kHz	29.3kHz	39.1kHz	48.8kHz
8MHz	15.6kHz	23.4kHz	31.2kHz	39kHz

4.3 静音模式电流整流器

静音模式是电压PWM模式，自动整流功能（pwm_autoscale=1，pwm_autograd=1）将电流调整到需要的设定值。自动整流用于自动调整程序的一部分，用来追踪电机的参数变化。驱动器检测斩波器导通期的实际电流并且使用一个比例整流器来调整PWM_SCALE_AUTO的值，来将电机 电流匹配到目标值。PWM_REG 是比例因子的设定寄存器。理论上来说，比例因子越小越好，这样能得到更加稳定与柔和的调整过程，同时又要足够大，以满足驱动器快速对电机的电流变化做成反应（如改变 VREF）。在自动调整的 AT#2 初始阶段，PWM_REG 也用来补偿电机的速度变化。因此，AT#2器件的快加速需要一个较大的 PWMs_REG的设定值。PWM_REG设定需要满足加速与减速的斜率要求。PWM_REG设置值是否合理，在自动调整的 AT#2 以及自动调整程序完成后（或无自动调整功能时 PWM_OFS与 PWM_GRAD），可以监测加速期的电流来判断，如图 4.2：

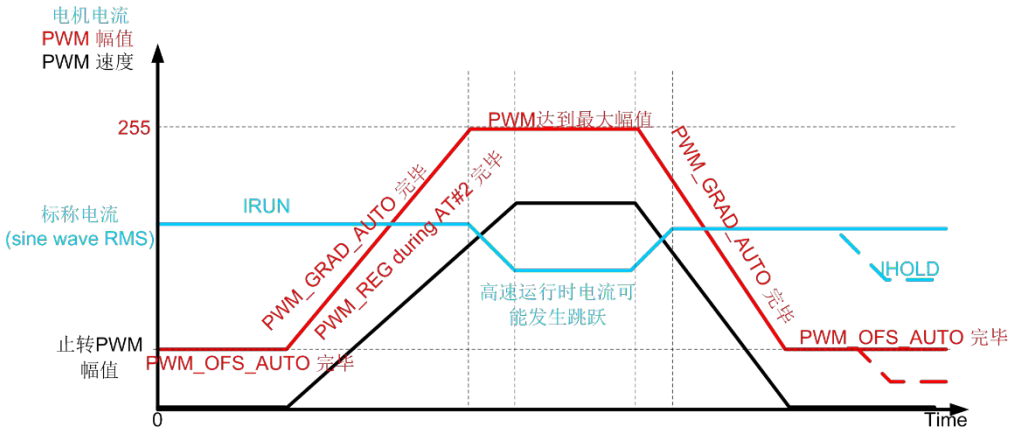


图 4.2 PWM_GRAD 和 PWM_OFS 设定

4.3.1 最小电流限制

静音模式斩波电流调整器需要一个最低的电机电流值。由于线圈电流只能在斩波控制器的打开时期，通过检测采样电阻上的电压来得到，最小的斩波整流电流由空白时间 TBL 与斩波频率共同决定。电源电压上升与斩波频率上升会增大这个最小电流，而小的 TBL 设置会减小这个最小电流。正确确定 PWM_OFS_AUTO 的值非常重要，在自动调整的 AT#1 阶段，电流由采样电阻，VREF，IRUM 共同决定。更低的电流（比如静止的低功耗态）由PWM_OFS_AUTO 与 PWM_GRAD_AUTO 决定，而这两个寄存器由 PWM_OFS，PWM_GRAD 两个非自动电流整流的寄存器决定。自由旋转态下电机电流可以到 0.

最低线圈电流在自动整流模式下，限制公式如下：

$$I_{lowerlimit}=t_{BLANK}*f_{PWM}*V_M/R_{COIL}$$

V_M 是电机电源电压,R_{COIL} 是电机线圈内阻。 $I_{lowerlimit}$可以看作由IRUN设置的最小正常工作电流。

示例如下：

电机内阻 5Ω，电源电压 24V。TBL 设置%01，PWM_FREQ=%00，t_{BLANK} 等于 24 个时钟周期，等与 2/（1024 个时钟周期），计算：

$$I_{lowerlimit}= 24*t_{CLK}*2/1024/t_{CLK}*24V/5Ω=225mA$$

这就意味着，电机自动调整的目标电流需要大于 225mA。最低电流同样会影响 VREF 对电流的设定。

注意事项：当在自动调整模式，注意最小电流限制。在自动调整的AT#1阶段必须大于这个最小电流。这个最小电流可以用电流探针来测量，通过设定运行电流IRUN或者保持电流 IHOLD。

4.4 速度整流

静音模式速度整流功能，基于两个步进信号之间的时间，如基于TSTEP，单位是时钟周期。这个理论原则上不需要电流测试，因为反馈环路不是必须的。当 pwm_autoscale =0简单的速度调整用 UART 编程实现。基础理论是需要一个近似线性的电压去得到一个目标电机电流。电机的内阻是 R，由电流公式 I=U/R，U是电源电压被 PWM调制后的值。初始的 PWM_AMPL的计算如下：

$$PWM\_APML=374*R_{COIL}*I_{COIL}/V_M$$

有效的 PWM电压 U_{PWM}（峰值电压的 0.707）从 8bit 对峰值 248的正弦波的细分得到：

$$U_{PWM}=V_M*PWM\_SCALE/256*248/256*0.707=V_M*PWM\_SCALE/374$$

当电机速度加快，电机会产生一个反向电动势，电动势的值与电机速度成正比，会减小加载到线圈电阻上的有限电压而减小电流。6609 提供第二个速度参数 PWM_GRAD 来补偿这个反向电动势。总的有效 PWM 幅度（PWM_SCALE_SUM）在这个模式下，依赖于细分微步的频率，计算如下：

$PWM\_SCALE\_SUM=PWM\_OFS+PWM\_GRAD*256*f_{STEP}/f_{CLK}$

PWM_GRAD 一阶近似计算公式：

$PSM\_GRAD=C_{BEMF}[V/（rad*s）]*2\pi*f_{CLK}*1.46/（V_M*MSPR）$

C_{BEMF} 是反向电动势常数，单位是伏特每 rad 每秒，MSPR 是电机每圈的细分步进数，如 51200=256 微步 x200 整步每圈（1.8°的电机）

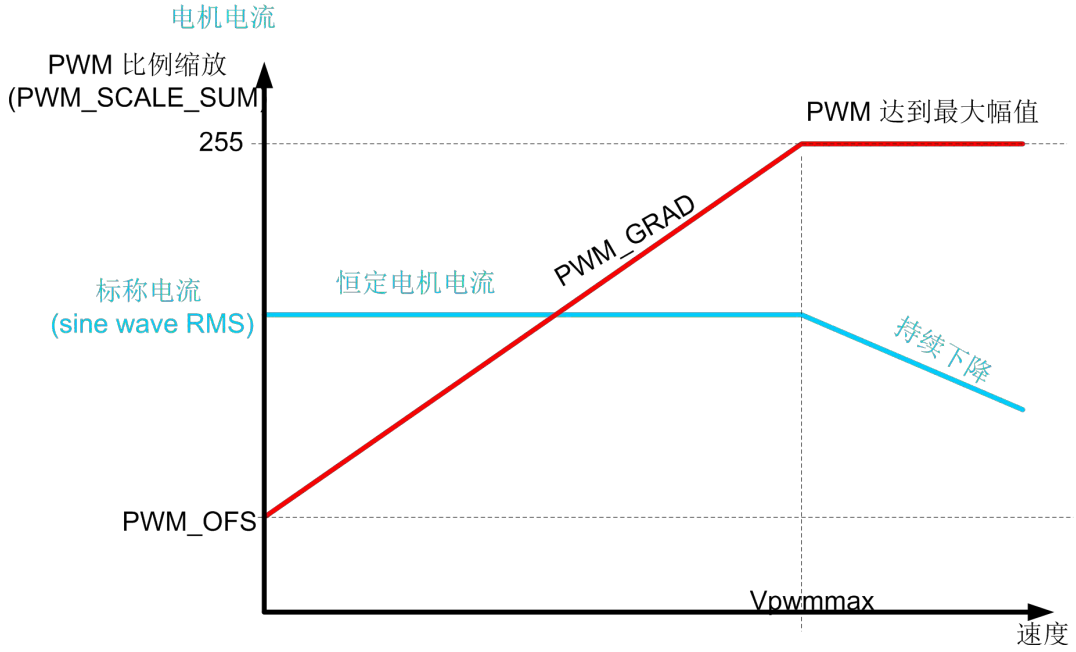


图 4.3 基于速度的 PWM 幅值(pwm_auto_scale=0)

小结：PWM_OFS与 PWM_GRAD可以通过用示波器追踪电机电流来得到合适的值。可选择自动调整模式来确定以及通过寄存器读出 PWM_OFS_AUTO 与PWM_GRAD_AUTO。

电机的反向电动势常数：

反电动势来源与电机的转动，一般的电机不会标明这项参数，而是标明力矩与电流的衰减。在国际单位中，反向电动势常数 C_{BEMF} 与力矩常数的单位一致。例如电机转动在 1rps（1 圈每秒 =6.28rad/s），将产生 6.28V 的反向电动势。因此，反向电动势常数计算公式：

$C_{BEMF}[V/（rad/s）]=HoldingTorque[Nm]/I_{COILNOM}[A]/2$

$I_{COILNOM}$ 是电机特定相电流下对应的保持力矩。HoldingTorque 是电机的保持力矩，对应两个线圈电流都是 $I_{COILNOM}$。力矩单位是[Nm] ，1Nm=100Ncm=1000mNm。

电压是有效电压，在每个线圈中都有效，所以公式中电流需要乘2.

4.5 静音模式与快速模式的合并使用

一些高速运动模式下，快速模式更加稳定。6609可以合并静音模式与快读模式，通过设置一个速度转换点。这个速度转换点（TPWMTHRS）可以写入OTP。这种模式下，静音模式工作在低速。

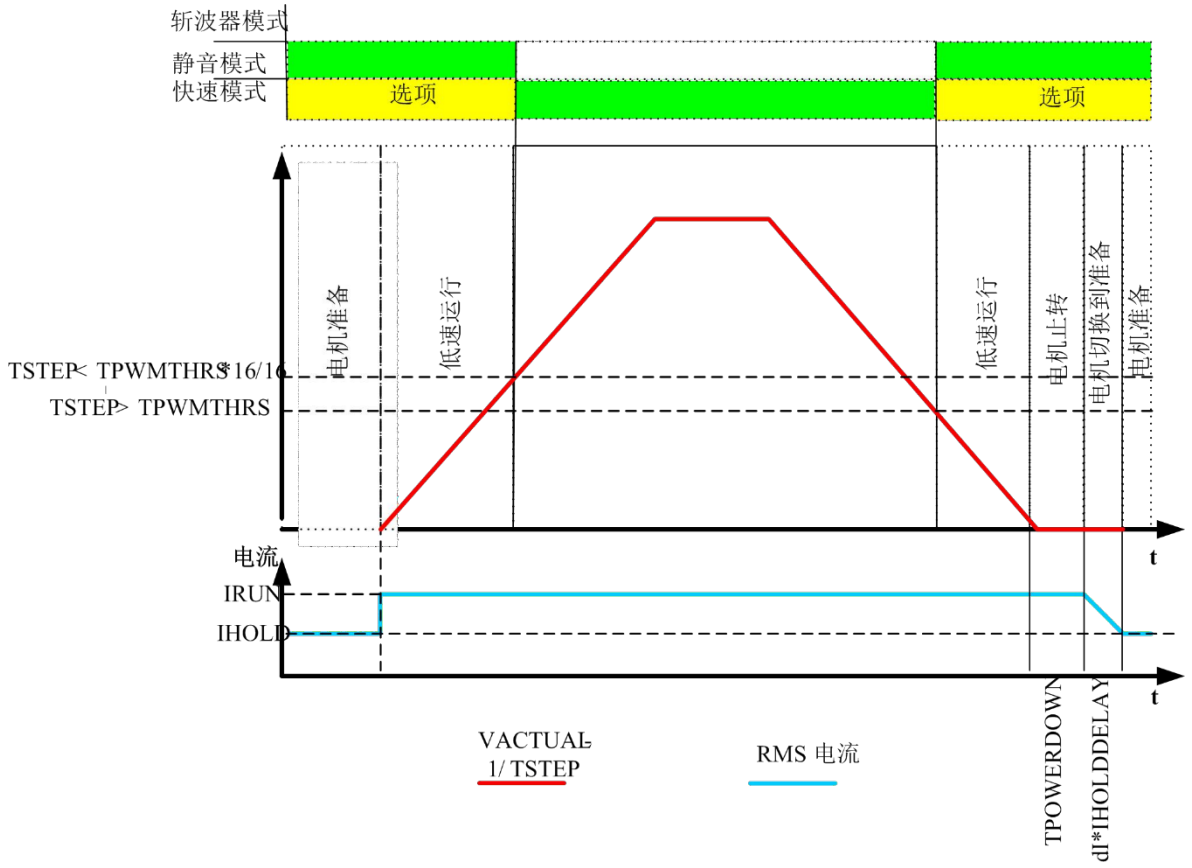


图 4.4 可选的 TPWMTHRS 控制静音和快速混合模式

第一步，两种模式的参数需要单独设置好（快速模式参数可以写入OTP），下一步，设定一个转换速度，让低速时工作在静音模式，高速时工作在快速模式。TPWMTHRS用来设置这个转换速度。可以在电机需要的速度运转时读出 TSTEP，然后写入 TPWMTHRS。使用一个低的转换速度 来避免反转时的电流过冲。

当在高速进行模式转换时会有大的电流过冲毛刺，由于电机的反向电动势（随速度增大）会带来电机的电压与电流相位差90°，当高速时两种模式之间的切换会导致毛刺剧烈增大。这个毛刺电流大到甚至超过过流点（取决于电机阻抗）。在低速下（如 1to10 RPM）下，这个毛刺对于大多数电机都忽略不计。因此，考虑到这个电流毛刺，如果你只是要需要静音模式，将TPWMTHRS 设置为0.

当第一次使用静音模式时使用自动电流调整功能，电机需要处于静止态以进行适当的电流调整。当电机从高速转到静音模式，斩波器逻辑加载上次电流整流的结果，直到电机返回到更低速 度 。这样，静音模式开始时，整流器回到低速时有了已知的开始点。因此，当模式切换时，不需 要考虑速度转换点与电压变化，由于电机不会失步或者瞬态电流过大或者过小。

电机堵转或速度的突然变化可能会认为检测到短路或者会引起电流的自动调整，该状态不能自己恢复。这时候需要清除错误的指示信号，并且重启电机从零速开始恢复到这个状态。

电机堵转可能造成过流条件，取决于电机速度与电机的线圈阻抗。低速下，反电动势只是电

源电压的一小部分，没有触发短路的风险。

小结：当第一次转到静音模式时，开始电机与静止模式，并且停止至少128个斩波周期以进行初始静态电流控制。

4.6 静音模式的指示信号

静音模式使用电压模式驱动，状态指示器基于电流检测，速度较慢，特别是驱动器由于反电动势突变如电机堵转会师指示器延迟。

4.6.1 负载开路指示信号

在静音模式里，状态指示信息与速度模式下的不一样。

-- 非对称的检测电阻或线圈会产生闪烁的 OLA或者OLB

-- 中断的电机线圈带来持续的输出线圈开路指示

-- 如果在过去的几个整步内，电流整流不能到达目标电流（没有电机接触或者速度超过PWM限制），一个或者两个指示器会有效。

如果有需要，做一个负载开路输出的测试，用快速模式，因为它输出最安全结果。静态模式下通过读 PWM_SCALE_SUM 的值可以检测线圈内阻。

4.6.2 PWM_SCALE_SUM 反馈电机状态

在自动整流模式下，PWM_SCALE_SUM 可以反馈电机的工作状态。因为这个参数反应了特定电流下实际加载到电机的有效电压，它取决与几个因素：电机负载，线圈阻抗，电源电压，电流设定等等。当检测这个值到达极限（255），电流整流器则无法满足电流要求。

4.7 自由旋转与被动刹车

静音模式为电机静态下提供了不同的模式。这些模式通过设置静态电流 IHOLD 为 0 并且通过FREEWHEEL选择需要的模式。这些需要的模式在一段时间(由 TPOWERDOWN与 IHOLD_DELAY 设置的延迟)后起效。当电机目标电流为零时，电流整流关闭以快速启动。自由旋转模式可以实现自 由 旋转与被动刹车。被动刹车是旋窝电流刹车，消耗电流很小，因为没有实际电流注入线圈。因此，被动刹车可以允许连续力矩下电机慢换向。

小结：当使用静音模式下使用电机来操作，最好带上机械负载电机表现会更好，这样可以防止空载时的机械振荡。

静音模式相关参数			
参数	功能描述	值	解释
en_Quick	静音模式无效位，输入脚 QUICK 与这个寄存器信号做异或运算	1	不使用静音模式
		0	使用静音模式
TPWMTHRS	设定了静音模式的最高速度，当工作到需要的 频率点是读出 TSTEP（两个微步之间的值）的值	0 … 1048575	TSTEP 低 于 TPWMTHRS 时静 音模式无效
PWM_LIM	设定限制从快速模式转到静音模式的电流毛刺	0 … 15	8 位输出幅度寄存器的高 4 位 (默认=12)
pwm_ autoscale	使用自动电流调整模式或预调整模式	0	与调整模式
		1	自动电流模式

pwm_ autograd	设定自动频率补偿	0	非自动模式，使用 PWM_GRAD 的值
		1	自动模式
PWM_FREQ	PWM 频率选择.设置越低效果越好. 在输出端测试到的斩波频率是 1/2 的f_{PWM}.	0	$f_{PWM}=2/1024\ f_{CLK}$
		1	$f_{PWM}=2/683\ f_{CLK}$
		2	$f_{PWM}=2/512\ f_{CLK}$
		3	$f_{PWM}=2/410\ f_{CLK}$
PWM_REG	用户定义的 PWM 幅度（衰减），当 *pwm_autoscale*=1，基于速度或者环路调制衰减	1 … 15	Results in 0.5 to 7.5 steps for *PWM_SCALE_AUTO* regulator per fullstep
PWM_OFS	用户定义的 PWM 幅度（偏置）基于速度调制以及 *PWM_OFFS_AUTO* 自动速度调制的初始值	0 … 255	*PWM_OFS*=0 将使基于电流设定的线性电流调整失效
PWM_GRAD	用户自定义的 PWM 幅度（衰减）基于速度调制以及 PWM_GRAD_AUTO 的初始化	0 … 255	重启值可以通过编写 OTP 来实现
FREEWHEEL	静态设置选项当电流为零时（I_HOLD=0）。只有静音模式下有效。这个选项将使电机能容易转动，两个线圈短路设定实现被动刹车。	0	普通操作
		1	自由旋转
		2	线圈通过 LS 短接
		3	线圈通过 HS 短接
PWM_SCALE_AUTO	可以读出静音模式下 PWM 电压的实际调整矫正值	-255... 255	快速模式下这个值被冻结
PWM_SCALE_AUTO，PWM_OFS_AUTO	允许监控自动调节并且允许 PWM_OFS 与决定PWM_GRAD 的初值	0...255	只读
TOFF	一般情况下做为使能驱动，实际的值对静音模式不起作用	0	驱动关闭
		1...15	驱动使能
TBL	比较器空白时间。这个时间需要安全地覆盖 开关时间以及检测电阻上的振铃时间。一般 设置 1 或者 2.对于高负载电容，设置 3. 低的设置允许静音模式	0	$16t_{CLK}$
		1	$24t_{CLK}$
		2	$32t_{CLK}$
		3	$40t_{CLK}$

六．检测电阻

可以通过选择一个合适的灵敏电阻来设置所需要的最大马达电流。下表为平均电流和选取的灵敏电阻对应关系。

Rsense 电阻的选择及对应的最大马达电流		
R_{SENSE} [Ω]	RMS 电流 [A] VREF=2.5V 或者悬空 ； IRUN=31， Vsence=0（标准）	适用的马达类型
1.00	0.23	300mA 马达
0.82	0.27	
0.75	0.30	
0.68	0.33	400mA 马达
0.50	0.44	500mA 马达
0.47	0.47	
0.39	0.56	600mA 马达
0.33	0.66	700mA 马达
0.27	0.79	800mA 马达
0.22	0.96	1A 马达
0.18	1.15	1.2A 马达
0.15	1.35	1.5A 马达
0.12	1.64	
0.10	1.92	

灵敏电阻需要谨慎的选择。马达所有电流都会流过灵敏电阻。由于斩波开关输出功率管，会产生脉冲电流。因此，需要一种低电感类型，如薄膜或合成电阻，以防止电压尖峰引起感应电压 输入端的振铃，从而导致测量结果不稳定。此外，低电感，低电阻的印刷电路板布局是必不可少 的。必须避免两个感测电阻的任何公共接地路径，因为这将导致两个电流感测信号之间的耦合。 一个巨大的地平面是最好的。

在马达待机状态下，灵敏电阻需要能够传输峰值线圈电流，除非待机功耗降低。在正常情况 下，由于在慢衰减阶段没有电流流过感测电阻，感测电阻的导电性小于线圈的均方根电流。对于 大多数应用，0.5W 类型足以达到 1.2A rms 电流。

注意使用对称的感测电阻布局和长度相同的短而直的感测电阻轨迹。匹配良好的感测电阻确保最佳性能。布局紧凑，接地面积大，避免寄生电阻效应。

七．电机电流控制

电机的电流由感应电阻器的电阻设置。有几种情况允许降低电机电流，例如为了适应不同的电机，或在静止或低负载情况下。

测量电机电流的方法			
方法	参数	范围	基本应用
VREF 脚电压（9.1 章）	VREF 可用来调整 IRUN 和IHOLD，可以通过 GCONF.i_scale_analog 使能	2.5V: 100%... 0.5V: 20% >2.5V 或者悬空: 100% <0.5V: 不推荐	-微调电机电流以适应电机类型 -通过 poti 手动调节 -延迟或者软启动 -静态电流降低（仅适用于扩展循环）
ENN 脚	禁用/启用输出驱动	0： 电机使能 1： 电机禁用	-禁用电机以允许自由旋转
PDN_UART 脚	禁用/启用 静止电流降低 至IHold	0：电流降低已启用 1：禁用	-启用电流减小以减少静止状态温度升高
OTP 存储	OTP_IHOLD OTP_IHOLDDELAY	9%至 78%静止电流。 减少约 300 ms 至2.5 s	-调整电流来适应各种应用， 从而实现高效率，低发热
OTP 存储	OTP_internalRsense	0: 使用外部感应电阻 1: 使用内部感应电阻	-在 BOM 上节省两个感应电阻， 通过一个便宜的 0603 电阻设置 电流
UART 接口	IHOLD_IRUN TPOWERDOWM OTP	IRUN,IHOLD 满量程电流的1/32 至 32/32	-运行和保持（静止）电流的精细编程 -改变特定情况下电机电流的 IRun -设置 OTP 选项
UART 接口	CHOPCONFIG.vsense flag	0: 正常，最稳定 1: 降低电压	-使用更小的 0.25W 电阻时设置 vsense 得到更低的电压

选择合适的感应电阻器在满电流标度（Vref=2.5V）下为电机提供足够的电流。默认的缩放比例（irun=31）。

独立模式 rms 运行电流计算公式：

$$\frac{I_{RMS}}{R_{SENSE}} = \frac{300mV}{+30m\Omega} * \frac{1}{\sqrt{2}} * \frac{V_{VREF}}{2.5V}$$

UART 配置或者保持电流设置 RMS 电流计算公式：

$$I_{RMS} = \frac{CS+1}{32} * \frac{V_{FS}}{R_{SENSE} + 30m\Omega} * \frac{1}{\sqrt{2}}$$

CS 是 ihold 和 irun 的当前比例设置

V_{FS} 是由 vsense 控制位确定的满幅电压（请参阅电气特性、vsrtl 和 vsrth）。默认值为 300mV

。当设置为模拟 V_{FS}（I_scale_analog=1，默认值），产生的电压 V_{FS} 的计算公式如下：

$$V_{FS}' = V_{FS} * \frac{V_{VREF}}{2.5V}$$

其中 V_{VREF} 值为管脚脚 VREF 上的电压在 0 到 2.5V 之间。
为了得到最佳精度的电流设置，需要在应用中不断的测量和微调电流。

马达电流控制参数			
参数	描述	设置	备注
IRUN	电机运行时的电流刻度。根据内部正弦波表产生线圈电流值。对于高精度电机操作，使用 16 到 31 之间的电流比例因子，因为减小电流值会使微步变得更粗，从而降低有效的微步分辨率。	0...31	比例因子 1/32，2/32，…32/32 irun 在独立模式下为满刻度 （设置 31）
IHOLD	与 IRUN 相同，电机处于静止状态下起作用		
IHOLD DELAY	允许从运行电流平稳降低到保持电流。iholdDelay 以 2^18 时钟的增量控制断电后电机断电的时钟周期数：0=瞬间断电，1..15：每电流步的电流减少延迟是 2^18 时钟的倍数。 示例：当使用 irun=31 和 ihold=16 时，保持电流减小需要 15 个电流步骤。因此，iholdDelay 设置为 4 会导致断电时间为 4*15*2^18 个时钟周期，即在 16MHz 时钟频率下大约一 秒钟	0	Instant IHOLD
		1 ... 15	$1*2^{18}$ ...$15*2^{18}$ 每个电流衰减时钟数
TPOWER DOWN	设置从静止模式（stst）到电机关闭的延迟时间。时间范围约为 0 到 5.6 秒。	0 ... 255	0 ... ((2^8)-1)*2^18 t_{CLK} 最小设置为 2，以允许自动调整 PWM_OFFS_AUTO
vsense	允许控制满标度电流的感测电阻电压范围。低电压范围可以降低感测电阻的功耗。	0	V_{FS}=0.3V
		1	V_{FS}=0.165V

7.1 模拟电流设置 VREF

当需要高灵活度的输出电流大小时，我们可以通过驱动器的模拟输入来进行电流控制，而不是选择一组不同的感测电阻或通过接口使用 irun 或 ihold 参数缩小运行电流。这样，一个简单的 分压器实现一个电路板适应不同的电机。

VREF 调整电机电流

6609为电流控制提供一个内部参考电压，直接从 5伏电源分压。或者，也可以使用外部参考电压。运用到斩波比较器的参考电压会按比例降低。斩波比较器将 SENSE1和 SENSE2上的电压与用于电流调节的标度参考电压进行比较。当启用 GCONF中 的 I_scale_analytic（I_scale_模拟）时（默认），VREF 上的外部电压被放大和过滤，并用作参考电压。2.5伏电压（或 2.5伏和 5伏之间的任何电压）提供与内部参考电压相同的电流标度。0伏 和 2.5伏之间的电压则能够线性地缩放电流。在 0 和感应电阻设置定义的电流标度之间。不建议参考电压工作在低于 0.5 伏至 1伏 ，因为较低VREF电压由数字电路和电源纹波会对斩波精度产生较大的影响。为了获得最佳精度，请以所需最大值的方式选择感测电阻，Vref 选 择 在2V至 2.4V之间，确保为电机正常运行电流。

VREF 驱动方法

向 VREF提供电压的最简单方法是使用来自稳定电源电压的分压器。或微控制器的 DAC输出。同时还允许采用 PWM信号进行电流控制。PWM在VREF引脚上使用附加的 R/C低通转换为模拟电压。PWM信号控制模拟电压。选择 R和C值形成一个低通滤波器，使用脉宽调制频率需要远高于 10 kHz。另外，VREF 提供 3.5KHz 带宽的内部低通滤波器。

使用低参考电压（例如低于1V），如果采用大电流驱动器去适应低电流电机将导致模拟性能下降。为了达到最佳效果调整感应电阻以适合所需的电机。

3 UART串口通讯配置寄存器举例

通常，大多数寄存器都可以使用默认值。只有少数寄存器是客户常用的，其他寄存器可以忽略。

6609 写寄存器函数:

```
void Write_6609_REG(uint8_t WReg_addrr, uint32_t Write_data)
{
// Write 6609 register
  Syn = 0x05;  // Synchronization + reservation
  Slave = 0x00; // 8-bit slave address
  CRC_data = 0x00;  // CRC check initialization 0
  Write_data3 = Write_data >> 24;
  Write_data2 = Write_data >> 16;
  Write_data1 = Write_data >> 8;
  Write_data0 = Write_data;  //

  Write_data_temp[0]=Syn;
  Write_data_temp[1]=Slave;
  Write_data_temp[2]=WReg_addrr|0x80;
  Write_data_temp[3]=Write_data3;  // Write data high 4 bits
  Write_data_temp[4]=Write_data2;  // Write data high 3 bits
  Write_data_temp[5]=Write_data1;  // Write data high 2 bits
  Write_data_temp[6]=Write_data0;  // Write data high 1 bits
  Write_data_temp[7]=CRC_data;     //CRC Check value

  swuart_calcCRC(Write_data_temp,8);  // Calculate CRC check value


  Delay_ms(1);  //delay 1ms
  Send_char(Write_data_temp,8);  // Send 8 bytes of data to 6609 through serial
  port
}
```

6609 读寄存器函数:

```
void Read_6609_REG(uint8_t RReg_addrr)
{

  Syn = 0x05;  // Synchronization + reservation
  Slave = 0x00; //8-bit slave address
  RReg_addr = RReg_addrr; //Read register address
  CRC_data = 0x00;

  Write_data_temp[0]=Syn;
  Write_data_temp[1]=Slave;
  Write_data_temp[2]=RReg_addr&0x7F;
  Write_data_temp[3]=CRC_data;

  swuart_calcCRC(Write_data_temp,4);  // Calculate CRC check value
  Send_char(Write_data_temp,4);  // Send 4 bytes to 6609 through serial port
}
```

For example, use SCC mode:

比如，需要将0x00寄存器的第2位更改为1：

```
Write_6609_REG(0x00,  0x00000105);// 0x00 is a register with address ,0ther
                                  register values should not be changed at will,
                                  but the default value should be used
```

For example, use SC2 mode:

比如，需要将0x00寄存器的第2位更改为0：

```
Write_6609_REG(0x00,  0x00000101);// 0x00 is a register with address ,0ther
                                  register values should not be //changed at
                                  will, but the default value should be used
```

由于外部MS1和MS2只能细分为16，如果要使用更高的细分128（最高可到256），需要执行以下操作:

```
Write_6609_REG(0x00, 0x00000185); //  mstep_reg_select set to 1

Write_6609_REG(0x6c, 0xc4010053); //  intpol set to 0

Write_6609_REG(0x6c, 0xc1010053); //  MRES set to 0x0001
```

读寄存器举例:

Read 0x00 register:

Read_6609_REG(0x00);

然后通过串口中断接收8个字节数据:

```
void USART1_IRQHandler(void)          // Serial port interrupt reception
{


        if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
        {
            USART_ClearITPendingBit(USART1,USART_IT_RXNE);

          {
         RxBuffer1[RxCounter1++] = USART_ReceiveData(USART1);


              if(RxBuffer1[0] == 0x05)
              {

                 if( (RxBuffer1[1] == 0xFF) && (RxCounter1>=2)) //
                 {
                      if(RxCounter1==RxBufferSize1)
                      {
                          RxCounter1 = 0;
                             Get_CRC = RxBuffer1[7];


                          swuart_calcCRC(RxBuffer1,8);


                          if( (RxBuffer1[7] == Get_CRC) )
                          {

                             for(Kii=0; Kii<RxBufferSize1; Kii++)
                             {
                          Read_data_temp[Kii] = RxBuffer1[Kii];
                             }

                             rec_f=1;
```

```
                                    }
                                    else
                                    {

                                        RxCounter1 = 0;


                                      Clear_Buff(RxBuffer1,RxBufferSize1);

                                    }



                                }

                            }
                    else if( (RxBuffer1[1] != 0xFF) && (RxCounter1>=2))
                            {
                                RxCounter1 = 0;
                            }
                            else
                            {
                                ;
                            }

                        }

                    else
                    {
                        RxCounter1 = 0;

                    }
                }


        }


}
```

Read_ data_ Temp is an array that receives 8 bytes