Application Notes
ZHCAA13 – 6.2020

基于 C2000 InstaSPIN FOC 下桥三电阻采样方式的电机电流重构方法

JOHNSON CHEN/ EP FAE

摘要

在三相电机 FOC 控制算法里，电机三相电流重构对电流环的稳定性和电机运行效率都非常关键。在采用下桥三电阻采样方案时，如果没有有效方法的重构三相电流，电机在高调制率下将无法稳定运行。传统的方法是限制调制率，从而保证电机不会运行在高调制率下，这个方法牺牲了母线电压的利用率，从而导致电机无法高速运行或者高速运行效率降低。

本文重点介绍针对采用下桥三电阻采样方式的一种电流重构算法，通过本文介绍的电流重构算法，可以使电机运行在全调制范围（包括过调制情况下），可以充分利用母线电压，从而可以让电机运行在高速区间并提升电机高速运行效率。

本方法已在 C2000 InstaSPIN FOC 芯片如 TMS320F28069F， TMS320F28027F，TMS320F28054F 上实现，并进行验证。

Contents

Figures

1 介绍

在某些应用场景如空调，电动自行车，水泵等，因为成本的关系，通常使用基于下桥的三相电阻采样方案（如图 1 所示），此时只有下桥开通时才有电流流过检流电阻，因此只有在下桥开通时才能采样到有效电流。

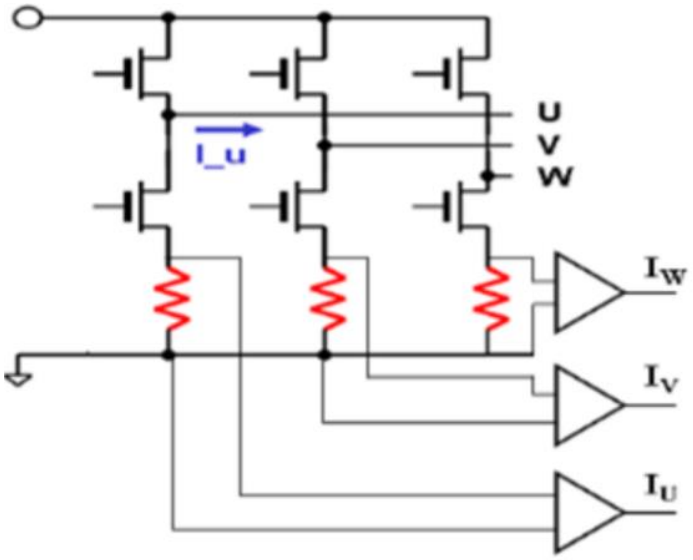

Figure 1. 下桥三电阻采样方案

使用传统的三相电流重构算法，在高调制率时因下桥开通时间太短，所以无法采到有效三相电流，从而导致电机无法正常运行。此时通常通过限制调制率的方法来保证电机可靠运行，但这样会限制母线电压利用率，从而影响电机运行效率。

本文将介绍一种新的三相电机电流重构算法，通过本文介绍的电流重构算法，电机可以运行在全调制范围内（包括过调制情况下），可以充分利用母线电压，从而使电机可以运行在高速区间并提升了电机高速运行时的效率。

本文会重点介绍此方法的原理，基于 C2000 InstaSPIN FOC 的软件实现， 及相应测试结果。

本文分析基于以下场景：

1. PWM 定时器的计数值等于零时触发 ADC 采样
2. PWM 输出高有效

其中 $T_{Minwidth}=T_{Deadtime}+T_{Delay}+T_{Adc\_sample}$。

$T_{Deadtime}$ 为死区时间，T_{Delay} 为 IGBT/MOSFET 导通延迟时间，T_{Adc_sample} 为 ADC 采样三通道电流所需时间。

2 传统三相电流重构算法介绍

传统电流重构方法在三相下桥开通时采样两相电流，基于 Iu+Iv+Iw=0 原理，计算出另外一相电流。由于 PWM 计数器通常采取对称计数模式，通常在 PWM 定时器的计数值等于零或者等于周期值时触发 ADC 进行采样。

下图为下桥三相 PWM 控制输出波形。

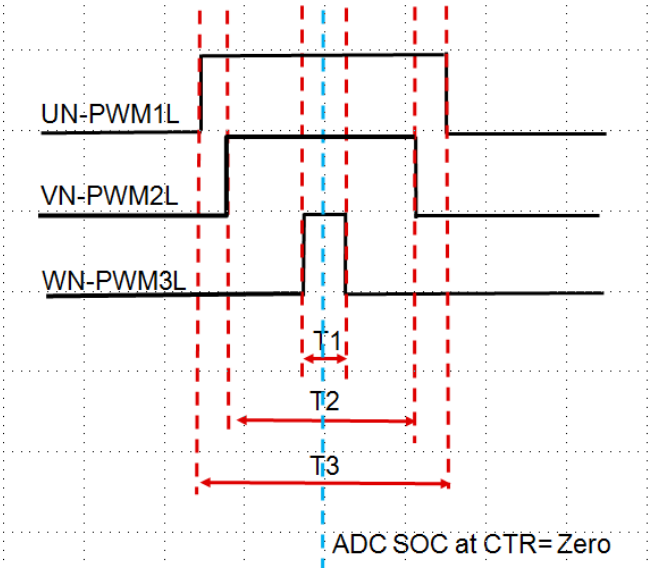

Figure 2. 传统电流重构方法采样点

传统电流重构方法是：当 T1< = T2 <=T3 时，只采样 U 相和 V 相电流，那么 W 相电流 Iw=-(Iu+Iv).

此方法在下面情况下会有问题：

1. 当 T1 < **T**MinWidth,当采样 U 相和 V 相电流时，W 相将会有 PWM 开关，因此采样到的 U 相和 V 相电流会有噪音。

2. 当 T1 < **T**Minwidth 且 T2 < **T**Minwidth, 当采样 U 相电流时，V 相和 W 相将会有 PWM 开关,因此采样的 U 相电流也会有噪音。

3 新三相电流重构算法介绍

在高调制率下，三相下桥 PWM 开通时间较短，在采样其中某相电流时可能会碰到其它相 PWM 的开关，因开关噪音导致采样不到真实的电流信号，因此无法重构电流。

因此我们需要寻找合适的采样点来采样电流信号。

新电流重构算法如下：

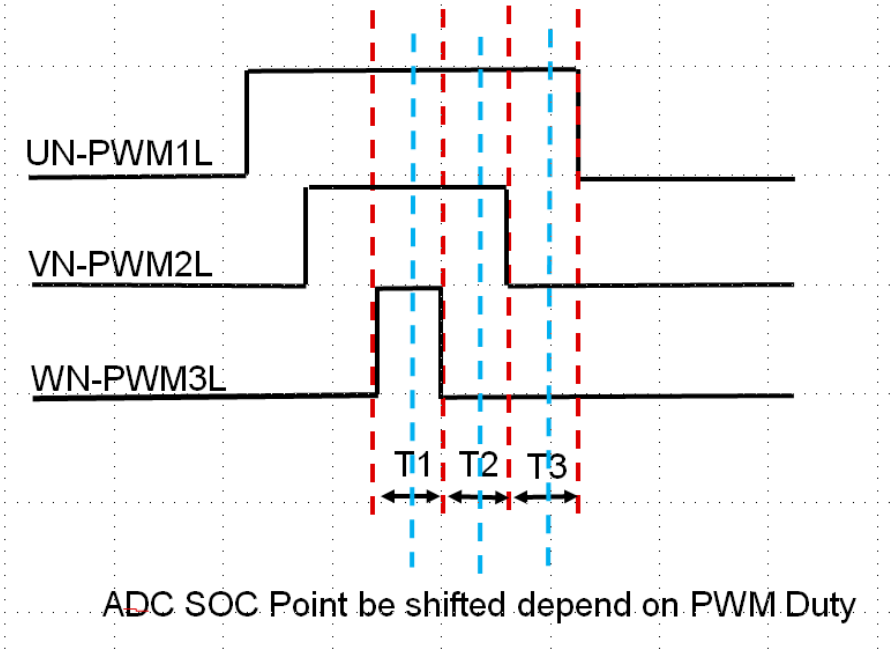

Figure 3. 新电流重构方法采样点

基于不同的 PWM 占空比输出情况, 使用 CMPB 来**选择合适的** ADC 触发**采样点**。

情况一：当 T1>TMinWidth, 在 T1 区域内触发 ADC 采样三相相电流。这时候所有采样到的电流信号都是有效的，可以直接使用。

情况二： T1<=TMinWidth 且 T2 >=TMinwidth，没有足够的时间采样 W 相电流，因此在 T2 区域内触发 ADC 采样 U 相和 V 相电流信号，W 相电流 Iw=-(Iu+Iv)。

情况三：T1 <=TMinwidth 且 T2 < TMinwidth，在 T1 和 T2 区间均没有没有足够采样三相电流，因此在 T3 区域内触发 ADC 采样三相电流信号。

此时只有 U 相电流信号是有效的, V 相和 W 相电流必须忽略，所以无法通过 Iu+Iv+Iw=0 计算出另外两相电流。因从考虑从软件算法上来解决这个问题，软件后台一直会对 UVW 三相电流进行低通滤波，此时 V 相和 W 相电流将使用低通滤波的结果。

4 软件框图

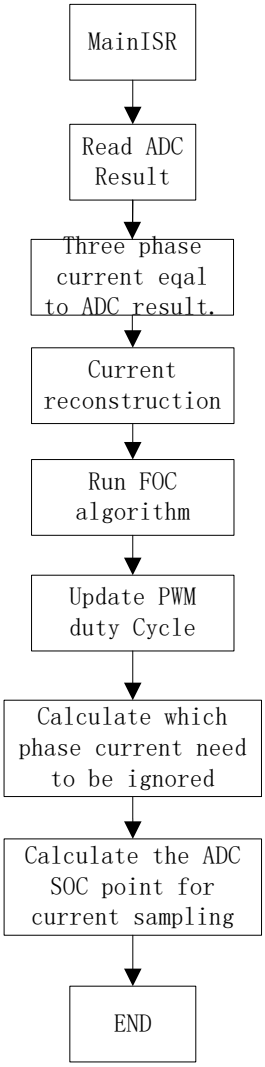

Figure 4. FOC 中断函数软件流程图

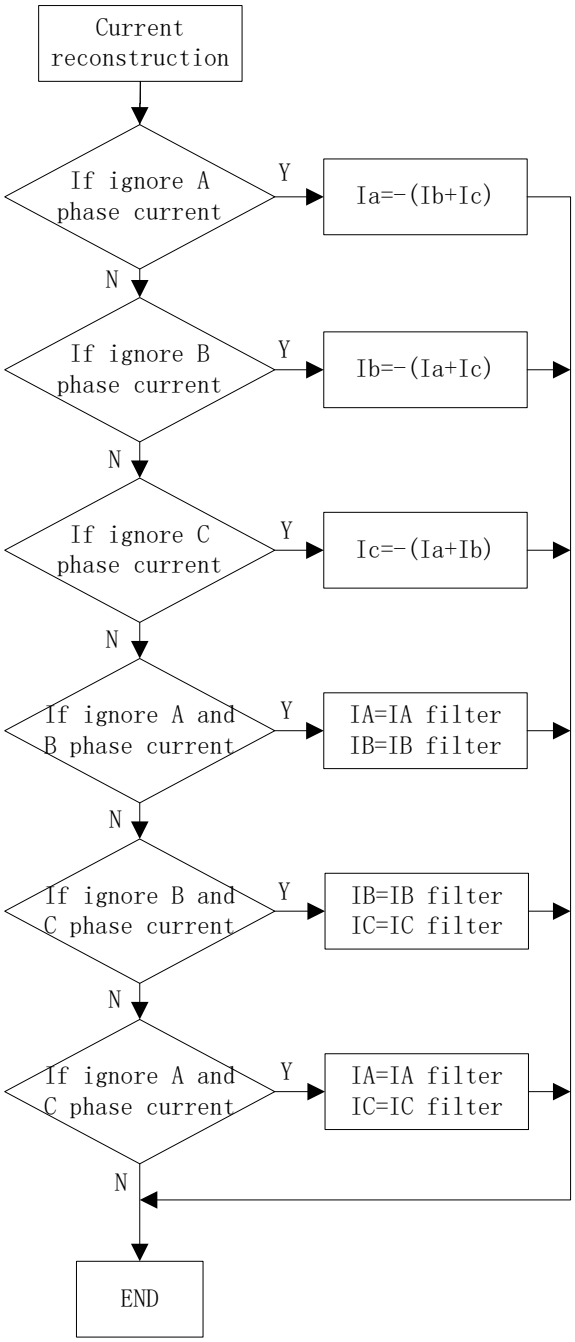

Figure 5.　电流重构流程图一

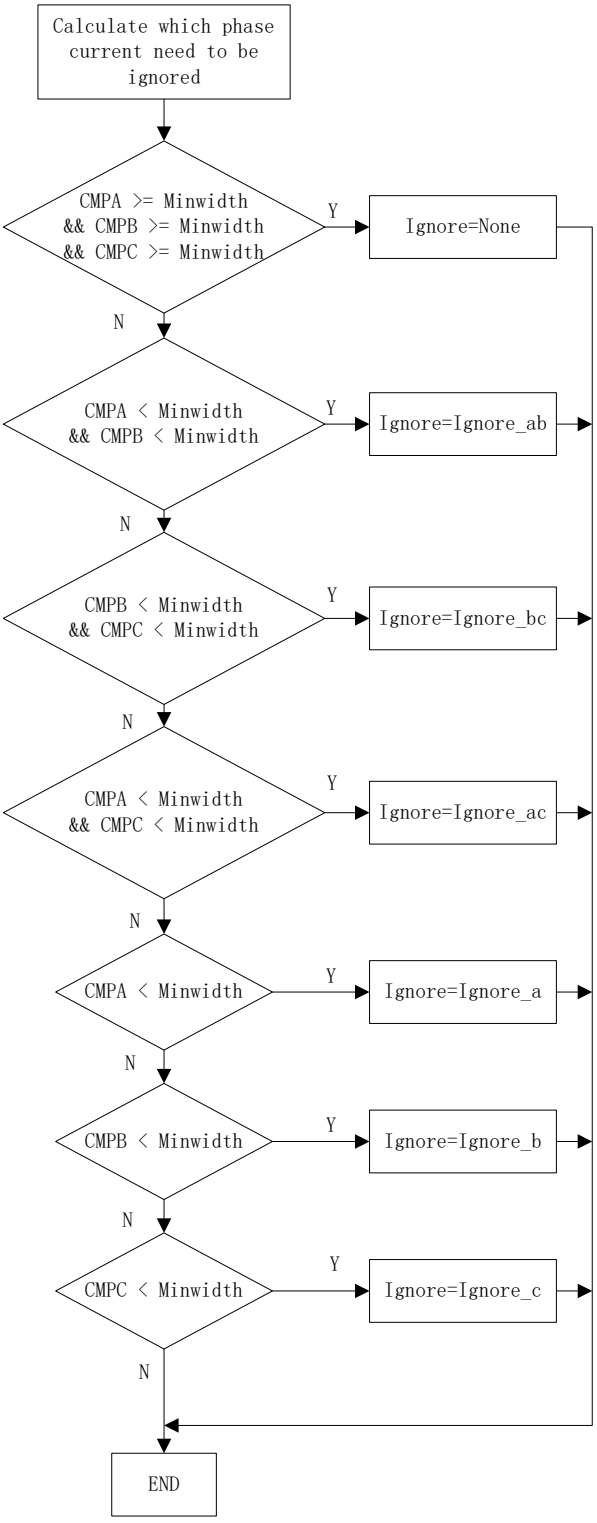

Figure 6.　电流重构流程图二

5 软件实现

```
//Since this S/W using CMP1B to triggle ADCSOC, so please make sure the CMPB be loading immediate,
//when configure the PWM in drv.c. If the HW didn't using ePWM1 for motor driver,please change the
//to using other ePWM module CMPB event to tirggle ADCSOC.
// setup the Counter-Compare Control Register (CMPCTL)
PWM_setShadowMode_CmpB(obj->pwmHandle[cnt],PWM_ShadowMode_Immediate);

//variable define in proj_lab10a.c.
//////////////////////////////////////////
// need to trunning based on H/W
uint16_t gCmpOffset=200; //Deadtime+delay, need to turning based on HW
uint16_t gCmpOffset2=200; //Deadtime+delay,can be same as gCmpOffset, for ignore two phase current
                          //case.
uint16_t minwidth2=400; //Deadtime+delay+ TADC sampling, need to turning based on HW
uint16_t minwidth=200; //（Deadtime+delay+ TADC samplin)/2, need to turning based on HW
uint16_t gIavg_shift = 1; //Filter coefficient, need to turning based on application
MATH_vec3 gIavg;
uint16_t pwmValue_1 = 0;
uint16_t pwmValue_2 = 0;
uint16_t pwmValue_3 = 0;
//////////////////////////////////////

// Initialize and setup the 100% SVM generator
svgencurrentHandle = SVGENCURRENT_init(&svgencurrent,sizeof(svgencurrent));
SVGENCURRENT_setMinWidth(svgencurrentHandle, minwidth);
gIavg.value[0] = 0;
gIavg.value[1] = 0;
gIavg.value[2] = 0;

//code after DRV_readAdcData(drvHandle,&gAdcData) in MainISR.
// convert the ADC data
DRV_readAdcData(drvHandle,&gAdcData);
// run the current reconstruction algorithm
SVGENCURRENT_RunRegenCurrent(svgencurrentHandle, (MATH_vec3 *)(gAdcData.I.value));
//////////////////////////////////////////////////////////////////////////////////////////////
//For HVM isssue
//////////////////////////////////////////////////////////////////////////////////////////////
gIavg.value[0] += (gAdcData.I.value[0] - gIavg.value[0])>>gIavg_shift;
gIavg.value[1] += (gAdcData.I.value[1] - gIavg.value[1])>>gIavg_shift;
gIavg.value[2] += (gAdcData.I.value[2] - gIavg.value[2])>>gIavg_shift;
if(svgencurrent.IgnoreShunt == ignore_ab)
{
gAdcData.I.value[0] = gIavg.value[0];
gAdcData.I.value[1] = gIavg.value[1];
}
else if(svgencurrent.IgnoreShunt == ignore_ac)
{
gAdcData.I.value[0] = gIavg.value[0];
gAdcData.I.value[2] = gIavg.value[2];
}
else if(svgencurrent.IgnoreShunt == ignore_bc)
{
gAdcData.I.value[1] = gIavg.value[1];
gAdcData.I.value[2] = gIavg.value[2];
```

```
}
//End
////////////////////////////////////////////////////////////////////////////////////////////////


//code after DRV_writePwmData(drvHandle,&gPwmData) in MainISR.
// write the PWM compare values
DRV_writePwmData(drvHandle,&gPwmData);
////////////////////////////////////////////////////////////////////////////////////////////////
//For HVM isssue
////////////////////////////////////////////////////////////////////////////////////////////////
pwmValue_1 = (DRV_readPwmCmpA(drvHandle,PWM_Number_1) + DRV_readPwmCmpAM(drvHandle,PWM_Number_1))>>1;
pwmValue_2 = (DRV_readPwmCmpA(drvHandle,PWM_Number_2) + DRV_readPwmCmpAM(drvHandle,PWM_Number_2))>>1;
pwmValue_3 = (DRV_readPwmCmpA(drvHandle,PWM_Number_3) + DRV_readPwmCmpAM(drvHandle,PWM_Number_3))>>1;
R_CMP1A = DRV_readPwmCmpA(drvHandle,PWM_Number_1);
R_CMP2A = DRV_readPwmCmpA(drvHandle,PWM_Number_2);
R_CMP3A = DRV_readPwmCmpA(drvHandle,PWM_Number_3);
// run the current ignore algorithm
// SVGENCURRENT_RunIgnoreShunt(svgencurrentHandle,pwmValue_1,pwmValue_2,pwmValue_3);
DRV_setTrigger(SVGENCURRENT_getMinWidth(svgencurrentHandle),minwidth2,gCmpOffset,gCmpOffset2);
//
DRV_SetADCSOC_Point(drvHandle,svgencurrentHandle);
//end
////////////////////////////////////////////////////////////////////////////////////////////////
```

本算法已经在motorware15开始的版本里面正式实现，详细代码可以参考Lab10a --- Overmodulation 例程。

6 测试结果

测试条件：

a. motorware_1_01_00_10

b. TMS320F28069F

c.调制率：gMotorVars.OverModulation = _IQ(1.25) // 最大为 4/3=1.333333333

d.测试电机：松下单转子压缩机型号：5RS092ZKB21

e.电机速度：4.5KRPM

实际测试波形如下：

传统电流重构算法波形：

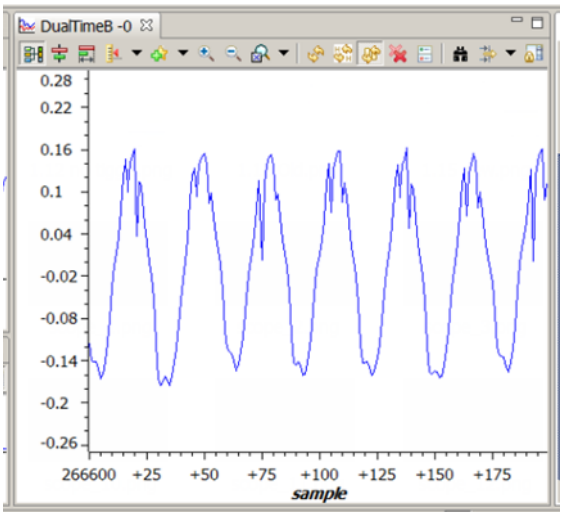

Figure 7. 传统电流重构方法相电流波形

从图 7 可以看到，使用传统电流重构算法，在 1.25 调制率下，实际采样的电流采样会有较大噪音，无法正常重构电机电流。

使用本文新电流重构方法波形：

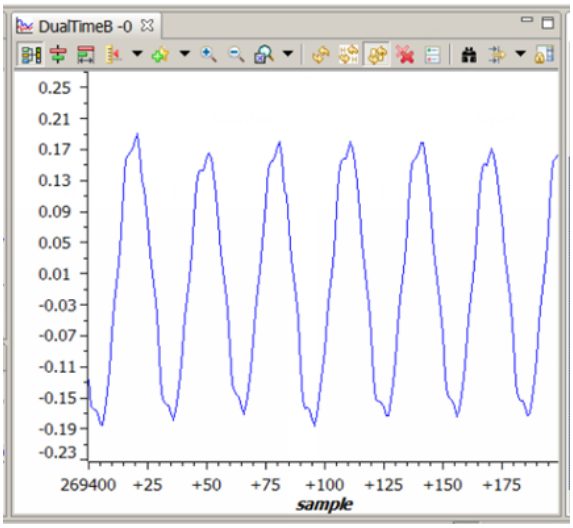

Figure 8.　新电流重构算法相电流波形

从图 8 可以看到，使用新电流重构算法，在 1.25 调制率下，PWM 100% 输出时，依然可以有效重构电机电流。

7 注意事项

不同版本的 motorware 因系数调整，最大调制率值(USER_MAX_VS_MAG_PU)会不一样,具体参考不同版本 USER_MAX_VS_MAG_PU 注释。

motorware_1_01_00_13 为 4/3=1.3333333333333

motorware_1_01_00_16 为 2/3=0.6666666666667

8 参考文献

1. ***TMS320F28069F Piccolo Technical Reference Manual (SPRUH18)***
2. ***TMS320F28054F InstaSPIN-FOC™ Software Technical Reference Manual (*SPRUHW0*)***
3. ***TMS320F28027F Piccolo Technical Reference Manual (*SPRUI09*)***
4. ***InstaSPIN-FOC™ and InstaSPIN-MOTION™ User's Guide (SPRUHJ1)***

重要声明和免责声明

TI 均以“原样”提供技术性及可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证其中不含任何瑕疵，且不做任何明示或暗示的担保，包括但不限于对适销性、适合某特定用途或不侵犯任何第三方知识产权的暗示担保。

所述资源可供专业开发人员应用TI 产品进行设计使用。您将对以下行为独自承担全部责任：(1) 针对您的应用选择合适的TI 产品；(2) 设计、验证并测试您的应用；(3) 确保您的应用满足相应标准以及任何其他安全、安保或其他要求。所述资源如有变更，恕不另行通知。TI 对您使用所述资源的授权仅限于开发资源所涉及TI 产品的相关应用。除此之外不得复制或展示所述资源，也不提供其它TI或任何第三方的知识产权授权许可。如因使用所述资源而产生任何索赔、赔偿、成本、损失及债务等，TI对此概不负责，并且您须赔偿由此对TI 及其代表造成的损害。

TI 所提供产品均受TI 的销售条款 (http://www.ti.com.cn/zh-cn/legal/termsofsale.html) 以及ti.com.cn上或随附TI产品提供的其他可适用条款的约束。TI提供所述资源并不扩展或以其他方式更改TI 针对TI 产品所发布的可适用的担保范围或担保免责声明。

邮寄地址：上海市浦东新区世纪大道 1568 号中建大厦 32 楼，邮政编码：200122