PAW3395DM-T6QU：光学游戏导航芯片

一般说明

PAW3395DM-T6QU是原相影像的新型低功耗高端游戏导航芯片，采用16引脚模压引线框架DIP封装，带有照明源。它提供一流的游戏体验，具有高速、高分辨率、高精度和可选升降检测高度的增强功能，可满足专业游戏玩家的需求。它旨在与LM19-LSI或L0AE-LSI1一起使用以实现最佳性能。

主要特征

- 运行模式（HP模式）下典型1.7 mA的低功耗
- 带850nm照明光源的16引脚模压引线框架DIP封装
- 增强的可编程性
  - 游戏模式
    - 高性能模式（HP模式）
    - 低功耗模式（LP模式）
    - 有线游戏模式
  - 提升检测选项
    - 1mm和2mm设置
    - 手动升降切断校准
- 可选分辨率高达26000 cpi，步长为50 cpi
- 角度捕捉
- 角度可调
- 分辨率误差为0.4%（典型值）在5000cpi上QCK高达200ips
- 高速运动检测650ips*和加速度50g*
- 自调节可变帧率以获得最佳性能
- 内部振荡器——无需时钟输入
- 4线串行端口接口(SPI)
- 运动中断输出

应用

- 有线和无线光学游戏鼠标
- 集成输入设备

关键参数

参数	值
电源范围	VDD：1.8至2.1V VDDIO：1.8至3.3V
镜头倍率	1:1
界面	4线串口接口
典型工作电流@ VDD = 1.9V 注：包括LED电流	运行：1.7 mA（HP模式） 运行：1.3 mA（LP模式）Rest1：580 µA Rest2: 11 µA Rest3: 6 µA 掉电：4µA
解析度	高达26000 cpi
跟踪速度	650* ips
加速度	50* g
外形尺寸（封装与LM19-LSI镜头组装）	10.90 x 16.20 x 9.81 mm^3

注意：* - HP模式

订购信息

零件号	描述	封装类型	包装类型	MOQ
PAW3395DM-T6QU	光学游戏导航芯片	16针拨码	管子	1,000
LM19-LSI	圆形镜片	圆形镜片	托盘	1,000
L0AE-LSI1	修剪镜头	修剪镜头	托盘	1,000

RoHS compliant

表中的内容

图列表

表格列表

1.0 介绍

1.1 芯片概览

PAW3395DM-T6QU是一款针对高端有线和有线游戏鼠标的光学导航芯片。它封装含一个作为图像采集系统(IAS)的像素阵列、一个数字信号处理器(DSP)、一个4线串行端口、一个电源控制电路和与红外LED集成在一个封装中的内置LED驱动器，如图所示框图。该芯片通过光学获取连续表面图像（帧）并通过数学方法确定运动的方向和幅度来测量位置变化。IAS通过镜头和照明系统获取微观表面图像。这些图像由DSP处理以确定运动的方向和距离。DSP计算Δx和Δy相对位移值。外部微控制器从芯片串行端口读取Δx和Δy信息。然后微控制器将数据转换为USB或RF信号，然后再将它们发送到主机PC或游戏机。

注意：在本文档中，PAW3395DM-T6QU被称为芯片。

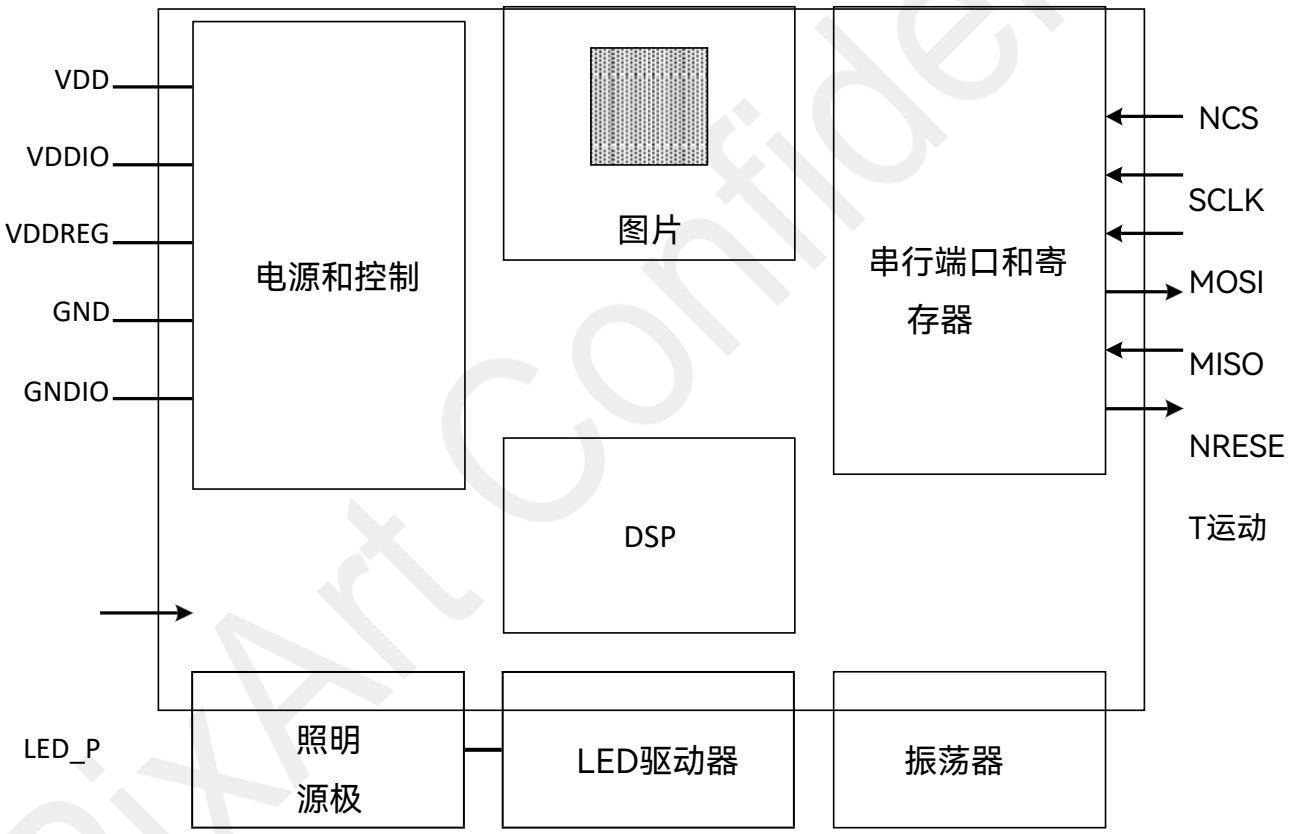


图1.框图

1.2 引脚配置

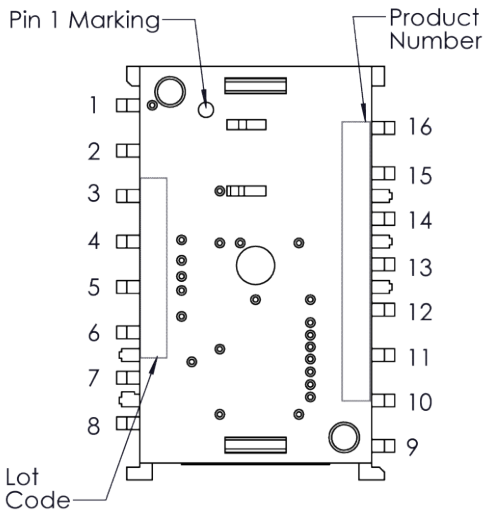


图2.器件引脚排列

表1.引脚定义

引脚编号	功能	象征	类型	描述
1个	保留	NC	NC	无连接
2个	保留	NC	NC	无连接
3	电源接地	GND	接地	接地
4个	电源电压	VDD	力量	输入电源
5个	低压差输出	VDDREG	力量	数字核心的LDO输出（仅供内部使用）
6个	保留	NC	NC	无连接
7	输入/输出电压	VDDIO	力量	输入/输出电源
8个	输入输出地	GNDIO	接地	输入输出地
9	运动输出	运动	输出	运动检测
10	4-wire SPI	SCLK	输入	串行数据时钟
11		MOSI	输入	串行数据输入
12		MISO	输出	串行数据输出
13		NCS	输入	片选（低电平有效）
14	重置控制	复位	输入	芯片复位（低电平有效）
15	LED	LED_P	输入	LED阳极
16	保留	NC	NC	无连接

2.0 电气规格

2.1 监管要求

- 当使用铁氧体磁珠并遵循PixArt的建议组装到带有屏蔽USB电缆的鼠标中时，通过FCC“第15部分，B子部分，B类”，“ICES-003:2016第6期，B类”和“ANSI C63.4:2014”。
- 通过IEC 62471: 2006灯和灯系统的光生物安全。

2.2 绝对最大额定值

表2.绝对最大额定值

参数	象征	最小值	最大限度	单位	笔记
贮存温度	T_S	-40	85	°C	
铅焊温度	T_{SOLDER}		260	°C	持续7秒，低于座椅平面1.6毫米
电源电压	VDD	-0.5	2.1	V	
	VDDIO	-0.5	3.3	V	
ESD	ESDHB		2	kV	所有引脚上的人体模型
电压	V_{IN}	-0.5	3.3	V	所有I/O引脚

笔记:

1. 在室温下。
2. 最大额定值是指超过该值可能会损坏设备的值。
3. 暴露于这些条件或超出指示的条件可能会对设备可靠性产生不利影响。不暗示绝对最大额定条件下的功能操作。

2.3 推荐工作条件

表3.推荐工作条件

参数	象征	最小值	类型。	最大限度	单位	笔记
工作温度	T_A	0		40	°C	
电源电压	VDD	1.8	1.9	2.1	V	不包括电源噪声
	VDDIO	1.8	1.9	3.3	V	不包括电源噪声。（VDDIO必须等于或大于VDD）
电源上升时间	t_{RT}	0.15		20	ms	0至VDD分钟
电源噪声峰峰值	V_{NA}			100	mV	10 kHz —75 MHz
串口时钟频率	f_{SCLK}			10	MHz	50%占空比
与镜头参考的距离 平面到跟踪表面	Z	2.2	2.4	2.6	mm	
速度 ▪ 高性能模式 ▪ 低功耗模式 ▪ 有线游戏模式 ▪ 办公模式	S	 650 480 650 200			ips	在45度运行模式
加速度 ▪ 高性能模式 ▪ 低功耗模式 ▪ 有线游戏模式 ▪ 办公模式	A	 50 40 50 10			g	
分辨率错误 ▪ 高性能模式 ▪ 低功耗模式 ▪ 有线游戏模式	Res_{Err}		 0.4 0.4 0.4		%	在5000cpi的QCK上高达200ips
提升截止1mm设置	$Lift_{1mm}$		1个		mm	PixArt标准游戏界面
提升截止2mm设置	$Lift_{2mm}$		2个		mm	PixArt标准游戏界面

2.4 交流电气规格

表4.交流电气规格

超过推荐工作条件的芯片电气特性。典型值为25℃、VDD = 1.9V且VDDIO=1.9V

参数	象征	最小值	典型的	最大限度	单元	笔记
重置后的运动延迟	$t_{MOT-RST}$	50			ms	从重置到有效运动，假设存在运动
关闭	t_{STDWN}			500	ms	从关断模式激活到低电平当前的
从关机中唤醒	t_{WAKEUP}	50			ms	从关闭模式无效到有效运动。 注意：RESET必须在关闭后断言。 请参阅部分《关机注意事项》
味噌上升时间	t_{r-MISO}		6		ns	C_L = 20pF
味增下降时间	t_{f-MISO}		6		ns	C_L = 20pF
SCLK后的MISO延迟	$t_{DLY-MISO}$			35	ns	从SCLK下降沿到MISO数据有效 C_L = 20pF
味噌保持时间	$t_{hold-MISO}$	25			ns	数据保持到下一个SCLK下降沿边缘
MOSI保持时间	$t_{hold-MOSI}$	25			ns	SCLK上升沿后数据有效的时间量
MOSI建立时间	$t_{setup-MOSI}$	25			ns	从数据有效到SCLK上升沿
写命令之间的SPI时间	t_{SWW}	5个			µs	从第一个数据字节最后一位的SCLK上升沿到第二个数据字节最后一位的SCLK上升沿
写入和读取之间的SPI时间命令	t_{SWR}	5个			µs	从第一个数据字节的最后一位SCLK上升，到最后一个SCLK上升第二个地址字节的位
读取和后续命令之间的SPI时间	t_{SRW} t_{SRR}	2个			µs	从第一个数据字节的最后一位的SCLK上升沿到下一个命令的地址字节的第一位的SCLK下降沿
SPI读取地址数据延迟	t_{SRAD}	2个			µs	从地址字节最后一位的SCLK上升沿到第一个SCLK下降沿 正在读取的数据位
运动爆发后NCS不活动	t_{BEXIT}	500			ns	下一次SPI使用前运动突发后的最小NCS非活动时间
NCS到SCLK有效	$t_{NCS-SCLK}$	120			ns	从最后一个NCS下降沿到第一个SCLK上升沿

SCLK到NCS无效（用于读操作）	$t_{SCLK-NCS}$	120			ns	从最后一个SCLK上升沿到NCS上升沿，用于有效的MISO数据传输

参数	象征	最小值	典型的	最大限度	单元	笔记
SCLK到NCS无效（用于写操作）	$t_{SCLK-NCS}$	1			µs	从最后一个SCLK上升沿到NCS上升沿，对于有效的MOSI数据转接
NCS到MISO高阻态	$t_{NCS-MISO}$			500	ns	从NCS上升沿到MISO高阻态
运动上升时间	$t_{r-MOTION}$		300		ns	C_L = 20pF
运动下降时间	$t_{f-MOTION}$		300		ns	C_L = 20pF
输入电容	锡		10		pF	SCLK, MOSI, NCS
负载电容	C_L			20	pF	味噌、MOTION
瞬态电源电流	I_{DDT}			70	mA	从0V到V_{DD}的电源斜坡期间的最大电源电流，上升时间最短为150 µs，最长为20ms。（不包括充电电流对于旁路电容）
	I_{DDTIO}			60	mA	从0V到V_{DDIO}的电源斜坡期间的最大电源电流，上升时间最短为150 µs，最长为20ms。（不包括充电电流对于旁路电容）

2.5 直流电气规格

表5.直流电气规范

超过推荐工作条件的芯片电气特性。典型值是在25℃、VDD = 1.9V、VDDIO = 1.9V且LED电流为50mA时。

参数	象征	最小值	类型。	最大限度	单元	笔记
直流电源电流（高性能模式）	IDD_{RUN} IDD_{REST1} IDD_{REST2} IDD_{REST3}		1.7 580 11 6		mA µA µA µA	高达200ips ▪ IDD_{RUN}：平均电流消耗，包括1ms轮询的LED电流 ▪ IDD_{REST}：平均电流消耗，包括LED电流
直流电源电流(低功耗模式)	IDD_{RUN} IDD_{REST1} IDD_{REST2} IDD_{REST3}		1.3 580 11 6		mA µA µA µA	高达200ips ▪ IDD_{RUN}：平均电流消耗，包括1ms轮询的LED电流 ▪ IDD_{REST}：平均电流消耗，包括LED电流
直流电源电流(有线游戏模式)	IDD_{RUN}		10		mA	高达650ips IDD_{RUN} :平均电流消费，包括LED电流 轮询时间为0.125毫秒
直流电源电流（办公模式）	IDD_{RUN} IDD_{RUN} IDD_{REST1} IDD_{REST2} IDD_{REST3}		0.6 0.4 70 11 6		mA mA µA µA µA	高达200ips高达30ips ▪ IDD_{RUN} :平均电流消耗，包括8ms轮询的LED电流 ▪ IDD_{REST}：平均电流消耗，包括LED电流
关断电流	IPD		4		µA	
输入低电压	V_{IL}			0.3xVDDIO	V	SCLK, MOSI, NCS
输入高电压	V_{IH}	0.7xVDDIO			V	SCLK, MOSI, NCS
输入迟滞	V_{I_HYS}		100		mV	SCLK, MOSI, NCS
输入漏电流	I_{leak}		±1	±10	µA	V_{in}=VDDIO或0V, SCLK, MOSI, NCS
输出低电压	V_{OL}			0.45	V	I_{out} = 1mA味噌 I_{out} = 0.1mA用于运动
输出高电压	V_{OH}	VDDIO -0.45			V	I_{out} = -1mA味噌 I_{out} = -0.1mA用于运动

3.0 机械规格

本节涵盖芯片在芯片、镜头和PCB组件方面的指南和建议。

3.1 芯片封装尺寸

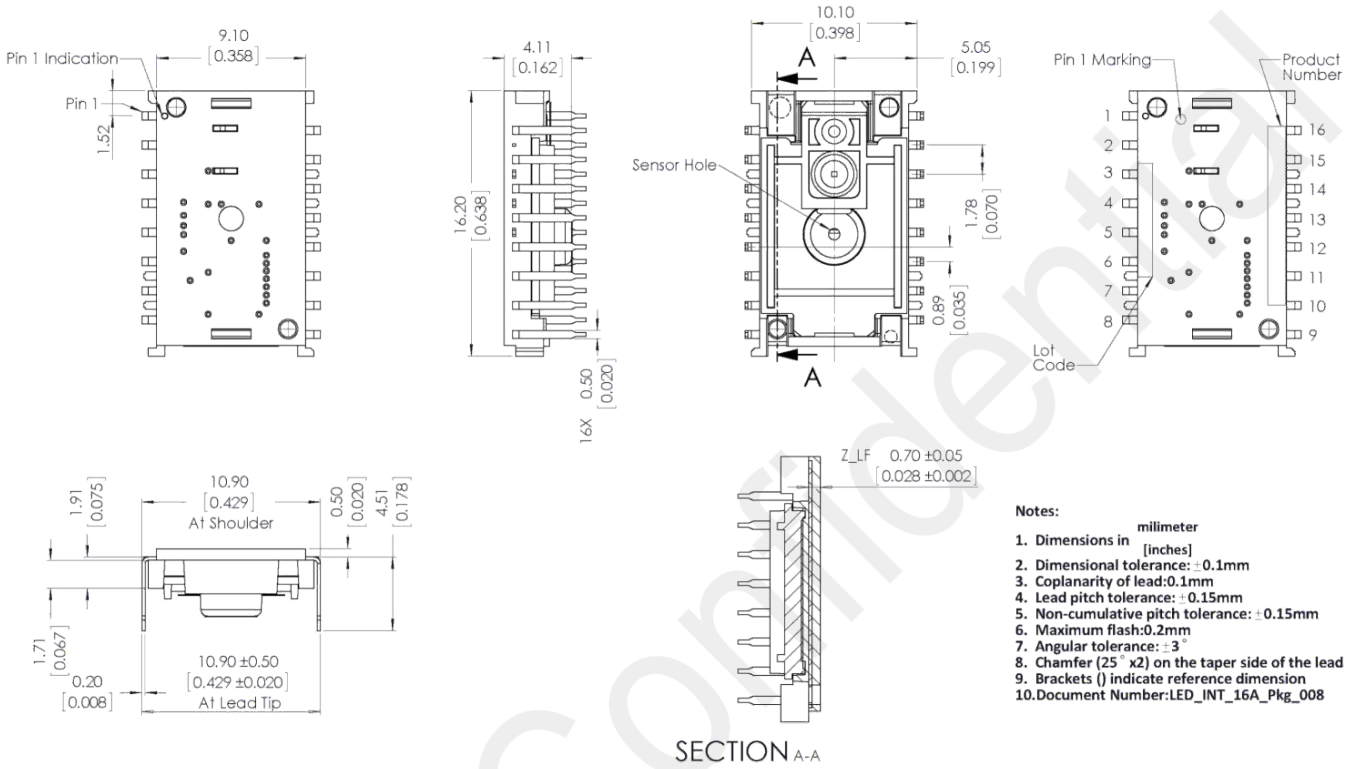


图3.封装外形图

注意：建议在处理和组装此组件时采取正常的静电放电预防措施，以防止ESD可能引起的损坏和/或退化。

3.2 封装标记

表6.封装标记说明

项目	打标	评论
产品编号	PAW3395DM-T6QU	
批号	AYWWXXXXX	A：装配厂房Y： 年份 WW：周 XXXXX：PixArt参考

3.3 芯片组装图

强烈建议遵循图4中的芯片方向以获得最佳跟踪性能。

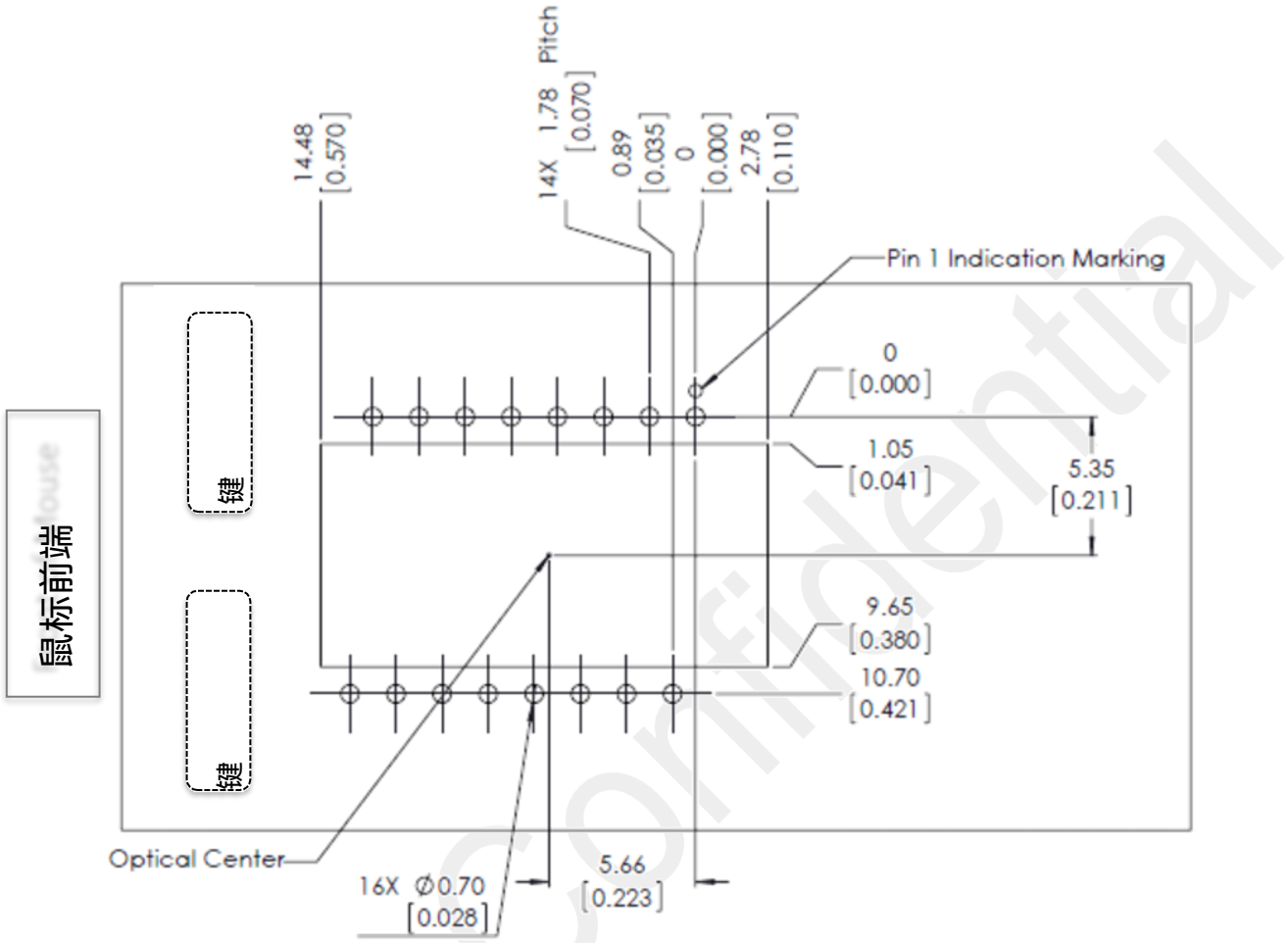


图4.推荐的芯片方向、机械切口和间距（顶视图）

3.4 镜头装配图

3.4.1 与LM19-LSI镜头组装

有关详细信息，请参阅LM19-LSI镜头数据表。

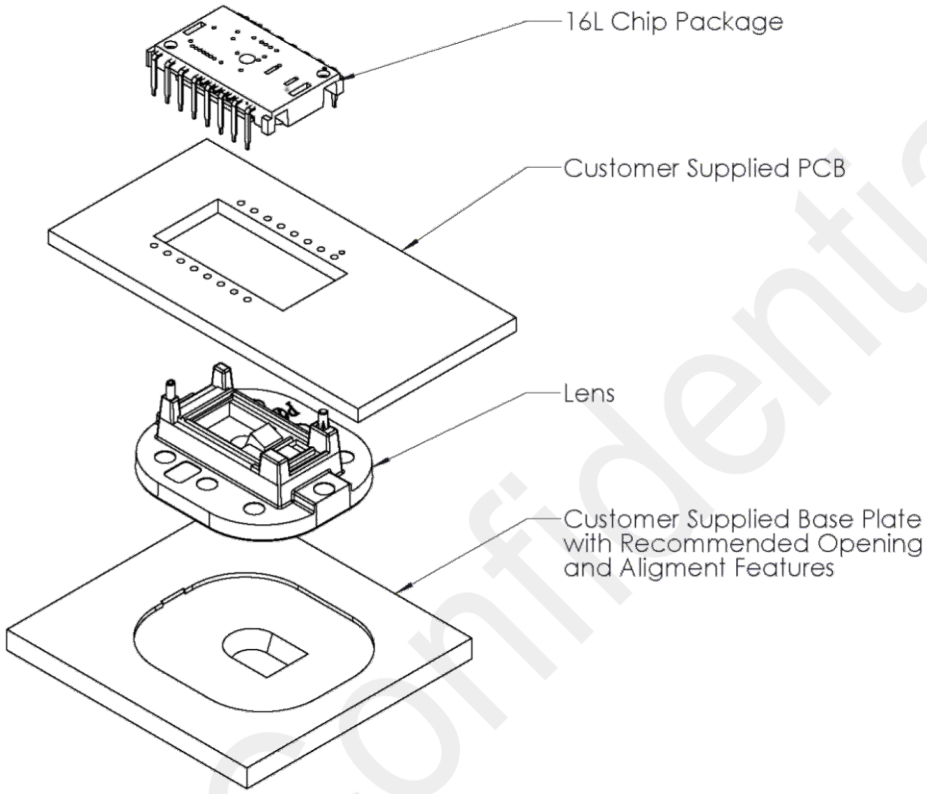


图5.装配LM19-LSI镜头的分解图

3.4.2 装配L0AE-LSI1镜头

有关详细信息，请参阅L0AE-LSI1镜头数据表。

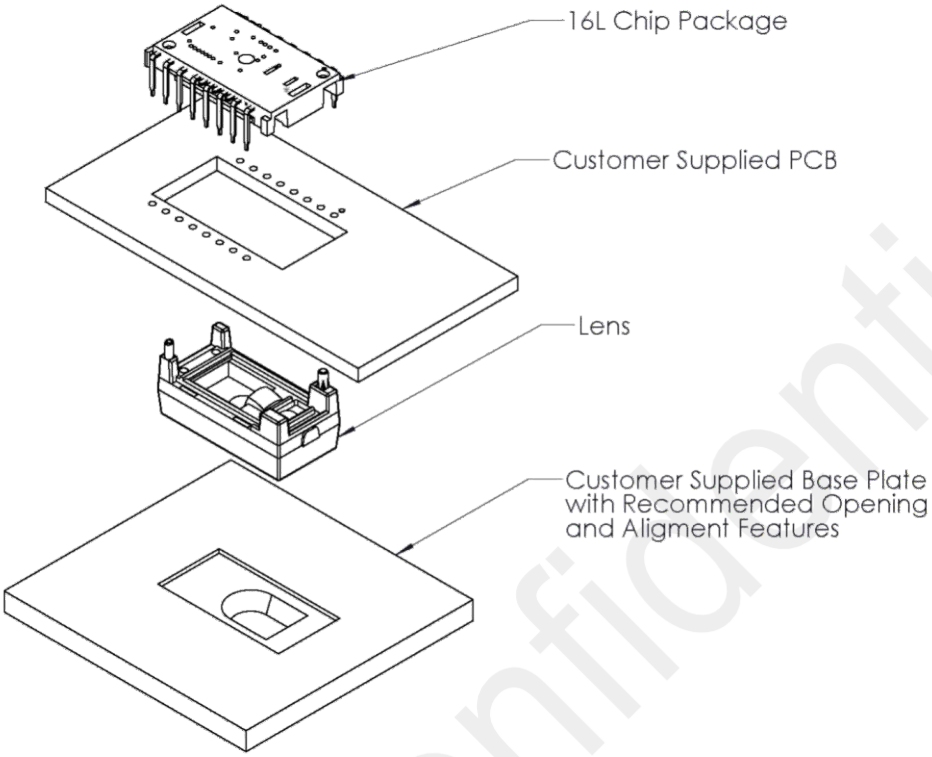


图6.装配L0AE-LSI1的分解图

3.5 PCB组装建议

1. 将集成芯片和所有其他电子元件插入PCB。
2. 使用焊接夹具在免洗焊接工艺中对整个组件进行波峰焊。需要焊接夹具来保护芯片免受助焊剂喷涂和波峰焊膏的影响。
3. 避免将任何助焊剂沾到芯片主体上，因为助焊剂有可能渗入芯片封装，焊接夹具的设计应仅将芯片引线暴露在助焊剂喷雾和熔融焊料中，同时屏蔽芯片主体和光学孔径。夹具还应将芯片设置在PCB上的正确位置和高度。
4. 将镜头放在底板上。必须小心避免光学表面受到污染。
5. 从芯片的光学孔径上取下保护性Kapton胶带。必须小心防止污染物进入孔。在整个鼠标组装过程中，请勿将PCB芯片朝上放置。取下Kapton胶带时垂直握住PCB。
6. 将镜头上的PCB组件插入底板对准柱以固定PCB组件。芯片封装将通过导柱自动对准镜头。PCB的光学位置参考由底板和透镜设置。请注意，必须尽量减少因按下按钮而引起的PCB运动，以保持光学对准。
7. 建议：通过热熔工艺将透镜的导柱熔化在芯片上，可以将透镜永久固定在芯片封装上。请参阅标题为“LM19-LSI镜头：PCB组装和镜头热熔建议”的应用说明，了解有关镜头热熔工艺的详细信息和建议。
8. 安装鼠标顶壳。顶壳中必须有一个功能可以向下压到PCB组件上，以确保所有组件堆叠或互锁到正确的垂直高度。
9. 建议将鼠标脚放在底板开口周围，以稳定鼠标在表面上的跟踪。

3.6 包装信息

物品	描述
产品编号	PAW3395DM-T6QU
封装类型	16L浸渍
每管数量	25 pcs
内盒数量	1,000 pcs
运输箱数量	12,000 pcs
管径	500 x 13.5 x 7.0 mm^3
内盒尺寸	89 x 540 x 58 mm^3
运输箱尺寸	310 x 560 x 270 mm^3

3.6.1 包装管

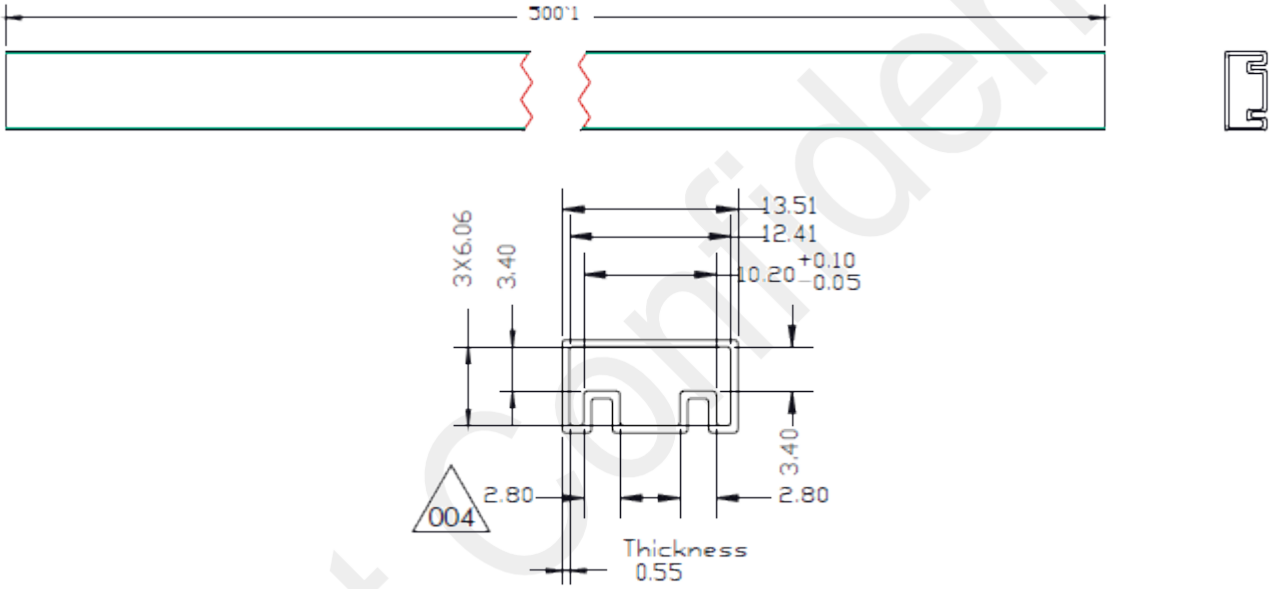
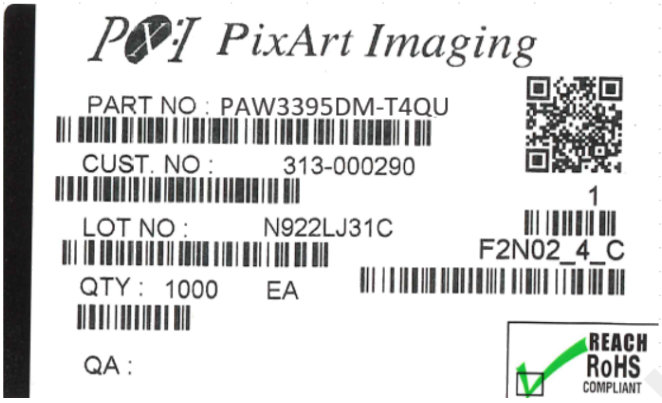


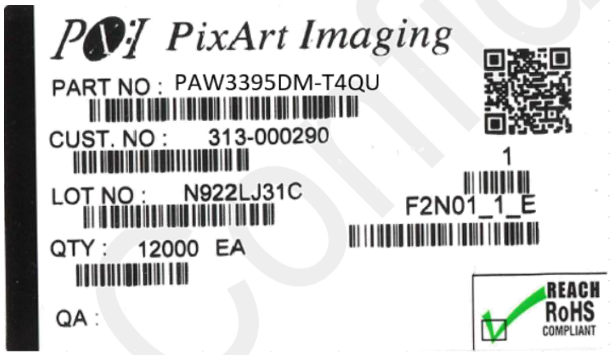
图7.填料管

3.7 封装处理信息

3.7.1 内箱标签样本



3.7.2 装运箱标签样本



4.0 参考示意图

建议不要让NRESET引脚悬空，它应该由微控制器的输出引脚持续驱动以建立其状态。

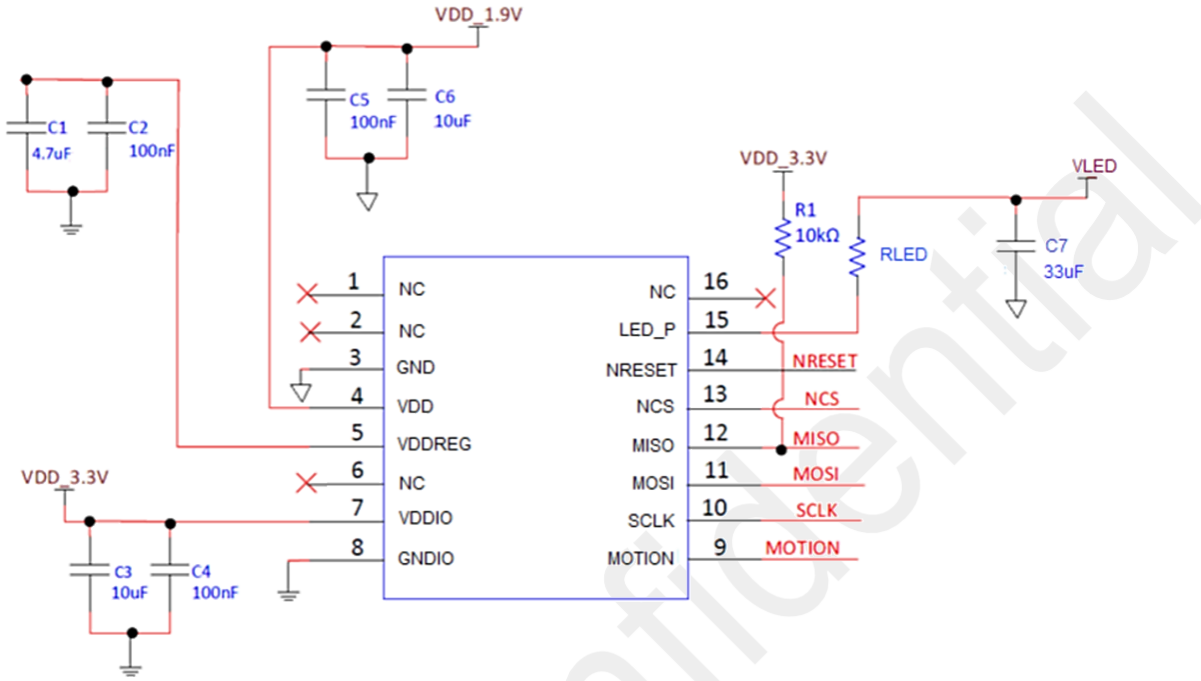


图8.参考示意图

表3显示了为LED获得50mA电流的R_{LED}和V_{LED}的推荐值。推荐使用R_{LED} 1%公差。

表7.推荐的R_{LED}

VLED (V)	推荐RLED (Ω)
1.9V	5.6
2.0V	6.8

5.0 串行外设接口(SPI)

5.1 信号说明

同步串口用于读写芯片中的寄存器。

该端口是一个4线端口。主机微控制器始终启动通信。该芯片从不启动任何数据传输。SCLK、MOSI和NCS可以直接由微控制器驱动。端口引脚可以与其他SPI从设备共享。当NCS引脚被驱动为高电平时，输入信号被忽略并且输出为三态。

表8. SPI端口信号说明

信号名称	功能说明
SCLK	时钟输入，由主机（微控制器）生成。
MOSI	输入数据。（主输出/从输入）
MISO	输出数据。（主输入/从输出）
NCS	片选输入（低电平有效）。NCS需要拉低激活串口；否则，MISO将为高阻态，MOSI和SCLK将被忽略。NCS也可用于在出现错误时重置串行端口。

5.2 运动引脚时序

运动引脚是低电平有效输出，在运动发生时向微控制器发出信号。每当设置运动位时，运动引脚就会降低；换言之，每当Delta_X_L、Delta_X_H、Delta_Y_L或Delta_Y_H寄存器中存在非零数据时。清除运动位（通过读取Delta_X_L、Delta_X_H、Delta_Y_L或Delta_Y_H寄存器）会将运动引脚置为高电平。

5.3 片选操作

NCS变低后，串口被激活。如果在交易过程中出现NCS，则整个交易将中止并且串行端口将被重置。事务中止后，在开始下一个事务之前需要正常的地址到数据或事务到事务延迟。为了提高通信可靠性，所有串行事务都应由NCS成帧。换句话说，端口在不使用期间不应保持启用状态，因为任何ESD和EFT/B事件都可能被解释为串行通信并使芯片进入未知状态。此外，每个突发模式事务完成后或终止突发模式操作后，都必须提高NCS。在突发模式终止之前，该端口不可用于进一步使用。

5.4 写入操作

写操作，定义为从微控制器到芯片的数据，总是由微控制器发起，由两个字节组成。第一个字节包含地址（七位），并以“1”作为其MSB以指示数据方向。第二个字节包含数据。芯片在SCLK的上升沿读取MOSI。

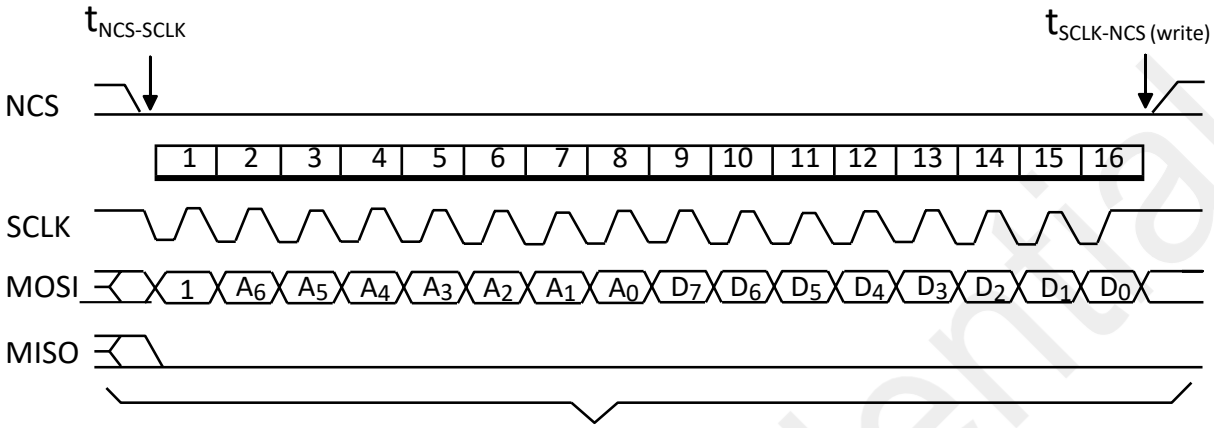


图9.写操作

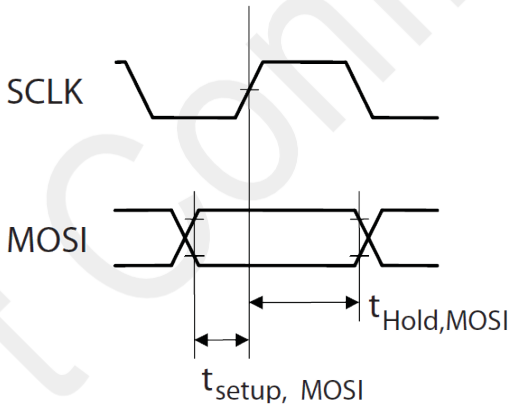


图10. MOSI建立和保持时间

5.5 读取操作

读取操作，定义为从芯片到微控制器的数据，总是由微控制器启动，由两个字节组成。第一个字节包含地址，由微控制器通过MOSI发送，其MSB为“0”以指示数据方向。第二个字节包含数据并由芯片通过MISO驱动。芯片在SCLK的下降沿输出MISO位，并在SCLK的每个上升沿采样MOSI位。

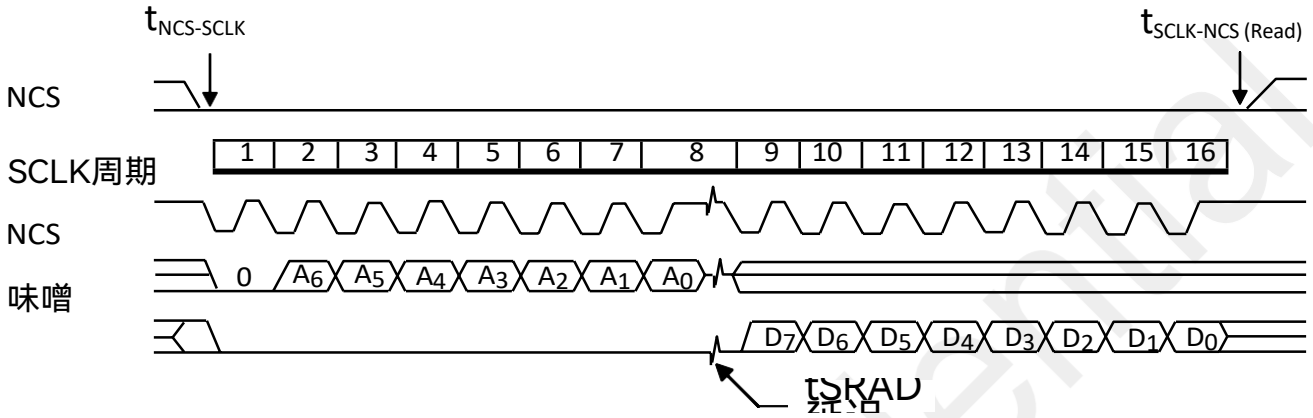


图11.读取操作

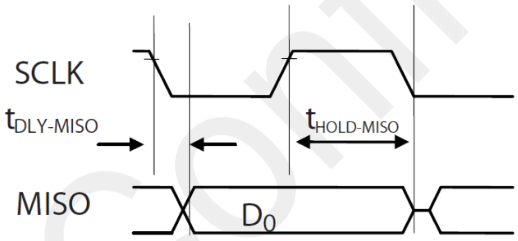


图12. MISO延迟和保持时间

注意：SCLK的最小高电平状态也是芯片的最小MISO数据保持时间。由于SCLK的下降沿实际上是下一个读或写命令的开始，所以芯片会保持MISO上数据的状态，直到SCLK的下降沿。

5.6 读取和写入命令之间所需的时序(tsxx)

串行端口上的读取和写入命令之间有最低时序要求。

如果第二个写命令的最后一个数据位的SCLK上升沿出现在t SWW延迟之前，那么第一个写命令可能无法正确完成。

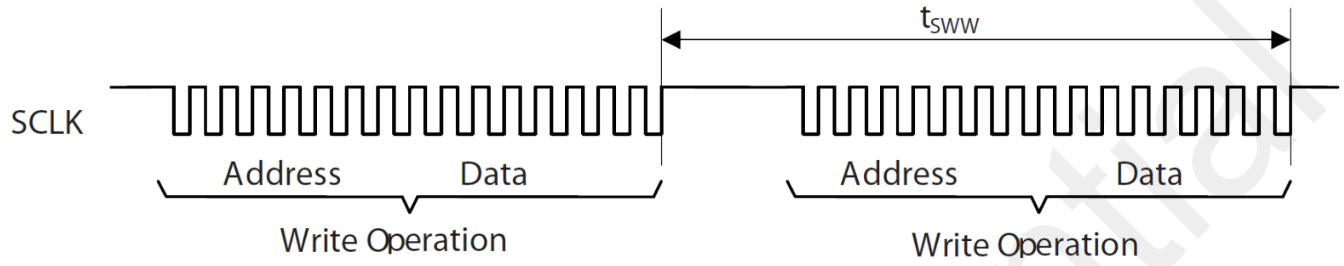


图13.两个写命令之间的时序

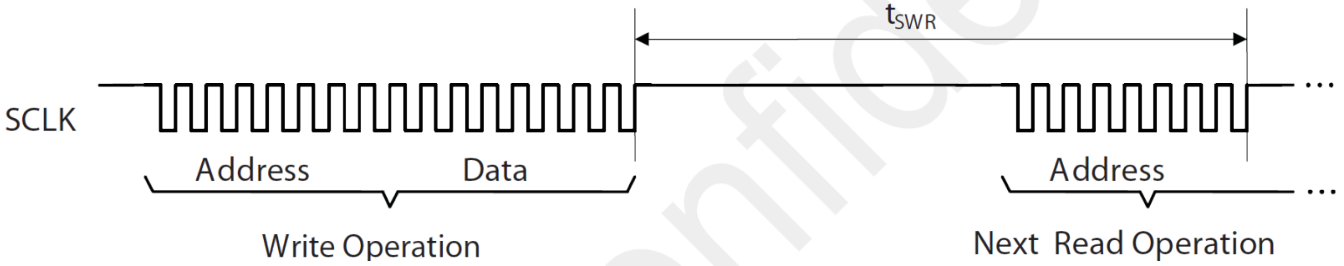


图14.写入命令与写入命令或后续读取命令之间的时序

如果读取命令的最后一个地址位的SCLK上升沿发生在t SWR所需的延迟之前，则写入命令可能无法正确完成。在读取操作期间，SCLK应在最后一个地址数据位之后至少延迟tSRAD，以确保芯片有时间准备请求的数据。

读取或写入命令的第一个地址位的SCLK下降沿必须在前一个读取操作的最后一个数据位的最后一个SCLK上升沿之后至少tSRR或tSRW。此外，在读取操作期间，SCLK应在最后一个地址数据位之后延迟，以确保芯片有时间准备请求的数据。

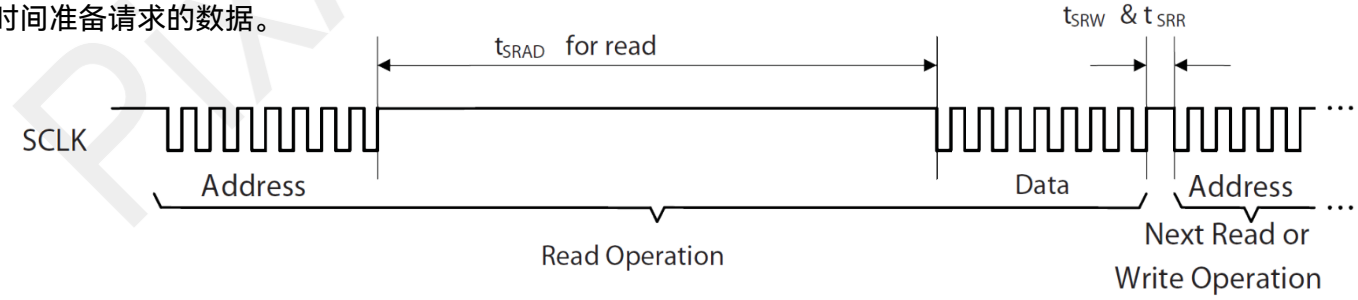


图15.读取命令与写入命令或后续读取命令之间的时序

5.7 突发模式操作

突发模式是一种特殊的串行端口操作模式，用于减少预定义寄存器的串行事务时间。速度的提高是通过连续数据时钟进出多个寄存器而无需指定寄存器地址并且不需要数据字节之间的正常延迟周期来实现的。

5.7.1 动作阅读

读取Motion_Burst寄存器会激活运动读取模式。芯片将按此顺序响应以下运动突发报告。

BYTE[00] =运动

BYTE[01] =观察

BYTE[02] =

Delta_X_L BYTE[03]

= Delta_X_H

BYTE[04] =

Delta_Y_L BYTE[05]

= Delta_Y_H

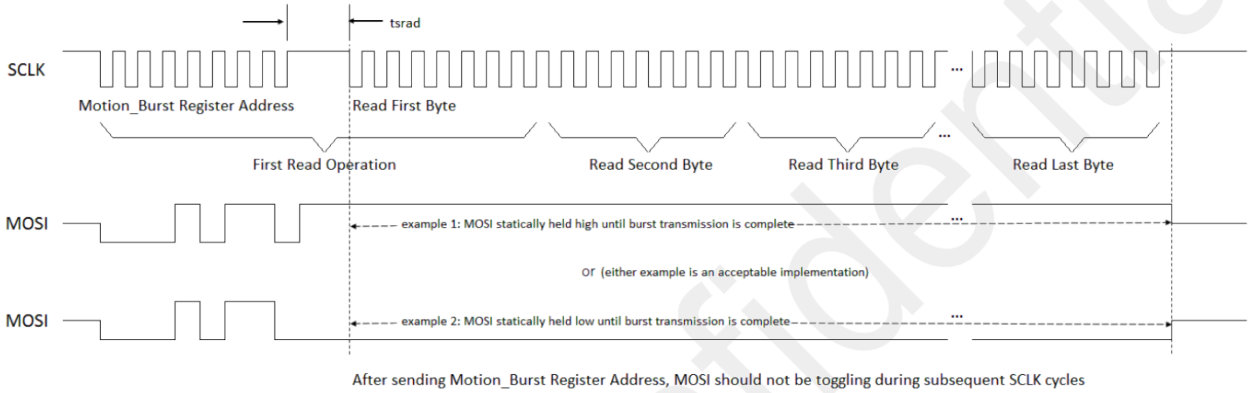
BYTE[06] = SQUAL

BYTE[07] = RawData_Sum
BYTE[08] = Maximum_RawData
BYTE[09] = Minimum_Rawdata
BYTE[10] = Shutter_Upper
BYTE[11] = Shutter_Lower

发送Motion_Burst寄存器地址后，微控制器必须等待t_{SRAD}，然后开始读取数据。通过以正常速率驱动SCLK，可以在字节之间无延迟地读取所有数据位。在接收到最后一个地址位后，数据被锁存到输出缓冲器中。突发传输完成后，微控制器必须拉高NCS线至少t_{BEXIT}以终止突发模式。串行端口在使用NCS重置之前不可用，即使是第二次突发传输。

5.7.2 启动Motion Burst的程序

1. 降低NCS。
2. 等待$t_{NCS-SCLK}$
3. 发送Motion_Burst地址(0x16)。发送此地址后，MOSI应保持静态（高电平或低电平），直到突发传输完成。
4. 等待t_{SRAD}
5. 开始连续读取最多12个字节的SPI数据。可以通过将NCS拉高持续at来终止运动突发至少t_{BEXIT}。
6. 要读取新的运动突发数据，请从步骤1开始重复。



注意：即使在运行或静止模式下，也可以从Burst_Motion_Read寄存器读取运动突发数据。充电序列

图14 运动读取序列

6.0 上电顺序

6.1 开机顺序

虽然芯片执行内部上电自复位，但仍建议将Power_Up_Reset
每次上电时都会写入寄存器。推荐的芯片上电顺序如下:

1. 以任何顺序为VDD和VDDIO供电，每次供电之间的延迟不超过100ms。确保所有供应稳定。
2. 等待至少50毫秒。
3. 将NCS拉高，然后拉低以重置SPI端口。
4. 将0x5A写入Power_Up_Reset寄存器（或者切换NRESET引脚）。
5. 等待至少5ms。
6. 加载上电初始化寄存器设置。
7. 无论运动位状态如何，都读取寄存器0x02、0x03、0x04、0x05和0x06一次。

6.2 上电初始化寄存器设置

1. 将值0x07写入寄存器0x7F
2. 将值0x41写入寄存器0x40
3. 将值0x00写入寄存器0x7F
4. 将值0x80写入寄存器0x40
5. 将值0x0E写入寄存器0x7F
6. 将值0x0D写入寄存器0x55
7. 将值0x1B写入寄存器0x56
8. 将值0xE8写入寄存器0x57
9. 将值0xD5写入寄存器0x58
10. 将值0x14写入寄存器0x7F
11. 将值0xBC写入寄存器0x42
12. 将值0x74写入寄存器0x43
13. 将值0x20写入寄存器0x4B
14. 将值0x00写入寄存器0x4D
15. 将值0x0E写入寄存器0x53
16. 将值0x05写入寄存器0x7F
17. 将值0x04写入寄存器0x44
18. 将值0x06写入寄存器0x4D
19. 将值0x40写入寄存器0x51
20. 将值0x40写入寄存器0x53
21. 将值0xCA写入寄存器0x55
22. 将值0xE8写入寄存器0x5A
23. 将值0xEA写入寄存器0x5B
24. 将值0x31写入寄存器0x61
25. 将值0x64写入寄存器0x62
26. 将值0xB8写入寄存器0x6D
27. 将值0x0F写入寄存器0x6E

28. 将值0x02写入寄存器0x70
29. 将值0x2A写入寄存器0x4A
30. 将值0x26写入寄存器0x60
31. 将值0x06写入寄存器0x7F
32. 将值0x70写入寄存器0x6D
33. 将值0x60写入寄存器0x6E
34. 将值0x04写入寄存器0x6F
35. 将值0x02写入寄存器0x53
36. 将值0x11写入寄存器0x55
37. 将值0x01写入寄存器0x7A
38. 将值0x51写入寄存器0x7D
39. 将值0x07写入寄存器0x7F
40. 将值0x10写入寄存器0x41
41. 将值0x32写入寄存器0x42
42. 将值0x00写入寄存器0x43
43. 将值0x08写入寄存器0x7F
44. 将值0x4F写入寄存器0x71
45. 将值0x09写入寄存器0x7F
46. 将值0x1F写入寄存器0x62
47. 将值0x1F写入寄存器0x63
48. 将值0x03写入寄存器0x65
49. 将值0x03写入寄存器0x66
50. 将值0x1F写入寄存器0x67
51. 将值0x1F写入寄存器0x68
52. 将值0x03写入寄存器0x69
53. 将值0x03写入寄存器0x6A
54. 将值0x1F写入寄存器0x6C

55. 将值0x1F写入寄存器0x6D
56. 将值0x04写入寄存器0x51
57. 将值0x20写入寄存器0x53
58. 将值0x20写入寄存器0x54
59. 将值0x0C写入寄存器0x71
60. 将值0x07写入寄存器0x72
61. 将值0x07写入寄存器0x73
62. 将值0x0A写入寄存器0x7F
63. 将值0x14写入寄存器0x4A
64. 将值0x14写入寄存器0x4C
65. 将值0x19写入寄存器0x55
66. 将值0x14写入寄存器0x7F
67. 将值0x30写入寄存器0x4B
68. 将值0x03写入寄存器0x4C
69. 将值0x0B写入寄存器0x61
70. 将值0x0A写入寄存器0x62
71. 将值0x02写入寄存器0x63
72. 将值0x15写入寄存器0x7F
73. 将值0x02写入寄存器0x4C
74. 将值0x02写入寄存器0x56
75. 将值0x91写入寄存器0x41
76. 将值0x0A写入寄存器0x4D
77. 将值0x0C写入寄存器0x7F
78. 将值0x10写入寄存器0x4A
79. 将值0x0C写入寄存器0x4B
80. 将值0x40写入寄存器0x4C
81. 将值0x25写入寄存器0x41
82. 将值0x18写入寄存器0x55
83. 将值0x14写入寄存器0x56
84. 将值0x0A写入寄存器0x49
85. 将值0x00写入寄存器0x42
86. 将值0x2D写入寄存器0x43
87. 将值0x0C写入寄存器0x44
88. 将值0x1A写入寄存器0x54
89. 将值0x0D写入寄存器0x5A
90. 将值0x1E写入寄存器0x5F
91. 将值0x05写入寄存器0x5B
92. 将值0x0F写入寄存器0x5E
97. 将值0x00写入寄存器0x51
98. 将值0x5B写入寄存器0x54
99. 将值0x00写入寄存器0x53
100. 将值0x64写入寄存器0x56
101. 将值0x00写入寄存器0x55
102. 将值0xA5写入寄存器0x58
103. 将值0x02写入寄存器0x57
104. 将值0x29写入寄存器0x5A
105. 将值0x47写入寄存器0x5B
106. 将值0x81写入寄存器0x5C
107. 将值0x40写入寄存器0x5D
108. 将值0xDC写入寄存器0x71
109. 将值0x07写入寄存器0x70
110. 将值0x00写入寄存器0x73
111. 将值0x08写入寄存器0x72
112. 将值0xDC写入寄存器0x75
113. 将值0x07写入寄存器0x74
114. 将值0x00写入寄存器0x77
115. 将值0x08写入寄存器0x76
116. 将值0x10写入寄存器0x7F
117. 将值0xD0写入寄存器0x4C
118. 将值0x00写入寄存器0x7F
119. 将值0x63写入寄存器0x4F
120. 将值0x00写入寄存器0x4E
121. 将值0x63写入寄存器0x52
122. 将值0x00写入寄存器0x51
123. 将值0x54写入寄存器0x54
124. 将值0x10写入寄存器0x5A
125. 将值0x4F写入寄存器0x77
126. 将值0x01写入寄存器0x47
127. 将值0x40写入寄存器0x5B
128. 将值0x60写入寄存器0x64
129. 将值0x06写入寄存器0x65
130. 将值0x13写入寄存器0x66
131. 将值0x0F写入寄存器0x67
132. 将值0x01写入寄存器0x78
133. 将值0x9C写入寄存器0x79
134. 将值0x00写入寄存器0x40

93.将值0x0D写入寄存器0x7F

94.写寄存器0x48带值0xdd

95.写寄存器0x4f用值0x03

96.写寄存器0x52带值0x49

135. 用值0x02写寄存器0x55

136. 用值0x70写寄存器0x23

137. 写寄存器0x22带值0x01

138. 等待1ms

139. 以1ms的间隔读取寄存器0x6C，直到获得值0x80或最多读取60次，此寄存器读取间隔必须以1ms的间隔进行，时间容差为±1%

如果60次后仍未从寄存器0x6C中获取到0x80的值:

a. 将值0x14写入寄存器0x7F

b. 将值0x00写入寄存器0x6C

c. 将值0x00写入寄存器0x7F

140. 将值0x00写入寄存器0x22

141. 将值0x00写入寄存器0x55

142. 将值0x07写入寄存器0x7F

143. 将值0x40写入寄存器0x40

144. 将值0x00写入寄存器0x7F

在上电过程中，电源处于高电平之后但正常运行之前会有一段时间。下表显示了上电和复位期间各种引脚的状态。

表9. VDD有效后信号引脚的状态

引脚	重置期间	重置后
NRESET	功能性	功能性
NCS	忽略	功能性
MISO	不明确的	取决于NCS
SCLK	忽略	取决于NCS
MOSI	忽略	取决于NCS
运动	不明确的	功能性

6.3 NRESET

NRESET引脚用于执行芯片全芯片复位。置位时，它执行与Power_Up_Reset_Register相同的复位功能。NRESET引脚需要置位（保持为逻辑0）至少100 ns的持续时间，以便芯片复位。

注：NRESET管脚内置弱上拉电路。在低电平有效复位阶段，NRESET引脚可吸收高达600μA的静态电流。

7.0 操作指南

7.1 原始数据输出

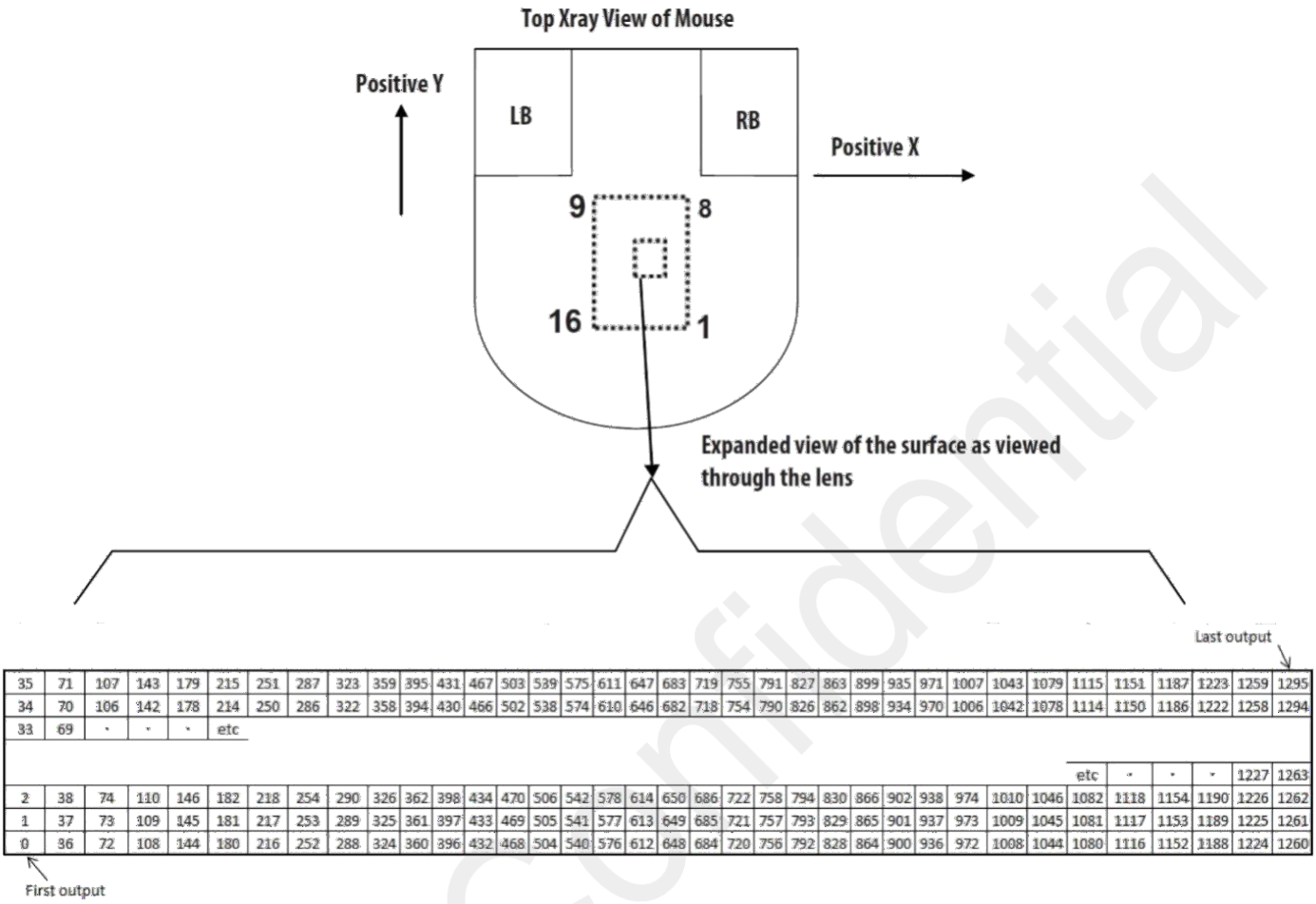
本节介绍下载完整RawData值数组的方法。

为了触发RawData输出，写入RawData_Grab寄存器。在RAWDATA_GRAB_STATUS寄存器报告PG_VALID为TRUE后，通过使用寄存器读取方法读取RAWDATA_GRAB寄存器来检索RawData的第1个元素。在RawData输出过程中，必须将鼠标置于静止位置。

RawData输出程序:

1. 芯片应该正确上电和复位。
2. 将值0x00写入寄存器0x7F
3. 将值0x80写入寄存器0x40
4. 连续读取寄存器0x02 (Motion)，直到OP_Mode1和OP_Mode0都为0。
5. 将值0x01写入寄存器0x50
6. 将值0x04写入寄存器0x55
7. 将值0xFF写入寄存器0x58
8. 不断读取寄存器0x59，直到PG_FIRST和PG_VALID都为“1”
9. 从寄存器0x58读取第一个原始数据
10. 不断读取寄存器0x59，直到PG_VALID为“1”。
11. 读取7位ADC数据(RAWDATA 6-0)的寄存器0x58。重复（10）和（11）1295次，形成完整的像素阵列信息。
12. 将值0x00写入寄存器0x40
13. 将值0x00写入寄存器0x50
14. 将值0x00写入寄存器0x55

原始数据地图



35	71	107	143	179	215	251	287	323	359	395	431	467	503	539	575	611	647	683	719	755	791	827	863	899	935	971	1007	1043	1079	1115	1151	1187	1223	1259	1295
34	70	106	142	178	214	250	286	322	358	394	430	466	502	538	574	610	646	682	718	754	790	826	862	898	934	970	1006	1042	1078	1114	1150	1186	1222	1258	1294
33	69	·	·	·	etc																														
																														etc	·	·	·	1227	1263
2	38	74	110	146	182	218	254	290	326	362	398	434	470	506	542	578	614	650	686	722	758	794	830	866	902	938	974	1010	1046	1082	1118	1154	1190	1226	1262
1	37	73	109	145	181	217	253	289	325	361	397	433	469	505	541	577	613	649	685	721	757	793	829	865	901	937	973	1009	1045	1081	1117	1153	1189	1225	1261
0	36	72	108	144	180	216	252	288	324	360	396	432	468	504	540	576	612	648	684	720	756	792	828	864	900	936	972	1008	1044	1080	1116	1152	1188	1224	1260

图17. RawData映射（参考表面）

7.2 关闭

可以通过将值0xB6写入关断寄存器0x3B来将芯片设置为关断模式。除上电命令（将0x5A写入寄存器0x3A）外，在断言关断模式时不应访问SPI端口。只要芯片的NCS引脚未被激活，就可以访问同一SPI总线上的其他IC。

要解除断言关断模式，请从第2步开始执行上电序列。

表10.关断模式下的引脚状态

引脚	地位
NRESET	高的
NCS	High*1
MISO	Hi-Z*2
SCLK	如果NCS = 1*3，则忽略
MOSI	如果NCS = 1*4，则忽略

备注：*

1. 如果SPI总线与其他设备共享，则NCS引脚必须保持为1（高电平）。建议在关机期间保持为1（高），除非给芯片上电。如果芯片要从关机状态重新上电（将0x5A写入寄存器0x3a），它必须保持为0（低）。
2. 为了满足数据表中的低功耗规范，应在关机期间上拉MISO。
3. 如果NCS为1（高），则忽略SCLK。如果NCS为0（低），则它起作用。
4. 如果NCS为1（高），则忽略MOSI。如果NCS为0（低），MOSI引脚上出现的任何命令都将被忽略，除了上电命令（将0x5A写入寄存器0x3A）之外。

注意：关机后的唤醒时间很长。在鼠标正常移动期间，不应将关机用于电源管理。

7.3 游戏和办公模式设置

根据下表中的寄存器设置，可以将芯片编程为不同的游戏和办公模式。请注意，根据推荐的上电顺序启动芯片时，芯片默认设置为高性能模式。

高性能模式（默认）	低功耗模式	办公模式
将值0x05写入寄存器0x7F将值0x40写入寄存器0x51将值0x40写入寄存器0x53将值0x31写入寄存器0x61将值0x0F写入寄存器0x6E将值0x0F写入寄存器0x7F将值0x32写入寄存器0x42将值0x00写入寄存器0x43写入寄存器0x7F用值0x0D写寄存器0x51用值0x00写寄存器0x52用值0x49写寄存器0x53用值0x00写寄存器0x54用值0x5B写寄存器0x55用值0x00写寄存器0x56用值0x64写寄存器0x57用值0x02写寄存器0x58用值0xA5写入寄存器0x7F，值为0x00写入寄存器0x54，值为0x54写入寄存器0x78，值为0x01写入寄存器0x79，值为0x9C写入寄存器0x40位[1:0]与值0x0	将值0x05写入寄存器0x7F将值0x40写入寄存器0x51将值0x40写入寄存器0x53将值0x3B写入寄存器0x61将值0x1F写入寄存器0x6E将值0x1F写入寄存器0x7F将值0x07写入寄存器0x42将值0x32写入寄存器0x43写入寄存器0x00写入寄存器0x7F用值0x0D写寄存器0x51用值0x00写寄存器0x52用值0x49写寄存器0x53用值0x00写寄存器0x54用值0x5B写寄存器0x55用值0x00写寄存器0x56用值0x64写寄存器0x57用值0x02写寄存器0x58用值0xA5写入寄存器0x7F，值为0x00写入寄存器0x54，值为0x54写入寄存器0x78，值为0x01写入寄存器0x79，值为0x9C写入寄存器0x40位[1:0]与	将值0x05写入寄存器0x7F将值0x28写入寄存器0x51将值0x30写入寄存器0x61将值0x3B写入寄存器0x61将值0x1F写入寄存器0x6E将值0x07写入寄存器0x7F将值0x32写入寄存器0x42将值0x00写入寄存器0x43写入寄存器0x7F用值0x0D写寄存器0x51用值0x00写寄存器0x52用值0x49写寄存器0x53用值0x00写寄存器0x54用值0x5B写寄存器0x55用值0x00写寄存器0x56用值0x64写寄存器0x57用值0x02写寄存器0x58用值0xA5写入寄存器0x7F，值0x00写入寄存器0x54，值0x52写入寄存器0x78，值0x0A写入寄存器0x79，值0x0F写入寄存器0x40位[1:0]与值0x02

笔记:

需要对寄存器0x40采取特殊预防措施，以避免覆盖寄存器中的其他位。写入bit[1:0]配置不同模式时，需要先读取并存储其当前值，然后应用位掩码将新值写回寄存器。详见8.3节。

有线游戏模式
将值0x05 写入寄存器0x7F 将值0x40写入寄存器0x51将值0x40写入寄存器0x53将值0x40写入寄存器0x61将值0x31写入寄存器0x6E将值0x0F写入寄存器0x7F将值0x07写入寄存器0x42将值0x2F写入寄存器0x43将值0x00写入寄存器0x7F用值0x0D写寄存器0x51用值0x12写寄存器0x52用值0xDB写寄存器0x53用值0x12写寄存器0x54用值0xDC写寄存器0x55用值0x12写寄存器0x56用值0xEA写寄存器0x57用值0x15写寄存器0x58用值0x2D将值0x00写入寄存器0x7F将值0x55写入寄存器0x54 将值0x83写入寄存器0x40

7.4 通用提升切断

芯片提供1mm和2mm通用lift cut off设置，该设置适用于所有垫子，lift cut off设置配置详见LIFT_CONFIG寄存器。在按照数据表中推荐的上电顺序启动时，芯片默认设置为1mm提升切断设置。

7.5 手动提升切断校准

该芯片能够通过调整特定游戏垫或跟踪表面上的参数来优化其提升性能，此功能涉及最终用户交互。

7.5.1 提升切断校准程序

1. 确保芯片按照第6.1节中的上电顺序上电。

2. 提示用户手动升降切断校准即将开始，并确保鼠标名义上放置在表面上（鼠标未抬起）。

3. 通过依次加载以下寄存器值来启动校准程序。
   a. 将值0x00写入寄存器0x7F
   b. 读取寄存器0x40并将其值存储到Var_Mode
   c. 将值0x80写入寄存器0x40
   d. 将值0x05写入寄存器0x7F
   e. 将值0xE7写入寄存器0x43
   f. 将值0x04写入寄存器0x7F
   g. 将值0xC0写入寄存器0x40
   h. 将值0x10写入寄存器0x41
   i. 将值0x0C写入寄存器0x44
   j. 将值0x0C写入寄存器0x45
   k. 将值0x0C写入寄存器0x46
   l. 将值0x0C写入寄存器0x47
   m. 将值0x0C写入寄存器0x48
   n. 将值0x0C写入寄存器0x49
   o. 将值0x0C写入寄存器0x4A
   p. 将值0x0C写入寄存器0x4B
   q. 将值0xC1写入寄存器0x40
4. 校准程序可以通过向用户发出的SW提示启动，也可以由用户通过鼠标单击事件启动。建议将鼠标移动超过20英寸的距离以覆盖垫子的大部分区域。
5. 将值0x40写入寄存器0x40以停止校准过程。
6. 连续读取寄存器0x4C位[3:0]以检查校准过程的状态。

如果返回值等于5，则表示校准成功。校准可以继续下一步或继续直到用户启动鼠标单击事件。

否则，校准失败，加载以下寄存器值返回通用1mm设置，校准过程需要从步骤2重新开始。

   a. 将值0x08写入寄存器0x4E
   b. 将值0x05写入寄存器0x7F
   c. 将值0xE4写入寄存器0x43
   d. 将值0x00写入寄存器0x7F
   e. 使用Var_Mode写入寄存器0x40

7. 如果校准成功，依次写入如下一组寄存器值，
   a. 读取寄存器0x4D并将其值存储到VarA
   b. 将值0x0C写入寄存器0x7F
   c. 将值0x0C存储到VarB
   d. 将值0x30存储到VarC
   e. 将值0x08写入寄存器0x4E
   f. 将值0x05写入寄存器0x7F
   g. 将值0xE4写入寄存器0x43
   h. 将值0x00写入寄存器0x7F
   i. 使用Var_Mode写入寄存器0x40
8. 执行固件辅助手动校准，请参阅第7.5.2节。（选修的）
9. 提示用户校准过程已完成，继续第7.5.3节启用Lift Cut off Calibration Register Setting。

7.5.1.1固件辅助手动提升切断校准

固件辅助手动提升切断校准提供了额外的步骤，以确保手动校准提供良好的跟踪，提升切断高度在独特的表面上约为1.2毫米。此部分是可选的，不一定需要成功完成第7.5.1节中的手动校准。它不会影响其他表面的提升切断性能。

程序:

1. 当鼠标在表面上移动时，以突发模式操作连续读取和累加SQUAL、RawData_Sum和SQUAL2寄存器的值。累计至少3000组样本后，计算每个寄存器的平均值，分别存入Var_SQUAL_Avg、Var_RawData_Sum_Avg、Var_SQUAL2_Avg。

   运动突发报告顺序:

   BYTE[00] =运动
   BYTE[01] =观察
   BYTE[02] =
   Delta_X_L BYTE[03]
   = Delta_X_H
   BYTE[04] =
   Delta_Y_L BYTE[05]
   = Delta_Y_H
   BYTE[06] = SQUAL
   BYTE[07] = RawData_Sum
   BYTE[08] =
   Maximum_RawData BYTE[09]
   = Minimum_Rawdata
   BYTE[10] = Shutter_Upper
   BYTE[11] = Shutter_Lower
   BYTE[12] =保留

BYTE[13] =保留

BYTE[14] = SQUAL2

笔记:

a. 建议为累加器分配32位无符号整数，以防止数据溢出。

b. 有关突发模式操作的详细信息，请参阅第5.8节。

2. 检查RawData_Sum_Avg:

   如果(RawData_Sum_Avg < 48)，将值23存储到Var_Squal_TH

   。否则，将值30存储到Var_Squal_TH

3. 计算SQUAL比率并存储到Var_SQUAL_Ratio:

   Var_SQUAL_Ratio = (*Var_SQUAL2_Avg x* 100)/*Var_SQUAL_Avg*

4. 检查Var_SQUAL_Ratio和RawData_Sum_Avg:

   如果((Var_SQUAL_ratio < Var_Squal_TH) && (RawData_Sum_Avg < 68)),

   检测到独特的表面，将值0x25存储到VarA，将值0x0C存储到VarB，将0x2D存储到VarC。否则，未检

   测到独特的表面，VarA、VarB和VarC没有变化。

7.5.2 启用提升切断校准寄存器设置

写入以下一组寄存器值以启用特定游戏垫上的提升切断校准寄存器设置。本节将使用从7.5.1节中获得的VarA、VarB和VarC。

1. 将值0x0C写入寄存器0x7F
2. 用VarA写寄存器0x41
3. 用VarC写寄存器0x43
4. 用VarB写寄存器0x44
5. 将值0x08写入寄存器0x4E
6. 将值0x0D写入寄存器0x5A
7. 将值0x05写入寄存器0x5B
8. 将值0x05写入寄存器0x7F
9. 将值0x0F写入寄存器0x6E
10. 将值0x09写入寄存器0x7F
11. 将值0x0C写入寄存器0x71
12. 将值0x00写入寄存器0x7F

7.5.3 禁用提升切断校准寄存器设置

写入以下一组寄存器值以禁用第7.5.2节中的提升切断校准寄存器设置，并恢复为默认的通用1毫米提升切断设置。

1. 将值0x0C写入寄存器0x7F
2. 将值0x25写入寄存器0x41
3. 将值0x2D写入寄存器0x43
4. 将值0x0C写入寄存器0x44
5. 将值0x10写入寄存器0x4A
6. 将值0x0C写入寄存器0x4B
7. 将值0x40写入寄存器0x4C
8. 将值0x08写入寄存器0x4E
9. 将值0x16写入寄存器0x53
10. 将值0x1A写入寄存器0x54
11. 将值0x18写入寄存器0x55

12. 将值0x14写入寄存器0x56
13. 将值0x0D写入寄存器0x5A
14. 将值0x05写入寄存器0x5B
15. 将值0x1E写入寄存器0x5F
16. 将值0x30写入寄存器0x66
17. 将值0x05写入寄存器0x7F
18. 将值0x0F写入寄存器0x6E
19. 将值0x09写入寄存器0x7F
20. 将值0x0C写入寄存器0x71
21. 将值0x07写入寄存器0x72
22. 将值0x00写入寄存器0x7F

7.6 无线模式的电源管理

该芯片具有三种省电模式。每种模式都有不同的运动检测周期及其对鼠标运动的相应响应时间。响应时间是检测到运动时芯片从静止模式唤醒所需的时间。当闲置时，芯片会自动从运行模式更改或降档到Rest1，然后到Rest2，最后到消耗最少电流的Rest3。

电流消耗在Rest3最低，在Rest1最高。然而，芯片响应运动所需的时间从Rest1最短，从Rest3最长。降档时间是从现有模式到下一个模式的经过时间（在无运动条件下）。例如，处于Rest1模式的芯片需要10s才能转换（降档）到Rest2模式。每种模式的响应时间和降档时间如下表所示。

但是，用户可以通过寄存器0x77到0x7C更改每个休息模式的时序设置。

表11.静止模式响应和降档时间

休息模式	HP模式/ LP模式		办公模式	
	响应时间	降档时间	响应时间	降档时间
Rest1	< 2ms	1s	< 20ms	1s
Rest2	< 200ms	10s	< 200ms	10s
Rest3	< 1000ms	600s	< 1000ms	600s

注：时序基于第6.1节中的上电初始化寄存器设置和第7.3节中的游戏和办公模式设置。如果更新寄存器0x77到0x7C中的任何值，则时序会发生变化。

8.0 寄存器

8.1 寄存器汇总表

芯片寄存器可通过串行端口访问。寄存器用于读取运动数据和状态以及设置设备配置。

地址	登记	使用权	重置值	地址	寄存器	使用权	重置值
0x00	Product_ID	R	0x51	0x4A	**Resolution_Y_Low**	R/W	0x63
0x01	Revision_ID	R	0x00	0x4B	**Resolution_Y_High**	R/W	0x00
0x02	Motion	R/W	0x00	0x56	Angle Snap(角度捕捉)	R/W	0x0D
0x03	Delta_X_L	R	0x00	0x58	RawData output	R	0x00
0x04	Delta_X_H	R	0x00	0x59	RawData status	R	0x00
0x05	Delta_Y_L	R	0x00	0x5A	Ripple_Control(纹波控制)	R/W	0x00
0x06	Delta_Y_H	R	0x00	0x5B	Axis_Control(坐标系翻转)	R/W	0x60
0x07	SQUAL	R	0x00	0x5C	Motion_Ctrl	R/W	0x02
0x08	RawData_Sum	R	0x00	0x5F	Inv_Product_ID	R	0xAE
0x09	Maximum_RawData	R	0x00	0x77	Run_Downshift	R/W	0x14
0x0A	Minimum_RawData	R	0x00	0x78	Rest1_Period	R/W	0x01
0x0B	Shutter_Lower	R	0x00	0x79	Rest1_Downshift	R/W	0x90
0x0C	Shutter_Upper	R	0x01	0x7A	Rest2_Period	R/W	0x19
0x15	Observation	R/W	0x80	0x7B	Rest2_Downshift	R/W	0x5E
0x16	Motion_Burst	R/W	0x00	0x7C	Rest3_Period	R/W	0x3F
0x3A	Power_Up_Reset	W	N/A	0x7D	Run_Downshift_Mult	R/W	0x07
0x3B	Shutdown	W	N/A	0x7E	Rest_Downshift_Mult	R/W	0x55
0x40	Performance	R/W	0x00	0x0577*	Angle_Tune1(角度调整)	R/W	0x00
0x47	**Set_Resolution**(更新CPI)	W	0x00	0x0578*	Angle_Tune2(角度调整使能)	R/W	0x00
0x48	**Resolution_x_low**	R/W	0x63	0x0C4E*	**Lift_Config**	R/W	0x08
0x49	**Resolution_x_high**	R/W	0x00				

笔记:

1. R =读，w =写，读/写= Rw
2. * -为了访问寄存器:
   a. 在地址中写下带有MSB（字节）值的寄存器0x7F。
   b. 读/编写寄存器值，并使用较低的字节地址。
   c. 用值0x00写寄存器0x7f。
3. 例如：将值0x01写入寄存器0x0C4E (Lift_Config)
   a. 将值0x0C写入寄存器0x7F，
   b. 将值0x01写入寄存器0x4E，
   c. 将值0x00写入寄存器0x7F

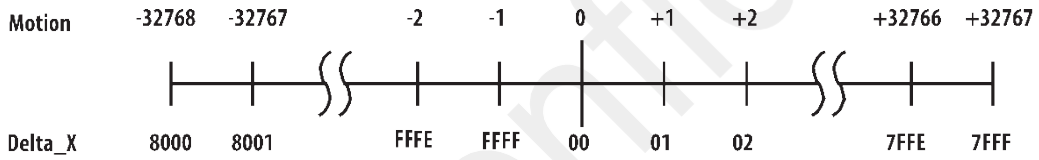
8.2 寄存器说明

注册名称	PRODUCT_ID							
银行	-			住址		0x00		
使用权	R			默认值		0x51		
位	7	6	5	4	3	2	1	0
域	PID [7:0]							
描述	该寄存器包含分配给芯片的唯一标识。该寄存器中的值不更改，它可用于验证串行通信链路是否正常工作。							

注册名称	REVISION_ID							
银行	-			住址		0x01		
使用权	R			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	反转[7:0]							
描述	该寄存器包含当前的IC版本。							

注册名称	运动							
银行	-			住址		0x02		
使用权	R			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	MOT	保留	保留	保留	Lift_Stat	保留	OP_Mode1	OP_Mode0
描述	该寄存器允许用户确定自上次读取后是否发生了运动。读取运动寄存器（Delta_X_L、Delta_X_H、Delta_Y_L和Delta_Y_H）的过程如下: 1. 读取运动寄存器。这将冻结Delta_X_L、Delta_X_H、Delta_Y_L和Delta_Y_H寄存器值。 2. 如果设置了MOT位，则应按给定顺序读取Delta_X_L、Delta_X_H、Delta_Y_L和Delta_Y_H寄存器以获得累积运动。 3. 要读取一组新的运动数据（Delta_X_L、Delta_X_H、Delta_Y_L和Delta_Y_H），请从步骤1开始重复。 向该寄存器写入任何值将清除所有运动数据。 注意：如果在第二次读取运动寄存器之前没有读取Delta_X_L、Delta_X_H、Delta_Y_L和Delta_Y_H寄存器，Delta_X_L、Delta_X_H、Delta_Y_L和Delta_Y_H中的数据将丢失。							

位域	名称	默认值	描述
7	MOT	0	自上次报告以来的动议 0：无动议 1：运动发生，数据准备好读取Delta_X_L，Delta_X_H、Delta_Y_L和Delta_Y_H寄存器
3	Lift_Stat	0	表示芯片的抬升状态 0: Chip on surface 1: 抬起
1:0	OP_Mode[1:0]	0	0：运行模式 1：休息1模式 2：休息2模式 3：休息3模式

注册名称	DELTA_X_L							
银行	-			住址		0x03		
使用权	R			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	X7	X6	X5	X4	X3	X2	X1	X0
注册名称								
银行	-			住址		0x04		
使用权	R			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	X15	X14	X13	X12	X11	X10	X9	X8
描述	上8位是delta_x_h，下部8位为delta_x_l。总体delta_x是 16位2的补充编号。 自上次报告以来，X运动是计数的。绝对值由分辨率确定。  首先需要读取Delta_x_l的Delta_x_l，以具有完整的运动数据。 注意：建议将寄存器0x02、0x03、0x04、0x05和0x06进行顺序读取。							
银行	-			住址		0x05		
使用权	R			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
银行	-			住址		0x06		
使用权	R			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	Y15	Y14	Y13	Y12	Y11	Y10	Y9	Y8

描述	上8位是delta_y_h，下部8位为delta_y_l。总体delta_y是 16位2的补充编号。 自上次报告以来，Y运动是计数的。绝对值由分辨率确定。 需要首先读取Delta_y_l，以使用Delta_y_h具有完整的运动数据。 注意：建议将寄存器0x02、0x03、0x04、0x05和0x06进行顺序读取。

注册名称	SQUAL(特征点寄存器)							
银行	-			住址		0x07		
使用权	R			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	SQ7	SQ6	SQ5	SQ4	SQ3	SQ2	SQ1	SQ0
描述	SQUAL（表面质量）寄存器用于衡量芯片在当前帧中可见的有效特征的数量。使用以下公式计算有效特征的总数。 功能数量= Squal寄存器值x 4 最大鼠尾形寄存器值为0xB6。由于当前框架的小变化会导致Squal的变化，因此预期在表面时，Squal的变化。 仅当芯片处于运行模式时，肮脏的值才有效。在测量Squal之前禁用休息模式。							

注册名称	RAWDATA_SUM							
银行	-			住址		0x08		
使用权	R			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	RDS7	RDS6	RDS5	RDS4	RDS3	RDS2	RDS1	RDS0
描述	该寄存器用于查找芯片平均RAWDATA值。它报告了一个18位计数器的上字节，该字节总计当前框架中所有1296个RAWDATA。要找到平均rawdata值遵循以下公式: 平均像素值= PIX_ACCUM x 1024/1296 最大寄存器值为0xA0（HEX）或160（DEC），最小寄存器值为0。数据总值可以更改每个帧。 在阅读RAWDATA_SUM值之前禁用REST模式。							

注册名称	MAXIMUM_RAWDATA							
银行	-			住址		0x09		
使用权	R			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	MaxRD7	MaxRD6	MaxRD5	MaxRD4	MaxRD3	MaxRD2	MaxRD1	MaxRD0
描述	当前帧中的最大生data值。最小值= 0，最大值=127。最大rawData值可以更改每个帧。							

注册名称	MINIMUM_RAWDATA							
银行	-			住址		0x0A		
使用权	R			默认值		0x7F		
位	7	6	5	4	3	2	1	0
域	MinRD7	MinRD6	MinRD5	MinRD4	MinRD3	MinRD2	MinRD1	MinRD0

描述	当前框架中的最小rawdata值。最小值= 0，最大值=127。最小生data值可以改变每个帧。

注册名称	SHUTTER_LOWER							
银行	-			住址		0x0B		
使用权	R			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	S7	S6	S5	S4	S3	S2	S1	S0

注册名称	SHUTTER_UPPER							
银行	-			住址		0x0C		
使用权	R			默认值		0x01		
位	7	6	5	4	3	2	1	0
域	保留	保留	保留	保留	S11	S10	S9	S8
描述	shutter_lower低8位，shutter_upper是12位快门寄存器的高4位。 先读取shutter_upper，然后连续读取shutter_lower。 调整快门以将平均RAWDATA值保持在正常操作范围内。当在默认模式下操作时，在每个帧上需要检查快门值并自动调整为新值。 快门单元是内部振荡器（名义68MHz）的时钟循环。							

注册名称	CHIP_OBSERVATION							
银行	-			住址		0x15		
使用权	R			默认值		0x80		
位	7	6	5	4	3	2	1	0
域	CO7	CO6	CO5	CO4	CO3	CO2	CO1	CO0
描述	用户必须通过编写0x00来清除寄存器，等待最小tdly_obs ms并读取寄存器。如果芯片正常工作，则chip_observation的寄存器值应为0xb7或0xbf。该寄存器可以用作恢复方案的一部分，以检测由EFT/B或ESD事件引起的问题。 T_{dly_obs}定义为最长的帧周期+ 10%变化。最长的帧周期是芯片处于REST3模式时。需要考虑时钟频率公差值。例如，如果默认使用REST3期500ms，然后T_{dly_obs} = 500ms + 50ms							

登记名称	BURST_MOTION_READ							
银行	-			住址		0x16		
使用权	R			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0

描述	burp_motion_read寄存器用于高速访问运动，观察，delta_x_l，delta_x_h，delta_y_l，delta_y_l，delta_y_h，squal，squal，rawdata_sum，rawdata_sum，maxima_rawdata，mixum _rawdata，miniumum_rawdata，shutter_upper_upper和shutter_upper和shutter_lower_lower_lower_lower_lower_lower coblersels。有关更多详细信息，请参阅第5.7 节。

注册名称	POWER_UP_RESET							
银行	-			住址		0x3A		
使用权	R			默认值		N/A		
位	7	6	5	4	3	2	1	0
域	PRST7	PRST6	PRST5	PRST4	PRST3	PRST2	PRST1	PRST0
描述	将0x5a写入此寄存器以重置芯片，所有设置都将恢复为默认值。从关闭模式恢复后需要重置。							

登记名称	关闭							
银行	-			住址		0x3B		
使用权	R			默认值		N/A		
位	7	6	5	4	3	2	1	0
域	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
描述	编写0xB6以将芯片设置为关闭模式。有关恢复过程的更多详细信息，请参阅第7.2 节。							

注册名称	表现							
银行	-			住址		0x40		
使用权	R/W			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	苏醒	保留	保留	保留	保留	保留	保留	保留
描述	此寄存器配置芯片的操作模式。							

位域	名称	默认值	描述
7	苏醒	0	0：启用休息模式 1：禁用休息模式

登记名称	SET_RESOLUTION							
银行	-			住址		0x47		
使用权	W			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	保留	保留	保留	保留	保留	保留	保留	SET_RES
描述	更新分辨率设置后，在RESOLUTION_X或/和RESOLUTION_Y中，将值0x01写入SET_RESOLUTION，以便芯片使用新的分辨率设置。							

少量字段	名称	默认值	描述
0	SET_RES	0	1：更新分辨率设置

注册名称	RESOLUTION_X_LOW							
银行	-			住址		0x48		
使用权	R/W			默认值		0x63		
位	7	6	5	4	3	2	1	0
域	RESX7	RESX6	RESX5	RESX4	RESX3	RESX2	RESX1	RESX0
注册名称								
银行	-			住址		0x49		
使用权	R/W			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	RESX15	RESX14	RESX13	RESX12	RESX11	RESX10	RESX9	RESX8

描述

该寄存器允许更改芯片X轴的分辨率。

RESOLUTION_X_LOW为低8位，RESOLUTION_X_HIGH为16位的高8位RESOLUTION_X寄存器。

先写入RESOLUTION_X_LOW，然后连续写入RESOLUTION_X_HIGH。

设置X轴分辨率:

0x0000: 50CPI
0x0001: 100CPI
0x0002: 150CPI
:
0x0063: 5000CPI (Default)
:
0x018F: 20000CPI
:
0x0207: 26000CPI (max)

更新分辨率设置后，在RESOLUTION_X或/和RESOLUTION_Y中，将值0x01写入SET_RESOLUTION，以便芯片使用新的分辨率设置。

注意：建议在RIPPLE_CONTROL寄存器中设置bit-7，以在选择9000 CPI及以上时启用纹波控制。

注册名称	RESOLUTION_Y_LOW							
银行	-			住址		0x4A		
使用权	R/W			默认值		0x63		
位	7	6	5	4	3	2	1	0
域	RESY7	RESY6	RESY5	RESY4	RESY3	RESY2	RESY1	RESY0
注册名称								
银行	-			住址		0x4B		
使用权	R/W			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	RESY15	RESY14	RESY13	RESY12	RESY11	RESY10	RESY9	RESY8
描述	该寄存器允许更改芯片Y轴的分辨率。 RESOLUTION_Y_LOW为低8位，RESOLUTION _Y_HIGH为16位的高8位 RESOLUTION_Y寄存器。 先写入RESOLUTION_Y_LOW，然后连续写入RESOLUTION_Y_HIGH。 设置Y轴分辨率: 0x0000: 50CPI 0x0001: 100CPI 0x0002: 150CPI : 0x0063: 5000CPI (Default) : 0x018F: 20000CPI : 0x0207: 26000CPI (max) 更新分辨率设置后，在RESOLUTION_X或/和RESOLUTION_Y中，将值0x01写入 SET_RESOLUTION，以便芯片使用新的分辨率设置。 注意：建议在RIPPLE_CONTROL寄存器中设置bit-7以启用纹波控制 选择9000 CPI及以上。								

登记名称	ANGLE_SNAP							
银行	-			住址		0x56		
使用权	R/W			默认值		0x0D		
位	7	6	5	4	3	2	1	0
域	EN	0	0	0	1	1	0	1
描述	写入该寄存器以启用角度捕捉功能。							
少量字段	名称		默认值	描述				

7	EN	0	0：角度捕捉禁用 1：角度捕捉使能

注册名称	RAWDATA_GRAB							
银行	-			住址		0x58		
使用权	R/W			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	RAWDATA7	RAWDATA6	RAWDATA5	RAWDATA4	RAWDATA3	RAWDATA2	RAWDATA1	RAWDATA0
描述	当启用RawData Grab过程时，该寄存器包含原始数据级别。RawData Grab的详细过程请参考7.1节。							

登记名称	RAWDATA_GRAB_STATUS							
银行	-			住址		0x59		
使用权	R/W			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	PG_VALID	PG_FIRST	保留	保留	保留	保留	保留	保留
描述	写入该寄存器以启用角度捕捉功能。							

少量字段	名称	默认值	描述
7	PG_VALID	0	1：RawData Grab有效
6	PG_FIRST	0	1：先抓RawData

登记名称	RIPPLE_CONTROL							
银行	-			住址		0x5A		
使用权	R/W			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	CTRL8	保留	保留	1	保留	保留	保留	保留
描述	写入该寄存器以启用或禁用纹波控制功能。根据第6.1节中推荐的上电序列启动芯片后，默认禁用纹波控制。							

少量字段	名称	默认值	描述
7	CTRL8	0	0：纹波控制禁用 1：纹波控制使能

注册名称	AXIS_CONTROL							
银行	-			住址		0x5B		
使用权	R/W			默认值		0x60		
位	7	6	5	4	3	2	1	0
域	Swap_XY	INV_Y	INV_X	保留	保留	保留	保留	保留
描述	该寄存器设置芯片上报的轴方向							

位域	名称	默认值	描述
7	Swap_XY	0	1：交换XY方向
6	INV_Y	1	1：反转Y方向
5	INV_X	1	1：反转X方向

登记名称	MOTION_CTRL							
银行	-			住址		0x5C		
使用权	R/W			默认值		0x02		
位	7	6	5	4	3	2	1	0
域	MOT_Set	保留	保留	保留	保留	保留	RES_MOD	保留
描述	配置运动引脚设置并选择X轴和Y轴分辨率模式。							

少量字段	名称	默认值	描述
7	MOT_Set	0	0：运动低电平有效（默认） 1：运动高电平有效
2	RES_Mod	1	0：X轴和Y轴分辨率均由RESOLUTION_X_LOW和RESOLUTION_X_HIGH定义； 1：X轴分辨率由RESOLUTION_X_LOW和RESOLUTION_X_HIGH定义，Y轴分辨率由RESOLUION_Y_LOW和RESOLUION_Y_HIGH定义（默认） 0为两个寄存器配置，1为四个寄存器配置配置。

登记名称	INV_PROD_ID							
银行	-			住址		0x5F		
使用权	R/W			默认值		0xAE		
位	7	6	5	4	3	2	1	0
域	IPID [7:0]							
描述	该寄存器值是Product_ID寄存器值的倒数。用于测试SPI口硬件							

注册名称	RUN_DOWNSHIFT							
银行	-			住址		0x77		
使用权	R/W			默认值		0x14		
位	7	6	5	4	3	2	1	0
域	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
描述	此寄存器将运行设置为REST1降档时间。使用以下公式进行计算。 运行降档时间（MS）= RUN_DOWNSHIFT [7：0] x RUN_DOWNSHIFT_mult（默认256）x 50μs 根据建议的电力序列，在芯片启动时，run_downshift设置为默认值为1。 默认运行降档= 79（0x4f）x 256 x 50μs = 1s 最小值为0x01。0x00的值将在内部剪辑至0x01。上述所有值的公差预期为±10%。							

注册名称	REST1_PERIOD							
银行	-			住址		0x78		
使用权	R/W			默认值		0x01		
位	7	6	5	4	3	2	1	0
域	R1R7	R1R6	R1R5	R1R4	R1R3	R1R2	R1R1	R1R0
描述	此寄存器设置剩下的1个周期。 REST1周期= REST1_PERIOD [7：0] x 1ms 根据建议的电力序列，在芯片启动时，REST1_PERIOD设置为1ms，为默认值。 默认REST1周期= 1（0x01）x 1ms = 1ms 最小值为0x01。值为0x00无效。上述所有值的公差预期为±10%。							

登记名称	REST1_DOWNSHIFT							
银行	-			住址		0x79		
使用权	R/W			默认值		0x90		
位	7	6	5	4	3	2	1	0
域	R1D7	R1D6	R1D5	R1D4	R1D3	R1D2	R1D1	R1D0
描述	此寄存器将REST1设置为REST2降档时间。使用以下公式进行计算。 REST1降档时间（MS）= REST1_DOWNSHIFT [7：0] x REST1_DOWNSHIFT_MULT（默认64）x REST1期（默认1MS） 根据建议的电力序列，将REST1_Downshift设置为默认值10。 默认值= 156（0x9c）x 64 x 1ms = 9984ms = 10s 最小值为0x01。0x00的值将在内部剪辑至0x01。上述所有值的							

	公差预期为±10％。

注册名称	REST2_PERIOD							
银行	-			住址		0x7A		
使用权	R/W			默认值		0x19		
位	7	6	5	4	3	2	1	0
域	R2P7	R2P6	R2P5	R2P4	R2P3	R2P2	R2P1	R2P0
描述	该寄存器设置Rest2周期。 Rest2周期= Rest2_Period[7:0] x慢时钟(1ms) x 4 根据推荐的上电序列启动芯片后，REST2_PERIOD默认设置为100ms。 默认Rest2周期= 25 (0x19) x 1ms x 4= 100ms最小 值为0x01。值0x00无效。 上述所有值预计具有± 10%的公差。							

注册名称	REST2_DOWNSHIFT							
银行	-			住址		0x7B		
使用权	R/W			默认值		0x5E		
位	7	6	5	4	3	2	1	0
域	R2D7	R2D6	R2D5	R2D4	R2D3	R2D2	R2D1	R2D0
描述	该寄存器设置Rest2到Rest3的降档时间。使用下面的公式进行计算。 Rest2降档时间(ms) = Rest2_Downshit[7:0] x REST2_DOWNSHIFT_MULT（默认64）x rest2_period（默认100ms） 芯片按照推荐的上电顺序启动后，Rest2降档时间默认设置为10分钟。 默认= 94(0x5E) x 64 x 100ms = 601.6s = 10min 最小值为0x01。0x00的值将在内部剪辑至0x01。上述所有值的公差预期为±10%。							

注册名称	REST3_PERIOD							
银行	-			住址		0x7C		
使用权	R/W			默认值		0x3F		
位	7	6	5	4	3	2	1	0
域	R3P7	R3P6	R3P5	R3P4	R3P3	R3P2	R3P1	R3P0
描述	该寄存器设置Rest3周期。 Rest3周期= Rest3_Period[7:0] x慢时钟(1ms) x 8 根据推荐的上电序列启动芯片后，Rest3周期默认设置为504ms。 默认Rest3周期= 63(0x3F) x 1ms x 8 = 504ms最小 值为0x01。值0x00无效。							

	上述所有值预计具有± 10%的公差。

注册名称	RUN_DOWNSHIFT_MULT							
银行	-			住址		0x7D		
使用权	R/W			默认值		0x07		
位	7	6	5	4	3	2	1	0
域	保留	保留	保留	保留	RUN_M3	RUN_M2	RUN_M1	RUN_M0
描述	该寄存器设置运行降档乘数。 (详见寄存器RUN_DOWNSHIFT中的公式)							

位域	名称	默认值	描述
3:0	RUN_M[3:0]	7	RUN_DOWNSHIFT_MULT 0的寄存器值 : 2 1: 4 2: 8 3: 16 4: 32 5: 64 6: 128 7: 256 8: 512 9: 1024 10: 2048

注册名称	REST_DOWNSHIFT_MULT							
银行	-			住址		0x7E		
使用权	R/W			默认值		0x55		
位	7	6	5	4	3	2	1	0
域	保留	REST_M6	REST_M5	REST_M4	保留	REST_M2	REST_M1	REST_M0
描述	该寄存器设置REST降档乘数。 (参考寄存器REST1_DOWNSHIFT和REST2_DOWNSHIFT中的公式)							

位域	名称	默认值	描述
6:4	REST_M[6:4]	5	REST2_DOWNSHIFT_MULT 0的寄存器值 : 2 1: 4 2: 8 3: 16 4: 32 5: 64 6: 128 7: 256
2:0	REST_M[2:0]	5	REST1_DOWNSHIFT_MULT 0的寄存器值 : 2 1: 4 2: 8 3: 16 4: 32 5: 64 6: 128 7: 256

注册名称	ANGLE_TUNE1							
银行	-			住址		0x0577		
使用权	R/W			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	ANGLE7	ANGLE6	ANGLE5	ANGLE4	ANGLE3	ANGLE2	ANGLE1	ANGLE0
描述	该寄存器设置REST降档乘数。 (参考寄存器REST1_DOWNSHIFT和REST2_DOWNSHIFT中的公式)							

位域	名称	默认值	描述
7:0	ANGLE[7:0]	0x00	0xE2：-30度 0xF6：-10度 0x00：0度（默认） 0x0F：+15度

			0x1E：+30度

注册名称	ANGLE_TUNE2							
银行	-			住址		0x0578		
使用权	R/W			默认值		0x00		
位	7	6	5	4	3	2	1	0
域	EN	保留	保留	保留	保留	保留	保留	保留
描述	写入该寄存器以启用角度调整功能							

位域	名称	默认值	描述
7	EN	0	0：角度调整禁用（默认） 1：角度调整使能

注册名称	LIFT_CONFIG							
银行	-			住址		0x0C4E		
使用权	R/W			默认值		0x08		
位	7	6	5	4	3	2	1	0
域	保留	保留	保留	保留	1	保留	LIFT1	LIFT0
描述	写入该寄存器以启用角度调整功能							

位域	名称	默认值	描述
			0：1mm设置（默认） 2：2mm设置

8.3 寄存器写入的位掩码

需要特别注意一些寄存器有“保留”位。为了覆盖寄存器中的特定位，需要先读取并存储其当前值，然后应用位掩码并将新值写回到寄存器中。这是通过使用按位运算符实现的，例如AND(&)、OR(|)或INVERSE(~)。

例子:

禁用寄存器0x40中的Rest模式（将位7设置为1）

读取寄存器0x40并存储在VarA

VarA |= 0x80
将值VarA写入寄存器0x40

在寄存器0x40中启用休息模式（将位7设置为0）

读取寄存器0x40并存储在VarA VarA

&= ~ 0x80

将值VarA写入寄存器0x40

修订记录

修订号	日期	描述
0.8	2021年1月13日	初始创建