GPIO&LED 基础应用说明

1. 概述

本文是 AW GPIO&LED 技术原理、功能介绍、软件和设计指南相关信息的集合，它提供了利用 AW GPIO&LED 开发工作所需的一切。

本指南被分为以下部分。

原理介绍 - 本章概述了三通道直驱 LED 工作原理，应用场景，以及器件家族。

功能 – 介绍了 AW 三通道直驱 LED 芯片模式说明和功能。

原理图设计说明 – 为三通道直驱 LED 典型硬件原理图设计和布局提供指导。

芯片用法 – AW 三通道直驱 LED GUI 是一款快速调试工具，本章详细介绍开发人员如何通过 AW 三通道直驱 LED demo 板搭配 GUI 完成项目开发调试。

软件移植指南 – 此章节讲解了 AW 三通道直驱 LED MCU/MTK 驱动移植指南和软件使用流程

常见问题 – 此章节介绍了 AW 三通道直驱 LED 常见的问题 FAQ

2. 简介

AW952X 系列产品是 I2C 控制的 16 路输出多功能芯片，包括 AW9523BTQR 和 AW9527QNR 两款产品，既可以作为 GPIO 扩展，也可以作为 LED 驱动使用；作 GPIO 扩展时所有接口都可以独立地设置为输入或输出，作 LED 驱动时支持 256 级调光；其中 AW9527 支持呼吸功能，且封装更小，AW9523BTQR 的封装为 QFN 4×4-24L，AW9527 的封装为 QFN 3×3-24L。

AW9527 支持 GCC 线性调节电流，同时支持自主呼吸模式，具备灵活高效的照明效果编程功能，可以减少控制器资源占用。

AW952X 是一款既可以作为 GPIO 扩展，又可以作为 LED 驱动的芯片，这就决定了其具有更多的应用场景。AW952X 常用于各类设备（如键盘、智能门锁等）内部电路中控制单元的 GPIO 扩展，以及各类设备（如 IoT、运动设备等）的 LED 驱动。

电竞键盘

游戏手柄

智能门锁

智能茶具

智能跳绳

图 2-1 产品应用示例

3. 功能

3.1 I2C 地址

AW9523X_AW9527、AW91XXX 系列芯片的 I2C 地址由引脚 AD0 和 AD1 配置，支持 4 个设备地址。I2C 地址的 Bit7 ~ Bit3 为固定值“10110”，Bit2 和 Bit1 的值分别由 AD1 和 AD0 的电平决定，Bit0 表示 I2C 协议的读写控制位，AD0/AD1 和 I2C 地址的关系如图 2-1

Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	1	1	0	AD1	AD0	R/W

图 3-1 AD0/AD1 和 I2C 地址的关系

当 AD1=GND，AD0=GND 时，Bit2-Bit1 为“00”，I2C 七位地址为 0x58，

当 AD1=GND，AD0=VCC 时，Bit2-Bit1 为“01”，I2C 七位地址为 0x59，

当 AD1=VCC，AD0=GND 时，Bit2-Bit1 为“10”，I2C 七位地址为 0x5A，

当 AD1=VCC，AD0=VCC 时，Bit2-Bit1 为“11”，I2C 七位地址为 0x5B。

3.2 中断

当端口配置成 GPIO 输入模式并且使能中断时，AW9523X_AW9527、AW91XXX 系列芯片会检测输入口的电平状态。当输入口电平状态变化时，芯片会将 INTN 引脚拉低。主控可通过读取 INPUT_PORT0（00H）或 INPUT_PORT1（01H）寄存器来清除芯片的中断状态。中断清除后，芯片将停止拉低 INTN 引脚。在应用中，INTN 引脚需要外接上拉电阻。中断触发时序参考图 3-2。

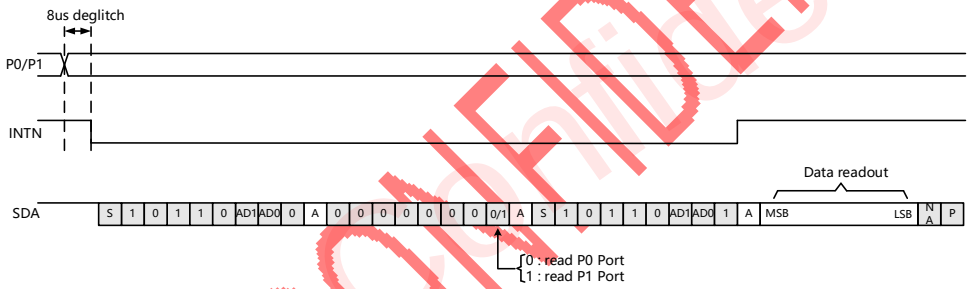


图 3-2 中断触发时序图

1. 当使能中断的输入端口检测到电平变化，从高电平到低电平或者从低电平到高电平。芯片内部会自动进行 8μs 抗尖峰干扰处理，确保状态变化的有效性。

2. 当输入端口电平变化有效，AW9523X_AW9527、AW91XXX 系列芯片会拉低 INTN 引脚。

3. 当触发中断导致 INTN 引脚被拉低后，主控制器可通过读取 INPUT_PORT0（00H）和 INPUT_PORT1（01H）寄存器来清除中断事件。清除中断后，芯片将停止拉低 INTN 引脚。INTN 引脚状态将由外部电平决定。

3.3 上电时序

AW9523X_AW9527、AW91XXX 在使用时，需要严格遵守上电时序，如图 3-3，RSTN 需要在上电 100μs 后方可拉高使芯片进入工作模式，再 5ms 之后才可以进行 I²C 配置。

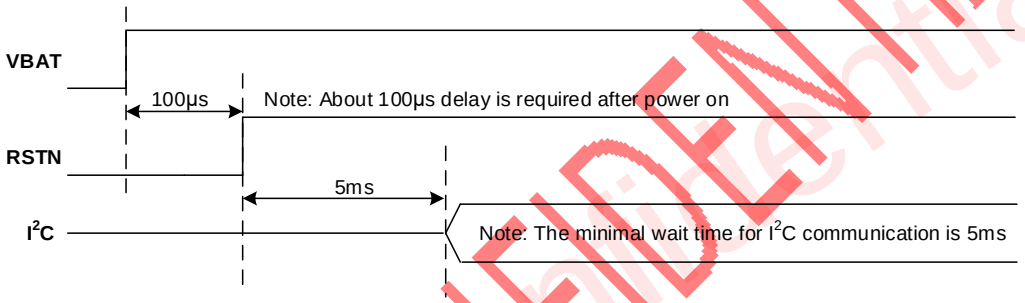


图 3-3 AW952X 上电时序图

当主控芯片(MCU)的 GPIO 不够，需将 RSTN 直接拉高时，可以通过一个 RC 延时电路，将 RSTN 的激活延时至 VCC 上电后 100μs，如图 3-4 所示：

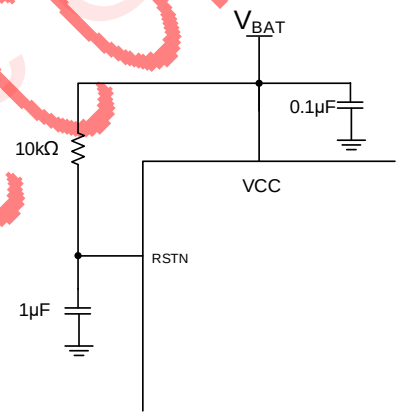


图 3-4 AW952X RSTN 固定拉高连接示意图

应注意的是，当作为开漏 GPIO 应用时，上拉电压不应早于 AW9523X_AW9527、AW91XXX 供电。

3.4 芯片复位

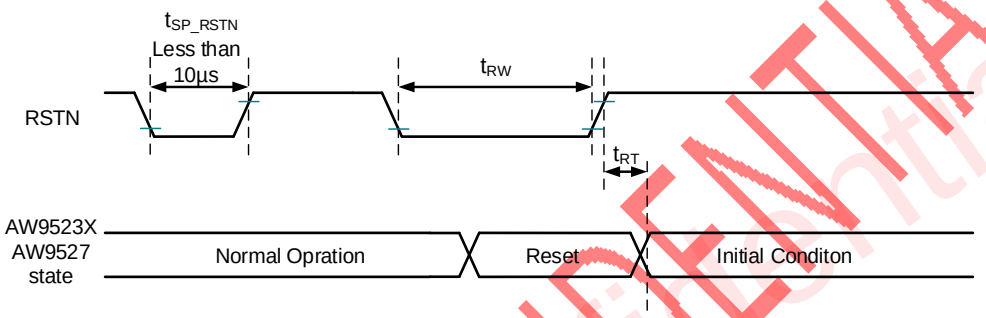


图 3-5　AW9523X_AW9527 硬复位时序图

AW9523X_AW9527、AW91XXX 芯片的复位功能如下：

- 上电复位：上电 5ms 后，芯片初始化至默认状态。
- 硬复位：RSTN 引脚维持低电平状态持续 20μs 以上，芯片内部所有电路被复位。
- 软复位：对 SW_RSTN（7FH）寄存器写 00H，芯片所有内部电路被复位。

芯片软复位参考代码如下：

（说明：AW91XXX 与 AW9523X_AW9527 软复位功能相同，函数名称有差别 aw91xxx_soft_rst）

```
1.  /* AW9523X_AW9527 software reset */
2.  static void aw9523x_aw9527_soft_rst(void)
```

```
3.  {
4.      aw_i2c_write_reg(SW_RSTN, 0x00);
5.      /* delay 2ms at least */
6.      delay_ms(2);
7.  }
```

4. 原理图/PCB LAYOUT 设计

4.1 原理图设计

4.1.1 AW952X GPIO应用电路设计

AW952X 可以作为 GPIO 扩展或 LED 驱动使用，作 GPIO 扩展时的参考应用原理图如图 4-1：

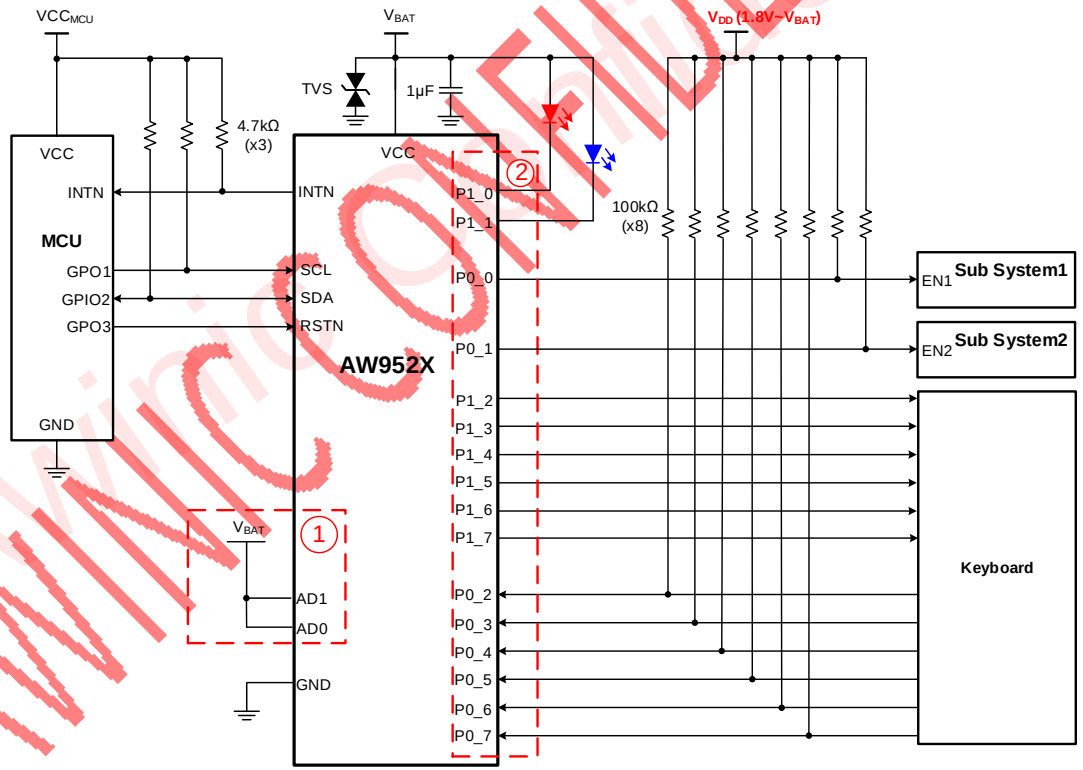
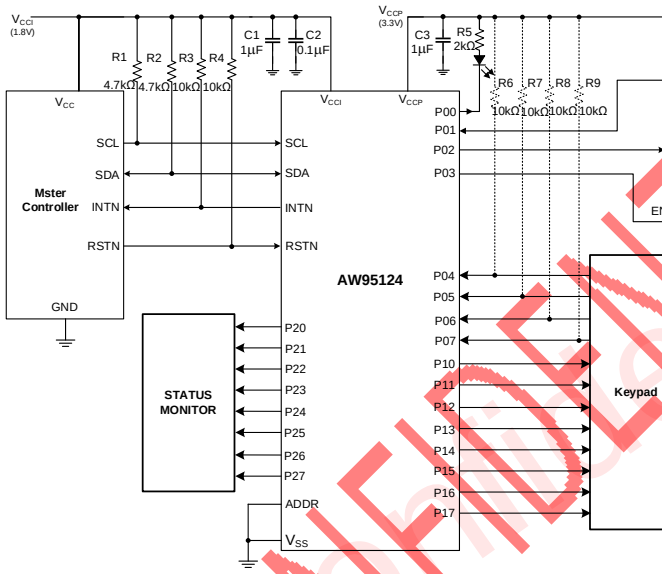


图 4-1 AW952X 作 GPIO 扩展应用电路图

1. 输入电容推荐使用 1μF 的 X5R 或 X7R 电容，SCL、SDA、INTN 需要配置一个上拉电阻；
2. AW952X I2C 地址可通过 AD1、AD0 选择，在输出存在 LED 时，需将 AD1、AD0 连接至 VBAT，保证 GPIO 的初始状态为高电平或高阻态，从而使得 LED 不会在系统上电时被错误地点亮；
3. AW952X 的每个输出都可以独立的被配置为 GPIO 或 LED 驱动。作 GPIO 输出时，P1 端口为 Push-Pull 输出；P0 端口默认为 Open-Drain 输出，也可以将其配置为 Push-Pull 输出，当作为 Open-Drain 输出时则需要为其准备上拉电阻；
4. AW952X 在驱动开关器件如 MOSFET 或 BJT 时，建议在 GPIO 输出端预留一组 RC 延时电路；
5. 建议在电源端，以及需要外接到 PCB 板外的接口处预留 TVS 位置用于防止可能会发生的 ESD 或浪涌电压。

4.1.2 AW9XXX GPIO应用电路设计

AW95124 作为 I2C 从设备通常位于主机的远程位置，靠近主机需要监控的 GPIO 端，图 4-2 为 AW95124 的典型应用电路，AW95124 作为 IO 扩展器通常用于控制 LED（用于反馈或状态灯）、控制其他设备的启用或重置信号，甚至读取其他设备或按键的输出状态等。



(1) In this application schematic, P00, P02, P03 andP10~P27 are configured as outputs.
(2) P01 and P04~P07 are configured as inputs.
(3) Device address is configured as 0100010 for this example.

图 4-2 AW95xxx 作 GPIO 扩展应用电路图

1. AW95124 的电压范围为 VCCI：1.08V～3.6V，VCCP：1.08V～3.6V，电源输入端的并联电容推荐使用 1μF 和 0.1μF，当电源电压 VCCI 不大于 3.3V 时，C1~C4 选用额定电压为不小于 6.3V 的电容；当电源大于 3.3V 时，C1~C4 选用额定电压为不小于 10V 的电容。

2. AW95124 芯片的 GPIO 口配置为输入状态时，可通过寄存器配置内部的上下拉功能（芯片内部的上拉电阻为 100kΩ，内部的下拉电阻为 100kΩ）；输入的高电平需不小于 Vccp，AW95124 应用中不会增加系统功耗，若输入高电平低于 VCCP，则 AW95124 芯片功耗会增加(AW95124 每路会增加功耗)。

3. AW95124 芯片的 IO 口配置为输出状态时，默认是推挽输出模式，当 GPIO 口配置成推挽输出模式时，对应的 GPIO 口不要与其他电压相连，若外加的电压大于 VCCP，则会出现往芯片内部倒灌电流的现象，若外加的电压低于 VCCP 或若在 IO 外部拉低时，则会进一步增加系统功耗。

4. 当 GPIO 口配置为开漏输出时，外加的电压与 VCCP 电压大小无关，高于或者低于 VCCP 都不会出现倒灌电流和芯片的整体功耗增加问题(外加的电压不能高于 3.6V)，注意端口的上拉电压与相连芯片的 VIL/VIH 匹配；

5. 当 AW95124 的 IO 口为输出状态时，内部的上下拉电阻配置无法使用，在应用时需注意；

6. AW95124 芯片的 GPIO 口作为输入端口，默认输入为高阻态，不需要的管脚推荐接 VCCP 或 GND（需与固定电平相连，否则容易受外部干扰造成状态的变化，可以选择芯片内部上下拉电阻配置或预留外部电阻连接至 VCCP 或 GND），但选择内部上下拉电阻配置时，芯片上电默认状态为输入悬空态，应用时需注意；

7. 每个 I/O 灌电流必须外部限制为最大 25mA，每组 IO 口（P07~P00、P17~P10 和 P27~P20）必须限制为最大 100mA 电流，芯片总电流为 200mA。

8. 每个 I/O 口的输出电流限制为 10mA，每组 IO 口（P07~P00、P17~P10 和 P27~P20）必须限制为最大 80mA 电流，芯片总电流为 160mA。

9. I2C 端口是开漏类型，需要上拉电阻 R1、R2 至 VCCI 才能够正常通信，在应用时，SCL、SDA 信号的上拉电平不能低于 VCCI，建议接上拉电阻至 VCCI；当输入高电平低于 VCCI 时，芯片的功耗会增加（每路会增加额外的功耗）且需注意通讯电平是否满足对应端口的 VIH/VIL 要求；

10. 当 I2C 通信速率为是 400kHz 时，默认上拉电阻 R1 与 R2 取值为 2.2kΩ，若主机端已有上拉电阻，则 SCL/SDA 端的外部上拉电阻可省略，R1/R2 具体阻值应根据以下推荐的外部上拉电阻 RP 计算方式进行选择；当 I2C 通信速率为是 1000kHz 时，默认上拉电阻 R1 与 R2 取值为 1kΩ，若主机端已有上拉电阻，则 SCL/SDA 端的外部上拉电阻可省略，R1/R2 具体阻值应根据以下推荐的外部上拉电阻 RP 计算方式进行选择；

    外部上拉电阻 RP 的计算方式如下：

    RP(min)是 VCCI，VOL(max)与 IOL 的函数：$R_P(\min) = \frac{V_{CCI}-V_{OL}(max)}{I_{OL}}$；

    RP(max)是最大上升时间 tr（针对快速模式，tr=300ns，fSCL=400kHz；针对快速+模式，tr=120ns，fSCL=1000kHz）与总线电容 Cb 的函数：$R_P(\max) = \frac{t_r}{0.8473\times C_b}$；

其中，参数 VOL，IOL，tr 与总线电容 Cb 可参考下表中的规范所示。总线电容 Cb 可以通过将 AW95124 上增加的电容，SCL 引脚电容 Ci 或者 SDA 引脚电容 Cio，走线与连接处等的寄生电容以及同一总线上其他从设备的电容相加来近似评估，最终根据功耗与速度的要求选择出合适的上拉电阻。对于 I2C 总线的标准模式或快速模式，最大总线电容 Cb 均不得超过 400pF。对于 I2C 总线的快速+模式，最大总线电容 Cb 均不得超过 550pF。

参数		标准模式		快速模式		快速模式+		单位
		Min	Max	Min	Max	Min	Max	
t_r	SDA 和 SCL 信号的上升时间	-	1000	20	300	-	120	ns

C_b	每条总线的容性负载	-	400	-	400	-	550	pF
V_{OL}	低电平输出电压(在3mA灌电流，V_{CCI} >2V时)	0	0.4	0	0.4	0	0.4	V
	低电平输出电压(在2mA灌电流，V_{CCI}≤2V时)	0	-	0	0.2*V_{CCI}	0	0.2*V_{CCI}	V
I_{OL}	低电平输出电流	3	-	3	-	20	-	mA

11. INTN 端口是开漏输出，需要电阻 R3 上拉至 VCCI，推荐使用 10kΩ，具体阻值可根据调试进行选择；与 INTN 端口相连的 MCU 的 IO 口，需要支持唤醒功能。

12. 当 IO 口只用做输出模式时，中断 INTN 可以 NC 处理，或者将 INTN 引脚固定上拉。

13. RSTN 端口是输入引脚，需要电阻 R4 上拉至 VCCI，推荐使用 10kΩ，具体阻值可根据调试进行选择；与 RSTN 端口相连的 MCU 的 IO 口为 OD 输出引脚。上拉的电源建议直接接 VCCI，当输入高电平低于 VCCI 时，AW95124 芯片会增加功耗且且需注意上拉电平是否满足对应端口的 VIH/VIL 要求；

14. AW95124 的 I2C 地址可通过 ADDR 进行拓展，ADDR 引脚可直接接 GND、SCL、SDA、高电平，需要接高电平时，建议直接接 VCCI，当输入高电平低于 VCCI 时，AW95124 会增加功耗且需注意上拉电平是否满足对应端口的 VIH/VIL 要求；AW95124 的 7 位 I2C 地址可设置为 0x20~0x23，同一路 I2C 总线上最多可支持 4 个 AW95124 器件，AW95124 的 I2C 地址选择详见下表：

固定位	ADDR	I2C 地址
0100	SCL	0x20
	SDA	0x21
	GND	0x22
	V_{CCI}	0x23

4.1.3 AW91XXX GPIO应用说明

1. 使用 AW9110C/AW9106C 作为 GPIO 来应用时，需要注意的是，默认为 HiZ 的端口不可直接用作 GPIO 输入使用，如需使用，需要在该端口增加若下拉或若上拉，以保证该端口的状态在初始化后为稳定状态；
2. 在使用中断功能时，INTN 引脚内部为 open drain 架构，因此外部需要通过 kΩ级上拉电阻上拉。一旦产生中断，需要对每个端口都进行读清操作，即对 00H、01H 寄存器分别进行读操作。

4.1.4 AW952X LED应用电路设计

AW952X 作为 LED 驱动时参考应用图如图 4-3 所示：

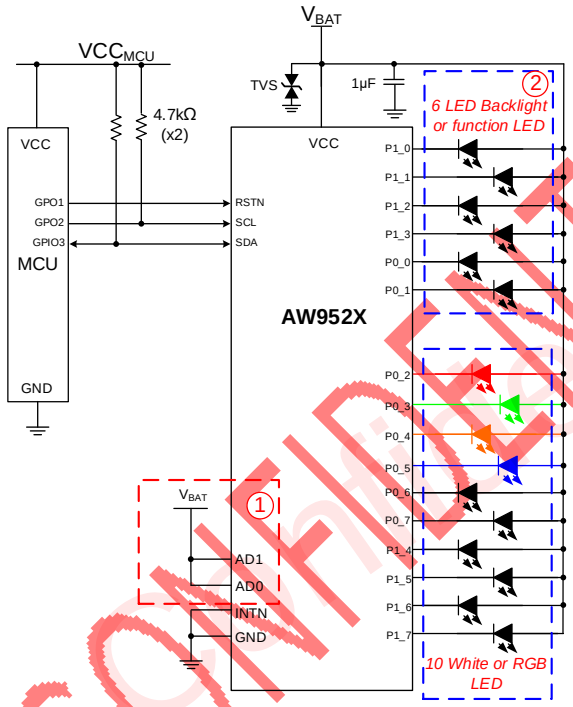


图 4-3 AW952X 作 LED 驱动应用电路图

1. 输入电容推荐使用 1μF 的 X5R 或 X7R 电容，SCL、SDA 需要配置一个上拉电阻；
2. AW952X 作 LED 驱动时，AD1 和 AD0 必须连接至 VBAT，以保证输出端口的初始状态为高电平或高阻态，避免 LED 被误点亮；
3. AW952X 有六个通道的 Dropout 电压被专门优化过：P1_0~P1_3、P0_0~P0_1，这些通道在需要时可以作为 LED 背光使用；
4. 建议在电源端，以及需要外接到 PCB 板外的接口处预留 TVS 位置用于防止可能会发生的 ESD 或浪涌电压；
5. AW9527 在小电流应用时，可以作为矩阵灯驱动，如图 4-4 所示。

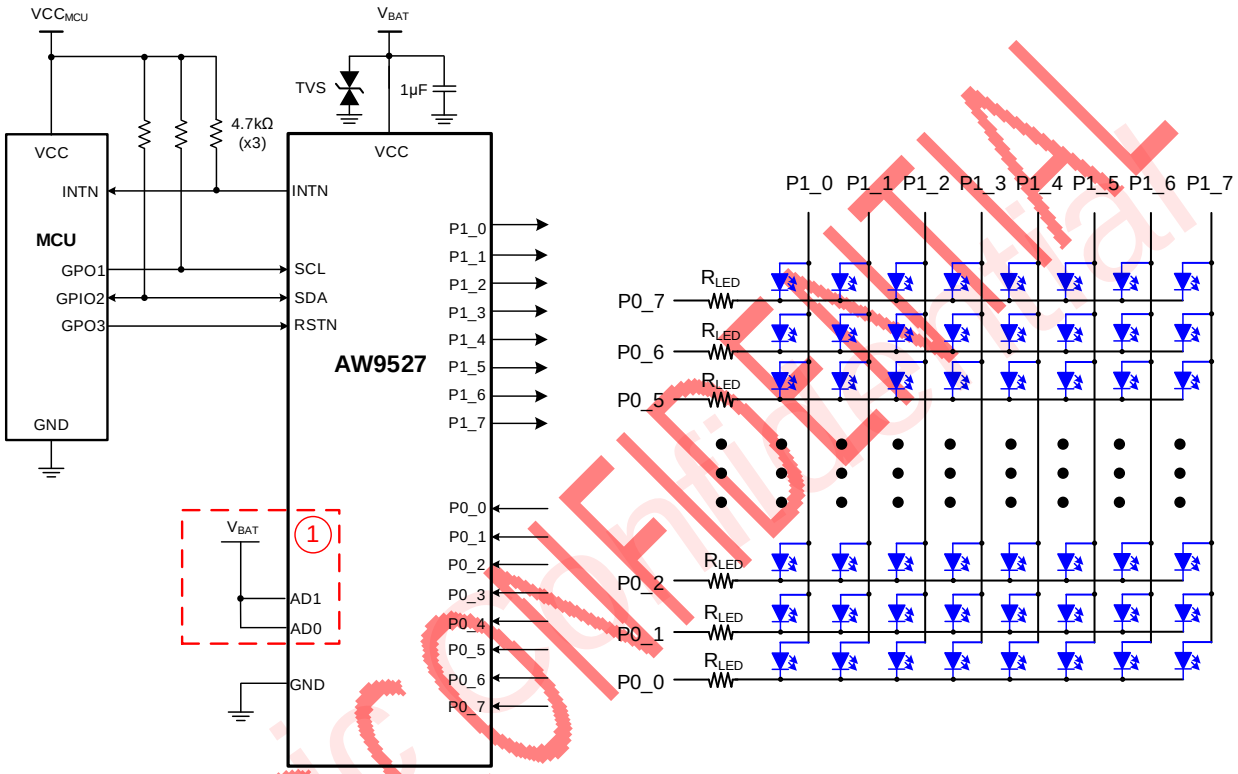


图 4-4 做矩阵 LED 应用电路图

4.1.5 AW91XXX LED应用电路设计

以 AW9110C 为例，采用 1 颗芯片驱动 10 路 LED 或者 6 路背光的应用参考电路图如图 4-5：

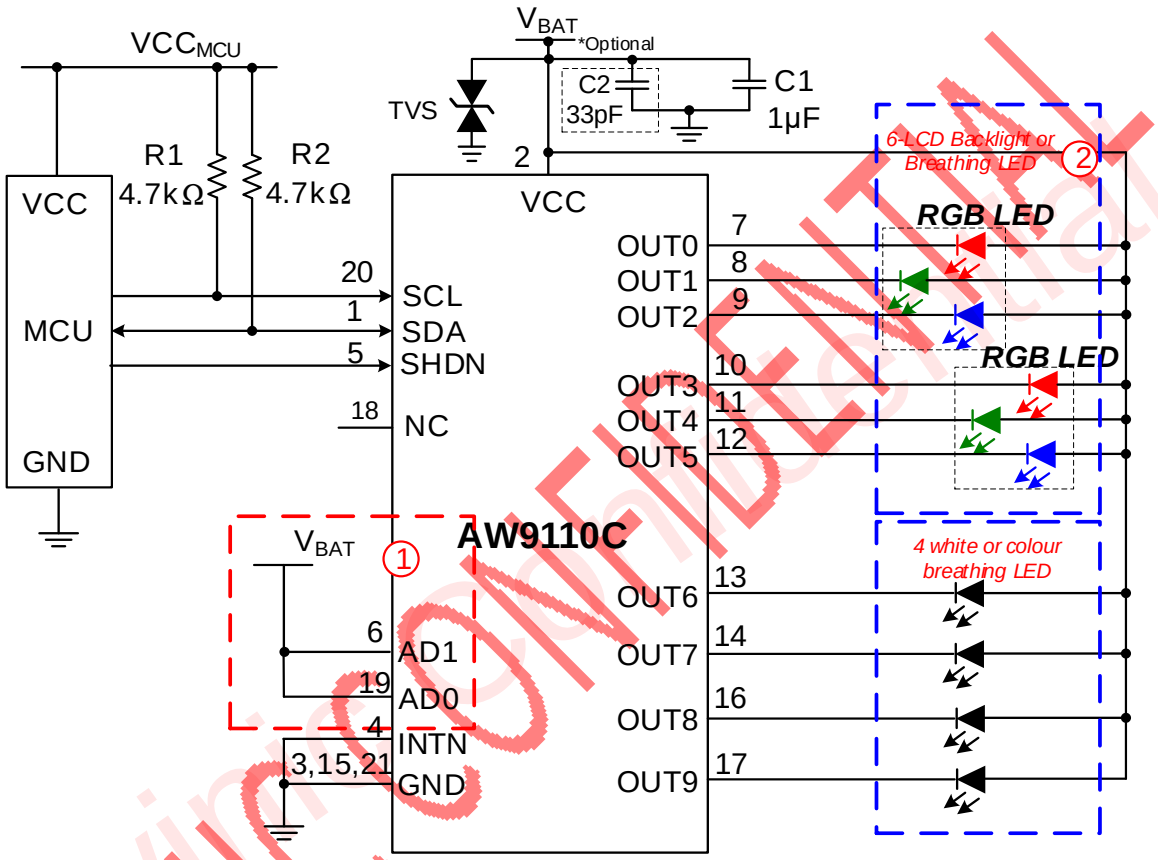


图 4-5 AW9110C 应用电路图

当 LED 阳极供电为 VBAT 时， AD1/AD0 需要接至 VBAT，以保证芯片在上电后 OUTx 的初始态为高，而后再通过寄存器配置芯片进入 LED 模式。如未按照此硬件连接，芯片在上电后，灯会出现误点亮的现象。

使用 AW9110C/AW9106C 作为 LED 模式，配置呼吸参数（对应 15H、16H 寄存器）时，需要注意 fade-on、fade-off、all-on、all-off 这四个呼吸参数 3-bit 仅可以配置为 001、010、011、100、101，对应关系如下：

3-bit 呼吸参数	呼吸时间

001	315ms
010	630ms
011	1260ms
100	2520ms
101	5040ms

4.2 PCB Layout 参考设计

4.2.1 AW9527 PCB LAYOUT

1. 电源去耦电容靠近芯片 VCC 引脚放置，应保持 GND 的完整性；
2. 输入电容应选用 X5R 或 X7R 的陶瓷电容，容值推荐为 0.1~1μF，可以根据电源稳定情况来决定；
3. 做矩阵灯应用时，应注意保持走线之间的安全间距以减小寄生效应，寄生电容过大可能会导致鬼影现象。

4.2.2 AW95XXX PCB LAYOUT

1. AW95124 是一款带有 I2C 接口的 24 位 IO 扩展器，电源去耦电容尽可能靠近芯片 VCCI 和 VCCP 引脚放置，且遵循电流先经过大电容再经过小电容，最后到芯片的准则；
2. 在应用中，根据 IO 口的驱动能力设置合适的 IO 口走线线宽；
3. AW95124 芯片 GND 引脚若通过地孔与主板地相连，为了达到更好的散热效果，过孔应尽可能多些。

4.2.3 AW9110C_AW9106C PCB LAYOUT

1. 电源去耦电容（C1）靠近芯片 VCC 管脚放置，为了防止应用中可能出现的高频毛刺电压，可在 VCC 引脚预留一个小电容 C2，容值可选 nF 或 pF 级别；
2. I2C 信号线 SCL、SDA 注意包地处理；
3. 保证芯片地的完整性，并与主地有较好的连接，避免因地的不完整产生较大的寄生。

5. 芯片用法

5.1 GUI 和 EVB 使用方法

5.1.1 EVB板名称

AW9523_AW91XX_AW9527_EVB_V1.0。

5.1.2 EVB板概述

AW9523_AW91XX_AW9527_EVB_V1.0 是艾为测试 GPIO 扩展和呼吸灯功能的测试板，可以对 AW9523B、AW9110C、AW9106C、AW9527 进行演示和测试。

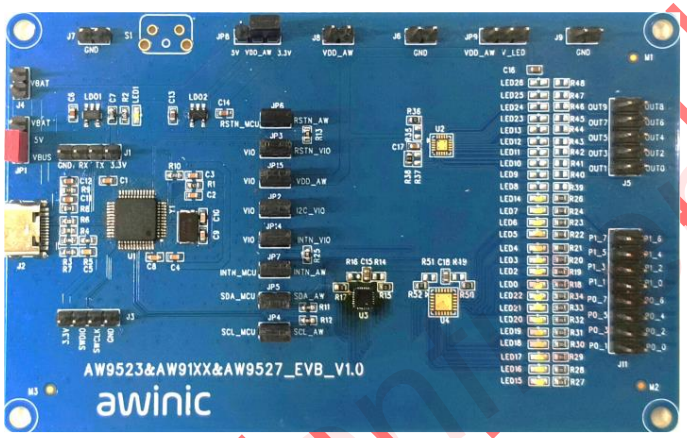
5.1.3 EVB描述与特性

AW9523_AW91XX_AW9527_EVB_V1.0 可支持芯片型号为 AW9523B、AW9110C、AW9106C 与 AW9527。

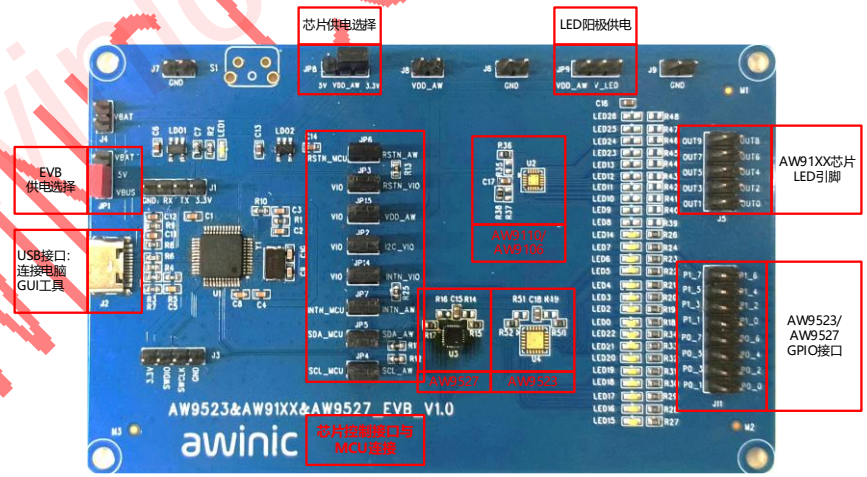
AW9523B 和 AW9527 支持 16 路 GPIO 扩展和 LED 驱动，AW9110C 支持 10 路 GPIO 扩展和 LED 驱动，AW9106C 支持 6 路 GPIO 扩展和 LED 驱动。应用时，每一路可以单独配置为 GPIO/LED 模式。GPIO 模式时，也可以任意配置输入、输出模式。

AW952X 和 AW91XX 均支持外部引脚电压选择 I2C 地址，共四个地址可选，方便在多颗使用或者地址冲突时进行修改。芯片上电后引脚默认为输出模式，具体输出状态和芯片地址相关，具体对应关系参考手册。

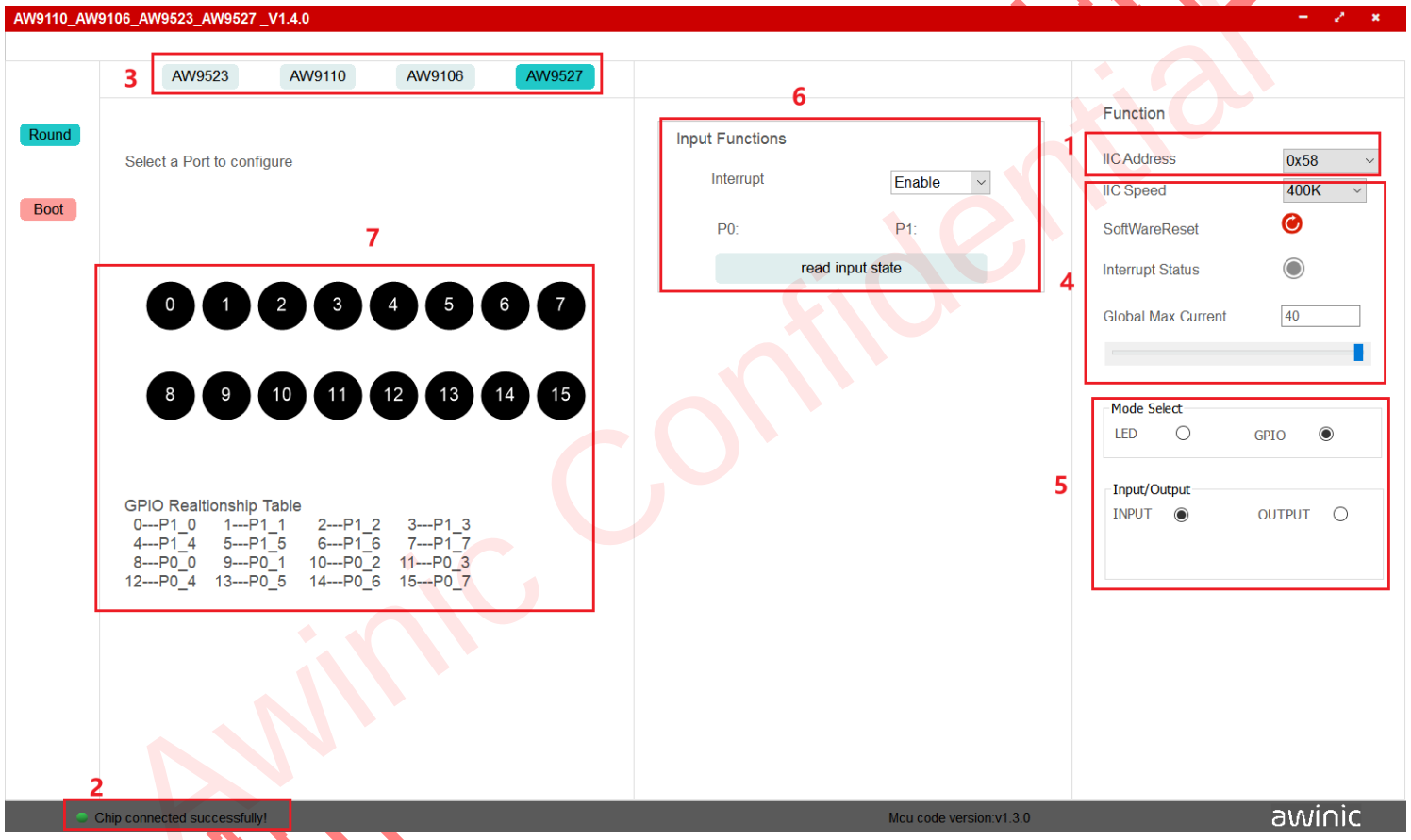
5.1.4 评估板图片



5.1.5 硬件概述



5.1.6 环境搭建



1. 芯片地址选择，可根据外部电阻更改地址，按照板上实际电阻选择对应地址即可；

2. 连接状态，插入 USB 线后，显示 Chip connected successfully!即为连接成功；

3. demo 板上实际贴片的芯片，插入 USB 连接 UI 后自动选择；

4. 芯片软复位以及 LED 模式下全局电流调节；

5. 芯片模式选择，可选择 LED 或者 GPIO 模式，以及 GPIO 模式下 IO 口输入输出模式选择；

6. 在 5 处选择不同模式后，显示对应的调试界面。上图为默认输出模式时界面，可选择 open drain 或者 push pull 输出以及输出电平；

一、GPIO 模式测试

1. 芯片供电 2.5V-5.5V，成功连接 UI；

2. 更改芯片为 GPIO 模式，输入或者输出模式：

   A．输出模式：先选择开漏或者推挽输出，开漏模式需要外接上拉电阻。然后选择需要输出的状态，高电平或者低电平。

   B．输入模式：按照需求使能对应的 IO 口，默认为全部使能。外部对对应 IO 口进行拉高拉低操作，点击 read input state，即可读取 P0 和 P1 口的状态。

二、LED 模式测试

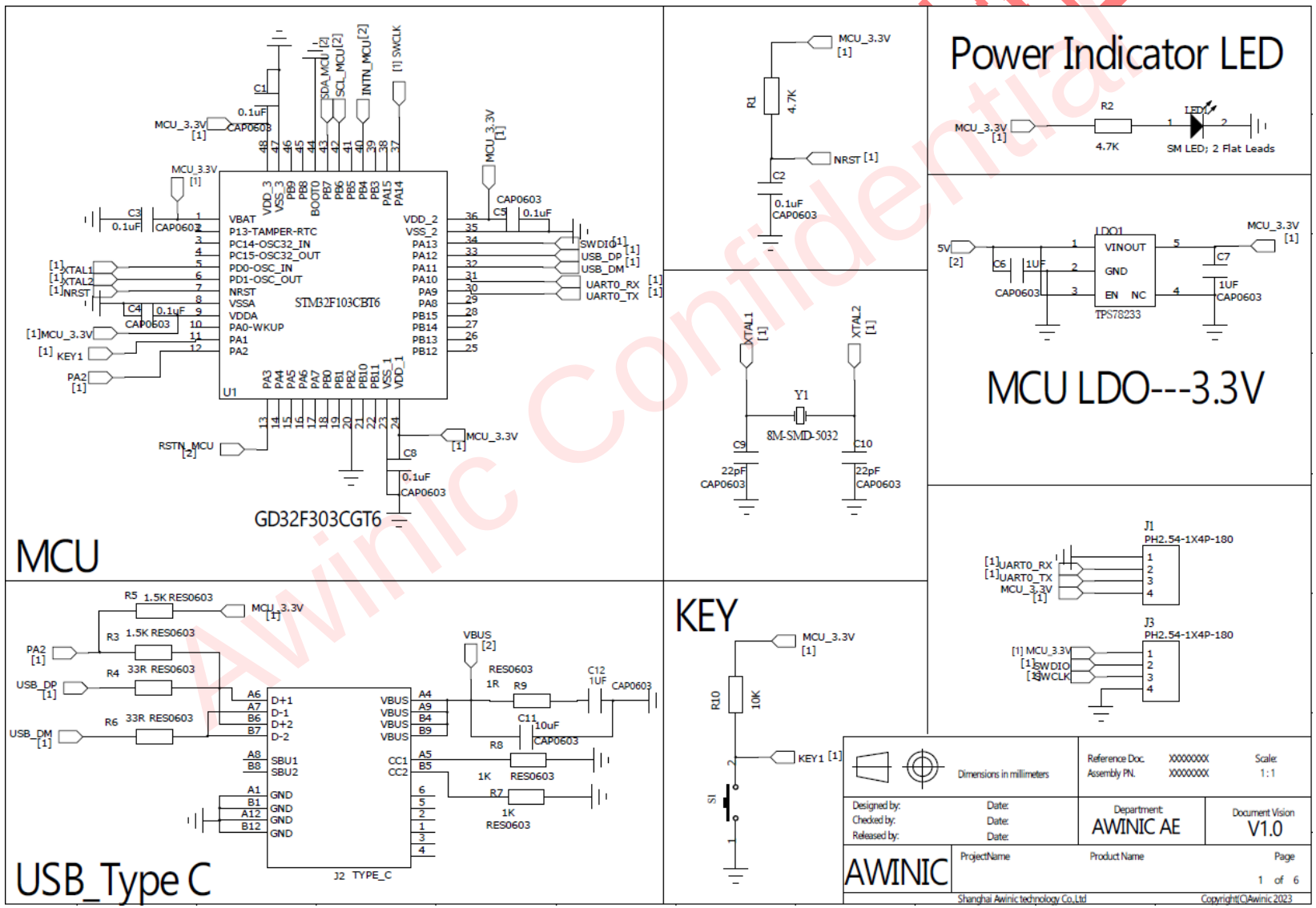
1. 板上有预留灯焊盘的位置（LED0-LED26），使用时将灯焊接上去；

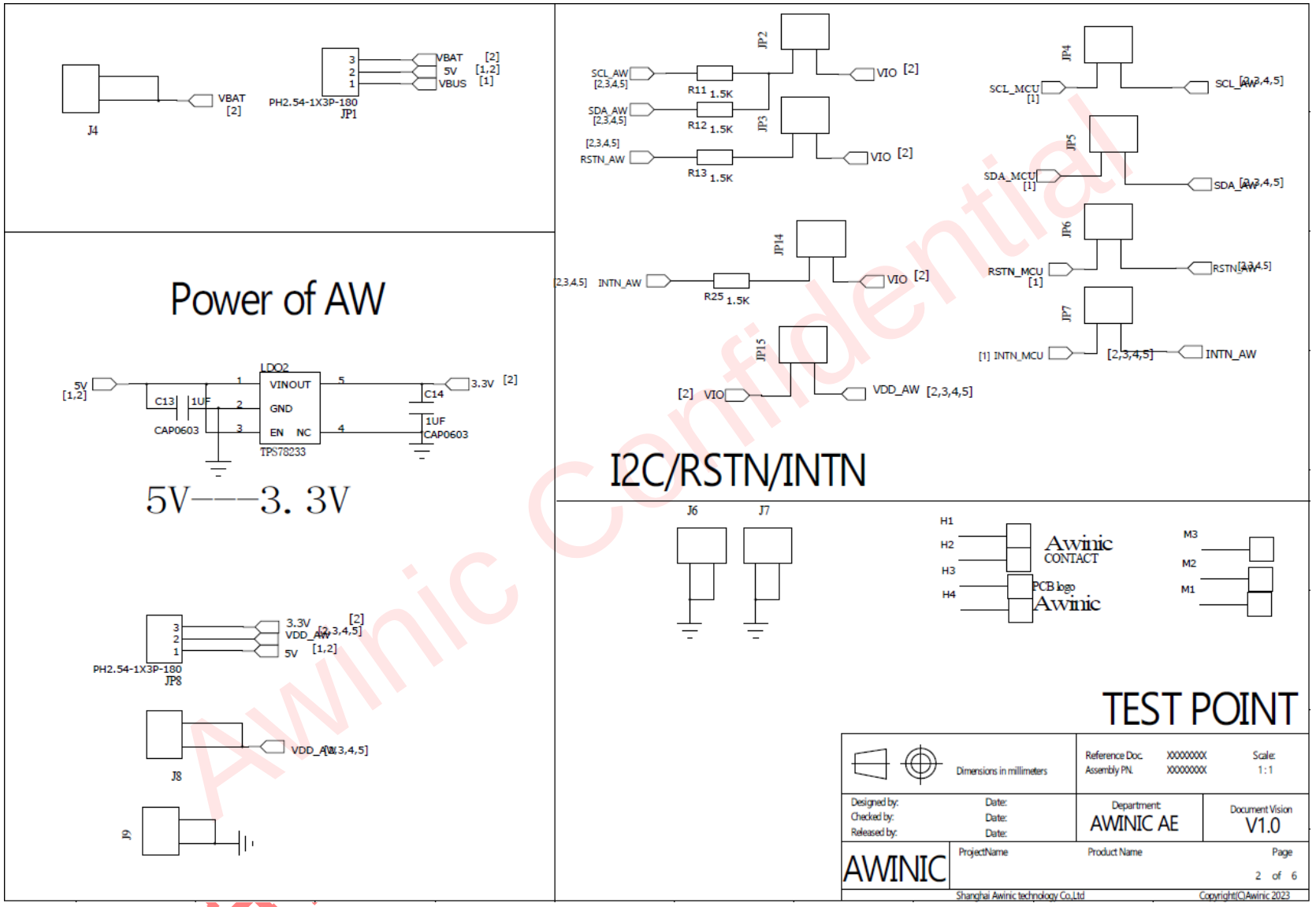
2. 灯阳极先不供电，芯片模式配置完成之后，再对灯的阳极供电。因为芯片默认为输出模式，输出状态和 IIC 地址相关，直接供电有烧灯的风险；

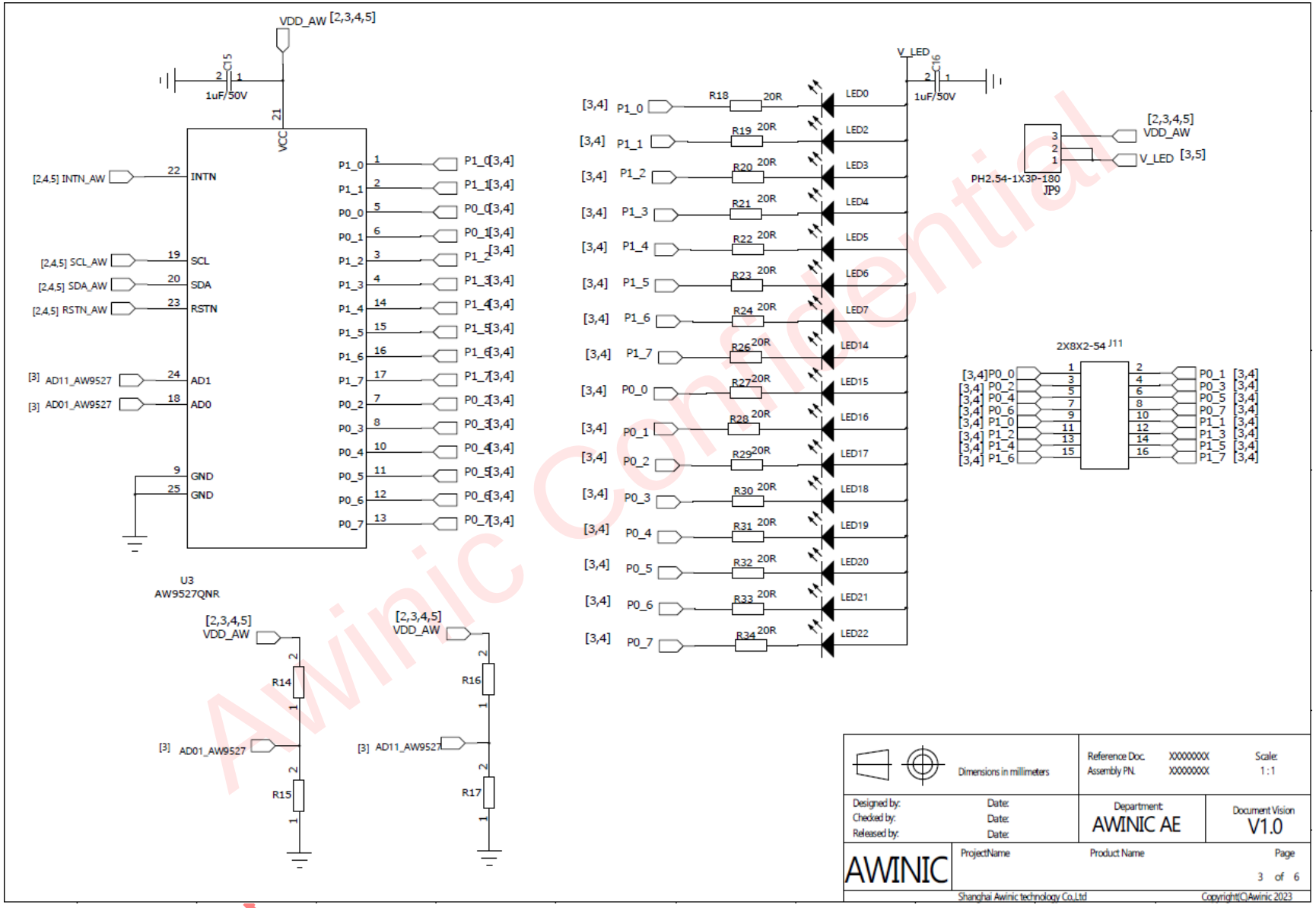
3. 芯片供电 2.5V-5.5V，成功连接 UI，配置为 LED 模式；

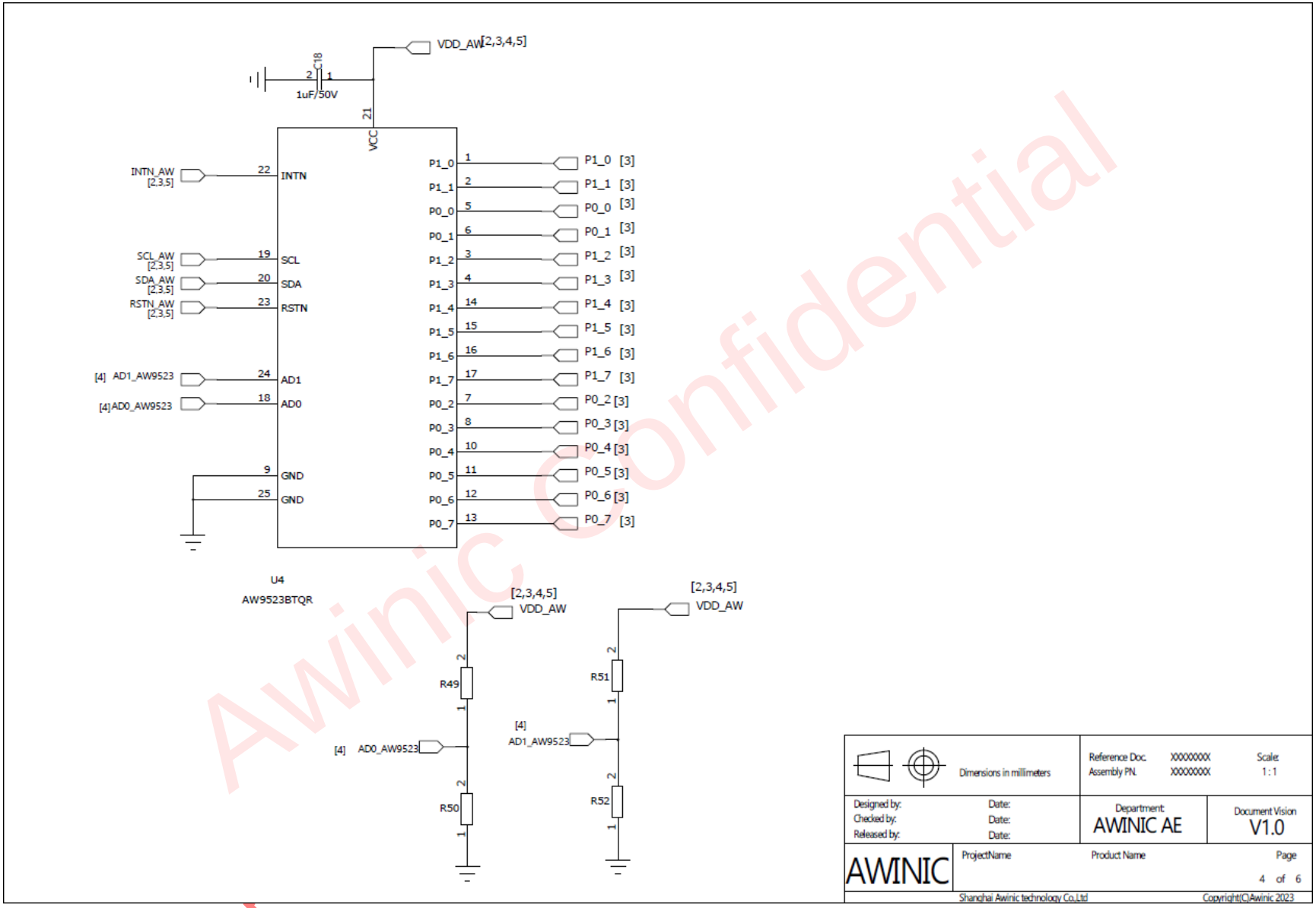
4. 根据需求配置电流和呼吸效果进行调试。

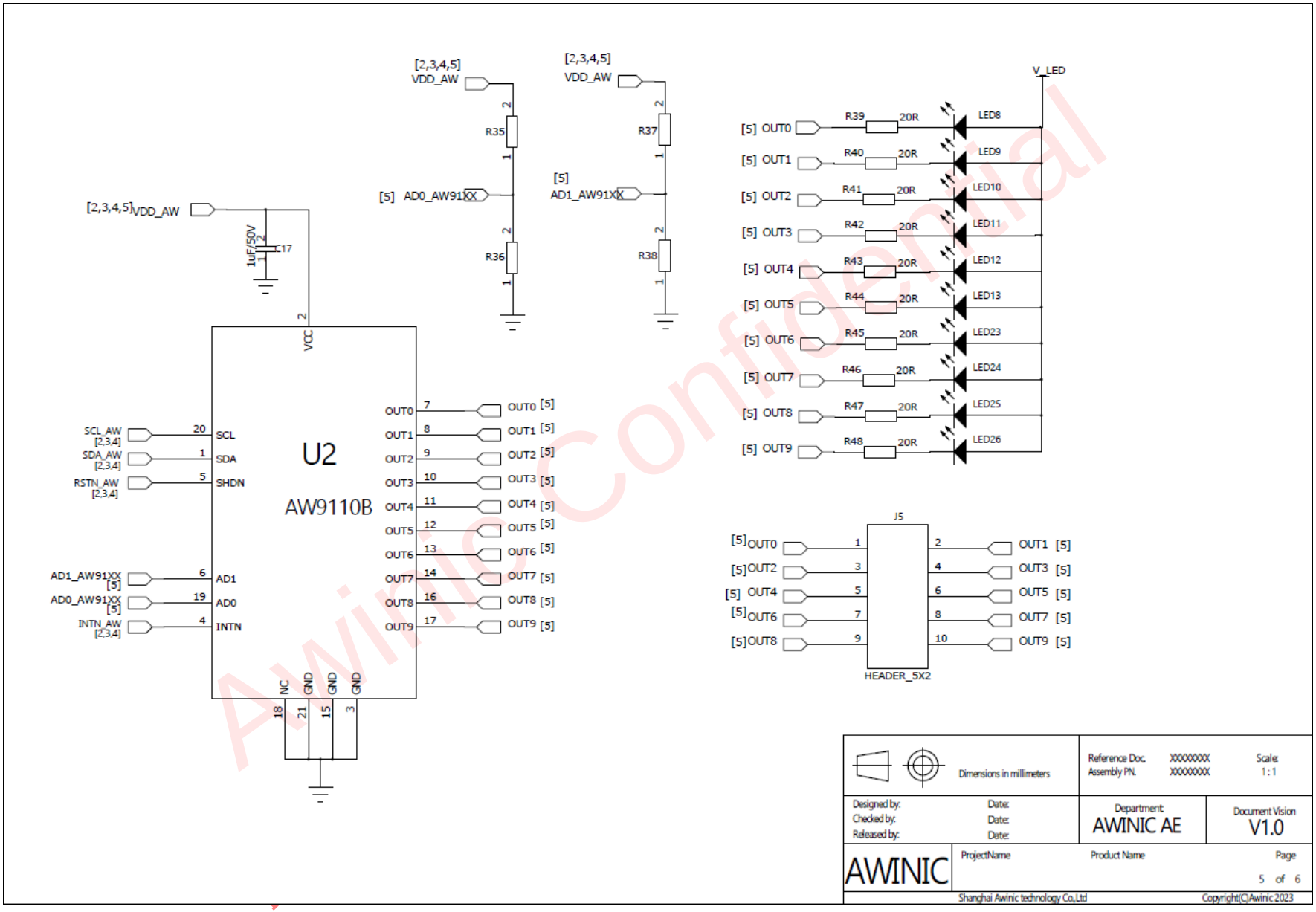
5.1.7 demo原理图



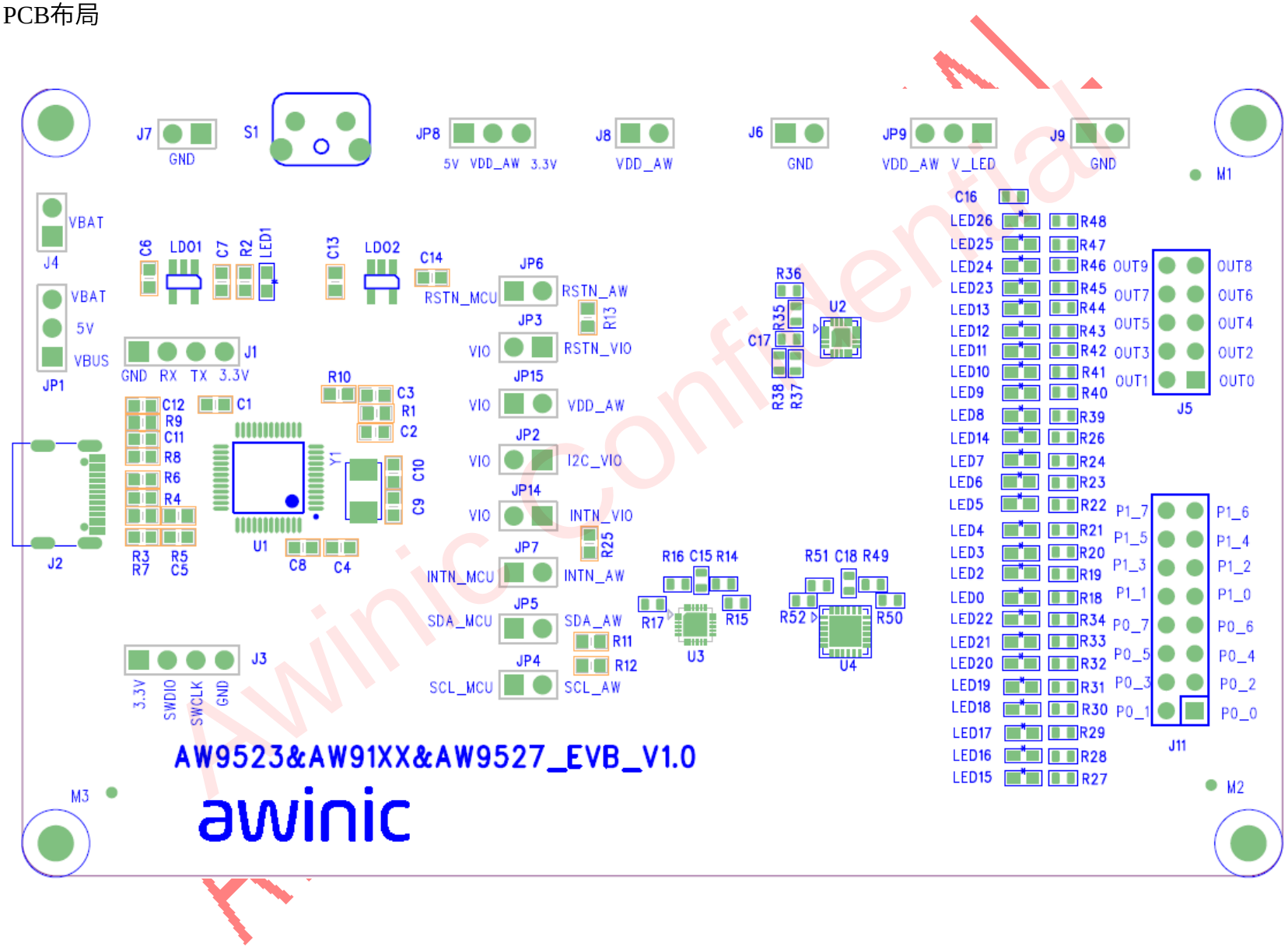


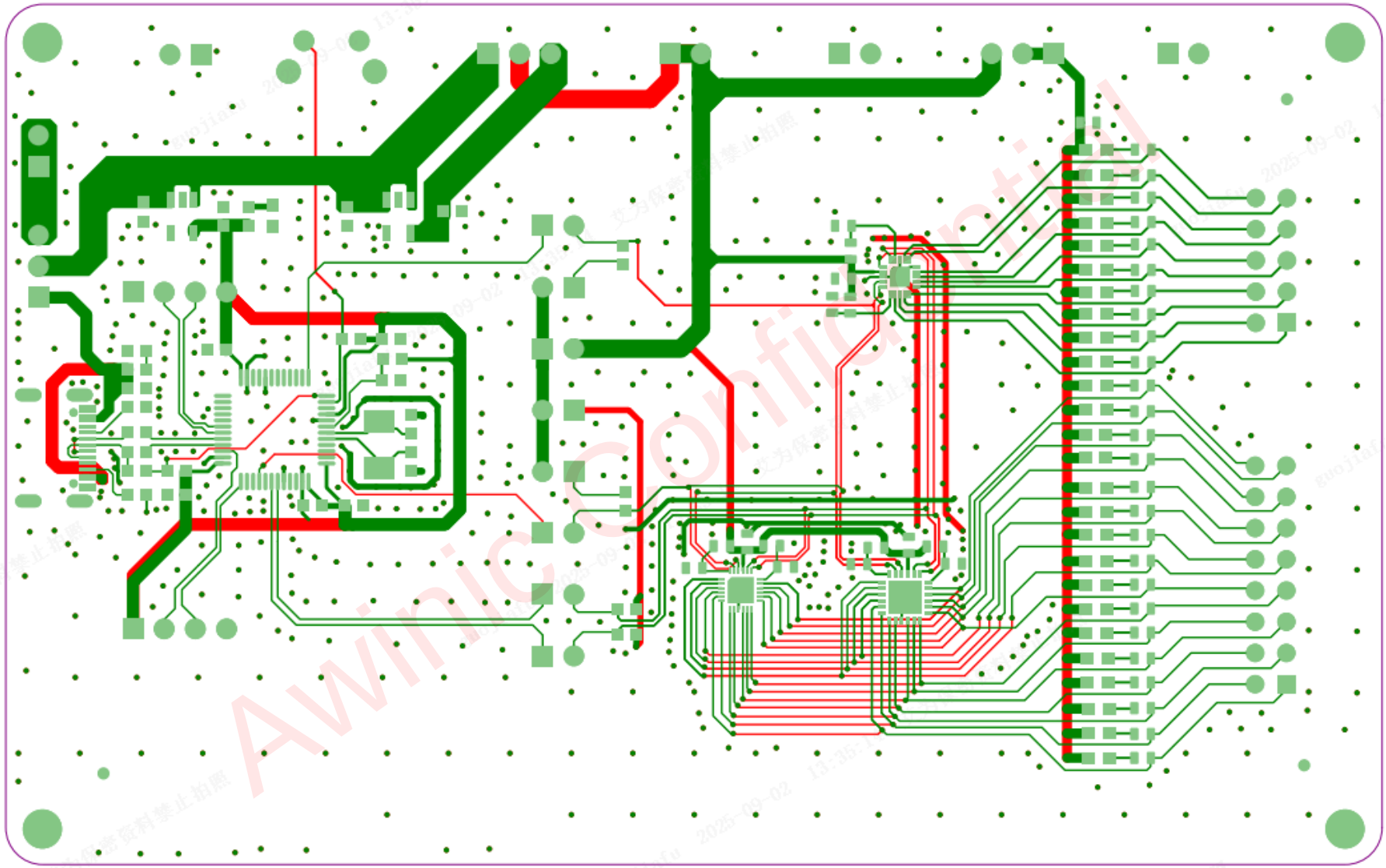






5.1.8 demo PCB布局





5.2 芯片工作模式及配置

5.2.1 AW9523X_AW9527 LED调光

AW9523X_AW9527 系列芯片作为 LED 驱动时，提供 4 个最大电流的等级配置，所有端口都可配置成 LED 驱动模式，每一路端口支持独立控制的 256 级调光，最大电流为 37mA。

配置芯片为 LED 调光模式的步骤如下：

1. 芯片上电，拉高 RSTN 引脚，延时至少 5ms。
2. 根据应用配置对应端口为 LED 工作模式。P0 的 P0_7-P0_0 端口由 12H 寄存器的[Bit7-Bit0]控制，P1 的 P1_7-P1_0 端口由 13H 寄存器的[Bit7-Bit0]控制，当端口对应寄存器的 Bit 配置为 0 时，表示端口工作在 LED 模式。
3. 配置全局的最大输出电流，配置 CTL/GCR（11H）寄存器的 ISEL[Bit1-Bit0]。寄存器的控制逻辑为：当 ISEL[Bit1-Bit0]=00 时，最大调光电流为 Imax；当 ISEL[Bit1-Bit0]=01 时，最大调光电流为 Imax*3/4；当 ISEL[Bit1-Bit0]=10 时，最大调光电流为 Imax*2/4；当 ISEL[Bit1-Bit0]=11 时，最大调光电流为 Imax*1/4。其中 Imax=37mA。
4. 配置端口对应的亮度值。通过配置 DIMx（x=0~15）（20H-2FH）寄存器来调节不同端口的电流值，即改变 LED 的亮度。寄存器与端口的映射关系参考下图：

Address	Name	Description	Default
20H	DIM0	P1_0 port LED current control	00H
21H	DIM1	P1_1 port LED current control	00H
22H	DIM2	P1_2 port LED current control	00H
23H	DIM3	P1_3 port LED current control	00H
24H	DIM4	P0_0 port LED current control	00H
25H	DIM5	P0_1 port LED current control	00H
26H	DIM6	P0_2 port LED current control	00H
27H	DIM7	P0_3 port LED current control	00H
28H	DIM8	P0_4 port LED current control	00H
29H	DIM9	P0_5 port LED current control	00H
2AH	DIM10	P0_6 port LED current control	00H
2BH	DIM11	P0_7 port LED current control	00H
2CH	DIM12	P1_4 port LED current control	00H
2DH	DIM13	P1_5 port LED current control	00H

2EH	DIM14	P1_6 port LED current control	00H
2FH	DIM15	P1_7 port LED current control	00H

5. 如果只需要将某些 LED 关闭，可已将 LED 对应通道的电流值配置为 0。

6. 若整个芯片都不需要工作，可将 RSTN 引脚拉低以降低功耗，再次点亮需要重新执行步骤 1-4。

LED 调光模式软件配置参考代码如下：

```
1.  //Function declaration
2.  void aw_i2c_write_byte(unsigned char reg_addr，unsigned char reg_val);
3.  unsigned char aw_i2c_read_byte(unsigned char reg_addr);
4.  void set_rstn_pin_high(); //pull RSTN pin
5.  /* init chip */
6.  set_rstn_pin_high(); //pull up RSTN pin
7.  delay_ms(10);
8.  aw_i2c_write_byte(0x12，0x00); //set P0 port is LED mode
9.  aw_i2c_write_byte(0x13，0x00); //set P1 port is LED mode
10. val = aw_i2c_read_byte(0x11);
```

```
11. aw_i2c_write_byte(0x11，val&(~0x03)|0x00); //set globe LED Imax is 37mA
12. /* LED brightness control */
13. aw_i2c_write_byte(0x20，0xFF); // Set the LED brightness to port P1_0 to max is 0xFF
14. aw_i2c_write_byte(0x21，0xFF); // Set the LED brightness to port P1_1 to max is 0xFF
15. aw_i2c_write_byte(0x22，0xFF); // Set the LED brightness to port P1_2 to max is 0xFF
16. aw_i2c_write_byte(0x23，0xFF); // Set the LED brightness to port P1_3 to max is 0xFF
17. aw_i2c_write_byte(0x24，0xFF); // Set the LED brightness to port P0_0 to max is 0xFF
18. aw_i2c_write_byte(0x25，0xFF); // Set the LED brightness to port P0_1 to max is 0xFF
19. aw_i2c_write_byte(0x26，0xFF); // Set the LED brightness to port P0_2 to max is 0xFF
20. aw_i2c_write_byte(0x27，0xFF); // Set the LED brightness to port P0_3 to max is 0xFF
21. aw_i2c_write_byte(0x28，0xFF); // Set the LED brightness to port P0_4 to max is 0xFF
22. aw_i2c_write_byte(0x29，0xFF); // Set the LED brightness to port P0_5 to max is 0xFF
23. aw_i2c_write_byte(0x2A，0xFF); // Set the LED brightness to port P0_6 to max is 0xFF
24. aw_i2c_write_byte(0x2B，0xFF); // Set the LED brightness to port P0_7 to max is 0xFF
25. aw_i2c_write_byte(0x2C，0xFF); // Set the LED brightness to port P1_4 to max is 0xFF
```

```
aw_i2c_write_byte(0x2D，0xFF); // Set the LED brightness to port P1_5 to max is 0xFF
aw_i2c_write_byte(0x2E，0xFF); // Set the LED brightness to port P1_6 to max is 0xFF
aw_i2c_write_byte(0x2F，0xFF); // Set the LED brightness to port P1_7 to max is 0xFF
/* if you need to close led，sush as close P1_0 */
aw_i2c_write_byte(0x20，0x00);
```

注意：以上代码部分是伪代码，只用于配置流程作为参考。

5.2.2 AW9523X_AW9527 BLINK模式

AW9527 的端口中 P0_0~P0_1、P1_0~P1_3 支持 BLINK 模式。在 BLINK 模式下，可自动完成周期性的呼吸效果，直到退出 BLINK 模式或关闭呼吸功能。

配置芯片端口工作在 BLINK 模式的流程如下：

1. 芯片上电，拉高 RSTN 引脚，至少延时 5ms。
2. 根据应用设置对应端口为 LED 工作模式。P0_1~P0_0 由 P0WKMD（12H）寄存器的[Bit1-Bit0]控制，P1_3~P1_0 由 P1WKMD（13H）寄存器的[Bit3-Bit0]控制，设置为 0 时，表示对应端口工作在 LED 模式。
3. 设置呼吸端口的电流值为 0，P1_0~P1_3 的电流值分别由 DIM0~DIM3（20H~23H）寄存器控制，P0_0~P0_1 的电流值分别由 DIM4~DIM5（24H~25H）寄存器控制。

4. 根据应用使能端口的呼吸功能，配置 PATEN（14H）寄存器的[Bit5-Bit0]为 1。
5. 根据应用选择端口的工作模式为 BLINK，配置 CONFIG_PORT0（04H）寄存器的[Bit1-Bit0]和 CONFIG_PORT1（05H）寄存器的[Bit3-Bit0]为 1。
6. 设置 LED 的淡入淡出时间，配置 FDTMR（15H）寄存器的[Bit5-Bit0]。
7. 设置 LED 的全亮全暗时间，配置 FLTMR（16H）寄存器的[Bit5-Bit0]。
8. 设置 LED 的最大工作电流，配置 GCR（11H）寄存器的[Bit1-Bit0]。
9. 触发 BLINK 呼吸功能，配置 GCR（11H）寄存器的[Bit7]为 1。
10. 关闭 BLINK 呼吸，需要按顺序执行以下步骤，确保设置 DIMx（x = 0~5）的操作是在关闭呼吸使能的情况下执行。
    a) 先关闭呼吸功能，配置 PATEN（14H）寄存器的[Bit5-Bit0]为 0。
    b) 再将 LED 对应的 DIM 值清 0，配置 DIM0~DIM5（20H~25H）寄存器的值为 0，使芯片退出 BLINK 模式后，LED 处于熄灭的状态。
11. 若整芯片都不需要工作，可将 RSTN 引脚拉低以降低功耗。

注意：AW9527 只允许启用一次 GO 控制位（GCR（11H）寄存器的[Bit7]位）。在重新启用 GO 控制位之前，需要先将芯片复位，并重新配置 BLINK 模式相关寄存器。

BLINK 模式配置参考代码如下：

```
1.  //Function declaration
2.  void aw_i2c_write_byte(unsigned char reg_addr，unsigned char reg_val);
```

```
3.  unsigned char aw_i2c_read_byte(unsigned char reg_addr);
4.  void set_rstn_port_high();
5.  /* init chip */
6.  set_rstn_port_high();
7.  delay_ms(10);
8.  aw_i2c_write_byte(0x12，0x00); //set mode of P0_0-P0_1 is led
9.  aw_i2c_write_byte(0x13，0x00); //set mode of P1_0-P1_3 is led
10. aw_i2c_write_byte(0x11，0x00); //set dim range 0-37mA
11. aw_i2c_write_byte(0x20，0x00); //set brightness of P1_0 is 0x00
12. aw_i2c_write_byte(0x21，0x00); //set brightness of P1_1 is 0x00
13. aw_i2c_write_byte(0x22，0x00); //set brightness of P1_2 is 0x00
14. aw_i2c_write_byte(0x23，0x00); //set brightness of P1_3 is 0x00
15. aw_i2c_write_byte(0x24，0x00); //set brightness of P0_0 is 0x00
16. aw_i2c_write_byte(0x25，0x00); //set brightness of P0_1 is 0x00
17. aw_i2c_write_byte(0x14，0x1F); //enable breath mode
```

```
aw_i2c_write_byte(0x04，0x03); //select P0_0-P0_1 is blink mode
aw_i2c_write_byte(0x05，0x0F); //select P1_0-P1_3 is blink mode
aw_i2c_write_byte(0x15，0x09);
//set fade off/on param of blink mode <00[001][001]> fade on = fade off = 315ms
aw_i2c_write_byte(0x16，0x12);
//set stay off/on param of blink mode <00[010][010]> stay on = stay off = 630ms
val = aw_i2c_read_byte(0x11);
aw_i2c_write_byte(0x11，val|0x01<<7); //start blink
```

注意：以上代码部分是伪代码，只用于配置流程作为参考。

5.2.3 AW9523X_AW9527 SMART-FADE模式

SMART-FADE 模式为半自主呼吸模式，AW9527 芯片端口中 P0_0~P0_1、P1_0~P1_3 支持 SMART-FADE 模式。

SMART-FADE 模式配置流程如下：

1. 芯片上电，拉高 RSTN 引脚，至少延时 5ms。

2. 根据应用设置对应端口为 LED 工作模式。P0_1~P0_0 由 P0WKMD（12H）寄存器的[Bit1-Bit0]控制，P1_3~P1_0 由 P1WKMD（13H）寄存器的[Bit3-Bit0]控制，设置为 0 时，表示对应端口工作在 LED 模式。
3. 根据应用使能端口的呼吸功能，配置 PATEN（14H）寄存器的[Bit5:Bit0]为 1。
4. 选择 LEDx（x = 0~5）的工作模式为 SMART-FADE，配置 CONFIG_PORT0（04H）寄存器的[Bit1-Bit0]和 CONFIG_PORT1（05H）寄存器的[Bit3-Bit0]为 0。
5. 设置 LED x（x = 0~5）的淡入淡出时间，配置 FDTMR（15H）寄存器的[Bit5-Bit0]。
6. 开启 LEDx（x = 0~5）的淡入，配置 OUTPUT_PORT0（02H）寄存器的[Bit1-Bit0]和 OUTPUT_PORT1（03H）寄存器的[Bit3-Bit0]为 1，LEDx（x = 0~5）开始变亮。
7. 等待设定的 LEDx（x = 0~5）变亮的时间。
8. 开启 LEDx（x = 0~5）的淡出，配置 OUTPUT_PORT0（02H）寄存器的[Bit1-Bit0]和 OUTPUT_PORT1（03H）寄存器的[Bit3-Bit0]为 0，LEDx（x = 0~5）开始变暗。
9. 等待设定的 LEDx（x = 0~5）变暗的时间。
10. 如果需要点亮和关闭，重复 6-9。
11. 如果需要修改呼吸参数则需要重新执行步骤 5-9。
12. 如果不需要芯片工作，可拉低 RSTN 引脚以降低功耗。 再次执行 SMART-FADE 需要执行 1-9。

SMART-FADE 模式配置参考代码如下：

```
1.  //Function declaration
2.  void aw_i2c_write_byte(unsigned char reg_addr，unsigned char reg_val);
3.  unsigned char aw_i2c_read_byte(unsigned char reg_addr，unsigned char reg_val);
4.  void set_shdn_port_high();
5.  /* init chip */
6.  set_shdn_port_high();
7.  delay_ms(10);
8.  aw_i2c_write_byte(0x12，0x00); //set mode of P0_0-P0_1 is led
9.  aw_i2c_write_byte(0x13，0x00); //set mode of P1_0-P1_3 is led
10. aw_i2c_write_byte(0x11，0x00); //set dim range 0-37mA
11. aw_i2c_write_byte(0x20，0x00); //set brightness of P1_0 is 0x00
12. aw_i2c_write_byte(0x21，0x00); //set brightness of P1_1 is 0x00
13. aw_i2c_write_byte(0x22，0x00); //set brightness of P1_2 is 0x00
14. aw_i2c_write_byte(0x23，0x00); //set brightness of P1_3 is 0x00
```

```
aw_i2c_write_byte(0x24，0x00); //set brightness of P0_0 is 0x00
aw_i2c_write_byte(0x25，0x00); //set brightness of P0_1 is 0x00
aw_i2c_write_byte(0x14，0x1F); //enable breath mode
aw_i2c_write_byte(0x04，0x00); //select P0_0-P0_1 is smart-fade mode
aw_i2c_write_byte(0x05，0x00); //select P1_0-P1_3 is smart-fade mode
aw_i2c_write_byte(0x15，0x09);
//set fade off/on param of blink mode <00[001][001]> fade on = fade off = 315ms
aw_i2c_write_byte(0x02，0x03); //P0_0-P0_1 start fade on
aw_i2c_write_byte(0x03，0x0F); //P1_0-P1_3 start fade on
aw_i2c_write_byte(0x02，0x00); //P0_0-P0_1 start fade off
aw_i2c_write_byte(0x03，0x00); //P1_0-P1_3 start fade off
```

注意：以上代码部分是伪代码，只用于配置流程作为参考。

5.2.4 AW9523X_AW9527 GPIO功能

AW9523X_AW9527 系列芯片的 16 个端口都可以配置成 GPIO 模式，GPIO 模式下可配置为输入和输出两种方向。端口配置成 GPIO 模式的步骤如下：

1. 芯片上电，拉高 RSTN 引脚，至少延时 5ms。
2. 将 GPIO 输出全部配置为低电平。P0 的 P0_7-P0_0 端口可分别配置 OUTPUT_PORT0（02H）寄存器的[Bit7-Bit0]配置为 0，P1 的 P1_7-P1_0 端口可分别配置 OUTPUT_PORT1（03H）寄存器的[Bit7-Bit0]配置为 0。
3. 配置端口为 GPIO 模式。P0 的 P0_7-P0_0 端口可分别配置 12H 寄存器的[Bit7-Bit0]为 1，P1 的 P1_7-P1_0 端口可分别配置 13H 寄存器的[Bit7-Bit0]为 1。
4. 设置端口为输入或者输出模式。P0 的 P0_7-P0_0 端口模式通过 CONFIG_PORT0（04H）寄存器的[Bit7-Bit0]控制，P1 的 P1_7-P1_0 端口模式通过 CONFIG_PORT1（05H）寄存器的[Bit7-Bit0]控制。当端口对应的 Bit 配置为 0 时表示端口工作在输出模式，当端口对应的 Bit 配置为 1 时表示端口工作在输入模式。
5. 在输出模式下，P0_7-P0_0 端口的输出状态由 OUTPUT_PORT0（02H）寄存器的[Bit7-Bit0]控制，P1_7-P1_0 端口的输出状态由 OUTPUT_PORT1（03H）寄存器的[Bit7-Bit0]控制。当端口对应的 Bit 配置为 0 时表示端口输出高电平，当端口对应的 Bit 配置为 1 时表示端口输出低电平。

   a)、P1 端口设置为输出时，默认为推挽输出，不可再配置。

   b)、P0 端口设置为输出时，默认为开漏。可通过 CTL/GCR（11H）寄存器的[Bit4]来设置成推挽输出或开漏输出。
6. 如果设置为输入模式，P0_7-P0_0 端口的状态可以分别由 INPUT_PORT0（00H）寄存器的[Bit7-Bit0]表示，P1_7-P1_0 端口的状态可以分别由 INPUT_PORT1（01H）寄存器的[Bit7-Bit0]表示。INPUT_PORT0（00H）和 INPUT_PORT1（01H）为只读寄存器，不可以写入。在输入模式的基础上，

可以配置 INT_PORT0（06H）和 INT_PORT1（07H）控制 P0_7-P0_0 和 P1_7-P1_0 端口的输入中断，对应 Bit 为 1 时关闭中断，对应 Bit 为 0 时使能中断。

7. 若整个芯片都不需要工作，可以将 RSTN 引脚拉低以降低功耗。

以 P0 口作为输入，P1 口作为输出。软件配置参考代码如下：

```
1.  //Function declaration
2.  void aw_i2c_write_byte(unsigned char reg_addr，unsigned char reg_val);
3.  unsigned char aw_i2c_read_byte(unsigned char reg_addr);
4.  void set_shdn_port_high();
5.  /* Suppose P0 is set as input and P1 set outputt */
6.  /* chip init */
7.  set_rstn_pin_high(); // pull up RSTN pin
8.  delay_ms(10);
9.  aw_i2c_write_byte(0x02，0x00); //set P0 port output low level
10. aw_i2c_write_byte(0x03，0x00); //set P1 port output low level
11. aw_i2c_write_byte(0x12，0xff); //set P0 port is GPIO mode
```

```
12. aw_i2c_write_byte(0x13，0xff); //set P1 port is GPIO mode
13. aw_i2c_write_byte(0x04，0xff); //set P0 input mode
14. aw_i2c_write_byte(0x06，0x00); //enable P0 interrupt
15. aw_i2c_write_byte(0x05，0x00); //set P1 output mode
16. val = aw_i2c_read_byte(0x11);
17. aw_i2c_write_byte(0x11，val|0x01<<4); //set P0 Push-Pull output mode
18. /* gpio control */
19. /* set P1 all high */
20. aw_i2c_write_byte(0x03，0xff); //set P1 port all high
21. /* get P0 input state */
22. val = aw_i2c_read_byte(0x00);
23. /* set P1_x state */
24. val = aw_i2c_read_byte(0x03);
25. aw_i2c_write_byte(0x03，val&(~0x01<<0)); //set P1_0 low
26. aw_i2c_write_byte(0x03，val&(~0x01<<1)); //set P1_1 low
```

```
/* When using input, pay attention to clear the interrupt after triggering the interrupt, and the clear interrupt must be separated and read 0x00 and 0x01 registers*/
val = aw_i2c_read_byte(0x00);
val = aw_i2c_read_byte(0x01);
```

注意：以上代码部分是伪代码，只用于说明配置流程。

5.2.5 AW91XXX LED调光

AW91XXX 的所有端口都支持 LED 调光模式，每一路端口支持独立控制的 256 级调光。配置芯片为

LED 调光模式的步骤如下：

1. 芯片上电，拉高 SHDN 引脚，至少延时 5mS。
2. 根据应用设置对应端口为 LED 工作模式。AW9110C、AW9110DOUT9-OUT4 由 P0WKMD（12H）寄存器的[Bit5-Bit0]控制，AW9106COUT5-OUT4 由 P0WKMD（12H）寄存器的[Bit1-Bit0]控制， OUT3-OUT0 由 P1WKMD（13H）寄存器的[Bit3-Bit0]控制，设置为 0 时，表示工作在 LED 模式。
3. 如果使用的是 OUT5-OUT0 端口，需要配置 PATEN（14H）寄存器的[Bit5-BitO]为 0，关闭对应端口的 BREATH 功能。
4. 设置 LEDx 的最大工作电流，配置 GCR（11H）寄存器的[Bit1-Bit0]。

5. 根据应用配置 DIMx（AW9110C、AW9110D:x=0~9;AW9106C:x=0~5）（AW9110C、AW9106C：20H~25H）寄存器的值来调节不同 PORT 的电流值，即改变 LEDx 的亮度。AW9110C OUT0-OUT9 的亮度与寄存器 DIMx（x=0~9）（20H~29H）的值依次对应，AW9106C OUT0-OUT5 的亮度与寄存器 DIMx（x=0~5）（20H~25H）的值依次对应。
6. 如果只需要将某些 LED 关闭，可以将对应的电流值设置为 0。
7. 如果整个芯片都不需要工作，为降低功耗，可以将 SHDN 引脚拉低
8. 在 SHDN 拉低后，如需重新调光，重复 1-5 步骤。

LED 调光模式软件配置参考代码如下：

```
1. // Function declaration

2. void aw_i2c_write_byte(unsigned reg_addr，unsigned reg_val);

3. unsigned char aw_i2c_read_byte(unsigned reg_addr);

4. void set_shdn_port_high();

6. /* power on */
```

```
7. set_shdn_port_high();

8. delay_ms(5);

9. /* init chip */

10. aw_i2c_write_byte(0x12，0x00); //set mode of out4-out9 is led

11. aw_i2c_write_byte(0x13，0x00); //set mode of out0-out3 is led

12. aw_i2c_write_byte(0x11，0x00); //set dim range 0-37mA

13. aw_i2c_write_byte(0x14，0x00); //disable out0-out5 breath

15. /* set dim */

16. aw_i2c_write_byte(0x20，0xff); //set brightness of out0 is 255

17. aw_i2c_write_byte(0x21，0xff); //set brightness of out1 is 255

18. aw_i2c_write_byte(0x22，0xff); //set brightness of out2 is 255
```

```
19. aw_i2c_write_byte(0x23，0xff); //set brightness of out3 is 255

20. aw_i2c_write_byte(0x24，0xff); //set brightness of out4 is 255

21. aw_i2c_write_byte(0x25，0xff); //set brightness of out5 is 255

22. aw_i2c_write_byte(0x26，0xff); //set brightness of out6 is 255

23. aw_i2c_write_byte(0x27，0xff); //set brightness of out7 is 255

24. aw_i2c_write_byte(0x28，0xff); //set brightness of out8 is 255

25. aw_i2c_write_byte(0x29，0xff); //set brightness of out9 is 255

27. aw_i2c_write_byte(0x20，0x00); //set brightness of out0 is 0，close led

28. aw_i2c_write_byte(0x23，0x3F); //set brightness of out3 is 0x3F，half of max_brightness
```

5.2.6 AW91XXX BLINK模式

AW91XXX 的端口中 OUTO-OUT5 支持 BLINK 模式。在 BLINK 模式下，可自动完成周期性的呼吸效果，直到退出 BLINK 模式或关闭呼吸功能。配置芯片端口工作在 BLINK 模式的流程如下：

1. 芯片上电，拉高 SHDN 引脚，至少延时 5ms。
2. 根据应用设置对应端口为 LED 工作模式。AW9110C OUT9-OUT4 由 P0WKMD（12H）寄存器的[Bit5-Bit0]控制，AW9106C OUT5-OUT4 由 P0WKMD（12H）寄存器的[Bit1-Bit0]控制， OUT3-OUT0 由 P1WKMD（13H）寄存器的[Bit3-BitO]控制，设置为 0 时，表示对应端口工作在 LED 模式。
3. 设置呼吸端口的电流值为 0，OUT0-OUT5 的电流值分别由 DIM0~DIM5（20H~25H）寄存器控制。
4. 根据应用使能 PORT（OUT0-OUT5）的呼吸功能，配置 PATEN（14H）寄存器的[Bit5-BitO]为 1。
5. 根据应用选择 PORT（OUTO-OUT5）的工作模式为 BLINK，配置 PODIR（O4H）寄存器的[bit1-bitO]和 P1DIR（05H）寄存器的[Bit3-Bit0]为 1。
6. 设置 LED 的淡入淡出时间，配置 FDTMR（15H）寄存器的[Bit5-Bit0]。
7. 设置 LED 的全亮全暗时间，配置 FLTMR（16H）寄存器的[Bit5-Bit0]。
8. 设置 LED 的最大工作电流，配置 GCR（11H）寄存器的[Bit1-Bit0]。
9. 触发 BLINK 呼吸功能，配置 GCR（11H）寄存器的[Bit7]为 1。
10. 关闭 BLINK 呼吸，需要按顺序执行以下步骤，确保设置 DIMx（x=0~5）的操作是在关闭呼吸使能的情况下执行。

    a) 先关闭呼吸功能，配置 PATEN（14H）寄存器的[Bit5-Bit0]为 0。

b）再将 LED 对应的 DIM 值清 0，配置 DIM0~DIM5（20H~25H）寄存器的值为 0，使芯片退出 BLINK 模式后，LED 处于熄灭的状态。

11. 重新触发 BLINK 呼吸功能，需要按顺序执行以下步骤：

a) 先关闭呼吸功能，配置 PATEN（14H）寄存器的[Bit5-Bit0]为 0。

b) 使能呼吸模式，配置 PATEN（14H）寄存器的[Bit5-Bit0]为 1。执行步骤 9，触发 BLINK 呼吸功能。

12. 若需要改变呼吸参数，需要执行步骤 5-9。

13. 若整芯片都不需要工作，可将 SHDN 引脚拉低。重新使能需重新执行 1-9。

注意：AW9110C、AW9106C 只允许启用一次 GO 控制位（GCR（11H）寄存器的[Bit7]位）。在重新启用 GO 控制位之前，需要先将芯片复位，并重新配置 BLINK 模式相关寄存器。

BLINK 模式配置参考代码如下：

```
1. //Function declaration

2. void aw_i2c_write_byte(unsigned reg_addr，unsigned reg_val);

3. unsigned char aw_i2c_read_byte(unsigned reg_addr);

4. void set_shdn_port_high();
```

```
6. /* init chip */

7. set_shdn_port_high();

8. delay_ms(5);

9. aw_i2c_write_byte(0x12，0x00); //set mode of out4-out5 is led

10. aw_i2c_write_byte(0x13，0x00); //set mode of out0-out3 is led

11. aw_i2c_write_byte(0x11，0x00); //set dim range 0-37mA

13. aw_i2c_write_byte(0x20，0x00); //set brightness of out0 is 0x00

14. aw_i2c_write_byte(0x21，0x00); //set brightness of out1 is 0x00

15. aw_i2c_write_byte(0x22，0x00); //set brightness of out2 is 0x00

16. aw_i2c_write_byte(0x23，0x00); //set brightness of out3 is 0x00

17. aw_i2c_write_byte(0x24，0x00); //set brightness of out4 is 0x00
```

```
18. aw_i2c_write_byte(0x25，0x00); //set brightness of out5 is 0x00

20. aw_i2c_write_byte(0x14，0x1F); //enable out0-out5 breath mode

21. aw_i2c_write_byte(0x04，0x03); //select out4-out5 is blink mode

22. aw_i2c_write_byte(0x05，0x0F); //select out0-out3 is blink mode

24. aw_i2c_write_byte(0x15，0x09); //set fade off/on param of blink mode <00[001][001]> fade on = fade off = 315ms

25. aw_i2c_write_byte(0x16，0x12); //set stay off/on param of blink mode <00[010][010]> stay on = stay off = 630ms

27. val = aw_i2c_read_byte(0x11);

28. aw_i2c_write_byte(0x11，val|0x01<<7); //start blink
```

5.2.7 AW91XXX SMART-FADE模式

SMART-FADE 模式为半自主呼吸模式，AW91XXX 芯片端口中 OUT0-OUT5 支持 SMART-FADE 模式。

SMART-FADE 模式配置流程如下：

1. 芯片上电，拉高 SHDN 引脚，至少延时 5ms。
2. 根据应用设置对应端口为 LED 工作模式。AW9110C OUT9-OUT4 由 P0WKMD（12H）寄存器的[Bit5-Bit0]控制，AW9106C OUT5-OUT4 由 POWKMD（12H）寄存器的[Bit1-Bit0]控制，OUT3-OUT0 由 P1WKMD（13H）寄存器的[Bit3-Bit0]控制，设置为 0 时，表示对应端口工作在 LED 模式。
3. 根据应用使能端口 OUT5-OUT0 的呼吸功能，配置 PATEN（14H）寄存器的[Bit5:Bit0]为 1。
4. 选择 LEDx（x=0~5）的工作模式为 SMART-FADE，配置 PODIR（04H）寄存器的[Bit1-BitO]和 P1DIR（05H）寄存器的[Bit3-Bit0]为 0。
5. 设置 LEDx（x=0~5）的淡入淡出时间，配置 FDTMR（15H）寄存器的[Bit5-Bit0]。
6. 开启 LEDx（x=0~5）的淡入，配置 P0DO（02H）寄存器的[Bit1-Bit0]和 P1DO（03H）寄存器的[Bit3-Bit0]为 1，LEDx（x=0~5）开始变亮。
7. 等待设定的 LEDx（x=0~5）变亮的时间。
8. 开启 LEDx（x=0~5）的淡出，配置 P0DO（02H）寄存器的[Bit1-Bit0]和 P1DO（03H）寄存器的[Bit3-Bit0]为 0，LEDx（x=0~5）开始变暗。
9. 等待设定的 LEDx（x=0~5）变暗的时间。
10. 如果需要点亮和关闭，重复 6-9。
11. 如果需要修改呼吸参数则需要重新执行步骤 5-9。
12. 如果不需要芯片工作，可拉低 SHDN 引脚以减低功耗。再次执行 SMART-FADE 需要执行 1-9

SMART-FADE 模式配置参考代码如下：

```
//Function declaration
```

```
2. void aw_i2c_write_byte(unsigned reg_addr，unsigned reg_val);

3. unsigned aw_i2c_read_byte(unsigned reg_addr，unsigned reg_val);

4. void set_shdn_port_high();

6. /* init chip */

7. set_shdn_port_high();

8. delay_ms(10);

9. aw_i2c_write_byte(0x12，0x00); //set mode of out4-out5 is led

10. aw_i2c_write_byte(0x13，0x00); //set mode of out0-out3 is led

11. aw_i2c_write_byte(0x11，0x00); //set dim range 0-37mA

13. aw_i2c_write_byte(0x20，0x00); //set brightness of out0 is 0x00

14. aw_i2c_write_byte(0x21，0x00); //set brightness of out1 is 0x00
```

```
15. aw_i2c_write_byte(0x22，0x00); //set brightness of out2 is 0x00

16. aw_i2c_write_byte(0x23，0x00); //set brightness of out3 is 0x00

17. aw_i2c_write_byte(0x24，0x00); //set brightness of out4 is 0x00

18. aw_i2c_write_byte(0x25，0x00); //set brightness of out5 is 0x00

20. aw_i2c_write_byte(0x14，0x1F); //enable out0-out5 breath mode

22. aw_i2c_write_byte(0x04，0x00); //select out4-out5 is smart-fade mode

23. aw_i2c_write_byte(0x05，0x00); //select out0-out3 is smart-fade mode

24. aw_i2c_write_byte(0x15，0x09); //set fade off/on param of blink mode <00[001][001]> fade on = fade off = 315ms

26. aw_i2c_write_byte(0x02，0x03); //out4-out5 start fade on

27. aw_i2c_write_byte(0x03，0x0F); //out0-out3 start fade on

29. aw_i2c_write_byte(0x02，0x00); //out4-out5 start fade off
```

```
30. aw_i2c_write_byte(0x03，0x00); //out0-out3 start fade off
```

5.2.8 AW91XXX GPIO模式

AW91XXX 芯片的各端口之间功能独立，可根据应用配置对应的端口为 GPIO 模式。端口配置成 GPIO 模式的步骤如下：

1. 芯片上电，拉高 SHDN 引脚，至少延时 5mS。
2. 设置端口为 GPIO 模式。AW9110C OUT9-OUT4 由 P0WKMD（12H）寄存器的[Bit5-Bit0]控制，AW9106COUT5-OUT4 由 P0WKMD（12H）寄存器的[Bit1-Bit0]控制，OUT3-OUT0 由 P1WKMD（13H）寄存器的[Bit3-Bit0]控制，设置为 1 时，表示工作在 GPIO 模式。
3. 设置成输入或者输出模式，AW9110C OUT9-OUT4 由 P0DIR（04H）寄存器的[Bit5-Bit0]控制，AW9106C OUT5-OUT4 由 P0DIR（04H）寄存器的[Bit1-Bit0]控制，OUT3-OUT0 由 P1DIR（05H）寄存器的[Bit3-Bit0]控制，设置为 1 时，表示输入，设置为 0 时，表示输出。
4. 若设置成输出模式，OUT0-OUT3 固定为推挽输出。AW9110C 的 OUT4-OUT9、AW9106C 的 OUT4-OUT5 默认为开漏输出。AW9110C OUT4-OUT9、AW9106C 0UT4-OUT5 可通过配置 GCR（11H）寄存器的[Bit4]修改输出类型。
5. 在输出模式下，AW9110C 可以通过设置 P0DO（02H）寄存器的[Bit5-Bit0]改变 OUT9-OUT4 的输出状态，AW9106C 可以通过设置 P0DO（02H）寄存器的[Bit1-Bit0]改变 OUT5-OUT4 的输出状态。设置 P1DO（03H）寄存器的[Bit3-Bit0]的值改变 OUT0-OUT3 的输出状态。
6. 如设置为输入模式，AW9110C 可通过读取 P0DI（00H）寄存器的[Bit5-Bit0]获取 OUT9-OUT4 的输入状态，AW9106C 可通过读取 P0DI（00H）寄存器的[Bit1-Bit0]获取 OUT5-OUT4 的输入状态，读取 P1DI（01H）寄存器的[Bit3-Bit0]获取 OUT3-OUT0 的输入状态。

7. 在输入模式下,AW9110C 可以通过设置 P0MSK(06H)寄存器的[Bit5-Bit0]配置 OUT9-OUT4 的中断开启与关闭,AW9106C 可以通过设置 P0MSK(06H)寄存器的[Bit1-Bit0]配置 OUT5-OUT4 的中断开启与关闭，P1MSK（07H）寄存器的[Bit3-Bit0]配置 OUT3-OUT0 的中断开启与关闭。
8. 芯片产生中断请求时，中断请求将被保留，直到读取 P0DI（00H）/P1DI（01H）寄存器后，中断请求状态才能被清除。

AW9110C GPIO 模式配置参考代码如下：

```
//Function declaration

2. void aw_i2c_write_byte(unsigned reg_addr，unsigned reg_val);

3. unsigned char aw_i2c_read_byte(unsigned reg_addr);

4. void set_shdn_port_high();

6. /* init chip */

7. set_shdn_port_high(); //set SHDN pin HIGH

8. delay_ms(10);

9. aw_i2c_write_byte(0x12，0x3F); //set mode of out4-out9 is gpio
```

```
10. aw_i2c_write_byte(0x13，0x0F); //set mode of out0-out3 is gpio

12. aw_i2c_write_byte(0x11，0x10); //set out4-out9 PUSH-PULL mode

14. /* set out0-out4 is input */

15. aw_i2c_write_byte(0x05，0x07); //set out0-out3 input

16. val = aw_i2c_read_byte(0x04);

17. aw_i2c_write_byte(0x04，val|0x01<<0); //set out4 input

19. /* set out5-out9 is ouput */

20. val = aw_i2c_read_byte(0x04);

21. aw_i2c_write_byte(0x04，val&(~(0x1F<<1))); //set out5-out9 output

23. /* get input datate */

24. val1 = aw_i2c_read_byte(0x00); //[bit5-bit0] valid -> out9-out4
```

```
25. val2 = aw_i2c_read_byte(0x01); //[bit3-bit0] valid -> out3-out0

27. /* set out5-out9 high */

28. val = aw_i2c_read_byte(0x02);

29. aw_i2c_write_byte(0x02，val|(0x1F<<1)); //set out5-out9 output high

31. /* set out5-out9 low */

32. val = aw_i2c_read_byte(0x02);

33. aw_i2c_write_byte(0x02，val&(~(0x01F<<1))); //set out3 high
```

6. 驱动移植指南

6.1 MCU 软件移植

6.1.1 I2C通信接口配置

根据实际使用的平台实现 I2C 读写接口。

```
static AW_U8 aw_i2c_write_reg(AW_U8 reg_addr, AW_U8 reg_data)

static AW_U8 aw_i2c_read_reg(AW_U8 reg_addr, AW_U8 *reg_data)

static AW_U8 aw_i2c_write(AW_U8 reg_addr, AW_U8 *reg_data, AW_U16 num)

static AW_U8 aw_i2c_read(AW_U8 reg_addr, AW_U8 *reg_data, AW_U16 num)
```

6.1.2 延迟接口配置

根据实际使用的平台实现延时接口。

```
static void delay_ms(AW_U32 ms)
```

6.1.3 使能引脚配置

根据实际使用的平台实现芯片使能接口。

```
static void aw952xx_chip_enable(AW_BOOL enable)
```

6.1.4 中断功能配置

如果使能了aw952xx.h 中定义的宏AW952XX_SINGLE_KEY_MODE_ENABLE 或者AW952XX_MATRIX_KEY_MODE_ENABLE，需要根据实际使用的平台实现中断接口，并在中断接口中调用如下两个函数。

```
aw952xx_enbale_interrupt_by_mask(AW952XX_INPUT_PORT_MASK, 0);

aw952xx_key_work();
```

6.1.5 入口函数移植

在主程序合适的位置调用入口函数。

```
AW_BOOL aw952xx_init(void)
```

6.1.6 驱动功能选择配置_I2C地址配置

在 aw952xx.h 中可以通过修改宏定义 AW952XX_DEVICE_ADDR 直接修改 i2c 地址。

6.1.7 驱动功能选择配置_选择使能具体的GPIO

驱动在 aw952xx.h 中提供两个端口控制的宏，宏的每 bit 位可以单独控制使能一个 gpio 端口。AW9523B_INPUT_PORT_MASK 代表使能为输入端口，AW9523B_OUTPUT_PORT_MASK 代表为使能为输出端口。同一时刻一个端口不能即使能为输入端口，又使能输出端口。

6.1.8 驱动功能选择配置_驱动功能选择

驱动在 aw952xx.h 中提供了 4 个参考功能，包括：

led 模式（AW952XX_LED_MODE_ENABLE）；

gpio 模式（AW952XX_GPIO_MODE_ENABLE）；

单按键模式（AW952XX_SINGLE_KEY_MODE_ENABLE）；

矩阵按键模式（AW952XX_MATRIX_KEY_MODE_ENABLE）。

可以通过开启不同的宏，初始化不同的功能模式。

6.2 QCOM 软件移植

6.2.1 驱动移植

1. 修改dtsi

```
&i2c_3 {

status = "ok";

aw952xx_led@58 {

compatible = "awinic,aw952xx";

reg = <0x58>; //addr

reset-gpio = <&tlmm 20 0>; //rst gpio

irq-gpio = <&tlmm 72 0>; //irq gpio

// aw952xx_power-supply = <&pm8909_l6>; //VCC
```

```
status = "okay";

aw952xx,single_key_enable = <0>; //enable single key function, 0 to disable

aw952xx,matrix_key_enable = <1>; //enable matrix key function, 0 to disable

aw952xx,led_enable = <0>; //enabel led function, 0 to disable

aw952xx,gpio_enable = <0>; //enable gpio function, 0 to disable

aw952xx,key{

aw952xx,wake_up_enable = <0>; //enabel key function in suspend mode,0 to disable

aw952xx,input_port_mask = <0x0F00>; // 0000 1111 0000 0000 Identifies the pin port for the input.

here: P1_0-P1_3

aw952xx,output_port_mask = <0xF000>; //1111 0000 0000 0000 Identifies the pin port for the output

here: P1_4-P1_7
```

```
};

aw952xx,led {

aw952xx,default_imax = <0x00>;

led1{

aw952xx,name = "lcd-backlight"; //cdev name

aw952xx,idx_count = <1>; //The number of ports used

aw952xx,idx = <0x00>; //The specific port identifier used, This is used here: P0_0

aw952xx,default_brightness = <255>;

aw952xx,max_brightness = <255>;

};

led2{
```

```
aw952xx,name = "led2";

aw952xx,idx_count = <1>;////The number of ports used

aw952xx,idx = <0x01>;//The specific port identifier used, This is used here:P0_1

aw952xx,default_brightness = <255>;

aw952xx,max_brightness = <255>;

};

led3{

aw952xx,name = "led3";

aw952xx,idx_count = <1>;

aw952xx,idx = <0x02>; //The specific port identifier used, This is used here:P0_2

aw952xx,default_brightness = <255>;
```

```
aw952xx,max_brightness = <255>;

};

led4{

aw952xx,name = "led4";

aw952xx,idx_count = <1>;

aw952xx,idx = <0x03>; //The specific port identifier used, This is used here:P0_3

aw952xx,default_brightness = <255>;

aw952xx,max_brightness = <255>;

};

};

aw952xx,gpio{
```

```
aw952xx,gpio_mode = <1>;

gpio1{

aw952xx,gpio_idx = <4>; // The specific port identifier used, This is used here:P0_4

aw952xx,gpio_dir = <1>; // The specific port work in output(1) or input(0)

aw952xx,gpio_default_val = <0>;

};

gpio2{

aw952xx,gpio_idx = <5>; // The specific port identifier used, This is used here:P0_5

aw952xx,gpio_dir = <1>; //The specific port work in output(1) or input(0)

aw952xx,gpio_default_val = <0>;

};
```

```
gpio3{

aw952xx,gpio_idx = <6>; // The specific port identifier used, This is used here:P0_6

aw952xx,gpio_dir = <1>; //The specific port work in output(1) or input(0)

aw952xx,gpio_default_val = <0>;

};

gpio4{

aw952xx,gpio_idx = <7>; // The specific port identifier used, This is used here:P0_7

aw952xx,gpio_dir = <1>; //The specific port work in output(1) or input(0)

aw952xx,gpio_default_val = <0>;

};

};
```

```
};

};
```

在aw952xx.dtsi 中配置模式（ single_key_enable 、matrix_key_enable 、led_enable 和gpio_enable），同时可分别配置他们的port 模式和输入输出状态。测试的AW952XX 的demo，当用作按键时，其输入引脚需要接外部上拉，默认为高电平，当按键按下时，output 会将input 拉低，IC 此时会产生中断信号。

注：该调试平台默认已经配置好 IRQ 使用 GPIO72 和 rst 使用 GPIO20 两个引脚，中断需要默认拉高，如果未配置，则需要先在平台的 dtsi 中配置。此外，用户可通过配置 wake_up_enable 的初始值来决定 suspend mode 下按键扫描功能是否开启，置 1 时休眠模式下按键功能可正常使用，若置 0 休眠模式下按键功能将会被禁用。

2 驱动配置

（1）驱动移植

将驱动包中所有驱动文件移植到drivers/input/misc/aw952xx/目录下面。

（2）内核编译配置

a. defconfig编译配置

在defconfig 中添加CONFIG_AW952XX=y

b. Kconfig配置

在drivers/input/misc/Kconfig 中添加source ” drivers/input/misc/ aw952xx /Kconfig”

c. Makefile配置

在 drivers/input/misc/Makefile 中添加 obj-$(CONFIG_ AW952XX) += aw952xx /

6.2.2 调试接口

AW952XX Driver 创建的节点文件路径为：sys/bus/i2c/drivers/ aw952xx /*-00xx，其中*为i2c，bus number，xx 为i2c address。

（1）reg

功能描述：用于读写aw952xx 的所有寄存器

使用方法：

读寄存器值：cat reg

写寄存器值：echo reg_addr reg_data > reg （16 进制操作，reg_addr和reg_data 均为单字节）

参考例程：

cat reg （获取所有可读寄存器上的值）

echo 0x1a 0x88> reg （向0x1a 寄存器写入0x88 的值）

(2) sys/class/leds/<led-name>/brightness

功能描述：Led 模式，调节led 的亮度，写入值的范围0-255

使用方法：echo 10 > brightness

(3) aw952xx_gpio

功能描述：Gpio 模式时，查询gpio 的端口状态

使用方法：cat aw952xx_gpio

(4) imax

功能描述：修改led 驱动可调节的最大电流值

其中imax_val 的值为0x00-0x03，值的意义如下:

0x00 0-Imax

0x01 0-3/4*Imax

0x02 0-2/4*Imax

0x03 0-1/4*Imax

Imax的典型值为37mA

使用方法：echo imax_val > imax

7. 注意事项

7.1 上电默认状态

AW952X 根据 AD1、AD0 引脚的不同连接属性，16 路输出也有着不同的初始状态：

AD1	AD0	P1_7	P1_6	P1_5	P1_4	P1_3	P1_2	P1_1	P1_0	P0_7	P0_6	P0_5	P0_4	P0_3	P0_2	P0_1	P0_0
GND	GND	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GND	VBAT	0	0	0	0	1	1	1	1	0	0	0	0	Hi-Z	Hi-Z	Hi-Z	Hi-Z
VBAT	GND	1	1	1	1	0	0	0	0	Hi-Z	Hi-Z	Hi-Z	Hi-Z	0	0	0	0
VBAT	VBAT	1	1	1	1	1	1	1	1	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z

注意：默认状态为 Hi-Z 的端口不能作为 GPIO 输入使用

7.2 GPIO 模式应用说明

1. 芯片默认为 HiZ 的端口不可以直接用作 GPIO 使用，需要在该端口增加弱上拉或弱下拉，以保证该端口在初始化之后为稳定状态。
2. 在使用中断功能时，INTN 引脚内部为开漏架构，因此外部需要通过 KΩ级上拉电阻上拉。一旦产生中断，需要对每个端口都进行读寄存器清中断操作，即对 INPUT_PORT0（00H）和 INPUT_PORT1（01H）寄存器分别进行读操作。

7.3 LED 模式应用说明

使用 AW9527 配置端口作为 LED 模式时，配置呼吸参数（对应 FDTMR（15H）、FLTMR（16H）寄存器）时，需要注意 fade-on、fade-off、all-on、all-off 这四个呼吸参数 3-bit 仅可以配置为 001、010、011、100、101 对应关系如下：

3bit 呼吸参数	呼吸时间
001	315ms
010	630ms
011	1260ms
100	2520ms
101	5040ms

8. 常见问题

8.1 Q：AW9523B 在 AD0 和 AD1 拉高时，将 P0 输出配置为推挽输出，P0_0 会拉高，为什么?

A：AW9523B 默认输出状态由寄存器 02，03 决定，这两个寄存器的默认值由 AD0 和 AD1 的接法决定，在默认状态为 Hi-Z 时，可以理解为开漏的高，所以在 AD0 和 AD1 拉高的时候默认值为 FF，所以配到推挽的时候会直接拉高。

8.2 Q：AW9523b 的驱动代码支持哪些应用?

A：独立按键、矩阵按键、呼吸灯。

8.3 Q：AW9523b 的 GPIO 模式支持内部上拉或者下拉吗?

A：不支持。

8.4 Q：AW95016 模式支持内部上拉或者下拉吗?

A：支持

8.5 Q：AW95016 的锁存功能是什么?

A：中断触发之后，如果开启了锁存功能，则会保存当前触发中断的状态，直到读清该状态。

8.6 Q：AW9523B/AW95016 矩阵键盘的常规设计，最多支持几个按键组合?

A：2 个。

8.7 Q：AW9523B 是否可以做自主呼吸?

A：AW9523B 不支持自主呼吸功能，如需要实现呼吸功能，需要主控通过 I2C 不停的对芯片寄存器进行写操作。 其中 DIMx 寄存器为亮度控制寄存器，在对应 P0x 或 P1x 为 LED 模式下，对 DIMx 写 00H，灯为灭状态，对 DIMx 写 FFH，灯为最亮状态，实现呼吸功能时，则需要把 00H~FFH 根据呼吸时间写入 DIMx 中即可。

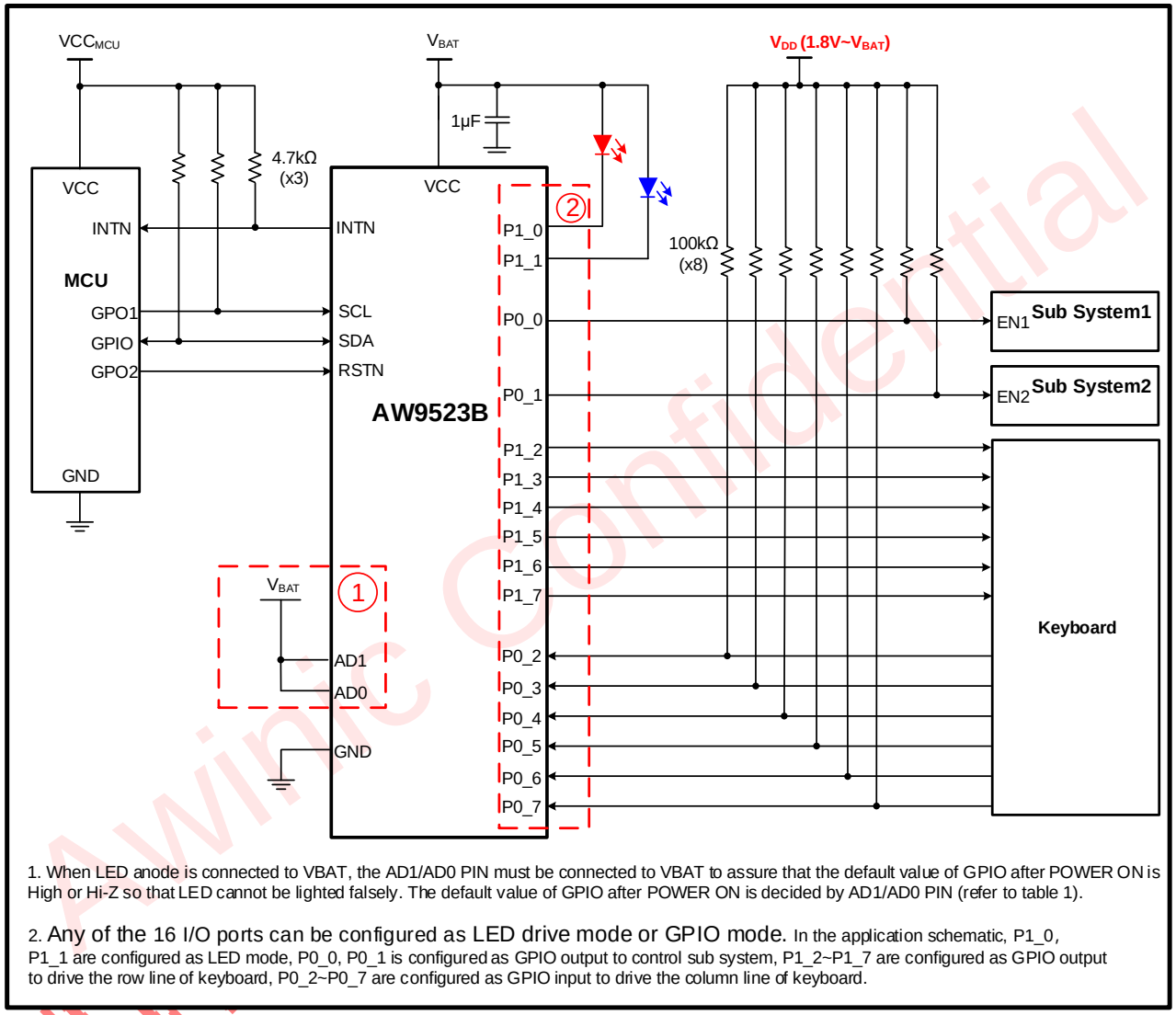
8.8 Q：AW9106C 上电后，为什么灯全部是亮的状态?

A：AW9106C 上电后，所有引脚状态和 AD0、AD1 接法相关，具体如下表， 在 LED 应用时，AD0、AD1 都为 GND 时，芯片上电后所有引脚输出低电平，而芯片 OUTx 连接的是 LED 阴极，导致 LED 全亮，甚至有烧毁 LED 风险，因此在 LED 应用时，建议 AD0、AD1 连接至高电平，芯片上电后所有引脚为高阻或输出高电平。

AD1	AD0	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
GND	GND	0	0	0	0	0	0
GND	VCC	Hi-Z	Hi-Z	1	1	1	1
VCC	GND	0	0	0	0	0	0
VCC	VCC	Hi-Z	Hi-Z	1	1	1	1

8.9 Q：AW9523B 是否支持矩阵式键盘?

A：支持。 芯片配置到 GPIO 模式，P0 和 P1 部分 GPIO 可分别配置为输入和输出模式。当有按键按下时芯片会产生一个中断拉低 P0，然后轮询去拉高 P1 的状态，这样就可以通过判断 P0 那边 pin 脚的状态改变来判断哪个按键按下了。



1. When LED anode is connected to VBAT, the AD1/AD0 PIN must be connected to VBAT to assure that the default value of GPIO after POWER ON is High or Hi-Z so that LED cannot be lighted falsely. The default value of GPIO after POWER ON is decided by AD1/AD0 PIN (refer to table 1).

2. Any of the 16 I/O ports can be configured as LED drive mode or GPIO mode. In the application schematic, P1_0, P1_1 are configured as LED mode, P0_0, P0_1 is configured as GPIO output to control sub system, P1_2~P1_7 are configured as GPIO output to drive the row line of keyboard, P0_2~P0_7 are configured as GPIO input to drive the column line of keyboard.

8.10 Q：AW9523B 如何识别多按键触发?

A：用定时线程轮询的方式来做，当一个按键按下时，启动线程一直轮询监控是否有第二个按键按下，直到所有按键释放才取消线程。

8.11 Q：使用 2 个 AW9523B 如何连接?

A：AW9523B 的 I2C 地址由 AD1/AD0 决定，所以有两种接法：I2C 地址相同时，需要用两组 I2C，RSTN 可共用，也可以分开。I2C 地址不同时，可共用一组 I2C，RSTN 可共用，也可以分开。

8.12 Q：AW9523B P0 和 P1 有什么区别？AW9523B P0 和 P1 默认状态如何设置?

A：GPIO 模式下 P0 可以做开漏输出或者推挽输入输出，P1 只能做推挽输入输出。建议 P0 接上拉电阻。设置细节请参考规格书中详细描述，不同设定默认状态有不同，使用时需要重点核对。

8.13 Q：AW9523B 用作 GPIO 口时需要注意什么?

A：

1. P1 口可配置为输入或输出，推挽模式。P0 口可配置为输入或输出，推挽模式或开漏模式。P0/P1 在上电后默认状态和 AD0/AD1 接法有关，可参考备注中图表。

2. GPIO 模式下每个 I/O 口最大带载能力为 20mA。

3. 可通过 0x04 寄存器设置 P0_0 到 P0_7 作为输入或者输出模式；0x05 寄存器设置 P1_0 到 P1_7 作为输入输出模式。

8.14 Q：AW9523B 用作键盘输入时，需要注意什么？

A：

1. AW9623B 用作键盘时，应用按键数最多不能超过 64 个 KEY（8x8 矩阵）。

2. P1_0～P1_7 设定成输出，对按键逐行扫描；P0_0～P0_7 设定成输入，对按键逐列扫描，同时检测列状态。

3. AW9523B 的中断输出和 I2C 总线需要外接 4.7KΩ 上拉电阻，上拉电平大小和主控接口电平一致。

8.15 Q：AW9523B GPIO 口可以任意配置吗？输入功能、输出功能和按键应用。

A：

1. 可以单独配置每个 I/O 口的输入/输出功能。

2. 按键应用时需要通过键盘扫描方式，应用按键数最多不能超过 64 个 KEY。可根据需要的按键数调整 I/O 口数量。

3. 按键应用时建议把 P0 对应按键的端口设置为输入，把 P1 对应按键的端口设置为输出。

8.16 Q：AW9523B 供电电压范围是多少?

A：AW9523B 供电电压范围是 2.5V~5.5V。如果做键盘扩展或 I/O 扩展，P0 端口上拉需要与平台的 I/O 电平一致。如果驱动 LED，LED 阳极直接接到电池 VBAT 上，AD1/AD0 PIN 也连接到 VBAT，以确保 POWER ON 后的 GPIO 默认值为 High 或 Hi-Z，LED 不会误亮。

8.17 Q：AW9523B P0 无法输出高电平，如何解决?

A：AW9523B，P0 默认为开漏模式，P1 默认为推挽模式。P1 可以正常输出高电平，但 P0 无法输出高电平，需要外部上拉才能输出高电平。此外，软件配置 0x11 寄存器，配置成 0x10，使 P0 为推挽模式，也可以实现输出高电平。

8.18 Q：AW9523B、AW9106C、AW9110C 驱动 LED 灯，需要注意哪些?

A：

1. 芯片 P0/P1 或 OUT 上电后默认状态，和 AD1/AD0 有关，要确保 POWER ON 后，LED 不会误亮。LED 阴极接 P0/P1 或 OUT，默认状态必须是 High 或 Hi-Z，LED 阳极接 P0/P1 或 OUT，默认状态必须是 Low 或 Hi-Z。细节可以查阅规格书
2. LED 模式时，LED 阳极接电源，阴极接芯片 P0/P1 或 OUT，亮度可调。

3. GPIO 模式，只有亮和灭两种状态，亮度不可调。接法：①LED 阳极接电源，阴极接芯片 P0/P1 或 OUT，电流不能超过 20mA，电路上需要增加限流电阻。另外，注意电源电量降低时亮度也会变化。②LED 阳极接芯片，阴极接地，电流不能超过 20mA，电路上需要增加限流电阻。P0/P1 或 OUT 电压随 VCC 电压变化，当 VCC 接电源，电源电量降低的时 P0/P1 或 OUT 电压也会降低，这个时候亮度也会变化。

8.19 Q：AW9106C/AW9110C/AW9523B 分别有几个通道可以自主呼吸？

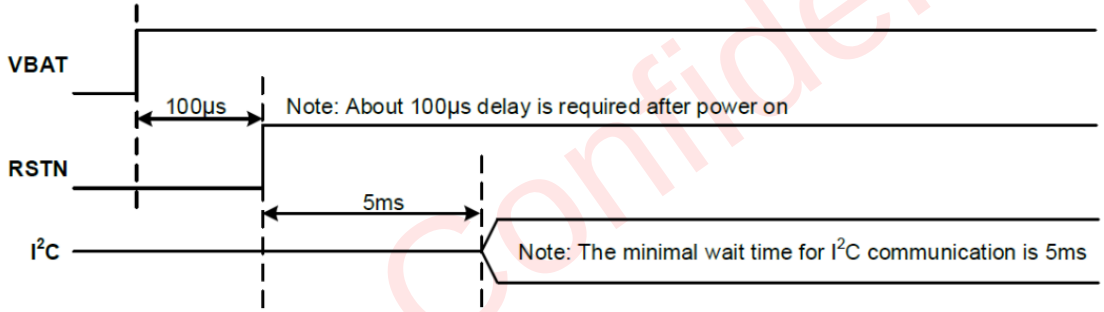
A：AW9106C 一共有 6 个通道，6 个通道都可以自主呼吸；AW9110C 一共有 10 个通道，其中只有 6 个通道（OUT0-OUT5）可以自主呼吸；AW9523B 一共有 16 个通道，所有通道均不能自主呼吸。

8.20 Q：AW9106C、AW9110C、AW9523B 的 INTN 引脚要怎么接？

A：INTN 引脚的功能：当通道配置为 GPIO 口输入模式时，端口检测到输入电平变化后会有中断信号产生。主控识别到中断信号后可以去读 IC 的 REG00H 和 REG01H 寄存器获得端口的电平状态；也可以不采用此方式，而采取定时轮询的方式来读。所以中断引脚可以使用，也可以不用。对于实时性要求比较高的应用场景，如矩阵键盘，建议要使用 INTN 引脚；对于实时性要求比较低的应用场景，如常规的 GPIO 口扩展，不用 INTN 引脚也是可以的。如果不用 INTN 引脚，悬空或者接地；如果使用 INTN 引脚，需通过上拉电阻上拉至 VIO，并连接至主控的 GPIO 口。

8.21 Q：AW9106C、AW9110C、AW9523B 的 SHDN（RSTN）引脚要怎么接呢?

A：上电时需要满足以下上电时序，否则芯片可能不能正常初始化所以有两种接法：接至主控的 GPIO 口，或通过 RC 延时电路接到高电平。优先建议接至主控的 GPIO 口，不建议直接接高，因为接至主控的 GPIO 口更容易控制上电时序，而且保留硬复位功能，让整个系统更加可靠如果 GPIO 口资源非常紧张，也可以将高电平通过 RC 延时电路接至 RSTN 引脚，但需实测上电波形，保证满足上图中的上电时序



8.22 Q：AW9106C、AW9110C、AW9523B 的中断引脚什么情况下会产生中断？中断信号是什么？中断信号是否需要释放？如何释放?

A: AW9106C、AW9110C、AW9523B 的中断引脚只有一种情况下会产生中断：端口配置为输入模式，并且对应端口的中断使能位打开（REG06H、REG07H），一旦监测到端口的电平状态发生改变（高电平变为低电平、低电平变为高电平，8us deglitch），就会产生中断。表现为 INTN 引脚拉低。产生中断信号后，INTN 引脚会被持续拉低，除非通过一定的操作才能释放中断信号。INTN 信号释放方法：主控通过 IIC 分别读 REG00H 和 REG01H 寄存器（不能连读）可以释放中断。

8.23 Q：AW9106C、AW9110C、AW9523B 作为 LED 驱动时，芯片 VCC 和灯的阳极可以分开供电吗？

A：首先 VCC 和 LED 阳极电源的大小主要受限于 VCC 引脚和各端口的耐压，建议最大不超过 5.5V。在 LED 应用中，芯片 VCC 与灯阳极可为同一电源，也可分开独立供电，但要注意两点: （1）灯阳极的电压不能大于芯片 VCC 电压； （2）上电时序要求：VCC 上电不能晚于 LED 阳极电源上电； 同时，还要注意，当 AW9106B、AW9110B、AW9523B 的全部端口均作为 LED 驱动时，AD0AD1 引脚必须均接高电平，以确保上电后各端口的默认状态为高阻或高电平，保证灯不会误亮。

8.24 Q：AW9523B 的 POR 的电压是多少？

A：AW9523B 的 POR 电压为 2V。

8.25 Q：AW9110C GPIO 作为输出，最大电流多少？

A：作为输出，需要配置成推挽模式， 芯片路径上电流可以到 60mA，也即带载不能超过 60mA。

8.26 Q：AW9523B 作为 LED 驱动应用时，P0/P1 配置 LED 模式， 连接 LED 阴极，LED 阳极接 VCC 也可以独立供电；LED 阳极电压如何计算？

A：VLED（阳极）= VF+R*I+0.4（恒流源正常工作压降）+ 0.2V（走线等线损预留值）。

8.27 Q：用 9523B 驱动 LED，IO 口接 LED 阳极可以吗？不行的原因?

A:

1. AW9523B 的 IO 做阳极，无法控制输出的电压高低，只能默认输出 VCC 给过来的电压，若 VCC 电压过高或 VCC 电压有波动，有烧灯风险。
2. 若 AW9523B 的 IO 做阳极，IO 需工作在推挽模式高电平，此时无法控制输出电流，即无法控制灯亮度，只能控制亮灭。且电流能力未知（推挽模式估计只有几 mA 能力）
3. AW9523B 的 IO 做阳极，各引脚之间的电流能力可能有差异导致灯亮度不均匀。推挽模式的电流能力精度我们不做卡控，常规应用场景也无需卡控。

8.28 Q：AW9523b/AW95016 外部 GPIO 电平是否可以比 VCC 高?

A：不建议，会往 VCC 漏电。

8.29 Q：AW9523B，AD 口都拉低，RSTN 也拉低，IO 低电平的内部架构是怎样的呢?

A：内部开漏架构，是由 NMOS 连接到地。

8.30 Q：当 AW95016 VDD 还有供电，RESET 拉低后，GPIO 口的状态如何？等到 VDD 再掉电后,GPIO 状态又如何?

A：RESET 拉低后(VDD 供电正常时)，芯片的 IO 口恢复到默认状态；随着 VDD 再掉电后（电压还没调到 BOR，AW95016A 的 BOR 与 POR 的电压相等），芯片的 IO 口保持在默认状态；VDD 芯片低于 BOR 时候，此时芯片状态不可控。

8.31 Q：AW95016 芯片的 BOR 值是多少?

A：芯片的 BOR 模块和 POR 模块是相同模块，因此 AW95016A 的 BOR 与 POR 的电压相等。

8.32 Q：AW9523B 的 IO 口的上拉电平需要满足什么要求?

A：

1. AW9523B 的 IO 口作为输入引脚：上拉电源需要不超过 VDD 的电平；
2. AW9523B 的 IO 口作为 OD 输出引脚：上拉电源需要不超过 VDD 的电平；
3. AW9523B 的 IO 口作为 PP 输出引脚：上拉电源需要不超过 VDD 的电平。

8.33 Q：AW95124FOR 的 I2C 上拉电平与 VDD_I2C 关系?

A：AW95124FOR 的 I2C 上拉电平与 VDD_I2C 相等，如果 I2C 上拉电平低于 VDD_I2C，I2C 通讯可能会无法通讯。

9. 版本记录

版本	日期	说明
V1.0	2026-06	初版

免责声明

此文档中包含的信息被认为是准确、可靠的。但是，上海艾为电子技术股份有限公司（以下简称艾为）对这些信息的准确性或完整性均不作任何明示或暗示的陈述或保证，且对这些信息的使用后果不承担任何责任。

艾为保留在任何时间、没有任何通报的前提下修改本文档中发布的信息，包括但不限于产品资料和规格的权利。客户在下订单前应自行获取最新的相关信息，并验证这些信息是最新且完整的。本文档信息覆盖并取代所有先于此次公布的文档。

此文档为产品应用的参考文档，请根据实际项目应用进行修改，不保证没有瑕疵且不做任何担保。

对于艾为产品技术文档的信息，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况下才允许进行复制。艾为对篡改过的文件不承担任何责任或义务。复制第三方的信息可能需要遵守额外的限制条件。