版本历史

历史版本	修改内容	版本日期
V0.1	初始版本	2019-03-28
V0.2	删除 MCU 频率限制说明以及 LED 限制说明	2019-09-06
V0.3	补充章节内容	2019-09-29
V0.4	完善心率测量章节&补充说明章节	2020-01-04
V0.5	补充心率测量标准	2020-01-04
V0.6	更改格式	2022-07-06

目 录

1 产品概述

1.1 芯片简介

CSU18M91/CSU18M92 是一款支持称重测量、阻抗测量、心率测量(**心率测量仅 CSU18M91 支持**)、显示驱动的 SOC。芯片包含人体阻抗测量模块 BIM、24bit Sigma-Delta ADC、8K×16 位 MTP 程序存储器、128 字节 EEPROM 和 896 字节数据存储器。

外设资源：31 个双向 I/O 口，11 个内部中断、2 个外部中断，1 路蜂鸣器，1 组输入全差分 24bit Sigma-Delta 型 ADC，6 路单端输入的 10 位 SAR 型 ADC，1 个 MDU 运算单元，2 路 UART，1 路 I2C 从机，1 路 SPI，1 路低电压比较，支持 4*24、6*22 点阵 LCD 或 4*14 点阵 LED。

封装类型：CSU18M91：LQFP48、LQFP64；CSU18M92：裸片、LQFP48

注 1：CSU18MD92 是 CSU18M92 的裸片型号。

注 2：为了统一描述，下面的 CSU18M9x 泛指 CSU18M91 和 CSU18M92 整个系列。

注 3：CSU18M91-LQFP48 与 CSU18M92-LQFP48 管脚兼容。

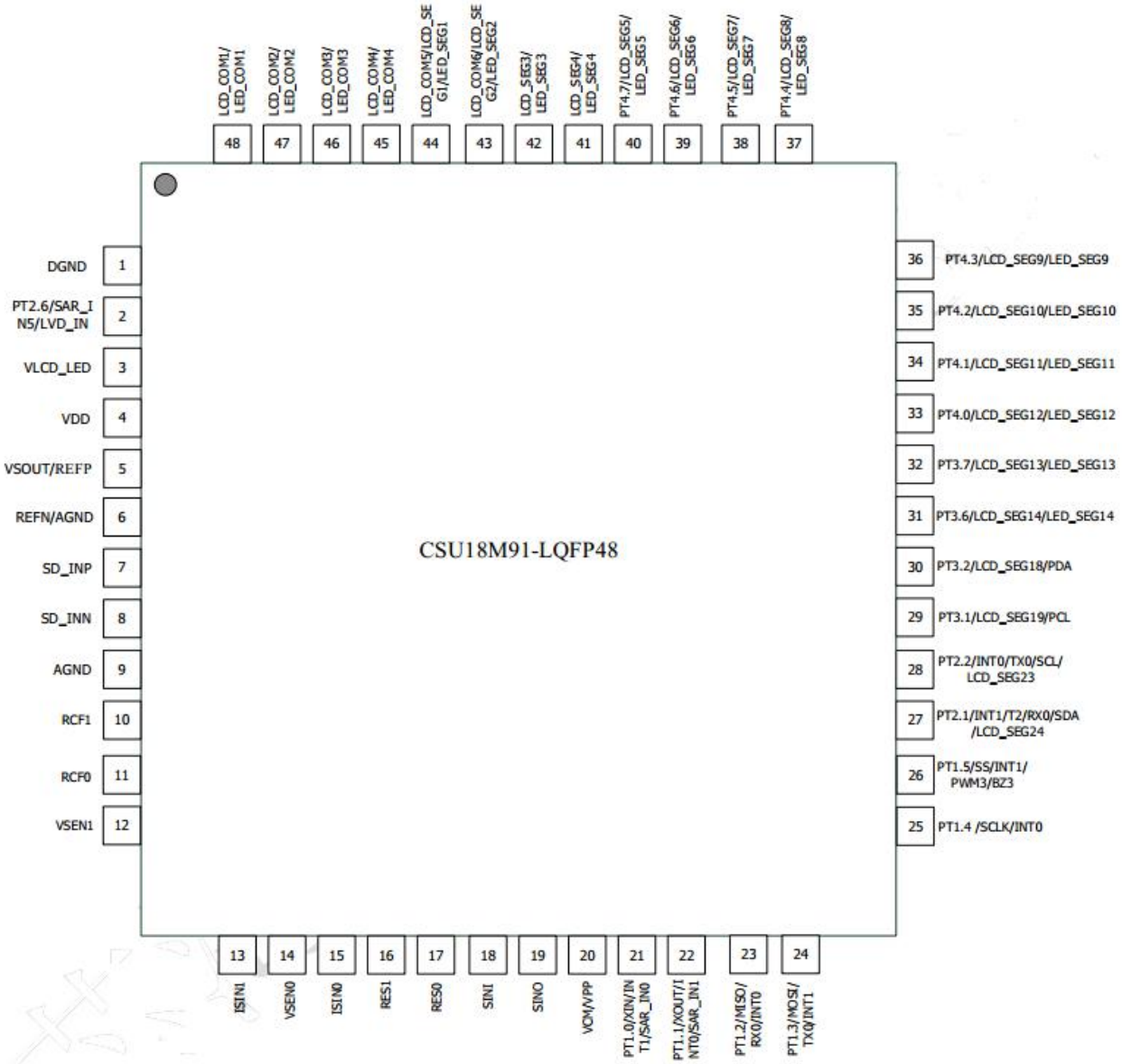


图 1-1 芯片管脚分布图(CSU18M91-LQFP48)

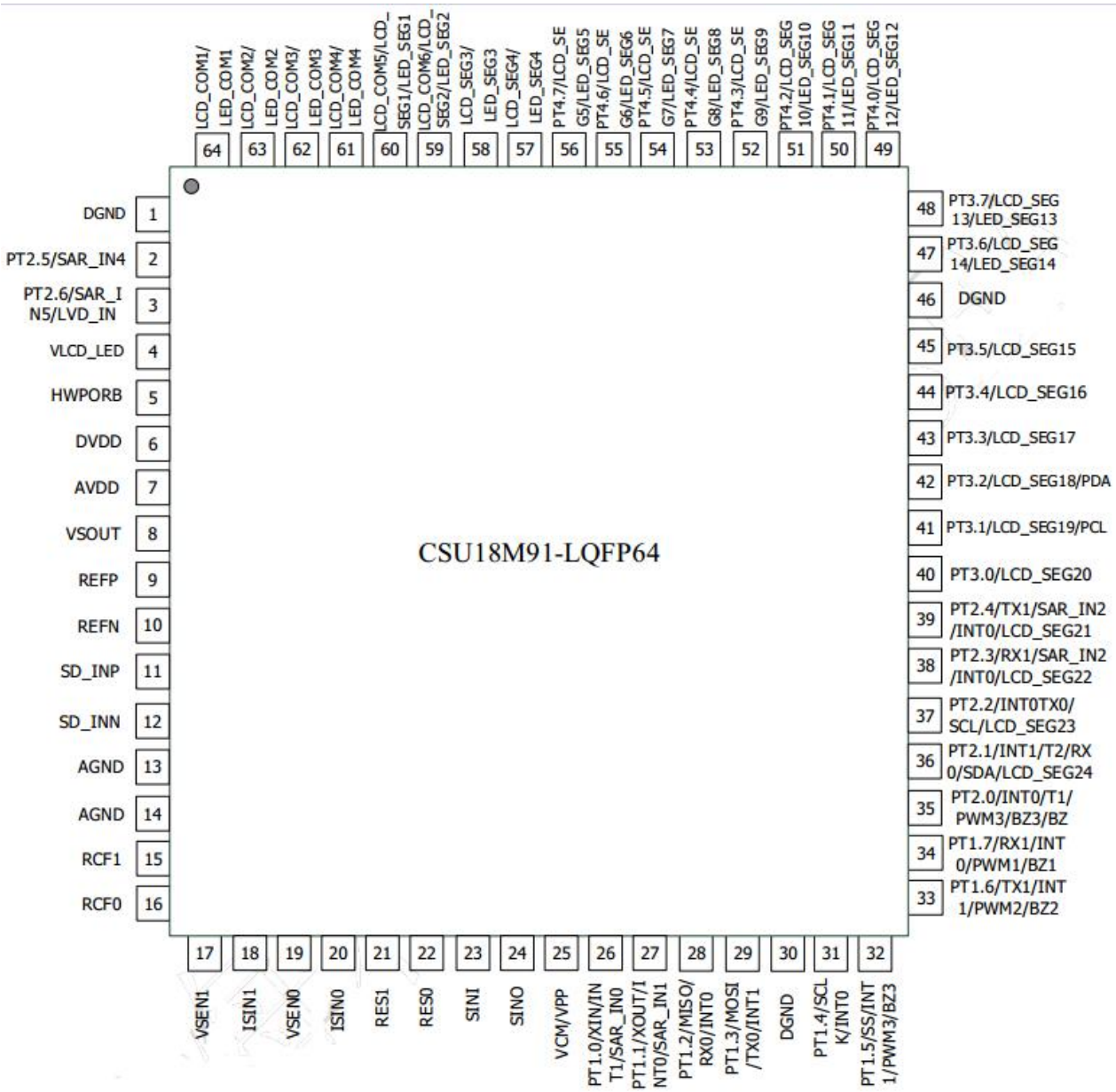


图 1-2 芯片管脚分布图(CSU18M91-LQFP64)

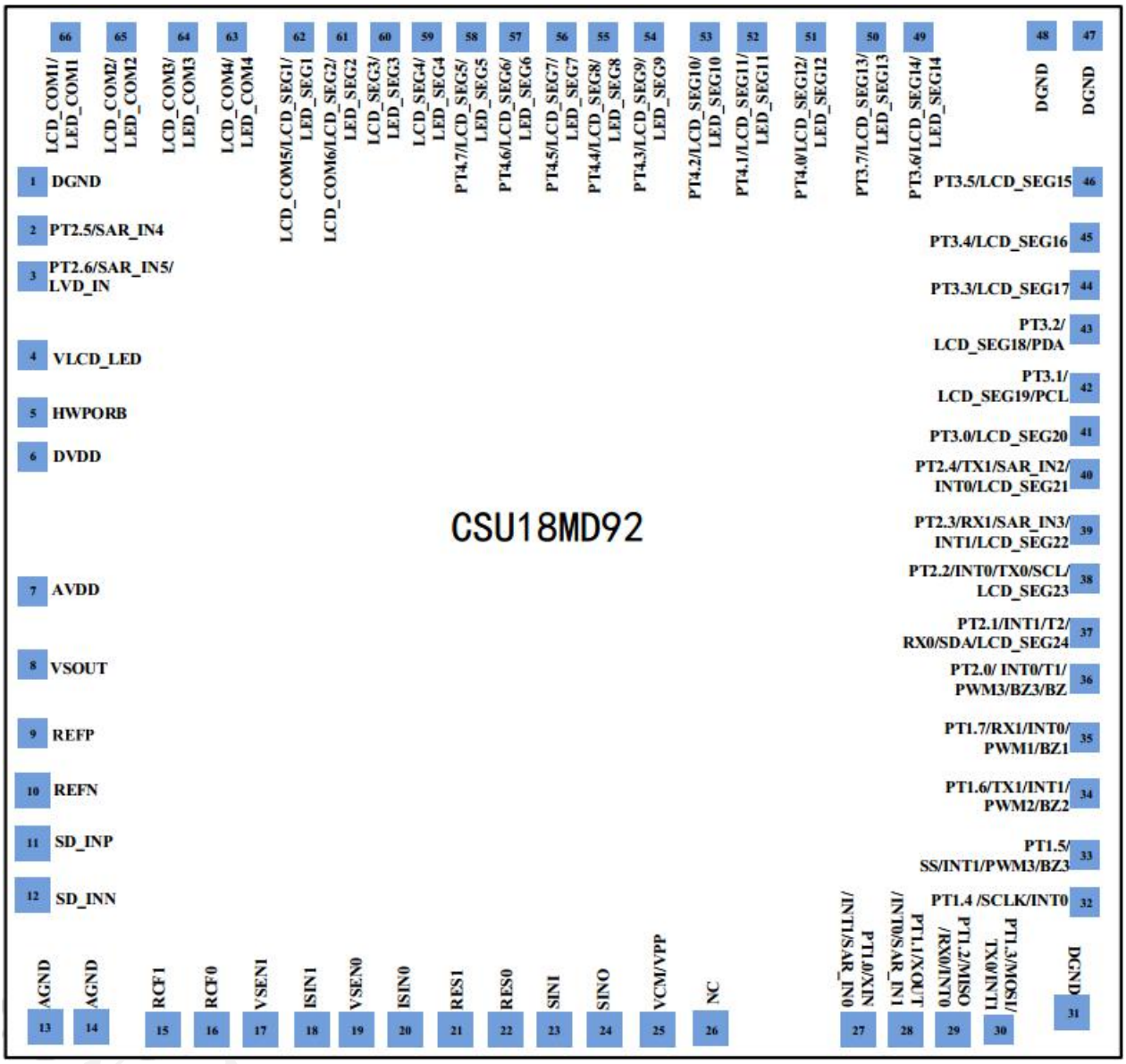


图 1-3 芯片管脚分布图(CSU18MD92)

1.2 开发环境

芯海 CSU-IDE V5.4.0（或以上版本）支持 CSU18M9x 系列芯片的开发、编译和仿真。

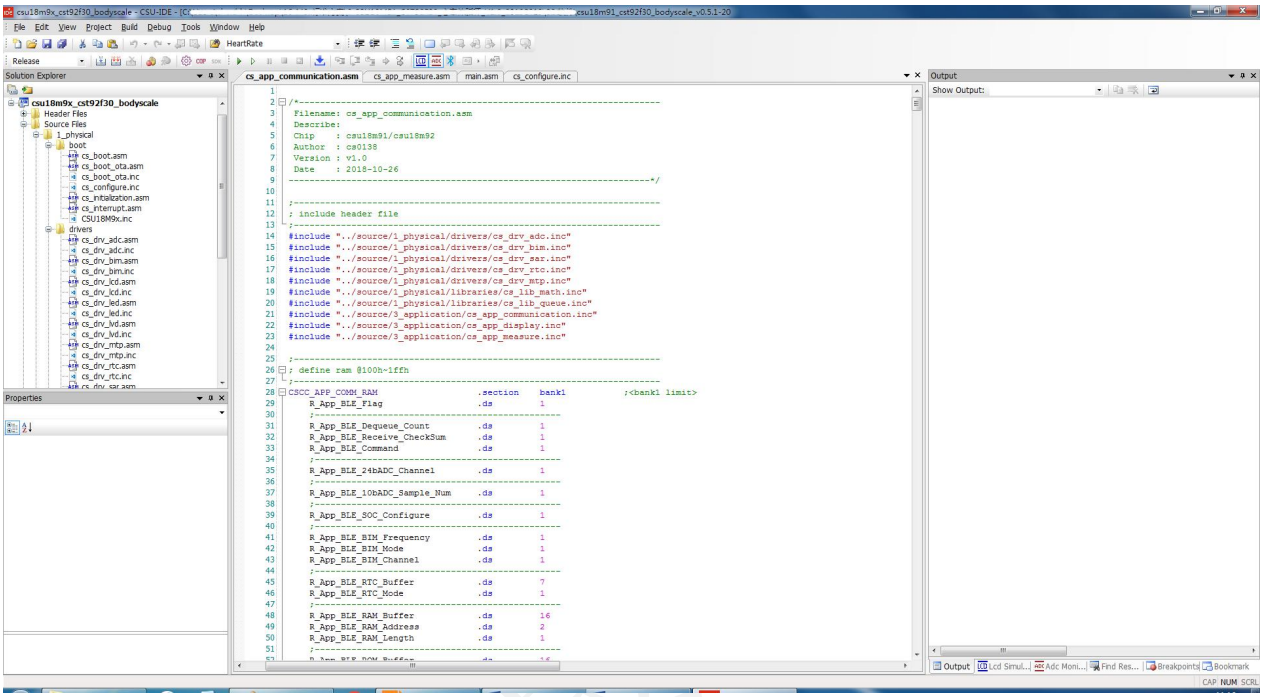


图 1-4 CSU18M9x 开发环境截图

由于 V5 版本 IDE 与 V4 版本 IDE 存在差异，所以建议阅读[Help]里面的帮助文档：User_Manual，CSU_ASM_Programming_Guide，CSU_IDE_FAQ 等。

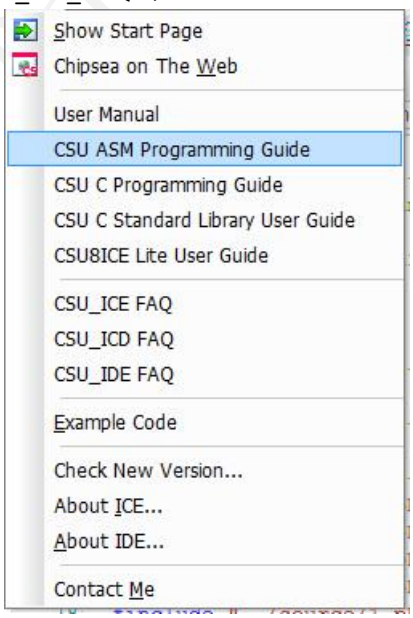


图 1-5 Help 文档链接

新建工程选择型号：

1) **CSU18M91-LQFP48/CSU18M91-LQFP64 选择 CSU18M91 新建工程**

2) **CSU18M92-LQFP48/CSU18MD92 选择 CSU18M92 新建工程**

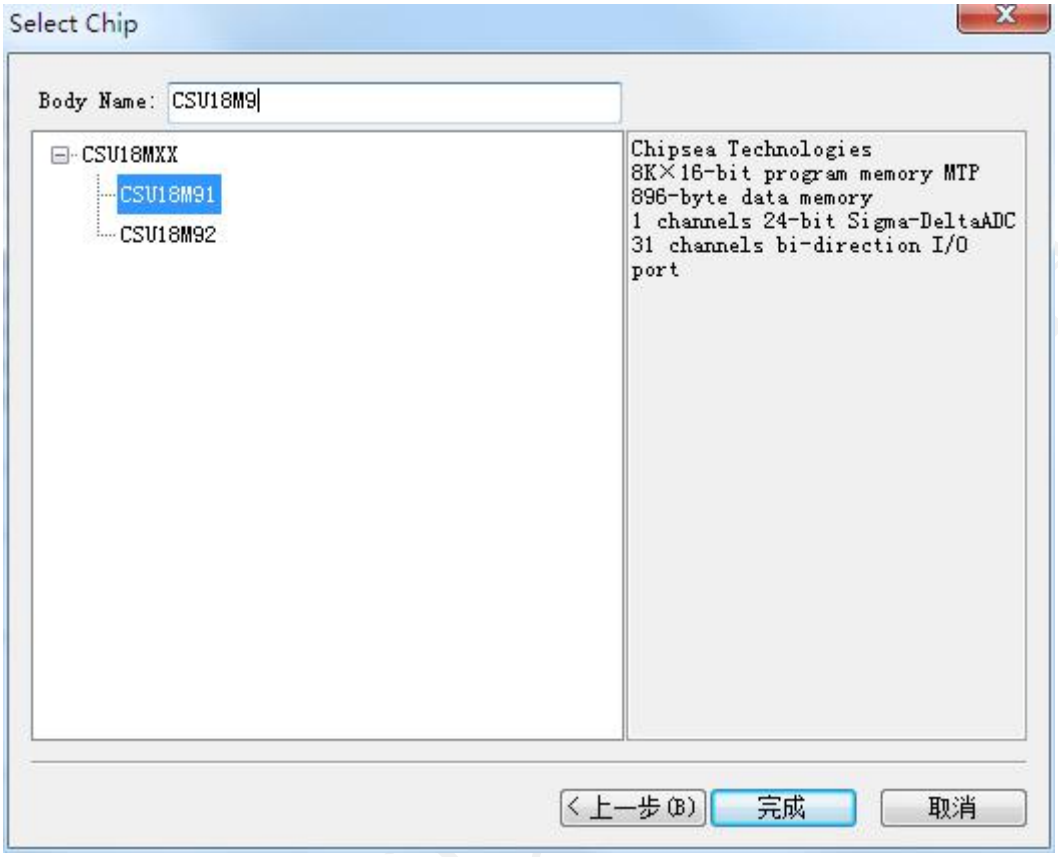


图 1-6 选择芯片型号

推荐使用 Visual Studio Code 作为代码编辑工具。对于汇编工程，可以安装“chipsea”插件用于高亮关键字。

需要注意“chipsea”插件目前对某些关键字（例如 INTF3）不支持，导致无法高亮显示，这不影响代码的编辑使用。

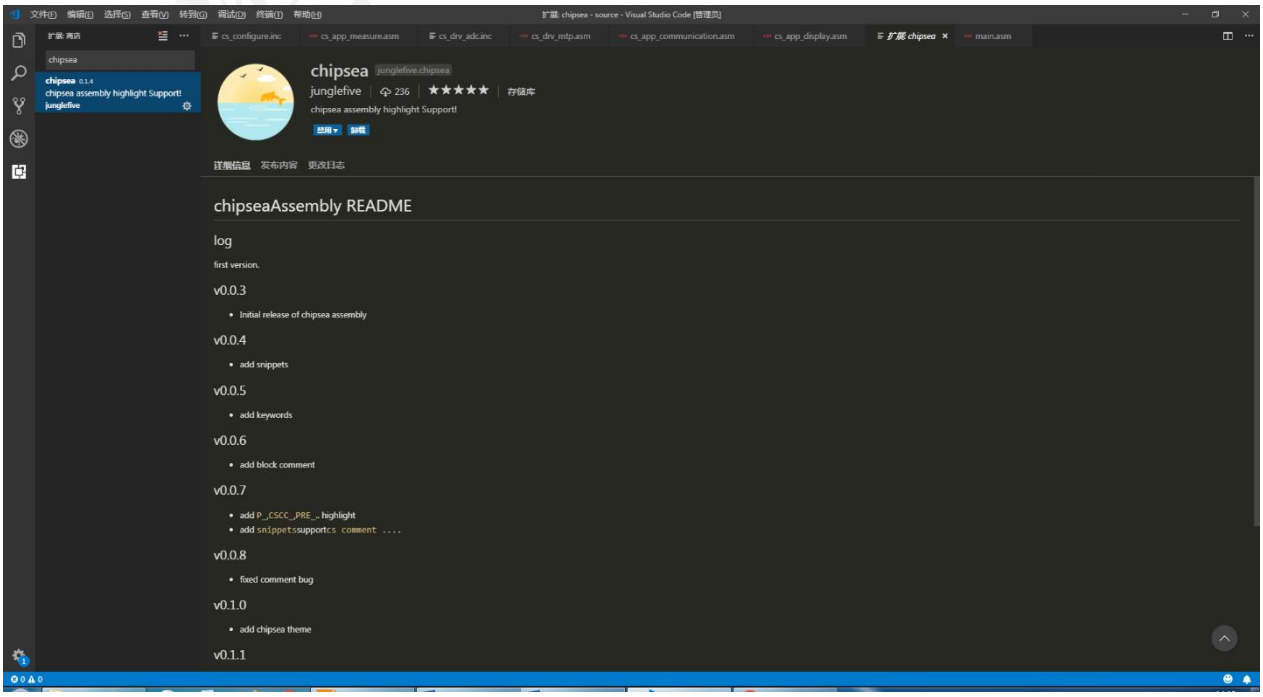


图 1-7 Visual Studio Code 截图

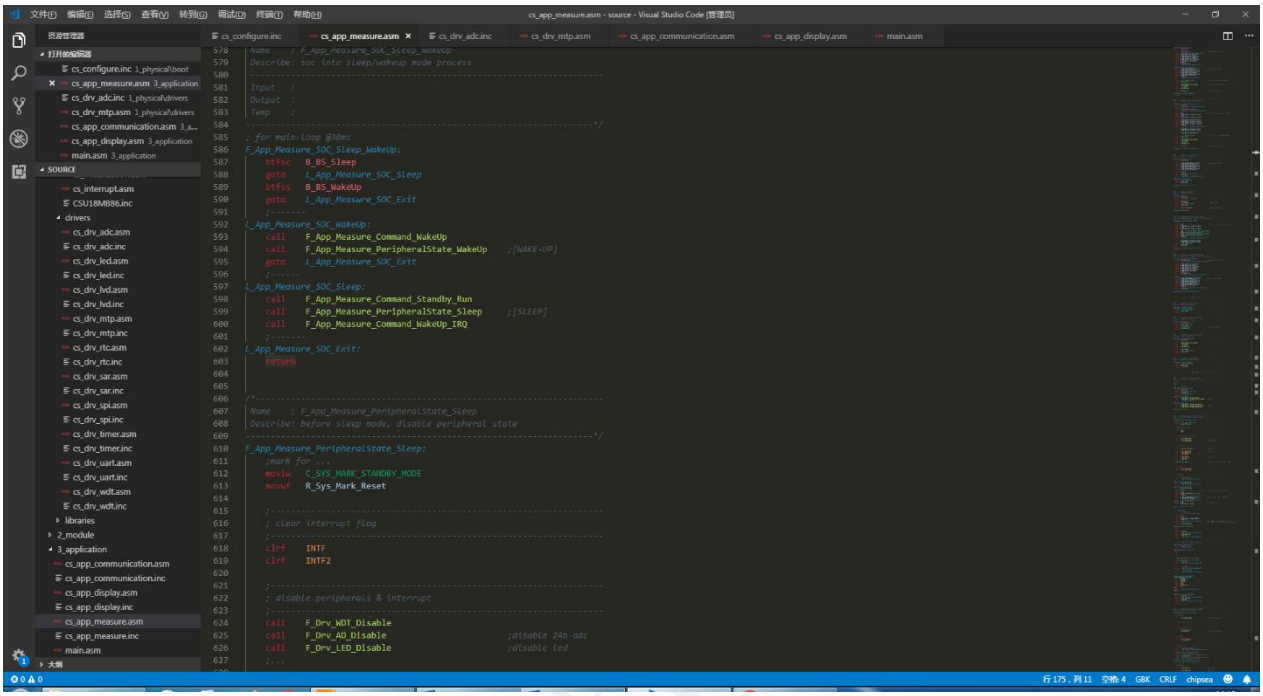


图 1-8 chipsea 插件效果

1.3 仿真配置

打开/新建工程后，需要设置代码选项，点击[Project]→[Set CodeOption]，弹出设置界面，具体设置可以参考下图：

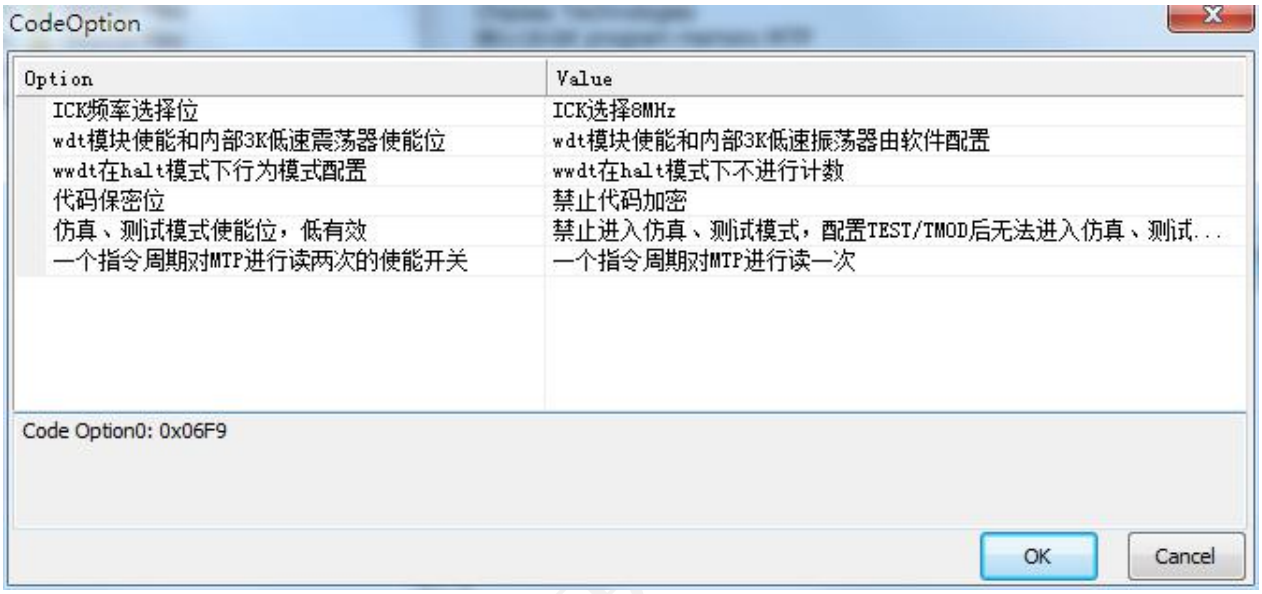


图 1-9 代码选项设置

如果需要对代码进行加密，则选择“使能代码加密”，如下所示：

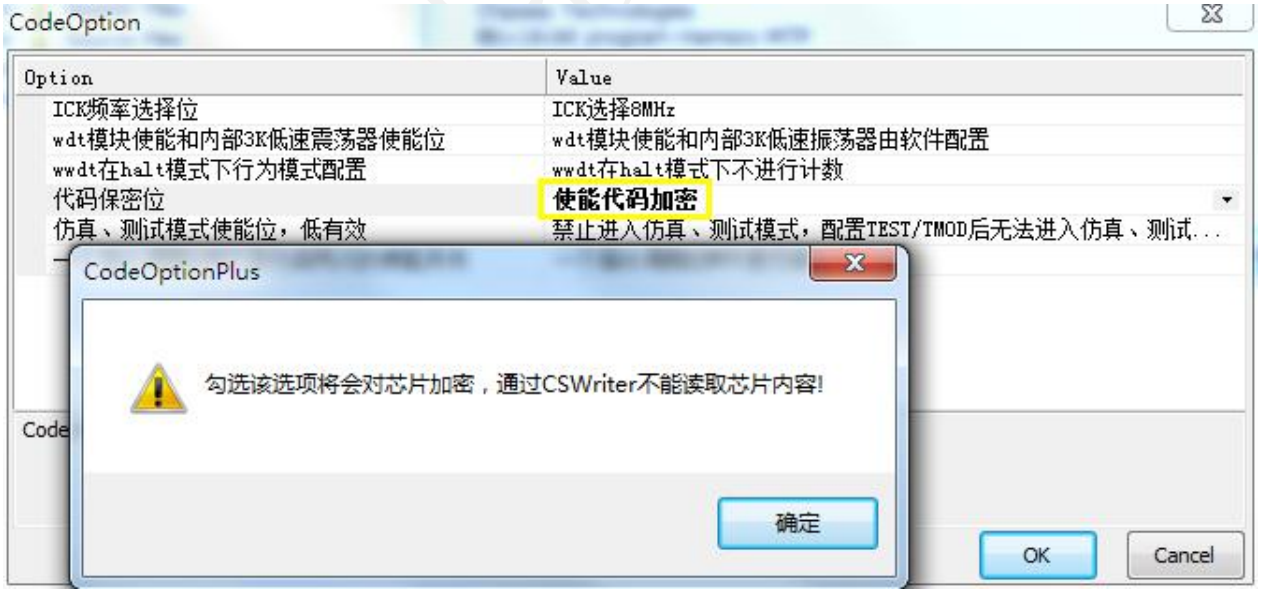


图 1-10 代码加密设置

点击[Project]→[Settings]，弹出工程设置界面，设置 LCD/LED 段码数量，然后点击[LCD]按钮设置仿真段码，规则详见[Help]→[User Manual]。

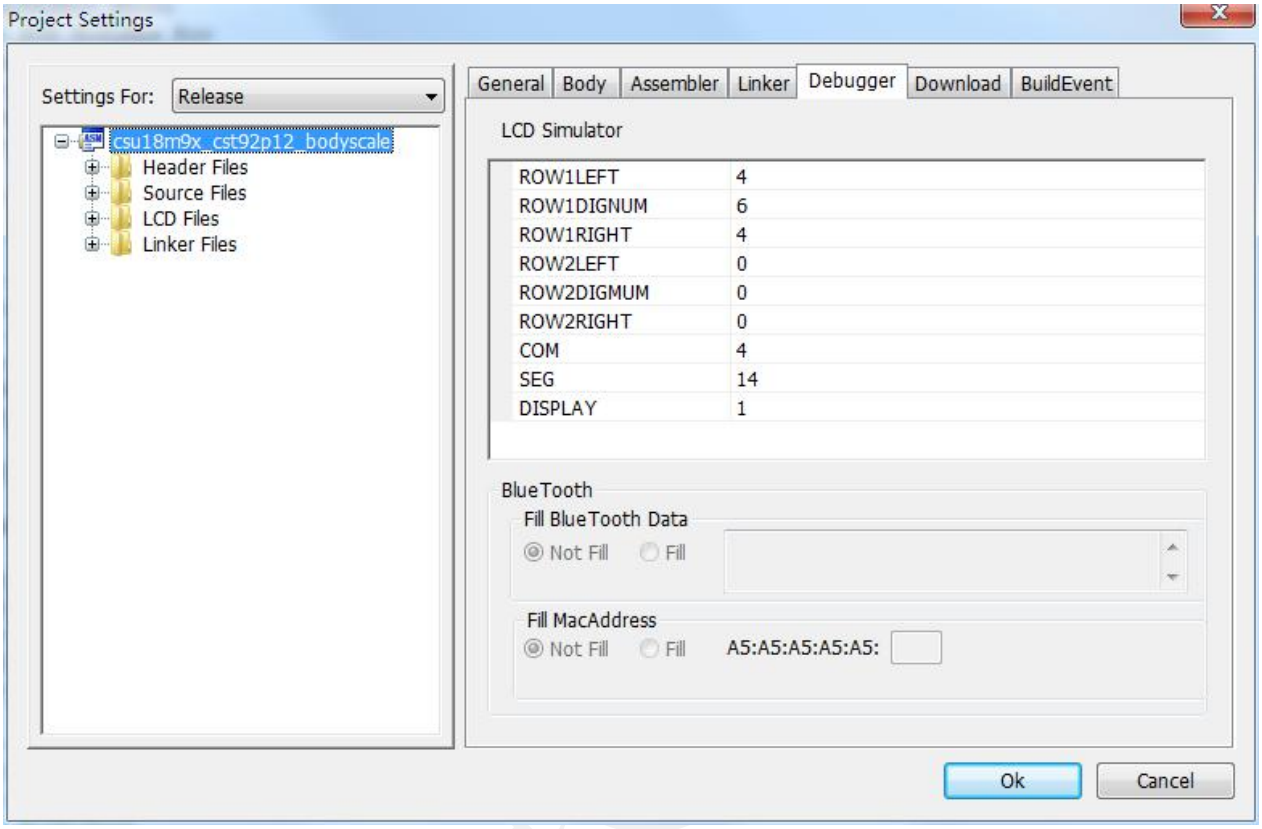
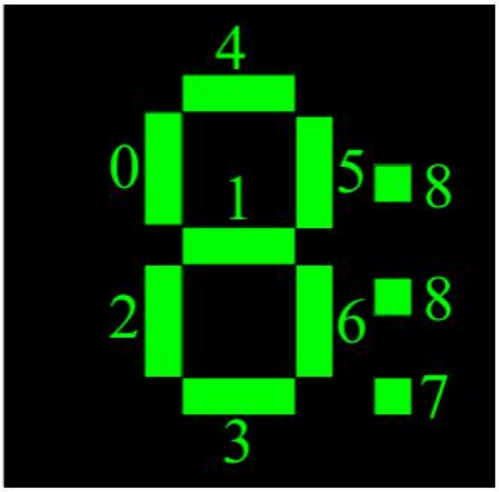


图 1-11 工程设置界面

ST： LCD 第一行的数码管。

SD： LCD 第二行的数码管。

其中数码管的段号进行如下约定：



再加上一些数字，就可以指定四个红点具体的某一个；或是第几个数码管的第几段。

对于红点，只需在 LT、 LD、 RT、 RD 后面加上 0 到 3 的数字，红点从上往下数，分别对应 0 到 3。例如：RT2 则表示 LCD 第一行右边的第 3 个红点（从上往下数）。对于数码管，在 ST 和 SD 后加数字表示第几个数码管（从左往右数），之后加下划线“ _”，再加数字（ 0 到 8）。其中数字表示这个数码管对应的段号。例如：ST4_5 则表示 LCD 第一行数码管的第四个数码管的第 5 段。

注意： LT、 LD、 RT、 RD、 ST、 SD 后加的数字受 LCD 配置的参数约束，如果超出约束则称为无效映射关系字符。 如在 LCD Setting 中把 LCD 第一行数码管的个数配为 6，那么之后 ST 后的数字范围为 1 到 6；把 LCD 第二行左边的红点设为 3，则 LD 后的数字范围为 0 到 2。

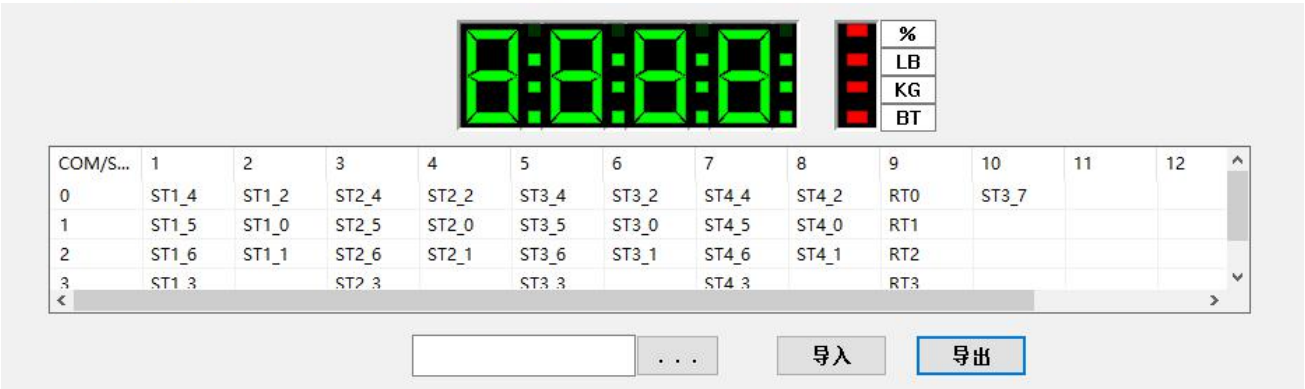


图 1-12 LCD/LED 设置界面

1.4 仿真工具

方案开发需要使用 CSU8ICE Lite 进行仿真，仿真小板为 EVL-18MX91。

仿真小板的阻抗测量端口为：VOR、VOL、FOR、FOL，其中 FOR、FOL 为电流激励端口，VOR、VOL 为电压测量端口。仿真时可以使用精密电阻来模拟人体阻抗。

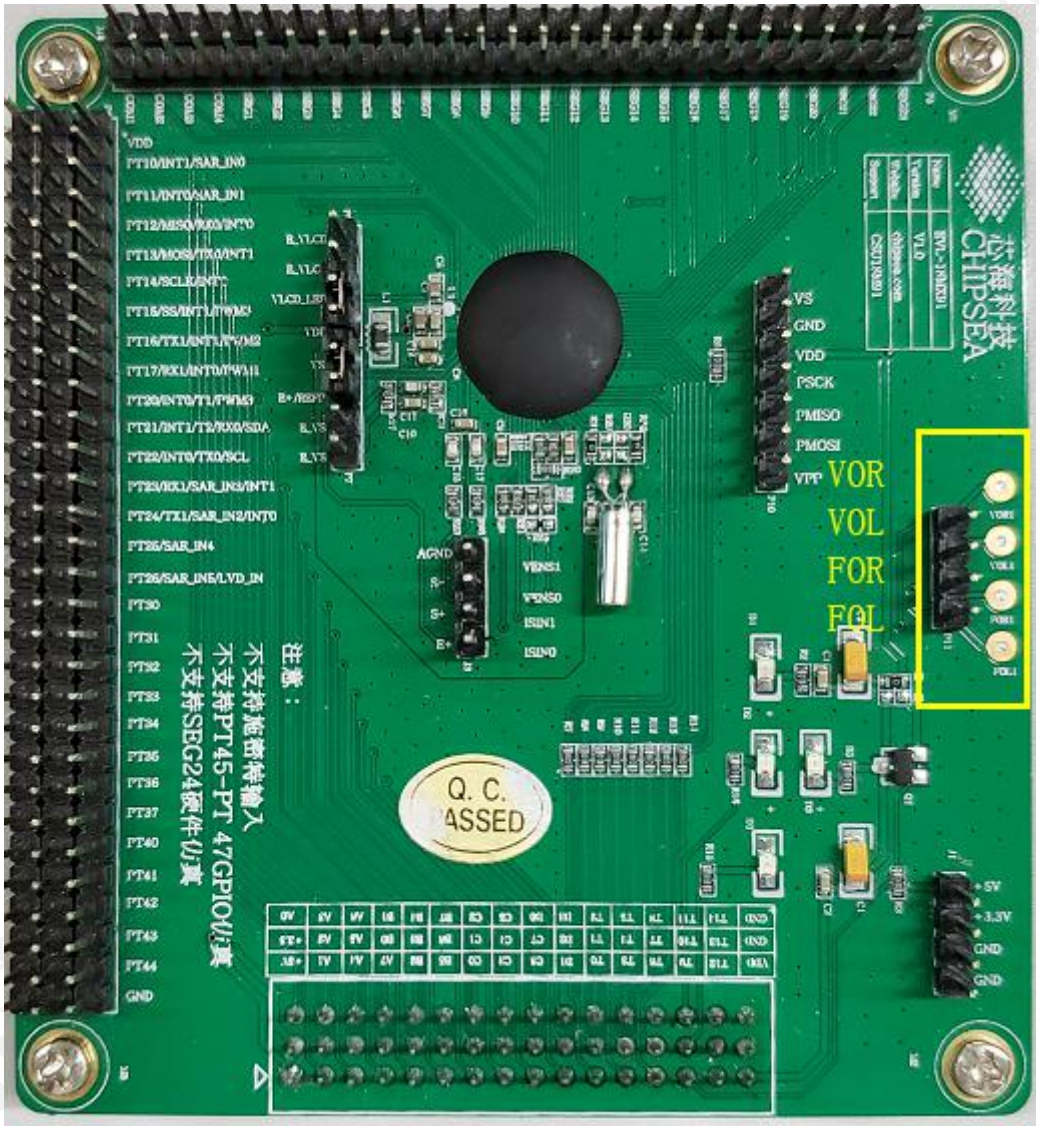


图 1-12 仿真小板

用户指南详见[Help]→[CSU8ICE Lite User Guide]。

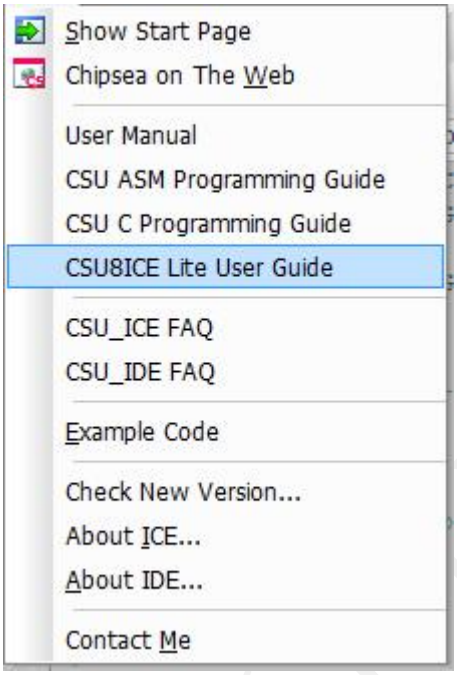


图 1-13 Help 文档链接

1.5 烧录工具

烧录上位机软件需要 CSWrite V2.3.2 或以上版本，支持 CSU18M91、CSU18M92 型号。**CSU18MD92 选择的烧录型号是 CSU18M92-LQFP48**。

CSU18M9x 烧录采用 5 线制：GND/VDD/PCL/PDA/VPP。详见用户手册描述。

对于引出 HWPORB 引脚的芯片烧录的时候需要把该引脚上拉到电源才能进行烧录。

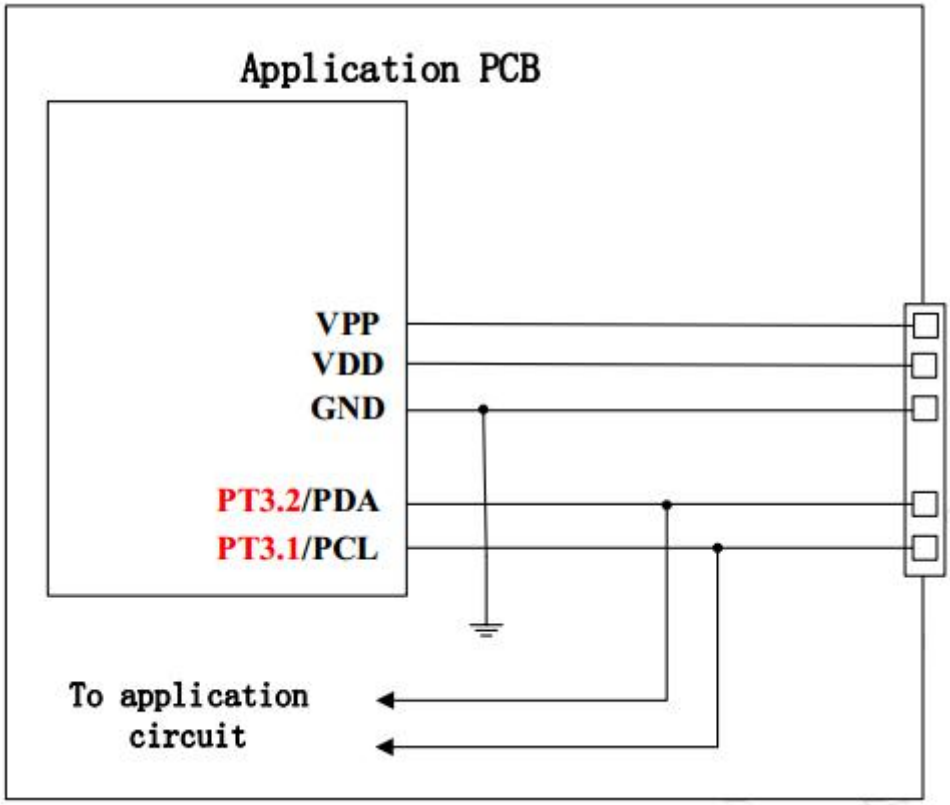


图 1-14 烧录接口示意图

2 软件设计

2.1 方案简介

以下为 CSU18M9x 的典型应用方案（智能体脂秤）：

表 2-1 CSU18M9x 方案组合

方案组合	说明
CSU18M9x+CST92P12	支持 BLE 通信 支持 LED/LCD 显示 支持 4 电极阻抗测量 CSU18M91 支持心率测量
CSU18M9x+CST92P23	**支持无线广播** 支持 LED/LCD 显示 支持 4 电极阻抗测量
CSU18M9x+CST92F30	支持 BLE 通信 支持 LED/LCD 显示 支持 4 电极阻抗测量 CSU18M91 支持心率测量
CSU18M9x+CST92F30+CSM64F02	支持 BLE 通信 支持 WIFI 通信 支持 LED/LCD 显示 支持 4 电极阻抗测量 CSU18M91 支持心率测量

2.1.1 CSU18M9x+CST92P12 方案

CSU18M9x 为系统主控，主要负责体重测量、阻抗测量、心率测量以及显示驱动，CST9P12 负责蓝牙广播与透传。CST92P12 的加载代码需要预烧录在 Flash 里。

体重测量、阻抗测量与心率测量需要串行处理，一般而言，体重锁定后切换至阻抗测量，测量完成后再进行心率测量，最后返回称重模式。

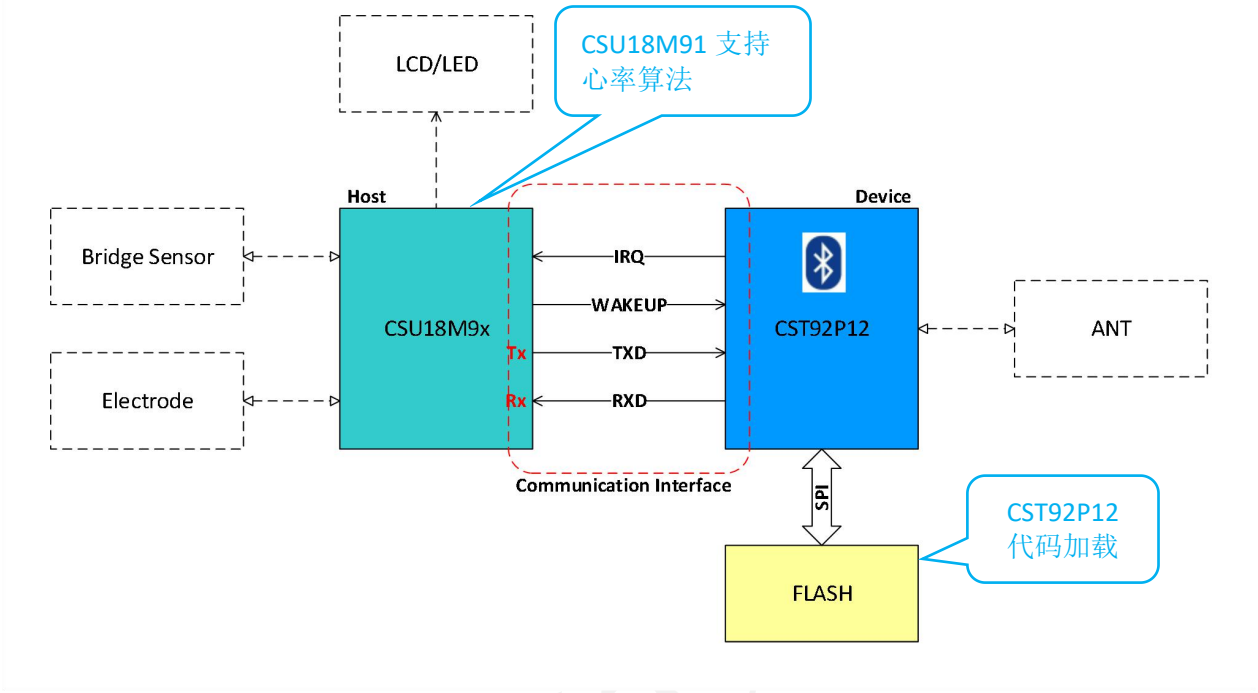


图 2-1 CSU18M9x+CST92P12 系统框图

2.1.2 CSU18M9x+CST92P23 方案

CSU18M9x 为系统主控，主要负责体重测量、阻抗测量以及显示驱动，CST9P23 负责无线广播。

由于《OKOK 健康秤 APP 接入协议-单向广播体脂秤 V3》不支持心率数据传输，所以方案不具备心率测量功能。

CST92P23 是一款低功耗、低成本、高集成度的无线收发芯片，片上集成发射机、接收机、GFSK 调制解调器、基带等。该芯片采用 SOP14 封装，外围电路简单，搭配低成本 MCU 和少量外围被动器件，即可实现无线信号传输。

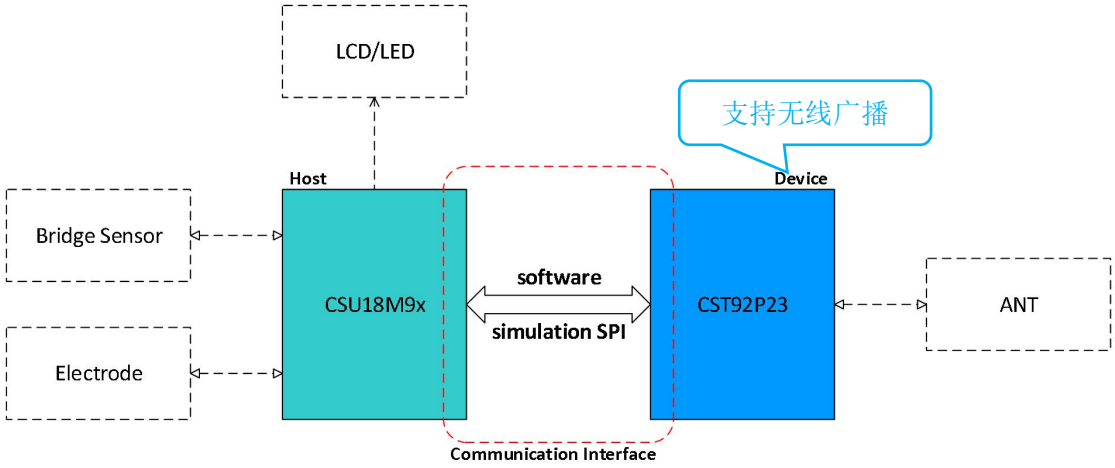


图 2-2 CSU18M9x+CST92P23 系统框图

2.1.3 CSU18M9x+CST92F30 方案

系统设计可以把 CSU18M9x 作为 Device，CST92F30 作为 Host。Device 主要负责数据采集、显示驱动和休眠管理，其余全部工作由 Host 完成，包括心率算法、蓝牙传输和用户数据管理。

CST92F30 是一颗高集成度的低功耗蓝牙 SOC 芯片，基于低功耗蓝牙 5.0 协议栈。芯片内置 32 位 Cortex-M0 CPU、512KB Flash、138KB SRAM。

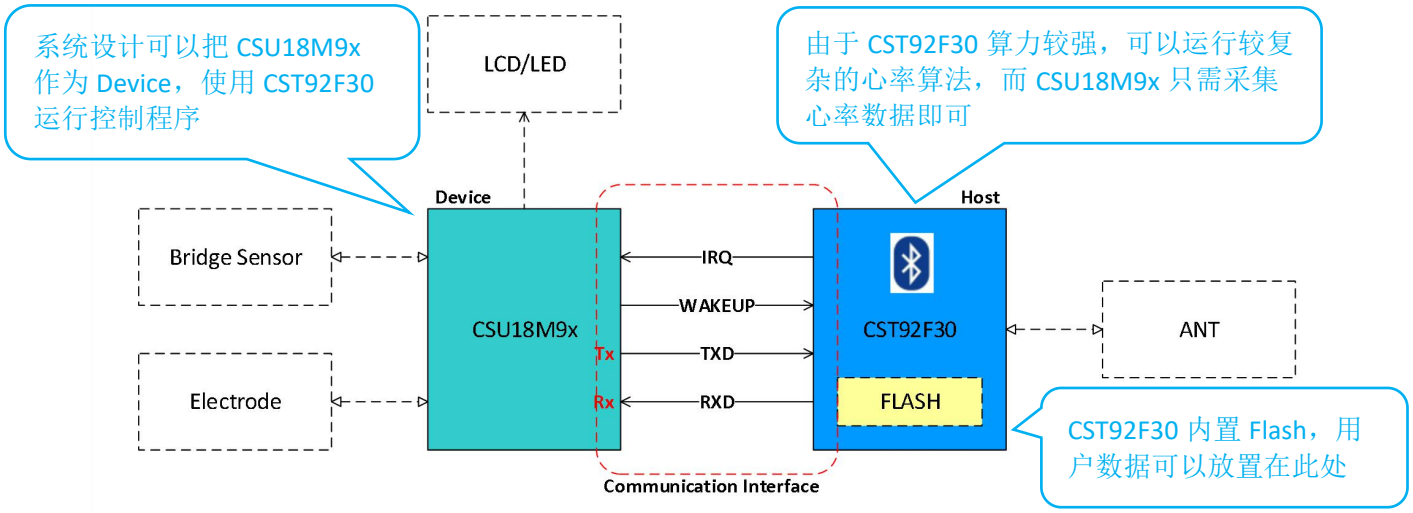


图 2-3 CSU18M9x+CST92F30 系统框图

2.1.4 CSU18M9x+CST92F30+CSM64F02 方案

系统设计可以把 CSU18M9x 作为 Device，CST92F30 作为 Host。Device 主要负责数据采集、显示驱动和休眠管理，其余全部工作由 Host 完成，包括心率算法、蓝牙传输和用户数据管理。

Host 与 WIFI 采用 SPI 接口进行通信，SPI 接口以 Host 为 master 模式，WIFI 为 slave 模式。WIFI 主要作为透传模块，交互协议由 Host 来处理。由于 WIFI 功耗较高，所以待机时需要断开电源。

CSM64F02 是面向物联网市场的 802.11b/g/n Wi-Fi 模块。模块集成了 32 位高速 CPU、TCP/IP 协议栈、RTOS、WiFi 射频前端、板载天线，支持 ADC/ SPI/ UART/ I2C/ PWM 等 IO。

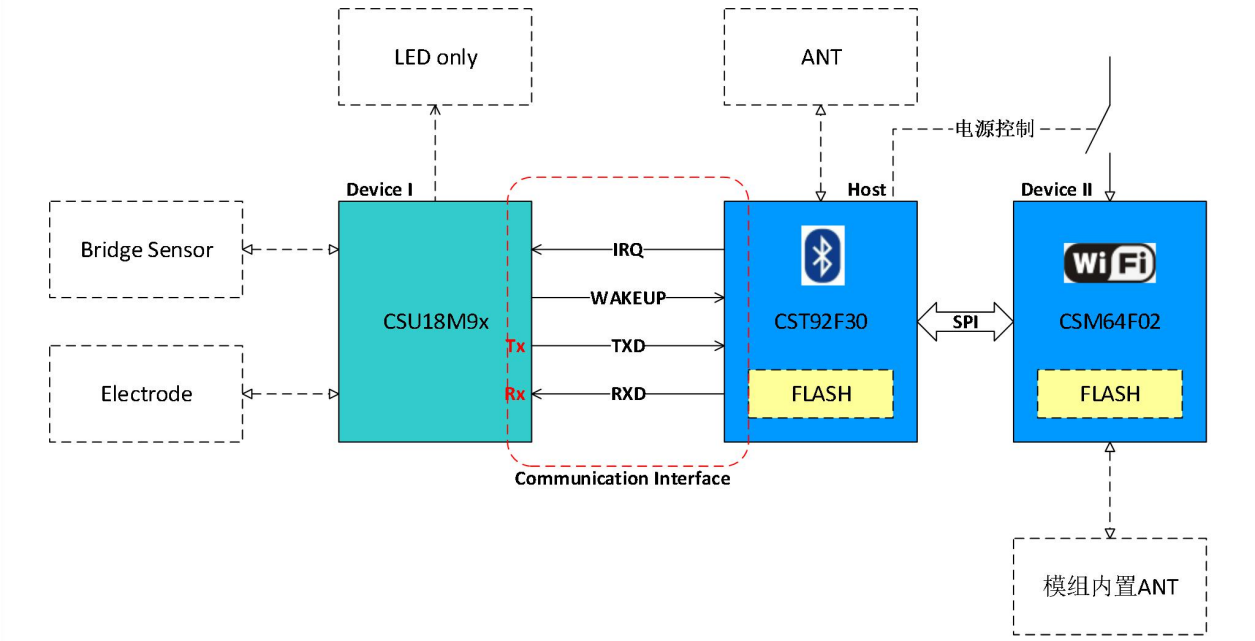


图 2-4 CSU18M9x+CST92F30+CSM64F02 系统框图

2.2 系统架构

根据章节<2.1 方案简介>的内容，从 CSU18M9x 是否作为系统主控的角度出发，智能体脂秤的软件架构可以分成两种：

1) CSU18M9x 为系统主控，主要负责体重测量、阻抗测量、心率测量以及显示驱动，BLE/WIFI 仅仅用于数据传输。详细内容可以参考《SOC Hostless 应用指南》。
2) CSU18M9x 作为 Device，BLE 作为 Host。Device 主要负责数据采集、显示驱动和休眠管理，其余全部工作由 Host 完成，包括心率算法、蓝牙传输和用户数据管理。详细内容可以参考《SOC Hosted 应用指南》。

2.3 应用指南

下文是关于 CSU18M9x 的应用指南，推荐配置与示例代码仅供参考，用户需要根据实际需求选择合适的配置。

2.3.1 体重测量

用户可以参考用户手册《DS_CSU18M9x_Vx.x_cn》关于 24bit Sigma Delta ADC 的应用场景和配置推荐。配置 ANACFG 寄存器时需要确保 24b-ADC 处于关闭状态。对于人体秤应用，VS 推荐使用 2.3V 或 2.4V 档位，VS 配置完毕需要延时一段时间让模拟电源稳定后再使能 24b-ADC。

表 2-2 体重测量参考配置

应用场景	寄存器位	推荐配置	
人体秤	SDCKS	=1	SD_SMPCK=500KHz
	SDGAIN[1:0]	=10/11	GAIN=64/128
	SDREFS[1:0]	=00	SDADC 参考电压为（REFP-REFN）
	SDCHP[1:0]	=00	SDADC 斩波频率 1
	ADM[3:0]	=1111	降采样率 16384（3+2 阶 Comb）
	SDSINL[1:0]	=00	SDADC 输入信号来自片外压力传感器

为了实现自动上秤功能，可以设置每隔 1s 唤醒系统检测是否有人上秤。为了节省功耗，ADC 输出速率通常选择 1953Hz，采集几组数据后（丢弃前三组）根据 AD 值判断是否有人上秤。如果没有人上秤则继续处于休眠状态，如果有人上秤则返回测量状态。

当从 24bit Sigma Delta ADC 发生配置变换时，都需要丢掉至少三笔数据。

表 2-3 自动上秤参考配置

应用场景	寄存器位	推荐配置	
快速上秤	SDCKS	=1	SD_SMPCK=500KHz
	SDGAIN[1:0]	=10/11	GAIN=64/128
	SDREFS[1:0]	=00	SDADC 参考电压为（REFP-REFN）
	SDCHP[1:0]	=00	SDADC 斩波频率 1
	ADM[3:0]	=0001	降采样率 256（3 阶 Comb）
	SDSINL[1:0]	=00	SDADC 输入信号来自片外压力传感器

温度补偿寄存器 TEMPC 主要用于称重传感器的温度增益误差补偿，具体补偿值需要整机进行高低温测试来确定。人体秤一般取值+45ppm/+90ppm。

示例代码：

```
/*---------------------------------------------------------------------
 Name    : F_AD_Change_To_NormalSpeed
 Describe: set adc speed as high-resolution(slow-speed) mode
---------------------------------------------------------------------*/
F_Drv_AD_Init:
F_AD_Change_To_NormalSpeed:
    M_ADC_SDAD_DISABLE                  ;disable 24b-adc
    movlw   C_ADC_ANACFG                ;ldo enable; vs=2.4v; sd_inp/sd_inn; 24b-
adc disable
    movwf   ANACFG
    movlw   C_ADC_SDCFG                 ;sf=500KHz; pga=128
    movwf   SDCFG
    movlw   C_ADC_SDCON_NORMAL_SPEED    ;dr=30.5hz
    movwf   SDCON
    movlw   C_ADC_TEMPC                 ;tc=+45ppm -> depending on bridge sensors
    movwf   TEMPC
    ;-------
    movlw   0xFF                        ;<delay>
    decfsz  WORK,F
    goto    $-1
    ;-------
    M_ADC_SDAD_ENABLE                   ;enable 24b-adc
    bcf     INTF,SDIF
    bsf     INTE,SDIE                   ;enable 24b-adc interrupt
    bsf     INTE,GIE                    ;enable global interrupt
    return




/*---------------------------------------------------------------------
 Name    : F_AD_ChangeToHighSpeed
 Describe: set adc speed as high-speed mode
---------------------------------------------------------------------*/
F_AD_ChangeToHighSpeed:
    movlw   C_ADC_ANACFG                ;ldo enable; vs=2.4v; sd_inp/sd_inn; 24b-
adc disable
    movwf   ANACFG
    movlw   C_ADC_SDCFG                 ;sf=500KHz; pga=128
```

```
movwf   SDCFG
movlw   C_ADC_SDCON_HIGH_SPEED      ;dr=1953hz
movwf   SDCON
movlw   C_ADC_TEMPC                 ;tc=+45ppm -> depending on bridge sensors
movwf   TEMPC
;-------
movlw   0xFF                        ;<delay>
decfsz  WORK,F
goto    $-1
;-------
M_ADC_SDAD_ENABLE                   ;enable 24b-adc
bcf     INTF,SDIF
bsf     INTE,SDIE                   ;enable 24b-adc interrupt
bsf     INTE,GIE                    ;enable global interrupt
return
```

2.3.2 阻抗测量

CSU18M9x 集成了一个交流人体阻抗测量(BIM)模块，主要用于人体阻抗测量，然后软件结合其它参数计算人体成分。

测量原理是将人体等效为一个阻容网络，输出一路电流流过该网络，以产生一个和网络阻抗成正比的压降。通过 Sigma-Delta ADC 测得该压降即可换算出阻容网络的等效阻抗，然后通过查询人体阻抗数据表格，将人体等效阻抗换算成人体的组成成分。人体阻抗数据表格包含人体阻抗的电学模型、人体阻抗和人体脂肪含量的关系等，通常和人的年龄、性别、身高体重、以及人种有关。

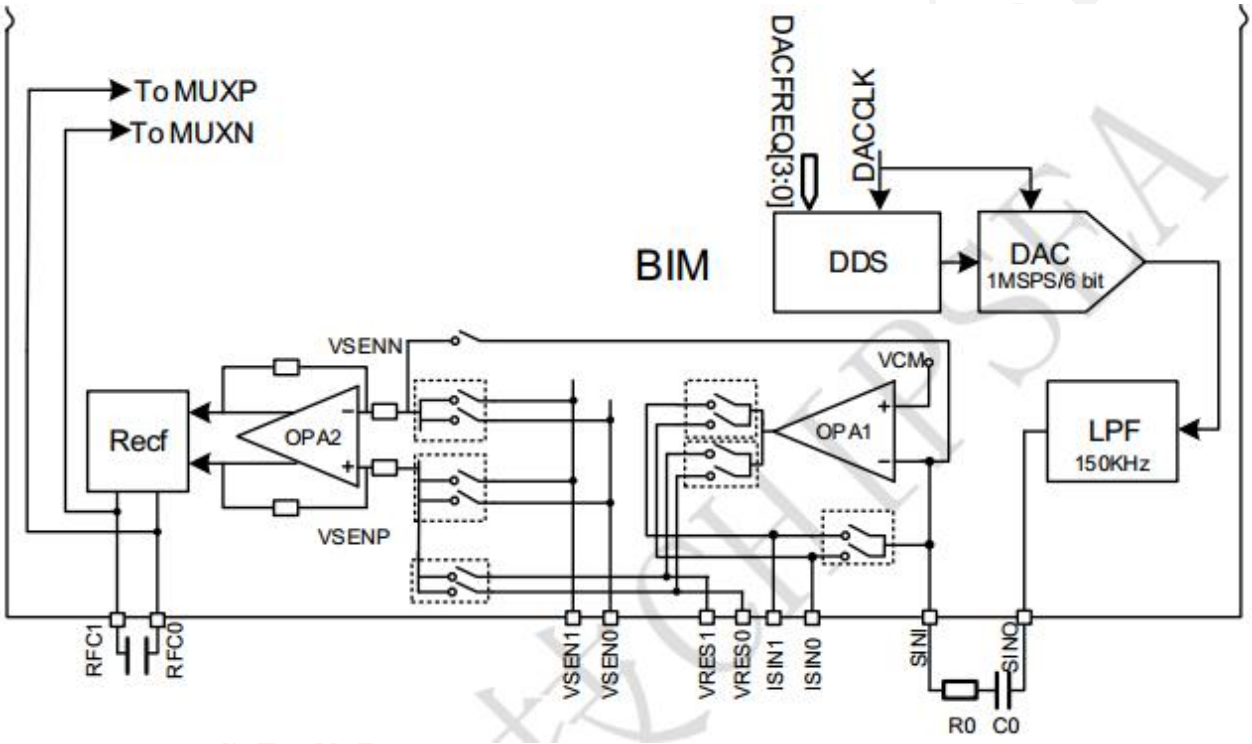


图 2-5 BIM 模块结构图

如上图所示为 BIM 的模块结构图，其中正弦信号发生器（包括 DDS、DAC、LPF 等）可以产生正弦波信号，该信号经过 C0、R0 做高通滤波及限流后转换为正弦电流，并通过至少一对激励电极在人体的等效阻容网络上形成一个电压降。端口 ISINx， x=0/1，安装在人体不同的两个部位，一个负责发射正弦电流激励信号，一个负责接收该激励信号。通过测量电极（端口 VSENy， y=0/1）探测该电压降信号后，芯片内部对其进行整流滤波等处理，然后送入 ADC 中测量将模拟电压信号转换为数字信号，从而得到人体阻抗上的电压降值。

由于 BIM 模块与称重测量模块共用一个 Sigma-Delta ADC(24b-ADC)，因此**称重测量与阻抗测量需要串行工作**。

上电初始化时，需要对 BIM 模块进行初始化并获取校准电阻 Res0=300Ω、Res1=1KΩ对应的 AD 平均值，用于人体阻抗的计算。初始化完成切换为称重测量模式（默认）。

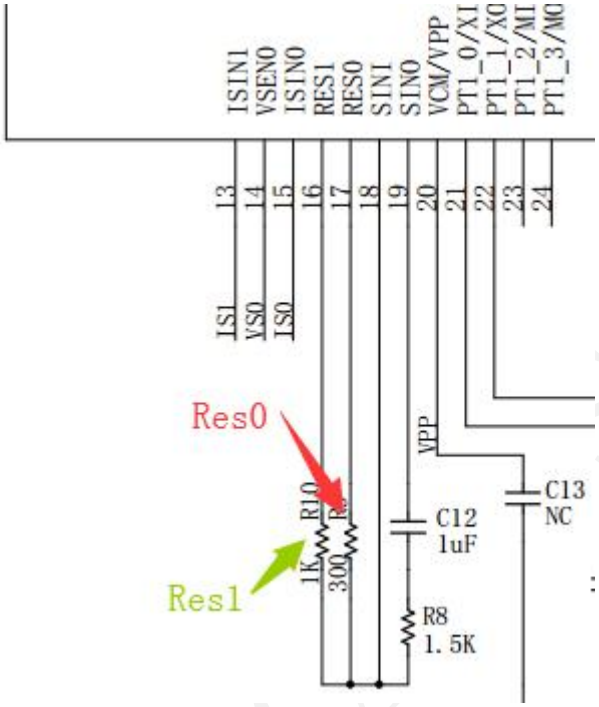


图 2-6 CSU18M9x 局部电路

测量人体阻抗时，需要注意的事项：**测量校准电阻 RES0、RES1 的参考电压、激励频率要与测量人体的一致。多频阻抗测量模式下，计算阻抗时需要利用不同频率下 RES0、RES1 对应的 AD 值。**

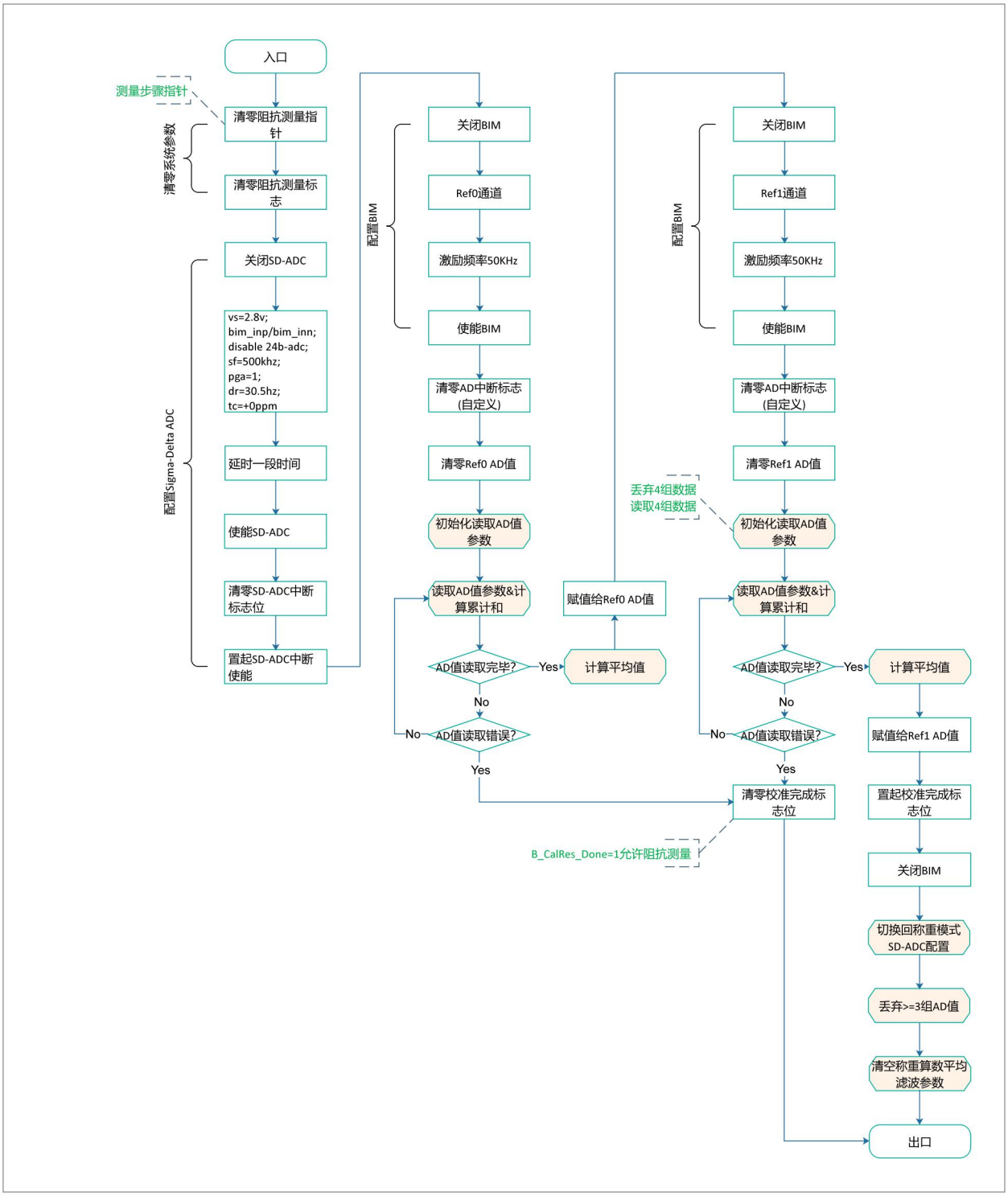


图 2-7 BIM 初始化(阻抗校准)流程图

示例代码：

```
/*---------------------------------------------------------------------
 Name    : F_Drv_BIM_Init
 Describe: init BIM & calibrate BIM (RES0/RES1)
---------------------------------------------------------------------*/
F_Drv_BIM_Init:
```

```
    clrf    R_AFE_Pointer                       ;clear pionter
    clrf    R_AFE_Flag_Sta                      ;clear state flag
    clrf    R_AFE_Flag_Err                      ;clear error flag

    M_AFE_SDAD_DISABLE                          ;disable 24b-adc
    movlw   C_AFE_ANACFG                        ;vs=2.8v; bim_inp/bim_inn;
disable 24b-adc
    movwf   ANACFG
    movlw   C_AFE_SDCFG                         ;sf=500khz; pga=1
    movwf   SDCFG
    movlw   C_AFE_SDCON_DR_30HZ                 ;dr=30.5hz
    movwf   SDCON
    movlw   C_AFE_TEMPC                         ;tc=+0ppm
    movwf   TEMPC
    movlw   0xFF                                ;<delay>
    decfsz  WORK,F
    goto    $-1
    M_AFE_SDAD_ENABLE                           ;enable 24b-adc
    bcf     INTF,SDIF
    bsf     INTE,GIE                            ;enable global interrupt
    bsf     INTE,SDIE                           ;enable 24b-adc interrupt

    ;-----------------------------------------------
    ; Get_Res0_AD
    ;-----------------------------------------------
output:R_AFE_Res0_ADL,R_AFE_Res0_ADH
L_Drv_BIM_CalBIM_Ref0:
    M_AFE_BIM_DISABLE                           ;disable BIM
    movlw   C_AFE_BIM0_RES0                     ;mode -> ref0 of BIM
    movwf   BIM0
    movlw   C_AFE_BIM1_50K                      ;frequecy -> 50KHz
    movwf   BIM1
    M_AFE_BIM_ENABLE                            ;enable BIM

    bcf     B_INT_GetADC
    clrf    R_AFE_Res0_ADL
    clrf    R_AFE_Res0_ADH
    call    F_Drv_BIM_Init_Filter_RAM_Ref

L_Drv_BIM_CalBIM_Ref0_Loop:
    call    F_Drv_BIM_Filter_AD                 ;read ad & calc sum
    btfsc   B_BIM_AD_Done
    goto    L_Drv_BIM_CalBIM_Ref0_Done
    btfss   B_BIM_AD_Err
    goto    L_Drv_BIM_CalBIM_Ref0_Loop
    goto    L_Drv_BIM_CalBIM_Err

L_Drv_BIM_CalBIM_Ref0_Done:
```

```
    call    F_Drv_BIM_AD_SUM_RRF2                   ;(R_AFE_SumHH/H/M/L)/4
    movfw   R_AFE_SumH
    movwf   R_AFE_Res0_ADH
    movfw   R_AFE_SumM
    movwf   R_AFE_Res0_ADL

    ;-----------------------------------------------
    ; Get_Res1_AD
    ;-----------------------------------------------
output:R_AFE_Res1_ADL,R_AFE_Res1_ADH
L_Drv_BIM_CalBIM_Ref1:
    M_AFE_BIM_DISABLE                               ;disable BIM
    movlw   C_AFE_BIM0_RES1                         ;mode -> ref1 of BIM
    movwf   BIM0
    movlw   C_AFE_BIM1_50K                          ;frequecy -> 50KHz
    movwf   BIM1
    M_AFE_BIM_ENABLE                                ;enable BIM

    bcf     B_INT_GetADC
    clrf    R_AFE_Res1_ADL
    clrf    R_AFE_Res1_ADH
    call    F_Drv_BIM_Init_Filter_RAM_Ref

L_Drv_BIM_CalBIM_Ref1_Loop:
    call    F_Drv_BIM_Filter_AD                     ;read ad & calc sum
    btfsc   B_BIM_AD_Done
    goto    L_Drv_BIM_CalBIM_Ref1_Done
    btfss   B_BIM_AD_Err
    goto    L_Drv_BIM_CalBIM_Ref1_Loop
    goto    L_Drv_BIM_CalBIM_Err

L_Drv_BIM_CalBIM_Ref1_Done:
    call    F_Drv_BIM_AD_SUM_RRF2                   ;(R_AFE_SumHH/H/M/L)/4
    movfw   R_AFE_SumH
    movwf   R_AFE_Res1_ADH
    movfw   R_AFE_SumM
    movwf   R_AFE_Res1_ADL

    ;-----------------------------------------------
L_Drv_BIM_CalBIM_Done:
    bsf     B_CalRes_Done                           ;cal-res done
    goto    L_Drv_BIM_CalBIM_Exit

    ;-----------------------------------------------
L_Drv_BIM_CalBIM_Err:
    bcf     B_CalRes_Done                           ;cal-res fail

    ;-----------------------------------------------
```

```
L_Drv_BIM_CalBIM_Exit:
    M_AFE_BIM_DISABLE                                   ;disable BIM
    call    F_AD_Change_To_NormalSpeed                  ;configure sigma-delta ADC
    call    F_AD_Lost3p
    call    F_AD_ClrSumBuf
    return
```

用户可以参考用户手册《DS_CSU18M9x_Vx.x_cn》关于 24bit Sigma Delta ADC 的应用场景和配置推荐。测量人体阻抗模式下，建议配置 **VS=2.8V/3.0V**、**BIM_DACFQ=50KHz**。注意配置 ANACFG 寄存器时需要确保 24b-ADC 处于关闭状态，VS 配置完毕需要延时一段时间，然后再使能 24b-ADC。

表 2-4 阻抗测量参考配置

应用场景	寄存器位	推荐配置	
人体阻抗	SDCKS	=1	SD_SMPCK=500KHz
	SDGAIN[1:0]	=00	GAIN=1
	SDREFS[1:0]	=00	SDADC 参考电压为（REFP-REFN）
	SDCHP[1:0]	=00	SDADC 斩波频率 1
	ADM[3:0]	=1111	降采样率 16384（3+2 阶 Comb）
	SDSINL[1:0]	=11	SDADC 输入信号来自片内人体测量模块

如果满足称重锁定、视重不为零（不清零、不去皮的称重范围）这两个条件，系统切换到阻抗测量状态，测量完毕返回称重模式。阻抗测量期间，称重处理无效。阻抗测量需要另外一个前置条件：初始化 BIM 模块。上电初始化时，调用 BIM 初始化函数，完成校准工作。

为了改善用户体验，需要加入上下阈值判断站姿异常或上下秤，还需要加入超时处理，确保可以返回称重模式。

人体阻抗的计算公式如下：

$$Z = \frac{(R\_AFE\_Foot\_AD - R\_AFE\_Short\_AD) \times (Res1 - Res0)}{(R\_AFE\_Res1\_AD - R\_AFE\_Res0\_AD - KB)}$$

1) Z：表示人体阻抗值
2) R_AFE_Foot_AD：表示人体阻抗 AD 平均值
3) R_AFE_Short_AD：表示内短 AD 平均值
4) Res1：表示 1KΩ校准电阻（如果 Z 分辨率为 0.1Ω，则表示为 10000）
5) Res0：表示 300Ω校准电阻（如果 Z 分辨率为 0.1Ω，则表示为 3000）
6) R_AFE_Res1_AD：表示 Res1=1KΩ校准电阻对应的 AD 平均值
7) R_AFE_Res0_AD：表示 Res0=300Ω校准电阻对应的 AD 平均值
8) KB：误差常数

从上述公式可以知道，R_AFE_Res1_AD、R_AFE_Res0_AD 已经在初始化得出相应的数值；KB 可以根据调试得出典型值，默认值为 0（如果想得出更精确的数值，则需要进行校准操作，本质就是上述公式的逆推）；R_AFE_Short_AD 为内短 AD 平均值，在测量人体阻抗前得出该数值；然后配置 BIM 相关的寄存器，取得人体阻抗的 AD 平均值 R_AFE_Foot_AD。

误差常数的计算公式如下：

$$KB = (R\_AFE\_Res1\_AD - R\_AFE\_Res0\_AD) - \frac{(R\_AFE\_Cal\_AD - R\_AFE\_Short\_AD) * (Res1 - Res0)}{Zcal}$$

1) Zcal：表示标准电阻值
2) R_AFE_Cal_AD：表示标准电阻值 AD 平均值

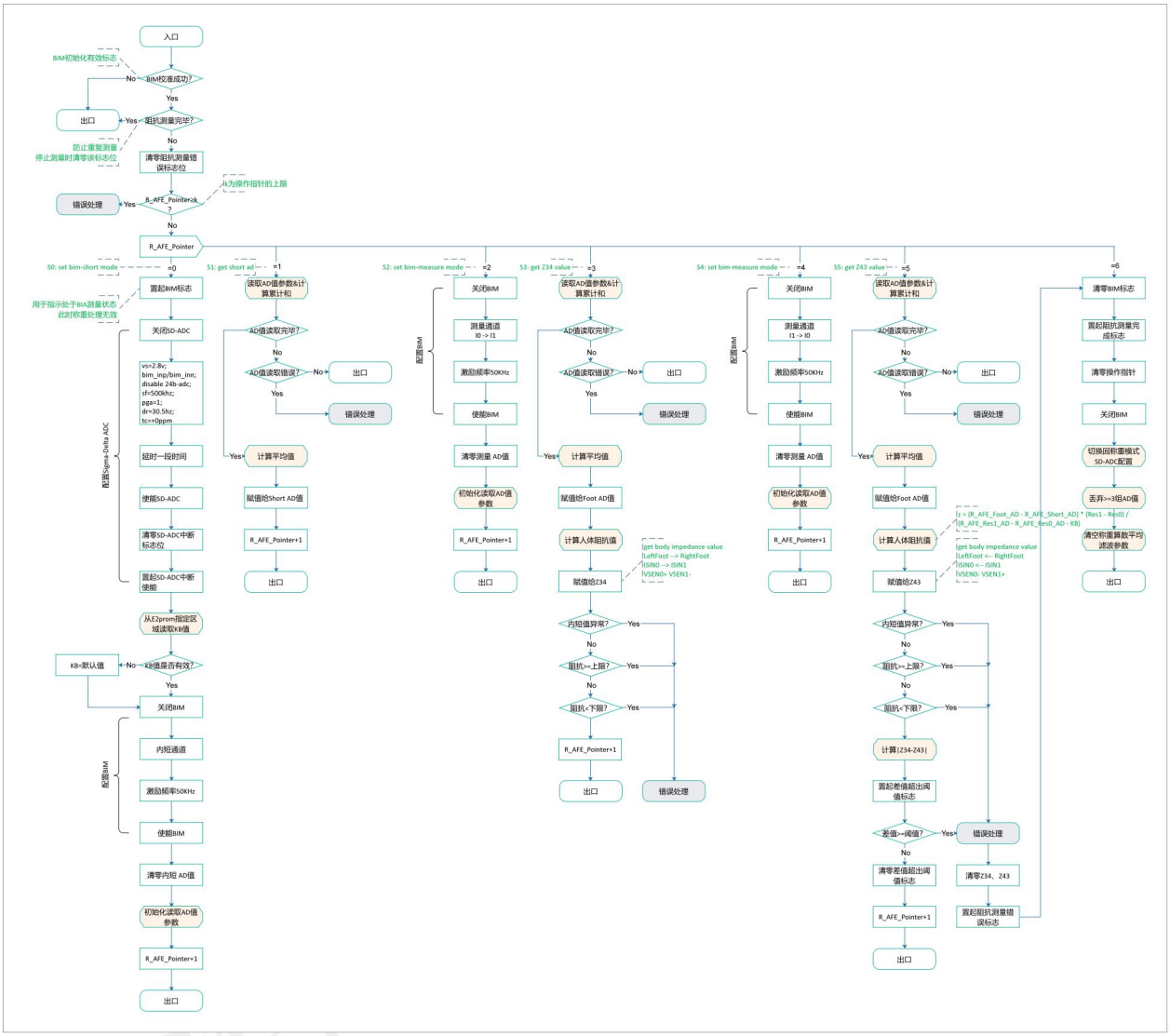


图 2-8 人体阻抗测量流程图

示例代码：

```
/*-------------------------------------------------------------------------
 Name    : F_Drv_BIM_BodyRes
 Describe: measure body-res & calc Z34/Z43
-------------------------------------------------------------------------*/
F_Drv_BIM_BodyRes:
    btfss   B_CalRes_Done
    goto    L_Drv_BIM_BodyRes_Exit
    btfsc   B_BodyRes_Done
    goto    L_Drv_BIM_BodyRes_Exit
    bcf     B_BodyRes_Err

    ;-----------------------------------------------
    movlw   7
    subwf   R_AFE_Pointer,0
    btfsc   STATUS,C
    goto    L_Drv_BIM_BodyRes_Err
```

```
    ;-------
    movfw   R_AFE_Pointer
    addpcw
    goto    L_Drv_BIM_BodyRes_S0        ;<0>
    goto    L_Drv_BIM_BodyRes_S1        ;<1>
    goto    L_Drv_BIM_BodyRes_S2        ;<2>
    goto    L_Drv_BIM_BodyRes_S3        ;<3>
    goto    L_Drv_BIM_BodyRes_S4        ;<4>
    goto    L_Drv_BIM_BodyRes_S5        ;<5>
    ;-------
    goto    L_Drv_BIM_BodyRes_Done      ;<n+1>

    ;-----------------------------------------------
    ; S0: set bim-short mode
    ;-----------------------------------------------
L_Drv_BIM_BodyRes_S0:
    bsf     B_BIM_Enable                        ;BIM work flag
    ;-------
    M_AFE_SDAD_DISABLE                          ;disable 24b-adc
    movlw   C_AFE_ANACFG                        ;vs=2.8v; bim_inp/bim_inn;
disable 24b-adc
    movwf   ANACFG
    movlw   C_AFE_SDCFG                         ;sf=500khz; pga=1
    movwf   SDCFG
    movlw   C_AFE_SDCON_DR_30HZ                 ;dr=30.5hz
    movwf   SDCON
    movlw   C_AFE_TEMPC                         ;tc=+0ppm
    movwf   TEMPC
    movlw   0xFF                                ;<delay>
    decfsz  WORK,F
    goto    $-1
    M_AFE_SDAD_ENABLE                           ;enable 24b-adc
    bcf     INTF,SDIF
    bsf     INTE,GIE                            ;enable global interrupt
    bsf     INTE,SDIE                           ;enable 24b-adc interrupt
    ;-------
    ;R_Math_A0 ->> R_AFE_Cal_KL
    ;R_Math_A1 ->> R_AFE_Cal_KH
    ;R_Math_A2 ->  C_E2PROM_DATA_CHECK
    M_DRV_E2PROM_DATA_READ  C_E2PROM_KB_ADDRH, C_E2PROM_KB_ADDRL, C_E2PROM_KB_CNT,
R_Math_A0, 1
    ;-------
    movfw   R_Math_A0
    movwf   R_AFE_Cal_KL
    movfw   R_Math_A1
    movwf   R_AFE_Cal_KH
    movlw   C_E2PROM_DATA_CHECK
    xorwf   R_Math_A2,0
```

```
    btfsc   STATUS,Z
    goto    L_Drv_BIM_BodyRes_S0_RdCalKB_Done
    movlw   C_AFE_VAL_KB_L                          ;<default KB value>
    movwf   R_AFE_Cal_KL
    movlw   C_AFE_VAL_KB_H
    movwf   R_AFE_Cal_KH
L_Drv_BIM_BodyRes_S0_RdCalKB_Done:
    ;-------

    M_AFE_BIM_DISABLE                               ;disable BIM
    movlw   C_AFE_BIM0_SHORT                        ;mode -> short of BIM
    movwf   BIM0
    movlw   C_AFE_BIM1_50K                          ;frequecy -> 50KHz
    movwf   BIM1
    M_AFE_BIM_ENABLE                                ;enable BIM
    ;-------
    clrf    R_AFE_Short_ADL
    clrf    R_AFE_Short_ADH
    call    F_Drv_BIM_Init_Filter_RAM_Body
    incf    R_AFE_Pointer,1                         ;<R_AFE_Pointer+1>
    goto    L_Drv_BIM_BodyRes_Exit

    ;-----------------------------------------------L_Drv_BIM_BodyRes_Short
    ; S1: get short ad
    ;-----------------------------------------------
L_Drv_BIM_BodyRes_S1:
    call    F_Drv_BIM_Filter_AD                     ;read ad & calc sum
    btfsc   B_BIM_AD_Done
    goto    L_Drv_BIM_BodyRes_S1_Done
    btfss   B_BIM_AD_Err
    goto    L_Drv_BIM_BodyRes_Exit
    goto    L_Drv_BIM_BodyRes_Err
    ;-------
L_Drv_BIM_BodyRes_S1_Done:
    call    F_Drv_BIM_AD_SUM_RRF2                   ;/4
    call    F_Drv_BIM_AD_SUM_RRF1                   ;/2
    movfw   R_AFE_SumH
    movwf   R_AFE_Short_ADH
    movfw   R_AFE_SumM
    movwf   R_AFE_Short_ADL
    incf    R_AFE_Pointer,1                         ;<R_AFE_Pointer+1>
    goto    L_Drv_BIM_BodyRes_Exit

    ;-----------------------------------------------
    ; S2: set bim-measure mode
    ;-----------------------------------------------
L_Drv_BIM_BodyRes_S2:
    M_AFE_BIM_DISABLE                               ;disable BIM
```

```
    movlw   C_AFE_BIM0_MEASURE_LR                   ;mode -> measure(LR) of BIM
    movwf   BIM0
    movlw   C_AFE_BIM1_50K                          ;frequecy -> 50KHz
    movwf   BIM1
    M_AFE_BIM_ENABLE                                ;enable BIM
    ;-------
    clrf    R_AFE_Foot_ADL
    clrf    R_AFE_Foot_ADH
    call    F_Drv_BIM_Init_Filter_RAM_Body
    incf    R_AFE_Pointer,1                         ;<R_AFE_Pointer+1>
    goto    L_Drv_BIM_BodyRes_Exit

    ;-----------------------------------------------
    ; S3: get Z34 value
    ;-----------------------------------------------
    ; LeftFoot --> RightFoot
    ; ISIN0    --> ISIN1
    ; VSEN0+       VSEN1-
    ;-----------------------------------------------
L_Drv_BIM_BodyRes_S3:
    call    F_Drv_BIM_Filter_AD                     ;read ad & calc sum
    ;-------
    ;jdg body still on scale? ad-value?
    ;-------
    btfsc   B_BIM_AD_Done
    goto    L_Drv_BIM_BodyRes_S3_Done
    btfss   B_BIM_AD_Err
    goto    L_Drv_BIM_BodyRes_Exit
    goto    L_Drv_BIM_BodyRes_Err
    ;-------
L_Drv_BIM_BodyRes_S3_Done:
    call    F_Drv_BIM_AD_SUM_RRF2                   ;/4
    call    F_Drv_BIM_AD_SUM_RRF1                   ;/2
    movfw   R_AFE_SumH
    movwf   R_AFE_Foot_ADH
    movfw   R_AFE_SumM
    movwf   R_AFE_Foot_ADL
    call    F_Drv_BIM_Calc_BodyRes                 ;calc body-res Z34
    movfw   R_Math_A0
    movwf   R_AFE_Z34_L
    movfw   R_Math_A1
    movwf   R_AFE_Z34_H
    ;-------
    btfsc   B_BodyResErr_Short
    goto    L_Drv_BIM_BodyRes_Err
    btfsc   B_BodyResErr_High
    goto    L_Drv_BIM_BodyRes_Err
    btfsc   B_BodyResErr_Low
```

```
    goto    L_Drv_BIM_BodyRes_Err
    ;-------
    incf    R_AFE_Pointer,1                         ;<R_AFE_Pointer+1>
    goto    L_Drv_BIM_BodyRes_Exit

    ;-----------------------------------------------
    ; S4: set bim-measure mode
    ;-----------------------------------------------
L_Drv_BIM_BodyRes_S4:
    M_AFE_BIM_DISABLE                               ;disable BIM
    movlw   C_AFE_BIM0_MEASURE_RL                   ;mode -> measure(RL) of BIM
    movwf   BIM0
    movlw   C_AFE_BIM1_50K                          ;frequecy -> 50KHz
    movwf   BIM1
    M_AFE_BIM_ENABLE                                ;enable BIM
    ;-------
    clrf    R_AFE_Foot_ADL
    clrf    R_AFE_Foot_ADH
    call    F_Drv_BIM_Init_Filter_RAM_Body
    incf    R_AFE_Pointer,1                         ;<R_AFE_Pointer+1>
    goto    L_Drv_BIM_BodyRes_Exit

    ;-----------------------------------------------
    ; S5: get Z43 value
    ;-----------------------------------------------
    ;LeftFoot <-- RightFoot
    ;ISIN0    <-- ISIN1
    ;VSEN0-       VSEN1+
    ;-----------------------------------------------
L_Drv_BIM_BodyRes_S5:
    call    F_Drv_BIM_Filter_AD                     ;read ad & calc sum
    ;-------
    ;jdg body still on scale? ad-value?
    ;-------
    btfsc   B_BIM_AD_Done
    goto    L_Drv_BIM_BodyRes_S5_Done
    btfss   B_BIM_AD_Err
    goto    L_Drv_BIM_BodyRes_Exit
    goto    L_Drv_BIM_BodyRes_Err
    ;-------
L_Drv_BIM_BodyRes_S5_Done:
    call    F_Drv_BIM_AD_SUM_RRF2                   ;/4
    call    F_Drv_BIM_AD_SUM_RRF1                   ;/2
    movfw   R_AFE_SumH
    movwf   R_AFE_Foot_ADH
    movfw   R_AFE_SumM
    movwf   R_AFE_Foot_ADL
    call    F_Drv_BIM_Calc_BodyRes
```

```
    movfw   R_Math_A0
    movwf   R_AFE_Z43_L
    movfw   R_Math_A1
    movwf   R_AFE_Z43_H
    ;-------
    btfsc   B_BodyResErr_Short
    goto    L_Drv_BIM_BodyRes_Err
    btfsc   B_BodyResErr_High
    goto    L_Drv_BIM_BodyRes_Err
    btfsc   B_BodyResErr_Low
    goto    L_Drv_BIM_BodyRes_Err
    ;-------
    movfw   R_AFE_Z34_L
    movwf   R_Math_B0
    movfw   R_AFE_Z34_H
    movwf   R_Math_B1
    call    F_Lib_Math_Abs16                    ;R_Math_A1~0=|R_Math_A1~0 -
R_Math_B1~0|
    ;-------
    bsf     B_BodyResErr_Over
    movlw   C_AFE_RES_OR_L
    subwf   R_Math_A0,0
    movlw   C_AFE_RES_OR_H
    subwfc  R_Math_A1,0
    btfsc   STATUS,C
    goto    L_Drv_BIM_BodyRes_Err               ;|Z34-Z43|>=a, invalid value
    bcf     B_BodyResErr_Over                   ;|Z34-Z43|< a, valid value
    incf    R_AFE_Pointer,1                     ;<R_AFE_Pointer+1>
    goto    L_Drv_BIM_BodyRes_Exit

    ;-----------------------------------------------
L_Drv_BIM_BodyRes_Err:
    clrf    R_AFE_Z34_L                         ;Z34=0 & Z43=0 -> bodyres-err
    clrf    R_AFE_Z34_H
    clrf    R_AFE_Z43_L
    clrf    R_AFE_Z43_H
    ;-------
    bsf     B_BodyRes_Err

    ;-----------------------------------------------
L_Drv_BIM_BodyRes_Done:
    bcf     B_BIM_Enable                        ;BIM work flag
    bsf     B_BodyRes_Done
    clrf    R_AFE_Pointer
    M_AFE_BIM_DISABLE
    call    F_AD_Change_To_NormalSpeed
    call    F_AD_Lost3p
    call    F_AD_ClrSumBuf
```

```
    ;-----------------------------------------------
L_Drv_BIM_BodyRes_Exit:
    return
```

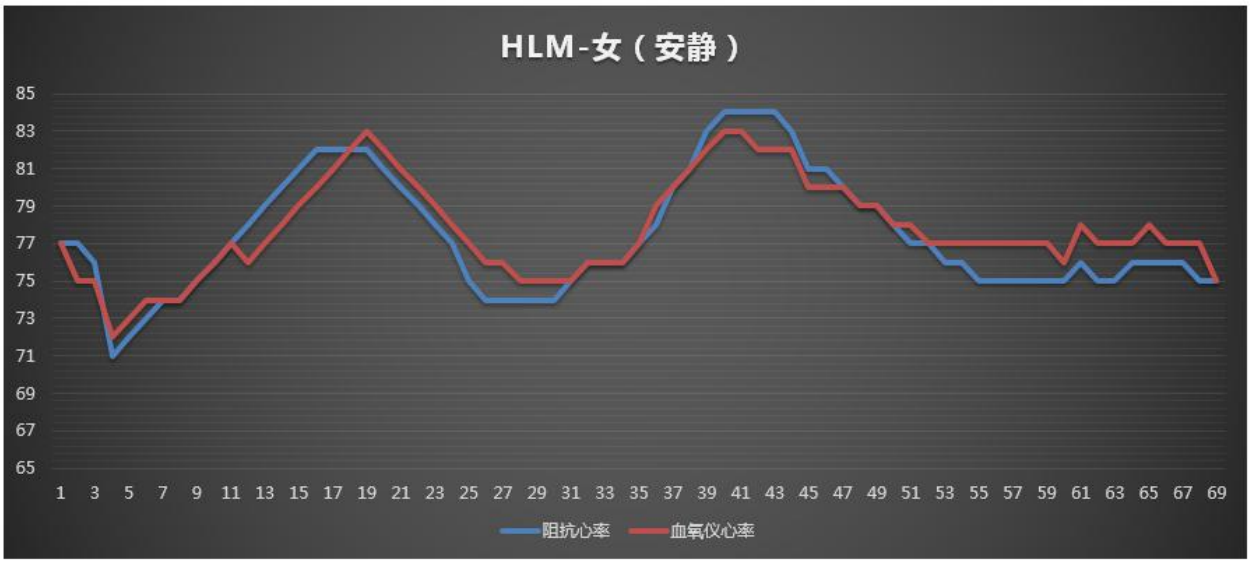
多频阻抗测量的做法是：先测完一个频率的阻抗，然后切换配置再测量下一个频率的阻抗，以此类推，直到全部频率的阻抗测量完毕。

需要注意两个事项：（1）切换配置后需要丢弃前几笔 AD 值；（2）需要获得不同频率下的校准电阻 AD 平均值，用于人体阻抗的计算。

由于多频阻抗测量耗时较长，为了提高用户体验，可以加入上下阈值判断站姿异常或上下秤，确保可以及时返回称重模式。

2.3.3 心率测量

CSU18M91 支持心率测量功能。测量原理是：心脏跳动引起的血液流动导致阻抗微小变化，系统利用内置模拟前端测量两脚之间阻抗微小的动态变化，通过心率算法从阻抗变化提取心率信号。



用医疗级脉搏血氧仪，男女各6人，运动及安静状态对比测试，综合误差<=+/-5bpm。

图 2-9 心率对比测试示意图

由于 CSU18M91 内置一个 24bit Sigma-Delta ADC，因此体重测量、阻抗测量与心率测量需要串行处理，一般而言，体重锁定后切换至阻抗测量，测量完成后再进行心率测量，最后返回称重模式。

表 2-5 心率测量参考配置

应用场景	寄存器位	推荐配置	
人体心率	SDCKS	=1	SD_SMPCK=500KHz
	SDGAIN[1:0]	=00	GAIN=1
	SDREFS[1:0]	=00	SDADC 参考电压为（REFP-REFN）
	SDCHP[1:0]	=00	SDADC 斩波频率 1
	ADM[3:0]	**=1101**	**Data Rate 122Hz**
	SDSINL[1:0]	=11	SDADC 输入信号来自片内人体测量模块

测量人体心率模式下，建议配置 **VS=2.8V/3.0V**、**BIM_DACFQ=25/50KHz**。注意配置 ANACFG 寄存器时需要确保 24b-ADC 处于关闭状态，VS 配置完毕需要延时一段时间，然后再使能 24b-ADC。

对于心率测量应用，24bit Sigma Delta ADC 的输出速率一般采用 61Hz ~ 244Hz，具体配置根据心率算法的需求来确定。如果采用 CSU18M91（内置）心率算法则采用 **122Hz**。为了取得更好的信噪比，正弦电流输出频率一般选择 **BIM_DACFQ=25/50KHz**。

由于心率信号非常微弱，而且是叠加在阻抗信号上的，因此不能使用 PGA 进行信号放大，否则会导致信号溢出。此时可以使能 METCH[4]=1（BIM Gain=2），心率测量完毕关闭 METCH[4]=0。

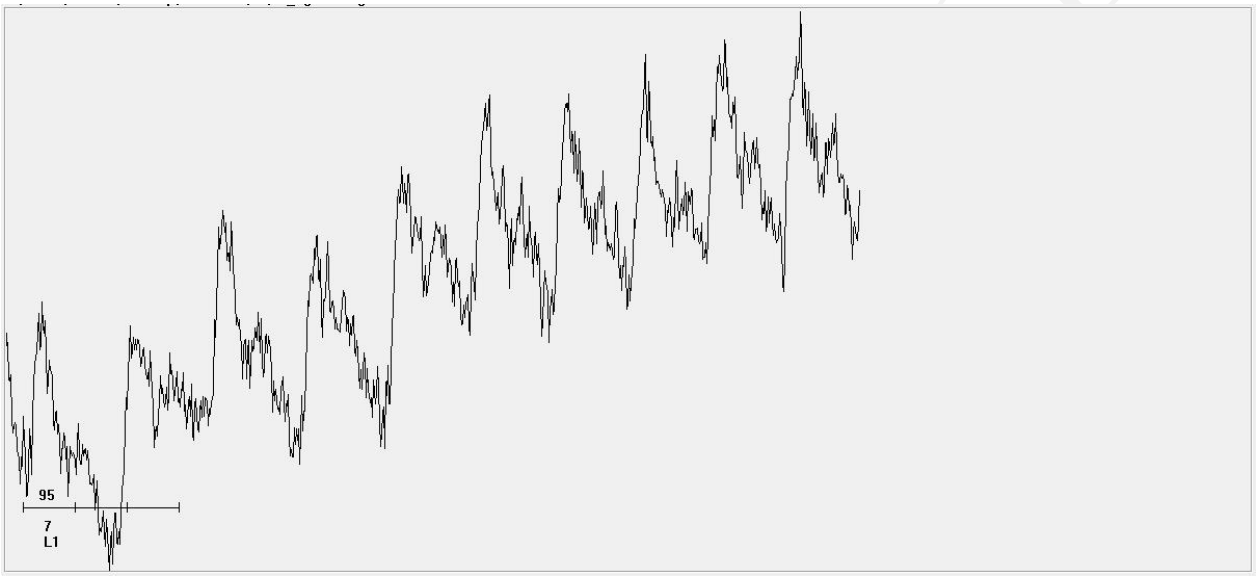


图 2-10 心率原始信号

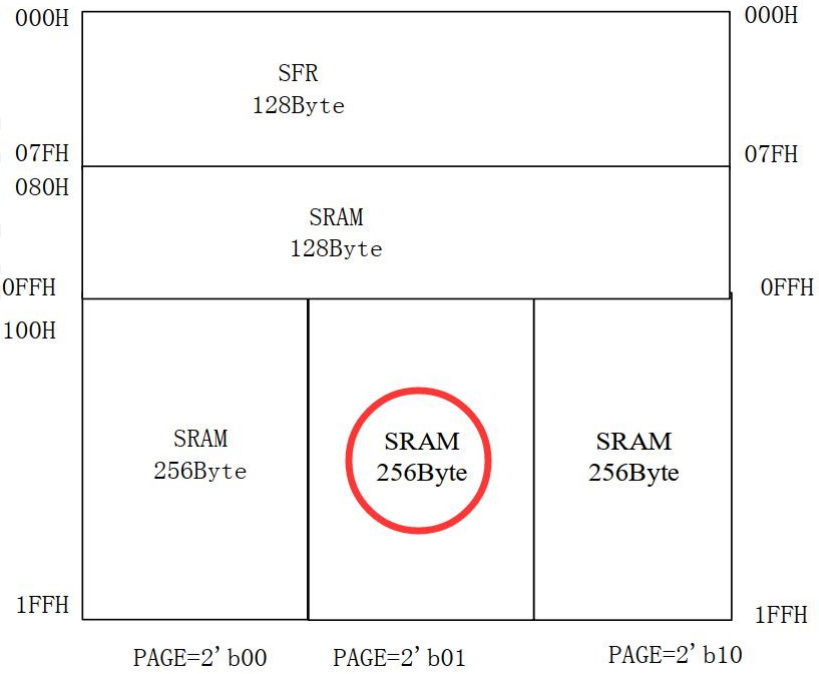


图 2-11 心率算法 SRAM 区域

由于心率算法使用 Bank2 SRAM(PAGE=2'b01)，因此用户需要注意应用代码不要与心率算法冲突，导致算法产生错误。参考代码详见源文件 **cs_drv_bim.asm/cs_drv_bim.inc/cs_app_bia_measure.asm**。

心率测量主要涉及 3 个子函数：

1) F_Drv_BIM_Suspend

-- 清零 BIM 测量的相关标志位与操作指针。

2) F_Drv_BIM_HeartRate

-- 采集心率数据和阈值判断。如果测量完成或测量异常，则置起相应的标志位并退出该函数。

3) F_Lib_HeartRate_Process

-- 心率算法库函数。系统通过" F_Drv_BIM_HeartRate"函数采集心率 AD 值，然后调用心率算法库函数" F_Lib_HeartRate_Process"即可。为了防止心率算法出现异常，调用算法需要满足一系列前置条件，详见图 2-12 心率测量流程图。

由于心率算法的变量是使用 Bank2 区域的 SRAM，因此为了便于参数传递，**算法的输入、输出变量必须使用 Bank0 的变量**。需要注意的是**变量的名称不能随意更改。**

1) R_HeartRate_ADL/ R_HeartRate_ADH/ R_HeartRate_ADHH

-- 输入变量，心率 AD 值。"F_Drv_BIM_HeartRate"函数首先配置心率测量模式，然后初始化心率算法，再把经过预处理的心率 AD 值赋给指定的算法输入变量"R_HeartRate_ADL / R_HeartRate_ADH / R_HeartRate_ADHH"。 用户可以不关注该变量。

2) R_HeartRate

-- 输出变量，心率值。心率算法库函数" F_Lib_HeartRate_Process" 在 5s~20s 内输出心率值，如果心率信噪比良好，一般 7s 后即可输出心率值。

3) R_HeartRate_Level

-- 输出变量，心率值等级。用于评判心率信号质量的指标，=1 表示信号质量优异，=2 表示信号质量良好，=3 表示信号质量一般。用户可以不关注该变量。

4) B_HeartRate_En

-- 标志位，心率数据有效。

5) B_HeartRate_Done

-- 标志位，心率测量完成。无论是否心率异常，测量结束时都会置起该标志位表示测量完成。

6) B_HeartRate_Err

-- 标志位，心率测量错误。

系统可以根据标志位"B_HeartRate_Done"是否为 1，来判断心率测量是否完成，然后再判断标志位"B_HeartRate_Err"是否为 0，如果为 0 表示心率测量数据有效。

标志位"B_HeartRate_En"表示心率数据有效，允许调用心率算法函数"F_Lib_HeartRate_Process"。

由于心率测量时间较长，因此系统需要调整超时（关机）时间，防止测量过程中进入待机状态。至于心率测量允许条件，用户可以根据需求自定义，如果不满足心率测量条件则调用测量终止函数”F_Drv_BIM_Suspend”（@cs_drv_bim.asm），清零测量标志与操作指针，以便于下一次测量。

如果满足称重锁定、视重不为零、阻抗测量完成这些前置条件，系统切换到心率测量状态，测量完毕返回称重模式。心率测量期间，称重处理无效。

当视重为零时，系统清零心率测量完成标志位与操作指针，用于下一次测量。

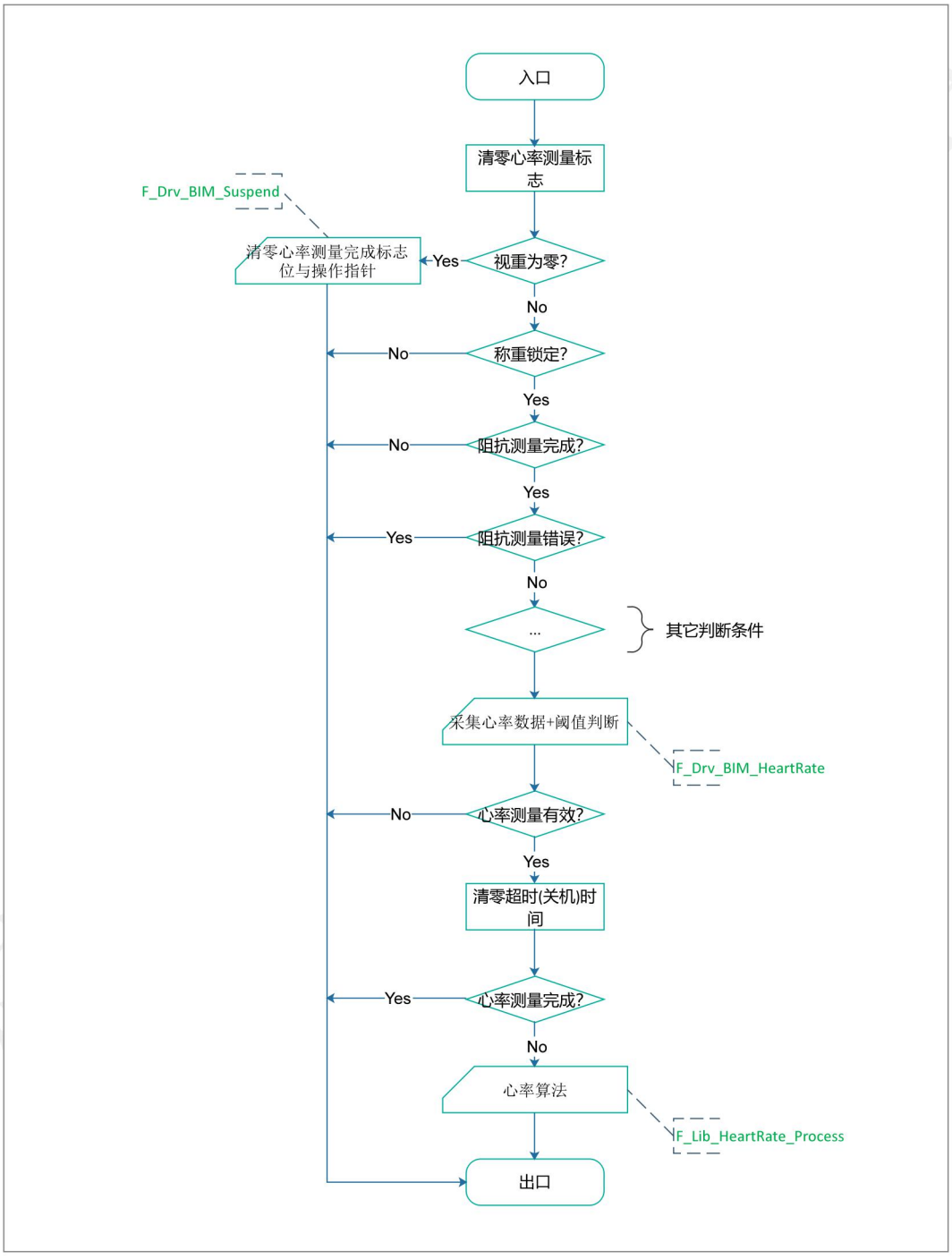


图 2-12 心率测量流程图

示例代码：

```
/*---------------------------------------------------------------------
 Name    : F_App_HeartRate_Measure
 Describe: measure human-
body resting heart rate; suspend measure when exceeding threshold
 ---------------------------------------------------------------------
```

```
 Input   :
    B_BS_IsZero             --> weight value zero flag (w < zero threshold)
    B_BS_Lock               --> weight value lock flag
    B_BodyRes_Done          --> completion flag
    B_BodyRes_Err           --> error flag
    B_Dsp_Flag_Blink_Start  --> weight value lock fliker flag
 Output  :
    B_BIM_Enable            --> bim function enable flag
    B_HeartRate_Done        --> completion flag
    B_HeartRate_Err         --> error flag
    R_HeartRate             --> resting heart rate value
    R_TimeOut_TimeBase           --> time-
out counter (clear it when measuring heart-rate)
 Temp    :
    B_HeartRate_En          --> heart-rate measuring flag
    R_HeartRate_ADL,R_HeartRate_ADH     --> valid heart-rate adc value
-------------------------------------------------------------------------*/
; for main-loop, after impedance function
.if PRE_OPTION_BIA_HARTRATE
F_App_HeartRate_Measure:
    bcf     B_HeartRate_En                  ;clear heart-rate data enable flag

    ;-----------------------------------------------------------------
    ; heart-rate measuring condition
    ;-----------------------------------------------------------------
    ;pre-conditions
    btfss   B_BS_IsZero
    goto    L_App_HeartRate_Measure_Valid
    call    F_Drv_BIM_Suspend
    goto    L_App_HeartRate_Measure_Exit
    ;-------
L_App_HeartRate_Measure_Valid:
    btfss   B_BS_Lock                       ;after weighing lock
    goto    L_App_HeartRate_Measure_Exit

    ;-----------------------------------------------------------------
.if PRE_OPTION_BIA_IMPEDANCE
    btfss   B_BodyRes_Done                  ;after human-body impedance measuring
    goto    L_App_HeartRate_Measure_Exit
    btfsc   B_BodyRes_Err
    goto    L_App_HeartRate_Measure_Exit
.endif

    ;-----------------------------------------------------------------
    ; heart rate measure
    ;-----------------------------------------------------------------
    call    F_Drv_BIM_HeartRate             ;@cs_drv_bim.asm
    ;-------
```

```
    btfsc   B_HeartRate_En
    clrf    R_TimeOut_TimeBase              ;clear time-out register
    ;-------
    btfss   B_HeartRate_En                  ;
    goto    L_App_HeartRate_Measure_Exit
    btfsc   B_HeartRate_Done
    goto    L_App_HeartRate_Measure_Exit

    ;---------------------------------------------------------------
    ; Input: R_HeartRate_ADL, R_HeartRate_ADH, R_HeartRate_ADHH
    ; Output: R_HeartRate, R_HeartRate_Level
    ;---------------------------------------------------------------
    call    F_Lib_HeartRate_Process           ;<heart-rate algorithm>

    ;---------------------------------------------------------------
L_App_HeartRate_Measure_Exit:
    return
.endif
```

心率算法的输入变量是”R_HeartRate_ADL”、”R_HeartRate_ADH”、”R_HeartRate_ADHH”，输出变量为” R_HeartRate”、”R_HeartRate_Level”。其中用户只需关注输出变量”R_HeartRate”和标志位”B_HeartRate_Done”、 ”B_HeartRate_Err”即可。

要实现心率数据采集，首先需要进行配置切换，具体步骤可以参考流程图与示例代码。需要注意的是输出速率要与心率算法要求的一致，否则会导致结果错误。

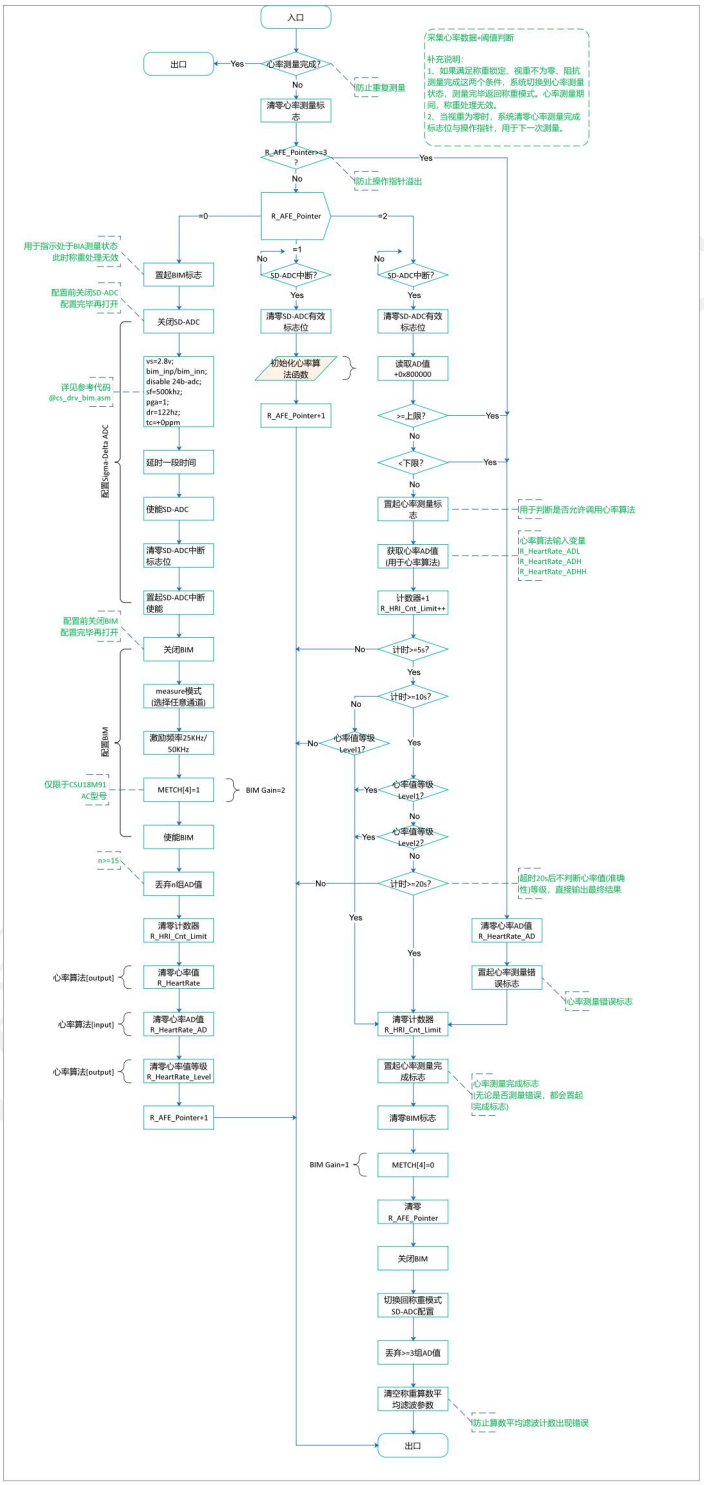


图 2-13 采集心率数据(含阈值判断)流程图

由于测量人体心率时间较长(典型值 10s)，所以可以根据应用场景(例如单脚站立、穿鞋测量等异常站姿)加入上下阈值的判断，阈值根据经验值或调试值来设置。如果测量值超出阈值则返回称重模式。

以下阈值仅供参考，用户应根据实际需求调整上下阈值。

示例代码：

```
#define     C_HR_UPPER_LIMIT_H      0xAA
#define     C_HR_UPPER_LIMIT_M      0x0F
#define     C_HR_UPPER_LIMIT_L      0xE0

#define     C_HR_LOWER_LIMIT_H      0x8B
#define     C_HR_LOWER_LIMIT_M      0x47
#define     C_HR_LOWER_LIMIT_L      0x00
```

示例代码：

```
/*-----------------------------------------------------------------------
 Filename: cs_drv_bim.asm
 Describe: hri driver function
 Chip    : csu18m91
 Author  : chipsea
 Version : v1.0
 Date    : 2019-11-11
-----------------------------------------------------------------------*/
.if PRE_OPTION_BIA_HARTRATE
;-----------------------------------------------------------------------
; define ram @000h~0ffh
;-----------------------------------------------------------------------
;<bank0 for cs_lib_heart_rate>
;<don't change these variables's name>
CSCC_DRV_HRI_RAM            .section        bank0    ;<bank0 limit>
    R_HeartRate             .ds             1        ;heart rate value, <output>
    R_HeartRate_ADL         .ds             1        ;hri adc value, <input>
    R_HeartRate_ADH         .ds             1
    R_HeartRate_ADHH        .ds             1
    R_HeartRate_Level       .ds             1        ;confidence level, <output>,
 <1=excellent, 2=good, 3=commonly>
    ;
    R_HRI_Flag              .ds             1
    R_HRI_Cnt_LimitL        .ds             1        ;measure counter
    R_HRI_Cnt_LimitH        .ds             1
.ends
.endif
/*-----------------------------------------------------------------------
 Name    : F_Drv_BIM_HeartRate
 Describe: heart-rate data
 ----------------------------------------------------------------------
 Input   :
    B_HeartRate_Done
 Output  :
    B_HeartRate_Done;
    B_HeartRate_Err;
    B_BIM_Enable;
```

```
    R_HeartRate_ADL,R_HeartRate_ADH,R_HeartRate_ADHH;
 Temp    :
    R_AFE_Pointer;
    R_HRI_Cnt_LimitL,R_HRI_Cnt_LimitH;
    R_AFE_ADLL,R_AFE_ADL,R_AFE_ADH;
-----------------------------------------------------------------------*/
.if PRE_OPTION_BIA_HARTRATE
F_Drv_BIM_HeartRate:
    btfsc   B_HeartRate_Done
    goto    L_Drv_BIM_HeartRate_Exit
    bcf     B_HeartRate_Err

    movlw   3
    subwf   R_AFE_Pointer,0
    btfsc   STATUS,C
    goto    L_Drv_BIM_HeartRate_Err

    movfw   R_AFE_Pointer
    addpcw
    goto    L_Drv_BIM_HeartRate_S0                  ;<0>
    goto    L_Drv_BIM_HeartRate_S1                  ;<1>
    goto    L_Drv_BIM_HeartRate_S2                  ;<2>

    ;-----------------------------------------------
    ; S0: set heart rate measure mode
    ;-----------------------------------------------
L_Drv_BIM_HeartRate_S0:
    bsf     B_BIM_Enable                            ;BIM working flag

    M_AFE_SDAD_DISABLE                              ;disable 24b-adc
    movlw   C_AFE_ANACFG                            ;vs=2.8v; bim_inp/bim_inn; di
sable 24b-adc
    movwf   ANACFG
    movlw   C_AFE_SDCFG                             ;sf=500khz; pga=1
    movwf   SDCFG
    movlw   C_AFE_SDCON_DR_120HZ                    ;dr=122hz
    movwf   SDCON
    movlw   C_AFE_TEMPC                             ;tc=+0ppm
    movwf   TEMPC
    call    F_Lib_Delay_1ms                         ;<@cs_lib_delay_time>
    M_AFE_SDAD_ENABLE                               ;enable 24b-adc
    bcf     INTF,SDIF
    bsf     INTE,GIE                                ;enable global interrupt
    bsf     INTE,SDIE                               ;enable 24b-adc interrupt

    M_AFE_BIM_DISABLE                               ;disable BIM
    movlw   C_AFE_BIM0_MEASURE_LR                   ;mode -> measure(LR) of BIM
    movwf   BIM0
```

```
    movlw   C_AFE_BIM1_50K
    movwf   BIM1
    bsf     METCH,4                         ;<double differential signal>
    M_AFE_BIM_ENABLE                        ;enable BIM

    ;<at least 6 times>
    call    F_AD_Lost3p                     ;drop 3 times adc-
value @cs_drv_adc.asm
    call    F_AD_Lost3p
    call    F_AD_Lost3p
    call    F_AD_Lost3p
    call    F_AD_Lost3p
    call    F_AD_Lost3p
    call    F_AD_Lost3p

    clrf    R_HRI_Cnt_LimitL
    clrf    R_HRI_Cnt_LimitH

    clrf    R_HeartRate
    clrf    R_HeartRate_ADL
    clrf    R_HeartRate_ADH
    clrf    R_HeartRate_ADHH
    clrf    R_HeartRate_Level

    incf    R_AFE_Pointer,F                 ;<R_AFE_Pointer+1>
    goto    L_Drv_BIM_HeartRate_Exit

    ;-----------------------------------------------
    ; S1: get base line value
    ;-----------------------------------------------
L_Drv_BIM_HeartRate_S1:
    btfss   B_INT_GetADC                    ;<cs_drv_adc.inc>
    goto    $-1
    bcf     B_INT_GetADC
    call    F_Lib_HeartRate_Init            ;<init heart-rate algorithm>
    incf    R_AFE_Pointer,F                 ;<R_AFE_Pointer+1>
    goto    L_Drv_BIM_HeartRate_Exit

    ;-----------------------------------------------
    ; S2: get heart rate ad
    ;-----------------------------------------------
L_Drv_BIM_HeartRate_S2:
    btfss   B_INT_GetADC                    ;<cs_drv_adc.inc>
    goto    $-1
    bcf     B_INT_GetADC

    movfw   SDOH
    addlw   0x80                            ;+0x800000
```

```
    movwf   R_AFE_ADH
    movfw   SDOL
    movwf   R_AFE_ADL
    movfw   SDOLL
    movwf   R_AFE_ADLL

    ;-------
    movlw   C_HR_UPPER_LIMIT_L                      ;<upper limit>
    subwf   R_AFE_ADLL,W
    movlw   C_HR_UPPER_LIMIT_M
    subwfc  R_AFE_ADL,W
    movlw   C_HR_UPPER_LIMIT_H
    subwfc  R_AFE_ADH,W
    btfsc   STATUS,C
    goto    L_Drv_BIM_HeartRate_Err

    movlw   C_HR_LOWER_LIMIT_L                      ;<lower limit>
    subwf   R_AFE_ADLL,W
    movlw   C_HR_LOWER_LIMIT_M
    subwfc  R_AFE_ADL,W
    movlw   C_HR_LOWER_LIMIT_H
    subwfc  R_AFE_ADH,W
    btfss   STATUS,C
    goto    L_Drv_BIM_HeartRate_Err

    bsf     B_HeartRate_En                          ;measuring heart-rate flag

    movfw   R_AFE_ADLL
    movwf   R_HeartRate_ADL
    movfw   R_AFE_ADL
    movwf   R_HeartRate_ADH
    movfw   R_AFE_ADH
    movwf   R_HeartRate_ADHH                        ;<heart rate valid value for
heart-rate algorithm>

    ;-------------------------------------------------
    movlw   0x01
    addwf   R_HRI_Cnt_LimitL,F
    movlw   0x00
    addwfc  R_HRI_Cnt_LimitH,F

    ;-------
    movlw   LOW     C_HR_CNT_LIMIT0
    subwf   R_HRI_Cnt_LimitL,W
    movlw   HIGH    C_HR_CNT_LIMIT0
    subwfc  R_HRI_Cnt_LimitH,W
    btfss   STATUS,C
    goto    L_Drv_BIM_HeartRate_Exit
```

```
    ;
    movlw   LOW     C_HR_CNT_LIMIT1
    subwf   R_HRI_Cnt_LimitL,W
    movlw   HIGH    C_HR_CNT_LIMIT1
    subwfc  R_HRI_Cnt_LimitH,W
    btfsc   STATUS,C
    goto    L_Drv_BIM_HeartRate_Ret                 ;C=1, >=10s
    ;                                               ;C=0, < 10s
    movlw   C_HR_HEARTRATE_LEVEL1
    xorwf   R_HeartRate_Level,W
    btfsc   STATUS,Z
    goto    L_Drv_BIM_HeartRate_Done
    goto    L_Drv_BIM_HeartRate_Exit
    ;
L_Drv_BIM_HeartRate_Ret:
    movlw   C_HR_HEARTRATE_LEVEL1
    xorwf   R_HeartRate_Level,W
    btfsc   STATUS,Z
    goto    L_Drv_BIM_HeartRate_Done                ;Z=1, =level1
    movlw   C_HR_HEARTRATE_LEVEL2
    xorwf   R_HeartRate_Level,W
    btfsc   STATUS,Z
    goto    L_Drv_BIM_HeartRate_Done                ;Z=1, =level2

    ;-------
    movlw   LOW     C_HR_CNT_LIMIT2
    subwf   R_HRI_Cnt_LimitL,0
    movlw   HIGH    C_HR_CNT_LIMIT2
    subwfc  R_HRI_Cnt_LimitH,0
    btfss   STATUS,C
    goto    L_Drv_BIM_HeartRate_Exit
    goto    L_Drv_BIM_HeartRate_Done

    ;-----------------------------------------------
L_Drv_BIM_HeartRate_Err:
    clrf    R_HeartRate_ADL
    clrf    R_HeartRate_ADH
    clrf    R_HeartRate_ADHH
    bsf     B_HeartRate_Err                         ;heart rate flag

    ;-----------------------------------------------
L_Drv_BIM_HeartRate_Done:
    clrf    R_HRI_Cnt_LimitL
    clrf    R_HRI_Cnt_LimitH
    bsf     B_HeartRate_Done
    ;-------
    bcf     B_BIM_Enable
    bcf     METCH,4                                 ;
```

```
    clrf    R_AFE_Pointer
    M_AFE_BIM_DISABLE
    call    F_AD_Change_To_NormalSpeed
    call    F_AD_Lost3p
    call    F_AD_ClrSumBuf

    ;-----------------------------------------------
L_Drv_BIM_HeartRate_Exit:
    return
.endif
```

2.3.4 温度测量

对于温度测量应用，VS 推荐使用 2.3V 或 2.4V 档位，VS 配置完毕需要延时一段时间让模拟电源稳定后再使能 24b-ADC。配置 ANACFG 寄存器时需要确保 24b-ADC 处于关闭状态。

当 24bit Sigma Delta ADC 从其它通道切换到温度传感器测量时，需要丢掉前三笔数据。

表 2-6 温度测量参考配置

应用场景	寄存器位	推荐配置	
人体秤	SDCKS	=1	SD_SMPCK=500KHz
	SDGAIN[1:0]	=00	GAIN=1
	SDREFS[1:0]	=00	SDADC 参考电压为（REFP-REFN）
	SDCHP[1:0]	=10	SDADC 斩波频率 2
	ADM[3:0]	=1111	降采样率 16384（3+2 阶 Comb）
	SDSINL[1:0]	=10	SDADC 输入信号来自片内温度传感器

温度传感器精度受限于工艺偏差，出厂时需要进行标定。用当前环境温度进行标定，记录标定温度和标定温度 AD 值。

温度标定计算公式（摄氏温标）：

$$\text{标定的每度码值变化值} = \frac{\text{标定温度 AD 值}}{(273.15 + \text{标定温度})}$$

$$\text{实际测量的温度} = \frac{\text{实际测量温度 AD 值}}{\text{标定的每度码值变化值}} - 273.15$$

2.3.5 LED 显示

使能 LED 显示前需要设置相应的端口为 SEG 模式。其中”SEGCON3”、” SEGCON4”寄存器分别控制 PT3* SEG 、PT4* SEG 功能，赋值”0xFF”表示使能 LED/LCD 驱动端口，赋值”0x00”表示禁止 LED/LCD 驱动端口。

需要注意的是，**PT3* SEG、PT4* SEG 只能整体控制，不能对单独的端口进行控制**。举例：如果 LED 驱动使用了 PT3.7 端口，则意味着 PT3.6 ~ PT3.0 端口此时不能作为 GPIO 使用。

示例代码：

```
;-----------------------------------------------------------------------
; define register value
;-----------------------------------------------------------------------
M_LED_IO_ENABLE     macro
    movlw   C_LED_SEGCON3           ;=1, enable led driver port
    movwf   SEGCON3
    movlw   C_LED_SEGCON4
    movwf   SEGCON4
endm

/*-----------------------------------------------------------------------
 Name    : F_Drv_LED_Init
 Describe: init led driver registers
-----------------------------------------------------------------------*/
F_Drv_LED_Init:
    M_LED_IO_ENABLE                 ;enable led driver port
    ;-------
    M_LED_CURRENT_SETUP             ;set-up led driver current
    ;-------
    ;clear display buffers
    movlw   1
    movwf   R_LED_ADDRESS
L_Drv_LED_Init_Lp:
    movfw   R_LED_ADDRESS
    movwf   LCDADR
    movlw   0x00                    ;<NULL>
    movwf   LCDDAT
    incf    R_LED_ADDRESS,F
    movlw   15
    subwf   R_LED_ADDRESS,W
    btfss   STATUS,C
    goto    L_Drv_LED_Init_Lp
    ;-------
    movlw   C_LED_LEDENR
    movwf   LEDENR
    return
```

用户可以通过调节恒流源的大小来调节 LED 亮度。LED 驱动的原理是在每一路 LED_SEG*下拉恒定电流源，所有 LED_SEG*的恒流源都可以独立配置电流值， 6.25~10mA 之间，步长 0.25mA。自动扫描模式下，用户也可以通过调整占空比” LED_DUTY”来实现亮度的调节。

LED_SEG1 有四组寄存器配置电流，分置对应不同 COM。**多色指示灯可以放在 SEG1 端口，便于进行亮度调节，使其亮度一致**。

由于 LED 采用共阳方式驱动，因此休眠时建议把 LED_SEG 端口设为 GPIO 模式，输出高电平。

示例代码：

```
/*---------------------------------------------------------------------
 Name    : F_Drv_LED_Disable
 Describe: led driver disable
---------------------------------------------------------------------*/
F_Drv_LED_Disable:
    ;clear display buffers
    movlw   1
    movwf   R_LED_ADDRESS
L_Drv_LED_Disable_Lp:
    movfw   R_LED_ADDRESS
    movwf   LCDADR
    movlw   0x00                    ;<NULL>
    movwf   LCDDAT
    incf    R_LED_ADDRESS,F
    movlw   15
    subwf   R_LED_ADDRESS,W
    btfss   STATUS,C
    goto    L_Drv_LED_Disable_Lp
    ;-------
    ;disable led function
    clrf    LEDENR                  ;disable led driver
    M_LED_IO_DISABLE                ;disable led driver port
    ;-------
    ;close led gpio - high level
.if C_LED_SEGCON3 == C_LED_SEGCON3_SEG
    movlw   0b11111111
    movwf   PT3EN
    movlw   0b00000000
    movwf   PT3PU
    movlw   0b11111111
    movwf   PT3
.endif
.if C_LED_SEGCON4 == C_LED_SEGCON4_SEG
    movlw   0b11111111
    movwf   PT4EN
    movlw   0b00000000
    movwf   PT4PU
```

```
    movlw   0b11111111
    movwf   PT4
.endif
    return
```

2.3.6 电压测量

电压检测和低电报警可以使用 10Bit SAR-ADC 实现。电池电压经过比例分压（例如 1 : 2、1 : 3）进入模拟端口，SAR-ADC 参考电压默认采用 VDD。软件根据公式计算当前电压值，然后判断当前电压是否低于低压阈值，如果是则置起低电报警标志。

在 3V 电池（两节干电池或一节纽扣电池）应用中，由于不需要稳压的 LDO，因此系统电源 VDD 是随着电量下降而变低的。此时 SAR-ADC 可以配置成：参考电压选择 VS， 通道选择 DVDD/2。

电池电压计算公式：

$$\text{电池电压} = \frac{\text{参考电压值} \times \text{当前 AD 值}}{(2^{10}-1) \times \text{分压比例系数}}$$

由于 CSU18M9x 不支持浮点数运算，因此实际计算电压值时需要人为放大一定比例，防止计算损失精度。

$$\text{电池电压} \times \text{放大系数}\eta = \frac{\text{参考电压值} \times \text{当前 AD 值} \times \text{放大系数}\eta}{(2^{10}-1) \times \text{分压比例系数}}$$

为了提高电压测量准确性和稳定性，软件需要避开电流变化大的情况下采集 AD 值，防止出现电压波动明显的问题。其中一种处理方法是：软件在开启显示前进行电压测量，避免 LED 显示造成电源波动，同时每次测量电压时都要采集多组数据并剔除毛刺数据。注意每笔 AD 值之间需要间歇一小段时间，让 ADC 输入处于稳定状态。

如果 CSU18M9x 进入待机模式，检测电压的端口需要保持为模拟端口，不要设置为数字端口，防止电流倒灌导致功耗超标。

10Bit SAR-ADC 的操作步骤：

1) 配置 SAR-ADC 参考电压与输入通道
2) 使能 SAREN
3) 延时一段时间
4) 使能 SARSTART
5) 等待 SARSTART=0，然后读取 SARO[9:0]
6) 跳转到步骤 3)继续执行，直到完成额定次数的 AD 值采集
7) 关闭 SAREN

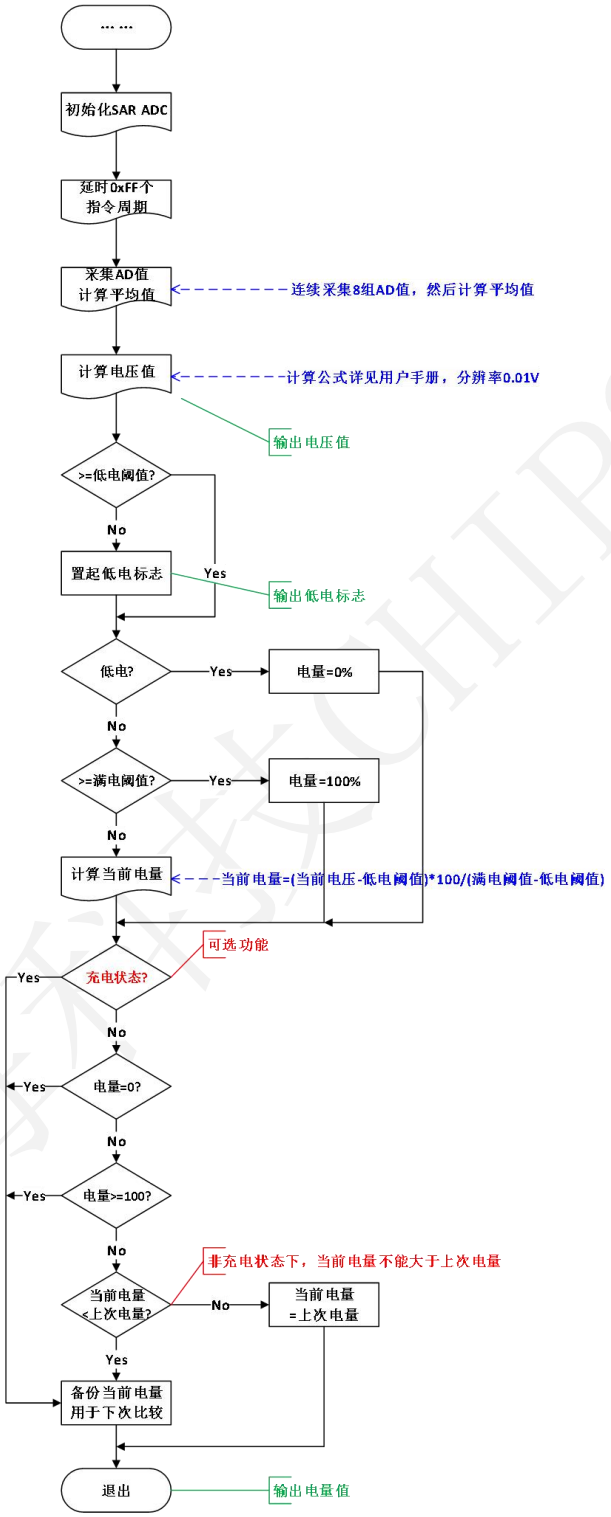


图 2-14 电量检测流程图

3 硬件设计

3.1 原理图设计

具体电路可以参考相关硬件资料。

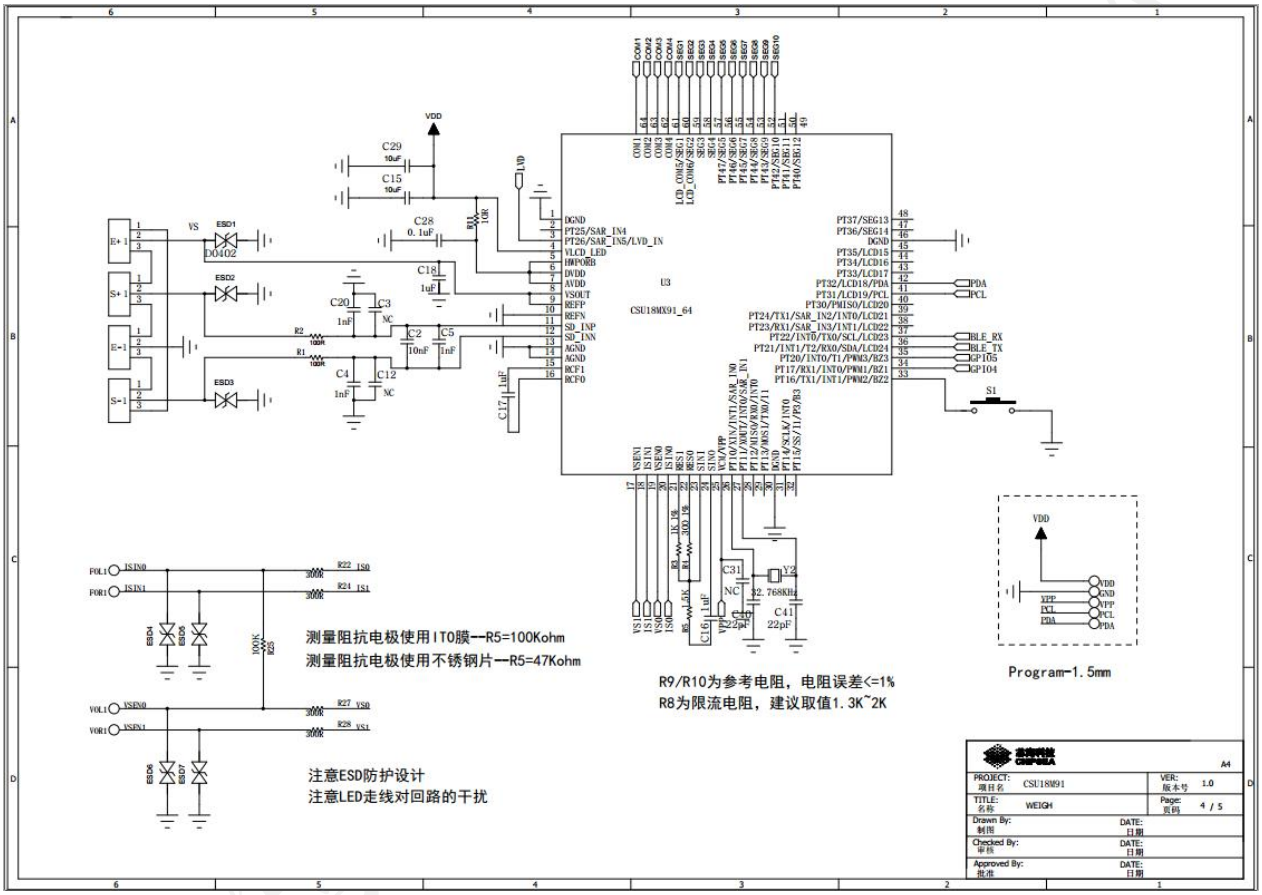


图 3-1 测量系统原理图(CSU18M91-LQFP64)

1、VLCD_LED 为 LCD/LED 驱动电路、LCD IO 的电压源。

LED 秤：VLCD_LED 接外部电源 2.4V ~ 3.6V，由于 LED 显示时会造成电源波动较大，建议使用 10μF 或更大容量的电容进行去耦。

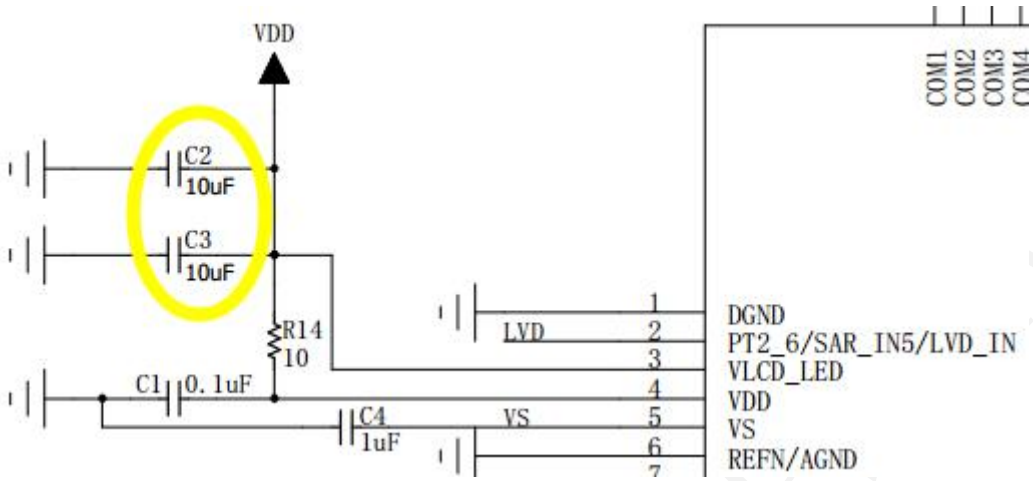


图 3-2 局部电路(a)

LCD 秤：**如果 LCD 驱动电压与 VDD 不一致，则 VLCD_LED 接 1PCS 1μF（典型值）电容到 GND，使能升压泵功能，提供电压源给 LCD IO**。寄存器配置详见用户手册《DS_CSU18M9x_Vx.x_cn》。

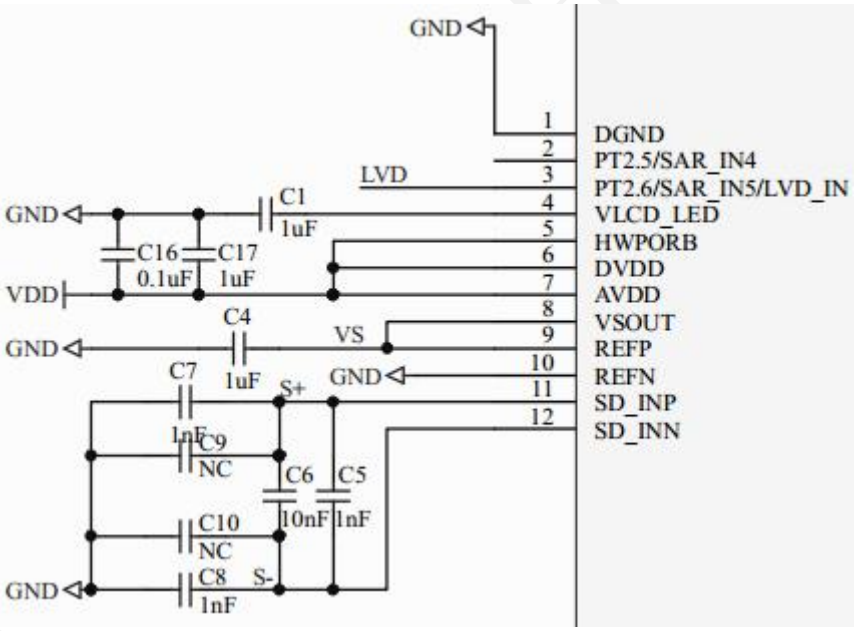


图 3-3 局部电路(b)

2、为了防止 LED 电源对芯片模拟电源干扰，可以考虑使用 R-C、L-C、FB-C 等低通滤波的方式滤除电源毛刺。（注：R 表示电阻、L 表示电感、FB 表示磁珠、C 表示电容）

用户可以根据实际情况选择合适的旁路参数和去耦参数。

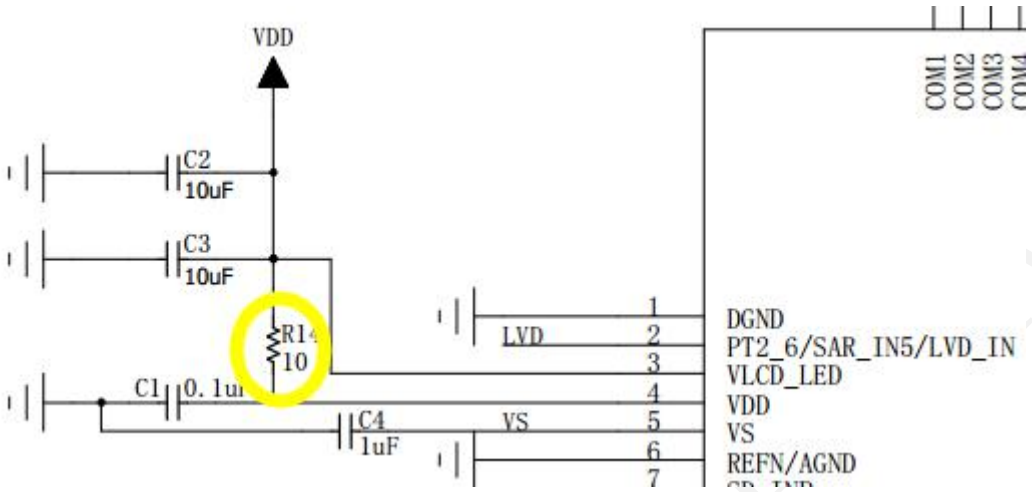


图 3-4 局部电路(c)(参数仅供参考)

3、**HWPORB 为外部复位引脚，如果不需要用到该功能，直接接到 VDD 即可**。如果方案需要预留 RST 引脚（例如锂电池方案，防止系统死机），则可以单独引出 HWPORB 引脚，HWPORB 引脚通过 10KΩ上拉电阻接到 VDD，然后接 1μF（典型值）电容到 GND。当按键按下时 HWPORB 变成低电平，松开按键后完成复位动作。**HWPORB 引脚不能悬空处理**。

其中**裸片/LQFP64 支持 HWPORB 引脚**；LQFP48 受限于管脚数量，不支持 HWPORB 引脚。

对于引出 HWPORB 引脚的芯片烧录的时候需要把该引脚上拉到电源才能进行烧录。

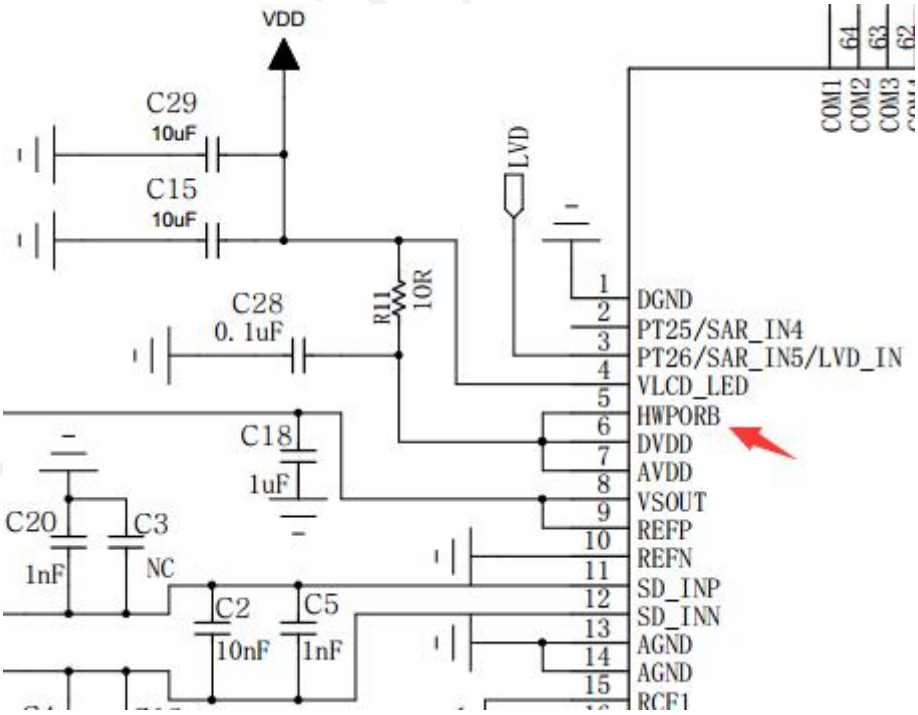


图 3-5 局部电路(d)

4、模拟输入滤波电路可以参考下图，采用 RC 滤波可以改善 ESD、RS 干扰的影响，共模滤波电容使用 C7=C8 =1nF，差模滤波电容使用 C6=10nF//C5=1nF。如果系统成本比较紧张，可以去除 R1、R2、C9、C10 和 C5；如果系统成本允许，可以添加 C9=C10= 10nF。

VS 为传感器激励源和电压参考源，至少需要 1PCS 1μF ~ 2.2μF 电容滤波。对于人体秤应用，VS 滤波电容不宜取值太大，否则休眠检重时会增加功耗。下图中 C5/C6/C7/C8/C9/C10 是典型参考值，实际应根据滤波效果调整取值。

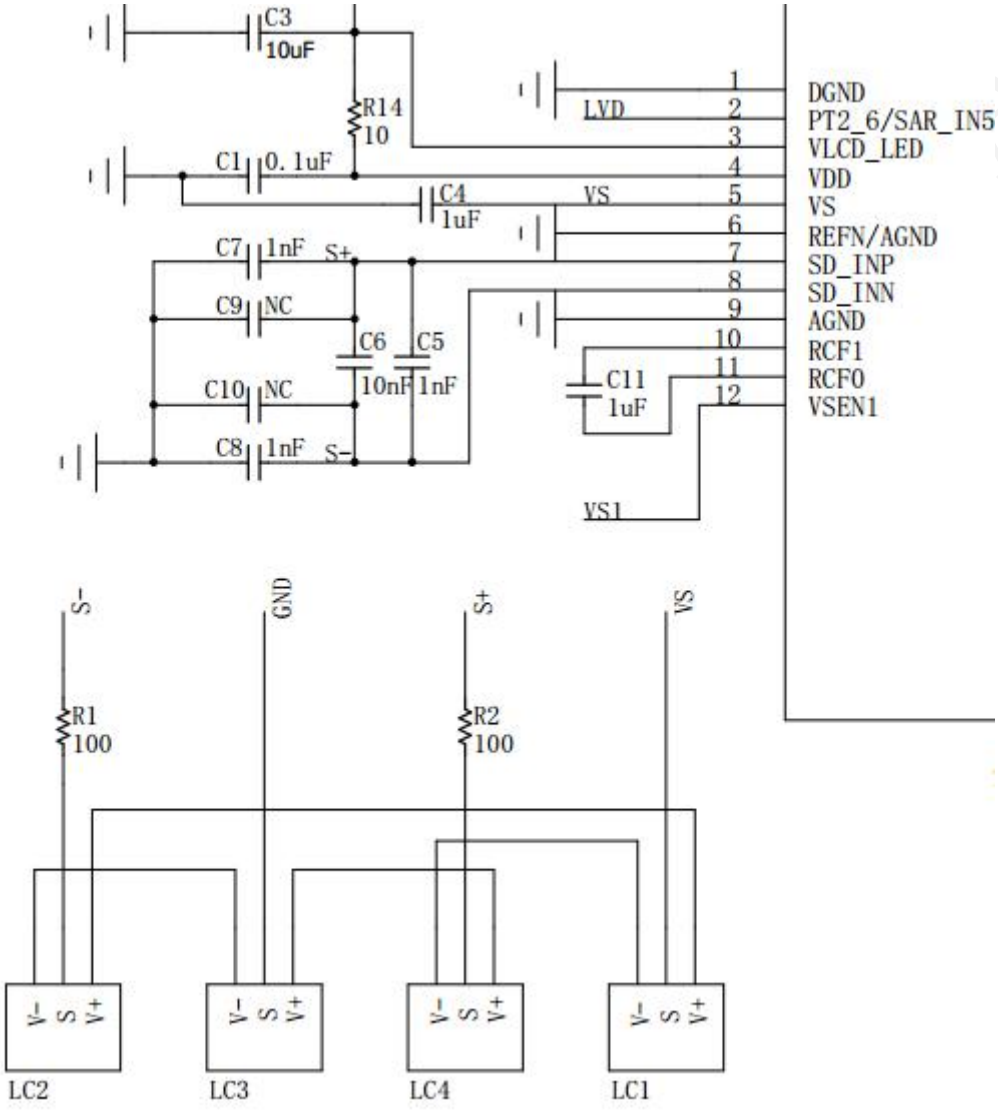


图 3-6 局部电路(e)

5、BIM 电路可以参考下图。其中 **C12 是滤波电容，建议取值 0.1μF**。**R8 是限流电阻，建议取值 1KΩ**，有助于心率信号的测量。R9=300Ω、R10=1KΩ为校准电阻，建议采用误差<=1%的电阻。C11 是整流滤波电容，建议取值 1μF。

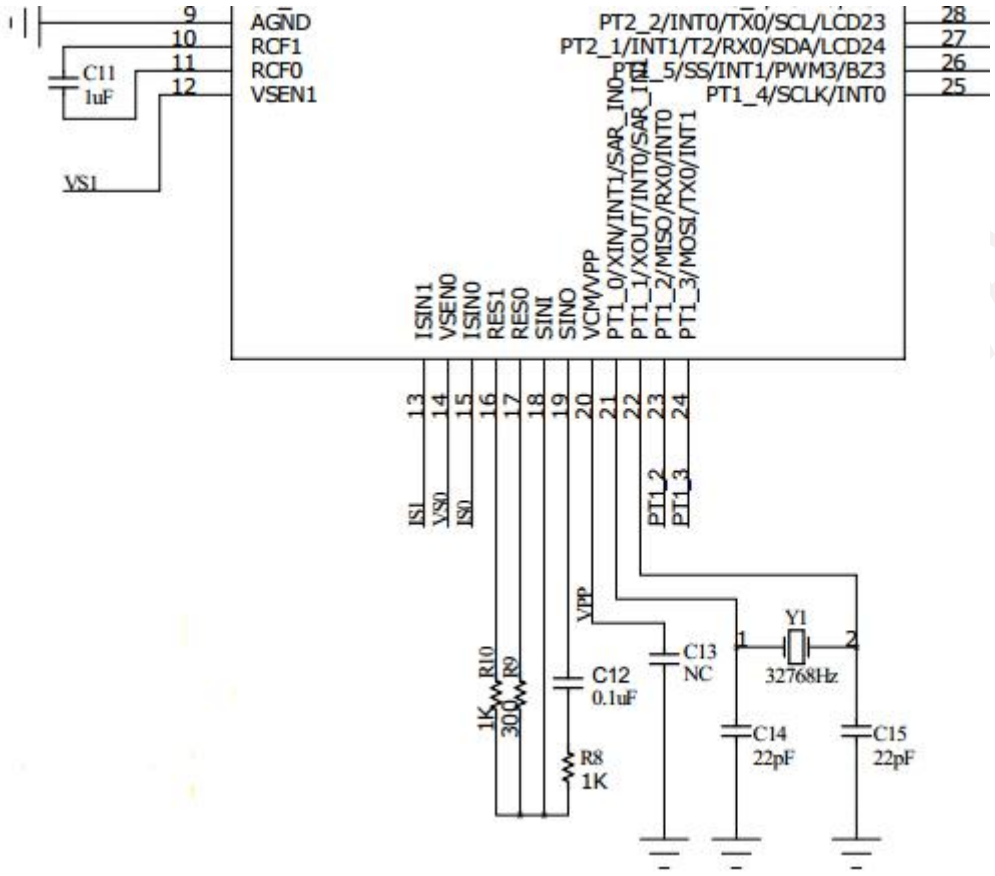


图 3-7 局部电路(f)

由于 IS0--IS1 回路存在保护电阻 Resd、接触电阻 Rcon，会导致 Rx 测量范围变小。当（恒流）激励电流越大，Rx 的范围就越小。如图 3-7 所示，为了满足人体安全需求以及 Rx 测量范围需求，R8 建议取值 1.0KΩ ~ 2KΩ。

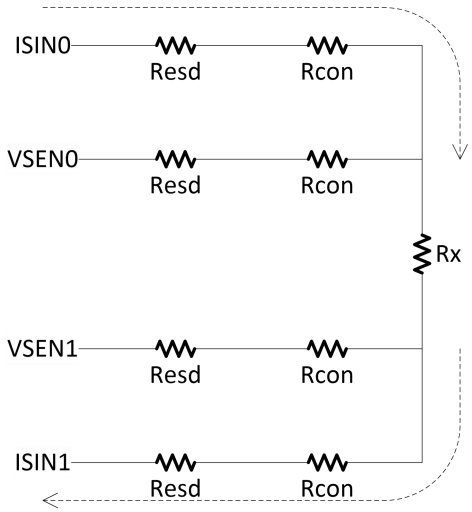


图 3-8 测量等效电路图

VSEN0、VSEN1 为电压测量端口，ISIN0、ISIN1 为激励电流端口，分别连接到 4 个电极片上。建议 ISIN 接到上电极，VSEN 接到下电极，如下所示：

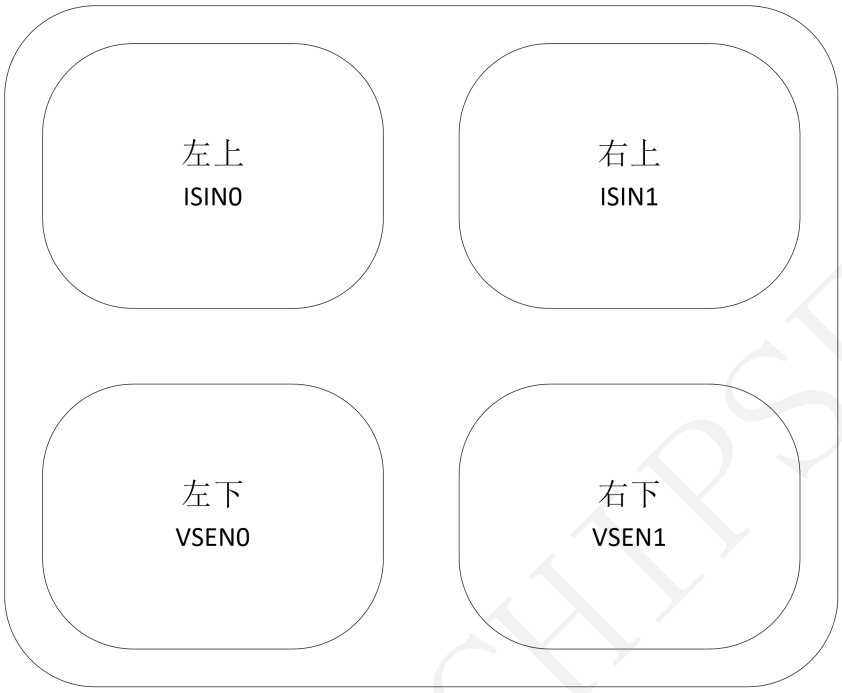


图 3-9 电极片示意图(a)

图 3-10 电极片示意图(b)

为了防止静电损坏 SOC，需要在测量回路中间串接保护电阻，电阻尽量靠近 PCB 接线端，一般取值 300Ω左右。如果需要满足 ESD>6KV（直接接触放电），可以额外加上 ESD 二极管。需要注意 ESD 二极管 V_{RWM}、V_{BR}、C_J 等参数是否符合应用需求，建议选择击穿电压 V_{BR} ≤6V、结电容 C_J<5pF 的型号。

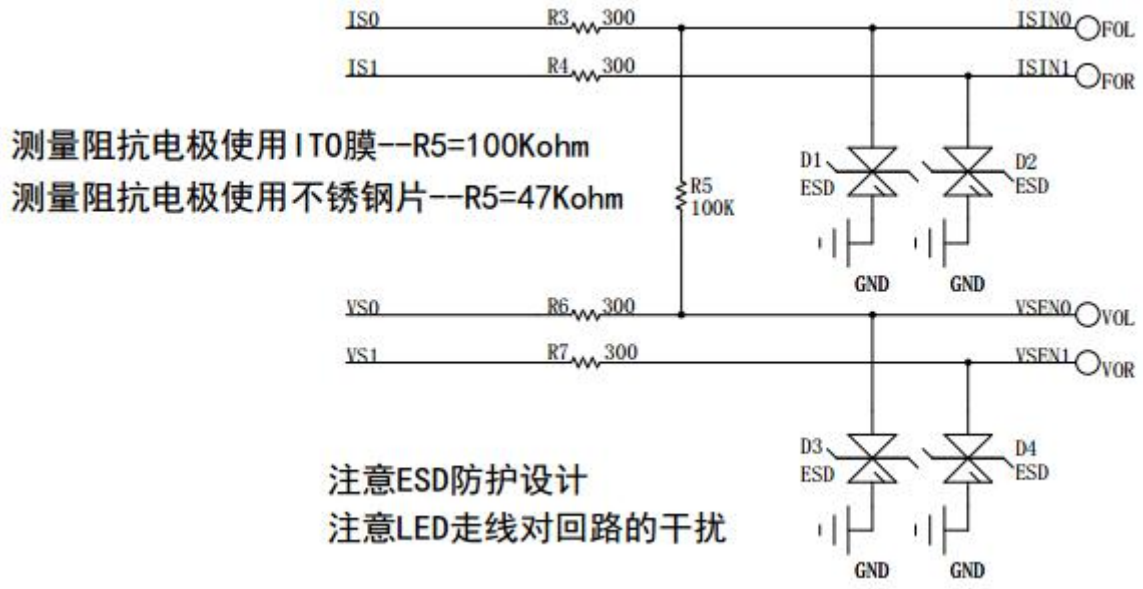


图 3-11 BIM 保护电路

如图 3-11 所示，R5 为了防止空秤时 BIM 输入处于不定态。如果电极使用 ITO 膜，则 R5=100KΩ（参考值）；如果电极使用不锈钢，则 R5=47KΩ。

电极片的设计可以参考《体脂秤电极设计规范 V0.2》。为了保证阻抗测量的重复性，尽量增大电极面积，确保与脚掌充分接触。

6、实时时钟（RTC）单元提供给用户实时时间以及日历信息。RTC 的时钟源由外部 32768Hz 晶振提供。时钟精度取决于晶振误差以及匹配电容值。

关于最优的匹配电容 C_L 的计算公式为：$\mathbf{C_L=C_A*C_B/(C_A+C_B)+C_S+C_{LAYOUT}+C_{WIRE}}$ (C_S 指 PCB 的分布电容，典型值为 5pF；C_{LAYOUT} 指 PCB 走线分布电容，C_{WIRE} 指芯片邦线分布电容，数值根据实测结果而定)

负载电容 C_A 和 C_B 的计算：

例：晶振规格书 C_L 为 15pF。通过公式计算得知，当 C_A= C_B =20pF 的时候，C_L 等于 15pF（忽略 C_{LAYOUT}、C_{WIRE}）。

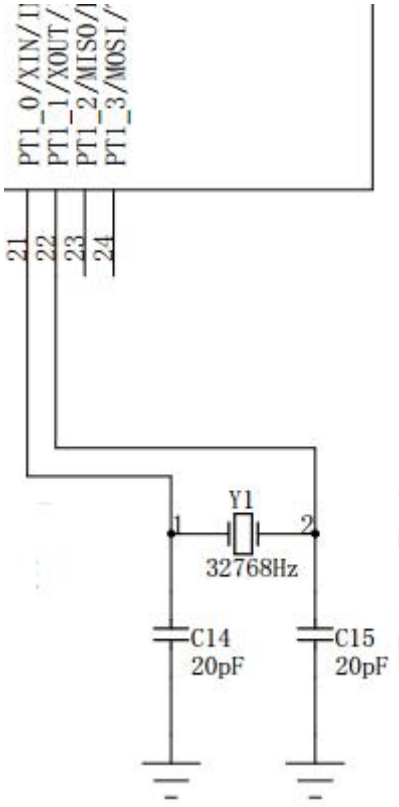


图 3-12 局部电路(g)(参数仅供参考)

7、芯片有 14 个 LED_SEG 驱动口和 4 个 LED_COM 驱动口，最多可以驱动 56 个发光二极管，支持硬件自动扫描刷新 LED。模块的电源来自于 VLCD_LED 管脚，可以应用于白光和蓝光 LED。LED 驱动的原理是在每一路 LED_SEG*下拉恒定电流源，所有 LED_SEG*的恒流源都可以独立配置电流值， 6.25~10mA 之间，步长 0.25mA。

LED_SEG1 有四组寄存器配置电流，分置对应不同 COM。**多色指示灯可以放在 SEG1 端口，便于进行亮度调节，使其亮度一致**。

通常使用 2 个 SEG 端口构成一个”8”字，并且每个数字的段码位置是一致的，例如”1A”和”2A”都对应 COM1 引脚，如图 3-15 所示。

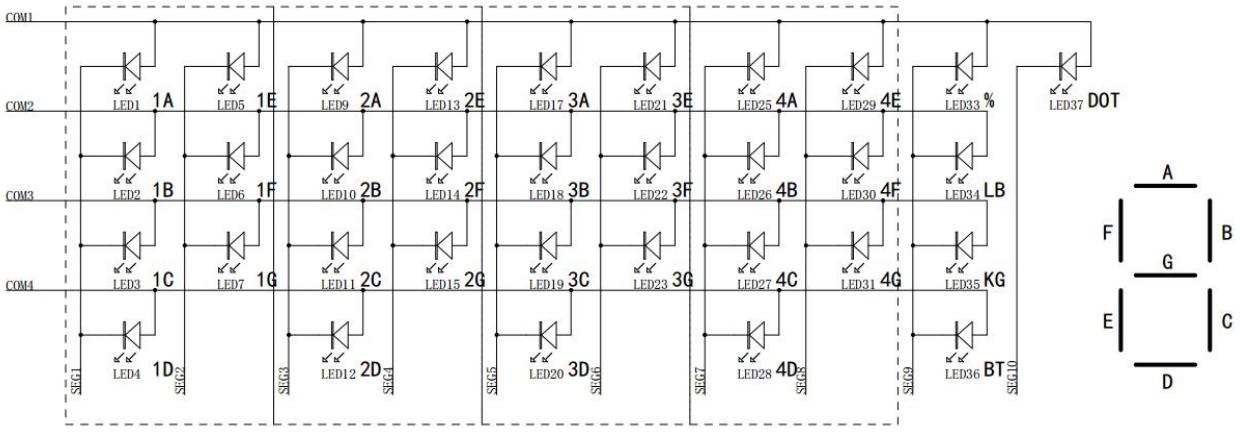


图 3-13 LED 电路(仅供参考)

选用 LED 型号时，注意 Forward Voltage、Luminous Intensity 等参数，通常白光 LED 需要 3.0V 或以上的电压才能正常工作，必要时可以使用 V_{OUT}=3.6V 的 LDO。如果选择 V_{OUT}=3.3V 的 LDO，需要注意选择 Forward Voltage 小，Luminous Intensity 高的型号。

8、选用 LDO 型号时，注意 I_{SS}、V_{IN}、ΔV_{OUT} 等参数，LED 秤建议选用 I_{OUT}>100mA 的型号，减少显示变化大时（例如锁定闪烁画面）产生电源抖动。满足亮度的条件下，尽量调小 LED 驱动电流，可以减少电源抖动。

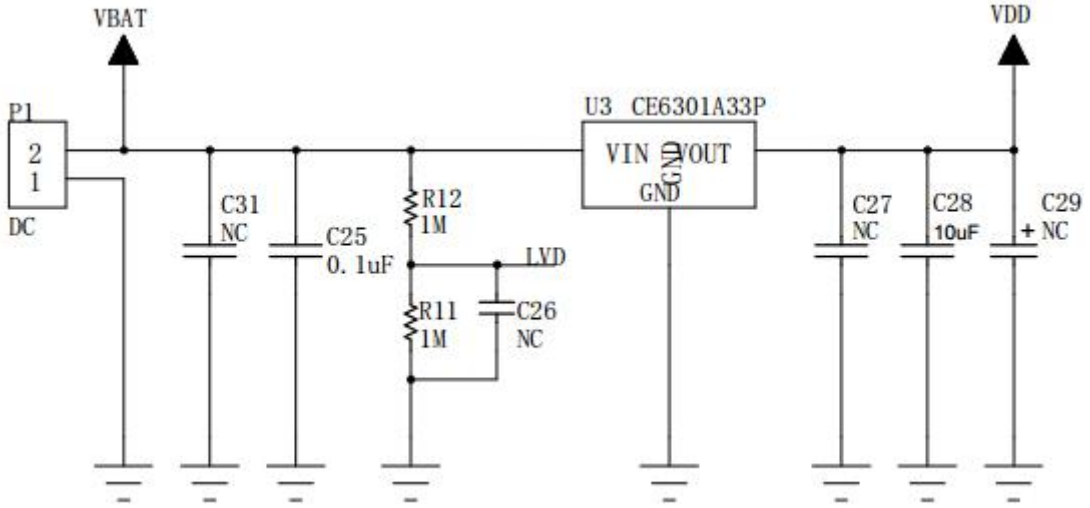


图 3-14 LDO 电路

3.2 PCB 设计

具体电路可以参考相关硬件资料。

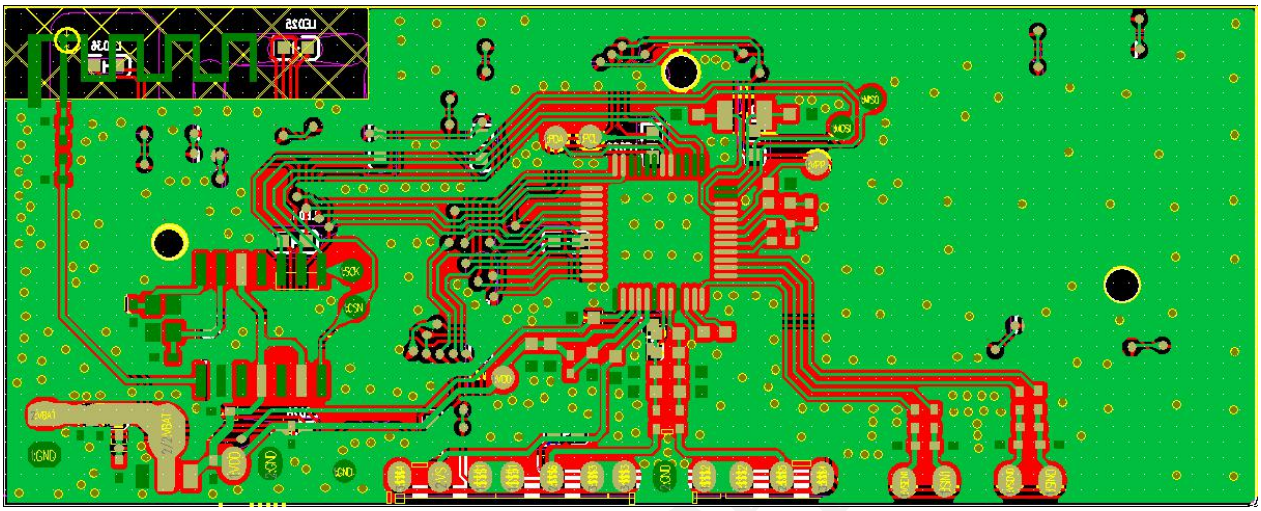


图 3-15 CSU18M9x_CST92P23 电路图

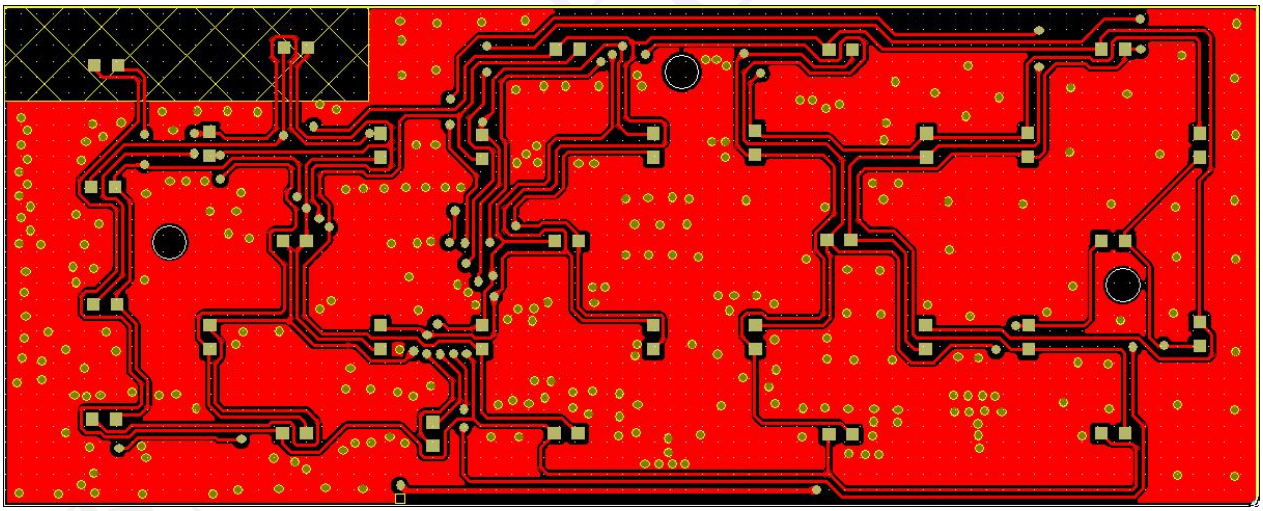


图 3-16 CSU18M9x_CST92P23 Top Layer

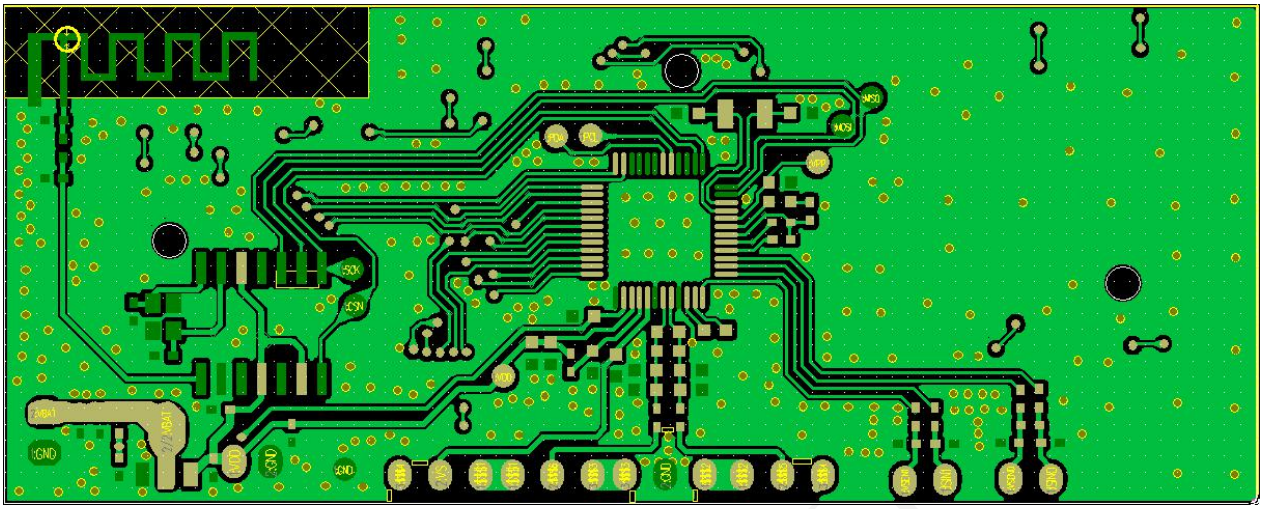


图 3-17 CSU18M9x_CST92P23 Bottom Layer

1、模拟走线与数字走线尽量分开，防止数字信号干扰模拟信号。模拟信号主要指检测电压回路、电源回路、称重回路以及 BIM 回路。

2、Top Layer 与 Bottom Layer 走线交错的时候，尽量垂直相交，如下图所示。

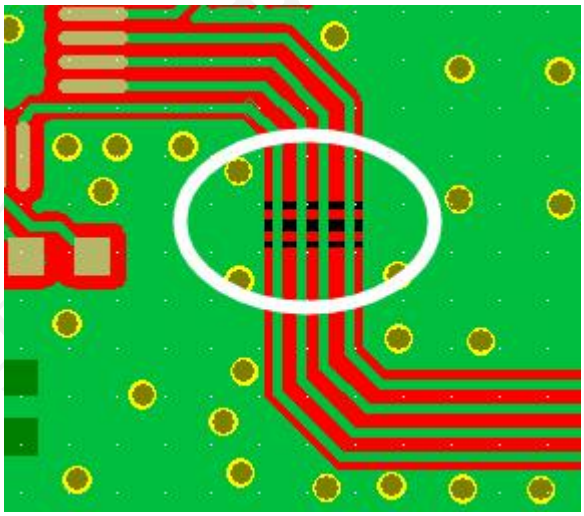


图 3-18 PCB Layout 局部(a)

3、（称重）模拟输入电路为差分信号，需要减少回路面积，走线尽量对称，中间不要有 GND，滤波器件靠近 IC。

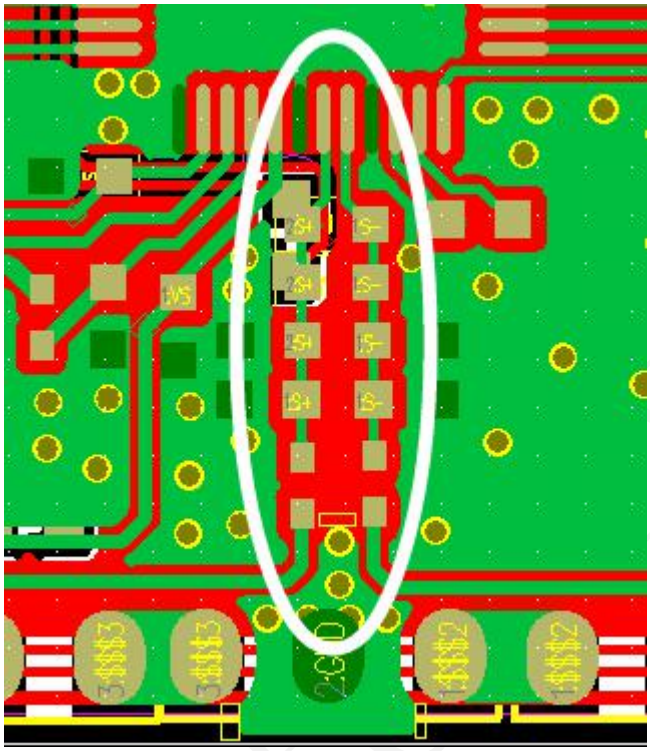


图 3-19 称重测量滤波电路

4、人体秤一般采用半桥传感器，所以需要 4 组传感器来构成一个全桥传感器，中间抽头分别是 E+、S+、E-和 S-，如下所示：

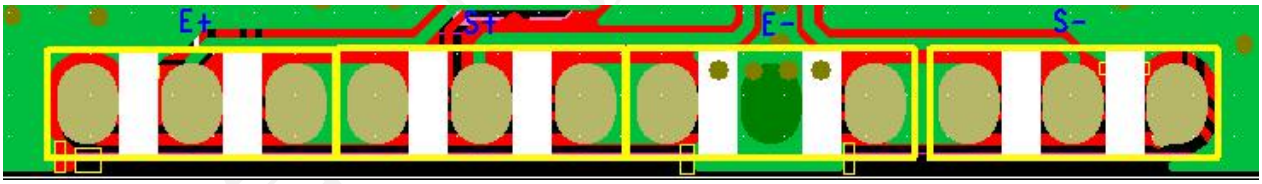


图 3-20 称重传感器接口

5、在同一层，电源走线应避免 LED 走线、UART/SPI/IIC 走线的干扰，可以使用 GND 隔离。Layout 时需要注意 VDD-GND 回路面积要尽量缩小，VDD 走线尽量在同一平面上，GND 敷铜面积足够大，以免 ESD 干扰造成系统复位等异常情况。

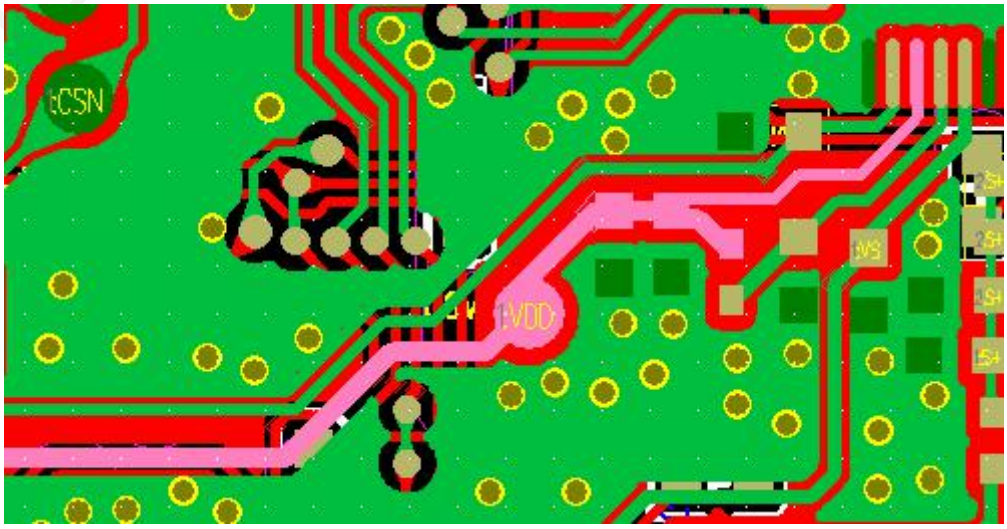


图 3-21 PCB Layout 局部(b)

Bottom Layer 由于走线导致 GND 回路不完整，可以使用 Top Layer 的敷铜联通 GND。尽量多摆放一些过孔，减少 GND 回路阻抗。

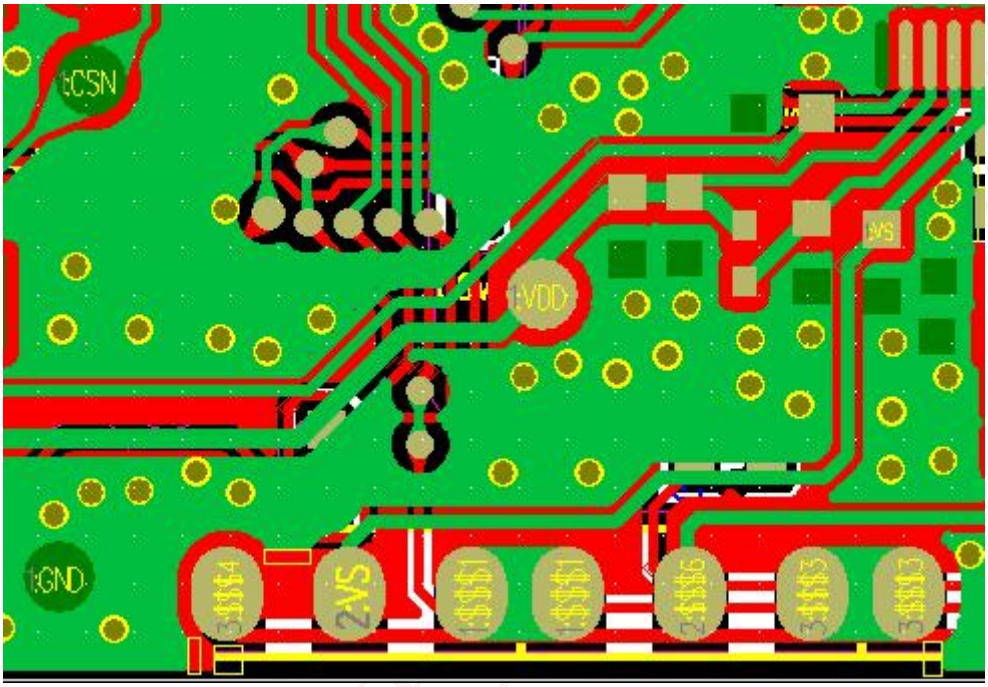


图 3-22 PCB Layout 局部(c)

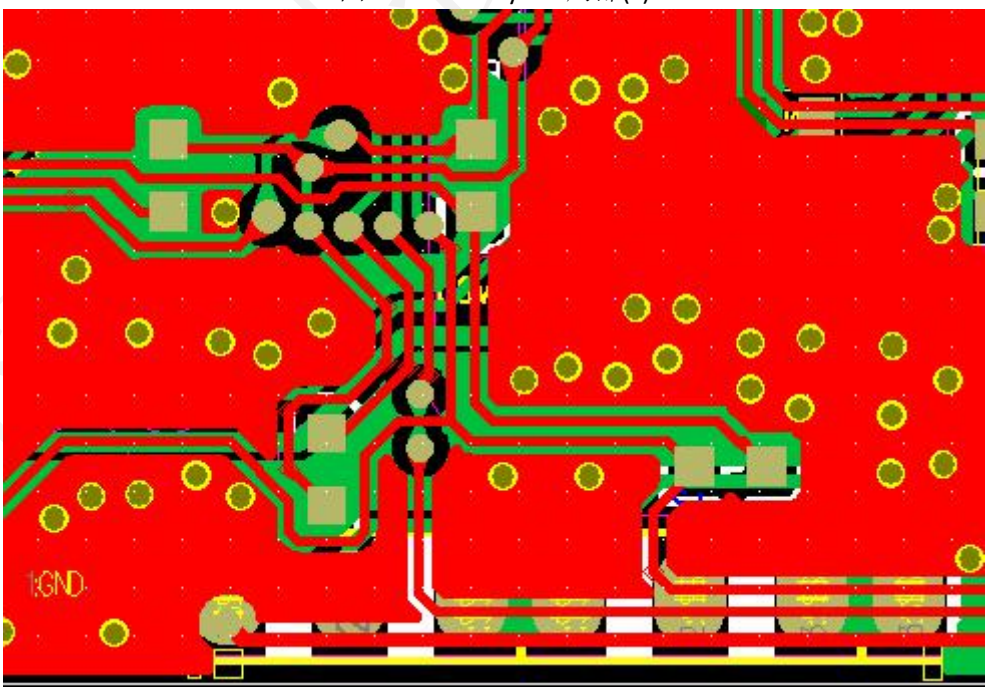


图 3-23 PCB Layout 局部(d)

6、为了防止静电损坏 IC，需要在 BIM 测量回路添加保护器件。ESD 保护器件尽量靠近 PCB 接线端，然后紧跟（串接）保护电阻，一般取值 300Ω左右。

BIM 的测量接口分成 VSEN0/ISIN0 和 VSEN1/ISIN1 两组接口，分别接到左右两边的电极片。但实际上 VSENX 与 ISINX 不是一组差分信号，ISIN0/ ISIN1 和 VSEN0/ VSEN1 才是差分信号。因此 Layout 时注意 VSENX 与 ISINX 不要太靠近，防止产生串扰。

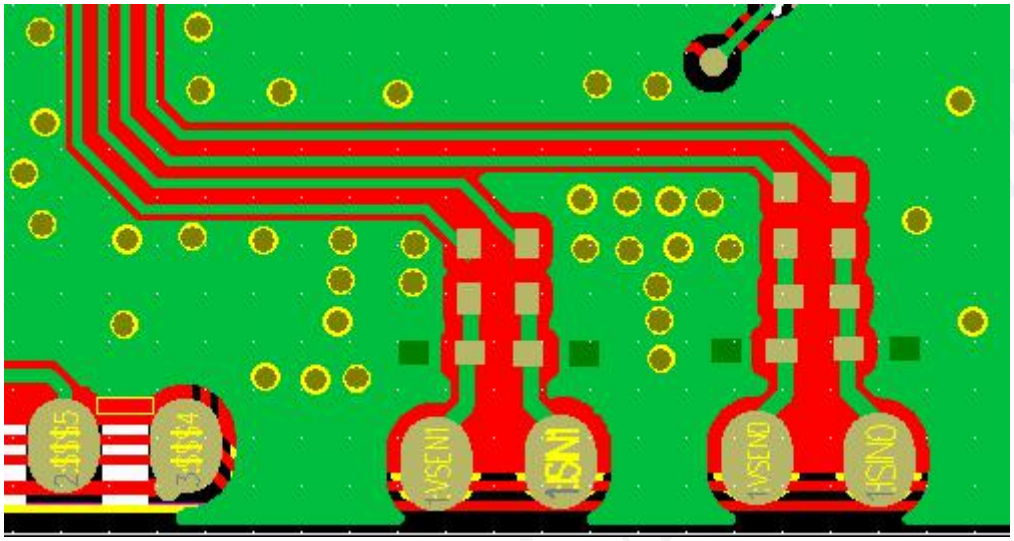


图 3-24 BIM 测量保护电路

7、蓝牙天线需要保证足够的 PCB 净空，要保证足够的 GND 敷铜（详见《低功耗蓝牙硬件设计》）。实际由于成本限制，往往无法确保理想的 PCB 净空，所以 Top Layer 的 LED 走线需要像图 3-25 那样，尽量与 ANT 走线垂直，缩小回路面积，增加 PCB 净空。

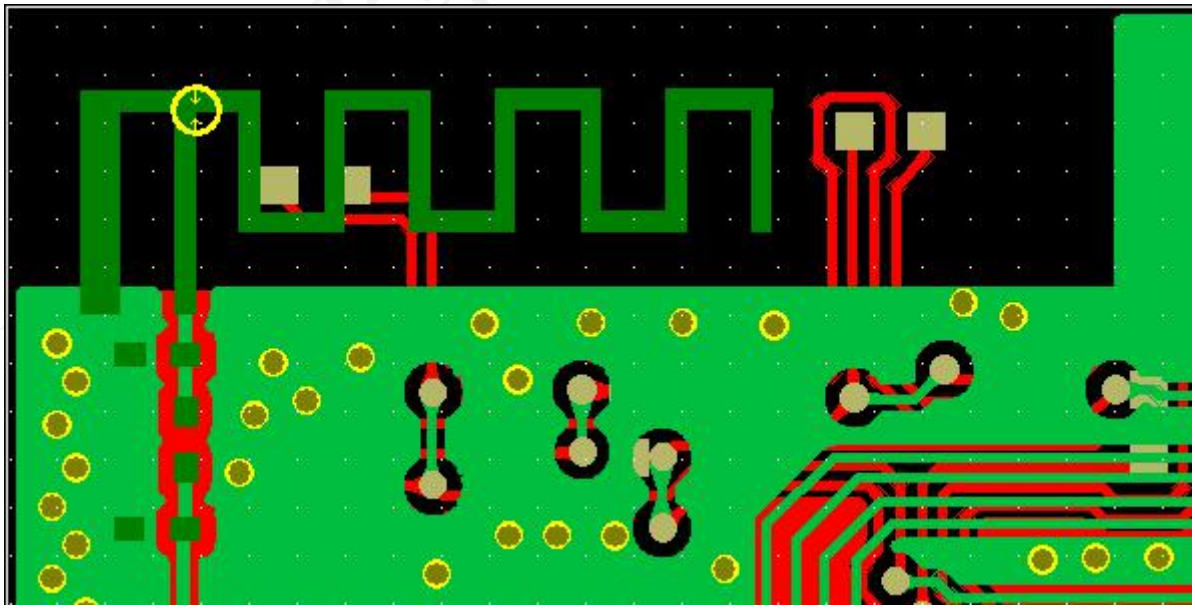


图 3-25 ANT 电路

8、由于 Bottom Layer 走线的缘故，导致 GND 回路不完整，因此需要 Top Layer 敷铜作为 GND 回路。为了确保 GND 敷铜面积，可以尽量使相邻的 LED 走线靠近（因为 LED 是分时驱动，所以不必担心相互串扰）。

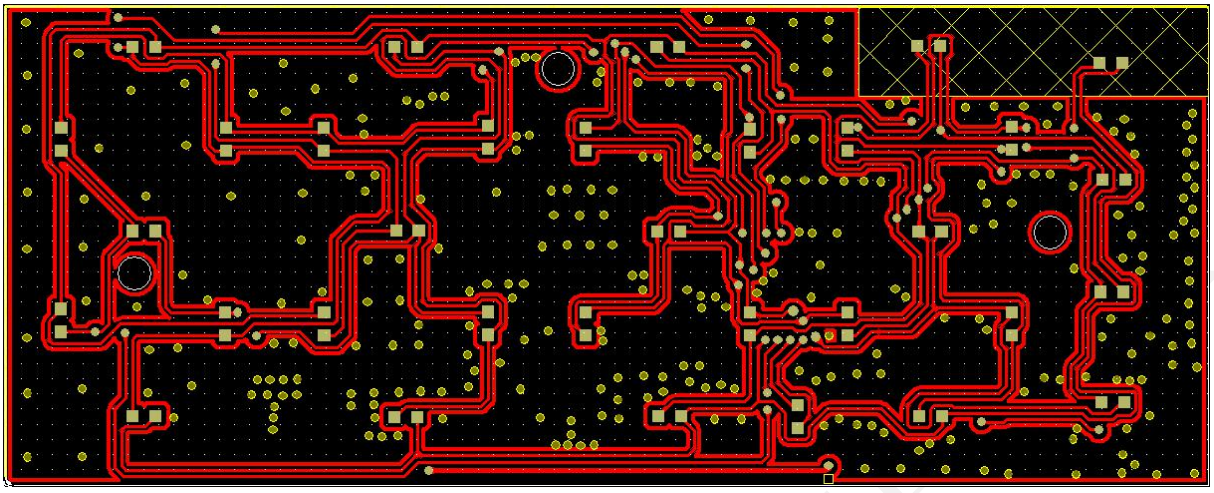


图 3-26 LED 电路(a)

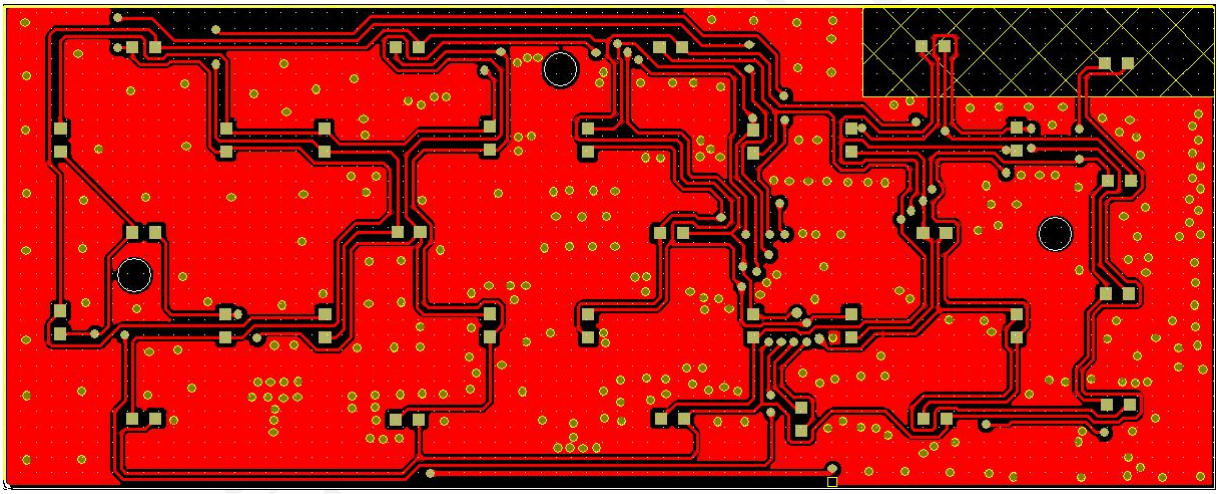


图 3-27 LED 电路(b)

9、Layout 应注意可生产性与可测试性。例如元器件需要与 PCB 接线端隔开一段距离，防止焊线时碰到元器件；PCB 接线端下方尽量不要走线，防止阻焊层脱落导致焊线短路，如果需要走线则建议放置丝印层增加安全性。

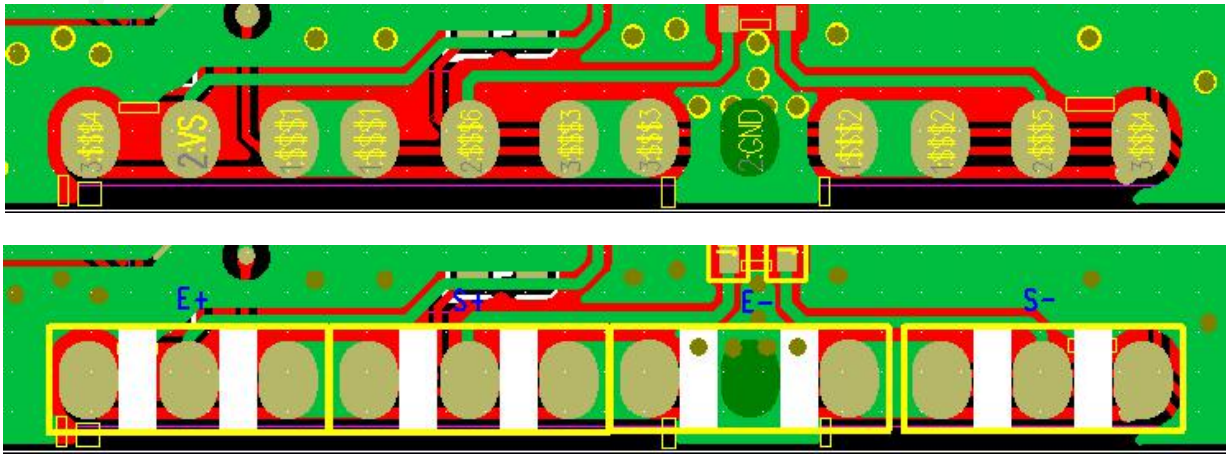


图 3-28 PCB Layout 局部(e)

3.3 BOM 说明

以 CSU18M91+CST92P12 方案为例，需要注意几点：

1、BIM 校准电阻 R9、R10 需要选择误差≤1%的型号。

2、LED 建议选择导通电压低、亮度高的型号。

3、CST92P12 的晶振需要选择 24MHz/10ppm 的型号。

4、晶振的匹配电容需要根据实际情况调整参数，建议选择误差小的型号，理论计算公式详见《3.1 原理图设计》。

5、天线匹配器件需要根据实测值进行调整参数，C24=1.2pF 仅供参考。为了减少寄生参数，建议使用 0402 或 0201 封装。

6、成本允许的条件下，建议选用参数更好的型号来替换，有助于改善性能。

例如：称重测量滤波电路中 R1、R2 和 C7、C8 选用误差小的型号，可以更好地抑制共模干扰。

1	贴片电容	1.2pF ±0.25pF 50V C0G	C0402	1	C24	ANT参数以实际为准
2	贴片电容	10pF ±5% 50V C0G	C0402	2	C16, C17	XTAL参数以实际为准
3	贴片电容	20pF ±5% 50V C0G	C0402	2	C14, C15	XTAL参数以实际为准
4	贴片电容	1nF ±10% 50V X7R	C0402	3	C5, C7, C8	
5	贴片电容	10nF ±10% 50V X7R	C0402	2	C6, C20	
6	贴片电容	100nF ±10% 10V X5R	C0402	6	C1, C18, C19, C22, C25, C30	
7	贴片电容	1uF ±10% 10V X5R	C0402	5	C4, C11, C12, C21, C23	
8	贴片电容	10uF ±10% 10V X5R	C0603	3	C2, C3, C28	
9	贴片电阻	10Ω ±5% 1/16W	R0402	1	R14	
10	贴片电阻	100Ω ±1% 1/16W	R0402	2	R1, R2	
11	贴片电阻	300Ω ±1% 1/16W	R0402	5	R3, R4, R6, R7, R9	
12	贴片电阻	1KΩ ±1% 1/16W	R0402	1	R10	
13	贴片电阻	1.5KΩ ±1% 1/16W	R0402	1	R8	
14	贴片电阻	10KΩ ±5% 1/16W	R0402	2	R13, R17	
15	贴片电阻	100KΩ ±5% 1/16W	R0402	1	R5	
16	贴片电阻	1MΩ ±1% 1/16W	R0402	2	R11, R12	

图 3-29 BOM 表(a)

（接上表）

17	贴片发光二极管	超亮白光	LED0603	33	LED1, LED2, LED3, LED4, LED5, LED6, LED7 LED9, LED10, LED11, LED12, LED13, LED14, LED15 LED17, LED18, LED19, LED20, LED21, LED22, LED23 LED25, LED26, LED27, LED28, LED29, LED30, LED31 LED33, LED34, LED35, LED36, LED37	
18	贴片AISC	CSU18M91-LQFP48	LQFP48	1	U1	
19	贴片AISC	CST92P12-SSOP24L	SSOP24	1	U2	
20	贴片LDO	CE6301A33P	SOT89	1	U3	
21	贴片FLASH	FM25F04A	SOP8	1	U4	
22	贴片无源晶振	32.768KHz ±20ppm 12.5pF	FC135	1	Y1	
23	贴片无源晶振	16MHz ±10ppm 9pF	SMD3225	1	Y2	
24	线路板	FR4 沉金 白油黑字 79×31.5×1.0mm	PCB	1	PCB	双面板

图 3-30 BOM 表(b)

4 补充说明

4.1 功耗说明

对于 CSU18M9x 智能体脂秤应用，系统时钟建议选择 8MHz。

如果需要实现心率测量功能，则选用 2 分频或 4 分频选项，因为心率算法耗时较长，需要较快的 CPU 主频来处理。如果方案没有心率功能，可以选择较低的 CPU 主频，例如选择 4 分频或 8 分频，这样可以降低运行功耗。

CSU18M91 内置心率算法耗时~2ms @CPUCLK=2MHz。

表 4-1 CSU18M9x 典型功耗

CPU Clock	全速运行功耗
4MHz	~3mA
2MHz	~2.2mA
1MHz	~1.8mA

用户可以通过调节恒流源的大小来调节 LED 亮度，在满足亮度需求的前提下，尽量降低驱动电流。如果使用最低驱动电流 6.25mA 还不能满足功耗需求，可以通过调整占空比” LED_DUTY”来实现亮度的进一步调节（自动扫描模式）。

在满足系统精度的前提下，可以选择激励电压 VS=2.3V，降低称重传感器的激励电流。

CSU18M9x 进入休眠模式时， 应避免 IO 电流倒灌、不定态的情况（例如 UART 端口），防止待机功耗异常。

4.2 环境温度测量

在低成本应用中，用户可以使用 CSU18M9x 内置的温度传感器检测环境温度，如果芯片的驱动电流不大，则可以把片内温度视作环境温度。具体配置见章节<2.3.4 温度测量>。

由于 LED 显示电流较大，导致 CSU18M9x 产生温升，从而导致内置温度传感器不能如实反映环境温度。因此需要做一些特殊处理。例如：上电时测量当前温度，当启动 LED 显示后就暂停温度测量。关闭 LED 显示后（例如待机状态），开始计时，超过一定时间后认为芯片已经“冷却”了，才允许下一次检测温度。

4.3 低电报警

低电报警可以使用 10Bit SAR-ADC 实现。软件根据公式计算当前电压值，然后判断当前电压是否低于低压阈值，如果是则置起低电报警标志。具体配置见章节<2.3.6 电压测量>。

由于 LED 秤驱动电流较大，因此可以适当调高低电阈值，避免系统在低电状态下（电源不稳定）测量阻抗或心率出现误差大的情况。

4.4 显示端口设置

使能 LED 显示前需要设置相应的端口为 SEG 模式。其中”SEGCON3”、” SEGCON4”寄存器分别控制 PT3* SEG 、PT4* SEG 功能，赋值”0xFF”表示使能 LED/LCD 驱动端口，赋值”0x00”表示禁止 LED/LCD 驱动端口。

需要注意的是，**PT3* SEG、PT4* SEG 只能整体控制，不能对单独的端口进行控制**。举例：如果 LED 驱动使用了 PT3.7 端口，则意味着 PT3.6 ~ PT3.0 端口此时不能作为 GPIO 使用。

PT3* SEG 、PT4* SEG 端口设置详见参考代码 cs_drv_led.inc。

4.5 优化心率信噪比

如果想进一步提升心率信噪比，减小心率测量误差，可以在心率测量模式下使用外部参考源。

称重模式依然使用 VS 作为参考源/激励源，节省系统功耗。当切换到心率测量模式时，使能外部参考源（可以使用有使能引脚的低噪声 LDO）作为电压基准，同时参考源输出采用 LC 滤波，测量完毕后关闭外部参考源。

使能/关闭外部参考源时注意延时一小段时间，使电源系统稳定下来。

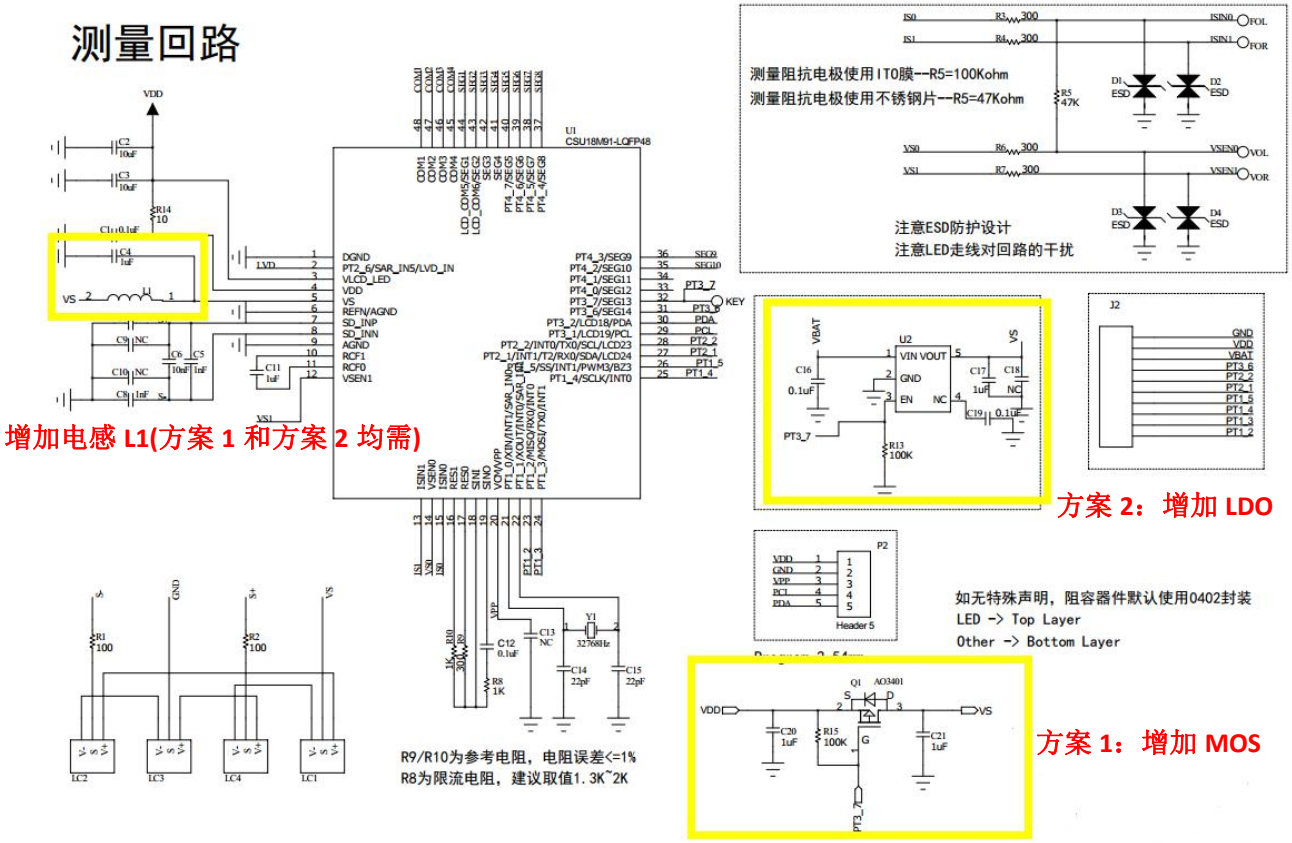


图 4-1 CSU18M91 心率秤改进方案

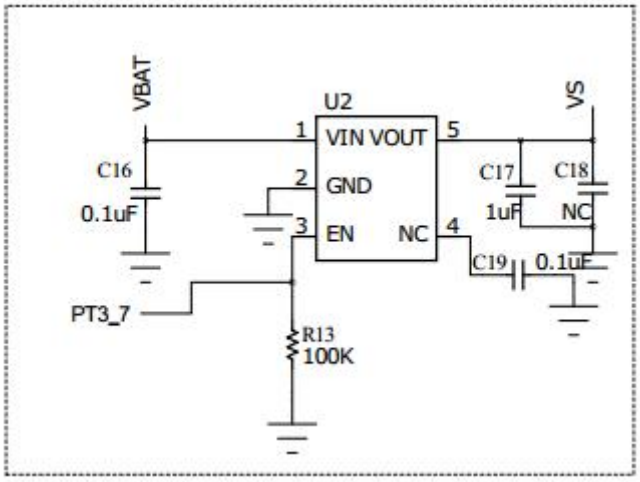


图 4-2 CSU18M91 参考电路(a)

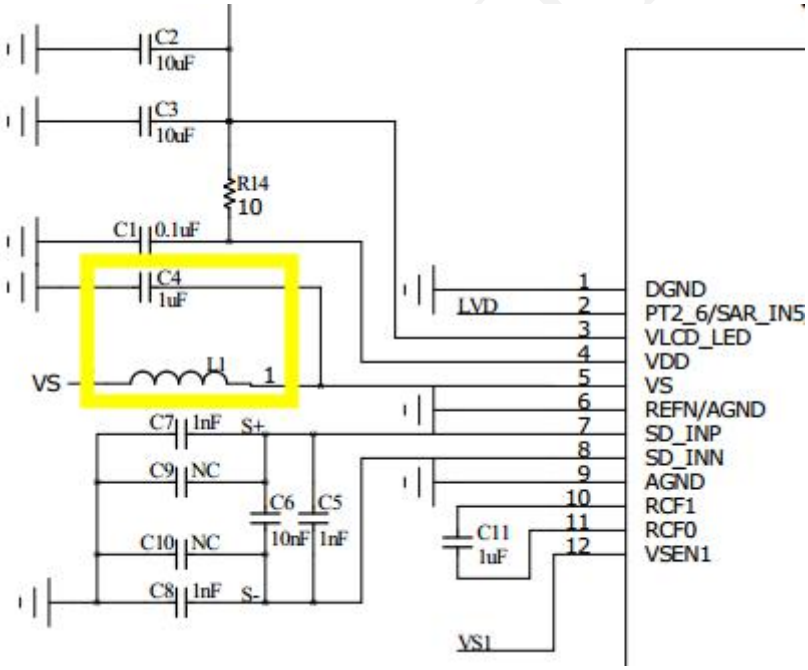


图 4-3 CSU18M91 参考电路(b)

外部参考源（LDO）建议选择输出电压>=3.0V 的型号，提高心率信噪比，需要注意的是输出电压不能大于 VDD。LDO 参考型号：ME6211C33M5G。**滤波电感取值>=270nH**。

图 4-4 心率原始信号(内部参考源)

图 4-5 心率原始信号(外部参考源，LDO 电路)

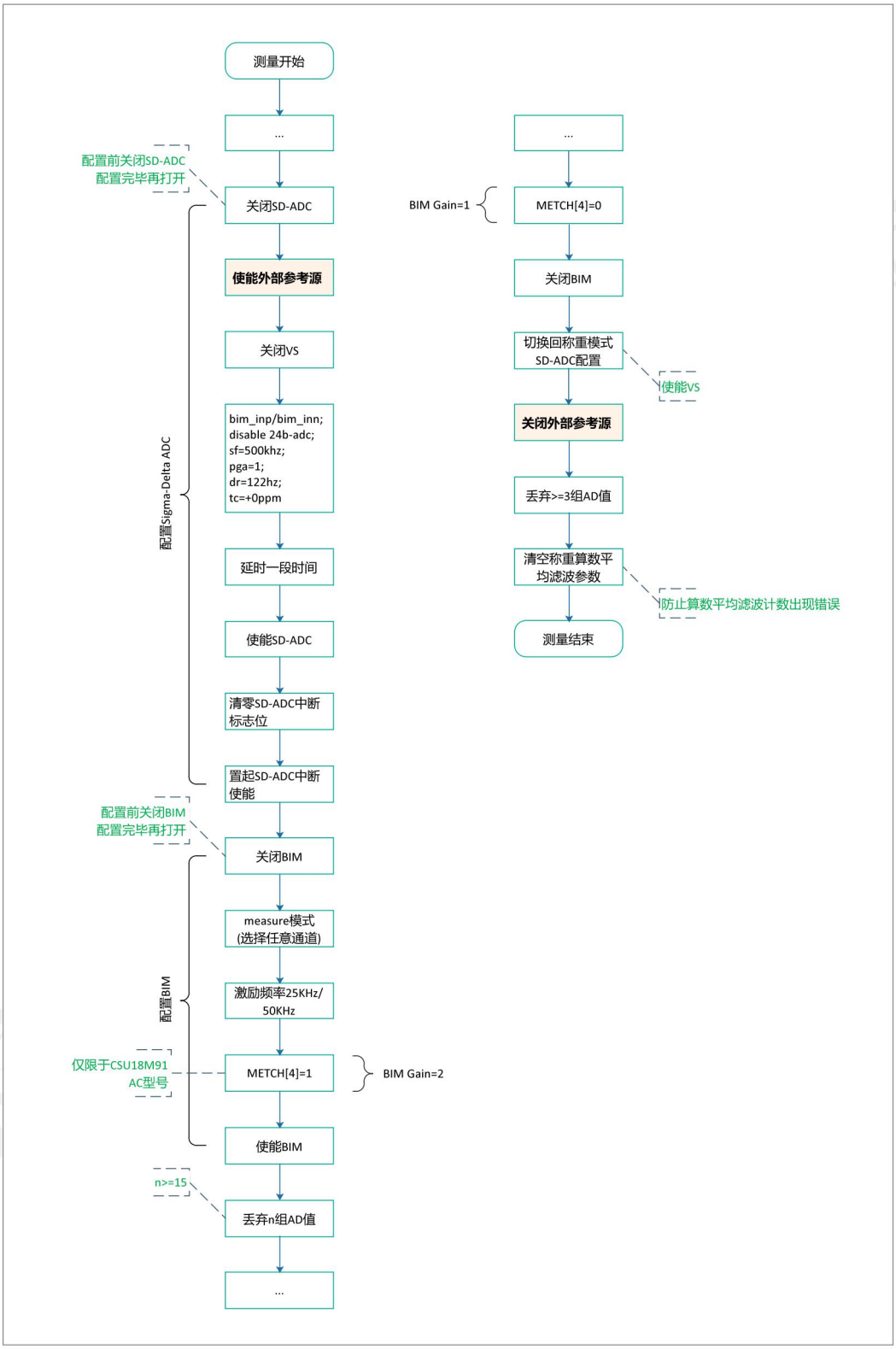


图 4-6 使用外部参考源配置流程图

如果方案成本紧张，可以使用 P-MOS 控制电路使能系统电源作为参考源。当切换到心率测量模式时，使能外部参考源作为电压基准，同时**参考源输出采用 LC 滤波**（见图 4-3），测量完毕后关闭外部参考源。

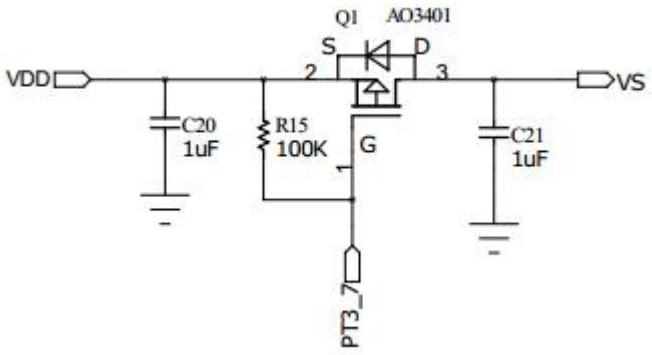


图 4-7 CSU18M91 参考电路(c)

由于 P-MOS 控制电路（见图 4-7）低电平有效，因此**需要注意测量体重、测量阻抗、上电校准 BIM（F_Drv_BIM_Init）、系统待机时，控制端口保持高电平以关闭外部参考源**。

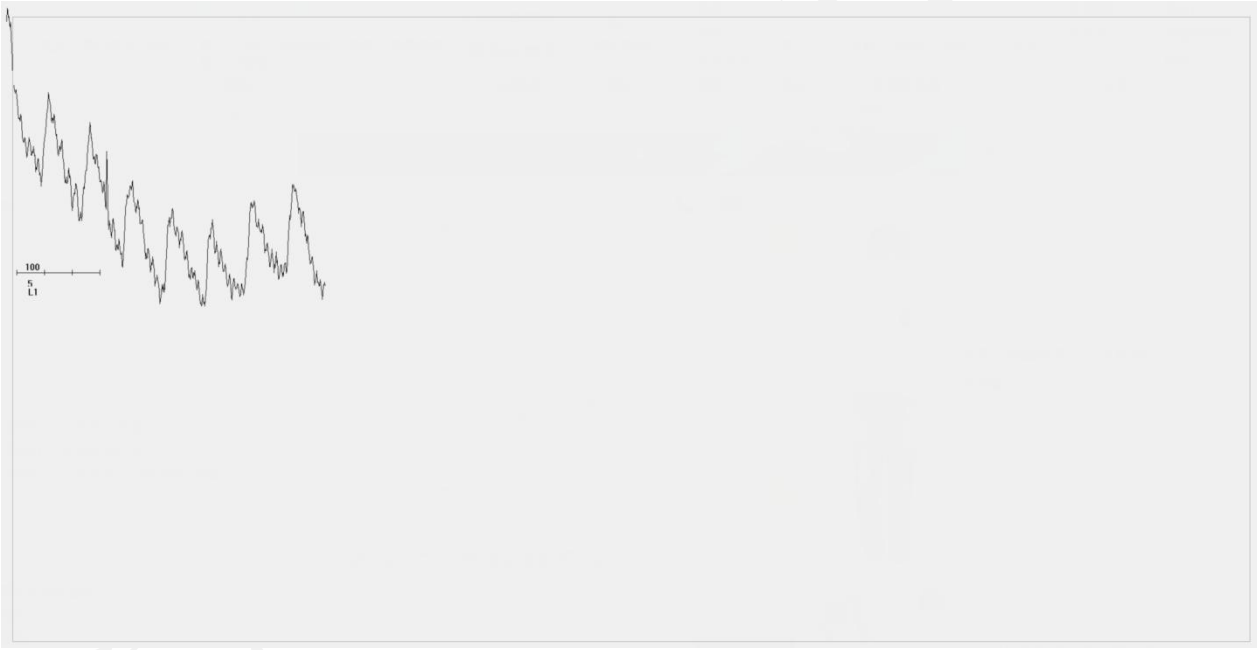


图 4-8 心率原始信号(外部参考源，MOS 电路)

如果使用外部参考源效果不佳，有可能是心率测量的电路存在器件损坏或虚焊的情况。可以重点检查以下电路。

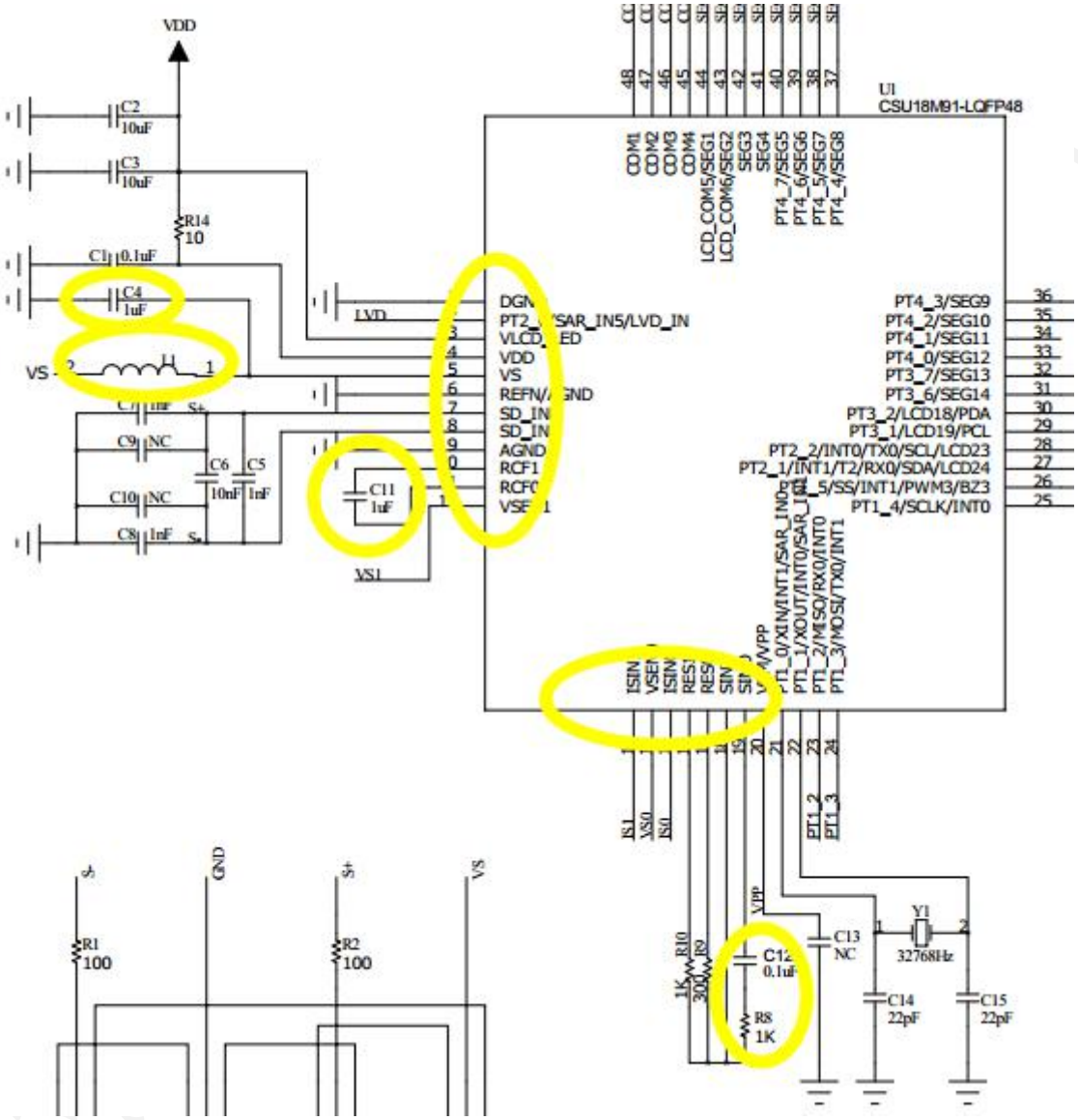


图 4-9 心率测量电路示意图

图 4-10 心率原始信号(电路修复前)

图 4-11 心率原始信号(电路修复后)

需要注意的是，假如外部参考源损坏，测量心率时（VS 已经关闭）会导致模拟电路工作异常。为了防止这种情况发生，软件应有防止死机的机制，例如定时检测 AD 缓冲区是否有变化，然后作出相应的处理。

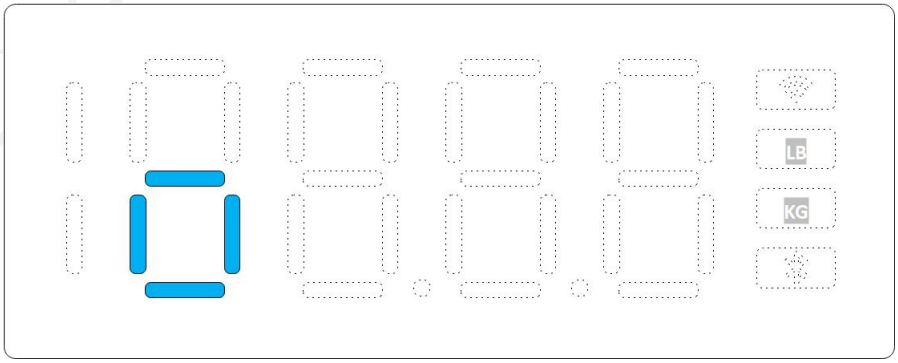
如果没有软件防死机措施，可以考虑不关闭 VS，测量心率时直接加载外部参考源，但需要外部参考源需要大于 VS。

图 4-12 心率原始信号（不关闭 VS，加载外部参考源）

由于系统电源容易受到其它电路的干扰，例如 LED 闪烁显示时会导致电源输出波动，从而影响测量效果。因此在**测量心率时尽量避免电源的大幅度波动，例如重量锁定闪烁时不要进行心率测量，心率测量画面尽量平滑过渡（避免电流负载波动）**。

图 4-13 电流突变产生信号"尖峰" (外部参考源，MOS 电路)

可以参考下图的测量提示画面，循环显示电流相对恒定的画面，减少负载的变化。



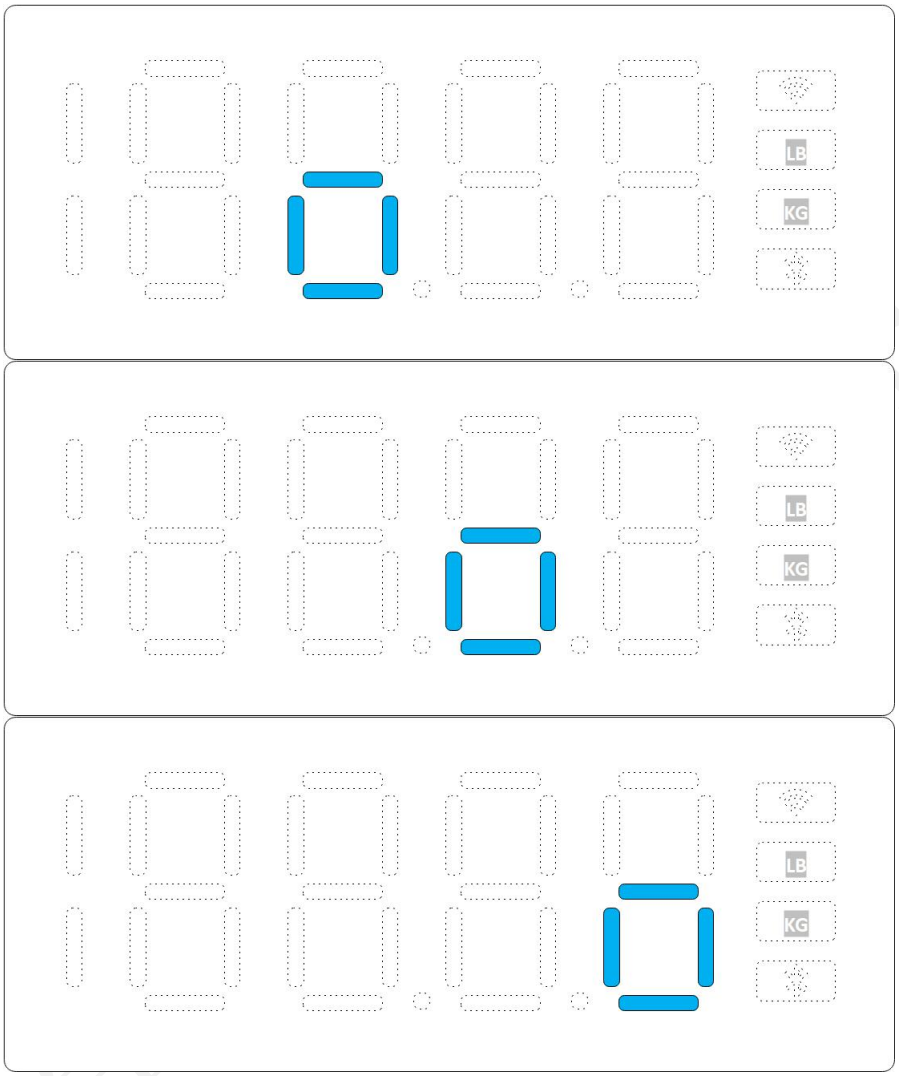


图 4-14 心率测量显示画面(仅供参考，以实际需求为准)

4.6 心率测试及验收指南

测试目的：

验证心率秤心率数据输出的准确性。

测试方法：

对比测试，对比受试者同时使用心率秤与 PC-60NW（推荐型号）脉搏血氧饱和度仪测量输出的心率数据，根据验收标准判断心率秤的心率数据输出是否合格。

建议进行 10 人 10 台 10 次对比测试。

测试人群：

要求 BMI 的范围为 12~35，年龄的范围为 18~50，男性 5 人，女性 5 人。

测试设备：

心率体脂秤 10 台，PC-60NW 脉搏血氧饱和度仪 1 台。

图 4-15 脉搏血氧饱和度仪

测试步骤：

1) 确保两台设备能正常工作，开启设备；
2) 让受试者静坐 2min，保持心率平稳；
3) 让受试者脱鞋脱袜，手指佩戴脉搏血氧饱和度仪，双脚平稳地踏上心率秤，足底与电极片保持紧密接触，开始测试，记录心率秤输出的心率值与脉搏血氧饱和度仪显示的心率值，每人重复测试 10 次，每次一组数据。

注意事项：

1) 测试过程中保持直立、双手不要乱动；
2) 保持脚掌与电极片接触充分；

3) 确保脉搏血氧饱和度仪处于稳定测量状态（显示下图锯齿波形），否则对比数据将会是错误的；

图 4-16 稳定测量状态

4) 为了获取最佳结果，不要穿厚重衣服、佩戴首饰和携带移动设备；

5) 测试过程中保持放松，不能讲话，身体不能晃动。

验收标准：

对比数据差值在+-5bpm 内的占比>=90%；对比数据差值在+-10bpm 内的占比>=95%。