目录

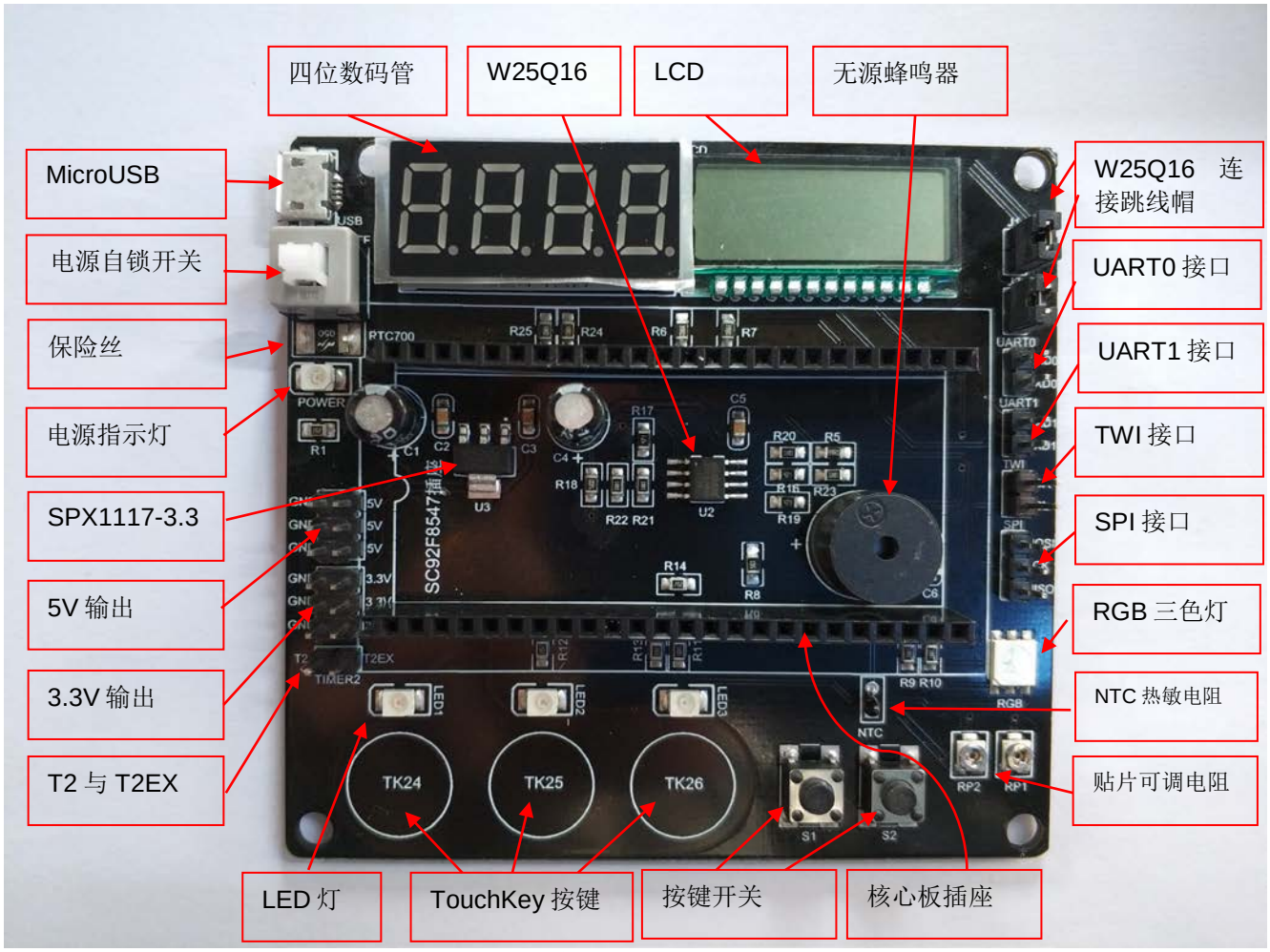
1 SDK101X 开发板简介

1.1 SDK101X 开发板概述

赛元 SDK101x 开发板，由开发板底板 SDK1010 以及核心板 SDK1011/ 1012/ 1013/ 1014 组成。本开发板配备常用的单片机外围资源，自带调试下载接口，配合开发板提供的示例程序，可以让用户在最短的时间，熟悉并掌握赛元 MCU 相关的编程方法。本开发板非常适合初步接触赛元 MCU 的用户自学使用。

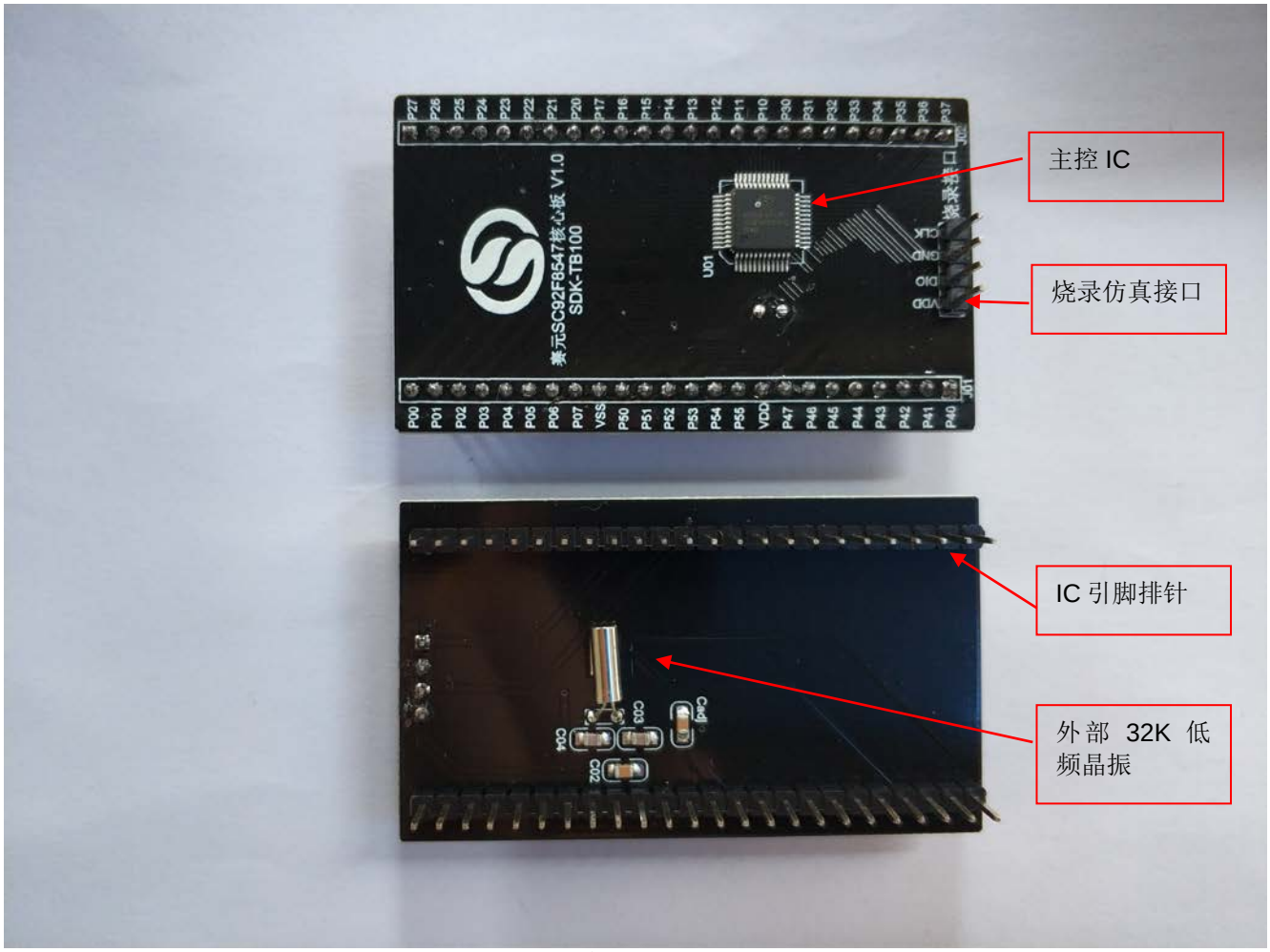
1.2 SDK101X 开发板硬件资源汇总

下图为开发板底板 SDK1010 资源简图。



底板 SDK1010

开发板核心板资源简图以 SDK1011 为例，如下，上部分为正面，下部分为反面。底板与核心板拼接时，核心板赛元 logo 一端应对准底板丝印下凹一端。



核心板 SDK1011

SDK1011 核心板板载资源如下：

- CPU：SC92F8547/7547，工作电压为 2.4V~5.5V，ROM 为 32KB，RAM 为 2KB，系统时钟可选 16/8/4/1.33MHz
- 一组烧录仿真引脚
- 一个外接 32K Hz 低频晶振

SDK1012 核心板板载资源如下：

- CPU：SC92F8446B/7446B，工作电压为 2.4V~5.5V，ROM 为 16KB，RAM 为 1KB，系统时钟可选 16/8/4/1.33MHz
- 一组烧录仿真引脚
- 一个外接 32K Hz 低频晶振

SDK1013 核心板板载资源如下：

- CPU：SC92F8463B，工作电压为 2.4V~5.5V，ROM 为 16KB，RAM 为 1KB，系统时钟可选 12/6/4/2MHz，内建高频 24M Hz 振荡器；
  SC92F7463B，工作电压为 2.4V~5.5V，ROM 为 16KB，RAM 为 1KB，系统时钟可选 16/8/4/1.33MHz
- 一组烧录仿真引脚
- 一个外接 16M Hz 低频晶振

SDK1014 核心板板载资源如下：

- CPU：SC92F7423，工作电压为 2.4V~5.5V，ROM 为 16KB，RAM 为 512bytes，系统时钟可选 16/8/4/1.33MHz
- 一组烧录仿真引脚
- 一个外接 32K Hz 低频晶振

SDK101x 开发板底板 SDK1010 板载资源如下：

- 一个 MicroUSB 外部供电接口
- 一个电源自锁开关，使用外部供电时控制整个板子电源
- 一个保险丝，防止芯片烧坏
- 一个电源指示灯，红光
- 一个 SPX1117-3.3 芯片，提供 3.3V 的稳压电源
- 一组 5V 电源供应口
- 一组 3.3V 电源供应口
- 一个无源蜂鸣器
- 一个四位共阴数码管
- 一个四位段码 LCD，3.3V，1/4Duty，1/3Bias
- 一个 W25Q16 存储芯片，16M 存储空间，2.7~3.6V
- 一个 RBG 三色灯
- 一个 NTC 热敏电阻
- 两个按键开关
- 两个贴片可调电阻
- 三个 LED 灯，红光
- 三个 TouchKey 按键

以上板载资源，搭配主控 IC 自带硬件资源，足够用户熟悉并掌握赛元 MCU 用法。

1.3 SDK101X 开发板软件资源汇总

为了方便用户尽快熟悉赛元 MCU，在结合开发板资源的基础上，我们提供了 MCU 相关的例程代码，用户可通过烧录接口烧录 hex 文件，可直接在开发板上观察到对应现象。

开发板的例程列表如下表所示：

例程	描述	
GPIO	SDK1011	通过配置 GPIO 来操作 LED 灯，现象为 LED1/LED2/LED3 三盏灯同时亮灭一段时间
	SDK1012	
	SDK1013	通过配置 GPIO 来操作 LED 灯，现象为 LED1 亮灭一段时间
	SDK1014	
EXTI	SDK1011	通过将 P05 配置为外部中断，下降沿触发。现象为当按下开关按键 S2 时，LED1 灯的亮灭状态翻转
	SDK1012	
	SDK1013	通过将 P07 配置为外部中断，下降沿触发。现象为当按下开关按键 S2 时，LED1 灯的亮灭状态翻转
	SDK1014	
PWM	SDK1011	通过不断循环改变 PWM40、PWM41、PWM53 的占空比，实现 RGB 三色灯的循环变色
	SDK1012	
	SDK1013	通过不断改变 PWM3 的占空比，实现 LED1 灯的呼吸灯效果
	SDK1014	
Timer0	SDK1011	使用 Timer0 定时功能，使 LED1 灯每隔一秒亮灭一次
	SDK1012	
	SDK1013	
	SDK1014	
Timer1	SDK1011	将 Timer1 设为计数器，计数值为 1。现象为当开关按键 S1 按下时，LED1 灯的亮灭状态改变一次。由于按键抖动， S1 按下一次 LED1 灯状态可能改变多次
	SDK1012	
	SDK1013	
	SDK1014	
Timer2	SDK1011	将 Timer2 设为捕获模式，将 Timer0 在 P12（SCK）口产生的方波接在 T2EX 引脚上，LED 数码管显示该方波的周期值，单位为微秒
	SDK1012	
	SDK1013	将 Timer2 设为捕获模式，将 Timer0 在 P05（SCK）口产生的方波接在 T2EX 引脚上，LED 数码管显示该方波的周期值，单位为微秒
	SDK1014	
DDIC_LED	SDK1011	启动 LED 硬件驱动电路，使 4 位数码管显示的值每秒加 1。（开发板 LCD 与数码管 SEG 口相同，故使用 LED 时 LCD 会显示乱码，正常现象）
	SDK1012	

	SDK1013 SDK1014	无 LED 硬件驱动电路，例程为软件 LED 扫描，4 位数码管显示的值每秒加 1（开发板 LCD 与数码管 SEG 口相同，故使用 LED 时 LCD 会显示乱码，正常现象）
DDIC_LCD	SDK1011 SDK1012	启动 LCD 硬件驱动电路，使 4 位 LCD 显示屏显示的值每秒加 1
	SDK1013 SDK1014	-
ACMP	SDK1011 SDK1012	模拟比较器功能，通过调节 RP1 电阻可控制比较器正端输入电压（P43 口电压），调节 RP2 电阻可控制比较器负端比较电压（P44 口电压），当正端电压大于负端电压时，LED2 灯亮，反之 LED3 灯亮
	SDK1013 SDK1014	-
BTM	SDK1011 SDK1012	低频时钟定时器，使用外部晶振，需勾选 option 项，此时有且只有 LED1 灯以 0.5s 周期闪烁，其余灯灭
	SDK1013	低频时钟定时器，使用内部晶振，LED1 灯以 0.5s 周期闪烁
	SDK1014	低频时钟定时器，使用外部晶振，LED1 灯以 0.5s 周期闪烁
ADC	SDK1011 SDK1012 SDK1013 SDK1014	ADC 采集 NTC 热敏电阻电压，转换为当前室温，数码管显示室温
IAP	SDK1011 SDK1012 SDK1013 SDK1014	通过 IAP 将数组的值写进 EEPROM，再读出 EEPROM 的值与数组的值比较，若相同，则 UART0 发送 EEPROM START 与 END 至上位机，若不相同，上位机将显示详细信息
MDU	SDK1011 SDK1012	乘除法器，若乘法结果正确，则 LED1 灯亮，若除法结果正确，则 LED2 灯亮
	SDK1013	乘除法器，若乘除法结果都正确，则 LED1 灯常亮，否则 LED1 灯闪烁
	SDK1014	-
UART0	SDK1011 SDK1012 SDK1013 SDK1014	UART0 与上位机通信，波特率 9600，上位机显示“UART0 is OK ！”
UART1	SDK1011 SDK1012 SDK1013 SDK1014	UART1 与上位机通信，波特率 9600，上位机发送数据到 MCU，MCU 将所接收到的内容发出
SPI	SDK1011 SDK1012 SDK1013	通过 SPI 操作 W25Q16，将数组数据 0~9 写入 W25Q16，再将 W25Q16 中的值读出来并通过 UART0 发送给上位机，上位机显示 0~9
	SDK1014	通过 SSI1_SPI 操作 W25Q16，将数组数据 0~9 写入 W25Q16，再将 W25Q16 中的值读出来并通过 SSI0_UART 发送给上位机，上位机显示 0~9
TWI	SDK1011 SDK1012 SDK1013 SDK1014	与 IIC 协议相同，但只可做从机，开发板无相关硬件，只提供程序配置
TK	SDK1011 SDK1012	按下 TouchKey 按键，上方对应的 LED 灯亮，蜂鸣器鸣响； SC92F7547 及 SC92F7446 无 TK 功能
	SDK1013	SC92F8463B 只有 TK24 有效，按下 TouchKey 按键，上方对应的 LED 灯亮灭，蜂鸣器鸣响； SC92F7463B 无 TK 功能
	SDK1014	-
功能复用程序	SDK1011 SDK1012	将多数软件资源整合，用户可通过按键选择所要观察的现象。 SC92F8547/SC92F8446B 通过 TK24/TK25/TK26 按键进行模式选择，S1 进入当前模式，S2 退出当前模式；

		SC92F7547/SC92F7446B 通过 S1 进行模式选择，S2 确认进入/退出当前模式
	SDK1013	将多数软件资源整合，用户可通过按键选择所要观察的现象； SC92F8463B 通过 TK24 模式选择，S2 确认进入/退出当前模式； SC92F7463B 通过 S1 模式选择，S2 确认进入/退出当前模式
	SDK1014	将多数软件资源整合，用户可通过按键选择所要观察的现象； S1 模式选择，S2 确认进入/退出当前模式

2 赛元 MCU 软硬件开发平台介绍

2.1 开发平台 KEIL C

赛元 MCU，采用 Keil C 即 KEIL uVISION 平台来开发，支持汇编语言以及 C 语言编写。

KEIL uVISION，是众多单片机应用开发软件中最优秀的软件之一，它支持众多不同公司的 MCS51 架构的芯片甚至 ARM，它集编辑，编译，仿真等于一体，界面与常用的微软 VC++界面相似，界面简洁，易学易用，在调试程序，软件仿真方面也有很强大的功能。

有关 KEIL C 的使用，请参考赛元官网资料“赛元 MCU 工具使用说明”文档的第二部分“赛元 MCU 的开发平台——Keil C”，有 Keil C 的安装及新建工程等使用说明。

2.2 烧录仿真工具

赛元目前使用的烧录工具有 SC-LINK，DPT52，PRO52。烧录工具使用前请安装赛元仿真插件。SC-LINK 适用于赛元 92F/93F 系列 IC 的脱机烧录、在线烧写、仿真以及 92F/93F 系列触控 IC 的 Touch 调试。在线工具 DPT52 适用于赛元所有系列 IC 的在线编程、触控系列 IC 的调试以及部分系列 IC 的仿真。量产编程工具 PRO52 适用于赛元所有系列 IC 的量产烧写。有关赛元烧录仿真工具的使用与仿真插件的安装，请参考赛元官网资料“赛元烧录仿真工具 SC LINK 使用说明”、“赛元在线开发工具 DPT52 使用说明”、“赛元量产编程工具 PRO52 使用说明”。

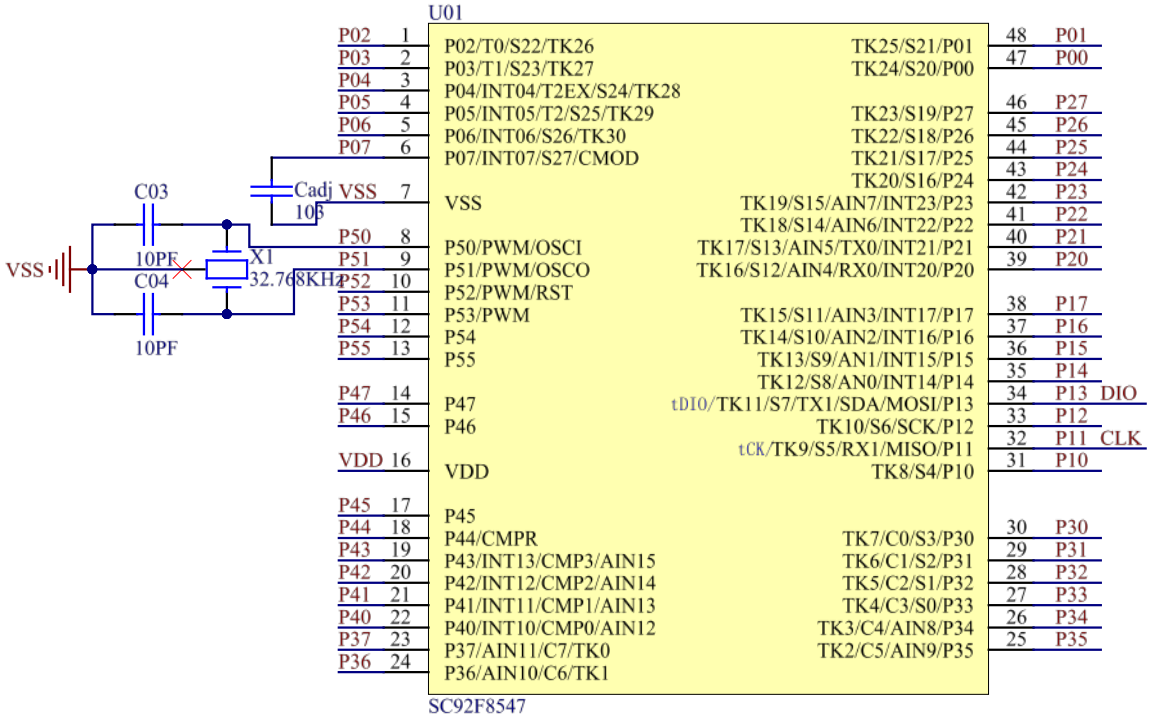
2.3 PC 端烧录软件 SOC PRO51

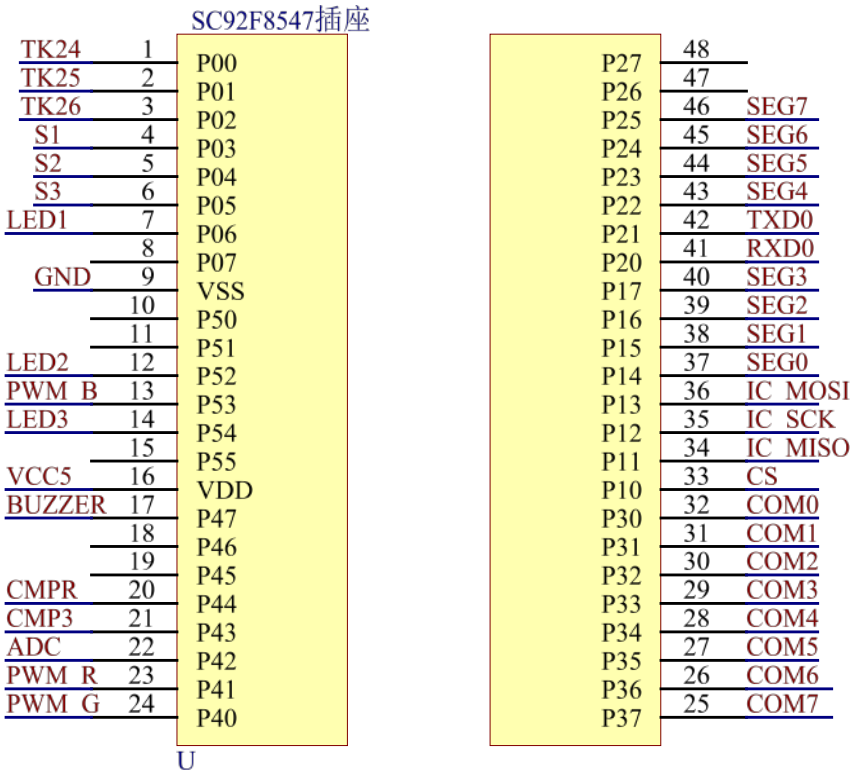
赛元 PC 端烧录上位机 SOC PRO51，支持 SC-LINK、DPT52、PRO52 等全系列烧录仿真工具。关于 SOC PRO51 的安装步骤与使用说明请参考赛元官网资料“赛元 MCU 工具使用说明”文档“1.3 软件安装步骤”与“1.4 开发工具功能说明与操作流程”（文档中烧录软件界面为旧版 V3.05 界面）。

3 开发板各模块电路图

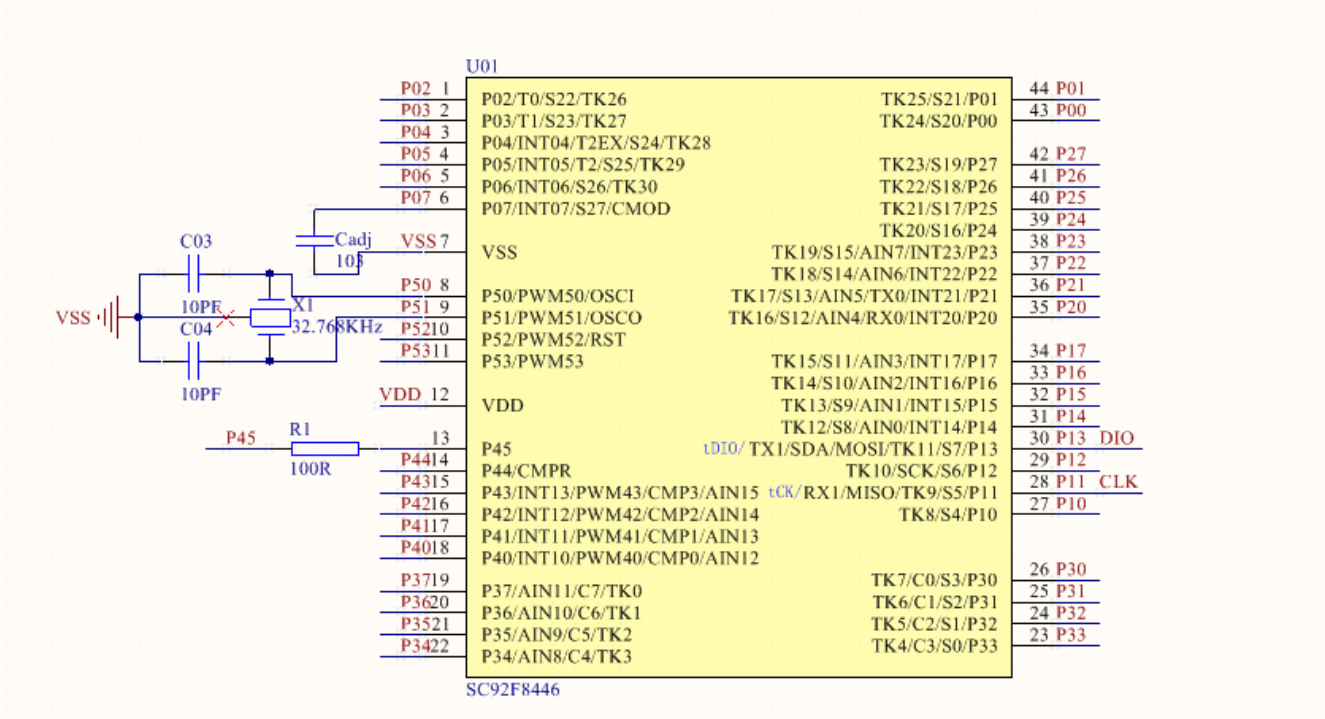
3.1 主控 MCU 及外部晶振

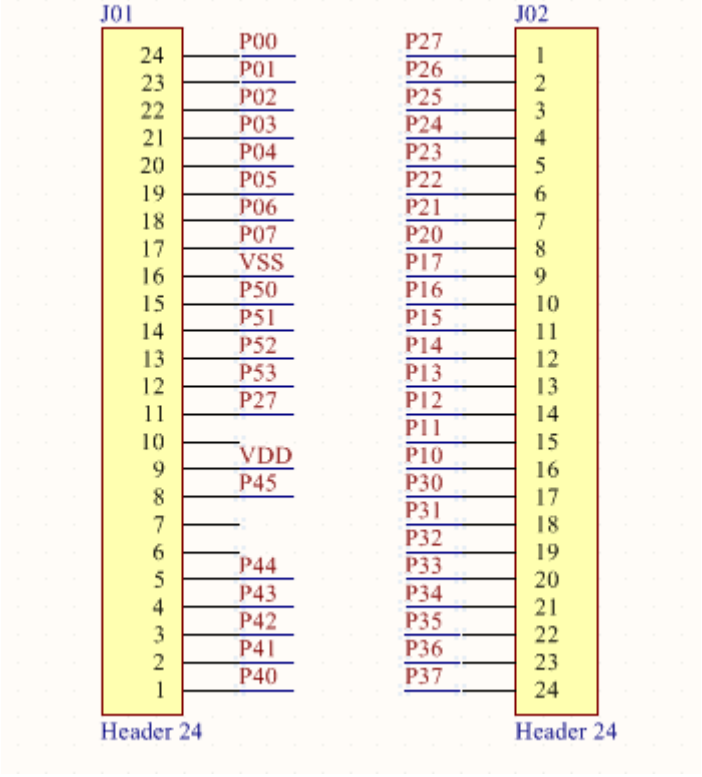
SDK1011 核心板使用 SC92F8547/7547 作为主控芯片。同时外接 32K Hz 低频晶振，可作为 BTM 低频时钟定时器振荡器。



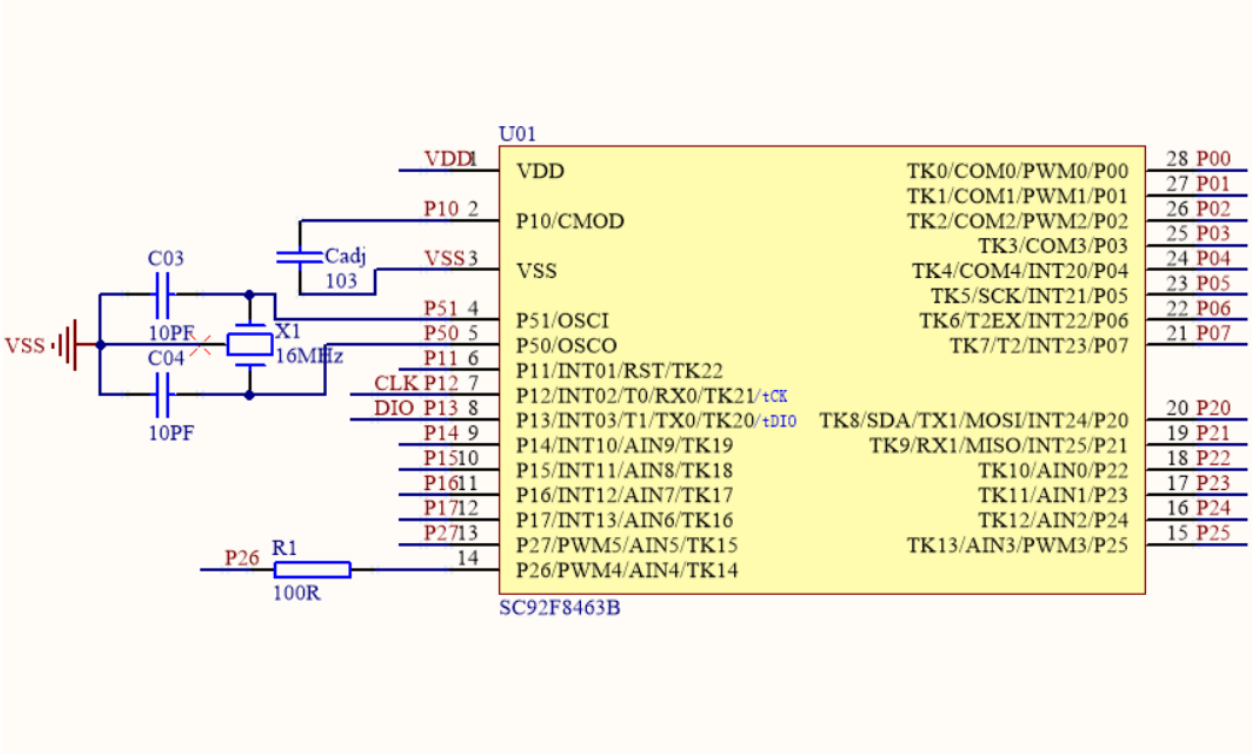


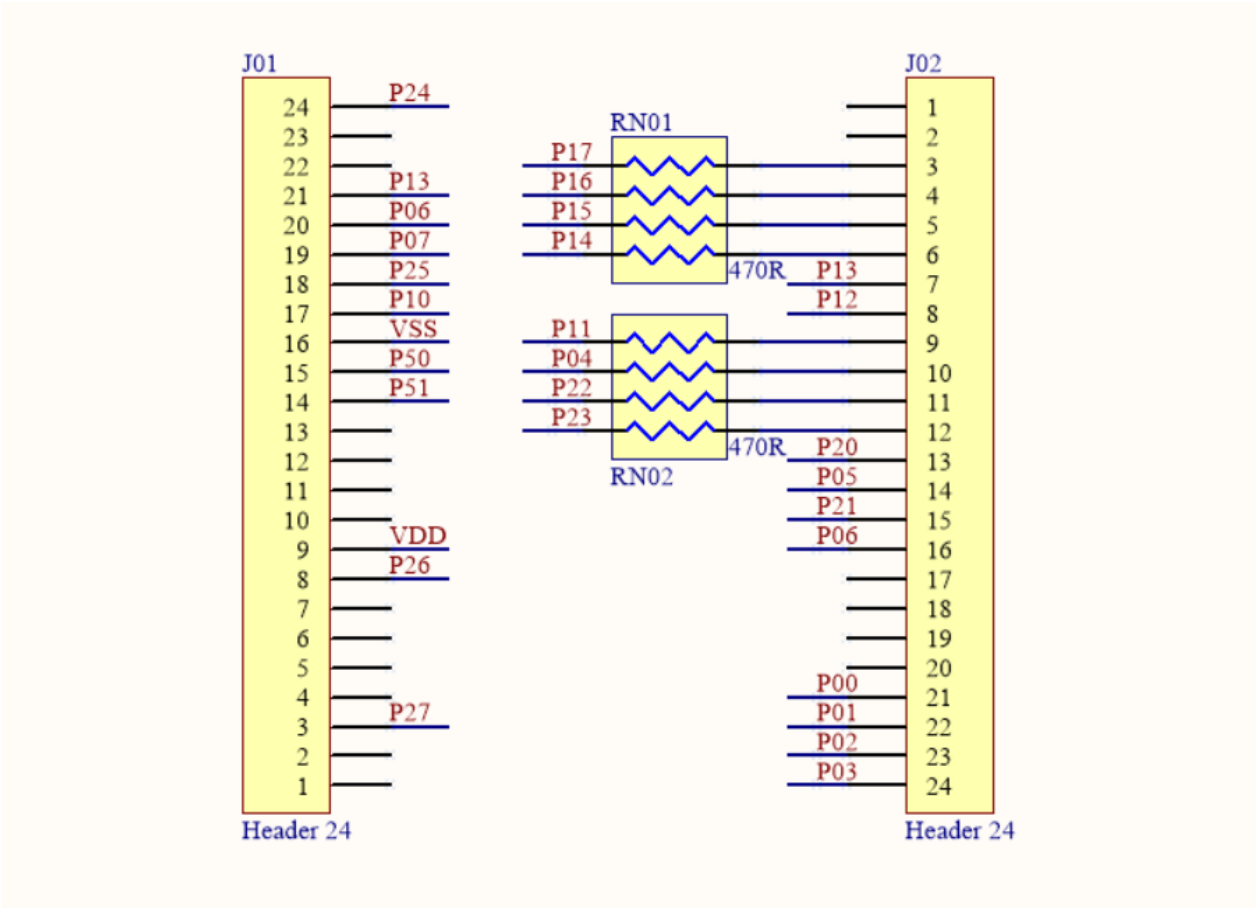
SDK1012 核心板使用 SC92F8446B/7446B 作为主控芯片。同时外接 32KHz 低频晶振，可作为 BTM 低频时钟定时器振荡器。





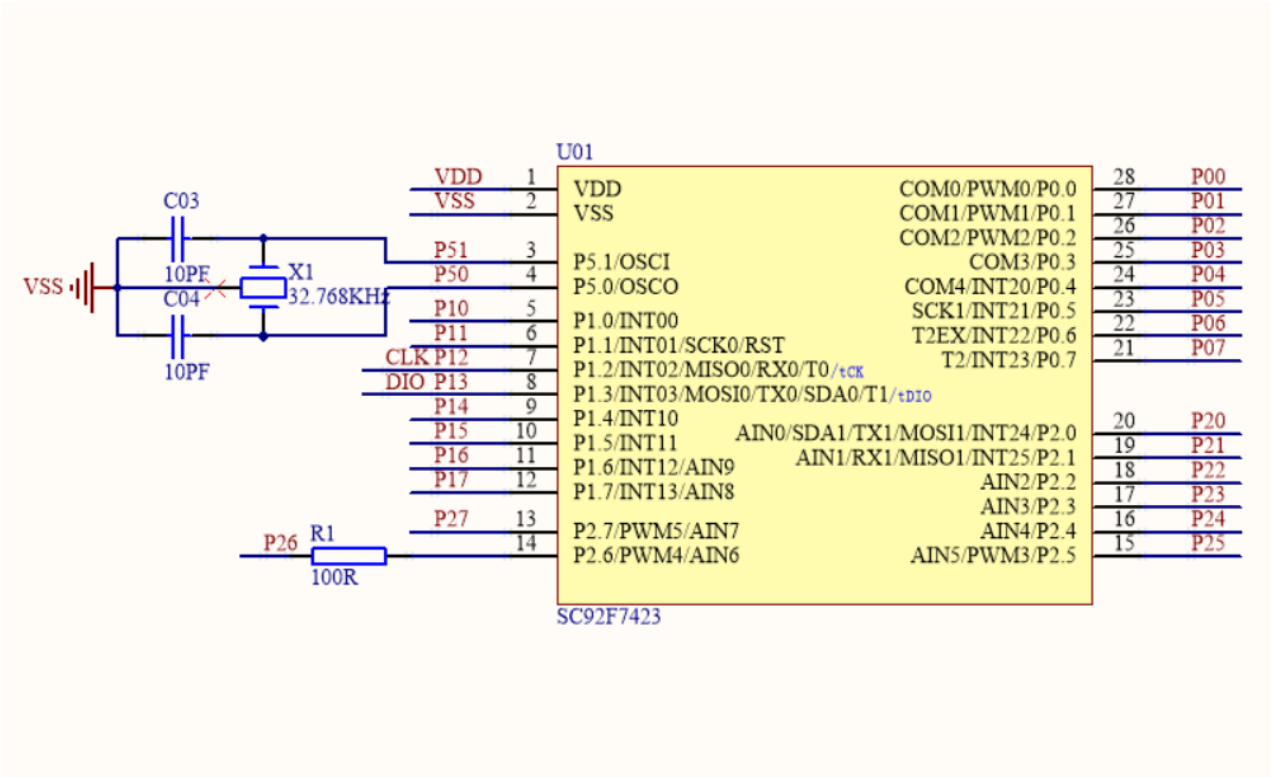
SDK1013 核心板使用 SC92F8463B/7463B 作为主控芯片。同时外接 16MHz 高频晶振，可作为系统时钟。

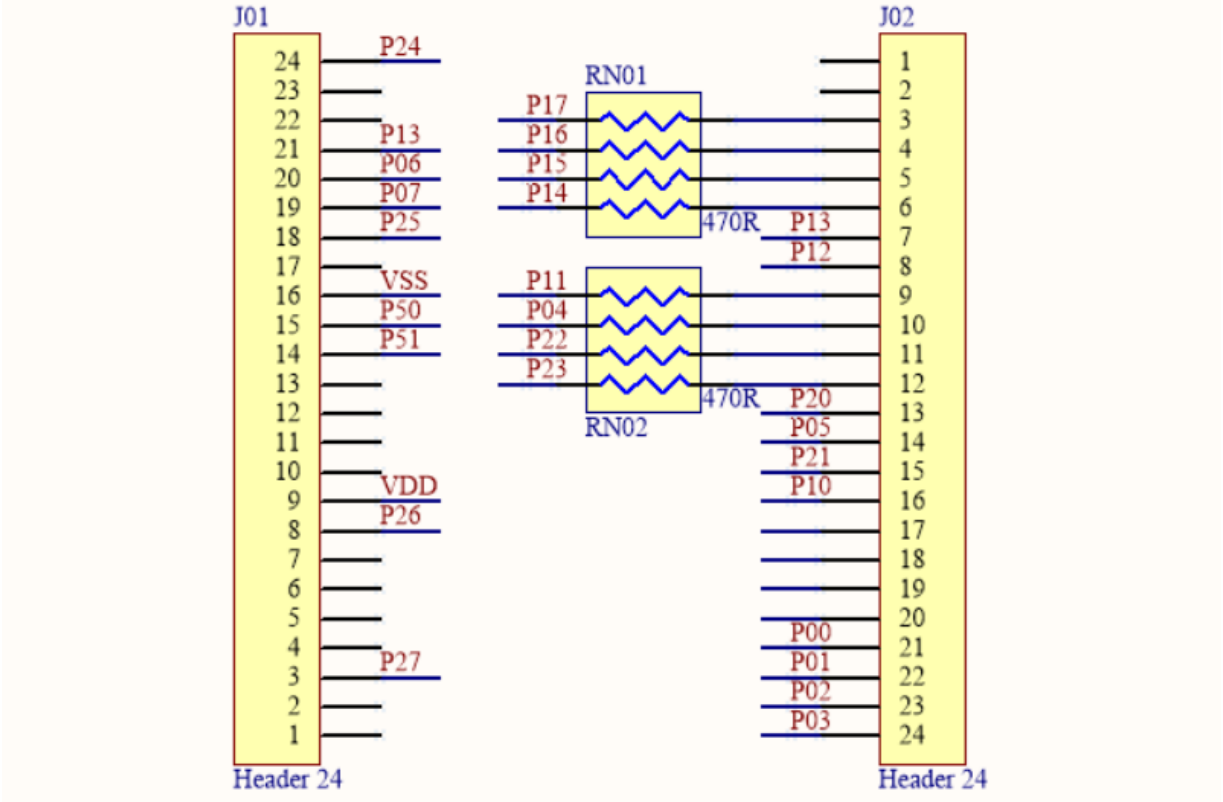




SDK1014 核心板使用 SC92F7423 作为主控芯片。同时外接 32KHz 低频晶振，可作为 BTM 低频时钟定时器

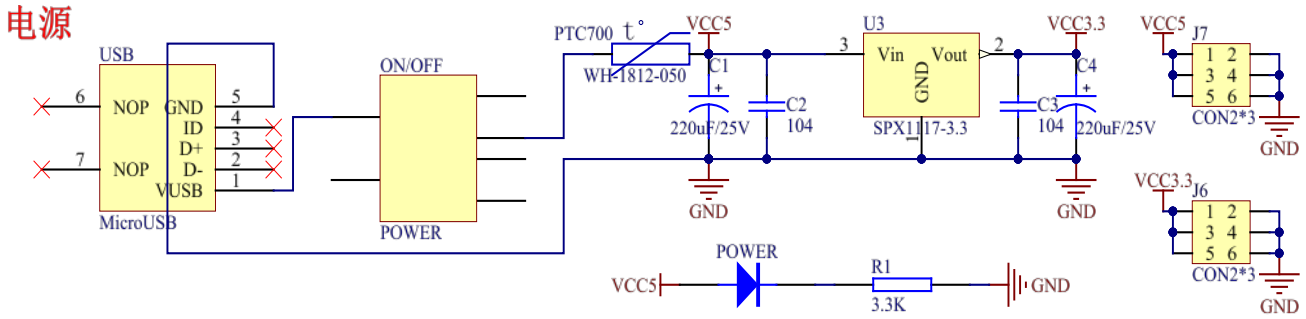
振荡器。





3.2 电源模块

外接 MicroUSB 时，可通过自锁开关控制开发板电源，由电源指示灯 POWER 亮灭指示电源通断。通过 SPX1117-3.3 芯片将 V_{DD} 降压为 3.3V，并提供 V_{DD} 与 3.3V 的电源供给接口。

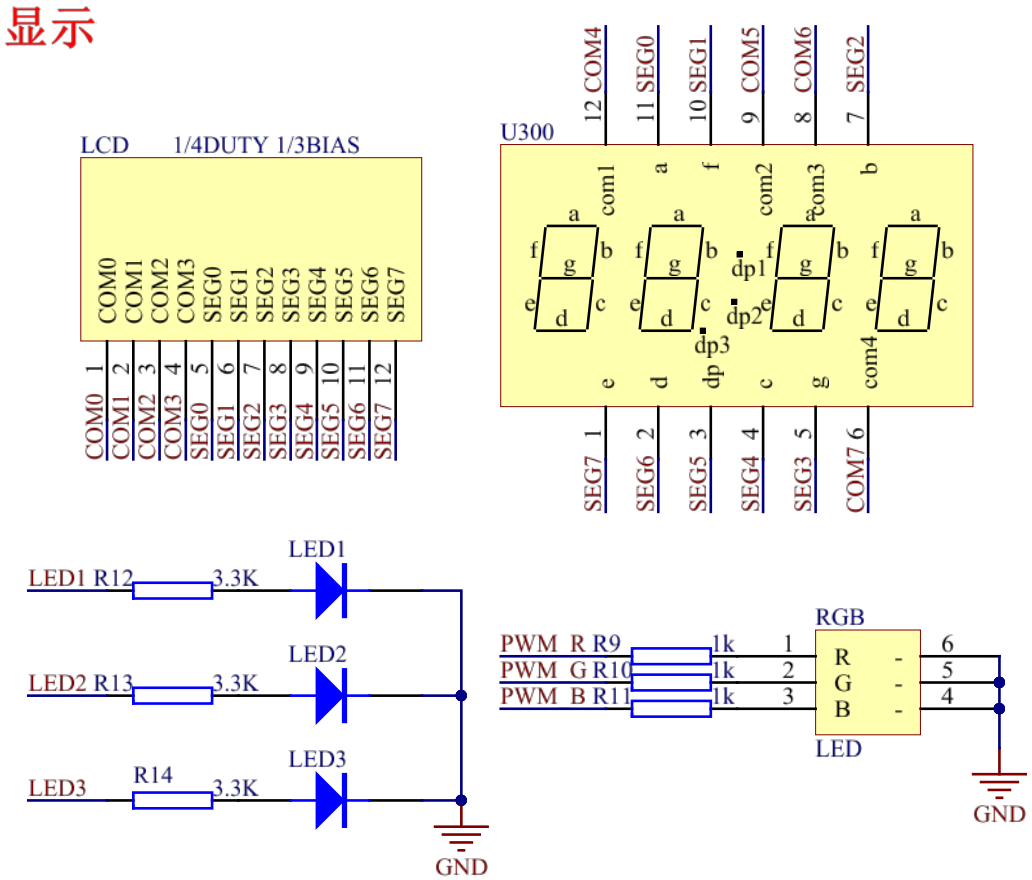


3.3 LED/RGB/LCD/数码管显示单元

SDK1010 提供三个 LED 灯，一个三色灯 RGB，一个 1/4duty、1/3bias LCD 显示屏与四位 LED 数码管。其中，LCD 显示屏与四位 LED 数码管共用 SEG 口，同一时刻只可使用 LCD 或 LED。

SDK1011/1012 核心板可控制三个 LED 灯，PWM 接口与三色灯 RGB 相连，可调节 PWM 占空比决定当前 RGB 颜色。SDK1011/1012 开发板上数码管和 LCD 屏的 SEG 口及 COM 口与 MCU 硬件显示驱动电路相连。

SDK1013/1014 的主控 IC 为 28pin MCU，资源较 SDK1011/1012 少，因此 SDK1013/1014 只可控制一个 LED 灯接口——LED1，此灯与 PWM 接口相连，可调节 PWM 占空比以调节 LED 灯亮度，这两款核心板均不提供 RGB 灯驱动接口。SDK1013/1014 的主控 IC 无硬件显示驱动电路，只支持 1/2bias LCD 屏软件驱动，又因 SDK1010 主板板资源只提供 1/3bias LCD 屏，故 SDK1013/1014 只提供 LED 数码管显示驱动接口，不提供 LCD 屏显示驱动接口。



3.4 按键与蜂鸣器

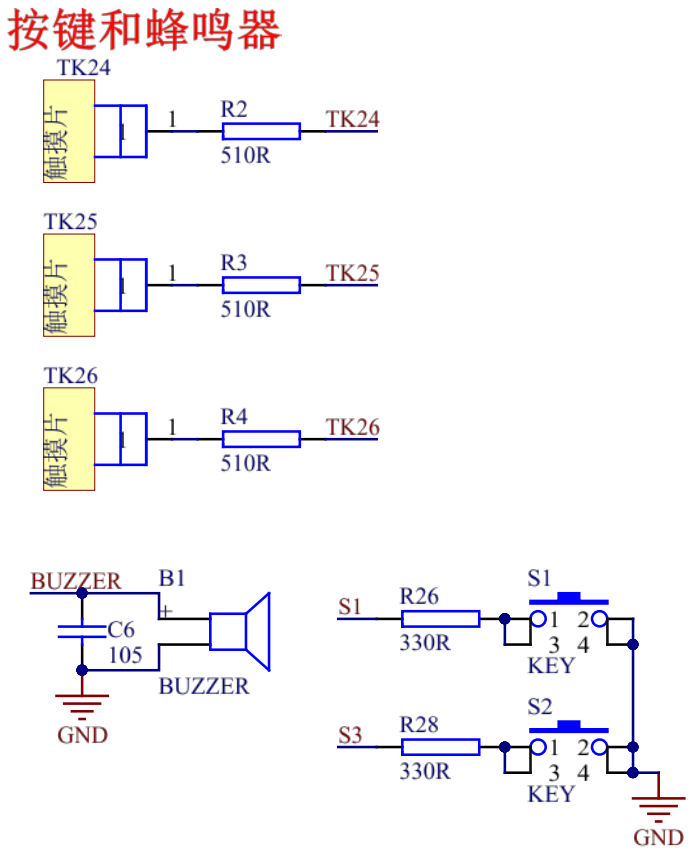
开发板提供三个 TouchKey 按键，帮助用户熟悉 MCU 的 TouchKey 功能。提供两个开关按键，按键 S1 与 Timer1 外部输入脚 T1 相连，按键 S2 与 Timer2 外部输入脚 T2 相连。各核心板使用 IO 口直接驱动无源蜂鸣器，但提供的 IO 不同。

SDK1011 核心板 MCU SC92F8547 提供三个 TouchKey 按键接口，SC92F7547 无 TK 功能。

SDK1012 核心板 MCU SC92F8446B 提供三个 TouchKey 按键接口，SC92F7446B 无 TK 功能。

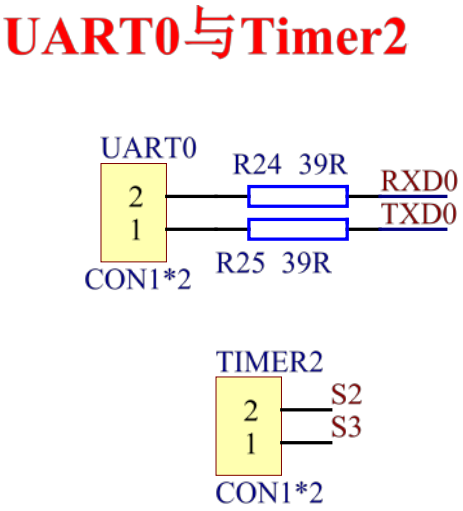
SDK1013 核心板 MCU SC92F8463B IO 口资源较少，仅提供一个 TouchKey 按键接口与 TK24 相连，SC92F7463B 无 TK 功能。

SDK1014 开发板 MCU SC92F7423 无 TK 功能。



3.5 UART0 与 TIMER2

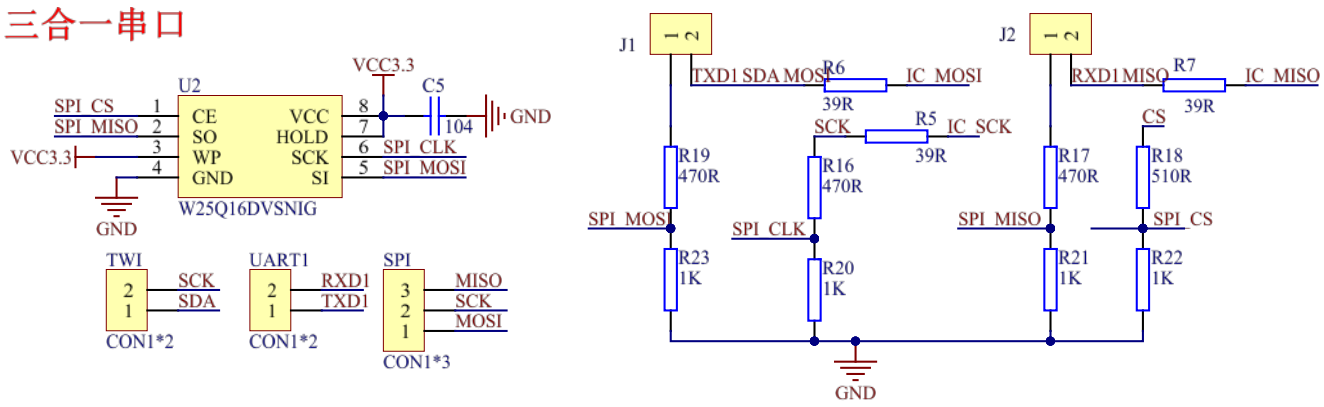
MCU 支持一路全双工的串行口 UART0，通讯模式可选模式 0、模式 1、模式 3，TXD0 与 RXD0 引脚接出如下。SDK1014 的 MCU 为 SC92F7423,其有两个三合一串行接口 SSI，使用 UART0 时将 SSI0 配置为 UART 模式即可，但只支持模式 1 和模式 3，TXD0 与 RXD0 引脚接出相同。其同时，将 Timer2 外部输入脚 T2 与外部捕获输入脚 T2EX 引出。



3.6 三选一串行接口 SSI

MCU 内部集成三选一串行接口 SSI，可将 SSI 配置为 SPI、TWI 和 UART1 其中一种。TWI 兼容 IIC，但只可做从机。串口 UART1 支持模式 1 与模式 3。SPI 接口与 W25Q16 存储芯片相连，使用前需接上跳线帽。

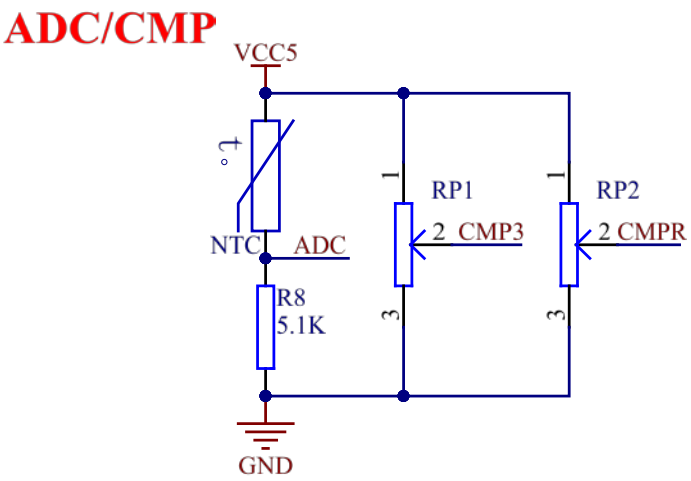
SDK1014 的 MCU 为 SC92F7423,其有两个三合一串行接口 SSI，使用 SDK1014 时，SSI 资源指 SSI1 资源。



3.7 ADC 与 CMP

VCC 连接 NTC 热敏电阻与 5.1K 电阻 R8 到地，ADC 引脚采集 NTC 与 R8 之间电压，热敏电阻阻值随温度变化，ADC 采样值亦随之变化。贴片电阻 RP1 连接模拟比较器正输入端 CMP3，RP2 连接模拟比较器负输入端 CMPR，可调节 RP1、RP2 阻值使正负端电压变化。

SDK1013/1014 不提供模拟比较器资源接口。



4 SDK101X 开发板示例

4.1 GPIO 示例

GPIO 端口可配置为三种模式，包括强推挽输出模式，带上拉的输入模式，高阻输入模式。GPIO 口模式配置由寄存器 PxCON 与 PxPH 控制，PxCON 控制 IO 为输入或输出模式，当端口作为输入时，每个 IO 端口带有由 PxPH 控制的内部上拉电阻。

强推挽输出模式下，能够提供持续的大电流驱动。带上拉的输入模式下，输入口上恒定接一个上拉电阻，仅当输入口上电平被拉低时，才会检测到低电平信号。

4.1.1 SDK1011/1012GPIO 示例

SDK1011 核心板 MCU SC92F8547/7547 提供了最多 46 个可控制的双向 GPIO 端口，SDK1012 核心板 MCU SC92F8446B/7446B 提供了最多 42 个可控制的双向 GPIO 端口。

GPIO 示例通过配置 GPIO 为强推挽输出模式，驱动 3 个 LED 灯，现象为 LED 灯以固定的频率闪烁。

```
void main(void)
{
    uint16_t i,j;
    GPIO_Init(GPIO0,GPIO_PIN_6,GPIO_MODE_OUT_PP);    //将P06设置为强推挽输出

    while(1)
    {
        GPIO_WriteHigh(GPIO0,GPIO_PIN_6);            //P06写高
        for(i=0;i<100;i++)                           //延时一段时间
        {
            for(j=0;j<3000;j++);
        }

        GPIO_WriteLow(GPIO0,GPIO_PIN_6);             //P06写低
        for(i=0;i<100;i++)                           //延时一段时间
        {
            for(j=0;j<3000;j++);
        }
    }
}
```

示例首先将与 LED 相连的三个 IO 端口配置为强推挽输出模式，在循环中将 IO 口高低电平状态以一定的延时不断改变。

4.1.2 SDK1013/1014 GPIO 示例

SDK1013 核心板 MCU SC92F8463B/7463B 和 SDK1014 核心板 MCU SC92F7423 提供了最多 26 个可控制的双向 GPIO 端口。

GPIO 示例通过配置 GPIO 为强推挽输出模式，驱动 LED 灯，现象为 LED1 以固定的频率闪烁。

```
void main(void)
{
    uint16_t i,j;
    GPIO_Init(GPIO2,GPIO_PIN_5,GPIO_MODE_OUT_PP);    //将P25设置为强推挽输出

    while(1)
    {
        GPIO_WriteHigh(GPIO2,GPIO_PIN_5);            //P25写高
        for(i=0;i<100;i++)                           //延时一段时间
        {
            for(j=0;j<3000;j++);
        }

        GPIO_WriteLow(GPIO2,GPIO_PIN_5);             //P25写低
        for(i=0;i<100;i++)                           //延时一段时间
        {
            for(j=0;j<3000;j++);
        }
    }
}
```

4.2 外部中断示例

外部中断 INT0~2，当外部中断口有中断条件发生时，外部中断就被触发。用户可以根据需要设成上沿、下沿或者双沿中断，可通过设置 SFR （INTxF 和 INTxR）来实现。用户可通过 IP 寄存器来设置每个中断的优先级级别。外部中断 INT0~2 还可以唤醒单片机的 STOP。

4.2.1 SDK1011/1012 EXTI 示例

```
void main(void)
{
    GPIO_Init(GPIO0, GPIO_PIN_6, GPIO_MODE_OUT_PP);              //设置P00,P06为强推挽输出

    GPIO_Init(GPIO0, GPIO_PIN_5, GPIO_MODE_IN_PU);               //设置INT05：P05为输入带上拉
    EXTI_SetExtInt0xTriggerMode(INT05, EXTI_TRIGGER_FALL_ONLY); //设置INT0下降沿中断
    EXTI_ITConfig(INT0, ENABLE, LOW);                            //使能INT0，低中断优先级

    enableInterrupts();                                          //开总中断

    while(1)
    {
    }
}
void EXTI0Interrupt()       interrupt 0
{
    if(GPIO_ReadPin(GPIO0,GPIO_PIN_6))          //翻转IO
    {
        GPIO_WriteLow(GPIO0,GPIO_PIN_6);
    }
    else
    {
        GPIO_WriteHigh(GPIO0,GPIO_PIN_6);
    }
}
```

外部中断示例通过将 IO 口 P05 配置为带上拉的输入模式，外部中断模式为下降沿触发的外部中断并使能外部中断 0 及总中断，而 P05 通过按键 S2 与地相连。当开关按键 S2 按下时产生一个下降沿，外部中断发生，此时在外部中断子程序中改变 LED1 灯的亮灭状态。主意，按键按下时产生的抖动可能会使灯的状态多次改变。

4.2.2 SDK1013/1014 EXTI 示例

```
void main(void)
{
    GPIO_Init(GPIO2, GPIO_PIN_5, GPIO_MODE_OUT_PP);              //设置P25为强推挽输出

    GPIO_Init(GPIO0, GPIO_PIN_7, GPIO_MODE_IN_PU);               //设置INT23：P07为输入带上拉
    EXTI_SetExtInt2xTriggerMode(INT23, EXTI_TRIGGER_FALL_ONLY); //设置INT23下降沿中断
    EXTI_ITConfig(INT2, ENABLE, LOW);                            //使能INT2，低中断优先级

    enableInterrupts();                                          //开总中断

    while(1);
}
void EXTI2Interrupt()        interrupt 10
{
    if(GPIO_ReadPin(GPIO2,GPIO_PIN_5))       //翻转IO
    {
        GPIO_WriteLow(GPIO2,GPIO_PIN_5);
    }
    else
    {
        GPIO_WriteHigh(GPIO2,GPIO_PIN_5);
    }
}
```

外部中断示例通过将 IO 口 P07(INT23)配置为带上拉的输入模式，外部中断模式为下降沿触发的外部中断并使能外部中断 2 及总中断，而 P07 通过按键 S2 与地相连。当开关按键 S2 按下时产生一个下降沿，外部中断发生，此时在外部中断子程序中改变 LED1 灯(P25)的亮灭状态。

4.3 PWM 示例

4.3.1 SDK1011 /1012 PWM 示例

SC92F8547/7547、SC92F8446B/7446B 提供了最多 8 路共用周期、单独可调占空比的 12 位 PWM 输出.。

PWM 具有的功能为：

1. 12 位 PWM 精度
2. 8 路 PWM 周期相同，但占空比可单独设置
3. 输出可设置正反向

PWM 可支持周期及占空比的调整，寄存器 PWMCG、PWMCON 控制 PWM 的状态及周期，各路 PWM 的打开及输出波形占空比可单独调整，需注意：**PWM 占空比调节寄存器只可写，不可读。**

示例通过 3 路输出的 PWM 波形控制 RGB 三色灯，现象为三色灯循环变色。

```
void PWM_INIT(void)
{
    PWM_DeInit();
    PWM_Init(PWM_PRESSEL_FOSC_D2, 159);                  //PWM时钟源1分频，周期为（599+1）*2/FOSC
    PWM_OutputStateConfig(PWM40, PWM_OUTPUTSTATE_ENABLE);//使能PWM40
    PWM_OutputStateConfig(PWM41, PWM_OUTPUTSTATE_ENABLE);//使能PWM41
    PWM_OutputStateConfig(PWM53, PWM_OUTPUTSTATE_ENABLE);//使能PWM53
    PWM_IndependentModeConfig(PWM40, 0);                 //PWM40占空比设置为0
    PWM_IndependentModeConfig(PWM41, 0);                 //PWM41占空比设置为0
    PWM_IndependentModeConfig(PWM53, 0);                 //PWM53占空比设置为0
    PWM_Cmd(ENABLE);                                     //打开PWM总开关
    PWM_ITConfig(ENABLE, LOW);                           //打开PWM中断
}

void Delay_Some_Time(uint16_t Some_Time)
{
    uint16_t i,j;
    for(i=Some_Time;i>0;i--)
    for(j=100;j>0;j--);
}
```

```
void main(void)
{
    uint16_t i;
    PWM_INIT();

    while(1)
    {
        for(i=0;i<80;i++)
        {
            PWM_IndependentModeConfig(PWM40, i);
            Delay_Some_Time(30);
            PWM_IndependentModeConfig(PWM41, 80-i);
            Delay_Some_Time(30);
            PWM_IndependentModeConfig(PWM53, i);
            Delay_Some_Time(30);
        }
        for(i=0;i<80;i++)
        {
            PWM_IndependentModeConfig(PWM40, 80-i);
            Delay_Some_Time(30);
            PWM_IndependentModeConfig(PWM41, i);
            Delay_Some_Time(30);
            PWM_IndependentModeConfig(PWM53, 80-i);
            Delay_Some_Time(30);
        }
    }
}
```

示例首先设置 PWM 时钟源为 F_{OSC}/2，周期为（159+1）/(F_{OSC}/2) = 20μs，在主循环中，对与 RGB 相连的三路 PWM 使能，输出的 PWM 波形不反向，占空比随 for 循环不断改变，从而实现三色灯的循环变色。

4.3.2 SDK1013 /1014 PWM 示例

SC92F8463B/7463B 提供了最多 6 路共用周期、单独可调占空比的 10 位 PWM 输出.。

SC92F8463B/7463B 的 PWM 具有的功能为：

1. 10 位 PWM 精度；
2. 输出可设置正反向；
3. 分为独立模式和互补模式：
   1） 独立模式下，PWM0-5 周期相同，但每一路 PWM 输出波形的占空比单独可设置；
   2） 互补模式下可同时输出三组互补、带死区的 PWM 波形；
4. 提供 1 个 PWM 溢出的中断；

SC92F7423 提供了最多 6 路共用周期、单独可调占空比的 8 位 PWM 输出.。

SC92F7423 的 PWM 具有的功能为：

1. 8 位 PWM 精度；
2. 输出可设置正反向；
3. PWM0-5 周期相同，但占空比单独可设置；
4. 提供 1 个 PWM 溢出的中断；

PWM 支持周期调整，各路 PWM 的打开及输出波形占空比单独调整。

以 SDK1013 为例，通过配置 PWM 工作在独立模式下，调整 PWM3 输出不同占空比波形控制灯 LED1，现象为红色呼吸灯。

```
uint8_t code BrightAdjust[47]=
{
    1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
    2,2,2,2,2,2,2,2,2,2,2,2,2,2,
    3,3,3,3,3,3,3,3,3,3,3,
};

void PWM_INIT(void)
{
    PWM_DeInit();

    //PWM时钟源1分频，周期为（159+1）*2/FOSC
    PWM_Init(PWM_PRESSEL_FOSC_D2, 159);
    PWM_OutputStateConfig(PWM3, PWM_OUTPUTSTATE_ENABLE);//使能PWM3
    PWM_IndependentModeConfig(PWM3, 0);                 //PWM3占空比设置为0
    PWM_Cmd(ENABLE);                                    //打开PWM总开关
}

void Delay_Some_Time(uint16_t Some_Time)
{
    uint16_t i,j;
    for(i=Some_Time;i>0;i--)
    for(j=100;j>0;j--);
}
void main(void)
{
    uint8_t i = 0,Brightness = 0;

    PWM_INIT();

    while(1)
    {
        for(i=0;i<47;i++)
        {
            Brightness = Brightness + BrightAdjust[i];
            PWM_IndependentModeConfig(PWM3, Brightness);
            Delay_Some_Time(100);
        }
        Delay_Some_Time(300);
        for(i=45;i>0;i--)
        {
            Brightness = Brightness - BrightAdjust[i];
            PWM_IndependentModeConfig(PWM3, Brightness);
            Delay_Some_Time(200);
        }
        PWM_IndependentModeConfig(PWM3, 0);
        Brightness = 0;
        Delay_Some_Time(3000);
    }
}
```

4.4 TIMER0 示例

Timer0 是一个 16 位的定时计数器，具有计数方式和定时方式两种工作模式，特殊功能寄存器 TMOD 中有一个控制位 C/T0 来选择 T0 是定时器还是计数器，T0 本质上都是一个加法计数器，只是计数的来源不同。定时器的来源为系统时钟或者其分频时钟，但计数器的来源为外部管脚的输入脉冲。只有在 TR0=1 的时候，T0 才会被打开计数。

T0 有 4 种工作模式

1. 模式 0：13 位定时器/计数器模式
2. 模式 1：16 位定时器/计数器模式
3. 模式 2：8 位自动重载模式

4. 模式 3：两个 8 位定时器/计数器模式

下面以 SDK1011 核心板为例，示例为演示 Timer0 的定时功能，现象为 LED1 灯每 600 毫秒亮灭状态改变一次。

```
void Timer0Init()
{
    TIM0_TimeBaseInit(TIM0_PRESSEL_FSYS_D12,TIM0_MODE_TIMER);   //Timer0 12分频,做定时器
    TIM0_WorkMode0Config(4192);                                 //设置Timer0工作模式0，初值为4192，定时3ms
    TIM0_ITConfig(ENABLE, LOW);                                 //使能Timer0中断
    TIM0_Cmd(ENABLE);                                           //使能Timer0计数
}

void IOInit()
{
    GPIO_Init(GPIO0, GPIO_PIN_6, GPIO_MODE_OUT_PP); //P00强推输出模式
}

void main(void)
{
    IOInit();
    Timer0Init();
    enableInterrupts();
    while(1);
}

void Timer0Interrupt()      interrupt 1
{
    if(++T0CNT == 200)
    {
        T0CNT = 0;
        TIM0_Mode0SetReloadCounter(4192);
        if(GPIO_ReadPin(GPIO0, GPIO_PIN_6))     //LED1灯翻转
        {
            GPIO_WriteLow(GPIO0,GPIO_PIN_6);
        }
        else
        {
            GPIO_WriteHigh(GPIO0,GPIO_PIN_6);
        }
    }
}
```

T0 设置为定时器模式，工作方式为模式 0，时钟源 12 分频，初值为（8912-4000），即 3 毫秒进入一次中断，中断标志位累加到 200，即 600 毫秒时翻转 LED1 灯的状态。

4.5 TIMER1 示例

Timer1 是一个 16 位的定时计数器，具有计数方式和定时方式两种工作模式，特殊功能寄存器 TMOD 中有一个控制位 C/T1 来选择 T1 是定时器还是计数器，T1 本质上都是一个加法计数器，只是计数的来源不同。定时器的来源为系统时钟或者其分频时钟，但计数器的来源为外部管脚的输入脉冲。只有在 TR1=1 的时候，T1 才会被打开计数。

T1 有 3 种工作模式

1. 模式 0：13 位定时器/计数器模式
2. 模式 1：16 位定时器/计数器模式
3. 模式 2：8 位自动重载模式

下面以 SDK1011 核心板为例，示例为演示 Timer1 的计数功能，现象为每按下开关 S1，LED1 灯亮灭状态改变一次。

```
extern bit T1FLG;

void Timer1Init()
{
    TIM1_TimeBaseInit(TIM1_PRESSEL_FSYS_D1,TIM1_MODE_COUNTER);  //Timer1不分频,做计数器
    TIM1_WorkMode1Config(65535);                                //设置Timer1工作模式1，初值为65535
    TIM1_Cmd(ENABLE);                                           //使能Timer1计数
    TIM1_ITConfig(ENABLE, LOW);                                 //使能Timer1中断
}

void IOInit()
{
    GPIO_Init(GPIO0, GPIO_PIN_6, GPIO_MODE_OUT_PP); //P06强推输出模式
    GPIO_Init(GPIO0, GPIO_PIN_3, GPIO_MODE_IN_PU);
}

void main(void)
{
    IOInit();           //IO初始化
    Timer1Init();       //Timer1初始化

    enableInterrupts(); //开总中断

    while(1)
    {
        if(T1FLG)
        {
            T1FLG = 0;
            if(GPIO_ReadPin(GPIO0, GPIO_PIN_6))
            {
                GPIO_WriteLow(GPIO0, GPIO_PIN_6);
            }
            else
            {
                GPIO_WriteHigh(GPIO0, GPIO_PIN_6);
            }
        }
    }
}
void Timer1Interrupt()      interrupt 3
{
    TIM1_WorkMode1Config(65535);
    T1FLG = 1;
}
```

T1 设置为计数器模式，工作方式为模式 1，时钟源不分频，初值为（65536-1），即计数一次进入中断，中断标志置起后，翻转 LED1 灯的状态。

4.6 TIMER2 示例

Timer2 具有计数方式和定时方式两种工作模式。特殊功能寄存器 T2CON 中有一个控制位 C/T2 来选择 T2 是定时器还是计数器。它们本质上都是一个加法计数器，只是计数的来源不同。定时器的来源为系统时钟或者其分频时钟，但计数器的来源为外部管脚的输入脉冲。TR2 是 T2 在定时器/计数器模式计数的开关控制，只有在 TR2=1 的时候，T2 才会被打开计数。

计数器模式下，T2 管脚上的每一个脉冲，T2 的计数值分别增加 1。

定时器模式下，可通过特殊功能寄存器 TMCON 来选择 T2 的计数来源是 $f_{SYS}/12$ 或 f_{SYS}。

定时器/计数器 T2 有 4 种工作模式：

1. 模式 0：16 位捕获模式
2. 模式 1：16 位自动重载定时器模式
3. 模式 2： 波特率发生器模式（SC92F7423 无此模式）
4. 模式 3：可编程时钟输出模式

示例将 Timer2 配置为 16 位捕获模式，四位数码管作显示用，用 Timer0 与 SCK 口产生的一个周期合适的方波，接在 T2EX 上，数码管显示该方波的周期，单位为 us。

```
void TIMER2_Init(void)
{
    TIM2_TimeBaseInit(TIM2_PRESSEL_FSYS_D1, TIM2_MODE_TIMER, TIM2_COUNTDIRECTION_UP);
    TIM2_WorkMode0Config(0);      //工作模式设置为捕获模式
    TIM2_Cmd(ENABLE);
    TIM2_ITConfig(ENABLE, HIGH); //打开中断，优先级设置为高
}

void DDICInit(void)
{
    //占空比1/4,P14~P17,P22~P25,P34~P37为DDIC输出脚
    DDIC_Init(DDIC_DUTYCYCLE_D4, 0X00, 0XF0, 0X3C, 0XF0);
    DDIC_LEDConfig();           //DDIC配置为LED模式
    DDIC_OutputPinOfDutycycleD4(SEG0_27COM4_7);       //占空比1/4时SEG为S0~S27,COM为C4~C7
    DDIC_Cmd(ENABLE);           //使能
}

void main(void)
{
    GPIO_Init(GPIO1, GPIO_PIN_2, GPIO_MODE_OUT_PP);
    DDICInit();
    TIMER2_Init();
    TIMER0_Init();
    enableInterrupts();

    while(1)
    {
        if(T0Flag120ms)
        {
            T0Flag120ms = 0;
            if(CaptureState & 0X80)
            {
                LedDataTab[0] = (uint16_t)Cycle / 1000;    //显示数据装载到数组
                LedDataTab[1] = (uint16_t)Cycle / 100 % 10;
                LedDataTab[2] = (uint16_t)Cycle / 10 % 10;
                LedDataTab[3] = (uint16_t)Cycle % 10;
                CaptureState=0;
            }
            Led_Display();
        }
    }
}
```

```
void Timer2Interrupt()      interrupt 5
{
    static uint16_t CaptureValue1, CaptureValue2;
    static uint16_t ReloadCount = 0;
    if((CaptureState&0X80) == 0)
    {
        if(TIM2_GetFlagStatus(TIM2_FLAG_TF2))
        {
            if(CaptureState & 0X40)
            ReloadCount++;                  //ReloadCount为T2计数溢出次数
        }
        if(TIM2_GetFlagStatus(TIM2_FLAG_EXF2))
        {
            if(CaptureState & 0X40)
            {
                //CaptureValue2为捕获第二个下降沿时记录的捕获值
                CaptureValue2 = ((uint16_t)RCAP2H << 8) + RCAP2L;
                //捕获的方波周期为 （T2溢出时间+第二次捕获值-第一次捕获值）
                Cycle = ((65536*ReloadCount) + CaptureValue2 - CaptureValue1) / 16;
                TL2 = 0X00;     //清除计数值
                TH2 = 0X00;
                CaptureState |= 0X80;
                ReloadCount = 0;
            }
            else
            {
                CaptureValue1 = ((uint16_t)RCAP2H << 8) + RCAP2L;
                CaptureState = 0X40;
                ReloadCount = 0; //CaptureValue1为捕获第一个下降沿时记录的捕获值
            }
        }
    }
    if(TIM2_GetFlagStatus(TIM2_FLAG_TF2))
    {
        TIM2_ClearFlag(TIM2_FLAG_TF2);
    }
    if(TIM2_GetFlagStatus(TIM2_FLAG_EXF2))
    {
        TIM2_ClearFlag(TIM2_FLAG_EXF2);
    }
}
```

T2EX 上的下降沿触发 T2 捕获模式的中断，在中断服务程序中记录当前计数值，在下一个下降沿到来后记录计数值，此时计数值相减即可求周期。

4.7 LED 显示驱动示例

SC92F8547/7547、SC92F8446B/7446B 内部集成了硬件的 LCD/LED 显示驱动电路，可方便用户实现 LCD 和 LED 的显示驱动。其主要特点如下：

1. LCD 和 LED 显示驱动二选一；
2. LCD 和 LED 显示驱动共用相关 IO 口和寄存器。

LED 显示驱动功能如下：

1. 4 种显示驱动模式可选：8 X 24、6 X 26、5 X 27、或 4X 28 段；
2. seg 口驱动能力 4 级可选；
3. 显示驱动电路可选择内建 128K LRC 或外部 32K 振荡器作为时钟源，帧频约为 64Hz。

示例 code 启动 LED 硬件驱动电路，现象为 LED 数码管每秒的显示值加 1，从 0~F 循环显示。

而 SDK1013/1014 无硬件的 LCD/LED 显示驱动电路，可以软件方式取得数码管。

```
void DDICInit(void)
{
     //占空比1/4,P14~P17,P22~P25,P34~P37为DDIC输出脚
    DDIC_Init(DDIC_DUTYCYCLE_D4, 0X00, 0XF0, 0X3C, 0XF0);
    DDIC_LEDConfig();           //DDIC配置为LED模式
    DDIC_OutputPinOfDutycycleD4(SEG0_27COM4_7);//占空比1/4时SEG为S0~S27,COM为C4~C7
    DDIC_Cmd(ENABLE);           //使能
}

void main(void)
{
    uint8_t LedDisplayNum = 0;
    Timer0Init();
    DDICInit();
    enableInterrupts();

    while(1)
    {
        if(T0Flag10ms)
        {
            T0Flag10ms = 0;
            Led_Display();
        }
        if(T0Flag1s)
        {
            T0Flag1s = 0;
            LedDisplayNum++;    //显示数据加一
            if(LedDisplayNum==16)
            {
                LedDisplayNum = 0;
            }
            LedDataTab[0] = LedDisplayNum;     //显示数据装载到数组
            LedDataTab[1] = LedDisplayNum;
            LedDataTab[2] = LedDisplayNum;
            LedDataTab[3] = LedDisplayNum;
        }
    }
}
```

在主函数中，配置 T0 为定时器模式，计数值为 1600，定时时间为 100 微秒，置起中断标志 LedDisplayFlag，累加中断标志中断标志 Timer0Flag，Timer0Flag 累加 10000 次，即 1 秒使显示值加 1。开启 LED 硬件扫描，在主循环中， 每隔 10 毫秒更新 LED 数码管显示数据。SDK1010 主板 LED 与 LCD 共用 SEG 口，使用 LED 时，LCD 会显示乱码。

4.8 LCD 显示驱动示例

SC92F8547/7547、SC92F8446B/7446B 内部集成了硬件的 LCD/LED 显示驱动电路。

LCD 显示驱动功能如下：

1. 4 种显示驱动模式可选：8 X 24、6 X 26、5 X 27、或 4X 28 段；
2. 2 种偏置方式可选：1/4 Bias 和 1/3 Bias；
3. com 口驱动能力 4 级可选；
4. 显示驱动电路可选择内建 128K LRC 或外部 32K 振荡器作为时钟源，帧频约为 64Hz。

示例 code 启动 LCD 硬件驱动电路，现象为 LCD 屏每秒的显示值加 1，从 0~F 循环显示。

```
void DDICInit(void)  //DDIC配置为LCD，显示函数和配置需根据LCD屏进行修改
{
    DDIC_Init(DDIC_DUTYCYCLE_D4, 0X00, 0XF0, 0X3C, 0X0F);
    //LCD电压为(17+4)*VDD/32=3.3V,内部分压电阻100K,偏置电压3.3V
    DDIC_LCDConfig(4, DDIC_ResSel_100K, DDIC_BIAS_D3);
    DDIC_OutputPinOfDutycycleD4(SEG4_27COM0_3);
    DDIC_Cmd(ENABLE);
}
```

```
void main(void)
{
    uint8_t LcdDisplayNum = 0;
    Timer0Init();
    DDICInit();
    enableInterrupts();

    while(1)
    {
        if(T0Flag10ms)
        {
            T0Flag10ms = 0;
            Lcd_Display();
        }
        if(T0Flag1s)
        {
            T0Flag1s = 0;
            LcdDisplayNum++;    //显示数据加一
            if(LcdDisplayNum==16)
            {
                LcdDisplayNum = 0;
            }
            LcdDataTab[0] = LcdDisplayNum;     //显示数据装载到数组
            LcdDataTab[1] = LcdDisplayNum;
            LcdDataTab[2] = LcdDisplayNum;
            LcdDataTab[3] = LcdDisplayNum;
        }
    }
}
```

在主函数中，配置 T0 为定时器模式，计数值为 1600，定时时间为 100 微秒，置起中断标志 LcdDisplayFlag，累加中断标志中断标志 Timer0Flag，Timer0Flag 累加 10000 次，即 1 秒使显示值加 1。开启 LCD 硬件扫描，在主循环中， 每隔 10 毫秒更新 LCD 屏显示数据。

4.9 模拟比较器示例

4.9.1 SDK1011 /1012 模拟比较器示例

SC92F8547/7547、SC92F8446B/7446B 内建一个模拟比较器 ，此比较器具有四个模拟信号正输入端：CMP0~3 ，可通过 CMPIS [1:0]切换选择。负输入端电压可通过 CMPRF[3:0]切换为 CMPR 脚上的外部电压或内部的 16 档比较电压中的一种。

通过 CMPIM[1:0]可以方便的设定比较器的中断模式，当 CMPIM[1:0]所设定的中断条件发生时比较器中断标志 CMPIF 会被置 1，该中断标志需要软件清除。

示例 code 启动模拟比较器功能，通过调节 RP1 电阻可控制比较器正端输入电压（P43 口电压），调节 RP2 电阻可控制比较器负端比较电压（P44 口电压），现象为当正端电压大于负端电压时，LED1 灯亮，反之 LED2 灯亮。

```
void main(void)
{
    GPIO_Init(GPIO0, GPIO_PIN_6, GPIO_MODE_OUT_PP); //LED1灯口设置为强推挽
    GPIO_Init(GPIO5, GPIO_PIN_2, GPIO_MODE_OUT_PP); //LED2灯口设置为强推挽
    //模拟比较器正向输入端为通道3，负端比较电压为外部电压
    ACMP_Init(ACMP_VREF_EXTERNAL, ACMP_CHANNEL_3);
    ACMP_SetTriggerMode(ACMP_TRIGGER_RISE_FALL);
    ACMP_ITConfig(ENABLE, LOW);     //开比较器中断
    enableInterrupts();             //开总中断
    ACMP_Cmd(ENABLE);

    while(1)
    {
        if(ACMP_GetFlagStatus(ACMP_FLAG_CMPSTA))//读比较器输出状态位,正端电压大于负端电压时置一
        {
            GPIO_WriteHigh(GPIO0, GPIO_PIN_6);
            GPIO_WriteLow(GPIO5, GPIO_PIN_2);
        }
        else
        {
            GPIO_WriteHigh(GPIO5, GPIO_PIN_2);
            GPIO_WriteLow(GPIO0, GPIO_PIN_6);
        }
    }
}
```

示例将 LED1 灯与 LED2 灯配置为强推挽模式，开启模拟比较器功能，正端输入电压端口选择 CMP3，负端比较电压选择外部 CMPR 电压，模拟比较器中断触发条件为双沿中断。通过读取输出状态位的状态，控制 LED1 灯、LED2 灯的亮灭。

4.10 低频时钟定时器 BTM 示例

4.10.1 SDK1011/1012 BTM 示例

SC92F8547/7547、SC92F8446B/7446B 内建一个频率为 128kHz 的 RC 及 32.768kHz 晶体振荡电路，都可作为低频时钟定时器 Base Timer 的时钟源。该振荡器直接连接一个 Base Timer，可以把 CPU 从 STOP mode 唤醒，并且产生中断。

示例 code 开启 BTM，现象为 LED1 灯每 0.5 秒亮灭状态改变一次。

```
void main(void)
{
    GPIO_Init(GPIO0,GPIO_PIN_6,GPIO_MODE_OUT_PP);    //将LED1设置为强推挽输出
    GPIO_Init(GPIO5,GPIO_PIN_2,GPIO_MODE_OUT_PP);    //将LED2设置为强推挽输出
    GPIO_Init(GPIO5,GPIO_PIN_0,GPIO_MODE_IN_PU);     //将P50设置为带上拉输入

    BTM_Init(BTM_TIMEBASE_250MS);                    //BTM溢出时间250ms
    BTM_ITConfig(ENABLE, LOW);                       //使能BTM中断
    BTM_Cmd(ENABLE);                                 //使能BTM计数
    enableInterrupts();                              //打开总中断

    while(1)
    {
        if(GPIO_ReadPin(GPIO5,GPIO_PIN_0))           //P50为外部晶振输入脚，外部晶振使能时为低
        {
            GPIO_WriteHigh(GPIO5,GPIO_PIN_2);
        }
        else
        {
            GPIO_WriteLow(GPIO5,GPIO_PIN_2);
        }
    }
}
```

```
void BTMInterrupt()          interrupt 9
{
    if(GPIO_ReadPin(GPIO0,GPIO_PIN_6))          //翻转IO
    {
        GPIO_WriteLow(GPIO0,GPIO_PIN_6);
    }
    else
    {
        GPIO_WriteHigh(GPIO0,GPIO_PIN_6);
    }
}
```

示例 code 在 option 项中勾选使用外部 32.768 晶体振荡电路，此时内建 128k Hz RC 停止工作。BTM 设置为每 0.5 秒产生一个中断，中断标志置起后翻转 LED1 灯状态。若 option 项不勾选外部晶振，此时内部低频 RC 工作，LED1 灯依旧 0.5 秒改变一次状态，由于外部低频晶振不工作，OSCI（P50）为 IO 口，且为上拉状态，故 LED2 灯将常亮。

4.10.2 SDK1013/1014 BTM 示例

SC92F8463B/7463B 内建一个频率为 128kHz 的 RC，可作为低频时钟定时器 Base Timer 的时钟源。

SC92F7423 内建一个频率为 128kHz 的 RC 及 32.768kHz 晶体振荡电路，均可作为低频时钟定时器 Base Timer 的时钟源。

该振荡器直接连接一个 Base Timer，可以把 CPU 从 STOP mode 唤醒，并且产生中断。

示例 code 开启 BTM，现象为 LED1 灯每 0.25 秒亮灭状态改变一次。

```
void main(void)
{
    //将LED1设置为强推挽输出
    GPIO_Init(GPIO2,GPIO_PIN_5,GPIO_MODE_OUT_PP);

    BTM_Init(BTM_TIMEBASE_250MS);    //BTM溢出时间250ms
    BTM_ITConfig(ENABLE, LOW);       //使能BTM中断
    BTM_Cmd(ENABLE);                 //使能BTM计数
    enableInterrupts();              //打开总中断

    while(1);
}
void BTMInterrupt()          interrupt 9
{
    if(GPIO_ReadPin(GPIO2,GPIO_PIN_5))          //翻转IO
    {
        GPIO_WriteLow(GPIO2,GPIO_PIN_5);
    }
    else
    {
        GPIO_WriteHigh(GPIO2,GPIO_PIN_5);
    }
}
```

SDK1013 示例 code 在 option 项中勾选不使能外部晶体振荡电路，使用内部 128kHz RC 振荡器工作。BTM 设置为每 0.25 秒产生一个中断，中断标志置起后翻转 LED1 灯状态。

SDK1014 示例 code 在 option 项中勾选使用外部 32.768 晶体振荡电路，此时内建 128k Hz RC 停止工作。BTM 设置为每 0.25 秒产生一个中断，中断标志置起后翻转 LED1 灯状态。也可在 option 项不勾选外部晶振，此时内部低频 RC 工作，LED1 灯依旧 0.25 秒改变一次状态。

4.11 ADC 示例

SC92F8547/7547、SC92F8446B/7446B 内建一个 12-bit 17 通道的高精度逐次逼近型 ADC，SC92F846XB/746XB 内建一个 12-bit 11 通道的高精度逐次逼近型 ADC，SC92F7423 内建一个 12-bit 11 通道的高精度逐次逼近型 ADC。外部的 ADC 通道和 IO 口的其它功能复用。内部的一路可接至 1/4 V_{DD}，配合内部 2.4V 参考电压用于测量 V_{DD} 电压。

ADC 的参考电压可以有 2 种选择：

1. VDD 管脚(即直接是内部的 V_{DD})；
2. 内部 Regulator 输出的参考电压精准的 2.4V（此时 MCU 供电电压 V_{DD} 不可低于 2.9V）。

注意：f_{ADC} 直接由内部 f_{HRC} 分频所得，用户在配置时要注意 ADC 的时钟频率 f_{ADC} 不可大于系统时钟的频率 f_{SYS}，否则会引起 ADC 转换结果异常！

以 SC92F8547 为例，利用 ADC 采集电压，当热敏电阻 NTC 的阻值随温度改变时，ADC 所采集的电压亦随之改变，由热敏电阻的温度特性可通过当前的电压值求得当前的温度值。现象为 LED 数码管显示当前室温，用手触摸 NTC，数码管显示温度亦随之变化。

```
void ADCInit(void)
{
    ADC_Init(ADC_PRESSEL_FOSC_D6, ADC_Cycle_6Cycle);
    ADC_ChannelConfig(ADC_CHANNEL_14, ENABLE);
    ADC_Cmd(ENABLE);
    ADC_VrefConfig(ADC_VREF_VDD);
}

void main(void)
{
    DDICInit();      //LED驱动电路初始化，四分之一占空比
    ADCInit();       //ADC初始化
    Timer0Init();    //TIMER0初始化
    enableInterrupts();

    while(1)
    {
        if(T0Flag500ms)
        {
            T0Flag500ms = 0;
            GetADCValue();        //得到温度值的ADC转换值
            GetTemperature();     //通过数组ADCValueToTemp[]求得温度值
            Led_Display();
        }
    }
}
```

示例利用 T0 的定时功能，定时更新 LED 数码管的数据显示，开启 ADC 功能，选择通道 14 为采样通道，设置采样时间为 36 个采样时钟周期，ADC 的时钟频率 f_{ADC} 为 $f_{HRC}/6$。

4.12 串口 UART0 示例

4.12.1 SDK1011/1012/1013 UART0 示例

MCU支持一个全双工的串行口UART0，UART0的功能及特性如下：

1. 三种通讯模式可选：模式 0、模式 1 和模式 3；
2. 可选择定时器 1 或定时器 2 作为波特率发生器；
3. 发送和接收完成可产生中断 RI/TI，该中断标志需要软件清除。

以 SDK1011 示例 code 为例，使用 UART0 与串口上位机通信，波特率为 9600，现象为上位机显示“UART0 is OK！”。

```c
#include "stdio.h"

void Uart0Init(void)       //波特率9600，TIMER1做时钟源，允许接收
{
    GPIO_Init(GPIO2, GPIO_PIN_1, GPIO_MODE_IN_PU);
    UART0_Init(16000000, 9600, UART0_Mode_10B, UART0_CLOCK_TIMER1, UART0_RX_ENABLE);
    UART0_ITConfig(ENABLE, LOW);     //使能uart0中断
}

char putchar(char c)
{
    UART0_SendData8(c);
    while(!Uart0SendFlag);
    Uart0SendFlag = 0;
    return c;
}

void main(void)
{
    Uart0Init();
    enableInterrupts();

    while(1)
    {
        printf("UART0 is OK!\n");
    }
}
```

示例配置 UART0 为模式 1，使能 UART0 接收，使用 T1 作为波特率发生器，波特率为 9600。在主循环中不断往上位机发送字符串“UART0 is OK !”。

4.12.2 SDK1014 UART0 示例

SDK1014 核心板 MCU 为 SC92F7423，此 MCU 支持两路三合一串口 SSI，这里选用 SSI0。SSI_UART0 的功能及特性如下：

1. 两种通讯模式可选：模式 1 和模式 3；
2. 发送和接收完成可产生中断 RI/TI，该中断标志需要软件清除。

示例 code 将 SSI0 设置为 UART 模式，使用 UART0 与串口上位机通信，波特率为 9600，现象为上位机显示“UART0 is OK !”。

```c
void Uart0Init(void)
{
    GPIO_Init(GPIO1, GPIO_PIN_3, GPIO_MODE_IN_PU);
    //波特率9600，允许接收
    SSI0_UART_Init(16000000, 9600, UART_Mode_10B, UART_RX_ENABLE);
    SSI0_ITConfig(ENABLE, LOW); //使能uart0中断
}
char putchar(char c)
{
    SSI0_UART_SendData8(c);
    while(!Uart0SendFlag);
    Uart0SendFlag = 0;
    return c;
}
void main(void)
{
    Uart0Init();
    enableInterrupts();

    while(1)
    {
        printf("UART0 is OK!\n");
    }
}
```

示例配置 SSI0 为 UART，UART0 工作在模式 1，使能 UART0 接收，波特率为 9600。在主循环中不断往上位机发送字符串“UART0 is OK !”。

4.13 IAP 示例

IAP 操作空间范围有两种模式可选：

1. 最高位地址的 128 bytes EEPROM 可以作为数据存储使用；
2. IC 整个 ROM 空间范围及 128 bytes EEPROM 内都可进行 IAP 操作，主要用作远程程序更新使用。

IAP 操作空间选择作为 Code Option 在编程器写入 IC 时选择。

示例 code 通过 IAP 将数组的值写进 EEPROM，再读出 EEPROM 的值与数组的值比较，若相同，则 UART0 发送 EEPROM START 与 END 至上位机，若不相同，上位机将显示详细信息。

```
void main(void)
{
    uint16_t i;
    uint8_t IAP_Value = 0;

    GPIO_Init(GPIO1, GPIO_PIN_3, GPIO_MODE_IN_PU);      //TX设为输入带上拉
    //波特率9600
    UART0_Init(12000000, 9600, UART0_Mode_10B, UART0_CLOCK_TIMER1, UART0_RX_ENABLE);
    UART0_ITConfig(ENABLE, LOW);                        //使能uart0中断

    IAP_SetHoldTime(IAP_HOLDTIME_1MS);                  //HoldTime时间为1MS
    IAP_SetOperateRange(IAP_OPERATERANGE_ONLY_EEPROM);  //只允许EEPROM进行IAP操作

    enableInterrupts();                                 //开总中断

    printf("IAP EEPROM START\n");
    for(i=0;i<128;i++)
    {
        IAP_ProgramByte(i, i, IAP_MEMTYPE_EEPROM, 0xf0);
        IAP_Value = IAP_ReadByte(i,IAP_MEMTYPE_EEPROM);
        if(i!=IAP_Value)
        {
            printf("IAP Fail! Error message is\n");
            printf("Adress:%u\n",i);
            printf("Data:0X%BX\n",IAP_Value);
        }
    }
    printf("IAP EEPROM END\n");

    while(1);
}
```

示例使用 IAP 功能在 128Bytes 的 EEPROM 地址上，进行读写操作。读出的值若与写的值不同，上位机将显示不同处的地址，以及对应地址内的数据。

4.14 乘除法器示例

4.14.1 SDK1011/1012 MDU 示例

SC92F8547 提供了 1 个 16 位的乘除法器，由扩展累加器 EXA0~EXA3、扩展 B 寄存器 EXB 和运算控制寄存器 OPERCON 组成。可取代软件进行 16 位×16 位乘法运算和 32 位/16 位除法运算。

示例 code 开启乘除法器功能，现象为若乘法结果正确，则 LED1 灯亮，若除法结果正确，则 LED2 灯亮。

```
void main(void)
{
    GPIO_Init(GPIO0, GPIO_PIN_6, GPIO_MODE_OUT_PP);
    GPIO_Init(GPIO5, GPIO_PIN_2, GPIO_MODE_OUT_PP);

    MDU_MultiplicationConfig(10000, 50);     //设置乘数与被乘数11000*50
    MDU_StartOperation();                    //启动运算
    if(MDU_GetProduct() == 500000)
    {
        GPIO_WriteHigh(GPIO0, GPIO_PIN_6);  //结果正确LED1灯亮
    }

    MDU_DivisionConfig(10000, 50);           //设置被除数与除数10000/50
    MDU_StartOperation();                    //启动运算
    if((MDU_GetQuotient()==200) && (MDU_GetRemainder()==0))
    {
        GPIO_WriteHigh(GPIO5, GPIO_PIN_2);  //结果正确LED2灯亮
    }

    while(1);

}
```

示例将 LED1 与 LED2 配置为强推挽输出模式。乘法功能中入参为乘数与被乘数，除法功能中入参为被除数与除数。由函数返回值可得乘除法结果。

4.14.2 SDK1013 MDU 示例

SDK1013 核心板 MCU 提供了 1 个 16 位的乘除法器，由扩展累加器 EXA0~EXA3、扩展 B 寄存器 EXB 和运算控制寄存器 OPERCON 组成。可取代软件进行 16 位×16 位乘法运算和 32 位/16 位除法运算。

示例 code 开启乘除法器功能，现象为若乘除法结果都正确，则 LED1 灯常亮，否则 LED1 灯闪烁。

```
void main(void)
{
    uint32_t Product;
    uint16_t i;
    GPIO_Init(GPIO2, GPIO_PIN_5, GPIO_MODE_OUT_PP);

    MDU_MultiplicationConfig(10000, 50);     //设置乘数与被乘数10000*50
    MDU_StartOperation();                    //启动运算
    Product = MDU_GetProduct();

    MDU_DivisionConfig(10000, 50);           //设置被除数与除数10000/50
    MDU_StartOperation();                    //启动运算

    while(1)
    {
        if((Product==500000) && (MDU_GetQuotient()==200) && (MDU_GetRemainder()==0))
        {
            GPIO_WriteHigh(GPIO2, GPIO_PIN_5);  //结果正确LED1灯亮
        }
        else
        {
            GPIO_WriteHigh(GPIO2, GPIO_PIN_5);  //否则LED1闪烁
            for(i=0;i<50000;i++);
            GPIO_WriteLow(GPIO2, GPIO_PIN_5);
            for(i=0;i<50000;i++);
        }
    }
}
```

示例将 LED1 配置为强推挽输出模式。乘法功能中入参为乘数与被乘数，除法功能中入参为被除数与除数。

4.15 UART1 示例

SDK1011/1012/1013/1014 核心板 MCU 内部集成了三选一串行接口电路（简称 SSI），可配置为串口模式 UART1。UART 模式可工作在模式 1（10 位全双工异步通信）和模式 3（11 位全双工异步通信）。

以 SDK1014 为例，示例 code 使用 UART1 与串口上位机通信，使能接收功能，波特率为 9600，当上位机发生数据到 MCU，MCU 将同样的数据发送到上位机。

```
void main(void)
{
    GPIO_Init(GPIO2, GPIO_PIN_0, GPIO_MODE_IN_PU);  //TX1脚设置为上拉输入模式

    //波特率9600，使能接收
    SSI1_UART_Init(16000000, 9600, UART_Mode_10B, UART_RX_ENABLE);
    SSI1_ITConfig(ENABLE, LOW);
    enableInterrupts();

    while(1);
}
void SSI1Interrupt()        interrupt 7
{
    if(SSI1_GetFlagStatus(UART_FLAG_TI))
    {
        SSI1_ClearFlag(UART_FLAG_TI);
    }
    if(SSI1_GetFlagStatus(UART_FLAG_RI))
    {
        SSI1_ClearFlag(UART_FLAG_RI);
        SSI1_UART_SendData8(SSI1_UART_ReceiveData8());
    }
}
```

4.16 SPI 示例

SDK1011/1012/1013/1014 核心板 MCU 内部集成了三选一串行接口电路（简称 SSI），可配置为高速串行通信接口 SPI，允许 MCU 与外围设备(包括其它 MCU)进行全双工，同步串行通信。

SPI 可配置为主模式或从属模式中的一种。以 SDK1011 为例，SPI 模块的配置和初始化通过设置 SSCON0 寄存器(SPI 控制寄存器)和 SSCON1(SPI 状态寄存器)来完成。配置完成后，通过设置 SSCON0，SSCON1，SSDAT(SPI 数据寄存器)来完成数据传送。通过软件设置 SSCON0 寄存器的 CPOL 位和 CPHA 位，用户可以选择 SPI 时钟极性和相位的四种组合方式。

示例 code 通过 SPI 操作 W25Q16，将数组数据 0~9 写入 W25Q16，再将 W25Q16 中的值读出来并通过 UART0 发送给上位机，现象为上位机显示 0~9。

```
void SPIInit(void)
{
    //SPI主模式，空闲状态SCK低电平，第一沿采集中断，时钟速率为Fsys/256，MSB位优先发送
    SSI_SPI_Init(SPI_FIRSTBIT_MSB, SPI_BAUDRATEPRESCALER_256, SPI_MODE_MASTER ,
                 SPI_CLOCKPOLARITY_LOW, SPI_CLOCKPHASE_1EDGE, SPI_TXE_DISINT);
    SSI_SPI_Cmd(ENABLE);
    SSI_ITConfig(ENABLE, LOW);
}
```

```
void main(void)
{
    uint16_t i;
    GPIO_Init(GPIO1, GPIO_PIN_0, GPIO_MODE_OUT_PP);
    for(i=0;i<10;i++)
    {
        Send_DATA[i] = i;  //发送数组赋值
        Rec_DATA[i] = 2;   //接收数组赋值
    }

    Uart0Init();
    SPIInit();
    enableInterrupts();

    W25_SectorErase(0x000000);          //W25Q16擦除0x0000起始的4k空间
    SPI_Flash_Write_NoCheck(Send_DATA,0x000000,10);//往0X0000地址写10个数，数值为Send_DATA[i]值
    SPI_Flash_Read(Rec_DATA,0x000000,10);   //从0X0000读10个数,存放到Rec_DATA[10]

    while(1)
    {
        printf("\n Rec_DATA[10] value = \n{");       //串口打印Rec_DATA数组值，观察是否为写入值

        for(i=0;i<10;i++)
        {
            printf("%c",Rec_DATA[i]+0x30);
        }
        printf("}");
    }
}
```

示例使用 UART0 与串口上位机通信，SPI 配置 SCK 在空闲状态下为低电平，第一沿采集数据，SPI 时钟速率为 f_{SYS} /256，MSB 优先发送。将数组 Send_DATA[10]的值写进 W25Q16，再从 W25Q16 将这些值读出来放在数组 Rec_DATA[10]中，通过 UART0 将 Rec_DATA[10]发出。

4.17 触控按键电路 TOUCH KEY 示例

SC92F8547/SC92F8446B 内建一个 31 通道的高灵敏度电容触控按键 Touch Key Sensor 电路，可实现隔空按键触控、接近感应等操作。SC92F8463B 内建一个 23 通道的双模电容触控按键 Touch Key Sensor 电路，可配置为高灵敏度模式或高可靠模式，高灵敏度模式可实现隔空按键触控、接近感应等灵敏度要求较高的触控应用。高可靠模式具有很强的抗干扰能力，可通过 10V 动态 CS 测试。

用户通过使用赛元提供的高灵敏度触控按键库文件，可快速简单实现隔空按键、接近感应等功能。

示例 code 调用赛元高灵敏度触控库，配置三个 TouchKey 按键，按下触控按键，按键上方对应的 LED 灯亮，同时蜂鸣器鸣响。

```
void main(void)
{
    GPIO_Init(GPIO0, GPIO_PIN_6, GPIO_MODE_OUT_PP);     //LED1灯强推挽模式
    GPIO_Init(GPIO5, GPIO_PIN_2, GPIO_MODE_OUT_PP);     //LED2灯强推挽模式
    GPIO_Init(GPIO5, GPIO_PIN_4, GPIO_MODE_OUT_PP);     //LED3灯强推挽模式
    GPIO_Init(GPIO4, GPIO_PIN_7, GPIO_MODE_OUT_PP);     //蜂鸣器强推挽模式
    GPIO_Init(GPIO0, GPIO_PIN_3, GPIO_MODE_OUT_PP);

    TIMER0_Init();
    TouchKeyInit();           //调用库函数，初始化TouchKey

    while(1)
    {
        Sys_Scan();      //TK按键扫描
    }
}
```

```
void Sys_Scan(void)
{
    if(TKScanflag == 1)
    {
        //重要步骤2：  触摸键扫描一轮标志，是否调用TouchKeyScan()一定要根据此标志位置起后
        if(SOCAPI_TouchKeyStatus&0x80)
         {
            SOCAPI_TouchKeyStatus &=0x7f;    //重要步骤3：清除标志位，  需要外部清除。
            exKeyValueFlag = TouchKeyScan();
            ChangeTouchKeyvalue();           //键值转换
            TKScanflag=0;
            return ;
        }
    }
    if(TKScanflag==0)
    {                                        //LED灯亮灭随TK按键改变
        LED();
        SOCAPI_TouchKeyStatus &=0x7f;    //重要步骤3：清除标志位，  需要外部清除。
        TouchKeyRestart();               //启动下一轮转换
        TKScanflag=1;
    }
}
```

4.18 功能复用示例

4.18.1 SDK1011/1012 功能复用示例

功能复用示例是对开发板各种资源整体上的一个运用。示例通过 TK 按键与开关按键选择演示开发板不同的功能模块。TK 按键可选择进入不同的模式，开关按键为确认键与退出键。

1. SC92F8547/SC92F8446B 按键说明：

开关按键 S1：演示功能确认键

开关按键 S2：退出当前演示的功能

TK24：代表二进制 0

TK25：代表二进制 1

TK26：重新输入

若选择模式 3，则需输入二进制 0011，即按两下 TK24，两下 TK25（此过程中按下 TK26 可重新选择模式）。再按确认键即可看到模式 3 所演示功能，按退出键则返回模式选择界面。

2. SC92F7547/SC92F7446B 按键说明：

开关按键 S1：模式选择，每按一次模式加一

开关按键 S2：演示功能确认键或退出当前演示功能

若选择模式 3，则需通过 S1 按键选至模式 3。再按 S2 按键即可看到模式 3 所演示功能，再按 S2 则退出，返回模式选择界面。

3. 模式说明：

模式 1：PWM 控制三色灯，现象为三色灯循环变色。

模式 2：Timer1 计数，若按下 S1 键两次则 LED1 亮灭状态改变。

模式 3：UART1 通过 TXD1 口往串口助手上位机发送数据，上位机显示字符串“TEST”，波特率为 9600

模式 4：ADC 测温，显示当前温度。

模式 5：SPI 操作 W25Q16，将 10 个数写进 W25Q16，通过 TXD0 口发送数据到串口，观察所读的值是否为所写的值。需将 J1 与 J2 用跳线帽短接。

模式 6：T2 捕获模式，将 SCK 脚与 T2EX 脚用杜邦线短接，LED 上显示 SCK 脚方波的周期。

模式 7：IAP 操作 EEPROM，将 10 个数写进 EEPROM，通过 TXD1 口发送数据到串口，观察所读的值是否为所写的值。

```
void main(void)
{
    GPIOInit();          //IO初始化
    Timer0Init();        //Timer0初始化
    DDIC_LCDInit();      //LCD初始化
    TouchKeyInit();      //TK初始化
    enableInterrupts(); //开总中断
    GPIO_Init(GPIO4, GPIO_PIN_4, GPIO_MODE_OUT_PP);
    while(1)
    {
        Sys_Scan();      //TK按键扫描
        if(T0Flag10ms)
        {
            T0Flag10ms = 0;
            Lcd_Display();

        }
        if(GetS1State() != RESET)
        {
            Mode = LcdDataTab[3] + LcdDataTab[2]*2 + LcdDataTab[1]*4 + LcdDataTab[0]*8;  //模式
```

4.18.2 SDK1013 功能复用示例

功能复用示例是对开发板各种资源整体上的一个运用。示例通过 TK 按键与开关按键选择演示开发板不同的功能模块。TK 按键可选择进入不同的模式，开关按键为确认键与退出键。

1. SC92F8463B 按键说明：

开关按键 S2：演示功能确认键或退出当前演示功能

TK24：模式选择，每按一次模式加一

通过 TK24 进行模式选择，再按 S2 进入模式即可看到该模式演示的功能，再按 S2 则返回模式选择。

2. SC92F7463B 按键说明：

开关按键 S1：模式选择，每按一次模式加一

开关按键 S2：演示功能确认键或退出当前演示功能

通过按键 S1 进行模式选择，再按 S2 进入模式即可看到该模式演示的功能，再按 S2 则返回模式选择。

3. 模式说明：

模式 1：PWM 控制 LED1，现象为红色呼吸灯。

模式 2：Timer1 计数，若按下 S1 键两次则 LED1 亮灭状态改变。

模式 3：UART1 通过 TXD1 口往串口助手上位机发送数据，上位机显示字符串“TEST”，波特率为 9600

模式 4：ADC 测温，显示当前温度。

模式 5：SPI 操作 W25Q16，将 10 个数写进 W25Q16，通过 TXD0 口发送数据到串口，观察所读的值是否为所写的值。需将 J1 与 J2 用跳线帽短接。

模式 6：T2 捕获模式，将 SCK 脚与 T2EX 脚用杜邦线短接，LED 上显示 SCK 脚方波的周期。

模式 7：IAP 操作 EEPROM，将 10 个数写进 EEPROM，通过 TXD1 口发送数据到串口，观察所读的值是否为所写的值。

```
void main(void)
{
    GPIOInit();          //IO初始化
    Timer0Init();        //Timer0初始化
    DDICInit();
    TouchKeyInit();      //TK初始化
    enableInterrupts(); //开总中断
    while(1)
    {
        if(T0Flag1ms)
        {
            T0Flag1ms = 0;
            Sys_Scan();      //TK按键扫描
            KeyScan();

            if(Mode)
            {
                LedDataTab[0] = 17;
                LedDataTab[1] = 17;
                LedDataTab[2] = 17;
                LedDataTab[3] = Mode;
            }
            else
            LedDataTab[0] = LedDataTab[1] = LedDataTab[2] = LedDataTab[3] = 16;
        }
        if(S2Flag)
        {
            S2Flag = 0;
            ComAllClose;
            SegAllClose;
            GPIO_WriteLow(GPIO2, GPIO_PIN_5);
            GPIO_Init(GPIO2, GPIO_PIN_6, GPIO_MODE_IN_PU);
            exKeyValue = 0;
            switch(Mode)
            {
                case 1: Mode1_PWM();              break;
                case 2: Mode2_Timer1Counter();    break;
                case 3: Mode3_UART1();            break;
                case 4: Mode4_ADC();              break;
                case 5: Mode5_SPI();              break;
                case 6: Mode6_Timer2Capture();    break;
                case 7: Mode7_IAP();              break;
                default:                          break;
            }
            GPIO_Init(GPIO2, GPIO_PIN_6, GPIO_MODE_OUT_PP);
        }
    }
}
```

4.18.3 SDK1014 功能复用示例

功能复用示例是对开发板各种资源整体上的一个运用。示例通过开关按键选择演示开发板不同的功能模块。开关按键 S1 可选择进入不同的模式，开关按键 S2 为确认键与退出键。

1. 按键说明：

开关按键 S1：模式选择，每按一次模式加一

开关按键 S2：演示功能确认键或退出当前演示功能

2. 模式说明：

模式 1：PWM 控制 LED1，现象为红色呼吸灯。

模式 2：Timer1 计数，若按下 S1 键两次则 LED1 亮灭状态改变。

模式 3：UART1 通过 TXD1 口往串口助手上位机发送数据，上位机显示字符串“TEST”，波特率为 9600

模式 4：ADC 测温，显示当前温度。

模式 5：SPI 操作 W25Q16，将 10 个数写进 W25Q16，通过 TXD0 口发送数据到串口，观察所读的值是否

为所写的值。需将 J1 与 J2 用跳线帽短接。

模式 6：T2 捕获模式，将 SCK 脚与 T2EX 脚用杜邦线短接，LED 上显示 SCK 脚方波的周期。

模式 7：IAP 操作 EEPROM，将 10 个数写进 EEPROM，通过 TXD1 口发送数据到串口，观察所读的值是否为所写的值。

示例 code 有 7 种演示模式，通过 S1 进行模式选择，再按 S2 进入模式即可看到该模式演示的功能，再按 S2 则返回模式选择。

```
void main(void)
{
    GPIOInit();          //IO初始化
    Timer0Init();        //Timer0初始化
    DDICInit();          //LCD初始化
    LedDataTab[0] = LedDataTab[1] = LedDataTab[2] = LedDataTab[3] = 16;
    enableInterrupts(); //开总中断

    while(1)
    {
        if(T0Flag3ms)
        {
            T0Flag3ms = 0;
            KeyScan();
            Led_Display();

            if(Mode)
            {
                LedDataTab[0] = 17;
                LedDataTab[1] = 17;
                LedDataTab[2] = 17;
                LedDataTab[3] = Mode;
            }
        }
        if(S1Flag)
        {
            S1Flag = 0;
            Mode++;
            if(Mode > 7)
            {
                Mode = 1;
            }
        }

        if(S2Flag)
        {
            S2Flag = 0;
            ComAllClose;
            SegAllClose;
            switch (Mode)
            {
                case 1: Mode1_PWM();              break;
                case 2: Mode2_Timer1Counter();    break;
                case 3: Mode3_UART1();            break;
                case 4: Mode4_ADC();              break;
                case 5: Mode5_SPI();              break;
                case 6: Mode6_Timer2Capture();    break;
                case 7: Mode7_IAP();              break;
                default:                          break;
            }
        }
    }
}
```

规格更改记录

版本	记录	日期
V1.0	初版	2019 年 1 月