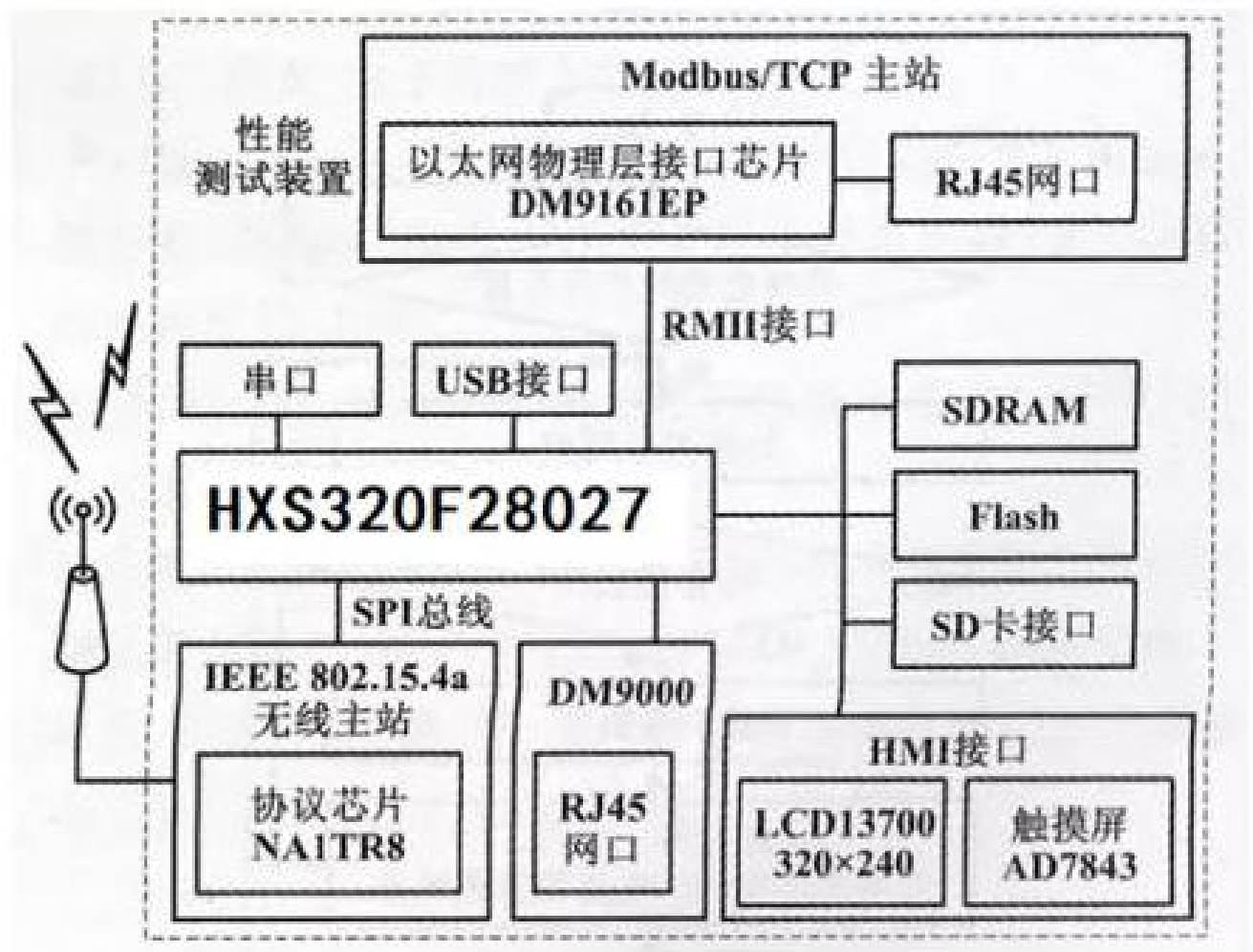
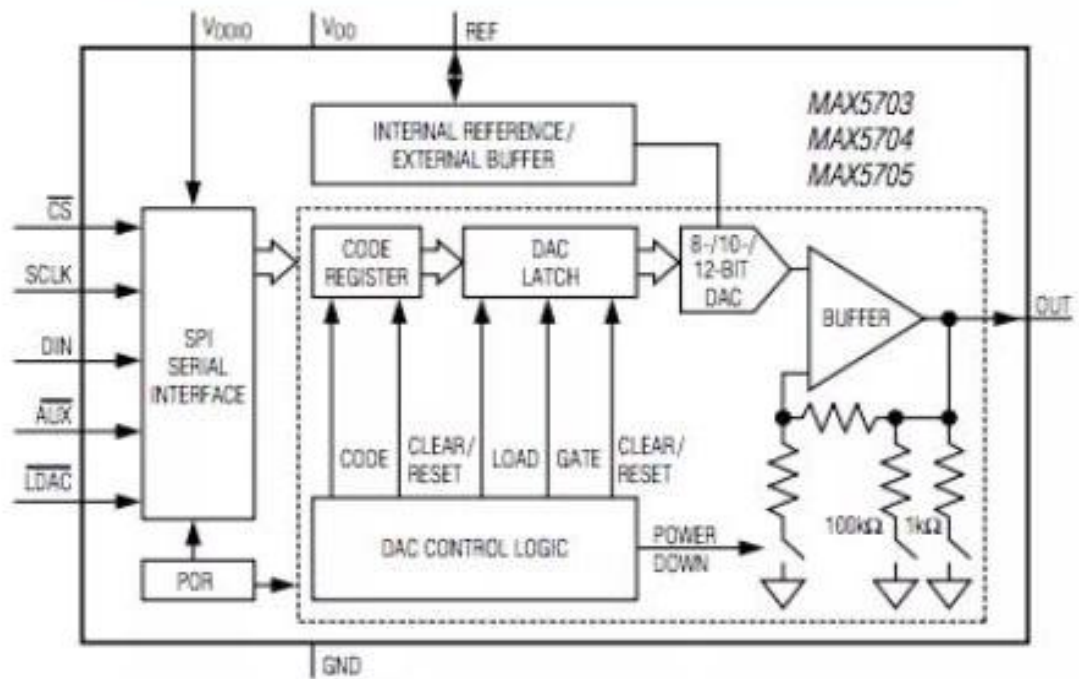
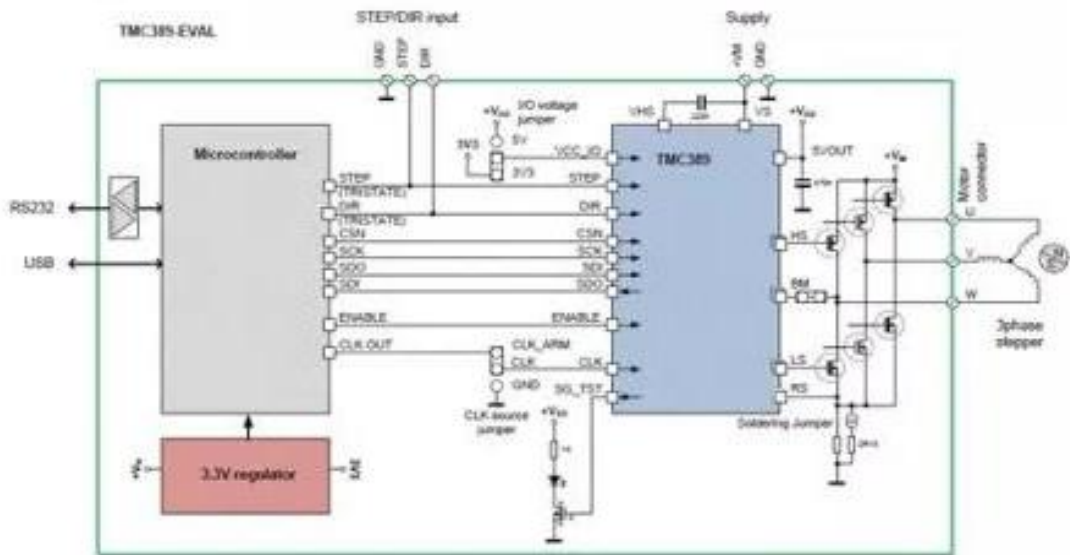


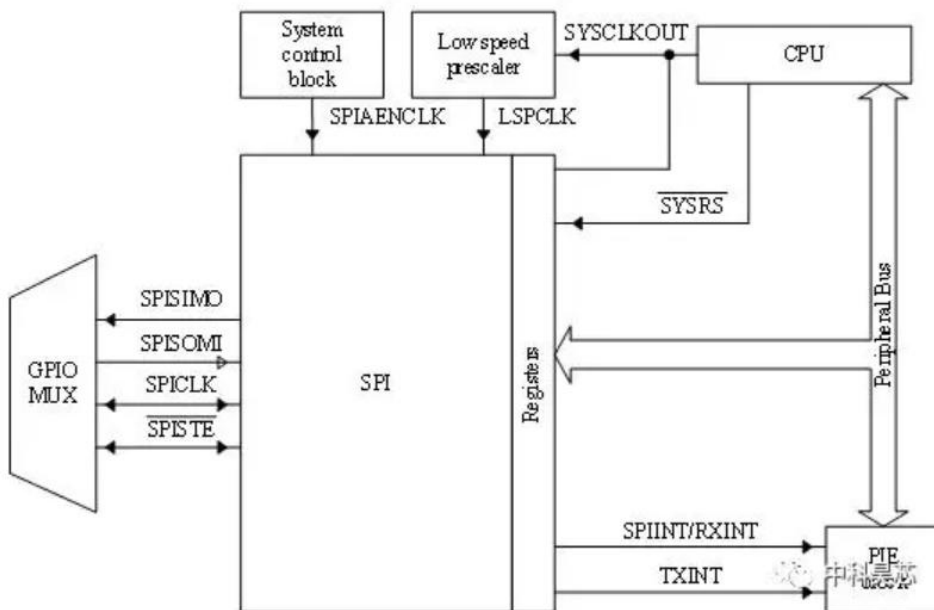
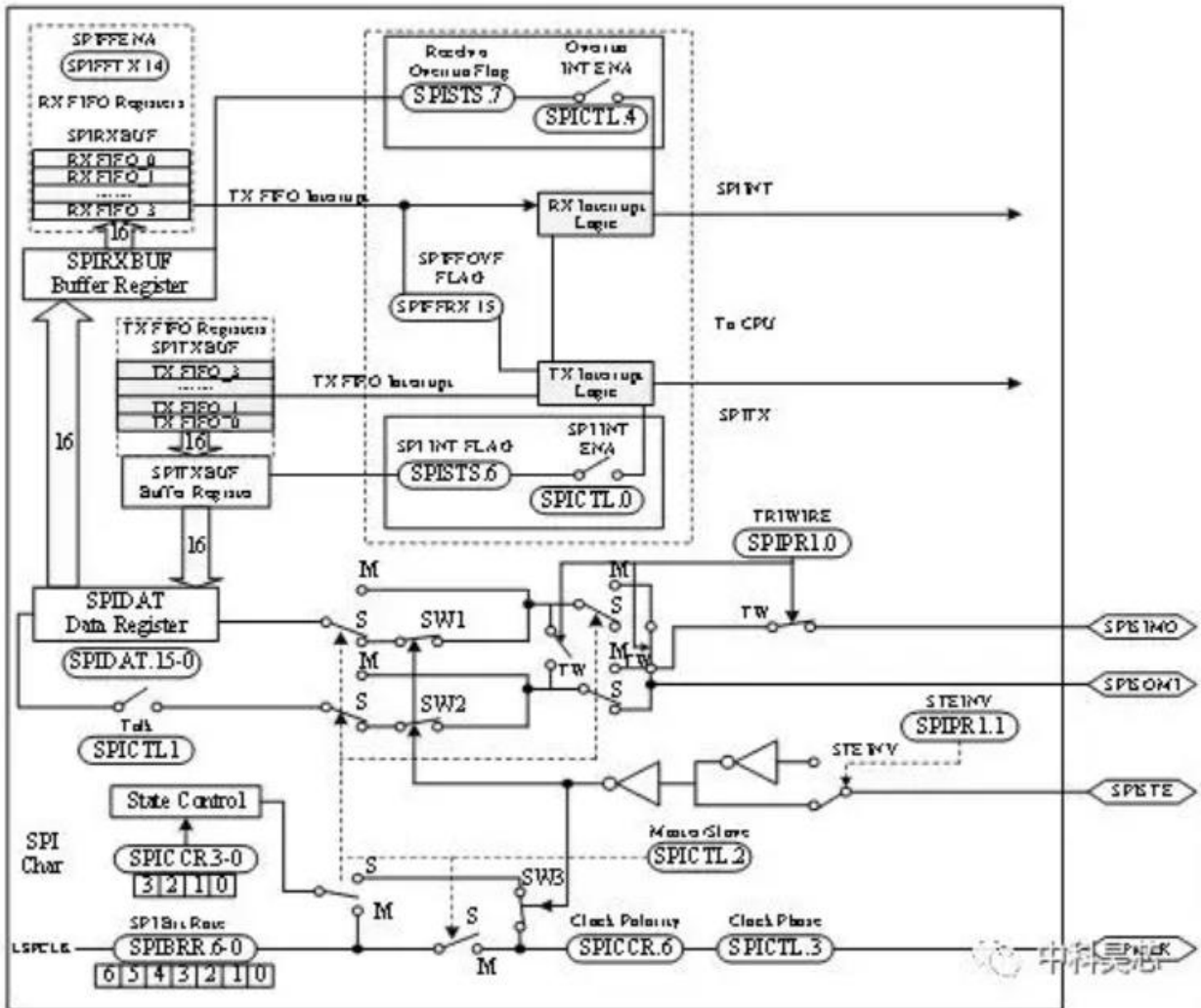
全球智能环保优先，高效便捷的数据处理已成为未来必然趋势。SPI 协议因其传输速率高，通讯简单等优势，在电机转向转速指令收发、射频通讯与 A/D、D/A 传感器 LCD/OLED 显示等方面得到快速应用。





在平头哥半导体有限公司的剑池集成开发环境（简称“CDK”）V2.10.1 版本开始支持中科昊芯 HX2000 系列芯片开发与调试后，本文以 HXS320F28027 的 PWM 输出实例对 PWM 电机调速原理及程序开发展开介绍。

HX2000 系列 SPI 通讯原理如下，通过 IO MUX 设置外设引脚功能，CPU 通过主控制器输出预分频与时钟使能，通过时钟引脚为通信网络提供时钟，通过 SPIBRR 寄存器配置波特率，数据写到 SPIDAT 或 SPITXBUF 时会启动发送最高有效位，之后数据移入 SPIDAT 最低有效位，以右对齐方式存储到 SPIRXBUF 中。



SPI 模块使用前，需先进行：

(1) 复位初始化操作：

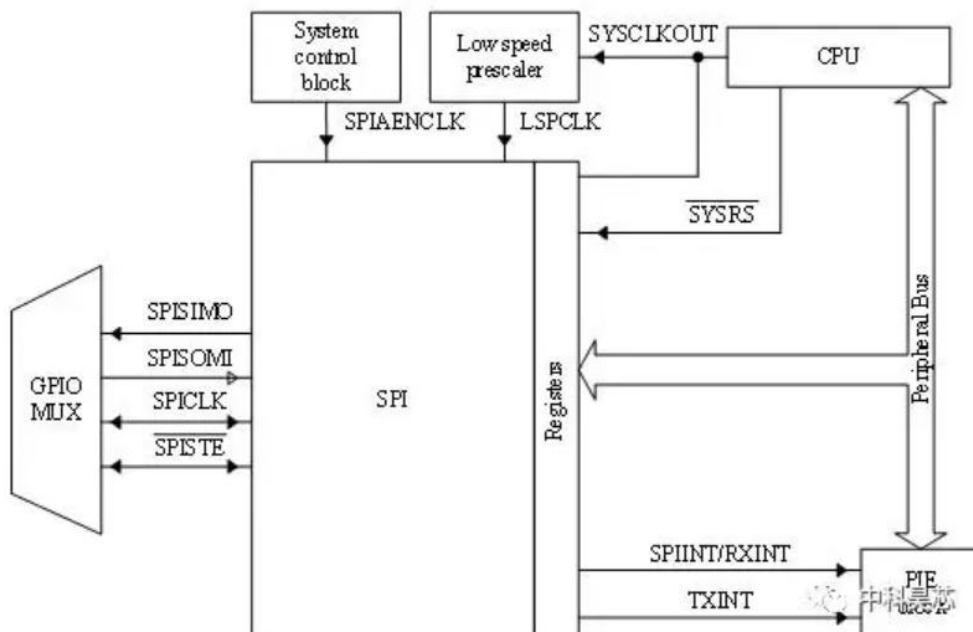
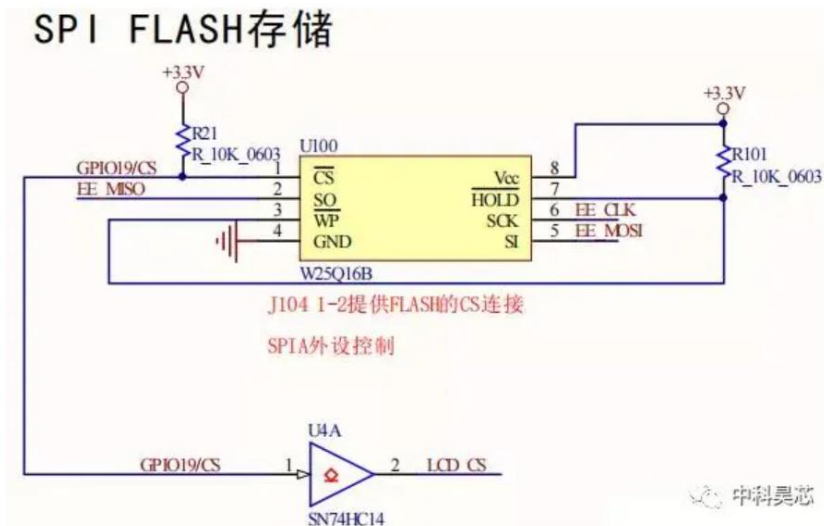
- ①通过 SPICCR[SPI SW RESET]位清零复位 SPI；
- ②根据需要进行 SPI 的初始化、格式与波特率及管脚功能配置；
- ③SPI SW RESET 位置 1，使 SPI 脱离复位。

(2) FIFO 配置：

- ①通过 SPIRST 在任一阶段复位 FIFO 模式；
- ②通过 SPIFFTX[SPIFFENA]置位使能 FIFO 模式，以激活 SPI 及其 FIFO 寄存器；
- ③发送 SPITXINT，接收错误与溢出中断 SPIRXINT 配置；
- ④通过 SPIFFCT 调整传输速率延迟 0~255 个 SPCLK 周期，以匹配外设通讯速率；
- ⑤通过 TXFFST 或 RXFFST 判断发送与接收到的字的数量，确认收发成功。

本例程主要完成 SPI 与 W25Q64 系列 FLASH 数据访问功能，通过 IO MUX 配置 GPIO19 在传输期间维持低电平，复位初始化期间高电平输出，与 SPI 外设的选通 CS 引脚相连，实现 FLASH 数据读写。

其连接原理如下图，访问过程为：




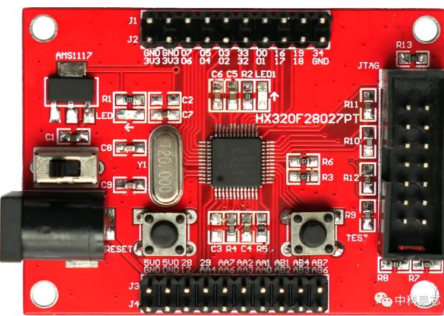

- ①读 Flash. ID 与 Flash. SReg 状态;
- ②写使能, 读取读与写寄存器状态;
- ③写使能, 擦除芯片;
- ④延时等待 4 个周期, 等待芯片数据擦除完成;
- ⑤写使能, 向 FLASH 写入数据;
- ⑥读出写入 FLASH 芯片的数据;

注: 每次读写后需延时等待, 以匹配外设传输速率。

详细介绍参见公众号 B 站视频讲解, 二维码为下图:



工欲善其事必先利其器, 程序开发前准备阶段如下表:

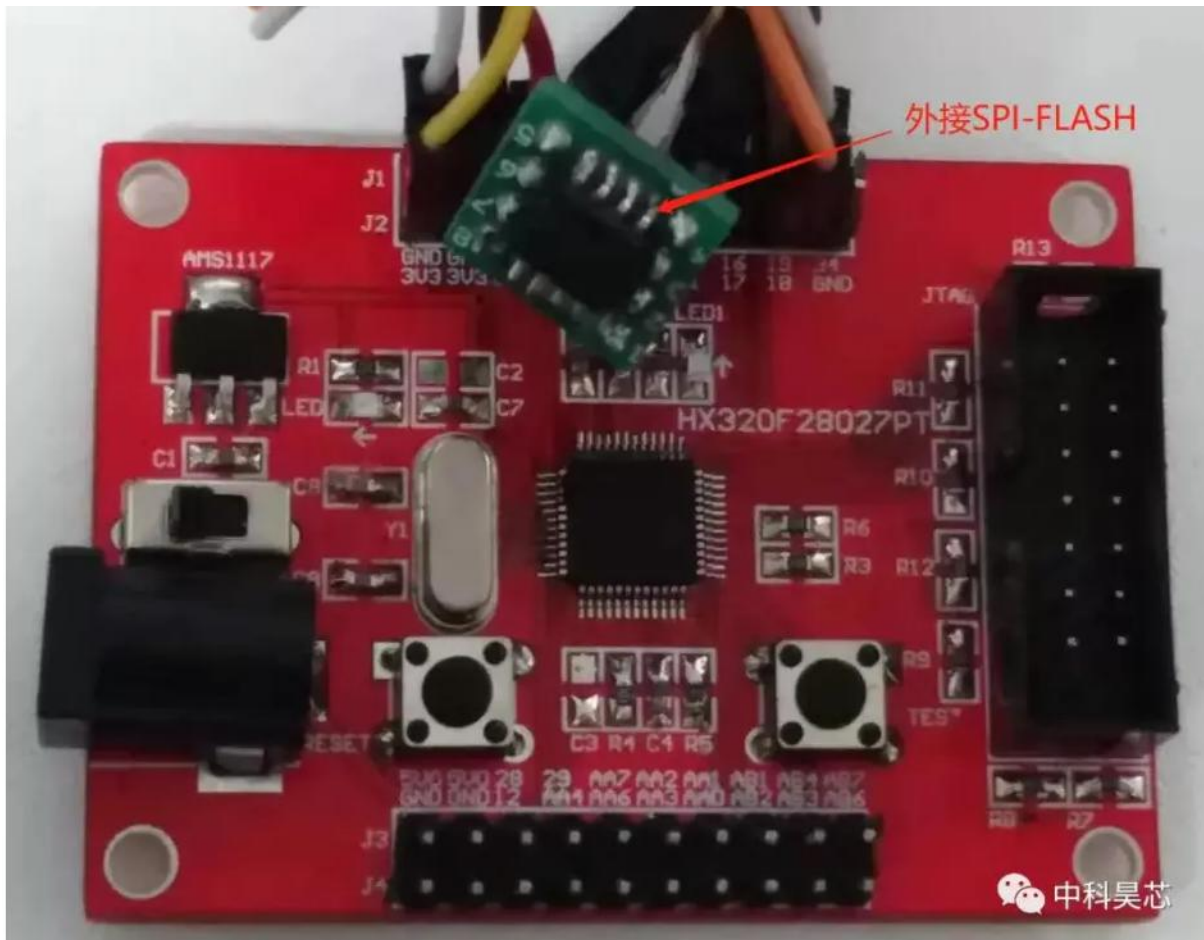
开发环境	开发板	仿真器
剑池集成开发环境 V2.10.1	Core_DSC28027 核心板	HX100V2
		
下载地址: https://occ.t-head.cn/community/download?id=575997419775328256	资料地址: http://haawking.cn/core28027	申请地址: http://haawking.cn/DSP-EMULATOR

下载界面如下:



准备好开发工具后就可以开始程序开发。详细的“CDK”安装及创建工程方法请看第一篇推送《芯教程 | 平头哥 CDK 助力中科昊芯 HX2000 系列芯片系统开发》。

例程选用 HXS320F28027 核心板与 W25Q64 系列 FLASH 进行数据访问，硬件连接如下：



接着在 CDK 上开发 SPI 访问 FLASH 读写数据程序，

代码包括：

- ①外设 GPIO 引脚、复位初始化操作与 FIFO 配置；
- ②SPI 发送与接收、W25Q64 系列 FLASH 芯片访问程序；

```
int main(void)
{
    /*系统初始化控制*/
    InitSysCtrl();
    /*LED 灯 GPIO 配置，用于判断 SPI 访问 FLASH 数据读写状态*/
    InitLED();
}
```

```

/*定义两个循环变量，用于延时操作*/
int i, j=0;
/*SPI 的外设引脚 GPIO 配置*/
SPI_I0init();
/*SPI FIFO 配置*/
SPI_fifo_init();
/*SPI 复位初始化操作*/
spi_init();
/*读取 W25Q64 系列 Flash 的 ID 信息*/
SPIFlash.Jedec_ID=Jedec_ID_Read();
/*读取 W25Q64 系列 Flash 的 SReg 寄存器状态*/
SPIFlash.SReg=Read_Status_2Reg();
/*写使能*/

WREN();
/*读取 W25Q64 系列 Flash 的写寄存器状态*/
WrSReg(0x0000);
SPIFlash.SReg=Read_Status_2Reg();
/*延时等待，等待写寄存器状态读写完成*/
Wait_Busy();
SPIFlash.SReg=Read_Status_2Reg();

WREN();
/*擦除芯片*/
Chip_Erase();
/*延时等待，等待 FLASH 芯片擦除成功*/
Wait_Busy();
Wait_Busy();
Wait_Busy();
Wait_Busy();
/*读出 FLASH 芯片数据，确认 FLASH 芯片数据擦除成功*/
ReadData(0, upper_128, 128);

WREN();
/*向 FLASH 芯片写入数据*/
PageProgram(0, upper_100, 128);
/*延时等待，等待数据写入完成*/

```

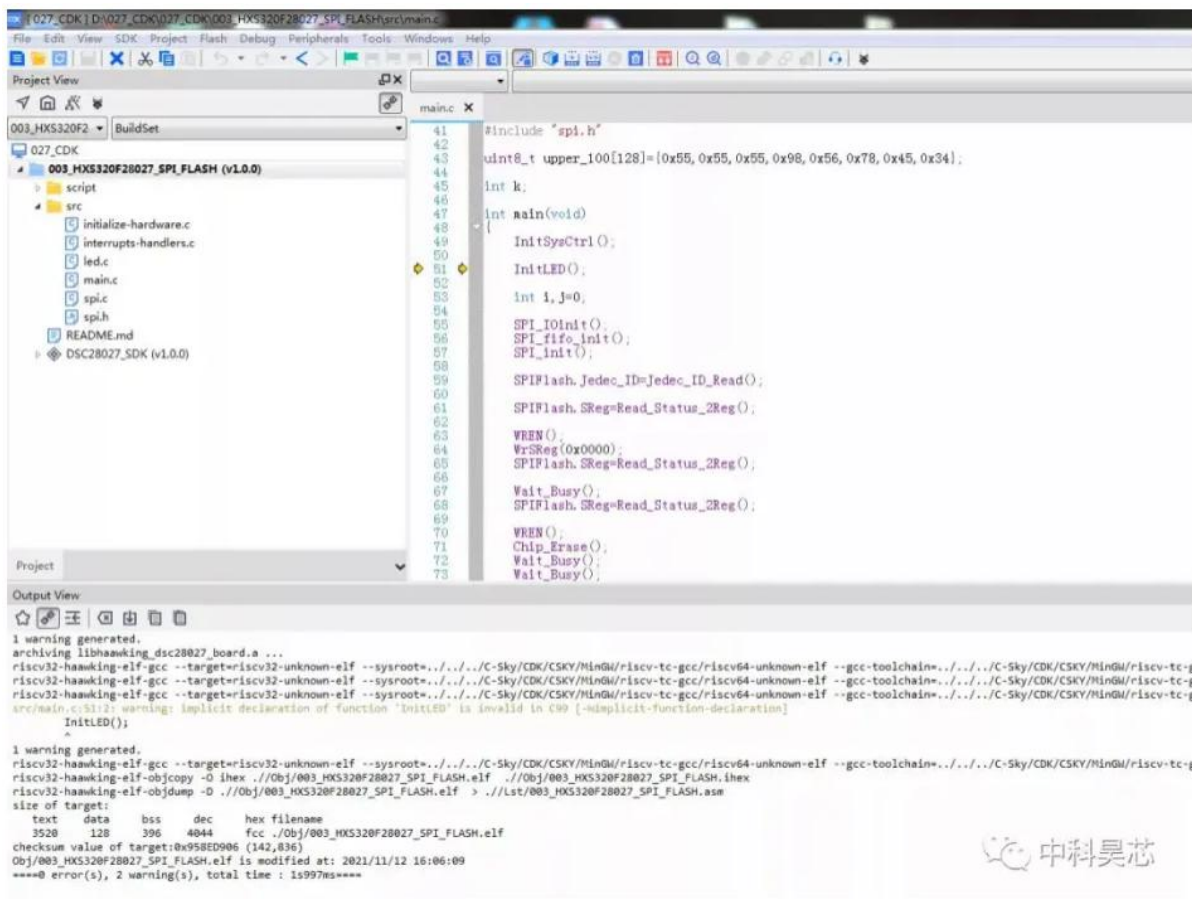
```

Wait_Busy();
/*读出 FLASH 芯片所写入的数据*/
ReadData(0, upper_128, 128);

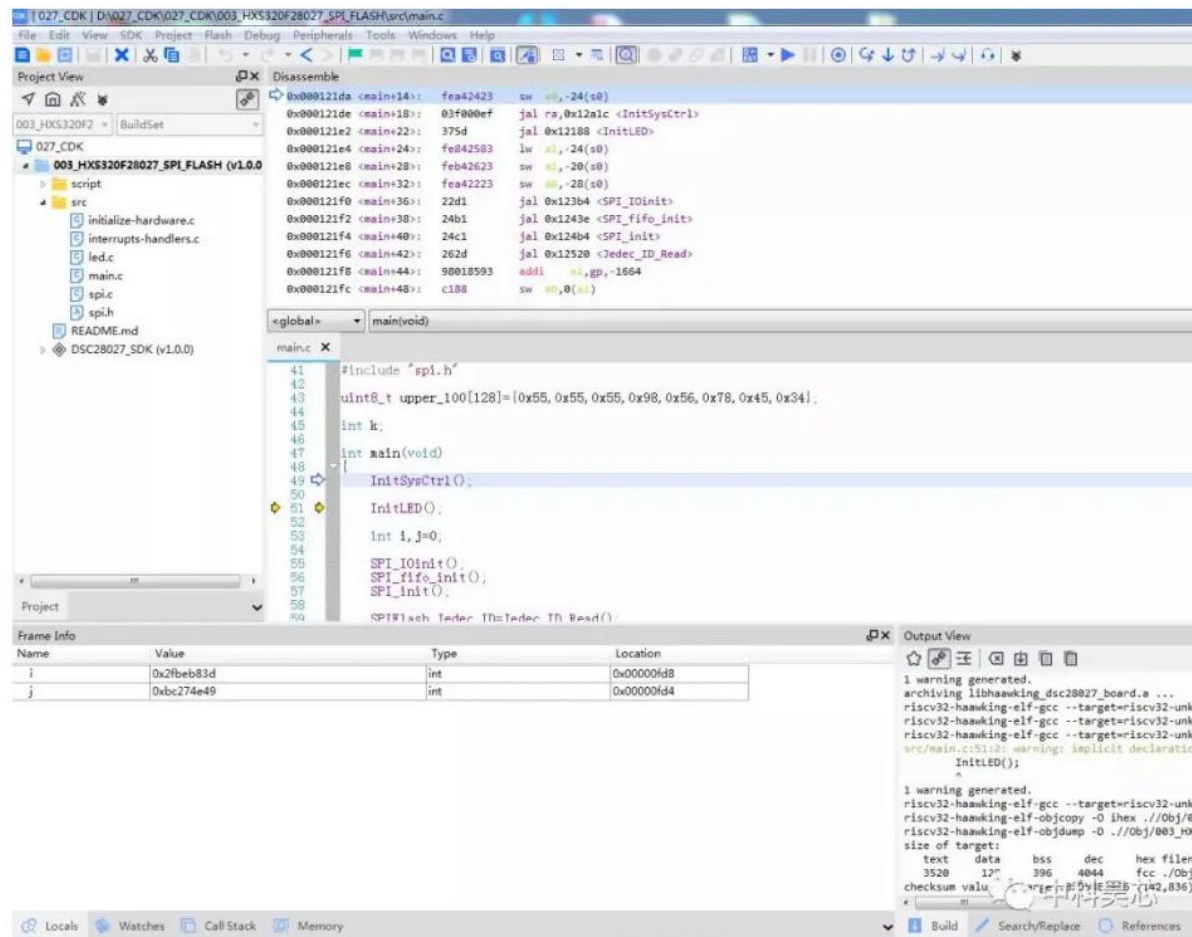
while(1)
{
    for(k=0;k<129;k++)
    {
        /*判断写入 FLASH 的数据与 FLASH 接收数据是否一致，一致在亮灯 GPI00*/
        if(upper_128[k]==upper_100[k])
        {
            GpioDataRegs.GPACLEAR.bit.GPI00=1;
            /*延时，使亮灯维持较长时间，以便于观察*/
            for(i=0;i<1000;i++)
            {
                for(j=0;j<1000;j++)
                {
                }
            }
        }
        /*FLASH 接收数据与 SPI 发送数据不一致，则 GPI00 对应 LED 灯灭*/
    }
    else
    {
        GpioDataRegs.GPASET.bit.GPI00=1;
        for(i=0;i<1000;i++)
        {
            for(j=0;j<1000;j++)
            {
            }
        }
    }
}
return 0;
}

```


在 CDK 上开发 SPI 访问 FLASH 读写数据程序的截图如下：



编译成功后就可进入“Debug”模式调试了，其界面如下：



下图一为 HX2000 系列芯片 SPI 发送的数据 upper_100，图二为 FLASH 接收的数据 upper_128，由图可得 upper_128 数组与 upper_100 数组相等，故 SPI 读写 FLASH 数据成功。

Frame Info			
Expression	Value	Type	Location
upper_100	[128]	uint8_t [128]	0x00012e80
upper_128	[128]	uint8_t [128]	0x00012f80
0	85 'U'	uint8_t	0x00012f80
1	85 'U'	uint8_t	0x00012f81
2	85 'U'	uint8_t	0x00012f82
3	152 '\230'	uint8_t	0x00012f83
4	86 'V'	uint8_t	0x00012f84
5	120 'x'	uint8_t	0x00012f85
6	69 'E'	uint8_t	0x00012f86
7	52 '4'	uint8_t	0x00012f87
8	0 '\000'	uint8_t	0x00012f88
9	0 '\000'	uint8_t	0x00012f89
10	0 '\000'	uint8_t	0x00012f8a

接收端

(图一)

Frame Info			
Expression	Value	Type	Location
upper_100	[128]	uint8_t [128]	0x00012e80
0	85 'U'	uint8_t	0x00012e80
1	85 'U'	uint8_t	0x00012e81
2	85 'U'	uint8_t	0x00012e82
3	152 '\230'	uint8_t	0x00012e83
4	86 'V'	uint8_t	0x00012e84
5	120 'x'	uint8_t	0x00012e85
6	69 'E'	uint8_t	0x00012e86
7	52 '4'	uint8_t	0x00012e87
8	0 '\000'	uint8_t	0x00012e88
9	0 '\000'	uint8_t	0x00012e89
10	0 '\000'	uint8_t	0x00012e8a
11	0 '\000'	uint8_t	0x00012e8b

发送端

(图二)

关于中科昊芯

“智由芯生 创享未来”，中科昊芯是数字信号处理器专业供应商。作为中国科学院科技成果转化企业，瞄准国际前沿芯片设计技术，依托多年积累的雄厚技术实力及对产业链的理解，以开放积极的心态，基于开源指令集架构 RISC-V，打造多个系列数字信号处理器产品，并构建完善的处理器产品生态系统。产品具有广阔的市场前景，可广泛应用于数字信号处理、工业控制及电机驱动、数字电源、消费电子、白色家电等领域。