SPGT62C19B 电机控制模组

使用说明书

V1.0 – 2005.8.16

凌阳科技大学计划教育推广中心
北京海淀上地信息产业基地中黎科技园 1 号楼 5 层
TEL: 86-10-62981668　　FAX: 86-10-62962425　　E-mail:unsp@sunplus.com.cn　　http://www.unsp.com.cn

版权声明

目　　录

1 模组简介

SPGT62C19B 电机控制模组是为学生以及单片机爱好者学习步进电机和直流电机控制而设计的学习套件。模组采用凌阳 SPGT62C19B 电机驱动芯片，配置两相步进电机和直流电机各一台，并提供 4 位 LED 数码管用来显示电机转速等信息。模组针对 SPCE061A 单片机设计，可以方便地用排线与 SPCE061A 精简开发板（即“61 板”）连接，可作为单片机教学、产品开发前期验证等辅助工具使用。模组的平面图如图 1.1 所示：

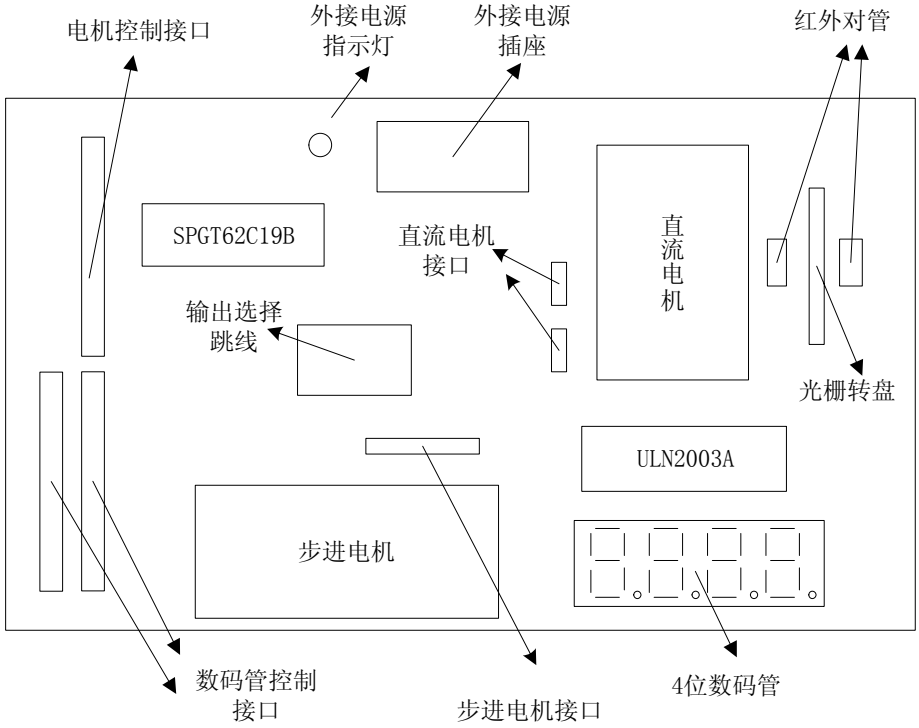


图 1.1 电机控制模组结构图

上述结构图中各部分说明如下：

电机控制接口：模组与单片机的接口，为 10PIN 排针，可以直接与“61 板”连接，实现电机控制。

数码管控制接口：模组与单片机的接口，为两组 10PIN 排针，可以直接与“61 板”连接，实现对 4 位 LED 数码管的控制。

SPGT62C19B：电机驱动芯片，可驱动一台双极性两相步进电机，或者两台直流电机。

外接电源指示灯： SPGT62C19B 电机驱动芯片的逻辑控制电源与电机驱动电源是各自独立供电的，可以外接 5V~12V 的电机驱动电源。当接通了电机驱动电源时，外接电源指示灯会点亮。

外接电源插座：为 SPGT62C19B 提供电机驱动电源的插座。共有两组电源插座，分别为 2PIN 针座（可接 61 板电池盒或其他直流电源）和 DC 稳压电源插座（可接直流稳压电源）。使用时可选择其中一组插座作为电机驱动电源输入端。

输出选择跳线：该组跳线用来选择 SPGT62C19B 芯片控制的电机。模组提供了步进电机和直流电机各一台，可通过对输出选择跳线的设定来切换当前工作的电机类型。

步进电机接口：该接口为 4PIN 插针形式，用于连接 SPGT62C19B 驱动芯片和两相步进电机。

步进电机：35BYJ26 型号永磁式步进减速电机，工作方式为两相四拍。在步进电机面板上安装有刻度盘，以便于在实验中观察电机的转动状态。

直流电机接口：由于 SPGT62C19B 可同时驱动两台直流电机，因此留出了两组直流电机接口，在模组上分别标示为 J11 和 J12。可以将模组提供的直流电机接在其中一组接口上。

直流电机：电机型号为 310CA，工作电压 3V~12V，在 5V 电压下空载转速约 4000 转/分。

光栅转盘和红外对管：在直流电机的转轴上安置了光栅转盘，光栅转盘的两侧分别装有鼠标用红外发射和接收管。当直流电机转动时，光栅将不断改变红外对管的通断状态，从而实现对直流电机转速的测量。

ULN2003A：ULN2003A 是单片式 7 路达林顿三极管阵列，在本模组中用来驱动 4 位 LED 数码管。

4 位数码管：4 位 8 段共阳极 LED 数码管，可用作电机转速显示，也可用于显示其他内容。

2 硬件模块说明

2.1 步进电机

模组配备的步进电机为 35BYJ26 型永磁步进电机，工作方式为双极性两相四拍。步进电机是一种将电脉冲转化为角位移的执行机构。当步进电机接收到一个脉冲信号，它就按设定的方向转动一个固定的角度(称为“步距角”)。可以通过控制脉冲个数来控制角位移量，从而达到准确定位的目的；同时可以通过控制脉冲频率实现步进电机的调速。

为使步进电机获得更大的转动力矩和更高的精度，电机内部加入了减速装置，通过变速齿轮组将电机的输出转速减小至 1/64。这样，步进电机每接收到一个脉冲信号，输出转动的角度（即步距角）可以达到 15° 的 1/64。

电机共引出 4 根控制线：

表 2.1 步进电机控制线

控制线颜色	蓝	黄	粉	橙
控制线名称	1A	1B	2A	2B

其中，1A 与 1B 是电机内部一组线圈的两个抽头，2A 与 2B 是另一组线圈的两个抽头。只需以一定的顺序控制两组线圈中的电流方向即可使步进电机按指定方向转动。例如，下面的通电时序可使电机逆时针转动：

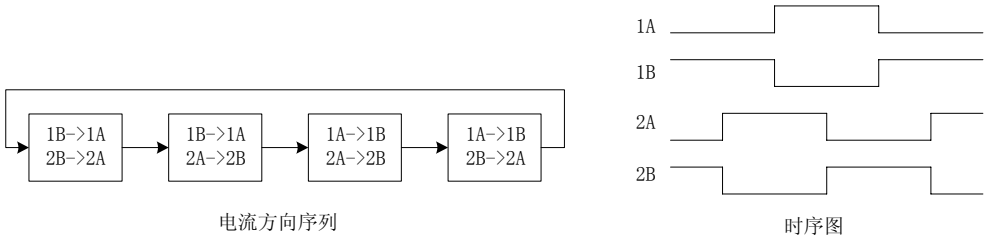


图 2.1 使步进电机逆时针转动的控制时序

35BYJ26 步进电机的主要技术参数如下：

表 2.2 35BYJ26 步进电机技术参数

电压	相电阻	步距角	减速比	启动转矩（g.cm）	启动频率（P.P.S）	定位转矩（g.cm）
5~12V	250Ω	15°/64	1:64	≥200	≥250	≥200

此外，步进电机面板上还加装了刻度盘和指针，以便于观察步进电机的转动情况。

2.2 直流电机

直流电机的控制方法比较简单，只需给电机的两根控制线加上适当的电压即可使电机转动起

来，电压越高则电机转速越高。对于直流电机的速度调节，可以采用改变电压的方法，也可采用 PWM 调速方法。PWM 调速就是使加在直流电机两端的电压为方波形式，通过改变方波的占空比实现对电机转速的调节。

模组配备的直流电机型号为 310CA，其电压范围为 3V~12V，在 5V 的供电电压下，其空载转速在 4000 转/分左右，空载电流约 20mA，堵转电流约 300mA。

直流电机转轴上加装了光栅转盘，可用来测量电机的转速，也可便于观察电机的转动情况。光栅转盘遮挡在红外发射管和红外接收管之间。如下图所示，光栅转盘的圆面上开了 4 个通光槽，电机每转动一周，红外接收管将接收到 4 次红外光，从而可以实现电机测速功能。

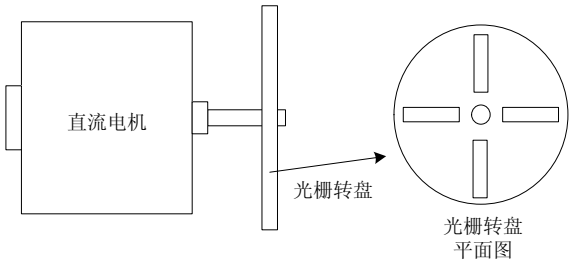


图 2.2 直流电机的光栅转盘

2.3 用 SPGT62C19B 芯片实现电机控制

2.3.1 SPGT62C19B 芯片简介

SPGT62C19B 是低电压单片式步进电机驱动器集成电路芯片，可驱动一台两相步进电机，或者两台直流电机。它带有双路 H 桥，可分别驱动两个独立的 PNP 功率管。每一个 H 桥都有各自独立的使能引脚，因此非常适合于需要独立控制的步进电机驱动系统。

SPGT62C19B 输出电压可达 40v，输出电流可达 750mA，由输入的逻辑电平来决定输出脉冲的宽度及频率，所以由这款芯片组成的电机驱动系统将脉冲发生器、脉冲分配器、脉冲放大器合为一体，省去了很多外围器件。

SPGT62C19 的内部由两组完全相同的控制电路组成了两路输出通道。其中一路通道的控制电路原理如图 2.3 所示。输入控制信号经前级缓冲后送入片内控制器，然后由控制部分进行处理并驱动晶体管，最后由 OUT 端口输出驱动信号以控制电机的运行。

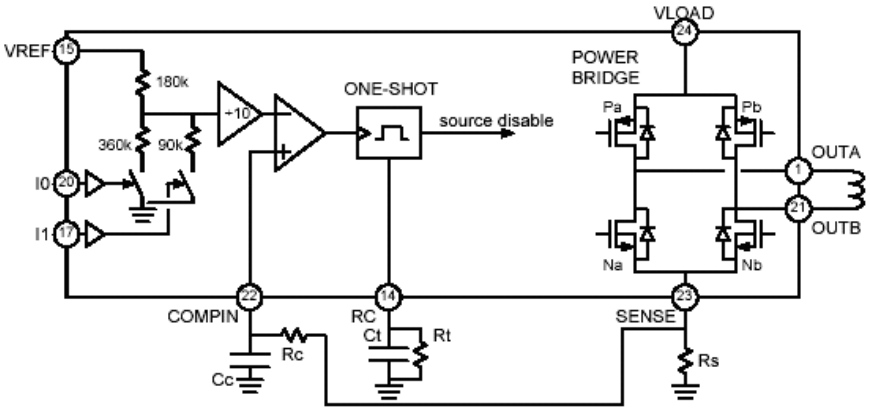


图 2.3 SPGT62C19B 工作原理

SPGT62C19B 的控制脚有如下 6 个：

表 2.3 SPGT62C19B 的控制引脚

引脚	名称	用途
20	I01	通道 1 的电流大小控制
17	I11	通道 1 的电流大小控制
16	PHASE1	通道 1 的电流方向控制
8	I02	通道 2 的电流大小控制
9	I12	通道 2 的电流大小控制
10	PHASE2	通道 2 的电流方向控制

以通道 1 为例，控制口 I01 与 I11 的不同逻辑组合可使通道 1 输出端产生不同大小的电流输出：

表 2.4 控制脚 I01 与 I11 逻辑组合与输出电流的关系

I01 逻辑值	I11 逻辑值	输出电流
0	0	Imax
1	0	2/3*Imax
0	1	1/3*Imax
1	1	0

上表中，Imax 是输出电流的上限值，它与图 2.3 中 Vref 和 Rs 的值有关。其关系式为：

Imax = Vref /10*Rs：

PHASE1 的逻辑电平值决定了该通道的电流输出方向。PHASE1 与电流方向的对应关系为：

表 2.5 控制脚 PHASE1 与输出电流的关系

PHASE1 逻辑值	输出电流方向
0	OUT1B -> OUT1A
1	OUT1A -> OUT1B

对于 SPGT62C19B 的更详细说明可参考模组配套资料中的 SPGT62C19B Datasheet。

2.3.2 利用 SPGT62C19B 控制直流电机

SPGT62C19B 的两个输出通道可以分别控制一台直流电机。仍以通道 1 为例，只需设定 PHASE1 的逻辑电平，即可实现电机的正反转控制。而电机调速可以通过不断改变 I01 和 I11 的高低电平状态，使输出通道产生 PWM 波形信号，从而利用 PWM 的占空比来调节电机转速。

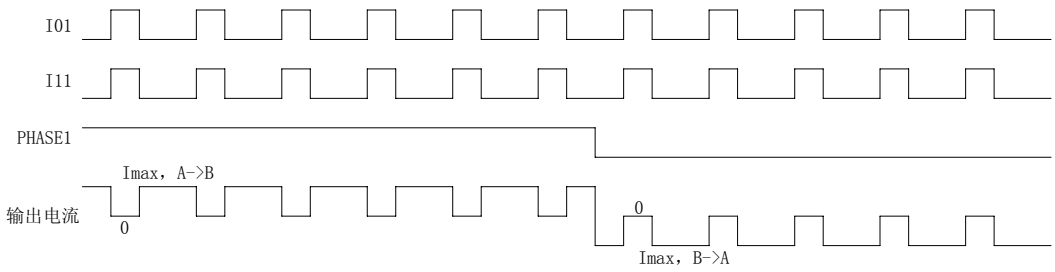


图 2.4 输出 PWM 控制直流电机

2.3.3 利用 SPGT62C19B 控制步进电机

两相步进电机的 4 根引线分为两组，可分别连接 SPGT62C19B 的两个输出通道。这样，就可以通过 SPGT62C19B 的 6 个控制引脚使两个输出通道发出驱动步进电机所需的脉冲信号。步进电机与 SPGT62C19B 的连接方法如下：

表 2.6 步进电机与 SPGT62C19B 的连接方法

电机引线颜色	电机引线名称	SPGT62C19B 引脚序号	SPGT62C19B 引脚名称
蓝	1A	1	OUT1A
黄	1B	21	OUT1B
粉	2A	2	OUT2A
橙	2B	5	OUT2B

由 2.3.1 节可知，每通道的输出电流可以有 4 种状态，这为步进电机提供了多种控制方式，可实现“整步（Full-Step）”、“半步（Half-Step）”、“优化半步（Modified Half-Step）”等工作模式。这几种工作模式的控制时序如下图所示：

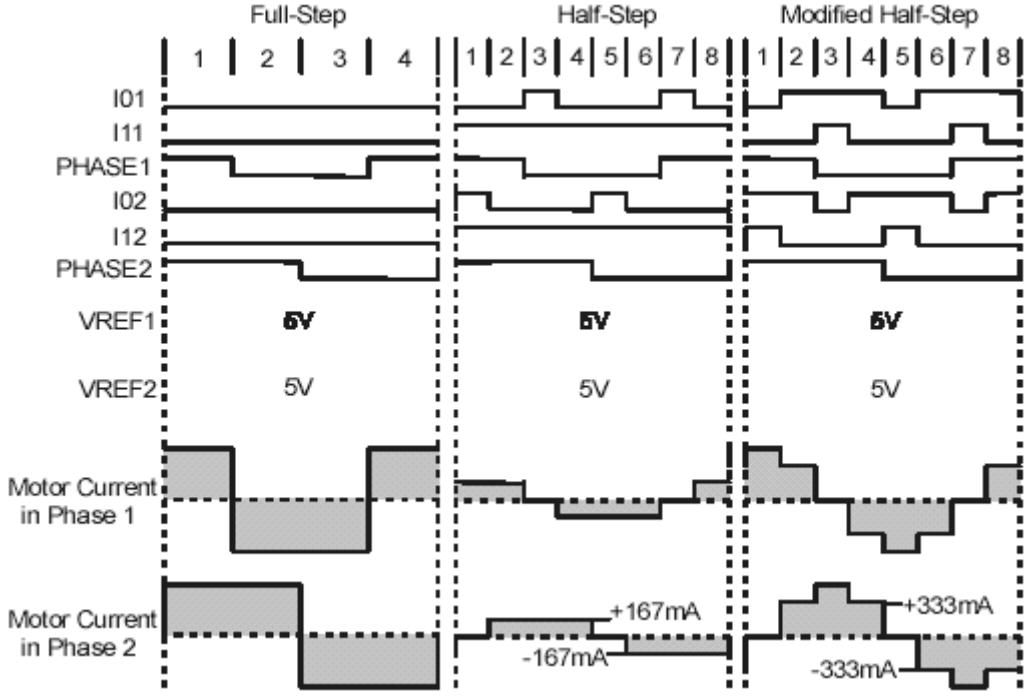


图 2.5 步进电机不同控制模式的时序图

对于半步模式和优化半步模式，与整步模式相比，步进电机每相中的电流变化更为平缓，这使得步进电机转动更加平滑，在一定程度上也可起到提高步距角分辨率的作用。

2.4 转速测量电路

转速测量采用一组鼠标上用的红外对管实现。其电路原理如图 2.6 所示。

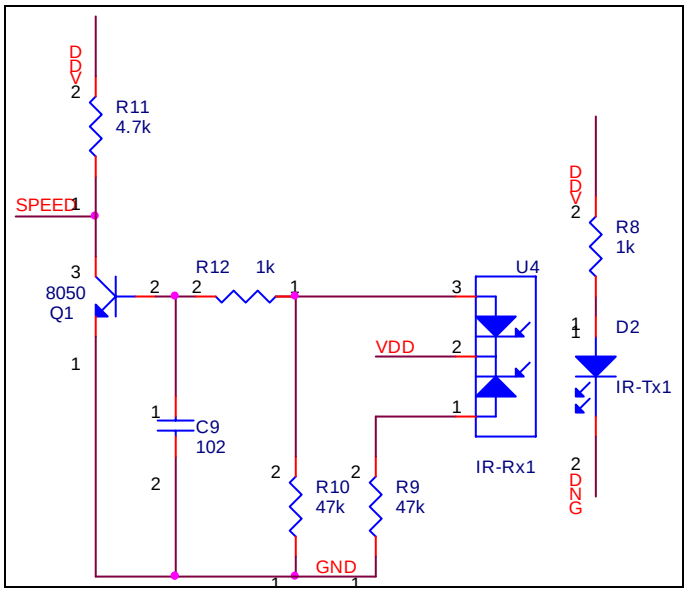


图 2.6 转速测量电路

当红外发射管与红外接收管之间被直流电机光栅转盘的不透明部分遮挡时，红外接收管处于截止状态，此时图中的 SPEED 输出高电平。反之，当光栅转盘的通光槽转至红外对管之间时，红外接收管处于导通状态，此时 SPEED 输出低电平。将 SPEED 连接到单片机的 I/O 口，即可通过定时计数的方法计算出电机转动速度。

2.5 数码管显示电路

模组提供了 4 位共阴极 LED 数码管，数码管采用 ULN2003A 为其提供驱动电流。ULN2003A 是 7 路达林顿三极管阵列，这里用到了其中的 4 路，分别连接到数码管的 4 个位选脚 G1~G4，如下图所示。

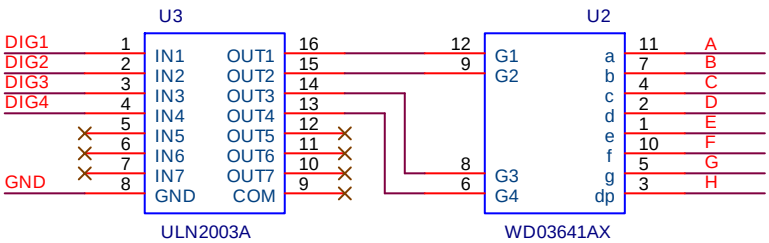


图 2.7 数码管显示电路

选取单片机的 8 位 I/O 作为数码管的“段控制”口，连接到数码管的 A~H 这 8 个段控制脚；再用 4 位 I/O 作为数码管的“位控制”口，连接到驱动芯片 ULN2003A 的 DIG1~DIG4，即可实现数码管显示控制。数码管的各段、位与控制引脚的对应关系如下图所示。

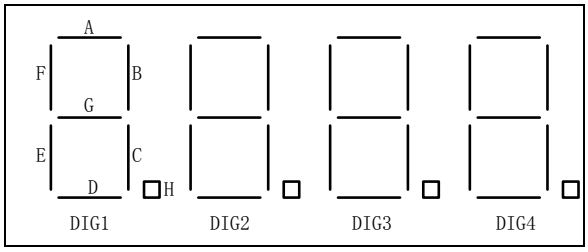


图 2.8 数码管段、位与控制端口的对应关系

。

3 硬件连接与跳线配置

3.1 电机控制部分

电机控制的相关接口包括三个方面：电机控制芯片 SPGT62C19B 与单片机之间的接口，步进电机和直流电机驱动输出接口，以及电机供电电源接口。此外，还有输出选择跳线用于步进电机与直流电机之间的切换。

3.1.1 SPGT62C19B 与单片机之间的接口

SPGT62C19B 与单片机之间的接口在模组上标示为“J3”，是 10PIN 排针的形式。该接口在模组中的位置和各管脚名称如下：

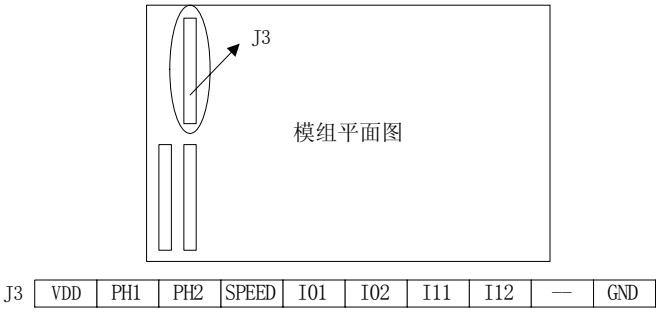


图 3.1 SPGT62C19B 与单片机的接口

其中 PH1、PH2、I01、I02、I11、I12 对应着 SPGT62C19B 的 6 个控制引脚（参见 2.3.1 节）；而 SPEED 则是速度检测信号输出脚（参见 2.4）。可以直接用 10PIN 排线将 J3 接口与 61 板 IOB 的低 8 位（即 61 板的 J6）相连。应注意的是，**模组接口标示为“VDD”的脚应与 61 板接口标示为“+”的脚相对应，不能接反。**另外，61 板的 I/O 供电电压应在 4.5V~5.5V 之间。因此，**要把 61 板的 I/O 电压选择跳线（61 板的 J5）跳至“5V”位置，并保证 61 板供电电压在 4.5V 以上**，建议使用 5V 稳压电源给 61 板供电。

3.1.2 步进电机和直流电机驱动输出接口

步进电机和直流电机驱动输出接口是 SPGT62C19B 驱动芯片与电机之间的连接端口。步进电机接口在模组上标示为 J10，是 4PIN 排针形式，对应着步进电机的 4 根引线，以及 SPGT62C19B 的 OUT1A、OUT1B、OUT2A、OUT2B 这 4 个输出端。直流电机接口共有两组，在模组上标示为 J11 和 J12，分别对应 SPGT62C19B 的 OUT1 和 OUT2 两个输出通道；直流电机的两根引线可以与其中一组接口相接。

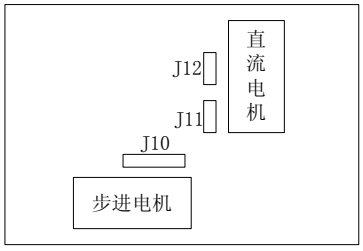


图 3.2 步进电机与直流电机接口在模组中的位置

3.1.3 电机供电电源接口

SPGT62C19B 芯片有两个电源输入口，分别给逻辑控制电路和电机驱动电路供电。因此，需要外接独立的电源给电机供电，要求外接电源的电压在 5V~12V 之间，并可以提供 750mA 以上的电流。模组提供了两种形式的电机供电电源接口，分别是 2PIN 针座和直流稳压电源插座；可以将外部电源连接在其中一组电源接口上。

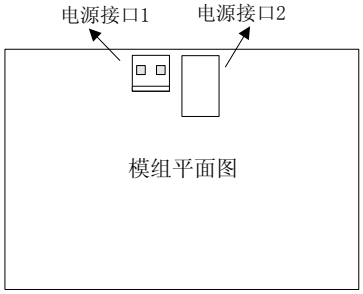


图 3.3 电机供电电源接口在模组中的位置

3.1.4 输出选择跳线

由于 SPGT62C19B 支持一台步进电机或两台直流电机，而不能同时控制步进电机与直流电机，因此设置了“输出选择跳线”用于在步进电机与直流电机之间切换，如图 3.4 所示。当使用直流电机时，要将 4 个短接帽都跳接到“DC MOTOR”一端；使用步进电机时，要将 4 个短接帽跳接到“STEP MOTOR”一端。

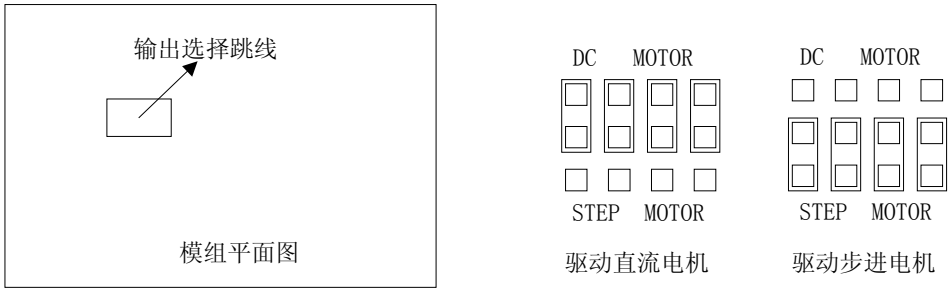
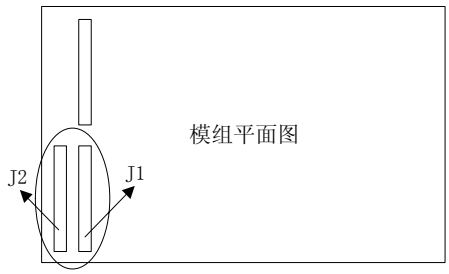


图 3.4 输出选择跳线位置以及配置方法

3.2 数码管显示部分

数码管显示接口包括“段控制”接口和“位控制”接口。它们在模组中被标示为“J1”和“J2”。它们在模组中的位置以及接口中各引脚的名称如图 3.5 所示。可以用 10PIN 排线将模组的 J1 与 61 板的 IOA 低 8 位（即 61 板的 J8）相连，模组的 J2 与 61 板的 IOA 高 8 位（即 61 板的 J9）相连。连接时要注意模组接口标示为“VDD”的脚应与 61 板接口标示为“+”的脚相对应，不能接反。



J1	VDD	A	B	C	D	E	F	G	H	GND
J2	VDD	DIG1	DIG2	DIG3	DIG4	—	—	—	—	GND

图 3.5 数码管与单片机的接口

4 模组驱动程序使用说明

为便于用户学习和使用，模组提供了基于 SPCE061A 单片机的驱动程序。它包括步进电机驱动程序、直流电机驱动程序和 LED 数码管驱动程序。用户可以直接调用驱动程序中的函数实现对步进电机、直流电机以及数码管的控制。

4.1 步进电机驱动程序

步进电机驱动程序包括 StepMotor.asm、StepMotor.inc 和 StepMotor.h 三个文件。StepMotor.asm 文件中定义了驱动步进电机所需的函数，而 StepMotor.inc 和 StepMotor.h 两个“头文件”对这些函数进行了声明，可以被用户应用程序所包含（扩展名为.inc 的文件可被汇编程序文件包含，扩展名为.h 的文件可被 C 语言程序包含）。步进电机驱动函数说明如下：

1. StepMotor_Init（步进电机驱动初始化）

格式： C 语言： void StepMotor_Init(void)

汇编语言： F_StepMotor_Init

参数： 无

返回值： 无

说明： 步进电机初始化函数，它将开启 IRQ4_4KHz 时基中断。在控制步进电机时，首先要执行该函数。

2. StepMotor_SetMode（设置步进电机工作模式）

格式： C 语言： void StepMotor_SetMode(unsigned ModeCode)

汇编语言： F_StepMotor_SetMode

参数： ModeCode（R1）- 步进电机工作模式，可以是下列数值之一：

STEP_FULL – 整步模式

STEP_HALF - 半步模式

STEP_MOHALF - 优化半步模式

返回值： 无

说明： 该函数用于设定步进电机的工作模式。步进电机驱动程序提供了三种可选的工作模式——“整步模式”、“半步模式”和“优化半步模式”。

3. StepMotor_SetSpeed（设置步进电机转速）

格式： C 语言： void StepMotor_SetSpeed (unsigned Speed)

汇编语言： F_StepMotor_SetSpeed

参数： Speed（R1）- 步进电机转速，单位为拍/秒（P.P.S）

返回值： 无

说明： 执行该函数可以对电机转速进行设置，转速的上限由步进电机自身性能等因素决定。可修改 StepMotor.asm 文件中定义的 MOTOR_MAXSPEED 值，指定一个转速上限，这样，当函数的 Speed 参数值大于 MOTOR_MAXSPEED 时，程序会自动将 Speed 值调整为 MOTOR_MAXSPEED。

4. StepMotor_Forward（使步进电机正向转动）

格式： C 语言： void StepMotor_Forward(unsigned Step)

汇编语言： F_StepMotor_Forward

参数： Step（R1）- 使电机转动的拍数，范围从 1~65535。当 Steps 设置为 0 时，电机将持续转动

返回值： 无

说明： 电机的转动方向与电机接口的插接方向有关，因此所谓“正向”仅是一个相对概念，其实际转动方向有可能是顺时针，也有可能是逆时针。

5. StepMotor_Backward（使步进电机反向转动）

格式： C 语言： void StepMotor_Backward(unsigned Step)

汇编语言： F_StepMotor_Backward

参数： Step（R1）- 使电机转动的拍数，范围从 1~65535。当 Steps 设置为 0 时，电机将持续转动

返回值： 无

说明： “反向”是相对于“正向”而言的。实际转动方向有可能是顺时针，也有可能是逆时针。

6. StepMotor_Drive（步进电机驱动时序发生函数）

格式： C 语言： void StepMotor_Drive(void)

汇编语言： F_StepMotor_Drive

参数： 无

返回值： 无

说明： 该函数是在 IRQ4_4KHz 中断服务程序中调用的。使用步进电机驱动程序时，要创建 IRQ4 中断服务函数，并在 4KHz 中断服务中调用 StepMotor_Drive 函数。

7. StepMotor_Stop（步进电机停止转动）

格式： C 语言： void StepMotor_Stop (void)

汇编语言： F_StepMotor_Stop

参数： 无

返回值： 无

说明： 当执行了 StepMotor_Forward 或 StepMotor_Backward 函数使电机处于转动状态时，

执行 StepMotor_Stop 函数将使电机停止转动。

8. StepMotor_Status（获取步进电机当前转动状态）

格式： C 语言： unsigned StepMotor_Status (void)

汇编语言： F_StepMotor_Status

参数： 无

返回值： r1 - 电机转动状态，可能是下列数值：

0 - 停止状态

1 - 正向转动

0xffff - 反向转动

说明： 该函数可以查询电机当前的状态。例如执行了 StepMotor_Forward(100)，使电机转动 100 拍，可利用 StepMotor_Status 函数查询电机是否已经转动完成。

4.2 直流电机驱动程序

直流电机驱动程序包含 DCMotor.asm、DCMotor.inc 和 DCMotor.h 三个文件。DCMotor.asm 文件中定义了驱动直流电机所需的函数，而 DCMotor.inc 和 DCMotor.h 两个“头文件”对这些函数进行了声明，可以被用户应用程序所包含。直流电机驱动程序包括下列函数：

1. DCMotor_Init（直流电机驱动初始化）

格式： C 语言： void DCMotor_Init(void)

汇编语言： F_DCMotor_Init

参数： 无

返回值： 无

说明： 直流电机初始化函数，它将开启 IRQ4_4KHz 时基中断。在驱动直流电机时，首先要执行该函数。

2. DCMotor_SetSpeed（设置直流电机转速等级）

格式： C 语言： void DCMotor_SetSpeed(unsigned Motor,unsigned Speed)

汇编语言： F_DCMotor_SetSpeed

参数： Motor（R1）- 选择对哪个直流电机接口进行转速等级设定：

MOTOR_1 直流电机接口 1

MOTOR_2 直流电机接口 2

Speed（R2）– 速度等级，事实上是 PWM 占空比，范围 0-63，单位 1/64

返回值： 无

说明： 该函数用来对电机的转速进行设定，它是通过调节电机控制信号的 PWM 占空比实现的。参数 Speed 表示电机的转速等级，范围从 0~63，实际上就是控制信号的 PWM

占空比（0/64~63/64）。但对于较低的速度等级，可能无法使直流电机转动起来，一般 Speed 的值应大于 15。

3. DCMotor_Forward（使直流电机正向转动）

格式： C 语言： void DCMotor_Forward(unsigned Motor)

汇编语言： F_DCMotor_Forward

参数： Motor（R1）- 选择向哪个直流电机接口发出正转信号：

MOTOR_1 直流电机接口 1

MOTOR_2 直流电机接口 2

返回值： 无

说明： 执行该函数将使指定的直流电机转动。所谓“正向转动”是一个相对概念，电机的实际转动方向与电机接口的插接方向有关，可能是顺时针转动，也可能是逆时针转动。参数 Motor 可以为 MOTOR_1、MOTOR_2 或 MOTOR_1+MOTOR2。

4. DCMotor_Backward（使直流电机反向转动）

格式： C 语言： void DCMotor_Backward(unsigned Motor)

汇编语言： F_DCMotor_Backward

参数： Motor（R1）- 选择向哪个直流电机接口发出反转信号：

MOTOR_1 直流电机接口 1

MOTOR_2 直流电机接口 2

返回值： 无

说明： 执行该函数将使指定的直流电机转动。“反向转动”是相对与“正向转动”而言的，电机的实际转动方向可能是顺时针转动，也可能是逆时针转动。参数 Motor 可以为 MOTOR_1、MOTOR_2 或 MOTOR_1+MOTOR2。

5. DCMotor_Stop（使直流电机停止转动）

格式： C 语言： void DCMotor_Backward(unsigned Motor)

汇编语言： F_DCMotor_Backward

参数： Motor（R1）- 选择向哪个直流电机接口发出停转信号：

MOTOR_1 直流电机接口 1

MOTOR_2 直流电机接口 2

返回值： 无

说明： 执行该函数将使指定的直流电机停止转动。参数 Motor 可以为 MOTOR_1、MOTOR_2 或 MOTOR_1+MOTOR2。

6. DCMotor_Drive（产生直流电机驱动信号）

格式： C 语言： void DCMotor_Drive(void)

汇编语言： F_DCMotor_Drive

参数： 无

返回值： 无

说明： 该函数是在 IRQ4_4KHz 中断服务程序中调用的。使用直流电机驱动程序时，要创建 IRQ4 中断服务函数，并在 4KHz 中断服务中调用 DCMotor_Drive 函数。

7. DCMotor_GetSpeed（获取电机转速）

格式： C 语言： unsigned DCMotor_GetSpeed(void)

汇编语言： F_DCMotor_GetSpeed

参数： 无

返回值： r1 - 测得的电机转速值,单位(转/秒)

说明： 调用该函数可获得电机的转速。

4.3 LED 数码管驱动程序

数码管驱动程序包括 DIG.asm、DIG.inc 和 DIG.h 三个文件。DIG.asm 文件中定义了 LED 数码管显示相关函数，而 DIG.inc 和 DIG.h 两个“头文件”对这些函数进行了声明，可以被用户应用程序所包含。数码管驱动程序主要包括下列函数：

1. DIG_Init（数码管显示初始化）

格式： C 语言： void DIG_Init(void)

汇编语言： F_DIG_Init

参数： 无

返回值： 无

说明： 数码管显示初始化函数，它将开启 IRQ4_4KHz 时基中断。在使用数码管显示驱动程序时，首先要执行该函数。

2. DIG_Set（设置数码管某一位的显示内容）

格式： C 语言： void DIG_Set(unsigned DigPos, unsigned DigBuffer)

汇编语言： F_DIG_Set

参数： DigPos（R1）- 要设置的位，范围 1~4，自左至右对应 4 个数码管

DigBuffer（R2）- 数码管的显示内容

返回值： 无

说明： 该函数用来改变 4 位数码管中某一位的显示内容。参数 DigPos 指定要改变内容的位，DigBuffer 的低 8 位代表了 8 段数码管的八个“段”，其对应关系如下图所示：

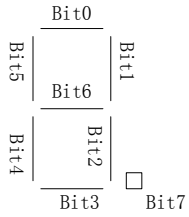


图 4.1 数码管各段对应的二进制位

3. DIG_SetAll（设置所有数码管的显示内容）

格式： C 语言： void DIG_SetAll(unsigned *DigBuffer)

汇编语言： F_DIG_SetAll

参数： DigBuffer（R1）- 数码管的显示内容的起始地址

返回值： 无

说明： 该函数用来设定 4 位数码管中每一位的显示内容。参数 DigBuffer 是一个指针，指向存放数码管显示内容的数组，该数组应含有 4 个 Word 数据，分别存储每位数码管的内容。

4. DIG_Drive（数码管显示执行函数）

格式： C 语言： void DIG_Drive(void)

汇编语言： F_DIG_Drive

参数： 无

返回值： 无

说明： 该函数是在 IRQ4_4KHz 中断服务程序中调用的。使用数码管显示驱动程序时，要创建 IRQ4 中断服务函数，并在 4KHz 中断服务中调用 DIG_Drive 函数。

5 应用举例

5.1 步进电机控制示例

例：设计程序使步进电机以 50 P.P.S 的速度正转 270 拍，然后以 100 P.P.S 的速度反转 590 拍。采用“半步”工作方式。

第 1 步：打开 unSP IDE，新建工程。这里将工程命名为 StepM，保存在 D 盘根目录下。

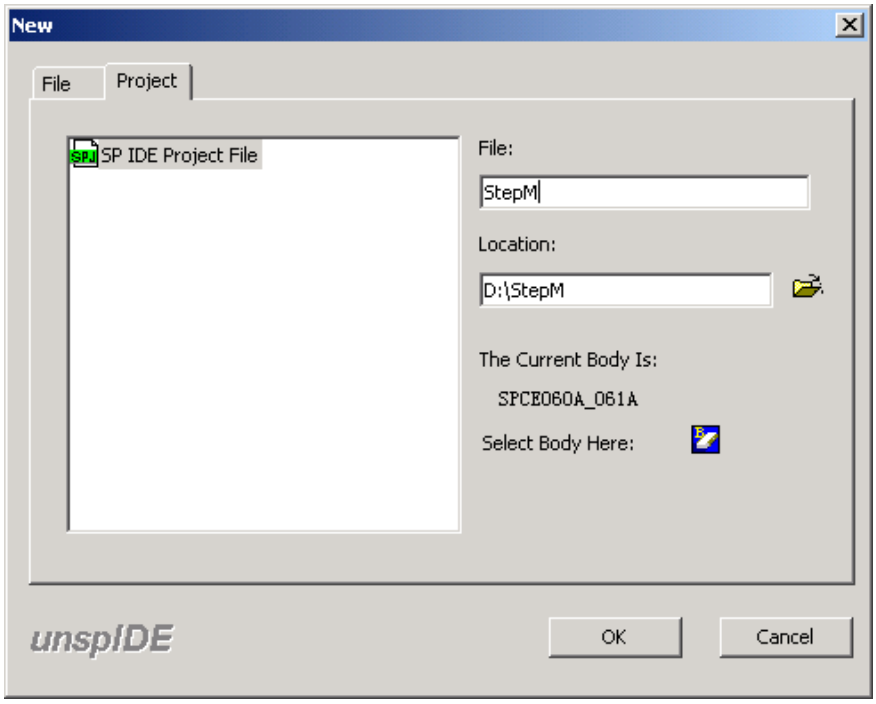


图 5.1 新建工程

第 2 步：将步进电机驱动程序文件复制到工程所在的文件夹，即 D:\StepM\。步进电机驱动程序文件可以在模组配套资料中找到，包括 StepMotor.asm、StepMotor.inc 和 StepMotor.h 三个文件。

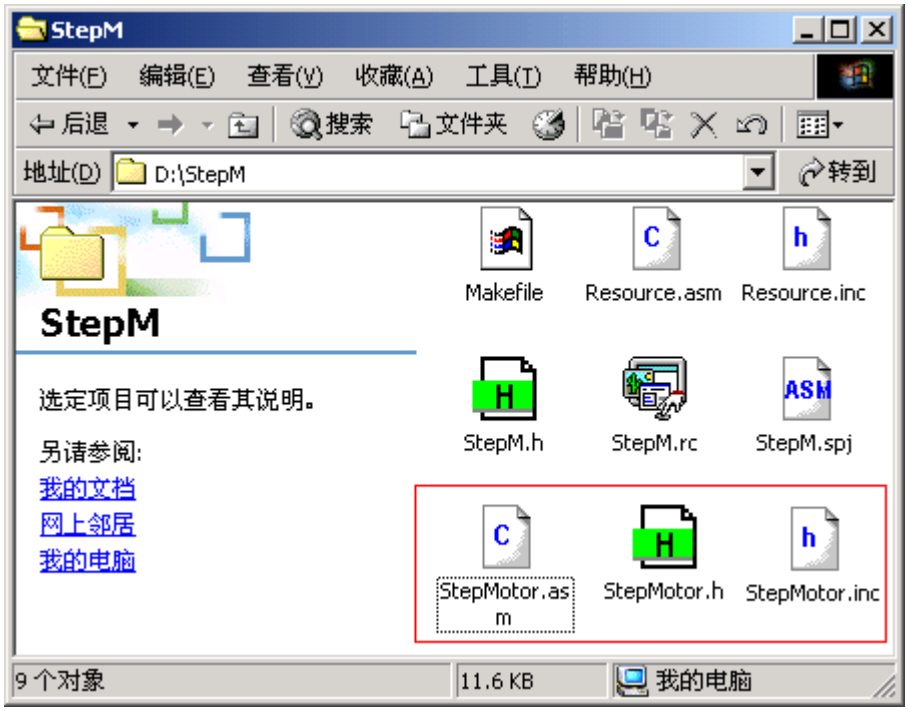


图 5.2 复制驱动程序文件

第 3 步：把复制过来的三个程序文件添加到工程中。方法是选择 unSP IDE 的“Project”菜单下的“Add to project”-> “Files”选项。在弹出的文件选择对话框中选中三个驱动程序文件，点击“打开”，如图 5.3 所示。这样，驱动程序就添加到工程中了。

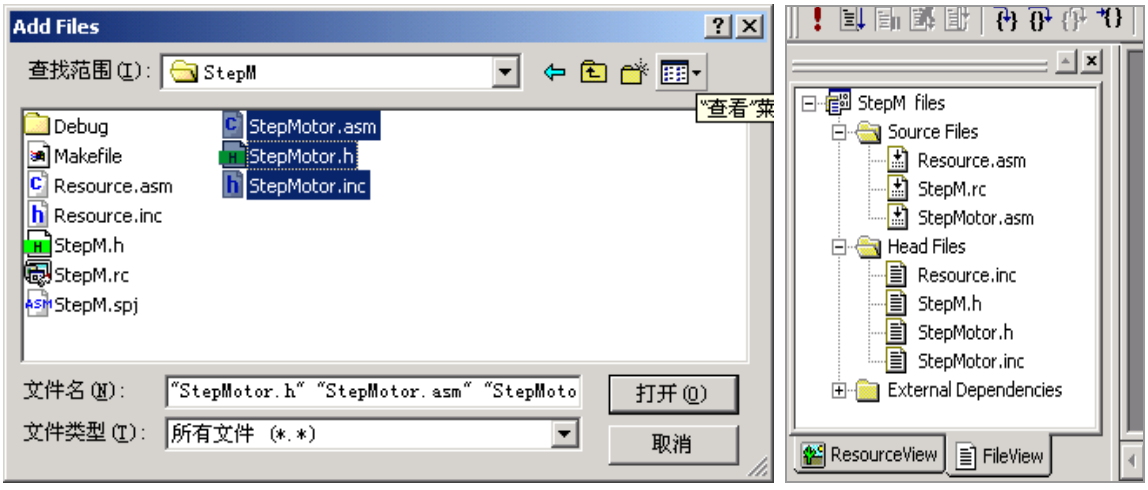


图 5.3 将驱动程序文件添加到工程

第 4 步：建立主程序文件，编写代码。在 IDE 的 File 菜单中选择“New”，新建一个 C 程序文件，可命名为“main.c”。

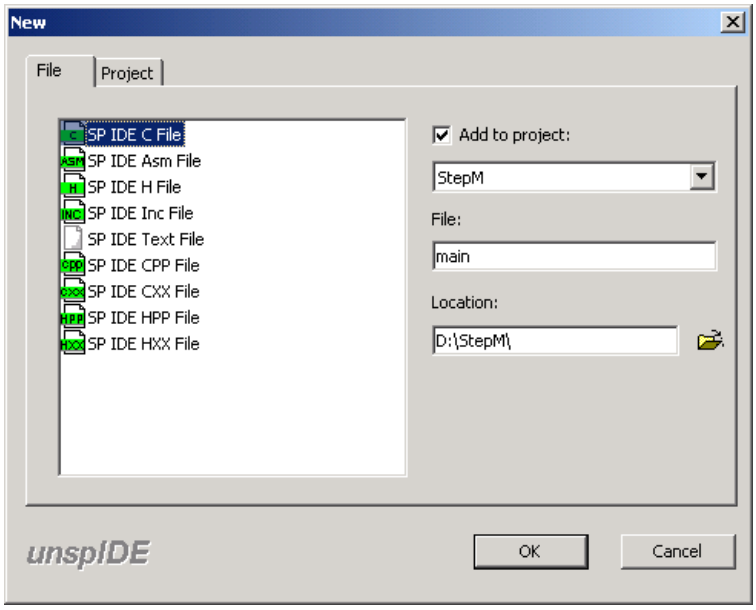


图 5.4 新建 C 程序文件

在建立的 C 程序文件中编写代码如下：

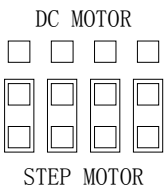
```
#include "StepMotor.h"                          // 包含步进电机驱动头文件

#define CLR_WDT()    *(unsigned *)0x7012=1      // 清看门狗

main()
{
    StepMotor_Init();                           // 初始化驱动程序
    StepMotor_SetMode(STEP_HALF);               // 设置为半步工作方式
    StepMotor_SetSpeed(50);                     // 速度设定为 50P.P.S
    StepMotor_Forward(270);                     // 正转 270 拍
    while(StepMotor_Status()!=0)                // 等待正转 270 拍结束
    {
        CLR_WDT();
    }
    StepMotor_SetSpeed(100);                    // 速度设定为 100P.P.S
    StepMotor_Backward(590);                    // 反转 590 拍
    while(StepMotor_Status()!=0)                // 等待反转 590 拍结束
```

```
    {
        CLR_WDT();
    }

    while(1)CLR_WDT();                          // 程序完成，进入死循环
}
```

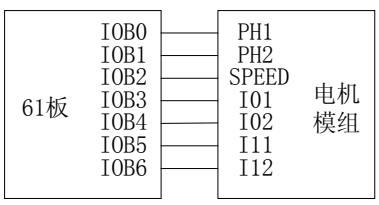
第 5 步：建立中断服务程序文件，编写中断服务函数。由于步进电机驱动需要使用 IRQ4_4KHz 中断，因此要建立中断服务程序。中断服务程序可以用 C 语言编写，也可用汇编语言编写。这里使用 C 语言编写，可以直接在 main.c 的 main()函数后面继续书写下列代码：

```
#define P_INT_Ctrl      (unsigned int*)0x7010       // 硬件端口定义
#define P_INT_Clear     (unsigned int*)0x7011       // 硬件端口定义
#define C_IRQ4_4KHz  0x0040                         // 常数助记符定义

void IRQ4(void)__attribute__((ISR));                // 将 IRQ4 函数声明为中断服务函数

void IRQ4(void)                                     // IRQ4 函数体
{
    if((*P_INT_Ctrl&C_IRQ4_4KHz)!=0x0000)           // 如果是 4KHz 中断
    {
        *P_INT_Clear = C_IRQ4_4KHz;                 // 清中断标志位
        StepMotor_Drive();                          // 调用步进电机驱动函数
    }
}
```

第 6 步：连接硬件，下载并运行程序。将模组的输出选择跳线跳至“STEP MOTOR”位置，并用排线将模组的 J3 连接到 61 板的 IOB 低 8 位（即 61 板的 J6），如下图所示。



输出选择跳线
配置方法



61板与模组的
连接方法

图 5.5 硬件连接示意图

需要注意的两点是，61 板的“J5”跳线要跳到“5V”状态；模组 J3 接口标号为“VDD”的脚要与 61 板 J6 接口标号为“+”的脚相接，不能接反。

第七步：接通模组上的电机供电电源（选择模组中两个电源接口其中之一，连接直流电源；外接电源的电压范围应在 5V~12V 之间），即完成了硬件连接。

在 IDE 的工具栏中选中绿色的“Use ICE”按钮（如图 5.6 所示），然后按 F8 键对程序进行 Build 和下载。

图 5.6 选择 Use ICE 按钮

程序下载成功后，按 F5 键执行之，观察步进电机面板上的指针转动情况。

5.2 直流电机控制示例

例：通过直流电机接口 2 输出 47/64 占空比的 PWM，控制直流电机正向转动。

第 1 步：打开 unSP IDE，新建工程。这里将工程命名为 DCM，保存在 D 盘根目录下。可参考 5.1 节的第 1 步。

第 2 步：将直流电机驱动程序文件复制到工程文件夹（即“D:\DCM”）。直流电机驱动程序包括 DCMotor.asm、DCMotor.inc 和 DCMotor.h 三个文件。可参考 5.1 节的第 2 步。

第 3 步：把三个驱动程序文件加入工程中。仿照 5.1 节的第 3 步，在 IDE 的 Project 菜单下选择“Add to project”-〉“Files”，在弹出的文件列表中选定工程文件夹下的三个驱动程序文件，添加到工程。

第 4 步：建立主程序文件，编写代码。仿照 5.1 节的第 4 步，在 IDE 下新建 C 程序文件（这里命名为 main.c），写入下列代码：

```
#include "DCMotor.h"                          // 包含直流电机驱动头文件

#define CLR_WDT()    *(unsigned *)0x7012=1    // 清看门狗

main()
{
    DCMotor_Init();                           // 初始化驱动程序
    DCMotor_SetSpeed(MOTOR_2,47);             // 将电机接口 2 的速度等级设置为 47
    DCMotor_Forward(MOTOR_2);                 // 向电机接口 2 发出正转信号
```

```
    while(1)CLR_WDT();                          // 等待中断到来，驱动电机转动
}
```

第 5 步：建立中断服务程序文件。直流电机驱动需要使用 IRQ4_4KHz 中断，因此要建立中断服务程序。中断服务程序可以用 C 语言编写，也可用汇编语言编写。这里使用汇编语言编写。在 IDE 下，选择“File”菜单的“New”，新建一个 asm 文件，可命名为“ISR.asm”。

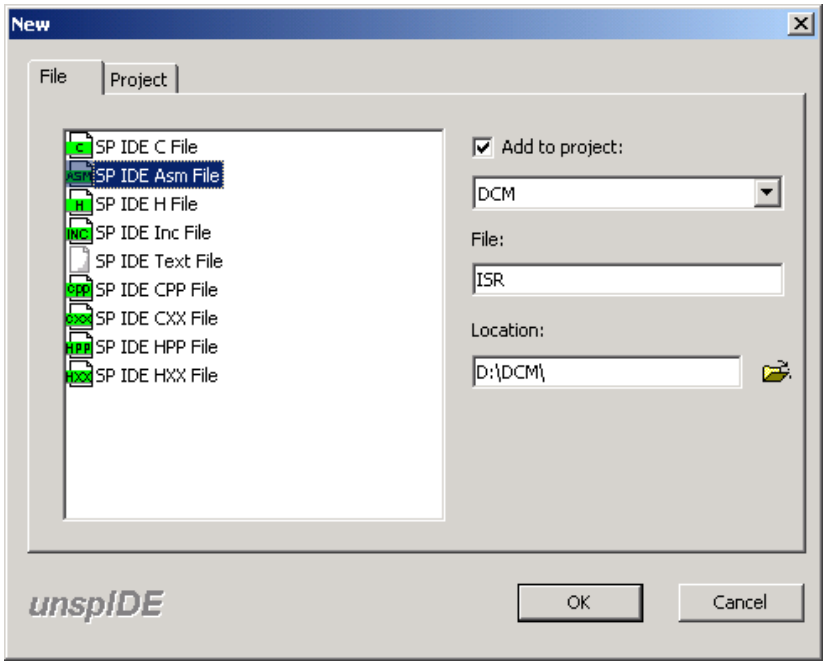


图 5.7 新建汇编语言文件

在 ISR.asm 中编写中断服务程序代码如下：

```
.INCLUDE    DCMotor.inc                     // 包含直流电机驱动头文件

.DEFINE     P_INT_Ctrl      0x7010          // 硬件端口定义
.DEFINE     P_INT_Clear     0x7011          // 硬件端口定义
.DEFINE     C_IRQ4_4KHz     0x0040          // 常数助记符定义

.PUBLIC _IRQ4                               // 将 IRQ4 声明为公有函数

.TEXT
_IRQ4:
    push r1 to [sp]
    r1 = [P_INT_Ctrl]                       // 判断中断源
```

```
    test r1,C_IRQ4_4KHz
    jz    ?Exit                          // 如果不是 4KHz 中断源则直接退出
    r1 = C_IRQ4_4KHz
    [P_INT_Clear] = r1                   // 清中断标志位
    call F_DCMotor_Drive                 // 调用直流电机驱动函数
?Exit:
    pop r1 from [sp]
reti
```

第 6 步：连接硬件，下载运行程序。将模组的输出选择跳线跳至“DC MOTOR”位置，并用排线将模组的 J3 连接到 61 板的 IOB 低 8 位（即 61 板的 J6），如图 5.8 所示，然后接好模组的电机供电电源。可参考 5.1 节的第 6 步。

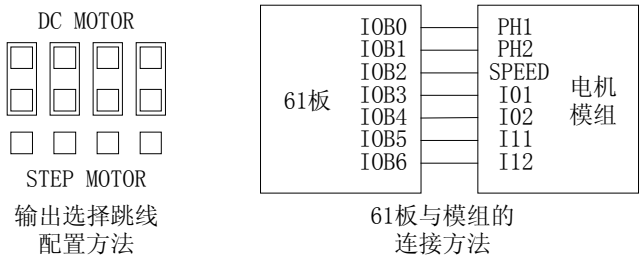


图 5.8 硬件连接示意图

在 IDE 的工具栏中选中绿色的“Use ICE”按钮（如图 5.6 所示），然后按 F8 键对程序进行 Build 和下载。按 F5 键运行程序，观察直流电机的转动。

在上述程序 main()函数最后的循环中，可以调用速度检测函数 DCMotor_GetSpeed()获取当前的电机转速。

5.3 数码管显示程序示例

例：利用 4 位 LED 数码管显示“1234”四个数字。

第 1 步：打开 unSP IDE，新建工程。参考 5.1 节的第 1 步，这里将工程命名为 DigLed，保存在 D 盘根目录下。

第 2 步：将数码管驱动程序复制到工程所在的文件夹下（即“D:\DigLed”）。数码管驱动程序包括 DIG.asm、DIG.inc 和 DIG.h 三个文件。可参考 5.1 节的第 2 步。

第 3 步：参考 5.1 节的第 3 步，把三个驱动程序文件加入工程。

第 4 步；建立主程序文件，编写代码。参考 5.1 节的第 4 步，新建 main.c 程序，编写代码如下：

```
#include "DIG.h"                                    // 数码管显示驱动头文件

#define CLR_WDT()    *(unsigned*)0x7012=1           // 清看门狗

const unsigned DigNum[]={                           // 0~9 十个数字对应的数码管显示编码
      0x3f,0x06,0x5b,0x4f,0x66,
      0x6d,0x7d,0x27,0x7f,0x6f
      };

main()
{
      DIG_Init();                                   // 显示初始化
      DIG_Set(1,DigNum[3]);                         // 第一位显示'3'
      DIG_Set(2,DigNum[4]);                         // 第二位显示'4'
      DIG_Set(3,DigNum[5]);                         // 第三位显示'5'
      DIG_Set(4,DigNum[6]);                         // 第四位显示'6'
      while(1)CLR_WDT();                            // 等待 4KHz 中断来临，实现显示
}
```

第 5 步：建立中断服务程序文件。数码管显示驱动程序用到 IRQ4_4KHz 中断，可以使用汇编语言或 C 语言建立 IRQ4 中断服务程序。这里采用 C 语言，可以直接在 main.c 已有代码的下方继续编写下列代码：

```
#define P_INT_Ctrl      (unsigned int*)0x7010       // 硬件端口定义
#define P_INT_Clear     (unsigned int*)0x7011       // 硬件端口定义
#define C_IRQ4_4KHz  0x0040                         // 常数助记符定义

void IRQ4(void)__attribute__((ISR));                // 将 IRQ4 函数声明为中断服务函数

void IRQ4(void)                                     // IRQ4 函数体
{
      if((*P_INT_Ctrl&C_IRQ4_4KHz)!=0x0000)         // 如果是 4KHz 中断
```

```
    {
        *P_INT_Clear = C_IRQ4_4KHz;          // 清中断标志位
        DIG_Drive();                         // 调用数码管驱动函数
    }
}
```

第 6 步：连接硬件，下载运行程序。将模组的 J1 接口用 10PIN 排线连接到 61 板的 IOA 低 8 位（即 61 板的 J8），模组的 J2 连接到 61 板的 IOA 高 8 位（即 61 板的 J9）。在 IDE 的工具栏中选中绿色的“Use ICE”按钮（如图 5.6 所示），然后按 F8 键对程序进行 Build 和下载。按 F5 键运行程序，观察数码管的显示状态。

6 常见问题解答

Q: 模组接口的“VDD”已经与 61 板的“+”连接了，为何还要外接直流电源？

A: 61 板的“+”负责给 SPGT62C19B 芯片提供逻辑电源，而驱动电机转动的供电电源是与逻辑电源相分离的。这样可以保证 61 板和电机各自供电充足，也降低了电机对逻辑电路部分的干扰。

Q: 可否同时控制模组上的直流电机和步进电机转动？

A: 由于 SPGT62C19B 芯片支持一台步进电机**或者**两台直流电机，因此无法同时控制步进电机和直流电机。

Q: 模组有两组直流电机接口（J11 和 J12），直流电机应插在哪组接口上？

A: 直流电机可以插在任意一组接口上，但应使程序所控制的接口与电机实际使用的接口相一致。

Q: 我的硬件连接和程序都没问题，为什么电机不工作？

A: 首先检查“输出选择跳线”是否配置正确（见 3.1.4 节），其次应保证 61 板的 I/O 口供电电压在 4.5V~5.5V 之间。如果使用电池盒，要保证电池电压充足。建议用 5V 稳压电源为 61 板供电。

Q: 步进电机为什么转得这么慢？

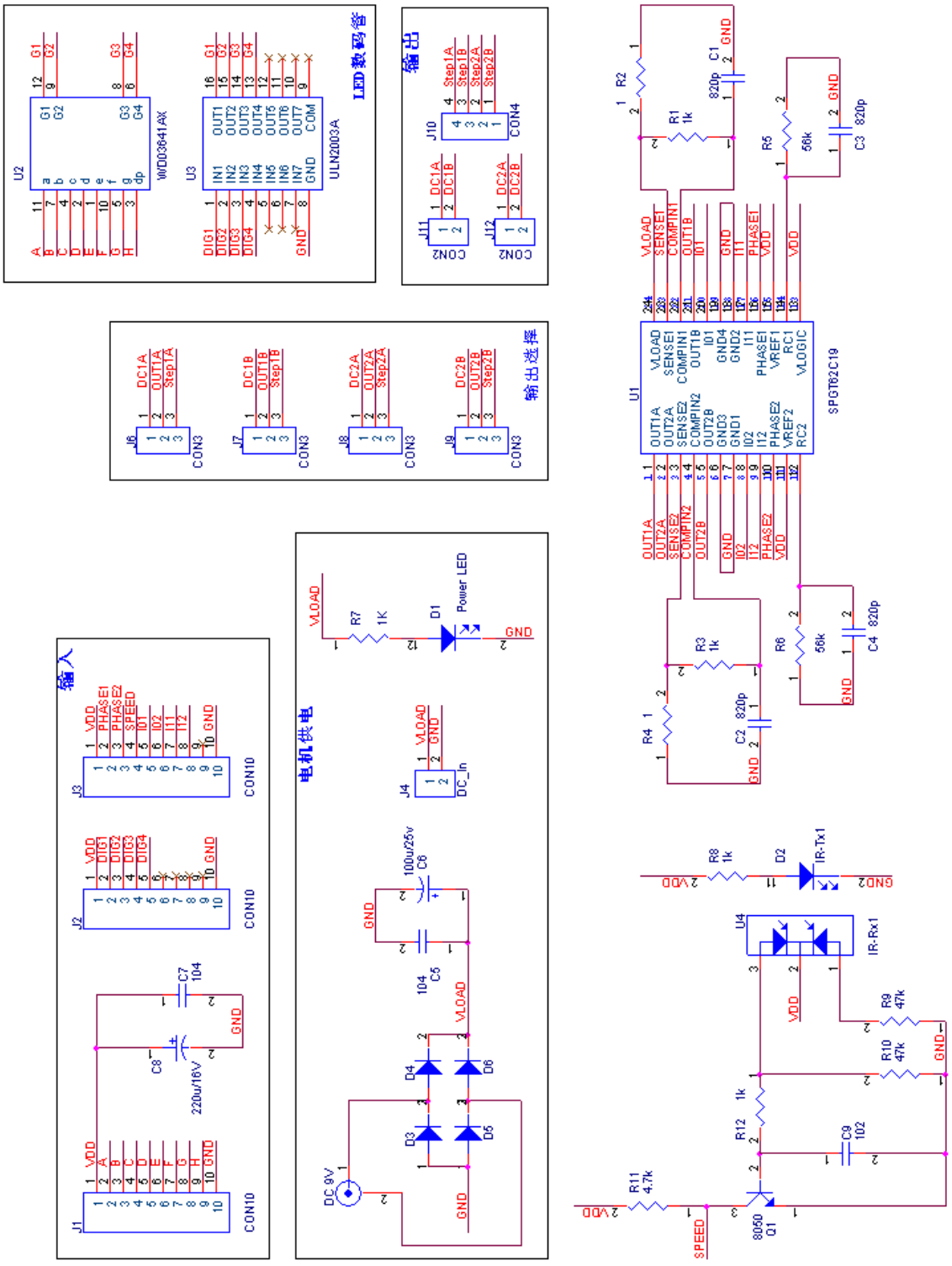
A: 这种步进电机内置了减速齿轮组，将输出转速变换为内部转速的 1/64。这样做的优点是可以增大步进电机的力矩，也可提高步进精度。

Q: 为什么系统刚刚上电复位时直流电机就开始转动了？

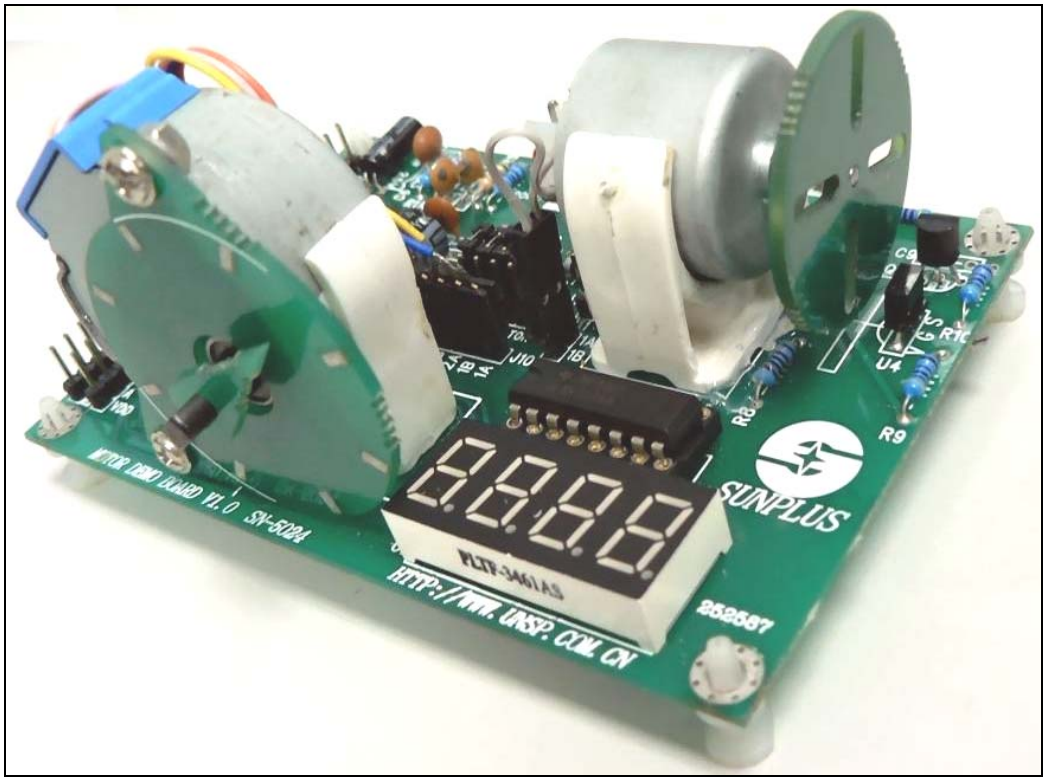
A: SPGT62C19B 的控制逻辑是，当 I01 与 I02 都为 0 时，输出电流最大。而 SPCE061A 上电复位时 I/O 状态为“下拉输入”，表现在端口上的电平状态是 0，因此会使直流电机发生转动。

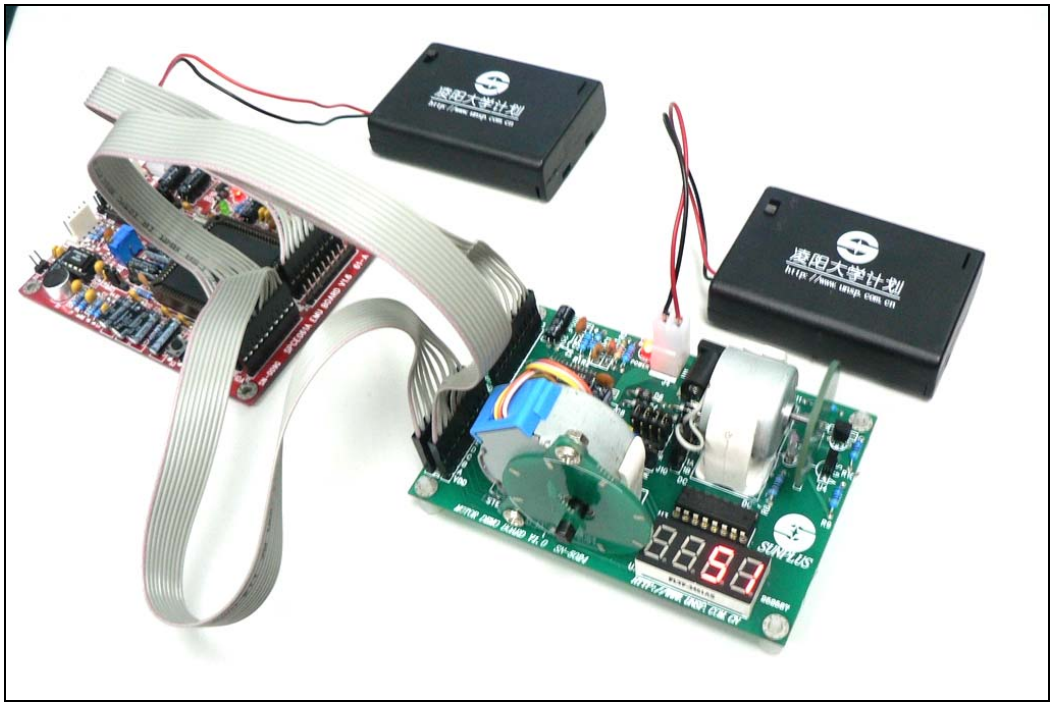
7 附录

7.1 电路原理图


LED数码管
U2
WDD3641AX
U3
ULN2003A
输出
J10
CON4
J11
CON2
J12
CON2
输出选择
J6
J7
J8
J9
CON3
输入
J3
J2
J1
CON10
电机供电
J4
DC_In
DC 9V
Power LED
D1
D2
D3
D4
D5
D6
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
C1
C2
C3
C4
C5
C6
C7
C8
C9
U1
SPGT62C19
U4
IR-Tx1
IR-Rx1
Q1
8050
SPEED
VLOAD
VDD
GND

7.2 模组实物图





7.3 公司联系方式

- 客服专线：**010-62981668-2911**
- 技术支持：**010-62981668-2919**
- 传真号码：**010-62962425**
- 咨询信箱：**unsp@sunplus.com.cn**
- 欢迎登陆：**http://www.unsp.com.cn**
- 技术论坛：**http://www.unsp.com.cn/dvbbs/**
- 邮政编码：1**00085**
- 联系地址：北京市海淀区上地信息产业基地中黎科技园 1 号楼 5 层