目录

项目四 洗衣机模拟装置的制作

项目简介

图 4- 1 常使用的洗衣机

洗衣机是利用电能产生机械作用来洗涤衣物的清洁电器。按其额定洗涤容量分为家用和集体用两类。中国规定洗涤容量在 6 千克以下的属于家用洗衣机：家用洗衣机主要由箱体、洗涤脱水桶（有的洗涤和脱水桶分开）、传动和控制系统等组成，有的还装有加热装置。洗衣机一般专指使用水作为主要的清洗液体，有别于使用特制清洁溶液，及通常由专人负责的干洗。

本项目主要任务：做一个模拟的洗衣机装置。它需要具备以下功能：显示界面→操作按键→电机转动。按照功能把本项目细分为以下任务：

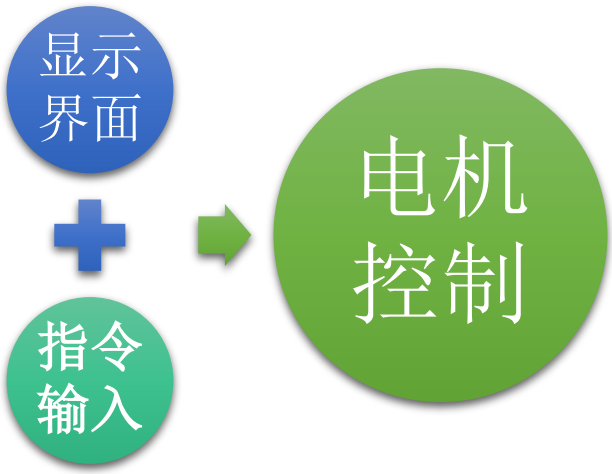

图 4- 2 衣机机模拟构成示意图

任务一　洗衣机模拟装置的硬件制作

一、任务要求

搭建洗衣机模拟装置，首先得有硬件作为基础。在随书附送的 PCB 中，取出洗衣机模拟装置模块。其正面与背面效果如图 4- 3 所示：

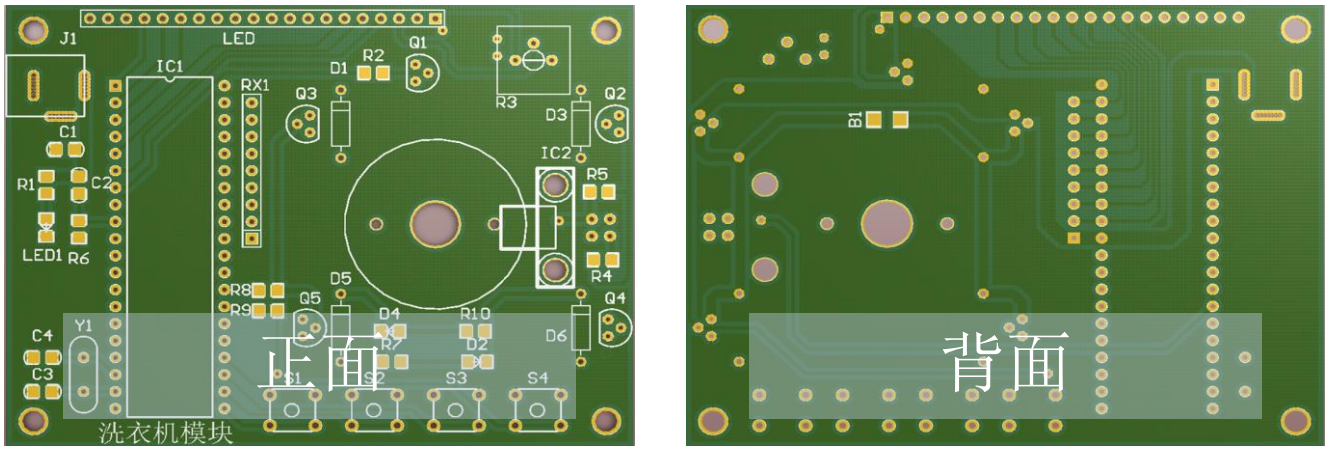

图 4- 3 PCB 正面与背面 PCB 图

本次任务要求为把洗衣机模拟装置电路板焊接完成。需要注意的是：由于我们使用最小系统板进行程序编写与调试，所以 PCB 正面左侧的单片机最小系统可以不焊，只需在 IC1 位置焊上底座就可以了如图 4- 4 所示。若不使用最小系统板，这些元器件需要焊接。

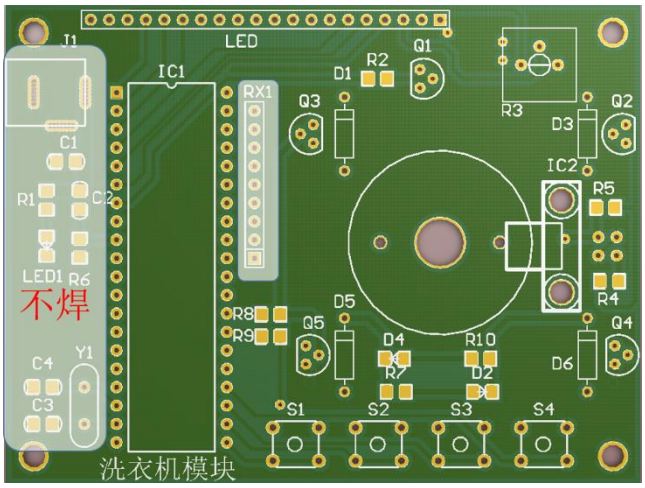

图 4- 4 使用最小系统板不需要装配

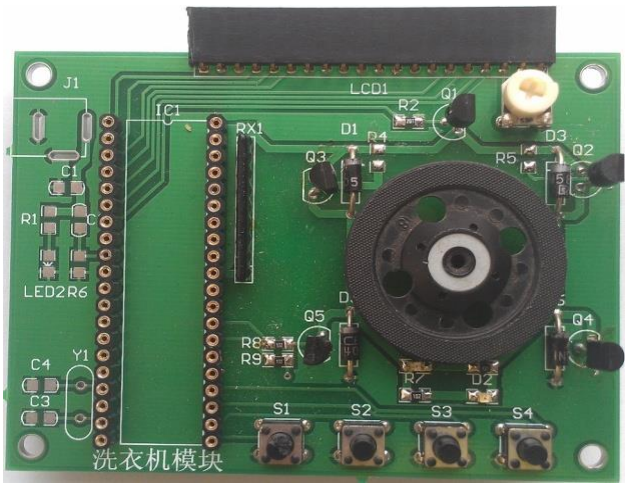

图 4- 5 实物演示板部分

图 4- 5 为实物演示图，此图仅仅作为参考。

本任务需要同学根据电路原理图及装配图装配要求完成实物 PCB 的焊接。

二、相关知识

下面就本次任务需要用到的元器件做一个简单介绍。

1、直流电动机

直流电动机是指能将直流电能转换成机械能（直流电动机）或将机械能转换成直流电能（直流发电机）的旋转电机。它是能实现直流电能和机械能互相转换的电动机。当它作电动机运行时是直流电动机，将电能转换为机械能；作发电机运行时是直流发电机，将机械能转换为电能。

此项目用的电动机如图 4- 6 所示：

图 4- 6 直流电实物图

2、液晶模块 12864

LCD12864 液晶显示模块是一款无字库的点阵型图形显示器，其屏幕由 128（列）×64（行）点阵组成，也称为 12864LCD。它可以显示 16 点阵的 4 行×8 列汉字 32 个，8 点阵的 8 行×8 列字母 64 个或者为 128×64 全屏幕复杂点阵图形。外形如图 4- 7 所示，引脚排如图 4- 8 所示。

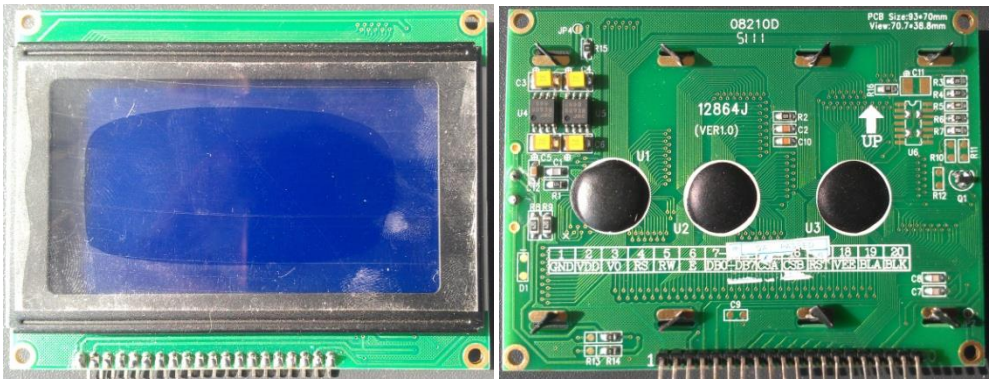

图 4- 7 液晶模块

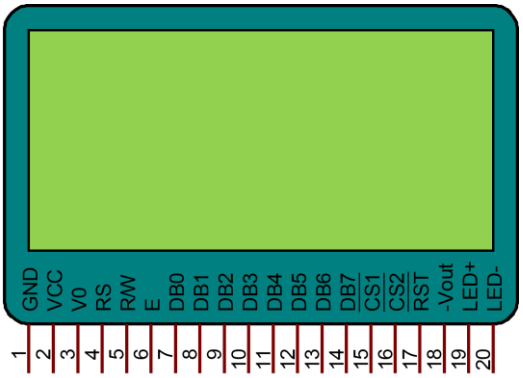

图 4- 8 液晶模块引脚排列图

3、二极管 1N4007

二极管又称晶体二极管,简称二极管(diode) 在半导体二极管内部有一个 PN 结两个引线端子，这种电子器件按照外加电压的方向，具备单向电流的转导性。洗衣机任务使用的是整流二极管 1N4007（图 4- 9）。其中银色一端为二极管的负极，对应 PCB 上的二极管封装的横线。此二极管在本项目中主要起释放继电器线圈电流从而保护电路的作用。

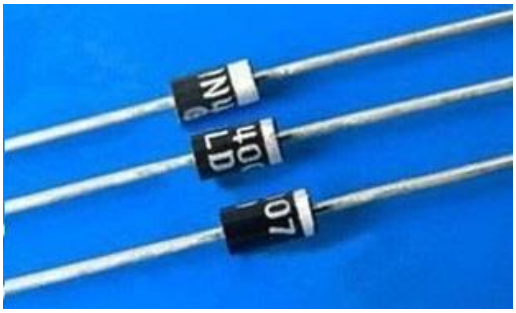

图 4- 9 1N4007 实物图

4、三极管 S8050 和 S8550

半导体三极管又称“晶体三极管”或“晶体管”。在半导体锗或硅的单晶上制备两个能相互影响的 PN 结，组成一个 PNP（或 NPN）结构。中间的 N 区（或 P 区）叫基区，两边的区域叫发射区和集电区，这三部分各有一条电极引线，分别叫基极 B、发射极 E 和集电极 C，是能起放大、振荡或开关等作用的半导体电子器件。这个项目使用两种不同型号的三级管，一个是 NPN 型的三极管 S8050 一个是 PNP 型的三极管 S8550。在这个项目主要的作用是控制电动机的转动。三极管实物如图 4- 10 所示。

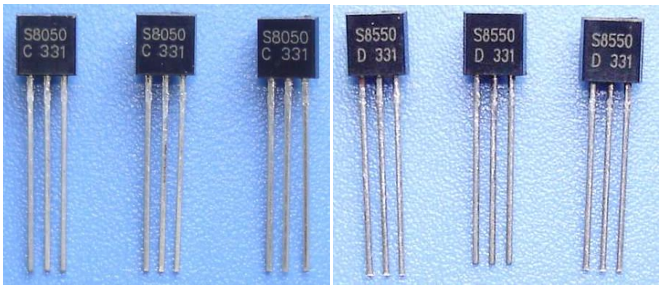

图 4- 10 S8050 和 S8550 实物图

5、光电开关

本项目使用的光电开关为对射式光电开关（图 4- 11）。对射式光电开关由发射器和接收器组成，其工作原理是：通过发射器发出的光线直接进入接收器，当被检测物体经过发射器和接收器之前阻断光线时，光电开关就产生开关信号。与反射式光电开关不同之处在于，前者是通过电→光→电的转换，而后者是通过介质完成。

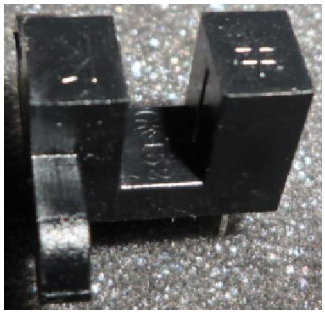

图 4- 11 光电开关实物

三、操作训练

按照电路原理图、元器件清单和 PCB 装配图完成 PCB 板的焊接。

1、装接工艺要求

1.1、贴片电阻、贴片发光二极管水平安装，贴紧印制板。电阻标号方向应一致，发光二极管的正负要注意。

1.2、电位器，二极管，按键开关均采用水平安装，元件贴紧电路板。

1.3、三极管采用直立安装方式，高度要求为管底离印刷板 5mm。

1.4、电动机安装要求先用螺丝把电动机固定在 PCB 板上。然后把电动机线焊接在 PCB 底面的 B1 处焊盘上。

1.5、连接导线的长度要适中，焊接要求符号常规工艺要求:所有插入焊盘孔的元器件引脚及导线均采用直脚焊接，剪脚留头在焊面以上 0.5mm，且不能损伤焊接面。

1.6、自检：对已完成的装配、焊接的工件要求仔细检查质量，符合焊接和装配工艺要求。未述之处按常规工艺要求安装。

2、电路原理图

图 4- 12 是洗衣机模拟装置原理。

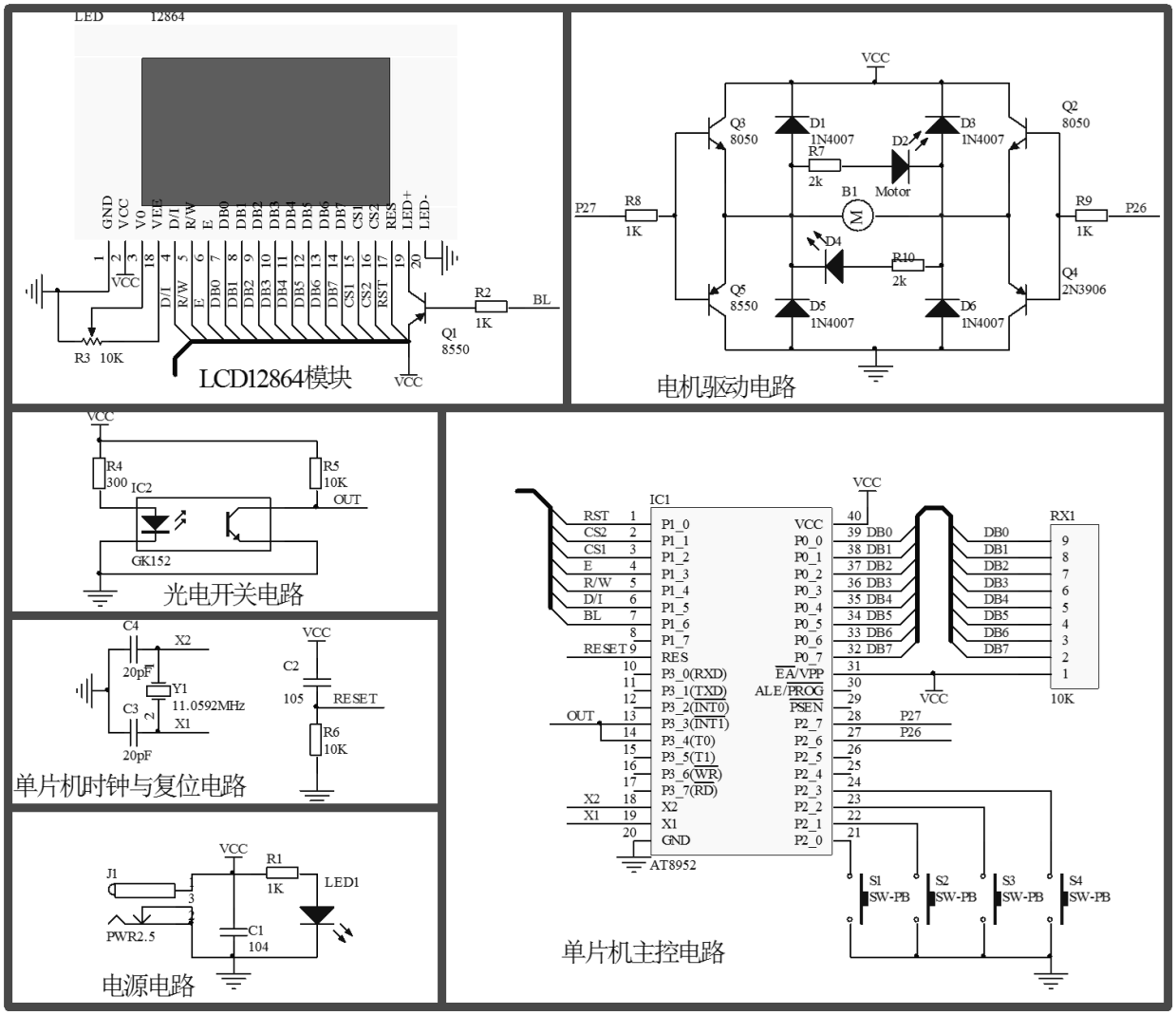

图 4- 12 电路原理图

3、PCB 板装配图

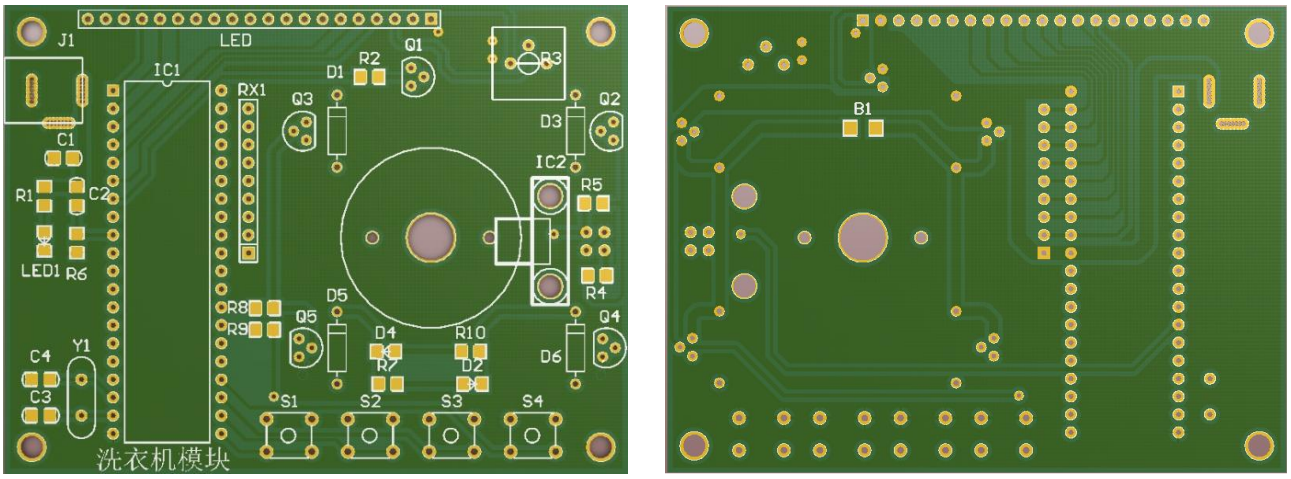

图 4- 13 PCB 装配图

4、元件清单

元器件清单如表 4- 1 所示：

表 4- 1　洗衣机模拟装置需装配元器件清单

序号	元件名	参数	标号	封装	数量	备注
1	电动机	Motor	B1		1	
2	二极管	1N4007	D1, D3, D5, D6	DIODE-0.4	4	
3	发管二极管	红色	D2,	LED0805	1	
4	发管二极管	绿色	D4	LED0805	1	
5	单片机底座	圆孔 IC-40P	IC4	DIP40	1	
6	光电开关	GK152	IC2	光电开关	1	
7	液晶	12864	LED	HDR1X20	1	
8	三极管	8550	Q1, Q5	TO-92M	2	
9	三极管	8050	Q2, Q3	TO-92M	2	
10	三极管	9012	Q4	TO-92M	1	
11	电阻	1K	R2, R8, R9	0805R	3	
12	电位器	10K	R3	3362	1	
13	电阻	300	R4	0805R	1	
14	电阻	10K	R5	0805R	1	
15	电阻	2k	R7, R10	0805R	2	
16	按键	SW-PB	S1, S2, S3, S4	6x6x6	4	

若使用单片机最小系统进行调试，下表中器件不需安装。若直接安装单片机在 PCB 板上，则表 4- 2 中元件必须安装。

表 4- 2　数洗衣机模拟装置非必须装配元器件

序号	元件名	参数	标号	封装	数量	备注
17	电容	104	C1	0805C	1	不焊
18	电容	105	C2	0805C	1	不焊
19	电容	20pF	C3, C4	0805C	2	不焊
20	电阻	1K	R1	0805R	1	不焊
21	电阻	10K	R6	0805R	1	不焊
22	排阻	10K	RX1	SIP-9	1	不焊
23	发光二级管	红色	LED1	LED-0805	1	不焊
24	晶振	11.0592MHz	Y1	X1	1	不焊
25	电源接口	PWR2.5	J1	POWER 接口 1	1	不焊

四、任务评价

表 4- 3　任务一完成情况评价

项目名称			评价时间		年　月　日		
小组名称		小组成员					
评价内容	评价要求	权重	评价标准	学生自评得分	小组评价得分	教师评价得分	合计
职业与安全意识	1. 工具摆放、操作符合安全操作规程；2．遵守纪律、爱惜设备和器材、工位整洁；3. 具有团队协作精神。	10%	好（10） 较好（8） 一般（6） 差（<6）				
印制线路板插件	1、插件正确，电阻标志方向一致。 2、元器件高度符合工艺要求，元器件装插平整。焊点光亮、焊料适量，无虚焊、漏焊、假焊、搭锡、溅锡、铜箔起翘等现象,无毛刺、孔隙留脚长度，焊面以上 0.5mm~1mm。	15%	好（60） 较好（45） 一般（30） 差（<30）				
印制线路板焊接	焊点光亮、焊料适量，无虚焊、漏焊、假焊、搭锡、溅锡、铜箔起翘等现象,无毛刺、孔隙留脚长度，焊面以上 0.5mm~1mm。	60%	好（60） 较好（45） 一般（30） 差（<30）				
印制线路板的总装	装配正确，无烫伤、划伤工件和导线，紧固件装配牢固可靠。	15%	好（15） 较好（12） 一般（9） 差（<6）				
教师签名			学生签名			总分	
项目评价=学生自评（0.2）+小组评价（0.3）+教师评价（0.5）							

五、思考与练习

请写一份装配报告，注明安装中遇到的问题与思考。

任务二　LCD12864 液晶显示界面的制作

一、任务要求

LCD12864 液晶显示界面制作有关说明：

在上面一个项目中我们使用 1602 液晶显示，我们在这个项目中要使用另外一种液晶显示模块 12864，这种液晶是图形点阵型，可以弥补 LCD1602 只能显示 ASCII 码的缺点，它可以显示汉字、ASCII 和图片。

任务要求：

使用 LCD12864 构建一个洗衣机的液晶显示界面。其具体要求如下：

LCD12864 液晶显示模块显示三行字符：

第一行居中显示“欢迎使用洗衣机”；

第二行显示“模式 1：自动”。

第三行显示“模式 2：手动”。

其显示效果如图 4-14。所示。

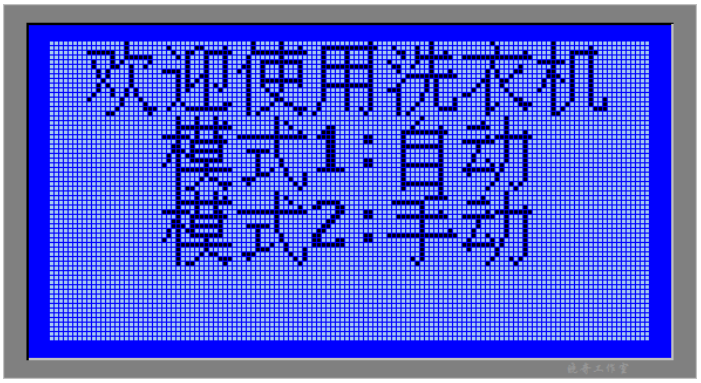

图 4- 14 液晶显示效果

二、相关知识——LCD12864 简介

要完成本项目，首先需要了解 LCD12864 液晶显示屏。

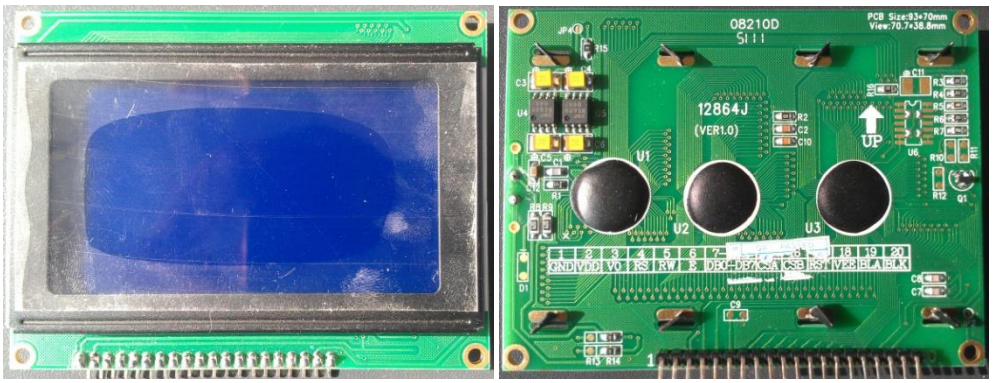

图 4- 15 LCD12864 实物图

LCD12864 液晶显示模块是一款无字库的点阵型图形显示器，其屏幕由 128（列）×64（行）点阵组成，也称为 12864LCD。它可以显示 16 点阵的 4 行×8 列汉字 32 个，8 点阵的 8 行×8 列字母 64 个或者为 128×64 全屏幕复杂点阵图形。

1、LCD12864 模块引脚功能和原理图

1.1 LCD12864 模块引脚功能

如图 4- 16 是 12864 液晶引脚排列图。

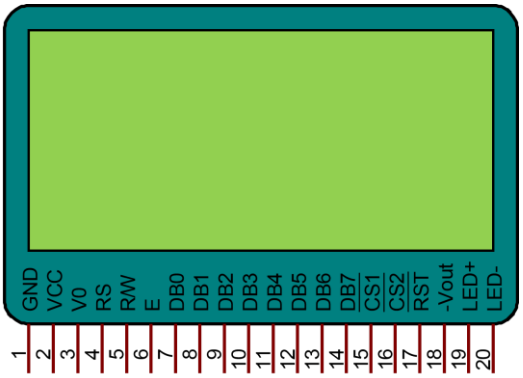

图 4- 16 LCD12864 引脚图

LCD1602 模块 16 个引脚其各引脚功能如表 4- 4 所示。

表 4- 4　LCD12864 模块引脚及功能

序号	符号	电平	描　述
1	VSS	0V	电源地。
2	VDD	+5.0V	模组逻辑供电电压正极。
3	V0	--	液晶显示对比度调节。
4	D/I	H/L	寄存器与显示内存操作选择。1：对寄存器指令操作。0：对数据操作。
5	R/W	H/L	MCU 读写控制器信号。
6	E	H/L	读写使能信号。
7	DB0	H/L	数据总线。
8	DB1	H/L	
9	DB2	H/L	
10	DB3	H/L	
11	DB4	H/L	
12	DB5	H/L	
13	DB6	H/L	
14	DB7	H/L	
15	CS1	H/L	片选信号，高电平有效。
16	CS2	H/L	片选信号，高电平有效。
17	/RST	H/L	复位信号。
18	VEE	--	由内部提供液晶显示驱动电压
19	LED+	+5.0V	LED 背光电源输入正。(+5V)
20	LED-	0V	LED 背光电源输入负。

1.2 LCD12864 模块原理图

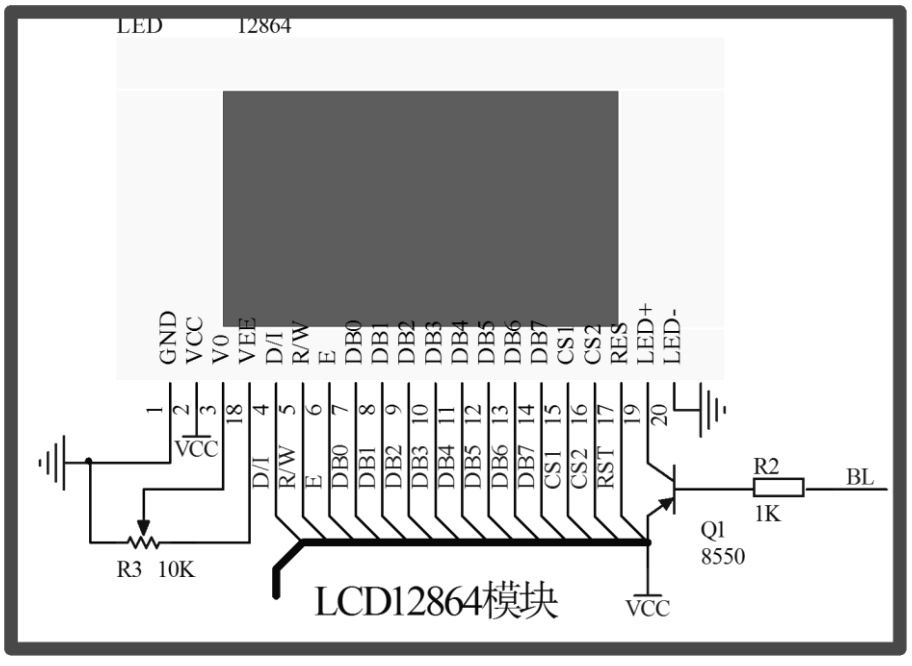

图 4- 17 液晶模块原理图

(1)元件作用

整个液晶模块的元器件包括一个电位器一个三极管和一个电阻，当然还有最重要的液晶模块。下面就上面用到的元件做个功能介绍。

电位器是来调节我们液晶的对比度。当液晶模块显示时字体颜色淡时或则没有显示时可以通过调节它达到显示清楚的效果。

三极管和电阻在这里起到对液晶背光的控制作用。对液晶背光控制可以为减小很多的功耗。和手机一样，当超过一定的时间可以自动关闭液晶背光和显示从而来降低功耗。当给三极管的基极一个低电平三极管导通了液晶模块的背光就点亮了，如果是高电平液晶模块背光就熄灭了。

液晶模块当然是用来显示的。

(2)液晶模块和单片机的连接关系

液晶模块需要和单片机来连接，从而达到用单片机来控制液晶模块。下面来看一下液晶模块和单片机引脚的连接关系。

液晶的数据口 DB0-DB7 连接到单片机 P0.0-P0.7 口上。

液晶的控制线 D/I 连接到单片机 P1.5 口上。

液晶的控制线 R/W 连接到单片机 P1.4 口上。

液晶的控制线 E 连接到单片机 P1.3 口上。

液晶的控制线 CS1 连接到单片机 P1.2 口上。

液晶的控制线 CS2 连接到单片机 P1.1 口上。

液晶的复位引脚 RES 连接到单片机 P1.0 口上。

液晶的背光控制通过三极管连接到单片机 P1.6 口上。

2、LCD12864 模块工作原理

2.1 LCD12864 模块内部结构

LCD12864 点阵液晶图形显示器显示字符或图形的原理与点阵屏显示字符的原理相同。可以将点阵图形显示器的显示屏看出由若干个点构成的点阵，通过控制点阵中不同的点的亮或暗，亮和暗的点阵按一定规律可以组成字符、图形或曲线。实际上在点阵图形显示器内部有对应显示屏幕的显示缓存，单片机与

液晶屏连接后将代表点阵的亮或暗（1 或 0）的信息写入到显示缓存中，由液晶显示器内部的行列驱动自动扫描缓存并刷新驱动显示屏，在液晶显示屏上显示用户期望的信息。所以对使用者来说，对液晶屏写入信息就是对液晶屏内部的缓存写入信息。由 12864 液晶模块的引脚可以看出，数据总线宽度为 8 位，所以写入到显示缓存中的数据是按字节为单位来进行读写的。

下面以显示一个汉字字符“高”来说明液晶显示器显示字符原理。字符点阵如图 4- 18 所示。

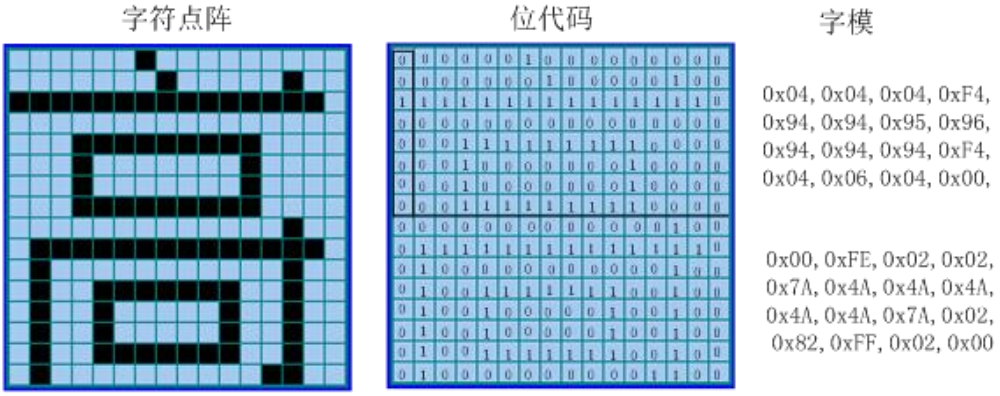

图 4- 18 字“高”的字模

图 4- 18 中汉字“高”的字符大小取 16 行 16 列的点阵来显示。缓存中一个字节为 8 位，使用一位表示一个点（即由 8 个点组成一个字节），这样将点阵信息与字节对应起来，一个 16×16 点阵的汉字需要 32 个字节的编码数据，这些编码数据就反应了 16×16 点阵中亮和暗的控制信息。在图 4- 18 中，将需要点亮显示的点为 1，不显示（暗）的点为 0，这样就得到了“高”字符的每一位代表亮或暗的位代码，将位代码按照一定的规律取成一个个字节即取字模，将送入到液晶显示缓存中的对应位置就可以在液晶显示屏相应的位置显示汉字 “高”了。图 4- 18 中汉字“高”的字模的取模方式是这样的：将 16×16 点阵分成上（8 行）下（8 行）两部分，每一部分 16 列，每一列正好可以取成一个字节，每个字节的高位在下，低位在上，即字模按照“取模：纵向取模，高位在下；数据：从左到右，从上到下”的规律取出的。

2.2 LCD12864 液晶显示器内部 RAM 与屏上显示点的关系

由于 12864 LCD 不带有字库，所以在 12864 上要显示的字符或图形等信息都必须采用取模软件来获得相应的字模的编码数据，然后按照控制要求发送到 12864 的显示缓存中去。

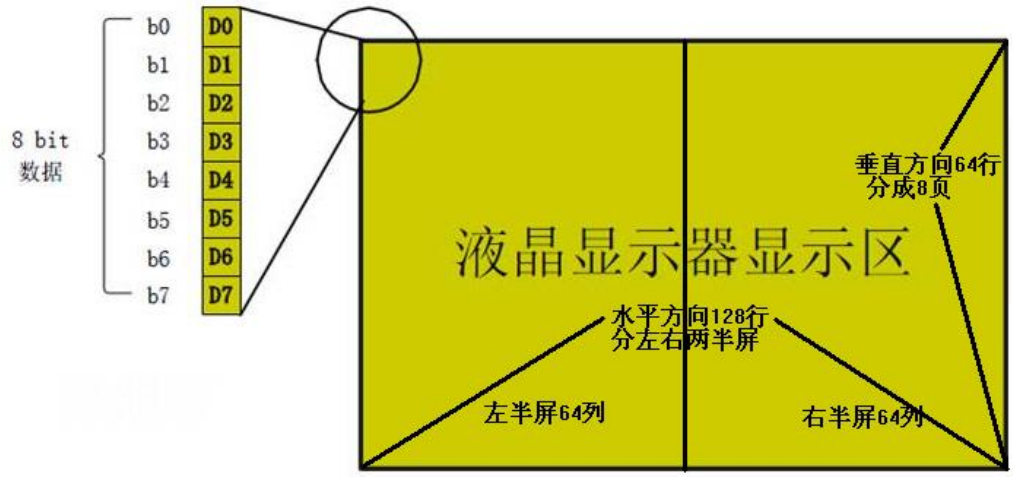

图 4- 19　TG12864 显示 RAM 与屏幕点的对应关系示意图

12864 显示模块显示屏上的点与内部显示 RAM 之间是一一对应的，液晶屏显示区共有 128×64 个点，对应了显示 RAM 的 128×64 个位，显示屏上的每一个点对应了显示 RAM 的一个位。显示 RAM 按照字节为单位划分为 8 个页（page），每一页为 8 行，每一行为 128 个列（其中左半屏为 64 个列，右半屏为 64 个列）。内部驱动芯片的显示 RAM 区对应屏上点的排列方式为“纵向排列，高位在下”。显示 RAM 与屏幕上点的关系示意图如图 4-6 所示，其映射关系如表 4- 5。

表 4- 5　12864 显示 RAM 与屏幕点的映射表

		列 / 行	LCD显示器横坐标（从左往右）									
			左半屏					右半屏				
			0	1	2	……	63	64	……	125	126	127
LCD显示器纵向坐标（从上往下）	第0页	0	D0	D0	D0	……	D0	D0	……	D0	D0	D0
		1	D1	D1	D1	……	D1	D1	……	D1	D1	D1
		2	D2	D2	D2	……	D2	D2	……	D2	D2	D2
		3	D3	D3	D3	……	D3	D3	……	D3	D3	D3
		4	D4	D4	D4	……	D4	D4	……	D4	D4	D4
		5	D5	D5	D5	……	D5	D5	……	D5	D5	D5
		6	D6	D6	D6	……	D6	D6	……	D6	D6	D6
		7	D7	D7	D7	……	D7	D7	……	D7	D7	D7
	第1页	8	D0	D0	D0	……	D0	D0	……	D0	D0	D0
		9	D1	D1	D1	……	D1	D1	……	D1	D1	D1
		……	……	……	……	……	……	……	……	……	……	……
		15	D7	D7	D7	……	D7	D7	……	D7	D7	D7
	……	……	……	……	……	……	……	……	……	……	……	……
	第7页	56	D0	D0	D0	……	D0	D0	……	D0	D0	D0
		……	……	……	……	……	……	……	……	……	……	……
		61	D5	D5	D5	……	D5	D5	……	D5	D5	D5
		62	D6	D6	D6	……	D6	D6	……	D6	D6	D6
		63	D7	D7	D7	……	D7	D7	……	D7	D7	D7

所以要点亮屏上某一点就是对屏的内部对应的 RAM 位置 1，根据此对应关系在程序中控制写入显示 RAM 的数据就控制了屏的显示。注意：市场上不同的

LCD 显示模块的屏幕点与内部显示 RAM 的对应关系不完全相同，必须先了解清楚之后再进行编程。

3、LCD12864 液晶显示模块的指令

表 4- 6　12864 液晶显示模块指令集

命令	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	功能
开关显示	L	L	L	L	H	H	H	H	H	L/H	控制显示开或关，内部状态及显示静态数据无效。 L:关;　H:开
设置 Y 地址	L	L	L	H	Y 地址（0-63）						在 Y 地址计数器中设置 Y 值。
设置 X 地址	L	L	H	L	H	H	H	页（0-7）			在 X 寄存器中设置 X 的值。
显示开始线（Z 地址）	L	L	H	H	显示开始线（0-63）						在显示屏最上层显示数据静态寄存器中的图征。
读取状态	L	H	忙	L	开/关	复位	L	L	L	L	读取状态: 忙　L: 准备 H:在运行中 开/关　L:显示开 H:显示关 复位　L:正常 H:复位
写入显示数据	H	L	写数据								写数据到显示数据静态寄存器，在写指令后，Y 地址自动加 1。
读取显示数据	H	H	读数据								从显示数据静态寄存器中读取数据到数据总线。

12864 液晶显示模块的指令集如表 4- 6 所示。使用时需要注意以下几点：

（1）模块在接收单片机的指令或数据之前必须先确认模块内部是否处于非忙的状态（读取忙标志 DB7 为 0）方可进行指令或数据的传送。如果不进行检查，可以在两次指令或数据操作过程中间插入一段延时。

（2）欲在某个位置显示字符时，首先应设定显示字符的位置，即设定显示地址：开始线（0-63），页地址（0-7）和列地址（0-63），然后再写入要显示字符的字模数据，显示连续字符时只要设定一次地址，模块自动对地址加 1。

12864 液晶显示模块的开始线是由内部的 Z 地址计数器控制的，开始线地址设定范围为 0-63 行范围内任意一行，指令中使用 DB0-DB5 用于选择扫描起始线 0-63 中任意一行。Z 地址计数器具有循环计数功能，用于显示行的扫描同步，当扫描完一行后自动加一。12864 液晶显示模块第 0 行起始线的地址为 0XC0。

12864 液晶显示模块的页地址就是 X 地址，8 行为 1 页，12864 中共有 64 行，即 8 页，指令用 DB2-DB0 用于选择第 0-7 页。页地址存储在 X 地址计数器中，读写数据对页地址没有影响，复位后页地址计数器内容为 0。12864 液晶显示模块第 0 页地址为 0XB8。

12864 液晶显示模块的列地址就是 Y 地址，列地址存储在 Y 地址计数器中。由于该模块分成左右两半屏（由 CS1 与 CS2 来选择），故 Y 地址范围为 0-63。指令使用 DB0-DB5 设置列地址。当单片机对液晶模块内部 RAM 中的数据进行读或写操作时，y 地址计数器会加 1，指向下一个地址。12864 液晶显示模块第 0 列地址为 0X40。

（3）显示开关时，不影响显示 RAM 中的内容。

（4）对 1264 读/写操作的时序如图 4- 21 和图 4- 20 所示。

4、LCD12864 液晶子程序的编写

4.1 写操作子程序的编写

写操作子程序分为两个部分：写数据和写命令。

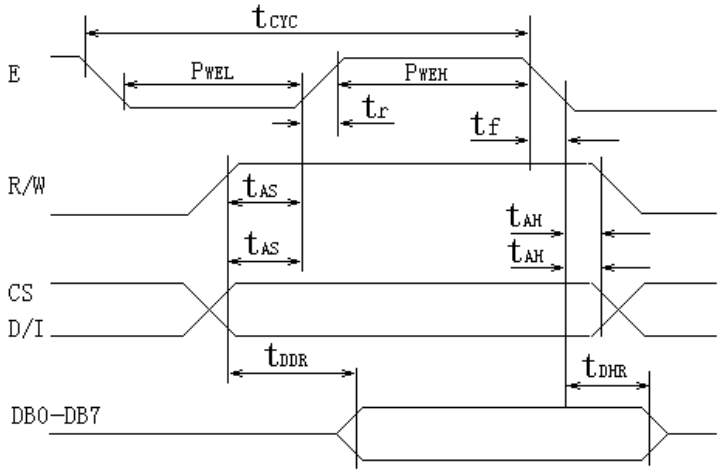

图 4- 20　12864 写操作时序

(1)写命令子程序的编写

LCD12864 写时序如图 4- 20 所示，可以参考其时序图和原理图写出 LCD12864 的写数据子程序：

LCD12864 写数据子程序

```
sbit E=P1^2;                    // E 为锁存使能,下降沿锁存数据
sbit RS=P1^0;                   // 数据/指令选择，0 为数据，1 为指令
sbit RW=P1^1;                   // 读/写选择，0 为写，1 为读
#define   DATAPORT     P0       // P0 端口为 LCD 数据总线
/******写数据到 LCD 函数 参数：dat 数据******/
void write_data(unsigned char dat)          //写数据
{
    check_busy_12864()
    RW=0;          //写操作
    RS=1;          //写数据
    DATAPORT=dat;     //P0 口传输数据
    E=1;
    E=0;       //下降沿(使能)，将数据送入
    RS=0;          //置为写命令状态，减少噪点发生率
}
```

(2)写命令子程序的编写

LCD12864 写时序如图 4- 20 所示，可以参考其时序图和原理图写出 LCD12864 的写命令子程序：

LCD12864 的写命令子程序：

LCD12864 写命令子程序

```
sbit E=P1^2;                    // E 为锁存使能,下降沿锁存数据
sbit RS=P1^0;                   // 数据/指令选择，0 为数据，1 为指令
sbit RW=P1^1;                   // 读/写选择，0 为写，1 为读
#define   DATAPORT     P0       // P0 端口为 LCD 数据总线
/******写命令到 LCD 函数 参数：cmd 命令******/
void write_cmd(unsigned char dat)           //写命令
{
    check_busy_12864()
    RW=0;          //写操作
    RS=0;          //写命令
    DATAPORT=dat;     //P0 口传输命令
    E=1;
```

```
    E=0;      //下降沿(使能)，将命令送入
}
```

4.2 读操作子程序的编写

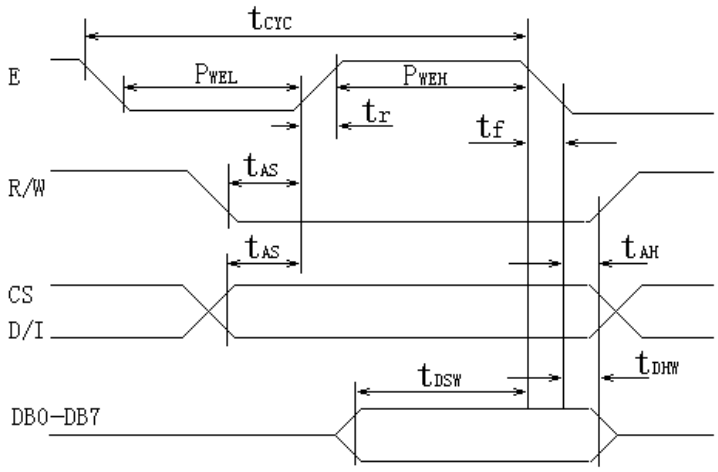

图 4- 21　12864 读操作时序

LCD12864 读时序如图 4- 21 所示，可以参考其时序图和原理图写出 LCD12864 的读操作子程序,不过，由于 LCD12864 显示的字符较少，所以编程中对 LCD12864 的读取基本上只限于需要判忙时，基本上不会对写入的数据进行读取。其判忙程序如下所示：

LCD12864 读操作子程序

```
sbit E=P1^2;                    // E 为锁存使能,下降沿锁存数据
sbit RS=P1^0;                   //  数据/指令选择，0 为数据，1 为指令
sbit RW=P1^1;                   //  读/写选择，0 为写，1 为读
#define   DATAPORT     P0       // P0 端口为 LCD 数据总线
/******忙检测函数******/
void check_busy_12864()
{
        uchar i=0,dat;                  //定义一个临时变量
        RS=0;                           //对指令操作
        RW=1;                           //读操作
        do{                             //先执行下面指令
          DATAPORT=0x00;                //口线置 1 防止干扰
          E=1;                          //下降沿的高电平部分
          dat=DATAPORT;                     //读 LCD 总线上的数据到临时变量
          E=1;                          //短暂延时
```

```
        E=0;
        dat=0x80&dat;          //取出数据的 D7 位，该位是忙检测(busy)信号
        i++;
      }while((dat==0x80)&&(i<100)); //LCD 内部不忙或超时，退出
}
```

4.3 LCD12864 驱动参考头文件

根据上述指令与控制字表格，可以编写 LCD12864 头文件。通过把一些基本参数与硬件接口描述写入头文件中为以后的程序移植打下基础。

采用端口方式编写的驱动参考程序（LCD.H）如下：

LCD12864 驱动头文件库程序 LCD.H

```
#include<reg51.h>                  // 包含 51 单片机头文件
/******12864LCD 引脚定义******/
#define DATAPORT   P0              // P0 端口为 LCD 数据总线
sbit CS1=P1^3;                     // 位定义 P2.0，CS1 为左半屏片选信号
sbit CS2=P1^4;                     // 位定义 P2.1，CS2 为右半屏片选信号
sbit E=P1^2;                       // E 为锁存使能,下降沿锁存数据
sbit RS=P1^0;                      // 数据/指令选择，0 为数据，1 为指令
sbit RW=P1^1;                      // 读/写选择，0 为写，1 为读
sbit RST=P1^5;                     // TG12864 复位信号
sbit BL=P1^6;                      // 背光选择
/******毫秒级延时函数******/
void delay_ms(unsigned int ms)
{
  unsigned char i;                      //临时延时变量 i
  while(--ms)                    //减 1 判断是否为 0
  {
     for(i=0;i<125;i++);       //加 1 判断是否达到 125
  }
}
void selectscreen(unsigned char screen)//选屏 0:全屏 1:左半屏 2:右半屏
{
    switch (screen)
    {
        case 0:CS1=1;CS2=1;    break;        //全屏
        case 1:CS1=1;CS2=0;    break;        //左半屏
```

```
            case 2:CS1=0;CS2=1;     break;          //右半屏
            default:   break;
        }
}
void write_cmd(unsigned char dat)           //写命令
{
    RW=0;           //写操作
    RS=0;           //写命令
    DATAPORT=dat;     //P0 口传输命令
    E=1;
    E=0;       //下降沿(使能)，将命令送入
}
void write_data(unsigned char dat)          //写数据
{
    RW=0;           //写操作
    RS=1;           //写数据
    DATAPORT=dat;     //P0 口传输数据
    E=1;
    E=0;       //下降沿(使能)，将数据送入
    RS=0;           //置为写命令状态，减少噪点发生率
}
void check_busy_12864()//忙检测函数
{
            uchar i=0,dat;                  //定义一个临时变量
            RS=0;                               //对指令操作
            RW=1;                               //读操作
            do{                                 //先执行下面指令
               DATAPORT=0x00;                   //口线置 1 防止干扰
               E=1;                             //下降沿的高电平部分
               dat=DATAPORT;                    //读 LCD 总线上的数据到临时变量
               E=1;                             //短暂延时
               E=0;
               dat=0x80&dat;        //取出数据的 D7 位，该位是忙检测(busy)信号
               i++;
          }while((dat==0x80)&&(i<100));//LCD 内部不忙或超时，退出
}
void set_onoff(unsigned char onoff)         //开关
```

```
{
    onoff|=0x3e;        //控制字
    write_cmd(onoff);  //写命令,将控制字写入
}

void set_page(unsigned char page)           //选行
{
    page&=0x07;             //限位
    page|=0xb8;             //控制字
    write_cmd(page);   //写命令,将控制字写入
}
void set_column(unsigned char column)     //选列
{
    if(column>63)selectscreen(2);     //当列<63 时，则选择左半屏
    else selectscreen(1);          //当列超过左半屏时，则选择右半屏
    column&=0x3f;           //限位
    column|=0x40;           //控制字
    write_cmd(column);     //写命令,将控制字写入
}
void set_startline(unsigned char line)   //选起始行
{
    line&=0x3f;             //限位
    line|=0xc0;             //控制字
    write_cmd(line);    //写命令,将控制字写入
}

void clearnscreen() //清屏
{
    unsigned char i,j;
    selectscreen(0);     //选屏
    for (i=0;i<8;i++)
    {
        set_page(i);          //选行
        for (j=0;j<64;j++)    //列会自动加 1
        {
            write_data(0); //写数据
        }
```

```
    }
}
void LCD_INIT()//LCD 初始化
{
        BL=0;                   //背光
        RST=0;                  //复位
        delay_ms(10);           //延时等待
        RST=1;                  //置位
        delay_ms(10);           //延时等待
        set_onoff(1);           //开关
        set_startline(0);       //起始行
        clearnscreen();         //清屏
}
```

把上述程序保存在 LCD.H 这个头文件中。在今后若需要使用 LCD12864 时只需调用此头文件就可以了。

三、操作训练

1、任务分析

硬件完成了，加上前面的知识储备，我们就可以开始完成 LCD12864 液晶显示屏的制作了。本任务要求显示如图 4- 14 所示效果。根据前面的基础知识已经学习了 LCD12864 的初始化及其控制字的配置。要在 LCD1 上显示字符，还需要两个程序模块，1，写汉字的子程序。2、写 ASCII 的子程序。

1.1 写入汉字程序

本程序需要控制写入的汉字显示在 LCD12864 的第几行，第几列。首先要选择屏幕是左半屏还是右半屏，然后选择页，最后选择列，把所需要的数据写入进去。其参考程序如下：

LCD1284 写入汉字子程序

```
//写一个汉字
void write_HZ(unsigned char page,unsigned char column,unsigned char *p)
{
        unsigned char i;
        for (i=0;i<16;i++)
        {
           set_column(column+i);//选列
           set_page(page);          //选行
           write_data(*p);     //输入数据
           p++; //指针加 1
        }
        for (i=0;i<16;i++)
        {
           set_column(column+i);//选列
           set_page(page+1); //选行(行加 1，写下半个字的数据)
           write_data(*p);     //输入数据
           p++; //指针加 1
        }
}
```

1.2 写入 ASCII 程序

写 ASIIC 码和写汉字程序基本一样。只是显示的列数不一样。其他都是一样的。

LCD12864 写入 ASCII 码子程序

```
void write_ASC(unsigned char page,unsigned char column,unsigned char *p)   // 写一个 ASC 码
{
        unsigned char i;
        for (i=0;i<8;i++)
        {
           set_column(column+i);//选列
           set_page(page);          //选行
           write_data(*p);     //输入数据
           p++; //指针加 1
        }
        for (i=0;i<8;i++)
        {
           set_column(column+i);//选列
           set_page(page+1);  //选行(行加 1，写下半个 ASC 码的数据)
           write_data(*p);     //输入数据
           p++; //指针加 1
        }
}
```

2、主函数流程图

主函数流程图如图 4- 22 所示。

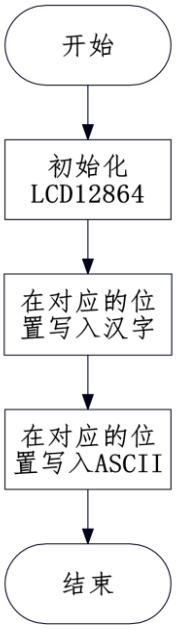

图 4- 22 LCD12864 液晶显示界面主程序流程图

3、参考程序

根据图 4- 22 LCD12864 液晶显示界面主程序流程图编写程序：

LCD12864 液晶显示界面程序 LCD12864.c

```
#include<reg51.h>                    // 包含 51 单片机头文件
#include<lcd.h>                      // 包含液晶子程序头文件
unsigned char code HZ[][32]=
{
/*--  文字:  欢  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x14,0x24,0x44,0x84,0x64,0x1C,0x20,0x18,0x0F,0xE8,0x08,0x08,0x28,0x18,0x08,0x00,
0x20,0x10,0x4C,0x43,0x43,0x2C,0x20,0x10,0x0C,0x03,0x06,0x18,0x30,0x60,0x20,0x00,
/*--  文字:  迎  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x40,0x41,0xCE,0x04,0x00,0xFC,0x04,0x02,0x02,0xFC,0x04,0x04,0x04,0xFC,0x00,0x00,
0x40,0x20,0x1F,0x20,0x40,0x47,0x42,0x41,0x40,0x5F,0x40,0x42,0x44,0x43,0x40,0x00,
/*--  文字:  使  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x40,0x20,0xF0,0x1C,0x07,0xF2,0x94,0x94,0x94,0xFF,0x94,0x94,0x94,0xF4,0x04,0x00,
0x00,0x00,0x7F,0x00,0x40,0x41,0x22,0x14,0x0C,0x13,0x10,0x30,0x20,0x61,0x20,0x00,
/*--  文字:  用  --*/
```

```
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x00,0x00,0xFE,0x22,0x22,0x22,0x22,0xFE,0x22,0x22,0x22,0x22,0xFE,0x00,0x00,
0x80,0x40,0x30,0x0F,0x02,0x02,0x02,0x02,0xFF,0x02,0x02,0x42,0x82,0x7F,0x00,0x00,
/*--  文字:  洗  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x10,0x61,0x06,0xF0,0xA0,0x98,0x8E,0x88,0x88,0xFF,0x88,0x88,0x88,0x80,0x80,0x00,
0x04,0x04,0xFF,0x00,0x40,0x20,0x18,0x07,0x00,0x00,0x3F,0x40,0x40,0x40,0x70,0x00,
/*--  文字:  衣  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x04,0x04,0x04,0x04,0x84,0x44,0x25,0x3E,0xC4,0x04,0x04,0x84,0x64,0x44,0x04,0x00,
0x08,0x04,0x02,0x41,0xFF,0x40,0x20,0x00,0x00,0x03,0x0D,0x10,0x20,0x60,0x20,0x00,
/*--  文字:  机  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x08,0x08,0xC8,0xFF,0x48,0x88,0x08,0x00,0xFE,0x02,0x02,0x02,0xFE,0x00,0x00,0x00,
0x04,0x03,0x00,0xFF,0x00,0x41,0x30,0x0C,0x03,0x00,0x00,0x00,0x3F,0x40,0x78,0x00,
/*--  文字:  模  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x10,0xD0,0xFF,0x50,0x90,0x04,0xF4,0x54,0x5F,0x54,0x54,0x5F,0xF4,0x04,0x00,0x00,
0x03,0x00,0xFF,0x00,0x00,0x84,0x85,0x45,0x35,0x0F,0x15,0x25,0x65,0xC4,0x44,0x00,
/*--  文字:  式  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x08,0x88,0x88,0x88,0x88,0x88,0x08,0xFF,0x08,0x09,0x0E,0x0A,0x08,0x00,0x00,
0x00,0x20,0x60,0x30,0x1F,0x10,0x08,0x08,0x00,0x07,0x18,0x20,0x40,0x80,0x70,0x00,
/*--  文字:  自  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x00,0x00,0xF8,0x48,0x48,0x4C,0x4B,0x4A,0x48,0x48,0x48,0xF8,0x00,0x00,0x00,
0x00,0x00,0x00,0xFF,0x44,0x44,0x44,0x44,0x44,0x44,0x44,0x44,0xFF,0x00,0x00,0x00,
/*--  文字:  动  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x20,0x24,0x24,0xE4,0x24,0x24,0x24,0x20,0x10,0x10,0xFF,0x10,0x10,0xF0,0x00,0x00,
0x08,0x1C,0x0B,0x08,0x0C,0x05,0x4E,0x24,0x10,0x0C,0x03,0x20,0x40,0x3F,0x00,0x00,
/*--  文字:  手  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x24,0x24,0x24,0x24,0x24,0x24,0xFE,0x22,0x22,0x22,0x22,0x22,0x20,0x00,0x00,
0x02,0x02,0x02,0x02,0x02,0x42,0x82,0x7F,0x02,0x02,0x02,0x02,0x02,0x02,0x02,0x00,
};
unsigned char code ASC[11][16]=
```

```
{
    0xF8,0xFC,0x04,0xC4,0x24,0xFC,0xF8,0x00,    // -0-
    0x07,0x0F,0x09,0x08,0x08,0x0F,0x07,0x00,
    0x00,0x10,0x18,0xFC,0xFC,0x00,0x00,0x00,    // -1-
    0x00,0x08,0x08,0x0F,0x0F,0x08,0x08,0x00,
    0x08,0x0C,0x84,0xC4,0x64,0x3C,0x18,0x00,    // -2-
    0x0E,0x0F,0x09,0x08,0x08,0x0C,0x0C,0x00,
    0x08,0x0C,0x44,0x44,0x44,0xFC,0xB8,0x00,    // -3-
    0x04,0x0C,0x08,0x08,0x08,0x0F,0x07,0x00,
    0xC0,0xE0,0xB0,0x98,0xFC,0xFC,0x80,0x00,    // -4-
    0x00,0x00,0x00,0x08,0x0F,0x0F,0x08,0x00,
    0x7C,0x7C,0x44,0x44,0xC4,0xC4,0x84,0x00,    // -5-
    0x04,0x0C,0x08,0x08,0x08,0x0F,0x07,0x00,
    0xF0,0xF8,0x4C,0x44,0x44,0xC0,0x80,0x00,    // -6-
    0x07,0x0F,0x08,0x08,0x08,0x0F,0x07,0x00,
    0x0C,0x0C,0x04,0x84,0xC4,0x7C,0x3C,0x00,    // -7-
    0x00,0x00,0x0F,0x0F,0x00,0x00,0x00,0x00,
    0xB8,0xFC,0x44,0x44,0x44,0xFC,0xB8,0x00,    // -8-
    0x07,0x0F,0x08,0x08,0x08,0x0F,0x07,0x00,
    0x38,0x7C,0x44,0x44,0x44,0xFC,0xF8,0x00,    // -9-
    0x00,0x08,0x08,0x08,0x0C,0x07,0x03,0x00,
    0x00,0x00,0x00,0x30,0x30,0x00,0x00,0x00,    // -:-
    0x00,0x00,0x00,0x06,0x06,0x00,0x00,0x00,
};
void main(void)//主函数
{
    LCD_INIT();//液晶初始化
    write_HZ(0,16*0+8,&HZ[0]);//欢
    write_HZ(0,16*1+8,&HZ[1]);//迎
    write_HZ(0,16*2+8,&HZ[2]);//使
    write_HZ(0,16*3+8,&HZ[3]);//用
    write_HZ(0,16*4+8,&HZ[4]);//洗
    write_HZ(0,16*5+8,&HZ[5]);//衣
    write_HZ(0,16*6+8,&HZ[6]);//机

    write_HZ(2,16*1+8,&HZ[7]);//模
    write_HZ(2,16*2+8,&HZ[8]);//式
```

```
    write_ASC(2,16*3+8,&ASC[1]);//1
    write_ASC(2,16*4,&ASC[10]);//:
    write_HZ(2,16*4+8,&HZ[9]);//自
    write_HZ(2,16*5+8,&HZ[10]);//动

    write_HZ(4,16*1+8,&HZ[7]);//模
    write_HZ(4,16*2+8,&HZ[8]);//式
    write_ASC(4,16*3+8,&ASC[2]);//2
    write_ASC(4,16*4,&ASC[10]);//:
    write_HZ(4,16*4+8,&HZ[11]);//自
    write_HZ(4,16*5+8,&HZ[10]);//动
    while (1)
    {
    }
}
```

LCD12864 驱动头文件库程序

LCD.H

```
#include<reg51.h>                    // 包含 51 单片机头文件
/******12864LCD 引脚定义******/
#define DATAPORT   P0                // P0 端口为 LCD 数据总线
sbit CS1=P1^3;                       // 位定义 P2.0，CS1 为左半屏片选信号
sbit CS2=P1^4;                       // 位定义 P2.1，CS2 为右半屏片选信号
sbit E=P1^2;                         // E 为锁存使能,下降沿锁存数据
sbit RS=P1^0;                        // 数据/指令选择，0 为数据，1 为指令
sbit RW=P1^1;                        // 读/写选择，0 为写，1 为读
sbit RST=P1^5;                       // TG12864 复位信号
sbit BL=P1^6;                        // 背光选择
/******毫秒级延时函数******/
void delay_ms(unsigned int ms)
{
   unsigned char i;                        //临时延时变量 i
   while(--ms)                         //减 1 判断是否为 0
   {
       for(i=0;i<125;i++);        //加 1 判断是否达到 125
   }
}
void selectscreen(unsigned char screen)//选屏 0:全屏 1:左半屏 2:右半屏
```

```
{
    switch (screen)
    {
        case 0:CS1=1;CS2=1;    break;        //全屏
        case 1:CS1=1;CS2=0;    break;        //左半屏
        case 2:CS1=0;CS2=1;    break;        //右半屏
        default:   break;
    }
}
void write_cmd(unsigned char dat)          //写命令
{
    RW=0;          //写操作
    RS=0;          //写命令
    DATAPORT=dat;     //P0 口传输命令
    E=1;
    E=0;       //下降沿(使能)，将命令送入
}
void write_data(unsigned char dat)          //写数据
{
    RW=0;          //写操作
    RS=1;          //写数据
    DATAPORT=dat;     //P0 口传输数据
    E=1;
    E=0;       //下降沿(使能)，将数据送入
    RS=0;          //置为写命令状态，减少噪点发生率
}
void check_busy_12864()//忙检测函数
{
        uchar i=0,dat;                  //定义一个临时变量
        RS=0;                           //对指令操作
        RW=1;                           //读操作
        do{                             //先执行下面指令
          DATAPORT=0x00;                //口线置 1 防止干扰
          E=1;                          //下降沿的高电平部分
          dat=DATAPORT;                 //读 LCD 总线上的数据到临时变量
          E=1;                          //短暂延时
          E=0;
```

```
            dat=0x80&dat;          //取出数据的 D7 位，该位是忙检测(busy)信号
            i++;
        }while((dat==0x80)&&(i<100));//LCD 内部不忙或超时，退出
}
void set_onoff(unsigned char onoff)          //开关
{
    onoff|=0x3e;        //控制字
    write_cmd(onoff);  //写命令,将控制字写入
}

void set_page(unsigned char page)            //选行
{
    page&=0x07;              //限位
    page|=0xb8;              //控制字
    write_cmd(page);   //写命令,将控制字写入
}
void set_column(unsigned char column)      //选列
{
    if(column>63)selectscreen(2);      //当列<63 时，则选择左半屏
    else selectscreen(1);              //当列超过左半屏时，则选择右半屏
    column&=0x3f;            //限位
    column|=0x40;            //控制字
    write_cmd(column);      //写命令,将控制字写入
}
void set_startline(unsigned char line)   //选起始行
{
    line&=0x3f;              //限位
    line|=0xc0;              //控制字
    write_cmd(line);    //写命令,将控制字写入
}

void clearnscreen() //清屏
{
    unsigned char i,j;
    selectscreen(0);      //选屏
    for (i=0;i<8;i++)
    {
```

```
            set_page(i);          //选行
            for (j=0;j<64;j++)    //列会自动加 1
            {
                  write_data(0); //写数据
            }
      }
}
void LCD_INIT()//LCD 初始化
{
        BL=0;                   //背光
        RST=0;                  //复位
        delay_ms(10);           //延时等待
        RST=1;                  //置位
        delay_ms(10);           //延时等待
        set_onoff(1);           //开关
        set_startline(0);       //起始行
        clearnscreen();         //清屏
}
```

4、程序说明

本程序首先调用初始化程序 LCD_INIT()对液晶进行初始化。包括液晶的背光开启，复位液晶，以及设置液晶的初始列和清屏。完成了液晶的初始化之后只需要调用 write_HZ 和 write_ASC 在液晶屏相应的位置写入汉字和 ASCII 码。

5、任务实施

5.1 建立工程

打开 keil 软件，通过菜单“Project”，新建立一个工程”new Project”项目 LCD12864，然后再建一个文件名为 LCD12864.C 的源程序文件，将上面的参考程序输入并保存。建立的 LCD12864.C 文件添加入本项目中。

5.2 编译并生成 HEX

单击”Build target”按钮，对源程序进行编译和链接，产生 HEX 文件。

5.3 烧录芯片

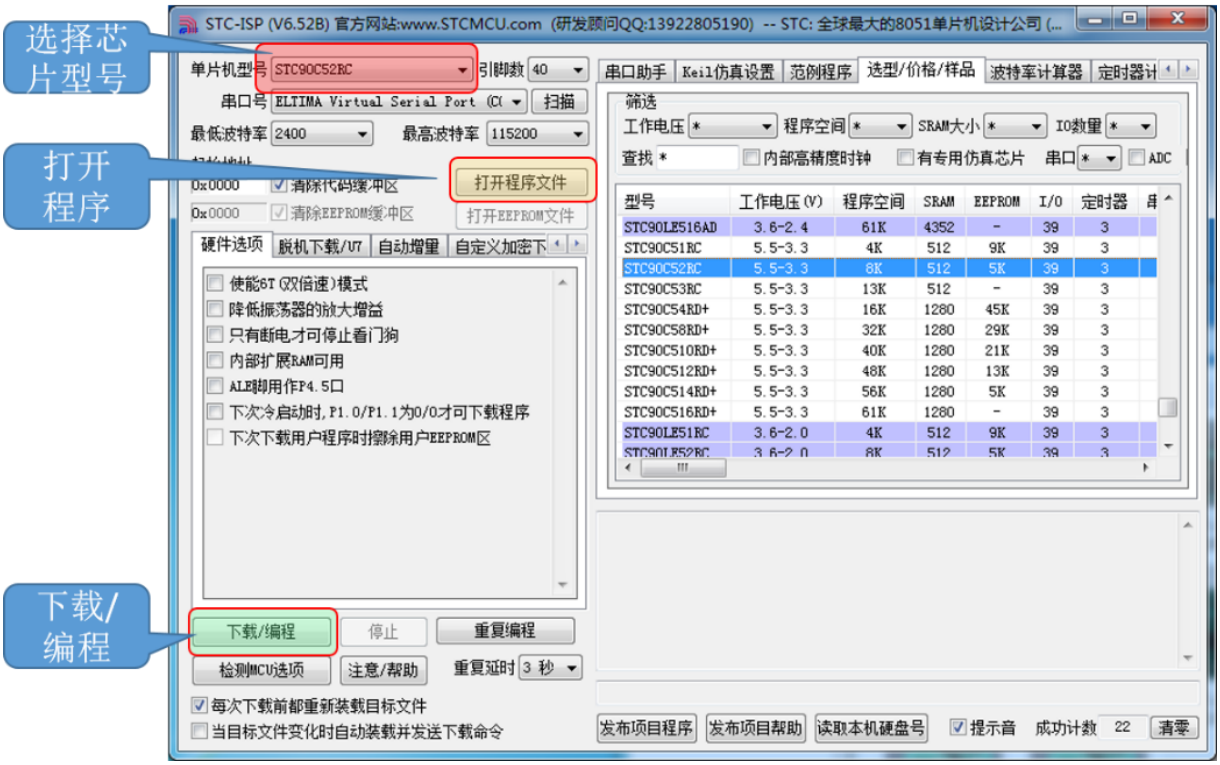

图 4- 23 STC 单片机程序烧写示意图

打开 STC-ISP 下载软件，选中单片机型号为“STC90C52RC”,选择最低波特率为 2400 最高波特率为 115200（为默认值一般不需修改）。点击打开程序文件按钮，在刚才建立的 LCD12864 文件夹中生成的 LCD12864.HEX 文件。然后点击“下载/编程”按钮。最后给最小系统通电，等待电脑端串口与单片机握手成功。就能把程序载入到单片机中去了。其详细操作图如**错误!未找到引用源。**所示：

5.4 硬件调试

程序下载完成后，把单片机最小系统接入电路中，调试电路使其能正常工作，并在 LCD12864 上显示初始化界面。

四、任务评价

表 4- 7　任务二完成情况评价表

项目名称			评价时间		年　月　日		
小组名称		小组成员					
评价内容	评价要求	权重	评价标准	学生自评得分	小组评价得分	教师评价得分	合计
职业与安全意识	1. 工具摆放、操作符合安全操作规程；2. 遵守纪律、爱惜设备和器材、工位整洁；3. 具有团队协作精神。	10%	好（10） 较好（8） 一般（6） 差（<6）				
印制线路板调试	1. 能够根据任务要求进行印制线路板的调试；2. 根据任务要求，能够对对应电路部分进行硬件电路的检查与故障检修。	15%	好（15） 较好（12） 一般（9） 差（<9）				印制线路板调试
项目功能测试	1. 编写的程序能成功编译；2. 程序能正确烧写到芯片中；3. 能够在 LCD12864 上正常的显示汉字和 ASCII 码。	60%	好（60） 较好（45） 一般（30） 差（<30）				
问题与思考	1. 如何显示大小不同的汉字 2. 如何反白显示汉字和数字；	15%	好（15） 较好（12） 一般（9） 差（<6）				
教师签名			学生签名			总分	
项目评价=学生自评（0.2）+小组评价（0.3）+教师评价（0.5）							

五、知识拓展

一、液晶显示模块分类

市场上液晶显示模块按功能分主要有三种：字段式，点阵字符式和点阵图形式。

字段式液晶显示器与 LED 数码管显示器有些相同，它是以长条笔划状或一些特殊固定图形与汉字显示像素组成的液晶显示器件，一般只能只能显示数字和一些标识符号，常用字段式液晶显示器件大多应用在便携、袖珍设备上。

点阵字符式液晶显示模块是由点阵字符液晶显示器件和专用的行、列驱动器、控制器及必要的连接件，结构件装配而成的，可以显示数字和西文字符。这种点阵字符模块本身具有字符发生器，可以显示多种字符、数字、字母、标点符号等点阵图形，显示容量大，功能丰富。常见的有 1 行、2 行、4 行，每行可显示 8、12、16、24 个 5×7 点阵字符的通用液晶显示器，例如 1602 液晶显示器就是一个 2 行，每行有 16 个 5×7 点阵字符字符式液晶显示器。

点阵图形式液晶显示器一般显示面积大于点阵式液晶显示器，其特点是点阵像素连续排列，行和列在排布中均没有空隔。因此可以显示了连续、完整的图形。由于它也是有 X-Y 矩阵像素构成的，所以除显示图形外，也可以显示字符。例如 TG12864 液晶显示器就是一款像素总量为 128×64，可以显示 8 列×4 行汉字的点阵图形式的液晶显示器。

二、12864 液晶显示模块功能拓展

1．反白显示

实现字符反白显示可以用如下方法来实现：

将要反白显示的字符字模数据在送入液晶显示 RAM 之前进行按位求反。任务一中显示函数可以修改反白显示的参考函数如下：

反白显示的参考子程序

```
void disp_num8x16(uchar row,uchar col,uchar n,uchar fb)
{//row 页，col 列，n 第几个字符,fb 反白
    uchar i,dz;                          //定义 i,dz 为局部变量
```

```
        if(col<64){CS1=1,CS2=0;}              //选左半屏
        else{CS1=0,CS2=1;col-=64;}                 //选右半屏
        for(i=0;i<8;i++)                      //for 循环控制 8 列扫描字符
        {
            wcmd_12864(0xb8+row);                  //设定页
            wcmd_12864(0x40+col+i);                //设定列
            if(fb==1)dz=~ num8X16[i+n*16];   //反白显示点阵数据
            else dz=num8X16[i+n*16];         //正常显示点阵数据
      wdat_12864(dz);                    //写入显示 RAM
      wcmd_12864(0xb8+row+1);          //设定页
            wcmd_12864(0x40+col+i);                //设定列
            if(fb==1)dz=~ num8X16[i+n*16];   //反白显示点阵数据
            else dz=num8X16[i+n*16];         //正常显示点阵数据
            wdat_12864(num8X16[i+n*16+8]);      //写入
}
}
```

如果编写的显示函数具有绘图功能，则该显示函数必须先要读出液晶相应位置显示 RAM 的数据，这样可以在读出数据后进行按位求反后写入。

2．滚动或闪烁显示

可以使用下面的函数使得扫描起始线改变即可。参考子函数如下：

滚动显示的参考子程序

```
void    lcd_rol()                          //向上滚屏
{
      unsigned char i;
      for(i = 0; i < 0xc0; i++)
      {
            CS1=1;                                  //  片选 1
            CS2=1;
            SendCommandToLCD(0xc0+i);                  //向上滚
            delay(10);
      }
}
```

六、思考与练习

1．使用洗衣机模拟装置硬件完成任务二《洗衣机界面》的模拟制作。

2．如何来控制 LCD12864 背光亮度的控制？

3．怎样在 LCD12864 液晶屏上面显示一幅图片？

任务三　直流电动机正反转控制

一、任务要求

控制直流电动机实现正反转动的效果。其具体要求如下：

按下按键 S1 电动机正转

按下按键 S2 电动机反转

按下按键 S3 电动机停止转动

二、相关知识——直流电动机简介

1、直流电动机介绍

1.1 直流电动机定义

要完成本项目，首先得认识一下直流电动机（如图 4- 24）;什么是直流电动机呢？直流电动机是指能将直流电能转换成机械能（直流电动机）或将机械能转换成直流电能（直流发电机）的旋转电机。它是能实现直流电能和机械能互相转换的电动机。当它作电动机运行时是直流电动机，将电能转换为机械能；作发电机运行时是直流发电机，将机械能转换为电能。

图 4- 24 直流电实物图

1.2 直流电动机的分类

直流电动机按结构及工作原理可划分：无刷直流电动机和有刷直流电动机

（1）无刷直流电动机

无刷直流电动机由电动机主体和驱动器组成，是一种典型的机电一体化产品。无刷电动机是指无电刷和换向器（或集电环）的电动机，又称无换向器电动机。

（2）有刷直流电动机

有刷电动机的定子上安装有固定的主磁极和电刷，转子上安装有电枢绕组和换向器。直流电源的电能通过电刷和换向器进入电枢绕组，产生电枢电流，电枢电流产生的磁场与主磁场相互作用产生电磁转矩，使电动机旋转带动负载。

本项目中我们使用的是直流有刷电动机。

1.2 直流电动机的工作原理

直流电动机里边固定有环状永磁体，电流通过转子上的线圈产生安培力，当转子上的线圈与磁场平行时，再继续转受到的磁场方向将改变，因此此时转子末端的电刷跟转换片交替接触，从而线圈上的电流方向也改变，产生的洛伦兹力方向不变，所以电动机能保持一个方向转动。

直流发电机的工作原理就是把电枢线圈中感应的交变电动势，靠换向器配合电刷的换向作用，使之从电刷端引出时变为直流电动势。从而带动电机转动。

2、直流电动机驱动

由于我们的单片机每个 I/0 的输出电流是有限的。这个时候就需要直流电动机驱动电路来驱动电动机。直流电动机驱动电路其实就是对单片机 I/0 的电流就行放大以达到电动机转动所需要的电流。

2.1 根据直流电动机需要达到的转动效果选择驱动电路

直流电动机是单向还是双向转动？需不需要调速？

1）对于单向的直流电动机驱动，只要用一个大功率三极管或场效应管或继电器就可以控制电动机的转动或者不转动。

2）当电动机需要双向转动时，可以使用由 4 个功率元件（三极管或场效应管）组成的 H 桥电路或者使用一个双刀双掷的继电器来控制电动机的正转或者反转。

3）如果不需要对直流电动机进行调速，只要使用继电器即可实现正反转；

4）如果需要调速，可以使用三极管，场效应管等开关元件实现 PWM（脉冲宽度调制）调速。

本项目选用的是第四种电路可以对直流电动机进行正反转控制，和正反两个方向的速度控制。

2.2 直流电动机驱动原理分析

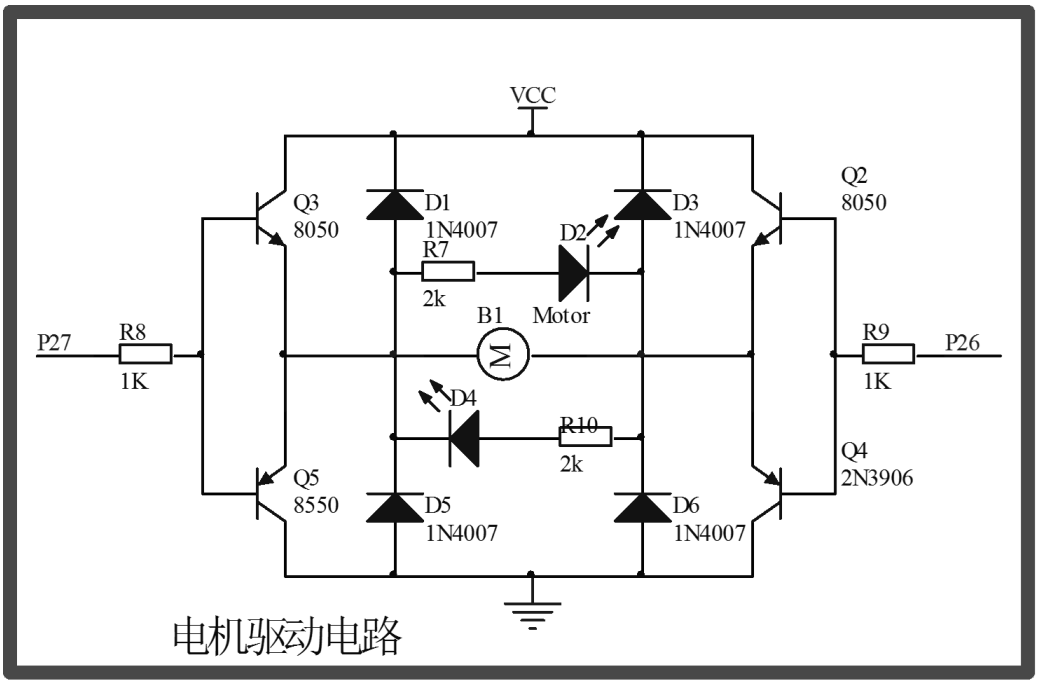

图 4- 25 电动机驱动原理图

电动机驱动电路主要由四个三极管 Q2(8050)、Q3(8050)、Q4(8550)、Q5(8550)组成的 H 桥驱动电路和二极管 D1、D3、D5、D6 组成的保护电路以及 D2、D4 组成的电动机方向指示灯构成的。下面我们主要来掌握 H 桥驱动电路原理以及控制方法。我们通过验证的方法来看一下电动机如何实现正反转的。

1）P2.6 和 P2.7 都置成低电平。三极管 Q2 和 Q3 都截止，电动机两端没有电压所以电动机停转。

2）P2.6 置成高电平 P2.7 置成低电平。那么三极管 Q2 和 Q5 导通，Q3 和 Q4 截止。电流通过三级管 Q2 流向电动机后在通过三极管 Q5 流向地（如图 4-26 所示），这样电动机中就有电流流过了，电动机就可以转动起来了。

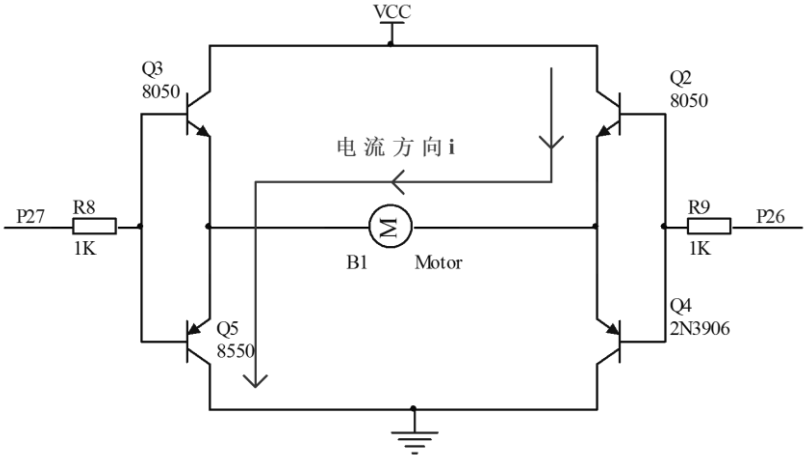

图 4- 26　电动机反转电流流向示意图

3）P2.6 置成低电平 P2.7 置成高电平。那么三极管 Q3 和 Q4 导通，Q2 和 Q5 截止。电流通过三级管 Q3 流向电动机后在通过三极管 Q4 流向地（如图 4-27 所示），这样电动机中就有反向的电流流过了，电动机就可以以相反的方向转动起来了。

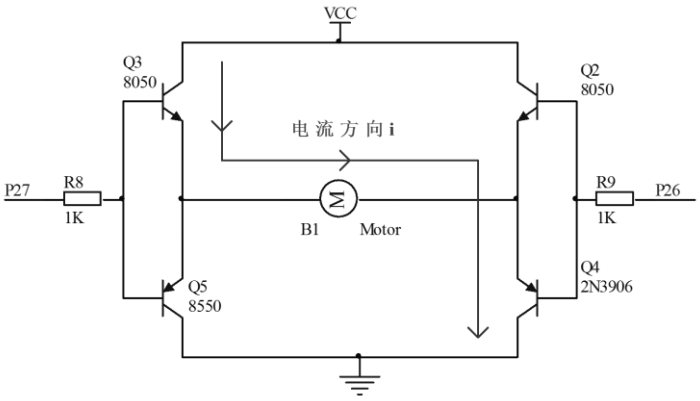

图 4- 27　电动机正转电流流向示意图

4）P2.6 和 P2.7 都置成高电平。三极管 Q5 和 Q4 都截止，Q2 和 Q3 都导通，电动机两端都是电源电压导致没有电流流过电动机，所以电动机停转。

3、直流电动机软件编程

3.1 电动机正转程序的编写

有了上面直流电动机驱动原理后要想让电动机转动起来就变得非常的简单了，只要在电动机一端加正电压另外一端接地，这样电动机就可以转动起来了，通过调整两端的电压就可以实现电动机的正向转动。以下为正转范例程序。

电动机正转操作子程序

```
sbit ZPort      =     P2^7;//配置 P2.7 为电动机正端
sbit FPort      =     P2^6; //配置 P2.6 为电动机负端
void M_Zheng()//电动机正转
{
      ZPort = 1；//电动机正端和电源连接
      FPort = 0；//电动机负端和地连接
}
```

3.2 电动机反转子程序的编写

反转和正转原理相同。以下为反转范例程序。

电动机反转操作子程序

```
sbit ZPort      =     P2^7;//配置 P2.7 为电动机正端
sbit FPort      =     P2^6; //配置 P2.6 为电动机负端
void M_Zheng()//电动机反转
{
      ZPort = 0；//电动机正端和地连接
      FPort = 1；//电动机负端和电源连接
}
```

3.3 电动机停止程序的编写

电动机停止只要让电动机的两端都没有电流流过就可以实现了。

电动机停止操作子程序

```
sbit ZPort      =     P2^7;//配置 P2.7 为电动机正端
sbit FPort      =     P2^6; //配置 P2.6 为电动机负端
void M_Stop()//电动机停止
```

```
{
    ZPort = 0；//电动机正端和地连接
    FPort = 0；//电动机负端和地连接
}
```

三、操作训练

1、任务分析

本程序所有的动作都是通过按键进行控制的，当某一个按键按下之后，通过调用相应的电动机程序来实现电动机正反转的功能。

2、主函数流程图

主函数流程图如图 4- 28 所示。

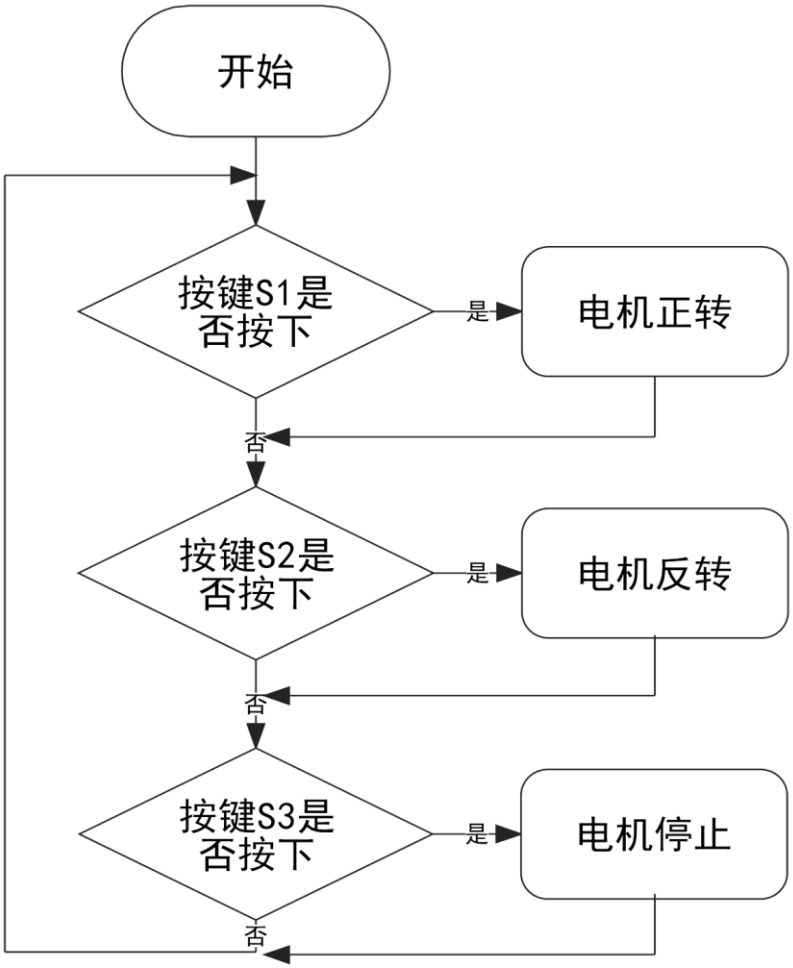

图 4- 28 直流电动机正反转控制流程图

3、参考程序

根据图 4- 28 图 4- 28 编写程序：

直流电动机正反转控制主程序 LCD1602.C

```
#include<reg51.h>              // 包含 51 单片机头文件
/**********************电动机控制*******************/
sbit ZPort = P2^7;//配置 P2.7 为电动机正端
sbit FPort = P2^6;//配置 P2.6 为电动机负端
void M_Zheng()//电动机正转
{
    ZPort=1;//电动机正端和电源连接
    FPort=0;//电动机负端和地连接
}
void M_Fan()//电动机反转
{
    ZPort=0;//电动机正端和地连接
    FPort=1;//电动机负端和电源连接
}
void M_Stop()//电动机停转
{
    ZPort=0;//电动机正端和地连接
    FPort=0;//电动机负端和地连接
}

/************************************************/
/***************按键子程序************************/
unsigned char key_value;//按键键值
sbit S1 = P2^0;//配置按键 1 为 P2.0  电动机正转
sbit S2 = P2^1;//配置按键 2 为 P2.1  电动机反转
sbit S3 = P2^2;//配置按键 3 为 P2.2  电动机停止
void KEY()//按键
{
    unsigned char key_time;
    S1=S2=S3=1;//将按键口全置 1
    if(!S1||!S2||!S3)//判断是否有按键按下
    {
```

```
            if(++key_time==10)//计时，防抖动
            {
                if(!S1)key_value=1;//赋值键值为 1
                if(!S2)key_value=2;//赋值键值为 2
                if(!S3)key_value=3;//赋值键值为 3
            }
        }
        else key_time=0;//计时清零
}
void Read_KEY()//读键值
{
        if(key_value)//判断是否有按键按下
        {
            switch (key_value)
            {
                case 1:M_Zheng();break;//电动机正转
                case 2:M_Fan();break;//电动机反转）
                case 3:M_Stop();break;//电动机停止
                default:   break;
            }
            key_value=0;//键值清零
        }
}
void main(void)//主函数
{
        while(1)
        {
            KEY();
            Read_KEY();//读按键值执行相应的动作
        }
}
```

4、程序说明

本程序通过调用 KEY()函数来获取当前哪个按键被按下。然后把按键按下的数值给 Read_KEY()函数。通过 Read_KEY()函数调用相应的电动机程序。

5、任务实施

5.1 建立工程

打开 keil 软件，通过菜单“Project”，新建立一个工程”new Project”项目 DZKZ，然后再建一个文件名为 DJZK.C 的源程序文件，将上面的参考程序输入并保存。建立的 DJZK.C 文件添加入本项目中。

5.2 编译并生成 HEX

单击”Build target”按钮，对源程序进行编译和链接，产生 HEX 文件。

5.3 烧录芯片

打开 STC-ISP 下载软件，选中单片机型号为“STC90C52RC”,选择最低波特率为 2400 最高波特率为 115200（为默认值一般不需修改）。点击打开程序文件按钮，在刚才建立的 DJZK 文件夹中生成的 DJZK.HEX 文件。然后点击“下载/编程”按钮。最后给最小系统通电，等待电脑端串口与单片机握手成功。就能把程序载入到单片机中去了。其详细操作图如**错误!未找到引用源。**所示：

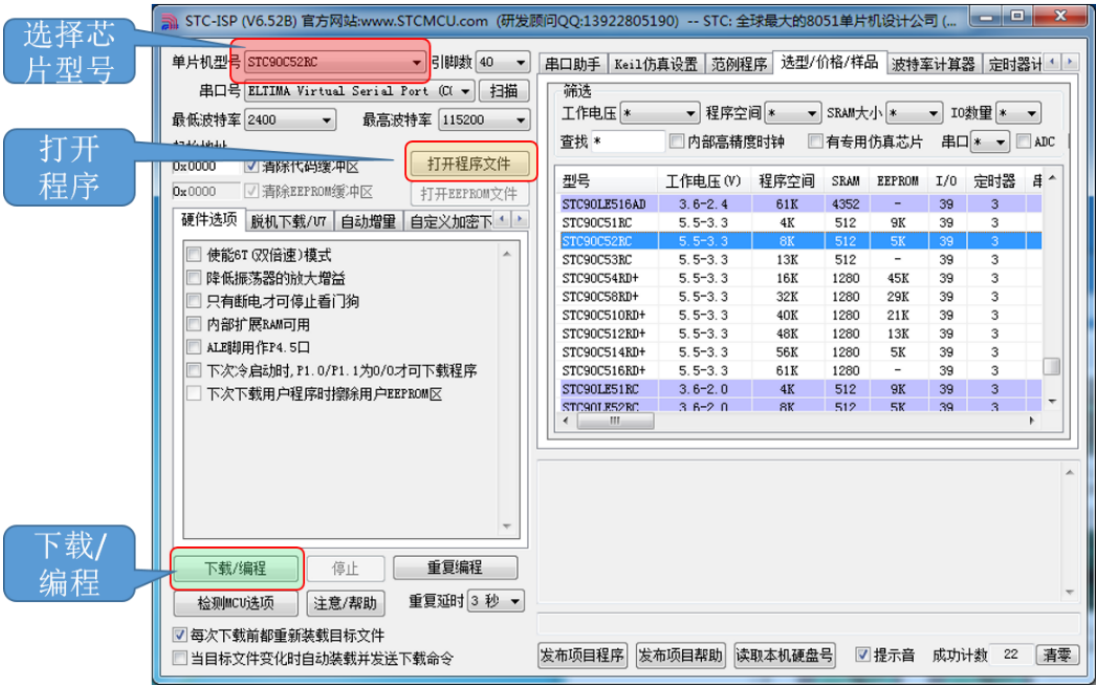

图 4- 29 STC 单片机程序烧写示意图

5.4 硬件调试

程序下载完成后，把单片机最小系统接入电路中，调试电路使其能正常工作，并且按下按键后电动机可以执行相应的动作。

四、任务评价

表 4- 8　任务三完成情况评价表

项目名称			评价时间		年　月　日		
小组名称		小组成员					
评价内容	评价要求	权重	评价标准	学生自评得分	小组评价得分	教师评价得分	合计
职业与安全意识	1. 工具摆放、操作符合安全操作规程；2．遵守纪律、爱惜设备和器材、工位整洁；3. 具有团队协作精神。	10%	好（10） 较好（8） 一般（6） 差（<6）				
印制线路板调试	1．能够根据任务要求进行印制线路板的调试；2. 根据任务要求，能够对对应电路部分进行硬件电路的检查与故障检修。	15%	好（15） 较好（12） 一般（9） 差（<9）				
项目功能测试	1．编写的程序能成功编译；2. 程序能正确烧写到芯片中；3. 通过按下相应的按键电动机能过执行相应的动作。	60%	好（60） 较好（45） 一般（30） 差（<30）				
问题与思考	1．怎样改变电动机的转速；2. 如何知道电动机的转速	15%	好（15） 较好（12） 一般（9） 差（<6）				
教师签名			学生签名			总分	
项目评价=学生自评（0.2）+小组评价（0.3）+教师评价（0.5）							

五、知识拓展

1、步进电动机

步进电机（图 4- 30）是将电脉冲信号转变为角位移或线位移的开环控制元步进电机件。在非超载的情况下，电机的转速、停止的位置只取决于脉冲信号的频率和脉冲数，而不受负载变化的影响，当步进驱动器接收到一个脉冲信号，它就驱动步进电机按设定的方向转动一个固定的角度，称为“步距角”，它的旋转是以固定的角度一步一步运行的。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来控制电机转动的速度和加速度，从而达到调速的目的。

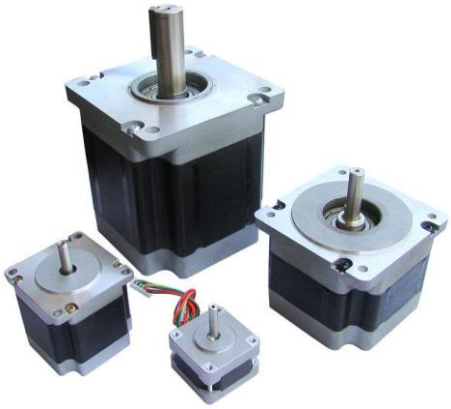

图 4- 30 步进电动机

2、交流电动机

交流电机（图 4- 31）是用于实现机械能和交流电能相互转换的机械。由于交流电力系统的巨大发展，交流电机已成为最常用的电机。交流电机与直流电机相比，由于没有换向器，因此结构简单，制造方便，比较牢固，容易做成高转速、高电压、大电流、大容量的电机。交流电机功率的覆盖范围很大，从几瓦到几十万千瓦、甚至上百万千瓦。

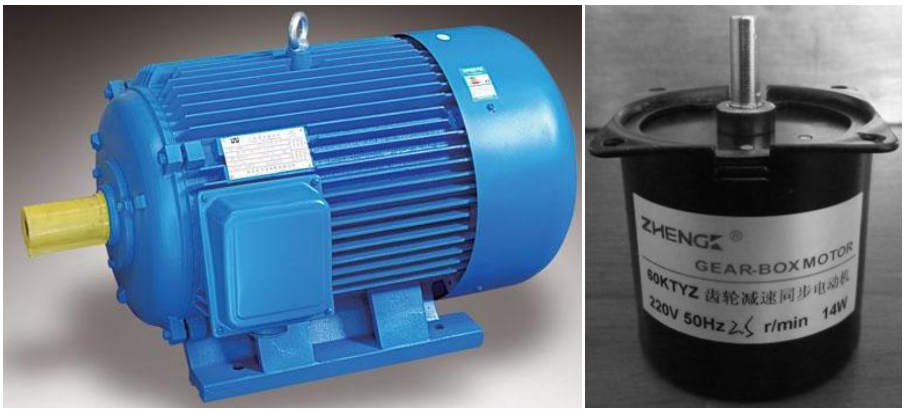

图 4- 31 交流电动机

六、思考与练习

1．使用洗衣机模拟装置硬件完成任务三《直流电动机正反转的控制》的制作。

2．思考如何在液晶上显示电动机的转动方向？

任务四　PWM 直流电动机调速控制

一、任务要求

控制直流电动机实现调速的效果。其具体要求如下：

按下按键 S1 电动机正转

按下按键 S2 电动机加速旋转

按下按键 S3 电动机减速旋转

按下按键 S4 电动机停止转动

二、相关知识——电动机调速简介

1、PWM 介绍

脉冲宽度调制(PWM)，简称脉宽调制，是利用微处理器的数字输出来对模拟电路进行控制的一种非常有效的技术，广泛应用在从测量、通信到功率控制与变换的许多领域中。

2、PWM 驱动直流电动机原理

直流电动机的转速计算公式如下：n=(U-IR)/KΦ，其中 U 为电枢端电压，I 为电枢电流，R 为电枢电路总电阻，Φ 为每极磁通量，K 为电动机结构参数。可以看出，转速和 U、I 有关，并且可控量只有这两个，我们可以通过调节这两个量来改变转速。我们知道，I 可以通过改变电压进行改变，而我们常提到的 PWM 控制也就是用来调节电压波形的常用方法，这里我们也就是用 PWM 控制来进行电动机转速调节的。通过单片机输出一定频率的方波，方波的占空比大小绝对平均电压的大小，也决定了电动机的转速大小。

3、PWM 调速直流电动机驱动的性能

性能：对于 PWM 调速的电动机驱动电路，主要有以下性能指标。

1）输出电流和电压范围：它决定着电路能驱动多大功率的电动机。

2）效率：高的效率不仅意味着节省电源，也会减少驱动电路的发热。要提高电路的效率，可以从保证功率器件的开关工作状态和防止共态导通（H 桥或推挽电路可能出现的一个问题，即两个功率器件同时导通使电源短路）入手。

3）对控制输入端的影响：功率电路对其输入端应有良好的信号隔离，防止有高电压大电流进入主控电路，这可以用高的输入阻抗或者光电耦合器实现隔离。

4）对电源的影响：共态导通可以引起电源电压的瞬间下降造成高频电源污染；大的电流可能导致地线电位浮动。

5）可靠性：电动机驱动电路应该尽可能做到，无论加上何种控制信号，何种无源负载，电路都是安全的。

4、PWM 调速直流电动机驱动电路分析

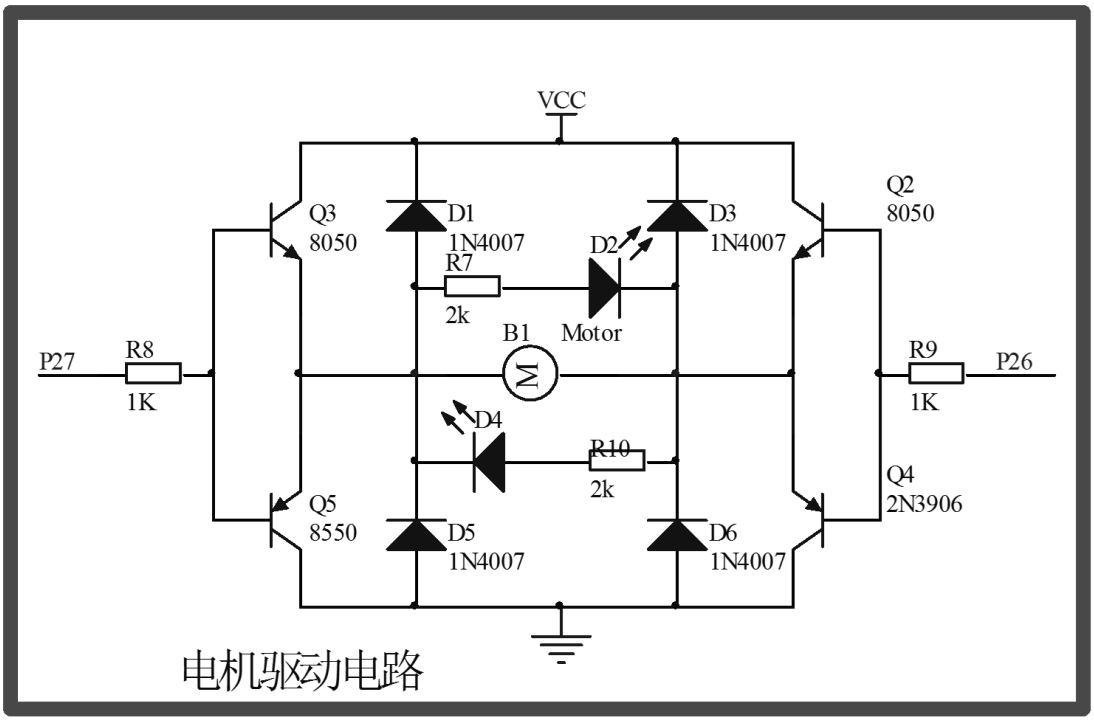

图 4- 32 电动机驱动电路

PWM 调速电路和电动机正反转使用的是同一个电路如图 4- 32 所示。

调速时我们只需要控制 P2.6 口为低电平。这样当控制 P2.7 的波形的占空比时就可以进行调速了。

5、PWM 软件编程

有上面 PWM 的理论基础之后。我们就可以来产生 PWM 波形了。通过定时器产生一个基本时间，我们通过对它进行一个的累加，累加到一个固定的值（例程为 100）就可以得到一个固定的频率（频率=1/（固定值定×时器时间））。在累加的过程中我们可以控制波形是高电平还是低电平，从而得到我们想要的占比。例如我们想要得到 40%的占空比，那么我们可以在开始时把 IO 口置成高电平当累加值到达 40（pwm_val 的值）时置成低电平，到达 100 置成高平并复位累加值，如此反复我们就可以得到 40%的占空比的波形（如图 4- 33 所示）。

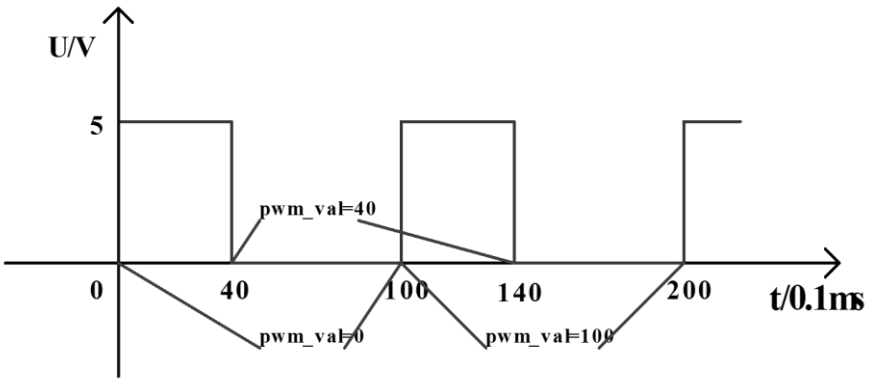

图 4- 33 PWM 波产生示意图

PWM 操作程序

```
void timer0(void) interrupt 1 //定时器中断
{
    static unsigned char count=0;//波形的频率控制计数值
    count++;                    //计数值加一
    if(count==pwm_val)          //判断计数值是否到达占空比的值
    {
        PWM1=1;                 //把波形置 1
    }
    if(count==100)              //判断一个波形是否完成
    {
        count=0;                //计数值清零
        PWM1=0;                 //把波形置 2
    }
}
```

三、操作训练

1、任务分析

这任务中涉及到了电动机速度的控制，这边通过 PWM 来进行电动机的调速控制。这个任务中我们首先通过按键让电动机转动起来。然后在转动的基础上在进行调速。调速主要是产生不同占空比的波形来驱动电动机，占空比越大的波形给电动机的电压也就越高电动机也转的越快，相反占空比越小给电动机的电压也就越低电动机也就转的越慢。这样就可以达到电动机调速的目的了。

2、主函数流程图

主函数流程图如图 4- 34。所示。

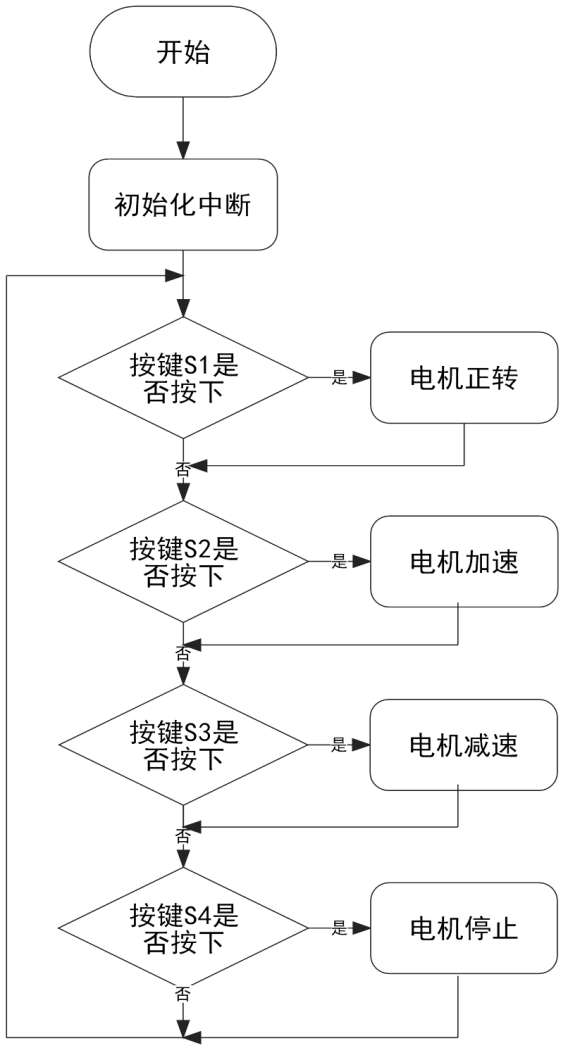

图 4- 34 PWM 直流电动机调速流程图

3、参考程序

根据图 4- 34 PWM 直流电动机调速流程图编写程序：

PWM 直流电动机调速程序

```
#include<reg51.h>                    // 包含 51 单片机头文件
unsigned char pwm_val=100; //占空比设置值
/**********************电动机控制*******************/
sbit ZPort = P2^7;//配置 P2.7 为电动机正端
sbit FPort = P2^6;//配置 P2.6 为电动机负端
void M_Zheng()//电动机正转
{
    ZPort=1;//电动机正端和电源连接
    FPort=0;//电动机负端和地连接
}
void M_Stop()//电动机停转
{
    ZPort=0;//电动机正端和地连接
    FPort=0;//电动机负端和地连接
}
/***************按键子程序************************/
unsigned char key_value;//按键键值
sbit S1 = P2^0;//配置按键 1 为 P2.0
sbit S2 = P2^1;//配置按键 2 为 P2.1
sbit S3 = P2^2;//配置按键 3 为 P2.2
sbit S4 = P2^3;//配置按键 4 为 P2.3
void KEY(void)//按键
{
    unsigned char key_time;
    S1=S2=S3=S4=1;//将按键口全置 1
    if(!S1||!S2||!S3||!S4)//判断是否有按键按下
    {
        if(++key_time==10)//计时，防抖动
        {
            if(!S1)key_value=1;//赋值键值为 1
            if(!S2)key_value=2;//赋值键值为 2
            if(!S3)key_value=3;//赋值键值为 3
```

```
            if(!S4)key_value=4;//赋值键值为 4
        }
    }
    else key_time=0;//计时清零
}
void Read_KEY()//读键值
{
    if(key_value)//判断是否有按键按下
    {
        switch (key_value)
        {
            case 1:M_Zheng();
                    pwm_val=100;//赋值占空比的初始值
                     EA=1;          //打开定时器
                    break;//电动机正转
            case 2:
                     pwm_val++;   //加大占空比  电动机加速
                     if(pwm_val==101)//占空比最大为 100  占空比限制
                     pwm_val=100;
                     break;//
            case 3:
                     pwm_val--;//减小占空比  电动机减速
                     if(pwm_val==255)//占空比最小为 0  占空比限制
                     pwm_val=0;
                     break;//电动机加速
            break;//电动机减速
            case 4:M_Stop();
                     EA=0;
                    break;//电动机停止
            default:   break;
        }
        key_value=0;//键值清零
    }
}
void INIT_TIME0(void)//初始化定时器 0
{
    TMOD = 0x02;//使用定时器模式 2 自动加载
```

```
    TH0=0xA1; //定时时间 0.1mS
    TL0=0xA1;
    ET0=1;      //允许定时器 0 中断
    TR0=1;      //开始计数
}
void main(void)//主函数
{
    INIT_TIME0();
    while(1)
    {
        KEY();
        Read_KEY();//读按键值执行相应的动作
    }
}
void timer0(void) interrupt 1 //定时器中断
{
    static unsigned char count=0;//波形的频率控制计数值
    count++;                    //计数值加一
    if(count==pwm_val)          //判断计数值是否到达占空比的值
    {
        ZPort=0;                //把波形置 0
    }
    if(count==100)              //判断一个波形是否完成   100HZ 波形
    {
        count=0;                //计数值清零
        ZPort=1;                //把波形置 1
    }
}
```

4、程序说明

本程序开始先调用 INIT_TIME0 函数对定时进行初始化。定时时间为 0.1 毫秒。在主程序中调用按键程序和按键处理程序。当相应的按键按下时函数会调用相应的子程序包括 M_Zheng 和 M_Stop。如果是按键是执行电机转速控制的，那在按键中就对 PWM 占空比的值 pwm_val 就行设置以达到电机转速调节的效果。

5、任务实施

5.1 建立工程

打开 keil 软件，通过菜单“Project”，新建立一个工程”new Project”项目 PWM，然后再建一个文件名为 PWM.C 的源程序文件，将上面的参考程序输入并保存。建立的 PWM.C 文件添加入本项目中。

5.2 编译并生成 HEX

单击”Build target”按钮，对源程序进行编译和链接，产生 HEX 文件。

5.3 烧录芯片

打开 STC-ISP 下载软件，选中单片机型号为“STC90C52RC”,选择最低波特率为 2400 最高波特率为 115200（为默认值一般不需修改）。点击打开程序文件按钮，在刚才建立的 PWM 文件夹中生成的 PWM.HEX 文件。然后点击“下载/编程”按钮。最后给最小系统通电，等待电脑端串口与单片机握手成功。就能把程序载入到单片机中去了。其详细操作图如**错误!未找到引用源。**所示：

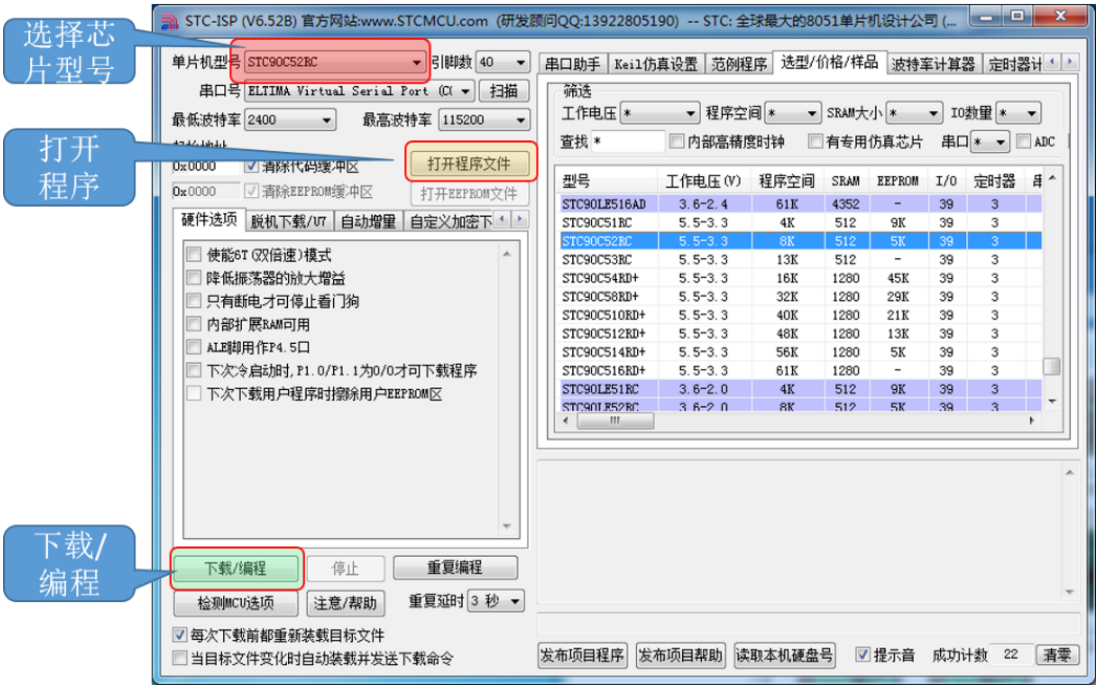

图 4- 35 STC 单片机程序烧写示意图

5.4 硬件调试

程序下载完成后，把单片机最小系统接入电路中，调试电路使其能正常工作，并且按下按键后电动机可以执行相应的动作。

四、任务评价

表 4- 9　任务四完成情况评价表

项目名称			评价时间		年　月　日		
小组名称		小组成员					
评价内容	评价要求	权重	评价标准	学生自评得分	小组评价得分	教师评价得分	合计
职业与安全意识	1. 工具摆放、操作符合安全操作规程；2. 遵守纪律、爱惜设备和器材、工位整洁；3. 具有团队协作精神。	10%	好（10） 较好（8） 一般（6） 差（<6）				
印制线路板调试	1. 能够根据任务要求进行印制线路板的调试；2. 根据任务要求，能够对对应电路部分进行硬件电路的检查与故障检修。	15%	好（15） 较好（12） 一般（9） 差（<9）				印制线路板调试
项目功能测试	1. 编写的程序能成功编译；2. 程序能正确烧写到芯片中；3. 按下正转按键电动机能够正转，按下停止按键电动机能够停止转动，按下调速按键电机转速应能改变。	60%	好（60） 较好（45） 一般（30） 差（<30）				
问题与思考	1. 当降低调速频率时电动机会有什么现象？2. 如何实现反转调速？	15%	好（15） 较好（12） 一般（9） 差（<6）				
教师签名			学生签名			总分	
项目评价=学生自评（0.2）+小组评价（0.3）+教师评价（0.5）							

五、知识拓展

1、步进电动机转速控制—步进电动机驱动器

步进电机驱动器（图 4- 36）是一种将电脉冲转化为角位移的执行机构。当步进驱动器接收到一个脉冲信号，它就驱动步进电机按设定的方向转动一个固定的角度(称为“步距角”)，它的旋转是以固定的角度一步一步运行的。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来控制电机转动的速度和加速度，从而达到调速的目的。

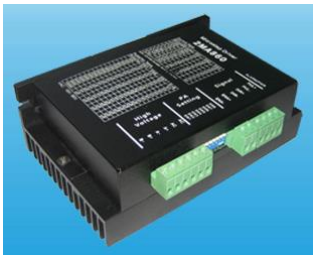

图 4- 36 步进电动机驱动器

2、交流电动机转速控制—变频器

变频器（Variable-frequency Drive，VFD）（图 4- 37）是应用变频技术与微电子技术，通过改变电机工作电源频率方式来控制交流电动机的电力控制设备，它可以通过改变频率来达到控制交流电动机转速。变频器主要由整流（交流变直流）、滤波、逆变（直流变交流）、制动单元、驱动单元、检测单元微处理单元等组成。变频器靠内部 IGBT 的开断来调整输出电源的电压和频率，根据电机的实际需要来提供其所需要的电源电压，进而达到节能、调速的目的，另外，变频器还有很多的保护功能，如过流、过压、过载保护等等。随着工业自动化程度的不断提高，变频器也得到了非常广泛的应用。

图 4- 37 变频器

六、思考与练习

1．使用洗衣机模拟装置硬件电路完成任务四《PWM 直流电动机调速控制》的模拟制作。

2．完成电机正反双向的调速程序。

3．思考当控制 PWM 频率越来越小电机会产生什么样的现象？

任务五　直流电动机转速测量

一、任务要求

任务要求：

控制直流电动机实现调速，在调速的过程中实时的通过液晶屏显示电动机当前的转速。其具体要求如下：

按下按键 S1 电动机正转

按下按键 S2 电动机加速旋转

按下按键 S3 电动机减速旋转

按下按键 S4 电动机停止转动、

在上面的过程中在液晶屏上第二行居中显示“电动机当前转速为”

第三行显示转速“xxx.xr/min”（如图 4- 38 电动机测速显示界面）其中 000 表示每分钟的转速。显示要求每秒钟更新一次。

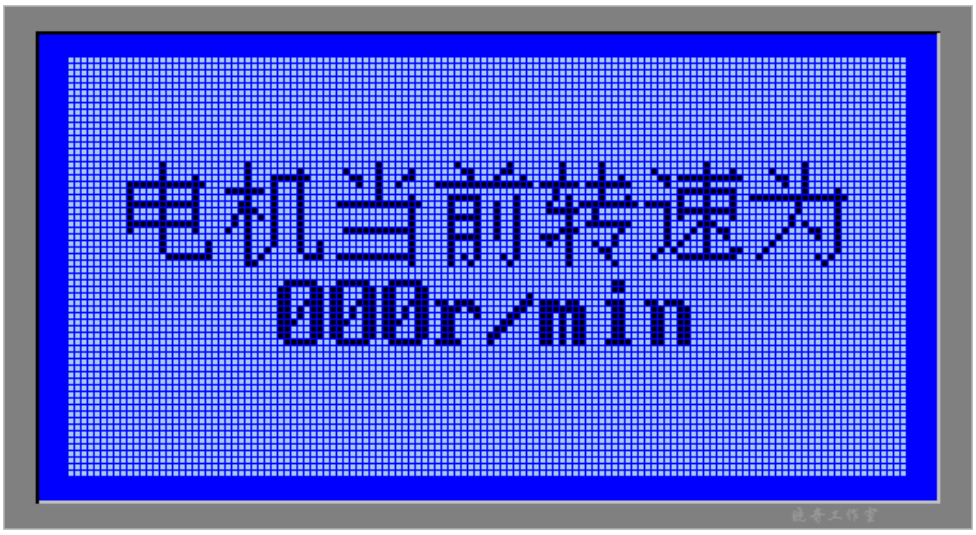

图 4- 38 电动机测速显示界面

二、相关知识——测速原理

说道测速原理我们必须来介绍本项目中用的测速元件关电开关。

1、光电开关介绍

光电开关是传感器大家族中的成员，它把发射端和接收端之间光的强弱变化转化为电流的变化以达到探测的目的。由于光电开关输出回路和输入回路是电隔离的（即电缘绝），所以它可以在许多场合得到应用。

常用光电开关的分类方法：按检测方式可分为两大类反射式、对射式

1.1 反射式光电开关

反射式光电开关（如图 4- 39）也属于红外线不可见光产品，是一种小型光电元器件，它可以检测出其接收到的光强的变化。在前期是用来检测物体有无感应到的,它是由一个红外线发射管跟一个红外线接收管组合而成，它的发射波长是 780nm-1mm，发射器带一个校准镜头，将光聚焦射向接收器，接收器出电缆将这套装置接到一个真空管放大器上。检测对象是当它进入间隙的开槽开关和块光路之间的发射器和检测器，当物体接近到灭弧室，接收器的一部分收集的光线从对象反射到光电元件上面。它是利用物体对红外线光束遮光或反射，由同步回路选通而检测物体的有无的，其物体不限于金属，对所有能反射光线的物体均可检测。

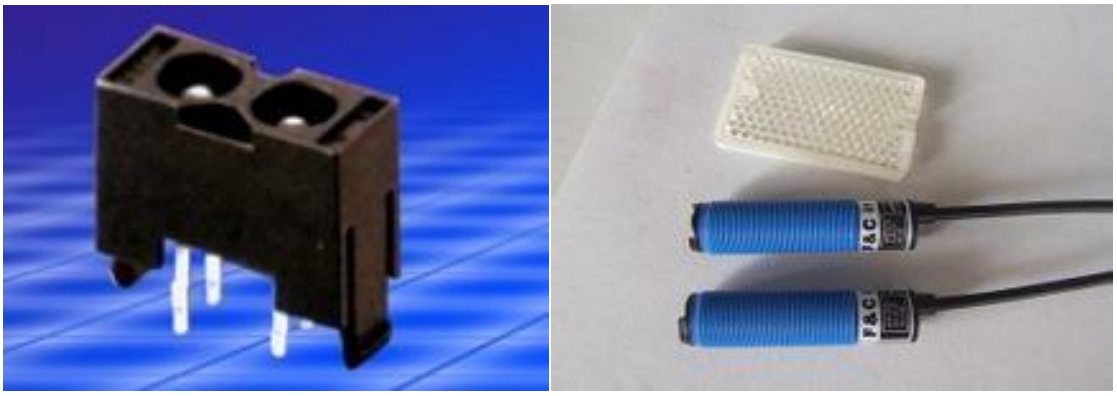

图 4- 39 反射式光电开关

1.2 对射式光电开关

对射式光电开关（如图 4- 40）由发射器和接收器组成，其工作原理是：通过发射器发出的光线直接进入接收器，当被检测物体经过发射器和接收器之前阻断光线时，光电开关就产生开关信号。与反射式光电开关不同之处在于，前者是通过电-光-电的转换，而后者是通过介质完成。对射式光电开关的特点在于：可辨别不透明的反光物体，有效距离大，不易受干扰，高灵敏度，高解

析，高亮度，低功耗，响应时间快，使用寿命长，无铅，广泛应用于：投币机，小家电，投币机，自动感应器，传真机，扫描仪等设备上面。

图 4- 40 对射式光电开关实物图

我们这个任务选用是对射式光电开，如图 4-40 所示。

2、光电开关原理图

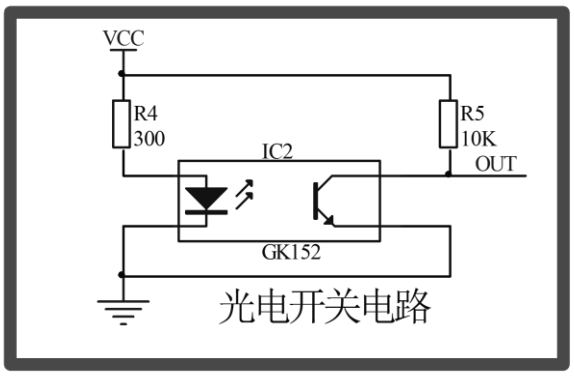

图 4- 41 光电开关原理图

光电开关电路的组成非常简单（如图 4- 41）一共只有三个元件，两个电阻加一个光电开关。从电路上我们可以看到光电开关的发射管是一直点亮的，接收管通过一个上拉电阻接到电源。它工作的原理是当有东西遮挡在光电开关检测槽中时，接收管接收不到发射管发出的光，接收管成高阻态，电路输出端 OUT 为高电平。当没有东西遮挡在光电开关检测槽中时，接收管接接收到发射管发出的光，接收管成低阻态，电路输出端 OUT 为低电平。

3、测速原理

我们通过光电码盘（如图 4- 42）测速法来测量电动机的转速。光电码盘是一种通过测出转速信号的频率或周期来测量电动机转速的一种无接触测速方

法。光电码盘一般安装在电动机转子端轴上，随着电动机的转动，光电码盘也跟着一起转动。如果我们有一个光电开关，我们可以让光电码盘部分放到我们的光电开关检测槽中。这样当光电码盘转动就会在我们的光电开关接收到的光的次数就是码盘的编码数。如果测量时间为 t，测量到的脉冲数为 N，则 n=N/t

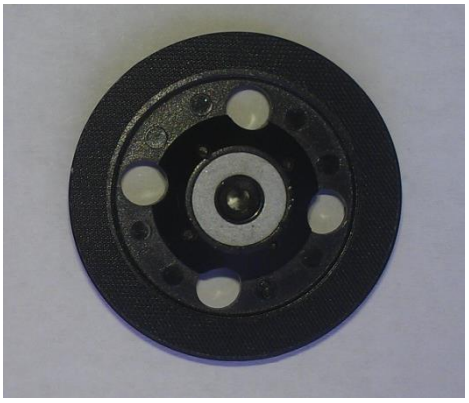

图 4- 42 **光电码盘**

4、测速软件编程

有了上面的原理后我们可以发现测量电动机的速度就是记录光电开关的频率。这里使用外部中断来计数使用定时器每秒中读出外部中断的频率，从而计算出电动机的转速。

电动机测速程序

```
void timer1(void) interrupt 3 //定时器中断
{
    static unsigned char i=0;
    TH1=0x4C; //定时时间 50mS
    TL1=0x00;
    i++;                        //计数值加一
    if(i==20)          //一秒钟时间
    {
        i=0;              //计数清零
        zhuansu=InitCount;//获取编码器的计数值
        InitCount=0;      //清空编码器的计数值从新计数
        zhuansu=zhuansu*60;//获取的编码器值是一秒中的 转换成一份钟计数值
        zhuansu=zhuansu/4;//编码器每计数 4 个表示电动机转动一圈
        flag1s=1;
```

```
    }

}

void init0(void) interrupt 2 //定时器中断
{
    InitCount++;
}
```

三、操作训练

1、任务分析

本任务上次的任务中添加了电动机测速部分和液晶显示转速部分。我们这节主要的任务就是电动机测速。电动机测速我们在之前已经介绍了。我本只需要读出转速然后显示显示出来。

2、主函数流程图

主函数流程图如图 4- 43。所示。

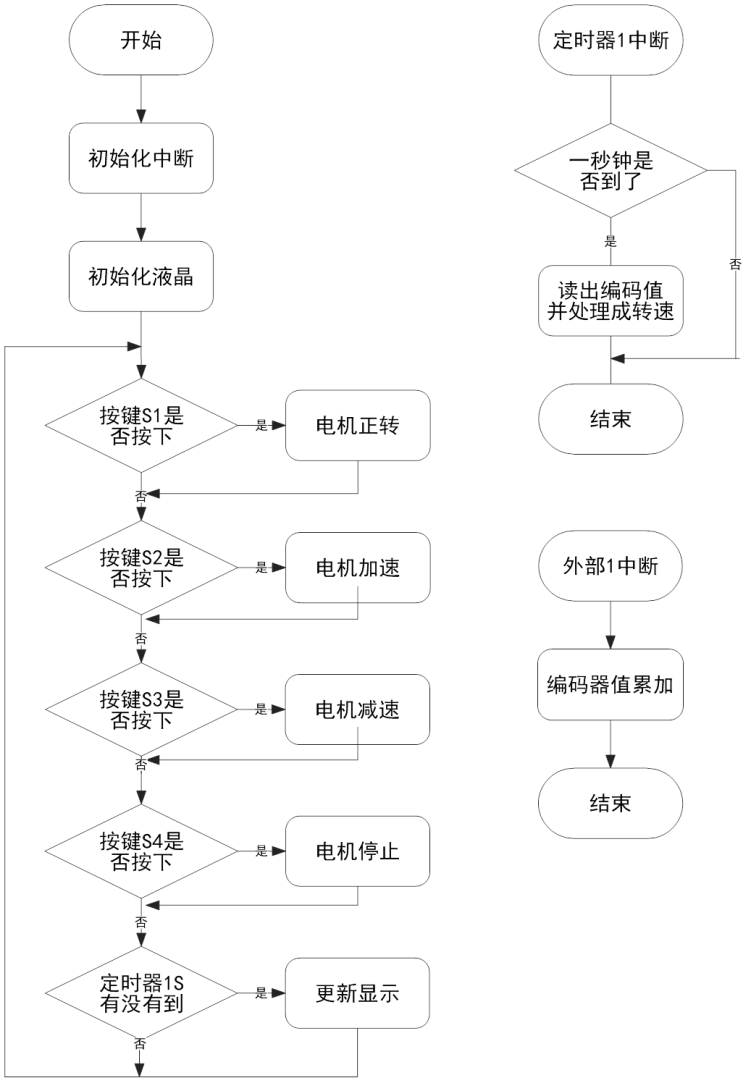

图 4- 43 直流电动机测速流程图

3、参考程序

根据图 4- 43 直流电动机测速编写直流电动机测速程序：

直流电动机测速程序

```
#include<reg51.h>                    // 包含 51 单片机头文件
unsigned char code HZ[][32]=
{
/*--  文字:  电  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x00,0xF8,0x88,0x88,0x88,0x88,0xFF,0x88,0x88,0x88,0x88,0xF8,0x00,0x00,0x00,
0x00,0x00,0x1F,0x08,0x08,0x08,0x08,0x7F,0x88,0x88,0x88,0x88,0x9F,0x80,0xF0,0x00,
/*--  文字:  机  --*/
```

```
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x10,0x10,0xD0,0xFF,0x90,0x10,0x00,0xFE,0x02,0x02,0x02,0xFE,0x00,0x00,0x00,0x00,
0x04,0x03,0x00,0xFF,0x00,0x83,0x60,0x1F,0x00,0x00,0x00,0x3F,0x40,0x40,0x78,0x00,
/*--  文字:  当  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x40,0x42,0x44,0x58,0x40,0x40,0x7F,0x40,0x40,0x50,0x48,0xC6,0x00,0x00,0x00,
0x00,0x40,0x44,0x44,0x44,0x44,0x44,0x44,0x44,0x44,0x44,0xFF,0x00,0x00,0x00,
/*--  文字:  前  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x08,0x08,0xE8,0x29,0x2E,0x28,0xE8,0x08,0x08,0xC8,0x0C,0x0B,0xE8,0x08,0x08,0x00,
0x00,0x00,0xFF,0x09,0x49,0x89,0x7F,0x00,0x00,0x0F,0x40,0x80,0x7F,0x00,0x00,0x00,
/*--  文字:  转  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0xC8,0xB8,0x8F,0xE8,0x88,0x88,0x40,0x48,0x48,0xE8,0x5F,0x48,0x48,0x48,0x40,0x00,
0x08,0x18,0x08,0xFF,0x04,0x04,0x00,0x02,0x0B,0x12,0x22,0xD2,0x0A,0x06,0x00,0x00,
/*--  文字:  速  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x40,0x40,0x42,0xCC,0x00,0x04,0xF4,0x94,0x94,0xFF,0x94,0x94,0xF4,0x04,0x00,0x00,
0x00,0x40,0x20,0x1F,0x20,0x48,0x44,0x42,0x41,0x5F,0x41,0x42,0x44,0x48,0x40,0x00,
/*--  文字:  为  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x20,0x22,0x2C,0x20,0x20,0xE0,0x3F,0x20,0x20,0x20,0x20,0xE0,0x00,0x00,0x00,
0x80,0x40,0x20,0x10,0x08,0x06,0x01,0x00,0x01,0x46,0x80,0x40,0x3F,0x00,0x00,0x00,
};
unsigned char code ASC[][16]=
{
    0xF8,0xFC,0x04,0xC4,0x24,0xFC,0xF8,0x00,    // -0-
    0x07,0x0F,0x09,0x08,0x08,0x0F,0x07,0x00,
    0x00,0x10,0x18,0xFC,0xFC,0x00,0x00,0x00,    // -1-
    0x00,0x08,0x08,0x0F,0x0F,0x08,0x08,0x00,
    0x08,0x0C,0x84,0xC4,0x64,0x3C,0x18,0x00,    // -2-
    0x0E,0x0F,0x09,0x08,0x08,0x0C,0x0C,0x00,
    0x08,0x0C,0x44,0x44,0x44,0xFC,0xB8,0x00,    // -3-
    0x04,0x0C,0x08,0x08,0x08,0x0F,0x07,0x00,
    0xC0,0xE0,0xB0,0x98,0xFC,0xFC,0x80,0x00,    // -4-
    0x00,0x00,0x00,0x08,0x0F,0x0F,0x08,0x00,
    0x7C,0x7C,0x44,0x44,0xC4,0xC4,0x84,0x00,    // -5-
```

```
    0x04,0x0C,0x08,0x08,0x08,0x0F,0x07,0x00,
    0xF0,0xF8,0x4C,0x44,0x44,0xC0,0x80,0x00,    // -6-
    0x07,0x0F,0x08,0x08,0x08,0x0F,0x07,0x00,
    0x0C,0x0C,0x04,0x84,0xC4,0x7C,0x3C,0x00,    // -7-
    0x00,0x00,0x0F,0x0F,0x00,0x00,0x00,0x00,
    0xB8,0xFC,0x44,0x44,0x44,0xFC,0xB8,0x00,    // -8-
    0x07,0x0F,0x08,0x08,0x08,0x0F,0x07,0x00,
    0x38,0x7C,0x44,0x44,0x44,0xFC,0xF8,0x00,    // -9-
    0x00,0x08,0x08,0x08,0x0C,0x07,0x03,0x00,
    0x20,0xE0,0xC0,0x60,0x20,0x60,0xC0,0x00,    // -r-
    0x08,0x0F,0x0F,0x08,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x80,0xC0,0x60,0x30,0x00,    // -/-
    0x0C,0x06,0x03,0x01,0x00,0x00,0x00,0x00,
    0xE0,0xE0,0x60,0xC0,0x60,0xE0,0xC0,0x00,    // -m-
    0x0F,0x0F,0x00,0x0F,0x00,0x0F,0x0F,0x00,
    0x00,0x00,0x20,0xEC,0xEC,0x00,0x00,0x00,    // -i-
    0x00,0x00,0x08,0x0F,0x0F,0x08,0x00,0x00,
    0x20,0xE0,0xC0,0x20,0x20,0xE0,0xC0,0x00,    // -n-
    0x00,0x0F,0x0F,0x00,0x00,0x0F,0x0F,0x00,
};
unsigned char pwm_val=100; //占空比设置值
/**********************电动机控制*******************/
sbit ZPort = P2^7;//配置 P2.7 为电动机正端
sbit FPort = P2^6;//配置 P2.6 为电动机负端
void M_Zheng()//电动机正转
{
    ZPort=1;//电动机正端和电源连接
    FPort=0;//电动机负端和地连接
}
void M_Stop()//电动机停转
{
    ZPort=0;//电动机正端和地连接
    FPort=0;//电动机负端和地连接
}
/***************按键子程序************************/
unsigned char key_value;//按键键值
sbit S1 = P2^0;//配置按键 1 为 P2.0
```

```
sbit S2 = P2^1;//配置按键 2 为 P2.1
sbit S3 = P2^2;//配置按键 3 为 P2.2
sbit S4 = P2^3;//配置按键 4 为 P2.3
void KEY(void)//按键
{
    unsigned char key_time;
    S1=S2=S3=S4=1;//将按键口全置 1
    if(!S1||!S2||!S3||!S4)//判断是否有按键按下
    {
        if(++key_time==100)//计时，防抖动
        {
            if(!S1)key_value=1;//赋值键值为 1
            if(!S2)key_value=2;//赋值键值为 2
            if(!S3)key_value=3;//赋值键值为 3
            if(!S4)key_value=4;//赋值键值为 4
        }
    }
    else key_time=0;//计时清零
}
void Read_KEY()//读键值
{
    if(key_value)//判断是否有按键按下
    {
        switch (key_value)
        {
            case 1:M_Zheng();
                   pwm_val=100;//赋值占空比的初始值
                    TR0=1;          //打开定时器
                   break;//电动机正转
            case 2:
                    pwm_val++;   //加大占空比 电动机加速
                    if(pwm_val==101)//占空比最大为 100 占空比限制
                    pwm_val=100;
                    break;//
            case 3:
                    pwm_val--;//减小占空比 电动机减速
                    if(pwm_val==255)//占空比最小为 0 占空比限制
```

```
                        pwm_val=0;
                        break;//电动机加速
                break;//电动机减速
                case 4:M_Stop();
                        TR0=0;
                       break;//电动机停止
                default:   break;
            }
            key_value=0;//键值清零
        }
}
void INIT_TIME0(void)//初始化定时器 0
{
    TMOD |= 0x02;//使用定时器模式 2 自动加载
    TH0=0xA1; //定时时间 0.1mS
    TL0=0xA1;
    ET0=1;      //允许定时器 0 中断
}
void INIT_TIME1(void)//初始化定时器 1
{
    TMOD |= 0x10;//使用定时器模式 2 自动加载
    TH1=0x4C; //定时时间 50mS
    TL1=0x00;
    ET1=1;      //允许定时器 1 中断
    TR1=1;      //开始计数
}
void INIT_INIT1(void)//初始化外部中断 1
{
    IT1=1;      //下降沿触发中断
    EX1=1;      //允许外部中断 1 中断
}
bit flag1s=0;//1S 钟标记为用来每秒刷新液晶一次
unsigned int zhuansu=0;//电动机转速
unsigned char InitCount=0;//编码器计数值
void main(void)//主函数
{
    INIT_TIME0();
```

```
    INIT_TIME1();
    INIT_INIT1();
    EA=1;
    LCD_INIT();//液晶初始化
    write_HZ(2,16*0+8,&HZ[0]);//电
    write_HZ(2,16*1+8,&HZ[1]);//机
    write_HZ(2,16*2+8,&HZ[2]);//当
    write_HZ(2,16*3+8,&HZ[3]);//前
    write_HZ(2,16*4+8,&HZ[4]);//转
    write_HZ(2,16*5+8,&HZ[5]);//速
    write_HZ(2,16*6+8,&HZ[6]);//为
    write_ASC(4,16*1+8,&ASC[0]);//0
    write_ASC(4,16*2,&ASC[0]);//0
    write_ASC(4,16*2+8,&ASC[0]);//0
    write_ASC(4,16*3,&ASC[10]);//r
    write_ASC(4,16*3+8,&ASC[11]);///
    write_ASC(4,16*4,&ASC[12]);//m
    write_ASC(4,16*4+8,&ASC[13]);//i
    write_ASC(4,16*5,&ASC[14]);//n
    while(1)
    {
        KEY();
        Read_KEY();//读按键值执行相应的动作
        if(flag1s==1)
        {
            flag1s=0;
            write_ASC(4,16*1+8,&ASC[zhuansu/100%10]);//转速百位
            write_ASC(4,16*2,&ASC[zhuansu/10%10]);//转速十位
            write_ASC(4,16*2+8,&ASC[zhuansu%10]);//转速个位
        }
    }
}
void timer0(void) interrupt 1 //定时器中断
{
    static unsigned char count=0;//波形的频率控制计数值
    count++;                    //计数值加一
    if(count==pwm_val)          //判断计数值是否到达占空比的值
```

```
    {
        ZPort=0;                //把波形置 0
    }
    if(count==100)              //判断一个波形是否完成  100HZ 波形
    {
        count=0;                //计数值清零
        ZPort=1;                //把波形置 1
    }
}
void timer1(void) interrupt 3 //定时器中断
{
    static unsigned char i=0;
    TH1=0x4C; //定时时间 50mS
    TL1=0x00;
    i++;                        //计数值加一
    if(i==20)           //一秒钟时间
    {
        i=0;            //计数清零
        zhuansu=InitCount;//获取编码器的计数值
        InitCount=0;    //清空编码器的计数值从新计数
        zhuansu=zhuansu*60;
//获取的编码器值是一秒中的 转换成一份钟计数值
        zhuansu=zhuansu/4;//编码器每计数 4 个表示电动机转动一圈
        flag1s=1;
    }

}
void init0(void) interrupt 2 //定时器中断
{
    InitCount++;
}
```

LCD12864 驱动头文件库程序

LCD.H

```
#include<reg51.h>                   // 包含 51 单片机头文件
/******12864LCD 引脚定义******/
#define DATAPORT   P0               // P0 端口为 LCD 数据总线
sbit CS1=P1^3;                      // 位定义 P2.0，CS1 为左半屏片选信号
```

```
sbit CS2=P1^4;                  // 位定义 P2.1，CS2 为右半屏片选信号
sbit E=P1^2;                    // E 为锁存使能,下降沿锁存数据
sbit RS=P1^0;                   // 数据/指令选择，0 为数据，1 为指令
sbit RW=P1^1;                   // 读/写选择，0 为写，1 为读
sbit RST=P1^5;                  // TG12864 复位信号
sbit BL=P1^6;                   // 背光选择
/******毫秒级延时函数******/
void delay_ms(unsigned int ms)
{
   unsigned char i;                    //临时延时变量 i
   while(--ms)                  //减 1 判断是否为 0
   {
       for(i=0;i<125;i++);       //加 1 判断是否达到 125
   }
}
void selectscreen(unsigned char screen)//选屏  0:全屏  1:左半屏  2:右半屏
{
      switch (screen)
      {
            case 0:CS1=1;CS2=1;     break;          //全屏
            case 1:CS1=1;CS2=0;     break;          //左半屏
            case 2:CS1=0;CS2=1;     break;          //右半屏
            default:   break;
      }
}
void write_cmd(unsigned char dat)           //写命令
{
      RW=0;          //写操作
      RS=0;          //写命令
      DATAPORT=dat;     //P0 口传输命令
      E=1;
      E=0;       //下降沿(使能)，将命令送入
}
void write_data(unsigned char dat)          //写数据
{
      RW=0;          //写操作
      RS=1;          //写数据
```

```
    DATAPORT=dat;     //P0 口传输数据
    E=1;
    E=0;      //下降沿(使能)，将数据送入
    RS=0;          //置为写命令状态，减少噪点发生率
}
void check_busy_12864()//忙检测函数
{
        uchar i=0,dat;                  //定义一个临时变量
        RS=0;                           //对指令操作
        RW=1;                           //读操作
        do{                             //先执行下面指令
          DATAPORT=0x00;                //口线置 1 防止干扰
          E=1;                          //下降沿的高电平部分
          dat=DATAPORT;                 //读 LCD 总线上的数据到临时变量
          E=1;                          //短暂延时
          E=0;
          dat=0x80&dat;        //取出数据的 D7 位，该位是忙检测(busy)信号
          i++;
       }while((dat==0x80)&&(i<100));//LCD 内部不忙或超时，退出
}
void set_onoff(unsigned char onoff)          //开关
{
    onoff|=0x3e;        //控制字
    write_cmd(onoff);  //写命令,将控制字写入
}

void set_page(unsigned char page)            //选行
{
    page&=0x07;              //限位
    page|=0xb8;              //控制字
    write_cmd(page);   //写命令,将控制字写入
}
void set_column(unsigned char column)      //选列
{
    if(column>63)selectscreen(2);      //当列<63 时，则选择左半屏
    else selectscreen(1);            //当列超过左半屏时，则选择右半屏
    column&=0x3f;            //限位
```

```
    column|=0x40;            //控制字
    write_cmd(column);      //写命令,将控制字写入
}
void set_startline(unsigned char line)    //选起始行
{
    line&=0x3f;             //限位
    line|=0xc0;             //控制字
    write_cmd(line);    //写命令,将控制字写入
}

void clearnscreen() //清屏
{
    unsigned char i,j;
    selectscreen(0);     //选屏
    for (i=0;i<8;i++)
    {
        set_page(i);        //选行
        for (j=0;j<64;j++)    //列会自动加 1
        {
            write_data(0); //写数据
        }
    }
}
void LCD_INIT()//LCD 初始化
{
    BL=0;                   //背光
    RST=0;                  //复位
    delay_ms(10);           //延时等待
    RST=1;                  //置位
    delay_ms(10);           //延时等待
    set_onoff(1);           //开关
    set_startline(0);       //起始行
    clearnscreen();         //清屏
}
```

4、程序说明

本程序通过按键来控制电动机的转速。然后使用外部中断 1 进行计数，当一秒中时间到了。我们从中断读出计数值进行处理从而得到电动机的转速。最后通过液晶把转速显示出来。

5、任务实施

5.1 建立工程

打开 keil 软件，通过菜单“Project”，新建立一个工程”new Project”项目 DJCS，然后再建一个文件名为 DJCS.C 的源程序文件，将上面的参考程序输入并保存。建立的 DJCS.C 文件添加入本项目中。

5.2 编译并生成 HEX

单击”Build target”按钮，对源程序进行编译和链接，产生 HEX 文件。

5.3 烧录芯片

打开 STC-ISP 下载软件，选中单片机型号为“STC90C52RC”,选择最低波特率为 2400 最高波特率为 115200（为默认值一般不需修改）。点击打开程序文件按钮，在刚才建立的 DJCS 文件夹中生成的 DJCS.HEX 文件。然后点击“下载/编程”按钮。最后给最小系统通电，等待电脑端串口与单片机握手成功。就能把程序载入到单片机中去了。其详细操作图如图 4- 44 所示：

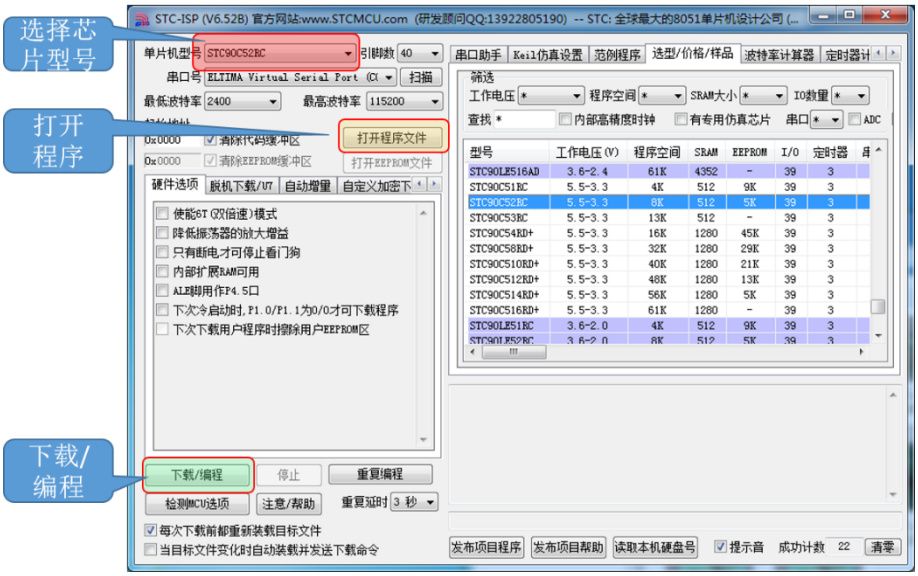

图 4- 44 STC 单片机程序烧写示意图

5.4 硬件调试

程序下载完成后，把单片机最小系统接入电路中，调试电路使其能正常工作，并在 LCD12864 上显示电机当前的转速。

四、任务评价

表 4- 10　任务五完成情况评价表

项目名称			评价时间		年　月　日		
小组名称		小组成员					
评价内容	评价要求	权重	评价标准	学生自评得分	小组评价得分	教师评价得分	合计
职业与安全意识	1. 工具摆放、操作符合安全操作规程；2. 遵守纪律、爱惜设备和器材、工位整洁；3. 具有团队协作精神。	10%	好（10） 较好（8） 一般（6） 差（<6）				
印制线路板调试	1. 能够根据任务要求进行印制线路板的调试；2. 根据任务要求，能够对对应电路部分进行硬件电路的检查与故障检修。	15%	好（15） 较好（12） 一般（9） 差（<9）				印制线路板调试
项目功能测试	1. 编写的程序能成功编译；2. 程序能正确烧写到芯片中；3. 通过按键能够使电动机进行调速，同时可以在液晶屏显示出电动机的当前转速。	60%	好（60） 较好（45） 一般（30） 差（<30）				
问题与思考	1. 如何显示出电动机转速的小数位 2. 除了使用外部中断进行测速还可以使用什么来测速 3. 如何提高测速的精度	15%	好（15） 较好（12） 一般（9） 差（<6）				
教师签名			学生签名			总分	
项目评价=学生自评（0.2）+小组评价（0.3）+教师评价（0.5）							

五、知识拓展

1、霍尔传感器测速

1.1 霍尔传感器

霍尔传感器是根据霍尔效应制作的一种磁场传感器。霍尔效应是磁电效应的一种，这一现象是霍尔（A.H.Hall，1855—1938）于 1879 年在研究金属的导电机构时发现的。后来发现半导体、导电流体等也有这种效应，而半导体的霍尔效应比金属强得多，利用这现象制成的各种霍尔元件，广泛地应用于工业自动化技术、检测技术及信息处理等方面。霍尔效应是研究半导体材料性能的基本方法。通过霍尔效应实验测定的霍尔系数，能够判断半导体材料的导电类型、载流子浓度及载流子迁移率等重要参数。

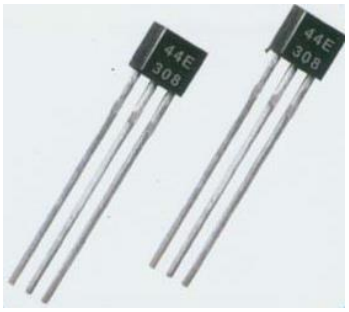

图 4- 45　霍尔传感器

1.2 霍尔传感器测速原理

霍尔传感器测速原理就是使用霍尔传感器获得脉冲信号。脉冲获得很简单，只要在电动机的转轴粘上一粒磁钢，让霍尔传感器开关靠近磁钢，就会有信号输出，电动机转动时，就会不断地产生脉冲信号输出。这样可以通过测量输出的信号的频率，就可以得到电机的转速了。电机的转速就等于输出信号的频率。

六、思考与练习

1．使用洗衣机模拟装置硬件电路完成任务五《直流电动机转速测量》的制作。

2．考虑一下，如何使用定时器的计数功能来测量电动机转速？

3．思考怎样提高电动机转速测量的精度。

任务六　模拟洗衣机控制器制作

一、任务要求

完成洗衣机模拟装置的程序的编写具体要求如下：

通电后显示如图 4- 46 所示的液晶初始界面。

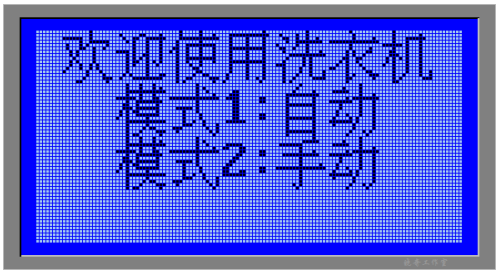

图 4- 46 洗衣机初始界面

洗衣机有两种模式一种是自动模式一种是手动模式。通过按键 S1 和 S2 来进行选择。其中 S1 为自动模式按键，S2 为手动模式按键。

当按下 S1 按键时就进入了自动模式了。当按下 S2 按键时就进入了手动模式了。下面先来看一下自动模式。

1、自动模式

自动模式一共包括四个步骤：进水、洗涤、排水、脱水四个步骤。

1.1 进水

液晶屏显示如图 4- 47 所示界面，剩余时间从 10 秒开始倒计时。当时间剩余 0 秒时液晶显示进水完成等待 2 秒进入第二步骤洗涤。

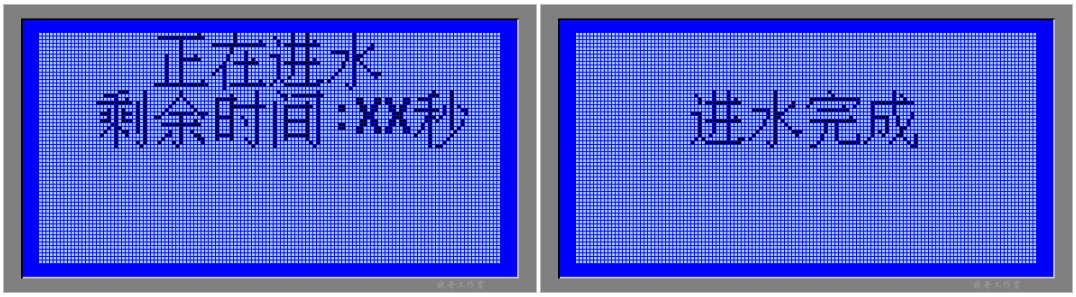

图 4- 47 洗衣机进水页面

1.2 洗涤

液晶屏显示如图 4- 48 所示界面，剩余时间从 20 秒开始倒计时。同时电动机实现正转 1 秒反转 1 秒。当时间剩余 0 秒时电动机停止，液晶显示洗涤完成等待 2 秒进入第三步骤排水。

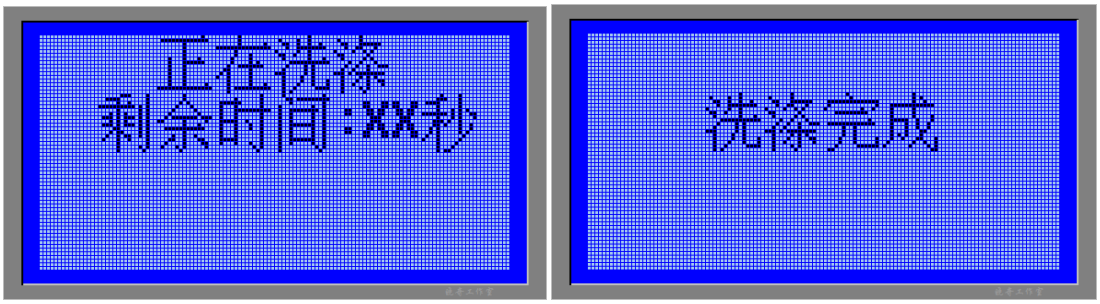

图 4- 48 洗衣机洗涤界面

1.3 排水

液晶屏显示如图 4- 49 所示界面，剩余时间从 10 秒开始倒计时。当时间剩余 0 秒时液晶显示排水完成等待 2 秒进入第四步骤脱水。

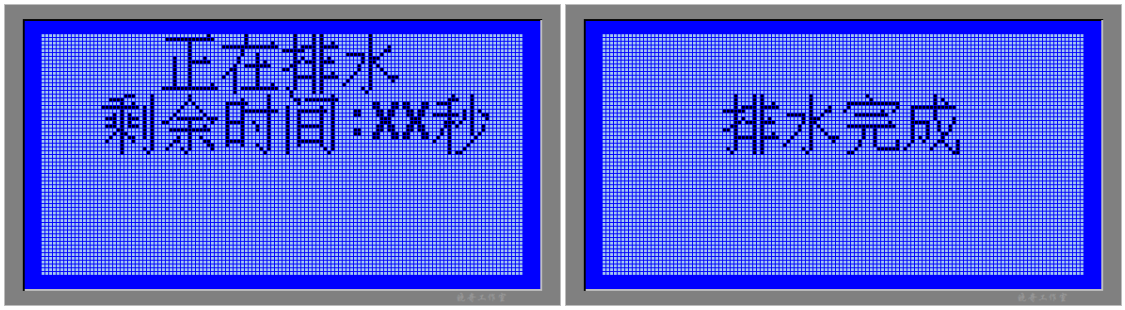

图 4- 49 洗衣机排水界面

1.4 脱水

液晶屏显示如图 4- 50 所示界面，剩余时间从 5 秒开始倒计时。同时电动机实现正转。时间剩余 0 秒时电动机停止，液晶显示脱水完成等待 2 秒返回到初始界面完成一次自动模式下的洗衣工作。

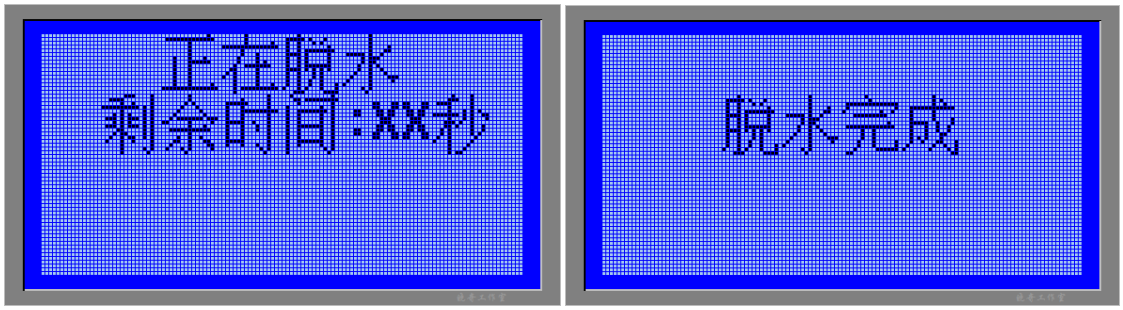

图 4- 50 洗衣机脱水界面

2、手动模式

手动模式是把自动模式分解四个步骤：进水、洗涤、排水、脱水。可以单独的选着其中的一个步骤就行操作。通过四个按键 S1、S2、S3、S4 来选择当前工作的状态。

2.1 当按下按键 S1 时进入进水的状态，液晶显示如图 4- 47 所示界面，剩余时间从 10 秒开始倒计时。当时间剩余 0 秒时液晶显示进水完成等待 2 秒返回到初始界面。完成一次手动状态下的进水。

2.2 当按下按键 S2 时进入洗涤的状态，液晶显示如图 4- 48 所示界面，剩余时间从 20 秒开始倒计时。同时电动机实现正转 1 秒反转 1 秒。当时间剩余 0 秒时电动机停止，液晶显示洗涤完成等待 2 秒返回到初始界面。完成一次手动状态下的洗涤。

2.3 当按下按键 S3 时进入排水的状态，液晶显示如图 4- 49 所示界面，剩余时间从 10 秒开始倒计时。当时间剩余 0 秒时液晶显示排水完成等待 2 秒返回到初始界面。完成一次手动状态下的排水。

2.4 当按下按键 S4 时进入脱水的状态，液晶显示如图 4- 50 所示界面，剩余时间从 5 秒开始倒计时。同时电动机实现正转。时间剩余 0 秒时电动机停止，液晶显示脱水完成等待 2 秒返回到初始界面。完成一次手动状态下的脱水。

二、操作训练

1、任务分析

这个任务需要使用到洗衣机模拟装置的所有模块的。包括液晶，电动机及其驱动，按键。

这个任务所有的程序都是通过按键来触发的。所以在主程序中只要对按键进行判断。当有按键按下时，执行相应的程序。比如按键 S1 按下就调用自动模式子程序。那么自动模式子程序就开始执行。执行完成后返回到初始界面。

2、主函数流程图

主函数流程图如图 4- 51 所示。

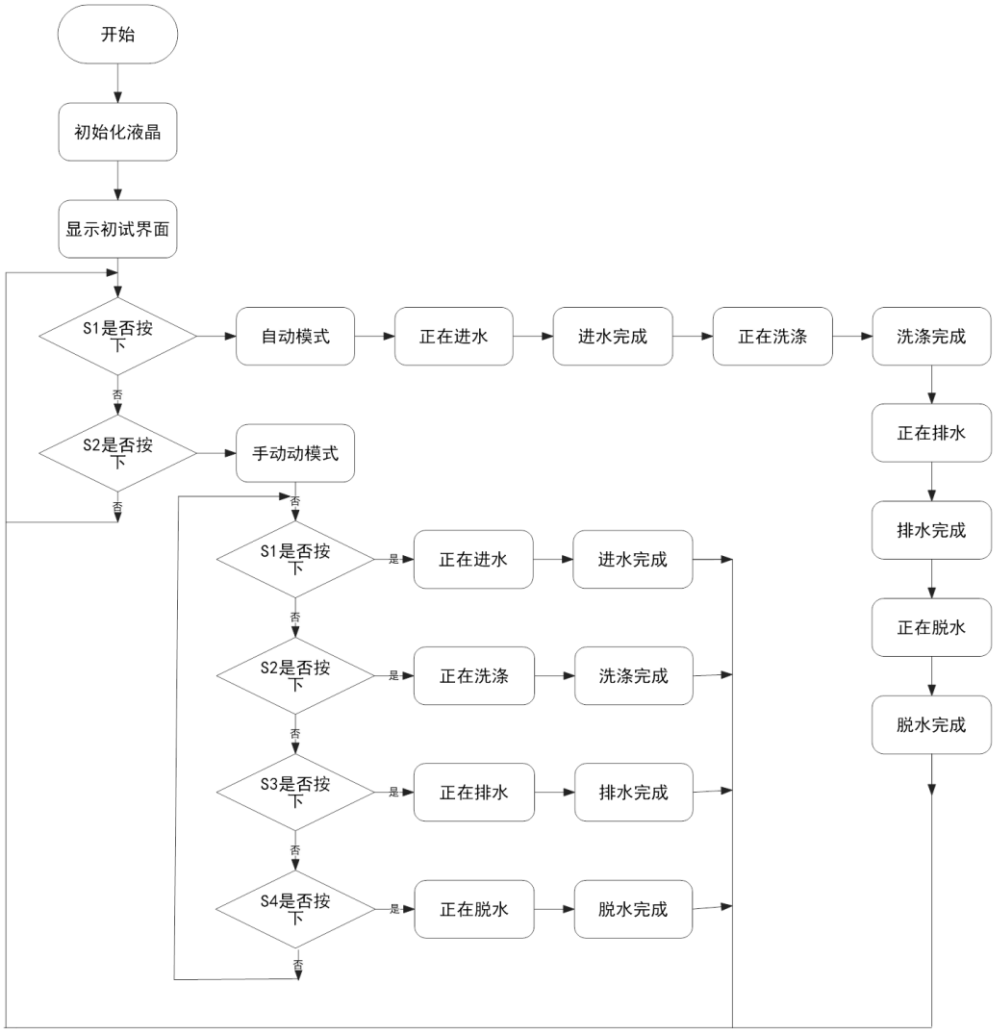

图 4- 51 洗衣机模拟装置流程图

3、参考程序

根据图 4- 51 洗衣机模拟装置流程图编写程序：

洗衣机模拟装置程序

```
#include<reg51.h>                    // 包含 51 单片机头文件
#include<lcd.h>                      // 包含液晶子程序头文件
unsigned char code HZ[27][32]=
{
/*--  文字:  欢  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x14,0x24,0x44,0x84,0x64,0x1C,0x20,0x18,0x0F,0xE8,0x08,0x08,0x28,0x18,0x08,0x00,
0x20,0x10,0x4C,0x43,0x43,0x2C,0x20,0x10,0x0C,0x03,0x06,0x18,0x30,0x60,0x20,0x00,
/*--  文字:  迎  --*/
```

```
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x40,0x41,0xCE,0x04,0x00,0xFC,0x04,0x02,0x02,0xFC,0x04,0x04,0x04,0xFC,0x00,0x00,
0x40,0x20,0x1F,0x20,0x40,0x47,0x42,0x41,0x40,0x5F,0x40,0x42,0x44,0x43,0x40,0x00,
/*--  文字:  使  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x40,0x20,0xF0,0x1C,0x07,0xF2,0x94,0x94,0x94,0xFF,0x94,0x94,0x94,0xF4,0x04,0x00,
0x00,0x00,0x7F,0x00,0x40,0x41,0x22,0x14,0x0C,0x13,0x10,0x30,0x20,0x61,0x20,0x00,
/*--  文字:  用  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x00,0x00,0xFE,0x22,0x22,0x22,0x22,0xFE,0x22,0x22,0x22,0x22,0xFE,0x00,0x00,
0x80,0x40,0x30,0x0F,0x02,0x02,0x02,0x02,0xFF,0x02,0x02,0x42,0x82,0x7F,0x00,0x00,
/*--  文字:  洗  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x10,0x61,0x06,0xF0,0xA0,0x98,0x8E,0x88,0x88,0xFF,0x88,0x88,0x88,0x80,0x80,0x00,
0x04,0x04,0xFF,0x00,0x40,0x20,0x18,0x07,0x00,0x00,0x3F,0x40,0x40,0x40,0x70,0x00,
/*--  文字:  衣  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x04,0x04,0x04,0x04,0x84,0x44,0x25,0x3E,0xC4,0x04,0x04,0x84,0x64,0x44,0x04,0x00,
0x08,0x04,0x02,0x41,0xFF,0x40,0x20,0x00,0x00,0x03,0x0D,0x10,0x20,0x60,0x20,0x00,
/*--  文字:  机  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x08,0x08,0xC8,0xFF,0x48,0x88,0x08,0x00,0xFE,0x02,0x02,0x02,0xFE,0x00,0x00,0x00,
0x04,0x03,0x00,0xFF,0x00,0x41,0x30,0x0C,0x03,0x00,0x00,0x00,0x3F,0x40,0x78,0x00,
/*--  文字:  模  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x10,0xD0,0xFF,0x50,0x90,0x04,0xF4,0x54,0x5F,0x54,0x54,0x5F,0xF4,0x04,0x00,0x00,
0x03,0x00,0xFF,0x00,0x00,0x84,0x85,0x45,0x35,0x0F,0x15,0x25,0x65,0xC4,0x44,0x00,
/*--  文字:  式  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x08,0x88,0x88,0x88,0x88,0x88,0x08,0xFF,0x08,0x09,0x0E,0x0A,0x08,0x00,0x00,
0x00,0x20,0x60,0x30,0x1F,0x10,0x08,0x08,0x00,0x07,0x18,0x20,0x40,0x80,0x70,0x00,
/*--  文字:  自  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x00,0x00,0xF8,0x48,0x48,0x4C,0x4B,0x4A,0x48,0x48,0x48,0xF8,0x00,0x00,0x00,
0x00,0x00,0x00,0xFF,0x44,0x44,0x44,0x44,0x44,0x44,0x44,0x44,0xFF,0x00,0x00,0x00,
/*--  文字:  动  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
```

```
0x20,0x24,0x24,0xE4,0x24,0x24,0x24,0x20,0x10,0x10,0xFF,0x10,0x10,0xF0,0x00,0x00,
0x08,0x1C,0x0B,0x08,0x0C,0x05,0x4E,0x24,0x10,0x0C,0x03,0x20,0x40,0x3F,0x00,0x00,
/*--  文字:  手  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x24,0x24,0x24,0x24,0x24,0x24,0xFE,0x22,0x22,0x22,0x22,0x22,0x20,0x00,0x00,
0x02,0x02,0x02,0x02,0x02,0x42,0x82,0x7F,0x02,0x02,0x02,0x02,0x02,0x02,0x02,0x00,
/*--  文字:  进  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x80,0x82,0x9C,0x88,0x00,0x88,0x88,0xFF,0x88,0x88,0x88,0xFF,0x88,0x88,0x80,0x00,
0x00,0x40,0x20,0x1F,0x20,0x50,0x4C,0x43,0x40,0x40,0x40,0x5F,0x40,0x40,0x40,0x00,
/*--  文字:  水  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x10,0x10,0x10,0x90,0x70,0x00,0xFF,0x20,0x60,0x90,0x08,0x04,0x00,0x00,0x00,
0x10,0x10,0x08,0x06,0x01,0x40,0x80,0x7F,0x00,0x00,0x01,0x06,0x0C,0x18,0x08,0x00,
/*--  文字:  涤  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x10,0x60,0x02,0xEC,0x00,0x20,0x90,0x8C,0x57,0xA4,0x54,0x4C,0x84,0x80,0x80,0x00,
0x04,0x04,0xFF,0x00,0x41,0x21,0x1A,0x42,0x82,0x7F,0x02,0x0A,0x12,0x62,0x00,0x00,
/*--  文字:  排  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x10,0x10,0x10,0xFF,0x90,0x40,0x10,0xFE,0x00,0x00,0xFE,0x10,0x10,0x10,0x10,0x00,
0x04,0x44,0x82,0x7F,0x08,0x09,0x09,0xFF,0x00,0x00,0xFF,0x09,0x09,0x09,0x08,0x00,
/*--  文字:  脱  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x00,0xFE,0x12,0x12,0xFE,0x00,0xF9,0x8E,0x8A,0x88,0x8C,0x8B,0xFA,0x00,0x00,
0x40,0x38,0x07,0x21,0x41,0x3F,0x80,0x40,0x30,0x0F,0x00,0x3F,0x40,0x40,0x70,0x00,
/*--  文字:  正  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x02,0x02,0xC2,0x02,0x02,0x02,0x02,0xFE,0x82,0x82,0x82,0x82,0x82,0x02,0x00,
0x20,0x20,0x20,0x3F,0x20,0x20,0x20,0x20,0x3F,0x20,0x20,0x20,0x20,0x20,0x20,0x00,
/*--  文字:  在  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x04,0x04,0xC4,0x64,0x9C,0x87,0x84,0x84,0xE4,0x84,0x84,0x84,0x84,0x04,0x00,
0x04,0x02,0x01,0x7F,0x00,0x20,0x20,0x20,0x20,0x3F,0x20,0x20,0x20,0x20,0x20,0x00,
/*--  文字:  秒  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x12,0x12,0xD2,0xFE,0x91,0x11,0xC0,0x38,0x10,0x00,0xFF,0x00,0x08,0x10,0x60,0x00,
```

```
0x04,0x03,0x00,0xFF,0x00,0x83,0x80,0x40,0x40,0x20,0x23,0x10,0x08,0x04,0x03,0x00,
/*--  文字:  剩  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x08,0x2A,0xAA,0xFA,0x0A,0xFE,0x09,0xF9,0x49,0xA8,0x00,0xF8,0x00,0x00,0xFF,0x00,
0x21,0x11,0x08,0x05,0x02,0x7F,0x02,0x04,0x19,0x09,0x00,0x07,0x20,0x40,0x3F,0x00,
/*--  文字:  余  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x40,0x40,0x20,0x20,0x30,0x28,0x24,0xE3,0x24,0x28,0x28,0x30,0x20,0x60,0x20,0x00,
0x00,0x40,0x21,0x1D,0x09,0x41,0x81,0x7F,0x01,0x05,0x09,0x39,0x11,0x01,0x00,0x00,
/*--  文字:  时  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0xFC,0x44,0x44,0x44,0xFC,0x10,0x90,0x10,0x10,0x10,0xFF,0x10,0x10,0x10,0x00,
0x00,0x07,0x04,0x04,0x04,0x07,0x00,0x00,0x03,0x40,0x80,0x7F,0x00,0x00,0x00,0x00,
/*--  文字:  间  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0xF8,0x01,0x06,0x00,0xF0,0x92,0x92,0x92,0x92,0xF2,0x02,0x02,0xFE,0x00,0x00,
0x00,0xFF,0x00,0x00,0x00,0x07,0x04,0x04,0x04,0x04,0x07,0x40,0x80,0x7F,0x00,0x00,
/*--  文字:  完  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x90,0x8C,0xA4,0xA4,0xA4,0xA5,0xA6,0xA4,0xA4,0xA4,0xA4,0x94,0x8C,0x04,0x00,
0x00,0x80,0x40,0x20,0x18,0x07,0x00,0x00,0x00,0x3F,0x40,0x40,0x40,0x70,0x00,0x00,
/*--  文字:  成  --*/
/*--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16    --*/
0x00,0x00,0xF8,0x48,0x48,0x48,0xC8,0x08,0xFF,0x08,0x09,0x0A,0xC8,0x88,0x08,0x00,
0x40,0x30,0x0F,0x00,0x08,0x50,0x4F,0x20,0x10,0x0B,0x0C,0x12,0x21,0x40,0xF0,0x00,
};//欢迎使用洗衣机模式自动手进水涤排脱正在秒剩余时间完成
unsigned char code ASC[11][16]=
{
      0xF8,0xFC,0x04,0xC4,0x24,0xFC,0xF8,0x00,    // -0-
      0x07,0x0F,0x09,0x08,0x08,0x0F,0x07,0x00,
      0x00,0x10,0x18,0xFC,0xFC,0x00,0x00,0x00,    // -1-
      0x00,0x08,0x08,0x0F,0x0F,0x08,0x08,0x00,
      0x08,0x0C,0x84,0xC4,0x64,0x3C,0x18,0x00,    // -2-
      0x0E,0x0F,0x09,0x08,0x08,0x0C,0x0C,0x00,
      0x08,0x0C,0x44,0x44,0x44,0xFC,0xB8,0x00,    // -3-
      0x04,0x0C,0x08,0x08,0x08,0x0F,0x07,0x00,
      0xC0,0xE0,0xB0,0x98,0xFC,0xFC,0x80,0x00,    // -4-
```

```
    0x00,0x00,0x00,0x08,0x0F,0x0F,0x08,0x00,
    0x7C,0x7C,0x44,0x44,0xC4,0xC4,0x84,0x00,    // -5-
    0x04,0x0C,0x08,0x08,0x08,0x0F,0x07,0x00,
    0xF0,0xF8,0x4C,0x44,0x44,0xC0,0x80,0x00,    // -6-
    0x07,0x0F,0x08,0x08,0x08,0x0F,0x07,0x00,
    0x0C,0x0C,0x04,0x84,0xC4,0x7C,0x3C,0x00,    // -7-
    0x00,0x00,0x0F,0x0F,0x00,0x00,0x00,0x00,
    0xB8,0xFC,0x44,0x44,0x44,0xFC,0xB8,0x00,    // -8-
    0x07,0x0F,0x08,0x08,0x08,0x0F,0x07,0x00,
    0x38,0x7C,0x44,0x44,0x44,0xFC,0xF8,0x00,    // -9-
    0x00,0x08,0x08,0x08,0x0C,0x07,0x03,0x00,
    0x00,0x00,0x00,0x30,0x30,0x00,0x00,0x00,    // -:-
    0x00,0x00,0x00,0x06,0x06,0x00,0x00,0x00,
};
#define Input_water_time      10
#define Washing_time          20
#define Output_water_time     10
#define Dehydration_time      5
#define show_ok_time          2
unsigned char flag;//工作状态
unsigned char mode;//工作模式
/**********************LCD 显示界面******************/
void show_welcome()//显示"欢迎使用洗衣机"
{
    write_HZ(0,16*0+8,&HZ[0]);//欢
    write_HZ(0,16*1+8,&HZ[1]);//迎
    write_HZ(0,16*2+8,&HZ[2]);//使
    write_HZ(0,16*3+8,&HZ[3]);//用
    write_HZ(0,16*4+8,&HZ[4]);//洗
    write_HZ(0,16*5+8,&HZ[5]);//衣
    write_HZ(0,16*6+8,&HZ[6]);//机
}
void show_auto()//显示"模式 1:自动"
{
    write_HZ(2,16*1+8,&HZ[7]);//模
    write_HZ(2,16*2+8,&HZ[8]);//式
    write_ASC(2,16*3+8,&ASC[1]);//1
```

```
    write_ASC(2,16*4,&ASC[10]);//:
    write_HZ(2,16*4+8,&HZ[9]);//自
    write_HZ(2,16*5+8,&HZ[10]);//动
}
void show_manu()//显示"模式 2:手动"
{
    write_HZ(4,16*1+8,&HZ[7]);//模
    write_HZ(4,16*2+8,&HZ[8]);//式
    write_ASC(4,16*3+8,&ASC[2]);//2
    write_ASC(4,16*4,&ASC[10]);//:
    write_HZ(4,16*4+8,&HZ[11]);//自
    write_HZ(4,16*5+8,&HZ[10]);//动
}
void show_menu()//显示菜单主界面
{
    show_welcome();//显示"欢迎使用洗衣机"
    show_auto();//显示"模式 1:自动"
    show_manu();//显示"模式 2:手动"
}
void show_Input_water()//显示"正在进水"
{
    write_HZ(0,16*2,&HZ[17]);//正
    write_HZ(0,16*3,&HZ[18]);//在
    write_HZ(0,16*4,&HZ[12]);//进
    write_HZ(0,16*5,&HZ[13]);//水
}
void show_Washing()//显示"正在洗涤"
{
    write_HZ(0,16*2,&HZ[17]);//正
    write_HZ(0,16*3,&HZ[18]);//在
    write_HZ(0,16*4,&HZ[04]);//洗
    write_HZ(0,16*5,&HZ[14]);//涤
}
void show_Output_water()//显示"正在排水"
{
    write_HZ(0,16*2,&HZ[17]);//正
    write_HZ(0,16*3,&HZ[18]);//在
```

```
	write_HZ(0,16*4,&HZ[15]);//排
	write_HZ(0,16*5,&HZ[13]);//水
}
void show_Dehydration()//显示"正在脱水"
{
	write_HZ(0,16*2,&HZ[17]);//正
	write_HZ(0,16*3,&HZ[18]);//在
	write_HZ(0,16*4,&HZ[16]);//脱
	write_HZ(0,16*5,&HZ[13]);//水
}
void show_time(unsigned char S)//显示"剩余时间:XX 秒" input:时间
{
	write_HZ(2,16*1,&HZ[20]);//剩
	write_HZ(2,16*2,&HZ[21]);//余
	write_HZ(2,16*3,&HZ[22]);//时
	write_HZ(2,16*4,&HZ[23]);//间
	write_ASC(2,16*5,&ASC[10]);//:
	write_ASC(2,16*5+8,&ASC[S/10%10]);//秒十位
	write_ASC(2,16*6,&ASC[S%10]);//秒个位
	write_HZ(2,16*6+8,&HZ[19]);//秒
}
void show_Input_water_ok()//显示"进水完成"
{
	clearnscreen(); //清屏
	write_HZ(2,16*2,&HZ[12]);//进
	write_HZ(2,16*3,&HZ[13]);//水
	write_HZ(2,16*4,&HZ[24]);//完
	write_HZ(2,16*5,&HZ[25]);//成
}
void show_Washing_ok()//显示"洗涤完成"
{
	clearnscreen(); //清屏
	write_HZ(2,16*2,&HZ[04]);//洗
	write_HZ(2,16*3,&HZ[14]);//涤
	write_HZ(2,16*4,&HZ[24]);//完
	write_HZ(2,16*5,&HZ[25]);//成
}
```

```
void show_Output_water_ok()//显示"排水完成"
{
	clearnscreen(); //清屏
	write_HZ(2,16*2,&HZ[15]);//排
	write_HZ(2,16*3,&HZ[13]);//水
	write_HZ(2,16*4,&HZ[24]);//完
	write_HZ(2,16*5,&HZ[25]);//成
}
void show_Dehydration_ok()//显示"脱水完成"
{
	clearnscreen(); //清屏
	write_HZ(2,16*2,&HZ[16]);//脱
	write_HZ(2,16*3,&HZ[13]);//水
	write_HZ(2,16*4,&HZ[24]);//完
	write_HZ(2,16*5,&HZ[25]);//成
}
void show_mode()//显示模式
{
	write_HZ(0,16*2,&HZ[12]);//进
	write_HZ(0,16*3,&HZ[13]);//水
	write_HZ(0,16*4,&HZ[7]);//模
	write_HZ(0,16*5,&HZ[8]);//式

	write_HZ(2,16*2,&HZ[4]);//洗
	write_HZ(2,16*3,&HZ[14]);//涤
	write_HZ(2,16*4,&HZ[7]);//模
	write_HZ(2,16*5,&HZ[8]);//式

	write_HZ(4,16*2,&HZ[15]);//排
	write_HZ(4,16*3,&HZ[13]);//水
	write_HZ(4,16*4,&HZ[7]);//模
	write_HZ(4,16*5,&HZ[8]);//式

	write_HZ(6,16*2,&HZ[16]);//脱
	write_HZ(6,16*3,&HZ[13]);//水
	write_HZ(6,16*4,&HZ[7]);//模
	write_HZ(6,16*5,&HZ[8]);//式
```

```
}
/*************************************************************/
/**********************电动机控制*****************************/
sbit ZPort = P2^7;//配置 P2.7 为电动机正端
sbit FPort = P2^6;//配置 P2.6 为电动机负端
void M_Zheng()//电动机正转
{
	ZPort=1;//电动机正端和电源连接
	FPort=0;//电动机负端和地连接
}
void M_Fan()//电动机反转
{
	ZPort=0;//电动机正端和地连接
	FPort=1;//电动机负端和电源连接
}
void M_Stop()//电动机停转
{
	ZPort=0;//电动机正端和地连接
	FPort=0;//电动机负端和地连接
}
/*************************************************************/
/*************************************************************/
unsigned char time;//剩余时间
bit Input_water(unsigned char S)//进水   input:时间
{
	show_Input_water();//显示"正在进水"
	show_time(S);//显示"剩余时间:XX 秒"
	if(S==0)return 1;//时间为零时输出 1 跳转下一步
	else return 0;//保持此状态
}
bit Washing(unsigned char S)//洗涤 input:时间
{
	show_Washing();//显示"正在洗涤"
	show_time(S);//显示"剩余时间:XX 秒"
	if(S%2==0)M_Zheng();//电动机正转
	else M_Fan();//电动机反转
	if(S==0)
```

```
    {
        M_Stop();//电动机停转
        return 1;//时间为零时输出 1 跳转下一步
    }
    else return 0;//保持此状态
}
bit Output_water(unsigned char S)//排水input:时间
{
    show_Output_water();//显示"正在排水"
    show_time(S);//显示"剩余时间:XX 秒"
    if(S==0)return 1;//时间为零时输出 1 跳转下一步
    else return 0;//保持此状态
}
bit Dehydration(unsigned char S)//脱水   input:时间
{
    show_Dehydration();//显示"正在脱水"
    show_time(S);//显示"剩余时间:XX 秒"
    if(S%2==0)M_Zheng();//电动机正转
    else M_Fan();//电动机反转
    if(S==0)
    {
        M_Stop();//电动机停转
        return 1;//时间为零时输出 1 跳转下一步
    }
    else return 0;//保持此状态
}
/*****************************按键子程序*********************/
unsigned char key_value;//按键键值
sbit S1 = P2^0;//配置按键 1 为 P2.0
sbit S2 = P2^1;//配置按键 2 为 P2.1
sbit S3 = P2^2;//配置按键 3 为 P2.2
sbit S4 = P2^3;//配置按键 4 为 P2.3
void KEY()//按键
{
    unsigned char key_time;
    S1=S2=S3=S4=1;//将按键口全置 1
    if(!S1||!S2||!S3||!S4)//判断是否有按键按下
```

```
    {
        if(++key_time==10)//计时，防抖动
        {
            if(!S1)key_value=1;//赋值键值为 1
            if(!S2)key_value=2;//赋值键值为 2
            if(!S3)key_value=3;//赋值键值为 3
            if(!S4)key_value=4;//赋值键值为 4
        }
    }
    else key_time=0;//计时清零
}
void Read_KEY()//读键值
{
    if(key_value)//判断是否有按键按下
    {
        switch (key_value)
        {
            case 1:

    if(flag==2&&!mode)mode=1,time=Input_water_time,clearnscreen();
                if(flag==0)flag=1;break;//进水模式（自动）
            case2:
                if(flag==2&&!mode)mode=2,time=Washing_time,clearnscreen();
                if(flag==0)mode=0,flag=2,clearnscreen();break;//洗涤模式（手动）
            case3:

    if(flag==2&&!mode)mode=3,time=Output_water_time,clearnscreen();
                break;//排水模式
            case
4:if(flag==2&&!mode)mode=4,time=Dehydration_time,clearnscreen();
                break;//脱水模式
            default:    break;
        }
        key_value=0;//键值清零
    }
}
/************************工作**************************************/
```

```
void Read_flag()//读状态
{
	switch (flag)
	{
		case 0:	show_menu();break;//显示菜单主界面
		case 1:
			if(!mode)mode=1,time=Input_water_time,clearnscreen();
			break;//进入进水步骤
		case 2:	if(!mode)show_mode();break;//显示模式
		default:	break;
	}
}
void Work()//工作
{
	switch (mode)
	{
		case 1:
		if(Input_water(time))//判断进水是否到时
		{
			time=1;while(time);//等待 1 秒
			show_Input_water_ok();//显示"进水完成"
			time=show_ok_time;//延时两秒
			while(time);//等待计时为零
			if(flag==1)mode=2,time=Washing_time;//进入洗涤模式
			if(flag==2)mode=0,flag=0;//返回主菜单
			clearnscreen();//清屏
		}
		break;//进水模式
		case 2:
		if(Washing(time))//判断洗涤是否到时
		{
			time=1;while(time);//等待 1 秒
			show_Washing_ok();//显示"洗涤完成"
			time=show_ok_time;//延时两秒
			while(time);//等待计时为零
			if(flag==1)mode=3,time=Output_water_time;//进入排水模式
			if(flag==2)mode=0,flag=0;//返回主菜单
```

```
                clearnscreen();//清屏
            }
            break;//洗涤模式
            case 3:
            if(Output_water(time))//判断排水是否到时
            {
                time=1;while(time);//等待 1 秒
                show_Output_water_ok();//显示"排水完成"
                time=show_ok_time;//延时两秒
                while(time);//等待计时为零
                if(flag==1)mode=4,time=Dehydration_time;//进入脱水模式
                if(flag==2)mode=0,flag=0;//返回主菜单
                clearnscreen();//清屏
            }
            break;//排水模式
            case 4:
            if(Dehydration(time))//判断脱水是否到时
            {
                time=1;while(time);//等待 1 秒
                show_Dehydration_ok();//显示"脱水完成"
                time=show_ok_time;//延时两秒
                while(time);//等待计时为零
                mode=0,flag=0;//返回主菜单
                clearnscreen();//清屏
            }
            break;//脱水模式
            default:    break;
        }
}
/*************************************************************************
********/
unsigned int ms;
void TIME0_ROUTING() interrupt 1  //定时器 0  定时时间:1ms
{
    TH0=0xfc; //寄存器高字节
    TL0=0x66;//寄存器低字节
    KEY();//按键扫描
```

```
    if(time)
    {
        if(++ms==1000)ms=0,time--;//倒计时
    }
}
void TIME0_INIT()
{
    TMOD=0x01;//模式选择
    TH0=0xfc;//寄存器高字节
    TL0=0x66;//寄存器低字节
    ET0=1;//定时器 0 中断使能
    TR0=1;//定时器 0 计数开关
    EA=1;//定时器使能总开关
}
void main()
{
    TIME0_INIT();//定时器初始化
    LCD_INIT();//液晶初始化
    while (1)
    {
        Read_KEY();//读键值
        Read_flag();//读状态
        Work();//工作
    }
}
```

LCD12864 驱动头文件库程序

LCD.H

```
#include<reg51.h>                  // 包含 51 单片机头文件
/******12864LCD 引脚定义******/
#define DATAPORT   P0              // P0 端口为 LCD 数据总线
sbit CS1=P1^3;                     // 位定义 P2.0，CS1 为左半屏片选信号
sbit CS2=P1^4;                     // 位定义 P2.1，CS2 为右半屏片选信号
sbit E=P1^2;                       // E 为锁存使能,下降沿锁存数据
sbit RS=P1^0;                      // 数据/指令选择，0 为数据，1 为指令
sbit RW=P1^1;                      // 读/写选择，0 为写，1 为读
sbit RST=P1^5;                     // TG12864 复位信号
```

```
sbit BL=P1^6;                          //  背光选择
/******毫秒级延时函数******/
void delay_ms(unsigned int ms)
{
   unsigned char i;                          //临时延时变量 i
   while(--ms)                          //减 1 判断是否为 0
   {
       for(i=0;i<125;i++);       //加 1 判断是否达到 125
   }
}
void selectscreen(unsigned char screen)//选屏  0:全屏  1:左半屏  2:右半屏
{
      switch (screen)
      {
            case 0:CS1=1;CS2=1;     break;          //全屏
            case 1:CS1=1;CS2=0;     break;          //左半屏
            case 2:CS1=0;CS2=1;     break;          //右半屏
            default:   break;
      }
}
void write_cmd(unsigned char dat)            //写命令
{
      RW=0;           //写操作
      RS=0;           //写命令
      DATAPORT=dat;     //P0 口传输命令
      E=1;
      E=0;       //下降沿(使能)，将命令送入
}
void write_data(unsigned char dat)           //写数据
{
      RW=0;           //写操作
      RS=1;           //写数据
      DATAPORT=dat;     //P0 口传输数据
      E=1;
      E=0;       //下降沿(使能)，将数据送入
      RS=0;           //置为写命令状态，减少噪点发生率
}
```

```
void check_busy_12864()//忙检测函数
{
        uchar i=0,dat;                  //定义一个临时变量
        RS=0;                           //对指令操作
        RW=1;                           //读操作
        do{                             //先执行下面指令
          DATAPORT=0x00;                //口线置 1 防止干扰
          E=1;                          //下降沿的高电平部分
          dat=DATAPORT;                 //读 LCD 总线上的数据到临时变量
          E=1;                          //短暂延时
          E=0;
          dat=0x80&dat;       //取出数据的 D7 位，该位是忙检测(busy)信号
          i++;
       }while((dat==0x80)&&(i<100));//LCD 内部不忙或超时，退出
}
void set_onoff(unsigned char onoff)         //开关
{
    onoff|=0x3e;        //控制字
    write_cmd(onoff);  //写命令,将控制字写入
}

void set_page(unsigned char page)           //选行
{
    page&=0x07;             //限位
    page|=0xb8;             //控制字
    write_cmd(page);   //写命令,将控制字写入
}
void set_column(unsigned char column)      //选列
{
    if(column>63)selectscreen(2);      //当列<63 时，则选择左半屏
    else selectscreen(1);           //当列超过左半屏时，则选择右半屏
    column&=0x3f;           //限位
    column|=0x40;           //控制字
    write_cmd(column);      //写命令,将控制字写入
}
void set_startline(unsigned char line)   //选起始行
{
```

```
    line&=0x3f;                 //限位
    line|=0xc0;                 //控制字
    write_cmd(line);    //写命令,将控制字写入
}

void clearnscreen() //清屏
{
    unsigned char i,j;
    selectscreen(0);    //选屏
    for (i=0;i<8;i++)
    {
        set_page(i);            //选行
        for (j=0;j<64;j++)    //列会自动加 1
        {
            write_data(0); //写数据
        }
    }
}
void LCD_INIT()//LCD 初始化
{
      BL=0;                     //背光
      RST=0;                    //复位
      delay_ms(10);             //延时等待
      RST=1;                    //置位
      delay_ms(10);             //延时等待
      set_onoff(1);             //开关
      set_startline(0);         //起始行
      clearnscreen();           //清屏
}
```

4、程序说明

本程序主要通过液晶菜单来控制洗衣机的工作。Read_KEY():读键值子程序，用来根据按下的按键来控制系统的运行。Read_flag():读状态子程序，用来读取系统状态，来显示不同状态下的初始界面。Work():工作子程序，用来对四种模式的工作操作。flag 变量:用来标志系统的为停止、自动或手动模

式。mode 变量:用来标志系统为停止模式、进水模式、洗涤模式、排水模式或脱水模式。

5、任务实施

5.1 建立工程

打开 keil 软件，通过菜单“Project”，新建立一个工程”new Project”项目 XYJ，然后再建一个文件名为 XYJ.C 的源程序文件，将上面的参考程序输入并保存。建立的 XYJ.C 文件添加入本项目中。

5.2 编译并生成 HEX

单击”Build target”按钮，对源程序进行编译和链接，产生 HEX 文件。

5.3 烧录芯片

打开 STC-ISP 下载软件，选中单片机型号为“STC90C52RC”,选择最低波特率为 2400 最高波特率为 115200（为默认值一般不需修改）。点击打开程序文件按钮，在刚才建立的 XYJ 文件夹中生成的 XYJ.HEX 文件。然后点击“下载/编程”按钮。最后给最小系统通电，等待电脑端串口与单片机握手成功。就能把程序载入到单片机中去了。其详细操作图如图 4- 52 所示：

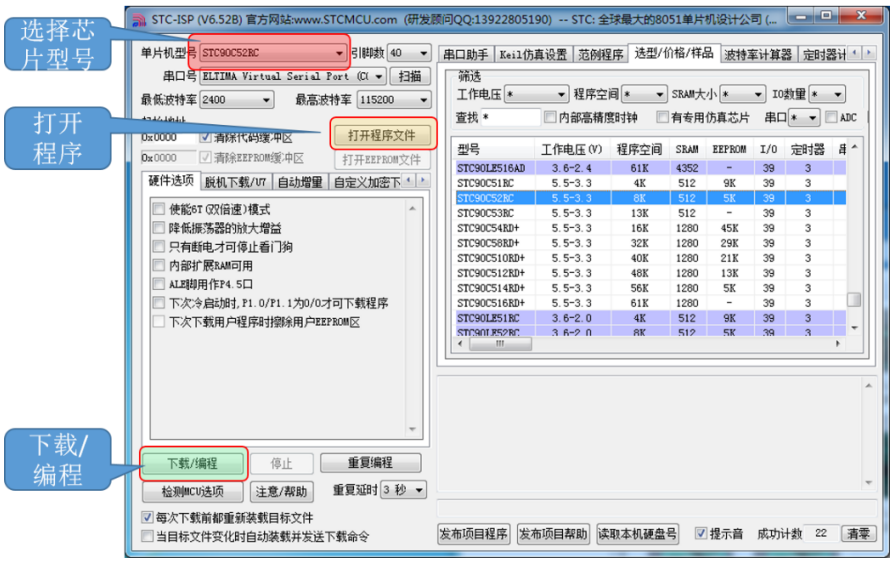

图 4- 52 STC 单片机程序烧写示意图

5.4 硬件调试

程序下载完成后，把单片机最小系统接入电路中，调试电路使其能正常工作。

三、任务评价

表 4- 11　任务六完成情况评价表

项目名称			评价时间		年　月　日		
小组名称		小组成员					
评价内容	评价要求	权重	评价标准	学生自评得分	小组评价得分	教师评价得分	合计
职业与安全意识	1. 工具摆放、操作符合安全操作规程；2．遵守纪律、爱惜设备和器材、工位整洁；3. 具有团队协作精神。	10%	好（10） 较好（8） 一般（6） 差（<6）				
印制线路板调试	1．能够根据任务要求进行印制线路板的调试；2. 根据任务要求，能够对对应电路部分进行硬件电路的检查与故障检修。	15%	好（15） 较好（12） 一般（9） 差（<9）				
项目功能测试	1．编写的程序能成功编译；2. 程序能正确烧写到芯片中；3. 能够实现洗衣机模拟装置的功能	60%	好（60） 较好（45） 一般（30） 差（<30）				
问题与思考	1．洗衣机的工作时间是怎么确定的？2．增加洗衣机的安全性，当盖子打开是应该直接停止工作。可以通过四个按键中任意按键模拟盖子打开。	15%	好（15） 较好（12） 一般（9） 差（<6）				
教师签名			学生签名			总分	
项目评价=学生自评（0.2）+小组评价（0.3）+教师评价（0.5）							

四、思考与练习

1．使用洗衣机模拟装置完成任务六《洗衣机模拟装置》的程序编写。

2．如何使洗衣机有更多的功能？