

教你学单片机(十二)

■周兴华

汇编语言的程序设计

单片机的体积小、价格便宜、控制功能强大,可广泛应用于各个领域。但单片机本身毕竟只是一片微控制器,用它组成应用系统,还需要一个研制过程,该过程称为对单片机进行“开发”。

单片机应用系统就是为某应用目的所设计的专门的用户系统。虽然单片机各应用系统功能和用途不尽相同,硬件结构和应用软件差异也很大,但是其研制方法和设计过程却大致相同。从用户提出的任务开始,以芯片为基础进行设计,最后构成一个实用系统,大致可分为以下几个阶段:

1. 确定任务

单片机应用系统的研制过程是以确定系统的功能和技术指标开始的。首先要细致分析、研究实际问题,明确各项任务和要求。从考虑系统的先进性、可靠性、可维护性以及成本、经济效益出发,拟定出合理可行的技术性能指标。

2. 总体设计

在对应用系统进行总体设计时,应根据应用系统提出的各项技术性能指标,拟定出性能/价格比最高的一套方案。首先应依据任务的繁杂程度、技术指标的要求来选择单片机机型,在机型选定之后,再选择系统中要用到的其它元器件。在总体方案设计过程中,必须对软件和硬件综合考虑。原则上能够由软件来完成的任务就尽可能用软件来实现,以降低硬件成本,简化硬件结构。同时还要求大致规定各接口电路的地址、软件的结构和功能、上下位机的通信协议、程序的驻留区域及工作缓冲区等。总体设计方案一旦确定下来,系统的大致规模及软件的基本框架就确定了。

3. 硬件设计(电路设计)

硬件设计是指应用系统的电路设计,包括主机、控制电路、存储器、I/O

接口、A/D和D/A转换电路等。硬件设计时应考虑留有充分余量,电路设计力求正确无误,因为系统调试中不易修改硬件结构。

4. 软件设计(程序设计)

要想使单片机完成某一具体的工作任务,必须按序执行一条条指令。这种按工作要求编排指令序列的过程称为程序设计。程序设计可采用汇编语言或高级语言来做,对初学者来说,为掌握单片机的硬件结构,一般可从汇编语言开始学,等以后熟悉了单片机的硬件结构后再用高级语言来做程序会觉得十分方便。

汇编语言程序设计步骤

使用汇编语言作为程序设计语言的编程步骤与高级语言编程步骤类似,但又略有差异。其程序设计步骤大致可分为以下几步:

1. 熟悉与分析工作任务,明确其要求和要达到的工作目的、技术指标等。
2. 确定解决问题的计算方法和工作步骤。
3. 画程序状态工作流程图。
4. 分配内存工作单元,确定程序与数据区存放地址。
5. 按流程图编写源程序。
6. 上机调试、修改及最后确定源程序。

在进行程序设计时,必须根据实际问题和所使用的单片机特点来确定算法,然后按照尽可能使程序简短和缩短运行时间两个原则编写程序。编程技巧需经大量实践后逐渐地加以提高。

下面我们介绍一些常用的程序设计方法。

一、顺序程序设计

顺序结构程序是一种最简单、最基本的程序,其特点是按程序编写的顺序依次执行,程序流向不变。顺序结构程序是所有复杂程序的基础及

基本组成部分。图1为顺序结构程序的状态流程。

下面我们学习设计一个简单的顺序结构程序,让读者朋友从简单开始逐步掌握程序设计。

之前在《手把手教你学单片机(五)》讲座中,我们曾用移位指令设计了一个程序,使S1板PO口的输出为:一个发光二极管点亮右移循环,形成流水灯。这里我们使用顺序结构程序,同样完成在S1板PO口上的右移循环流水灯。

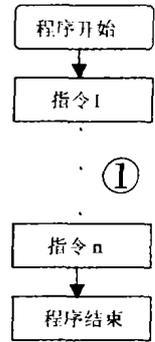
在我的文档中建立一个文件目录(S20),然后建立一个S20.uv2的工程项目,最后建立源程序文件(S20.asm)。

输入下面的程序:

```

序号:1      ORG 0000H
2           LJMP MAIN
3           ORG 030H
4 MAIN:    CLR P0.7
5           ACALL DEL
6           SETB P0.7
7           CLR P0.6
8           ACALL DEL
9           SETB P0.6
10          CLR P0.5
11          ACALL DEL
12          SETB P0.5
13          CLR P0.4
14          ACALL DEL
15          SETB P0.4
16          CLR P0.3
17          ACALL DEL
18          SETB P0.3
19          CLR P0.2
20          ACALL DEL
21          SETB P0.2
22          CLR P0.1
23          ACALL DEL
24          SETB P0.1
25          CLR P0.0
26          ACALL DEL
27          SETB P0.0
28          AJMP MAIN
29 DEL:    MOV R7,#0FFH
30 DEL1:   MOV R6,#0FFH
31 DEL2:   MOV R5,#01FH
32 DEL3:   DJNZ R5,DEL3
33         DJNZ R6,DEL2
34         DJNZ R7,DEL1

```



35 RET
36 END

编译通过后,将其烧录到 89C51 芯片中,将芯片插入到 S1 型 LED 输出试验板上,在 S1 实验板上通电运行后,PO 口的输出状态为:一个发光管点亮并右移循环,形成流水灯。

我们对程序进行分析解释。

序号 1(程序解释,以下同):程序开始。

序号 2:跳转到 MAIN 主程序处。

序号 3:主程序 MAIN 从地址 0030H 开始。

序号 4:PO.7 置低电平,点亮对应的发光管。

序号 5:调用延时子程序,维持发光管点亮。

序号 6:PO.7 置高电平,熄灭对应的发光管。

序号 7:PO.6 置低电平,点亮对应的发光管。

序号 8:调用延时子程序,维持发光管点亮。

序号 9:PO.6 置高电平,熄灭对应的发光管。

序号 10:PO.5 置低电平,点亮对应的发光管。

序号 11:调用延时子程序,维持发光管点亮。

序号 12:PO.5 置高电平,熄灭对应的发光管。

序号 13:PO.4 置低电平,点亮对应的发光管。

序号 14:调用延时子程序,维持发光管点亮。

序号 15:PO.4 置高电平,熄灭对应的发光管。

序号 16:PO.3 置低电平,点亮对应的发光管。

序号 17:调用延时子程序,维持发光管点亮。

序号 18:PO.3 置高电平,熄灭对应的发光管。

序号 19:PO.2 置低电平,点亮对应的发光管。

序号 20:调用延时子程序,维持发光管点亮。

序号 21:PO.2 置高电平,熄灭对应的发光管。

序号 22:PO.1 置低电平,点亮对应的发光管。

序号 23:调用延时子程序,维持发光管点亮。

序号 24:PO.1 置高电平,熄灭对应的发光管。

序号 25:PO.0 置低电平,点亮对应的发光管。

序号 26:调用延时子程序,维持发光管点亮。

序号 27:PO.0 置高电平,熄灭对应的发光管。

序号 28:跳转到标号 MAIN 处进行循环运行。

序号 29-35:延时子程序。

序号 36:程序结束。

二、循环程序设计

在程序设计中,有时要求对某一段程序重复执行多次,在这种情况下可用循环程序结构,有助于缩减程序长度。一个循环程序的结构由以下 3 部分组成:

1.循环初态。在循环开始时,往往需要设置循环过程工作单元的初始值,如工作单元初值、计数器初值等。

2.循环体。即要求重复执行的程序段部分,它用于完成主要的计算或操作任务。

3.循环控制部分。在循环程序中必须给出循环结束的条件,否则就成为死循环。循环控制就是根据循环结束条件,判断是否结束循环。

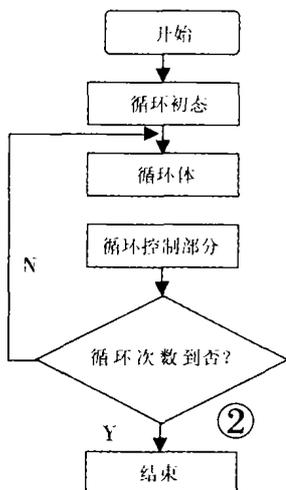


图 2 为循环结构程序的状态流程。

下面在 S1 板做一个实验,内部 RAM 从 40H 单元开始,有一个无符号数数据块,其长度存于 30H 单元,试求出数据块中最大的数,并存入 35H 单元,同时送 PO 口显示。

在我的文档中建立一个文件目录(S21),然后建立一个 S21.uv2 的工程项目,最后建立源程序文件(S21.asm)。

输入下面的程序:

```

序号:1   LEN DATA 30H
2       MAX DATA 35H
3       BLOCK DATA 40H
4       ORG 0000H
5       LJMP MAIN
6       ORG 030H
7   MAIN: MOV LEN,#10
8       MOV A,#100
9       MOV R1,#BLOCK
10  LOOP1: MOV @R1,A
11       INC R1
12       INC A
13       DJNZ LEN,LOOP1
14       MOV LEN,#15
15       MOV A,#50
16  LOOP2: MOV @R1,A
17       INC R1
18       INC A
19       DJNZ LEN,LOOP2
20       CLR A
21       MOV LEN,#35
22       MOV R2,LEN
23       MOV R1,#BLOCK
24  LOOP:  CLR C
25       SUBB A,@R1
26       JNC NEXT1
27       MOV A,@R1
28       SJMP NEXT2
29  NEXT1: ADD A,@R1
30  NEXT2: INC R1
31       DJNZ R2,LOOP
  
```

```

32       MOV MAX,A
33       MOV P1,MAX
34       ACALL DEL
35       SJMP $
36  DEL:  MOV R7,#0FFH
37  DEL1: MOV R6,#0FFH
38  DEL2: MOV R5,#1FH
39  DEL3: DJNZ R5,DEL3
40       DJNZ R6,DEL2
41       DJNZ R7,DEL1
42       RET
43       END
  
```

编译通过后,将其烧录到 89C51 芯片中,将芯片插入到 S1 型 LED 输出试验板上,在 S1 实验板上通电运行后,P1 口的输出状态为:01101101,换算成 10 进制数为 109。这个结果对不对呢?对程序进行分析便知。

我们对程序进行分析解释。

序号 1(程序解释,以下同):定义 30H 数据单元地址为 LEN。

序号 2:定义 35H 数据单元地址为 MAX。

序号 3:定义 40H 数据单元地址为 BLOCK。

序号 4:程序从地址 0000H 开始。

序号 5:跳转到 MAIN 主程序处。

序号 6:主程序 MAIN 从地址 0030H 开始。

序号 7:向 30H 单元送立即数 10,此为第一个数据块长度。

序号 8:向累加器 A 中送立即数 100。

序号 9:向寄存器 R1 中送立即数 40H。

序号 10:将累加器 A 内容送 40H 单元中。

序号 11:寄存器 R1 内容加 1。

序号 12:累加器 A 内容加 1。

序号 13:30H 内容减 1。若不为 0 转 LOOP1;若为 0 向下执行。这样,即可将从 100 开始的 10 个立即数(100、101、102、103、104、105、106、107、108、109)送入从 40H 单元开始的数据块中。

序号 14:再向 30H 单元送立即数 15,此为第二个数据块长度。

序号 15:向累加器 A 中送立即数 50。

序号 16:将累加器 A 中内容送入以 R1 内容为地址的单元,由于 R1 已加至 50H,故第二个数据块首与第一个数据块尾连接在一起。

序号 17:寄存器 R1 内容加 1。

序号 18:累加器 A 内容加 1。

序号 19:30H 内容减 1。若不为 0 转 LOOP2;若为 0 向下执行。这样,即可将从 50 开始的 15 个立即数(50、51、52、53、54、55、56、57、58、59、60、61、62、63、64)送入从 50H 单元开始的数据块中。

序号 20:向累加器 A 中送立即数 00H。

序号 21:按下来的动作是判别数据块中的最大数。由于两个数据块合在一起,总长度为 35,故向 30H 单元送立即数 35。

序号 22:将立即数 35 转送至寄存器 R2 中。

序号 23:R1 中送入数据块首地址。

序号 24:清除进位位 C。
 序号 25:将 A 中内容减以 R1 内容为地址的单元内容,结果存 A 中。
 序号 26:若 C 为 0,说明 A 中内容大于等于以 R1 内容为地址的单元内容,程序转 NEXT1;否则顺序执行。
 序号 27:将以 R1 内容为地址的单元内容送至累加器 A 中。
 序号 28:跳转至标号 NEXT2 处。
 序号 29:将减后结果与减数相加,恢复被减数并存入 A 中。
 序号 30:R1 内容(即数据块地址)加 1。
 序号 31:R2 内容(即总的数据块长度)减 1,若结果不为 0,跳转至 LOOP 处循环执行;若为 0,向下执行。序号 24~31 这段程序的意思是:将 A 中的数作为被减数(从 0 开始)与数据块中的数逐个相减,若 A 中内容大于等于数据块中减数,保留 A 中内容;若 A 中内容小于数据块中减数,则由减数取代 A 中内容。
 序号 32:将判别出的最大数送入 MAX(35H)单元。
 序号 33:将最大数送 P1 口显示。
 序号 34:调用延时子程序,以便观察清楚。
 序号 35:动态停机。
 序号 36~42:延时子程序。
 序号 43:程序结束。

好了,这样我们分析清楚了,数据块中的最大数为 109。单片机自己找出了数据块中的最大数。可能的读者要问,这样一个找最大数的小程序有什么实用价值呢?现在我们已初步学会了一些简单程序的设计,但我们不会永远停留在这一阶段,将来设计的程序会更复杂,会用到许多数学运算及判断,而一个复杂程序大多由一系列的功能子程序组成,因此学习这类程序设计,就是在向提高自己的程序设计能力进军,是很有必要的。

我们在程序中经常调用的延时子程序也属于循环程序结构。延时的实现是靠执行一条条指令延时来实现的,因此编程使得某指令循环若干次,即可达到延时目的。延时子程序一般用以下指令来实现。

DJNZ Rn,rel

说明:Rn=Rn-1,Rn 不等于 0 时,PC=PC+3+rel。Rn 等于 0 时,PC=PC+3。

该指令的长度为 3 个字节,指令执行周期为两个机器周期,Rn 为工作寄存器,当 Rn-1 不等于 0 时,程序执行该段循环延时程序。Rn-1 等于 0 时,则执行下一条指令。工作寄存器 Rn 作为该指令循环次数指针。编程模式如下:

```
DELAYn:MOV Rn,#datan
.....
DELAY6:MOV R6,#data6
DELAY7:MOV R7,#data7
DJNZ R7,$
DJNZ R6, DELAY7
.....
DJNZ Rn, DELAYn+1
.....
```

附表

晶振频率(MHz)	12	6	4	3	2	1
机器周期(μS)	1	2	3	4	6	12

延时子程序的延长时间=7xR6x.....xRnx(2x 机器周期),其中 Rn 为工作寄存器 R1~R7,机器周期由单片机的晶振主频决定,附表为 MCS-51 单片机晶振频率与机器周期对照表。

下面在 S1 板做一个实验,把内部 RAM 中起始地址为 20H 的数据块传送到地址为 80H 的一区域,直到发现“\$”字符的 ASCII 码(24H)为止。将“\$”字符的 ASCII 码(24H)在 P0 口显示出。同时规定数据块最大长度为 48 个字节。

在我的文档中建立一个文件目录(S22),然后建立一个 S22.uv2 的工程项目,最后建立源程序文件(S22.asm)。

输入下面的程序:

```
序号:1      ORG 0000H
2          LJMP MAIN
3          ORG 030H
4  MAIN:   MOV R2,#48
5          MOV R1,#00H
6          MOV R0,#20H
7  THERE:  MOV A,R1
8          MOV @R0,A
9          INC R0
10         INC R1
11         DJNZ R2, THERE
12         MOV R2,#48
13         MOV R0,#20H
14         MOV R1,#80H
15  LOOP:   MOV A,@R0
16         CJNEA,#24H,LOOP2
17         MOV P1,A
18         ACALL DEL
19         AJMP LOOP1
20  LOOP2:  MOV @R1,A
21         MOV P0,A
22         ACALL DEL
23         INC R0
24         INC R1
25         DJNZ R2,LOOP
26  LOOP1:  SJMP $
27  DEL:    MOV R7,#0FFH
28  DEL1:   MOV R6,#0FFH
```

```
29  DEL2:  MOV R5,#08H
30  DEL3:  DJNZ R5,DEL3
31         DJNZ R6,DEL2
32         DJNZ R7,DEL1
33         RET
34         END
```

编译通过后,将其烧录到 89C51 芯片中,将芯片插入到 S1 型 LED 输出试验板上,在 S1 试验板上通电运行后,P0 口从 00H 开始,每隔一秒钟刷新显示数据块中传送的内容。当显示 23H 后,传送停止。同时 P1 口显示“\$”字符的 ASCII 码(24H)。

我们对程序进行分析解释。

序号 1(程序解释,下同):程序开始。
 序号 2:跳转到 MAIN 主程序处。
 序号 3:主程序 MAIN 从地址 0030H 开始。
 序号 4:向 R2 中送数 48(即数据块长度)。
 序号 5:向寄存器 R1 中送数 00H。
 序号 6:向寄存器 R0 中送数 20H(源数据块首址)。
 序号 7:R1 的内容送累加器 A。
 序号 8:A 中内容送以 R0 内容为地址的单元。
 序号 9:寄存器 R0 内容加 1。
 序号 10:寄存器 R1 内容加 1。
 序号 11:R2 内容(数据块长度)减 1,若不为 0,跳转至 THERE 执行;若为 0,顺序执行。序号 4~11 的作用是将 00H~2FH 立即数存入从 20H 单元开始的源数据块中,长度为 48。
 序号 12:再向 R2 中送立即数 48(即数据块长度)。
 序号 13:R0 中存入立即数 20H(源数据块首地址)。
 序号 14:R1 中存入立即数 80H(目标数据块首地址)。
 序号 15:源数据块内容送 A。
 序号 16:判 A 中内容是否为 24H?若不等,转至 LOOP2;若相等,顺序执行。
 序号 17:A 中内容送 P1 口显示。
 序号 18:调用延时子程序,以便观察清楚。
 序号 19:跳转至 LOOP1。
 序号 20:A 中内容送目标数据块中。
 序号 21:同时 A 中内容送 P0 显示。
 序号 22:调用延时子程序,以便观察清楚。
 序号 23:R0 内容(源数据块地址)加 1。
 序号 24:R1 内容(目标数据块地址)加 1。
 序号 25:R2 内容(数据块长度)减 1,若不为 0,跳转至 LOOP 循环执行;若为 0,顺序执行。序号 12~25 的作用是将 20H 单元开始的源数据块传送到 80H 单元开始的目标数据块中。长度为 48。若数据块中内容不为 24H,传送进行,同时送 P0 口显示;若数据块中内容等于 24H,传送停止,同时将 24H 送 P1 口显示。
 序号 26:动态停机。
 序号 27~33:延时子程序。

序号 34:程序结束。

这段程序也是相当有用的,例如我们要将一个数据块的部分内容读出时,可以设一个标志(如 24H),一旦程序运行到标志时,数据读出即结束。

三、子程序设计及调用、返回

当程序中出现多次执行同一程序段的时候,就可以把多次执行的程序段编成子程序,在需要的时候可以被其它程序多次调用,这就是所谓的子程序结构。子程序结构减少了主程序的长度,使程序的层次结构更加清楚。使用子程序的过程,称为调用子程序。子程序执行完后返回主程序的过程称为子程序返回。

子程序的结构特点

子程序是一种具有某种功能的程序段,其资源需要为所有调用程序共享,因此,子程序在功能上应具有通用性,在结构上应具有独立性。它在结构上与一般程序的主要区别是在子程序末尾有一条子程序返回指令(RET),其功能是执行完子程序后将堆栈内的断点地址弹出至 PC 返回到主程序中。

编写子程序时的注意要点:

1. 给每个子程序赋一个名字,即入口地址的标号。

2. 为了能正确地传递参数。要有入口条件,说明进入子程序时它所要处理的数据如何得到(例如,是把它放在 ACC 中还是放在某工作寄存器中等)。另外,要有出口条件,即处理的结果是如何存放的。

3. 注意保护现场和恢复现场。在执行子程序时,可能要使用累加器或某些工作寄存器。而在调用子程序之前,这些寄存器中可能存放有主程序的中间结果,这些中间结果是不允许被破坏的。因而,在子程序使用累加器和这些工作寄存器之前,要将其中的内容保存起来,即保护现场。当子程序执行完,即将返回主程序之前,再将这些内容取出,送回到累加器或原来的工作寄存器中,这一过程称为恢复现场。保护和恢复现场通常用堆栈来进行。对于需要保护现场的情况,编写子程序时要在子程序的开始使用压栈指令,把需要保护的寄存器内容压入堆栈。当子程序执行完,在

返回指令前使用弹出指令,把堆栈中保护的内容弹出到原来的寄存器,这样即恢复了现场。

4. 为了使子程序具有一定的通用性,子程序中的操作对象应尽量用地址或寄存器形式,而不用立即数形式。另外,子程序中如含有转移指令,应尽量用相对转移指令,以便它不管放在内存的哪个区域,都能正确执行。

子程序的调用与返回

主程序调用子程序是通过子程序调用指令 LCALL add16 和 ACALL add11 来实现的。LCALL add16 称为长调用指令,指令的操作数给出 16 位的子程序首地址;ACALL add11 称为绝对调用指令,它的操作数提供子程序的 11 位入口地址,此地址与程序计数器 PC 的高 5 位并在一起,构成 16 位的转移地址(即子程序入口地址)。子程序调用指令的功能是将 PC 中的内容(调用指令的下一条指令地址称断点)压入堆栈(即保护断点),然后将调用地址送入 PC,使程序转向子程序的入口地址。

子程序的返回是通过返回指令 RET 实现的。RET 指令的功能是将堆栈中存放的返回地址(即断点)弹出堆栈,送回到 PC 去,使程序继续从断点处执行。

子程序嵌套

子程序嵌套(或称多重转子)是指,在子程序执行过程中还可以调用另一个子程序。子程序嵌套的次数从理论上说是无限的,但实际上,由于受堆栈深度的限制,嵌套次数是有限的。

下面在 S1 板做一个实验,采用上一个实验的延时子程序,让 PO 口的 8 个 LED 闪烁 20 次。

在我的文档中建立一个文件目录(S23),然后建立一个 S23.uv2 的工程项目,最后建立源程序文件(S23.asm)。

输入下面的程序:

```

序号:1      ORG 0000H
2          LJMP MAIN
3          ORG 030H
4  MAIN:    MOV R0,#0
5          MOV A,#0FFH
6  LOOP:    CPL A
7          MOV PO,A
8          ACALL DEL
9          CJNE R0,#40,LOOP
10         SJMP $

```

```

11  DEL:    MOV R7,#0FFH
12  DEL1:   MOV R6,#0FFH
13  DEL2:   MOV R5,#08H
14  DEL3:   DJNZ R5,DEL3
15          DJNZ R6,DEL2
16          DJNZ R7,DEL1
17          INC R0
18          RET
19          END

```

编译通过后,将其烧录到 89C51 芯片中,将芯片插入到 S1 型 LED 输出试验板上,在 S1 实验板上通电运行后,PO 口的输出状态为:8 个发光二极管点亮/熄灭,共重复 20 次。

我们对程序进行分析解释。

序号 1(程序解释,以下同):程序开始。

序号 2:跳转到 MAIN 主程序处。

序号 3:主程序 MAIN 从地址 0030H 开始。

序号 4:将寄存器 R0 清零,这也是子程序调用时的入口条件。

序号 5:向累加器 A 送数 FFH(二进制为 11111111)。

序号 6:累加器 A 内容取反(二进制为 00000000)。

序号 7:将 A 中内容送 PO 口,点亮或熄灭发光管(低电平点亮,高电平熄灭)。

序号 8:调用延时子程序,以便观察清楚。

序号 9:判 R0 内容是否等于十进制数 40。若不等,跳转至 LOOP;若相等,向下执行。

序号 10:动态停机(原地踏步)。

序号 11~18:延时子程序。其中序号 17 的指令作用为,每调用一次子程序,R0 的内容加 1。由于本子程序只要求调用 40 次,因此有入口条件(R0=#0),但不需出口条件。并且也不要求对现场进行保护和恢复。

序号 19:程序结束。

配文优惠邮购(每次邮费保价费 12 元):Keil 51 Windows 集成开发环境(已汉化光盘,邮购代号:K1):46 元。TOP851 多功能编程器(邮购代号:B1):400 元。LED 输出试验板(邮购代号:S1):90 元。LED 数码管输出试验板(邮购代号:S2):140 元。5V 高稳定专用稳压电源(邮购代号:D1):35 元。。邮购时只需在附言栏中写明邮购代号及数量并附上联系电话即可。

邮购地址:201103 上海市闵行区莲花路 2151 弄 57 号 201 室

联系人:吕超亚

电话:021-64066571 13044152947

技术支持 E-mail:zxh2151@sohu.com ◀