项目三

IAP15W4K58S4
单片机的并行I/O口与应用编程

无论什么单片机，外部的命令以及处理结果都要通过单片机的并行 I/O 口进行通信。本项目要达到的目标包括：一是掌握 IAP15W4K58S4 单片机并行 I/O 口的工作模式与设置；二是掌握 C 语言程序的结构、数据类型以及特殊功能寄存器的定义；三是学会 IAP15W4K58S4 单片机并行 I/O 口应用的 C 语言编程。

知识点：

- IAP15W4K58S4 单片机并行 I/O 口的工作模式以及负载能力；
- IAP15W4K58S4 单片机并行 I/O 口的准双向工作模式中“准”字的含义；
- C 语言的程序结构、数据类型以及变量的定义；
- C 语言的算术运算、关系运算、逻辑运算语句；
- C 语言的控制语句(if、switch/case、while、for 等语句)；
- C 语言(C51)中特殊功能寄存器地址以及位地址的定义与赋值。

技能点：

- IAP15W4K58S4 单片机并行 I/O 口工作模式的设置；
- C 语言(C51)中特殊功能寄存器地址以及位地址的定义与赋值；
- IAP15W4K58S4 单片机并行 I/O 口应用的 C 语言编程。

任务 1　IAP15W4K58S4 单片机并行 I/O 口的输入/输出

IAP15W4K58S4 单片机的并行 I/O 口需要通过地址来访问。作为一种特殊功能寄存器，首先要将 IAP15W4K58S4 单片机的并行 I/O 口名称进行地址定义；再利用简单赋值语句，实现 IAP15W4K58S4 单片机并行 I/O 口输入/输出的应用编程。

1. IAP15W4K58S4 单片机的并行 I/O 口与工作模式

1) I/O 口功能

IAP15W4K58S4 单片机最多有 62 个 I/O 口(P0.0～P0.7、P1.0～P1.7、P2.0～P2.7、P3.0～P3.7、P4.0～P4.7、P5.0～P5.5、P6.0～P6.7、P7.0～P7.7)。LQFP44 封装的 IAP15W4K58S4 单片机共有 42 个 I/O 端口线,分别为 P0.0～P0.7、P1.0～P1.7、P2.0～P2.7、P3.0～P3.7、P4.0～P4.7、P5.4、P5.5,可用作准双向 I/O。其中,大多数 I/O 口线具有 2 个以上功能。各 I/O 口线的引脚功能名称前已介绍,详见表 2-2-1～表 2-2-6。

2) I/O 口的工作模式

IAP15W4K58S4 单片机的所有 I/O 口均有 4 种工作模式:准双向口(传统 8051 单片机 I/O 模式)、推挽输出、仅为输入(高阻状态)与开漏模式。每个 I/O 口的驱动能力均可达到 20mA。但 40 引脚及以上单片机整个芯片最大工作电流不要超过 120mA;20 引脚以上、32 引脚以下单片机整个芯片最大工作电流不要超过 90mA。每个口的工作模式由 PnM1 和 PnM0(n=0,1,2,3,4,5)两个寄存器的相应位来控制。例如,P0M1 和 P0M0 用于设定 P0 口的工作模式,其中 P0M1.0 和 P0M0.0 用于设置 P0.0 的工作模式,P0M1.7 和 P0M0.7 用于设置 P0.7 的工作模式,以此类推,设置关系如表 3-1-1 所示。IAP15W4K58S4 单片机上电复位后所有的 I/O 口(除与增强型 PWM 有关的引脚之外)均为准双向口模式。

表 3-1-1 I/O 口工作模式的设置

控制信号		I/O 口工作模式
PnM1[7:0]	PnM0[7:0]	
0	0	准双向口(传统 8051 单片机 I/O 模式):灌电流可达 20mA,拉电流为 150～230μA
0	1	推挽输出:强上拉输出,可达 20mA,要外接限流电阻
1	0	仅为输入(高阻)
1	1	开漏:内部上拉电阻断开,要外接上拉电阻才可以拉高。此模式可用于 5V 器件与 3V 器件电平切换

2. IAP15W4K58S4 单片机的并行 I/O 口的结构

如前所述,IAP15W4K58S4 单片机的所有 I/O 口均有 4 种工作模式:准双向口(传统 8051 单片机 I/O 模式)、推挽输出、仅为输入(高阻状态)与开漏模式,由 PnM1 和 PnM0(n=0,1,2,3,4,5)两个寄存器的相应位来控制 P0～P5 端口的工作模式。下面介绍 IAP15W4K58S4 单片机的并行 I/O 口不同模式的结构与工作原理。

1) 准双向口工作模式

准双向口工作模式下,I/O 口的电路结构如图 3-1-1 所示。

准双向口工作模式下,I/O 口可用于直接输出,不需重新配置口线输出状态。这是因为当口线输出为“1”时,驱动能力很弱,允许外部装置将其拉低电平。当引脚输出为低电平时,它的驱动能力很强,可吸收相当大的电流。

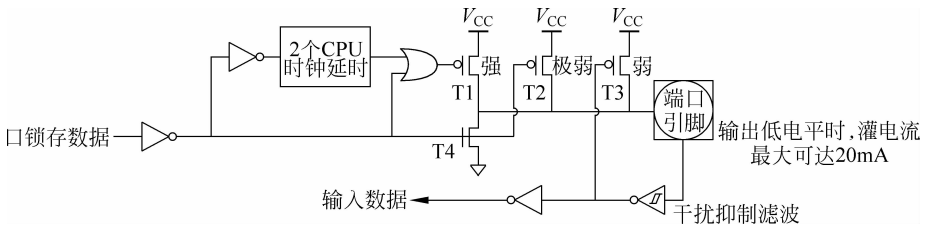

图 3-1-1　准双向口工作模式下 I/O 口的电路结构

每个端口都包含一个 8 位锁存器，即特殊功能寄存器 P0～P5。这种结构在数据输出时，具有锁存功能，即在重新输出现新数据之前，口线上的数据一直保持不变，但对输入信号是不锁存的，所以外设输入的数据必须保持到取数指令执行为止。

准双向口有三个上拉场效应管 T1、T2 和 T3，以适应不同的需要。其中，T1 称为“强上拉”，上拉电流可达 20mA；T2 称为“极弱上拉”，上拉电流一般为 30μA；T3 称为“弱上拉”，一般上拉电流为 150～270μA，典型值为 200μA。输出低电平时，灌电流最大可达 20mA。

当口线寄存器为“1”且引脚本身也为“1”时，T3 导通。T3 提供基本驱动电流，使准双向口输出为“1”。如果一个引脚输出为“1”，而由外部装置下拉到低电平时，T3 断开，而 T2 维持导通状态，为了把这个引脚强拉为低电平，外部装置必须有足够的灌电流，使引脚上的电压降到门槛电压以下。

当口线锁存为“1”时，T2 导通。当引脚悬空时，这个极弱的上拉源产生很弱的上拉电流，将引脚上拉为高电平。

当口线锁存器由“0”到“1”跳变时，T1 用来加快准双向口由逻辑“0”到逻辑“1”的转换。当发生这种情况时，T1 导通约两个时钟，使引脚能够迅速地上拉到高电平。

准双向口带有一个施密特触发输入以及一个干扰抑制电路。

当从端口引脚输入数据时，T4 应一直处于截止状态。假定在输入之前曾输出锁存过数据“0”，则 T4 是导通的。这样，引脚上的电位始终被钳位在低电平，使输入高电平无法读入。因此，若要从端口引脚读入数据，必须先将端口锁存器置“1”，使 T4 截止。

2）推挽输出工作模式

推挽输出工作模式下，I/O 口的电路结构如图 3-1-2 所示。

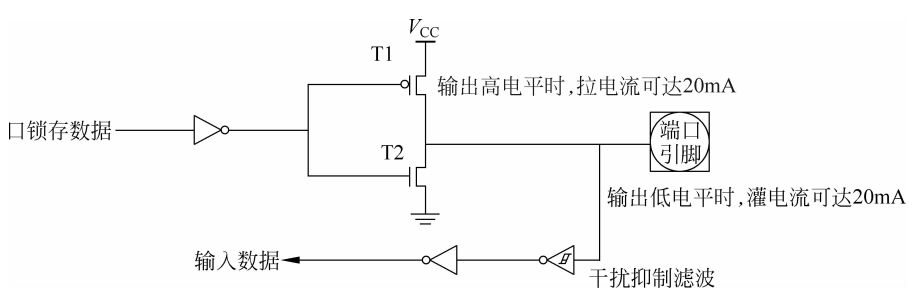

图 3-1-2　推挽输出工作模式下 I/O 口的电路结构

推挽输出工作模式下，I/O 口输出的下拉结构、输入电路结构与准双向口模式是一致的；不同的是，推挽输出工作模式下，I/O 口的上拉是持续的“强上拉”。若输出高电平，输

出拉电流最大可达 20mA；若输出低电平时，输出灌电流最大可达 20mA。

当从端口引脚输入数据时，必须先将端口锁存器置“1”，使 T2 截止。

3）仅为输入（高阻）工作模式

仅为输入（高阻）工作模式下，I/O 口的电路结构如图 3-1-3 所示。

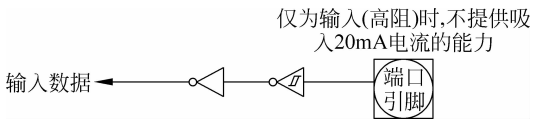

图 3-1-3　仅为输入（高阻）工作模式下 I/O 口的电路结构

仅为输入（高阻）工作模式下，可直接从端口引脚读入数据，不需要先对端口锁存器置“1”。

4）开漏输出工作模式

开漏输出工作模式下，I/O 口的电路结构如图 3-1-4 所示。

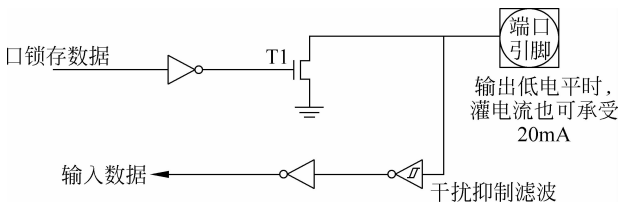

图 3-1-4　开漏输出工作模式下 I/O 口的电路结构

开漏输出工作模式下，I/O 口输出的下拉结构与推挽输出/准双向口一致，输入电路与准双向口一致，但输出驱动无任何负载，即开漏状态。输出应用时，必须外接上拉电阻。

3. IAP15W4K58S4 单片机并行 I/O 口的使用注意事项

1）典型三极管控制电路

单片机 I/O 引脚本身的驱动能力有限，如果需要驱动较大功率的器件，可以采用单片机 I/O 引脚控制晶体管进行输出的方法。如图 3-1-5 所示，如果用弱上拉控制，建议加上拉电阻 R_1，阻值为 3.3～10kΩ；如果不加上拉电阻 R_1，建议 R_2 的取值在 15kΩ 以上，或用强推挽输出。R_3 为要驱动的负载电阻。

2）典型发光二极管驱动电路

采用弱上拉驱动时，利用灌电流方式驱动发光二极管，如图 3-1-6(a)所示；采用推挽输出（强上拉）驱动时，利用拉电流方式驱动发光二极管，如图 3-1-6(b)所示。

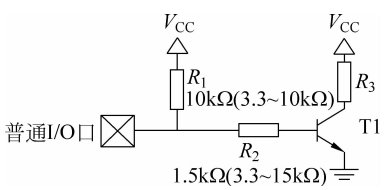

图 3-1-5　典型三极管控制电路

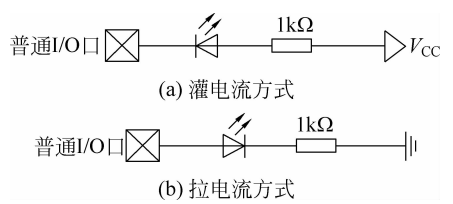

图 3-1-6　典型发光二极管驱动电路

在实际使用时，应尽量采用灌电流驱动方式，而不要采用拉电流驱动，以便提高系统的负载能力和可靠性。有特别需要时，可以采取拉电流方式，如供电线路要求比较简单时。

做行列矩阵按键扫描电路时，也需要加限流电阻。因为实际工作时可能出现两个 I/O 口均输出低电平的情况，并且在按键按下时短接在一起，而 CMOS 电路的两个输出脚不能直接短接在一起。在按键扫描电路中，一个口为了读另外一个口的状态，必须先置高，而单片机的弱上拉口在由 0 变为 1 时，会有两个时钟的强推挽输出电流，输出到另外一个输出低电平的 I/O 口，有可能造成 I/O 口损坏。因此，建议在按键扫描电路的两侧各加 300Ω 限流电阻，或者在软件处理上，不要出现按键两端的 I/O 口同时为低电平的情况。

3）如何让 I/O 口上电复位时控制输出为低电平

IAP15W4K58S4 单片机上电复位时，普通 I/O 口为弱上拉高电平输出，而很多实际应用要求上电时某些 I/O 口控制输出为低电平，否则所控制的系统（如电动机）就会误动作。为了解决这个问题，可采用以下两种方法。

（1）通过硬件实现高、低电平的逻辑取反功能。例如，在图 3-1-5 中，单片机上电复位后，晶体管 T1 的集电极输出低电平。

（2）由于 IAP15W4K58S4 单片机既有弱上拉输出模式又有强推挽输出模式，可在单片机 I/O 口上加一个下拉电阻（1kΩ、2kΩ 或 3kΩ），这样上电复位时，虽然单片机内部 I/O 口是弱上拉/高电平输出，但由于内部上拉能力有限，外部下拉电阻又较小，无法将其拉为高电平，所以该 I/O 口上电复位时外部输出低电平。如果要将此 I/O 口驱动为高电平，可将此 I/O 口设置为强推挽输出。此时，I/O 口驱动电流达 20mA，将该口驱动为高电平输出。实际应用时，先串联一个大于 470Ω 的限流电阻，再接下拉电阻到地，如图 3-1-7 所示。

特别提示：IAP15W4K58S4 单片机的 P2.0（RSTOUT_LOW）引脚上电复位后输出电平，通过 STC-ISP 在线编程软件设置为低电平输出。

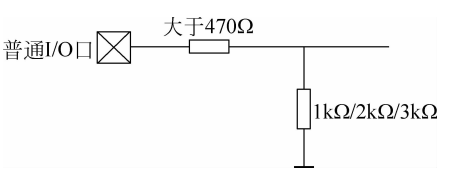

图 3-1-7　让 I/O 口上电复位时控制输出为低电平的驱动电路

4）增强型 PWM 模块输出端口的复位初始状态

所有与增强型 PWM 有关的输出端口（P3.7、P2.1、P2.2、P2.3、P1.6、P1.7）在上电复位后均为高阻状态，在正常应用时，需将其设置为准双向口。通常做法是用 C 语句将 IAP15W4K58S4 单片机所有 I/O 口设置为双向口模式，构成一个 I/O 的初始化函数（如本教材中的 gpio()），并存为一个独立的头文件（如本教材中的 gpio.h）。在编写 C 语言函数时，先用 #include<gpio.h> 将 gpio.h 头文件包含到应用程序中，再在主函数中直接调用 gpio.h 头文件中的 I/O 初始化文件 gpio()。

4. C51 基础（1）

C51 程序是在 ANSI C 的基础上拓展的，增加了针对 8051 单片机内部资源进行操作的语句，包括特殊功能寄存器与特殊功能寄存器可寻址位的地址定义、8051 单片机存储器存储类型的定义、中断函数等功能操作。

1）C 语言程序结构

（1）结构形式

```
#include<reg51.h>          //8051 单片机特殊功能寄存器与特殊功能寄存器位地址定义的文件
#include<intrins.h>        //8051 单片机常用函数的头文件(循环移位与空操作函数等)
#define uint unsigned int  //宏定义,uint 定义为无符号整型数据类型
#define uchar unsigned char //宏定义,uchar 定义为无符号字符型数据类型
/* ------------- 功能子函数 1 ------------------- */
fun1()
{
    函数体 1
}
/* ------------- 功能子函数 2 ------------------- */
fun2()
{
    函数体 2
}
⋮
/* ------------- 功能子函数 n ------------------- */
funn()
{
    函数体 n
}
/* ------------- 主函数 n ------------------- */
main()
{
    主函数体
}
```

（2）结构说明

函数是 C 语言程序的基本单位。一个 C 语言程序可包含多个不同功能的函数，但一个 C 语言程序中只能有一个且必须有一个名为 main()的主函数。主函数的位置可在其他功能函数的前面、之间或最后。当功能函数位于主函数的后面位置时，在调用主函数时，必须“先声明”。

C 语言程序总是从 main()主函数开始执行。主函数可通过直接书写语句或调用功能子函数来完成任务。功能子函数可以是 C 语言本身提供的库函数，也可以是用户自己编写的函数。

（3）库函数与自定义函数

库函数是针对一些经常使用的算法，经前人开发、归纳、整理形成的通用功能子函数。Keil C51 内部有数百个库函数供用户调用，调用 Keil C51 的库函数时，只需要包含具有该函数说明的相应的头文件即可，如#include＜reg51.h＞。当使用不同类型的单片机时，可包含其相应的头文件。若无专门的头文件，首先应包含典型的头文件，即 reg51.h，其他新增的功能符号直接用 sfr 语句定义其地址。

自定义函数是用户自己根据需要编写的子函数。

2）C51 变量定义

（1）标识符与关键字

标识符是用来标识源程序中某个对象的名字。这些对象可以是语句、数据类型、函数、变量、常量、数组等。

一个标识符由字符串、数字和下划线组成。第一个字符必须是字母和下划线。通常以下划线开头的标识符是编译系统专用的，因此在编写 C 语言源程序时一般不使用以下划线开头的标识符，而将下划线用作分段符。C51 编译器在编译时，只对标识符的前 32 个字符编译，因此在编写源程序时标识符的长度不要超过 32 个字符。在 C 语言程序中，字母是区分大小写的。

关键字是编程语言保留的特殊标识符，也称为保留字，它们具有固定名称和含义。在 C 语言的程序编写中，不允许标识符与关键字相同。ANSI C 标准一共规定了 32 个关键字，如表 3-1-2 所示。

Keil C51 编译器的关键字除了有 ANSI C 标准规定的 32 个之外，还根据 8051 单片机的特点扩展了相关关键字。对于在 Keil C51 开发环境的文本编辑器中编写的 C 程序，系统把保留字以不同颜色表示，默认颜色为蓝色。Keil C51 编译器扩展的关键字如表 3-1-3 所示。

表 3-1-2　ANSI C 规定的关键字

关　键　字	类　　型	作　　用
auto	存储种类说明	说明局部变量，默认值为此
break	程序语句	退出最内层循环体
case	程序语句	switch 语句中的选择项
char	数据类型说明	单字节整型数据或字符型数据
const	存储类型说明	在程序执行过程中不可更改的常量值
continue	程序语句	转向下一次循环
default	程序语句	switch 语句中的失败选择项
do	程序语句	构成 do…while 循环结构
double	数据类型说明	双精度浮点数
else	程序语句	构成 if…else 选择结构
enum	数据类型说明	枚举
extern	存储种类说明	在其他程序模块中说明了的全局变量
float	数据类型说明	单精度浮点数
for	程序语句	构成 for 循环结构
goto	程序语句	构成 goto 循环结构
if	程序语句	构成 if…else 选择结构
int	数据类型说明	基本整型数据
long	数据类型说明	长整型数据
register	存储种类说明	使用 CPU 内部寄存器变量
return	程序语句	函数返回
short	数据类型说明	短整型数据
signed	数据类型说明	有符号数据
sizeof	运算符	计算表达式或数据类型的字节数
static	存储种类说明	静态变量

续表

关 键 字	类 型	作 用
struct	数据类型说明	结构类型数据
switch	程序语句	构成 switch 选择结构
typedef	数据类型说明	重新进行数据类型定义
union	数据类型说明	联合类型数据
unsigned	数据类型说明	无符号数据
void	数据类型说明	无类型数据
volatile	数据类型说明	该变量在程序执行中可被隐含地改变
while	程序语句	构成 while 和 do...while 循环结构

表 3-1-3 Keil C51 编译器扩展的关键字

关 键 字	类 型	作 用
bit	位标量声明	声明一个位标量或位类型的函数
sbit	可寻址位声明	定义一个可位寻址变量地址
sfr	特殊功能寄存器声明	定义一个特殊功能寄存器(8 位)地址
sfr16	特殊功能寄存器声明	定义一个 16 位的特殊功能寄存器地址
data	存储器类型说明	直接寻址的 8051 单片机内部数据存储器
bdata	存储器类型说明	可位寻址的 8051 单片机内部数据存储器
idata	存储器类型说明	间接寻址的 8051 单片机内部数据存储器
pdata	存储器类型说明	“分页”寻址的 8051 单片机外部数据存储器
xdata	存储器类型说明	8051 单片机的外部数据存储器
code	存储器类型说明	8051 单片机程序存储器
interrupt	中断函数声明	定义一个中断函数
reentrant	再入函数声明	定义一个再入函数
using	寄存器组定义	定义 8051 单片机使用的工作寄存器组
small	变量的存储模式	所有未指明存储区域的变量都存储在 data 区域
large	变量的存储模式	所有未指明存储区域的变量都存储在 xdata 区域
compact	变量的存储模式	所有未指明存储区域的变量都存储在 pdata 区域
_at_	地址定义	定义变量的绝对地址
far	存储器类型说明	用于某些单片机扩展 RAM 的访问
alicn	函数外部声明	C 函数调用 PL/M-51,必须先用 alicn 声明
task	支持 RTX51	指定一个函数是一个实时任务
priority	支持 RTX51	指定任务的优先级

(2) 数据类型

C 语言的数据结构是以数据类型决定的。数据类型分为基本数据类型和复杂数据类型,复杂数据类型由基本数据类型构造而成。

C 语言的基本数据类型为 char、int、short、long、float、double。

① Keil C51 编译器支持的数据类型:对于 Keil C51 编译器来说,short 型与 int 型相同,double 型与 float 型相同。表 3-1-4 所示为 Keil C51 编译器支持的数据类型。

表 3-1-4　Keil C51 编译器支持的数据类型

数据类型定义符号	数据类型名称	长　　度	值　　域
unsigned char	无符号字符型数据	单字节	0～255
signed char	有符号字符型数据	单字节	－128～＋127
unsigned int	无符号整型数据	双字节	0～65535
signed int	有符号整型数据	双字节	－32768～＋32767
unsigned long	无符号长整型数据	4 字节	0～4294967295
signed long	有符号长整型数据	4 字节	－2147483648～＋2147483647
float	浮点数据	4 字节	±1.175494E－38～±3.402823E＋38
*	指针类型	1～3 字节	对象的地址
bit	位变量	位	0 或 1
sfr	8 位的特殊功能寄存器	单字节	0～255
sfr16	16 位特殊功能寄存器	双字节	0～65535
sbit	特殊功能寄存器位	位	0 或 1

② 数据类型分析如下所述。

- char：字符类型，有 unsigned char 和 signed char 之分，默认值为 signed char，长度为 1 字节，用于存放 1 个单字节数据。对于 signed char 型数据，其字节的最高位表示该数据的符号，“0”表示正数，“1”表示负数，数据格式为补码形式，表示的数值范围为－128～＋127；而 unsigned char 型数据是无符号字符型数据，表示的数值范围为 0～255。
- int：整型，有 unsigned int 和 signed int 之分，默认值为 signed int，长度为 2 字节，用于存放双字节数据。signed int 是有符号整型数，unsigned int 是无符号整型数。
- long：长整型，有 unsigned long 和 signed long 之分，默认值为 signed long，长度为 4 字节。signed long 是有符号长整型数，unsigned long 是无符号长整型数。
- float：浮点型，是符合 IEEE 754 标准的单精度浮点型数据。float 浮点型数据占用 4 字节(32 位二进制数)，其存放格式如下所示。

字节(偏移)地址	＋3	＋2	＋1	＋0
浮点数内容	SEEEEEEE	EMMMMMMM	MMMMMMMM	MMMMMMMM

其中：

- S 为符号位，存放在最高字节的最高位。“1”表示负，“0”表示正。
- E 为阶码，占用 8 位二进制数。E 值是以 2 为底的指数再加上偏移量 127，其目的是避免出现负的阶码值，而指数可正可负。阶码 E 的正常取值范围是 1～254，实际指数的取值范围为－126～＋127。
- M 为尾数的小数部分，用 23 位二进制数表示。尾数的整数部分永远为 1，因此不予保存，但它是隐含存在的。小数点位于隐含的整数位“1”的后面。一个浮点数的数值表示是$(-1)^{S}\times 2^{E-127}\times(1.M)$。
- 指针型：指针型数据不同于以上 4 种基本数据类型，它本身是一个变量，但在此变量

中存放的不是普通的数据，而是指向另一个数据的地址。指针变量也要占据一定的内存单元。在 Keil C51 中，指针变量的长度一般为 1～3 字节。指针变量也具有类型，其表示方法是在指针符号“*”的前面冠以数据类型符号，如“char *point”是一个字符型指针变量。指针变量的类型表示该指针所指向地址中数据的类型。

- bit：位标量。这是 C51 编译器的一种扩充数据类型，利用它可以定义一个位标量。

(3) 变量的数据类型选择

选择变量数据类型的基本原则如下所述。

① 若能预算变量的变化范围，可根据变量长度选择变量类型。应尽量缩短变量的长度。

② 如果程序中不需要使用负数，选择无符号数类型的变量。

③ 如果程序中不需要使用浮点数，要避免使用浮点数变量。

④ 数据类型之间的转换：在 C 语言程序的表达式或变量的赋值运算中，有时会出现运算对象的数据类型不一样的情况，C 语言程序允许在标准数据类型之间隐式转换。隐式转换按以下优先级别(由低到高)自动进行：

bit→char→int→long→float→signed→unsigned

一般来说，如果有几个不同类型的数据同时运算，先将低级别类型的数据转换成高级别类型，再做运算处理，并且运算结果为高级别类型数据。

3) 简单赋值运算

在 C 语言中，最常见的赋值运算符为“=”。利用赋值运算符将一个变量与一个表达式连接起来的式子称为赋值表达式。在赋值表达式后面加一个“;”，便构成语句。例如：

```
y = 6;                    //将 6 赋值给变量 y
y = x;                    //变量 x 的值赋给变量 y
```

4) 8051 单片机并行 I/O 口的 C51 编程

8051 单片机的并行 I/O 口(P0、P1、P2、P3)属于特殊功能寄存器，每个并行 I/O 口都有一个固定的地址，当使用 C51 的关键字(sfr)对并行 I/O 口进行地址定义后，并行 I/O 口的符号名称(P0、P1、P2、P3)在 C51 的编程中就可以直接使用了。

定义格式为：

```
sfr 特殊功能寄存器名 = 特殊功能寄存器的地址常数;
```

例如：

```
sfr P0 = 0x80;            //定义特殊功能寄存器 P0 口的地址为 80H
```

经过上述定义后，P0 这个特殊功能寄存器名称可直接使用。例如：

```
P0 = 0x80;                //将数值 80H 赋值给 P0 口
```

特别提示：Keil C 编译器包含对 8051 系列单片机各特殊功能寄存器及可寻址位定义的头文件 reg51.h。在程序设计时，只要利用包含指令将头文件 reg51.h 包含进来即可。对于增强型 8051 单片机，新增特殊功能寄存器需要重新定义。对于 STC 系列单片机，利用 STC-ISP 在线编程软件工具，可生成 STC 各系列单片机有关特殊功能寄存器以及可位寻址