单片机C语言中指针的应用

曹　聪　岳继光

摘　要　本文主要结合51单片机系统的硬件资源特点，从指针结构声明，对于不同存储区（片内、片外数据存储区和程序存储区）的指针寻址的实现等方面阐述了单片机C语言指针的应用。

关键词　单片机C　语言　指针

C是一种编译型语言，有高级语言的特点，并具备汇编语言的功能，移植性能好，便于自顶向下结构化程序设计。C语言在单片机中的应用，给开发者带来了很大的方便，软件开发者不需要对单片机硬件的结构有很深入的了解，编译器可以自动完成变量存储单元的分配，使得单片机的程序设计更加简单可靠。

指针、地址、数组及其相互关系是C语言中最有特色的部分。在编写单片机的应用程序时，常常需要对端口及存储单元进行寻址，因此，掌握指针在这些寻址过程的工作原理是很有必要的，这有利于编写灵活高效的程序。

一、指针的声明与结构

要灵活使用单片机C语言的指针，对其声明和结构有比较清晰的了解是必要的。

1. 单片机存储结构及其C语言存储类型

在进行单片机指针声明之前，首先应该先了解单片机的存储类型与存储区关系，物理上存储区可以分为四个部分：片内数据存储区、片外数据存储区、片内程序存储区和片外程序存储区。程序在运行时，内外部程序存储区是透明的，所以在逻辑上可以分为片内数据存储区、片外数据存储区和程序存储区。单片机由于型号不同，其片内RAM的大小是不一样的，如8051有128Byte片内RAM，8052有256Byte。

单片机C语言存储类型的存储区关系说明如下：

data：可寻址片内RAM，地址空间00H－7FH

bdata：可位寻址的片内RAM，20H－2FH的128bit位地址。

Idata：可寻址片内RAM，允许访问全部内部RAM，要使用单片机II28Byte以后存储空间时必须使用该存储类型。

Pdata：分页寻址片外RAM，汇编中采用MOVX@RO（256 BYTE/页）

Xdata：可寻址片外RAM（64K地址范围）

Code：程序存储区（64K地址范围），汇编中采用MOVE @DPTR

2. 指针的声明

C语言中，对于指针的声明采用如下形式：

类型标识符 * 指针变量名

由于单片机存储区的关系，所以单片机C语言的指针声明格式有别于普通C语言，其格式为：

类型标识符 ［存储区类型］ * ［指针变量存储区类型］ 指针变量名

①　②　③　④

单片机C语言的指针定义比普通C语言的指针定义多个两个部分：存储区类型是指指针变量所指向的数据的存储区，可以是以上所有的数据存储类型；指针变量存储区类型是指指针变量的存放区域，可以是data、idata、xdata或pdata。如：

unsigned char xdata * data yc

①　②　③ ④

是指在片内RAM区（③data）分配一个指针变量yc（④），这个指针指向一个无符号字符（①unsigned char），该无符号字符存放于xdata区（②xdata）。

应用时格式如下：

```
unsigned char xdata Indata[6]; (1)
unsigned char xdata * data yc;
yc＝InData;
```

其编译后的汇编为：

```
MOV 08H, #00H;0x08和0x09是在片内RAM区分配的yc指针变量地址空间
MOV 09H, #00H;
```

3. 指针声明的注意事项

声明格式中存储区类型和指针变量存储区类型可以省略，如果省略，不同的编译器的默认值是不一样的。建议最好不要省略，在声明中写明存储区类型能够大大提高程序的效率，也便于阅读。同时在片内RAM资源不是很紧张时最好将指针本身存放于data区，这样有利于提高程序的效率，以下这段C程序及其汇编后的汇编程序说明了这个问题。

```
Unsigned char * xdata yc;
Yc＝IdData;
```

编译后的汇编代码如下：

```
MOV DPTR, #0006H;yc指针变量地址空间0006H－0008H
MOV A, #02H
MOVX @DPTR, A
MOV A, #00H
```

曹　聪　同济大学电子与信息工程学院　上海　200092
岳继光　同济大学电子与信息工程学院　上海　200092

```
INC DPTR
MOVX @DPTR,A
INC DPTR
MOV A,#00H
MOVX @DPTR,A
```

与(1)中的程序相比,可知,这样的定义浪费了程序空间,也降低了效率(以上采用KeiC 编译器,不同的51 编译器产生的汇编代码不全相同)。

另外,一般说来基于51 系列的系统架构单片机的内部RAM 资源都很紧张,最好在定义函数内部或程序段内部的局部变量使用片内RAM,而尽量不要把全局变量声明为片内RAM 区中。

二、指针寻址的实现

单片机C 语言的数组寻址和普通C 语言的基本上实现是一样的,只是当数组存储在片内时,由于片内RAM 资源十分有限,所以很难有比较复杂的数据结构,而且在编程过程中也尽量避免在片内RAM 中使用较大的数组。对于片外数据存储和程序存储空间的指针访问实现,单片机有其自身的特点。

1. 指向data 区的指针寻址实现

这是最基本的寻址方法,比如一个检测系统中,通过A/D 转换把外部数据输入单片机,单片机对这6 次采样数据求和,这6 次的数据都在data 区中。程序实现例1。

```
Unsigned char data Input—Data[6];
Unsigned int data sum, I;
Unsigned char data * data yc;
Void main()
{
sum=0;
Collect—Data();/ * 采集数据函数,输入到数组Input—Data 中 * /
Yc=Input—Data;
For(I=0;I<=6;I++,yc++)
Sum+= * yc;
}
```

可以看出,这里指针的应用和普能C 语言是差不多的。

2. 指向片外数据存储区的拭指针寻址实现

由于单片机的片内数据存储区有限,所以在单片机应用中常常要扩展片外数据存储区,对于片外数据存储区和扩展端口采用指针寻址可以方便的进行数据的读写操作。

笔者在开发汉字字库程序时有这样的要求:汉字的显示码固化于外部扩展的32K 的EEPROM 中。每个汉字为16×16 的点阵,所以每个汉字含有32 字节的点阵显示码,在操作过程中,根据汉字的区位码,计算出汉字的存储首地址,从这个首地址中读出汉字的显示码送至液晶显示模块。大体程序如下:

```
主程序部分
{
...
unsigned char Code—Dotarray[4];/ * 定义字符数组,前两位存储汉字区位码,后两位用于存储返回的点阵显示码的首地址 * /
...
Area—Dot(unsigned char Code—Dotarray);/根据区位码得到点阵显示码的函数 * /
Led—Dis(unsigned char * Code—Dotarray);/ * 调用液晶显示函数 * /
...
}
液晶显示函数部分
void Led—Dis(unsigned char * Code—Dotarray)
{
unsigned char xdata * data yc;/ * 定义指向首地址的指针 * /
yc=(unsigned char xdata * )(Code—Dotarray[2]+9(unsigned int)
(Code—Dotarray[3]))*0x100+0x50);/ * 通过强制计算得到汉字点阵字符的首地址yc,引用 * yc 及可以得到相应的显示码字符。 * /
}
```

本例中,也可以直接返回指针变量,只需要将函数定义返回形式稍作改动,和C 语言中的一样,不再做说细介绍。

以上是对于大容量程序存储区的指针寻址实现形式,对于一些固定的端口可以先用XBYTE 定义,再用指针指向端口。如:

```
# define 0809—IN0 XBYTE[0x8000]/ * 定义0809 通道0 地址0x8000x/
unsigned char xdata * data ad—adr;/ * 定义指针 * /
```

引用时用如下语句:

```
ad—adr=0809—INO;
```

3. 指向程序存储区的指针寻址实现

指针指向数据存储区其实质就是C 语言中指向函数的指针这一概念,可以利用这种指针来实现用函数指针调用函数。指向函数的指针变量的定义格式为:

```
类型标识符(*指针变量名)([参数1],[参数2],...);
```

在定义好指针后就可以给指针变量赋值,使其指向某个函数的开始存地址,然后用(*指针变量名)([参数],[参数2,...]

即可调用这个函数。

例如,主程序中要引用一个键盘扫描scan 函数,程序如下:

```
void scan()
void main()
{
void( * yc)();/ * 定义指向函数指针 * /
yc=scan;//给指针yc 赋值,yc 指向程序存储区
for(;;)
```

```
{
(*yc)();//调用函数
...
}
}
```

如果scan函数本身已经固化在程序存储区，地址已知，在0x6000中时，也可以直接给yc指定数值：

```
yc=0x6000;
```

调用格式相同。

三、结束语

单片机C语言的应用是十分灵活的，要充分发挥C语言的优势，对内外部数据和程序进行方便自如的操作，必须要掌握好指针的应用。文中所述是作者在利用单片机开发智能楼宇对讲系统、汉字字库系统和磁滞电机监控系统等项目时的一些应用体会，旨在抛砖引玉，一起探讨如何更好的发挥单片机C语言指针的作用。

参考文献

[1]马忠梅等 单片机的C语言应用程序设计 北京航空航天大学出版社 2001

[2]朱宇光等 单片机应用新技术教程 北京电子工业出版社 2000

[3]张友德等 单片微型机原理、应用与实验 上海复旦大学出版社 1998

(收稿日期:2003年5月3日)

(上接49页)

```
Sender)
    {DynamicArray<byte>ByteReceive;-//定义动态数组以接收数据
    int number—byte;
    ByteReceive=Comm1—>ReadInputByte();//接收字节数据
    number—byte=ByteReceive.Length;//数据的长度
    //然后可对接收的每个字节数据分别处理，如ByteReceive[0]、ByteReceive[1]……
    DynamicArray<byte>send;//定义动态数组以存储要发送的数据
    send.Length=7;//发送数据的长度
    send[0]=OxAA;send[1]=OxAA;send[2]=0x9F;send[3]=OxE0;
    send[4]=Ox00;send[5]=OxAA;send[6]=OxAD;
    Comml—>OutputByte(send);//发送返回数据，告诉检测装置计算机已接收到数据
    ……
    }
```

③ 关闭通信端口，系统退出后关闭通信资源

```
Comml->PortOpen==false;
```

五、结束语

随着计算机应用领域的不断扩展，计算机与外部设备之间的通信也越来越广泛。目前讨论串行通信的技术文献虽然很多，但基于C++Builder字节通信的文献介绍较少。本文对创建TComm通信控件的方法做了探讨，并介绍了其使用。实践证明，利用TComm控件开发基于字节的串口通信程序具有代码简洁、应用方便等优点，克服了MSComm控件的缺点。详细代码需要的读者可与作者联系。Email:zm10308@sina.com ZML0308@163.com

参考文献

[1]范逸之，江文贤，陈立元. C++Builder与RS—232串行通信控制，北京：清华大学出版社，2002

[2]罗日成，李卫国. C++Builder5中基于MSComm控件串口通信的编程与实现. 机电工程. 2002(1). 24—27

(收稿日期:2003年3月12日)

(上接56页)

法的进一步研究，如何降低网络的通信流量以及提高搜索的准确性上面。

参考文献

[1]朱国进、陈家训，Web资源查找机理剖析，微型电脑应用，1999年第6期

[2]潘春华、常敏、武港山，面向Web的信息收集工具的设计与开发，计算机应用研究，2002年第6期

[3]何凌云、孙恒、王命延，Web信息自动搜索系统的设计与研究，计算机与现代化，2002年第6期，总第82期

[4]霍艳蓉，Web信息检索的关键技术，现代图书情报技术，2002年第6期，总第97期

[5]Subrahmanyam Allamaraju等，J2EE服务器端高级编程，机械工业出版社，2001年9月

[6]James William Allamaraju, James W. Cooper, Jave Design Patterns

[7]RFC—2068, 2616 http://www.ietf.org/iesg/lrfc—index.txt

(收稿日期:2003年6月25日)