1. Warning 280:'i':unreferenced local variable

说明局部变量 i 在函数中未作任何的存取操作解决方法消除函数中 i 变量的宣告及即定义的参数在程序中并未调用

2 Warning 206:'Music3':missing function-prototype

说明 **Music3( )**函数未作宣告或未作外部宣告所以无法给其他函数调用

解决方法将叙述 **void Music3(void)**写在程序的最前端作宣告如果是其他文件的函数则要写成 **extern void Music3(void),**即作外部宣告

3Error:318:can't open file 'beep.h'

说明在编译 **C:\8051\MANN.C** 程序过程中由于 **main.c** 用了指令＃**include "beep.h",**但却找不到所致解决方法编写一个 **beep.h** 的包含档并存入到 **c:\8051** 的工作目录中

4 Error 237:'LedOn':function already has a body

说明 **LedOn( )**函数名称重复定义即有两个以上一样的函数名称

解决方法修正其中的一个函数名称使得函数名称都是独立的

5 ***WARNING 16:UNCALLED SEGMENT,IGNORED FOR OVERLAY PROCESS

SEGMENT: ?PR?_DELAYX1MS?DELAY

说明 **DelayX1ms( )**函数未被其它函数调用也会占用程序记忆体空间

解决方法去掉**DelayX1ms( )**函数或利用条件编译**#if …..#endif,**可保留该函数并不编译

6 ***WARNING 6 :XDATA SPACE MEMORY OVERLAP

FROM : 0025H

TO: 0025H

说明外部资料 **ROM** 的 **0025H** 重复定义地址

解决方法外部资料 **ROM** 的定义如下 **Pdata unsigned char XFR_ADC _at_0x25** 其中 **XFR_ADC** 变量的名称为 **0x25,**请检查是否有其它的变量名称也是定义在 **0x25** 处并修正它

7 WARNING 206:'DelayX1ms': missing function-prototype

C:\8051\INPUT.C

Error 267 :'DelayX1ms ':requires ANSI-style prototype

C:\8051\INPUT.C

说明程序中有调用 **DelayX1ms** 函数但该函数没定义即未编写程序内容或函数已定义但未作宣告

解决方法编写 **DelayX1ms** 的内容编写完后也要作宣告或作外部宣告可在 **delay.h** 的包含档宣告成外部以便其它函数调用

8 ***WARNING 1:UNRESOLVED EXTERNAL SYMBOL

SYMBOL:MUSIC3

解决办法：
1.是文件没有添加到工程里。
2.可能是因为存在没有被调用的已经定义的函数。
3.不知道你有没有把 Source group 组下面的 A51.C 删掉，如果没有删，在 A51.c 上点右键，选择 remove file " ".
4.建一个新的 c 文件，里面写一个空的函数，把该文件添加到 project 中，注意该文件不能再选 generate assembler SRC file 和 assemble SRC file 选项。重新编译工程，如果警告该函数没被调用，在主文件中调一下。
5. 建一个新的 c 文件，把主文件中的几个函数移至该文件，把该文件添加到 project 中，注意该文件不能再选 generate assembler SRC file 和 assemble SRC file 选项。重新编译工程

9*WARNING 2:REFERENCE MADE TO UNRESOLVED**

EXTERNAL

SYMBOL:MUSIC3

MODULE:C:\8051\MUSIC.OBJ(MUSIC)

ADDRESS:0018H

在 **MUSIC3** 函数里面 **MUSIC** 这个参数有使用，没有申明。或者申明了没有实体。也就是说对于这个参数，编译器无法解析。

10 ***ERROR 107:ADDESS SPACE OVERFLOW

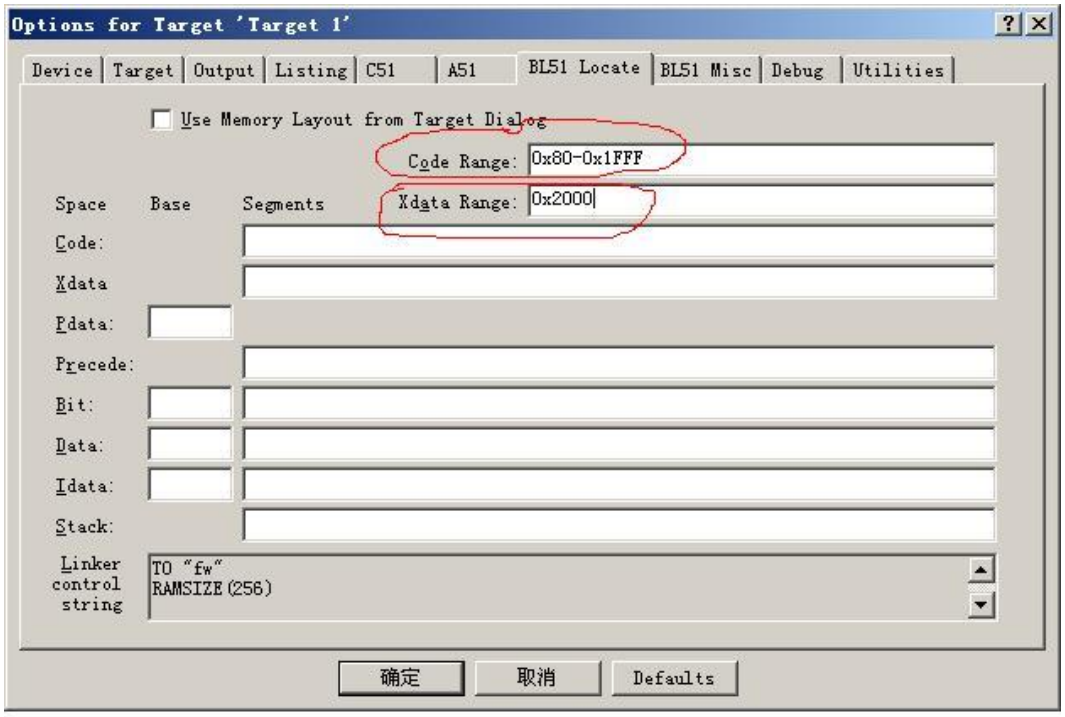
SPACE: DATA

SEGMENT: _DATA_GOUP_

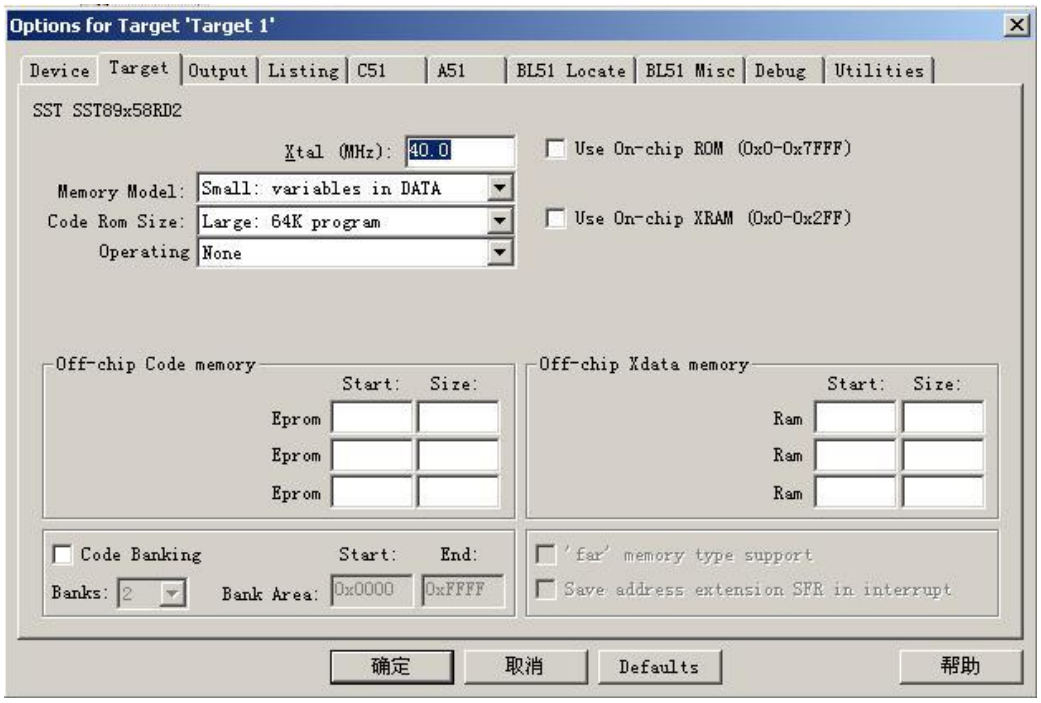
LENGTH: 0018H

说明 data 空间已经不够用，原因是你可能有好多函数，而函数内部的局部变量又没有定义其空间，这种情况下，系统会将变量分配到你在 Otions for Target 对话框里的设置的空间。如果你在下图所示中的 Memory Model 里设置成 Small:variables in DATA,则 DATA 空间很快便用完，导致 data 空间不够用。解决的办法有两种，一是通过更改 Memory Model 设置，可以设置成 pdata 或 xdata，以便有足够大的空间，但这又带来新的问题，程序运行速度减慢，而且 code 代码也会加大，因为如果一个局部变量被存放在了 xdata 空间，汇编语言访问 xdata 空间的代码大小要比访问 data 空间的代码大，变量一旦很多，程序的代码也会逐渐增大；二是根据自己的要求设置变量的空间。所以这涉及到代码优化的问题，遇到具

体问题时，在运行速度和代码大小之间取得适合自己的情况



Project---->Option for target---->BL51 Locate 选项卡，如上图红圈部分所示，根据自己系统的存储器分布情况，可以设置代码区间和 XDATA 区间。通常默认情况下，代码区间很小，所以会造成 107 号错误，根据需求，调大该范围即可。



11ERROR L105: PUBLIC REFERS TO IGNORED SEGMENT

SYMBOL: USARTDATACOUNT

SEGMENT: ?DT?USART_READWRITE

Ram 空间不足：

外部变量：定义处不用加 External，声明处要加 External

将以 data 型别定义的公共变量修改为 idata 型别的定义

12*ERROR 118: REFERENCE MADE TO ERRONEOUS EXTERNAL**

SYMBOL: VOLUME

MODULE: C:\8051\OSDM.OBJ (OSDM)

ADDRESS: 4036H

定义了某某函数或全部变量在不同文件里面想调用它,却在包含头文件里面少了 extern 语句,或只有主程序和包含头文件没有(EXTERN.....定义语句(函数)).

如果调试时有些 if 结构里的语句符合条件没有执行，或者某些语句不符合条件也被执行，那是因为 if 和 else 里有相同的语句，编译的时候作优化处理了。

13 WARNING 15 (MULTIPLE CALL TO SEGMENT) 症状

原因

Warning 15 向我们表明了 linker 发现了一个函数，这个函数不仅在 main code 里被调用了，而且在 ISR（或者被 ISR 调用的函数中）被调用了。或者是被同时被多个 ISR 同时调用了。

这样会产生一个问题，就是在此函数不是一个可重入函数，而当此函数已经在执行时它可能被另一个 ISR 所调用。这样就会导致结果是可变的而且很可能会导致一些参数的错误。

另一个问题就是本地变量和参数所使用的内存可能被其他函数的内存覆盖。如果函数是由中断所调用的，则此函数的内存就会被使用。这会引起其它函数的内存错误。

解决方法

有几种方法去解决这个问题

如果你 100%确认这个函数的两个副本都不会同时执行（如果此函数是被 main 调用并且中断是未被使能的）并且此函数没有使用内存（只使用的寄存器），那么你就可以忽略此警告

如果此函数使用了内存，你就要使用 OVERLAY directive 来将此函数从覆盖分析（overlay anaysis）中移除。举例如下：

```
OVERLAY (?PR?_WRITE_GMVLX1_REG?D_GMVLX1 ! *)
```

如上语句能阻止被此函数使用的内存遭到覆盖。如果这个函数调用了你程序中其他的在别处的函数，那么你可能需要将这些函数也排除在覆盖分析之外。

如果当此函数在执行时可以被调用，那么事情就会变得比较的复杂。你可能需要：无论何时当从 main 中调用此函数时，需要关闭中断。你可能需要对被调用的函数使用#pragma disable。你也必须使用

OVERLAY directive 将此函数从 overlay analysis 中移除。 为此函数创建两个副本。一个给 main，一个给 ISR。

使此函数可重入。

14E:\VCWORK\2815.C(826): error C236: '_wrbyte': different length of parameter lists

子函数里的形参声明的方式不对，需要每个参数都定义一下类型

E:\VCWORK\2815.C(743): error C183: unmodifiable lvalue

出现 error C183: unmodifiable lvalue 的错误，最后发现时存在一个数组是 uchar code xx[5],后边把它用作接受串口的缓冲区，显示 uchar code 是不能改变的，是写在 rom 中的。应该改成 uchar xx[5]，这是写在 ram 中的

原因：修改了不能改变的变量，

E:\VCWORK\2815.C(799): error C242: 'array[]': too many initializers

15 ERROR L104: MULTIPLE PUBLIC DEFINITIONS SYMBOL: _WRITE_DATA MODULE: .\ds18b20start.obj (DS18B20

c/c++语言中有很多地方要用到 extern，但是如果没有真正的了解它的意义，会给编程带来很大的麻烦，为了使大家少走弯路，特祥细的说明一下。

对于比较小的程序，一般只有一个 c 文件和一个头文件,全局变量我们通常会直接定义在 c 文件中，在程序之前加 int i 定义。如果要在头文件中定义有以下两种方法：用 extern 来声明:extern int i;这一句只是对变量 i 进行声明，在 c 文件的程序之前必须加上 int i 进行定义。extern int i=0;这一句声明和定义都做了。

对于大一点的程序，有很多 c 文件和头文件，这个时候全局变量就必须在头文件中声明(不需要初始化)，然后在一个 c 文件中定义(该初始化的要初始化)。如果在头文件中定义，则编译的时候会出现重复定义的错误。如果只有头文件中声明就会出现没有定义有警告。

*** ERROR L104: MULTIPLE PUBLIC DEFINITIONS
SYMBOL: K
MODULE: 222.obj (222)

出现上述错误则是因为变量 k 重复定义，把你的头文件中的变量定义前加 extern(只是变量声明不用初始化)，再在某一个你要调用该变量的 c 文件的程序之前再定义(注意第一个调用的 c 文件要负责附带初始化该变量，其他调用的 c 文件就不需要初始化过程啦)

14MAIN.C(85): warning C259: 'parameter': pointer: different mspace

原因，函数调用时候的实参和声明时候的形参存储空间不同，修改成一致即可。

16 E:\VC\2815\2815\FTOC.C(32): warning C231: '_memcpy': attempt to redefine intrinsic function

17*** ERROR L121: IMPROPER FIXUP

访问内存指令超出指令的寻址范围了,例如 MOVX @Ri 指令超出了 PDATA 段的范围,或者是 ACALL 指令超出了 2k 的寻址范围.

检查你的调用子函数的命令.特别是那些 LCALL,ACALL 等

18*** WARNING L2: REFERENCE MADE TO UNRESOLVED EXTERNAL

SYMBOL: MAIN
MODULE: C:\KEIL\C51\LIB\C51S.LIB (?C_INIT)
ADDRESS: 080DH

在 main 函数里面 C_INIT 这个参数有使用，没有申明。或者申明了没有实体。也就是说对于这个参数，编译器无法解析。

19 keil4 warning C316:unterminated conditionals

今天用 Keil4 写程序时遇到这个问题：warning C316:unterminated conditionals

跑了几个论坛，在审视了一遍代码之后，知道了原因：

像类似 XX.C(99):warning C316:unterminated conditionals 这种警告的话 XX.c 文件有一个凌乱的条件编译或预编译。因为 C 语言中有些头文件中的预编译或宏定义，那么条件编译就避免不了。写条件编译时，可能有忘写一个基本的语句。比如说，你用了条件编译 #ifndef 而忘记写#endif。因为他们本来就是配套的。有前者必有后者。不能丢掉其中任何一个。一个 include 文件最后的#endif 少了# 前缀或者没有#endif，都会出现类似警告。就像你写 C 语句，你不会写了 int i 而不能忘记写 " ; " ，否者就不能把一个语句表达完整。

总之，出现上述问题。先看看整个 C 文件中是否出现上述错误，或整个工程中自己写的那些头文件中里面的条件编译是否都写对了，即：前面写了#ifndef，后面是否有对应的 #endif。

20 DS1302.C(86): error C183: unmodifiable lvalue

code 的内容只能读,不能改. 定义数组时把 code 去掉。

21 keil 编译警告‘Argument':conversion:pointer to non-pointer 是什么问题

应该是参数传递错误，指针参数处传递了非指针参数。

22 *** ERROR L114: SEGMENT DOES NOT FIT

块大小与目标设备不符。段溢出了,你的 DATA 区超过了 256 字节
你的 idata 变量太大（CEH），与器件容量不匹配。可能你的单片机型号选成 31 了，选个 256 字节内部 RAM 的应该就行，将定义为 data 的变量定义为 xdata 类型，问题解决了。

23 error C193 :bad operand type

% 取模不能用浮点数，
frequence 要转成整型来取模，小数位可以乘 10 后转整型来得到。

24 常见错误

error 1: Out of memory 内存溢出
error 2: Identifier expected 缺标识符
error 3: Unknown identifier 未定义的标识符
error 4: Duplicate identifier 重复定义的标识符
error 5: Syntax error 语法错误
error 6: Error in real constant 实型常量错误
error 7: Error in integer constant 整型常量错误
error 8: String constant exceeds line 字符串常量超过一行
error 10: Unexpected end of file 文件非正常结束
error 11: Line too long 行太长
error 12: Type identifier expected 未定义的类型标识符
error 13: Too many open files 打开文件太多
error 14: Invalid file name 无效的文件名
error 15: File not found 文件未找到
error 16: Disk full 磁盘满
error 17: Invalid compiler directive 无效的编译命令
error 18: Too many files 文件太多
error 19: Undefined type in pointer def 指针定义中未定义类型
error 20: Variable identifier expected 缺变量标识符
error 21: Error in type 类型错误
error 22: Structure too large 结构类型太长
error 23: Set base type out of range 集合基类型越界
error 24: File components may not be files or objectsfile 分量不能是文件或对象
error 25: Invalid string length 无效的字符串长度

error 26: Type mismatch 类型不匹配
error 27：error 27：Invalid subrange base type 无效的子界基类型
error 28：Lower bound greater than upper bound 下界超过上界
error 29：Ordinal type expected 缺有序类型

error 30：Integer constant expected 缺整型常量
error 31：Constant expected 缺常量
error 32：Integer or real constant expected 缺整型或实型常量
error 33：Pointer Type identifier expected 缺指针类型标识符
error 34：Invalid function result type 无效的函数结果类型
error 35：Label identifier expected 缺标号标识符
error 36：BEGIN expected 缺 BEGIN
error 37：END expected 缺 END
error 38：Integer expression expected 缺整型表达式
error 39：Ordinal expression expected 缺有序类型表达式
error 40：Boolean expression expected 缺布尔表达式
error 41：Operand types do not match 操作数类型不匹配
error 42：Error in expression 表达式错误
error 43：Illegal assignment 非法赋值
error 44：Field identifier expected 缺域标识符
error 45：Object file too large 目标文件太大
error 46：Undefined external 未定义的外部过程与函数
error 47：Invalid object file record 无效的 OBJ 文件格式
error 48：Code segment too large 代码段太长
error 49：Data segment too large 数据段太长
error 50：DO expected 缺 DO

error 51：Invalid PUBLIC definition 无效的 PUBLIC 定义
error 52：Invalid EXTRN definition 无效的 EXTRN 定义
error 53: Too many EXTRN definitions 太多的 EXTRN 定义
error 54：OF expected 缺 OF
error 55：INTERFACE expected 缺 INTERFACE
error 56：Invalid relocatable reference 无效的可重定位引用
error 57：THEN expected 缺 THEN
error 58：TO or DOWNTO expected 缺 TO 或 DOWNTO
error 59：Undefined forward 提前引用未经定义的说明
error 61：Invalid typecast 无效的类型转换
error 62：Division by zero 被零除
error 63：Invalid file type 无效的文件类型
error 64：Cannot read or write variables of this type 不能读写此类型变量
error 65：Pointer variable expected 缺指针类型变量
error 66：String variable expected 缺字符串变量
error 67：String expression expected 缺字符串表达式
error 68：Circular unit reference 单元 UNIT 部件循环引用
error 69：Unit name mismatch 单元名不匹配
error 70：Unit version mismatch 单元版本不匹配
error 71：Internal stack overflow 内部堆栈溢出
error 72：Unit file format error 单元文件格式错误
error 73：IMPLEMENTATION expected 缺 IMPLEMENTATION

error 74：Constant and case types do not match　　常量和 CASE 类型不匹配
error 75：Record or object variable expected　　缺记录或对象变量

error 76：Constant out of range　　常量越界
error 77：File variable expected　　缺文件变量
error 78：Pointer expression expected　　缺指针表达式
error 79：Integer or real expression expected　　缺整型或实型表达式
error 80：Label not within current block　　标号不在当前块内
error 81：Label already defined　　标号已定义
error 82：Undefined label in preceding statement part　在前面未定义标号
error 83：Invalid @ argument　　无效的@参数
error 84：UNIT expected　　缺 UNIT
error 85: ";" expected　　缺“；”
error 86： ":" expected　　缺“：”
error 87： "," expected　　缺“，”
error 88： "(" expected　　缺“（”
error 89： ")" expected　　缺“）”
error 90： "=" expected　　缺“=”
error 91： ":=" expected　　缺“:=”
error 92： "[" or "(." Expected　　缺“[”或“（.”
error 93: "]" or ".)" expected　　缺“］”或“.）”
error 94： "." expected　　缺“.”
error 95: ".." expected　　缺“..”
error 96：Too many variables　　变量太多
error 97：Invalid FOR control variable　　无效的 FOR 循环控制变量
error 98：Integer variable expected　　缺整型变量
error 99：Files and procedure types are not allowed here　该处不允许文件和过程类型
error 100：String length mismatch　　字符串长度不匹配

25 error C2085: 'Delete' : not in formal parameter list

不在形参列表中，意思是在函数定义后丢掉了‘;’

25..\QXJ\TEST.C(371): error C100: unprintable character 0xA1 skipped

在语句后面出现了不该出现的字符。

二、C 语言浮点数的存储方式

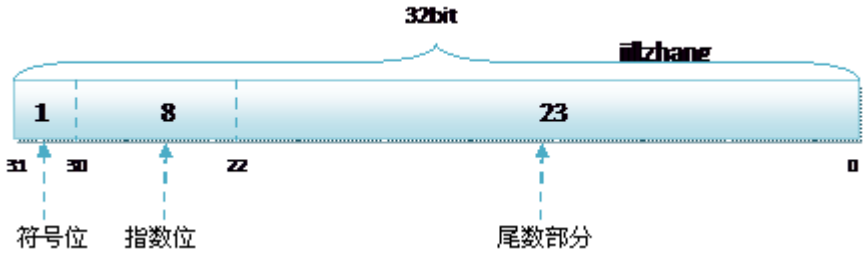
C 语言和 C#语言中，对于浮点类型的数据采用单精度类型（float）和双精度类型(double)来存储，float 数据占用 32bit,double 数据占用 64bit,我们在声明

一个变量 float f= 2.25f 的时候，是如何分配内存的呢？如果胡乱分配，那世界岂不是乱套了么，其实不论是 float 还是 double 在存储方式上都是遵从 IEEE 的规范的，float 遵从的是 IEEE R32.24 ，而 double 遵从的是 R64.53。

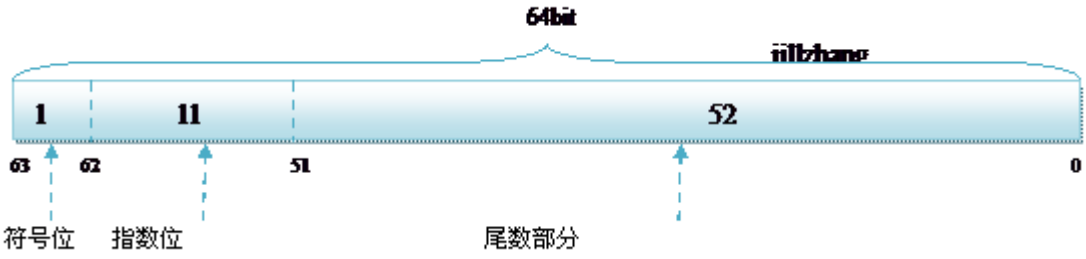
无论是单精度还是双精度在存储中都分为三个部分：

1. 符号位(Sign) ： 0 代表正，1 代表为负
2. 指数位（Exponent）:用于存储科学计数法中的指数数据，并且采用移位存储
3. 尾数部分（Mantissa）：尾数部分

其中 float 的存储方式如下图所示：



而双精度的存储方式为:

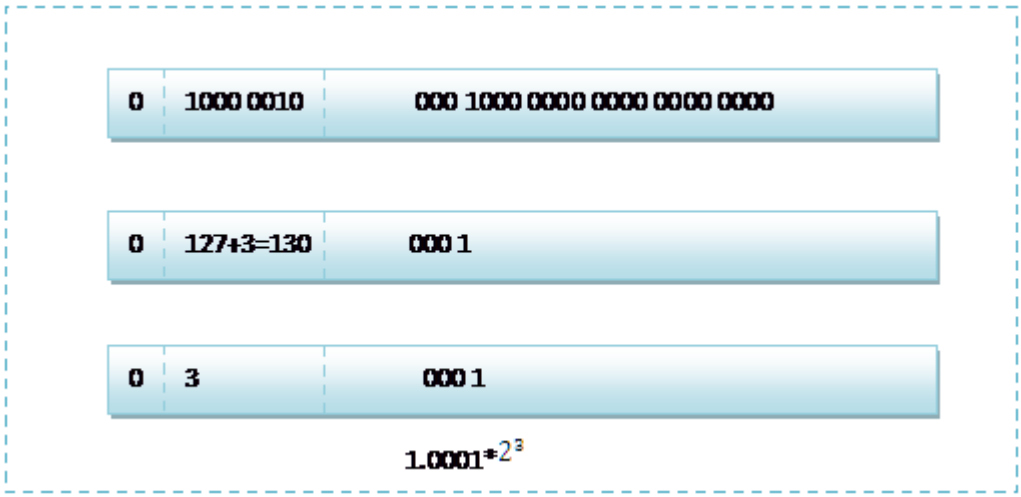


R32.24 和 R64.53 的存储方式都是用科学计数法来存储数据的，比如 8.25 用十进制的科学计数法表示就为:$8.25*10^0$,而 120.5 可以表示为:$1.205*10^2$,这些小学的知识就不用多说了吧。而我们傻蛋计算机根本不认识十进制的数据，他只认识 0，1，所以在计算机存储中，首先要将上面的数更改为二进制的科学计数法表示，8.25 用二进制表示可表示为 1000.01,我靠，不会连这都不会转换吧?那我估计要没辙了。120.5 用二进制表示为：1110110.1 用二进制的科学计数法表示 1000.01 可以表示为 $1.0001*2^3$,1110110.1 可以表示为 $1.1101101*2^6$,任何一个数都的科学计数法表示都为 $1.xxx*2^n$,尾数部分就可以

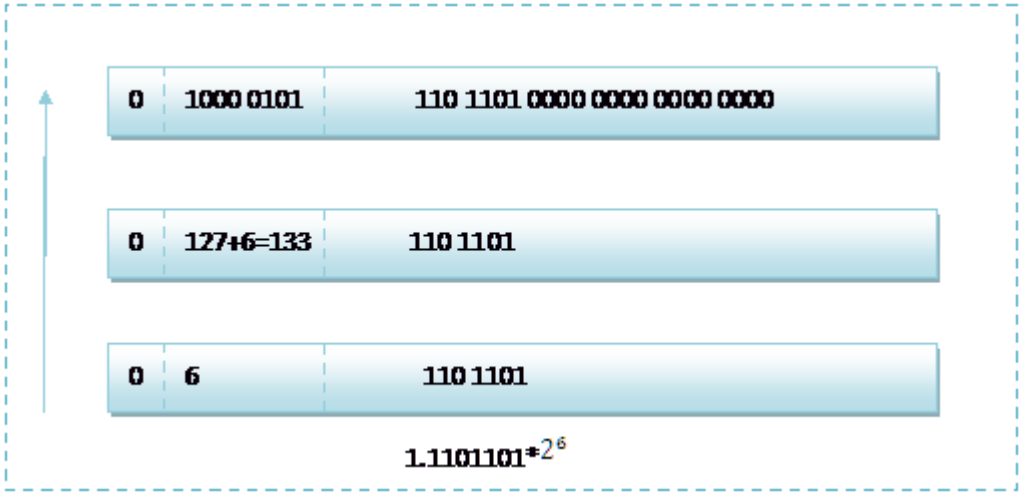
表示为 xxxx,第一位都是 1 嘛，干嘛还要表示呀？可以将小数点前面的 1 省略，所以 23bit 的尾数部分，可以表示的精度却变成了 24bit，道理就是在这里，那 24bit 能精确到小数点后几位呢，我们知道 9 的二进制表示为 1001，所以 4bit 能精确十进制中的 1 位小数点，24bit 就能使 float 能精确到小数点后 6 位，而对于指数部分，因为指数可正可负，8 位的指数位能表示的指数范围就应该为:-127-128 了，所以指数部分的存储采用移位存储，存储的数据为元数据+127，下面就看看 8.25 和 120.5 在内存中真正的存储方式。

首先看下 8.25，用二进制的科学计数法表示为:$1.0001*2^3$

按照上面的存储方式，符号位为:0，表示为正，指数位为:3+127=130 ,位数部分为,故 8.25 的存储方式如下图所示:



而单精度浮点数 120.5 的存储方式如下图所示:



那么如果给出内存中一段数据，并且告诉你是单精度存储的话，你如何知道该数据的十进制数值呢？其实就是对上面的反推过程，比如给出如下内存数据：

01000010111011010000000000000，首先我们现将该数据分段，0 10000 0101 110 1101 0000 0000 0000 0000，在内存中的存储就为下图所示：

根据我们的计算方式，可以计算出，这样一组数据表示为:$1.1101101*2^{6}$=120.5

而双精度浮点数的存储和单精度的存储大同小异，不同的是指数部分和尾数部分的位数。所以这里不再详细的介绍双精度的存储方式了，只将 120.5 的最后存储方式图给出，大家可以仔细想想为何是这样子的

下面我就这个基础知识点来解决一个我们的一个疑惑，请看下面一段程序，注意观察输出结果

```
float f = 2.2f;
double d = (double)f;
Console.WriteLine(d.ToString("0.0000000000000"));
f = 2.25f;
d = (double)f;
Console.WriteLine(d.ToString("0.0000000000000"));
```

可能输出的结果让大家疑惑不解，单精度的 2.2 转换为双精度后，精确到小数点后 13 位后变为了 2.2000000476837，而单精度的 2.25 转换为双精度后，变为了 2.2500000000000，为何 2.2 在转换后的数值更改了而 2.25 却没有更改呢？很奇怪吧？其实通过上面关于两种存储结果的介绍，我们已经大概能找到答案。首先我们看看 2.25 的单精度存储方式，很简单 0 1000 0001 001 0000 0000 0000 0000 0000，而 2.25 的双精度表示为:0 100 0000 0001 0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000，这样 2.25 在进行强制转换的时候，数值是不会变的，而我们再看看 2.2 呢，2.2 用科学计数法表示应该为：将十进制的小数转换为二进制的小数的方法为将小数*2，取整数部分，所以 0.282=0.4，所以二进制小数第一位为 0.4 的整数部分 0，0.4×2=0.8，第二位为 0，0.8*2=1.6，第三位为 1，0.6×2 = 1.2，第四位为 1，0.2*2=0.4，第五位为 0，这样永远也不可能乘到=1.0，得到的二进制是一个无限循环的排列 00110011001100110011...，对于单精度数据来说，尾数只能表示 24bit 的精度，所以 2.2 的 float 存储为:

但是这样存储方式，换算成十进制的值，却不会是 2.2 的，应为十进制在转换为二进制的时候可能会不准确，如 2.2，而 double 类型的数据也存在同样的问题，所以在浮点数表示中会产生些许的误差，在单精度转换为双精度的时候，也会存在误差的问题，对于能够用二进制表示的十进制数据，如 2.25，这个误差就会不存在，所以会出现上面比较奇怪的输出结果

三、xdata idata pdata data

DATA: 固定指前面 0x00-0x7f 的 128 个 RAM,可以用 acc 直接读写的,速度最快,生成的代码也最小。

IDATA:固定指前面 0x00-0xff 的 256 个 RAM,其中前 128 和 dATa 的 128 完全相同,只是因为访问的方式不同。

IDATA 是用类似 C 中的指针方式访问的。汇编中的语句为：mox ACC,@Rx.(不重要的补充：c 中 IDATA 做指针式的访问效果很好)

XDATA: 外部扩展 RAM,一般指外部 0x0000-0xffff 空间,用 DPTR 访问。

PDATA: 外部扩展 RAM 的低 256 个字节,地址出现在 A0-A7 的上时读写,用 movx ACC,@Rx 读写。这个比较特殊,而且 C51 好象有对此 BUG, 建议少用。但也有他的优点,具体用法属于中级问题,这里不提。

STARTUP.A51 的作用,和汇编一样,在 C 中定义的那些变量和数组的初始化就在 STARTUP.A51 中进行,如果你在定义全局变量时带有数值,如 unsigned char dATa xxx="100";,那 STARTUP.A51 中就会有相关的赋值。如果没有=100, STARTUP.A51 就会把他清 0。（STARTUP.A51==变量的初始化）。这些初始化完毕后,还会设置 SP 指针。对非变量区域, 如堆栈区,将不会有赋值或清零动作。有人喜欢改 STARTUP.A51 为了满足自己一些想当然的爱好,这是不必要的,有可能错误的。比如掉电保护的时候想保存一些变量, 但改 STARTUP.A51 来实现是很笨的方法,实际只要利用非变量区域的特性,定义一个指针变量指向堆栈低部：0xff 处就可实现。, 为什么还要去改? 可以这么说：任何时候都可以不需要改 STARTUP.A51,如果你明白它的特性。

bit 是在内部数据存储空间中 20H .. 2FH 区域中一个位的地址，这在 DATA 的 20H 以后以字节形式出现，可互相参照。另外加上 8051 可寻址 的 SFR，但刚刚试过，只是 00H--7FH 起作用，也就是说当数据有变化时颜色变红，以后的从 80H 到--FFH 就不是位寻址区了，是位寻址的特殊寄存 器,如涉及到了可位寻址的那 11 个当然会有反应。复位后,程序计数器 PC 的内容为 0000H，内部 RAM 各单元的值不确定。各功能寄存器的复位值如下：堆栈指针 SP 的复位值为 07H，累加器 ACC、寄存器 B 的复位值为 00H，数据指针 DPTR 的复位值为 0000H，而 p0、p1、p2、p3 四个口的复位值为 0FFH。其他 SFR 如 PSW、TCON、TMOD、TL0、TH0、TL1、TH1 的复位值也为 00H。

wave 中是低 128 字节和高 128 字节（0-7FH），低 128 字节是片内 RAM 区，高 128 字节（80-FFH）是 SFR（特殊功能寄存器）bit 则是位于低 128 字节的 20H .. 2FH 区域，即 data 的 20H .. 2FH 区域

code 是在 0000H .. 0FFFFH 之间的一个代码地址。

我用 ORG 5000H

TAB: DB 22H,3BH,43H,66H,5H,6DH,88H 后，

CODE 从 5000H 开始以后变成 DB 各位

DATA 是 在 0 到 127 之间的一个数据存储器地址，或者加 128 .. 255 范围内的一个特殊功能寄存器（SFR）地址。两者访问的方式不同。实际上由于 PSW 的复位设置 PSW.3=RS0 和 PSW.4=RS1 皆为 0，所以通用工 作寄存器区就是第 0 区，所以 data 的 00--07H 部分是与 REG 栏中的 R0--R7

对应的。以后的则仅代表低 128 字节的内部 RAM。
IDATA 是 0 to 255 范围内的一个 IDATA 存储器地址, IDATA 与 DATA 重合低 128 字节，有的地方只有 DATA 表示 256 字节的片内 RAM,
XDATA 是 0－ 65535 范围内的一个 XDATA 存储器地址。

指针类型和存储区的关系详解
一、存储类型与存储区关系
data ---> 可寻址片内 ram
bdata ---> 可位寻址的片内 ram
idata ---> 可寻址片内 ram，允许访问全部内部 ram
pdata ---> 分页寻址片外 ram (MOVX @R0) (256 BYTE/页)
xdata ---> 可寻址片外 ram (64k 地址范围 FFFFH)
code ---> 程序存储区 (64k 地址范围),对应 MOVC @DPTR

二、指针类型和存储区的关系

对变量进行声明时可以指定变量的存储类型如:
uchar data x 和 data uchar x 相等价都是在内 ram 区分配一个字节的变量。同样对于指针变量的声明,因涉及到指针变量本身的存储位置和指针所指向的存储区位置不同而进行相应的存储区类型关键字的使用如：
uchar xdata * data pstr
是指在内 ram 区分配一个指针变量("*"号后的 data 关键字的作用)，而且这个指针本身指向 xdata 区("*"前 xdata 关键字的作用)，可能初学 C51 时有点不好懂也不好记。没关系，我们马上就可以看到对应“*”前后不同的关键字的使用在编译时出现什么情况。
uchar xdata tmp[10]; //在外 ram 区开辟 10 个字节的内存空间,地址是外 ram 的 0x0000－0x0009
.第 1 种情况:
uchar data * data pstr;
pstr="tmp";
首先要提醒大家这样的代码是有 bug 的, 他不能通过这种方式正确的访问到 tmp 空间。为什么？我们把编译后看到下面的汇编代码：
MOV 0x08,#tmp(0x00) ;0x08 是指针 pstr 的存储地址
看到了吗！本来访问外 ram 需要 2 byte 来寻址 64k 空间，但因为使用 data 关键字(在"*"号前的那个)，所以按 KeilC 编译环境来说就把他编译成指向内 ram 的指针变量了，这也是初学 C51 的朋友们不理解各个存储类型的关键字定义而造成的 bug。特别是当工程中的默认的存储区类为 large 时，又把 tmp[10] 声明为 uchar tmp[10] 时，这样的 bug 是很隐秘的不容易被发现。
第 2 种情况:
uchar xdata * data pstr;
pstr = tmp;
这种情况是没问题的，这样的使用方法是指在内 ram 分配一个指针变量("*"号后的 data 关键字的作用)，而且这个指针本身指向 xdata 区("*"前 xdata 关键字的作用)。编译后的汇编代码如下。
MOV 0x08,#tmp(0x00) ;0x08 和 0x09 是在内 ram 区分配的 pstr 指针变量地址空间
MOV 0x09,#tmp(0x00)
这种情况应该是在这里所有介绍各种情况中效率最高的访问外 ram 的方法了，请大家记住他。
第 3 种情况:

uchar xdata * xdata pstr;
pstr="tmp";
这中情况也是对的，但效率不如第 2 种情况。编译后的汇编代码如下。
MOV DPTR, #0x000A ;0x000A,0x000B 是在外 ram 区分配的 pstr 指针变量地址空间
MOV A, #tmp(0x00)
MOV @DPTR, A
INC DPTR
MOV A, #tmp(0x00)
MOVX @DPTR, A
这种方式一般用在内 ram 资源相对紧张而且对效率要求不高的项目中。
第 4 种情况:
uchar data * xdata pstr;
pstr="tmp";
如果详细看了第 1 种情况的读者发现这种写法和第 1 种很相似，是的，同第 1 种情况一样这样也是有 bug 的，但是这次是把 pstr 分配到了外 ram 区了。编译后的汇编代码如下。
MOV DPTR, #0x000A ;0x000A 是在外 ram 区分配的 pstr 指针变量的地址空间
MOV A, #tmp(0x00)
MOVX @DPTR, A
第 5 种情况:
uchar * data pstr;
pstr="tmp";
大家注意到"*"前的关键字声明没有了，是的这样会发生什么事呢？下面这么写呢！对了用齐豫的一首老歌名来说就是 “请跟我来”，请跟我来看看编译后的汇编代码，有人问这不是在讲 C51 吗？ 为什么还要给我们看汇编代码。C51 要想用好就要尽可能提升 C51 编译后的效率，看看编译后的汇编会帮助大家尽快成为生产高效 C51 代码的高手的。还是看代码吧！
MOV 0x08, #0X01 ;0x08－0x0A 是在内 ram 区分配的 pstr 指针变量的地址空间
MOV 0x09, #tmp(0x00)
MOV 0x0A, #tmp(0x00)
注意：这是新介绍给大家的，大家会疑问为什么在前面的几种情况的 pstr 指针变量都用 2 byte 空间而到这里就用 3 byte 空间了呢？这是 KeilC 的一个系统内部处理，在 KeilC 中一个指针变量最多占用 3 byte 空间，对于没有声明指针指向存储空间类型的指针，系统编译代码时都强制加载一个字节的指针类型分辩值。具体的对应关系可以参考 KeilC 的 help 中 C51 User's Guide。
第 6 种情况:
uchar * pstr;
pstr="tmp";
这是最直接最简单的指针变量声明，但他的效率也最低。还是那句话，大家一起说好吗！编译后的汇编代码如下。
MOV DPTR, #0x000A ;0x000A－0x000C 是在外 ram 区分配的 pstr 指针变量地址空间
MOV A, #0x01
MOV @DPTR, A
INC DPTR
MOV DPTR, #0x000A
MOV A, #tmp(0x00)
MOV @DPTR, A

INC DPTR
MOV A, #tmp(0x00)
MOVX @DPTR, A
这种情况很类似第 5 种和第 3 种情况的组合，既把 pstr 分配在外 ram 空间了又增加了指针类型的分辨值。

keil 错误

C51 编译器识别错类型有三种

1、致命错误：伪指令控制行有错，访问不存在的原文件或头文件等。

2、语法及语义错误：语法和语义错误都发生在原文件中。有这类错误时，给出提示但不产生目标文件，错误超过一定数量才终止编译。

3、警告：警告出现并不影响目标文件的产生，但执行时有可能发生问题。程序员应斟酌处理。

D.1 致命错误

C_51 FATAL_ERROR

ACTION: <当前行为>

LINE: <错误所在行>

ERROR: <错误信息> terminated

或 C_51 FATAL ERROR

ACTION: <当前行为>

FILE: <错误所在文件>

ERROR: <错误信息> terminated

C_51 TERMINATED C_51

(1) ACTION 的有关信息

*PARSING INVOKE-/#PRAGMA_LINE

在对#pragma 指明的控制行作此法分析时出错。

*ALLOCATING MEMORY

系统分配存储空间时出错。编译较大程序需要 512k 空间。

*OPENING INPUT_FILE

打开文件时，未找到或打不开源文件/头文件。

*CREATE LIST_FILE/OBJECT_FILE/WORK_FILE

不能创建上述文件。可能磁盘满或文件已存在而且写保护。

*PARSING SOURCE_FILE/ANALYZING DECLARATIONS

分析源程序时发现外部引用名太多。

*GENERATING INTERMEDIATE CODE

源代码被翻译成内部伪代码，错误可能来源于函数太大而超过内部极限。

*WRITING TO FILE

在向文件（work,list,prelist 或 object file）写时发生错误。

（2）ERROR 的有关信息

*MEMORY SPACE EXHAUSTED

所有可用系统空间耗尽。至少需要 512k 字节空间。没有足够空间，用户必须检查常驻内存的驱动程序是否太多。

*FILE DOES NOT EXIST

FILE 行定的文本文件名未发现。

*CAN'T CREAT FILE

FILE 行定义的文件不能被创建。

*SOURCE MUST COME FROMA DISK_FILE

源文件和头文件必须存在于硬盘或软盘上。控制台、CON、CI 或类似设备不允许作为输入文件。

*MORE THAN 256 SEGMENTS/PUBLICS/EXTERNALS

受 OMF_51 的历史限制，一个源程序不能超过 256 个各种函数的类型段，256 个全局变量，256 个公共定义或外部引用名。不使用为变量可以减少使用的段数。使用 static 存储类型说

明符可减少全局变量的使用数目。合理调整定义性说明的位置可减少外部引用名的使用数目。

*FILEWRITE ERROR

当向 list、prelist、work 或 object 文件中写内容时，由于空间不够而发生错误。

*NON_NULLARGUMENT EXPECTED

所选的控制参数需要一个括号内的变量，如一个文件夹或一个数。

*"（"AFTER CONTROL EXPECTED

变量的左括号丢失。

*"）"AFTER PARAMETER EXPECTED

变量的右括号丢失。

*RESPECIFIED OR CONFLICTING CONTROL

所选的控制参数与前面发生冲突或重复，例如 CODE 和 NOCOND

*BAD DECIMAL NUMBER

控制参数的数字含有非法数，需要使用十进制数。

*OUT OF RANGE DECIMAL NUMBER

控制参数的数字越界，例如 OPTIMIZE 的参量为 0-5。

*IDENTIFIER EXPECTED

控制参数 DEFINE 需要一个标识符做参量，与 C 语言的规则相同。

*PARSE STACK OVERFLOW

分析栈溢出。可能是源程序包含特别复杂的表达式，或功能块嵌套数超过 15。

*PREPOCESSOR: MACRO TO NESTED

宏扩展期间，预处理器的栈耗用太大。表明宏嵌套太多，或有递归宏定义。

*PREPOCESSOR: LINE TOO LONG(510)

宏扩展后行超过 510 个字符。

*CAN'T HAVE GENERAL CONTROL IN INVOCATION LINE

一般控制（如 EJECT）不能是命令行的一部分，应将它们放入源文件"pragma"预处理行中。

D.2 语法及语义错误

D.2.1 错误格式

这类错误在列表文件中产生如下格式的信息：

***ERROR<number>IN LINE<line>OF<file>:error message

***WARING<number>IN LINE<line>OF<file>:warning message

<number>表示错误行；

<line>表示源文件或头文件中与错误或警告相关的行；

<file>指明了错误所在的源文件或头文件；

D.2.2 错误信息及可能发生的原因

*ERROR100:unprintable character 0x??skipped

源文件中发现非法字符（注意，注解内的字符不做检查）。

*ERROR101:unclosed string

串未用引号结尾。

*ERROR 102:string too long

串不得超过 511 个字符。为了定义更长的串，用户必须使用续行符'\'逻辑的继续该串，在词汇分析时遇到以该符号结尾的行会与下行连接起来。

*ERROR 103: invalid character constant

试图再声明一个已定义的宏，已存在的宏可以用#undef 指令删除。预定义的宏不能删除。

*ERROR 104: identifier expected

预处理器指令期望产生一个标示符，如 ifdef<name>。

*ERROR 105: unclosed comment

当注解无结束界定符（*/）时产生此错误。

*EROOR 106: unbalanced#if-endif controls

endif 的数量与 if 或 ifdef 的数量不匹配。

*ERROR 107:include file nesting exceeds 9

include 指令后的文件名无效或丢失

*ERROR 108: expected string,如#error “string”

预处理器指令期望一个串变量。

*ERROR 109:<user error text>

由#error 伪指令引入的错误信息以错误信号形式显示。

*ERROR 110:missing directive

预处理行#后缺少伪指令。

*ERROR 111:unknown directive

预处理行#后不是伪指令。

*ERROR 112:misplaced ‘elif’

*ERROR 113:misplaced ‘else’

*ERROR 114:misplaced ‘endif’

指令 elif/else/endif 只有在 if、ifdef、ifndef 指令内才是合法的。

*ERROR 117:bad integer expression

if/elif 指令的数值表达式有语法错误。

*ERROR 118:missing’(‘after macro identifier

宏调用中实参表的左括号丢失

*ERROR 119：reuse of macro formal parameter

宏定义形参名重复使用

*ERROR 120：‘C’unexpected in formal list

形参表中不允许有字符‘c’，应用逗号代替

*ERROR 121：missing ‘)’after actual parameter

宏调用实参表的右括号丢失

*ERROR 122：illegal macro invocation

宏调用的实参表与宏定义中的形参表不同

*ERROR 123：missing macro name after ‘define’

#define 伪指令后缺预定义的宏

*ERROR 124：expected macro formal parameter

宏定义要求形参名

*ERROR 125：declarater too complex

说明过于复杂

*ERROR 126：type-stack underflow

对象的声明至多只能包含 20 个类型修饰符（【】，*，（），）错误 126 经常在错误 125 之前，两者一起发生。

*ERROR 127：invalid storage class

对象用无效的存储类所说明。当在函数外用 auto/register 存储时会发生这种情况。

*ERROR 128：memory space: illegal memory space ‘memory space ’used

函数参数的存储类由存储模式（SMALL LARGE COMPACT）决定用户不能改变，使用不同

于存储模式的自动变量应该为静态的存储类

*ERROR 129：missing’ ;’before ‘token’

该错误表示分号丢失，通常该错误会引发一连串的错误，引发的这些错误无关紧要。因为缺少分号后编译器不能做正确的语法分析

*ERROR 130：value out of range

using 或 interrupt 指令后参数越限。using 用的寄存器组号位 0-3，interrupt 需要 0-15 的中

断号

*ERROR 131：duplicate function-parameter

函数中形参名重复，形参名应彼此不同

*ERROR 132：not in formal parameter list

函数内参数声明使用的名字未出现在参数表中

*ERROR 133：char function(v0,v1,v2)

Char*v0,*v1,*v5;/ *’v5’在形参中未出现

*ERROR 134：xdata/idata/pdata/data on function not permitted

函数总是驻留于 0x5xxxx 的 code 存储区，不能位于 xdata/idata/pdata/data 空间

*ERROR 135：bad storage class for bit

位变量的定义可以接受 static 或 extern 的存储类，用 REGIESTER 和 ALIEN 都是非法的

*ERROR 136：‘void’on variable

‘void’类型只允许作为函数的返回类型或与指针类型合用（void*）

*ERROR 137：illegal parameter type:’function’

函数参数的类型不能是函数，然而函数指针可以作为参数

*ERROR 138：interrupt ()may not receive or return value (s)

中断函数既不能有参数又不能有返回值

*ERROR 139：illegal use of ‘alien’

关键字 alien 将函数定义为 PL/M51 规定的过程与函数结构。这意味着 C 函数中有参数的缩

记符号（即 funct(…);）时是不能用 alien 的

*ERROR 140：bit in illegal memory-space

位变量的定义可包含修饰符 DATA,如果无修饰符则假定为 DATA。因为位变量始终位于 0x4xxx 的内部数据存储器中，当试图采用其他存储空间就会产生这个错误。

*ERROR 141：NEAR<token>:expected<token>

编译器所见的单词是错误的。期望正确的单词

*ERROR 142：invalid base address

Sfr 说明中的基址有错。有效基址为 0x80-0xff。如果声明采用 base^pos 形式，则基址是 8 的

整数倍

*ERROR 143：invalid absolute bit position

Sbit 说明中位地址必须在 0x80-0xff 之间

*ERROR 144：base^pos：invalid bit position

Sbit 说明中位 pos 必须在 0~7 之间

*ERROR 145：undeclared sfr

Sfr 未说明

*ERROR 146：invalid sfr

绝对位地址说明（base^pos）包含无效的基地址。这个基地址必须与 sfr 名相对应

*ERROR 147：object too large

对象不能超过 65536（64k）字节

*ERROR 148：field not permitted in union

联合不能包含位成员，这个限制是由 8051 结构决定的。

*ERROR 149：function member in struct/union

结构或者联合不能包含函数类型的成员。但是指向函数的指针是允许的

*ERROR 150：bit member in struct/union

结构或者联合不能包含位类型的成员，这个限制是由 8051 结构决定的

*ERROR 151：self relative struct/union

结构或者联合不能包含自身

*ERROR 152：bit field type too small for number of bits

位域声明中指定的位数超过所给原型中位的数量

*ERROR 153：named bit-field cannot have 0 width

命名的域宽度为 0 错误，只有未命名的位域允许是 0 宽度

*ERROR 154：pro to field

无指向位域指针的类型

*ERROR 155：char/int required for fields

位域基类型要求 char 或 int 类型，unsigned char 或 unsigned int 也有效

*ERROR 156：alien permitted on function only

Alien 只能用于函数

*ERROR 157：var_parms on alien function

有变参数的函数不能用 alien ,因为 PL/M51 函数只能用固定数量的参数

*ERROR 158：function contains unnamed parameter

函数定义的参数表中包含无名参数。无名参数只允许用于函数的原型中

*ERROR 159：type follows void

函数原型声明中可含一个空的参数表 f（void）。Void 后不能再用其他类型定义

*ERROR 160：void invalid

Void 类型只能与指针合用或表明函数没有返回值

*ERROR 161：formal parameter ignored

函数内的外部函数引用声明使用了无类型的参数表。例如“ extern(a,b,c);”要求形参表

*ERROR 162：duplicate function-parameter

函数内参数名重复

*ERROR 163：unknown array size

一般的不管是一维数组还是多维数组或外部数组，都需要指定数组的大小，这个大小是由编译器的初始化时计算，这个错误表明试图为一个未定维的数组使用 sizeof 运算符，会哦着一

个多维数组的附加元素未定义。

*ERROR 164：ptr to null

这一个错误通常是由前一个错误造成的

*ERROR 165：ptr to bit

指向位的指针不是合法的类型

*ERROR 166：array of function

数组不能包含函数，但可能包含指向函数的指针。

*ERROR 167：array of fields

位域不能安排为数组

*ERROR 168：array of bit

数组没有位类型

*ERROR 169：function returns function

函数不能返回函数，但可以返回一个指向函数的指针

*ERROR 170：function returns array

函数不能返回数组，但可返回指向数组的指针

*ERROR 171：missing enclosing swith

Break/continue 语句只能出现在 for,while,do while 或 switch 语句中间。

*ERROR 172：missing enclosing swith

Case 语句只能用在 switch 语句中

*ERROR 173：missing return-expression

返回类型不是 integer 的函数必须包含一条代表达式的 return 语句。由于要与老版本兼容，编译器对返回整形值的函数不做检查

*ERROR 174：return-expression on void-function

Void 函数不能返回值，因此不能包含带表达式 return 的语句

*ERROR 175：duplicate case value

每个 case 语句必须包含一个常量表达式做其变量，这个值不能在 switch 语句的各级中出现

多次

*ERROR 176：more than one default

Switch 语句中不能包含多于一个的 default 语句

*ERROR 177：different struct/union

赋值或参数传递中使用了结构/联合的不同类型

*ERROR 178：struct/union comparison illegal

根据 ANSI C，两个结构或联合的比较是不允许的

*ERROR 179：cannot/cast from/to void-type

将 void 类型转化为其他类型数据或将其他类型转化为 void 类型都是非法的

*ERROR 180：cannot cast to’function’

转化为 function 是非法的，使用函数指针指向不同的函数

*ERROR 181：incompatible operand

在所给的运算符中至少有一个操作符类型是无效的

*ERROR 182：point to different object

报告指针使用不一致

*ERROR 183：unmodifiable value

预修改的对象位于 CODE 存储区，因而不能修改

*ERROR 184：sizeof :illegal operand

Sizeof 运算符不能决定函数或位域大小

*ERROR 185：different memory space

对象说明的存储空间与前面的不一致

*ERROR 186：invalid dereference

这条错误信息可能由编译器内部问题产生的

*ERROR 187：not an lvalue

所需参量必须是可变对象的地址

*ERROR 188：unknown object size

无法计算对象的大小，因为缺少数组的维数或因为通过 void 指针的间接访问

*ERROR 189：‘&’on bit/sfr illegal

地址操作符‘&’不允许用于位对象或 sfr

*ERROR 190：’&’:not an lvalue

地质部是可变的对象，不能作为左值

*ERROR 191：‘&’on constant

试图为所列类型常数建立指针

*ERROR 192：‘&’on array/function

地址操作符&不允许用于数组或函数，函数和数组本身都代表了地址

*ERROR 193：illegal op-type(s)

*ERROR 193：illegal add/sub on ptr

*ERROR 193：illegal operation on bit(s)

*ERROR 193：bad operand type

当一个表达式使用给定运算符的非法操作类型时就会出现该错误，使用个定运算符的非法操作类型的无效的表达式，例如 bit+bit,ptr+ptr,或 ptr*<any>。错误信息包括引起错误的运算符。

下列运算可使用位操作符

赋值（=）

OR/复合 OR（|，|=）

AND/复合 AND（&,&=）

XOR/复合 XOR（^,^=）

位或常数的安位比较（= =,!=）

取反（~）

ERROR 194：’’indirection to object of unknown size

间接操作符*不能用于 void 指针(void*)，因为指针所指的对象的大小时未知的

ERROR 195：’’illegal indirection

间接操作符*不能用于非指针变量

*ERROR 196：mspace probably invalid

产生此警告是因为某些常数值赋给指针并且常数没有形成一个有效的指针值，有效的指针常数类型为 long/unsigned long。编译器对指针对象采用 24bit（3 字节），低 16 位表示偏移，高

8 位表示存储类的选择，在低字节中，值从 1 到 5 表明了 xdata/pdata/idata/data/和 code 的存

储类

*ERROR 197：illegal pointer assignment

试图将一个非法变量赋给指针，只有另一个指针或指针变量可以赋给指针

*ERROR 198：size of returns zero

求某些对象长度得到 0，如果对象是外部的或一个数组中并非所有维的大小都是已知时得到 0，这时候该值可能是错的。

*ERROR 199：left size of’->’requires struct/union pointer

->操作符的左边变量必须是结构或变量

*ERROR 200：left size of ‘.’requires struct/union

操作符的左边变量必须是结构/联合

*ERROR 201：undefined struct/union tag

所给的结构/联合标记名是未知的

*ERROR 202：undefined identifier

所给的标示符未定义

*ERROR 203：bad storage class(nameref)

该错误表示编译器的内部有问题

*ERROR 204：undefined member

所给的结构/联合成员名未定义

*ERROR 205：cannot call an interrupt function

中断函数不能像普通函数那样调用，因为这种函数的头端和尾端是为中断特殊编码的

*ERROR 206：missing function-prototype

调用的函数缺少原型说明

*ERROR 207：declared with ‘void’ parameter list

用 void 参数说明的函数不接受调用者传来的参数

*ERROR 208：too many actual parameter

函数调用包含了多余的实参

*ERROR 209：too few actual parameter

函数调用时传递的实参过少

*ERROR 210：too many nested calls

超过了 10 个函数嵌套调用的极限

*ERROR 211：call not to a function

函数调用时没有函数的地址或未对指向函数的指针赋值

*ERROR 212：indirect call with parameter

由于参数传递方法的限制，通过指针的间接函数调用不能直接作为实参。这种参数传递方法要求被调用的函数名已知，因为参数的写入要被写入调用函数的数据段。然而间接调用时函数的的名字时未知的

*ERROR 213：left side of assign_op not an lvalue

在赋值操作符左边要求可变的对象

*ERROR 214：cannot cast non_pointer to pointer

非指针不能转化为指针

*ERROR 215：cannot cast pointer to not_int/pointer

指针可以转化为另一个指针或整数，但不能转化为其他类型

*ERROR 216：subscript on non_array or too many dimensions

对非数组使用了下标或数组维数过多

*ERROR 217：non_integral index

数组的下标表达式必须是整型类型

*ERROR 218：void_type in controlling expression

While,for 或 do while 语句中表达式不能是 void 类型

*ERROR 219：long constant truncated to int

企图把长整型常量截断为整型数是错误的

*ERROR 220：illegal constant expression

非法常量表达式

*ERROR 221：non_constant case/dim expression

Case 值或下标值（[]）要求用常量表达式

*ERROR 222：div by zero

*ERROR 223：mod by zero

编译器检测到 0 除或 0 模的错误

*ERROR 224：illegal operation on float/double

AND 和 NOT 一类的运算符不允许作用于 float/double

*ERROR 225：expression too complex ,simplify

表达式太复杂，必须简化

*ERROR 226：duplicate struct/union/enum tag

结构/联合/枚举类型中有重复标记

*ERROR 227：not a union tag

所给的标记名虽已定义，但不是联合的标记

*ERROR 228：not a struct tag

所给的标记名虽已定义，但不是结构的标记

*ERROR 229：not an enum tag

所给的标记名虽已定义，但不是枚举的标记

*ERROR 230：unknown struct/union/enum tag

所给的结构/联合/枚举标记名未定义

*ERROR 231：redefinition

所给的名字已经定义，不能再定义

*ERROR 232：duplicate label

所给的标号已经订义

*ERROR 233：undefined label

当对函数进行分析后，编译器检查到函数有未定义的标号，发出错误信息

*ERROR 234：‘{’scope stack overflow(31)

超过了最大为 31 个的功能嵌套极限，多余的块被忽略

*ERROR 235：parameter<number>:different type

函数实参类型与函数原型中的不同

*ERROR 236：different length of parameter lists

所给的函数是参量与函数原型中的不同

*ERROR 237：function already has body

试图定义已经定义过的函数

*ERROR 238：duplicate member

*ERROR 239：duplicate parameter

重复定义结构成员或函数参数

*ERROR 240：more than 128 local bit’s

位变量定义总数不能超过 128

*ERROR 241：auto segment too large

局部对象要求的空间超过了该模式的最大值。最大栈长定义如下：SMALL-128 字节,COMPACT-256 字节，LARGE-64k

*ERROR 242：too many initializers

初始化对象超限

*ERROR 243：string out of bounds

串中字符数超过了字符数组要求初始化的字符数

*ERROR 244：can’t initialize .bad type or class

试图初始化位或 sfr

*ERROR 245：unknown pragma, line ignored

未知的 pragma 语句，因此该行被忽略

*ERROR 246：floating point error

本错误发生在浮点变量超过 32 位有效字长时，32 位 IEEE 格式的浮点值的取值范围是±1.75494E-38~±3.402832E+38

*ERROR 247：non_address +/-constant initializer

有效的初始化表达式必须是非地址量+/-常量

*ERROR 248：aggregate initialization needs curly braces

所有的组合变量（数组/结构或联合）初始化时要用花括号括起来

*ERROR 249：segment<name>:segment too large

编译器检测到过大的数据段，最大数据段长取决于存储器空间

*ERROR 250：‘\esc’;value exceeds 255

串常数中\esc 转义序列的值超过有效域

*ERROR 251：illegal octal digit

不是有效的八进制数字

*ERROR 252：misplace primary control、line ignored

一次性使用的编译控制伪指令必须在 C 模块开头指定，在#INCLUDE 语句和变量说明之前

*ERROR 253：internal ERROR(ASMGEN\CLASS)

这种错误在以下情况下发生⑴内部函数（如 testbit）被不正确激活。它发生在函数原型和实参表不存在匹配问题的时候。基于这个原因，头文件中的使用要适当（intrins.h,string.h）。⑵C51 识别出存在内部一致性错误，请向您的销售代理商查询

*ERROR 255：switch expression has illegal type

Switch 语句中的 case 语句必须具有类型(u)char,(u)int 或（u）short,其他类型不允许（如 bit）

*ERROR 256：conflicting memory model

Alien 属性的函数只能使用 SMALL 模式。函数的参数必须位于内部数据存储空间

*ERROR 257：alien function can not be reentrant

“alien”属性的函数不能同时具有“reentrant”属性，函数的参数不能通过重入栈传递，这也适用于外部“alien”声明和“alien”函数

*ERROR 258：mspace illegal on struct/union member

不能为结构联合成员指定存储空间，但指向对象的指针可以

*ERROR 259：pointer: different mspace

当为指针赋值或做指针比较时，指针未指向存储在同一存储空间的对象时，会产生错误或者警告。如：

```
Char xdata *px;/*px to char in xdata memory*/
Char code *pc;/*pc to char in code memory*/
Void main()
{ char c;
If(px==pc)++c;/*warning 259*/
}
```

*ERROR 260：pointer truncation

指针转换时部分偏移量被截断，此时指针常量（如 char xdata）转为一个具有较小偏移区的指针（如 char idata）

*ERROR 261：bit in reentrant function

重入函数不能包含位变量，因为位变量不能存于重入栈，而只能位于 MCS51CPU 的可位寻址存储区中如：

```
Void test () reentrant
{ bit b0;/*illegal*/
Static bit b1;/*legal*/
}
```

*ERROR 262：’using/disable’: function returns bit

使用属性 using 选择寄存器组的函数或使用关中断（#pragma disable）功能的函数不能返回

bit 类型。如：

```
Bit test ()using 3/*error 261*/
{ bit b0;
return(b0);
}
```

*ERROR 263：save-stack overflow/underflow

“#pragama save”最大嵌套级为 8 级。Save 和 restore 指令按 FIFO 原则工作

*ERROR 264：intrinsic<intrinsic_name>:declaration/ activation error

内部参数定义不正确

*ERROR 265：<name>recursive call to non_reentrant function

发现非重入函数被递归调用。直接递归用生成代码可有效查出，间接递归调用由 L51 发现

L51 连接定位器使用错误提示

1 警告

警告并不终止 L51 的执行。这时产生的程序模块由程序员自己斟酌使用还是不使用。但是此时的列表文件和屏幕显示可能非常有用。

2 错误

错误并不终止 L51 的执行。这时产生的模块是不能使用的。但是此时的列表文件和屏幕显示可能非常有用。

3 致命错误

致命错误发生时立即终止 L51 的执行。

1 警告

*WARNING1:UNSOLVED EXTERNAL SYMBOLS

SYMBOLS: external_name

MODULE: filename (modulename)

指定模块的外部符号在 PUBLIC 符号表中找不到

*WARNING2:REFERENCE MADE TO UNSOVED EXTERNAL

SYMBOLS: external_name

MODULE: filename (modulename)

ADDRESS:code_address

访问了未能匹配的外部符号 code 地址

*WARNING4:DATA SPACE MEMORY OVERLAP

FROM:byte,bit,address

TO: byte,bit,address

数据空间指定范围出现覆盖

*WARNING5:CODE SPACE MEMORY OVERLAP

FROM:byte,bit,address

TO: byte,bit,address

程序空间指定范围出现覆盖

*WARNING6:XDATA SPACE MEMORY OVERLAP

FROM:byte,bit,address

TO: byte,bit,address

外部数据空间指定范围出现覆盖

*WARNING7:MODULE NAME NOT UNIQUE

MODULE:filename(modulename)

模块名重名。模块未处理

*WARNING8:MODULE NAME EXPLICITLY REQUESTED FROMANOTHER FILE

MODULE:filename(modulename)

其他文件指名要求本模块名

*WARNING9:EMPTYABSOLUTE SEGMENT

MODULE:filename(modulename)

本模块包含空的绝对段，因未定位，它可能在不通知的情况下随时被覆盖

*WARNING10:CANNOT DETERMINE ROOT SEGMENT

L51 对输入文件要求分辨是 C51 还是 PL/M 文件，然后进行流程分析，在无法确定的时候，发出本警告。它发生在主程序被汇编调用的时候，需要程序员用 OVERLAP 特殊控制选项进行干预

*WARNING11:CANNOT FIND SEGMENT OR FUNCTION NAME

NAME:overlap_control_name

在目标模块中找不到 OVERLAP 控制选项中规定的段或者函数间调用

*WARNING12:NO REFERENCE BETWEEN SEGMENTS

SEGMENT1:segment_name

SEGMENT2:segment_name

试图用 OVERLAP 控制选项删除本来不存在的段间访问或者函数间调用

*WARNING13:RECURSIVE CALL TO SEGMENT

SEGMENT:segment_name

CALLER:segment_name

CALLER 段递归调用 SEGMENT 段。PL/M51 和 C51 的非重入函数不允许递归调用

*WARNING14:IMCOMPIABLE MEMORYMODEL

MODULE:filename(modulename)

MODEL:memory_model

指定模块试图与以前不同的存储模式编译。

*WARNING15:MULTICALL TO SEGMENT

SEGMENT:segment_name

CALLER1:segment_name

CALLER2:segment_name

两个函数调用同一个函数（如主函数和中断函数），参数和局部变量将被覆盖

*WARNING15:UNCALLED SEGMENT,IGNORED FOR OVERLAP PROCESS

SEGMENT:segment_name

所给的段未被调用，已被排除在覆盖过程之外。

L51 错误

*ERROR101: SEGMENT COMBINATION ERROR

SEGMENT:segment_name

MODULE:filename(modulename)

由于连接错误所给段未能连入类型总段，并被忽略

*ERROR102:EXTERN ATTRIBUTE MISMATCH

SYMBOL:external_name

MODULE:filename(modulename)

所给外部符号名属性错，并被忽略

*ERROR103:EXTERN ATTRIBUTE DO NOTMATCH PUBLIC

SYMBOL:public_name

MODULE:filename(modulename)

所给外部符号名属性与公用符号名不匹配，并被忽略

*ERROR104:MUTIPULIC DEFINITION

SYMBOL:pulic_name

MODULE:filename(modulename)

所给公用符号重名

*ERROR105:PUBLIC REFERS TO IGNORED SEGMENT

SYMBOL:public_name

MODULE:filename(modulename)

所给外部符号名属性错，并被忽略

*ERROR106: SEGMENT OVERFLOW

SEGMENT:segment_name

所给段长超过 64，未处理

*ERROR107:ADDRESS SPACE OVERLAP

SPACE:space_name

SEGMENT:segment_name

由于存储空间不够，所给类型总段未能装入，已被忽略

*ERROR108:SEGMENT IN LOCATING CONTROL CANNOTALLOCATED

SEGMENT:segment_name

命令行定位控制中的段由于属性问题未能分配

*ERROR109:EMPTY RELOCATABLE SEGMENT

SEGMENT:segment_name

可在定位类型总段长度为零，未定位

*ERROR110:CANNOT FIND SEGMENT

SEGMENT:segment_name

命令行所给的段在输入模块中未找到，被忽略

*ERROR111:SPECIFIED BITADDRESS NOT ON BYTE MEMORY

SEGMENT:segment_name

位地址不在字界上，位段被忽略

*ERROR112:SEGMENT TYPE NOT LEGAL FOR COMMAND

SEGMENT:segment_name

命令行所给的段类型非法，被忽略

*ERROR114:SEGMENT DOES NOT FIT

SPACE:space_name

SEGMENT:segment_name

BASE:base_address

LENGTH: segment_length

由于所给段的长度或者基地址未定位，故被忽略

*ERROR115:INPAGE SEGMENT IS GREATER THAN 256 BYTES

SEGMENT:segment_name

所给 INPAGE 属性的段长于 256 字节未能连入类型总段，并被忽略

*ERROR116:INBLOCK SEGMENT IS GREATER THAN 2048 BYTES

SEGMENT:segment_name

所给 INBLOCK 属性的段长于 2048 字节未能连入类型总段，被忽略

*ERROR117:BITADDRESSABLLE SEGMENT IS GREATER THAN 16 BYTE

SEGMENT:segment_name

所给 BITADDRESSABLE 属性的段长于 16 字节未能连入类型总段，被忽略

*ERROR118:REFERENCE MADE TO ERRONEOUS EXTERNAL

SYMBOL:symbol_name

MODULE:file_name(modulename)

ADDRESS:code_address

企图访问错误的外部程序地址

*ERROR119:REFERENCE MADE TO ERRONEOUS SEGMENT

SYMBOL:symbol_name

MODULE:file_name(modulename)

ADDRESS:code_address

企图访问错误段的程序地址

*ERROR120:CONTENT BELONGS TO ERROREOUS SEGMENT

SEGMENT:segment_name

MODULE:file_name(modulename)

该内容属于有错误的段

*ERROR121:IMPROPER FIXUP

MODULE:file_name(modulename)

SEGMENTsegment_name

OFFSET:segment_name

根据所给段和偏移地址的到的是不当的地址

*ERROR122:CANNOT FIND MODULE

MODULE:file_name(modulename)

命令行所给的模块未能找到

L51 致命错误

*FATAL ERROR201:INVALID COMMAND LINE SYNTAX

Partial command line

命令行句法错。命令行显示到出错处。

*FATAL ERROR202:INVALID COMMAND LINE ， TOKEN TOO LONG

Partial command line

非法命令行，单词太长。命令行显示到出错处

*FATAL ERROR203:EXPECTED ITEM MISSING

Partial command line

缺项。命令行显示到出错处。

*FATAL ERROR204:INVALID KEYWORD

Partial command line

非法关键字

*FATAL ERROR205:CONSTANT TOO LONG

Partial command line

常量大于 0xffff。命令行显示到出错处。

*FATAL ERROR206:INVALID CONSTANT

Partial command line

命令行常量无效（如 16 进制数以字母开头）。命令行显示到出错处。

*FATAL ERROR207:INVALID NAME

Partial command line

模块名或段名无效。命令行显示到出错处。

*FATAL ERROR208:INVALID FILENAME

Partial command line

文件名无效。命令行显示到出错处。

*FATAL ERROR209:FILE USED IN CONLICTING CONTEXTS

FILE:filename

所给的文件名用于有矛盾之处。命令行显示到出错处。

*FATAL ERROR210:I/O ERROR ON INPUT FILE

System error message

FILE:filename

访问输入文件时检测到有错，并有后面的 EXCEPTION 给出具体的错误描述

*FATAL ERROR211:I/O ERROR ON OUTPUT FILE

System error message

FILE:filename

访问输出文件时检测到有错，并有后面的 EXCEPTION 给出具体的错误描述

*FATAL ERROR212:I/O ERROR ON LISTING FILE

System error message

FILE:filename

访问列表文件时检测到有错，并有后面的 EXCEPTION 给出具体的错误描述

*FATAL ERROR213:I/O ERROR ONWORK FILE

System error message

FILE:filename

访问工作文件时检测到有错，并有后面的 EXCEPTION 给出具体的错误描述

*FATAL ERROR214:I/O INPUT PHASE ERROR

MODULE:filename(modulename)

L51 在进行第二次扫描时遇到不同的数据发生该错误，可能是因汇编错误引起

*FATAL ERROR215:CHECK SUM ERROR

MODULE:filename(modulename)

校验和与文件内容不一致

*FATAL ERROR216:INSUFFICIENTMEMORY

MODULE:filename(modulename)

执行 L51 的内存空间不够

*FATAL ERROR217:NO MODULE TO BE PROCESSED

缺少应该被处理的模块

*FATAL ERROR218:NOTAN OBJECT FILE

FILE:filename

所给文件非目标文件

*FATAL ERROR219:NOTAN 8051 OBJECT FILE

FILE:filename

所给文件非 8051 目标文件

*FATAL ERROR220:INVALID INPUTMODULE

FILE:filename

所给输入模块无效，可能是由汇编错误引起的

*FATAL ERROR221:MODULE SPECIFIED MORE THAN ONCE

Partial command line

命令行上多次包含同一模块。命令行显示到出错处

*FATAL ERROR222:SEGMENT SPEXIFIED MORE THAN ONCE

Partial command line

命令行上多次包含同一段。命令行显示到出错处

*FATAL ERROR224:DUPLICATE KEYWORD OR CONFLICATING CONTROL

Partial command line

命令行上多次包含同一关键字或者存在相互矛盾的控制选项。命令行显示到出错处

*FATAL ERROR225:SEGMENTADDRESS ARE NOT IN ASCENDING ORDER

Partial command line

定位控制的段地址未按照升序显示。命令行显示到出错处

*FATAL ERROR226:SEGMENTADDRESS INVALID FOR CONTROL

Partial command line

定位控制的段的段地址无效。命令行显示到出错处

*FATAL ERROR227:PARAMETER OUT RANGE

Partial command line

所给 PAGEWIDTH 和 PAGELENGTH 参数越界。命令行显示到出错处

*FATAL ERROR228:PARAMETER OUT RANGE

Partial command line

命令行上 RAMSIZE 参数越界。命令行显示到出错处

*FATAL ERROR229:INTERAL PROCESS ERROR

Partial command line

L51 检测到内部处理错。请询问代理商

*FATAL ERROR230:STRARTADDRESS SPECIFIED MORE THAN ONCE

Partial command line

命令行上包含多个未命名组段的起始地址。命令行显示到出错处

*FATAL ERROR233:ILLEGAL USE OF *IN OVERLAY CONTROL

Partial command line

命令行 OVERLAY 定位选择非法使用了*号（如*！*或*~*）。命令行显示到出错处

E.5 异常信息

L51 某些错误的原因由系统的 EXCEPTION 给出。

*EXCEPTION 0021:PATH OR FILE NOT FOUND

路径名或文件名未找到。

*EXCEPTION 0026H:ILLEGAL FILE ACCESS

试图写或者删除写保护文件。

*EXCEPTION 0029H:ACCESS FILE DENIED

所给的文件实际是目录。

*EXCEPTION 002AH:I/O-ERROR

欲写的驱动器已满或未准备好。

*EXCEPTION 0101H:ILLEGAL CONTEXT

命令行的语意非法。如对打印机进行读操作。

附录 F C51 的极限值

*标示符最长 255 个字符，一般取 32 字符。大小写不敏感。

*case 语句的变量个数没有限制，仅受可用内存容量和函数的最大长度限制。

*函数嵌套调用最大深度为 10。

*功能块{….}最大嵌套深度为 15。

*宏最多嵌套为 8。

*函数以及宏的参数最多为 32 个。

*语句行和宏定义最多 510 个字符（宏扩展后是 510 个字符）

*头文件嵌套深度为 20。

*预处理器中的条件编译层最多为 20。

*关于 INTEL 目标模块格式（OMF-51）的极限值。

*函数类型段总和最多 256 个。

*全局符号（PUBLIC）最多 256 个。

*外部符号（EXTERNAL）最多 256 个。