分析 C51 单片机的一些误区和注意事项

1) C 忌讳绝对定位。常看见初学者要求使用_at_,这是一种谬误,把 C 当作 ASM 看待了。在 C 中变量的定位是编译器的事情,初学者只要定义变量和变量的作 用域,编译器就把一个固定地址给这个变量。怎么取得这个变量的地址?要用指针。比如 unsigned char data x;后,x 的地址就是&x, 你只要查看这个参数,就可以在程序中知道具体的地址了。所以俺一看见要使用绝对定位的人,第一印象就是:这大概是个初学者。

2) 设置 SP 的问题。原因和 1 差不对,编译器在把所有变量和缓冲区赋予地址后,自动把最后一个字节开始的地方,作为 SP 的开始位置,所以初学者是不必要去理会的。这体现 C 的优越性,很多事情 C 编译时候做了。

3) 用 C 的主程序结构: #include void main(void) { while(1); } 这是个最小的成功的 C 程序,包括头部文件和程序主体。 头部文件的名词解释:引用的外部资源文件,这个文件包括了硬件信息和外部模块提供的可使用的函数和变量的说明。可以用文本方 式打开 reg52.h,仔细研究下,会有一些写程序的体会。

4) 这样构成一个 C 项目 在 C 中,常用项目来管理。项目一般分为两大块:C 文件块和头部文件块。我们常把不同功能写在不同的 C 文件中,依据项目的管理,最后把所有文件连接起来,这样就可以得到可以烧录的 HEX 文件或 BIN 文件。这些 C 文件中,有且只有唯一一个包括 main()函数,和 3)中一样的 C 文件。用头部文件把各个不同的 C 互相连接起来。

一个 C 文件基本上要对应有一个 H 头部文件,这个 H 文件就包含本 C 文件中可以提供给外面使 用的变量和函数,没有在 H 文件中列出的募??梢运闶歉肃文件的内部函数和变量,外部 C 不能使用。 例子:a.C: unsigned char i; unsigned char mWork; void Test1(void) { mWork++; } void Test2(void) { i++; } a.h 文件中: extern unsigned char i; extern void Test1(void); 这样主程序 M.c 中: #include /*C 编译器内部自带的 H 文件,使用<>*/ #i nclude 'a.h' /*自定义的 H 文件,一般用''*/ void main(void) { Test1(); /*使用 a.c 模块文件中的函数*/ while(1){ i++; /*使用 a.c 模块文件中的变量*/ } }

5) 51 家族 核心都是基于 8031 的,有很多在此核心上进行扩展,有的把程序存储器放在内部:89c(S)51..,有的增加了 RAM:89c(S)52..,有的增加 了一些专用硬件 80C552...,有的改变时钟时序 W77E58...。市面上现在常用的主要有 ATMEL 公司的 AT89X 系列,PHILIPS 的 P87(89)x,台 湾 WINBOND 的 w77(78)x 系列,Cygnal 的 C8051Fx 系列。

6) 51 单片机结构的 C 描述 这里不讲 51 的具体结构,只是引导初学者快速理解 51 单片机的物理结构。寄存器和 IO 及其它硬件设备的地址名称,在相应的 C 头部文件 中可以找到。51 为 reg51.h,52 为 reg52.h,以次类推,比如 winbond 的 78E58 就为 w78e58.h 这些 H 文件中的描述:srf,定义一个 8 位的设备。srf16,定义一个 16 位的设备。 sbit,定义一个位的设备。 用这些语句定义后,就可以

在C中象汇编一样使用这些硬件设备,这是单片机应用比标准C特殊的地方,其它差别很少。

7） 在 51 系列中 data,idata,xdata,pdata 的区别 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。

8） 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。