



MS Micro Controller

MS Designed By Origin-gd Tech

---

**MS8xF 应用文档**

**用户应答手册**

**8 位 FLASH 全系列带 EEPROM 单片机**

# MS8xF用户应答手册

## 目录

<b>第一章 仿真器常见问题</b> .....	<b>1</b>
1.如何确定仿真器的版本，升级软件和固件？ .....	1
2.可以用仿真器将 HEX 下载到芯片吗？ .....	1
3.仿真时，PA0 和 PA1 是否可以继续作为普通 IO 口使用？ .....	1
4.怎么使用仿真器做调试？ .....	1
5.在仿真时，出现下载失败怎么处理？ .....	2
6.如何通过仿真器进行外部供电仿真？ .....	2
<b>第二章 烧录器及烧录器软件常见问题</b> .....	<b>3</b>
1.烧录器怎么供电，供电电压是多少？ .....	3
2.怎么看烧录器是否连接到电脑？ .....	3
3.烧录器软件的版本和烧录器的固件版本从哪里查看，怎么更新？ .....	3
4.烧录器软件如何导入 HEX 和 EEPROM.BIN？ .....	3
5.编译完成的 HEX 导入到烧录器前，是否需要设置配置位？ .....	3
6.怎么设置烧录器烧录电压？ .....	4
7.烧录器软件的 Auto Program 按钮和其他 Erase、Blank Check 等按钮的功能？ .....	4
8.烧录器软件没有 MS84Fxx04 系列型号？ .....	4
9.烧录成功或失败怎么提示？ .....	4
10.HEX 已经设置了加密，有信息可以用烧录器读到？ .....	4
11.怎样通过烧录器来读回芯片的校验码？ .....	4
12.烧录器怎样连接自动烧录机台烧录？ .....	5
13.怎样通过烧录器实现只检查 Checksum 不烧录的功能？ .....	5
14.烧录器只连接 VCC、GND、CLK、DAT 四根线可以烧录吗？ .....	5
15.烧录器第一次烧录芯片成功，第二次烧录失败是什么原因？ .....	5
16.如何用烧录器在板烧录芯片？ .....	6
17.支持跳线帽跳线，怎么用烧录夹座的型号？ .....	6
18.如何用烧录器以滚码方式烧录？ .....	11
<b>第三章 ORIGINIDE 常见问题</b> .....	<b>12</b>
1.可以用 MPLAB IDE 或者 MPLAB X IDE 这 2 个平台来开发 MS 单片机吗？ .....	12
2.不习惯用 ORIGINIDE 这个平台编辑代码，可以用其他代码编辑软件编辑吗？ .....	12

3.使用 C 语言编程时, 在编译时提示 “This Language Toolsuite is not Exists” ?	12
4.提示 “File not exist” 或 “No such fileor diretory”,包含的头文件怎么处理?	12
5.自定义的头文件怎么添加到工程里?	13
6.使用新的 ORIGINIDE 编译旧工程时, 弹出的编译选项一片灰白, 怎么处理?	13
7.仿真器仿真时, 芯片的 PA0 和 PA1 是否可以继续做普通 IO 口使用?	13
8.使用 ORIGINIDE 仿真时, 看门狗是否有作用?	13
9.同一个项目在别的电脑上可以编译通过, 在部分电脑上不能编译通过?	13
10.如何在 ORIGINIDE 里导出一个 EEPROM 文件?	14
11.只有 HEX 文件可以通过 ORIGINIDE 烧录吗?	14
<b>第四章 汇编常见问题</b>	<b>15</b>
1.头文件怎么调用?	15
2.在使用汇编编程时, 中断如何现场保护?	15
3.芯片初始化后,操作普通 IO 寄存器 PORTA/PORTC 为什么不起作用?	16
4.芯片 IO 做输入开内部上拉时, 为什么 IO 脚不是高电平?	16
5.芯片睡眠时怎样才能做到最低待机电流?	16
6.伪指令 BANKSEL 的作用是什么?	16
7.汇编指令时间的问题?	16
<b>第五章 C 语言常见问题</b>	<b>17</b>
1.关于头文件包含	17
2.关于 EEPROM 初始化	17
3.关于配置位配置	17
4.关于中断服务函数	17
5.关于中断现场保护	17
6.关于中断优先级	17
7.关于中断标志位	17
8.关于操作特殊寄存器和 RAM, BANK 的切换	18
9.关于_XTAL_FREQ 的定义	18
10.关于整型变量类型	18
11.关于常量类型与格式	19
12.关于位数据类型和变量	21

# MS8xF用户应答手册

---

13.关于标准类型限定符 .....	21
14.关于 C 编译器的优化 .....	22
<b>第六章 MS8xF 使用常见问题.....</b>	<b>24</b>
1.同一系列的型号，芯片内部电路是否一样？ .....	24
2.没有封装出来的 IO 口如何初始化？ .....	24
3.在低功耗应用应注意几个点？ .....	24
4.MS81F 的 IO 设置了 TRIS（比如 TRISA）寄存器后还是不能正常输入或输出？ .....	24
5.MS83F 的 IO 设置了 TRIS 和 CMCON0 寄存器后还是不能正常输入或输出？ .....	25
6.MS84F 的 IO 设置了 TRIS、CMCON0 和 ANSEL 后还是不能正常输入或输出？ .....	25
7.MS 单片机的时钟周期、指令周期分别指什么？ .....	25
8.用 IO 电平变化中断(IOCA)，如何清除中断？ .....	25
9.MS83F 和 MS84F 的内部固定参考电压注意事项 .....	25
10.使用 ADC 时，转换到的 AD 值不准 .....	26
11.汇编程序里超过 2K 空间的 LCALL 和 LJUMP 为什么会出错.....	26
<b>联系我们.....</b>	<b>27</b>

Origin-gd.com Tech

## 第一章 仿真器常见问题

### 1.如何确定仿真器的版本，升级软件和固件？

仿真工具分为三部分：

- 仿真器上位机软件（ORIGINIDE）
- 仿真器（Link）
- 仿真器的固件 FW

- 1、ORIGINIDE 的版本信息可以在 ORIGINIDE 界面的“菜单栏-Help-About...”里查看，如 1.9.5。通过 QQ 与 FAE 索取或在官网下载最新的 IDE 软件可以升级；
- 2、Link 的版本较少更新，版本丝印打在 PCB 面板上，如 Link\_V2.5；
- 3、仿真器的固件 FW 版本信息可以在 Link 插上电脑后，在 ORIGINIDE 右下角显示，如 FW Ver:VF0.00.00.12，一般随 IDE 的版本进行升级，如需手动升级或降级，可点击 IDE“菜单栏-Help-Update Firmware...”进行更新固件，更新文件保存在 ORIGINIDE“安装目录\Update\AppUpdate.bin”；

### 2.可以用仿真器将 HEX 下载到芯片吗？

不可以。如果要下载带芯片，只能通过点击 **build all**，待编译通过后把程序下载到芯片。也就是需要有可以有效编译成功的源代码，才可以把对应源代码的 HEX 程序下载到芯片。同时，通过 **build all** 下载后的程序的配置位是与用户设置的配置位一致。比如 CPD 已经选择了 **Enable**，下载完的芯片已经有代码保护。

### 3.仿真时，PA0 和 PA1 是否可以继续作为普通 IO 口使用？

不可以。仿真时 LINK 会占用 PA0 和 PA1 与芯片内部通信，所以在仿真时不能做普通 IO 口使用。下载完成后，与 LINK 断开连接，重新上电后，PA0 和 PA1 可以实现普通 IO 口功能。

### 4.怎么使用仿真器做调试？

1、将仿真器LINK用USB连接到电脑端后，在右上角的3个电压选项用跳帽选择好LINK给芯片的供电方式——仿真器给芯片供电和外部电源给芯片供电。

a.仿真器给芯片供电有3档电压选择：3.3V、3.7V、5V

b.外部电源给芯片供电要须将跳帽跳至External处，将外部供电VCC端接到仿真器P1端口的VCC，将外部供电GND端接到仿真器P1端口的GND，且外部供电电压不能超过5.5V；

**注意：在使用仿真器供电电压时，禁止外部供电给目标板。两个电源同时加载有很大风险会损坏Link**

2、连接好LINK右边的四根插针VCC、GND、CLK、DATA到芯片的VCC、GND、PA0、PA1；

3、.打开工程后点击Build All按钮会弹出编译选项，选择好编译选项后，ORIGINIDE即

# MS8xF用户应答手册

可通过LINK将程序下载到芯片里。下载完成后会自动进入DEBUG界面，使用DEBUG工具栏的图标即可进行仿真调试。

## 注意：通过仿真器下载好程序的芯片

需要注意的是因为仿真器在下载程序时需要MCU进行重新上电的过程，所以下载时目标板不能外接电源，且板子上最好不要有大电容影响芯片的重新上电，PA0和PA1最好不要有其他元件影响调试脚位的电平状态；

4、仿真调试时下载程序到芯片，不要同时开多个工程，以免出现下载错误；

## 5.在仿真时，出现下载失败怎么处理？

出现下载失败一般分两种情况分析，一是用Link单独连芯片下载失败，二是用在板调试时下载失败。

针对第一种情况，分以下几步分析：

- 1、检查是否同时开启多个工程；
- 2、检查ORIGINIDE和Link的固件版本是否最新；
- 3、检查Link和芯片的连线是否正确与稳定。注意芯片的PA0是时钟脚，接到Link的CLK，芯片的PA1是数据脚，接到Link的DATA。这个正好的PIC的相反(PIC的RA0是数据脚ICSPDAT，RA1是时钟脚ICSPCLK)；
- 4、检查编译选项里LVR的电压是否高过Link的输出电压。如果已经烧录成功的芯片的LVR设置的比Link的输出电压高，在还没进入烧录模式的时候，芯片就已经因为LVR低压复位了，所以会烧录失败。必须将芯片单独擦除一遍(菜单栏” Debug” ->” Erase”)，再烧录；
- 5、检查编译选项里WDTEN是否选择为Enable，应为Disable。如果已经烧录成功的芯片的WDTCON寄存器超时时间设置得较低，会导致再一次烧录的时候还没烧录成功就WDT超时复位了。同样需要将芯片单独擦除一遍(菜单栏” Debug” ->” Erase”)，再烧录；
- 6、检查芯片型号是否混料，更换芯片测试。
- 7、检查Link是否正常，更换Link测试；

针对第二种情况，分以下几步分析：

- 1、用Link单独连芯片，按第一种情况排查。如可以下载，则进行下一步判断；
- 2、检查在板芯片的VCC-GND之间是否有大电容，如有，请去掉；
- 3、检查芯片的CLK,DAT脚上是否有容性器件或其他元件影响下载通讯电平；
- 4、检查芯片的复位脚功能是否开启。将编译选项里 MCLRE 设置为 PA3/PA5；

## 6.如何通过仿真器进行外部供电仿真？

- 1、将仿真器的电压选择跳帽从3.3V/3.7V/5.0V档位换到External；
- 2、将外部供电VCC端接到仿真器P1端口的VCC；
- 3、将外部供电GND端接到仿真器P1端口的GND；
- 4、外部供电电压不能超过5.5V；
- 5、外部供电的电源线线阻尽量小；

注意：在使用仿真器供电电压时，禁止外部供电给目标板。两个电源同时加载有很大风险会损坏 Link

## 第二章 烧录器及烧录软件常见问题

### 1. 烧录器怎么供电，供电电压是多少？

烧录器供电电源有 2 个方式：

- a.通过 USB 口供电，供电电压为 5V B 型 USB；
- b.通过 DC 做供电，供电电压为 7.5V 或 9V；

**注意：电源输出电流须高于 500mA。如果上电后 LCD 显示不正常，请更换一个功率大一些的电**源。

### 2. 怎么看烧录器是否连接到电脑？

先把烧录器通过 USB 线连接到电脑，待驱动装完（驱动自带在烧录器上），打开烧录软件。如果烧录软件的右上处粤原点 LOGO 下面有出现烧录器的版本号，如：DRV Ver: 02.00.00.09，表示已经成功连接到电脑。

### 3. 烧录器软件的版本和烧录器的固件版本从哪里查看，怎么更新？

烧录器软件版本号在软件的左上角显示，如OriginWriter V3.4。

烧录器固件版本号在烧录器软件右上方图标下的DRV Ver后显示，如V2.1.4。

通过点击DRV Ver:上面的小图标可以进行固件升级或降级。点击图标后会弹出确认框，确认后等待2分钟左右，烧录器会自动完成更新。

固件更新完成后需要重新拔插 USB 重新连接，重新连接成功后 DRV Ver 后面会显示当前固件版本号。

### 4. 烧录器软件如何导入 HEX 和 EEPROM.BIN？

- a.选择芯片型号后，烧录器软件会自动弹出烧录档目录选项框；
- b.如果需要烧录HEX文件，则选择HEX文件。如果不需要烧录HEX文件，则选择取消。选择完成后烧录器自动弹出目录选项框，选择EEPROM文件；
- c.如果需要烧录EEPROM，则选择BIN文件。如果不需要烧录BIN文件，则选择取消；单独选择HEX文件，烧录器只会烧录对应的程序区，不会影响芯片原有的EEPROM数据；
- d.单独选择EEPROM文件，烧录器只会烧录对应的EEPROM区，不会影响芯片原有的程序数据；
- e.如果都不选择 HEX 和 EEPROM,那么只能通过烧录软件去读取烧录器上芯片的数据。

### 5. 编译完成的 HEX 导入到烧录器前，是否需要设置配置位？

不需要。在 ORIGINIDE 编译好的 HEX，已经包含了编译前配置好的配置位。所以，导入 HEX 到烧录器时，不用设置配置位。

## 6. 怎么设置烧录器烧录电压？

烧录器右上角有个跳帽，可以调整为 3V 烧录或者 5V 烧录。默认是跳在 3V 处。

## 7. 烧录器软件的 **Auto Program** 按钮和其他 **Erase**、**Blank Check** 等按钮的功能？

**Auto Program:** 烧录器便会依次执行“擦除”、“程序烧录”、“烧录校验”动作

**Erase:** 仅对芯片进行擦除操作（擦除可选择Flash和EEPROM）

**Blank Check:** 仅对芯片进行空白检查

**Program:** 仅对芯片执行烧录动作,不校验是否烧录成功

**Verify:** 检查烧录到芯片中的代码是否正确, (如果烧录程序使能了CPB, 则无法校验)

**Read:** 读出芯片信息（如果芯片有加密则只能读出EEPROM资料）

**Setting:** 设置烧录数量上限

**Rolling Code:** 滚码设置

**Calibration Frequency:** 校准频率（勾选后，检测到频率偏移芯片就会自动校准，但校准精度较差）

**Test Frequency:** 检测频率（勾选后，检测到频率偏移的芯片就会报错）

## 8. 烧录器软件没有 **MS84Fxx04** 系列型号？

点击Device/Load按钮后，会弹出选择芯片型号的对话框，如果没有当前芯片的系列号，需要检查：

- 1、烧录器是否正常联机。（DRV Ver是否显示烧录器固件号？）
- 2、烧录器是否为黄色版本烧录器。
- 3、如果为黄色版本，请确认烧录器固件号是否为最新的版本。

## 9. 烧录成功或失败怎么提示？

当按下烧录红色烧录按钮时，绿灯和红灯一起灭。之后如果烧录成功，绿灯会亮；如果烧录失败，红灯会亮和蜂鸣器会警报。

## 10. HEX 已经设置了加密，有信息可以用烧录器读到？

Flash 区程序的校验和 Checksum、EEPROM 区的数据。

## 11. 怎样通过烧录器来读回芯片的校验码？

将芯片按照烧录脚位放置在烧录器上或通过导线有效连接到烧录器上，按一下烧录器上的 KEY\_MODE 按键，烧录器会“滴”一声，并在显示屏上显示“IC's checksum: “XXXX”，如果未出现此结果，需要检查是否连接正确或者是否程序被配置成外部晶振模式。



## 12. 烧录器怎样连接自动烧录机台烧录？

V2.0 烧录器的 20 脚牛角座背面丝印的 PROGRAM、BUSY、OK、FAIL、GND 和 GND 下方未标注的引脚（+3V）分别连接到烧录机台的串口线的 START、BUSY、OK、NG、GND、VCC。在烧录器的设置选项里将启动电平、BUSY 电平、OK 电平、NG 电平均设置为 L，其他时间均使用默认设置即可。

## 13. 怎样通过烧录器实现只检查 Checksum 不烧录的功能？

将需要校验的程序下载到烧录器里，长按KKEY\_MODE按键，直到烧录器显示屏上显示烧录器的工作模式“Write Flash”，此时连续按下红色按钮切换烧录器模式到“Check checksum num”即可。

**Write**功能下有以下模式（下载程序时烧录器会自动选择模式，不建议手动切换）：

- a. Write Flash: 烧录芯片程序HEX
- b. Write Flash EEPROM: 烧录芯片程序HEX和EEPROM.BIN
- c. Write Fla EEP Rcode: 烧录芯片程序HEX和EEPROM.BIN烧录滚动码
- d. Write Fla EEP Rd FOSC: 烧录芯片程序HEX和EEPROM.BIN烧录滚动码并校准频率
- e. Write Flash Rcode: 烧录芯片程序HEX，烧录滚动码
- f. Write Fla Rd FOSC: 烧录芯片程序HEX，烧录滚动码并校准频率
- g. Write Fla FOSC: 烧录芯片程序HEX并校准频率
- h. Write Fla EEP FOSC: 烧录芯片程序HEX和EEPROM.BIN并校准频率
- i. Write EEPROM: 烧录EEPROM.BIN
- j. Write EEPROM R\_code: 烧录EEPROM.BIN和滚动码

**Check**功能下有三个模式：

- a. Check FOSC: 检查芯片频率（检查芯片频率会擦除芯片内部数据，检查完后需要重新烧录）
- b. Check Flash: 检查程序（只适用于代码不加密的情况CPB=Disable）
- c. Check checksum num: 检查芯片的 Checksum（可连机台自动检测）

## 14. 烧录器只连接 VCC、GND、CLK、DAT 四根线可以烧录吗？

可以连接。在烧录器软件往烧录器里导入好程序后，将烧录软件左下角的 Calibration Frequency 和 Test Frequency 勾选项去掉，即可通过四线烧录。

## 15. 烧录器第一次烧录芯片成功，第二次烧录失败是什么原因？

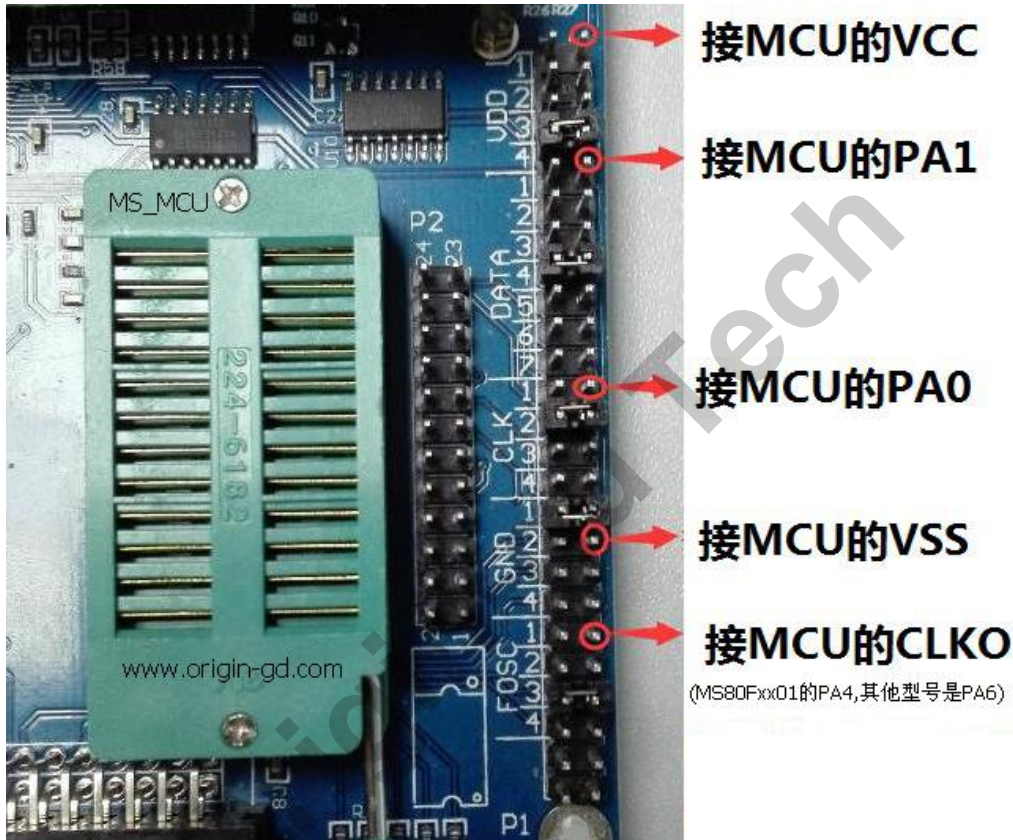
- 1、检查烧录器显示屏左边的烧录电压选择跳冒，如果程序LVR电压选择超过3V，且烧录器烧录电压为3V的情况下，会出现第一次烧录成功，第二次烧录失败的现象。将电压选择为5V可解决。因为第一次烧录的3V LVR在第二次烧录程序的时候已经生效，而烧录电压如果还是3V，这时候芯片就直接复位了，进入不了烧录模式。因此需要把烧录电压设置跳至5V烧录。
- 2、检查第一次烧录的程序的看门狗是否被开启。如果被开启，检查WDTCN寄存器的WDTPS[3:0]配置的分频比是否过低，导致烧录的时候还没烧录成功芯片就看门狗超时复位了。如果是这样，需要单独把芯片擦除一遍，再烧录。

# MS8xF用户应答手册

3、检查程序的编译选项里的FOSC是否为外部晶振模式，如果采取了外部晶振模式，且FCMEN时钟故障监视被禁止，导致第二次烧录时芯片无法启动进入烧录模式。

## 16. 如何用烧录器在板烧录芯片？

将烧录器 P1 的跳针按下图的方式接 MCU 对应的脚位即可。在线烧录线路规范：CLK 和 DAT 脚上是否有容性器件或其他元件影响下载通讯电平；VCC 和 GND 之间不要有大电容，以免影响重新上电。



## 17. 支持跳线帽跳线，怎么用烧录夹座的型号？

分别是 MS80F0801、MS81F0802、MS81F1402、MS81F1602、MS81F1802、MS81F2002、MS82F0802A、MS82F0802B、MS82F1402A、MS83F0802A、MS83F0802B、MS83F1402A 和 MS83F1602，其他型号只能通过烧录器的 5 个烧录脚连接到芯片来烧录。

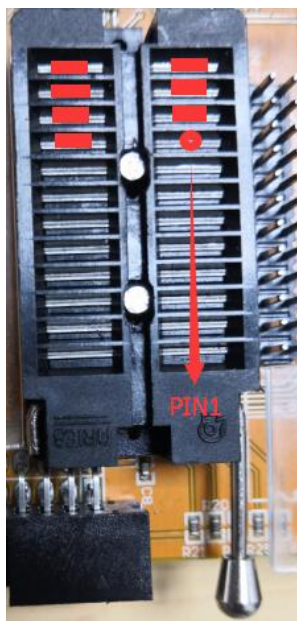
跳线帽跳线表：

Part NO.	VCC	DATA	CLK	GND	FOSC(CLKO)
MS80F0801	3	3	2	2	4
MS81F0802	3	2	1	2	4
MS81F1402	3	6	3	2	1
MS81F1602	2	7	4	3	2
MS81F1802	4	4	2	1	3

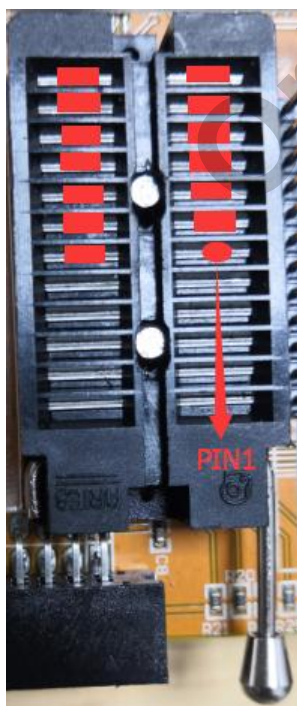
MA81F2002	4	4	2	1	3
MS82F0802A	1	5	3	4	2
MS82F0802B	1	5	3	4	2
MS82F1402A	1	5	3	4	2
MS83F0802A	1	5	3	4	2
MS83F0802B	1	5	3	4	2
MS83F1402A	1	5	3	4	2
MS83F1602	1	5	3	4	2

芯片放置的方向如下：

## 1、MS80F0801 和 MS81F0802:



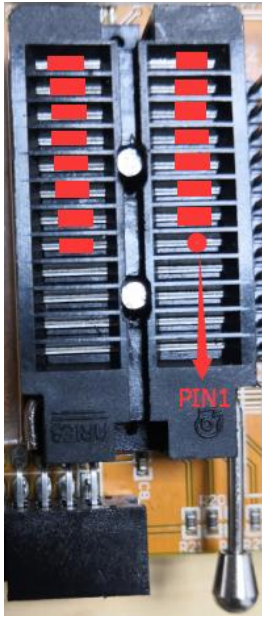
## 2、MS81F1402:



# MS8xF用户应答手册

---

## 3、MS81F1602 和 MS83F1602:

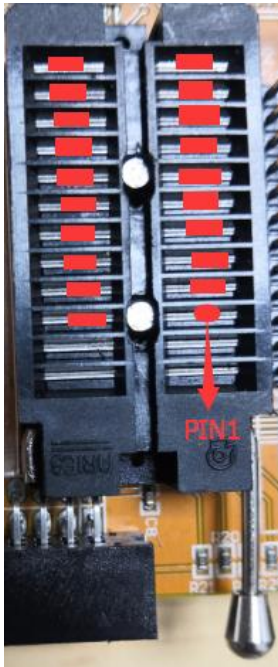


## 4、MS81F1802:



Origin-gd Tech

## 5、MS81F2002:



## 6、MS82F0802A、MS82F0802B、MS83F0802A 和 MS83F0802B:

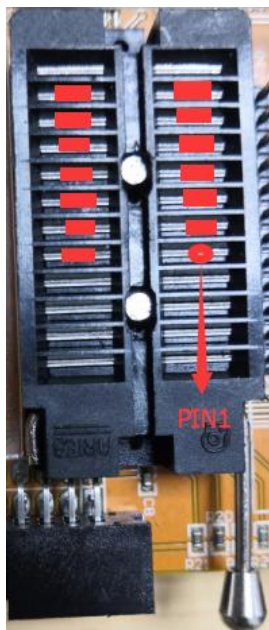


请注意：烧录这 4 个型号的芯片，只能用 4 线烧录。即跳好跳线帽和导入程序到烧录器后，在烧录软件左下角把 **Calibration Frequency** 和 **Test Frequency** 勾选项去掉。

# MS8xF用户应答手册

---

## 7、MS82F1402A 和 MS83F1402A:



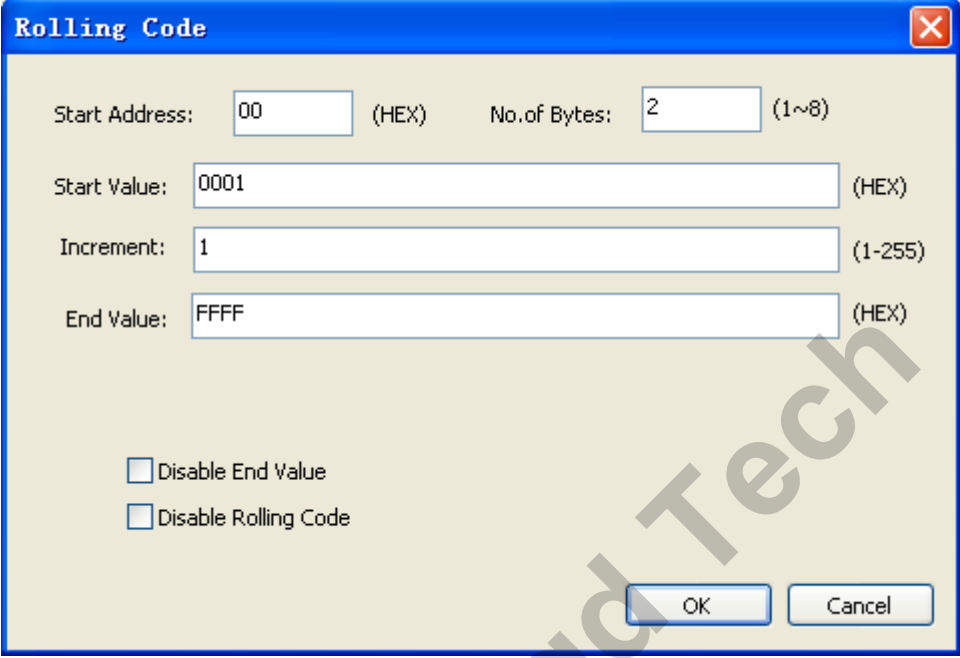
请注意：烧录这 2 个型号的芯片，只能用 4 线烧录。即跳好跳线帽和导入程序到烧录器后，在烧录软件左下角把 **Calibration Frequency** 和 **Test Frequency** 勾选项去掉。

Origin-gd Tech



## 18. 如何用烧录器以滚码方式烧录？

导入完程序后，点击烧录软件的“Rolling Code”，弹出烧录滚码配置信息表。请注意：滚码值只能在 EEPROM 区滚码。而不能像 PIC 一样，可以在 FLASH 区或者 EEPROM 滚码。所以使用 MS 单片机的滚码，必须在程序中用到 EEPROM 读指令，来读取存放在 EEPROM 区的滚码值。另外，滚码的方式只有自增模式，没有伪随机模式。



The image shows a software dialog box titled "Rolling Code" with a close button (X) in the top right corner. The dialog contains several input fields and checkboxes:

- Start Address:** Input field with "00" and "(HEX)" label.
- No. of Bytes:** Input field with "2" and "(1~8)" label.
- Start Value:** Input field with "0001" and "(HEX)" label.
- Increment:** Input field with "1" and "(1-255)" label.
- End Value:** Input field with "FFFF" and "(HEX)" label.
- Disable End Value
- Disable Rolling Code
- OK** and **Cancel** buttons at the bottom.

**Start Address:**滚码开始地址

**No. of Bytes:**滚码值字节数

**Start Value:**滚码开始值

**Increment:**滚码增加步距

**End Value:**滚码结束值

## 第三章 ORIGINIDE 常见问题

### 1. 可以用 MPLAB IDE 或者 MPLAB X IDE 这 2 个平台来开发 MS 单片机吗？

不可以，只能用 ORIGINIDE 这个平台来开发。因为 MPLAB IDE 和 MPLAB X IDE 没有 MS 相应的型号选择。

### 2. 不习惯用 ORIGINIDE 这个平台编辑代码，可以用其他代码编辑软件编辑吗？

可以，例如用 source insight。但须注意，要先建好工程再对工程的 c 代码编辑。因为用 ORIGINIDE 建工程，会自动生成一个与工程名一样的 c 文件，而这个 c 文件是无有效代码的。如果在建工程之前，已经有一个与工程名一样的 c 文件且已经编辑好代码的，这样建工程会把这个已经写好的文件替换掉。

### 3. 使用 C 语言编程时，在编译时提示 “This Language Toolsuite is not Exists” ？

一般是 C 编译器未安装或调用 C 编译器执行程序(pic.exe)的路径不对引起的。在 ORIGINIDE 目录下打开一个工程后，“菜单栏->Project->Select Language Toolsuite...”在弹出的框里点击 Browse...选择 C 编译器的安装目录下的编译器执行程序(pic.exe)即可。

注意：如果有用这个 C 编译器开发 PIC 的器件，请将把这个 C 编译器安装成功后的文件拷贝一份在另外一个盘符，例如 C 编译器安装一开始是安装在 C 盘，就把这个 C 编译器的文件复制一个在到 C 盘以外的另一个盘符中。然后使用 PIC 的器件开发就是用 C 盘的 C 编译器，用 MS 的器件开发就用另一个盘符的 C 编译器。

### 4. 编译时会提示“File not exist”或“No such file or directory”,包含的头文件怎么处理？

a.使用汇编编程时，头文件需要在最开始包含头文件：`#include <头文件.inc>`或`#include “头文件.inc”`。他们的差别在于：

- `#include <头文件.inc>`代表编译时直接在IDE软件路径中寻找里面是否有此文件。如果有，直接加载；如果没有，报错(File not exist);
- `#include “头文件.inc”`代表编译时先寻找你当前工程文件夹里面是否有此文件。如果有，优先加载这个文件。如果没有，就会在IDE软件路径中寻找里面是否有此文件。如果有，直接加载；如果没有，报错(File not exist);



b.使用 C 编程时，工程一定要包含#include "syscfg.h"。如果是 WIN7 和 WIN7 以上系统，需要关闭整个工程，然后让 ORIGINIDE 获得管理员权限。以 WIN7 为例：右键点击 ORIGINIDE 图标，然后管理员权限运行。

## 5. 自定义的头文件怎么添加到工程里？

目前只能通过文件里 include.inc 或.h 来添加自定义的头文件，编译通过后，就可以显示在 File View 的 Include File 目录下了。暂时不支持手动在 Include File 里直接添加。

## 6. 使用新的 ORIGINIDE 编译旧工程时，弹出的编译选项一片灰白，怎么处理？

出现这种情况，一般是工程的芯片型号有变动引起的，现在ORIGINIDE都按规格书上的器件名来处理,所以器件名不一样，引起编译选项空白。

解决方法：鼠标右键点击在 File View 显示区的 Device:xx，在弹出的菜单栏里选择“Project Options...”在弹出的选项框里选择正确的器件名 Device:xxxx，点击确定即可正常编译。

## 7. 仿真器仿真时，芯片的 PA0 和 PA1 是否可以继续做普通 IO 口使用？

不可以。仿真时 LINK 会占用 PA0 和 PA1 与芯片内部通信，所以在仿真时不能做普通 IO 口使用。下载完成后，与 LINK 断开连接，重新上电后，PA0 和 PA1 可以实现普通 IO 口功能。

## 8. 使用 ORIGINIDE 仿真时，看门狗是否有作用？

仿真时，如果启用看门狗，要注意清看门狗，否则看门狗溢出时会产生复位，使 IDE 无法获取正确的 PC 值，从而出现仿真错误。建议仿真时关闭看门狗，正常测试时再打开看门狗。

## 9. 同一个项目在别的电脑上可以编译通过，在部分电脑上不能编译通过？

1、头文件的问题。新版的IDE里使用C语言包含syscfg.h即可，汇编语言里包含芯片系列头文件即可，如MS80Fxx01.inc。

2、C 编译器的安装目录问题。因为对用户的权限管理，部分操作系统会禁止 IDE 去修改系统盘下 C 编译器里的文件。将 C 编译器装在非系统盘，或者使用管理员权限运行 ORIGINIDE 可解决此问题。

## 10. 如何在 ORIGINIDE 里导出一个 EEPROM 文件？

- 1、打开一个工程，IDE菜单栏Project->Export EEPROM Data 会生成一个.bin类型的文件，保存这个文件。
- 2、由于 IDE 不支持 EEPROM 文件的在线编辑，建议使用第三方文本编辑软件对 EEPROM 文件进行编辑，以 UltraEdit 为例，用 UltraEdit 打开文件,在编辑->十六进制函数->十六进制编辑，就可以对 256 字节的 EEPROM 数据进行编辑。

## 11. 只有 HEX 文件可以通过 ORIGINIDE 烧录吗？

目前只能通过 ORIGINIDE 的 Build All 来进行工程的编译和下载，单独 HEX 文件只能通过烧录器烧录芯片。

Origin-gd Tech

## 第四章 汇编常见问题

### 1. 头文件怎么调用？

- 1、 MS80F0601 和 MS80F0801 调用 MS80Fxx01.INC
- 2、 MS81F0802、MS81F1402、MS81F1602、MS81F1802 和 MS81F2002 调用 MS81Fxx02.INC
- 3、 MS82F0802A、MS82F0802B 和 MS82F1402 调用 MS82Fxx02.INC
- 4、 MS83F0802、MS83F0802、MS83F1402A 和 MS83F1602 调用 MS83Fxx02.INC
- 5、 MS84F1404 、 MS84F1604 、 MS84F2004A 和 MS84F2004B 调用 MS84Fxx04.INC

### 2. 在使用汇编编程时，中断如何现场保护？

芯片的中断向量地址为 04H，程序运行时发生中断事件时，会将程序当前 PC 值推入堆栈，然后从 04H 地址开始执行，直到遇到 RETI 取堆栈 PC 返回。中断保护现场和恢复现场程序需要保证中断事件发生前后 W 值，STATUS 值保持不变。如果在中断里改写了 PCLATH，那么也需要增加一个变量来保护 PCLATH。如果在中断里没有改写 PCLATH 值，可以不用保护。推荐的保护现场程序和恢复现场程序如下：

```
W_TMP EQU 0x70
S_TMP EQU 0x71
ORG 0004H
STR W_TMP
SWAPR STATUS,W
STR S_TMP
;BANKSEL PORTA
;中断处理程序
;中断处理程序
INT_RET:
SWAPR S_TMP,W
STR STATUS
SWAPR W_TMP,F
SWAPR W_TMP,W
RETI
```

W\_TMP和S\_TMP需要定义在0x70-0x7F这个所有BANK共用的地址区间，以防在其他BANK时进入中断会丢失中断信息。

中断保护现场后需要设置中断里的BANK。

如果中断处理程序里要处理BANK0里的寄存器，那么需要将BANK设置在BANK0。

如果中断处理程序里要处理BANK1里的寄存器，那么需要将BANK设置在BANK1。

如果不设置，程序在BANK1时进入中断时，操作到BANK0里的寄存器时就会产生错误，同理在BANK0时进入中断时，操作到BANK1里的寄存器也会产生错误。

用 C 语言编程不需要注意这些事宜，编译器会自动设置。

## 3. 芯片初始化后,操作普通 IO 寄存器 PORTA/PORTC 为什么不起作用?

- 1、检查ANSEL功能是否关闭, 设置ANSEL=0;
- 2、检查比较器功能是否关闭, 设置CMCON0=0x07;
- 3、设置TRISA/TRISC;
- 4、检查编译选项里MCLRE是否设置为MCLR;
- 5、检查编译选项里 Fosc 是否设置为 INTOSCIO;

## 4. 芯片 IO 做输入开内部上拉时, 为什么 IO 脚不是高电平?

设置 IO 为普通 IO 口还需要设置 WPUAX/WPUCX, 并将 OPTION.7 置 0 来开启内部上拉总开关。

## 5. 芯片睡眠时怎样才能做到最低待机电流?

- 1、检查IO的状态, 不要让IO处于输入悬空的状态;
- 2、检查ANSEL功能和比较器功能是否打开;
- 3、检查LVR功能是否在睡眠时打开, 如果是, 请关闭;
- 4、检查编译选项里Fosc是否设置为INTOSC, 如果是, 请选择INTOSCIO;
- 5、检测复位脚设置。如果没有用到或没有复位脚, 将编译选项MCLRE设置为MCLR;
- 6、检查看门狗是否打开, 根据需要打开或关闭;

## 6. 伪指令 BANKSEL 的作用是什么?

伪指令BANKSEL 是用于设置寄存器BANK的命令, 切换到BANKSEL后面的寄存器所在的BANK。

如MS80Fxx01和MS81Fxx02:

BANKSEL PORTA ,因为PORTA在BANK0, 所以这条指令等同于BCR STATUS,5

BANKSEL TRISA ,因为TRISA 在BANK1, 所以这条指令等同于BSR STATUS,5

在MS83Fxx02中, 因为寄存器页面分了BANK0, BANK1, BANK2。通过STATUS的BIT5, BIT6去设置, 所以BANKSEL PORTA, 等同于两条指令

```
BCR STATUS,5
```

```
BCR STATUS,6
```

同样BANKSEL TRISA, 也等同于两条指令

```
BSR STATUS,5
```

```
BCR STATUS,6
```

这在精确计算指令数时需要特别注意。

## 7. 汇编指令时间的问题

LJUMP、LCALL、RET、RETI、RETW需要跳转的指令为双指令周期, 其他指令都是单指令周期。

BTSS、BTSC、INCRSZ、DECRSZ 这些指令在满足跳转条件时为双指令周期, 不满足跳转条件为单指令周期。

## 第五章 C 语言常见问题

### 1. 关于头文件包含

在程序的最前面用#include引用包含头文件，其中必须包含编译器提供的“syscfg.h”文件，实现单片机内特殊寄存器和其它特殊符号的声明。全系列MS单片机都依赖这个头文件。不用单独包含芯片型号的头文件，ORIGINIDE会自动调用。

请注意：老版本的是调用pic.h、pic16f684.h和pic16f685.h，现在不用调用这3个头文件，都统一调用syscfg.h

### 2. 关于EEPROM初始化

PIC单片机用\_\_EEPROM()指令来初始化EEPROM，而MS单片机需要调用初始化EEPROM二进制文件(bin)来初始化EEPROM。使用MS单片机时，程序中不能有\_\_EEPROM()这个指令，防止烧录失败和程序运行错误。关于MS单片机EEPROM初始化细节，可以参照例程库的EEPROM说明。

### 3. 关于配置位配置

PIC单片机用\_\_CONFIG()指令来配置配置位，而MS单片机是在点击编译(build)按钮后，弹出配置位信息表，依次选好每一项，最后点击确定。配置位信息就被设置在HEX中了。使用MS单片机时，程序中不能有\_\_CONFIG()这个指令，防止烧录失败和程序运行错误。

### 4. 关于中断服务函数

MS单片机中断向量入口在0x0004，有且只有一个入口。因此C编译器规定中断函数名为：void interrupt ISR(void)。

### 5. 关于中断现场保护

C编译器会自动生成中断现场保护和恢复现场代码，无须用户自己写现场保护代码。是用汇编才须自己编写现场保护和恢复现场代码。

### 6. 关于中断优先级

因为只有一个中断向量入口，所以所有中断的优先级是由用户自己定义。在中断服务函数中，哪个中断先被判断，那个中断的优先级就高。

### 7. 关于中断标志位

对应中断源没有使能，一旦达到中断条件，标志位也是会置1，但不会触发进入中断函

# MS8xF用户应答手册

数。比如  $T0IE=0$ ，但  $TMR0$  溢出后， $T0IF$  是会置 1，但不会进入中断服务函数。这种方式多数应用在对中断触发不须非常及时作出动作，在主循环里边检查此标志判断是否需要执行相应的代码。

## 8. 关于操作特殊寄存器和 RAM, BANK 的切换

C 编译器会根据操作的寄存器分配的地址，而自动切换 BANK，用户不用自行切换。

## 9. 关于\_XTAL\_FREQ 的定义

`_XTAL_FREQ` 定义是告知 C 编译器当前代码运行振荡器频率。从而可以调用延时的宏。例如：`#define _XTAL_FREQ 4000000`。这是告知 C 编译器当前代码运行在振荡器为 4MHz 的环境下。如果需要延时一毫秒，就可以这样写：`__delay_ms(1)`；如果要延时十微秒，就可以这样写：`__delay_us(10)`；

同时需要注意的是——C 编译器默认是运行在 4T 的指令周期，因此当我们配置位配置为 2T 的指令周期时，这个定义的频率就必须乘以二。比如本身振荡器是 4MHz，就必须定义为 8MHz。这样用延时宏延时的时间才准。

## 10. 关于整型变量类型

C 编译器支持 1、2、3 和 4 字节长的整型数据类型，以及单个位的类型。`bit` 和 `short long` 类型是该实现中提供的非标准类型。`long long` 类型是 C99 标准类型。所有整型值都使用最低位 (LSb) 位于低地址的小尾数法格式表示。如果类型中未指定符号，则除 `char` 类型（总是为无符号）之外的类型均为有符号。`bit` 类型总是为无符号，符号位的概念毫无意义。有符号值以二进制补码整型值的形式存储。下表列出了数据类型及其相应的长度和算术类型：

类型	长度(位)	算术类型
<code>bit</code>	1	无符号整型
<code>signed char</code>	8	有符号整型
<code>unsigned char</code>	8	无符号整型
<code>signed short</code>	16	有符号整型
<code>unsigned short</code>	16	无符号整型
<code>signed int</code>	16	有符号整型
<code>unsigned int</code>	16	无符号整型
<code>signed short long</code>	24	有符号整型
<code>unsigned short long</code>	24	无符号整型
<code>signed long</code>	32	有符号整型
<code>unsigned long</code>	32	无符号整型
<code>signed long long</code>	32	有符号整型
<code>unsigned long long</code>	32	无符号整型

## 11. 关于常量类型与格式

常量用于在源代码中表示立即数，与可以存放相同值的变量形成对照。例如，123 是一个常量。与所有值一样，常量必须具有 C 类型。除了常量的类型之外，还可以使用几种格式之一来指定实际值。

### 1、整型常量

整型常量的格式会指定其基数。C 编译器支持 ANSI 标准基数说明符，以及用于在 C 代码中指定二进制常量的说明符。

下表列出了用于指定基数的格式。与用于指定十六进制数字的字母相同，用于指定二进制或十六进制基数的字母也不区分大小写。

基数	格式	示例
二进制	0b 数字或者 0B 数字	0b10101010
八进制	0 数字	0257
十进制	数字	129
十六进制	0x 数字或 0X 数字	0xF7

整型常量将具有 int、long int 或 long long int 类型，以便类型可以在不发生溢出的情况下存放其值。对于以八进制或十六进制指定的常量，如果它们对应的有符号常量太小而无法存放其值，则还可以分配 unsigned int、unsigned long int 或 unsigned long long int 类型。

可以通过在数字后添加一个后缀来更改常量的默认类型；例如 23U，其中的 U 为后缀。

下表列出了在分配类型时考虑的后缀和类型的可能组合。例如，如果指定了后缀 l，并且值为一个十进制常量，并且如果 long int 类型可以存放该常量，则编译器会分配该类型；否则，它会分配 long long int 类型。如果该常量被指定为一个八进制或十六进制常量，则也会考虑无符号类型。

后缀	十进制	八进制或十六进制
u 或 U	unsigned int unsigned long int unsigned long long int	unsigned int unsigned long int unsigned long long int
l 或 L	long int long long int	long int unsigned long int long long int unsigned long long int
u 或 U，以及 l 或 L	unsigned long int unsigned long long int	unsigned long int unsigned long long int
ll 或 LL	long long int	long long int unsigned long long int
u 或 U，以及 ll 或 LL	unsigned long long int	unsigned long long int

以下给出了一个可能由于分配给常量的默认类型不适合而发生失败的代码示例：

```
unsigned long int result;
unsigned char shifter;
void main(void)
{
    shifter = 20;
    result = 1 << shifter;
```

```
// code that uses result
}
```

常量 1 将被赋予 `int` 类型, 因此移位操作的结果将为 `int` 类型。虽然将该结果赋予 `long` 变量 `result`, 它的长度永远不会大于 `int` 的长度, 无论对该常量进行了多少位的移位。在此例中, 值 1 被左移 20 位将产生结果 0, 而不是 0x100000。

以下代码使用后缀来更改常量的类型, 从而确保移位结果具有 `unsigned long` 类型。  
`result = 1UL << shifter;`

## 2、浮点型常量

浮点型常量具有 `double` 类型, 除非加上后缀 `f` 或 `F`, 在这种情况下它是一个 `float` 常量。后缀 `l` 或 `L` 用于指定 `long double` 类型 C 编译器将其视为与 `double` 相同的类型。

## 3、字符和字符串常量

字符常量使用单引号字符'包围, 例如'a'。字符常量具有 `int` 类型, 虽然编译器之后可能会将其优化为 `char` 类型。

为了符合 ANSI C 标准, 编译器不支持在字符或字符数组中使用扩展字符集。实际上, 它们需要使用反斜杠字符进行转义, 如以下示例所示:

```
const char name[] = "Bj\370rk";
printf("%s's Resum\351", name); \\ prints "Bj\370rk's Resum\351"
```

该实现不支持多字节字符常量。

字符串常量或字符串字面值使用双引号字符"包围, 例如"hello world"。字符串常量的类型为 `const char*`, 构成字符串的字符存储在程序存储器中, 与使用 `const` 限定的所有对象一样。

将字符串字面值赋予一个未指定 `const` 目标的指针时, 会产生一条常见的相关警告, 例如:

```
char * cp = "hello world\n";
```

字符串字符不能进行修改, 但这种类型的指针允许进行写入, 因此会产生该警告。为了防止自己试图覆盖字符串, 请如下使用限定符对指针目标进行限定。

```
const char * cp = "hello world\n";
```

定义并使用字符串初始化数组 (即, 不是指针) 属于例外情况。例如:

```
char ca[] = "hello world\n";
```

实际上会将字符串字符复制到 RAM 数组中, 而不是像前面的示例一样将字符的地址赋予指针。字符串字面值保持为只读, 但该数组是可读写的。

对于具有相同字符序列的字符串, C 编译器将使用相同的存储单元和标号 (用于初始化驻留在数据空间中的数组的字符串除外)。例如, 在以下代码片段中

```
if(strncmp(scp, "hello", 6) == 0)
fred = 0;
if(strcmp(scp, "world") == 0)
fred--;
if(strcmp(scp, "hello world") == 0)
fred++;
```

字符串"world" 和字符串"hello world" 中的最后 6 个字符(最后一个字符为 `nul` 终结符) 将由存储器中的相同字符表示。字符串"hello" 不会与字符串"hello world" 中的相同字符重叠, 因为两者中的 `nul` 字符的位置不同。



编译器会连接两个相邻的字符串常量（即，两个字符串仅使用空格进行分隔）。因而：

```
const char * cp = "hello" "world";
```

将向指针赋予字符串“hello world”的地址。

## 12. 关于位数据类型和变量

C编译器支持bit整型，它可存放值0或1。单个位变量可以使用关键字bit声明，例如：

```
bit init_flag;
```

这些变量不能为自动变量或函数参数，但可以使用static限定，从而可以在函数内局部定义它们。例如：

```
int func(void)
{
    static bit flame_on;
    // ...
}
```

通过以通常方式在函数原型中使用bit关键字，函数可以返回一个位对象。值1或0将返回到在STATUS寄存器中的进位标志中。

位变量的行为在大多数方面类似于常规的unsigned char变量，但它们只能包含值0和1，因此提供了一种便捷高效的方法来存储标志。8个位对象打包到存储器的每个字节中，所以它们不会消耗大量的内部RAM。

只要可能，对位对象的操作都会使用单个位指令（bsf和bcf）执行，因而生成代码访问位对象的效率极高。

无法声明一个指向bit类型的指针或将位对象的地址赋予任何指针。也无法以静态方式初始化位变量，所以必须在代码自身中为它们赋予任意非零起始值（即1）。除非使用persistent进行限定，否则在启动时，位对象会被清零。

将更长的整型赋予位变量时，将仅使用LSb。例如，如果位变量bitvar以如下形式赋值：

```
int data = 0x54;
```

```
bit bitvar;
```

```
bitvar = data;
```

赋值会将它清零，因为 data 的 LSb 为零。

## 13. 关于标准类型限定符

类型限定符可以提供关于如何使用对象的附加信息。C编译器支持ANSI C限定符和一些额外的特殊限定符，这些特殊限定符对于嵌入式应用非常有用，并充分利用了8位MS MCU架构。

### 1、CONST 类型限定符

const 类型限定符用于指示编译器，某个对象是只读的，不会被修改。如果尝试修改声明为 const 类型的对象，编译器将发出警告或错误。

如果用户定义的对象声明为 const 类型，则它们会被放入一个链接到程序空间的特殊 psect 中。限定为 const 类型的对象可以为绝对对象。@address 构造用于将对象放置在程序存储器中的指定地址处，如下示例所示，对象 tableDef 被放置在地址 0x100 处。

```
const int tableDef[] @ 0x100 = { 0, 1, 2, 3, 4};
```

通常，const 对象必须在声明时进行初始化，因为它不能在运行时的任何时间点赋值。

例如：

```
const int version = 3;
```

会将 `version` 定义为将放置在程序存储器中的 `int` 变量，并且将总是包含值 `3`，程序永远无法修改它。但是，如果需要将某个对象放置在程序存储器中某个特定位置的其他对象之上，则可以定义未初始化的 `const` 对象，它会非常有用。通常，未初始化的 `const` 对象将被定义为绝对对象，如以下示例所示。

```
const char checksumRange[0x100] @ 0x800;
```

会将对象 `checksumRange` 定义为 `0x100` 字节的字符数组，位于程序存储器的地址 `0x800` 处。该定义不会在 `HEX` 文件中放入任何数据。

## 2、VOLATILE 类型限定符

`volatile` 类型限定符用于指示编译器，无法保证某个对象在两次连续访问之间会保留其值。这可以防止优化器删除对 `volatile` 对象看起来多余的引用，因为这可能会改变程序执行这种引用的行为。

所有可以由硬件修改或驱动硬件的 `SFR` 都限定为 `volatile` 类型，可能由中断程序修改的所有变量也应使用该限定符。例如：

```
volatile static unsigned int TACTL @ 0x160;
```

`volatile` 限定符并不保证所有访问都是原子操作，对于 `8` 位 `PIC MCU` 架构通常就不是。对于所有这些器件，每条指令最多都只能访问 `1` 个字节的数据。

编译器生成的用于访问 `volatile` 对象的代码可能会与访问普通变量的代码不同，并且用于 `volatile` 对象的代码通常会较长且较慢，因此仅在必需时才使用该限定符。但是，在需要时未使用该限定符可能会导致代码失败。

`volatile` 关键字的另一个用途是防止未在 `C` 源代码中使用的变量被删除。如果某个非 `volatile` 变量从不使用，或者其使用方式对程序的功能没有任何影响，那么它可能会在编译器生成代码之前被删除。

如果某条 `C` 语句仅包含一个 `volatile` 变量的名称，则会生成读取变量的存储单元并丢弃结果的代码。例如，以下整条语句：

```
PORTB;
```

将生成读取 `PORTB` 但不对该值执行任何操作的汇编代码。对于一些需要通过读取来复位中断标志状态的外设寄存器，这是非常有用的。通常，这种语句将不进行编码，因为它没有任何作用。

## 14. 关于 C 编译器的优化

`C` 编译器中的优化可以大致分为 `C` 代码级优化（在源代码转换为汇编代码之前对源代码执行）和汇编级优化（对由编译器生成的汇编代码执行）。

`C` 代码级优化在代码生成阶段的早期执行，所以具有流程上的好处：执行一项优化可能意味着可以之后应用另一项。

由于这些优化是在生成调试信息之前应用的，所以对程序的源代码级调试通常几乎没有影响。

其中一些优化是代码生成过程不可或缺的组成部分，因此无法通过选项禁止。以下几节给出了关于如何废除一些具体优化的建议。

在标准模式下，特别是在免费模式下，其中一些优化会被禁止。即使使能了它们，也只有在满足非常特定的条件时才会应用它们。因此，您可能会发现一些代码进行了优化，

但有些则没有。

以下列出了可简化或更改C表达式的主要C代码级优化。

## 1、未用变量：

程序中的未用变量会被删除。对于被删除的变量，将不会为它们保留存储器，它们不会出现在任何列表或映射文件中，也不会出现在调试信息中出现，因此无法在调试器中观察到。如果遇到未用变量，则会产生一条警告。

使用`volatile`限定的对象永远不会被删除。获取变量的地址或是在手写汇编代码中引用它的汇编域符号也构成对变量的使用。

## 2、冗余赋值：

后续未被使用的冗余赋值会被删除，除非变量为`volatile`类型。赋值语句将被完全删除，就如同原始源代码中从未出现它。不会为它生成任何代码，您也无法在调试器中在该行上设置断点。

## 3、未用函数：

程序中的未用函数会被删除。如果某个函数未被直接或间接地调用，也没有任何代码获取其地址，则认为它是一个未用函数。整个函数将被删除，就如同原始源代码中从未出现它。不会为它生成任何代码，您也无法在调试器中在该函数中的行上设置断点。在独立的手写汇编模块中引用函数的汇编域符号可以防止它被删除。汇编代码只需在

`GLOBAL`伪指令中使用该符号即可。

未用返回表达式：

函数中的未用返回表达式会被删除。如果对某个函数的所有调用的结果都丢弃返回值，则返回值会被视为未用。与返回值计算关联的代码将被删除，函数将被编码为如同它的返回类型为`void`。

## 4、常量传播：

在可以确定变量的数值内容时，将执行常量传播。不属于`volatile`类型，可以准确确定其值的变量会被替换为数字值。对于未初始化的全局变量，将假定它们在进行任何赋值之前包含零。

## 第六章 MS8xF 使用常见问题

### 1. 同一系列的型号，芯片内部电路是否一样？

是一样的。比如 MS80F0601 和 MS80F0801 这两个型号的内部电路是一样的，MS80F0601 只是没有把 PA3/MCLR<sub>B</sub> 和 PA5 封装出来。其他系列(MS81F、MS82F、MS83F 和 MS84F)也是一个道理。

### 2. 没有封装出来的 IO 口如何初始化？

没有封装的出来的 IO 口，如果是有 MCLR(复位)功能，在配置位表将它设置为 MCLR，不作为 IO 口。其他的没有封装出来的 IO，如果该 IO 口有内部上拉，那么将它设置为数字输入且开启内部上拉。如果该 IO 口没有内部上拉，那么将他设置为数字输出（数字输出高和数字输出低都可以）。

之所以要这么设置，是因为在低功耗的应用中，如果有悬空的输入脚（没有封装出来的引脚算是悬空脚），悬空输入脚电平不固定，处于浮空状态，会有一定功耗。

例如：MS80F0601 的 PA3/MCLR<sub>B</sub> 脚和 PA5 脚没有封装出来，应当将 PA3/MCLR<sub>B</sub> 在配置位表设置为 MCLR，将 PA5 设置为输入脚且开启内部上拉。

### 3. 在低功耗应用应注意几个点？

- 1、普通 IO 在睡眠时，应根据应用需求来设置。如果是普通输出口，则应该设置成省电的输出。如果是普通输入口，则应使输入口保持固定的电平，不能处于浮空的状态（这里请注意前面第 2 点说的——“没有封装出来的 IO 口如何初始化？”）。
- 2、芯片没有用的功能外设尽量关闭。特别注意：用到 MS83F 和 MS84F 等带 ADC 模块的型号，如果 ADCON0 寄存器的 ADON 位被置 1，必须在睡眠前把它置 0；
- 3、LVR 模块启动后会有 15uA 左右的电流，所以在睡眠时可根据实际情况来选择是否要开启 LVR。注意：LVR 是可以通过软件开启和关闭，所以有些应用环境是可以这样设置——唤醒时，开始 LVR；睡眠时，关闭 LVR。

### 4. MS81F 的 IO 设置了 TRIS（比如 TRISA）寄存器后还是不能正常输入或输出？

这个情况多数发生在 MS81Fxx02 系列，应当将 CMCON0 寄存器初始化的时候设置为 0x07。因为这个系列的有带比较器，而 MCU 上电复位时，CMCOM0 是等于 0（比较器关闭，比较器对应的输入脚位模拟输入脚）。所以必须将 CMCON0 寄存器设置为 0x07，然后比较器关闭，且将对应的模拟输入脚设置为普通数字输入脚。

## 5. MS83F 的 IO 设置了 TRIS 和 CMCON0 寄存器后还是不能正常输入或输出？

因为部分 IO 复用了比较器功能和 ADC 功能，而上电复位时，对应的引脚都是默认为模拟输入脚。须将设置为：CMCON0=0x07 和 ANSELx=0。

## 6. MS84F 的 IO 设置了 TRIS、CMCON0 和 ANSEL 寄存器后还是不能正常输入或输出？

检测编译选项里 USFRSEL 选项是否为 DisUSFR，修改成 ENUSFR 后并将 CMCON4 的 CMPON[3-0]设置为 0000。

## 7. MS 单片机的时钟周期、指令周期分别指什么？

### 1、时钟周期：

时钟周期就是振荡周期，是时钟脉冲的倒数，即系统振荡器频率的倒数。例如系统时钟设置为内部 RC 振荡器 4MHz，那么时钟周期= $(1/4\text{MHz})=0.25\mu\text{s}$ 。

### 2、指令周期：

指令周期是指执行单周期指令所需要的时间，由 4 个或 2 个时钟周期组成。例如：MS80F0801 的指令周期有 4 个机器周期组成。如果系统时钟设置为内部 RC 振荡器 4MHz，那么指令周期= $(1/4\text{MHz}) * 4T=1\mu\text{s}$ 。而 MS81F、MS82F、MS83F 和 MS84F 可以选择 2T 的指令周期。

根据指令周期可以算出 Timer 的溢出时间，比如 Timer0 预分频比为 1:4，初值为 6，指令周期为 1us。那么溢出时间= $(256-6) * 1\mu\text{s} * 4=1000\mu\text{s}$ 。

## 8. 用 IO 电平变化中断(IOCA)，如何清除中断？

每一个 PORTA 引脚均可分别配置为电平变化中断引脚。控制位 IOCx 使能或者禁止每个引脚的中断功能。PAIE 使能或者禁止电平变化中断，PAIF 为电平变化中断标志。对于已允许电平变化中断的引脚，则将改引脚上的值同上一次读 PORTA 时锁存的值进行比较。将上一次“不匹配”的输出一起作逻辑或运算，以便将 INTCON 寄存器中的 PAIF 电平变化中断标志位置 1。

发生改中断后，用户在中断服务程序中通过以下方式清除中断：

- 1、对 PORTA 进行一次读操作。这将结束引脚电平不匹配条件；
- 2、将标志位 PAIF 清零。

## 9. MS83F 和 MS84F 的内部固定参考电压注意事项

如果 ADC 采用内部参考电压 2V 的时候，VCC 的供电电压要求高于 2.5V

如果 ADC 采用内部参考电压 3V 的时候，VCC 的供电电压要求高于 3.5V

# MS8xF用户应答手册

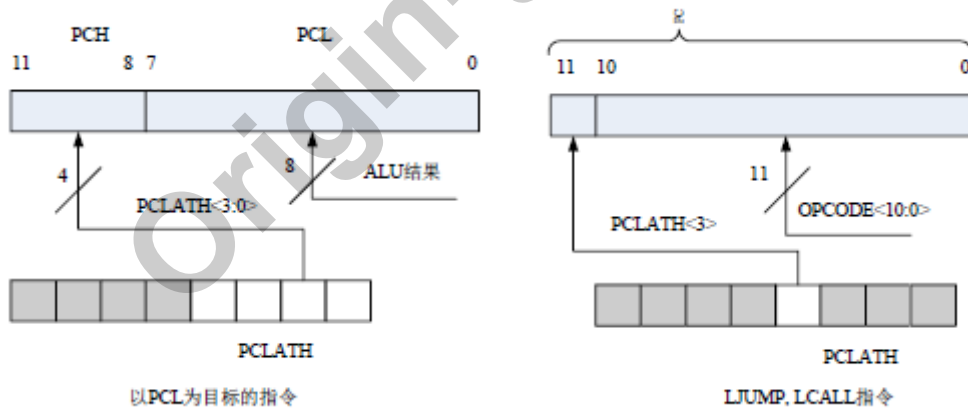
## 10. 使用 ADC 时，转换到的 AD 值不准？

- 1、检查参考源电压是否稳定，如果是用内部固定参考，检查是否符合第 9 点的要求；
- 2、检查 ADCON1 寄存器转换时钟的设置，须符合 ADC 时钟周期(Tad)表深色部分；
- 3、检查 ADC 程序，在把 ADON 置 1 后，有没有延时大概 20us，再 GO/DONE 置 1；

ADC 时钟周期(Tad)		系统时钟频率(Fsys)			
ADC 时钟源	ADCS[2:0]	16MHz	8MHz	4MHz	1MHz
Fsys/2	000	125ns	250ns	500ns	2.0us
Fsys/4	100	250ns	500ns	1.0us	4.0us
Fsys/8	001	500ns	1.0us	2.0us	8.0us
Fsys/16	101	1.0us	2.0us	4.0us	16.0us
Fsys/32	010	2.0us	4.0us	8.0us	32.0us
Fsys/64	110	4.0us	8.0us	16.0us	64.0us
Frc	x11	2~6us	2~6us	2~6us	2~6us

## 11. 汇编程序里超过 2K 空间的 LCALL 和 LJUMP 为什么会出错？

程序计数器(PC)为 12 位宽。其低 8 位来自可读写的 PCL 寄存器，高 4 位(PC[11:8])来自 PCLATH，不能直接读写。只要发生复位，PC 就将被清零。下图显示了装在 PC 值的两种情形。注意图右的 LCALL 和 LJUMP 指令，由于指令中的操作码为 11 位，而芯片的 PC 是 12 位，所以此时只用到 PCLATH 的第 3 位，和操作码的 11 位组成 12 位地址。



所以使用汇编编程时，直接操作 PCL 要设置 PCLATH[3:0]  
 2K 内的 LCALL 和 LJUMP 要设置 BCR PCLATH.3  
 2K 外的 LCALL 和 LJUMP 要设置 BSR PCLATH.3  
 使用 C 语言编程时，无须考虑此问题，编译器会自动处理。

## 联系我们

深圳市粤原点科技有限公司

SHENZHEN ORIGIN-GD TECH CO.,LTD

WEB:www.origin-gd.com

TEL: 0755-83666320

FAX: 0755-83666329

PHONE: 18344146830 13510476700 13902985185

FAE QQ: 2850507666

ADDRESS: 广东省深圳市龙岗区坂田街道环城南路 5 号坂田国际中心 E 栋 605 室

Origin-gd Tech