MPASM™汇编器至 MPLAB® XC8 PIC®汇编器移植指南

客户须知

本文档如同所有其他文档一样具有时效性。Microchip 将不断改进工具和文档以满足客户的需求，因此实际使用中有些对话框和/或工具说明可能与本文档所述之内容有所不同。请访问我们的网站（https://www.microchip.com）获取最新文档。

文档均标记有“DS”编号。该编号出现在每页底部的页码之前。DS 编号的命名约定为“DSXXXXXXXXA_CN”，其中“XXXXXXXX”为文档编号，“A”为文档版本。

欲了解开发工具的最新信息，请参见 MPLAB® IDE 在线帮助。从 Help（帮助）菜单选择 Topics（主题），打开现有在线帮助文件列表。

目录

1. 前言

1.1 本指南使用的约定

本指南采用以下文档约定：

表 1-1. 文档约定

说明	表示	示例
Arial 字体：		
斜体字	参考书目	*MPLAB® IDE 用户指南*
	需强调的文字	...*仅有*的编译器...
首字母大写	窗口	Output 窗口
	对话框	Settings 对话框
	菜单选择	选择 Enable Programmer
引用	窗口或对话框中的字段名	“Save project before build”
带右尖括号有下划线的斜体文字	菜单路径	*File>Save*
粗体字	对话框按钮	单击 **OK**（确定）
	选项卡	单击 **Power** 选项卡
N‘Rnnnn	verilog 格式的数字，其中 N 为总位数，R 为基数，n 为其中一位。	4‘b0010，2‘hF1
尖括号< >括起的文字	键盘上的按键	按下<Enter>，<F1>
Courier New 字体：		
常规 Courier New	源代码示例	`#define START`
	文件名	`autoexec.bat`
	文件路径	`c:\mcc18\h`
	关键字	`_asm，_endasm，static`
	命令行选项	`-Opa+，-Opa-`
	二进制位值	`0，1`
	常量	`0xFF，‘A’`
斜体 Courier New	可变参数	*`file.o`*，其中 *`file`* 可以是任何有效的文件名
方括号[ ]	可选参数	`mcc18 [options] file [options]`
花括号和竖线：{ | }	选择互斥参数；“或”选择	`errorlevel {0|1}`
省略号...	代替重复文字	`var_name [, var_name...]`

	表示由用户提供的代码	`void main (void)`
		`{ ...`
		`}`

1.2 推荐读物

本指南适用于拥有 MPASM 汇编器项目并希望将其移植到 MPLAB XC8 PIC 汇编器的客户。以下 Microchip 文档均已提供，并建议读者作为补充参考资料。

MPLAB® XC8 PIC Assembler User's Guide
本用户指南介绍了 MPLAB XC8 PIC 汇编器的用法和功能。

MPLAB® XC8 PIC Assembler Guide For Embedded Engineers
本入门指南介绍了 MPLAB XC8 PIC 汇编器示例项目和常用的编码序列。如果需要使用汇编器开发新项目，请使用本指南。

MPLAB® XC8 C Compiler Release Notes for PIC MCU
有关此汇编器的更改和缺陷修复的最新信息，请阅读 MPLAB XC8 安装目录的 docs 子目录下的自述文件。

开发工具发行说明
有关使用其他开发工具的最新信息，请阅读与该工具相关的自述文件，文件位于 MPLAB X IDE 安装目录的 docs 子目录下。

1.3 文档版本历史

版本 A（2020 年 3 月）

- 本文档的初始版本。

2. 简介

本指南重点介绍您决定将项目从 Microchip MPASM™汇编器（MPASM）移植到 MPLAB® XC8 PIC 汇编器（PIC 汇编器）时可能需要的源代码和编译更改。

移植项目时所需的大部分更改都涉及到汇编器伪指令。尽管某些指令操作数表达式可能需要使用不同的运算符或语法，但指令序列通常不需要更改。

移植的项目几乎肯定会生成一个十六进制文件，该文件与使用 MPASM 汇编器编译的原始 MPASM 项目生成的文件不同。当使用 PIC 汇编器进行编译时，对象和代码可能链接到不同的地址，这意味着指令将使用不同的地址操作数，并且可能存在不同的存储区和页选择序列。但是，指令序列应保持不变，因为 PIC 汇编器不执行任何优化或代码转换。

您可以在 MPLAB X IDE 中使用 PIC 汇编器。可以创建专用于此工具的项目，这些项目将使用项目属性中显示的自带选项集。

有关如何使用汇编器的完整信息以及有关汇编器的伪指令和语言的更多详细信息，请参见 *MPLAB® XC8 PIC Assembler User's Guide*。还有另外一个 *MPLAB® XC8 PIC Assembler Guide for Embedded Engineers* 文档包含代码、编译选项示例和入门信息。

2.1 文件类型

PIC 汇编器使用的源文件扩展名与 MPASM 使用的源文件扩展名不同。

为汇编源文件使用扩展名.s（小写）。如果为汇编源文件使用扩展名.S（大写），则必须先对这些文件进行预处理，然后才能传递给汇编器。或者，也可使用-xassembler-with-cpp 选项请求对源文件（无论其扩展名如何）进行预处理。

下表列出了常用的等效文件扩展名。

表 2-1. 等效文件扩展名

MPASM 文件扩展名	文件类型	PIC 汇编器等效选项
.asm	汇编源文件	.s 或.S
.inc	包含（头）文件	.inc
.hex	十六进制文件输出	.hex
.o	目标文件	.o
.lib	库归档	.a

PIC 汇编器将生成.hex 和.elf 输出文件，每个文件的基本名称与命令行上列出的第一个源文件相同。如果是从 MPLAB X IDE 中编译的，则文件的基本名称将是项目名称。请注意，通常由驱动程序运行的 Hexmate 可创建扩展名为.hxl 的日志文件。请勿将此文件与同样使用此扩展名的 MPASM 拆分十六进制文件混淆。

目标文件是汇编器使用的中间文件格式，但是请注意，MPASM 和 PIC 汇编器目标文件的格式有很大不同。PIC 汇编器无法读取 MPASM 创建的目标文件，因此不能将这些文件包含在 PIC 汇编器项目中，而是将移植的原始源代码（用于编译 MPASM 目标文件）包含在项目中。

通过归档器 xc8-ar 创建的库归档文件应使用扩展名.a。可通过命令行将库归档文件传递给汇编器。尽管这些归档可能包含 p 代码或目标模块的任何组合，但汇编器仅搜索目标模块。请注意，MPASM 和 PIC 汇编器库归档的格式有很大不同。PIC 汇编器无法读取 MPASM 创建的库文件，因此不能将这些文件包含在 PIC 汇编器项目中，而是将移植的原始源代码（用于编译 MPASM 库）包含在项目中。

PIC 汇编器不支持生成交叉引用文件。MPASM 汇编器能够生成这些文件（扩展名为.xrf），这些文件可用于获取程序符号列表。使用 PIC 汇编器时，模块使用的所有符号都显示在与该模块关联的汇编列表文件中。整个程序范围的所有全局符号也显示在映射文件中。

2.2 命令行选项

MPASM 使用的某些命令行选项在 PIC 汇编器中具有等效的选项。

所有 PIC 汇编器选项均以短线-或双短线--开头。选项区分大小写。

下表列出了 MPASM 命令行选项及其 PIC 汇编器的等效命令行选项。

表 2-2. 等效命令行选项

MPASM 选项	用途	PIC 汇编器等效运算符
?或 h	显示帮助	--help
a*hex-format*	指定十六进制文件输出和格式	使用-g 指定十六进制文件的类型；代码始终可重定位
c	控制是否区分大小写	无等效选项
d*label*[=*value*]	定义文字替换	-D*macro*=*text* 控制的预处理器功能类似
e	指定错误文件	无等效选项
l 或 l+	使能列表文件	-Wa,-a
l *path*	指定列表文件路径	无等效选项
m	使能宏扩展	无等效选项
o	使能目标文件并指定路径	无等效选项
p*device*	指定目标器件	-mcpu=*device*
q	安静模式	无等效选项
r*radix*	指定默认基数	无等效选项
s	显示进度窗口	无等效选项
t	设置制表符长度	无等效选项
w*value*	指定消息输出	-w，另请参见-mwarn。
x	使能交叉引用文件并指定路径	无等效选项
y	指定 PIC18 指令集	-misa=[std|xinst]

2.3 可重定位代码

MPLAB XC8 PIC 汇编器程序包包含一个链接器 hlink，并且始终以在 MPASM 文档中称为可重定位代码的方式来编译代码。也就是说，PIC 汇编器生成可重定位的目标模块，这些目标模块可与其他模块或归档库链接以形成最终程序映像。

通常，整个编译过程中始终运行 PIC 汇编器，以便生成项目的最终映像。但是，如有需要，可以使用-c 选项在链接步骤之前停止编译过程。这会生成目标文件输出，稍后必须链接该输出（如有需要，与其他目标模块或库归档链接在一起）以形成最终程序映像。

3. 表达式和运算符

与 MPASM 汇编器相比，PIC 汇编器中对指令和汇编器伪指令的操作数的表示和求值方式有所不同。

3.1 常量和基数

PIC 汇编器中常量的默认基数与 MPASM 使用的基数不同，并且每个汇编器使用的基数说明符也不同。

在没有任何基数说明符或伪指令时，PIC 汇编器中的数字常量将解析为十进制值。这些值在 MPASM 中将解析为十六进制。在移植的汇编代码中使用 `RADIX hex` 伪指令，以确保 PIC 汇编器采用的默认基数与 MPASM 使用的默认基数匹配。

下表列出了 MPASM 基数说明符和 PIC 汇编器使用的等效说明符。

表 3-1. 等效常数基数说明符

MPASM 常量形式	基数	PIC 汇编器等效说明符
B'*binary_digits*'	二进制	*binary_digits*B
O'*octal_digits*'	八进制	*octal_digits*[O|o|Q|q]
D'*decimal_digits*' 或.*decimal_digits*	十进制	*decimal_digits*[D|d|nothing]
H'*hexadecimal_digits*'或 0x*hexadecimal_digits*	十六进制	0*hexadecimal_digits*[H|h]或 0x*hexadecimal_digits*
A'*character*'或'*character*'	ASCII	'*character*'

请注意，PIC 汇编器使用的二进制位后缀（B）必须为大写，以避免与临时标号混淆。十六进制值必须始终以 0 开头。

以下示例给出了 PIC 汇编器使用的基数说明符。

```
movlw 10110011B  ;binary value
movlw 72q        ;octal value
movlw 34         ;decimal value
movlw 04Fh       ;hexadecimal value
movlw 'b'        ;ASCII value
```

3.2 标号

在标号的定义方面，PIC 汇编器比 MPASM 更加严格。

标号是一个赋值等于当前 psect 内当前存储单元的符号别名。此赋值通常由链接器执行。

在 PIC 汇编器中，标号定义由任意有效汇编标识符后跟一个冒号:组成。在 MPASM 中，冒号是选项。移植时，必须向 MPASM 代码中尚未使用冒号的任何标号添加冒号。

PIC 汇编器使用的标号标识符始终区分大小写。

标号标识符可以包含任意数量的以下字符：字母、数字和特殊美元字符$、问号?和下划线_。标识符的第一个字符不能为数字。标识符不能与任何汇编器伪指令、关键字或 psect 标志具有相同的名称。标号定义可以单独在一行上，也可以位于指令或汇编器伪指令的左侧。

下面给出了 PIC 汇编器中有效且惟一的标号的定义。

```
An_identifier:
  movlw 55
an_identifier:  movlw  0AAh
an_identifier1:  DW 0x1234
?$_12345:
  return
```

3.3 文件寄存器地址掩码

建议对操作数表示存储区偏移量的 PIC 文件寄存器指令使用的所有地址进行掩码。

标号（例如 RAM 中对象的标号）的值始终是标号在存储器中所处位置的完整地址。针对文件寄存器进行操作的大多数 PIC 指令仅要求将存储区内的偏移量指定为文件寄存器操作数。必须删除指示存储区编号的地址的高位。如果 PIC 汇编器检测到文件寄存器地址操作数中有多余的存储区信息，则会发出修正溢出错误（对于符号操作数）或警告（对于绝对操作数）。

可以采用两种常见方法对地址操作数进行掩码。

- 通常使用 BANKMASK()宏通过按位与（&运算符）掩码掉地址中的存储区信息。
- 通过按位异或（^运算符）掩码掉地址中的存储区信息。

这两种方法使用的掩码值因器件而异，因为不同器件系列具有不同大小的存储区。

下面列出了每个器件系列的地址组成。它们均由存储区值 b 和该存储区中的偏移量 o 组成。如下所示，对于数据存储区大小为 0x20 字节的低档器件，其地址由 5 位存储区偏移量和使用其余高位的存储区值组成。（并非每款器件上都实现了所示存储区信息的所有位）。例如，低档器件地址 0x65（二进制位模式 011**0 0101**）是存储区 3 中存储区偏移量 **5** 处的字节的地址。相比之下，PIC18 器件具有 8 位存储区偏移量，其余位为存储区值。

```
Address decomposition (b: bank value, o: bank offset)
bit position:   15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
Baseline:        b  b  b  b  b  b  b  b  b  b  b  o  o  o  o  o
Mid-range:       b  b  b  b  b  b  b  b  o  o  o  o  o  o  o  o
PIC18:           b  b  b  b  b  b  b  b  o  o  o  o  o  o  o  o
```

要通过逻辑与运算删除地址中的存储区值，应使用满足如下条件的掩码：与存储区偏移量对应的每个位置为 1，其余位置为 0。例如，在 PIC18 器件上，掩码为 0xFF；在中档器件上，掩码为 0x7F；在低档器件上，掩码为 0x1F。包含<xc.inc>之后，BANKMASK()宏可用，可基于选定器件使用正确的值来执行掩码。以下示例给出了该宏的用法。

```
#include <xc.inc>

copy:
  BANKSEL (src)               ;select the bank of src
  movf     BANKMASK(src),w    ;move from src, masking the address
  BANKSEL (dst)               ;select the bank of dst
  movwf    BANKMASK(dst)      ;move to dst, masking the address
```

另外，还可使用异或运算对地址进行掩码，这样便有机会执行其他检查，以确保关于对象位置的假设是正确的。应将地址与相应值进行异或运算，从而将*预期*存储区值清零，同时使存储区偏移量保持不变。在该值中，存储区偏移量位为 0，并且会将对象所在存储区的位模式指定为存储区值。例如，如果假定地址操作数是存储区 1 中的对象，则在 PIC18 器件上将其与掩码 0x100 进行异或运算；如果地址操作数将用作存储区 3 中的操作数，则将其与 0x300 进行异或运算。在中档器件上，将存储区 1 的对象与 0x80 进行异或运算；将存储区 3 的对象与 0x180 进行异或运算。在低档器件上，将存储区 1 的对象与 0x20 进行异或运算；将存储区 3 的对象与 0x60 进行异或运算，依此类推。在下面的中档器件示例中，如果 src 不在存储区 1 中或 dst 不在存储区 2 中，将产生错误。

```
#include <xc.inc>

copy:
  BANKSEL (src)               ;select the bank of src
  movf    src^080h,w          ;move from src (in bank 1)
  BANKSEL (dst)               ;select the bank of dst
  movwf   dst^0100h           ;move to dst (in bank 2)
```

此外，也可以将地址操作数与符号值进行异或运算。在下面的中档器件示例中，编程人员已假定上一个示例中使用的 scr 和 dst 位于同一存储区中，因此仅存在一个存储区选择序列，这样便可减小代码长度并加快执行速度。 可以使用这两个符号执行异或运算，以确保此假设有效。 在最后一条指令中，表达式 src&0FF80h 获取 src 的存储区值。该值随后与 dst 的完整地址进行异或运算。

```
#include <xc.inc>

copy:
  BANKSEL (src)               ;select the bank of src
```

```
movf    BANKMASK(src),w    ;move from src, masking the address
movwf   dst^(src&0FF80h)   ;move to dst, which should be in the same bank as src
```

如果上述代码中的 scr 和 dst 随后链接到同一个存储区，则对表示其存储区值的位进行异或运算后将得到 0，这样便可有效掩码掉 movwf 指令的操作数地址中的存储区值。如果 src 和 dst 链接到不同的存储区，则异或运算将使存储区值中产生会触发错误的非零位，这样便可驳倒编程人员的假设并防止代码在运行时失败。在这种情况下，这些对象位于哪个存储区中无关紧要，但它们必须位于同一个存储区中才能正确执行代码。

对于接受完整地址的指令（例如 movff 或 movffl 指令），不得对用作其操作数的任何地址进行掩码。如果对象位于存储区 0 中，则不需要地址掩码（因为其存储区值已为 0），但对所有地址操作数进行掩码是一个好的做法。

3.4 运算符

PIC 汇编器定义的运算符可用于大多数表达式。与 MPASM 中的运算符不同，PIC 汇编器运算符没有运算符优先级，它们都遵从从左到右的结合律。在移植复杂表达式时，可能需要使用括号，以确保其求值与在 MPASM 中一样。

下表列出了 MPASM 运算符及其 PIC 汇编器等效运算符。

表 3-2. 等效运算符

MPASM 运算符	用途	PIC 汇编器等效运算符
$	当前/返回程序计数器	$
(	左括号	(
)	右括号	)
!	逻辑取反（NOT）	无等效运算符
-	取补（二进制补码）	-
~	取反	not
low	低地址字节	low
high	高地址字节	high
upper	最高地址字节	low highword
*	乘法	*
/	除法	/
%	求模	%
+	加法	+
-	减法	-
<<	左移	<<或 shl
>>	右移	>>或 shr
>=	大于或等于	>=或 ge
>	大于	>或 gt
<	小于	<或 lt
<=	小于或等于	<=或 le
==	等于	=或 eq
!=	不等于	<>或 ne
&	按位与	&或 and

.....（续）

MPASM 运算符	用途	PIC 汇编器等效运算符
^	按位异或	^
|	按位或	|
&&	逻辑与	无等效运算符
||	逻辑或	无等效运算符
=	赋值	无等效运算符
+=	加法赋值	无等效运算符
-=	减法赋值	无等效运算符
*=	乘法赋值	无等效运算符
/=	除法赋值	无等效运算符
%=	求模赋值	无等效运算符
<<=	左移赋值	无等效运算符
>>=	右移赋值	无等效运算符
&=	逻辑与赋值	无等效运算符
|=	逻辑或赋值	无等效运算符
^=	逻辑异或赋值	无等效运算符
++	递增	无等效运算符
--	递减	无等效运算符

4. 汇编器伪指令

某些 MPASM 汇编器伪指令在 MPLAB XC8 PIC 汇编器中具有等效伪指令；但其他 MPASM 伪指令必须进行改写以适应新的语法或替换为其他伪指令序列。

下表列出了每个 MPASM 伪指令及其 PIC 汇编器最佳等效伪指令。以下各节中将更详细地讨论这些伪指令及其等效伪指令。请注意，看起来相同的伪指令在行为上可能存在细微差别，这可能导致程序出现意外行为。建议您将 *MPLAB® XC8 PIC Assembler User's Guide* 与 MPASM 文档进行对比，确保您移植的代码在所有情形下均能够按预期工作。

表 4-1. 等效伪指令

MPASM 伪指令	PIC 汇编器替代伪指令
ACCESS_OVR	ovrld 标志置 1 的 psect
__BADRAM 和__BADROM	无替代伪指令
BANSISEL	写入间接访问寄存器的指令序列
BANKSEL	BANKSEL 伪指令（无需更改）
CBLOCK	考虑使用 SET/EQU 伪指令或 DS。
CODE	code psect 或类似 psect
CODE_PACK	code psect 或类似 psect
__CONFIG	具有适当设置和值的 CONFIG 伪指令
CONFIG	具有适当设置和值的 CONFIG 伪指令
CONSTANT	EQU 伪指令
DA	考虑使用 DB 或 IRPC 伪指令
DATA	考虑使用 DW 伪指令
DB	DB 伪指令
DE	考虑使用适当的 psect 内部的 DB 伪指令
#DEFINE	#define 预处理器伪指令
DT	IRP 伪指令
DTM	IRP 伪指令
DW	DW 或 DB 伪指令
ELSE	ELSE 伪指令（无需更改）
END	END 伪指令（无需更改）
ENDC	无替代伪指令
ENDM	ENDM 伪指令（无需更改）
ENDW	无替代伪指令
EQU	EQU 伪指令（无需更改）
ERROR	ERROR 伪指令
ERRORLEVEL	考虑使用-w 驱动程序选项
EXITM	无替代伪指令

.....（续）

MPASM 伪指令	PIC 汇编器替代伪指令
EXPAND	EXPAND 伪指令（无需更改）
EXTERN	EXTRN 伪指令（注意不同的拼写）
FILL	考虑使用--fill 驱动程序选项
GLOBAL	GLOBAL 伪指令（无需更改）
IDATA	通过一个 psect 在程序存储器中存放初始值，通过另一个 psect 为数据对象保留空间
IDATA_ACS	通过一个 psect 在程序存储器中存放初始值，通过另一个 psect 为数据对象保留空间
IF	IF 伪指令（无需更改）
IFDEF	考虑使用#ifdef 预处理器伪指令
IFNDEF	考虑使用#ifndef 预处理器伪指令
#INCLUDE	#include 预处理器伪指令
LIST	LIST 伪指令，也可考虑使用其他汇编器选项
LOCAL	LOCAL 伪指令（无需更改）
MACRO	MACRO 伪指令（无需更改）
__MAXRAM 和__MAXROM	无替代伪指令
MESSG	MESSG 伪指令（无需更改）
NOEXPAND	NOEXPAND 伪指令（无需更改）
NOLIST	NOLIST 伪指令（无需更改）
ORG	考虑使用 ORG 伪指令
PAGE	无替代伪指令
PAGESEL	PAGESEL 伪指令（无需更改）
PAGESELW	考虑使用 PAGESEL 伪指令
PROCESSOR	PROCESSOR 伪指令（无需更改）
RADIX	RADIX 伪指令（无需更改）
RES	考虑使用 DS 伪指令
SET	SET 伪指令（无需更改）
SPACE	SPACE 伪指令（无需更改）
SUBTITLE	SUBTITLE 伪指令（无需更改）
TITLE	TITLE 伪指令（无需更改）
UDATA	udata_bank*n* psect 或类似 psect
UDATA_ACS	udata_acs psect 或类似 psect
UDATA_OVR	ovrld 标志置 1 的 psect

.....（续）

MPASM 伪指令	PIC 汇编器替代伪指令
UDATA_SHR	ovrld 标志置 1 的 psect
#UNDEFINE	#undefine 预处理器伪指令
VARIABLE	考虑使用 SET 伪指令
WHILE	考虑使用 REPT 伪指令

4.1 Access_ovr 伪指令

在 PIC18 器件上，MPASM ACCESS_OVR 伪指令声明快速操作存储区 RAM 中重叠数据段的开头。

建议的替代伪指令

定义 ovrld 标志置 1 的 psect。将 psect 与快速操作存储区链接器类 COMRAM 相关联，以将其链接到 PIC18 快速操作存储区中的某个位置。

以下示例给出了放置在 myData psect 中的对象，该 psect 在两个不同的模块（文件 1 和文件 2）中使用。两个模块中的 PSECT 伪指令使用相同的 psect 名称、ovrld 标志（用于指示 psect 将重叠）和 space=1 标志（用于指示 psect 将位于数据存储空间中）。该 psect 与 PIC18 COMRAM 链接器类相关联，该类定义了快速操作存储区，因此如果 psect 重叠，myData 将出现在该类定义的存储器中的任何位置。

```
;file 1
PSECT myData,space=1,ovrld,class=COMRAM
zero:
  DS 1
  ;leave a 1-byte gap for another object here
  ORG 2
two:
  DS 1

;file 2
PSECT myData,space=1,ovrld,class=COMRAM
  ORG 1
one:
  DS 1
```

请注意，每个模块中放置在重叠 psect 中的对象串连在一起，但每个模块的 psect 随后在链接时重叠。上述示例编译完成后，标号将按 zero、one、two 的顺序出现在存储器中。

上述示例中的 ORG 伪指令允许 psect 的内容交错。如果文件 1 中未使用该伪指令，则与标号 zero 和标号 one 关联的空间将重叠，并且这些对象将出现在相同的地址上。此类代码是合法的，可能为某些应用所需。

重叠的 psect 可以链接到任何 RAM 区域中，而不仅仅是快速操作存储区，并且只要选择适当的链接器类，此结构便可适用于任何器件。

4.2 Badram 和 Badrom 伪指令

MPASM __BADRAM 和__BADROM 伪指令与__MAXRAM 和__MAXROM 伪指令一起指定文件寄存器地址范围，如果代码使用这些地址范围，则应将其标记为无效。

建议的替代伪指令

这些伪指令没有替代伪指令。

使用-mreserve 驱动程序选项会将链接器类限制为所需的存储器范围。如果仅使用归为这些类的 psect 中定义的符号，则可避免使用无效的存储区。

此外，还可使用_RAMSIZE 和_ROMSIZE 预处理器宏，它们分别指示所选目标器件提供的最大数据和程序存储空间地址。

```
#define DEST 0x7D0
PSECT text,CLASS=code,reloc=2  ;for PIC18 devices
;for other devices: PSECT text,class=CODE,delta=2
storeIt:
  movlw 66
#if DEST > _ROMSIZE
#error "destination out of bounds"
#endif
  BANKSEL (DEST)
  movwf BANKMASK(DEST)
  return
```

4.3 Bankisel 伪指令

仅在中档器件上，MPASM BANKISEL 伪指令生成适用于间接访问其参数指定的寄存器地址的存储区选择代码。

建议的替代伪指令
PIC 汇编器中没有等效的伪指令。

根据后面代码的要求，将该指令替换为设置与间接访问关联的寄存器的指令。有关间接访问涉及的寄存器，请参见具体器件数据手册。

4.4 Banksel 伪指令

MPASM BANKSEL 伪指令生成适用于指定为参数的标号的存储区选择代码。

建议的替代伪指令
无需更改；继续使用 BANKSEL 汇编器伪指令。

该伪指令不区分大小写，可与任何器件以及数字或符号操作数一起使用。请注意，它可能会生成多条指令，因此不应紧跟在测试并跳过指令之后使用。

```
  movlw 66
  BANKSEL input
  movwf BANKMASK(input)
```

4.5 Cblock 伪指令

MPASM CBLOCK 伪指令生成具有连续地址的标号组。

建议的替代伪指令
PIC 汇编器中没有等效的伪指令。

使用 SET 或 EQU 伪指令明确定义所需的符号。如果要将存储与每个标号关联，则需在适当的 psect 中定义标号，后跟 DS 伪指令。

4.6 Code 伪指令

MPASM CODE 伪指令开始包含程序代码的段。

建议的替代伪指令
使用预定义的 code psect 或创建类似的 psect，确保标志适用于目标器件上包含可执行代码的段。如果必须按特定大小的倍数填充 psect，请在置于 psect 中的指令之后使用 ALIGN 伪指令。

必须使用 PSECT 伪指令将所有程序代码置于 psect（或段）中。包含<xc.inc>后，PIC 汇编器将提供 code psect。使用该 psect 时不必指定任何 psect 标志，例如：

```
PSECT code
;place code for any device here
```

或者，也可以使用任何名称和适当的 psect 标志定义自己的 psect。psect 的 space 标志必须为 0，指示 psect 应位于程序存储器中；不过，这是默认值。 通常，将使用 class 标志将 psect 分配给 CODE 链接器类（该类也是由驱动程序预定义的），因此 psect 将位于与该类关联的存储器中的某个位置，而无需指定任何链接器选项。此外，也可使用链接器的-p 选项（通过驱动程序的-Wl 选项传递给链接器）将该 psect 置于特定地址。

对于 PIC18 器件，将 psect 的 reloc 标志设置为 2，指示 psect 内容必须在偶数地址上对齐。使用默认 delta 值 1。例如：

```
PSECT myText,reloc=2,class=CODE
;PIC18 code goes here
```

对于所有其他器件，使用默认 reloc 标志值 1，但将 delta 值设置为 2，指示器件使用可（16 位）字寻址程序存储器。例如：

```
PSECT myText,delta=2,class=CODE
;Baseline/Mid-range code goes here
```

稍后可以使用具有 psect 名称的 PSECT 伪指令向源文件中的同一 psect 添加更多内容，但无需重复 psect 标志。例如，可以使用以下代码将更多内容串连到上面定义的 psect：

```
PSECT myText
;more content goes here
```

4.7 Code_pack 伪指令

MPASM CODE_PACK 伪指令开始包含程序代码的段，该段不会填充为偶数长度。

建议的替代伪指令
使用预定义的 code psect 或创建类似的 psect，确保标志适用于目标器件上包含可执行代码的段。

有关更多信息和示例，请参见 4.6 Code 伪指令。

4.8 __config 伪指令

MPASM __CONFIG 伪指令设置器件的配置位。

建议的替代伪指令
使用 CONFIG 伪指令以及与目标器件和应用相关的设置值对。请参见 4.9 Config 伪指令。

4.9 Config 伪指令

MPASM CONFIG 伪指令设置器件的配置位。

建议的替代伪指令
PIC 汇编器的 CONFIG 伪指令是直接替代伪指令，但需确保与伪指令一起使用的设置和值适合您的器件。

该伪指令可用于所有器件。以下示例给出了可用于编程各个配置位、整个配置字或 ID 存储单元值的不同方式。

```
; PIC18F67K22
; VREG Sleep Enable bit : Enabled
; LF-INTOSC Low-power Enable bit : LF-INTOSC in High-power mode during Sleep
; SOSC Power Selection and mode Configuration bits : High Power SOSC circuit selected
```

```
; Extended Instruction Set : Enabled
config RETEN = ON, INTOSCSEL = HIGH, SOSCSEL = HIGH, XINST = ON

; Alternatively, set the entire word
config CONFIG1L = 0x5D

; IDLOC @ 0x200000
config IDLOC0 = 0x15
```

有关该伪指令的完整信息，请参见 *MPLAB® XC8 PIC Assembler User's Guide*。

4.10 Constant 伪指令

MPASM CONSTANT 伪指令声明初始值不能更改的符号。

建议的替代伪指令

使用 PIC 汇编器的 EQU 伪指令，例如：

```
threshold EQU 0xFF
```

4.11 Da 伪指令

MPASM DA 伪指令将一个字符串中由两个 ASCII 字符组成的每个序列打包到程序存储器中的一个 14 位字中。

建议的替代伪指令

PIC 汇编器在适当的 psect 内部的 DB 伪指令执行类似的功能，但请注意，PIC 汇编器不支持打包的字符串，并且每个字符在存储区中将为完整字节。

包含<xc.inc>后，考虑将 IRPC 和/或 DB 伪指令置于提供的 data psect 中，例如：

```
#include <xc.inc>

PSECT data
myString:
  IRPC char,ABC
    DB    ' char'
  ENDM
```

但请注意，每个字符都将占用存储器的一个完整字节。

另外，也可以定义自己的 psect 并将其分配到程序存储器。确保 psect 的 space 标志设置为 0（默认值）。可通过将该 psect 与适当的链接器类（例如，PIC18 器件为 CONST，其他器件为 STRCODE）相关联，或者使用链接器的-P 选项（可使用-Wl 选项从驱动程序访问）明确定位该 psect，如以下 PIC18 示例所示。

```
PSECT romData,space=0,class=CONST
myString:
    DB    ' A' , 'B', 'C'
```

4.12 Data 伪指令

MPASM DATA 伪指令使用数据初始化程序存储器的一个或多个字。

建议的替代伪指令

此伪指令没有直接替代伪指令，但 PIC 汇编器的 DW 伪指令可执行类似的功能。有关更多信息和示例，请参见 4.18 Dw 伪指令。

4.13 Db 伪指令

MPASM DB 伪指令将字节置于程序存储器中。

建议的替代伪指令
PIC 汇编器在适当的 psect 内部的 DB 伪指令可执行类似的功能，但操作上存在一些差异。

该伪指令将其操作数的值以字节为单位置于当前 psect 中。该伪指令不支持字符串立即数操作数，但可以使用字符和多个逗号分隔的操作数，如下面的示例所示。

对于 PIC18 器件，指定的每个字节将占用程序存储器的一个字节。对于中档器件，每个程序字中将存储一个字节，而该字的高位为 0。要将数据封装到 retlw 指令中，可使用 retlw 指令代替该伪指令（见 4.16 Dt 伪指令中的示例）。

可以使用 data psect 保存定义的值。包含<xc.inc>后，此 psect 即已预定义。例如：

```
PSECT data
symbols:
  DB 76h
  DB 'A', -23
```

另外，也可以定义自己的 psect 并将其分配到程序存储器。确保 psect 的 space 标志设置为 0（默认值）。可通过将该 psect 与适当的链接器类（例如，PIC18 器件为 CONST，其他器件为 STRCODE）相关联，或者使用链接器的-P 选项（可使用-Wl 选项从驱动程序访问）明确定位该 psect，如以下 PIC18 示例所示。

```
PSECT romData,space=0,class=CONST
symbols:
  DB 76h
  DB 'A', -23
```

4.14 De 伪指令

MPASM DE 伪指令将字置于 EEPROM 中。

建议的替代伪指令
PIC 汇编器在适当的 psect 内部的 DW 伪指令可执行类似的功能。

可以使用 edata psect 保存定义的值。包含<xc.inc>后，此 psect 即已预定义。例如：

```
PSECT edata
  DW 9700h
  DW 'r', -48
```

另外，也可以定义自己的 psect 并将其分配到 EEPROM。 对于 PIC18 器件，确保 psect 的 space 标志设置为 0（默认值），对于任何其他支持 EEPROM 的器件，确保该标志设置为 3。可通过将该 psect 与 EEDATA 链接器类相关联，或者使用链接器的-P 选项（可使用-Wl 选项从驱动程序访问）明确定位该 psect，如以下中档器件示例所示。

```
PSECT eepromData,space=3,class=CONST
symbols:
  DW 76h
  DW 'A', -23
```

4.15 #define 伪指令

MPASM #DEFINE 伪指令用于定义宏的替代文本。

建议的替代伪指令
由于可以对汇编源文件进行预处理，因此可使用#define 预处理器伪指令（区分大小写）直接替代此汇编器伪指令，如以下示例所示。

与宏替代关联的所有常规预处理器功能均可用。确保汇编源文件的扩展名为.S，以便由汇编器对该文件进行预处理。

```
#define SUMSP(a, b) (a+2*b)
PSECT code
process:
  movlw SUMSP(5, 3)
  movwf volume
  ...
```

4.16 Dt 伪指令

MPASM DT 伪指令用于创建 retlw 指令表。

建议的替代伪指令
PIC 汇编器在适当的 psect 内部的 IRP 伪指令可执行类似的任务。

IRP 伪指令的工作方式与宏类似。由 ENDM 标记终止的代码块将针对参数名称之后指定的每个值输出一次。定义中出现的任何名称参数都会替换为正在处理的值。

可以使用 data psect 保存定义的值。包含<xc.inc>后，此 psect 即已预定义。例如：

```
PSECT data
symbols:
IRP number,48,65h,6Fh
    retlw number
ENDM
```

上述代码可展开为：

```
PSECT data
symbols:
  retlw 48
  retlw 65h
  retlw 6Fh
```

另外，也可以定义自己的 psect 并将其分配到程序存储器。确保 psect 的 space 标志设置为 0（默认值）。可通过将该 psect 与适当的链接器类（例如，PIC18 器件为 CONST，其他器件为 STRCODE）相关联，或者使用链接器的-P 选项（可使用-Wl 选项从驱动程序访问）明确定位该 psect，如以下 PIC18 示例所示。

```
PSECT romData,space=0,class=CONST
symbols:
IRP number,48,65h,6Fh
    retlw number
ENDM
```

4.17 Dtm 伪指令

MPASM DTM 伪指令用于创建 movlw 指令表。

建议的替代伪指令
PIC 汇编器在适当的 psect 内部的 IRP 伪指令可执行类似的任务。

有关更多信息和示例，请参见 4.16 Dt 伪指令。

4.18 Dw 伪指令

MPASM DW 伪指令将字置于程序存储器中。

建议的替代伪指令
PIC 汇编器在适当的 psect 内部的 DW 伪指令可执行类似的功能，但操作上存在一些差异。

该伪指令将其操作数的值以 16 位字为单位置于当前 psect 中。对于 PIC18 器件，每个操作数将占用两个地址（字节）。对于其他器件，汇编器会尝试将整个操作数值置于一个程序存储器字中，但是这些器件的程序存储器的宽度小于 2 字节，因此该值可能会被截断。通常，将使用 retlw 指令将数据存储到这些器件的程序存储器中，此指令会将每个字节存储到单独的存储单元中。

该伪指令不支持字符串立即数操作数，但可以使用字符和多个逗号分隔的操作数，如下面的示例所示。

可以使用 data psect 保存定义的值。包含<xc.inc>后，此 psect 即已预定义。例如：

```
PSECT data
modifiers:
  DW 1354h
  DW 's', -23
```

另外，也可以定义自己的 psect 并将其分配到程序存储器。确保 psect 的 space 标志设置为 0（默认值）。可通过将该 psect 与适当的链接器类（例如，PIC18 器件为 CONST，其他器件为 STRCODE）相关联，或者使用链接器的-P 选项（可使用-Wl 选项从驱动程序访问）明确定位该 psect，如以下 PIC18 示例所示。

```
PSECT myData,space=0,class=CONST
modifiers:
  DW 1354h
  DW 's', -23
```

4.19 Else 伪指令

MPASM ELSE 伪指令提供了备用汇编代码块，在 IF 伪指令条件求值为 false 时进行编译。

建议的替代伪指令
PIC 汇编器的 ELSE 伪指令可以直接替代此伪指令。

有关示例，请参见 4.34 If 伪指令。

4.20 End 伪指令

MPASM END 伪指令指示程序源代码结束。

建议的替代伪指令
PIC 汇编器的 END 伪指令可执行相同的任务。

END 伪指令的使用是可选项。一旦遇到使用该伪指令的情况，汇编器就将假定其后没有输入行，甚至 END 伪指令后的空行也会触发错误。

应将程序的开始标号指定为此伪指令的参数，以避免汇编器发出警告。

```
...
 return
 END  startMain ;the end of the program, folks
```

4.21 Endc 伪指令

MPASM ENDC 伪指令用于终止 CBLOCK 伪指令。

建议的替代伪指令
此伪指令没有直接替代伪指令。

4.22 Endm 伪指令

MPASM ENDM 伪指令用于终止宏定义。

建议的替代伪指令
PIC 汇编器的 ENDM 伪指令可以直接替代此伪指令。

有关示例，请参见 4.40 Macro 伪指令。

4.23 Endw 伪指令

MPASM ENDW 伪指令用于终止 WHILE 伪指令。

建议的替代伪指令
此伪指令没有直接替代伪指令。有关替代选项，请参见 4.62 While 伪指令。

4.24 Equ 伪指令

MPASM EQU 伪指令使值与符号相等。

建议的替代伪指令
PIC 汇编器的 EQU 伪指令可以直接替代此伪指令。

不得预先定义与 EQU 一起使用的符号。

4.25 Error 伪指令

MPASM ERROR 伪指令产生用户定义的错误消息。

建议的替代伪指令
PIC 汇编器的 ERROR 伪指令可以直接替代此伪指令。

4.26 Errorlevel 伪指令

MPASM ERRORLEVEL 伪指令控制要打印哪些汇编器消息。

建议的替代伪指令
-w 驱动程序选项可用于禁止汇编器产生的所有警告消息。另外，可考虑使用-mwarn 选项将警告限制在指定的严重程度。

4.27 Exitm 伪指令

MPASM EXITM 伪指令强制提前退出宏。

建议的替代伪指令
此伪指令没有替代伪指令。

4.28 Expand 伪指令

MPASM EXPAND 伪指令请求在列表文件中扩展汇编器宏。

建议的替代伪指令
PIC 汇编器的 EXPAND 伪指令可以直接替代此伪指令。

4.29 Extern 伪指令

MPASM EXTERN 伪指令将符号与在另一个模块中全局定义的符号链接在一起。

建议的替代伪指令
PIC 汇编器的 EXTRN 伪指令（注意拼写上的细微差别）可以直接替代此伪指令。

如果将此伪指令与当前模块中定义的符号一起使用，则会产生错误。

4.30 Fill 伪指令

MPASM FILL 伪指令用指定的值填充存储器。

建议的替代伪指令
PIC 汇编器的--fill 驱动程序选项可用于执行类似的任务。有关如何使用此功能的详细信息，请参见 *MPLAB® XC8 PIC Assembler User's Guide*。

4.31 Global 伪指令

MPASM GLOBAL 伪指令声明可由其他模块中的代码共用的符号。

建议的替代伪指令
PIC 汇编器的 GLOBAL 伪指令可以直接替代此伪指令。

在其他模块中，可以使用 GLOBAL 或 EXTRN 链接由此伪指令声明的符号。

4.32 Idata 伪指令

MPASM IDATA 伪指令为必须初始化的对象创建段。

建议的替代伪指令
使用 PIC 汇编器中具有以下功能的伪指令：可以在适当的数据存储器 psect 中保留空间，并将初始值整理到程序存储器内的独立 psect 中。

以下 PIC18 示例为存储区 1 中的变量保留了运行时存储器，并将这些变量的初始值置于程序存储器中。在本示例中使用了汇编器提供的 psect，但如有需要，可以自行定义适当的 psect。

```
#include <xc.inc>
;define space for the variables in RAM
PSECT udata_bank1
vars:
input:
  DS 2
output:
  DS 1

;place the initial values for the above in a matching order in program memory
PSECT data
iValues:
  DW 55AAh
  DB 67h
```

应用程序需要提供将初始值复制到保留数据存储空间的代码，如以下示例所示，将 iValues 的值复制到 vars 中。

```
PSECT text,class=CODE
copy0:
  movlw   low (iValues)
  movwf   tblptrl
  movlw   high(iValues)
  movwf   tblptrh
  movlw   low highword(iValues)
  movwf   tblptru
  tblrd*+
  movff   tablat, vars+0
  tblrd*+
  movff   tablat, vars+1
  tblrd*+
  movff   tablat, vars+2
  return
```

当复制大量数据时，考虑使用循环和 FSR 寄存器间接写入数据存储器中的变量。

4.33 Idata_acs 伪指令

MPASM IDATA_ACS 伪指令为必须初始化的 PIC18 快速操作存储区对象创建段。

建议的替代伪指令

使用 PIC 汇编器中具有以下功能的伪指令：可以在适当的数据存储器 psect 中保留空间，并将初始值整理到程序存储器内的独立 psect 中。

以下 PIC18 示例为快速操作存储区中的变量保留了运行时存储器，并将这些变量的初始值置于程序存储器中。在本示例中使用了汇编器提供的 psect，但如有需要，可以自行定义适当的 psect。

```
#include <xc.inc>
;define space for the variables in RAM
PSECT udata_acs
vars:
input:
  DS 2
output:
  DS 1

;place the initial values for the above in a matching order in program memory
PSECT data
iValues:
  DW 55AAh
  DB 67h
```

应用程序需要提供将初始值复制到保留数据存储空间的代码，如以下示例所示，将 iValues 的值复制到 vars 中。

```
PSECT text,class=CODE
copy0:
  movlw   low (iValues)
  movwf   tblptrl
  movlw   high(iValues)
  movwf   tblptrh
  movlw   low highword(iValues)
  movwf   tblptru
  tblrd*+
  movff   tablat, vars+0
  tblrd*+
  movff   tablat, vars+1
  tblrd*+
  movff   tablat, vars+2
  return
```

当复制大量数据时，考虑使用循环和 FSR 寄存器间接写入数据存储器中的变量。

4.34 If 伪指令

MPASM IF 伪指令开始条件汇编代码块。

建议的替代伪指令
PIC 汇编器的 IF 伪指令可以直接替代此伪指令。

操作数必须为绝对表达式，如果为非值零，则将汇编 IF 后的代码，直至遇到下一个匹配的 ELSE、ELSIF 或 ENDIF。如果操作数为零，则不会输出下一个匹配的 ELSE 或 ENDIF 之前的代码。在 ELSE 处，将反向解释条件编译的含义，而 ENDIF 将终止条件汇编块。

下面给出了用于在输出中包含两个调用之一的伪指令。

```
IF DEMO
  call demo_mode
ELSE
  call play_mode
ENDIF
```

对于汇编代码，无论条件为 true 还是 false，都会始终进行扫描和解析，但对于指令所对应的机器码，仅当条件匹配时才会将其输出。这意味着，无论条件表达式的状态如何，都将处理汇编器伪指令（例如 EQU），因此不应在 IF - ENDIF 结构内部使用这些伪指令。

尽管 MPASM 汇编器允许使用此伪指令的#IF 形式，但请注意，它仍然是一个汇编器伪指令。PIC 汇编器的#if 伪指令是一个*预处理器*伪指令，因此它将查看*预处理器*表达式（可能使用预处理器符号）而不是*汇编器*表达式（可能使用汇编器符号）的结果。如果移植代码以使用预处理器伪指令，请确保同时检查了涉及的表达式，并根据需要使用 #define 伪指令或-D 选项定义了等效预处理器宏。

4.35 Ifdef 伪指令

MPASM IFDEF 伪指令开始条件汇编代码块。

建议的替代伪指令
由于可以对汇编源文件进行预处理，因此可使用#ifdef 预处理器伪指令来替代此伪指令。确保汇编源文件的扩展名为.S，以便由汇编器对该文件进行预处理。例如：

```
#ifdef DBG
  movf state,w
  call diag
#endif
```

尽管 MPASM 汇编器允许使用此伪指令的#IFDEF 形式，但请注意，它仍然是一个汇编器伪指令。PIC 汇编器的 #ifdef 伪指令是一个*预处理器*伪指令，因此它将查看*预处理器*符号而不是*汇编器*符号的定义。如果移植代码以使用预处理器伪指令，请确保同时根据需要使用#define 伪指令或-D 选项定义了等效预处理器宏。

4.36 Ifndef 伪指令

MPASM IFNDEF 伪指令开始条件汇编代码块。

建议的替代伪指令
由于可以对汇编源文件进行预处理，因此可使用 PIC 汇编器的#ifndef 预处理器伪指令来替代此伪指令，如以下示例所示。确保汇编源文件的扩展名为.S，以便由汇编器对该文件进行预处理。例如：

```
#ifndef RUNMODE
  movf state,w
  call diag
#endif
```

尽管 MPASM 汇编器允许使用此伪指令的#IFNDEF 形式，但请注意，它仍然是一个汇编器伪指令。PIC 汇编器的 #ifndef 伪指令是一个*预处理器*伪指令，因此它将查看*预处理器*符号而不是*汇编器*符号的定义。如果移植代码以使用预处理器伪指令，请确保同时根据需要使用#define 伪指令或-D 选项定义了等效预处理器宏。

4.37 #include 伪指令

MPASM #include 伪指令以文本形式替换为指定的文件。

建议的替代伪指令

由于可以对汇编源文件进行预处理，因此可使用 PIC 汇编器的#include 预处理器伪指令直接替代此伪指令，如以下示例所示。

对于用尖括号括起来的文件名，将在标准头文件位置中搜索。而对于用引号括起来的文件名，将先在当前工作目录中搜索，然后在标准头文件位置中搜索。确保汇编源文件的扩展名为.S，以便由汇编器对该文件进行预处理。

```
#include <xc.inc>
#include "buttons.inc"
```

另外，也可以使用 PIC 汇编器的 INCLUDE "*file*”伪指令来替代此伪指令。不使用搜索路径。如果要包含的文件不在当前工作目录中，则必须使用伪指令来指定文件的完整路径。此伪指令不能用于包含带有预处理器伪指令的头文件，因此不能将其用于包含<xc.inc>。

```
INCLUDE "buttons.inc"
```

4.38 List 伪指令

MPASM LIST 伪指令使能和控制列表文件中的内容。

建议的替代伪指令

PIC 汇编器的 LIST *options* 伪指令执行此伪指令的许多功能，如表 4-2 所示。

表 4-2. 等效伪指令列表选项

MPASM 列表选项	用途	建议的 PIC 汇编器替代伪指令
b=*nnn*	设置制表符大小	没有可用替代伪指令
c=*nnn*	设置列宽	c=*nnn*
f=*format*	设置十六进制文件格式	使用-g 驱动程序选项
free	使用自由格式解析器	没有可用替代伪指令
fixed	使用固定格式解析器	没有可用替代伪指令
mm=[on|off]	打印列表中的存储器映射	使用-msummary 选项将存储器使用情况打印到控制台
n=*nnn*	设置每页行数	n=*nnn*
p=*device*	选择器件	p=*device*
pe=*type*	选择器件并使能 PIC18 扩展指令模式	将-mcpu 选项与-misa 选项结合使用
r=*radix*	设置源代码基数	使用 RADIX 汇编器伪指令
st=[on|off]	打印列表中的符号表	没有可用替代伪指令，但始终在列表中产生符号表
t=[on|off]	截断列表行	t=[on|off]

.....（续）		
MPASM 列表选项	**用途**	**建议的 PIC 汇编器替代伪指令**
w=[0|1|2]	设置消息级别	使用-w 选项禁止警告
x=[on|off]	使能宏扩展	x=[on|off]

4.39 Local 伪指令

MPASM LOCAL 伪指令在宏内部定义局部标号。

建议的替代伪指令
PIC 汇编器的 LOCAL 伪指令可以直接替代此伪指令。用逗号分隔多个标号名称

4.40 Macro 伪指令

MPASM MACRO 伪指令用于定义宏。

建议的替代伪指令
PIC 汇编器的 MACRO 伪指令可以直接替代此伪指令。

在宏定义内，&字符可用于将宏参数与其他文本相关联，但在实际扩展中会被删除。例如：

```
loadPort MACRO port, value
  movlw value
  movwf PORT&port
ENDM
```

如果在调用时 port 为 A，则将装入 PORTA，依此类推。宏中&标记的特殊含义表示必须仅使用 and 形式的按位与运算符。

通过使用两个分号;;作为注释的开头，可以在宏的扩展中禁止注释。

在调用某个宏时，参数列表必须用逗号分隔。如果希望在参数中包含逗号（或其他分隔符，如空格），可以使用尖括号<和>将其括起。

如果参数前面具有一个百分号%，则该参数将作为表达式进行求值，并以十进制数字而不是字符串的形式传递。如果宏主体内的参数求值会产生不同的结果，这将非常有用。

在宏中可以使用 nul 操作符来测试宏参数，例如：

```
IF nul     arg3  ;argument was not supplied.
...
ELSE             ;argument was supplied
...
ENDIF
```

4.41 Maxram 和 Maxrom 伪指令

MPASM __maxram 和__maxrom 伪指令与__badram 和__badrom 伪指令一起指定文件寄存器地址范围，如果代码使用这些地址范围，则应将其标记为无效。

建议的替代伪指令
这些伪指令没有替代伪指令。

有关更多信息，请参见 4.2 Badram 和 Badrom 伪指令。

4.42 Messg 伪指令

MPASM MESSG 伪指令生成用户定义的建议性消息。

建议的替代伪指令
PIC 汇编器的 MESSG 伪指令可以直接替代此伪指令。

4.43 Noexpand 伪指令

MPASM NOEXPAND 伪指令请求不在列表文件中扩展汇编器宏。

建议的替代伪指令
PIC 汇编器的 NOEXPAND 伪指令可以直接替代此伪指令。

4.44 Nolist 伪指令

MPASM NOLIST 伪指令使能和控制列表文件中的内容。

建议的替代伪指令
PIC 汇编器的 NOLIST 伪指令可以直接替代此伪指令。

4.45 Org 伪指令

MPASM ORG 伪指令将后续代码的初始位置计数器设置为指定的地址。

建议的替代伪指令
PIC 汇编器的 ORG 伪指令可执行与此伪指令类似的任务，但操作上存在一些差异。

ORG 伪指令将*当前 psect 内*位置计数器的值更改为指定的值。这意味着由 ORG 伪指令设置的地址将相对于 psect 的基址，通常直到链接时才会确定该基址。例如，在最终链接到地址 0x2000 的 psect 内使用 ORG 0100h 时，会将位置计数器移动到地址 0x2100。只有放置该伪指令的 psect 是绝对（使用 abs 标志）且重叠（ovrld 标志）的 psect 时，才会将位置计数器移动到指定的绝对地址处。

在程序中很少需要使用 ORG 伪指令。要使代码或数据位于特定地址，请将其放置在惟一的 psect 中，并使链接器将该 psect 置于所需位置。

4.46 Page 伪指令

MPASM PAGE 伪指令将页弹出操作插入到列表文件。

建议的替代伪指令
此伪指令没有直接替代伪指令。

4.47 Pagesel 伪指令

MPASM PAGESEL 伪指令生成适用于指定为参数的标号的页选择代码。

建议的替代伪指令
PIC 汇编器的 PAGESEL 伪指令可以直接替代此伪指令。

4.48 Pageselw 伪指令

MPASM PAGESELW 伪指令生成适用于指定为参数的标号的页选择代码（使用 WREG 作为中间寄存器）。

建议的替代伪指令

PIC 汇编器的 PAGESEL 伪指令可执行类似的功能，但不会使用 WREG 来调整页选择位。输出页选择代码将对必须翻转的位执行位操作。

4.49 Processor 伪指令

MPASM PROCESSOR 伪指令设置器件类型。

建议的替代伪指令

PIC 汇编器的 PROCESSOR 伪指令可以直接替代此伪指令。

必须使用-mcpu 驱动程序选项来选择器件，但可以使用此伪指令来确保为目标器件编译了源代码。如果由此伪指令指定的器件与选项设置的器件发生冲突，将产生错误。

4.50 Radix 伪指令

MPASM RADIX 伪指令为在汇编源文件中指定的常量设置基数。

建议的替代伪指令

PIC 汇编器的 RADIX 伪指令可以直接替代此伪指令。

4.51 Res 伪指令

MPASM RES 伪指令通过递增位置计数器来保留存储器。

建议的替代伪指令

PIC 汇编器的 DS 伪指令具有类似的功能，但操作上存在一些差异。

DS 伪指令递增位置计数器，从而允许将存储器分配给在伪指令之前定义的标号。通常，此伪指令在链接到数据空间的 psect 内部使用，可提供一种机制来为变量保留存储器，如以下示例所示。

```
PSECT udata_bank0
input:
  DS 2  ;allocate 2 bytes for input
output:
  DS 4  ;allocate 4 bytes for output
```

如果在分配到程序存储器的 psect 中使用此伪指令（space=0 标志），它将会移动位置计数器，但不会在 HEX 文件输出中放入任何内容。

此伪指令的地址单元由上下文确定，尤其是与其所在 psect 相关联的 delta 和 bit 标志值。如果 delta 标志设置为 1，则此伪指令会将位置计数器向前移动指定的字节数；如果设置为 2，则会将位置计数器向前移动指定的 16 位字数，依此类推。当使用 bit 标志时，地址分配单元为位。

4.52 Set 伪指令

MPASM SET 伪指令使值等于符号。

建议的替代伪指令

PIC 汇编器的 SET 伪指令可以直接替代此伪指令。

可以重新定义该符号，而不会出错。

4.53 Space 伪指令

MPASM SPACE 伪指令在汇编列表文件中插入空行。

建议的替代伪指令
PIC 汇编器的 SPACE 伪指令可以直接替代此伪指令。

4.54 Subtitle 伪指令

MPASM SUBTITLE 伪指令指定列表文件副标题字符串。

建议的替代伪指令
PIC 汇编器的 SUBTITLE 伪指令可以直接替代此伪指令。

4.55 Title 伪指令

MPASM TITLE 伪指令指定列表文件标题字符串。

建议的替代伪指令
PIC 汇编器的 TITLE 伪指令可以直接替代此伪指令。

4.56 Udata 伪指令

MPASM UDATA 伪指令为未初始化的对象创建段。

建议的替代伪指令
使用预定义的 udata_bank*n* psect（其中 *n* 表示 psect 应处于的存储区编号）或创建类似的 psect，确保标志适用于目标器件上包含变量的段。

包含<xc.inc>后，PIC 汇编器将提供 udata_bank*n* psect。使用该 psect 时不必指定任何 psect 标志，例如：

```
#include <xc.inc>

PSECT udata_bank1
;data goes here
```

或者，也可以使用任何名称和适当的 psect 标志定义自己的 psect。 psect 的 space 标志必须为 1，指示 psect 应位于数据存储器中。通常，使用 class 标志将 psect 分配给 BANK*N* 链接器类之一，其中 *N* 表示类定义的存储区编号。这些类也是由驱动程序预定义的，因此 psect 将位于与指定类关联的存储器中的某个位置，而无需指定任何链接器选项。此外，也可使用链接器的-p 选项（通过驱动程序的-Wl 选项传递给链接器）将该 psect 置于特定地址。例如：

```
PSECT myData,space=1,class=BANK1
;data goes here
```

4.57 Udata_acs 伪指令

对于 PIC18 器件，MPASM UDATA_ACS 伪指令为未初始化的快速操作存储区对象创建段。

建议的替代伪指令
使用预定义的 udata_acs psect 或创建类似的 psect，确保标志适用于 PIC18 器件上包含快速操作存储区变量的段。

包含<xc.inc>后，PIC 汇编器将提供 udata_acs psect。使用该 psect 时不必指定任何 psect 标志，例如：

```
#include <xc.inc>

PSECT udata_acs
;data goes here
```

或者，也可以使用任何名称和适当的 psect 标志定义自己的 psect。psect 的 space 标志必须为 1，指示 psect 应位于数据存储器中。通常，将使用 class 标志将 psect 分配给 COMRAM 链接器类之一（该类也是由驱动程序预定义的），因此 psect 将位于与指定类关联的存储器中的某个位置，而无需指定任何链接器选项。此外，也可使用链接器的-p 选项（通过驱动程序的-Wl 选项传递给链接器）将该 psect 置于特定地址。例如：

```
PSECT myData,space=1,class=COMRAM
;data goes here
```

4.58 Udata_ovr 伪指令

对于低档和中档器件，MPASM UDATA_OVR 伪指令会为未初始化的对象创建重叠段。

建议的替代伪指令

定义 ovrld 标志置 1 的 psect。将 psect 与快速操作存储区链接器类 COMMON 相关联，以使其链接到低档或中档器件上公共存储器中的某个位置。

以下示例给出了放置在 myData psect 中的对象，该 psect 在两个不同的模块（文件 1 和文件 2）中使用。两个模块中的 PSECT 伪指令使用相同的 psect 名称、ovrld 标志（用于指示 psect 将重叠）和 space=1 标志（用于指示 psect 将位于数据存储空间中）。psect 与 COMMON 链接器类相关联，该类定义了公共存储器，因此如果 psect 重叠，myData 将出现在该类定义的存储器中的任何位置。

```
;file 1
PSECT myData,space=1,ovrld,class=COMMON
zero:
  DS 1
  ;leave a 1-byte gap for another object here
  ORG 2
two:
  DS 1

;file 2
PSECT myData,space=1,ovrld,class=COMMON
  ORG 1
one:
  DS 1
```

请注意，每个模块中放置在重叠 psect 中的对象串连在一起，但每个模块的 psect 随后在链接时重叠。上述示例编译完成后，标号将按 zero、one、two 的顺序出现在存储器中。

上述示例中的 ORG 伪指令允许 psect 的内容交错。如果文件 1 中未使用该伪指令，则与标号 zero 和标号 one 关联的空间将重叠，并且这些对象将出现在相同的地址上。此类代码是合法的，可能为某些应用所需。

重叠的 psect 可以链接到任何 RAM 区域，而不仅仅是公共存储器。只要选择合适的链接器类，此结构便可适用于任何器件。

4.59 Udata_shr 伪指令

在低档和中档器件上，MPASM UDATA_SHR 伪指令会为未初始化的未分区对象创建段。

建议的替代伪指令

使用预定义的 udata_shr psect 或创建类似的 psect，确保标志适用于低档或中档器件上包含变量的段。

包含<xc.inc>后，PIC 汇编器将提供 udata_shr psect。使用该 psect 时不必指定任何 psect 标志，例如：

```
#include <xc.inc>

PSECT udata_shr
;data goes here
```

或者，也可以使用任何名称和适当的 psect 标志定义自己的 psect。psect 的 space 标志必须为 1，指示 psect 应位于数据存储器中。通常，将使用 class 标志将 psect 分配给 COMMON 链接器类之一（该类也是由驱动程序预定义的），因此 psect 将位于与指定类关联的存储器中的某个位置，而无需指定任何链接器选项。此外，也可使用链接器的-p 选项（通过驱动程序的-Wl 选项传递给链接器）将该 psect 置于特定地址。例如：

```
PSECT myData,space=1,class=COMMON
;data goes here
```

4.60 #define 伪指令

MPASM #define 伪指令用于定义宏的替代文本。

建议的替代伪指令
由于可以对汇编源文件进行预处理，因此可使用 PIC 汇编器的#define 预处理器伪指令直接替代此伪指令，如以下示例所示。此伪指令提供与宏替换关联的所有常规预处理器功能。

确保汇编源文件的扩展名为.S，以便由汇编器对该文件进行预处理。

```
#define SUMSP(a, b) (a+2*b)
PSECT text,class=CODE
process:
  movlw SUMSP(5, 3)
  movwf volume
```

4.61 Variable 伪指令

MPASM VARIABLE 伪指令创建一个设置为某个值的变量。

建议的替代伪指令
此伪指令没有替代伪指令，但 PIC 汇编器的 SET 伪指令可执行类似的任务。

SET 伪指令将值与符号相关联。尽管可以使用其他 SET 伪指令重新定义符号的值而不出错，但无法通过其他方式对符号的值执行操作。

4.62 While 伪指令

MPASM WHILE 伪指令在某些条件为 true 时对汇编代码块执行循环。

建议的替代伪指令
此伪指令没有替代伪指令，但 PIC 汇编器的 REPT 伪指令可执行类似的任务。

根据伪指令的参数，REPT 后由 ENDM 终止的代码块将输出多次。

以下示例显示了将执行 4 次的移位指令。

```
REPT 4
  rlncf mask,f
ENDM
```

5. 链接

使用 PIC 汇编器编译项目时，将始终调用链接器，除非通过-c 选项提前停止编译。必须执行链接器以获得可下载到硬件的最终程序映像。

链接器由其自带选项集进行控制。这些选项包括用于指定器件的存储器排列的选项。不使用也不能使用链接器脚本。

无需显式运行链接器来指定或更改链接器选项。一些 pic-as 驱动程序选项间接控制链接器，而-Wl,驱动程序选项可用于将链接器选项直接传递给链接器。该选项还允许改写由驱动程序发送给链接器的某些默认选项。

5.1 保留存储器

通过使用-mreserve 驱动程序选项，可以禁止代码和数据使用存储器。此外，也可以使用-mram 和-mrom 选项来实现相同的目的。例如-mreserve=ram@100:103 将删除数据存储器中的范围 100-103h。也可以使用-mram=default,-100-103。要保留程序存储器，可以使用-mreserve=rom@1800-1fff 或-mrom=default,-1800-1fff。

保留存储器时，会将其从所有包含该存储器范围的链接器类中删除，因此，置于这些类中的任何 psect 都不会使用该存储器。存储器保留不会影响已链接到绝对地址的任何 psect，也不会影响到相对于其他 psect 放置的 psect。要移动这些 psect，必须更改将其置于该位置的链接器选项。

5.2 将 Psect 放入存储器

所有代码和对象必须放在一个 psect（程序段）中，这只是一种在 psect 名称下将程序的各个部分组合在一起的方法。链接器会为所有 psect 分配存储器，之后，便可以确定在这些 psect 中定义的任何标号的值。

将 psect 放入存储器时，链接器将执行以下第一个符合条件的操作。

- 如果 psect 指定 abs 标志，则将其置于由 psect 的 space 标志指示的存储空间中的地址 0 处，或者在尚未指定空间的情况下将其置于程序存储器（默认）空间中。
- 如果-p 链接器选项引用 psect 名称，则将 psect 置于由 psect 的 space 标志指示的存储空间中该选项指定的位置，或者在尚未指定空间的情况下将其置于程序存储器（默认）空间中。
- 如果 psect 与链接器类相关联，则将 psect 置于由该类定义的地址范围内的任何空闲位置。
- 如果 psect 指定了空间编号，则将其置于该存储空间中的空闲位置（不建议）。
- 将 psect 置于程序存储器（默认）空间中的空闲位置（不建议）。

某些情况是非法的，例如，如果使用-p 选项来放置也使用 abs 标志的 psect，则会产生错误。建议始终使用 abs 标志、使用-p 选项或通过链接器类来链接 psect（上述方法的前三种）。如果链接器必须在没有用户指导的情况下放置 psect（上述方法的最后两种），则会产生类似于(526) psect "wanderer" not specified in -P option (first appears in "not_right.o")的警告。

在大多数情况下，psect 可以链接到器件指定的合适地址范围内的任何位置。例如，大多数可执行代码可以置于程序存储器中的任何位置，或者至少置于程序存储器页中的任何位置。数据对象通常可以置于数据存储区中的任何位置。在这种情况下，链接这些 psect 的最简单方式是将它们与链接器类相关联。如果使用 PIC 汇编器提供的 psect，则其已与适当的链接器类相关联，无需指定任何链接器选项即可将其正确链接。在以下示例中，

```
PSECT udata_bank1
myVar:
  DS 2
```

udata psect 已经与 RAM 链接器类相关联，并将链接到与该类关联的空闲存储器中的任何位置。

如果已创建自己的 psect，则可以使用具有 psect 定义的 class 标志将其与 PIC 汇编器提供的任何现有链接器类相关联。在以下示例中，程序员已创建了 psect 来代替 udata 进行使用。

```
PSECT machData,space=1,class=MDATA
myVar:
  DS 2
```

此 psect 使用了新类 MDATA，该类将需要通过链接器选项进行定义。为此，可使用-Wl,-AMDATA=050h-05fh 等驱动程序选项，以便将-A 链接器选项直接传递给链接器，并将指定的地址范围与 MDATA 类相关联。

不过，有时候必须将 psect 置于指定的地址处。复位向量代码便是一个很好的示例，中断程序也是。在这种情况下，将需要使用-p 链接器选项将 psect 置于所需的位置。此过程与提供绝对地址一样简单，例如，使用驱动程序选项-Wl,-pInterrupt=08h 将调用 Interrupt 的 psect 置于地址 8 处，但此选项有更高级的用法。

有关 PIC 汇编器提供的 psect 和链接器类以及本节提到的链接器和驱动程序选项的完整详细信息，请参见 *MPLAB® XC8 PIC Assembler User's Guide*。

Microchip 网站

Microchip 网站（www.microchip.com/）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。我们的网站提供以下内容：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题解答（FAQ）、技术支持请求、在线讨论组以及 Microchip 设计伙伴计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

产品变更通知服务

Microchip 的产品变更通知服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请访问 www.microchip.com/pcn，然后按照注册说明进行操作。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（ESE）
- 技术支持

客户应联系其代理商、代表或 ESE 寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 www.microchip.com/support 获得网上技术支持。

Microchip 器件代码保护功能

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下，Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿意与关心代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。

代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案（Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

法律声明

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc.及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc.的英文原版文档。

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担

保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和/或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。除非另外声明，否则在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

商标

Microchip 的名称和徽标组合、Microchip 徽标、Adaptec、AnyRate、AVR、AVR 徽标、AVR Freaks、BesTime、BitCloud、chipKIT、chipKIT 徽标、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、HELDO、IGLOO、JukeBlox、KeeLoq、Kleer、LANCheck、LinkMD、maXStylus、maXTouch、MediaLB、megaAVR、Microsemi、Microsemi 徽标、MOST、MOST 徽标、MPLAB、OptoLyzer、PackeTime、PIC、picoPower、PICSTART、PIC32 徽标、PolarFire、Prochip Designer、QTouch、SAM-BA、SenGenuity、SpyNIC、SST、SST 徽标、SuperFlash、Symmetricom、SyncServer、Tachyon、TempTrackr、TimeSource、tinyAVR、UNI/O、Vectron 和 XMEGA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的注册商标。

APT、ClockWorks、The Embedded Control Solutions Company、EtherSynch、FlashTec、Hyper Speed Control、HyperLight Load、IntelliMOS、Libero、motorBench、mTouch、Powermite 3、Precision Edge、ProASIC、ProASIC Plus、ProASIC Plus 徽标、Quiet-Wire、SmartFusion、SyncWorld、Temux、TimeCesium、TimeHub、TimePictra、TimeProvider、Vite、WinPath 和 ZL 均为 Microchip Technology Incorporated 在美国的注册商标。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BlueSky、BodyCom、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、INICnet、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNet 徽标、memBrain、Mindi、MiWi、MPASM、MPF、MPLAB Certified 徽标、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、Ripple Blocker、SAM-ICE、Serial Quad I/O、SMART-I.S.、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、ViewSpan、WiperLock、Wireless DNA 和 ZENA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的商标。

SQTP 是 Microchip Technology Incorporated 在美国的服务标记。

Adaptec 徽标、Frequency on Demand、Silicon Storage Technology 和 Symmcom 为 Microchip Technology Inc.在其他国家或地区的注册商标。

GestIC 是 Microchip Technology Inc.的子公司 Microchip Technology Germany II GmbH & Co. KG 在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。

ISBN： 978-1-5224-6007-7

质量管理体系

有关 Microchip 的质量管理体系的信息，请访问 www.microchip.com/quality。

全球销售及服务网点

美洲	亚太地区	亚太地区	欧洲
公司总部 2355 West Chandler Blvd. Chandler, AZ 85224-6199 电话：480-792-7200 传真：480-792-7277 技术支持： www.microchip.com/support 网址： www.microchip.com **亚特兰大** 德卢斯，佐治亚州 电话：678-957-9614 传真：678-957-1455 **奥斯汀，德克萨斯州** 电话：512-257-3370 **波士顿** 韦斯特伯鲁，马萨诸塞州 电话：774-760-0087 传真：774-760-0088 **芝加哥** 艾塔斯卡，伊利诺伊州 电话：630-285-0071 传真：630-285-0075 **达拉斯** 阿迪森，德克萨斯州 电话：972-818-7423 传真：972-818-2924 **底特律** 诺维，密歇根州 电话：248-848-4000 **休斯顿，德克萨斯州** 电话：281-894-5983 **印第安纳波利斯** 诺布尔斯维尔，印第安纳州 电话：317-773-8323 传真：317-773-5453 电话：317-536-2380 **洛杉矶** 米慎维荷，加利福尼亚州 电话：949-462-9523 传真：949-462-9608 电话：951-273-7800 **罗利，北卡罗来纳州** 电话：919-844-7510 **纽约，纽约州** 电话：631-435-6000 **圣何塞，加利福尼亚州** 电话：408-735-9110 电话：408-436-4270 **加拿大 - 多伦多** 电话：905-695-1980 传真：905-695-2078	**澳大利亚 - 悉尼** 电话：61-2-9868-6733 **中国 - 北京** 电话：86-10-8569-7000 **中国 - 成都** 电话：86-28-8665-5511 **中国 - 重庆** 电话：86-23-8980-9588 **中国 - 东莞** 电话：86-769-8702-9880 **中国 - 广州** 电话：86-20-8755-8029 **中国 - 杭州** 电话：86-571-8792-8115 **中国 - 香港特别行政区** 电话：852-2943-5100 **中国 - 南京** 电话：86-25-8473-2460 **中国 - 青岛** 电话：86-532-8502-7355 **中国 - 上海** 电话：86-21-3326-8000 **中国 - 沈阳** 电话：86-24-2334-2829 **中国 - 深圳** 电话：86-755-8864-2200 **中国 - 苏州** 电话：86-186-6233-1526 **中国 - 武汉** 电话：86-27-5980-5300 **中国 - 西安** 电话：86-29-8833-7252 **中国 - 厦门** 电话：86-592-2388138 **中国 - 珠海** 电话：86-756-3210040	**印度 - 班加罗尔** 电话：91-80-3090-4444 **印度 - 新德里** 电话：91-11-4160-8631 **印度 - 浦那** 电话：91-20-4121-0141 **日本 - 大阪** 电话：81-6-6152-7160 **日本 - 东京** 电话：81-3-6880- 3770 **韩国 - 大邱** 电话：82-53-744-4301 **韩国 - 首尔** 电话：82-2-554-7200 **马来西亚 - 吉隆坡** 电话：60-3-7651-7906 **马来西亚 - 槟榔屿** 电话：60-4-227-8870 **菲律宾 - 马尼拉** 电话：63-2-634-9065 **新加坡** 电话：65-6334-8870 **台湾地区 - 新竹** 电话：886-3-577-8366 **台湾地区 - 高雄** 电话：886-7-213-7830 **台湾地区 - 台北** 电话：886-2-2508-8600 **泰国 - 曼谷** 电话：66-2-694-1351 **越南 - 胡志明市** 电话：84-28-5448-2100	**奥地利 - 韦尔斯** 电话：43-7242-2244-39 传真：43-7242-2244-393 **丹麦 - 哥本哈根** 电话：45-4485-5910 传真：45-4485-2829 **芬兰 - 埃斯波** 电话：358-9-4520-820 **法国 - 巴黎** 电话：33-1-69-53-63-20 传真：33-1-69-30-90-79 **德国 - 加兴** 电话：49-8931-9700 **德国 - 哈恩** 电话：49-2129-3766400 **德国 - 海尔布隆** 电话：49-7131-72400 **德国 - 卡尔斯鲁厄** 电话：49-721-625370 **德国 - 慕尼黑** 电话：49-89-627-144-0 传真：49-89-627-144-44 **德国 - 罗森海姆** 电话：49-8031-354-560 **以色列 - 若那那市** 电话：972-9-744-7705 **意大利 - 米兰** 电话：39-0331-742611 传真：39-0331-466781 **意大利 - 帕多瓦** 电话：39-049-7625286 **荷兰 - 德卢内市** 电话：31-416-690399 传真：31-416-690340 **挪威 - 特隆赫姆** 电话：47-72884388 **波兰 - 华沙** 电话：48-22-3325737 **罗马尼亚 - 布加勒斯特** 电话：40-21-407-87-50 **西班牙 - 马德里** 电话：34-91-708-08-90 传真：34-91-708-08-91 **瑞典 - 哥德堡** 电话：46-31-704-60-40 **瑞典 - 斯德哥尔摩** 电话：46-8-5090-4654 **英国 - 沃金厄姆** 电话：44-118-921-5800 传真：44-118-921-5820