MPLAB® CXX Quick Reference Card

MICROCHIP

The Embedded Control Solutions Company®

MPLAB C17 Quick Reference

MPLAB C17 Command Switches

Command	Description
/?, /h	Display help screen
/D<macro>[=<text>]	Define a macro
/FO=<name>	Set object file name
/FE=<name>	Set error file name
/I<path>	Add include path
/NW<n>	Suppress message n
/O	Enable all optimizations
/Ob[+|-]	Branch optimization
/Oc[+|-]	Context optimization
/Ol[+|-]	Default static locals
/Or[+|-]	Register optimizer
/Ou[+|-]	Unreachable code removal
/Op	Far ram pointers are to GPRs
/P=<processor>	Set processor
/Q	Quiet mode
/W{1|2|3}	Set warning level
/M{s|m|c|l}	Select memory model

		RAM	ROM
s	small	near	near
m	medium	near	far
c	compact	far	near
l	large	far	far

MPLAB C17 Libraries and Precompiled Object Files

File	Use
cmath17.lib	Math routines
p17c???.o	SFR definitions
c0*17.o	Startup code
idata17.o	Initialized data support
int???*.o	Interrupt support
pmc???*.lib	Standard C and peripheral library routines

??? = processor type (e.g., 756 for PIC17C756)
* = memory model (i.e., s, c, m, l)

MPLAB C17 Types

Type	Bit Width	Range
void	N/A	none
char	8	-128 to 127
unsigned char	8	0 to 255
int	16	-32,768 to 32,767
unsigned int	16	0 to 65,535
short	16	-32,768 to 32,767
unsigned short	16	0 to 65,535

MPLAB C17 Types (Continued)

Type	Bit Width	Range
long	32	-2,147,483,648 to 2,147,483,647
unsigned long	32	0 to 4,294,967,295
float	32	1.7549435E-38 to 6.80564693E+38
double	32	1.7549435E-38 to 6.80564693E+38

Common MPLAB C17 Type Modifiers

Modifier	Use
auto	Variable exists only in block in which it was defined
const	Variable will not be modified
far	Variable is paged/banked regardless of memory model selected
extern	Variable is allocated in another module
near	Variable is not paged/banked regardless of memory model selected
static	Variable is retained unchanged between executions of the defining block

MPLAB C17 Interrupts

To create an interrupt service routine in your MPLAB C17 code, you may wish to use the following steps:

- Define interrupt routine in your source code using a `#pragma interrupt` statement.
- Specify which interrupt routine will be called for each type of interrupt used. Do this with the Install_ macros, replacing “isr” with the name of the ISR function:
  - `Install_INT(isr);`
  - `Install_TMR0(isr);`
  - `Install_T0CKI(isr);`
  - `Install_PIV(isr);`
- Include interrupt support routines (e.g., int756l.o) when invoking MPLINK™ object linker.

MPLAB C17 Inline Assembly

MPLAB C17 has an internal assembler with a syntax similar to the MPASM™ assembler, except that comments must be in the C (/* */) or C++ (//) style. The block of assembly code must begin with `_asm` and end with `_endasm`. For example:

```
_asm
movlw 7      // Load 7 into WREG
movwf PORTB  // and send it to PORTB
_endasm
```

Creating an MPLAB C17 Project in MPLAB IDE

The following are the basic steps required to create a MPLAB C17 based project in the MPLAB IDE. For a more detailed description, please see the MPLAB CXX User’s Guide (DS51217).

1. Specify the include, library, and linker script paths. The library path should be c:\mcc\lib, where c:\mcc is the installation directory for MPLAB C17.
2. Select the development mode (processor and debugging environment).
3. Select MPLINK object linker as the build tool for the target node.
4. Add C files using the **Add Node…** button, specifying the build tool for each as MPLAB C17.
5. Add a linker script file.
6. Add any needed libraries or precompiled object files.

Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224
Tel: 480.792.7200 Fax: 480.792.9210
Web Site Address: www.microchip.com

Printed in the U.S.A. 11/00 DS51225B

MPLAB C18 Quick Reference

MPLAB C18 Command Switches

Command	Description
`-?, -h`	Display help screen
-d<macro>[=<text>]	Define a macro
-fo=<name>	Set object file name
-fe=<name>	Set error file name
-i<path>	Add include path
-k	Default `char` is unsigned
-ls	Multi-bank stack
-nw<n>	Suppress message n
-O	Enable all optimizations
-Ob[+|-]	Branch optimization
-Oi[+|-]	Promote to integers
-Om[+|-]	Duplicate string merging
-On{0|1|2}	Set banking optimizer level
-Os[+|-]	Code straightening
-Ot[+|-]	Tail merging
-Ou[+|-]	Unreachable code removal
-p=<processor>	Set processor
-q	Quiet mode
-w{1|2|3}	Set warning level
-m{s|l}	Select memory model

		ROM
s	small	near
l	large	far

MPLAB C18 Libraries and Precompiled Object Files

File	Use
clib.lib	Standard C routines, math routines, startup code
c018i.o	Startup code with initialized data support
c018.o	Startup code without initialized data support
p18c???.lib	Peripheral library routines and SFR definitions

`???` = processor type (e.g., 452 for PIC18C452).

MPLAB C18 Types

Type	Bit Width	Range
void	N/A	none
char	8	-128 to 127
unsigned char	8	0 to 255
int	16	-32,768 to 32,767
unsigned int	16	0 to 65,535
short	16	-32,768 to 32,767
unsigned short	16	0 to 65,535
short long	24	-8,388,608 to 8,388,607
unsigned short long	24	0 to 16,777,215
long	32	-2,147,483,648 to 2,147,483,647
unsigned long	32	0 to 4,294,967,295
float	32	1.7549435E-38 to 6.80564693E+38
double	32	1.7549435E-38 to 6.80564693E+38

Common MPLAB C18 Type Modifiers

Modifier	Use
const	Variable will not be modified
far	Variable is paged/banked regardless of memory model selected
extern	Variable is allocated in another module
near	Variable is not paged/banked regardless of memory model selected
ram	Locate object in data memory
rom	Locate object in program memory
static	Variable is retained unchanged between executions of the defining block.
volatile	Variable may change from other sources (e.g., input port)

MPLAB C18 Interrupts

To create an interrupt service routine in your MPLAB C18 code, no additional libraries need be included. Simply do the following:

- Create a code section at the interrupt vector that contains a `goto isr` statement, either using inline assembly or a separate assembly file.
- Declare your interrupt routine in your source code using one of the following statements:

  High-priority interrupts – W, BSR, and STATUS are saved in shadow registers

  ```
  #pragma interrupt <isr> [save=symbol-list]
  ```

  Low-priority interrupts – W, BSR, and STATUS are saved on the software stack

  ```
  #pragma interruptlow <isr> [save=symbol-list]
  ```

MPLAB C18 Inline Assembly

MPLAB C18 has an internal assembler with a syntax similar to the MPASM assembler, except that comments must be in the C (/* */) or C++ (//) style. The block of assembly code must begin with `_asm` and end with `_endasm`. For example:

```
_asm
movlw 7       // Load 7 into WREG
movwf PORTB   // and send it to PORTB
_endasm
```

Creating an MPLAB C18 Project in MPLAB IDE

The following are the basic steps required to create an MPLAB C18 based project in the MPLAB IDE. For a more detailed description, please see the MPLAB CXX User's Guide.

1. Specify the include, library, and linker script paths. The library path should be c:\mcc\lib, where c:\mcc is the installation directory for MPLAB C18.
2. Select the development mode (processor and debugging environment)
3. Select MPLINK object linker as the build tool for the target node.
4. Add C files using the **Add Node…** button, specifying the build tool for each as MPLAB C18.
5. Add a linker script file.
6. Add any needed libraries or precompiled object files.

C Language Quick Reference

Operator Precedence

The following chart shows the order in which C language operators are processed. Those with higher precedence will always be processed before those with lower precedence. Operators at the same level are evaluated from left to right.

Highest Precedence
`{} [] -> .`
`! = ++ -- -` (*type cast*) `* & sizeof`
`* / %`
`+ -`
`<< >>`
`< <= > >=`
`== !=`
`&`
`^`
`|`
`&&`
`||`
`?`
`= += -= *= /=`
`,`
Lowest Precedence

Keywords

The ANSI C standard defines 32 keywords for use in the C language. The following table shows the ANSI C and the MPLAB CXX keywords, where MPLAB CXX keywords are shown in bold.

_asm	extern	short
_endasm	**far**	signed
auto	float*	sizeof
break	for	static
case	goto	struct
char	if	switch
const	int	typedef
continue	long	union
default	**near**	unsigned
do	**ram**	void
double	register**	volatile
else	return	while
enum	**rom**	

** has no effect in MPLAB CXX.