C11L11 HTC

了解使用HI-TECH C® PRO编译器的增强型中档PIC® MCU系列

课程安排

- 简介
- 编译器概述
- 数据类型和存储器
- 列表文件
- 映射文件
- 中断
- 运行时代码
- C语言和汇编语言

MASTERs CONFERENCE 2010

接下来...

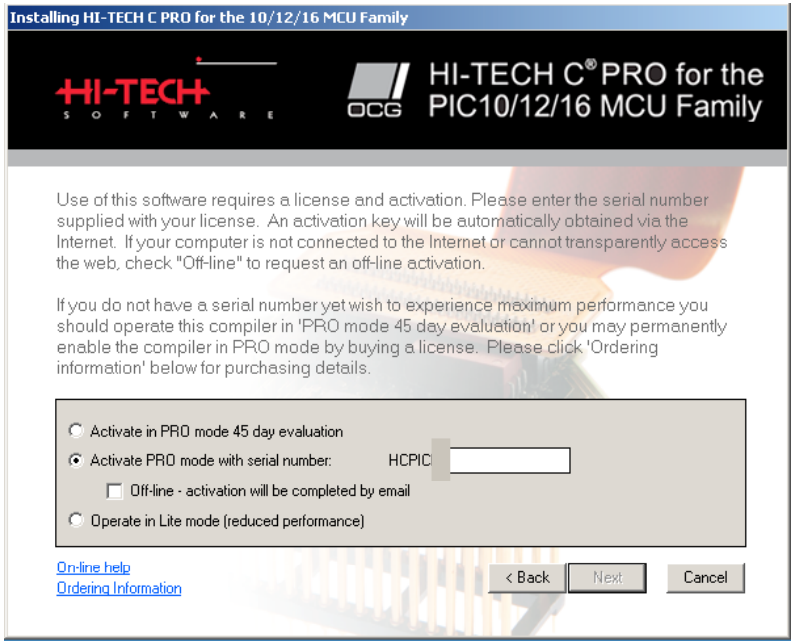
- 简介
- 编译器概述
- 数据类型和存储器
- 列表文件
- 映射文件
- 中断
- 运行时代码
- C语言和汇编语言

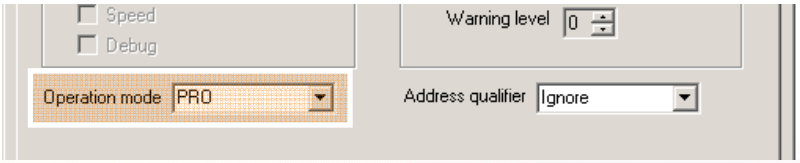
HI-TECH C® PRO模式

- **PRO**（需要许可证）
  - 可全面优化
- 标准（需要许可证）
  - 优化受限
- **Lite**（免费软件）
  - 基本优化，无限制
- 评估版包含**45**天的**PRO**模式，随后将恢复为**Lite**模式

安装

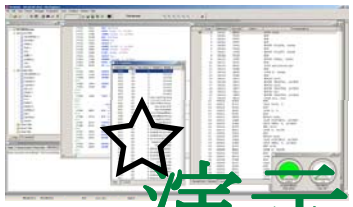
- 安装时进行在线激活
  - 通过代理服务器连接可能需要离线激活
- 可根据项目选择模式





与MPLAB® IDE集成

- **HI-TECH通用工具套件**
  - 安装在编译器上的插件
  - 可独立于编译器进行更新



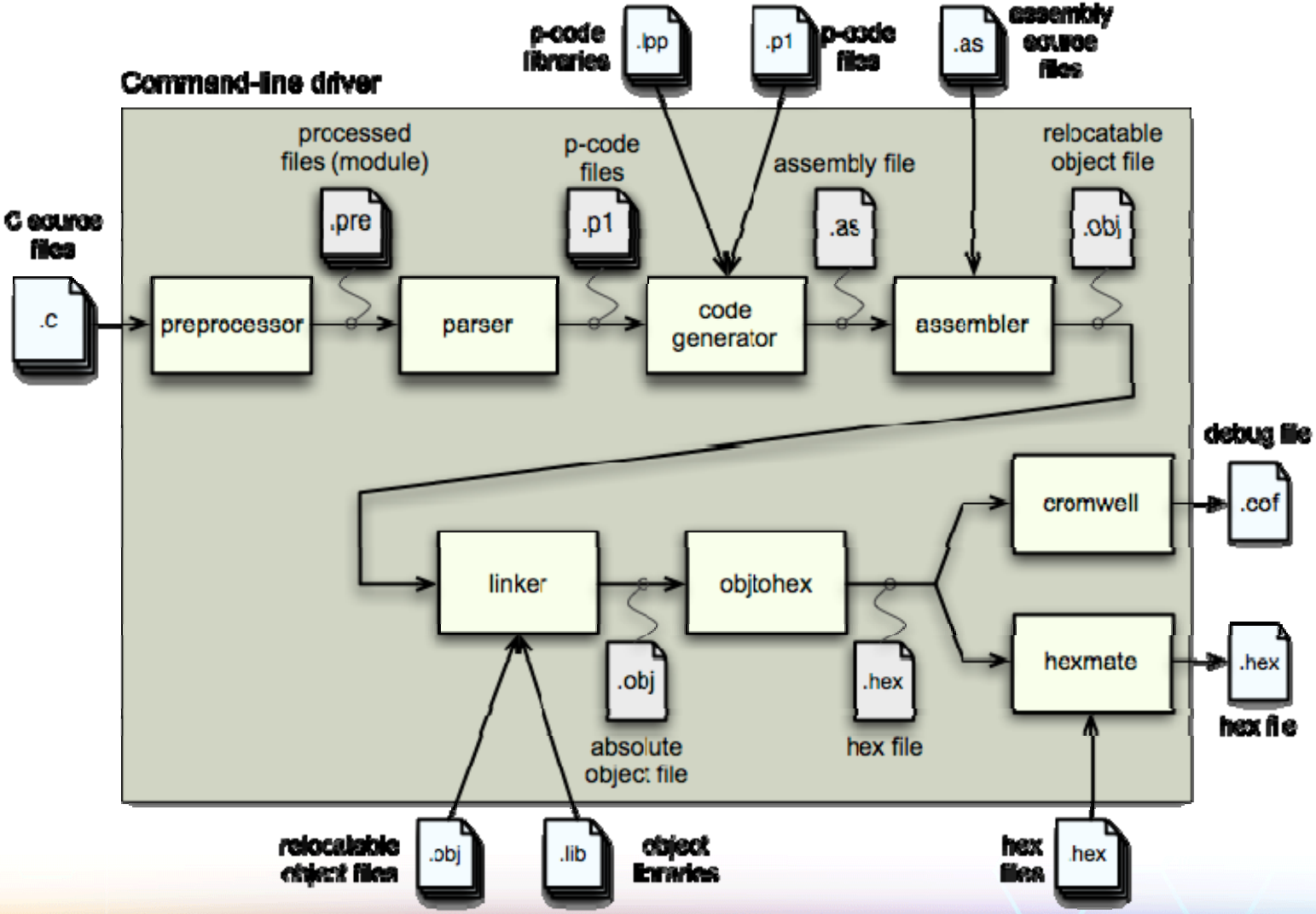
增强型PIC® MCU的改进

- 间接访问系统
  - 多个FSR
  - 通过FSR访问所有存储器
  - 新指令和寻址模式
- 用于中断的影子寄存器
- 新指令
  - Shifts、carry arithmetic和indexed call等
- 更多存储器

接下来...

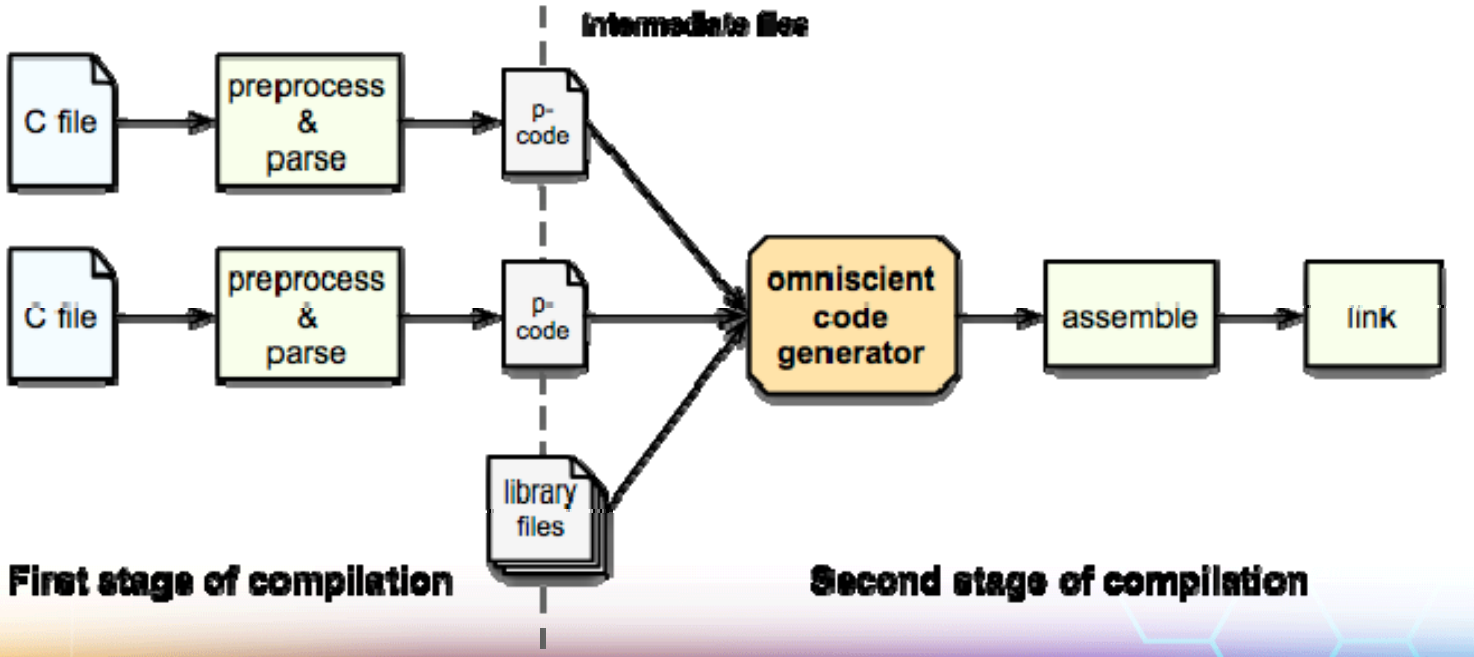
- 简介
- 编译器概述
- 数据类型和存储器
- 列表文件
- 映射文件
- 中断
- 运行时代码
- C语言和汇编语言

编译器应用程序


p-code libraries
.lpp
.p1
p-code files
.as
assembly source files
Command-line driver
processed files (module)
p-code files
assembly file
relocatable object file
C source files
.pre
.p1
.as
.obj
.c
preprocessor
parser
code generator
assembler
debug file
cromwell
.cof
linker
objtohex
hexmate
.hex
hex file
.obj
.hex
absolute object file
relocatable object files
.obj
.lib
object libraries
hex files
hex

编译顺序

- 所有C语言代码都并行编译
  - 包含库文件中的模块
  - P代码文件是中间文件



编译器消息传递

- 通过由驱动器控制的系统实现应用程序报告

```
ts002.c: 159: (762) constant truncated when assigned to bitfield (warning)
```

- 驱动器选项可用于：
  - 调整格式
  - 选择语言
  - 已禁止的警告

编译器消息传递

- **#pragma warning**允许单独控制错误和警告

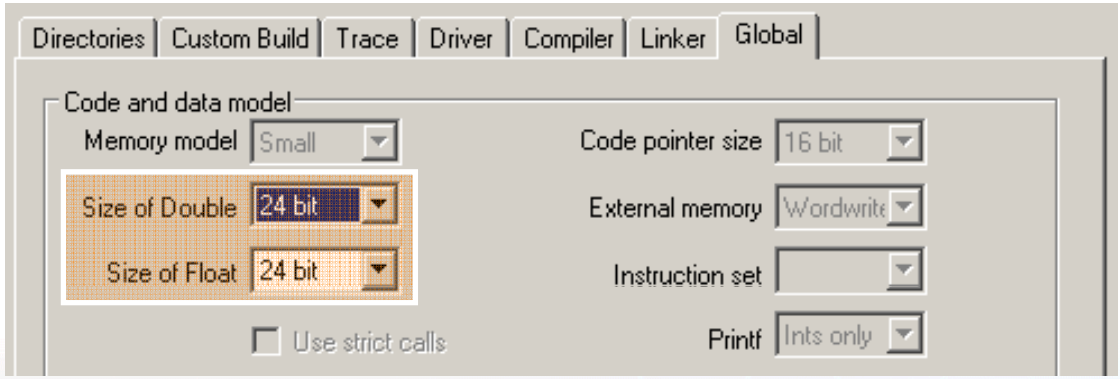
disable *list*	禁止消息
enable *list*	允许消息
push	保存当前状态
pop	获得先前状态
warning *list*	发出警告消息
error *list*	发出错误消息

接下来...

- 简介
- 编译器概述
- 数据类型和存储器
- 列表文件
- 映射文件
- 中断
- 运行时代码
- C语言和汇编语言

标准数据类型

- 标准算法类型
  - **char**型默认为无符号型
  - **double**和**float**型默认为24位宽
  - 用**--double=32**和**--float=32**指定32位



编译器指定类型

- **24位short long整型**
- 用作布尔值的**bit**型
  - 每字节打包8位变量
  - 调试文件中使用的位地址
  - 位变量不为**auto**
  - 采用截尾方式将整型转换为位类型

增强型PIC® MCU的优势

中档系列	增强型中档系列

;ptr.c: 30: a1 += a2;
  bsf 3,5
  bsf 3,6
  movf 110,w
  bcf 3,5
  bcf 3,6
  addwf 32,f
  btfsc 3,0
  incf 33,f
  bsf 3,5
  bsf 3,6
  movf 111,w
  bcf 3,5
  bcf 3,6
  addwf 33,f
162:
;ptr.c: 31: a1 <<= 2;
  bcf 3,0
  rlf 32,f
  rlf 33,f
  bcf 3,0
  rlf 32,f
  rlf 33,f
	;ptr.c: 30: a1 += a2;
	  movlb 3
	  movf 110,w
	  movlb 0
	  addwf 32,f
	  movlb 3
	  movf 111,w
	  movlb 0
	  addwfc 33,f
	156:
	;ptr.c: 31: a1 <<= 2;
	  lslf 32,f
	  rlf 33,f
	  lslf 32,f
	  rlf 33,f

标准限定符

- **const限定符**
  - 对象为只读（不能写入）
  - 存放于程序空间
  - 字符串常量（**"literal"**）具有**const char *型**

标准限定符

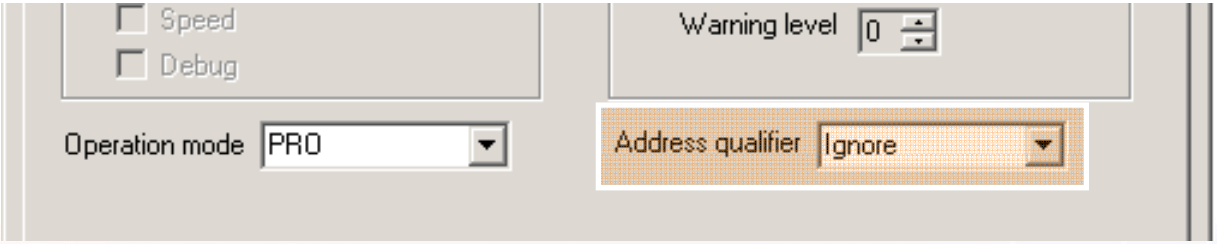
- `volatile`限定符
  - 值可从外部修改
  - 应当用于寄存器/变量：
    - 由硬件修改
    - 产生电信号
    - 由中断程序修改
  - 编译器将尝试原子访问
    - 在一条指令中修改值

编译器指定限定符

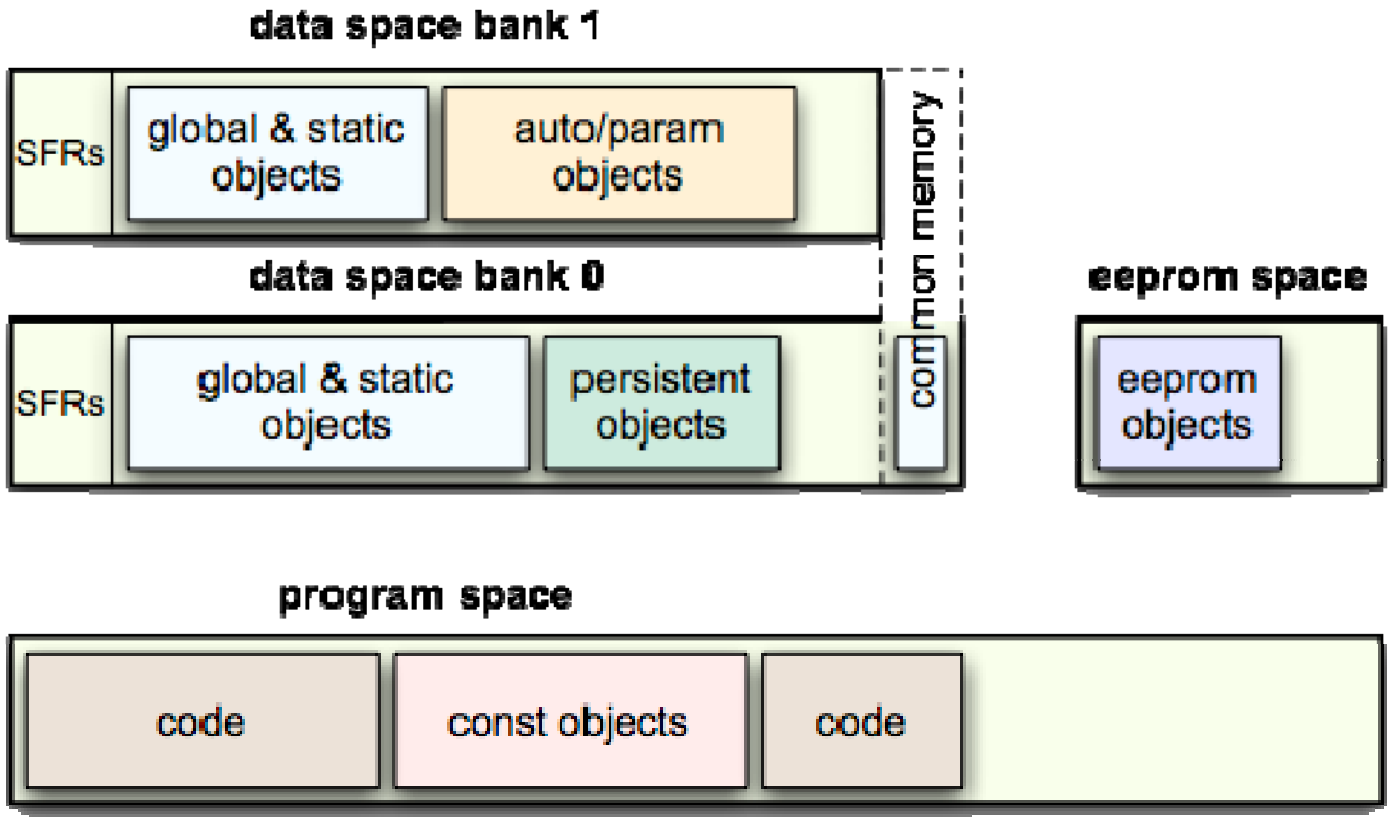
- **near限定符**
  - 尝试存储到通用存储器中
- **persistent限定符**
  - 不会被运行时启动代码清零
  - 复位后仍保持值
- **bank0/1/2/3限定符**
  - 尝试将对象放入指定存储区

控制限定符

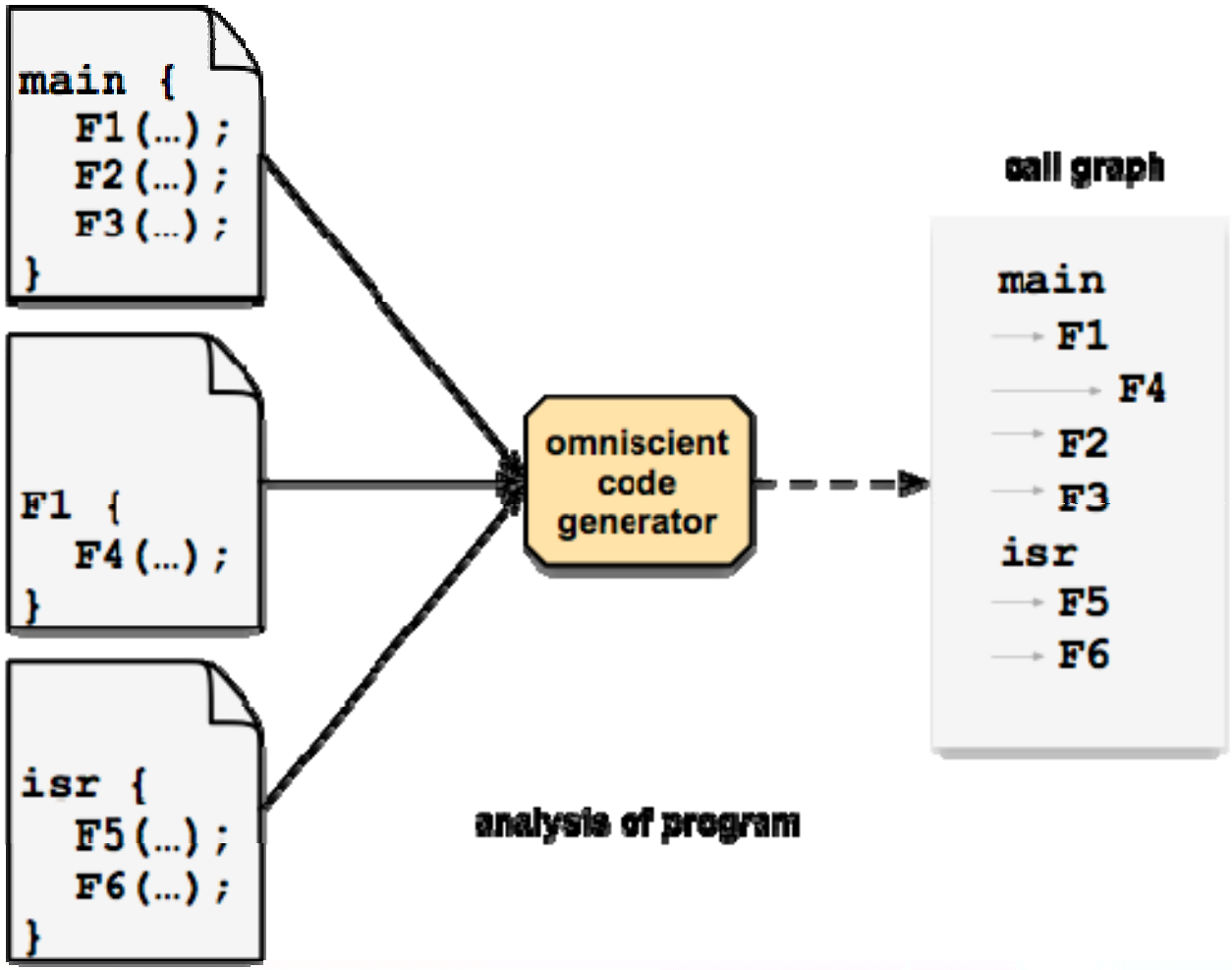
- 由--ADDRQUAL选项控制存储限定符的操作
  - 要求（Require）：必须服从限定符
  - 请求（Request）：如果可以则服从
  - 忽略（Ignore）：必须忽略限定符
  - 拒绝（Reject）：限定符将触发错误



存储空间分配


data space bank 1
SFRs
global & static objects
auto/param objects
common memory
data space bank 0
SFRs
global & static objects
persistent objects
eeprom space
eeprom objects
program space
code
const objects
code

调用图生成

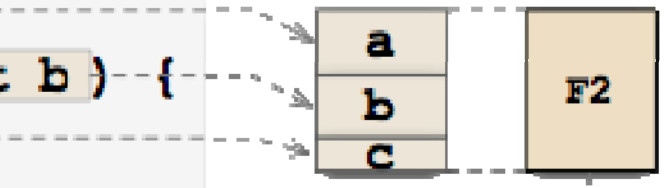

main {
F1(…);
F2(…);
F3(…);
}
F1 {
F4(…);
}
isr {
F5(…);
F6(…);
}
omniscient code generator
call graph
main
F1
F4
F2
F3
isr
F5
F6
analysis of program

编译后的堆栈生成

1 Formation of auto-parameter block (APB) for function F2

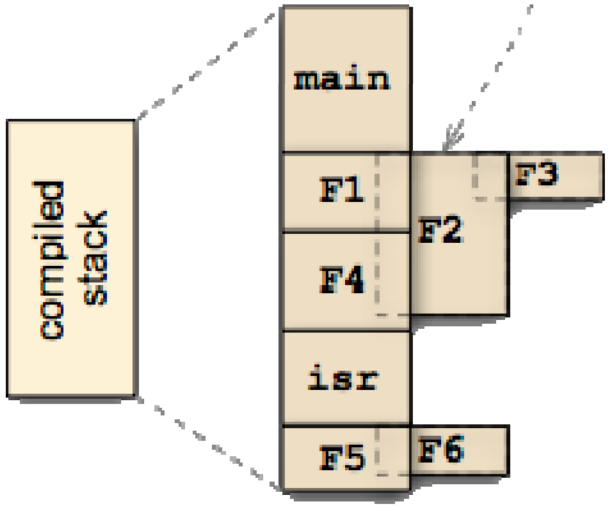
```
F2( int a, int b ){
   char c;

}
```



2 Analysis of call graph

```
main
  F1
    F4
  F2
  F3
isr
  F5
  F6
```



3 Overlap of non-concurrently active APBs to form compiled stack

绝对变量

- 主要用于映射**SFR**变量，例如

```
volatile near unsigned char PIR1 @ 0x0C;
```

- 在用于**GP**变量时要小心
- 头文件包含所有**SFR**的定义：

```
#include <htc.h>
```

指针

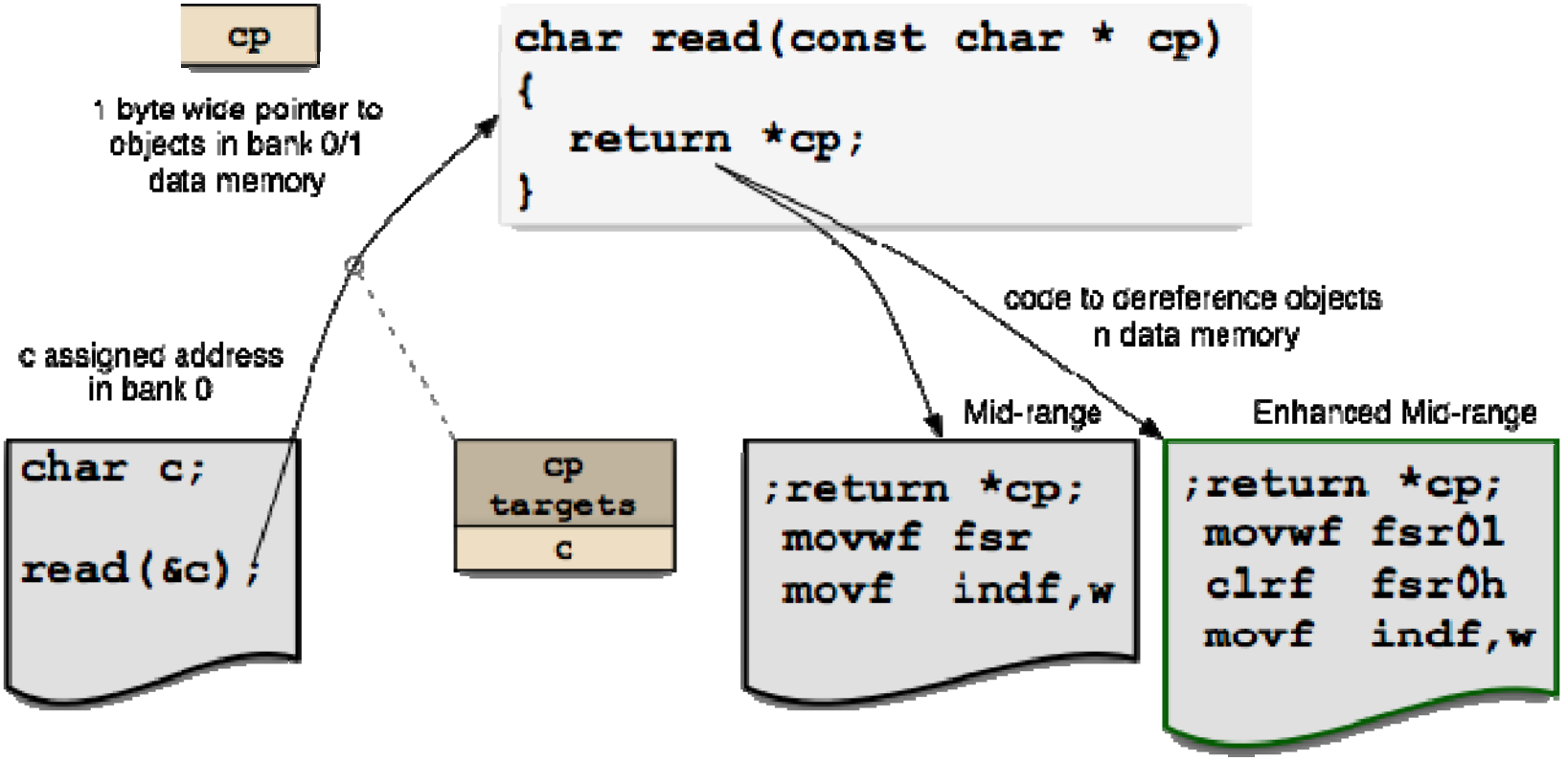
- 完全支持数据和函数指针
  - 定义通用格式及示例：

对象类型和限定符	*	指针限定符	指针名称及初始化
`char`	`*`		`cp ;`
`const char`	`*`		`cp ;`
`char`	`*`	`const`	`cp = address;`
`const char`	`*`	`const`	`cp = address;`

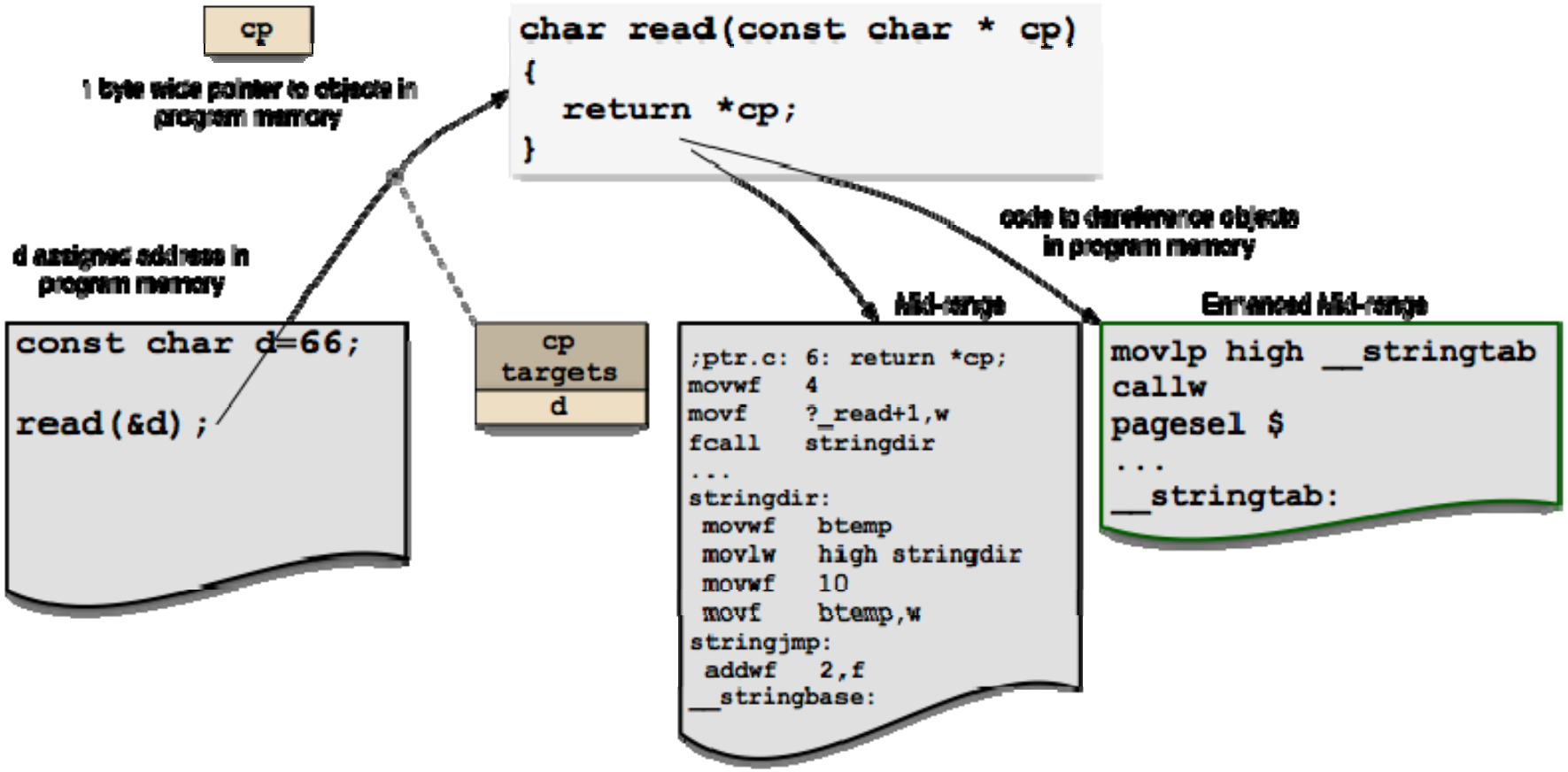
指针

- 独立指针大小和范围由使用情况决定
  - 每个目标的存储空间/区
  - 目标的大小（尤其是数组）
  - 特殊的限定符不再需要或考虑
    - 标准限定符维持原有意义

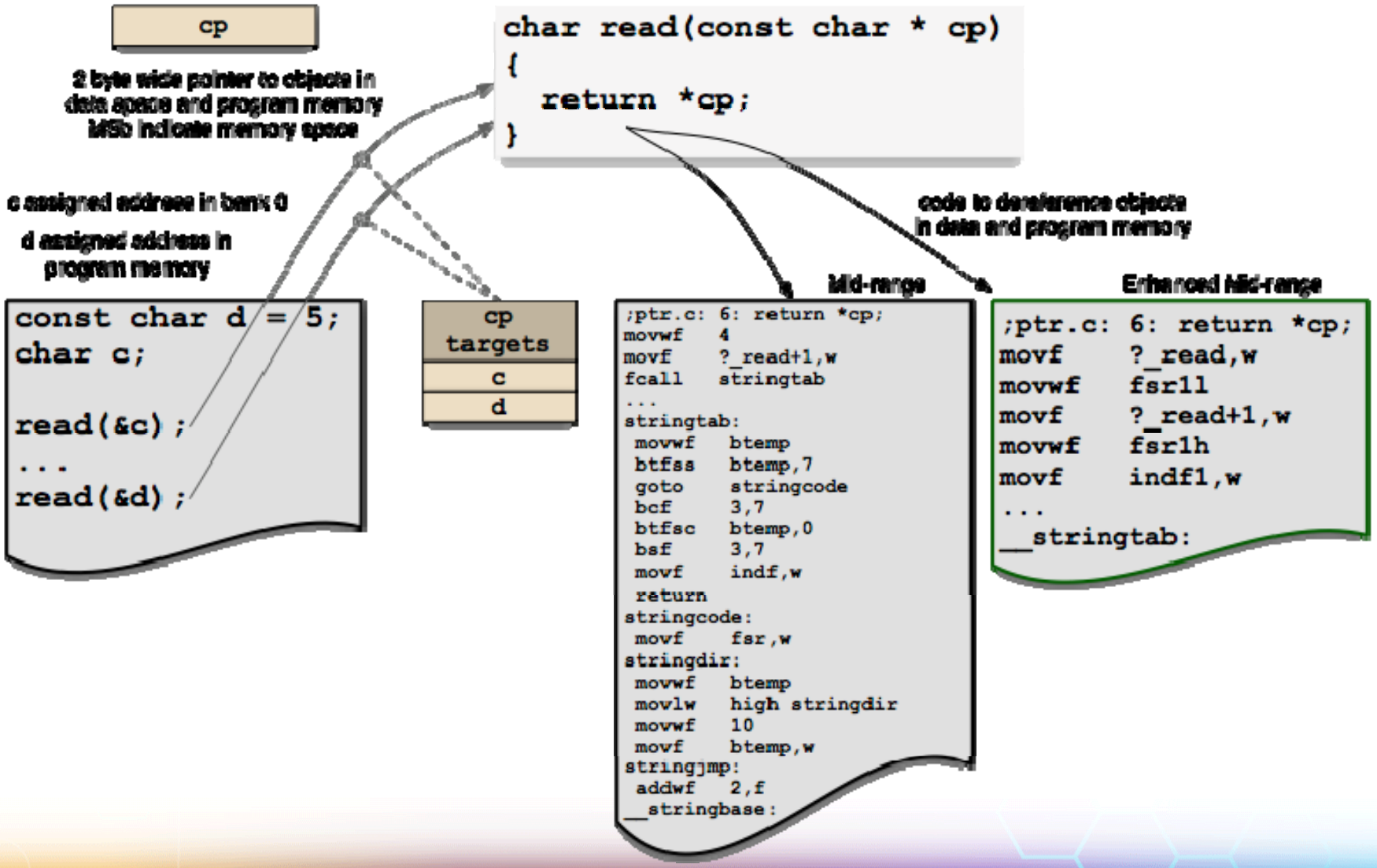
指针示例


cp
1 byte wide pointer to objects in bank 0/1 data memory
char read(const char * cp)
{
return *cp;
}
code to dereference objects n data memory
c assigned address in bank 0
Mid-range
Enhanced Mid-range
char c;
read(&c);
cp targets
c
;return *cp;
movwf fsr
movf indf,w
;return *cp;
movwf fsr01
clrf fsr0h
movf indf,w

指针示例


cp
1 byte wide pointer to objects in program memory
char read(const char * cp)
{
return *cp;
}
d assigned address in program memory
const char d=66;
read(&d);
cp targets
d
code to dereference objects in program memory
Mid-range
;ptr.c: 6: return *cp;
movwf 4
movf ?_read+1,w
fcall stringdir
...
stringdir:
movwf btemp
movlw high stringdir
movwf 10
movf btemp,w
stringjmp:
addwf 2,f
__stringbase:
Enhanced Mid-range
movlp high __stringtab
callw
pagesel $
...
__stringtab:

指针示例



增强型PIC® MCU的优势

中档系列	增强型中档系列

;ptr.c: 30: if(*cp == 'x')
movf 125,w
bcf pclath,3
bcf pclath,4
call stringdir
bcf pclath,3
bcf pclath,4
xorlw 120
btfss 3,2
goto l2
151
;ptr.c: 31: *lp = 0x1234;
movf 114,w
movwf 4
movlw 52
movwf indf
incf 4,f
movlw 18
movwf indf
stringdir:
movwf btemp
movlw high stringdir
movwf 10
movf btemp,w
addwf 2,f
	;ptr.c: 30: if(*cp == 'x')
	movf 125,w
	movlp high __stringtab
	callw
	pagesel $
	xorlw 120
	btfss 3,2
	bra l255
	155
	;ptr.c: 31: *lp = 0x1234;
	movf 114,w
	movwf 4
	clrf 5
	movlw 52
	movwi fsr0++
	movlw 18
	movwf indf

函数调用

- 所有器件使用堆栈，然后恢复为查找表
  - 堆栈意味着调用指令
  - 查找表速度较慢，但是允许无限制的调用深度
- 可定义绝对函数

```
void special(int a) @ 0x100
{
  …
```

行内延迟

- 特殊行内延迟“函数” _delay

```
;wait.c: 10: _delay(10);
  movlw   3
  movwf   ??_main
u857:
  decfsz  ??_main,f
  goto    u857
```

```
;wait.c: 12: _delay(2000);
  movlw   3
  movwf   ??_main+1
  movlw   151
  movwf   ??_main
u877:
  decfsz  ??_main,f
  goto    u877
  decfsz  ??_main+1,f
  goto    u877
  nop2
```

- 已提供的宏也使用_delay
  - __delay_ms和__delay_us

接下来...

- 简介
- 编译器概述
- 数据类型和存储器
- 列表文件
- 映射文件
- 中断
- 运行时代码
- C语言和汇编语言

汇编列表文件

- 汇编器列表文件显示：
  - C语言或汇编语言源代码
  - 输出汇编代码和伪指令
  - C语言程序的：
    - 函数信息
    - 符号表
    - 程序调用图
    - 指针引用图

符号映射——全局

- 在汇编符号前使用下划线
  - 应用到函数和变量
  - 采用偏移量来指定附加的字节

```
int foobar;

F2(int a , int b ) {
  char c;

  foobar &= a;
  c++;
  F4();
}
```

```
_F2:
    movf    F2@a,w
    andwf   _foobar
    movf    F2@a+1,w
    andwf   _foobar+1
    incf    F2@c
    fcall   _F4
    return
```

符号映射——局部

- 采用编译器生成的名称（基于定义符号的函数）

```
int foobar;

F2(int a , int b ) {
  char c;

  foobar &= a;
  c++;
  F4();
}
```

funcName@varName

```
_F2:
    movf    F2@a+0,w
    andwf   _foobar
    movf    F2@a+1,w
    andwf   _foobar+1
    incf    F2@c
    fcall   _F4
    return
```

接下来...

- 简介
- 编译器概述
- 数据类型和存储器
- 列表文件
- 映射文件
- 中断
- 运行时代码
- C语言和汇编语言

映射文件

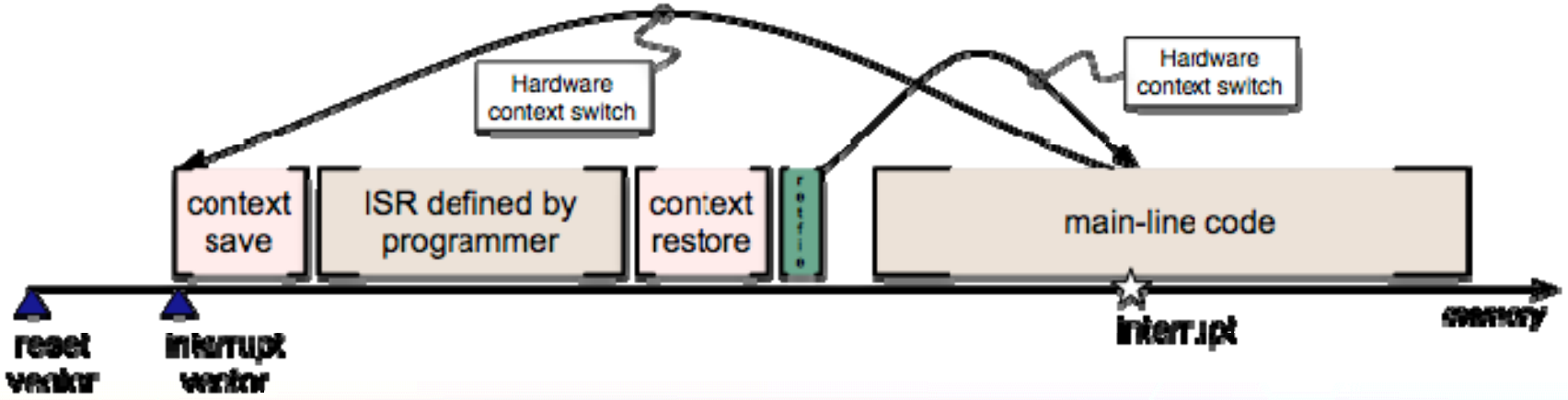
- 映射文件由以下各项组成：
  - 链接器使用的选项
  - 每个模块定义的Psect
  - 按类列出的Psect汇总
  - 未使用的存储单元
  - 程序符号表

接下来...

- 简介
- 编译器概述
- 数据类型和存储器
- 列表文件
- 映射文件
- 中断
- 运行时代码
- C语言和汇编语言

中断

- 一个中断向量（地址4）
  - 中断服务程序（ISR）链接到此处
- 许多中断源
  - 中断标志位决定中断源



软件中断代码

- **ISR可全部用C语言编写**
  - 由限定符**`interrupt`**识别
  - 不能包含参数并且必须是**`void`**返回类型
  - 不能由main程序代码调用
  - 将与中断向量链接
  - 将通过**`retfie`**指令返回

中断示例

- **ISR可使用任何有效的C标识符**
- 总是要检查中断源
  - 检查中断标志位和中断使能位

```
void interrupt isr(void)
{
  if(RCIF && RCIE)
    byte = RCREG;
}
```

现场切换

	中档	增强型
GIE禁止	✔	✔
硬件保存	PC	PC, PCLATH, WREG, STATUS, FSRs, BSR
软件保存	PCLATH, WREG, STATUS, FSR	
用户ISR	✔	✔
软件恢复	PCLATH, WREG, STATUS, FSR	
硬件恢复	PC	PC, PCLATH, WREG, STATUS, FSRs, BSR
返回	`retfie`	`retfie`
GIE使能	✔	✔

中断警告

- 仅保存**ISR**调用图所使用的寄存器
  - 不扫描行内汇编
- 在**ISR**内的任何位置不要重新使能**GIE**位

重入问题

- 所有函数均不能重入
  - 当前正在调用时重入将破坏局部变量

```
void
main(void) {
  while( ! ready)
    wait();
  // ...
```

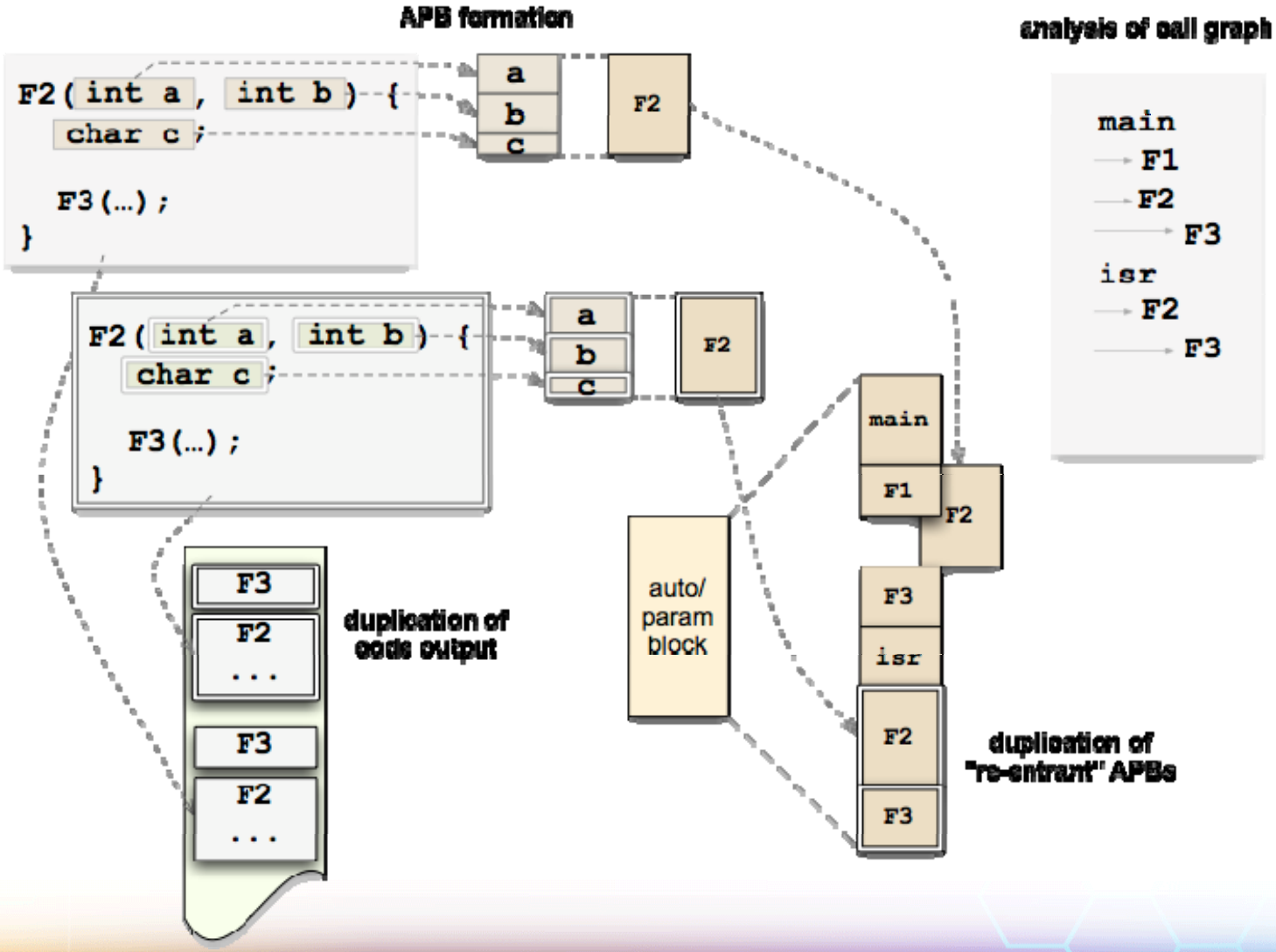
```
void interrupt
isr(void) {
  while( ! ready)
    wait();
  // ...
```

函数复制

- 以重入方式调用的函数会被复制
  - 复制标识符使用前缀**i1**
  - 可以使用**pragma**来禁止复制，假如：
    - 函数确定不会以重入方式调用；或者
    - 函数不包含参数、自动或临时变量。

```
#pragma interrupt_level 1
void wait(void) { ...
```

函数复制


APB formation
F2 (int a , int b) {
char c ;
F3 (…) ;
}
a
b
c
F2
F2 (int a , int b) {
char c ;
F3 (…) ;
}
a
b
c
F2
F3
F2
...
F3
F2
...
duplication of
code output
auto/
param
block
main
F1
F2
F3
isr
F2
F3
duplication of
"re-entrant" APBs
analysis of call graph
main
F1
F2
F3
isr
F2
F3

原子级操作

- 中断是一种汇编功能
- 一条C语言声明可能产生许多汇编指令
  - 可能会在表达式中间中断
  - 对于多字节对象很重要
  - **`Volatile`**并不能保证原子级操作

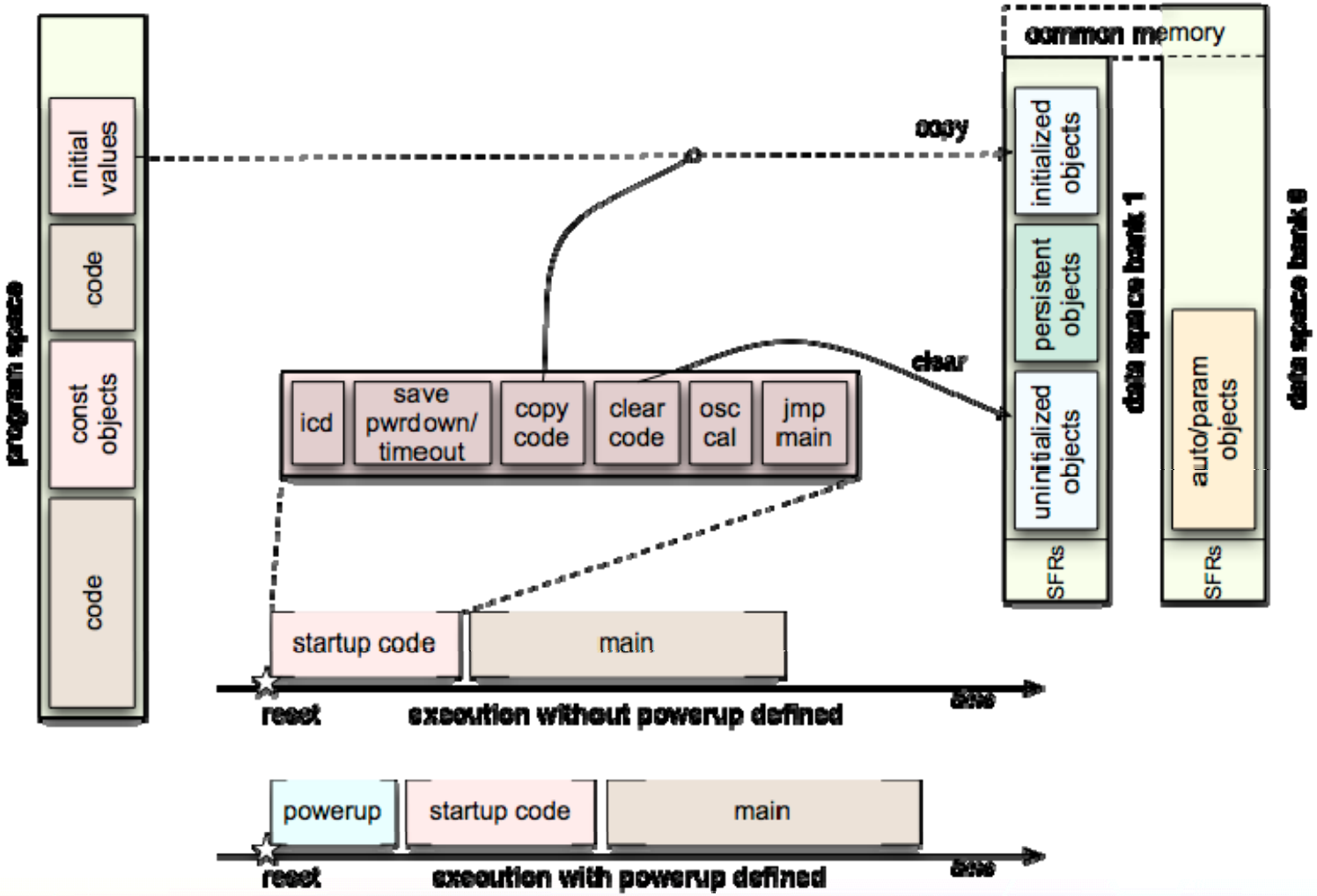
接下来...

- 指令
- 编译器概述
- 数据类型和存储器
- 列表文件
- 映射文件
- 中断
- 运行时代码
- C语言和汇编语言

库代码

- 驱动器选择相关的库文件
- 只包含使用的函数
- 先扫描定义的源代码
  - 请参见列表文件以确认
- 每次编译时定制需要的程序：
  - 运行时启动代码（汇编语言）
  - **Printf**函数（C语言代码）

运行时启动代码

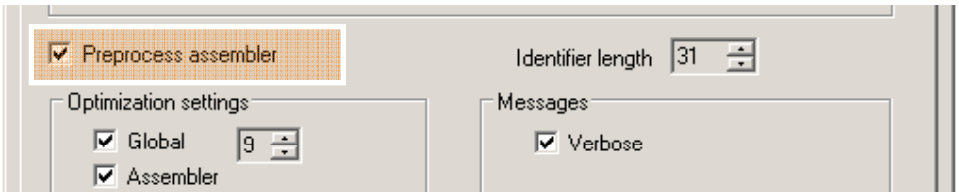

program space
initial values
code
const objects
code
common memory
copy
clear
initialized objects
persistent objects
uninitialized objects
SFRs
data space bank 1
auto/param objects
SFRs
data space bank 0
icd
save pwrdown/ timeout
copy code
clear code
osc cal
jmp main
startup code
main
reset
execution without powerup defined
time
powerup
startup code
main
reset
execution with powerup defined
time

Powerup程序

- 如果存在，则在复位后运行**Powerup**程序
- 提供自动使用：
  - 代码在**powerup** psect内
  - 完成时跳至**start**

Powerup示例

```
#include          "aspic.h”
    GLOBAL   powerup,start
    PSECT    powerup
powerup:
;   Insert powerup code here
        clrf     STATUS
        movlw    start>>8
        movwf    PCLATH
        goto     start & 0x7FF
```



Printf函数

- 采用额外步骤检测printf的使用
  - 说明了占位符和参数
- 定制的printf会被编译出
- 必须在putch中定义stdout

```
void putch(char data) {
  while( ! TXIF)
    continue;
  TXREG = data;
}
```

接下来...

- 简介
- 编译器概述
- 数据类型和存储器
- 列表文件
- 映射文件
- 中断
- 运行时代码
- **C语言和汇编语言**

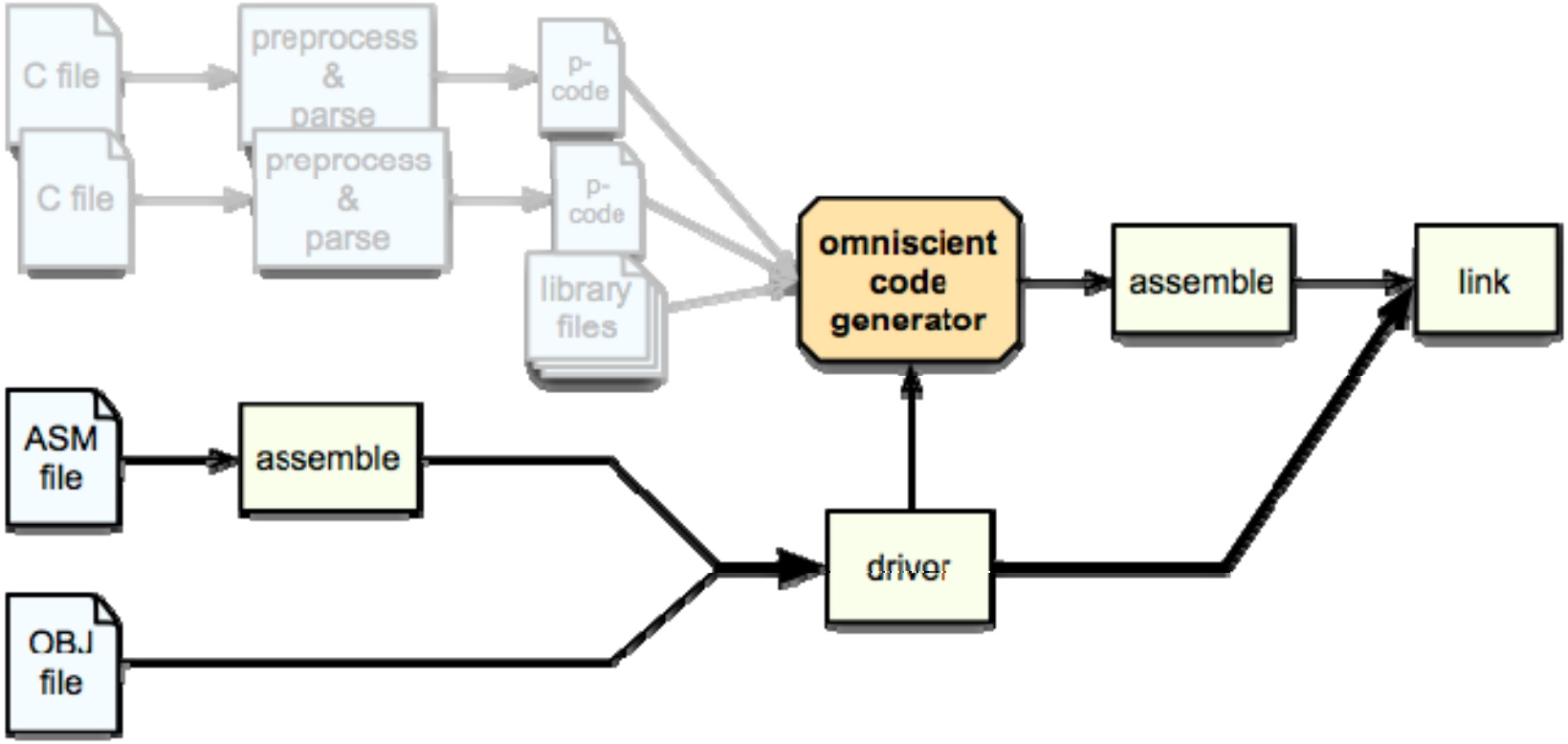
汇编语言代码

- 独立的汇编语言模块
- 借助以下任一种方法用C代码嵌入行内汇编：
  - 放置一条指令的**asm(“ … ”);**语句；或者
  - **#asm … #endasm**指令块
    - 从句法上来讲不是C语言代码的一部分
    - 不要用在循环中
  - 行内汇编代码可能会被汇编器优化程序更改

汇编语言与C语言的交互

- 针对以下项进行独立扫描：
  - 只用在汇编中的C符号
    - C符号为**volatile**
  - 汇编中的存储器要求
    - 存储器由**CGEN**保留

汇编语言与C语言的交互



总结

- 今天我们讨论了：
  - 编译器的组成
  - 存储器分配
  - 列表文件和映射文件中的有用信息
  - 如何处理中断
  - 运行时代码的产生
  - 哪些领域可采用增强型中档系列器件进行代码优化

其他资源

- **“Embedded C Programming”**
  - Chuck Hellebuyck
  - 第二版现已上市
    - 定时器、中断、通信和显示等

- 网上研讨会
  - http://www.microchip.com/webinars

商标

Microchip的名称和徽标组合、Microchip徽标、dsPIC、KeeLoq、KeeLoq徽标、MPLAB、PIC、PICmicro、PICSTART、PIC^{32}徽标、rfPIC和UNI/O均为Microchip Technology Incorporated在美国和其他国家或地区的注册商标。

FilterLab、Hampshire、HI-TECH C、Linear Active Thermistor、MXDEV、MXLAB、SEEVAL和The Embedded Control Solutions Company均为Microchip Technology Incorporated在美国的注册商标。

Analog-for-the-Digital Age、Application Maestro、CodeGuard、dsPICDEM、dsPICDEM.net、dsPICworks、dsSPEAK、ECAN、ECONOMONITOR、FanSense、HI-TIDE、In-Circuit Serial Programming、ICSP、Mindi、MiWi、MPASM、MPLAB Certified徽标、MPLIB、MPLINK、mTouch、Omniscient Code Generation、PICC、PICC-18、PICDEM、PICDEM.net、PICkit、PICtail、REAL ICE、rfLAB、Select Mode、Total Endurance、TSHARC、UniWinDriver、WiperLock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。

在此提及的所有其他商标均为各持有公司所有。