第四章 DSP的开发环境与工具

教材 **Page168**

参考作业: **(Page273) 4.1 4.2 4.3 4.4**

DSP开发环境和工具的重要性

- 如何提高开发速度，降低开发难度，是所有开发者所共同关心的。
- DSP的硬软件开发环境如何，开发工具的功能是否丰富，使用是否方便，其所得结果的效果怎样，等等，已经成为该DSP是否为用户所接受、从而得到推广使用得重要指标之一。

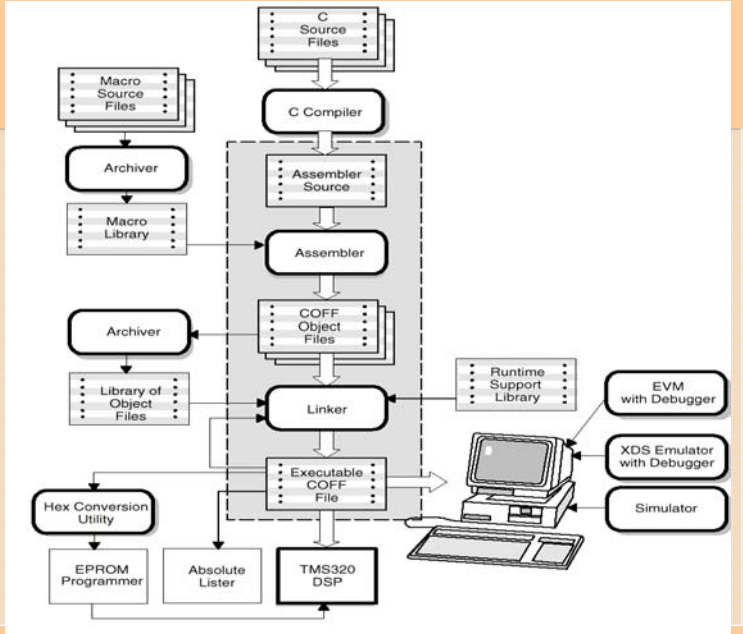
DSP开发环境和工具的重要性

- 各DSP生产厂商以及许多第三方公司作了极大的努力，为DSP系统集成和硬软件的开发提供了大量有用的工具，使其成为DSP发展过程中最为活跃的领域之一，并随着DSP技术的发展而不断地完善。
- 本章仍然以TI公司的TMS320系列DSP，尤其是目前广泛使用的‘C54xx为例，介绍目前使用得比较广泛的开发环境和工具。

DSP的开发环境和工具

- 集成开发环境CCS（编译器，连接器，优化C编译器，转换工具等）
- CMD内存定位文件
- DSP/BIOS
- 实时操作系统

DSP开发流程图



§4-1 代码生成工具

每个DSP系列都提供代码生成工具

- C编译器(C compiler)：将C源程序代码编译成为320系列对应汇编语言源代码。编译包中包括一个外壳程序（shell program）、一个优化器（optimizer）和一个内部列表公用程序（interlist utility）。
- 汇编器(assembler)：将汇编语言源文件转变为基于公用目标文件格式（COFF）的机器语言目标文件。

每个DSP系列都提供代码生成工具

- 连接器（linker）：将目标文件连接起来产生一个可执行模块。它能调整并解决外部符号参考。连接器的输入是可重新定位的COFF目标文件和目标库文件。
- 归挡器（archiver） 将一组文件归入一个归挡文件，也叫归档库。另外，归档器允许通过删除、替代、提取或增加文件来调整库。归档器最有用的应用之一是建立目标文件库。

每个DSP系列都提供代码生成工具

- 助记符到代数语言的转换公用程序（mnemonic-to-algebraic translator utility）：转换汇编语言源文件。该程序接收含有助记符的指令，并将其转换为代数指令，产生一个含有代数指令的汇编语言源文件。

每个DSP系列都提供代码生成工具

- 运行支持库公用程序（runtime-support utility） 建立用户的C语言运行支持库。标准运行支持库函数在rts.src里提供源代码，在rts.lib里提供目标代码。
- 运行支持库（runtime-support library）包含ANSI标准运行支持函数、编译器公用程序函数、浮点算术函数和被C54x编译器支持的C输入/输出函数。

每个DSP系列都提供代码生成工具

- TI DSP的调试器接收可执行的COFF文件作为输入，但大多可擦除存储器却不支持COFF文件。十六进制转化公用程序（hex conversion utility）将COFF目标文件转化为TI-Tagged，ASCII-hex，Intel，Motorola-S，Tektronix等目标格式，从而可以将转化文件装载在可擦除程序存储器里。同时该转换工具还可以自动生成BOOTLOADER程序所需要的引导信息。

每个DSP系列都提供代码生成工具

- 绝对列表器（absolute lister） 接收已经连接的目标文件作为输入，并产生.abs文件作为输出。汇编.abs文件后产生含有绝对地址的列表。
- 交叉引用列表（Cross-Reference Lister）用目标文件来产生一个交叉引用列表，列出符号、符号的定义、以及它们在已连接的源文件中的引用。

汇编器包括以下功能

- 处理汇编语言源文件中的源语句，产生一个可重新定位的目标文件
- 允许将代码分段
- 定义和引用全局符号
- 汇编条件块
- 支持宏调用，并允许在程序内或在库中定义宏

ASM500命令格式如下

- asm500 [input file [object file [listing file]]] [-options]

- **input file:** 汇编源文件名，缺省后缀为**.asm**
- **object file:** 编译输出的**OBJ**文件名,缺省后缀为**.obj**
- **listing file:** 产生的列表文件名,缺省后缀为**.lst**
- **options:** 编译器使用的各种选择。常用选项有：

- **-c**：若使用该选项，编译器忽略字母的大小写。例如**abc**与**ABC**是一样的。系统缺省为区分大小写。

- **-i：设置搜索路径。通知编译器在指定的搜索路径中去查找.copy,.include中的文件。用法举例：-ic:\c54x。**
- **-l: (小写的L）在编译时产生列表文件，缺省后缀为.lst**
- **-mg:** 汇编源程序使用代数指令集 或在源代码中使用 .Algebraic
- **-s：将所有的符号都放入符号表。若不使用该选项，编译器仅将全局变量放入符号表。**
- **-v: 指定版本。特别是需要使用BOOTLOADER时，应加-v548开关。**

不同系列使用不同的汇编工具

- 由于TI的各个DSP系列所使用的汇编指令、机器代码各不相同，所以应使用相应的编译器编译。
- 若使用TMS320C3X/4X系列的DSP，编译器命令为ASM30.EXE；‘C2000系列使用DSPA.EXE；‘C6000系列使用ASM6X.EXE。
- 有关编译器的详细的选项说明请参阅相应的技术手册。

§5-1-2 连接器

- 连接器将目标文件连接成一个可执行目标模块。
- 当它产生一个可执行模块时，执行重定位操作并解决外部引用的问题。
- 连接器指令允许合并目标文件、文件块，把块和标志合并到特定的地址中，以及定义或重新定义全局符号。

连接器有以下的功能

- 定义一个与目标系统存储器一致的存储模块
- 重新定位块，以赋予它们最后的地址
- 在目标系统存储器内将块重新定位到特定的区域
- 定义或重定义全局变量，以赋予它们特定的值
- 解决未定义的输入文件之间的外部引用

连接工具的使用

- 使用LNK命令可以将一个或多个OBJ文件连接为一个.OUT文件。
- 在连接时，可以通过段定位控制命令将不同的代码、数据写入不同的内存单元。
- 注意：LNK生成的OUT文件不是纯二进制代码文件，而是包含代码、符号表、代码定位信息的复合文件。
- 下面我们以TMS320中的C5000系列为例

LNK500的使用格式：

lnk500 [-options] filename 1 . ... filename n

LNK500命令常用的选项有：

- -e global_symbol: 定义程序的进入点。global_symbol必须在源程序中使用.global命令说明。
- -c：使用C编译器的ROM初始化模式。
- -cr: 使用C编译器的RAM初始化模式。

LNK500命令常用的选项有

- -i dir ： 指定库文件的路径。
- -l filename: 指定连接时使用的库文件名。
- -m filename: 生成MAP文件。
- -o filename: 指定生成的OUT文件名。系统缺省为a.out。

LNK500命令使用例子：

- 将file1.obj和file2.obj连接起来生成file.out文件，同时生成MAP文件。
- 通过查看MAP文件可以获得变量、子程序等符号的具体地址。
- 注意：MAP文件只列出全局变量的地址。即在汇编语言中用.global说明的符号。其命令如下：

```
lnk500 file1.obj file2.obj -o file.out -m file.map
```

不同系列使用不同的连接工具

- 若使用TMS320C3X/4X系列的DSP，连接器命令为LNK30.EXE；‘C2000系列使用DSPLNK.EXE；‘C6000系列使用LNK6X.EXE。有关连接器的详细的选项说明请参阅相应的技术手册。

§5-1-3 段的定义

- 为了有效利用**DSP**的内部或外部存储器，目标文件中的代码和数据被存放在不同的段中。
- 这些数据段或代码段在内存中可以是分开的，也可以是连续的。
- 在生成的**OUT**文件中包括三个缺省段：
- 汇编器和连接器允许创建、命名和连接已经命名的段。

段有两种基本的类型

- 已初始化的段：包括数据和代码。例如.text 和.data段都是已初始化段。在汇编语言中自己定义的用于存放代码的.sect段也是已初始化段。
- 未初始化的段：是为未初始化数据保留的内存空间。例如，.bss段是未初始化的，用于为变量保留空间。在汇编语言中还可以使用.usect指令说明一个未初始化段

连接器如何使用段

- 连接器有两个与段有关的功能：一个是在建立输出OUT文件时，连接器要使用OBJ目标文件中的段，合并输入段（当被连接的文件多于一个时）。另一个是连接器为输出段选择存储器地址。
- 这些连接器可以根据用户提供的CMD定位文件，来安排段的实际地址。这个CMD定位文件一般有两个部分：

在代码中三个缺省段

- **.text** 包括可执行代码
- **.data** 包括已经初始化了的数据
- **.bss** 为未初始化变量保存空间

- 还可以在汇编语言中自己定义的用于存放代码的段，如 **.sect**（代码），**.usect**、**.buffer**（数据）等

告诉连接器可以使用的存储器

- 存储器指令（**memory directive**） 定义目标系统的存储器图。用户可以给部分存储器命名，定义它的起始地址和长度。
- 段指令（**sections directive**） 告诉连接器如何将输入段合并到输出段中，及将这些输出段放在存储器的什么地方。用户可以用连接器的**SECTIONS**指令指定子段，如不特别指定，子段将和同基段名的段合并在一起。

内存定位CMD控制文件例子

```
MEMORY
{
   PAGE 0:  PROG:    origin =  1800h, length = 800h
   PAGE 4:  PROG:    origin =  0200h, length = 100h
   PAGE 1:  DATA:    origin =  0c00h, length = 100h
   PAGE 2:  DATA:    origin =  0d00h, length = 200h
   PAGE 3:  DATA:    origin =  0f00h, length = 100h
}
SECTIONS
{
   .text   > PROG PAGE 0
   .vect   > PROG PAGE 4
   .bss    > DATA  PAGE 1
   .data   > DATA  PAGE 2
   .buffer > DATA  PAGE 3
}
```

内存定位CMD控制文件说明

- 其中，包含两个部分：**MEMORY**和**SECTIONS**。在**MEMORY**中，主要说明目标系统中哪些存储器可以使用，它们的起始地址是多少，大小如何。
- 在**SECTIONS**部分中，主要完成段的具体地址的分配。例如，我们将**.text**段存放在**1800h**开始的程序空间中，将**.bss**段存放在**c00h**开始的数据空间中，将**.vect**段存放在程序空间的**200h**开始的地方。

内存定位CMD文件另一例子

```
asmlnk.obj
asmlnk1.obj
-m asmlnk.map
-o asmlnk.out

MEMORY
{
  PAGE 0: IPROG:    origin = 0x880,      len = 0x1f80
          VECT:     origin = 0x800,      len = 0x80
  PAGE 1: USERREGS: origin = 0x60,       len = 0x1c
          BIOSREGS: origin = 0x7c,       len = 0x4
          IDATA:    origin = 0x80,       len = 0x780
}
```

定位CMD控制文件另一例子

```
SECTIONS
{
    .vect:    {} > VECT PAGE 0
    .sysregs: {} > BIOSREGS PAGE 1
    .gblinit: {} > IPROG PAGE 0
    .bios:    {} > IPROG PAGE 0
    .text:    {} > IPROG PAGE 0
    .cinit:   {} > IPROG PAGE 0
    .pinit:   {} > IPROG PAGE 0
    .sysinit: {} > IPROG PAGE 0
    .bss:     {} > IDATA PAGE 1
    .far:     {} > IDATA PAGE 1
    .const:   {} > IDATA PAGE 1
    .switch:  {} > IDATA PAGE 1
    .sysmem:  {} > IDATA PAGE 1
    .sysheap: {} > IDATA PAGE 1
}
```

定位CMD文件可以包含参数

- 在上个例子中，我们将连接器使用的参数添加到内存定位CMD文件中，这样整个连接器命令可以简化为（以C5000系列为例，使用LNK500）：

  C:\C54X\LNK500　asmlnk.cmd

§5-1-4 优化C编译器

一. 优化C编译器的特点

- 标准化 C虽然是最易使用的一种程序语言，但它缺少标准化，特别在一些扩展范围内。ANSI 为这些扩展提供了标准。
- 兼容性 ANSI C是K$R标准的超集。大多数在K&R编译器下编译和运行的C程序在新的ANSI C优化编译器里也能运行。对于为其它处理器编写的代码，只需很少或根本不用增写代码就可以放入TMS320 C编译器中。

优化C编译器的特点

- 新类型 新的常量和变量类型，允许进行改进后的优化。
- 改进的性能协议 函数原形允许改进后的类型检查，并对调用规范进行优化。
- 标准库的支持 该编译器软件包内带有一个完整的运行支持库。库内的所有函数都符合ANSI C的函数标准。它包括字符串操作，动态存储器分配，数据转换，三角、指数和双曲线函数等。

优化C编译器的特点

- 高水平的优化 该编译器能够做高水平的优化，使用多种先进技术从C源代码产生高效汇编代码。通用优化可用于任何C代码，而对个片种的专门优化更适应该片种的结构特点。它通过简化循环、重新安排语句和表达式、使用寄存器变量、使用特殊指令等方法来改善执行速度，减小C程序的大小。

优化C编译器的特点

- 方便的汇编语言接口 该编译器具有灵活的汇编语言接口，使用直接调用协议，易于编写可以相互调用或混合汇编的C代码。
- 强大的预处理功能 C预先处理器（C preprocessor）和分析器（parser）集成在一起，作快速编译，其处理内容包括：宏定义和扩展，#include头文件、条件编译以及各种优先处理指令等。

二.TMS320 优化编译器的优化途径

- C编译器的效率决定于其所作优化的范围和数量。TMS320 C编译器做了许多的优化来提高编译代码的效率。
- TMS320的C优化编译器一般通过两种优化途径产生高效汇编语言代码：通用优化途径和针对特定片种的专门优化。下面分别介绍：

通用优化

- 代数式改变，字符简化，常数合并
- 合并无用代码，优化数据流
- 优化转移/简化控制流
- 循环相关变量的优化
- 以省时为基础的寄存器的使用

针对特定片种的专门优化

- 寄存器变量：将地址寄存器作为指针来使用。当数组下标结构成为循环变量时，这种优化特别有效。
- 自动增量寻址模式：DSP中提供了灵活寻址方式，对于*p++形式的指针表达式，可以通过辅助寄存器的自动增量寻址来实现，而不需要消耗额外的机器时钟。

针对特定片种的专门优化

- 块循环 ：对于大多数的DSP型号都提供了块循环指令RPTB和单指令循环RPT。优化编译器可以使用这些指令实现您零开销循环。
- 延时转移、调用和返回：许多DSP（如‘C5000，‘C3X，‘6000系列）都支持延时跳转、调用和返回。使用这些指令时比起它们的非延时的对应指令来要少两个机器时钟周期。

针对特定片种的专门优化

- 使用寄存器来传递函数变量:优化编译器支持新的选择性的调用顺序,将变量传递给寄存器,而不是压入堆栈,其结果显著地改善了性能。
- 并行指令:许多DSP(如‘C5000，‘C3X，‘6000系列）都提供了并行指令。当邻近的指令满足寻址要求时，编译器将它们组合起来，并行执行，从而大大加快执行速度。

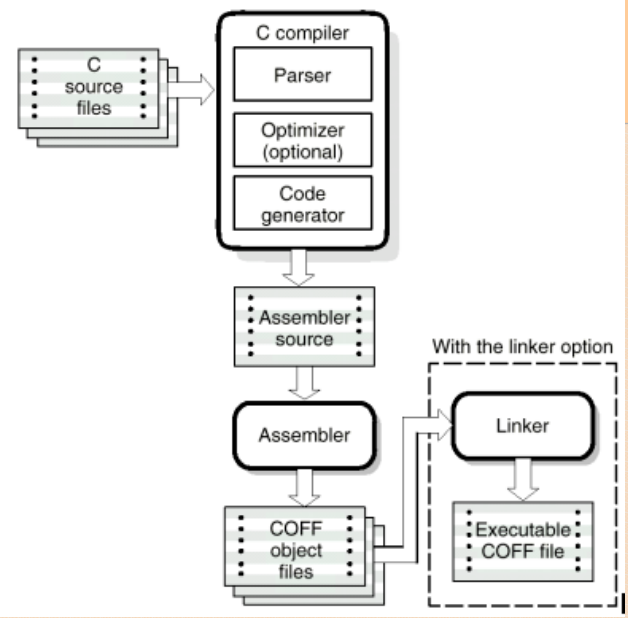
三. 优化C编译器的使用

- **TMS320 DSP**的C编译器主要有代码分析工具、优化器、汇编代码生成器、汇编工具和连接工具组成，其中汇编工具和连接工具与汇编语言共享。
- 编译分为两个阶段：第一阶段分析代码，第二阶段产生汇编语言源程序。完成第一阶段后，还可以选择一种优化途径。

优化C编译器的使用

- **TI DSP的C编译器一般都是通过一个shell程序来完成。例如，在‘C5000系列中，可以使用cl500.exe这个工具完成全部编译、优化、汇编和连接工作。**
- **其它系列的DSP也有类似的shell程序：如‘C2000系列使用 dspcl.exe，‘C3X使用cl30.exe，‘C6000系列使用cl6x.exe。**
- **C编译器的工作流程请参见下图。**

C编译器的工作流程图



使用CL500编译C程序

- Cl500程序按用户的要求，自动调用并运行编译器，优化器和汇编器，当然你也可以单独运行各个工具：

  ac500：C分析器
  opt500 ：优化器
  cg500 ：代码生成器
  asm500：汇编器
  lnk500: 连接器（使用-z 参数）

使用CL500编译C程序

- 使用CL500完成C程序的编译：

  Cl500 function.c

- CL500会自动调用ac500（C分析器）、cg500（代码生成器）和asm500（汇编器)
- 在使用优化参数（如-o）和-z参数后分别调用opt500（优化器）以及lnk500（连接器）
- 所有独立工具的参数，都可在CL500中用。

使用CL500编译C程序

- 下面是一些常用的选项：

  -g: 允许使用C代码级的调试。

  -k: 保留生成的汇编源程序。

  -ss: 产生C的注释汇编源程序。

  -z: 允许调用连接程序。

  -o: 完成优化

使用CL500编译C程序例子

```
cl500 -o function.c -z lnk.cmd -o function.out -
l rts.lib
```

- 使用CL500编译并连接，生成function.out文件（使用lnk.cmd内存映射定位文件）。同时使用优化器，以及标准的C语言库rts.lib。
- 在集成开发环境CCS中，可以通过工程文件添加或选择使用的参数！由CCS自动生成CL500命令的命令行参数。

使用优化编译器后的效果？

- 这段程序很简单，主要完成一个乘累加运算，结果放在变量sum中。为了便于比较优化器的效果，我们在编译时加入-ss参数，以保留C与汇编源代码。

```
sum=0;
for(k=0;k<20;k++)
   sum+=i[k]*j[k];
```

- 先看不使用优化器的情况：

```
;----------------------------------------------
;  12 | for(k=0;k<20;k++)
;----------------------------------------------
    ST      #0,*SP(2)              ; |12|
    SSBX    SXM                    ;
    LD      #20,A                  ; |12|
    SUB     *SP(2),A               ; |12|
    BC      L2,ALEQ      ; branch occurs ; |12|
L1:
;----------------------------------------------
;  13 | sum+=i[k]*j[k];
;----------------------------------------------
    MVDK    *SP(2),*(AR1)          ; |13|
    LD      *SP(3),A               ; |13|
    LD      *AR1(_j),T             ; |13|
    MAC     *AR1(_i), A            ; |13|
    STL     A,*SP(3)               ; |13|
    ADDM    #1,*SP(2)              ; |13|
    LD      #20,A                  ; |13|
    SUB     *SP(2),A               ; |13|
    BC      L1,AGT       ; branch occurs |13|
L2:
```

使用优化编译器后的效果？

- 请注意观察第12和第13行乘累加运算的实现。在未使用优化的情况时，编译器使用最普通的方法实现该运算，即先完成一次乘法，然后累加，再进行循环控制判断。当然，整个运算结果是正确的，只是多消耗时间。
- 如果在编译时使用优化参数，例如使用-o2，完成全局优化，其效果将大大不同。

例如：**cl500 a.c -ss -o(或-o2)**

使用优化参数后的效果

```
;----------------------------------------
;  10 | sum=0;
;  12 | for(k=0;k<20;k++)
;----------------------------------------
     LD      #0,A                    ; |10|
     RPT     #19          ; loop starts  |6|
L1:
;----------------------------------------
;  13 | sum+=i[k]*j[k];
;----------------------------------------
     MAC     *AR3+, *AR2+, A, A      ; |13|
                          ; loop ends    |13|
L2:
```

使用优化编译器后的效果

- 再观察第12和第13行，这次完成乘累加运算使用了MAC指令，并且使用了单指令循环操作，从而大大提高了运算效率。
- 同时，优化器还对数组变量i,j直接使用不同的辅助寄存器寻址，进一步提高了程序运行效率。
- 有关优化器的更详细的介绍、技巧，请参考TI提供的优化编译器手册和相关技术文档。

§5-1-5 十六进制转换工具

- 前面介绍的汇编器和连接器所产生的目标文件是采用COFF（公用目标文件格式）。
- 它不仅包含DSP执行代码，还包含有如符号表，代码和数据的定位等调试信息，所以它支持模块化编程，对管理代码段和目标系统的存储器更强有力和更灵活的方法。
- 该十六进制转换工具就是用来将COFF目标文件转换为若干种标准格式，以便通用EPROM编程器识别。

§5-1-6 其它辅助工具

- ‘C54xx的代数指令到助记符指令的翻译工具mnem2alg.exe 。使用该工具可以实现从代数指令到助记符指令的自动转换。
- F24XX的FLASH写入工具。TI或生产仿真器的第三方都提供了F24XX的片内FLASH的编程工具软件。用户可以使用仿真器、计算机的串口对片内FLASH编程。

§5-2 系统集成与调试工具

TI DSP有那些开发工具?

- TI公司为TMS320系统的集成与调试所提供的工具包括:
  - 软仿真器（Simulator）-------软件仿真器
  - DSP入门套件（DSK，DSP Starter Kit）
  - 标准评估模块（EVM）
  - 扩展开发系统XDS（eXtended Development System）---------硬件仿真器需软件配合使用
  - 集成开发软件CCS（Code Composer Stdio）

§5-2-1 软仿真器（Simulator）

- 软仿真器是一个软件程序，使用主机的处理器和存储器来仿真TMS320 DSP，从而进行软件开发和非实时的程序验证。
- 可以在没有目标硬件的情况下作DSP软件的开发和调试。
- 它可以直接使用由TMS320宏汇编器/连接器或ANSI C编译器所产生的目标代码(.out)文件为输入。

软仿真器的主要特性

- 在主机上执行用户的DSP程序。
- 修改和检查寄存器，进入时初始化寄存器。
- 显示和修改数据和程序存储器：在任何时候可以修改整个块。
- 设置断点：添加指令，读写内存，数据总线或程序总线上的数据类型，出错条件。
- 跟踪累加器、程序计数器、辅助寄存器等。

软仿真器的主要特性

- 单步执行指令。
- 在用户指定的时间产生中断。
- 用文件的方式快速存储和调用仿真参数。
- 反汇编能力，以便对源语句作编辑和重汇编。
- 存储器的内容可以同时显示为十六进制的16-bit值和汇编后的源代码。

Simulator被广泛使用

- 以前的软仿真器软件与其它开发工具如代码生成工具是分离的，使用起来不是太方便。现在，软仿真器作为CCS（Code Composer Studio，一种集成开发环境）的一个标准插件，已经被广泛应用于DSP的开发中。

§5-2-2 DSK系列评估工具以及标准评估模块（EVM）

- DSP入门套件（DSK，DSP Starter Kit）、评估模块（EVM，Evaluation Module）是TI或TI的第三方（Third Party）为TMS320 DSP的使用者设计和生产了一种评价DSP的平台。
- 目前可以为C2x、C3x、C5x、C54x、C6x等片种提供。
- 一般EVM板带有仿真器，DSK不一定。

DSK和EVM的特点：

- DSK或EVM板除了提供一个完整的DSP硬件系统外（包括A/D & D/A、外部程序/数据存储器、外部接口等）。
- 提供有完整的代码生成工具和调试工具。
- 用户可以使用DSK或EVM板来作DSP的实验，进行诸如控制系统、语音处理等测试应用；也可以用来编写和运行实时源代码，并对其进行评估；还可以用来调试用户自己的系统。

TMS320VC5402 DSK板

TMS320VC5402 DSK板的特点

- DSP为100 MHz TMS320VC5402
- 64K字（16 bits）的1个等待周期的外部SRAM
- 256K字的FLASH 存储器
- 该DSK板自带JTAG控制器，并使用并口与PC机相连
- DAA电话线接口
- 麦克风/扬声器音频接口

TMS320VC5402 DSK板的特点

- RS-232异步数据接口
- 外部扩展daughterboard接口
- 5V DC电源
- 为了配合硬件工作，该DSK中还包括一套DSK版的CCS（集成开发/调试工具）

§5-2-3 硬仿真器Emulators（XDS510）

硬仿真器是不可缺少的工具

- **TMS320扩展开发系统XDS（eXtended Development System）是功能强大的全速仿真器，用以系统级的集成与调试。**
- **扫描式仿真（Scan-Based Emulator）是一种独特的、非插入式的系统仿真、集成调试方法。使用这种方法，程序可以从片外或片内的目标存储器实时执行，在任何时钟速度下都不会引入额外的等待状态。**

硬仿真器是不可缺少的工具

- TMS320器件的结构通过内部的、可以由单一串口访问的移位寄存器扫描通道来实现扫描式仿真。
- 该扫描通道提供对内部的器件寄存器和状态机的访问，允许完全的可观察和控制。
- 即便DSP焊接到了目标系统中，这种非插入式的方法仍然可以工作。

硬仿真器是不可缺少的工具

- XDS510/XDS510WS仿真器是用户界面友好、以PC或SUN工作站为基础的开发系统，可以对C2xx、C3x、C4x、C5x、C54x、C8x、C6x等片种实施全速扫描式仿真。
- 用户可以使用XDS510来调试C程序、汇编语言程序，或两者的混合程序。

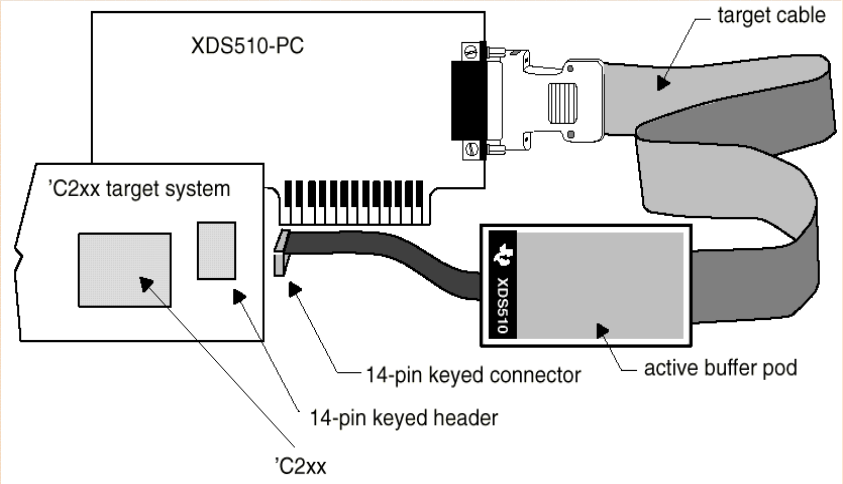
XDS510的主要特性：

- 通过一个十四脚的目标连接器，全速执行和监视目标系统中的器件
- 并行处理DSP的全局运行/停止/断点
- 高级语言（HLL）调试接口
- 最多可达200个断点的软件断点/跟踪和计时
- 对所有程序和数据地址作硬件断点/跟踪
- 单步执行

XDS510的主要特性：

- 与C/汇编源代码接口与调试
- 所有的寄存器和存储器的装载、检查与修改
- 时钟周期执行时间的标准程序检查
- 用户可以很方便地建立显示串口，以其本来格式编辑变量、数组、结构、指针等任何数据的值（浮点数、整数、字符、枚举或指针）。

使用XDS510调试DSP系统



不同DSP芯片都可使用XDS510

- 由于TI的各种型号的DSP中都提供了标准的扫描逻辑电路，使用不同的DSP时不用更换XDS510仿真器。
- 仅需要使用对应调试软件，并做相应的配置，如配置地址范围、内部或外部存储器资源以及片内存储器和外设及等待状态、访问权限等等。

完整的XDS510包括：

- **XDS510仿真器PC板（在PC-AT上工作，占用一个16-bit插槽）**
- **IEEE1149.1（JTAG）或MPSD目标电缆（除C30、C31、C32 XDS510使用MPSD电缆外，其余各片种都使用IEEE 1149.1目标电缆）**
- 各片种的调试软件

TI和第三方都提供仿真器

- 除TI自己提供XDS510仿真器外，还有很多的国内外厂家都提供类似产品。
- 国外厂商如SpectrumDigital等。
- 国内厂商如合众达、闻亭等。
- 他们都提供多种PC接口的仿真器，如特别适合于笔记本电脑使用的并行接口的仿真器XDS510PP。此外，还有PCI、USB接口的仿真器,以适应不同用户的要求。

§5-2-5 集成开发环境（CCS）

集成开发环境（CCS）

- CCS(Code Composer Studio)是一个完整的DSP集成开发环境，也是目前最优秀、最流行的DSP开发软件之一。
- CCS最早是由GO DSP公司为TI的‘C6000系列开发的，后来TI收购了GO DSP，并将CCS扩展到其它系列。
- 现在所有的TI DSP都可以使用该软件工具进行开发，‘C5000和‘C6000，’C2000的CCS中都提供DSP/BIOS功能

1.CCS包含哪些功能？

- 集成可视化代码编辑界面，可直接编写C，汇编、.H文件、.cmd文件等。
- 集成代码生成工具，包括汇编器、优化C编译器、连接器等等。
- 基本调试工具，如装入执行代码（.OUT文件），查看寄存器窗口，存储器窗口，反汇编窗口，变量窗口等，支持C源代码级调试。

CCS包含哪些功能？

- 支持多DSP调试
- 断点工具，包括硬件断点、数据空间读/写断点，条件断点（使用GEL编写表达式）等等。
- 探针工具（probe points），可用于算法仿真，数据监视等。
- 剖析工具（profile points），可用于评估代码执行的时钟数。

CCS包含哪些功能？

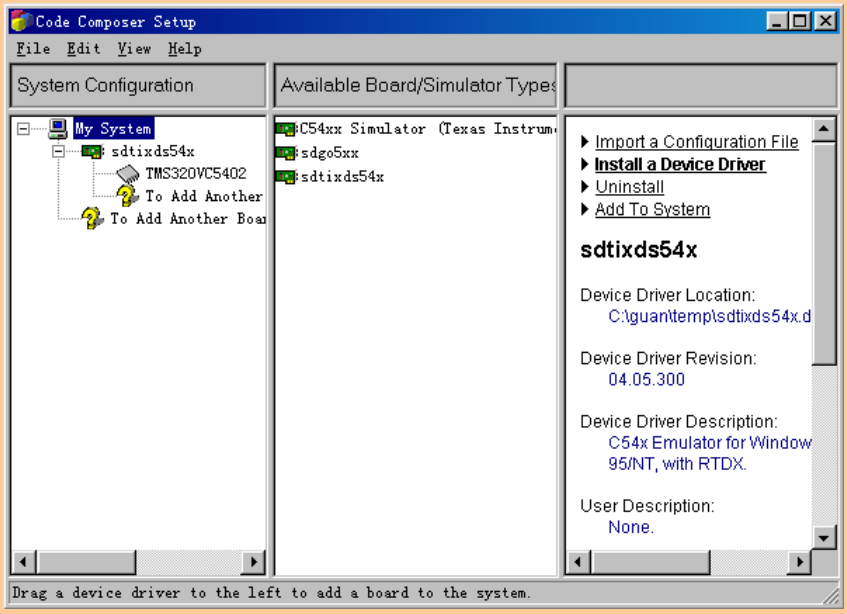
- 数据的图形显示工具，可绘制时域/频域波形，眼图，星座图，图象等，并可自动刷新（使用Animate命令运行）。
- 提供GEL工具，令用户可以编写自己的控制面板/菜单，从而方便直观地修改变量，配置参数等。
- 开放式的plug-ins技术，支持其它第三方的ActiveX插件，支持各种仿真器包括软仿真（只需安装相应的驱动程序）。

CCS包含哪些功能？

- 支持RTDX（Real Time Data eXchange）技术，利用该技术可在不中断目标系统运行的情况下，实现DSP与其他应用程序（OLE）实现数据交换。
- 提供DSP/BIOS工具，利用该工具可增强对代码的实时分析能力，如分析代码执行的效率，调度程序执行的优先级，方便管理或使用系统资源（代码/数据占用空间，中断服务程序的调用，定时器使用等等），从而减小开发人员对硬件资源熟悉程度的依赖性。

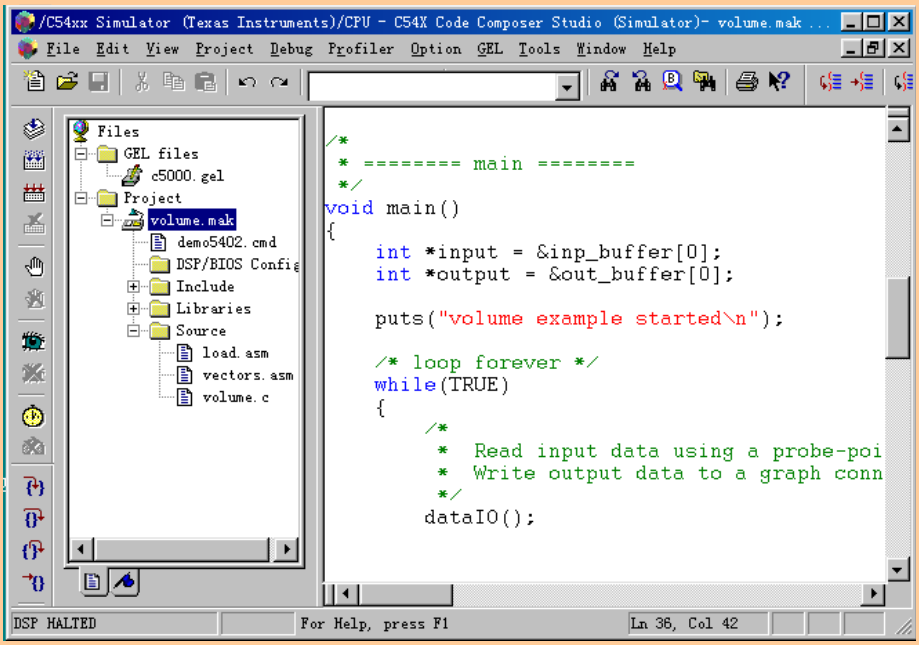
2. 为CCS安装设备驱动程序

- CCS支持软仿真器、各种型号硬仿真器、各种DSK和EVM板，你只需要向相应的生产厂家索取驱动程序，然后在CCS安装便可以使用了。
- 启动“Setup CCStudio ”应用程序安装CCS的设备驱动程序，添加设备驱动程序。然后选择一个驱动程序，将该驱动程序添加到CCS中。注意选择仿真器硬件使用的I/O端口。



3. 利用CCS开发DSP程序流程

- 在CCS环境中，你可以打开或新建工程文件，用C或汇编语言完成DSP程序代码的建立；
- 利用“build all ”命令调用代码生成工具完成编译，连接；
- 然后将生成的.out文件装入DSP的片内或外部扩展存储器，并完成调试、分析，统计或跟踪代码，确保算法的准确性、实时性和高效率。



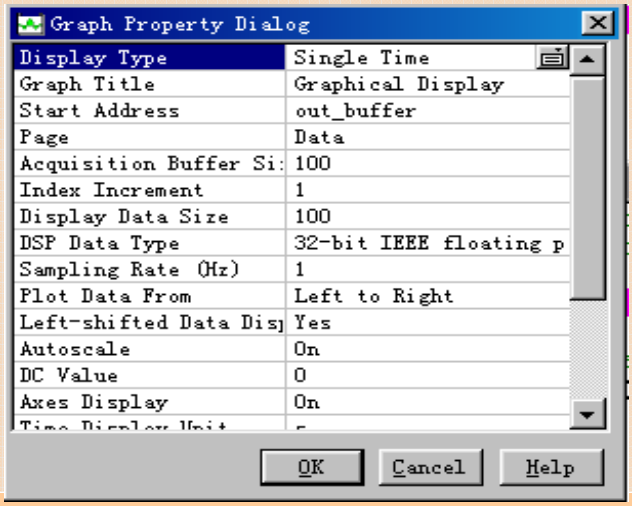
4.探针（probe points)工具使用

- 在CCS环境下可以设置探针，探针实际上是一种特殊的断点。
- 当程序运行到探针位置时，CCS中断目标系统的DSP程序的运行，然后从与该探针连接的数据文件（存放在PC机）中读出数据或输出结果，当完成数据的传输后自动恢复目标系统的DSP程序的运行。
- 探针工具特别适用于算法的仿真。探针工具可以运行在软仿真（Simulator）下。

5.图形工具的使用

- CCS提供了多种绘图工具，能将内存中的数据以各种图形方式显示，帮助用户直观地了解数据的意义。
- 需要注意的是图形窗口只有在断点时才刷新，所以应增加断点，并使用Animate方式运行。这样可以显示动画效果。
- 能提供的显示有：时域/频域波形，眼图，星坐图，图象。

设置图形显示对话框

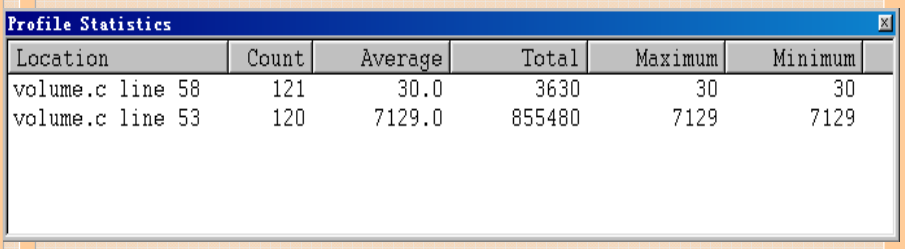


6.剖析工具（profile points）的使用

- 在CCS中，你可以利用代码剖析工具，计算代码执行了多少个机器时钟。
- 剖析工具报告的时钟个数是从前一个剖析点开始或程序起始处。
- 你可以使用鼠标右键单击某个行，选择“Toggle Profile Pt ”可以设置剖析点。然后在“Profiler ”菜单中选“Enable Clock ”。

用剖析工具分析代码执行时间

- 在启动程序运行后，你可以在“Profiler ”菜单中选“View Statistics ”，打开统计结果显示窗口。



7. DSP/BIOS的功能

- **DSP/BIOS是CCS提供的一套工具，用于支持系统实时分析。它本身仅占用极少的CPU资源。**
- 使用线程来管理程序，如硬件中断服务子程序，软件中断服务子程序，周期函数，**idle**函数等。
- 提供多种分析工具，评估代码。如图形化显示各个线程占用的CPU时间，代码执行时间统计，显示输出信息等。

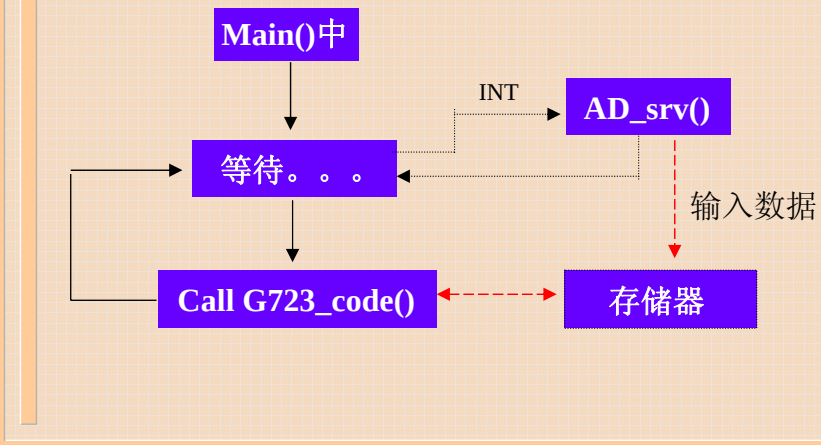
DSP/BIOS的功能

- 可以修改各个线程的优先级。
- CCS中的DSP/BIOS工具已经具有实时操作系统的很多功能特点，如任务的调度管理、任务间的同步和通讯、内存管理、实时时钟管理、中断服务管理等等。
- 需要强调的是只有‘C5000和‘C6000的CCS才带有DSP/BIOS。
- 当然，你也可以不使用DSP/BIOS工具，而使用传统的汇编和C编写DSP应用程序。

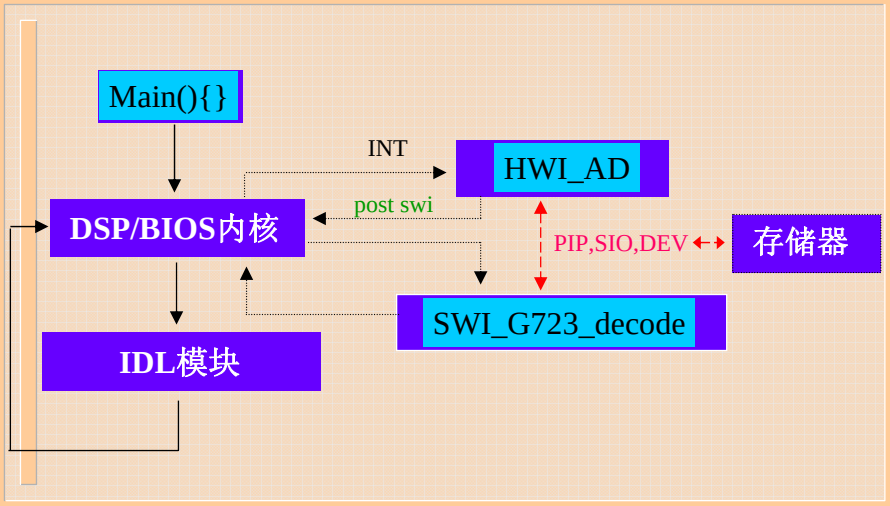
DSP/BIOS应用程序的结构

- DSP/BIOS API提供可伸缩的实时核，还提供了有优先级的多线程处理。它是专为那些需要实现实时调度、同步以及通讯的应用程序而设计。在一个包含DSP/BIOS内核的应用程序按优先级从低到高有四种主要线程：
  - 后台线程（IDL线程）
  - 任务（TSK模块）
  - 软件中断(SWIs)
  - 硬件中断(HWIs)

不使用DSP/BIOS时的程序结构



使用DSP/BIOS时的程序结构



DSP/BIOS的大小

- 对于C62x，最小需要程序存储空间的150字，数据存储空间575字。
  以C6201为例:占有的程序空间0.9%和数据空间3.5%
- 最多需要6500字，占C6201存储空间的20%。所以多数应用中是可接受的。
- DSP/BIOS的可裁减性：只把直接或间接调用的模块和API连接到目标文件中

DSP/BIOS支持的平台

- 支持C5000和C6000，C2000的DSP:
- 支持的平台：
  EVM, DSK，第三方开发的目标板，用户自己开发的板子 和 Simulator
- 配置模板(Configuration template)
  使用配置工具配置生成用户自己的.cdb文件
- Simulator 中使用DSP/BIOS片上时钟，CLK and PRD modules；Pin Connect tool配置中断管脚，仿真硬件中断。

DSP/BIOS提供的应用程序接口（API）模块

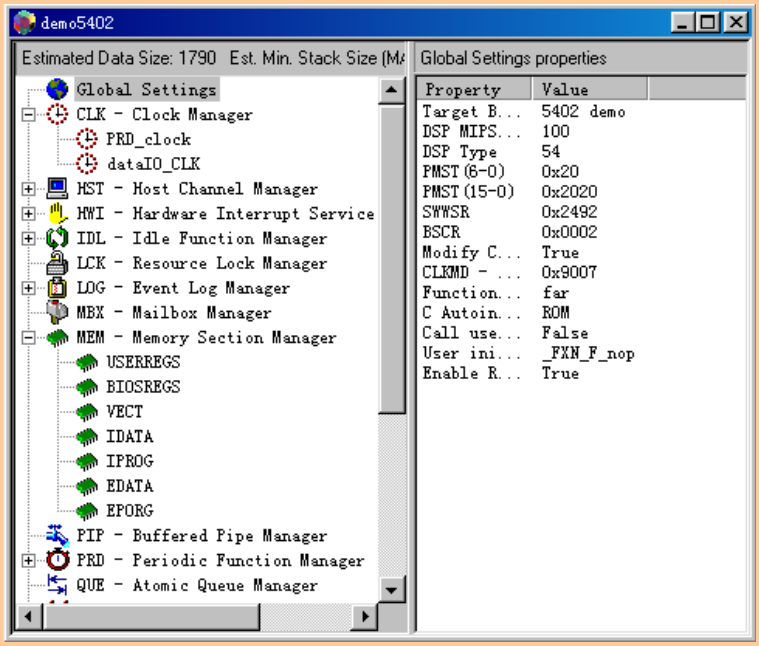
- CLK模块：用于片上的定时器管理，设置定时器中断的间隔时间。
- HST模块：用于实现主机与目标系统间数据的输入或输出。
- HWI模块：用于硬件中断管理，可设置相应的中断服务子程序。
- IDL模块：用于管理idle函数，该类函数具有最低优先级。

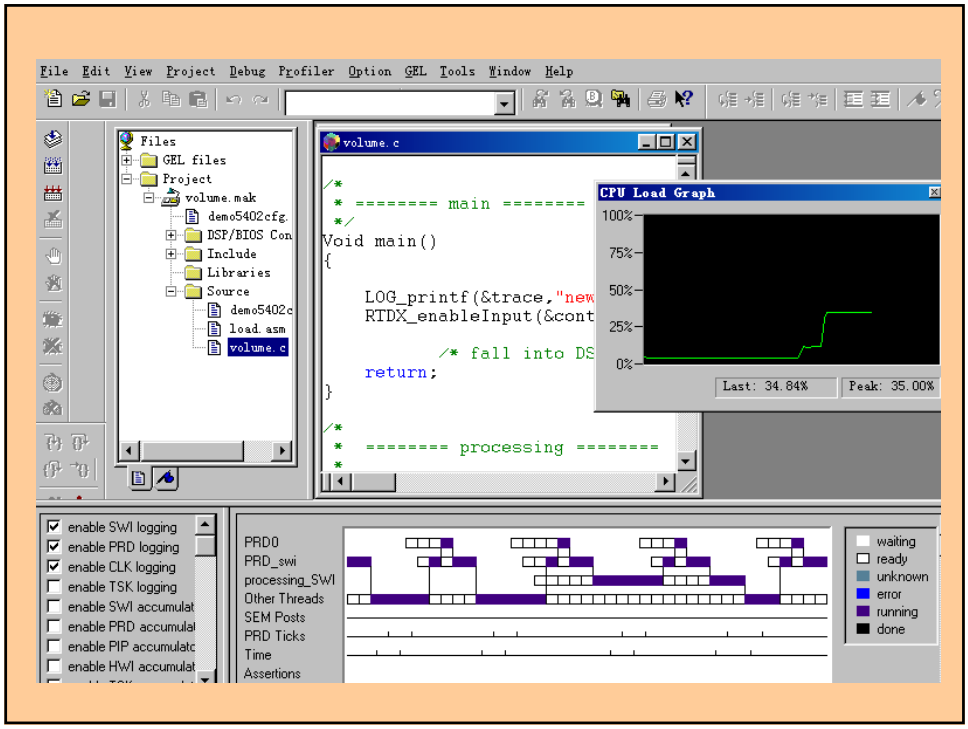
DSP/BIOS提供的模块

- LOG模块：用于事件的记录显示。例如，你可以通过该API输出调试信息。
- MEM模块：用于定义目标系统的内存使用。系统根据此信息自动产生.cmd文件。
- PIP模块：用于管道管理，可以实现线程间的数据交换。
- PRD模块：用于实现周期性的函数。该类函数的执行频率可以由CLK模块或自己调用PRD_tick决定。

DSP/BIOS提供的模块

- RTDX模块：用于主机与DSP目标系统间的实时数据传递。
- STS模块：用于状态统计管理，你可以在CCS下查看这些统计参数。
- SWI模块：用于管理软件中断。CCS将运行队列中的软件中断，并可以设置15个优先级，但都比硬件中断低。





§5-3 实时操作系统(Real Time Operation System,RTOS)

何时需要RTOS?

- 对于简单的DSP任务，用本章前面个章节的开发工具已经能满足任务的需要，往往不需要操作系统就可以进行。
- 但当DSP的任务增加和任务的复杂性提高，例如对实时性要求很高的多DSP并行操作的多任务系统，可能就需要某种操作系统来管理系统的资源，调度、安排任务的执行以及任务间的信息交换、通讯、同步。

1. 实时操作系统的功能

- 任务管理:实时操作系统一个最重要的功能就是多任务管理和基于优先级的任务调度。
- 任务间的同步和通讯:目前，主要的实时操作系统的任务间同步和通讯的机制有:消息、事件、信号量，而部分实时操作系统仍然在沿用油箱机制，另外一些实时操作系统提供了共享内存的任务间通讯机制。

实时操作系统的功能

- 内存管理:实时操作系统内存管理模式可以分为实模式和保护模式。但受DSP硬件的影响，内存管理功能都不强。主要考虑内核内存优化分配，以尽量减少整个系统的内存占有量的要求。
- 中断服务器管理:中断管理服务是操作系统的一个核心和基本功能。实时操作系统要求中断处理程序要更加短小精悍，以减少中断禁止时间和中断延迟时间。

实时操作系统的功能

- 实时时钟服务:实时时钟是系统调度的基础，也是系统定时服务器的基础一般包括定时唤醒(tm-wkafter或者tm-wkwhen)、定时事件(tm-evafter或者tm-evwhen)机制。另外部分优秀的实时操作系统提供了定时消息制度。是否提供灵活的、高精确度的定时器服务是衡量实时操作系统功能完整性的一个重要指标。

2. RTOS中的几个重要评价指标

- 任务调度机制:RTOS的实时性和多任务能力在很大程度上取决于它的任务调度机制。从调度策略上来将，分优先级调度策略和时间片轮转调度策略；从调度方式上来将，分可抢占、不可抢占、选择可抢占调度方式；从时间片看，分固定与可变时间片轮转。在大多数商用的实时系统中，大都提供了“抢占式任务调度”的功能。

RTOS中的几个重要评价指标

- 最小内存开销:在RTOS的设计中，其占用内存大小是一个很重要的指标，这是RTOS设计与其它操作系统设计的明显区别之一。
- 任务切换时间:是实时操作系统将控制权从一个任务的执行中取回，然后交给另外一个任务所需要的时间。

RTOS中的几个重要评价指标

- 中断禁止时间与中断延迟事件:当RTOS运行在核心状态或执行某些系统调用的时候，是不会因为外部中断的到来而中断执行的。只有当RTOS重新回到用户态时才响应外部中断请求，这一过程所需要的最大时间就是中断禁止时间。中断延时时间是指系统确认中断开始直到执行中断服务次序的第一条指令为止整个处理过程所需要的时间。

中断延迟时间由下列三个因素决定

- 处理器硬件电路的延迟时间，通常这个时间可以忽略
- 实时操作系统处理中断并将控制权转移给相关处理程序所需要的时间
- 实时操作系统的中断禁止时间
- 上述几项中，最大中断禁止时间和任务切换时间是评价一个RTOS实时性最重要的两个技术指标。

几种商用RTOS的简单介绍

1. OSE实时操作系统

- **OSE主要是由ENEA Data AB 下属的ENEA OSE Systems AB负责开发和技术服务的，一直以来都充当着实时操作系统以及分布式和容错性应用的先锋。公司网址：http:\\www.enea.com。**
- **OSE有四种类型的内核：普通实时操作系统内核、经过认证的实时操作系统内核、DSP的实时内核为中，小系统设计的内核。**

2. NUCLEUS实时多任务操作系统

- ATI 公司 (http:\\www.atinucleus.com) 推出的NUCLEUS+ 实时多任务操作系统以其微内核技术，原代码提供及广泛的CPU支持种类和易学易用等特点得到了国内众多用户的认可。目前，NUCLEUS+ 实时多任务操作系统在国内的通讯，医疗，控制及数据处理等领域得到了大量的应用。

3. Precise/MQX™实时操作系统

- Precise(http:\\www.psti.com) 在开发便携式嵌入式协议方面一直处于领先地位，该公司的RTOS被广泛应用在数字电话、PBXs、Xdsl、cable modems、机顶盒、PDA等领域。
- Precise一直致力于通讯协议的开发，并将大量的嵌入式通讯和网络协议集成到Precise/MQX™中。这对于开发嵌入式网络通讯产品的用户是一个好的选择。
- Precise的RTOS是免版税（royalty-free）的，并提供完整的源代码。Precise支持许多的CPU，其中包括TI的TMS320C6000 系列、TMS320C5000系列、TMS320C4X、TMS320C3X。

4. C EXECUTIVE和PSX实时操作系统

- JMI（http://www.jmi.com）开发的C EXECUTIVE是一个专为嵌入式系统设计的多任务、可ROM化的实时操作系统。它TI的TMS320C3X和TMS320C6000系列。
- 它具有快速文本切换、内核占用内存小的特性，并包括DOS兼容的文件系统、TCP/IP和SNMP等选件。
- 一般来讲，C EXECUTIVE适合较小的应用系统，增加PSX后可以满足中等大小的应用系统。以支持C6000的C EXECUTIVE为例，报价为$2500，PSX价格为$3750。C EXECUTIVE也是免版税（no-royalty）、一次性付款的RTOS，但没有提供源代码。
- 95%的C EXECUTIVE采用ANSI的标准C语言编写，而那些有严格时间要求的部分，如上下文切换、任务调度、中断处理都用优化汇编实现