DSP-算法- 精华资料 (全是 DSP 工程师遇到的问题)[转帖]

如何选择外部时钟？

DSP 的内部指令周期较高，外部晶振的主频不够，因此 DSP 大多数片内均有 PLL。但每个系列不尽相同。

1)TMS320C2000 系列：

TMS320C20x：PLL 可以÷2，×1，×2 和×4，因此外部时钟可以为 5MHz－40MHz。

TMS320F240：PLL 可以÷2，×1，×1.5，×2，×2.5，×3，×4，×4.5，×5 和×9，因此外部时钟可以为 2.22MHz－40MHz。

TMS320F241/C242/F243：PLL 可以×4，因此外部时钟为 5MHz。 TMS320LF24xx：PLL 可以由 RC 调节，因此外部时钟为 4MHz－20MHz。

TMS320LF24xxA：PLL 可以由 RC 调节，因此外部时钟为 4MHz－20MHz。

2)TMS320C3x 系列：

TMS320C3x：没有 PLL，因此外部主频为工作频率的 2 倍。

TMS320VC33：PLL 可以÷2，×1，×5，因此外部主频可以为 12MHz－100MHz。

3)TMS320C5000 系列：

TMS320VC54xx：PLL 可以÷4，÷2，×1-32，因此外部主频可以为 0.625MHz－50MHz。

TMS320VC55xx：PLL 可以÷4，÷2，×1-32，因此外部主频可以为 6.25MHz－300MHz。

4)TMS320C6000 系列：

TMS320C62xx：PLL 可以×1，×4，×6，×7，×8，×9，×10 和×11，因此外部主频可以为 11.8MHz－300MHz。

TMS320C67xx：PLL 可以×1 和×4，因此外部主频可以为 12.5MHz－230MHz。

TMS320C64xx：PLL 可以×1，×6 和×12，因此外部主频可以为 30MHz－720MHz

软件等待的如何使用？

DSP 的指令周期较快，访问慢速存储器或外设时需加入等待。等待分硬件等待和软件等待，每一个系列的等待不完全相同。

1)对于 C2000 系列： 硬件等待信号为 READY，高电平时不等待。 软件等待由 WSGR 寄存器决定，可以加入最多 7 个等待。其中程序存储器和数据存储器及 I/O 可以分别设置。

2)对于 C3x 系列： 硬件等待信号为/RDY，低电平是不等待。 软件等待由总线控制寄存器中的 SWW 和 WTCNY 决定，可以加入最多 7 个等待，但等待是不分段的，除了片内之外全空间有效。

3)对于 C5000 系列： 硬件等待信号为 READY，高电平时不等待。 软件等待由 SWWCR 和 SWWSR 寄存器决定，可以加入最多 14 个等待。其中程序存储器、控制程序存储器和数据存储器及 I/O 可以分别设置。

4)对于 C6000 系列（只限于非同步存储器或外设）： 硬件等待信号为 ARDY，高电平时不等待。 软件等待由外部存储器接口控制寄存器决定，总线访问外部存储器或设备的时序可以设置，可以方便的同异步的存储器或外设接口。

仿真工作正常对于 DSP 的基本要求

1)DSP 电源和地连接正确。 2)DSP 时钟正确。 3)DSP 的主要控制信号，如 RS 和 HOLD 信号接高电平。 4)C2000 的 watchdog 关掉。 5)不可屏蔽中断 NMI 上拉高电平。

CCS 或 Emurst 运行时提示“Can't Initialize Target DSP”

1)仿真器连接是否正常？ 2)仿真器的 I/O 设置是否正确？ 3)XDSPP 仿真器的电源是否正确？ 4)目标系统是否正确？ 5)仿真器是否正常？6)DSP 工作的基本条件是否具备。
建议使用目标板测试。

为什么 CCS 需要安装 Driver?

CCS 是开放的软件平台，它可以支持不同的硬件接口，因此不同的硬件接口必须通过标准的 Driver 同 CCS 连接。

Driver 安装的常见问题?

请认真阅读“安装手册”和 Driver 盘中的 Readme。 1)对于 SEED-XDS，安装 Readme 中的步骤，将 I/O 口设为 240/280/320/340。 2)对于 SEED-XDSPP，安装 Readme 中的步骤，将 I/O 口设为 378 或 278。 3)对于 SEED-XDSUSB，必须连接目标板，安装 Readme 中的步骤，将 I/O 口设为 A，USB 连接后，主机将自动激活相应的 Driver。 4)对于 SEED-XDSPCI，安装 Readme 中的步骤，将 I/O 口设为 240，PCI 接口板插入主机后，主机将自动激活相应的 Driver。 5)对于 Simulator，需要选择不同的 CFG 文件，以模拟不同的 DSP。 6)对于 C5402 DSK，将 I/O 口设为请认真阅读“安装手册”和 Driver 盘中的 Readme。 1)对于 SEED-XDS，安装 Readme 中的步骤，将 I/O 口设为 240/280/320/340。 2)对于 SEED-XDSPP，安装 Readme 中的步骤，将 I/O 口设为 378 或 278。注意主机 BIOS 中并口的型式必须同 xds510pp.ini 中一致。 3)对于 SEED-XDSUSB，必须连接目标板，安装 Readme 中的步骤，将 I/O 口设为 240/280/320/340，USB 连接后，主机将自动激活相应的 Driver。 4)对于 SEED-XDSPCI，安装 Readme 中的步骤，将 I/O 口设为 240/280/320/340，PCI 接口板插入主机后，主机将自动激活相应的 Driver。 5)对于 Simulator，需要选择不同的 CFG 文件，以模拟不同的 DSP。 6)对于 C5402 DSK，将 I/O 口设为 378 或 278。 7)对于 C6211/6711 DSK，将 I/O 口设为 378 或 278。 8)对于 C6201/C6701 EVM，将 I/O 口设为 0

Link 的 cmd 文件的作用是什么?

Link 的 cmd 文件用于 DSP 代码的定位。由于 DSP 的编译器的编译结果是未定位的，DSP 没有操作系统来定位执行代码，每个客户设计的 DSP 系统的配置也不尽相同，因此需要用户自己定义代码的安装位置。以 C5000 为例，基本格式为：

```
-o sample.out
-m sample.map
-stack 100
sample.obj meminit.obj
-l rts.lib
MEMORY {
PAGE 0: VECT: origin = 0xff80, length 0x80
PAGE 0: PROG: origin = 0x2000, length 0x400
PAGE 1: DATA: origin = 0x800, length 0x400
}
SECTIONS {
```

```
.vectors : {} >PROG PAGE 0
.text : {} >PROG PAGE 0
.data : {} >PROG PAGE 0
.cinit : {} >PROG PAGE 0
.bss : {} >DATA PAGE 1
}
```

如何将 OUT 文件转换为 16 进制的文件格式？

DSP 的开发软件集成了一个程序，可以从执行文件 OUT 转换到编程器可以接受的格式，使得编程器可以用次文件烧写 EPROM 或 Flash。对于 C2000 的程序为 DSPHEX；对于 C3x 程序为 HEX30；对于 C54x 程序为 HEX500；对于 C55x 程序为 HEX55；对于 C6x 程序为 Hex6x。以 C32 为例，基本格式为：

```
sample.out
-x
-memwidth 8
-bootorg 900000h
-iostrb 0h
-strb0 03f0000h
-strb1 01f0000h
-o sample.hex
ROMS {
EPROM: org = 0x900000,len=0x02000,romwidth=8
}
SECTIONS {
.text: paddr=boot
.data: paddr=boot
}
```

DSP 仿真器为什么必须连接目标系统(Target)？

DSP 的仿真器同单片机的不同，仿真器中没有 DSP，提供 IEEE 标准的 JTAG 口对 DSP 进行仿真调试，所以仿真器必须有仿真对象，及目标系统。目标系统就是你的产品，上面必须有 DSP。仿真器提供 JTAG 同目标系统的 DSP 相接，通过 DSP 实现对整个目标系统的调试。

仿真工作正常对于 DSP 的基本要求

1) DSP 电源和地连接正确。 2)DSP 时钟正确。 3)DSP 的主要控制信号，如 RS 和 HOLD 信号接高电平。 4)C2000 的 watchdog 关掉。 5)不可屏蔽中断 NMI 上拉高电平。

CCS 或 Emurst 运行时提示“Can't Initialize Target DSP”

1) 仿真器连接是否正常？ 2)仿真器的 I/O 设置是否正确？ 3)XDSPP 仿真器的电源是否正确？ 4)目标系统是否正确？ 5)仿真器是否正常？6)DSP 工作的基本条件是否具备。

建议使用目标板测试。

为什么 CCS 需要安装 Driver？

CCS 是开放的软件平台，它可以支持不同的硬件接口，因此不同的硬件接口必须通过标准的 Driver 同 CCS 连接。

Link 的 cmd 文件的作用是什么？

Link 的 cmd 文件用于 DSP 代码的定位。由于 DSP 的编译器的编译结果是未定位的，DSP 没有操作系统来定位执行代码，每个客户设计的 DSP 系统的配置也不尽相同，因此需要用户自己定义代码的安装位置。以 C5000 为例，基本格式为：

```
-o sample.out
-m sample.map
-stack 100
sample.obj meminit.obj
-l rts.lib
MEMORY {
PAGE 0: VECT: origin = 0xff80, length 0x80
PAGE 0: PROG: origin = 0x2000, length 0x400
PAGE 1: DATA: origin = 0x800, length 0x400
}
SECTIONS {
.vectors : {} >PROG PAGE 0
.text : {} >PROG PAGE 0
.data : {} >PROG PAGE 0
.cinit : {} >PROG PAGE 0
.bss : {} >DATA PAGE 1
}
```

如何将 OUT 文件转换为 16 进制的文件格式？

DSP 的开发软件集成了一个程序，可以从执行文件 OUT 转换到编程器可以接受的格式，使得编程器可以用次文件烧写 EPROM 或 Flash。对于 C2000 的程序为 DSPHEX；对于 C3x 程序为 HEX30；对于 C54x 程序为 HEX500；对于 C55x 程序为 HEX55；对于 C6x 程序为 Hex6x。以 C32 为例，基本格式为：

```
sample.out
-x
-memwidth 8
-bootorg 900000h
-iostrb 0h
-strb0 03f0000h
-strb1 01f0000h
-o sample.hex
ROMS {
EPROM: org = 0x900000,len=0x02000,romwidth=8
}
SECTIONS {
.text: paddr=boot
.data: paddr=boot
```

}

DSP 的 C 语言同主机 C 语言的主要区别？

1) DSP 的 C 语言是标准的 ANSI C，它不包括同外设联系的扩展部分，如屏幕绘图等。但在 CCS 中，为了方便调试，可以将数据通过 prinf 命令虚拟输出到主机的屏幕上。 2)DSP 的 C 语言的编译过程为，C 编译为 ASM，再由 ASM 编译为 OBJ。因此 C 和 ASM 的对应关系非常明确，非常便于人工优化。 3)DSP 的代码需要绝对定位；主机的 C 的代码有操作系统定位。 4)DSP 的 C 的效率较高，非常适合于嵌入系统。

为什么在 CCS 下编译工具工作不正常？

在 CCS 下有部分客户会碰到编译工具工作不正常，常见错误为： 1)autoexec.bat 的路径“out of memory”。修改 autoexec.bat，清除无用的 PATH 路径。 2)编译的输出文件（OUT 文件）写保护，无法覆盖。删除或修改输出文件的属性。 3)Windows 有问题。重新安装 windows。 4)Windows 下有程序对 CCS 有影响。建议用一“干净”的计算机。

在 CCS 下，如何选择有效的存储器空间？

CCS 下的存储器空间最好设置同你的硬件，没有的存储器不要有效。这样便于调试，CCS 会发现你调入程序时或程序运行时，是否访问了无效地址。 1)在 GEL 文件中设置。参见 CCS 中的示例。 2)在 Option 菜单下，选择 Memory Map 选项，根据你的硬件设置。注意一定要将 Enable Memory Mapping 置为使能。

在 CCS 下，OUT 文件加载时提示“Data verification failed...”的原因？

Link 的 CMD 文件分配的地址同 GEL 或设置的有效地址空间不符。中断向量定位处或其它代码、数据段定位处，没有 RAM，无法加载 OUT 文件。解决方法： 1)调整 Link 的 CMD 文件，使得定位段处有 RAM。 2)调整存储器设置，使得 RAM 区有效。

为什么要使用 BIOS？

1)BIOS 是 Basic I/O System 的简称，是基本的输入、输出管理。 2)用于管理任务的调度，程序实时分析，中断管理，跟踪管理和实时数据交换。 3)BIOS 是基本的实时系统，使用 BIOS 可以方便地实现多任务、多进程的时间管理。 4)BIOS 是 eXpress DSP 的标准平台，要使用 eXpress DSP 技术，必须使用 BIOS。

DSP 发展动态

1.TMS320C2000 TMS320C2000 系列包括 C24x 和 C28x 系列。C24x 系列建议使用 LF24xx 系列替代 C24x 系列，LF24xx 系列的价格比 C24x 便宜，性能高于 C24x，而且 LF24xxA 具有加密功能。 C28x 系列主要用于大存储设备管理，高性能的控制场合。

2.TMS320C3x TMS320C3x 系列包括 C3x 和 VC33，主要推荐使用 VC33。C3x 系列是 TI 浮点 DSP 的基础，不可能停产，但价格不会进一步下调。

3.TMS320C5x TMS320C5x 系列已不推荐使用，建议使用 C24x 或 C5000 系列替代。

4.TMS320C5000 TMS320C5000 系列包括 C54x 和 C55x 系列。 其中 VC54xx 还不断有新的器件出现，如：TMS320VC5471（DSP＋ARM7）。 C55x 系列是 TI 的第三代 DSP，功耗为 VC54xx 的 1/6，性能为 VC54xx 的 5 倍，是一个正在发展的系列。 C5000 系列是目前 TI DSP 的主流 DSP，它涵盖了从低档到中高档的应用领域，目前也是用户最多的系列。

5.TMS320C6000 TMS320C6000 系列包括 C62xx、C67xx 和 C64xx。此系列是 TI 的高档 DSP 系列。 其中 C62xx 系列是定点的 DSP，系列芯片种类较丰富，是主要的应用系列。 C67xx 系列是浮点的 DSP，用于需

要高速浮点处理的领域。 C64xx 系列是新发展，性能是 C62xx 的 10 倍。
6.OMAP 系列 是 TI 专门用于多媒体领域的芯片，它是 C55＋ARM9，性能卓越，非常适合于手持设备、Internet 终端等多媒体应用。

5V/3.3V 如何混接？
TI DSP 的发展同集成电路的发展一样，新的 DSP 都是 3.3V 的，但目前还有许多外围电路是 5V 的，因此在 DSP 系统中，经常有 5V 和 3.3V 的 DSP 混接问题。在这些系统中，应注意： 1)DSP 输出给 5V 的电路（如 D/A），无需加任何缓冲电路，可以直接连接。 2)DSP 输入 5V 的信号（如 A/D），由于输入信号的电压>4V，超过了 DSP 的电源电压，DSP 的外部信号没有保护电路，需要加缓冲，如 74LVC245 等，将 5V 信号变换成 3.3V 的信号。 3)仿真器的 JTAG 口的信号也必须为 3.3V，否则有可能损坏 DSP。

为什么要片内 RAM 大的 DSP 效率高？
目前 DSP 发展的片内存储器 RAM 越来越大，要设计高效的 DSP 系统，就应该选择片内 RAM 较大的 DSP。片内 RAM 同片外存储器相比，有以下优点： 1)片内 RAM 的速度较快，可以保证 DSP 无等待运行。 2)对于 C2000/C3x/C5000 系列，部分片内存储器可以在一个指令周期内访问两次，使得指令可以更加高效。3)片内 RAM 运行稳定，不受外部的干扰影响，也不会干扰外部。 4)DSP 片内多总线，在访问片内 RAM 时，不会影响其它总线的访问，效率较高。

为什么 DSP 从 5V 发展成 3.3V？
超大规模集成电路的发展从 1um，发展到目前的 0.1um，芯片的电源电压也随之降低，功耗也随之降低。DSP 也同样从 5V 发展到目前的 3.3V，核心电压发展到 1V。目前主流的 DSP 的外围均已发展为 3.3V，5V 的 DSP 的价格和功耗都价格，以逐渐被 3.3V 的 DSP 取代。

如何选择 DSP 的电源芯片？
TMS320LF24xx：TPS7333QD，5V 变 3.3V，最大 500mA。
TMS320VC33： TPS73HD318PWP，5V 变 3.3V 和 1.8V，最大 750mA。
TMS320VC54xx：TPS73HD318PWP，5V 变 3.3V 和 1.8V，最大 750mA； TPS73HD301PWP，5V 变 3.3V 和可调，最大 750mA。
TMS320VC55xx：TPS73HD301PWP，5V 变 3.3V 和可调，最大 750mA。
TMS320C6000： PT6931，TPS56000，最大 3A。

软件等待的如何使用？
DSP 的指令周期较快，访问慢速存储器或外设时需加入等待。等待分硬件等待和软件等待，每一个系列的等待不完全相同。
1)对于 C2000 系列： 硬件等待信号为 READY，高电平时不等待。 软件等待由 WSGR 寄存器决定，可以加入最多 7 个等待。其中程序存储器和数据存储器及 I/O 可以分别设置。
2)对于 C3x 系列： 硬件等待信号为/RDY，低电平是不等待。 软件等待由总线控制寄存器中的 SWW 和 WTCNY 决定，可以加入最多 7 个等待，但等待是不分段的，除了片内之外全空间有效。
3)对于 C5000 系列： 硬件等待信号为 READY，高电平时不等待。 软件等待由 SWWCR 和 SWWSR 寄存器决定，可以加入最多 14 个等待。其中程序存储器、控制程序存储器和数据存储器及 I/O 可以分别设置。
4)对于 C6000 系列（只限于非同步存储器或外设）： 硬件等待信号为 ARDY，高电平时不等待。 软件等待由外部存储器接口控制寄存器决定，总线访问外部存储器或设备的时序可以设置，可以方便的同异步的存储器或外设接口。

中断向量为什么要重定位？

为了方便 DSP 存储器的配置，一般 DSP 的中断向量可以重新定位，即可以通过设置寄存器放在存储器空间的任何地方。 注意：C2000 的中断向量不能重定位。

DSP 的最高主频能从芯片型号中获得吗？

TI 的 DSP 最高主频可以从芯片的型号中获得，但每一个系列不一定相同。

1)TMS320C2000 系列：

TMS320F206－最高主频 20MHz。

TMS320C203/C206－最高主频 40MHz。

TMS320F24x－最高主频 20MHz。

TMS320LF24xx－最高主频 30MHz。

TMS320LF24xxA－最高主频 40MHz。

TMS320LF28xx－最高主频 150MHz。

2)TMS320C3x 系列：

TMS320C30：最高主频 25MHz。

TMS320C31PQL80：最高主频 40MHz。

TMS320C32PCM60：最高主频 30MHz。

TMS320VC33PGE150：最高主频 75MHz。

3)TMS320C5000 系列：

TMS320VC54xx：最高主频 160MHz。

TMS320VC55xx：最高主频 300MHz。

4)TMS320C6000 系列：

TMS320C62xx：最高主频 300MHz。

TMS320C67xx：最高主频 230MHz。

TMS320C64xx：最高主频 720MHz。

DSP 可以降频使用吗？

可以，DSP 的主频均有一定的工作范围，因此 DSP 均可以降频使用。

如何选择外部时钟？

DSP 的内部指令周期较高，外部晶振的主频不够，因此 DSP 大多数片内均有 PLL。但每个系列不尽相同。

1)TMS320C2000 系列：

TMS320C20x：PLL 可以÷2，×1，×2 和×4，因此外部时钟可以为 5MHz－40MHz。

TMS320F240：PLL 可以÷2，×1，×1.5，×2，×2.5，×3，×4，×4.5，×5 和×9，因此外部时钟可以为 2.22MHz－40MHz。

TMS320F241/C242/F243：PLL 可以×4，因此外部时钟为 5MHz。 TMS320LF24xx：PLL 可以由 RC 调节，因此外部时钟为 4MHz－20MHz。

TMS320LF24xxA：PLL 可以由 RC 调节，因此外部时钟为 4MHz－20MHz。

2)TMS320C3x 系列：

TMS320C3x：没有 PLL，因此外部主频为工作频率的 2 倍。

TMS320VC33：PLL 可以÷2，×1，×5，因此外部主频可以为 12MHz－100MHz。

3)TMS320C5000 系列：

TMS320VC54xx：PLL 可以÷4，÷2，×1-32，因此外部主频可以为 0.625MHz－50MHz。

TMS320VC55xx：PLL 可以÷4，÷2，×1-32，因此外部主频可以为 6.25MHz－300MHz。

4)TMS320C6000 系列：

TMS320C62xx：PLL 可以×1，×4，×6，×7，×8，×9，×10 和×11，因此外部主频可以为 11.8MHz－300 MHz。

TMS320C67xx：PLL 可以×1 和×4，因此外部主频可以为 12.5MHz－230MHz。

TMS320C64xx：PLL 可以×1，×6 和×12，因此外部主频可以为 30MHz－720MHz

如何选择 DSP 的外部存储器？

DSP 的速度较快，为了保证 DSP 的运行速度，外部存储器需要具有一定的速度，否则 DSP 访问外部存储器时需要加入等待周期。

1)对于 C2000 系列： C2000 系列只能同异步的存储器直接相接。 C2000 系列的 DSP 目前的最高速度为 150MHz。建议可以用的存储器有：

CY7C199-15：32K×8，15ns，5V；

CY7C1021-12：64K×16，15ns，5V； CY7C1021V33-12：64K×16，15ns，3.3V。

2)对于 C3x 系列： C3x 系列只能同异步的存储器直接相接。 C3x 系列的 DSP 的最高速度，5V 的为 40MHz，3.3V 的为 75MHz，为保证 DSP 无等待运行，分别需要外部存储器的速度<25ns 和<12ns。建议可以用的存储器有：

ROM： AM29F400-70：256K×16，70ns，5V，加入一个等待；

AM29LV400-55(SST39VF400)：256K×16，55ns，3.3V，加入两个等待（目前没有更快的 Flash）。

SRAM： CY7C199-15：32K×8，15ns，5V；

CY7C1021-15：64K×16，15ns，5V；

CY7C1009-15：128K×8，15ns，5V；

CY7C1049-15：512K×8，15ns，5V；

CY7C1021V33-15：64K×16，15ns，3.3V；

CY7C1009V33-15：128K×8，15ns，3.3V；

CY7C1041V33-15：256k×16，15ns，3.3V。

3)对于 C54x 系列： C54x 系列只能同异步的存储器直接相接。 C54x 系列的 DSP 的速度为 100MHz 或 160MHz，为保证 DSP 无等待运行，需要外部存储器的速度<10ns 或<6ns。建议可以用的存储器有：

ROM： AM29LV400-55(SST39VF400)：256K×16，55ns，3.3V，加入 5 或 9 个等待（目前没有更快的 Flash）。

SRAM： CY7C1021V33-12：64K×16，12ns，3.3V，加入一个等待；

CY7C1009V33-12：128K×8，12ns，3.3V，加入一个等待。

4)对于 C55x 和 C6000 系列： TI 的 DSP 中只有 C55x 和 C6000 可以同同步的存储器相连，同步存储器可以保证系统的数据交换效率更高。

ROM： AM29LV400-55(SST39VF400)：256K×16，55ns，3.3V。

SDRAM： HY57V651620BTC-10S：64M，10ns。

SBSRAM： CY7C1329-133AC，64k×32；

CY7C1339-133AC，128k×32。

FIFO：CY7C42x5V-10ASC，32k/64k×18。

DSP 芯片有多大的驱动能力？

DSP 的驱动能力较强，可以不加驱动，连接 8 个以上标准 TTL 门。

调试 TMS320C2000 系列的常见问题？

1)单步可以运行，连续运行时总回 0 地址： Watchdog 没有关，连续运行复位 DSP 回到 0 地址。
2)OUT 文件不能 load 到片内 flash 中： Flash 不是 RAM，不能用简单的写指令写入，需要专门的程序写入。CCS 和 C Source Debugger 中的 load 命令，不能对 flash 写入。 OUT 文件只能 load 到片内 RAM，或片外 RAM 中。
3)在 flash 中如何加入断点： 在 flash 中可以用单步调试，也可以用硬件断点的方法在 flash 中加入断点，软件断点是不能加在 ROM 中的。硬件断点，设置存储器的地址，当访问该地址时产生中断。
4)中断向量： C2000 的中断向量不可重定位，因此中断向量必须放在 0 地址开始的 flash 内。在调试系统时，代码放在 RAM 中，中断向量也必须放在 flash 内。

调试 TMS320C3x 系列的常见问题？
1) TMS320C32 的存储器配置： TMS320C32 的程序存储器可以配置为 16 位或 32 位；数据存储器可以配置为 8 位、16 位或 32 位。
2)TMS320VC33 的 PLL 控制： TMS320VC33 的 PLL 控制端只能接 1.8V，不能接 3.3V 或 5V。

如何调试多片 DSP？
对于有 MPSD 仿真口的 DSP（TMS320C30/C31/C32），不能用一套仿真器同时调试，每次只能调试其中的一个 DSP； 对于有 JTAG 仿真口的 DSP，可以将 JTAG 串接在一起，用一套仿真器同时调试多个 DSP，每个 DSP 可以用不同的名字，在不同的窗口中调试。 注意：如果在 JTAG 和 DSP 间加入驱动，一定要用快速的门电路，不能使用如 LS 的慢速门电路。

在 DSP 系统中为什么要使用 CPLD？
DSP 的速度较快，要求译码的速度也必须较快。利用小规模逻辑器件译码的方式，已不能满足 DSP 系统的要求。 同时，DSP 系统中也经常需要外部快速部件的配合，这些部件往往是专门的电路，有可编程器件实现。 CPLD 的时序严格，速度较快，可编程性好，非常适合于实现译码和专门电路。

DSP 系统构成的常用芯片有哪些？
1) 电源： TPS73HD3xx，TPS7333，TPS56100，PT64xx...
2)Flash： AM29F400，AM29LV400，SST39VF400...
3)SRAM： CY7C1021，CY7C1009，CY7C1049...
4)FIFO: CY7C425，CY7C42x5...
5)Dual port： CY7C136，CY7C133，CY7C1342...
6)SBSRAM： CY7C1329，CY7C1339...
7)SDRAM： HY57V651620BTC...
8)CPLD： CY37000 系列，CY38000 系列，CY39000 系列...
9)PCI： PCI2040，CY7C09449...
10)USB： AN21xx，CY7C68xxx...
11)Codec：TLV320AIC23，TLV320AIC10...
12)A/D,D/A：ADS7805，TLV2543...
具体资料见 www.ti.com  http://www.cypress.com/

什么是 boot loader？
DSP 的速度尽快，EPROM 或 flash 的速度较慢，而 DSP 片内的 RAM 很快，片外的 RAM 也较快。为了使 DSP 充分发挥它的能力，必须将程序代码放在 RAM 中运行。为了方便的将代码从 ROM 中搬到 RAM 中，在

不带 flash 的 DSP 中，TI 在出厂时固化了一段程序，在上电后完成从 ROM 或外设将代码搬到用户指定的 RAM 中。此段程序称为“boot loader”。

TMS320C3x 如何 boot？
在 MC/MP 管脚为高时，C3x 进入 boot 状态。C3x 的 boot loader 在 reset 时，判断外部中断管脚的电平。根据中断配置决定 boot 的方式为存储器加载还是串口加载，其中 ROM 的地址可以为三个中的一个，ROM 可以为 8 位。

Boot 有问题如何解决？
1)仔细检查 boot 的控制字是否正确。 2)仔细检查外部管脚设置是否正确。 3)仔细检查 hex 文件是否转换正确。 4)用仿真器跟踪 boot 过程，分析错误原因。

DSP 为什么要初始化？
DSP 在 RESET 后，许多的寄存器的初值一般同用户的要求不一致，例如：等待寄存器，SP，中断定位寄存器等，需要通过初始化程序设置为用户要求的数值。 初始化程序的主要作用： 1)设置寄存器初值。 2)建立中断向量表。 3)外围部件初始化。

DSP 有哪些数学库及其它应用软件？
TI 公司为了方便客户开发 DSP，在它的网站上提供了许多程序的示例和应用程序，如 MATH 库，FFT，FIR/IIR 等，可以在 TI 的网页免费下载。

如何获得 DSP 专用算法？
TI 有许多的 Third Party 可以通过 DSP 上的多种算法软件。可以通过 TI 的网页搜索你所需的算法，找到通过算法的公司，同相应的公司联系。注意这些算法都是要付费的。

eXpressDSP 是什么？
eXpressDSP 是一种实时 DSP 软件技术，它是一种 DSP 编程的标准，利用它可以加快你开发 DSP 软件的速度。 以往 DSP 软件的开发没有任何标准，不同的人写的程序一般无法连接在一起。DSP 软件的调试工具也非常不方便。使得 DSP 软件的开发往往滞后于硬件的开发。 eXpressDSP 集成了 CCS(Code Composer Studio)开发平台，DSP BIOS 实时软件平台，DSP 算法标准和第三方支持四部分。利用该技术，可以使你的软件调试，软件进程管理，软件的互通及算法的获得，都便的容易。这样就可以加快你的软件开发进程。

1)CCS 是 eXpressDSP 的基础，因此你必须首先拥有 CCS 软件。
2)DSP BIOS 是 eXpressDSP 的基本平台，你必须学会所有 DSP BIOS。
3)DSP 算法标准可以保证你的程序可以方便的同其它利用 eXpressDSP 技术的程序连接在一起。同时也保证你的程序的延续性。

为什么要用 DSP？
3G 技术和 internate 的发展，要求处理器的速度越来越高，体积越来越小，DSP 的发展正好能满足这一发展的要求。因为，传统的其它处理器都有不同的缺陷。MCU 的速度较慢；CPU 体积较大，功耗较高；嵌入 CPU 的成本较高。 DSP 的发展，使得在许多速度要求较高，算法较复杂的场合，取代 MCU 或其它处理器，而成本有可能更低。

如何选择 DSP？

选择 DSP 可以根据以下几方面决定：

1)速度： DSP 速度一般用 MIPS 或 FLOPS 表示，即百万次/秒钟。根据您对处理速度的要求选择适合的器件。一般选择处理速度不要过高，速度高的 DSP，系统实现也较困难。

2)精度： DSP 芯片分为定点、浮点处理器，对于运算精度要求很高的处理，可选择浮点处理器。定点处理器也可完成浮点运算，但精度和速度会有影响。

3)寻址空间： 不同系列 DSP 程序、数据、I/O 空间大小不一，与普通 MCU 不同，DSP 在一个指令周期内能完成多个操作，所以 DSP 的指令效率很高，程序空间一般不会有问题，关键是数据空间是否满足。数据空间的大小可以通过 DMA 的帮助，借助程序空间扩大。

4)成本： 一般定点 DSP 的成本会比浮点 DSP 的要低，速度也较快。要获得低成本的 DSP 系统，尽量用定点算法，用定点 DSP。

5)实现方便： 浮点 DSP 的结构实现 DSP 系统较容易，不用考虑寻址空间的问题，指令对 C 语言支持的效率也较高。

6)内部部件：根据应用要求，选择具有特殊部件的 DSP。如：C2000 适合于电机控制；OMAP 适合于多媒体等。

要了解 DSP 芯片的性能，本网中的"DSP 及相关器件"中有介绍。

DSP 同 MCU 相比的特点？

1) DSP 的速度比 MCU 快，主频较高。

2)DSP 适合于数据处理，数据处理的指令效率较高。

3)DSP 均为 16 位以上的处理器，不适合于低档的场合。

4)DSP 可以同时处理的事件较多，系统级成本有可能较低。

5)DSP 的灵活性较好，大多数算法都可以软件实现。

6)DSP 的集成度较高，可靠性较好。

DSP 同嵌入 CPU 相比的特点？

1) DSP 是单片机，构成系统简单。 2)DSP 的速度快。 3)DSP 的成本较低。 4)DSP 的性能高，可以处理较多的任务。

如何编写 C2000 片内 Flash？

DSP 中的 Flash 的编写方法有三中：

1.通过仿真器编写：在我们的网页上有相关的软件，在销售仿真器时我们也提供相关软件。其中 LF240x 的编写可以在 CCS 中加入一个插件，F24x 的编写需要在 windows98 下的 DOS 窗中进行。具体步骤见软件中的 readme。有几点需要注意： a.必须为 MC 方式； b.F206 的工作频率必须为 20MHz； c.F240 需要根据 PLL 修改 C240_CFG.I 文件。建议外部时钟为 20MHz。 d.LF240x 也需要根据 PLL 修改文件。 d.如果编写有问题，可以用 BFLWx.BAT 修复。

2.提供串口编写：TI 的网页上有相关软件。注意只能编写一次，因为编写程序会破坏串口通信程序。

3.在你的程序中编写：TI 的网页上有相关资料。

如何编写 DSP 外部的 Flash？

DSP 的外部 Flash 编写方法：

1.通过编程器编写：将 OUT 文件通过 HEX 转换程序转换为编程器可以接受的格式，再由编程器编写。

2.通过 DSP 软件编写：您需要根据 Flash 的说明，编写 Flash 的编写程序，将应用程序和编写 Flash 的程序分别 load 到 RAM 中，运行编写程序编写。

对于 C5000，大于 48K 的程序如何 BOOT？

对于 C5000，片内的 BOOT 程序在上电后将数据区的内容，搬移到程序区的 RAM 中，因此 FLASH 必须在 RESET 后放在数据区。由于 C5000，数据区的空间有限，一次 BOOT 的程序不能对于 48K。解决的方法如下：

1.在 RESET 后，将 FLASH 译码在数据区，RAM 放在程序区，片内 BOOT 程序将程序 BOOT 到 RAM 中。

2.用户初试化程序发出一个 I/O 命令（如 XF），将 FLASH 译码到程序区的高地址。开放数据区用于其它的 RAM。

3.用户初试化程序中包括第二次 BOOT 程序（此程序必须用户自己编写），将 FLASH 中没有 BOOT 的其它代码搬移到 RAM 中。

4.开始运行用户处理程序。

DSP 外接存储器的控制方式

对于一般的存储器具有 RD、WR 和 CS 等控制信号，许多 DSP（C3x、C5000）都没有控制信号直接连接存储器，一般采用的方式如下：

1.CS 有地址线和 PS、DS 或 STRB 译码产生；

2./RD=/STRB+/R/W； 3./WR=/STRB+R/W。

GEL 文件的功能？

GEL 文件的功能同 emuinit.cmd 的功能基本相同，用于初始化 DSP。但它的功能比 emuinit 的功能有所增强，GEL 在 CCS 下有一个菜单，可以根据 DSP 的对象不同，设置不同的初始化程序。以 TMS320LF2407 为例：

```
#define SCSR1 0x7018 ； 定义 scsr1 寄存器
#define SCSR2 0X7019 ； 定义 scsr2 寄存器
#define WDKEY 0x7025 ； 定义 wdkey 寄存器
#define WDNTR 0x7029 ； 定义 wdntr 寄存器
StartUp() ； 开始函数
{
GEL_MapReset(); ； 存储空间复位 GEL_MapAdd(0x0000,0,0x7fff,1,1); 定义程序空间从 0000－7fff 可读写
GEL_MapAdd(0x8000,0,0x7000,1,1); 定义程序空间从 8000－f000 可读写
GEL_MapAdd(0x0000,1,0x10000,1,1); 定义数据空间从 0000－10000 可读写
GEL_MapAdd(0xffff,2,1,1,1); 定义 i/o 空间 0xffff 可读写
GEL_MapOn(); 存储空间打开
GEL_MemoryFill(0xffff,2,1,0x40); 在 i/o 空间添入数值 40h
*(int *)SCSR1=0x0200; 给 scsr1 寄存器赋值
*(int *)SCSR2=0x000C; 给 scsr2 寄存器赋值，在这里可以进行 mp/mc 方式的转换
*(int *)WDNTR=0x006f; 给 wdntr 寄存器赋值
*(int *)WDKEY=0x055; 给 wdkey 寄存器赋值
*(int *)WDKEY=0x0AA; 给 wdkey 寄存器赋值
```

使用 TI 公司模拟器件与 DSP 结合使用的好处。

1) 在使用 TI 公司的 DSP 的同时，使用 TI 公司的模拟可以和 DSP 进行无缝连接。器件与器件之间不需要

任何的连接或转接器件。这样即减少了板卡的尺寸，也降低了开发难度。
2)同为 TI 公司的产品，很多器件可以固定搭配使用。少了器件选型的烦恼
3)TI 在 CCS 中提供插件，可以用于 DSP 和模拟器件的开发，非常方便。

C 语言中可以嵌套汇编语言？
可以。在 ANSI C 标准中的标准用法就是用 C 语言编写主程序，用汇编语言编写子程序，中断服务程序，一些算法，然后用 C 语言调用这些汇编程序，这样效率会相对比较高

在定点 DSP 系统中可否实现浮点运算
当然可以，因为 DSP 都可以用 C,只要是可以使用 c 语言的场合都可以实现浮点运算。

JTAG 头的使用会遇到哪些情况
1) DSP 的 CLKOUT 没有输出，工作不正常。
2)Emu0，Emu1 需要上拉。
3)TCK 的频率应该为 10M。
4)在 3.3V DSP 中，PD 脚为 3.3V 供电，但是仿真器上需要 5V 电压供电，所以 PP 仿真器盒上需要单独供电。
4)仿真多片 DSP。在使用菊花链的时候，第一片 DSP 的 TDO 接到第二片 DSP 的 TDI 即可。注意当串联 DSP 比较多的时候，信号线要适当的增加驱动。

include 头文件（.h)的主要作用
头文件，一般用于定义程序中的函数、参数、变量和一些宏单元，同库函数配合使用。因此，在使用库时，必须用相应的头文件说明。

DSP 中断向量的位置
1) 2000 系列 dsp 的中断向量只能从 0000H 处开始。所以在我们调试程序的时候，要把 DSP 选择为 MP（微处理器方式），把片内的 Flash 屏蔽掉，免去每次更改程序都要重新烧写 Flash 工作。
2)3x 系列 dsp 的中断向量也只能在固定的地址。
3)5000，6000 系列 dsp 的中断向量可以重新定位。但是它只能被重新定位到 Page0 范围内的任何空间。

有源晶振与晶体的区别，应用范围及用法
1) 晶体需要用 DSP 片内的振荡器，在 datasheet 上有建议的连接方法。晶体没有电压的问题，可以适应于任何 DSP，建议用晶体。 2)有源晶振不需要 DSP 的内部振荡器，信号比较稳定。有源晶振用法：一脚悬空，二脚接地，三脚接输出，四脚接电压。

程序经常跑飞的原因
1) 程序没有结尾或不是循环的程序。
2)nmi 管脚没有上拉。
3)在看门狗动作的时候程序会经常跑飞。
4)程序编制不当也会引起程序跑飞。
5)硬件系统有问题。

并行 FLASH 引导的一点经验-阿哲
最近 BBS 上关于 FLASH 和 BOOT 的讨论很活跃，我也多次来此请教。前几天自制的 DSP 板引导成功，早

就打算写写这方面的东西。我用的 DSP 是 5416，以其为核心，做了一个相对独立的子系统（硬件、软件、算法），目前都已基本做好。 下面把在 FLASH 引导方面做的工作向大家汇报一下，希望能对大家有所帮助。本人经验和文笔都有限，写的不好请大家谅解。 硬件环境：

DSP：TMS320VC5416PGE160

FLASH：SST39VF400A-70-4C-EK 都是贴片的，FLASH 映射在 DSP 数据空间的 0x8000-0xFFFF

软件环境： CCS v2.12.01

主程序（要烧入 FLASH 的程序）： DEBUG 版，程序占用空间 0x28000-0x2FFFF（片内 SARAM），中断向量表在 0x0080-0x00FF（片内 DARAM），数据空间使用 0x0100-0x7FFF（片内 DARAM）。 因为 FLASH 是贴片的，所以需要自己编一个数据搬移程序，把要主程序搬移到 FLASH 中。在写入 FLASH 数据时，还应写入引导表的格式数据。最后在数据空间的 0xFFFF 处写入引导表的起始地址（这里为 0x8000）。

搬移程序： DEBUG 版，程序空间 0x38000-0x3FFFF（片内 SARAM），中断向量表在 0x7800-0x78FF（片内 DARAM），数据空间使用 0x5000-0x77FF（片内 DARAM）。 搬移程序不能使用与主程序的程序空间和中断向量表重合的物理空间，以免覆盖。 烧写时，同时打开主程序和搬移程序的 PROJECT，先 LOAD 主程序，再 LOAD 搬移程序，然后执行搬移程序，烧写 OK! 附：搬移程序（仅供参考）

```
volatile unsigned int *pTemp=(unsigned int *)0x7e00; unsigned int iFlashAddr;
int iLoop; /* 在引导表头存放并行引导关键字 */
iFlashAddr=0x8000;
WriteFlash(iFlashAddr,0x10aa);
iFlashAddr++; /* 初始化 SWWSR 值 */
WriteFlash(iFlashAddr,0x7e00);
iFlashAddr++; /* 初始化 BSCR 值 */
WriteFlash(iFlashAddr,0x8006);
iFlashAddr++; /* 程序执行的入口地址 */
WriteFlash(iFlashAddr,0x0002);
iFlashAddr++;
WriteFlash(iFlashAddr,0x8085);
iFlashAddr++; /* 程序长度 */
WriteFlash(iFlashAddr,0x7f00);
iFlashAddr++; /* 程序要装载到的地址 */
WriteFlash(iFlashAddr,0x0002);
iFlashAddr++;
WriteFlash(iFlashAddr,0x8000);
iFlashAddr++;
for (iLoop=0;iLoop<0x7f00;iLoop++)
{ /* 从程序空间读数据，放到暂存单元 */
asm(" pshm al");
asm(" pshm ah");
asm(" rsbx cpl");
asm(" ld #00fch,dp");
asm(" stm #0000h, ah");
asm(" MVDM _iLoop, al");
asm(" add #2800h,4,a");
asm(" reada 0h");
asm(" popm ah");
```

```
asm(" popm al");
asm(" ssbx cpl"); /* 把暂存单元内容写入 FLASH */
WriteFlash(iFlashAddr,*pTemp);
iFlashAddr++; } /* 中断向量表长度 */
WriteFlash(iFlashAddr,0x0080);
iFlashAddr++; /* 中断向量表装载地址 */
WriteFlash(iFlashAddr,0x0000);
iFlashAddr++;
WriteFlash(iFlashAddr,0x0080);
iFlashAddr++;
for (iLoop=0;iLoop<0x0080;iLoop++) { /* 从程序空间读数据，放到暂存单元 */
asm(" pshm al");
asm(" pshm ah");
asm(" rsbx cpl");
asm(" ld #00fch,dp");
asm(" stm #0000h, ah");
asm(" MVDM _iLoop, al");
asm(" add #0080h,0,a");
asm(" reada 0h");
asm(" popm ah");
asm(" popm al");
asm(" ssbx cpl"); /* 把暂存单元内容写入 FLASH */
WriteFlash(iFlashAddr,*pTemp);
iFlashAddr++;
} /* 写入引导表结束标志 */
WriteFlash(iFlashAddr,0x0000);
iFlashAddr++;
WriteFlash(iFlashAddr,0x0000); /* 在数据空间的 0xFFFF 写入引导表起始地址 */
iFlashAddr=0xffff;
WriteFlash(iFlashAddr,0x8000);
```

关于 LF2407A 的 FLASH 烧写问题的几点说明

TI 现在关于 LF24x 写入 FLASH 的工具最新为 c2000flashprogsw_v112。可以支持 LF2407、LF2407a、LF2401 及相关的 LF240x 系列。建议使用此版本。http://focus.ti.com/docs/tool/toolfolder.jhtml?PartNumber=C24XSOFTWARE

上可以下载到这个工具。我们仿真器自带的光盘中也有此烧写程序。 在使用这个工具时注意：

一,先解压，再执行 setup.exe。

二、进入 cc 中，在 tools 图标下有烧写工具；

1、关于 FLASH 时钟的选择，此烧写工具默认最高频率进行 FLASH 的操作。根据目标系统的工作主频重新要进行 PLL 设置。方法：先在 advance options 下面的 View Config file 中修改倍频。存盘后，在相应的目录下（tic2xx\algos\相应目录）运行 buildall.bat 就可以完成修改了。再进行相应的操作即可。

2、若是你所选的频率不是最高频率，还需要设定你自己的 timings.xx 来代替系统默认的最高频率的 timings.xx。例如 LF2407a 的默认文件是 timings.40。Timings.xx 可以利用 include\timings.xls 的 excel 工作表

来生成。然后在 advance options 下面的 View Config file 中修改相应的位置。存盘后，在相应的目录下运行 buildall.bat 就可以完成修改了。
3、对于 TMS320LF240XA 系列，还要注意：由于这些 DSP 的 FLASH 具有加密功能，加密地址为程序空间的 0x40-0X43H，程序禁止写入此空间，如果写了，此空间的数据被认为是加密位，断电后进入保护 FLASH 状态，使 FLASH 不可重新操作，从而使 DSP 报废，烧写完毕后一定要进行 Program passwords 的操作，如果不做加密操作就默认最后一次写入加密位的数据作为密码。
4、2407A 不能用 DOS 下的烧写软件烧写，必须用 c2000flashprogsw_v112 软件烧写；
5、建议如下：
1)、一般调试时，在 RAM 中进行；
2)、程序烧写时，避开程序空间 0x40-0x43H 加密区，程序最好小于 32k；
3)、每次程序烧写完后，将 word0，word1,word2，word3 分别输入自己的密码，再点击 Program password，如果加密成功，提示 Program is arrayed，如果 0x40－0x43h 中写入的是 ffff，认为处于调试状态，flash 不会加密；
4)、断电后，下次重新烧写时需要往 word0～word3 输入已设的密码，再 unlock，成功后可以重新烧写了；

6、VCPP 管脚接在＋5V 上，是应直接接的，中间不要加电阻。
7、具体事宜请阅读相应目录下的 readme1,readme2 帮助文件。
8.注意*.cmd 文件的编写时应该避开 40-43H 单元，好多客户由于没有注意到这里而把 FALSH 加密。

如何设置硬件断点？
在 profiler －>profile point -> break point

c54x 的外部中断是电平响应还是沿响应？
是沿响应，准确的说，它要检测到 100(一个 clk 的高和两个 clk 的低)的变化才可以。

参考程序，里面好象都要 disable wachdog,不知道为什么?
watchdog 是一个计数器，溢出时会复位你的 DSP，不 disable 的话，你的系统会动不动就 reset。

时钟电路选择原则
1,系统中要求多个不同频率的时钟信号时，首选可编程时钟芯片;
2,单一时钟信号时，选择晶体时钟电路;
3,多个同频时钟信号时，选择晶振;
4,尽量使用 DSP 片内的 PLL，降低片外时钟频率，提高系统的稳定性;
5,C6000、C5510、C5409A、C5416、C5420、C5421 和 C5441 等 DSP 片内无振荡电路，不能用晶体时钟电路;
6,VC5401、VC5402、VC5409 和 F281x 等 DSP 时钟信号的电平为 1.8V，建议采用晶体时钟电路

C 程序的代码和数据如何定位
1,系统定义:
.cinit 存放 C 程序中的变量初值和常量;
.const 存放 C 程序中的字符常量、浮点常量和用 const 声明的常量;
.switch 存放 C 程序中 switch 语句的跳针表;
.text 存放 C 程序的代码;
.bss 为 C 程序中的全局和静态变量保留存储空间;

.far 为 C 程序中用 far 声明的全局和静态变量保留空间;
.stack 为 C 程序系统堆栈保留存储空间，用于保存返回地址、函数间的参数传递、存储局部变量和保存中间结果;
.sysmem 用于 C 程序中 malloc、calloc 和 realloc 函数动态分配存储空间
2,用户定义:
#pragma CODE_SECTION (symbol, "section name");
#pragma DATA_SECTION (symbol, "section name")

cmd 文件
由 3 部分组成：
1)输入 / 输出定义：.obj 文件：链接器要链接的目标文件;.lib 文件：链接器要链接的库文件;.map 文件：链接器生成的交叉索引文件;.out 文件：链接器生成的可执行代码;链接器选项
2)MEMORY 命令：描述系统实际的硬件资源
3)SECTIONS 命令：描述“段”如何定位

为什么要设计 CSL?
1,DSP 片上外设种类及其应用日趋复杂
2,提供一组标准的方法用于访问和控制片上外设
3,免除用户编写配置和控制片上外设所必需的定义和代码

什么是 CSL?
1,用于配置、控制和管理 DSP 片上外设
2,已为 C6000 和 C5000 系列 DSP 设计了各自的 CSL 库
3,CSL 库函数大多数是用 C 语言编写的，并已对代码的大小和速度进行了优化
4,CSL 库是可裁剪的：即只有被使用的 CSL 模块才会包含进应用程序中
5,CSL 库是可扩展的：每个片上外设的 API 相互独立，增加新的 API，对其他片上外设没有影响

CSL 的特点
1,片上外设编程的标准协议：定义一组标准的 APIs：函数、数据类型、宏;
2,对硬件进行抽象，提取符号化的片上外设描述:定义一组宏，用于访问和建立寄存器及其域值
3,基本的资源管理:对多资源的片上外设进行管理;
4,已集成到 DSP/BIOS 中:通过图形用户接口 GUI 对 CSL 进行配置;
5,使片上外设容易使用:缩短开发时间，增加可移植.

为什么需要电平变换?
1) DSP 系统中难免存在 5V/3.3V 混合供电现象;
2)I/O 为 3.3V 供电的 DSP，其输入信号电平不允许超过电源电压 3.3V;
3)5V 器件输出信号高电平可达 4.4V;
4)长时间超常工作会损坏 DSP 器件;
5)输出信号电平一般无需变换

电平变换的方法
1,总线收发器（Bus Transceiver）：

常用器件： SN74LVTH245A（8 位）、SN74LVTH16245A（16 位）
特点：3.3V 供电，需进行方向控制，
延迟：3.5ns，驱动：-32/64mA，
输入容限：5V
应用：数据、地址和控制总线的驱动
2,总线开关（Bus Switch）
常用器件：SN74CBTD3384（10 位）、SN74CBTD16210（20 位）
特点：5V 供电，无需方向控制
延迟：0.25ns，驱动能力不增加
应用：适用于信号方向灵活、且负载单一的应用，如 McBSP 等外设信号的电平变换
3,2 选 1 切换器（1 of 2 Multiplexer）
常用器件：SN74CBT3257（4 位）、SN74CBT16292（12 位）
特点：实现 2 选 1，5V 供电，无需方向控制
延迟：0.25ns，驱动能力不增加
应用：适用于多路切换信号、且要进行电平变换的应用，如双路复用的 McBSP
4,CPLD
3.3V 供电，但输入容限为 5V，并且延迟较大：＞7ns，适用于少量的对延迟要求不高的输入信号
5,电阻分压
10KΩ 和 20KΩ 串联分压，5V×20÷（10＋20）≈3.3V

未用的输入／输出引脚的处理
1,未用的输入引脚不能悬空不接，而应将它们上拉活下拉为固定的电平
1)关键的控制输入引脚，如 Ready、Hold 等，应固定接为适当的状态,Ready 引脚应固定接为有效状态,Hold 引脚应固定接为无效状态
2)无连接（NC）和保留（RSV）引脚,NC 引脚：除非特殊说明，这些引脚悬空不接,RSV 引脚：应根据数据手册具体决定接还是不接
3)非关键的输入引脚,将它们上拉或下拉为固定的电平，以降低功耗
2,未用的输出引脚可以悬空不接
3,未用的 I/O 引脚:如果确省状态为输入引脚，则作为非关键的输入引脚处理，上拉或下拉为固定的电平;如果确省状态为输出引脚，则可以悬空不接