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 的控制信号 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？建议使用目标板测试。

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，如键盘输入、屏幕显示等。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。

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 文件转换为可以烧写 ROM 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 }

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。

TI DSP 的 C 语言的特点，如何使我编写的 C 更高效？

TI DSP 的 C 语言是标准的 ANSI 的 C，是一个专门优化的 C。对于 C3x/C5000/C6000 的用户，C 对于 ASM 的效率可以达近 1:1。 TI 的 DSP，每个系列有每个的特点。要编制高效的 C 程序，建议：

1)根据 DSP 的特点，调整程序编写的流程，任务的分配。

2)数据尽量放在片内。

3)对于要求高的子程序，用人工优化或 ASM 编写。

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。

如何从老的编译工具升级到 CCS？

1)在 Project 菜单下，创建 project。

2)将源文件（C 和 ASM 文件），加入 project 中。

3)将 Link 的 CMD 加入 project 中，并将 CMD 中的库文件设置去除。

4)将 include 文件和库文件加入 project 中。

5)设置编译的选项。

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。

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”。

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 有哪些数学库及其它应用软件？

MATH 库，FFT，FIR/IIR 等，可以在 TI 的网页免费下载，具体地址为 ftp://ftp.ti.com/pub/tms320bbs/00index.htm。

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。

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。

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。

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。

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。

DSP／BIOS 能否在 TMS320C54x 系列 DSP 的扩展内存中运行？

能。DSP／基本输入输出系统构置工具允许用户在 GlobalSetting 条件下选择适宜的库。DSP／基本输入输出系统要求基本输入输出系统、Sysinit 和 Vect 部分放到存储器（0x000000－0x008000）的重叠部分（OVLY＝1）。这些部分（.基本输入输出系统、.sysinit、.vect）含有轮询程序以支持扩展的存储器，并期望在起始序列中。余下的部分和对象可以置于存储器的任何位置。

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。

是否必需对浮点运算作人工的代码调整。C 编译器是否不能自动对浮点运算进行处理？

1.浮点不需要人工调整 ；

2.C 可以主动处理浮点运算

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

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")

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

相信对你有帮助的：