C2000 常见问题解答

ZHCU101

C2000 CLA 常见问题

架构、配置

1. CLA 是什么？

CLA 是一款与主 CPU 并行运行的 32 位浮点数学加速器。

2. CLA 独立于主 CPU 之外吗？

是的。 一旦 CLA 被主 CPU 配置，它可以独立于主 CPU 之外执行算法。 CLA 有自己的总线结构、寄存器组、管线和处理单元。 此外，CLA 可直接访问 ePWM，比较器和 ADC 结果寄存器。 这使得它非常适合于处理时间关键控制循环，但是它也可以用于滤波或数学算法。

3. CLA 是中断驱动的吗？

是的，2803x CLA 响应 ADC，ePWM 和 CPU 定时器 0 中断。 其他器件可对其他系统中断做出响应。 相关信息请参考你的器件专用文档。 也可参看 任务和中断。

4. CLA 中断的响应速度怎样？

CLA 不处理非时间关键中断（例如通信端口），并且没有中断嵌套。 此外，CLA 直接接收中断，而非通过外设中断扩展块 (PIE) 接收。 正是由于这一点，CLA 具有极低的中断响应延迟。 在中断之后的第七个周期上，第一条指令将位于管线的解码 2 (D2) 阶段。 此外，只要 ADC 结果寄存器可用，CLA 就能够轻松地读取其中内容。 也可参看 任务和中断 以及访问外设。

5. CLA 有寄存器吗？

有的，CLA 有自己的独立寄存器组。 CLA 寄存器可分为两组：

配置寄存器

这些寄存器中的一部分被主 C28x CPU 用来配置 CLA。 其他寄存器为主 CPU 提供状态信息。 例如，哪个中断已经被标记或者现在哪个任务正在运行。

执行寄存器。

这些寄存器包括四个浮点结果寄存器、两个辅助寄存器、一个状态寄存器和一个程序计数器。 这些寄存器可由主 C28x CPU 读取，但是不能被 C28x CPU 写入。

6. CLA 有累加器吗？

没有单个寄存器被指定为累加器 - 运算的结果进入结果寄存器 (MR0 - MR3)。

7. CLA 的运行频率是多少？

2803x，2806x 和 2805x 器件上的 CLA 的运行速度与 CPU 的运行速度一样 (SYSCLKOUT)。 其他器件也许会有所不同。 相关信息请参考你的器件专用文档。

8. 复位时 CLA 的状态是什么？

到 CLA 的时钟被禁用，并且所有 CLA 寄存器被清零。 在被主 CPU 配置为处理中断前，CLA 将不会开始处理中断。

9. 如何配置 CLA？

与任何其他模块或外设一样，CLA 由主 CPU 进行配置。

开发工具、调试等……

10. 我想知道有哪些代码开发工具可用，以及我如何调试针对 CLA 的代码。

请参考 C2000 CLA 调试 FAQ

任务和中断

11. ‘任务’是什么？

CLA 任务是由 CLA 执行的中断响应例程。

12. 支持多少个中断？

2803x 和 2806x CLA 都支持 8 个中断。

13. 哪些中断能够启动一个任务？

外设： 每个任务具有可以触发它的特定外设中断。 主 CPU 选择 MPISRCSEL 1 寄存器中的哪个中断？

需要理解的重要一点是触发资源只是任务的启动机制。 触发资源不限制任务可进行的操作。例 如，任务 1 可以读取任一/多个 ADC 结果寄存器，并且修改任何 ePWM1，ePWM2，ePWM3...ePWM7 寄存器，即使此任务是由 EPWM1_INT 启动时也是如此。

下面显示了 2803x 和 2806x 上的可用触发值。 其他器件也许会有所不同。 相关信息请参考你的器件专用文档。

在 2803x 上 中断触发值分配如下：

- 中断 1 = 任务 1 = ADCINT1 或 EPWM1_INT 或只为软件
- 中断 2 = 任务 2 = ADCINT2 或 EPWM2_INT 或只为软件
- 中断 3 = 任务 3 = ADCINT3 或 EPWM3_INT 或只为软件
- 中断 4 = 任务 4 = ADCINT4 或 EPWM4_INT 或只为软件
- 中断 5 = 任务 5 = ADCINT5 或 EPWM5_INT 或只为软件
- 中断 6 = 任务 6 = ADCINT6 或 EPWM6_INT 或只为软件
- 中断 7 = 任务 7 = ADCINT7 或 EPWM7_INT 或只为软件
- 中断 8 = 任务 8 = ADCINT8 或 CPU 定时器 0 或只为软件

在 2806x 上 中断触发值分配如下：

- 中断 1 = 任务 1 = ADCINT1 或 EPWM1_INT 或只为软件
- 中断 2 = 任务 2 = ADCINT2 或 EPWM2_INT 或只为软件
- 中断 3 = 任务 3 = ADCINT3 或 EPWM3_INT 或只为软件
- 中断 4 = 任务 4 = ADCINT4 或 EPWM4_INT 或 eQEP1/2 或 ECAP1/2/3 或只为软件

- 中断 5 = 任务 5 = ADCINT5 或 EPWM5_INT 或 eQEP1/2 或 ECAP1/2/3 或只为软件
- 中断 6 = 任务 6 = ADCINT6 或 EPWM6_INT 或 eQEP1/2 或 ECAP1/2/3 或只为软件
- 中断 7 = 任务 7 = ADCINT7 或 EPWM7_INT 或 eQEP1/2 或 ECAP1/2/3 或只为软件
- 中断 8 = 任务 8 = ADCINT8 或 CPU 定时器或 eQEP1/2 或 ECAP1/2/3 或只为软件

14. 主 CPU 能够通过软件启动任务吗?

可以！ 主 CPU 可以使用 IACK #16bit 指令随时标记一个中断。 例如，IACK 0x003 将标记中断 1 和中断 2。这与强制寄存器 (MIFRC) 中的设置位一样。

15. 我试图用 IACK 指令来强制任务执行，但是不起作用。 什么地方出错了吗？

- 请确保你已经在 MICTL 寄存器启用这个功能。
- 请确保在 MIER 寄存器中启用此中断。
- 请确保你在使用正确的 IACK 自变量。 例如，IACK #0x0003 将标记中断 1（位 0）和中断 2（位 1）。

16. 如果两个中断同时出现，哪个先执行呢？

被标记（MIFR 寄存器）且被使能（MIER 寄存器）的最高优先级任务被执行。 中断 1 / 任务 1 具有最高优先级，而中断 8 / 任务 8 的优先级最低。

17. 你可以嵌套 CLA 中断吗？

不可以。CLA 任务在它完成后执行。 一旦一个任务完成，那么被标记且被使能的最高优先级中断将自动开始。

18. CLA 能够中断主 CPU 吗？

CLA 将发送一个中断到 PIE（外设中断扩展块）来告知主 CPU 一个任务已经完成。 每个任务在 PIE 中有一个相关矢量。 这个中断在相关矢量完成时自动触发。 例如，当任务 1 完成时，PIE 中的 CLA1_INIT 将被标记。

PIE 中还有专门用于浮点上溢和下溢情况的中断。

19. 主 CPU 能够终止任务吗？

可以。 如果一个中断已经被标记，但是任务还未运行，那么主 CPU 可以使用 MICLR 寄存器清除此标记。如果任务已经运行，那么一个软复位（在 MCTL 中）将终止此任务并将 MIER 寄存器清零。 如果你希望将所有 CLA 寄存器清零，你可以使用 MCTL 寄存器中的硬复位选项。

20. 每个任务的起始地址是什么？ 起始地址是固定的吗？

起始地址是可配置的。 每个任务具有一个相关中断矢量（MVECT1 至 MVECT8）。 这个矢量保存任务的起始地址（作为第一个程序位置的偏移）。

21. 任务有大小限制吗？

没有限制，除了针对所有任务的全部指令需要与器件 CLA 程序存储器的大小相匹配。 所有 CLA 指令是 32 位的，所以在 4k x 16 的程序空间内，你可以拥有大约 2k 的 CLA 指令。

其他器件的准确程序存储器数量和程序计数器大小会有所不同。 相关信息请参考你的器件专用文档。

22. 我如何标明一个任务的末尾？

在一个任务开始后，CLA 将执行指令，直到遇到 "MSTOP" 指令。 MSTOP 表示任务的末尾。

23. CLA 自己可以标记其他任务吗？

CLA 不能对它自己的配置寄存器进行写入操作，所以它不能通过对强制寄存器的写入来启动一个任务。 然而，它可以写入 ePWM 寄存器，所以，从技术角度讲，它可以强制生成来自其中一个 ePWM 模块的中断。主 CPU 可以在任务完成时获得一个中断。 在这个中断内，你可以使用 IACK 指令启动另外一个任务。

24. 如果 CLA 被配置为对 ACDINT1 做出响应，那么 CPU 也能做出响应吗？

可以。 中断被发送至 CLA 和 PIE，所以它们中的一个或者它们两个都可以做出响应。

访问外设

25. CLA 可直接访问哪个外设？

下面显示了 CLA 在 2803x 和 2805x 上能够访问的外设。 在某些器件上会提供其他外设。 相关信息请参考你的器件专用文档。

2803x

CLA 可直接访问 ADC 结果，ePWM + HRPWM 和比较器寄存器。

2806x

CLA 可直接访问 ADC 结果，ePWM + HRPWM，eCAP，eQEP 和比较器寄存器。

26. 这些寄存器中你所提到的某些寄存器受到主 CPU 的 EALLOW 保护，防止破坏性写入。 CLA 也具有这一保护功能吗？

在 CLA 状态寄存器中有一个名为 MEALLOW 位，这个位启用/禁用对 CLA 写入的保护。 这个位由 MEALLOW/MEDIS CLA 指令置位和清零。 这个保护不受主 CPU 的 EALLOW 位的影响。 也就是说，主 CPU 可以经由 EALLOW 启用写入，但是寄存器仍将通过 MEALLOW 受到 CLA 的保护。

27. CLA 是如何“恰好”读取 ADC 结果寄存器的？

2803x 上的 ADC 可被配置成在采样窗口时间后将一个中断置位。 如果 CLA 被配置为对 ADC 中断做出响应，那么任务将在转换过程中开始。 任务中的第八条指令将恰好在 ADC 结果寄存器更新时读取它。

28. 如果 CLA 采取一个 ADC 中断，它可以清除 ADC 的中断标志吗？

不可以。CLA 无法访问 ADC 配置寄存器，所以它不能清除 ADC 中断标志。 这一操作有三个选项：

- 选项 1

将 ADC 置于连续模式。 在这个模式中，触发时下一个转换将启动，即使此标志仍被置位也是如此。

- 选项 2

用主 CPU 以及 CLA 来处理 ADC 中断，并且让主 CPU 清除标志。

- 选项 3

让主 CPU 处理 CLA 的任务中断的末尾，并清除标志。

29. 你曾提到 CLA 可以访问 ePWM 寄存器。 是器件上的所有 ePWM 模块吗？

所有任务都能够访问任一 ePWM 模块。 对此没有限制。

30. 我如何从 CLA 中对 GPIO 进行控制？

CLA 不能访问 GPIO 控制寄存器。 GPIO 控制通常由主 CPU 处理。 你可以进行的一项操作是用 CLA 访问 ePWM 寄存器来切换 ePWM 引脚。 如果你希望在任务末尾切换 GPIO 引脚，可以在主 CPU 处理 CLA 中断时由主 CPU 来完成。

31. 如果 CLA 正在使用 ePWM 或 ADC 结果寄存器，这是不是意味着主 CPU 不能访问这些寄存器？

不是。CLA 和主 CPU 都可以访问这些寄存器。 这些寄存器的仲裁机制可在 CLA 参考指南中找到。请牢记，如果主 CPU 对寄存器执行读取-修改-写入操作，并且在读取和写入操作之间 CLA 修改同一寄存器的话，CLA 所作出的更改会丢失。 总的来说，最好不要让两个处理器都对寄存器进行写入操作。

32. 我想让 CLA 任务 1 访问 ePWM1 和 ePWM2 寄存器，但是任务 1 只能由 EPWM1_INT 触发。

中断只是启动任务的方法。 它不会限制 CLA 在任务期间可以访问的资源。 任何 CLA 任务都可以访问所有 ADC 结果、比较和 ePWM 寄存器。 例如，假定 ADCINT1 触发任务 1。 然后，这个单个任务可读取 ADC RESULT0，执行一个比较算法，读取 ADC RESULT1，执行另外一个控制算法，等等。

存储器访问

33. CLA 具有对器件上所有存储器的访问权吗？

CLA 可以使用器件上的特定块。

- CLA 程序存储器

在 Piccolo 上（2803x 和 2806x），这是一个 4k x 16 块，单周期（无等待状态）。 这表示它可以保存 2048 条 CLA 指令（所有 CLA 指令均为 32 位）。 复位时，这个块被映射至主 CPU 存储器空间中，并且由 CPU 处理，处理方式与任何其他存储块相类似。 在被映射至 CPU 空间时，主 CPU 可以使用 CLA 程序代码来初始化存储器。 一旦存储器被初始化，CPU 将其映射至 CLA 程序空间。

- CLA 数据存储器

这些块均为 1k x 16，单周期。 复位时，这个块被映射至主 CPU 存储器空间中，并且由 CPU 处理，处理方式与任何其他存储块相类似。 在被映射至 CPU 空间时，主 CPU 可以使用数据表格以及用于 CLA 的系数来初始化存储器。 一旦存储器初始化为 CLA 数据，主 CPU 将其映射至 CLA 空间中。（每个块可被单独映射）。

- 2803x 数据 RAM

器件上有两个 CLA 数据存储器块。

每个数据 RAM 属于 CPU 或 CLA。 如果它被映射至 CLA，那么 CPU 将在它试图访问此块时读取 0x0000。 相似地，如果此块被映射至 CPU，CLA 将在它设法读取此块时读取所有 0x0000。

- 2806x 数据 RAM 访问

在器件上有三个 CLA 数据存储器块。

访问方法与 2803x 上的操作一样，除了第二配置位已经被添加，这样，即使在存储器块被映射至 CLA 的情况下，CPU 仍然可以对其进行读取/写入操作。

- 共用消息 RAM

有两个小存储器块用于 CLA 与主 CPU 之间的数据共享和通信。 在 Piccolo 上，这些块的大小为 128 x 16 个字。

- CLA 到 CPU 消息 RAM

CLA 可以读取/写入，主 CPU 只能读取

- CPU 到 CLA 消息 RAM

CPU 可以读取/写入，CLA 只能读取

34. CLA 和 CPU 能够同时访问同一消息 RAM 吗？

这些访问的仲裁方式说明如下。

- CLA 到 CPU 消息 RAM： 访问的优先级为（首先为最高优先级）：

  1. CLA 写入
  2. CPU 调试写入
  3. CPU 数据读取，程序写入，CPU 调试读取
  4. CLA 数据读取

- CPU 到 CLA 消息 RAM： 访问的优先级为（首先为最高优先级）：

  1. CLA 读取
  2. CPU 数据写入，程序写入，CPU 调试写入
  3. CPU 数据读取，CPU 调试读取
  4. CPU 程序写入

35. 如果我对数据手册的理解是正确的话，CLA 代码只能从 L3（CLA 程序 RAM）中运行。 它可由闪存中载入，并在 L3 中运行。

是的。 CLA 只能从指定为 CLA 程序存储器的存储器中执行。 2803x 上为 L3。 调试期间，你可以使用调试器直接加载 ram。 在独立系统中，主 CPU 需要使用 CLA 代码来将 RAM 初始化。 代码最有可能被存储在闪存中，并被复制到 L3 中。 要获得相关示例，请参考 controlSUITE 内 FIR 示例中的闪存项目内的 CLA FIR 来了解这一操作是如何完成的。

- 对于 2806x (C:\ti\controlSUITE\device_support\f2806x)
- 对于 2803x (C:\ti\controlSUITE\device_support\f2803x)
- 对于 2805x (C:\ti\controlSUITE\device_support\f2805x)

36. 我能将除 L3 以外的块用于 CLA 代码吗？

不可以 - 在 2803x 和 2806x 器件上，只有 L3 可被 CLA 用于程序存储器。 对于你所使用的特定器件，请检查数据手册中的存储器映射。

37. L3 存储器块被指定用于 CLA 程序。 我可以将其中的一部分分配给 CLA，剩下的用于 CPU 吗？

2803x： 不可以 - 这个存储器块或者属于主 CPU（复位时的缺省值）或者属于 CLA。 也就是说，二者都可以访问它。 如果此块被分配给 CLA，而主 CPU 试图提取或读取数据时，它将接收到一个 0x0000。

2805x 和 2806x： 可以 - 通过使用 MMEMCFG[RAMxCPUE] 位，你可以将数据存储器的读取和写入权限授予 CPU。 在更改配置后，始终等待 2 个 SYSCLKOUT 周期才能访问存储器。

38. 消息 RAM 可被用作 CLA 的数据存储器吗？

CLA 至 CPU 消息 RAM (128 x 16) 可由 CLA 读取/写入，并且可被用作 CLA 的数据 RAM。

39. 我如何在 CLA 至 CPU 消息 RAM 中初始化变量？

由于主 CPU 不能写入这个存储器，CLA 将需要初始化这些变量。 要进行这一操作，设置一个任务来执行初始化，然后用主 CPU 来触发此任务。 controlSUITE 中针对 2806x (C:\ti\controlSUITE\device_support\f2806x) 和 2803x (C:\ti\controlSUITE\device_support\f2803x) 的 FIR 示例显示这一操作是如何完成的。

40. 在我的应用程序中，CLA 不需要数据存储器。 这个存储器可由主 CPU 使用吗？

可以 - 如果 CLA 不需要此存储器，那么它就像任一其他块一样可由主 CPU 使用。 此外，两个数据存储器块可被单独地分配给 CLA 或主 CPU，所以你可以将一个块分配给主 CPU，另外一个分配给 CLA。

41. 在我的应用程序中，CLA 并未使用所有数据 RAM。 可以将一个块分配给 CLA，而将其他块分配给主 CPU 吗？

可以。 数据 RAM 被单独映射至 CPU 或 CLA。

42. 我想要执行一个乒乓 (ping-pong) 系统配置，在这系统配置中，CLA 使用一个数据 RAM，然后主 CPU 再使用这个数据 RAM。 这可行吗？

2803x

可以 - 在改变数据 RAM 的映射前，你需要确定几件事情：

- 在改变映射后（经由 MMEMCFG），始终等待 2 个 SYSCLKOUT 周期，然后再访问存储器。
- 在将存储器从 CPU 更改为 CLA 之前，请确保 CPU 未执行任何存储器访问。
- 在将存储器从 CLA 更改为 CPU 之前，请确保 CLA 未执行任何访问。 对这一点进行检查的方法是将 MIER 寄存器清零，等待几个周期，然后检查 MIRUN 是否被清零。

2806x

可以 - 有两个选项：

- 选项 1

使用 MMEMCFG[RAMxCPUE] 位，你可以授予 CPU 对数据存储器的读取和写入权限。 在更改配置后，始终等待 2 个 SYSCLKOUT 周期，然后再访问存储器。

- 选项 2

对于 2803x，按照同一步骤进行操作，并将存储器重新映射至 28x.

43. CLA 存储器是否受到代码安全模块 (CSM) 的保护？

是的 - 在 2803x 器件上，所有 CLA 存储器都受到 CSM 的保护。 CLA 配置和结果寄存器也受到保护。

CLA 和主 CPU 之间的通信

44. 主 CPU 和 CLA 之间如何进行通信？

通信由消息 RAM 和中断处理。

- CLA 可以通过 CLA 至 CPU 消息 RAM 将数据传递给主 CPU。
- 主 CPU 可以通过 CPU 至 CLA 消息 RAM 将数据传递给 CLA。
- 如果需要的话，主 CPU 可以使用 IACK 指令在软件中标记一个 CLA 中断/任务。
- CLA 可以通过一个到 PIE 的中断来通知主 CPU 一个任务已经完成。 在 PIE，每个任务都有一个中断矢量。 如果应用程序不作要求的话，主 CPU 不是必须处理来自 CLA 的中断。

45. 在我的代码中，如何在 CLA 和主 CPU 之间共用变量？

由于 CLA 和 C28x 器件代码在同一项目内，这很容易实现。 我的建议是：

- 创建一个包含常见常量和变量的共用头文件。 将这个文件包含在 C28x C 和 CLA.asm 代码中。
- 使用数据段 pragma 语句和链接器文件将变量放置在合适的消息 RAM 中。
- 在你的 C 语言代码中定义共用变量。
- 在 CPU 至 CLA 消息 RAM 中，用主 CPU 初始化变量。
- 在 CLA 至 CPU 消息 RAM 中，用 CLA 任务初始化变量。 这个初始化任务可由主 C28x 软件启动。

46. 我可以将 CLA 数据 RAM 用作消息 RAM 吗？

可以，但是需要牢记的是，不论何时，每个 CLA 数据 RAM 或者属于主 CPU 或者属于 CLA。 因此，要使其它处理器查看数据，你必须首先确保没有发生对 RAM 的访问，然后再更改映射。

指令集、代码执行

47. CLA 的指令执行速度有多快？

CLA 的时钟速率与主 CPU 一样 (SYSCLKOUT)，在 2803x 上最大值为 60MHz，在 2806x 上最大值为 80MHz。 所有 CLA 指令都是单周期的。 虽然跳转（分支/调用/返回）指令本身是单周期的，但是跳转指令的完成时间取决于指令附近使用了多少个“延迟槽”。 如果未使用延迟槽，那么无论是否跳转，指令都会在 7 个周期（最差情况）内完成。 无论是否跳转，典型的执行时间是 4 个周期。在这个情况下，指令之前的槽位被使用，但之后的未被使用。

48. 在此示例中，我注意到在每个 MSTOP 后有 3 个 MNOP。 为什么这么做？是要求这么做吗？

有一个限制条件，就是 MSTOP 不应在具有一条分支的 3 个指令内结束。 MNOPS 已经被添加，以确保始终满足这一要求，即使在任务之后的程序 RAM 未经初始化的情况下也是如此。 如果你确切的知道 MSTOP 之后的 3 条指令内没有分支指令，那么你可以删除 MNOPS。

49. CLA 使用浮点数学运算吗？

是的，CLA 支持 32 位（单精度）浮点数学运算。 这些运算遵循 IEEE 754 标准。

50. 为什么是浮点而不是定点？

浮点易于编码。 它是自我饱和的，因此免除了代码的饱和和缩放负担。 此外，它不会受到上溢/下溢符号反转问题的困扰。 最后，浮点编码算法周期有效性更高。

51. 由于主 CPU 是定点的，而 CLA 是浮点的，难道我必须对数字进行转换吗？

是的，CLA 利用数据转换指令简化了这个操作。 如果你正在读取存储器，这个转换会在值被读取时完成，所以效率很高。 例如，你可以读取 ADC 结果寄存器（无符号 16 位）并在读取时将其转换为 32 位浮点数。

52. CLA 支持 C28x+FPU 上的重复块 (RPTB) 指令吗？

不支持，但是你可以使用循环（指令或调用）来多次执行一组指令。 也没有针对 CLA 的单重复指令。 (RPT ||...)

53. CLA 支持分支指令吗？

是的，CLA 有其自己的分支 (MBCNDD)，调用 (MCCNDD)，和返回 (MRCNDD)。 所有这些是有条件且被延迟的。

54. 在 CLA 中有子硬件模块吗，模块中的每个组件对应某些算法，这样的话，用户只需设置寄存器，或者 CLA 只是浮点完全可编程？

硬件中没有 CLA 的内置算法。 CLA 完全可编程。

55. CLA 支持何种类型的指令？

要获得指令完整列表，请查阅参考指南。

2803x： TMS320x2803x Piccolo CLA 参考指南 (SPRUGE6)。

2806x： TMS320x2806x Piccolo 技术参考手册 (SPRUH18)。

以下表格显示 CLA 支持的指令类型概述。

类型	示例	周期		类型	示例	周期
负载（视条件而定）	MMOV32 MRa, mem32{,CONDF}	1		**存储**	MMOV32 mem32，MRa	1
加载数据移动	MMOVD32 MRa，mem32	1		**存储/加载 MSTF**	MMOV32 MSTF，mem32	1
浮点 比较、最小值、最大值 绝对值、负值	MCMPF32 MRa，MRb MABSF32 MRa，MRb	1		**转换 无符号整数至浮点数 整数至浮点数 浮点数至整数**	MUI16TOF32 MRa，mem16 MI32TOF32 MRa，mem32 MF32TOI32 MRa，MRb	1

			等等		
浮点数学 加、减、乘 **1/X** 估值 **1/sqrt(x)** 估值	MMPYF32 MRa，MRb，MRc MEINVF32 MRa，MRb MEISQRTF32 MRa，Rb	1	整数加载/存储	MMOV16 MRa，mem16	1
加载/存储辅助寄存器	MMOV16 MAR，mem16	1	分支/调用/返回	MBCNDD 16bitdest {,CNDF}	1-7
整数运算 **AND，OR，XOR** 加减 移位	MAND32 MRa，MRb，MRc MSUB32 MRa，MRb，MRc MLSR32 MRa，#SHIFT	1	停止指令 任务末尾 断点	MSTOP MDEBUGSTOP	1
并行指令 与并行加/减相乘 并行加载/存储数学运算	MMPYF32 MRa，MRb，MRc || MSUBF32 MRd，MRe，MRf	1/1	无运算	MNOP	1

CLA 与 C28x + FPU（C28x 加上浮点单元）对比

56. 基本 CLA 指令也可用于 28x + FPU 器件吗？

为了确保我们步调一致，我们来定义以下指令集：

C28x 指令集

这是最初的定点指令集。

C28x+FPU 指令集

这是 C28x 指令集加上支持本地单精度（32 位）浮点运算的附加指令。 虽然附加指令主要支持单精度浮点数学运算，但是也包括某些诸如 RPTB（重复块）的其他有用指令。 由于它们是扩展集的一部分，并且只在具有 FPU 的器件上提供，我们仍然把它们视为 FPU 指令的一部分。

CLA 指令集

CLA 指令集是 FPU 指令的子集。 在 CLA 上，有几个 FPU 指令不被支持 - 例如重复块就不被支持。 CLA 还有几个 FPU 没有的指令。 例如：CLA 具有一些本地整数数学指令以及一个本地分支/调用/返回。

57. 如果 CLA 指令集是 FPU 指令集的子集，我可以认为二者上的浮点 div，sin，cos 等的基准是一样的吗？

CLA 指令实际上是 C28x+FPU 的子集，而对于数学指令来说，它们二者有对等量，但是它们之间仍然有影响基准的差异。 例如：

- 乘法和转换（请见下一个问题）以及分支的周期差异
- 资源差异（例如： 8 个浮点结果寄存器与 4 个浮点结果寄存器）

58. CLA 的浮点乘法运算会快于常规 C28x+FPU 吗？

这真是一个容易使人上当的问题！ :) 请考虑以下情况：

- C28x FPU：乘法或转换花费 2p 个周期。 这意味着它们花费两个周期的时间来完成，但是请牢记你可以将另外一条指令放置在那个包含另外数学指令的延迟槽中。

- CLA：数学指令和转换花费 1 个周期 - 无需延迟槽。 所以，如果你不能使用 FPU 上的那个延迟周期进行有意义的操作，那么你可以说 CLA 速度较快，如果你只是数周期数量的话。

- 具有 CLA 的器件的运行速度远远低于（2803x 为 60MHz，2806x 为 80MHz）具有 FPU 单元的器件（80-300MHz）。

所以这取决于 FPU 延迟槽使用量以及器件的频率。

59. CLA 与 C28x + FPU 之间的主要差异是什么？

需牢记的最重要一点是 CLA 独立于主 CPU 之外，而 FPU 单元是 C28x 定点 CPU 顶部的扩展集。以下表格显示了二者之间的其他差异：

	2803x CLA	C28x+FPU
执行	独立于主 CPU 与主 CPU 并行执行浮点运算。	主 CPU 的一部分。 FPU 指令不与浮点运算并行执行。
浮点结果寄存器	4 (MR0 - MR3)	8 (R0H - R7H)
辅助寄存器	2，16 位，(MAR0, MAR1) 可以访问所有 CLA 数据	8，32 位，(XAR0 - XAR7) 与定点指令共用
管线	8 级管线 完全独立于主 CPU	取指令和解码阶段与定点指令共用 不能与定点运算并行执行
单步执行	将管线向前移动 1 个周期	完全清空管线
寻址模式	只使用 2 个寻址模式 直接 & 间接后递增 ~~(Direct & indirect with post increment)~~ 无数据页面指针	使用所有 C28x 寻址模式
被处理的中断	• 2803x： ADC，ePWM 和 CPU 定时器 0 中断 • 2806x： ADC，ePWM，CPU 定时器 0，eCAP 和 eQEP	所有可用中断
嵌套中断	不支持。 无堆栈指针。	支持堆栈指针。
指令集	独立指令集 FPU 指令的子集	浮点指令是除（扩展集） C28x 定点指令

	与 C28x+FPU 相似的记忆法，但是以 'M' 开头 ex: MMPYF32 MR0, MR1, MR2	之外的指令
重复指令	无单重复或重复块	重复 MACF32 & 重复块 (RPTB)
与 主 CPU 通信	通信方式为通过消息 RAM 和中断 主 CPU 能够读取 CLA 执行寄存器，但是不能写入	一个 CPU，但是你可以在定点和浮点寄存器之间复制信息 例如，将 R0H 复制到 ACC，或者将 STF 标志复制到 ST0
数学和转换	单周期	2p 周期（2 个管线周期）
整数运算	针对与、或、异或、加、减、移位等的原生指令	使用定点指令
流控	原生分支/调用/返回 有条件延迟	使用定点指令 需要将浮点标志复制到定点 ST0 中
分支/调用/返回	有条件 & 已延迟 后面的 3 条指令始终被执行 通过使用延迟周期可改进性能	使用定点流控 分支未被延迟 后面的指令只在未采用分支时执行
存储器访问	只能访问 CLA 程序、数据和消息 RAM 用于 CLA 程序的特定存储器 用于 CLA 数据的特定存储器	器件上的所有存储器 程序/数据存储器分配由用户来定
寄存器访问	• 2803x：ePWM+HRPWM，比较器，ADC 结果 • 2806x：ePWM+HRPWM，eQEP，eCAP，比较器，ADC 结果	器件上的所有寄存器
编程	CLA 汇编程序或 CLA C 语言编译器（要求 C28x codegen 6.1.0 或更新版本）	C/C++ 或汇编程序
工作频率（视器件而定）	• 2803x： 基于闪存的器件最高 60MHz • 2806x： 基于闪存的器件最高 80MHz	• 基于闪存的器件最高 15MHz (2833x) • 只有 RAM 的器件最高 300MHz (2834x)

在 CCS 上调试 C2000 CLA 常见问题

60. 我已安装 Code Composer Studio V4.0。 我如何获得调试 CLA 的支持？

除了下面的内容，还有一些对于 CCS 内使用 CLA C 语言编译器的要求。 请参考 C2000 CLA C 语言编译器维基网页 以获得更多信息。

- CLA 调试从 CCS 4.02 上开始提供。 如果你有一个较早的版本，那么请下载最新版或者检查软件更新。

普通 Code Composer Studio v4 信息请参考以下维基网页：

- Code Composer Studio V4
- 下载 CCS v4

61. 我有 Code Composer Studio V3.3 的完全版。 支持 CLA 的服务发布将在何时提供？

除了下面的内容，还有一些对于 CCS 内使用 CLA C 语言编译器的要求。 强烈建议使用 CCS 5.2 或更新版本。 CCS 3.3 还未经测试，不建议用于 CLA C 语言代码调试。 请参考 C2000 CLA C 语言编译器维基网页 以获得更多信息。

- 如果现在有 CCS 3.3 的完全版，那么请检查更新顾问（help ->update advisor）以获得完全版的链接，此版本支持 CLA 编译程序调试。

请注意：

- CCS V3.3 白金版 + SR12 不支持 CLA 编译程序调试。
- 没有发布将 CCS 3.3 + SR12 更新为支持 CLA 的服务补丁。
- 更新顾问中的链接是一个完全版，这个完全版将 SR12 和某些其他器件支持（其中包括 CLA，更新的编译器和 DSP/BIOS）组合在一起。

- 还提供一个支持 CLA 的 C2000 评估版（受限链接大小）供免费下载。

截至这篇文章成稿时，最新版本为这个链接中提供的构建 http://software-dl.ti.com/dsps/dsps_registered_sw/sdo_ccstudio/CCSv3/Free/C2000_Code_Composer_Studio_v3.3.83.19_limited.zip

62. 我需要从第三方获得新的驱动程序来调试 CLA 吗？

CCS v4： 在 CCS 4 中，你可以通过 Update Manager（更新管理器）来获得已更新的驱动程序。 它应该自动检查更新，但是你始终可以手工搜索更新。

CCS v3.3： 检查第三方的网页来获得更新。

对于 Blackhawk：

Blackhawk 已经发布一个支持 CLA 的补丁： http://www.blackhawk-dsp.com/support/Drivers33.aspx

对于 Spectrum Digital：

SD 已经发布更新的驱动程序 http://support.spectrumdigital.com/。

Signum

接至 2009 年 9 月 1 日，Signum 未将 CLA 支持集成在他们的驱动程序集中。 更多信息请向 Signum 查询。 http://www.signum.com/

63. 当我打开 CCS V3.3 时，并行调试管理器弹出。 我给如何使用它呢？

并行调试管理器将显示到主 CPU 的一个入口，以及一个到 CLA 的入口。 你可以通过 Open 菜单打开针对其中任何一个的调试窗口。 使用针对主 CPU 的调试窗口来构建你的项目、加载代码并运行主 CPU。

如果你不是在调试 CLA 代码，你可以断开 CLA 调试窗口并将其关闭。 当 CLA 调试窗口被断开时，所有 CLA 断点将被禁用，并且运行为 MNOP（无运算）。

如果你正在调试 CLA 代码，那么打开和连接 CLA 调试窗口。 这个窗口将显示反汇编假定 (assuming) CLA 操作码。 这个窗口中的步骤和运行函数也与 CLA 单步执行和运行相对应。 还请参考 断点、步进、运行、查看符号

普通调试工具问题

64. 我需要第二个仿真器来调试 CLA 吗？

不需要。主 CPU 和 CLA 可同时由同一 JTAG 端口进行调试。 两者均可单独进行调试。 例如，你可以在主 CPU 被暂停时运行 CLA。 相似地，你可以在 CLA 被暂停时运行主 CPU。

65. 有用于 CLA 的仿真器吗？

没有，目前 CLA 仿真器还未在计划之内。 我们提供超低成本的工具，诸如试验人员套件和 controlSTICK，这些工具使你能够使用真正的芯片。 真正的芯片要远远好于任何仿真器。

Codegen 工具，构建 CLA 代码。

66. 我需要哪个版本的 codegen 工具来创建 CLA 代码？

- C2000 Codegen 工具 V5.2.0 或之后的版本只支持 CLA 汇编级编程。
- C2000 Codegen 工具 V6.1.0 或之后的版本支持 CLA C 语言编译器 以及汇编级编程。

此外，你需要指定 --cla_support=cla0 选项来启用对 CLA 指令的支持。

67. 我们编译器/汇编程序抱怨说 CLA 指令是未知量。 为什么呢?

请确保你正在使用 C2000 Codegen 工具 V5.2.0 或之后的版本。 此外，你需要指定 --cla_support=cla0 选项来启用对 CLA 指令的支持。

68. CLA 汇编程序参考指南在哪里？

汇编程序本身与 28x 一样。 对 CLA 指令的支持最早被添加到 codegen V5.2.x 中 - 汇编程序能够通过其记忆法来分辨 CLA 指令。 28x 汇编程序指南在以下链接中 www.ti.com/lit/spru513

69. CLA C 语言编译器参考指南在哪里？

编译器本身从与 28x 编译器一样的外壳程序中被调用。 对 CLA C 语言代码的支持最早被添加到 codegen V6.1.x 中 - 编译器能够通过文件扩展名来分辨 CLA C 语言文件： .cla. 28x 编译器指南在以下链接中 www.ti.com/lit/spru514。 还请参考这个维基网页中的文章： CLA C 语言编译器。

70. Code Composer Studio 构建选项中的 --cla_support 标志在何处？

CCS v5.2： 在 CCS 5.2 中很容易找到这个选项。 请看 C2000_Compiler -> Processor Options 下的内容。如果你将 "CLA Support" 设定为 cla0，此开关 --cla_support=cla0 将被添加到构建选项中。

CCS v4： 这个选项在 project->properties->C/C++ Build, Tools Settings Tab, C2000 Compiler 下： 运行时间模型选项。 如果你将 "CLA Support" 设定为 cla0，此开关 --cla_support=cla0 将被添加到构建选项中。

CCS v3.3： 这个选项在编译器构建选项之下 (project->build options->compiler tab)。 如果你将 "CLA Support" 设定为 cla0，此开关 --cla_support=cla0 将被添加到构建选项中。

********** 注释 **********

v5.2.2 和 v5.2.3 codegen 工具的构建接口略去了 5.2.1 中提供的高级标签和 CLA Support 选项。 使用安装目录中的卸载程序来卸载之前的 C2000 编译器版本，下载更新的 5.2.3b 版本，并重新安装。

71. 我的 CLA 和主 C28x 代码可以驻留在同一项目中吗？

可以！ 可以驻留在一起，并且也应该如此！ 二者处于同一项目中使得变量和常量共享更加简单。 汇编程序将通过其独特的指令记忆法了解 CLA 代码，而编译器将通过其独特的扩展名 (.cla) 熟悉 CLA C 语言文件。唯一的限制就是 CLA 代码必须在其自己的汇编程序段内（使用 .sect 汇编命令）。 没有其他方面的限制。

72. 有针对 CLA 的编译器吗？

有。 编译器在 C28x codegen v6.1.0 中被引入。 CLA 架构不支持完整的 C 语言编译器，所以 CLA C 语言编译器具有某些如 CLA C 语言编译器维基网页 中所述和 C28x 编译器参考指南 www.ti.com/lit/spru514 中描述的限制。

CLA 寄存器

73. 我如何查看 CLA 寄存器？

CCS v5： 单击 CLA 调试会话然后选择： View->Registers. CLA 寄存器将显示在寄存器窗口中。

CCS v4： 单击 CLA 调试会话然后选择： View->Registers. CLA 寄存器将显示在寄存器窗口中。

CCS v3.3： 在 CLA 调试窗口中，你可以在 View->Registers->CLA 内查看 CLA 寄存器。 此外，由于它们是存储器映射的，你可以将它们添加到主 CPU 调试器窗口中的观察窗口内。

74. 我不知道如何以浮点格式查看 CLA 结果寄存器？

CCS v5： 右键单击寄存器窗口中的 "value"。 选择 "number format"，然后选择 "float"

CCS v4： 右键单击寄存器窗口中的 "value"。 选择 "format"，然后选择 "float"

CCS v3.3： 这一点已经被提交为改进请求。 目前，你可以用 CLA 开头的名称将结果寄存器添加到观察窗口中。 例如： CLA_MR0 或 CLA_MR3. 在观察窗口中，你可以将类型更改为 32 位浮点。 **更新： 这已经在评估工具的构建中修复。**

75. 为什么所有 CLA 寄存器回读 0？

CLA 寄存器受到代码安全模块 (CSM) 的保护。 请确保 CSM 未被锁定，以便查看寄存器。

76. 我为什么不能在调试器中编辑 CLA 执行寄存器 (MPC, MAR0, MAR1, MR0-MR3...)？

这些寄存器只能由主 C28x CPU 读取。 由于所有调试访问由主 CPU 处理，所以你不能在调试器中编辑它们。

77. MPC 和 CLA 调试窗口中的 "core PC" 间的区别是什么？

MPC 是寄存器本身；它是 CLA 程序空间开始的一个偏移。 MPC 指向当前处于 CLA 管线 D2 阶段的指令。 Code Composer Studio 还需要一个“核心 PC” ("core PC") 寄存器来管理并且跟踪反汇编窗口和源代码指令执行。 你将注意到 "core PC" 与实际存储器位置相对应。 例如，如果 CLA 程序空间从 0x9000 开始，而 MPC 的开始位置为 0x0120，那么 "core PC" 将为 0x9000+0x0120 = 0x9120。

断点、步进、运行、检查符号

78. 我已经按下 CLA 调试窗口中的指令执行（或运行）按钮，而 CCS 发出一个错误。 这是怎么了？

CLA 有可能空闲（也就是说，一个任务未被执行），或者正在运行（即，未在一个断点上暂停）。 为了发出一个“~~指令~~单步执行”或一个“运行”，CLA 应该执行一个任务（即，MIRUN != 0），并且 MPC 应该在一个断点上暂停（MDEBUGSTOP 指令）。

79. 我的 CLA 空闲 (MIRUN == 0)。 我如何在调试时启动一个任务？

有 3 种方式启动任务。 1) 外设中断 2) 主 C28x IACK 指令或 3) 写入 MIFRC（强制）寄存器。

启动一个希望单步执行的任务的最简单方法有可能是在调试窗口中写入强制 (MIFRC) 寄存器。 位 0 对应中断 1 / 任务 1，位 1 对应中断 2 / 位 2 等。

80. 在某些示例中，C28x 使用函数启动任务： Cla1ForceTask2andWait(). 这个函数的作用是什么？

这是一个在 Cla_defines.h 文件中定义的宏，在调试 CLA 代码时提供帮助。 每个任务有一个宏（即，Cla1ForceTask1andWait()，Cla1ForceTask2andWait()，等等。。。）

宏进行以下操作：

- 发出 IACK 指令来启动特定任务
- 等待几个周期（请见下一个问题）
- 轮询 MIRUN 位来“等待”，直到任务完成。

也有不用“等待”的版本。 ClaForceTask1() 等等。。。

81. 我已经使用 Cla1ForceTask8andWait() 宏。 C28x 不用“等待”且继续执行。有什么问题吗？

C28x 将轮询 MIRUN 位来查看任务是否仍在执行。 如果任务未启动，那么 C28x 将继续执行并且不等待。

如果情况如此，请检查：

- 宏使用 IACK 指令来启动一个任务。 请确保你已经在 MICTL 寄存器中启用这个功能。
- 请确保中断在 MIER 寄存器中被启用。

如果任务被启动，但是 C28x 仍然未等待：

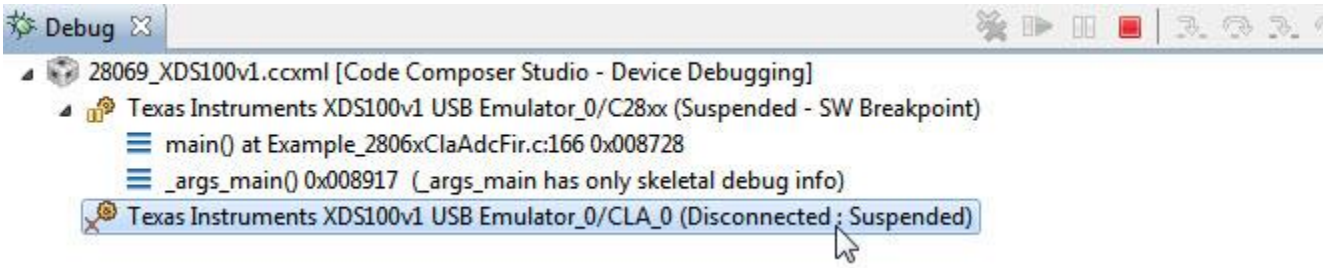
- 在某些示例代码中，Cla1ForceTaskxandWait() 宏在 IACK 指令和轮询 MIRUN 位之间只有两个 NOP。如果 C28x 过早地轮询 MIRUN 位，它仍将查看这个位是否被清零。 为了改正这个问题，在轮询 MIRUN 位前添加一个额外的 NOP。

82. 我如何启用和禁用 CLA 断点？

CCS v5.2：

- 在调试视图中，打开 "debug" 窗口 (View->Debug)。

- 单击调试窗口中的 CLA 处理器。 这将自动地把运行环境切换至 CLA 调试。 谨记，在 CLA 中为“断开”。这意味着 CLA 断点（MDEBUGSTOP 指令）被禁用且被 CLA 视为 MNOP。



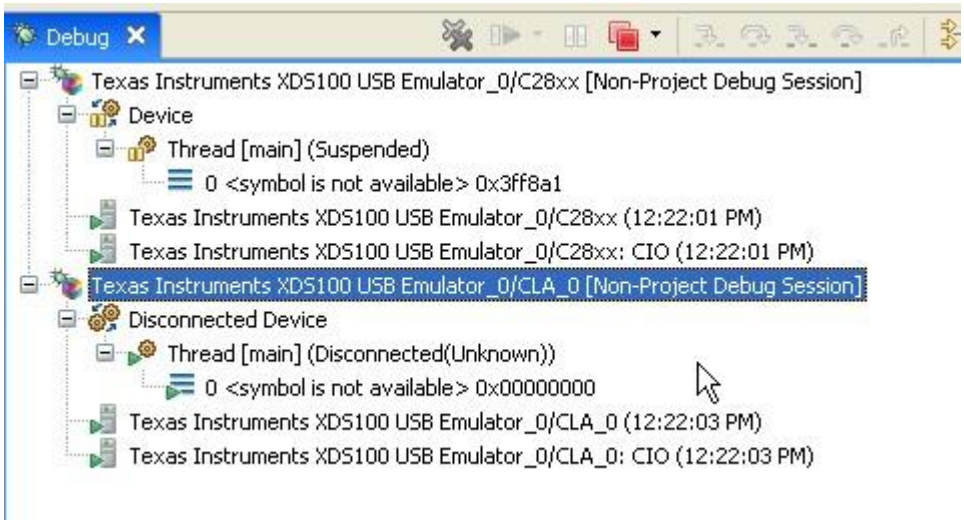
- 选择： Run->Connect Target（或者右键单击并选择 "Connect Target"）。 这将连接 CLA 调试器且启用 CLA 断点。 调试窗口中的状态将更改为表示 CLA 不再被断开。 当你连接 CLA 时，调试器也将自动连接 C28x。 这是所需要的运行方式；要调试 CLA，你还必须连接 C28x。

- 要禁用 CLA 断点，你将断开 CLA 调试窗口（Run->Disconnect Target 或者右键单击并选择 "Disconnect Target"）。 在这个情况下，CLA 断点将被视为 MNOP（无运算）。 在你断开前，请确保发出一个“运行”。 如果 CLA 在你断开前未自由运行，它将在你离开的位置保持暂停状态，并且将不执行其他任务。

CCS v4：

- 在调试视图中，打开“调试”窗口 (View->Debug)

- 单击调试窗口中的 CLA 处理器。 这将自动把运行环境切换至 CLA 调试。 谨记，在屏幕截图中，CLA 被“断开”。 这意味着 CLA 断点（MDEBUGSTOP 指令）被禁用，并被 CLA 视为 MNOP。



- 选择： Target->Connect Target. 这将连接 CLA 调试器，并且启用 CLA 断点。 调试窗口中的状态将更改为显示 CLA 不再被断开。 当你连接 CLA 时，调试器也将自动连接 C28x。 这是所需要的运行方式；要调试 CLA ，你还必须连接 C28x。

- 要禁用 CLA 断点，你将断开 CLA 调试窗口（Target->Connect Target - 这个选项在连接或断开之间进行转换）。 在这个情况下，CLA 断点将被视为 MNOP（无运算）。 在你断开前，请确保发出一个“运行”。 如果 CLA 在你断开前未自由运行，它将在你离开的位置保持暂停状态，并且将不执行其他任务。

CCS v3.3：

谨记：在 CCS v3.3 评估工具中，构建之前，断开 CLA 窗口不会禁用 CLA 断点。 这个问题已在构建中解决。

- 首先从调试管理器中打开 C28x 调试窗口

- 使用 Debug->connect 或 alt-c 命令连接 C28x 调试窗口。 在你连接 CLA 之前，C28x 必须被连接。

- 下一步，从调试管理器中打开 CLA 调试窗口

- 当你执行一个连接（Debug->connect 或 alt-c）CLA 调试窗口操作时，它将自动启用 CLA 断点。 当然，这一操作的前提假定 CLA 时钟已经被启用。 因此，我们已经在 Code Composer Studio gel 文件（用于具有 CLA 的器件）的 OnReset() 函数中添加一个 CLA 时钟使能。 这个 gel 函数在你每次复位器件时执行。

- 要禁用 CLA 断点，你将断开 CLA 调试窗口（debug->disconnect 或 alt-c）。 在这个情况下，CLA 断开将被视为 MNOP（无运算）。 在你断开前，请确保发出一个“运行”。 如果 CLA 在你断开前未自由运行，它将在你离开的位置保持暂停状态，并且将不执行其他任务。

83. 我如何将一个断点插入到 CLA 代码中？

- **汇编代码：** 首先将 MDEBUGSTOP 指令添加到你希望暂停的代码位置上。 然后，重建并加载代码。 请牢记，MDEBUGSTOP 指令一定不能在一条分支 (MBCNDD)，调用 (MCCNDD) 或返回 (MRCNDD) 指令的 3 条指令（之前或之后）以内。 当 CLA 暂停时，MDEBUGSTOP 指令将在管线的 D2（解码 2）阶段内。 你可以从此处单步执行 CLA 或运行到下一个断点或 MSTOP。
- **C 语言代码：** 使用 __mdebugstop() 固有函数来插入一个断点。 然后重建且载入代码。 当 CLA 暂停时，MDEBUGSTOP 指令将在管线的 D2（解码 2）阶段内。 你可以从此处单步执行 CLA 或运行到下一个断点或 MSTOP。

84. 为什么我必须重建和载入我的代码来插入一个 CLA 断点？

- 对于 C28x 主 CPU，当你插入一个断点时，调试器将在运行中插入一个 ESTOP0（C28x 断点）指令，然后在你暂停后用最初的操作码替代它。 C28x 管线被设计成无缝实现此操作。
- 然而，相对于 C28x 管线，CLA 管线被简化。 当遇到 MDEBUGSTOP（CLA 断点）指令时，管线在解码 2 阶段的断点出现时暂停。 此管线未被清空，并且 PC 未被向前移动。 此外，一旦你再次开始指令执行，指令将不会再重新取指令。 因此，MDEBUGSTOP 必须被插入代码中，并且重建代码。

85. 我可以在 CLA 调试窗口中查看符号吗？

完全可以！

CCS v5： 首先在调试窗口中单击 CLA 处理器。 然后，使用 Run->Load->Load Symbols 菜单载入符号。

CCS v4： 首先在调试窗口中单击 CLA 处理器。 然后，使用 Target->Load Symbols 菜单载入符号。

CCS v3.3： 在 CLA 调试窗口中，使用 File->load symbol 菜单来载入符号。 另外一个建议是也将 Code Composer Studio 项目载入，或者在 CLA 调试窗口中打开 CLA 汇编文件。 通过这个方法，调试器将为你显示单步执行时源代码中的位置。 谨记：如果你的代码图像发生变化，并且符号发生移动，那么你需要在这个窗口中重新载入正确的符号。

86. 我如何单步执行或运行 CLA？

一旦 CLA 在一个断点上暂停 (MDEBUGSTOP)，你可以使用 CLA 调试窗口中的正常指令执行和运行按钮。如果你已经将源代码和符号载入，你可以使用源代码单步执行。

87. CLA 调试窗口中的 Run->Reset->CPU Reset 功能是什么？

这将执行一个 CLA 的软复位。

在之前的 CCS 版本中为 debug->reset。

88. 如果我在一条指令前暂停，在我看到指令结果前，我必须多次执行 CLA。为何会这样呢？

这是由 CLA 的简化管线造成的。

当调试 C28x 代码时，你已经注意到管线在每次汇编单步执行时被清空。 这意味着每条指令在整个管线内被压入，这样的话，你可以在指令执行后立即看到指令的结果。

另一方面，CLA 管线在你每次执行指令时只向前移动一个周期。 这条由 CLA 程序计数器 (MPC) 表示的指令目前位于管线的 D2（解码 2）阶段中。 如果你指令执行一次，它将移动至 R1（读取 1）。 如果你再次执行，它将移动至 R2（读取 2）。 在第三次执行后，它将在 EXE（执行）阶段中。

89. 我已复位主 CPU 并运行。 一个外设或代码已启动任务，但是 CLA 并未在一个断点上停止。 有可能是什么原因呢？

当你复位器件时，CLA 断点被禁用，并且 CLA 时钟本身也被禁用。 为了使调试器重新启用 CLA 断点，CLA 时钟也需要被重新启用。 要自动完成这一操作，你可以在 GEL 文件的 OnReset() 函数中添加一条语句来启用 CLA 时钟。

注释： 更新版本的 GEL 文件已经将其包括在内。

90. 调试器允许 CLA 和主 CPU 的锁步单步执行吗？

你可以分别单步执行主 CPU 和 CLA。

91. 我发出了一条“运行”命令，但是并未在 MSTOP 上停止，而 CLA 开始执行另外一个任务。 为什么呢？

如果在你运行到 MSTOP 时，有一个任务等待并被启用，那么这个任务将自动开始执行。 为了防止这一情况发生，你可以修改 MIER 寄存器，这样的话，无任务被启用。

92. 我如何才能配置 CLA 代码？

不能通过 Code Composer Studio IDE 来直接完成配置。 但是，你可以使用一个空闲 PWM 模块来配置你的系统。 你可以使用一条指令来强制这个空闲 PWM 输出在 CLA 任务的开始位置上为高电平，然后在 CLA 任务的末尾强制 PWM 输出为低电平。 可使用 PWM 的 AQCFRCS 寄存器内的 CSFA 或 CSFB 位来完成 PWM 输出电平强制。 然后，可以使用一个示波器来测量完成一个特定任务的时间量。 谨记，在 CLA 任务触发值出现到 CLA 任务开始之间的时间内将使用 6 或 7 个额外周期。

其他问题

93. 为什么我的 Code Composer Studio GEL 文件在复位时启用 CLA 时钟？

当你复位器件时，CLA 断点被禁用，CLA 时钟本身也被禁用。 为了更加轻松地调试 CLA 代码，调试器将监视复位条件，并在复位后重新启用 CLA 断点。 正因如此，CLA 时钟本身必须被重新启用。 如果你不喜欢这个运行方式，你可以从 GEL 文件中注释掉或删除此行。

94. 我想从调试窗口中启动一个 CLA 任务，我该怎么做？

你可以使用 CLA 强制寄存器 MIFRC 来启动任务。 这个寄存器在 CLA 寄存器窗口中提供。

95. 我可以用 CLA 调试窗口，而非主 CPU 调试窗口加载代码吗？

可以，虽然你有可能看到一些奇怪的运行方式，但是这不会对调试环境造成任何影响。 例如，CLA 调试窗口也许尝试设定一个程序末尾断点，而它不能进行这一操作。在这个情况下，它将发出一个错误。 你可以将 CLA 窗口中的选项更改为在你加载程序时不设定断点。 CLA 调试窗口也可以将其源代码/反汇编显示设定至 C28x 代码的进入点。

96. CLA 程序存储器/反汇编窗口在 CLA 执行时回读全 0。 为何会这样呢？

在大多数情况下，你也许不会注意到这一点。 当 CLA 程序存储器被映射到 CLA 存储器空间时，CLA 取指令具有比 CPU 调试读取更高的优先级。 因此，CLA 有可能在 CLA 处于循环执行中时永久阻断 CPU 调试访问。 这有可能在最初开发 CLA 代码时出现，原因是一个导致无限循环的错误。 为了避免锁定主 CPU，程序存储器将在 CLA 处于运行中时对 CPU 的调试读取返回全 0x0000。 当 CLA 被暂停或空闲时，可以执行到 CLA 程序存储器的正常 CPU 调试读取和写入访问。

如果 CLA 被困在一个无限循环中，你可以使用软复位或硬复位来退出。 主 CPU 的复位也将从此情况中退出。

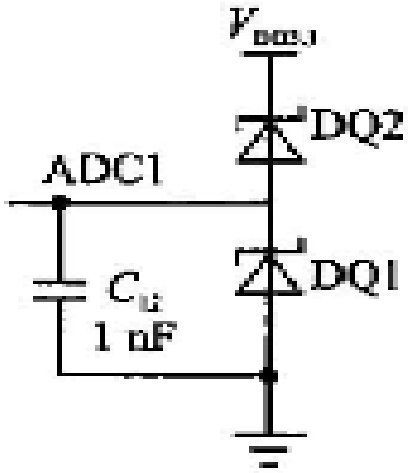
其他 C2000 常见问题

97. TMS320F2812 的低功耗模式有哪几种？如何退出低功耗模式？

- 空闲模式（IDLE） 只要把低功耗模块控制寄存器 LPMCR[1~0]（LPMECR 的 D0、D1 位）都设置成 0，那么低功耗模块 LPM 将不执行任何工作。处理器可以通过有效的中断或 NMI 中断来退出空闲模式。
- 暂停模式（HALT） 把低功耗模块控制寄存器 LPMCR[1~0]（LPMECR 的 D0、D1 位）分别设置成 1、0 或 1、1。只有复位 XRS 和 XNMI 外部信号能够从暂停方式唤醒 CPU。CPU 可以通过 XMNICR 寄存器来使能或者禁止 XNMI。
- 备用模式（STANDBY） 把低功耗模块控制寄存器 LPMCR[1~0]（LPMECR 的 D0、D1 位）分别设置成 0、1。所有在 LPMCR1 寄存器中被选中的信号，包括 XNMI 信号，都可以将 CPU 从备用方式唤醒。在唤醒处理器之前，要通过 OSCCLK 确定被选定的信号，OSCCLK 的周期数可以通过 LPMCR0 寄存器确定。

98. TMS320X2812 芯片 A/D 保护及校正电路如何设计？

TMS320X2812 模拟电压输入范围为 0~3V，但实际中使用 TMS320X2812 的 A/D 端口采样信号时，并不能保证所采集的信号在输入范围之内。由于 A/D 模块非常脆弱，当小于 0V 或者大于 3V 的信号输入 TMS320X2812 的 A/D 端口时可能会损坏 A/D 端口，使相应的 A/D 采样端口不能正常工作。A/D 保护电路可采用如图所示的钳位电路设计。



99. TMS320X2812 芯片外扩 RAM 如何设计？

TMS320X2812 芯片内部具有 18KX16 位的 RAM 空间，当程序代码长度大于 18KX16 位时，调试程序不方便，可以通过外扩 RAM 的方法来解决。在设计时，可以通过 TMS320X2812 的外部接口 XINTF 来外扩存储器，XINTF 是一种非多路选通的异步总线。关于 TMS320X2812 的 XINTF 接口介绍可以到官网 http://www.ti.com/product/tms320f2812?keyMatch=TMS320F2812&tisearch=Search-EN 下载相关数据手册查看。

100. 如何保证 TMS320X2812 系统正常工作？

TMS320X2812 芯片对电源要求很敏感，电源达不到工作电压或者操作不对，都有可能导致 TMS320X2812 不能正常工作。为保证 TMS320X2812 系统能正常工作，必须注意以下几点：

- 在每次上电之前，一定要检查电源与地是否相通。电源与地直接连接在一起，将导致板子损坏。
- 电源芯片产生的电压要稳定在 3.3V 和 1.9V。电源芯片上的电容的不匹配，有可能导致电源芯片里面的振荡电路工作一段时间后不再振荡，或者振荡频率对应的不是所要求输出的电压值。为解决这一问题，在设计电源时除了需要考虑电源的散热问题之外，还要考虑电容匹配问题。计算之后多次测量，取最佳值。
- 要按正常的步骤来运行或停止仿真器，带电停止或运行仿真器都有可能造成运行环境的死机。
- 根据数据手册设计复位电路，复位电路的设计错误也会导致系统不能正常运行。

101. F28M3x 器件硬件都没问题，但 JTAG 连接不上，识别不到芯片？

如果不用外部复位，直接利用 F28M3 器件内部上电复位功能的话，Reset 引脚必须外接一个上拉电阻到 Vcc，否则就会出现 JTAG 无法识别芯片的问题。

102. C2000 产品对于环境容忍度是多少？

TI 并没有公开提供这个指标，对于 C2000 芯片来说并不会有太大影响（如果一定要给个值，那么 85%RH，这是 JEDEC standards）,看到 C2000 芯片的 MSL-3 指标，不过该指标指的是芯片从防潮袋中取出暴露在空气中应该在 168 个小时内焊接到开发板上，以保证焊接可靠，这 168 个小时中周围的环境应为 ≤30 °C/60%RH。

103. TI DSP 采用片内 RAM 有哪些优点？

片内 RAM 同片外存储器相比,有以下优点：

- 片内 RAM 的速度较快,可以保证 DSP 无等待运行。
- 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。
- 片内 RAM 运行稳定,不受外部的干扰影响,也不会干扰外部。
- 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。

104. C2000 使用过程中出现周期性复位现象该如何解决？

系统硬件和软件问题都可能会引起周期性复位的问题。首先，要排查硬件原因，检查电源设计供电电流是否足够，电源管脚上的电容值是否合理，电路上是否有短路现象。另外就是检查软件设计问题，是否是看门狗复位导致，引导程序是否有 bug，调用子程序时候返回值是否会导致程序跑飞。以上方法可以排查出大部分周期性复位的原因。

105. C2000 系列的内部 AD 的最大输入范围是多少？

C2000 系列的内部 AD 的输入范围是 0-3V，管脚最大能承受的电压是 3.3V，超出 3V 以上的转换结果为最大值（4096）。

106. 常看到用 DSP（C2000 系列的）做产品的都带一个 CPLD，请问 CPLD 是必须的吗？

不是必须的。一般使用 CPLD 有两个目的。一个是扩展 IO（通用输入输出），一个是匹配时序。一般说来，如果项目所需 IO 较少，完全没有必要使用 CPLD，对速度要求不高的输入输出也没用必要使用 CPLD，对速度要求高但有可以使用的通用器件时（典型的应用是用总线扩张 IO）也可以使用通用器件，比如 245 等锁存器。

107. 如何正确使用内部 AD 并提高其精度？

TI 的 MCU 的内部 AD 一般都是 12 位的，可以满足大多数实时控制的一般需求，伪了保证其精度，一般可采取下列几种方法：

- 模拟部分和数字部分分开供电，模拟电源和数字电源分开是 AD 转换系统中最常见的做法。在实际应用中，如果不方便使用单独电影，可以把数字电源加磁珠隔离后供给模拟电电源。
- 线路板走线注意数字部分和模拟部分尽量不要有交叉和重叠部分，如果不可避免，尽量使用十字交叉（使用不同层），避免数字走线和模拟走下有较长的平行走线，尽可能的避免其直接的耦合。
- 为了提高 AD 的非线性，可以使用使用两个 AD 通道对其余通道进行校正，具体作法是利用两个 AD 通道对两个已知并且稳定的电压进行采样，利用简单的公式对其他通道进行校正。详细相信参考 TI 官方资料（暂时没有链接，等待 TI 给出）。这样做的好处是提高了 AD 的非线性误差，缺点是可以使用的 AD 通道减少了两个，同时计数量增大了。
- 为了适应较大的动态范围，保证宽量程时的精度，可以使用两个设置更多通道对同一原始信号的不同档位进入测量，利用不同通道的系数计算出正确的值。具体作法是把输入信号分成不同档位，即不同的放大（或者衰减）倍数，分别接入到不同的通道测量，不同的通道对应不同的档位，根据测量数据的范围选取合适的通道系数，从而保证了较宽量程，达到 16 位或者更高位数 AD 的动态范围。

108. C2000 系列能跑操作系统么？

可以支持，TI 推荐使用官方的 BIOS 实时操作系统，TI 官网可以找到这方面的教程，国内也有中文的教材讲这个操作系统的，可以试试。

109. C2000 系列除了用 JTAG 仿真工具烧写芯片以外，还有没有其他快速量产的方式？

原则上，C2000 利用 bootRom，可实现多种启动方式，这些方式都可以用来开发烧写芯片，将程序下载到 ram 中运行，即可实现烧写功能，市面上也有编程器厂商支持已经 C2000 量产烧写，烧写速度很快，但需要使用 hex 文件，out 文件不支持。

110. C2000 中每个系列都有多种芯片可供选择，如何快速进行选型呢？

选型的前提就是在某种系列中，比较各个芯片的差异，看是否能满足设计要求，以 Piccolo 2802x 系列为例，差异在于支持的最高频率不同，flash 容量不同，SRAM 容量不同，ADC 转换速率不同，是否支持高分辨率 ePWM 通道。

111. 当试图连接一个加密的 C2000 器件，CCS 提示错误，连接不上仿真器，如何解决？

在具有仿真代码安全逻辑电路(ECSL) 的器件上不支持硬件复位等待模式（如 Piccolo 器件）。如果此类器件加密了，当器件上电时，CPU 将开始运行并可执行一个指令来访问一个受保护的 ECSL 区域。如果这一情况发生，ECSL 将发生错误并使仿真器连接被断开。

解决办法：

1. 断开仿真器
2. 设置 boot 引脚到 wait 等待 boot 模式，注意：2833x/2823x 器件上是“loop to check”boot 模式
3. 复位器件
4. 重新连接仿真器

112. CCS 单步调试 C2000 时有些语句不执行，也不能设置断点，是什么原因？

有可能是编译器优化掉了。

建议：

1. 查看汇编，看看有无相应的操作
2. 关闭优化
3. 加 volatile 关键字

113. C2000 有的样片为什么是 TMX 打头？

每一颗 TMS320MCU 商用系列产品具有以下三个前缀中的一个：TMX，TMP 或者 TMS. 其中 TMS 是完全合格的产品器件，也是我们最常见到的，TMX 是试验器件，不一定代表最终器件的电气规范标准，TMP 是最终的芯片模型，符合器件的电气规范标准，但是未经完整的质量和可靠性验证。使用时需加注意。

114. C2000 的复位脚出现了周期性复位现象，怎么解决？

首先，排查硬件原因，检查电源设计供电电流是否足够，电源管脚上的电容值是否合理，电路上是否有短路现象，另外就是检查软件设计问题，是否是看门狗复位导致，引导程序是否有 bug，调用子程序时候返回值是否会导致程序跑飞。以上方法可以排查出大部分周期性复位的原因。

115. C2000 的 Piccolo 系列器件的电源供电管脚有 3.3V 的 VDDIO 以及 1.8V 的内核供电 VDD 管脚，在设计电路时，需要供应两路电压么？

Piccolo 器件有内部电压稳压器模块 VREG，可通过管脚 VREGENZ 来启用或者禁用，当 VREGENZ 为低电平时启用 VREG，此时内部产生 1.8V 的内核电压，外部无需再接 VDD 电源，只有当 VREGENZ 为高电平时，外部需另外提供 1.8V 内核电压。

116. C2000 的 2802x Piccolo 系列的 Bootloaler 支持哪些启动方式？

通过 GPIO37,GOIP34 和 TRST 管脚控制不同的 boot 模式,其中仿真引导（Emulation Boot）模式下，再根据 EMU_KEY 和 MEU_BMODE 的值定义不同的 boot 仿真模式,而取模式（GetMode）下，根据 OTP 的两个地址上的值 OTP_KEY 和 OTP_BMODE 来决定不同的 boot 模式,可见，Piccolo 系列的 Bootloaler 配置模式灵活多样。

	GPIO37 TDO	GPIO34 CMP2OUT	TRST	
Mode EMU	x	x	1	Emulation Boot
Mode 0	0	0	0	Parallel I/O
Mode 1	0	1	0	SCI
Mode 2	1	0	0	Wait
Mode 3	1	1	0	GetMode

117. Piccolo MCU 中的 CLA 技术有何优势？

CLA 是浮点控制律加速器技术，它是一款 32 位浮点数学加速器，可独立执行控制环路算法，并可与 C28-CPU 并行工作。从而将该 CPU 解放出来用以处理 I/O 与反馈回路，可使通用闭环应用性能提高五倍。此外，CLA 自带中断控制器，并可直接访问 ADC 与 EPWM 等外设。

118. 如何降低 DSP2812 电源纹波？

采用线性电源，模拟地和数字地加磁珠隔离，模拟电源和数字电源也用磁珠隔离。

119. DSP2812 烧录 FLASH 锁死如何解决？

每次联编完成，先不要烧 FLASH，全工程查找 003f7ff8-003f7fff8 个字节的数据，确保 PASSWDS 的 used 为 0，否则会烧进密码，把 DSP2812 内部的 FLASH 锁死。 如果已经被锁死，不要更改 DSP 的源程序，使用 CCS3.3 的在线 CPU 仿真模式，重新装载运行程序，此时看 View—Memory，看密码区被烧录进的数据，前 8 个数据即为实际烧录进 DSP 的密码，这样就可以解锁了。

120. 为什么 LM3S9B96 的开发板，16M 的外部晶振用示波器测试没有波形呢？9B96 不是没有内部震荡源吗？（程序可以正常下载、调试、运行）

LM3S9B96 是有内部振荡器的,所以用不用外部晶振对芯片的仿真使用都不影响。必须在 RCC[MOSCDIS] 中先置一使能外部晶振才能使外部晶振起振，才能测试得到震荡时钟。

你的代码中要设置时钟源，在你的代码中加入下面这句话:

ROM_SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_XTAL_16MHZ | SYSCTL_OSC_MAIN);

121. 请问 2812 和 28335 的区别有哪些？

28335 是个非常出色的芯片。我以前用过 2808 和 2812，其实 28335 是他们的综合体。

1、28335 借鉴了 2808 的 hrpwm，而且有 6 个。做电源很好。

2、28335 的内部 adc 比 2812 要好，外部引脚排序给人一种很舒服的感觉，可以节省 cpld 了。

总之，28335 很好。多了 FPU，硬件支持单精度浮点运算；

FLASH 翻倍，达到 256kWord；

除了 M0 外，内部 SRAM 翻倍；

XINTF 重新组织了，更加合理。

122. 812 的片内 AD 转换有几个问题：

1、从 AD 转换被触发到一次 AD 转换完毕发生中断这个过程有多长的时间？

2、AD 转换的结果在结果寄存器中是怎么保存的？不管设置的是用哪个通道转换，结果都是从 RESULT0 寄存器开始保存的吗？一次保存多少个数据？

3、我只设置了一个通道来进行转换，而且每次都是从 RESULT0 来把结果读到一个数组中，这个数组的长度设置有没有什么标准？我发现 AD 转换一次，这个存结果的数组中发生变化的个数很多而且每次都不一样，请问这正常吗？

1、这个设置一个断点，可以看两次中断之间转换执行了多少个时钟周期，就可以知道转换时间了

2、结果寄存器共 12 位，低四位无效，最小值对应 0，最大值对应 3

3、数组的长度大于等于 12 位就行了，当没有接转换模拟电压时，数组中的值当然变化很大，因为是随机的

123. 为什么有时候 C2000 对 GPIO 读写操作不稳定？

C2000 对 GPIO 的控制主要通过 GPxDAT、GPxSET、GPxCLEAR、GPxTOGGLE 四个寄存器完成。GPxDAT 叫做数据寄存器，可以用于对多组 GPIO 的读写操作，但是如果用 GPxDAT 寄存器对单个 GPIO 读写的时候会干扰到邻近的 GPIO 口的状态。所以对单个 GPIO 控制的时候需使用 GPxSET、GPxCLEAR、GPxTOGGLE 寄存器才能实现对单个 GPIO 的正常操作。

124. 如何在非标准频率下设置 C2000 ECAN 的波特率参数？

在非标准频率下设置 C2000 ECAN 的波特率参数的例子：

Calculation of the CAN bit timing :

System clock： f_sys = 24 MHz

System clock period： t_sys=1/f_sys = 41.666667 ns

BRP = 23

Desired bit rate is 125KBit/s, desired bit time is 8000 ns

CAN time quantum： tq = 24 * t_sys = 1000ns

Actual bit time = 8 * tq = ~= 8000 ns

Actual bit rate is 124999.999bit/s ~= 125Kbit/s

CAN bus length = 10 m, with 5 ns/m signal delay time.

Propagation delay time : 2 * (transceiver loop delay + bus line delay) = 400 ns

Prop_Seg = 1 * tq = 1000ns > 400ns.

Sync_Seg = 1 tq .

Phase_seg1 + Phase_Seg2 = (8-2) tq = 6 tq

Phase_seg1 = 3 tq and Phase_Seg2 = 3 tq

SJW = (min(Phase_Seg1, 4)) tq = 3 tq

TSEG1 = (Prop_Seg + Phase_Seg1 - 1) = 3

TSEG2 = (Phase_Seg2 - 1) = 2

SJW_p = (SJW - 1) = 2

Clock tolerance df :

A: df < min(Phase_Seg1, Phase_Seg2) / (2 * (13*bit_time - Phase_Seg2))

B: df < SJW / (20 * bit_time)

A: df < 3/(2*(13*11-3)) = 3/280 = 1.0714%

B: df < 3/(20*11) = 3/220 = 1.3636%

125. C2000 系列 SPI 通信，非 16bits 通信如何数据对齐？

2802x 系列的 C2000 内部 SPI 通信数据寄存器是 16bit 的，而且发送的数据是左对齐的。所以，当 spi 通信不是 16bit 时，应向将待发送的数据左移到 MSB，再发送相应位数。

126. 我是 C2000 的初学者，我如何用更加高效和快速来开发我的项目？

如果你正在使用 C 语言进行编程（现在很常见），在开始使用全新微控制器平台时最困难的是了解外设。CPU 本身不是问题，这是因为代码由 C 语言编写。因此，研究 TI 提供的可能性，使你在使用这里的 C2000 外设时更加轻松，其中包括：？ 示例代码： TI 提供很多针对大部分 C2000 器件的示例代码？ Code Composer Studio：Code Composer Studio 包含一整套用于开发和调试嵌入式应用的工具。它包含了用于优化的 C/C++ 编译器、源码编辑器、项目构建环境、调试器、描述器以及多种其他功能。直观的 IDE 提供了单个用户界面，可帮助您完成应用开发流程的每个步骤。熟悉的工具和界面使用户能够比以前更快地入手。？ controlSUITE：用于 C2000 微处理器的 controlSUITE 是一套全面的软件基础设施和软件工具集，旨在最大程度地缩短软件开发时间。从特定于器件的驱动程序和支持软件到复杂系统应用中的完整系统示例，controlSUITE 在每个开发和评估阶段都提供了程序库和示例。超越简单的代码段 - 立即使用实用型软件来开始设计您的实时系统。

127. DSP28335 编译时出现错误，怎么办？

安装 setup_C28XFPU_CSP_v3[1].3.1207.exe 浮点支持库然后在 CCS 里面设置一下打开 CCStudio v3.3 软件，选择菜单 “help” ”about” ”ComponentManager” ”Build Tools“ ”TMS320C28XX “ 选择新版 Code Generation Tool ，”Texas Instruments C2000 Code Generation Tools ＜v5.1.0＞ “ , ，关闭 CCStudio v3.3 软件，再打开 CCStudio v3.3 软件，有效！！！；对 F2833X 浮点 DSP,需设置浮点库：在 CCS3.3 软件中，Project/Build Option:Compiler/Advance/FloatingPointSupport:选“fpu32”Linker/Libraries/Incl. Libraries:加“rts2800_fpu32.li

128. 如何从 C2000LAUNCHPAD 中把 XDS100V2 仿真器接出来？如果用这个现成的仿真器怎么接出来？

把那两个 ISO7231 焊下来，下面是电阻的焊盘，从下面的焊盘把对应的信号线引出来就行了。当然这样就不能直接连后面的 28027 了，可以自己想办法，比如把电阻焊盘上用 0 欧电阻连起来，同时把信号引出来接一个接插件头，这样可以直接调试板子上的 28027，当把 LAUNCHPAD 的电源跳线跳开时就可以当普通仿真器用了。

板子上的 ISO7231 是起隔离作用的，但是如果把电源掉线连上，仿真器和 28027 用同一个电源，隔离就没有意义了， 可以直接用 0 欧电阻或者小电阻连起来。可以看一下官方的原理图。

129. 想用 C2000Lanuchpad 做四轴飞行器的主控板，然后加一个外设板。但是测试了很久读不出来 MPU6050 的数据。需要额外设置什么？

方案一，加上拉电阻

方案二，注意读写时 SDA 引脚的输入输出方向

130. C2000 产品对于环境湿度的容忍度是多少？

TI 并没有公开提供这个指标，您可以认为对于 C2000 芯片来说并不会有太大影响（如果一定要给个值，那么 85%RH，这是 JEDEC standards）

我估计之所以会有这一问，是看到 C2000 芯片的 MSL-3 指标，不过该指标指的是芯片从防潮袋中取出暴露在空气中应该在 168 个小时内焊接到开发板上，以保证焊接可靠，这 168 个小时中周围的环境应为 ≤30 °C/60%RH。

131. C2000 能当单片机使么？

如果不需要以太网（F28M35x 会有），就可以当成单片机来用，但是成本可能稍微有点高，如果有必要，建议选用低端的 Piccolo 系列。

132. C2000 实时支持库支持哪些 C 函数？

如库的说明一样，支持标志 C 函数库，所以我认为跟其他书籍的标志 C 函数库的函数也是一样的。

另外在编程的时候也可以多参考附件关于编译器的说明文档。

133. 我想知道有哪些代码开发工具可用，以及我如何调试针对 CLA 的代码？

请参考 C2000 CLA 调试 FAQ。

134. 2837xS 微控制器(MCU)什么样的产品？

2837xS 微控制器是功能强大的单核系列产品。这些单核 MCU 是业界首个可提供 4 个 16 位模数转换器(ADC)并且能在电源控制应用中进行精准反馈的产品。它们与最近发布的双核 C2000 Delfino F2837xD MCU 引脚和软件兼容，同时还能帮助加速从性能较高的工业控制应用向中等控制设计扩展时的开发过程。此外，对于那些使用前款 Delfino F2833x MCU 系列的客户而言，Delfino F2837xS MCU 是新一代单核解决方案。开发人员可将 F2833x MCU 现有的投资过渡到软件兼容的 F2837xS MCU，从而提供更高的 CPU 性能和更先进的模拟和控制外设。

135. F2837xS MCU 的特性与优势？

- C28x 与实时控制加速器(CLA)的强强联合，兼具 400MIPS 的浮点性能，可同时快速高效地管理多个控制任务。C28x CPU 在三角学和复杂数**算方面得到了进一步加速：利用在 C28x 内核中集成并自动在编译器中执行的全新三角数学单元(TMU)硬件加速器来快速执行用于转换与控制函数中的三角算法。
- 利用也是在 C28x 内核中集成并自动在编译器中执行的 Viterbi 复杂单元(VCU II)硬件加速器来加速编码通信应用中常见的复杂数**算。
- 通过从主 CPU(C28x 内核)向 CLA 卸载要求很高的控制回路分析来进行智能系统分区，从而产生额外的带宽并允许主处理器重点执行如系统诊断、应用管理或降频控制回路等其它任务。
- 可通过 4 个 16 位 ADC 在电源控制应用中进行精准反馈 —— 对单核 MCU 而言，这在业界尚属首例。
- 可提高系统吞吐量，如监测电机的三相电压和电流，同时完成软件解码高频解析器反馈。
- 可通过其它高完整性模拟和控制外设来最大化系统级集成并节省材料清单成本，例如 Σ-Δ 解调器、比较器、数模拟转换器(DAC)以及许多控制和通信外设。
- 8 个已调试的 Δ-Σ 通道，都可进行阈值比较，并配置了用于 TI AMC1204 隔离 Δ-Σ 调制器的无缝接口。
- 可从传统 F2833x MCU 轻松迁移。使用相同的 C28x 内核和许多相同的外设，以前的高性能 C2000 MCU 与全新 Delfino F2837xS MCU 软件兼容，从而提供更多的信号处理带宽、更佳的闪存性能，还可提供高级模拟和控制外设。
- 通过在全新的 F2837xS MCU 系列中创建不同性能等级的产品为双核 Delfino F2837xD MCU 和未来器件提供引脚和软件兼容的架构来加速产品上市进程并降低开发成本。
- 通过 controlSUITE™软件(可在 ti.com 免费下载)中的示例、头文件、应用库等来进行集中式软件开发。

136. C2000 如何生成 hex 烧写文件？

CCS 一般生成 out 文件直接下载使用，而其他的编程工具不支持 out 文件格式，所以需要将 out 文件转换成通用的 hex 格式文件，分两种情况：

1. CCS3.3 生成 HEX

1）找到 CCS3.3 的安装目录。在 C2000 目录找到一个叫 hex2000.exe 的可执行文件。如：

C:\CCStudio_v3.3PLA\C2000\cgtools\bin

2）把要转的 xxx.out 文件拷到这个目录下。

3）在电脑上进入 start->run->cmd 进入命令行窗口，用 DOS 命令进入 hex2000.exe 的安装目录。

4）键入 hex2000 -romwidth 16 -memwidth 16 -i-o xxx.hex xxx.out 命令就会在当前目录下生成一个叫 xxx.hex 的文件。

2. CCSv5 生成 HEX

1）在 IDE 中选择 菜单 project->propertise 打开界面。

2）在左侧选中 CCS Build，在右侧的选项卡中点击 Steps，在最下方 Apply Predefined Step 选中 Create flash image:Intel-HEX

3）在编译的时候会自动生成一个 xxx.hex 文件。

137. CCS 单步调试 C2000 时有些语句不执行，也不能设置断点，是什么原因？

有可能是编译器优化掉了。建议：1. 查看汇编，看看有无相应的操作；2. 关闭优化；3. 加 volatile 关键字。

138. C2000 系列能跑操作系统么？

可以支持，TI 推荐使用官方的 BIOS 实时操作系统，TI 官网可以找到这方面的教程，国内也有中文的教材讲这个操作系统的，可以试试。

139. C2000 系列除了用 JTAG 仿真工具烧写芯片以外，还有没有其他快速量产的方式？

原则上，C2000 利用 bootRom，可实现多种启动方式，这些方式都可以用来开发烧写芯片，将程序下载到 ram 中运行，即可实现烧写功能，市面上也有编程器厂商支持已经 C2000 量产烧写，烧写速度很快，但需要使用 hex 文件，out 文件不支持。

140. C2000 中每个系列都有多种芯片可供选择，如何快速进行选型呢？

选型的前提就是在某种系列中，比较各个芯片的差异，看是否能满足设计要求，以 Piccolo 2802x 系列为例，差异在于支持的最高频率不同，flash 容量不同，SRAM 容量不同，ADC 转换速率不同，是否支持高分辨率 ePWM 通道。

141. 当试图连接一个加密的 C2000 器件，CCS 提示错误，连接不上仿真器，如何解决？

在具有仿真代码安全逻辑电路(ECSL) 的器件上不支持硬件复位等待模式（如 Piccolo 器件）。如果此类器件加密了，当器件上电时，CPU 将开始运行并可执行一个指令来访问一个受保护的 ECSL 区域。如果这一情况发生，ECSL 将发生错误并使仿真器连接被断开。解决办法：

1．断开仿真器

2．设置 boot 引脚到 wait 等待 boot 模式，注意：2833x/2823x 器件上是“loop to check”boot 模式

3．复位器件

4．重新连接仿真器

142. C2000 有的样片为什么是 TMX 打头？

每一颗 TMS320MCU 商用系列产品具有以下三个前缀中的一个：TMX，TMP 或者 TMS. 其中 TMS 是完全合格的产品器件，也是我们最常见到的，TMX 是试验器件，不一定代表最终器件的电气规范标准，TMP 是最终的芯片模型，符合器件的电气规范标准，但是未经完整的质量和可靠性验证。使用时需加注意。

143. C2000 的复位脚出现了周期性复位现象，怎么解决？

首先，排查硬件原因，检查电源设计供电电流是否足够，电源管脚上的电容值是否合理，电路上是否有短路现象，另外就是检查软件设计问题，是否是看门狗复位导致，引导程序是否有 bug，调用子程序时候返回值是否会导致程序跑飞。以上方法可以排查出大部分周期性复位的原因。

144. C2000 的 Piccolo 系列器件的电源供电管脚有 3.3V 的 VDDIO 以及 1.8V 的内核供电 VDD 管脚，在设计电路时，需要供应两路电压么？

Piccolo 器件有内部电压稳压器模块 VREG，可通过管脚 VREGENZ 来启用或者禁用，当 VREGENZ 为低电平时启用 VREG，此时内部产生 1.8V 的内核电压，外部无需再接 VDD 电源，只有当 VREGENZ 为高电平时，外部需另外提供 1.8V 内核电压。

145. C2000 的 2802x Piccolo 系列的 Bootloaler 支持哪些启动方式？

通过 GPIO37,GOIP34 和 TRST 管脚控制不同的 boot 模式

	GPIO37 TDO	GPIO34 CMP2OUT	TRST	
Mode EMU	x	x	1	Emulation Boot
Mode 0	0	0	0	Parallel I/O
Mode 1	0	1	0	SCI
Mode 2	1	0	0	Wait
Mode 3	1	1	0	GetMode

其中仿真引导（Emulation Boot）模式下，再根据 EMU_KEY 和 MEU_BMODE 的值定义不同的 boot 仿真模式

Address	Name	Value	
0x0D00	EMU_KEY	if TRST == 1 and EMU_KEY == 0x55AA, then check EMU_BMODE for the boot mode, else { Invalid EMU_KEY Boot mode = WAIT_BOOT }	
0x0D01	EMU_BMODE	0x0000	Boot mode = PARALLEL_BOOT
		0x0001	Boot mode = SCI_BOOT
		0x0002	Boot mode = WAIT_BOOT
		0x0003	Boot mode = GET_BOOT (GetMode from OTP_KEY/OTP_BMODE)
		0x0004	Boot mode = SPI_BOOT
		0x0005	Boot mode = I2C_BOOT (1)
		0x0006	Boot mode = OTP_BOOT
		0x000A	Boot mode = RAM_BOOT
		0x000B	Boot mode = FLASH_BOOT
		Other	Boot mode = WAIT_BOOT

(1) I2C boot uses GPIO32 and GPIO33 which are not available on all packages.

而取模式（GetMode）下，根据 OTP 的两个地址上的值 OTP_KEY 和 OTP_BMODE 来决定不同的 boot 模式

Address	Name	Value	
0x3D 7BFE	OTP_KEY	GetMode will be entered if one of the two conditions is true: Case 1: TRST == 0, GPIO34 == 1 and GPIO37 == 1 Case 2: TRST == 1, EMU_KEY == 0x55AA and EMU_BMODE == GET_BOOT GetMode first checks the value of OTP_KEY: if OTP_KEY == 0x55AA, then check OTP_BMODE for the boot mode else { Invalid key: Boot mode = FLASH_BOOT }	
0x3D 7BFF	OTP_BMODE	0x0001	Boot mode = SCI_BOOT
		0x0004	Boot mode = SPI_BOOT
		0x0005	Boot mode = I2C_BOOT (1)
		0x0006	Boot mode = OTP_BOOT
		0x0007	Boot mode = CAN_BOOT
		Other	Boot mode = FLASH_BOOT

(1) The I2C boot loader uses GPIO32 and GPIO33 which are not available on all packages.

可见，Piccolo 系列的 Bootloaler 配置模式灵活多样。

146. 可以用串口烧写 2812 片上 flash 吗？

C2000 系列支持串口烧写片上 flash，注意与 bootmod 有关的引脚是否按要求上下拉。在软件中设置选择正确的 232 串口，握手过程会占用一定的时间，烧写速度可以参考软件配套的文档进行设置。注意所有烧录入 flash 的数据必须定义在正确的 page 里。