C2000 CLA FAQ

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 等等**	MUI16TOF32 MRa，mem16 MI32TOF32 MRa，mem32 MF32TOI32 MRa，MRb	1

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	4 (MR0 - MR3)	8 (R0H - R7H)

存器		
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 与 C28x+FPU 相似的记忆法，但是以 'M' 开头 ex: MMPYF32 MR0, MR1, MR2	浮点指令是除（扩展集） C28x 定点指令之外的指令
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)