在CCS上调试C2000 CLA FAQ

1. 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

2. 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

3. 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/

4. 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

5. 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。

6. 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。

7. 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。

8. 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。

9. 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

10. 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。

11. 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。

12. 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。

13. 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

14. 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。

15. 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。**

16. 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。

17. 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。

18. 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

19. 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）。

20. 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。

21. 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。。。

22. 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：

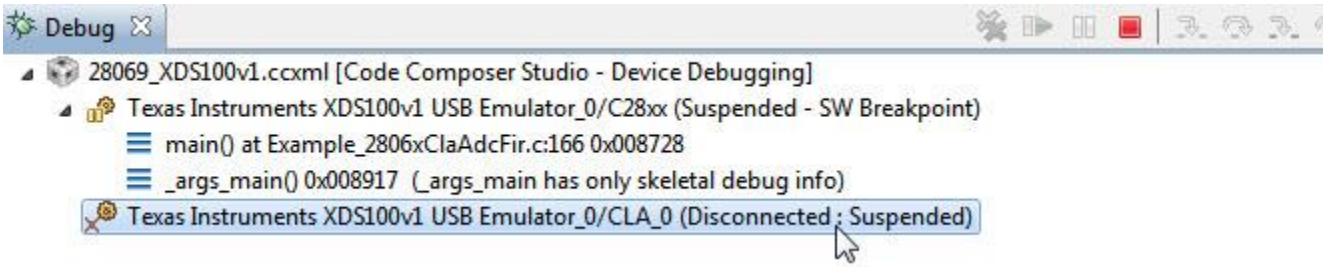
- 在某些示例代码中，Cla1ForceTaskxandWait() 宏在 IACK 指令和轮询 MIRUN 位之间只有两个 NOP。 如果 C28x 过早地轮询 MIRUN 位，它仍将查看这个位是否被清零。 为了改正这个问题，在轮询 MIRUN 位前添加一个额外的 NOP。

23. 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。



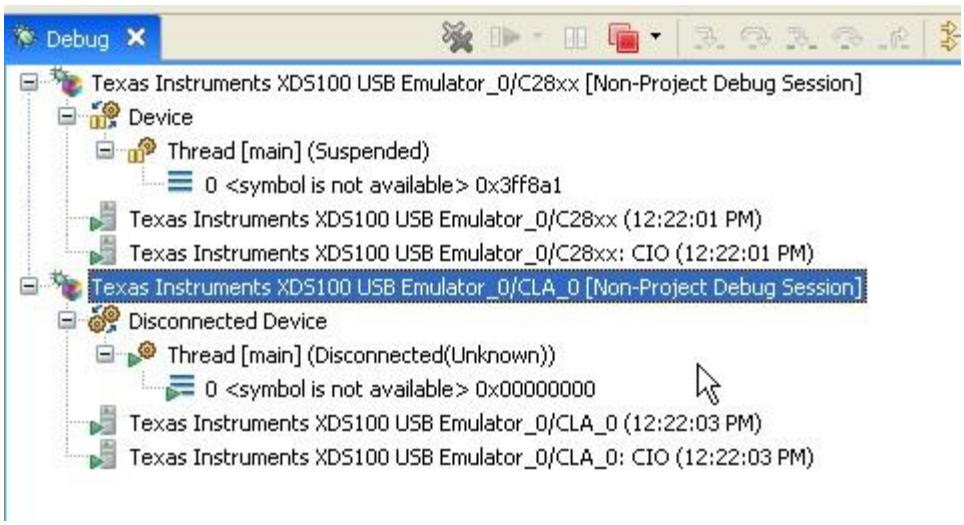
- 选择： Run->Connect Target（或者右键单击并选择 "Connect Target"）。 这将连接 CLA 调试器且启用 CLA 断点。 调试窗口中的状态将更改为表示 CLA 不再被断开。 当你连接 CLA 时，调试器也将自动连接 C28x。 这是所需要的运行方式；要调试 CLA，你还必须连接 C28x。

- 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。

24. 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。

25. 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。

26. 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。

27. 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。

28. 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。

29. 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。

30. 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。

31. 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。

32. 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。

33. 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

34. 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。

35. 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。

36. 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。

37. 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。