

应用说明

停止功能程序模板

AN00190-004

编写的代码示例说明在允许其它程序任务继续执行的同时处理 **Mint** 停止事件的一种简单方法



简介

Mint 是一种顺序语言（即前一代码执行完以后指定的代码行才会执行），它也使用户能够执行并行处理（通过多任务处理）和事件/中断执行操作（例如：通过数字输入事件），使语言整体上非常灵活，并且特别适合于机器和运动控制应用。

很多时候，即使在紧急停止情况下，应用程序也必须能够继续处理某些代码元素（例如，HMI 输入/输出、一般机器逻辑），这通过 **mint** 很容易实现，本应用说明如何实现以及基本 **Mint** 程序模板如何在大多数应用程序中再次使用。这种模板以 **Mint** 代码示例的形式提供结合本应用说明文档，它适用于 NextMove e100、MicroFlex e190 和 MotiFlex e180 控制器。它也可在 **Mint Workbench** 集成的 **Mint** 标准库中直接拖拽使用。

如果使用较旧的（经典或传统）运动产品，请联系您当地的 ABB 运动控制支持团队。

模板要求

Mint 模板的总体要求如下：

- 响应停止输入事件
- 在适用的情况下停止轴上的运动
- 确保停止输入事件不会被再次调用，直到停止输入被清除然后重新激活
- 确保在停止输入清除之前不会有进一步的运动

我们将在后面的章节中详细介绍在 **Mint** 程序代码中是如何实现这些功能的。

Mint 停止事件

Mint STOPINPUT 关键字允许用户配置用以连接来自机器的紧急停止信号的数字输入。通常，这种输入是低电平有效（即当输入未连接时被认为是有效的），因此是“失效保护型”。这样一来，**INPUTACTIVELEVEL** 中的相应位被设置为零。

Mint 关键字 **STOPMODE** 可用于配置当停止输入被激活时停止动作的执行模式。

在任何情况下，如果停止发生控制器就会调用‘Event Stop’。

通常使用一个常数来定义用作停止输入（以使代码更具可读性并便于此输入的更改，只需对代码进行最小限度的修改）。

对于我们的样本程序（专为在虚拟控制器上运行而设计），我们已将数字输入通道 0 分配为 **Mint** 停止输入。

下面是停止输入的相关配置

‘将输入 0 设为停止输入通道...

Const _nStopInput As Integer=0

‘将相关输入指定为停止输入...

STOPINPUT(0)=_nStopInput

‘设置控制器，在停止输入激活时快速停止轴...

STOPMODE(0)=_smCRASH_STOP（或 STOPINPUTMODE(0)=_siCANCEL，对于非 e100 控制器）

‘配置输入为低电平有效

INPUTACTIVELEVEL(0)=01111111111111111110

（请注意，在这种情况下，不需要将输入 0 配置为边沿触发）。

‘定义停止事件处理器，当停止输入激活时可供调用...

Event Stop

doHandleStop

End Event

当输入事件发生时，为了方便起见，将待执行代码置于一个子程序中（以方便重复调用）。

doHandleStop 子程序执行以下操作...

- 设置一个全局变量（bStopActive），用于整个应用程序，以确定停止输入是否处于活动状态。然后可使用该变量来防止程序执行（例如，防止运动）
- 取消运动（这是可选的，作为 STOPMODE/STOPINPUTMODE 响应动作）
- 等待轴变为 IDLE（即运动停止）
- 调用负责结束所有任务的一个子程序（doEndTasks），在紧急停止激活的情况下不得执行任何功能（在我们的例子中，这只是负责移动轴的任务，但实际上可能不仅限于任务执行动作）。
- 轴去使能
- 取消分配 Mint STOPINPUT（将它设为-1）。这可确保 Mint 停止事件不会被反复调用，并且最初定义为 STOPINPUT 的数字输入保持激活状态
- 运行（StopHandler）负责确定何时清除紧急停止的任务

StopHandlertask

当 Mint stop event 被激活时 MintStopHandler task 启动，Mint StopHandler 任务开始。现在，该任务等待与紧急停止关联的数字输入停用。一旦该输入停用，任务可以：

- 重新分配 Mint STOPINPUT（因此，如果输入激活，Stop Event 即可以再次发生）
- 清除全局变量(bStopActive)在程序代码中的互锁）

主程序

主程序分配 STOPINPUT，并在启动时初始化 bStopActive 的状态。这是为了确保每次重启时能够正确地初始化。

示例代码测试

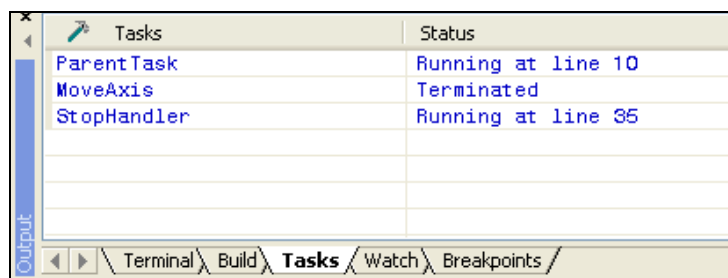
示例模板程序是基于虚拟控制器而编写，它包含允许在单个轴（0 轴）上启动运动的元素。示例代码可适应在任何应用中（以及任何数量的轴）中使用。

启动 Workbench，启动一个新项目并选择虚拟控制器。

从 File> Open File ... 菜单选择示例程序（它将出现在编辑器窗口）。

选定 Program>Compile（程序>编译），下载并运行它，从而将该文件下载到虚拟控制器。

在示例程序中，输入 0 被定义为停止输入（并且被默认为低电平有效）。由于没有连线到虚拟控制器，停止输入初始会被激活。“任务”观察窗口显示主程序和 StopHandler 正在运行..



Tasks	Status
ParentTask	Running at line 10
MoveAxis	Terminated
StopHandler	Running at line 35

如果将变量 ParentTask::bStopActive 输入观察窗口，您会发现它被设为 1。

在 Workbench 命令窗口中输入以下指令（并按回车键）...

INPUTACTIVELEVEL (0) =INPUTACTIVELEVEL (0) 或 1

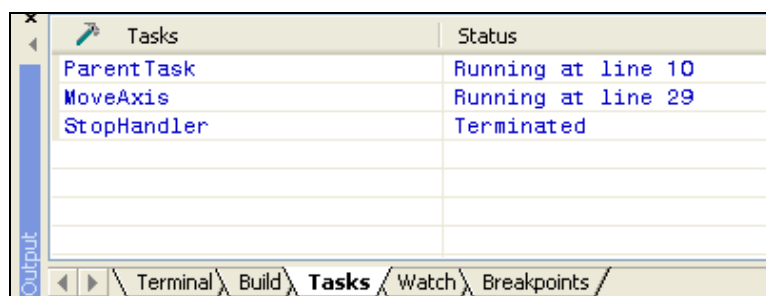
这将输入 0 设置为高态有效（退出停止输入）。您将从任务观察窗口看到 StopHandler 任务终止，并且变量观察窗口将显示 bStopActive 目前设置为 0。

现在，停止输入处于非激活状态，在 Workbench 命令窗口输入以下指令（并按回车键）...

COMMS(1)=1

从 Mint 程序以外写入 Comms（1）会导致 Mint 级别的 Comms1 事件发生（如有）。在我们的示例程序中，我们使用该事件来启动 MoveAxis 任务（如果它尚未运行）。

任务观察窗口将显示 MoveAxis 正在运行，而监视窗口可用于监视当索引产生时 0 轴的位置和速度变化...



Tasks	Status
ParentTask	Running at line 10
MoveAxis	Running at line 29
StopHandler	Terminated

现在，在 Workbench 命令窗口中输入以下指令（并按回车键）...

INPUTACTIVELEVEL(0)=INPUTACTIVELEVEL(0)和 0xFFFFFE

这会将输入 0 恢复为低电平有效。因此，停止输入被激活，0 轴上的运动将停止，任务 MoveAxis 会被终止，任务 StopHandler 将重新启动，变量 bStopActive 会被设为 1（_True）。

尝试在命令窗口再次写入 Comms（1）。您会发现没有运动发生。这是因为任务 MoveAxis 开始时的互锁检测到停止处于活动状态并立即结束任务...

If bStopActive=_trueThen End MoveAxis

这就是完整的应用说明。现在，如果提供示例代码，您应当能够理解和再使用它。

如果您希望将此代码存储在自己的 **Mint Library** 中（以便于将来拖入编辑器窗口），请使用鼠标/键盘选定/突出显示编辑器中的所有代码，然后将它拖入编辑器窗口的“**Mint Library**”方框。为此代码段输入一个合适的名称，然后，可以在需要时将其拖出库和拖入 **Mint** 应用程序。

Workbench 提供的标准库包含已命名为“**Stop Event Handler (ALL)**”的核心模板。

联系我们

要了解更多信息，请联系您当地的 ABB 代表或登录以下网站：

www.abb.com/motion
www.abb.com/drives
www.abb.com/drivespartners
www.abb.com/PLC

© ABB 公司，2012 年，版权所有。保留所有权利。

技术规格如有变更，恕不另行通知。