运动控制产品

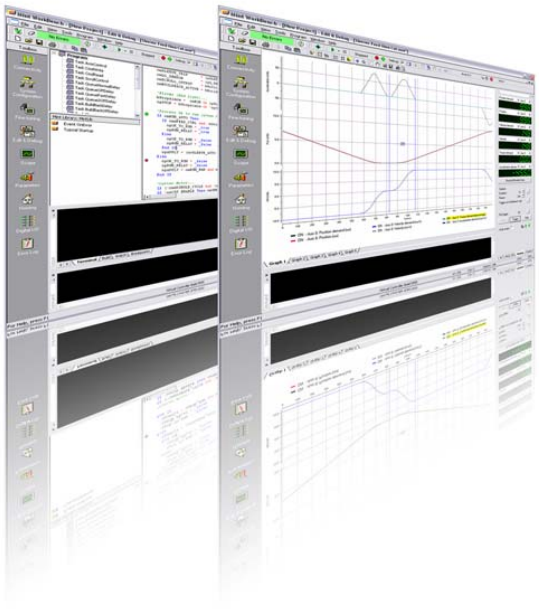
应用说明文档
多任务的基础知识

AN00107

版本 C（中文）

MintMT 引入了轻松编写包含多个任务的程序的能力。该工具功能强大，能用于开发控制多个轴的代码，免除了为各个轴人工交错发送命令的需要。任务不限于运动控制，可用于任何目的，例如 PLC 逻辑控制和/或用户界面。

分别调整各任务优先级的能力使用户能够维持对各任务的响应能力。例如，可向 HMI 任务分配较低的优先级，确保其不会阻碍更为重要的运动控制任务。



简介

任务通过关键词“Task”与“End Task”宣告。这些关键词包含任务运行时将要执行的语句，也可用于设置任务名称。请注意，在 MintMT 中，每个对象（无论是变量、子程序、定义，甚至任务）必须有唯一名称。以下示例简单展示了如何宣告任务。

```
Task myTask
  Print #1 “INX.0 = ”, INX(0)
End Task
```

执行任务

在 MintMT 中，默认初始仅执行父任务，所有其它任务开始均处于待用状态。实际上，父过程负责启动需运行的任务。这通过 Run 命令实现，可能会使用需启动的任务的名称。

```
Run myTask
```

```
Task myTask
  Print #1 “INX.0 = ”, INX(0)
End Task
```

请注意，因为默示执行过程在最后一个父任务命令完成后终止，因此 Run 命令之后无需有明确的 End 语句。这确保不会机械执行以下任务、子程序、功能或事件。

等待任务终止

以往的程序存在的问题在于任务在启动后立即停止！您可能会问“为什么”，这是因为一旦父任务终止，所有子任务也会立即终止。为解决这一问题，必须利用 Pause 命令使父任务一直等待，直至后面跟随的语句变为“真”。等待任务终止的最佳方式是利用 TaskStatus 功能监控任务的状态。任何任务将会是以下状态之一：
_tskTerminated
_tskRunning
_tskSuspended
_tskWaiting
示例中的程序变为：

```
Run myTask
Pause TaskStatus(myTask) = _tskTerminated

Task myTask
  Print #1 “INX.0 = ”, INX(0)
End Task
```

使任务连续执行

示例程序的当前形式并不十分有用，因为其在终止前仅会打印一次数字输入的值。这表示任务会持续运行至最后一个语句（End Task 定界符之前的语句），然后终止。这可能必要，但在该示例中，我们希望连续每隔 500 ms 打印一次。

```
Run myTask
Pause TaskStatus(myTask) = _tskTerminated

Task myTask
  Loop
    Print #1 “INX.0 = ”, INX(0)
    Wait = 500
  End Loop
End Task
```

Pause 语句现在有一点冗余，至少在当前形式下如此，可以安全地用“Pause _false”替代。尽管如此，我们会继续保留这种终止形式，但会为 HMI 添加另一任务。

利用 HMI 任务控制执行过程

典型地，一个应用程序会有多个主要用于控制机器的任务，还会有另外一个（例如）提供人机界面(HMI)的任务。该 HMI 任务可能是子任务，也可能是父任务。选用何种任务取决于个人喜好，但使用子任务时，可以将 HMI 特有的数据隐藏在任务中。其对以下程序看上去并无优势，但对使用菜单以及可能使用子菜单的较复杂用户界面确有优势。

```
Run myTask, hmiTask
Pause TaskStatus(hmiTask) = _tskTerminated

Task myTask
  Loop
    Print #1 “INX.0 = ”, INX(0)
    Wait = 500
  End Loop
End Task
```

```
Task hmiTask
  Print “Press ‘X’ to exit”
  Pause InKey(1) = ‘X’
End Task
```

请注意，在以上程序中，我们仅需监控 HMI 任务的状态，因为其它任务始终连续运行。

任务间的通信

有时，一项任务可能需与另一项任务共享信息，这可通过全局变量实现。全局变量在父任务中宣告，且在所有模块中可见，无论变量本身是任务、事件、子程序、功能还是启动块。如果修改 HMI 以调整输入位，从而在程序运行过程中进行监控，则应解释如何操作。
注：避免两种任务修改同一变量值十分重要。否则，由于排程器对多任务应用时间分段以及命令会交错，您可能不会得到预期的结果。

```
Dim nBit As Integer = 0

Run myTask, hmiTask
Pause TaskStatus(hmiTask) = _tskTerminated

Task myTask
  Loop
    Print #1 “INX.”, nBit, “ = ”, INX(nBit)
    Wait = 500
  End Loop
End Task

Task hmiTask
  Print “Press a digit to set the bit to monitor”
  Print “Or ‘X’ to exit”
  Loop
    Pause InKey(1)
    Select Case LastKey
      Case '0' To '7'
        nBit = LastKey - '0'
      Case 'X'
        Exit Task
    End Select
  End Loop
End Task
```

以上程序的优势在于规模小，便于看清变量 nBit 的作用，但程序较大时，所有变量的作用没有那么明显。程序越大，这一问题越越凸显。为避免全局数据增长，MintMTV 中的块允许数据在某一模块中被宣告，这会使其它模块无法访问数据，确保数据不会意外在其它地方被使用，因此有利。

您可能想了解这对任务之间共享数据有何用处，实际上，可通过范围覆盖操作符(::)使“被隐藏”的本地数据可见。该操作符用双冒号将变量名称与其前面的模块名称隔开，使变量所在的模块能够限定变量。以下程序展示了这一过程。

```
Run myTask, hmiTask
Pause TaskStatus(hmiTask) = _tskTerminated
```

```
Task myTask
  Dim nBit As Integer = 0
  Loop
    Print “INX.”, nBit, “ = ”, INX(nBit)
    Wait = 500
  End Loop
End Task

Task hmiTask
  Print “Press a digit to set the bit to monitor”
  Print “Or ‘X’ to exit”
  Loop
    Pause InKey(1)
    Select Case LastKey
      Case '0' To '7'
‘* Access the ‘nBit’ variable that belongs to ‘myTask’ using the override operator (::)
        myTask::nBit = LastKey - '0'
      Case 'X'
        Exit Task
    End Select
  End Loop
End Task
```

注：范围覆盖操作符(::)仅可用于访问任务、事件与启动类模块中的数据，不能用于访问子与功能类模块中的数据。

修改任务优先级

任务开始的优先级始终默认为十，这适合大多数应用程序，但有时某任务的执行速度可能有必要高于其它任务。反过来，某任务的执行速度小于其它任务可能更为合适。高优先级任务之一是控制运动的任务，低优先级任务可能是 HMI。

给出的示例程序实际不会因调整两个任务的优先级有任何收获，仅用于解释该过程如何执行。以下程序给 HMI 任务设置了低优先级，给输入监控任务设置了高优先级。

```
TaskPriority myTask, 20      ‘This task is important
TaskPriority hmiTask, 2      ‘This task is less important
Run myTask, hmiTask
Pause TaskStatus(hmiTask) = _tskTerminated
‘Remaining code as the previous example
```

联系我们

如需更多信息，请联系
您所在地的 ABB 代表或访问以下任一网站：

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