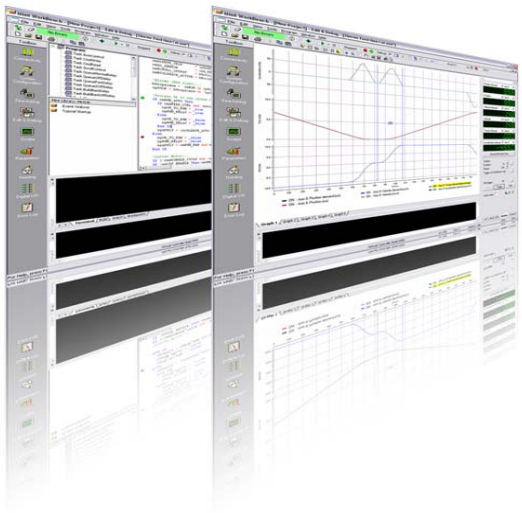
运动控制产品

应用说明文档
使用 Mint 中的时间功能与计时器

AN00214

版本 B（中文）

使用 Mint 内置的时间功能或创建自己的计时器对象，以满足应用需求。



简介

大多数应用通常要求实现以下某种或多种功能：

- 在继续执行程序前等待（规定的时间）
- 定期执行某种动作
- 对过程进行计时
- 在某条件得到满足的时间达到规定的时间时执行某种操作
- 在某条件得到满足的时间未达到规定的时间时执行某种操作

所有这些功能均能通过 Mint 编程语言轻松实现。Mint 提供一系列易于使用的时间相关功能和关键词，这些功能和关键词会在本应用说明文档中加以详述。此外，您还可以（通过结构的使用）创建自己的计时器对象（例如：创建功能非常类似于 IEC61131 计时器的计时器对象）。本应用说明文档提供了示例。

等待

Mint 是一种顺序语言（与循环 PLC 系统相反），因此在代码模块中执行 Mint 时，下一行在上一行完成后才会执行。这使得 Mint 成为了设置机器序列的理想之选，因为无需像使用循环系统时一样为处理序列中的每个元素而创建复杂的“机器状态图”。当然，Mint 非常灵活，如果需要，也可创建含内部状态图的循环任务。

在序列中的某一处，您可能需要等待规定的时间（例如：为了使气缸有时间伸展）。为此，Mint 提供 WAIT 命令，该命令采用以 ms 表示的时间参数。

例如：
开启输出 0，等待 250ms，然后将轴 2 移动 100 用户单位

```
OUTX（0） = _on
WAIT（250）  ‘suspend code in this module for 250ms
MOVER（2） = 100 : GO（2）
```

可以使用的 WAIT 命令的数量不限，可以按需求在整个程序中分布该命令。

定期执行某种动作

Mint 提供单一计时器事件。事件间隔用关键词 TIMEREVENT 设置。该关键词将时间（单位：ms）作为参数。

例如：
利用计时器事件，根据封装的编码器值（换算为 mm）每秒发生的变化计算机器速度（m/min）...

```
‘Main program
Dim fEncoderSnapshot
Dim fLastEncoderSnapshot
Dim fMachineSpeed

TIMEREVENT = 1000

Loop
 …
End Loop

Event TIMER
  fEncoderSnapShot = ENCODER（0）
  fMachineSpeed = 0.06 * WrapOffset（fLastEncoderSnapshot,fEncoderSnapShot,0,ENCODERWRAP（0））
  fLastEncoderSnapShot = fEncoderSnapShot
End Event
```

为禁用计时器事件，将 TIMEREVENT 设为 0。

对过程进行计时

Mint 主程序（父任务）与每一个 Mint 子任务均有各自的时间记录（单位：ms），该记录通过关键词 TIME 访问。例如，一个任务的 TIME 可设为 0，然后在需要时问询/读取，这完全独立于另一个任务的 TIME（可通过该任务内的逻辑复位与问询）。

因此，每个任务可独立地通过关键词 TIME 对某个过程进行计时。

例如：
测量气缸伸展所需的时间（例如：开启某输出，测量获得输入所需的时间）

```
Dim nExtendTime As Integer
TIME = 0
OUTX（1） = _on
PAUSE INX（4）
nExtendTime = TIME
```

对于运行 14 及以上版本的固件的控制器（例如：使用 5454 及以上版本的 NextMove ESB-2、e100、e180 和 e190 产品），也可宣告“时间”类型变量。

例如：Dim tTimer1 As Time

可为这些变量设置一个数值（例如：重置为 0），程序可问询这些变量的值（范围取决于变量在程序中的宣告位置）。可使用的时间变量的数量仅受程序内可用空间的限制。

在某条件得到满足的时间达到规定的时间时执行某种操作

有多种方法可用于实现该功能。一种方法是使用某种循环结构...

例如：
在输入 5 连续 3 秒开启的情况下继续执行程序...

```
Dim tConditionTimer As Time

tConditionTimer = 0
Repeat
  If INX（5） = _off Then tConditionTimer = 0
Until tConditionTimer >= 3000
```

该示例适用于希望一直“中断”代码，直至某条件在一定时间内始终为“真”的情况，但可能需要在执行程序剩余部分的“同时”监控该条件...在这种情况下，需使用 Mint 任务。

例如：
Mint 程序运行后台任务，该任务在检测到输入 5 连续 3 秒开启后开启输出 1...

```
‘Main program…
Run （ConditionMonitor）
…

Task ConditionMonitor
  Dim tConditionTimer As Time
  tConditionTimer = 0
  Repeat
    If INX（5） = _off Then tConditionTimer = 0
 Until tConditionTimer >= 3000
 OUTX（1） = _on
End Task
```

对于在某条件未得到满足（假）的时间达到规定时间的情况下采取措施的情况，代码与以上代码非常类似，仅需将逻辑颠倒，重置计时器。

创建 IEC 计时器功能

我们可以利用 Mint 关键词 STRUCTURE 创建自己的计时器对象...

```
Structure T_IECTimer
  IN As Integer
  PT As Integer
  ET As Time
  Q As Integer
End Structure
```

以上示例使用 IEC 规定的输入和输出参数的名称。为读取输入库，IN 最好不要与 Mint 关键词冲突，但即使这样，也不会造成问题，因为编译器能够识别。IN 是结构的成员变量（编辑器会改变文本的大小写与颜色，就好像文本被用作 Mint 关键词）。

为此，我们会创建 TON（延时开启）计时器-我们将模拟传感器检测连续的产品队列。如果队列在预设置的时间（示例时间为 2s）内一直存在，我们将设置计时器输出（这可能（例如）被用于启动机器过程）。
首先，必须宣告新类型的变量...

```
Dim tQueue As T_IECTimer
```

现在，需要设置用于开始/停止计时的逻辑...我们的“产品传感器”与输入 6 连接，因此我们在程序中将输入 6 设置成在出现尾部上升后沿时触发（如果需要进一步详细了解输入设置，请参阅 Mint 帮助文件中的 INPUTMODE、INPUTPOSTRIGGER 和 INPUTNEGTRIGGER）。
这样一来，我们可以利用 Mint Event IN6 检测产品传感器开启或关闭的时间...输入开启后，我们将加载预设置的时间（2s）、清除输出以及启用计时器。输入关闭后，我们将禁用计时器、关闭计时器输出以及将经过的时间复位...

```
Event IN6
  If INSTATEX（6） Then
    ‘Rising edge…
    tQueue.IN = _On
    tQueue.PT = 2000
    tQueue.Q = _Off
    tQueue.ET = 0
  Else
    ‘Trailing edge…
    tQueue.IN = _Off
    tQueue.Q = _Off
    tQueue.ET = 0
  End If
End Event
```

我们可以通过创建将计时器初始化的子程序以及将参数传输至该程序来使该代码更加“可重复使用/模块化”。然后，我们可以用该程序来启用/禁用可能包含在程序中的任何计时器...

```
Sub doInitTONTimer （ByRef tTimer As T_IECTimer, ByVal nPreset As Integer, ByVal bEnable As Integer）
  If bEnable Then
    tTimer.IN = _On
    tTimer.PT = nPreset
    tTimer.Q = _Off
    tTimer.ET = 0
  Else
    tTimer.IN = _Off
    tTimer.Q = _Off
    tTimer.ET = 0
  End If
End Sub
```

现在，输入事件可编写为...

```
Event IN6
  If INSTATEX（6） Then
    ‘Rising edge…
    doInitTONTimer （tQueue, 2000, _On）
  Else
    ‘Trailing edge…
    doInitTONTimer （tQueue, 2000, _Off）
  End If
End Event
```

在我们的程序逻辑中（例如：在一直循环执行后台逻辑测试的 Mint 任务中的程序逻辑，这与处理普通逻辑的 PLC 循环任务非常相似），我们可能希望检查计时器是否已完成。为使代码模块化且可重复使用，我们将创建测试是否应开启计时器输出的功能。

该功能会利用“AND”逻辑反馈计时器输出与启用输入的状态。此外，如果除输入事件外的程序逻辑将计时器禁用，我们可以利用该功能禁用计时器...

```
Function doTON （ByRef tTimer As T_IECTimer） As Integer
  If tTimer.IN Then
    ‘Timer is still enabled so check whether on time has elapsed…
    If tTimer.ET >= tTimer.PT Then tTimer.Q = _On
  Else
    ‘Timer has been disabled so turn it off…
    tTimer.ET = 0
    tTimer.Q = _Off
  End If
  doTON = （tTimer.Q AndAlso tTimer.IN）
End Function
```

我们将按以下方式在循环 Mint 任务中使用该功能...

```
Task PLCLogic
  Loop
    If doTON（tQueue） And TaskStatus（StartMachine） = _tskTerminated Then Run（StartMachine）
    …
  End Loop
End Task
```

如果想检查计时器属性（例如：经过的时间），可通过以下代码从程序的任何部分直接访问属性...

例如：nElapsedTime = tQueue.ET

TOF（延时关闭）计时器的逻辑与前述逻辑非常相似；请参阅本应用说明文档中的 Mint 程序示例，获得详细信息。Mint 程序示例设计用于通过作为 Mint Workbench 一部分的 Mint Virtual Controller 测试代码，从而利用 NETDATA 事件而非输入事件来模拟输入。在数据查看窗口或命令窗口中将 NETINTEGER（0）设为 0 或 1，以测试 TON 与 TOF 计时器的运行。将 TON 的预设置时间设为 5s，TOF 的预设置时间设为 3s。这给用户提供了更改 Netdata 值的时间。将 IEC 计时器变量添加到变量查看窗口，以观察计时器输出的运行状态（Q）或查看“StartMachine”任务的状态。除非 NETINTEGER（0）连续 5s 以上 > 0，否则任务不会开始；除非 NETINTEGER（0）至少连续关闭 3s，否则任务不会终止。

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