# AN2001 中断系统的应用

CREATE: 2008/12/30
UPDATE: 2010/06/25
Version 1.1
http://www.plcol.com
http://www.visiblecontrol.com

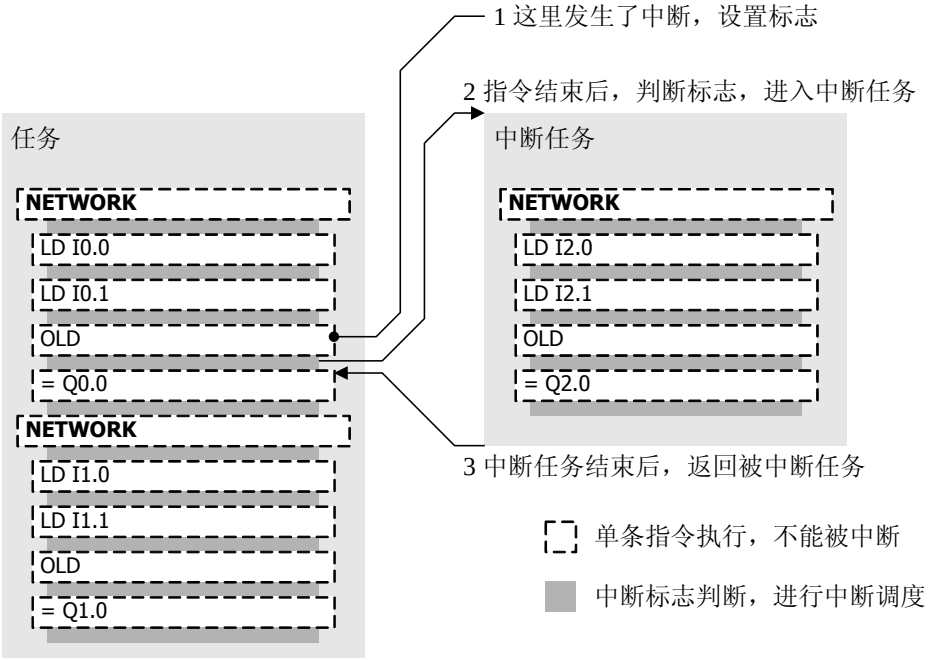
# 概述

除了 PLC 系统使用的中断服务，GUTTA 平台向 PLC 程序的编写者开放了一部分中断接口用于配置和编程。中断的发生使 PLC 放弃当前正在执行的任务转而去执行中断对应的任务。主循环扫描是 PLC 的一般性任务，它在 PLC 上电启动后重复执行且优先级最低，因此它可以被其他任何中断事件打断。不同的中断有不同的优先级。多个中断同时发生时，先运行中断优先级高的任务，然后运行中断优先级低的任务。GUTTA 系统中中断的使用主要包括中断事件与中断函数的连接，中断函数的程序编写这两部分内容。

纯粹的单片机控制器开发一般说来有下面几种形式：

1. **循环模式**：所有的工作都在主循环中完成。中断不被使能。这种模式最简单。
2. **中断模式**：所有的工作都在中断中完成。一但进入主循环就开始休眠。这种模式通常在低功耗设计中采用。
3. **前后台模式**：这种模式是前面两种模式的结合，一方面主循环程序在不停的工作，用于处理一般性逻辑或者运算。另一方面中断也是允许的，中断服务程序用于处理对响应要求较高的事件。
4. **合作调度模式**：在这种模式下，任务是程序的组织单元。中断程序里不执行任务，而是设置标志。中断结束后返回被中断的任务。该任务结束后调度系统根据前面中断设置的标志决定是否运行该中断对应的任务。需要注意的是，调度系统只有在某个任务结束后才会工作。因此任何任务的运行时间都不应该设计得太长，否则会导致其他任务被阻塞。
5. **抢占调度模式**：在抢占式调度模式下，中断不仅可以设置该中断对应任务的标志，如果需要的话，还可以打断当前任务。当前任务的现场由调度系统保存以等待其他任务运行结束后恢复。抢占式调度模式一般需要使用支持抢占的实时操作系统。

GUTTA PLC 系统本质上也是一个单片机程序。为了保证系统的最简化，GUTTA 采用了合作调度模式。在 PLC 正常运行的情况下，主循环重复调用 PLC 程序中的 MAIN（INT0）函数。从 PLC 的使用者看来，MAIN（INT0）就是 PLC 的主循环。其余对 PLC 用户开放的中断只设置中断标志。与普通合作调度模式相比，最大的优点是：PLC 程序是一条指令一条指令执行的。不论是解释型 PLC，还是编译型 PLC，也不论是在主循环程序中，还是在中断程序中，在每条 PLC 指令执行完后，系统都会对中断发生标志进行检查并决定是否中断当前任务。这样即使每个 PLC 函数程序很复杂耗时很长，中断的响应时间也最多不过是一条 PLC 指令的时间。




# 应用

## 指令介绍

在 GUTTA 系统中，中断的使用主要包括中断事件与中断函数的连接，中断函数的程序编写这两部分内容。若需要连接中断事件与函数，就需要用到 ATCH 指令。ATCH 指令有两个操作数：中断函数号 INT、中断事件号 EVENT。中断函数号 INT 应该是一个无符号单整数：数值 1 就表示中断函数 INT1，数值 2 就表示中断函数 INT2，依此类推。中断事件号 EVENT 也是一个无符号单整数。数值的含义由 PLC 的实现定义。

EVENT 含义

| EVENT | NAME | |
|---|---|---|
| 0 | INT_EVENT_INPUTP0 | 外部中断 0 |
| 1 | INT_EVENT_INPUTP1 | 外部中断 1 |
| 2 | INT_EVENT_TIMER0 | 时间中断 0 |
| 3 | INT_EVENT_TIMER1 | 时间中断 1 |
| 4 | INT_EVENT_TIMER_SECOND | 秒中断 |
| 5 | INT_EVENT_INPUTP2 | 外部中断 2 |
| 6 | INT_EVENT_INPUTP3 | 外部中断 3 |

EVENT 和实际硬件中断的对应关系

| EVENT | NAME | CPU-EC20 (AVR) | CPU-EC20 (8051) | CPU-EC20 (Cortex-M3) |
|---|---|---|---|---|
| 0 | INT_EVENT_INPUTP0 | INT7 | EXT1 | EXTI1_IRQHandler |
| 1 | INT_EVENT_INPUTP1 | × | × | × |

| | | | | |
|---|---|---|---|---|
| 2 | INT_EVENT_TIMER0 | TIMER0_OVF | TIMER1_OVF | SysTickHandler |
| 3 | INT_EVENT_TIMER1 | TIMER0_OVF | TIMER1_OVF | SysTickHandler |
| 4 | INT_EVENT_TIMER_SECOND | INT5 (PCF8563T) | EXT0 (PCF8563T) | RTC_IRQHandler |
| 5 | INT_EVENT_INPUTP2 | × | × | × |
| 6 | INT_EVENT_INPUTP3 | × | × | × |

EVENT 和实际硬件中断的对应关系（PLC 核）

| EVENT | NAME | EC30-EK51 | EC30-EKSTM32 |
|---|---|---|---|
| 0 | INT_EVENT_INPUTP0 | EXT0 | EXTI*_IRQHandler |
| 1 | INT_EVENT_INPUTP1 | EXT1 | EXTI*_IRQHandler |
| 2 | INT_EVENT_TIMER0 | TIMER1_OVF | SysTickHandler |
| 3 | INT_EVENT_TIMER1 | TIMER1_OVF | SysTickHandler |
| 4 | INT_EVENT_TIMER_SECOND | × | RTC_IRQHandler |
| 5 | INT_EVENT_INPUTP2 | × | EXTI*_IRQHandler |
| 6 | INT_EVENT_INPUTP3 | × | EXTI*_IRQHandler |

例如 ATCH 1,2 表示中断函数 INT1 连接时间中断 0。这也就意味着若时间中断 0 到达，PLC 系统就自动调用中断函数 INT1。

有中断连接指令 ATCH，就有中断分离指令 DTCH。和指令 ATCH 相反，指令 DTCH 取消中断事件与函数的连接。这个指令的操作数只有一个：EVENT。操作数 EVENT 的含义和 ATCH 指令中操作数 EVENT 含义一致。执行 DTCH 指令后，EVENT 中断事件相关的连接关系被取消。一个 EVENT 只能挂接一个中断函数 INT，一个中断函数 INT 可以同时服务于多个 EVENT。

有的时候，不希望某些程序被中断，就可以使用中断禁止指令 DISI。指令 DISI 全局性的禁止所有中断。等这段程序结束后，再运行中断允许指令 ENI。指令 ENI 全局性的允许所有中断。在中断禁止的这段时间，若发生了中断，对应的中断标志依然被置位但是不立即执行对应的中断函数，而是延迟到中断允许后执行。

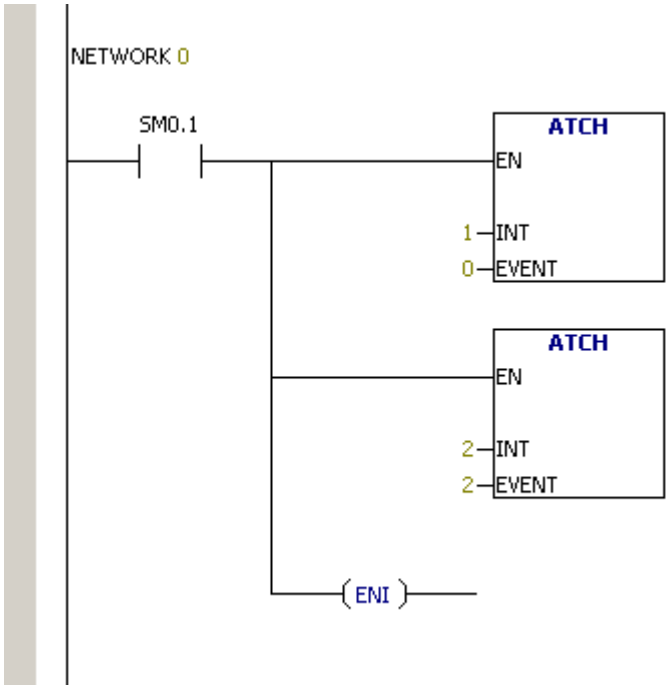
PLC 系统在复位后进入主循环扫描时，所有中断是被禁止的。

## 最简单的例子

http://www.plcol.com/technologies/anindex/an2001/Sample1.vcw

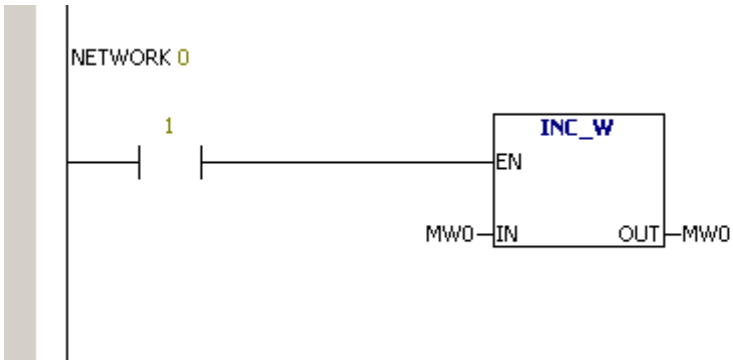
这个程序创建了 3 个中断函数：


第 1 个函数是主循环 MAIN（INT0）：



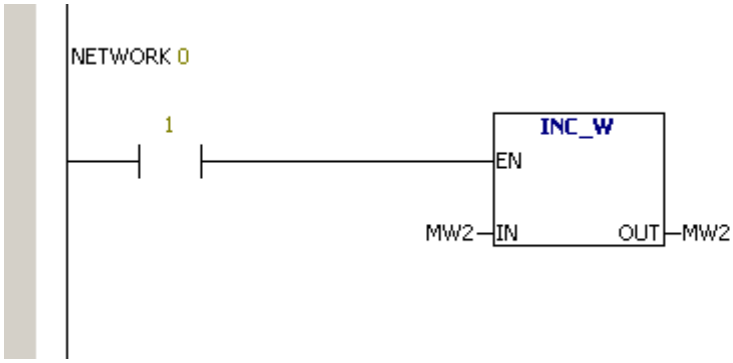

SM0.1 是 PLC 热启动标志位。PLC 每次启动后，SM0.1 在第一个扫描周期保持为 ON。第一个扫描周期结束后被系统设置为 OFF。MAIN 只有一个梯级，具体操作是：在系统热启动后，将 I0.1 上升沿事件和中断函数 INT_1（INT1）进行连接；将时间中断 0 事件和中断函数 INT_2（INT2）进行连接；同时允许中断。由于 SM0.1 只在第一个扫描周期保持为 ON，主循环扫描设置好中断后以后，就不进行任何操作了。

第 2 个函数是中断 INT_1，这个中断和 I0.1 上升沿事件相关联：




这个函数很简单，每执行一次，将 MW0 自加 1。其实也就是用 MW0 来记录 INT_1（INT1）的调用次数。

第 3 个函数是中断 INT_2，这个中断和时间中断 0 事件相关联：



这个函数很简单，每执行一次，将 MW2 自加 1。其实也就是用 MW2 来记录 INT_2（INT2）的调用次数。

同时，在系统块编辑窗口中设定时间中断 0 的时间间隔：


这里时间间隔设置为 50ms。也就是说每过 50ms 产生一个时间中断 0 事件。

将本程序下载到实验板或者是 PLC 模拟器 GUTTA Simulator 中，进入监控模式，观察变量 MW0 和 MW2 的变化。通过实验我们可以发现，每当按下 I0.1 时，MW0 发生一次自加。MW2 则根据时间连续的自加，由于中断事件设置为 50ms，意味着 MW2 每秒自加 20 次（1000ms / 50ms = 20）。

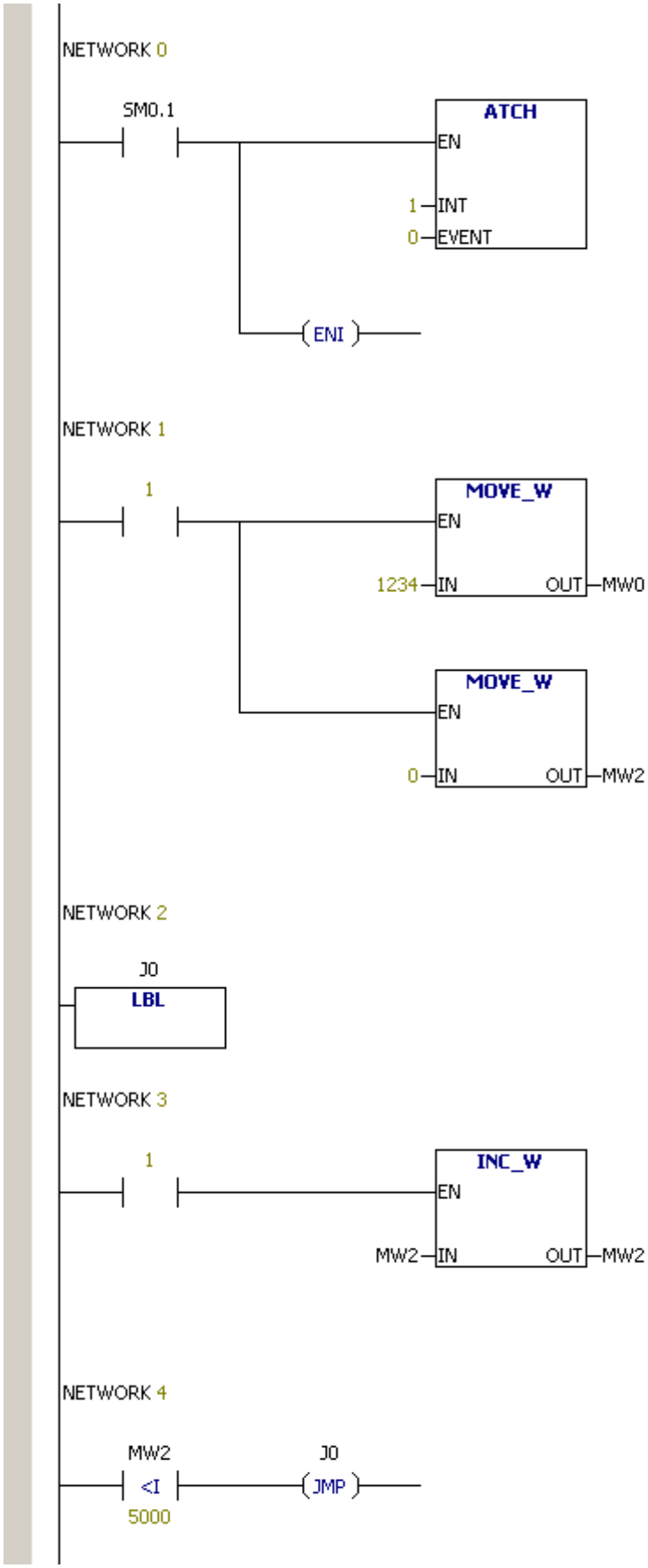
## 中断打断主循环的证明

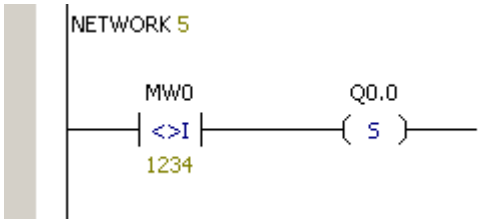
下面的这个程序是中断打断主循环的证明：

http://www.plcol.com/technologies/anindex/an2001/Sample2.vcw

这个程序创建了 2 个中断函数：


第 1 个函数是主循环 MAIN（INT0）：

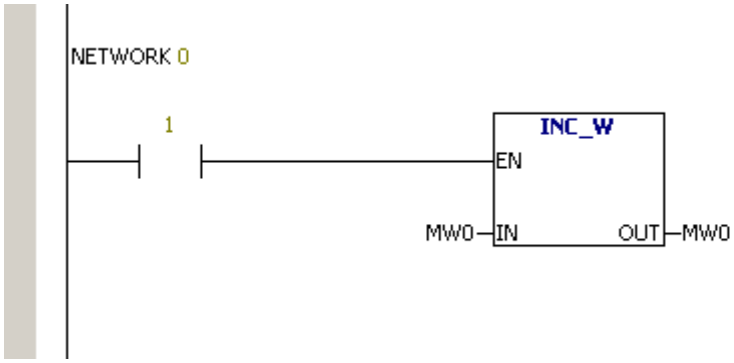

NETWORK 0
SM0.1
ATCH
EN
1
INT
0
EVENT
ENI
NETWORK 1
1
MOVE_W
EN
1234
IN
OUT
MW0
MOVE_W
EN
0
IN
OUT
MW2
NETWORK 2
J0
LBL
NETWORK 3
1
INC_W
EN
MW2
IN
OUT
MW2
NETWORK 4
MW2
<I
5000
J0
JMP




这个函数说明如下：

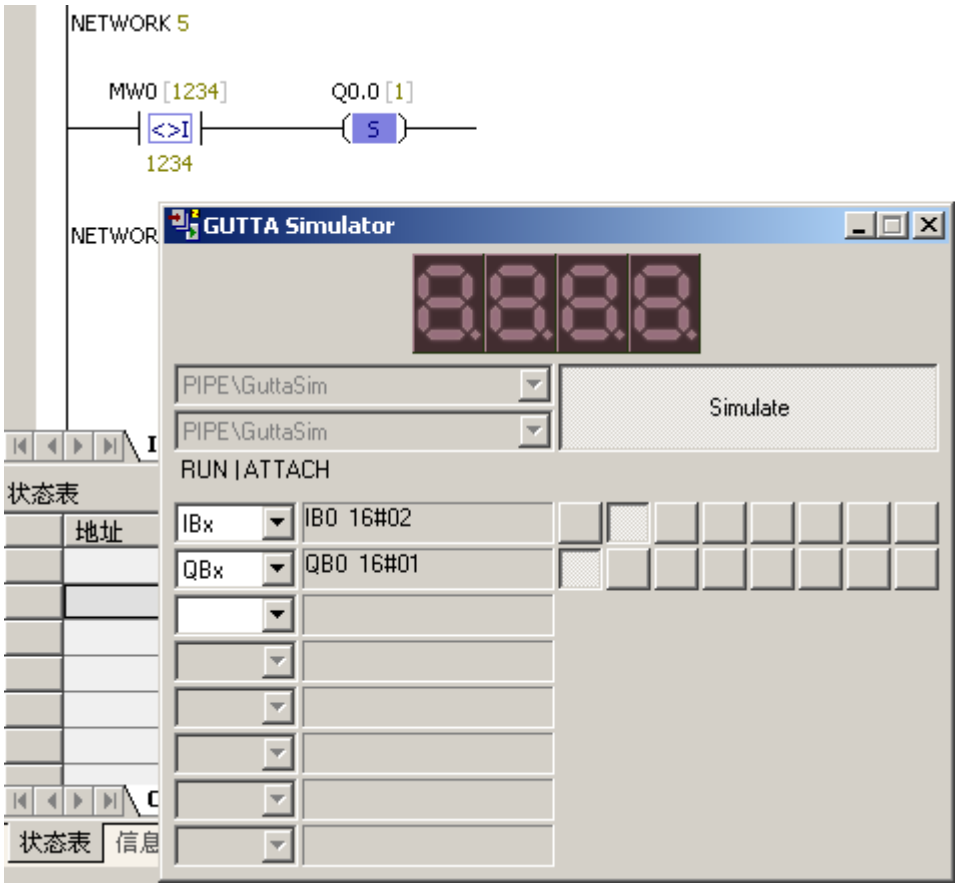
- NETWORK 0：将 I0.1 上升沿事件和中断函数 INT_1（INT1）进行连接；同时允许中断。
- NETWORK 1：初始化变量，将 MW0 设置为 1234，将 MW2 设置为 0。
- NETWORK 2、3、4：其实就是一段延时程序。MW2 不断的自加，然后和 5000 进行比较，若 MW2 小于 5000，转跳到 J0 继续自加，否则结束循环。（后面使用电脑的模拟器模拟，故循环次数比较大。若使用实验板，将此循环次数设为 100 即可。）
- NETWORK 5：判断 MW 是否依然等于 1234，如果不等于，则将 Q0.0 置位。

第 2 个函数是中断 INT_1（INT1），这个中断和 I0.1 上升沿事件相关联：




这个函数很简单，每执行一次，将 MW0 自加 1。

将本程序下载到实验版或者是 PLC 模拟器 GUTTA Simulator 中，进入监控模式，观察变量的值。




如上图，不断的点击输入 I0.1。直到 Q0.0 被置位。

观察主循环函数 MAIN（INT0）。MW0 在 NETWROK 1 被设置为 1234，延时一段时间后在 NETWORK 5 中判断 MW0 的值。由于 MW0 的值发生了改变，最终 Q0.0 被置位。这说明主循环 MAIN（INT0）的执行被中断。INT_1（INT1）的 INCW MW0 指令是造成 MW0 的值发生改变的唯一理由。