HOYAN ETH-A2 · STM32H743 · 固件 V2.77

HRL 语言规格

HOYAN Rule Language — 规则引擎语言参考

160 rules | 3 cond / 3 act | scan 20 ms | $VAR.0–63 | rule_def_t 247 B

目次

00 适用范围与版本
01 文件结构与容量
02 RULE 声明
03 IF 条件
04 THEN 动作
05 ON 触发模式
06 COOLDOWN 与属性
07 Tag 命名空间
08 扫描周期与执行模型
09 功能块对照表
10 完整范例:舱底水高位报警盘
11 诊断工具与开机行为
12 陷阱总表
13 快速参考卡

00 适用范围与版本

本手册说明 HOYAN ETH-A2 通信网关内置规则引擎所使用的 HOYAN Rule Language(HRL),涵盖语法、执行模型与运作限制,适用对象为负责现场逻辑组态的控制工程师与系统集成商。

本文件内容对应固件版本 V2.77。所述行为、容量上限与诊断接口均以该版本为准,后续版本可能变更,恕不另行通知。当本文件与设备实际行为不一致时,以设备实际行为为准。

HRL 是声明式的规则语言,**不是** IEC 61131-3。没有功能块、没有梯形图、没有 TON / TOF / R_TRIG / SR / CTU。全部逻辑由三个原语组成:

- `FOR <t>` — 条件必须持续成立
- `ON RISE` / `ON FALL` — 边缘触发
- `$VAR.0` ~ `$VAR.63` — 64 个 int32 中继变量,同一扫描周期内即时可见

解析器是逐行的,没有 AST。**不认得的行会被静默忽略,不报错** —— 这代表打错关键字的规则会安静地少掉一半语意。导入后务必逐条核对规则数与内容。

> **注意**
> 本产品的规则引擎为通用控制逻辑,非安全等级(SIL / PL)装置。涉及人员安全、防止设备损坏或环境保护的功能,必须由独立的硬件安全回路实现,不得仅依赖规则引擎。

01 文件结构与容量

```
# comment lines start with #
RULE <name up to 23 chars> :
  IF   <tag> <op> <literal>  [FOR <duration>]
  AND  <tag> <op> <literal>
  AND  <tag> <op> <literal>
  THEN <action>
  AND  <action>
  AND  <action>
  ON   LEVEL | RISE | FALL | CHANGE | BOOT | TRIGGER | OFFDELAY
  COOLDOWN <duration>
  GROUP <n>    PRIORITY <n>    DISABLED
```

缩进无意义。关键字不分大小写。`RULE` 开启新规则,不需 `END`。**一行一个指令** —— 解析器读完该行的第一个指令后直接跳到下一行。

`AND` / `OR` 的归属由「是否已看到 `THEN`」决定: `THEN` 之前填条件槽,之后填动作槽。这是解析器唯一的状态切换。

项目	上限	备注
规则数	160	
规则名称	23 字节	建议使用 ASCII
Tag 名称	23 字节	
条件 / 规则	3	左结合,无括号
动作 / 规则	3	超出的子句静默丢弃
群组	32	可整组启停

项目	上限	备注
用户变量	`$VAR.0` – `$VAR.63`	32 位有符号整数,断电保留
活动报警	16	
触发历史	64 条	规则页的「历史」按钮
`FOR` / `COOLDOWN` 时间	65535 ms	即 65.5 秒
每扫描周期输出点数	96	超出计入 `$SYS.WQ_FAIL`
单次导入规则数	64	更多请分批以 append 导入
单次导入请求大小	4096 字节	含标头;HRL 文字约 3.5 KB

> **陷阱**
> 超过 3 个条件或 3 个动作的子句会被
> **静默丢弃**
> 。写了第 4 个 `AND` 不会有任何错误消息,那行就是不存在。要更多条件请拆成多条规则,用 `$VAR` 串接。

> **陷阱 — 导入体积上限**
> 单次导入的整个请求(含标头)不得超过
> **4096 字节**
> 。超出的部分会被
> **静默丢弃**
> ,设备不返回任何错误。扣除标头后,HRL 文字实际可用约
> **3.5 KB**
> 。
>
> 大型项目请去除注释,并依规则边界切成多段:第一段用 `mode: "replace"` ,其余用 `mode: "append"` 。append 会依序 upsert 在既有规则之后,执行顺序完整保留。

02 RULE 声明

名称取到 `:` 或行尾为止,尾端空白去除,截断到 23 字符。冒号可省略。

名称是 `TRIGGER` 动作与 `/api/rules/fire` 的查找键(名称完全比对)。**名称重复时只会找到第一个。**

新规则默认值: `enable=1` 、 `priority=50` 、 `expr*=NONE` 、 `action3=NOOP` 。

> 规则名称建议用 ASCII。名称字段是 24 字节,UTF-8 中文一个字占 3 字节,只能放 7 个中文字,而且截断发生在字节边界,可能切出无效的 UTF-8 序列。

03 IF 条件

一般条件

```
IF DI16_#2.DI0 == 1                 # digital input closed
IF DI16_#2.DI0 == ON                # identical, ON parses to 1
IF AI6_#3.AI0 > 4000                # raw integer, no scaling
IF $VAR.12 != 0
IF $SYS.ETH_LINK == 0
IF @Pwr-1.va >= 220
```

运算符: `==` `!=` `>` `>=` `<` `<=` `ANY` 。未知运算符 fallback 为 `==` 。值: `ON` / `TRUE` → 1, `OFF` / `FALSE` → 0,其余以十进制整数解读。**阈值只能是常数,不能是另一个 tag。**

时间调度条件

```
IF EVERY 30s                        # -> TIMER mode
IF DAILY AT 22:30                   # -> SCHEDULE, every day
IF WEEKLY MON,TUE,FRI AT 08:30      # -> SCHEDULE, selected days
```

`EVERY` 解析度为整秒,小于 1s 会被强制成 1s 并打印 UART 警告,上限 65535s。星期字符串 `SUN MON TUE WED THU FRI SAT` ,逗号分隔无空格。 `DAILY` / `WEEKLY` 由 RTC 比对时分,同一分钟只触发一次。

AND / OR 追加条件

```
RULE pump_permit :
  IF  DI16_#2.DI0 == 1
  AND AI6_#3.AI0 > 500
  AND $VAR.5 == 0
  THEN DO16_#1.DO0 = 1
  ON LEVEL
```

第 1 个追加子句填 cond2,第 2 个填 cond3,之后丢弃。求值为**左结合、无括号**: `((c1 ⊕₁ c2) ⊕₂ c3)` 。

> **陷阱**
>
> `AND` 没有优先于 `OR` 。 `A OR B AND C` 在 HRL 中等于 `(A OR B) AND C` ,不是布尔代数的惯例。需要混合运算就用 `$VAR` 分阶段。

```
# Written intent:  A OR (B AND C)
# HRL evaluates:  (A OR B) AND C
  IF  DI16_#2.DI0 == 1
  OR  DI16_#2.DI1 == 1
  AND AI6_#3.AI0 > 500

# Correct approach - stage it through $VAR
RULE or_stage :
  IF  DI16_#2.DI0 == 1
  OR  DI16_#2.DI1 == 1
  THEN $VAR.30 = 1
  ON LEVEL

RULE or_stage_r :
  IF  DI16_#2.DI0 == 0
  AND DI16_#2.DI1 == 0
  THEN $VAR.30 = 0
  ON LEVEL

RULE combined :
  IF  $VAR.30 == 1
  AND AI6_#3.AI0 > 500
  THEN DO16_#1.DO0 = 1
  ON LEVEL
```

任一条件的 tag 读取失败 → 整条规则本次扫描视为无效,引擎记住的「上一次条件结果」与持续计时一并归零,并在触发历史留下 `e:cond`。这代表模块离线时规则的边缘记忆会被抹掉,重新上线会产生一次上升沿。

FOR — 持续时间 (dwell)

```
FOR 250ms        ->    250 ms
FOR 5s           ->   5000 ms
FOR 0.5s         ->    500 ms
FOR 1.25s        ->   1250 ms
FOR 30           ->  30000 ms   (bare number = seconds)
FOR 1m           ->  60000 ms
FOR 2m           ->  65535 ms   CLAMPED + UART warning
```

未达持续时间期间,引擎会把「上一次条件结果」视为不成立,所以 `FOR` + `ON RISE` 会在时间达成的那一次扫描正确产生上升沿。这个特性也是自激振荡闪烁源能成立的原因(见 §09)。

> **陷阱**
>
> `FOR` 套用在
> **三个条件的合并结果**
> 上,不是单一条件。多条件规则中任一条件变动都会重启 dwell 计时。去抖动逻辑必须隔离成单条件规则,输出到 `$VAR`。
>
> `FOR` 的时间
> **最大 65535 MS(65.5 秒)**
> 。`FOR 2m` 会被夹到 65535 ms。

04 THEN 动作

```
THEN DO16_#1.DO0 = 1
THEN DO16_#1.DO0 = ON               # identical
THEN AO4_#7.AO1  = 2048
THEN $VAR.3      = 0
THEN DO16_#1.DO0 = TOGGLE
THEN DO16_#1.DO0 = PULSE 500ms      # ON now, auto OFF after 500 ms
THEN ALARM "bilge high" HIGH
THEN MQTT plant/line1/state
THEN TRIGGER other_rule IMMEDIATE
THEN NTP_SYNC
THEN LOG
```

语法	行为
`<tag> = <value>`	写入常数
`<tag> = <operand> <op> <literal>`	算术／位运算式
`<tag> = TOGGLE`	读现值后写反相
`<tag> = PULSE [<dur>]`	立刻写 1,经指定时间(默认 3000 ms)后自动写 0
`ALARM "<name>" <sev>`	产生报警, `INFO` `WARN` `HIGH` `CRITICAL`
`MQTT <topic>`	发布 MQTT
`TRIGGER <rule> [IMMEDIATE]`	触发另一条规则
`NTP_SYNC`	触发网络校时
`LOG` / `NOOP`	只写日志

运算式

运算符 `+ - * / % & | ^`。 `/` 与 `%` 的 literal 为 0 时分别返回 operand 与 0,不会除零崩溃。

```
# Slot 1 (THEN) - operand may be ANY tag
THEN $VAR.10 = AI6_#3.AI0 / 10       # scale down
THEN $VAR.11 = $VAR.11 + 1           # counter
THEN AO4_#7.AO0 = AI6_#3.AI1 * 2     # analog follower
THEN $VAR.12 = DI16_#2.DI0 + 0       # COPY idiom
THEN $VAR.13 = $VAR.13 & 254         # clear bit 0
THEN $VAR.14 = $SYS.RTC_MIN % 15     # quarter-hour phase

# Slots 2 and 3 (AND) - operand must be $VAR.n
  AND $VAR.20 = $VAR.21 + 100        # OK
  AND $VAR.20 = AI6_#3.AI0 + 100     # NOT OK - operand ignored, writes 100
```

槽	operand 可以是	存储字段
动作 1 (`THEN`)	任意 tag — DI/DO/AI/AO/ `$VAR` / `$SYS` / `@ext`	完整 tag 名称
动作 2 / 3 (`AND`)	只能是 `$VAR.n`	仅 `$VAR` 编号

> **陷阱**
> 引擎内部支持直接复制,但 HRL **没有对应的书写方式**。`THEN DO = $VAR.5` 会被当成常数解读,结果是 **0**,而且完全不会报错。

```
THEN DO16_#1.DO0 = $VAR.5           # WRONG - writes 0, silently
THEN DO16_#1.DO0 = $VAR.5 + 0       # correct
THEN DO16_#1.DO0 = $VAR.5 * 1       # also correct
```

05 ON 触发模式

未指定时默认 `LEVEL`;未知字符串也 fallback 为 `LEVEL`。`EVERY` / `DAILY` / `WEEKLY` 由 `IF` 那行隐含设定,不需再写 `ON`。

模式	语意
`LEVEL`	条件为真 → **每个 tick 都执行动作**
`RISE`	条件 false→true 的那一 tick 执行一次
`FALL`	条件 true→false 的那一 tick 执行一次
`CHANGE`	条件任何变化执行一次
`BOOT`	开机时执行一次,正常 tick 中完全跳过
`TRIGGER`	只被别的规则唤醒;若有 `IF` 则当作闸门
`OFFDELAY` / `TOF`	延迟关,对标 IEC 61131-3 TOF

OFFDELAY (TOF)

```
RULE fan_runon :
  IF DI16_#2.DI5 == 1
  THEN DO16_#1.DO3 = 1
  ON OFFDELAY FOR 30s
```

条件成立 → 立刻写 `action_value`(为 0 则写 1),取消待关倒数。条件转 false → 启动倒数,倒数到才写 0。`FOR 0` 退化为一般 LEVEL。

> **陷阱**
> `ON OFFDELAY FOR ...` 的 `FOR` 是 **关闭延迟**,占用了该规则唯一的时间字段。同一条规则 **无法再用 `IF ... FOR` 做输入去抖动**。要「先去抖动再延迟关」得串两条规则。

规则连锁

```
RULE detect :
  IF AI6_#3.AI0 > 3800
  THEN TRIGGER shutdown IMMEDIATE
  ON RISE

RULE shutdown :
  THEN DO16_#1.DO0 = 0
  AND  DO16_#1.DO1 = 0
  AND  ALARM "overpressure trip" CRITICAL
  ON TRIGGER
```

`TRIGGER` = 下一 tick 执行(事件语意); `TRIGGER … IMMEDIATE` = 本 tick 执行(调用语意)。连锁保护:同一规则每 tick 最多 fire 1 次、每 tick 全局上限 64 次、IMMEDIATE 递归深度 8、工作串列 16。

开机规则

```
RULE init_state :
  IF $SYS.UPTIME >= 0
  THEN $VAR.0 = 0
  AND  $VAR.8 = 0
  AND  DO16_#1.DO0 = 0
  ON BOOT
```

06 COOLDOWN 与属性

`COOLDOWN` 为最小重触发间隔,最大 65535 ms。第一次触发不受限。被 cooldown 挡下时规则**不执行动作**,**但「上一次条件结果」仍会更新**,所以 RISE 边缘会被吃掉。

> `THEN <tag> = PULSE <dur>` 会把 `<dur>` 与独立的 `COOLDOWN` 行共用同一个时间字段。同一条规则不要同时使用两者。

属性	语法	说明
`GROUP`	`GROUP 3` / `GROUP Pumps`	1~32;字符串会 hash(`h = h*31 + c`)% 32 + 1。可整组启停。
`PRIORITY`	`PRIORITY 10`	1=最高,99=最低,默认 50。**不影响执行顺序**。
`DISABLED`	`DISABLED`	`enable = 0` ,规则保留但不执行。

> **陷阱 — PRIORITY 不影响执行顺序**
>
> `PRIORITY` 会被存档、也会被导出,但引擎在每个扫描周期是由上而下依序评估每一条规则,完全不参考这个字段。
>
> **执行顺序 = 规则在数组中的索引顺序 = HRL 文件中的书写顺序。**
>
> 要控制顺序就控制书写顺序。

07 Tag 命名空间

前缀	命名空间	范例	读	写
(无)	本地 IO 模块	`DI16_#2.DI0`	✔	✔

前缀	命名空间	范例	读	写
`$SYS.`	系统变量	`$SYS.UPTIME`	✔	只读
`$VAR.`	用户变量 0–63	`$VAR.12`	✔	✔
`@`	外部设备	`@Pwr-1.va`	✔	✔

```
DI16_#2.DI0        slave 2, coil point 0,  digital input
DO16_#1.DO15       slave 1, coil point 15, digital output
DO6TR_#5.DO3       slave 5, coil point 3,  relay output
AI6_#3.AI0         slave 3, register 0,    analog input
AO4_#7.AO1         slave 7, register 1,    analog output
$VAR.42            user variable, 32-bit signed, retained
$SYS.RTC_HOUR      system variable, read-only
@Pwr-1.va          external device via ext_device
```

本地 IO tag 的实际解析规则

本地 tag 的解析比表面看起来宽松很多:

- 找第一个 `#`,其后的十进制数字 = slave id
- 扫描整个字符串,只要出现子字符串 `AI` 或 `AO` → register;否则 coil
- 取字符串尾端的连续数字 = point index
- slave id 必须对应到一个已上线的模块,否则不算本地 tag

也就是说模块型号字符串本身不参与比对,`DO16_#1.DO7` 与 `XX_#1.7` 解析结果相同。仍建议照规范写,因为 Web UI 与 Modbus 映射表用的是完整名称。Coil point 限 0–15。

模块	ID	类别
`DI16`	0x471C	16 路数字输入
`DO16`	0x471D	16 路数字输出
`AI6`	0x471F	6 路模拟输入
`DO6TR`	0x4720	6 路继电器输出

$SYS 全清单

名称	内容
`UPTIME`	开机后秒数
`RTC_YEAR` … `RTC_SEC`	RTC 年月日时分秒
`RTC_WEEK`	0=周日 … 6=周六
`CPU_LOAD`	CPU 负载百分比
`RULE_ENABLED`	规则引擎总开关 0/1
`RULE_COUNT`	启用中的规则数
`MQTT_CONNECTED`	MQTT 连线 0/1

名称	内容
`ETH_LINK`	以太网连线 0/1
`SLAVE_COUNT`	在线 IO 模块数
`WQ_FAIL`	写入队列满而丢弃的累计数

名称**区分大小写**,必须全大写。 `$VAR` 是 32 位有符号整数,存于内部非易失性存储,重启保留 —— 若不希望保留请加一条 `ON BOOT` 归零规则。

08 扫描周期与执行模型

引擎以固定周期反复执行下列三个阶段。默认周期 20 ms(50 Hz),可调 5–100 ms。

```
1  INPUT      every local input point is sampled into a frozen
              snapshot; the output image starts as a copy of it

2  EVALUATE   every rule is evaluated in the order it appears in
              the file, then any rule woken by TRIGGER IMMEDIATE

3  UPDATE     only the points actually written during this scan
              are sent out to the field bus
```

读写语意的关键差异

Tag 类型	本 tick 读到的值	写入何时生效
本地 IO (DI/DO/AI/AO)	tick **开始时的冻结快照**	tick 结束刷出;同 tick 内后续规则读不到
`$VAR` / `$SYS` / `@ext`	**即时值**	**立即生效**,同 tick 内后续规则读得到

这是设计意图:本地 IO 冻结可根除同 tick 连锁抖动; `$VAR` 即时是为了让状态机能在一个扫描周期内完成推进。

> **陷阱**
>
> **不要用 DO 点当中继位。**
>
> 规则读不到自己或前面规则在同一 tick 写进去的 DO 值,状态机会延迟一整个扫描周期,而且行为会随规则顺序改变。所有中继逻辑走 `$VAR` 。

同一个 DO 被多条规则写

后执行者覆盖前执行者(都写进同一份输出映像),输出更新阶段只送出最终值。这让「先写默认值、后面条件成立再覆盖」的分层写法成为 HRL 最实用的技巧 —— 它绕开了「只有 3 条件、无括号、AND 不优先于 OR」的全部限制。

```
# Layer 1 - default OFF
RULE out_default :
  IF $VAR.21 == 0
  THEN DO16_#1.DO1 = 0
  ON LEVEL

# Layer 2 - steady ON
RULE out_steady :
  IF  $VAR.21 == 1
  AND $VAR.1 == 0
  THEN DO16_#1.DO1 = 1
  ON LEVEL

# Layer 3 - flashing overrides steady
RULE out_flash :
  IF $VAR.1 == 1
  THEN DO16_#1.DO1 = $VAR.0 + 0
  ON LEVEL

# Layer 4 - lamp test overrides everything (must be LAST in file)
RULE out_lamptest :
  IF DI16_#2.DI1 == 1
  THEN DO16_#1.DO1 = 1
  ON LEVEL
```

写入队列

单一扫描周期若写超过 96 个实体点,超出的会被丢弃,累计数暴露在 `$SYS.WQ_FAIL` ,可用规则自我监控:

```
RULE wq_watch :
  IF $SYS.WQ_FAIL > 0
  THEN ALARM "write queue full" HIGH
  AND  MQTT diag/wqfail
  ON RISE
  COOLDOWN 60s
```

09 功能块对照表

以下每个模式都只用 `FOR` 、 `ON RISE/FALL` 、 `$VAR` 三样东西,在 V2.77 语意下验证过。

IEC 61131-3	HRL 对应
TON	`IF ⋯ FOR t` + `ON RISE` ,输出到 `$VAR`
TOF	原生 `ON OFFDELAY FOR t`
R_TRIG / F_TRIG	原生 `ON RISE` / `ON FALL`
SR / RS	两条 `ON RISE` 规则写同一个 `$VAR`
CTU / CTD	`$VAR = $VAR + 1` 于 `ON RISE`
MOVE	`dst = src + 0` (动作槽 1)
BLINK	两条互锁的 `FOR` + `ON RISE` 规则
SEL / MUX	分层覆盖写法(§08)

IEC 61131-3	HRL 对应
HYSTERESIS	仅 Web UI / JSON API 可设,HRL 无语法

TON — 接通延迟

```
RULE ton_ls1 :
  IF DI16_#2.DI2 == 1 FOR 5s
  THEN $VAR.21 = 1
  ON RISE

RULE ton_ls1_r :
  IF DI16_#2.DI2 == 0
  THEN $VAR.21 = 0
  ON RISE
```

务必隔离成单条件规则。`FOR` 挂在多条件规则上会被其他条件的变动重启计时。上限 65.5 秒。

SR — 自保持

```
# Reset-dominant: reset rule is written AFTER set rule
RULE trip_set :
  IF AI6_#3.AI0 > 3800
  THEN $VAR.9 = 1
  ON RISE

RULE trip_reset :
  IF DI16_#2.DI0 == 1
  THEN $VAR.9 = 0
  ON RISE
```

置位优先或复位优先由书写顺序决定 —— 写在后面的覆盖前面的。上例是复位优先。

闪烁源(与扫描周期无关)

```
RULE blink_on :
  IF $VAR.0 == 0 FOR 500ms
  THEN $VAR.0 = 1
  ON RISE

RULE blink_off :
  IF $VAR.0 == 1 FOR 500ms
  THEN $VAR.0 = 0
  ON RISE

# consume it
RULE lamp_flash :
  IF $VAR.1 == 1
  THEN DO16_#1.DO1 = $VAR.0 + 0
  ON LEVEL
```

自激振荡成立的原因:持续时间未达时引擎把「上一次条件结果」视为不成立,达成当下必产生上升沿。500ms ON / 500ms OFF = 1 Hz。比 `IF EVERY 1s THEN $VAR.0 = TOGGLE` (受 `EVERY` 最小 1 秒限制,只能到 0.5 Hz)更灵活。

计数器

```
RULE cnt_up :
  IF DI16_#2.DI6 == 1
  THEN $VAR.30 = $VAR.30 + 1
  ON RISE

RULE cnt_done :
  IF $VAR.30 >= 100
  THEN DO16_#1.DO5 = 1
  AND  $VAR.30 = 0
  ON RISE

RULE cnt_clear :
  IF $SYS.UPTIME >= 0
  THEN $VAR.30 = 0
  ON BOOT
```

启停自保持 + 跳闸

```
RULE motor_start :
  IF  DI16_#2.DI0 == 1
  AND $VAR.40 == 0
  THEN $VAR.40 = 1
  ON RISE

RULE motor_stop :
  IF DI16_#2.DI1 == 1
  THEN $VAR.40 = 0
  ON RISE

RULE motor_trip :
  IF DI16_#2.DI2 == 1
  THEN $VAR.40 = 0
  ON LEVEL

RULE motor_out :
  IF $VAR.40 >= 0
  THEN DO16_#1.DO0 = $VAR.40 + 0
  ON LEVEL
```

`motor_trip` 用 `ON LEVEL` 而非 `ON RISE`:跳闸条件成立期间必须持续压住输出,不能只在边缘清一次。顺序上它写在 `motor_stop` 之后,所以跳闸优先于启动。

双泵轮替

```
RULE lead_swap :
  IF $VAR.50 == 0
  THEN $VAR.50 = 1
  ON RISE

RULE lead_swap_b :
  IF $VAR.50 == 1
  THEN $VAR.50 = 0
  ON RISE

RULE pump_a :
  IF  $VAR.41 == 1
  AND $VAR.50 == 0
  THEN DO16_#1.DO2 = 1
  ON LEVEL

RULE pump_b :
  IF  $VAR.41 == 1
  AND $VAR.50 == 1
  THEN DO16_#1.DO3 = 1
  ON LEVEL
```

通信监视

```
RULE bus_degraded :
  IF $SYS.SLAVE_COUNT < 4
  THEN ALARM "IO module offline" CRITICAL
  AND  DO16_#1.DO7 = 1
  ON RISE
  COOLDOWN 30s

RULE bus_ok :
  IF $SYS.SLAVE_COUNT >= 4
  THEN DO16_#1.DO7 = 0
  ON RISE
```

调度

```
RULE night_dim :
  IF DAILY AT 22:00
  THEN AO4_#7.AO0 = 400

RULE day_bright :
  IF DAILY AT 06:00
  THEN AO4_#7.AO0 = 2000
```

10 完整范例:舱底水高位报警盘

7 点船用舱底水报警盘,取代一台 20 点 PLC 加 7 颗 H3Y-4 延时继电器。行为:未确认时灯闪 1 Hz 加蜂鸣器;按确认钮蜂鸣器停、灯转常亮;液位复归灯灭;确认后若有新点成立蜂鸣器重响。共 54 条规则。

```
# ---- I/O map ------------------------------------------------
# DI16_#2.DI0     BS   buzzer stop / acknowledge
# DI16_#2.DI1     LT   lamp test pushbutton (wired direct to DI)
# DI16_#2.DI2~8   LS1..LS7 field level switches
# DO16_#1.DO0     BZ   buzzer      (DO16 PNP sourcing output, direct drive)
# DO16_#1.DO1~7   lamps R1..R7 (DO16 PNP sourcing output, no dimming)
# DO6TR_#5.DO0~5  CRT repeat 1..6  (dry contacts)
# DO6TR_#6.DO0    CRT repeat 7
# DO6TR_#6.DO1~2  local panel repeats (former T4 / T5 6-10 contacts)
#
# ---- $VAR map -----------------------------------------------
# $VAR.0          FLASH  1 Hz source
# $VAR.1~7        UNACK1..7
# $VAR.8          BUZ latch
# $VAR.21~27      DLY1..7  debounced alarm state (replaces H3Y-4)
```

每个报警点的核心两条规则

```
RULE p1_ton :
  IF DI16_#2.DI2 == 1 FOR 5s
  THEN $VAR.21 = 1
  AND  $VAR.1  = 1
  AND  $VAR.8  = 1
  ON RISE

RULE p1_rst :
  IF DI16_#2.DI2 == 0
  THEN $VAR.21 = 0
  AND  $VAR.1  = 0
  ON RISE
```

`p1_ton` 同时做三件事:去抖动后锁存 DLY、标记未确认、拉起蜂鸣器自保持。三个动作刚好填满一条规则的动作槽。`FOR 5s` 对应原机 H3Y-4 的 0–30 秒可调延时,防止船舶摇晃造成误报。

确认钮

```
RULE ack_a :
  IF DI16_#2.DI0 == 1
  THEN $VAR.8 = 0
  AND  $VAR.1 = 0
  AND  $VAR.2 = 0
  ON RISE

RULE ack_b :
  IF DI16_#2.DI0 == 1
  THEN $VAR.3 = 0
  AND  $VAR.4 = 0
  AND  $VAR.5 = 0
  ON RISE

RULE ack_c :
  IF DI16_#2.DI0 == 1
  THEN $VAR.6 = 0
  AND  $VAR.7 = 0
  ON RISE
```

一个上升沿要清 8 个标志,而一条规则只有 3 个动作槽,所以拆成 3 条同条件规则。三条各自记住自己的条件状态,同一个上升沿会同时触发,不需要 `TRIGGER` 串接。这三条必须写在所有 `p*_ton` 之前,否则同一 tick 内到来的新报警会被误消音。

灯输出

DO16 是 PNP 源型输出,指示灯与蜂鸣器直接挂在输出点上,不经继电器也不调光。原图的 DM:5K 调光器、调光模块与 RY1 中继在此设计中全部省略,灯测试按钮直接接入 DI。

灯的三态(灭／常亮／闪烁)加上灯测试覆盖,用 §08 的分层覆盖写法,见该节范例。四层依序写在文件中,灯测试那层必须是全档最后。

复示干接点

```
RULE p1_rep :
  IF $VAR.21 >= 0
  THEN DO6TR_#5.DO0 = $VAR.21 + 0
  ON LEVEL
```

> **陷阱**
> 复示接点必须追随 `DLY`,
> **不能追随灯输出**
> 。原图的复示是计时器的物理接点,与 PLC 的确认逻辑无关 —— 按了确认钮之后,机舱控制室那边仍须维持有报警。若误接灯输出,确认后的闪烁会让对方收到重复的假报警。

规则排列顺序

- `ON BOOT` 初始化
- 闪烁源 ×2
- 确认钮 ×3
- 去抖动 + 复归 ×14
- 灯输出三层 ×21
- 复示 ×9
- 蜂鸣器 ×1
- 灯测试 ×3(必须最后)

11 诊断工具与开机行为

触发历史

规则页左下的「历史」按钮开启,保留最近 64 条记录,字段为时间、规则、结果。

结果	意义
`skip`	条件不成立,未触发
`trig`	正常执行

结果	意义
`dry`	模拟模式,未实际写出
`e:cond`	条件的 tag 读取失败
`e:act`	动作写入失败
`e:chain`	连锁超过上限,或 `TRIGGER` 的规则名称找不到
`e:wq`	本次扫描有输出因队列已满而丢弃

调试模式

规则页的「调试」进入。把一条规则的评估过程拆成七个阶段逐一显示,每个阶段都看得到引擎实际读到的值与判定结果 —— 这是唯一能回答「为什么这条规则没动作」的工具。

阶段	检查什么
1 READ_TAG	条件的 tag 是否读得到,读到什么值
2 条件 1 评估	与阈值比较的结果
3 条件 2 合并	追加条件与 AND / OR 合并后的结果
4 持续检查	`FOR` 的时间是否已达成
5 边缘检测	依 `ON` 模式判断本次是否该触发
6 冷却检查	是否仍在 `COOLDOWN` 期间内
7 执行动作	实际写出的目标与值

范围可选全部规则、当前选中的规则、或单一群组;监控间隔 0.5 / 1 / 2 / 5 秒。另有「强制执行」直接跑一次动作,以及「条件注入」自动写入能让条件成立的值 —— 若来源是物理输入则无法注入,必须改变实际输入或改用虚拟模块。

监看与交叉参照

监看表可钉选任意 tag 观察实时值。交叉参照列出哪些规则**读取**、哪些规则**写入**该 tag —— 回答「这个值到底是谁在动」。异常筛选只列出 tag 读取失败或执行错误的规则;条件不成立不算异常。

「保持」冻结引擎,输出**维持现状**(不是停用归零),10 分钟无操作会自动恢复;「单步」放行一个扫描周期。「Force 强制置位」最多 16 点,优先于所有写入者,重启即清除,期间规则仍照常评估。

模拟模式

引擎设置里的开关。挡掉所有物理输出、MQTT 与报警,但**放行** `$VAR` **写入**,让状态机在模拟时仍能推进。注意 `$VAR` 的值是真的写进去了,退出模拟时不会自动恢复。

用规则自我监控

下列系统变量可直接写进规则,把诊断变成主动告警:

变量	监控什么
`$SYS.WQ_FAIL`	输出因队列已满被丢弃的累计次数

变量	监控什么
`$SYS.SLAVE_COUNT`	在线 IO 模块数,掉线即可告警
`$SYS.ETH_LINK`	以太网连线
`$SYS.MQTT_CONNECTED`	MQTT 连线
`$SYS.CPU_LOAD`	CPU 负载百分比

```
RULE wq_watch :
  IF $SYS.WQ_FAIL > 0
  THEN ALARM "write queue full" HIGH
  AND  MQTT diag/wqfail
  ON RISE
  COOLDOWN 60s
```

开机策略

在引擎设置的「启用时 BOOT 规则」下拉菜单设定:

选项	行为
BOOT 优先(跳过 Safety)	略过安全策略,直接跑 `ON BOOT` 规则(默认)
Safety 后 BOOT(先归零)	安全策略先把所有 DO 归零,等 1 秒,再跑 BOOT 规则
只用 Safety(不跑 BOOT)	不执行 BOOT 规则

`ON BOOT` 规则在正常扫描中完全跳过,只在开机、引擎总开关由关转开、以覆盖模式导入 HRL、或按下引擎设置里的「手动 fire BOOT」时执行。

12 陷阱总表

#	陷阱	对策
1	`PRIORITY` 不影响执行顺序	用书写顺序控制
2	本地 IO 读冻结快照,同 tick 读不到自己写的值	中继一律走 `$VAR`
3	`THEN dst = $VAR.5` 静默写成 0	写成 `= $VAR.5 + 0`
4	动作槽 2/3 的 operand 只能是 `$VAR.n`	复制 AI/$SYS 值要放动作槽 1
5	`FOR` 套用在合并条件上	去抖动隔离成单条件规则
6	`AND` 不优先于 `OR`	左结合,或改用分层覆盖写法
7	`FOR` / `COOLDOWN` 上限 65535 ms	长延迟用 `$SYS.UPTIME` 或 TIMER 计数
8	超过 3 条件／3 动作静默丢弃	拆成多条规则
9	`ON OFFDELAY FOR` 占用 dwell	串两条规则
10	`PULSE <dur>` 与 `COOLDOWN` 共用字段	同规则择一
11	不认得的行静默忽略	导入后逐条核对规则数与内容

#	陷阱	对策
12	规则名称重复只找得到第一个	名称唯一,23 字节内,建议 ASCII
13	条件 tag 读取失败会清掉边缘记忆	模块复线时预期一次假上升沿
14	cooldown 挡下时仍更新条件记忆	RISE 边缘会被吃掉,改用 `FOR` 抑制抖动
15	导入超过约 3.5 KB 被静默截断	去除注释并分段 append 导入

导出／导入不对称

设备导出的 HRL 文字**不保证能原样重新导入**:

导出内容	问题
`EVERY 30s`	解析器只认 `IF EVERY`,独立行被忽略 → TIMER 规则退化为 LEVEL
`DAILY AT 22:30`	同上,SCHEDULE 丢失
`THEN dst = operand`	COPY 形式,重新解析成常数 0
`AND ALARM "name"`	动作槽 2/3 未输出 severity
迟滞值	完全不输出

> **结论**
>
> HRL 文件要当作项目的
>
> **唯一真实来源**
>
> ,用版本控管维护。不要依赖「从设备导出再导入」的来回流程,导出只适合人工检视。

13 快速参考卡

```
RULE name :                          rule header, 23 chars max
  IF tag op literal [FOR dur]        condition 1   (op: == != > >= < <=)
  AND tag op literal                 condition 2   (OR also allowed)
  AND tag op literal                 condition 3   (left-associative)
  THEN dst = value                   action 1  (any-tag expression operand)
  AND  dst = value                   action 2  ($VAR-only operand)
  AND  dst = value                   action 3  ($VAR-only operand)
  ON LEVEL|RISE|FALL|CHANGE|BOOT|TRIGGER|OFFDELAY
  COOLDOWN dur                       min re-trigger interval
  GROUP n     PRIORITY n     DISABLED

IF EVERY dur | IF DAILY AT HH:MM | IF WEEKLY DAY,DAY AT HH:MM

value:   123 | ON | OFF | TRUE | FALSE | TOGGLE | PULSE [dur]
         operand + - * / % & | ^ literal
action:  tag = value | ALARM "x" INFO|WARN|HIGH|CRITICAL
         MQTT topic | TRIGGER rule [IMMEDIATE] | NTP_SYNC | LOG | NOOP
dur:     250ms | 5s | 0.5s | 1m | 1h        max 65535 ms
tag:     DI16_#n.DIm  DO16_#n.DOm  DO6TR_#n.DOm  AI6_#n.AIm
         $VAR.0-63    $SYS.NAME    @Device.field
```

打印本节即可作为现场速查卡。完整语意请回到对应章节。

泓洋电子科技有限公司 HOYAN Technologies Co., Ltd.
高雄市前鎮區 80665 明鳳十一街 33 巷 5 號
T +886 7 793-5098 F +886 7 793-5228 sc@hoyan.com.tw

© 2026 HOYAN Technologies Co., Ltd. All rights reserved.