HRL 語言規格

HOYAN Rule Language — 規則引擎語言參考

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

目次

00 適用範圍與版本
01 檔案結構與容量
02 RULE 宣告
03 IF 條件
04 THEN 動作
05 ON 觸發模式
06 COOLDOWN 與屬性
07 Tag 命名空間
08 掃描週期與執行模型
09 功能塊對照表
10 完整範例:艙底水高位警報盤
11 診斷工具與開機行為
12 陷阱總表
13 快速參考卡

00 適用範圍與版本

本手冊說明 HOYAN ETH-A2 通訊閘道器內建規則引擎所使用的 HOYAN Rule Language(HRL),涵蓋語法、執行模型與運作限制,適用對象為負責現場邏輯組態的控制工程師與系統整合商。

本文件內容對應韌體版本 V2.77。所述行為、容量上限與診斷介面均以該版本為準,後續版本可能變更,恕不另行通知。當本文件與裝置實際行為不一致時,以裝置實際行為為準。

HRL 是宣告式的規則語言,**不是** IEC 61131-3。沒有函式塊、沒有梯形圖、沒有 TON / TOF / R_TRIG / SR / CTU。全部邏輯由三個原語組成:

- `FOR <t>` — 條件必須持續成立
- `ON RISE` / `ON FALL` — 邊緣觸發
- `$VAR.0` ~ `$VAR.63` — 64 個 int32 中繼變數,同一掃描週期內即時可見

解析器是逐行的,沒有 AST。**不認得的行會被靜默忽略,不報錯** —— 這代表打錯關鍵字的規則會安靜地少掉一半語意。匯入後務必逐條核對規則數與內容。

> **注意**
> 本產品的規則引擎為一般用途控制邏輯,非安全等級(SIL / PL)裝置。涉及人員安全、防止設備損壞或環境保護的功能,必須由獨立的硬體安全迴路實現,不得僅依賴規則引擎。

01 檔案結構與容量

```
# 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
```

縮排無意義。關鍵字不分大小寫。`RULE` 開啟新規則,不需 `END`。**一行一個指令** —— 解析器讀完該行的第一個指令後直接跳到下一行。

`AND` / `OR` 的歸屬由「是否已看到 `THEN`」決定: `THEN` 之前填條件槽,之後填動作槽。這是解析器唯一的狀態切換。

項目	上限	備註
規則數	160	
規則名稱	23 位元組	建議使用 ASCII
Tag 名稱	23 位元組	
條件 / 規則	3	左結合,無括號
動作 / 規則	3	超出的子句靜默丟棄
群組	32	可整組啟停

項目	上限	備註
使用者變數	`$VAR.0` – `$VAR.63`	32 位元有號整數,斷電保留
作用中警報	16	
觸發歷史	64 筆	規則頁的「歷史」按鈕
`FOR` / `COOLDOWN` 時間	65535 ms	即 65.5 秒
每掃描週期輸出點數	96	超出計入 `$SYS.WQ_FAIL`
單次匯入規則數	64	更多請分批以 append 匯入
單次匯入請求大小	4096 位元組	含標頭;HRL 文字約 3.5 KB

> **陷阱**
> 超過 3 個條件或 3 個動作的子句會被
> **靜默丟棄**
> 。寫了第 4 個 `AND` 不會有任何錯誤訊息,那行就是不存在。要更多條件請拆成多條規則,用 `$VAR` 串接。

> **陷阱 — 匯入體積上限**
> 單次匯入的整個請求(含標頭)不得超過
> **4096 位元組**
> 。超出的部分會被
> **靜默丟棄**
> ,裝置不回傳任何錯誤。扣除標頭後,HRL 文字實際可用約
> **3.5 KB**
> 。
>
> 大型專案請去除註解,並依規則邊界切成多段:第一段用 `mode: "replace"`,其餘用 `mode: "append"`。append 會依序 upsert 在既有規則之後,執行順序完整保留。

02 RULE 宣告

名稱取到 `:` 或行尾為止,尾端空白去除,截斷到 23 字元。冒號可省略。

名稱是 `TRIGGER` 動作與 `/api/rules/fire` 的查找鍵(名稱完全比對)。**名稱重複時只會找到第一個。**

新規則預設值: `enable=1` 、 `priority=50` 、 `expr*=NONE` 、 `action3=NOOP` 。

> 規則名稱建議用 ASCII。名稱欄位是 24 位元組,UTF-8 中文一個字佔 3 位元組,只能放 7 個中文字,而且截斷發生在位元組邊界,可能切出無效的 UTF-8 序列。

03 IF 條件

一般條件

```
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
```

運算子: `==` `!=` `>` `>=` `<` `<=` `ANY` 。未知運算子 fallback 為 `==` 。值: `ON` / `TRUE` → 1, `OFF` / `FALSE` → 0,其餘以十進位整數解讀。**閾值只能是常數,不能是另一個 tag。**

時間排程條件

```
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` 解析度為整秒,小於 1s 會被強制成 1s 並印 UART 警告,上限 65535s。星期字串 `SUN MON TUE WED THU FRI SAT` ,逗號分隔無空白。`DAILY` / `WEEKLY` 由 RTC 比對時分,同一分鐘只觸發一次。

AND / OR 追加條件

```
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 個追加子句填 cond2,第 2 個填 cond3,之後丟棄。求值為**左結合、無括號**: $((c1 \oplus_1 c2) \oplus_2 c3)$ 。

> **陷阱**
>
> `AND` 沒有優先於 `OR` 。`A OR B AND C` 在 HRL 中等於 `(A OR B) AND C` ,不是布林代數的慣例。需要混合運算就用 `$VAR` 分階段。

```
# 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
```

任一條件的 tag 讀取失敗 → 整條規則本次掃描視為無效,引擎記住的「上一次條件結果」與持續計時一併歸零,並在觸發歷史留下 `e:cond`。這代表模組離線時規則的邊緣記憶會被抹掉,重新上線會產生一次上升緣。

FOR — 持續時間 (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
```

未達持續時間期間,引擎會把「上一次條件結果」視為不成立,所以 `FOR` + `ON RISE` 會在時間達成的那一次掃描正確產生上升緣。這個特性也是自我振盪閃爍源能成立的原因(見 §09)。

> **陷阱**
>
> `FOR` 套用在
>
> **三個條件的合併結果**
>
> 上,不是單一條件。多條件規則中任一條件變動都會重啟 dwell 計時。去彈跳邏輯必須隔離成單條件規則,輸出到 `$VAR`。
>
> `FOR` 的時間
>
> **最大 65535 MS(65.5 秒)**
>
> 。`FOR 2m` 會被夾到 65535 ms。

04 THEN 動作

```
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
```

語法	行為
`<tag> = <value>`	寫入常數
`<tag> = <operand> <op> <literal>`	算術／位元運算式
`<tag> = TOGGLE`	讀現值後寫反相
`<tag> = PULSE [<dur>]`	立刻寫 1,經指定時間(預設 3000 ms)後自動寫 0
`ALARM "<name>" <sev>`	產生警報, `INFO` `WARN` `HIGH` `CRITICAL`
`MQTT <topic>`	發佈 MQTT
`TRIGGER <rule> [IMMEDIATE]`	觸發另一條規則
`NTP_SYNC`	觸發網路校時
`LOG` / `NOOP`	只寫日誌

運算式

運算子 `+ - * / % & | ^` 。 `/` 與 `%` 的 literal 為 0 時分別回傳 operand 與 0,不會除零當機。

```
# 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	儲存欄位
動作 1 (`THEN`)	任意 tag — DI/DO/AI/AO/ `$VAR` / `$SYS` / `@ext`	完整 tag 名稱
動作 2 / 3 (`AND`)	只能是 `$VAR.n`	僅 `$VAR` 編號

> **陷阱**
> 引擎內部支援直接複製,但 HRL **沒有對應的書寫方式**。`THEN DO = $VAR.5` 會被當成常數解讀,結果是 **0**,而且完全不會報錯。

```
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 觸發模式

未指定時預設 `LEVEL`;未知字串也 fallback 為 `LEVEL`。`EVERY` / `DAILY` / `WEEKLY` 由 `IF` 那行隱含設定,不需再寫 `ON`。

模式	語意
`LEVEL`	條件為真 → **每個 tick 都執行動作**
`RISE`	條件 false→true 的那一 tick 執行一次
`FALL`	條件 true→false 的那一 tick 執行一次
`CHANGE`	條件任何變化執行一次
`BOOT`	開機時執行一次,正常 tick 中完全跳過
`TRIGGER`	只被別的規則喚醒;若有 `IF` 則當作閘門
`OFFDELAY` / `TOF`	延遲關,對標 IEC 61131-3 TOF

OFFDELAY (TOF)

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

條件成立 → 立刻寫 `action_value`(為 0 則寫 1),取消待關倒數。條件轉 false → 啟動倒數,倒數到才寫 0。`FOR 0` 退化為一般 LEVEL。

> **陷阱**
> `ON OFFDELAY FOR ...` 的 `FOR` 是 **關閉延遲**,佔用了該規則唯一的時間欄位。同一條規則 **無法再用 `IF ... FOR` 做輸入去彈跳**。要「先去彈跳再延遲關」得串兩條規則。

規則連鎖

```
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 執行(事件語意); `TRIGGER … IMMEDIATE` = 本 tick 執行(呼叫語意)。連鎖保護:同一規則每 tick 最多 fire 1 次、每 tick 全域上限 64 次、IMMEDIATE 遞迴深度 8、工作串列 16。

開機規則

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

06 COOLDOWN 與屬性

`COOLDOWN` 為最小重觸發間隔,最大 65535 ms。第一次觸發不受限。被 cooldown 擋下時規則**不執行動作**,但「**上一次條件結果**」仍會更新,所以 RISE 邊緣會被吃掉。

> `THEN <tag> = PULSE <dur>` 會把 `<dur>` 與獨立的 `COOLDOWN` 行共用同一個時間欄位。同一條規則不要同時使用兩者。

屬性	語法	說明
`GROUP`	`GROUP 3` / `GROUP Pumps`	1~32;字串會 hash(`h = h*31 + c`)% 32 + 1。可整組啟停。
`PRIORITY`	`PRIORITY 10`	1=最高,99=最低,預設 50。**不影響執行順序**。
`DISABLED`	`DISABLED`	`enable = 0` ,規則保留但不執行。

> **陷阱 — PRIORITY 不影響執行順序**
>
> `PRIORITY` 會被存檔、也會被匯出,但引擎在每個掃描週期是由上而下依序評估每一條規則,完全不參考這個欄位。
>
> **執行順序 = 規則在陣列中的索引順序 = HRL 檔案中的書寫順序。**
>
> 要控制順序就控制書寫順序。

07 Tag 命名空間

前綴	命名空間	範例	讀	寫
(無)	本地 IO 模組	`DI16_#2.DI0`	✔	✔

前綴	命名空間	範例	讀	寫
`$SYS.`	系統變數	`$SYS.UPTIME`	✔	唯讀
`$VAR.`	使用者變數 0–63	`$VAR.12`	✔	✔
`@`	外部設備	`@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 的實際解析規則

本地 tag 的解析比表面看起來寬鬆很多:

- 找第一個 `#`,其後的十進位數字 = slave id
- 掃描整個字串,只要出現子字串 `AI` 或 `AO` → register;否則 coil
- 取字串尾端的連續數字 = point index
- slave id 必須對應到一個已上線的模組,否則不算本地 tag

也就是說模組型號字串本身不參與比對, `DO16_#1.DO7` 與 `XX_#1.7` 解析結果相同。仍建議照規範寫,因為 Web UI 與 Modbus 映射表用的是完整名稱。Coil point 限 0–15。

模組	ID	類別
`DI16`	0x471C	16 路數位輸入
`DO16`	0x471D	16 路數位輸出
`AI6`	0x471F	6 路類比輸入
`DO6TR`	0x4720	6 路繼電器輸出

$SYS 全清單

名稱	內容
`UPTIME`	開機後秒數
`RTC_YEAR` … `RTC_SEC`	RTC 年月日時分秒
`RTC_WEEK`	0=週日 … 6=週六
`CPU_LOAD`	CPU 負載百分比
`RULE_ENABLED`	規則引擎總開關 0/1
`RULE_COUNT`	啟用中的規則數
`MQTT_CONNECTED`	MQTT 連線 0/1

名稱	內容
`ETH_LINK`	乙太網路連線 0/1
`SLAVE_COUNT`	線上 IO 模組數
`WQ_FAIL`	寫入佇列滿而丟棄的累計數

名稱**區分大小寫**,必須全大寫。`$VAR` 是 32 位元有號整數,存於內部非揮發記憶體,重開機保留 —— 若不希望保留請加一條 `ON BOOT` 歸零規則。

08 掃描週期與執行模型

引擎以固定週期反覆執行下列三個階段。預設週期 20 ms(50 Hz),可調 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
```

讀寫語意的關鍵差異

Tag 類型	本 tick 讀到的值	寫入何時生效
本地 IO (DI/DO/AI/AO)	tick 開始時的凍結快照	tick 結束刷出;同 tick 內後續規則讀不到
`$VAR` / `$SYS` / `@ext`	即時值	立即生效,同 tick 內後續規則讀得到

這是設計意圖:本地 IO 凍結可根除同 tick 連鎖抖動;`$VAR` 即時是為了讓狀態機能在一個掃描週期內完成推進。

> **陷阱**
>
> **不要用 DO 點當中繼位元。**
>
> 規則讀不到自己或前面規則在同一 tick 寫進去的 DO 值,狀態機會延遲一整個掃描週期,而且行為會隨規則順序改變。所有中繼邏輯走 `$VAR`。

同一個 DO 被多條規則寫

後執行者覆蓋前執行者(都寫進同一份輸出映像),輸出更新階段只送出最終值。這讓「先寫預設值、後面條件成立再覆蓋」的分層寫法成為 HRL 最實用的技巧 —— 它繞開了「只有 3 條件、無括號、AND 不優先於 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
```

寫入佇列

單一掃描週期若寫超過 96 個實體點,超出的會被丟棄,累計數暴露在 `$SYS.WQ_FAIL` ,可用規則自我監控:

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

09 功能塊對照表

以下每個模式都只用 `FOR` 、 `ON RISE/FALL` 、 `$VAR` 三樣東西,在 V2.77 語意下驗證過。

IEC 61131-3	HRL 對應
TON	`IF ⋯ FOR t` + `ON RISE` ,輸出到 `$VAR`
TOF	原生 `ON OFFDELAY FOR t`
R_TRIG / F_TRIG	原生 `ON RISE` / `ON FALL`
SR / RS	兩條 `ON RISE` 規則寫同一個 `$VAR`
CTU / CTD	`$VAR = $VAR + 1` 於 `ON RISE`
MOVE	`dst = src + 0` (動作槽 1)
BLINK	兩條互鎖的 `FOR` + `ON RISE` 規則
SEL / MUX	分層覆蓋寫法(§08)

IEC 61131-3	HRL 對應
HYSTERESIS	僅 Web UI / JSON API 可設,HRL 無語法

TON — 接通延遲

```
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
```

務必隔離成單條件規則。`FOR` 掛在多條件規則上會被其他條件的變動重啟計時。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
```

置位優先或復位優先由書寫順序決定 —— 寫在後面的覆蓋前面的。上例是復位優先。

閃爍源(與掃描週期無關)

```
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
```

自我振盪成立的原因:持續時間未達時引擎把「上一次條件結果」視為不成立,達成當下必產生上升緣。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)更靈活。

計數器

```
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
```

啟停自保持 + 跳脫

```
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`:跳脫條件成立期間必須持續壓住輸出,不能只在邊緣清一次。順序上它寫在 `motor_stop` 之後,所以跳脫優先於啟動。

雙泵輪替

```
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
```

通訊監視

```
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
```

排程

```
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 完整範例:艙底水高位警報盤

7 點船用艙底水警報盤,取代一台 20 點 PLC 加 7 顆 H3Y-4 延時繼電器。行為:未確認時燈閃 1 Hz 加蜂鳴器;按確認鈕蜂鳴器停、燈轉常亮;液位復歸燈滅;確認後若有新點成立蜂鳴器重響。共 54 條規則。

```
# ---- 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)
```

每個警報點的核心兩條規則

```
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` 同時做三件事:去彈跳後鎖存 DLY、標記未確認、拉起蜂鳴器自保持。三個動作剛好填滿一條規則的動作槽。`FOR 5s` 對應原機 H3Y-4 的 0–30 秒可調延時,防止船舶搖晃造成誤報。

確認鈕

```
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
```

一個上升緣要清 8 個旗標,而一條規則只有 3 個動作槽,所以拆成 3 條同條件規則。三條各自記住自己的條件狀態,同一個上升緣會同時觸發,不需要 `TRIGGER` 串接。這三條必須寫在所有 `p*_ton` 之前,否則同一 tick 內到來的新警報會被誤消音。

燈輸出

DO16 是 PNP 源型輸出,指示燈與蜂鳴器直接掛在輸出點上,不經繼電器也不調光。原圖的 DM:5K 調光器、調光模組與 RY1 中繼在此設計中全部省略,燈測試按鈕直接接入 DI。

燈的三態(滅／常亮／閃爍)加上燈測試覆蓋,用 §08 的分層覆蓋寫法,見該節範例。四層依序寫在檔案中,燈測試那層必須是全檔最後。

復示乾接點

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

> **陷阱**
> 復示接點必須追隨 `DLY`,
> **不能追隨燈輸出**
> 。原圖的復示是計時器的實體接點,與 PLC 的確認邏輯無關 —— 按了確認鈕之後,機艙控制室那邊仍須維持有警報。若誤接燈輸出,確認後的閃爍會讓對方收到重複的假警報。

規則排列順序

- `ON BOOT` 初始化
- 閃爍源 ×2
- 確認鈕 ×3
- 去彈跳 + 復歸 ×14
- 燈輸出三層 ×21
- 復示 ×9
- 蜂鳴器 ×1
- 燈測試 ×3(必須最後)

11 診斷工具與開機行為

觸發歷史

規則頁左下的「歷史」按鈕開啟,保留最近 64 筆紀錄,欄位為時間、規則、結果。

結果	意義
`skip`	條件不成立,未觸發
`trig`	正常執行

結果	意義
`dry`	模擬模式,未實際寫出
`e:cond`	條件的 tag 讀取失敗
`e:act`	動作寫入失敗
`e:chain`	連鎖超過上限,或 `TRIGGER` 的規則名稱找不到
`e:wq`	本次掃描有輸出因佇列已滿而丟棄

除錯模式

規則頁的「偵錯」進入。把一條規則的評估過程拆成七個階段逐一顯示,每個階段都看得到引擎實際讀到的值與判定結果 —— 這是唯一能回答「為什麼這條規則沒動作」的工具。

階段	檢查什麼
1 READ_TAG	條件的 tag 是否讀得到,讀到什麼值
2 條件 1 評估	與閾值比較的結果
3 條件 2 合併	追加條件與 AND / OR 合併後的結果
4 持續檢查	`FOR` 的時間是否已達成
5 邊緣偵測	依 `ON` 模式判斷本次是否該觸發
6 冷卻檢查	是否仍在 `COOLDOWN` 期間內
7 執行動作	實際寫出的目標與值

範圍可選全部規則、目前選中的規則、或單一群組;監控間隔 0.5 / 1 / 2 / 5 秒。另有「強制執行」直接跑一次動作,以及「條件注入」自動寫入能讓條件成立的值 —— 若來源是實體輸入則無法注入,必須改變實際輸入或改用虛擬模組。

監看與交叉參照

監看表可釘選任意 tag 觀察即時值。交叉參照列出哪些規則**讀取**、哪些規則**寫入**該 tag —— 回答「這個值到底是誰在動」。異常篩選只列出 tag 讀取失敗或執行錯誤的規則;條件不成立不算異常。

「保持」凍結引擎,輸出**維持現狀**(不是停用歸零),10 分鐘無操作會自動恢復;「單步」放行一個掃描週期。「Force 強制置位」最多 16 點,優先於所有寫入者,重開機即清除,期間規則仍照常評估。

模擬模式

引擎設定裡的開關。擋掉所有實體輸出、MQTT 與警報,但**放行** `$VAR` **寫入**,讓狀態機在模擬時仍能推進。注意 `$VAR` 的值是真的寫進去了,退出模擬時不會自動回復。

用規則自我監控

下列系統變數可直接寫進規則,把診斷變成主動告警:

變數	監控什麼
`$SYS.WQ_FAIL`	輸出因佇列已滿被丟棄的累計次數

變數	監控什麼
`$SYS.SLAVE_COUNT`	線上 IO 模組數,掉線即可告警
`$SYS.ETH_LINK`	乙太網路連線
`$SYS.MQTT_CONNECTED`	MQTT 連線
`$SYS.CPU_LOAD`	CPU 負載百分比

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

開機策略

在引擎設定的「啟用時 BOOT 規則」下拉選單設定:

選項	行為
BOOT 優先(跳過 Safety)	略過安全策略,直接跑 `ON BOOT` 規則(預設)
Safety 後 BOOT(先歸零)	安全策略先把所有 DO 歸零,等 1 秒,再跑 BOOT 規則
只用 Safety(不跑 BOOT)	不執行 BOOT 規則

`ON BOOT` 規則在正常掃描中完全跳過,只在開機、引擎總開關由關轉開、以覆蓋模式匯入 HRL、或按下引擎設定裡的「手動 fire BOOT」時執行。

12 陷阱總表

#	陷阱	對策
1	`PRIORITY` 不影響執行順序	用書寫順序控制
2	本地 IO 讀凍結快照,同 tick 讀不到自己寫的值	中繼一律走 `$VAR`
3	`THEN dst = $VAR.5` 靜默寫成 0	寫成 `= $VAR.5 + 0`
4	動作槽 2/3 的 operand 只能是 `$VAR.n`	複製 AI/$SYS 值要放動作槽 1
5	`FOR` 套用在合併條件上	去彈跳隔離成單條件規則
6	`AND` 不優先於 `OR`	左結合,或改用分層覆蓋寫法
7	`FOR` / `COOLDOWN` 上限 65535 ms	長延遲用 `$SYS.UPTIME` 或 TIMER 計數
8	超過 3 條件／3 動作靜默丟棄	拆成多條規則
9	`ON OFFDELAY FOR` 佔用 dwell	串兩條規則
10	`PULSE <dur>` 與 `COOLDOWN` 共用欄位	同規則擇一
11	不認得的行靜默忽略	匯入後逐條核對規則數與內容

#	陷阱	對策
12	規則名稱重複只找得到第一個	名稱唯一,23 位元組內,建議 ASCII
13	條件 tag 讀取失敗會清掉邊緣記憶	模組復線時預期一次假上升緣
14	cooldown 擋下時仍更新條件記憶	RISE 邊緣會被吃掉,改用 `FOR` 抑制抖動
15	匯入超過約 3.5 KB 被靜默截斷	去除註解並分段 append 匯入

匯出／匯入不對稱

裝置匯出的 HRL 文字**不保證能原樣重新匯入**:

匯出內容	問題
`EVERY 30s`	解析器只認 `IF EVERY`,獨立行被忽略 → TIMER 規則退化為 LEVEL
`DAILY AT 22:30`	同上,SCHEDULE 遺失
`THEN dst = operand`	COPY 形式,重新解析成常數 0
`AND ALARM "name"`	動作槽 2/3 未輸出 severity
遲滯值	完全不輸出

> **結論**
>
> HRL 檔案要當作專案的
>
> **唯一真實來源**
>
> ,用版本控管維護。不要依賴「從裝置匯出再匯入」的來回流程,匯出只適合人工檢視。

13 快速參考卡

```
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
```

列印本節即可作為現場速查卡。完整語意請回到對應章節。

泓洋電子科技有限公司 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.