HOYAN ETH-A2 · STM32H743 · ファームウェア V2.77

HRL 言語仕様書

HOYAN Rule Language — ルールエンジン言語リファレンス

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

目次

00 適用範囲とバージョン
01 ファイル構造と容量制限
02 RULE 宣言
03 IF 条件
04 THEN 動作
05 ON トリガモード
06 COOLDOWN と属性
07 タグ名前空間
08 スキャンサイクルと実行モデル
09 ファンクションブロック対応表
10 実装例:ビルジ高水位警報盤
11 診断ツールと起動動作
12 落とし穴一覧
13 クイックリファレンス

00 適用範囲とバージョン

本マニュアルは、HOYAN ETH-A2 通信ゲートウェイに内蔵されたルールエンジンで使用する HOYAN Rule Language(HRL)について、構文・実行モデル・動作上の制限を説明するものです。対象読者は、現場ロジックの構成を担当する制御技術者およびシステムインテグレーターです。

本書の記載内容はファームウェア V2.77 に対応します。記載された動作・容量上限・診断インターフェースは同バージョンに基づくものであり、以降のバージョンで予告なく変更される場合があります。本書の記載と実機の動作が異なる場合は、実機の動作を優先してください。

HRL は宣言型のルール言語であり、IEC 61131-3 **ではありません**。ファンクションブロックもラダー図も、TON / TOF / R_TRIG / SR / CTU もありません。すべてのロジックは 3 つのプリミティブから組み立てます:

- `FOR <t>` — 条件が一定時間成立し続けること
- `ON RISE` / `ON FALL` — エッジトリガ
- `$VAR.0` ~ `$VAR.63` — 64 個の int32 中間変数。同一スキャン内で即時に参照可能

パーサーは行単位で処理し、AST を持ちません。**認識できない行は黙って無視され、エラーになりません。** つまりキーワードのタイプミスはルールの意味を静かに半分削り取ります。インポート後は必ずルール数と内容を照合してください。

> **注意**
> 本製品のルールエンジンは汎用制御ロジックであり、安全度水準(SIL / PL)に適合した機器ではありません。人身安全、機器損傷の防止、環境保護に関わる機能は、独立したハードウェア安全回路によって実現する必要があり、ルールエンジンのみに依存してはなりません。

01 ファイル構造と容量制限

```
# 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
```

インデントに意味はありません。キーワードは大文字小文字を区別しません。`RULE` が新しいルールを開始し、`END` は不要です。**1 行 1 ディレクティブ** — パーサーは行の最初のディレクティブを読んだ時点で次の行へ移ります。

`AND` / `OR` が条件スロットと動作スロットのどちらを埋めるかは、`THEN` を既に読んだかどうかだけで決まります。これがパーサー唯一の状態遷移です。

項目	上限	備考
ルール数	160	
ルール名	23 バイト	ASCII 推奨

項目	上限	備考
タグ名	23 バイト	
条件数/ルール	3	括弧なしの左結合
動作数/ルール	3	超過分は黙って破棄
グループ	32	グループ単位で有効・無効化
ユーザー変数	`$VAR.0` – `$VAR.63`	32 ビット符号付き、電源断でも保持
アクティブ警報	16	
トリガ履歴	64 件	ルール画面の「履歴」ボタン
`FOR` / `COOLDOWN` の時間	65535 ms	= 65.5 秒
1 スキャンの出力点数	96	超過分は `$SYS.WQ_FAIL` に計上
1 回のインポート	64	それ以上は append で分割
インポート要求サイズ	4096 バイト	ヘッダ込み。HRL テキストは約 3.5 KB

> **落とし穴**
> 4 つ目以降の条件・動作句は
> **黙って破棄されます**
> 。4 番目の `AND` はエラーにすらならず、その行は存在しないものとして扱われます。条件を増やしたい場合はルールを分割し、`$VAR` で連結してください。

> **落とし穴 — インポートサイズ上限**
> 1 回のインポート要求は、ヘッダを含めて
> **4096 バイト**
> 以内に収まる必要があります。超過分は
> **黙って破棄され**
> 、機器はエラーを返しません。ヘッダを差し引くと、使用できる HRL テキストは約
> **3.5 KB**
> です。
>
> 大規模なプロジェクトではコメントを除去し、ルール境界で分割してください。最初のチャンクは `mode: "replace"` 、以降は `mode: "append"` で送信します。append は既存ルールの後ろに順次 upsert するため、実行順序は完全に保たれます。

02 RULE 宣言

名前は `:` または行末までを取り、末尾の空白は除去され、23 文字に切り詰められます。コロンは省略可能です。

名前は `TRIGGER` 動作と `/api/rules/fire` の検索キーです(名前の完全一致)。**名前が重複した場合、常に最初の 1 件だけが見つかります。**

新規ルールの既定値: `enable=1` 、 `priority=50` 、 `expr*=NONE` 、 `action3=NOOP` 。

ルール名は ASCII を推奨します。フィールドは 24 バイトで、UTF-8 の漢字は 1 文字 3 バイトのため 7 文字しか入らず、切り詰めはバイト境界で起きるため不正な UTF-8 列が生成される恐れがあります。

03 IF 条件

基本条件

```
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
```

演算子: `==` `!=` `>` `>=` `<` `<=` `ANY` 。不明な演算子は `==` にフォールバックします。値は `ON` / `TRUE` → 1、 `OFF` / `FALSE` → 0、その他は 10 進整数として解釈。**しきい値は定数のみで、他のタグは指定できません。**

スケジュール条件

```
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` の分解能は秒単位です。1 秒未満は 1 秒に強制され UART 警告が出ます。上限は 65535 秒。曜日文字列は `SUN MON TUE WED THU FRI SAT` 、カンマ区切りでスペースなし。 `DAILY` / `WEEKLY` は RTC の時分と照合し、同一分内では 1 回のみ発火します。

AND / OR による条件追加

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

1 つ目の追加句が cond2、2 つ目が cond3 を埋め、それ以降は破棄されます。評価は**括弧なしの左結合**です: $((c1 \oplus_1 c2) \oplus_2 c3)$ 。

> **落とし穴**
>
> `AND` は `OR` より優先されません。HRL では `A OR B AND C` は `(A OR B) AND C` であり、通常のブール代数の優先順位とは異なります。混在させたい場合は `$VAR` で段階化してください。

```
# 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
```

いずれかの条件のタグ読み取りに失敗すると、そのスキャンではルール全体が無効となり、記憶されている前回の条件結果と継続計時がクリアされ、トリガ履歴に `e:cond` が記録されます。つまりモジュールがオフラインになるとエッジ記憶が消え、復帰時に立ち上がりエッジが 1 回発生します。

FOR — 継続時間 (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
```

継続時間が未達成の間、エンジンは前回の条件結果を不成立として扱います。そのため `FOR` と `ON RISE` の組み合わせは、時間達成のスキャンで正しく立ち上がりエッジを生成します。この性質が自励発振フラッシャ(§09)が成立する理由でもあります。

> **落とし穴**
>
> `FOR` は
> **3 つの条件を合成した結果**
> に適用され、単一条件には適用されません。複数条件のルールではいずれかの条件が変化すると dwell が再スタートします。デバウンス処理は単一条件のルールに分離し、`$VAR` に出力してください。
>
> `FOR` の時間は
> **最大 65535 MS(65.5 秒)**
> です。`FOR 2m` は 65535 ms にクランプされます。

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
```

構文	動作
`<tag> = <value>`	定数を書き込む
`<tag> = <operand> <op> <literal>`	算術・ビット演算式
`<tag> = TOGGLE`	現在値を読んで反転して書く
`<tag> = PULSE [<dur>]`	即座に 1 を書き、指定時間(既定 3000 ms)後に自動で 0 を書く
`ALARM "<name>" <sev>`	警報を発生: `INFO` `WARN` `HIGH` `CRITICAL`
`MQTT <topic>`	MQTT に発行
`TRIGGER <rule> [IMMEDIATE]`	他のルールを起動
`NTP_SYNC`	NTP 時刻同期を起動
`LOG` / `NOOP`	ログのみ

演算式

演算子は `+ - * / % & | ^`。リテラルが 0 の場合、`/` は operand を、`%` は 0 を返し、ゼロ除算で落ちることはありません。

```
# 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
```

スロット	operand に指定可	格納フィールド
動作 1 (`THEN`)	**任意のタグ** — DI/DO/AI/AO/ `$VAR` / `$SYS` / `@ext`	完全なタグ名
動作 2 / 3 (`AND`)	`$VAR.n` **のみ**	`$VAR` 番号のみ

> **落とし穴**
> エンジン内部では単純なコピーに対応していますが、HRL には
> **それを書く方法がありません**
> 。`THEN DO = $VAR.5` は定数として解釈され、結果は
> **0**
> 。エラーは一切出ません。

```
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 トリガモード

省略時の既定は `LEVEL` で、不明な文字列も `LEVEL` にフォールバックします。`EVERY` / `DAILY` / `WEEKLY` は `IF` 行で暗黙にモードを設定するため `ON` は不要です。

モード	意味
`LEVEL`	条件が真 → 毎 tick 動作を実行
`RISE`	条件が false→true になった tick で 1 回
`FALL`	条件が true→false になった tick で 1 回
`CHANGE`	条件が変化するたびに 1 回
`BOOT`	起動時に 1 回。通常の tick では完全にスキップ
`TRIGGER`	他のルールからのみ起動。`IF` があればゲートとして機能
`OFFDELAY` / `TOF`	オフディレイ、IEC 61131-3 の TOF 相当

OFFDELAY (TOF)

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

条件成立 → ただちに `action_value` (0 なら 1)を書き、オフ待ちカウントダウンを取り消します。条件が false → カウントダウン開始、満了時に 0 を書きます。`FOR 0` は通常の LEVEL に退化します。

> **落とし穴**
> `ON OFFDELAY FOR ...` の `FOR` は
> **オフディレイ**
> であり、そのルール唯一の時間フィールドを占有します。したがって同一ルールで
> **`IF ... FOR` による入力デバウンスは併用できません**
> 。デバウンス→オフディレイは 2 ルールを連結してください。

ルール連鎖

```
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` は次の tick で実行(イベント的)、`TRIGGER ... IMMEDIATE` は同一 tick で実行(呼び出し的)です。連鎖保護: 同一ルールは 1 tick に 1 回、全体で 1 tick 64 回まで、IMMEDIATE の再帰深さ 8、ワークリスト 16。

起動ルール

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

06 COOLDOWN と属性

`COOLDOWN` は最小再トリガ間隔で、最大 65535 ms です。初回発火は対象外。cooldown でブロックされた場合、動作は実行されませんが**前回の条件結果は更新されます**。そのため RISE エッジは失われます。

> `THEN <tag> = PULSE <dur>` は `<dur>` は単独の `COOLDOWN` 行と同じ時間フィールドを使用します。1 つのルールで併用しないでください。

属性	構文	説明
`GROUP`	`GROUP 3` / `GROUP Pumps`	1~32。文字列はハッシュ(`h = h*31 + c`)% 32 + 1。グループ単位で有効・無効化できます。
`PRIORITY`	`PRIORITY 10`	1=最高、99=最低、既定 50。**実行順序には影響しません。**
`DISABLED`	`DISABLED`	`enable = 0` 。ルールは保持されますが実行されません。

> **落とし穴 — PRIORITY は実行順序を決めない**
>
> `PRIORITY` は保存もエクスポートもされますが、エンジンは各スキャンでルールを上から順に評価するだけで、このフィールドを参照しません。
>
> **実行順序 = 配列インデックス順 = HRL ファイル内の記述順です。**
>
> 順序を制御したければ記述順を制御してください。

07 タグ名前空間

接頭辞	名前空間	例	読	書
(なし)	ローカル IO モジュール	`DI16_#2.DI0`	✔	✔
`$SYS.`	システム変数	`$SYS.UPTIME`	✔	読取専用
`$VAR.`	ユーザー変数 0–63	`$VAR.12`	✔	✔
`@`	外部機器	`@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
```

ローカル IO タグの実際の解析規則

ローカルタグの解析は見た目よりはるかに緩やかです:

- 最初の `#` を探し、その後の 10 進数字が slave id
- 文字列全体を走査し、部分文字列 `AI` または `AO` があれば register、なければ coil
- 文字列**末尾**の連続する数字が point index
- slave id はオンラインのモジュールに解決できる必要があり、できなければローカルタグ扱いになりません

つまり型番文字列自体は照合に関与せず、`DO16_#1.DO7` と `XX_#1.7` は同じ結果になります。それでも Web UI と Modbus マップは完全な名前を使うため、正式表記で記述してください。Coil の point は 0–15 に制限されます。

モジュール	ID	種別
`DI16`	0x471C	16 点デジタル入力
`DO16`	0x471D	16 点デジタル出力
`AI6`	0x471F	6 点アナログ入力
`DO6TR`	0x4720	6 点リレー出力

$SYS 一覧

名称	内容
`UPTIME`	起動からの秒数
`RTC_YEAR` … `RTC_SEC`	RTC 年月日時分秒
`RTC_WEEK`	0=日曜 … 6=土曜
`CPU_LOAD`	CPU 負荷率
`RULE_ENABLED`	ルールエンジン全体スイッチ 0/1

名称	内容
`RULE_COUNT`	有効なルール数
`MQTT_CONNECTED`	MQTT 接続 0/1
`ETH_LINK`	イーサネットリンク 0/1
`SLAVE_COUNT`	オンライン IO モジュール数
`WQ_FAIL`	書込みキュー満杯による破棄の累計

名前は**大文字小文字を区別し**、すべて大文字である必要があります。`$VAR` は 32 ビット符号付き整数で、不揮発メモリに保存され再起動後も保持されます。保持したくない場合は `ON BOOT` でクリアするルールを追加してください。

08 スキャンサイクルと実行モデル

エンジンは以下の 3 段階を一定周期で繰り返します。既定周期は 20 ms(50 Hz)、5–100 ms の範囲で調整可能です。

```
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
```

読み書きセマンティクスの決定的な差

タグ種別	この tick で読める値	書込みの反映タイミング
ローカル IO (DI/DO/AI/AO)	tick **開始時の凍結スナップショット**	tick 終了時にフラッシュ。同一 tick 内の後続ルールからは見えない
`$VAR` / `$SYS` / `@ext`	**即時値**	**即座に反映**。同一 tick 内の後続ルールから参照可能

これは意図的な設計です。ローカル IO を凍結することで同一 tick 内のチャタリングを根絶し、`$VAR` を即時にすることで状態機械が 1 スキャン内で完全に進めるようにしています。

> **落とし穴**
> **DO 点を中間ビットとして使わないでください。**
> 同一 tick 内で書き込まれた DO 値は読み取れないため、状態機械はスキャン 1 周期分遅れ、挙動がルール順序に左右されます。中間ロジックはすべて `$VAR` を経由させてください。

同一 DO を複数ルールが書く場合

後から実行されたルールが優先されます(いずれも同じ出力イメージに書き込み、出力更新段階では最終値のみを出力)。この性質により「まず既定値を書き、後続の条件で上書きする」階層化手法が HRL で最も有用なテクニックになります。3 条件制限・括弧なし・AND/OR 優先順位の問題をまとめて回避できます。

```
# 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
```

書込みキュー

1 スキャンで 96 点を超える物理書込みは破棄され、累計数が `$SYS.WQ_FAIL` として公開されるため、ルールで自己監視できます:

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

09 ファンクションブロック対応表

以下のパターンはすべて `FOR`、`ON RISE/FALL`、`$VAR` の 3 つだけで構成され、V2.77 のセマンティクスで検証済みです。

IEC 61131-3	HRL 相当
TON	`IF ... FOR t` + `ON RISE`、出力は `$VAR` へ
TOF	ネイティブの `ON OFFDELAY FOR t`
R_TRIG / F_TRIG	ネイティブの `ON RISE` / `ON FALL`
SR / RS	同じ `$VAR` に書く 2 つの `ON RISE` ルール
CTU / CTD	`ON RISE` で `$VAR = $VAR + 1`
MOVE	`dst = src + 0` (動作スロット 1)
BLINK	相互にインターロックする 2 つの `FOR` + `ON RISE` ルール

IEC 61131-3	HRL 相当
SEL / MUX	階層的上書き(§08)
HYSTERESIS	Web UI / JSON API のみ。HRL に構文なし

TON — オンディレイ

```
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
```

必ず単一条件のルールに分離してください。複数条件のルールに `FOR` を付けると、他の条件の変化でタイマがリスタートします。上限は 65.5 秒です。

SR — 自己保持

```
# 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
```

セット優先かリセット優先かは記述順で決まります。後に書いたほうが優先されます。上の例はリセット優先です。

フラッシャ(スキャン周期に非依存)

```
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
```

自励発振する理由は、継続時間が未達成の間エンジンが前回の条件結果を不成立として扱うため、達成した瞬間に必ず立ち上がりエッジが生じるからです。500ms ON / 500ms OFF = 1 Hz。`EVERY` の下限 1 秒により 0.5 Hz が上限となる `IF EVERY 1s THEN $VAR.0 = TOGGLE` より柔軟です。

カウンタ

```
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
```

始動・停止自己保持とトリップ

```
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` は `ON RISE` ではなく `ON LEVEL` を使います。トリップ条件が成立している間は出力を押さえ続ける必要があり、エッジで 1 回クリアするだけでは不十分だからです。ファイル上 `motor_stop` の後に置くため、トリップが始動より優先されます。

2 台ポンプの交互運転

```
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
```

スケジュール

```
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 実装例:ビルジ高水位警報盤

7 点の船舶用ビルジ警報盤で、20 点 PLC と 7 個の H3Y-4 タイマリレーを置き換えます。動作: 未確認時はランプが 1 Hz で点滅しブザーが鳴動、確認ボタンでブザー停止・ランプ常時点灯、水位復帰でランプ消灯、確認後に新規点が成立するとブザーが再鳴動。ルール数 54。

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

各警報点の中核 2 ルール

```
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` は 3 つの処理を同時に行います: デバウンス後の DLY をラッチ、未確認フラグを立てる、ブザー自己保持をセット。3 つの動作でルールの動作スロットがちょうど埋まります。`FOR 5s` は元機の H3Y-4 の 0–30 秒可変遅延に相当し、船体動揺による誤報を防ぐためのものです。

確認ボタン

```
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
```

1 つの立ち上がりエッジで 8 個のフラグをクリアする必要がありますが、1 ルールの動作スロットは 3 つのため、同一条件の 3 ルールに分割します。各ルールが独自に条件状態を記憶するため、同じエッジで 3 つとも発火し、`TRIGGER` による連結は不要です。これらはすべての `p*_ton` より前に記述する必要があります。さもないと同一 tick に到来した新規警報が誤って消音されます。

ランプ出力

DO16 は PNP ソース出力のため、表示灯とブザーは出力点に直結し、中継リレーも調光も使いません。元図の DM:5K 調光器・調光モジュール・RY1 中継リレーはすべて省略され、ランプテスト押ボタンは DI に直結します。

ランプの 3 状態(消灯／常時点灯／点滅)とランプテストの上書きは §08 の階層化パターンを使います(同節の例を参照)。4 つの層をファイル順に並べ、ランプテストの層はファイル全体の最後に置きます。

復示ドライ接点

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

> **落とし穴**
> 復示接点は `DLY` に追従させる必要があり、
> **ランプ出力に追従させてはいけません**
> 。元図の復示はタイマの物理接点であり、PLC の確認ロジックとは無関係です。確認ボタンを押した後も機関制御室側は警報継続を見る必要があります。ランプ出力に接続すると、点滅開始後に相手側へ偽の繰り返し警報が送られます。

ファイル内のルール順序

- `ON BOOT` 初期化
- フラッシャ ×2
- 確認ボタン ×3
- デバウンス + 復帰 ×14
- ランプ出力 3 層 ×21
- 復示 ×9
- ブザー ×1
- ランプテスト ×3(最後に配置)

11 診断ツールと起動動作

トリガ履歴

ルール画面下部の「履歴」ボタンから開きます。直近 64 件が保持され、時刻・ルール・結果が表示されます。

結果	意味
`skip`	条件不成立、未発火

結果	意味
`trig`	正常実行
`dry`	シミュレーションモード、書込みなし
`e:cond`	条件のタグ読み取り失敗
`e:act`	動作の書込み失敗
`e:chain`	連鎖が上限超過、または `TRIGGER` 対象名が見つからない
`e:wq`	キュー満杯によりこのスキャンで出力が破棄された

デバッグモード

ルール画面の「デバッグ」から入ります。ルールの評価を 7 段階に分解し、各段階でエンジンが実際に読み取った値と判定を表示します。「なぜこのルールが動かないのか」に答えられる唯一のツールです。

段階	確認内容
1 READ_TAG	条件のタグが読めるか、値は何か
2 条件 1 評価	しきい値との比較結果
3 条件 2 の合成	追加条件を AND / OR で合成した結果
4 継続確認	`FOR` の時間が達成されたか
5 エッジ検出	`ON` モードに従い今回発火すべきか
6 クールダウン	`COOLDOWN` 期間中かどうか
7 動作実行	実際に書き込まれた対象と値

範囲は全ルール・選択中のルール・単一グループから選べ、監視間隔は 0.5 / 1 / 2 / 5 秒です。動作を 1 回実行する「強制実行」と、条件が成立する値を書き込む「条件注入」もあります。ソースが物理入力の場合は注入できないため、実際の入力を変えるか仮想モジュールを使用してください。

監視とクロスリファレンス

監視テーブルは任意のタグをピン留めしてリアルタイム値を表示します。クロスリファレンスはそのタグを**読む**ルールと**書く**ルールを一覧し、「この値を動かしているのは誰か」に答えます。異常フィルタはタグ読み取り失敗または実行エラーのルールのみを表示します。条件が不成立なだけでは異常ではありません。

「ホールド」はエンジンを凍結し、出力は**現状を維持**します(無効化してゼロにするのとは異なります)。10 分間操作がないと自動復帰します。「ステップ」は 1 スキャンだけ進めます。「強制置位」は最大 16 点で、あらゆる書き込み元より優先され、再起動でクリアされます。その間もルールは評価され続けます。

シミュレーションモード

エンジン設定内のスイッチです。すべての物理出力・MQTT・警報を遮断しますが、`$VAR` **への書込みは通す**ため状態機械は進行します。その `$VAR` の値は実際に書き込まれており、シミュレーション終了時に元へ戻らない点に注意してください。

ルールによる自己監視

以下のシステム変数はルールに直接記述でき、診断を能動的な警報に変えられます:

変数	監視対象
`$SYS.WQ_FAIL`	キュー満杯で破棄された出力の累計
`$SYS.SLAVE_COUNT`	オンライン IO モジュール数。脱落時に警報
`$SYS.ETH_LINK`	イーサネットリンク
`$SYS.MQTT_CONNECTED`	MQTT 接続
`$SYS.CPU_LOAD`	CPU 負荷率

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

起動ストラテジ

エンジン設定の起動ルール選択メニューで設定します:

選択肢	動作
BOOT 優先(Safety をスキップ)	セーフティポリシーをスキップし `ON BOOT` ルールを直接実行(既定)
Safety 後に BOOT(先にゼロ化)	セーフティポリシーで全 DO をゼロ化し 1 秒待機後、BOOT ルールを実行
Safety のみ(BOOT を実行しない)	BOOT ルールを実行しない

`ON BOOT` ルールは通常のスキャンでは完全にスキップされます。起動時、エンジンのマスタースイッチがオフ→オンになったとき、置換モードで HRL をインポートしたとき、エンジン設定の手動 BOOT 実行ボタンを押したときに実行されます。

12 落とし穴一覧

#	落とし穴	対策
1	`PRIORITY` は実行順序を決めない	記述順で制御
2	ローカル IO は凍結スナップショットを読む。同一 tick の書込みは見えない	中間値は `$VAR` 経由
3	`THEN dst = $VAR.5` は黙って 0 を書く	`= $VAR.5 + 0` と書く
4	動作スロット 2/3 の operand は `$VAR.n` のみ	AI/$SYS のコピーは動作スロット 1 へ
5	`FOR` は合成条件に適用される	デバウンスは単一条件ルールに分離
6	`AND` は `OR` より優先されない	左結合。階層的上書きを使う

#	落とし穴	対策
7	`FOR` / `COOLDOWN` は 65535 ms 上限	長時間は `$SYS.UPTIME` か TIMER カウンタ
8	3 条件・3 動作を超えると黙って破棄	複数ルールに分割
9	`ON OFFDELAY FOR` が dwell を占有	2 ルールを連結
10	`PULSE <dur>` と `COOLDOWN` はフィールド共有	1 ルールにつきどちらか一方
11	認識できない行は黙って無視	インポート後にルール数と内容を照合
12	重複したルール名は最初の 1 件のみ	名前は一意に、23 バイト以内、ASCII 推奨
13	条件タグの読取失敗でエッジ記憶が消える	モジュール復帰時に偽の立ち上がりエッジを想定
14	cooldown 中でも条件の記憶は更新される	RISE エッジが失われる。チャタリング抑制には `FOR` を使う
15	約 3.5 KB を超えるインポートは黙って切り詰められる	コメントを除去し append で分割インポート

エクスポート／インポートの非対称性

機器からエクスポートした HRL テキストは**そのまま再インポートできる保証がありません**:

エクスポート内容	問題
`EVERY 30s`	パーサーは `IF EVERY` のみ受理。単独行は無視され TIMER ルールが LEVEL に退化
`DAILY AT 22:30`	同上。SCHEDULE が失われる
`THEN dst = operand`	COPY 形式は定数 0 として再解釈される
`AND ALARM "name"`	動作スロット 2/3 の severity が出力されない
ヒステリシス	まったく出力されない

> **結論**
>
> HRL ファイルをプロジェクトの
>
> **唯一の真実の源**
>
> として扱い、バージョン管理下に置いてください。「エクスポートして再インポート」の往復に依存してはいけません。エクスポートは目視確認専用です。

13 クイックリファレンス

```
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
```

本節を印刷すれば現場用クイックカードになります。詳細なセマンティクスは各章を参照してください。

泓洋電子科技有限公司　HOYAN Technologies Co., Ltd.
No.5, Ln. 33, Mingfeng 11th St., Qianzhen Dist., Kaohsiung City 80665, Taiwan
T +886 7 793-5098　F +886 7 793-5228　sc@hoyan.com.tw

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