ENGINEERING
TOMORROW

Application Guide

Logic feature

MyDrive® Insight

www.danfoss.com | iC7

TABLE OF CONTENTS

1 Introduction

1.1 Version History

This guide is regularly reviewed and updated. All suggestions for improvement are welcome. The original language of this guide is English (US).

Table 1 : Version History

Version	Automation Product Version	Industry Version	Motion Version	MyDrive® Insight
01	iC7_Automation-2024.5.43 (GR3)	4.2.9	3.1.9	2.15.0
02	iC7_Automation-2024.11.45 (GR4)	4.3.0	3.2.0	2.17.0

1.2 Purpose of this Application Guide

This application guide provides an overview of Logic and its integration into MyDrive® Insight. It covers the following topics:

- What is Logic and its purpose
- How to configure Logic using MyDrive® Insight
- Understanding the operation of function blocks within Logic
- Example configurations to illustrate the usage of Logic
- A comprehensive list of all available function blocks
- Error handling within Logic

1.3 Intended Audience

This application guide is intended for trained personnel, automation engineers, and configurators who have experience with parameter operations and possess fundamental knowledge of drives.

1.4 Additional Resources

Additional resources are available with related information:

- The iC7 Series Motion Application Guide and iC7 Series Industry Application Guide provide information about the Automation applications that support Logic.
- The MyDrive Insight Application Guide covers the general usage of MyDrive® Insight.

2 General

2.1 What is Logic?

Logic is a versatile feature that allows customization and control of the operation of the drive without the need for a separate programming tool or language. By utilizing Logic, the operation of the drive can be customized using a limited number of programmable function blocks and a limited number of states.

Logic in MyDrive® Insight extends the features of the drive and provides increased flexibility. Logic enables applying conditional controls, implementing fault detection and diagnostics, creating sequencing, modes, states, and interlocking logic.

Each function block has three inputs and one output, the functionality of these blocks can be selected from a comprehensive list. These function blocks are executed sequentially on every application cycle.

Any monitoring value or parameter can be connected to the block inputs. The output signal of each programmable function block can be used as an input to another function block or to control the digital or analog outputs of the drive. Moreover, the value of most parameters can be freely set with Logic. The drive can be directly controlled by the function block outputs through setting references and control signals.

Function blocks can initiate state changes; upon entry into a state, a user-defined list of actions (similar to function block outputs but with fixed values) executes. This allows flexible drive reconfiguration based on the selected state. Function blocks can be assigned to operate within specific states only.

Logic can be easily configured using the graphical configuration tool integrated into MyDrive Insight.

2.2 Why use Logic?

Logic can be used for a wide range of applications and purposes, providing enhanced flexibility and customization options. Here are some common use cases for Logic:

1. Conditional Controls: Logic allows for the implementation of conditional controls based on various inputs or parameters. Logic can adjust system behavior based on specific conditions, such as drive run time, external events, or other defined criteria.
2. Fault Detection and Diagnostics: Logic can be used to implement fault detection and diagnostics algorithms. By monitoring various parameters and inputs, logic can be created that detects abnormal conditions or faults in the system, enabling proactive maintenance and troubleshooting.
3. Conditional control modes: Logic can change the motor configuration, enabling multi motor functionality. Logic can provide a backup operation in case of service, fieldbus fault, and so on.

These are just a few examples of what Logic can be used for. The versatility and flexibility of Logic make it a powerful tool for implementing customized functionality and adapting the system's behavior to meet specific requirements.

2.3 Configuration

Logic can be configured inside MyDrive Insight. However, the feature is only accessible if the drive supports Logic and a connection to the drive has been established.

2.4 Running Mode

NOTICE

RUNNING MODE

Before utilizing Logic, it is important to evaluate whether the installation is in a suitable state for making changes to parameters, digital outputs, and analog outputs. Logic can be in the following modes:

- *Disabled*: Logic is not executed. Outputs and parameters are not affected by Logic.
- *Programming*: Logic is running in debug mode – function blocks and state handling is executed, but outputs and parameters are not changed by Logic.
- *Executing*: The outputs are actively driven and reflect the configured Logic behavior.

To configure Logic, switch to *Programming* mode (stopping execution); select *Executing* to activate the configuration.
Disable Logic when unnecessary to reduce processing load.
Internal data resets to defaults on mode changes and is not retained across power cycles.

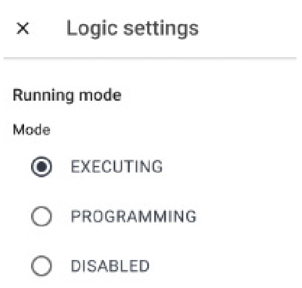


Illustration 1 : Running Mode Selection in the MyDrive® Insight GUI 2.17.0

2.5 Debugging

In Programming and Executing mode in the MyDrive® Insight GUI, it is possible to monitor the live values of block inputs and outputs.

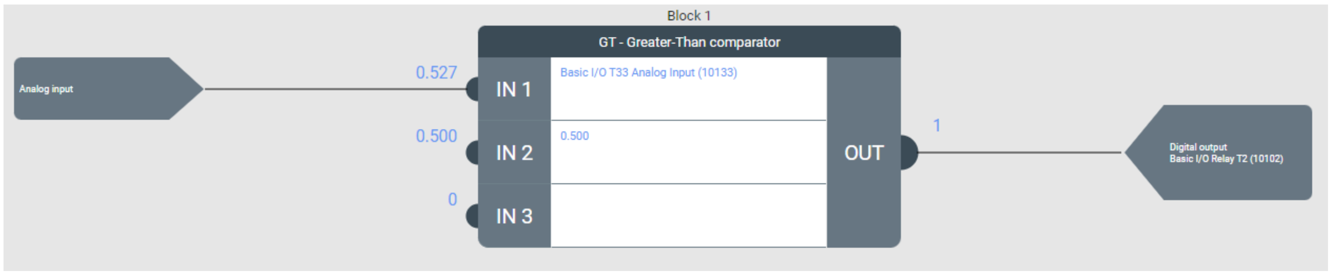

Block 1
GT - Greater-Than comparator
Analog input
0.527
0.500
0
IN 1
IN 2
IN 3
Basic I/O T33 Analog Input (10133)
0.500
OUT
1
Digital output
Basic I/O Relay T2 (10102)

Illustration 2 : Live debugging values shown on the inputs and outputs of a GT – Greater Than Comparator function block

2.6 Saving the Logic configuration

There is no need for a separate save command for the Logic configuration since it is automatically saved. It is saved as part of a backup and can then be restored. It is not possible to save or restore the Logic configuration separately from Parameter Settings.

2.7 Resource Limitations

The functionality of Logic is constrained by technical limitations and safeguards against processor and RAM overload.
For example, simultaneous parameter writes are limited to 30; if 15 function blocks are already writing parameters, only 15 additional parameters can be written in the state's On-entry actions. To write more parameters, use multiple states sequentially, splitting the parameter writes across them. Similar limits apply to digital and analog outputs.

Table 2 : Logic output limitations

Output resource	Global maximum limit
Number of function blocks	20
Number of states	5
Number of actions per state	20
Parameter write	30
Digital output write	10
Analog output write	10

NOTICE

PARAMETER SET LIMITATIONS

The parameters in the motor configuration and other hardware-related parameters cannot be adjusted while the motor is running. If the parameter cannot be written or if the parameter value does not meet the specified limits, a Logic Output Error warning is shown in the Events view of MyDrive Insight and the control panel.

Also, certain parameters are designated as ReadOnly and are not accessible for modification. These parameters are therefore not shown in the Logic ParamOut selection list.

2.8 Logic Virtual Terminals

Logic's virtual terminals store signals for use as inputs in other features. A function block output can write to a virtual terminal (for example, *Logic Digital I/O 1*), and that value can be read as an input elsewhere (for example, by setting parameter *4722 Advanced Start Input* as *Logic Digital I/O 1*). These virtual terminals function as boolean buffer parameters. The Logic virtual digital terminals can be selected wherever virtual terminals are available. For a detailed use case example, refer to Section 6.1 *Start based on Analog input T33*.

Table 3 : Logic virtual terminals

Terminal name	Description
Logic Digital I/O 1	Logic virtual digital I/O terminal 1.
Logic Digital I/O 2	Logic virtual digital I/O terminal 2.
Logic Digital I/O 3	Logic virtual digital I/O terminal 3.
Logic Digital I/O 4	Logic virtual digital I/O terminal 4.

2.9 Logic References

Logic includes reference parameters for each available control mode.
The Logic reference can be selected as a source in the control place configuration.
The Logic references are handled as parameters, and therefore count as a parameter write and are included in the maximum number of parameters written by Logic.
The specific references that are available are application dependent, for example, Logic position reference is available in Motion applications.

Table 4 : Logic References

Reference name	Description
Logic speed reference	provides a method for setting a speed reference directly from Logic.
Logic torque reference	Provides a method for setting a torque reference directly from Logic.
Logic process reference	Provides a method for setting a process reference directly from Logic.
Logic position reference	Provides a method for setting a position reference directly from Logic

3 Function blocks

Each function block in Logic can be configured by selecting the appropriate Function Block Type with the selection *Function*. This selector provides a wide range of function blocks such as AND, OR, MUL, DIV, EQ, GT, and more.

Every function block consists of three inputs and one output. Each input and output can be configured individually to meet the specific requirements of the application. The default value of all *Input Modes* is set to *Not Used*. If left unchanged, the input is treated as 0.0 (FALSE).

Each function block has mandatory inputs that must be configured. If optional inputs are set to *Not Used,* they are ignored. For details see the description of the individual function blocks in chapter *3.1 Functions*. A warning is issued if a required input is set to *Input Mode* = *Not Used*. For more information on error handling, refer to the Error Handling section.

3.1 Functions

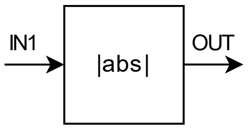
The functions available in the Logic blocks can be categorized into four groups:

4. Logic and Bit Operations: These functions provide boolean operators for common boolean algebra. They are used to perform logical operations on boolean signals. In logic operations, all inputs are converted to booleans. A numeric value of 0.0 is converted to boolean FALSE, while all other values are treated as boolean TRUE.
5. Math Operations: These functions provide numeric operators for elementary arithmetic operations. They are used to perform mathematical calculations on numeric values. Arithmetic operations are working with numeric values. All inputs are converted to numeric values.
6. Comparators: These functions provide comparative logic for numeric values. They are used to compare two values and determine their relationship, such as equality, inequality, or order. All inputs are converted to numerics. By default, the optional tolerance value is 10^-6, if the input is set to *Not Used*. The output is a boolean that can be used for boolean operators.
7. Special Operators: These functions can combine logic and arithmetic operations. They are used for advanced or specialized operations. The type of the output depends on the operator.

Table 5 : List of available function blocks

Logic operations	Math operations	Comparators	Special operations
AND – Logical AND	ABS – Numeric absolute	EQ – Equal comparator	CTUD – Rising edge up/down counter
F_TRIG – Falling edge flank trigger	ADD – Numeric sum	GE – Greater-or-Equal comparator	DELAY – Logical delay
NAND – Logical Not-AND	DIV – Numeric divide	GT – Greater-Than comparator	SEL – Selector/Relay
NOR – Logical Not_OR	FILTER – Low-pass filter	GT_LT – Combined GT and LT comparator	TIMER_ACUM – Accumulative timer
NOT – Logical NOT	LIMIT – Numeric limiter	LE – Less-or-Equal comparator	TIMER_EVENT – Event based timer
OR – Logical OR	MAX – Maximum value	LT – Less-Than comparator	
R_TRIG – Rising edge flank trigger	MEAN – Average value of used inputs	NE – Not-Equal comparator	
RS – Reset Set Flipflop	MIN – Minimum value		
SR – Set Reset Flipflop	MODULUS – Remainder of integer division		
XOR – Logical Exclusive-OR	MUL – Numeric multiply		
	MAX – Maximum value		
	MEAN – Average value of used inputs		
	MIN – Minimum value		
	MODULUS – Remainder of integer division		
	MUL – Numeric multiply		

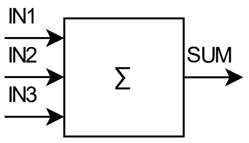
ABS – Numeric absolute	
IN1	OUT = | IN1 |
N/A	
N/A	



Description:
Absolute value function - removes the sign from a Input1.

$$OUT = |IN1| = \begin{cases} IN1, if\ IN1 \geq 0 \\ -IN1, if\ IN1 < 0 \end{cases}$$

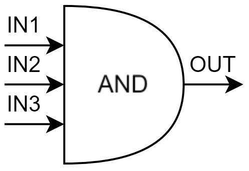
ADD – Numeric sum	
IN1	OUT = IN1 + IN2 + IN3
IN2	
IN3	



Description:
The output of the addition block is the sum of the 3 inputs. The third input is optional.

$$OUT = IN1 + IN2 + IN3$$

AND – Logical AND	
IN1	OUT = IN1 && IN2 && IN3
IN2	
IN3	



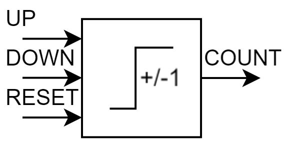
Description:
Logical AND Gate, the third input is optional. Output is TRUE (1) if all input signals are TRUE (IN X ≠ 0).

Table 6: AND Truth Table

IN1	IN2	IN3	OUT
0	0	0	0
0	1	0	0
1	0	0	0
1	1	0	0
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	1

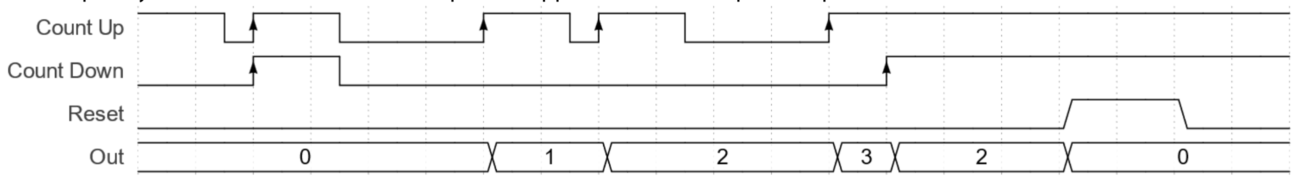
CTUD – Rising-edge up-down counter

COUNT UP	OUT = Nr. of RE(IN1) – Nr. of RE(IN2)
COUNT DOWN	
RESET	



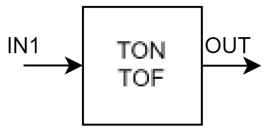
Description:
This function increments the output value by one for every rising edge on Input1 and decreases the output by one for every rising edge on Input2. The counter output is reset to 0 while the reset signal is TRUE and starts from 0 again afterwards. Either IN1 or IN2 must be used, IN3 is optional.

Example:
An up-down counter controls the speed of a conveyor belt. Count Up is triggered by a sensor detecting an item needing transport, increasing the belt speed. Count Down is triggered when the item reaches its destination, slowing the belt. Reset is used to stop the belt completely. The counter value on the output is mapped to a suitable speed setpoint for the drive.



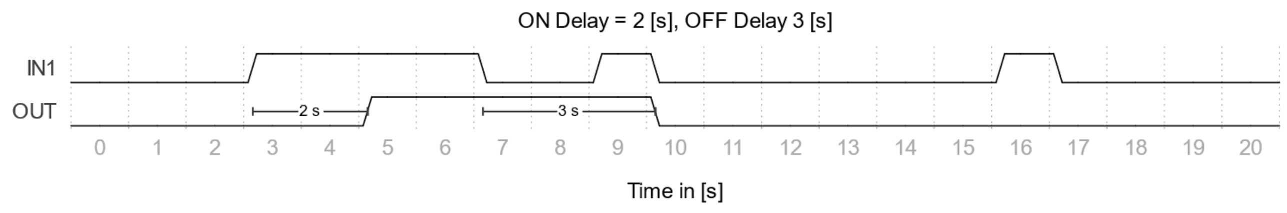
DELAY – Logical delay

IN1	OUT = Delayed IN1 transitions
ON DELAY [s]	
OFF DELAY [s]	

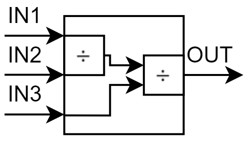


Description:
The function implements a timer with a configurable turn-on and turn-off delay. Input1 and at least one of the delays must be configured. Input1 and the Output are treated as booleans, while the delays in seconds are decimals.

Example:
Using a delay function to implement a soft-start/soft-stop sequence for a pump, preventing sudden starts and stops. When configured with a turn-on delay of, for example, 2 seconds and a turn-off delay of, for example, 3 seconds, the output follows the input with a delayed response on both rising- and falling-edge transitions as shown in the following figure.

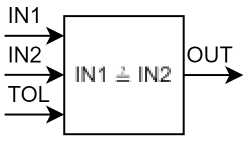


DIV– Numeric divide	
IN1	OUT = (IN1 / IN2) / IN3
IN2	
IN3	



Description:
Input1 divided with Input2 (optionally divided again with Input3).

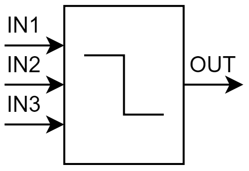
EQ – Equal comparator	
IN1	OUT = IN1 $\overset{?}{=}$ IN2 (+/-Tolerance)
IN2	
TOLERANCE	



Description:
Equal function. Output = Input1 == Input2, Input3(optional) = tolerance. By default, an optional tolerance value is 10^-6, if not configured.

$$OUT = \begin{cases} TRUE(1), if\ IN1 == IN2 \\ FALSE(0), else \end{cases}$$

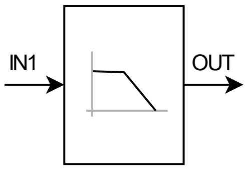
F_TRIG – Falling edge flank trigger	
IN1	OUT = F_TRIG(IN1) || F_TRIG(IN2) || F_TRIG(IN3)
IN2	
IN3	



Description:
The falling edge trigger block can detect a falling edge in a boolean signal and switch its output from FALSE to TRUE. The output stays active for one execution cycle (application dependent, for example Industry: 5 ms, Motion: 10 ms) when a falling edge is detected. At least one of the inputs must be configured.

Example:
Control a directional valve. Detect a falling edge on Input1, for example, a digital input connected to a limit switch. Connect the function block's output to the SET input of a RS-flipflop function block to store the desired valve state.

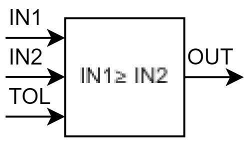
FILTER – Low-pass filter	
IN1	OUT = LowPass(IN1)
Filter time [s]	
N/A	



Description:
First-order low-pass filter function. Output = LowPass(Input1), Filter time = Input2 [s]. The Execution cycle is application dependent: Industry: 5 ms, Motion: 10 ms.

$$OUT = PrevOut + (IN1 - PrevOut) * \frac{Execution\ cycle}{MAX(IN2, Execution\ cycle)}, \qquad initially\ PrevOut = IN1$$

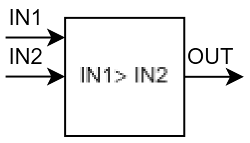
GE – Greater-or-Equal comparator	
IN1	OUT = IN1≥IN2 (-Tolerance)
IN2	
TOLERANCE	



Description:
Greater or Equal function. Output = Input1 >= Input2, Input3(optional) = tolerance. By default, the optional tolerance value is 10^-6, if not configured.

$$OUT = \begin{cases} TRUE(1), if\ IN1 \geq \text{IN2} \\ FALSE(0), else \end{cases}$$

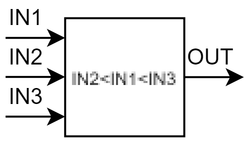
GT – Greater-Than comparator	
IN1	OUT = IN1> IN2
IN2	
N/A	



Description:
Greater Than function. Output = Input1 > Input2.

$$OUT = \begin{cases} TRUE(1), if\ IN1 > \text{IN2} \\ FALSE(0), else \end{cases}$$

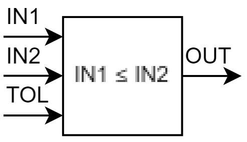
GT_LT – Combined GT and LT comparator	
IN1	OUT = IN2 < IN1 < IN3
IN2	
IN3	



Description:
Between but not equal to limits. Output = Input2 < Input1 < Input3.

$$OUT = \begin{cases} TRUE(1), if\ IN2 < \text{IN1} < IN3 \\ FALSE(0), else \end{cases}$$

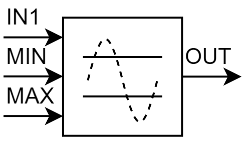
LE – Less-or-Equal comparator	
IN1	OUT = IN1 ≤ IN2 (+Tolerance)
IN2	
TOL	



Description:
Less or Equal function. Output = Input1 ≤ Input2, Input3(optional) = tolerance. By default, the optional tolerance value is 10^-6, if not configured.

$$OUT = \begin{cases} TRUE(1), if\ \mathrm{IN1} \leq IN2 \\ FALSE(0), else \end{cases}$$

LIMIT – Numeric limiter	
IN1	OUT = MIN(MAX(IN1, IN2), IN3).
MIN	
MAX	

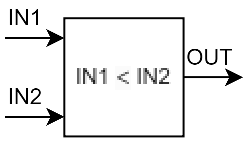


Description:
Limit function. Input2 must be smaller than Input3. Output = MIN(MAX(Input1, Input2), Input3).

$$OUT = \begin{cases} IN1, if\ IN2 < \mathrm{IN1} < IN3 \\ IN2, if\ IN1 < IN2 \\ IN3, if\ IN1 > IN3 \end{cases}$$

Example:
A limit function can dynamically constrain a process controller's torque reference (Input1), preventing motor stall (by limiting torque below Input2) and overload (by limiting torque above Input3). Unlike static torque limits set through application parameters such as *Positive* and *Negative Torque Limit*, this approach offers flexible, real-time control by allowing Input2 and Input3 to vary dynamically. The function's output can directly set the torque reference of the motor.

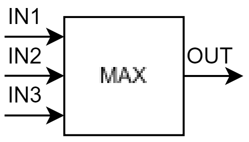
LT – Less-Than comparator	
IN1	OUT = IN1<IN2
IN2	
N/A	



Description:
Less Than function. Output = Input1 < Input2.

$$OUT = \begin{cases} TRUE(1), if\ \mathrm{IN1} < IN2 \\ FALSE(0), else \end{cases}$$

MAX – Maximum value	
IN1	OUT= MAX(MAX(IN1, IN2), IN3).
IN2	
IN3	

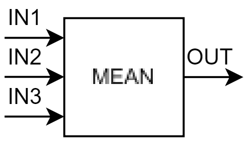


Description:
Maximum function. Returns the largest of the configured inputs. Input3 is optional.

$$OUT = \mathrm{MAX}(\ \mathrm{MAX}(\mathrm{IN1}, \mathrm{IN2}), \mathrm{IN3}).$$

Example:
Monitor different temperature sensors connected to Input1-3 and return the highest temperature to another function block that checks with a Greater Than-function, whether a temperature limit is exceeded.

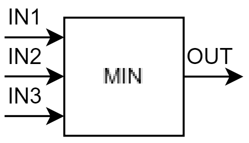
MEAN – Average value of used inputs	
IN1	OUT= (IN1 +IN2+ IN3)/ (nr. of inputs)
IN2	
IN3	



Description:
Mean function. Returns the average value of the configured inputs. At least one of the inputs must be configured.

$$OUT = \frac{\sum IN1, IN2, IN3}{nr.of\ configured\ inputs}$$

MIN – Minimum value	
IN1	OUT= MIN(MIN(IN1,IN2),IN3)
IN2	
IN3	

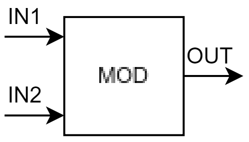


Description:
Returns the smallest of the configured inputs. Input3 is optional.

$$OUT = \mathrm{MIN}(\mathrm{MIN}(\mathrm{IN1}, \mathrm{IN2}), \mathrm{IN3})$$

Example:
Monitor different set points connected to Input1-3 and return the lowest value.

MODULUS – Remainder of integer division	
IN1	OUT= MOD(IN1, IN2)
IN2	
N/A	

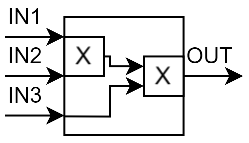


Description:
Integer modulus division function. This arithmetic function divides the operand connected to Iinput1 by the operand connected to Input2 and returns the remainder of the division. Be aware that the modulus operation treats the inputs as double integers and returns a double integer value.

$$OUT = \text{DINT(IN1) MOD DINT(IN2)}$$

Example:
When controlling a rotary indexing table with a limited number of positions, the modulus function can determine the current position. Input1: A counter representing the total number of steps the table has rotated. This counter could increment with each step motor pulse. Input2: The number of positions on the indexing table (for example, 12 positions). The modulus function (Input1 modulo Input2) returns the remainder. This remainder represents the current position on the table (0 to 11 in this example), providing a value that wraps around after reaching the maximum number of positions.

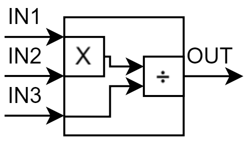
MUL – Numeric multiply	
IN1	OUT= IN1xIN2xIN3
IN2	
IN3	



Description:
Input1 multiplied with Input2 and Input3. Input3 is optional.

$$OUT = \text{IN1} * \text{IN2} * \text{IN3}$$

MULDIV – Comb. numeric multiply and divide	
IN1	OUT= (IN1xIN2)/IN3
IN2	
IN3	



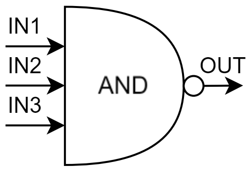
Description:
Combined multiply and divide function. Output = Input1 x Input2 / Input3. Input3 is optional.

$$OUT = \frac{\text{IN1} * \text{IN2}}{IN3}$$

Example:
Calculating the required motor speed of the driving pulley based on the desired driven pulley speed and pulley diameters:
Input1: Desired speed of the driven pulley [RPM]
Input2: Diameter of the driven pulley [m]
Input3: Diameter of the driving pulley [m]

NAND – Logical Not-AND

IN1	OUT = NOT(IN1 && IN2 && IN3)
IN2	
IN3	

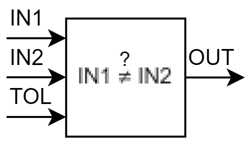


Description:
Inverted AND block. Output is FALSE (OUT = 0) if all inputs signals are TRUE. The third input is optional. It has the following truth table:

Table 7: NAND Truth Table

IN1	IN2	IN3	OUT
0	0	0	1
0	1	0	1
1	0	0	1
1	1	0	1
0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	0

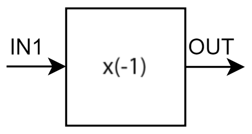
NE – Not-Equal comparator	
IN1	OUT = IN1 $\overset{?}{\neq}$ IN2 (+/-Tolerance)
IN2	
TOLERANCE	



Description:
Not-Equal function. Output = Input1 ≠Input2, Input3(optional) = tolerance. By default, the optional tolerance value is 10^-6, if Input3 is set to *Not Used*.

$$OUT = \begin{cases} TRUE(1), if\ IN1 \neq IN2 \\ FALSE(0), else \end{cases}$$

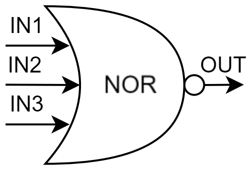
NEG – Numeric negate	
IN1	OUT = (-1)*IN1
N/A	
N/A	



Description:
Negated value if input1

$$OUT = (-1) * IN1$$

NOR – Logical Not OR	
IN1	OUT = NOR(IN1,IN2,IN3)
IN2	
IN3	



Description:
Logical NOR-function. Output = Input1 NOR Input2 NOR Input3(optional). Inverted OR block. Output is TRUE (OUT = 1) if all inputs signals are FALSE. It has the following truth table:

Table 8: NOR Truth Table

IN1	IN2	IN3	OUT
0	0	0	1
0	1	0	0
1	0	0	0
1	1	0	0
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	0

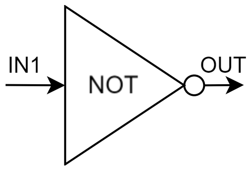
NORMALIZE – Scale for analog output	
IN1	OUT = RESCALE(IN1, MIN, MAX)
MIN	
MAX	

Description:
Normalize function. Scale Input1 between Input2 (min) and Input3 (max). Input2 must be smaller than Input3. Output is limited between 0.0 and 1.0.

$$OUT = ((\mathrm{MIN}(\mathrm{MAX}(\mathrm{IN1}, \mathrm{IN2}), \mathrm{IN3})) - \mathrm{IN2}) / (\mathrm{IN3} - \mathrm{IN2})$$

Example:
Dynamically rescale Analog Input T33 between T34 (equivalent to 0%) and 10 V (100%).
Input1: *T33 Analog input Value*, Parameter Nr. 1611
Input2: *T34 Analog Input Value*, Parameter Nr. 1612
Input3: *T33 Maximum Value*, Parameter Nr. 2271
Output: Percentage of Input1 relative to Input2(min) and Input3(max).
For example: 1611= 5.0 V, 1612=2.5 V, IN3=10 V
Out = 0.333

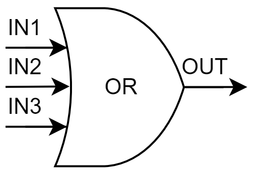
NOT – Logical NOT	
IN1	OUT = NOT(IN1)
N/A	
N/A	



Description:
Boolean complement function. Changes TRUE to FALSE and FALSE to TRUE

$$OUT = \begin{cases} TRUE(1), if\ IN1 = FALSE(0) .. \\ FALSE(0), if\ IN1 \neq FALSE\ (0) \end{cases}$$

OR – Logical OR	
IN1	OUT = OR(IN1, IN2, IN3)
IN2	
IN3	

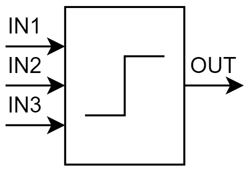


Description:
OR-Gate function. Output is TRUE(1) if one or more of the input signals are TRUE (IN X ≠ 0). Input3 is optional.

Table 9: OR Truth Table

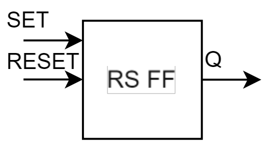
IN1	IN2	IN3	OUT
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	1
0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	1

R_TRIG – Rising edge flank trigger	
IN1	OUT = R_TRIG(IN1) || R_TRIG(IN2) || R_TRIG(IN3)
IN2	
IN3	

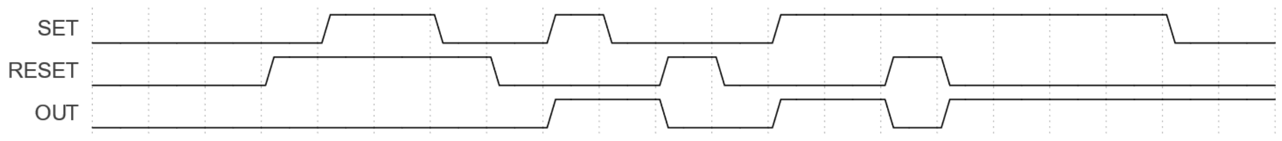


Description:
The rising edge trigger block can detect a rising edge in a boolean signal and switches its output from FALSE to TRUE. The output stays active for one execution cycle (application dependent, for example Industry: 5 ms, Motion: 10 ms) when a rising edge is detected. At least one of the inputs must be configured.

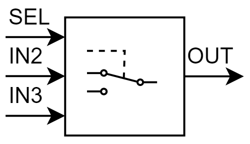
RS – Reset Set Flipflop	
SET	OUT =Q
RESET	
N/A	



Description:
The output is reset (OUT=0) if the RESET input is TRUE (≠0), regardless of the state of the SET input. If the SET input is TRUE (≠0) and RESET is FALSE (=0), the output pin is set (OUT=1). If both the inputs are FALSE (=0), the output preserves its previous value.
Example:
Manage a two-position system (on/off). Input1 giving a start and Input2 a stop signal.



SEL – Selector/Relay	
SEL	OUT =SEL(IN1, IN2, IN3)
IF_FALSE	
IF_TRUE	

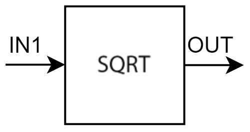


Description:
Select/relay function. Input2 and Input3 can be either numeric or boolean values. Output returns Input2 if Input1 is FALSE (0), else

$$OUT = \begin{cases} IN2, if\ IN1 = FALSE(0) \\ IN3, if\ IN1 \neq FALSE(0) \end{cases}$$

Example:
A select/relay function can select between two temperature setpoints (Input2 and Input3) based on a mode selection Input1.

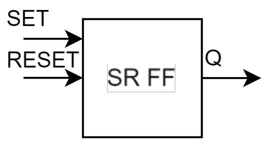
SQRT – Numeric square root	
IN1	OUT =SQRT(IN1)
N/A	
N/A	



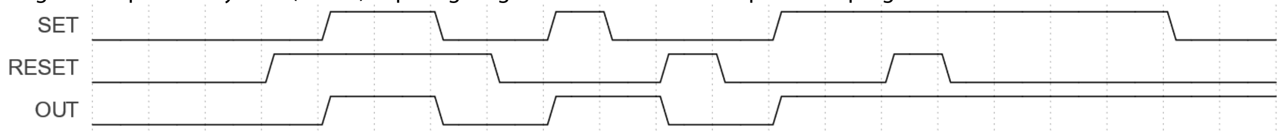
Description:
This numerical function calculates the square root of Input1.

$$OUT = \sqrt{IN1}$$

SR – Set Reset Flipflop	
SET	OUT =Q
RESET	
N/A	

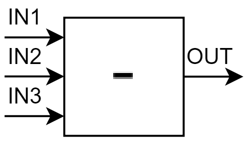


Description:
The output is set (OUT=1) if the SET input is TRUE (≠0), regardless of the state of the RESET input. If the RESET input is TRUE and SET is FALSE (=0), the output pin is cleared (OUT=0). If both the inputs are FALSE, the output preserves its previous value.
Example:
Manage a two-position system (on/off). Input1 giving a dominant start and Input2 a stop signal.



SUB – Numeric subtract

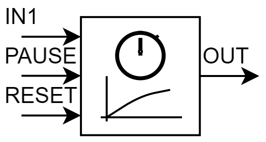
IN1	OUT =IN1-(IN2+IN3)
IN2	
IN3	



Description:
The subtract function subtracts Input2 and Input3 from Input1. Input3 is optional.
$OUT = IN1 - (IN2 + IN3)$

TIMER_ACCUM – Accumulative timer

IN1	OUT =Time Accum(IN1)
PAUSE	
RESET	

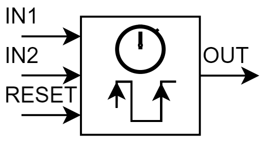


Description:
Accumulative Timer [s]. Accumulating time of high signal on Input1. If Input1=TRUE the timer starts/continues, if Input2(optional)=TRUE the timer pauses, if Input3 (optional)=TRUE the timer resets to 0. The output is the timer value [s] which is updated on every execution cycle. Input3 is a priority reset.

Example:
In a system monitoring the operational time of a piece of equipment, an accumulative timer could track the total uptime.
Input1: A Boolean signal indicating whether the equipment is running (TRUE = running, FALSE = stopped).
Input2 (optional): A Boolean signal to pause the timer (for example, during planned maintenance).
Input3 (optional): A Boolean signal to reset the timer to zero (for example, after a major service).
Output: The total accumulated runtime in seconds.
This provides a record of the total operational time of the equipment, which can be used for maintenance scheduling.

TIMER_EVENT – Event based timer

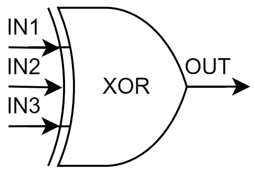
IN1	OUT =Time RS(IN1)<->RS(IN1) || RS(IN2)
IN2	
RESET	



Description:
Event-based Timer [s].
Measuring time between rising flanks on input 1 and 2 OR time between recurring flanks on input 1 only. A rising edge on Input1 updates the output and (re)starts the timer. A rising edge on Input2 updates the output and freezes the timer. A high signal on Input3(optional) stops the timer and resets the output and timer value

Example:
In a production line monitoring system, an event-based timer could measure the cycle time of a machine.
Input1: A sensor signal indicating the start of a machine cycle (rising edge).
Input2: A sensor signal indicating the end of a machine cycle (rising edge).
Input3 (optional): A signal to stop the timer (for example, during maintenance).
Output: The measured cycle time in seconds.

XOR – Logical Exclusive-OR

IN1	OUT = XOR(IN1, IN2, IN3)	
IN2		
IN3		

Description:
XOR-Gate function. Output is TRUE (1) if only one of the input signals are TRUE (IN X ≠ 0). Input3 is optional.

Table 10: XOR Truth Table

IN1	IN2	IN3	OUT
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0
0	0	1	1
0	1	1	0
1	0	1	0
1	1	1	0

3.2 Data Types

In Logic, all signals and values are internally handled as floating-point values. However, some of the selectable functions have inputs or outputs defined as boolean values (BOOL), such as AND, OR, RS, and others. These boolean values can have two different states: TRUE or FALSE.

With boolean values, the following conversion rule applies:

- If the input or output is not equal to 0.0, it is considered TRUE.
- If the input or output is equal to 0.0, it is considered FALSE.

For example, if a value of 0.534 is routed to a digital output, the digital output is active because it is interpreted as TRUE.

Similarly, in the example shown in Illustration 3, if an analog input with a value of 0.497 is routed to an OR function, the result is TRUE. Only when the analog input is precisely 0, is it interpreted as FALSE. Therefore, it is often not a good idea to use an analog input as input to a Boolean function block operation.

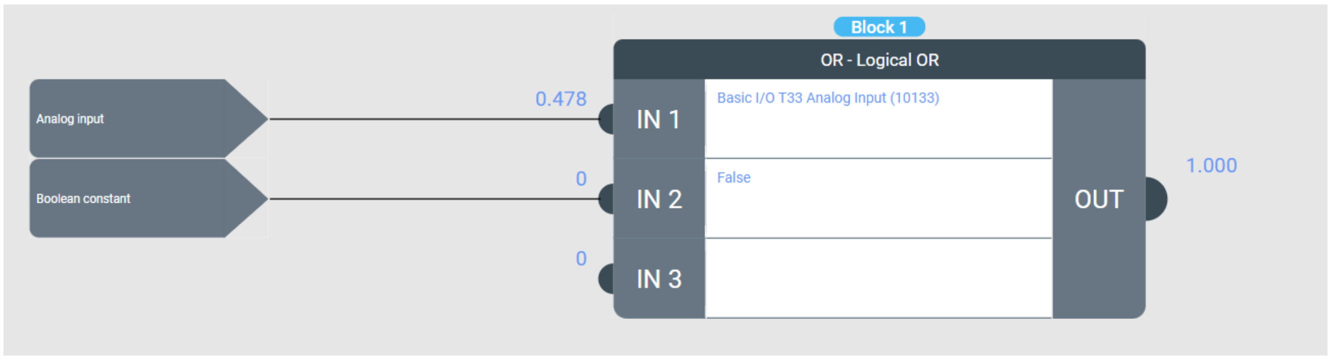


Illustration 3 : Data types

3.3 Block Inputs

In MyDrive® Insight, each input (IN1, IN2, and IN3) has a configuration selection. Click *Input Mode* to select the input signal. Depending on the selected input mode, additional configuration options such as Input Value, Bit/Index, and negate/invert become visible.

It is also possible to use the output of one block as the input to another block, enabling the creation of more complex logic configurations.

Table 11 : Input Modes

Selection name	Description	Additional information
Not used	Functionality is disabled.	Input does not fetch any values from any

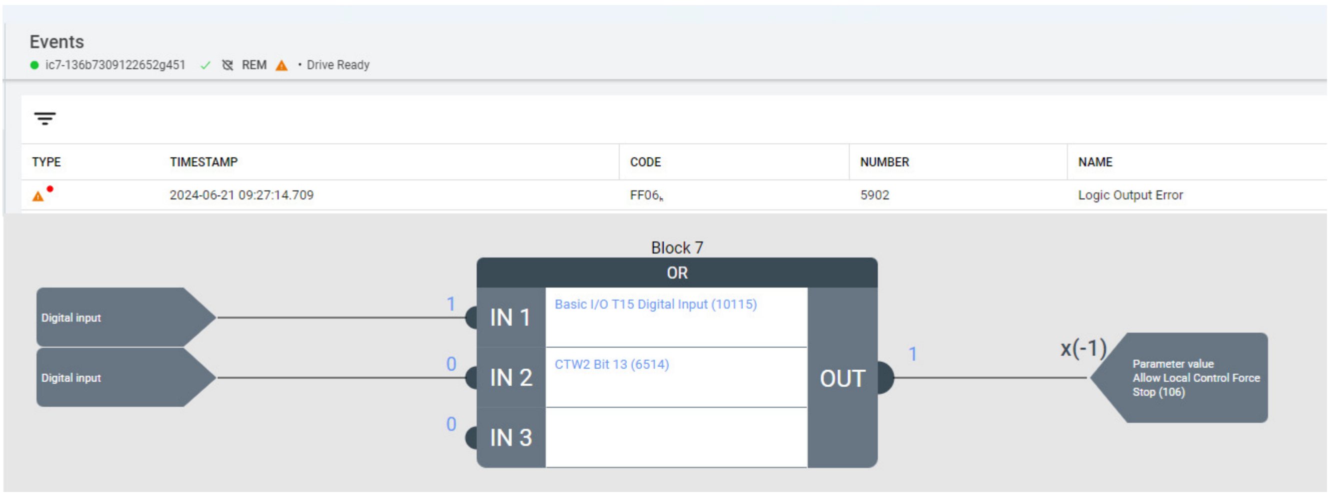
Selection name	Description	Additional information
		source. It returns 0.0 (FALSE). Input is treated as not configured. If the input is required for an operator, leaving it not configured triggers a Logic Block Configuration Error Event.
Digital input	Read the state of a digital input.	The input mode Digital input gives access to both physical inputs and virtual digital inputs, such as user-defined control word bits.
Parameter bit	Provides a method for fetching a specific bit from a word-type parameter value. The LSB has bit number 0.	
Boolean constant	Provides a method for setting a boolean value.	Set an input to constant TRUE or FALSE
Event active	Provides a method for reading if an event is active. Returns TRUE=1.0, if an event is active.	All events can be selected based on the event number in decimal form or the event name.
Event group active	Provides a method for reading if any event in an event group is active. Returns TRUE=1.0, if an event is active.	All event groups can be selected based on the event group number in hexadecimal form or the event name.
Analog input	Reads the value of an analog input terminal, returning a normalized value between 0.0 and 1.0.	Analog input returns the analog input value scaled between 0.0 and 1.0 and not the value in physical units. Using *Parameter Value* instead retrieves the analog input status.
Parameter value	Provides a method for fetching a parameter value.	
Numeric constant	Provides a method for inputting a numeric constant.	Use 0.0 for FALSE and 1.0 for TRUE if a boolean value is required, or use a Boolean constant input instead.
State	Provides a method for fetching current state or previous state.	This is useful in state handling, to enable or disable other parts of Logic, prepending on state. Value 0, means 'No State' equal state handling not used.
Time in Current State	Provides a method for fetching how long a Logic state has been active.	This is useful for creating timed states.
Block output	Output value from a selected block is passed on as the input value.	This makes it possible to link the output of a block to the input of another block.

3.4 Block Outputs

In MyDrive® Insight, the output (OUT) of a function block can be configured by clicking the configuration field. The output signal, and other configuration options, can be defined based on the selected output mode. These options include Output Value, Bit/Index, and negate/invert.

Some of the output modes are limited to a fixed number of simultaneous instances. For example, a maximum of 10 different digital input terminals can be accessed. See the additional information column in Table 12: Output Modes.

Using negation for output mode *Parameter value* means multiplying the value by -1. When setting a Boolean type parameter, this may not provide the expected result. Writing an invalid value, such as -1, to a boolean parameter that only accepts 0 or 1 results in a Logic Output Error Warning that can be seen in the MyDrive Insight Events view.



Events

ic7-136b7309122652g451 ✓ REM ▲ • Drive Ready

TYPE	TIMESTAMP	CODE	NUMBER	NAME
▲	2024-06-21 09:27:14.709	FF06	5902	Logic Output Error

Illustration 4: Logic output error event because -1 is not a valid choice for parameter number 106

Table 12 : Output Modes

Selection name	Description	Additional information
Not used	Block does not apply result anywhere.	
Digital output	Block output is applied to the selected digital output terminal.	Digital output is a limited resource, with a limited number of instances.
Analog output	Block output is applied to the selected analog output terminal.	Analog output is a limited resource, with a limited number of instances. The analog output is limited to values between 0.0 and 1.0 as it is giving the internal reference for the analog output. The value is then converted to physical units based on the analog output mode and min. and max. value configurations in the application parameters.
Parameter value	Block output is applied to the selected parameter.	Parameter write is a limited resource, with a limited number of instances. ReadOnly parameters cannot be written. Parameter values must be within the parameter's limits and a valid selection when limited by a selection list. Otherwise, a Logic Block Output Error event is triggered.
Logic State Request	Block output is used to request a state change.	Logic State request triggers a state change if the output is TRUE. If the state does not allow activation while running and the drive is running, the state change request is ignored.
Logic speed reference	Block output is applied to the Logic speed reference.	Provides a method for setting a speed reference directly from Logic. The Logic reference can be selected as a source in the control place configuration. Speed reference counts as a parameter write.
Logic torque reference	Block output is applied to the Logic torque reference.	Provides a method for setting a torque reference directly from Logic. The Logic reference can be selected as a source in the control place configuration. Torque reference counts as a parameter write.
Logic process reference	Block output is applied to the Logic process reference.	Provides a method for setting a process reference directly from Logic. The Logic

Selection name	Description	Additional information
		reference can be selected as a source in the control place configuration. Process reference counts as a parameter write.
Logic position reference	Block output is applied to the Logic position reference.	Provides a method for setting a position reference directly from Logic. The Logic reference can be selected as a source in the control place configuration. Position reference counts as a parameter write.

4 State Handling

Many applications require the drive to operate in modes beyond its primary function.
This can be for service, where limits are different, or parts of normal protection are disabled.
It can be a backup operation in case of fieldbus loss or other failures.
It can be for changing entire motor configurations on an application where one drive runs multiple motors, one at a time.

State handling can also be used as an active part of operation. This can be running states as a sequence based on time and/or other conditions.

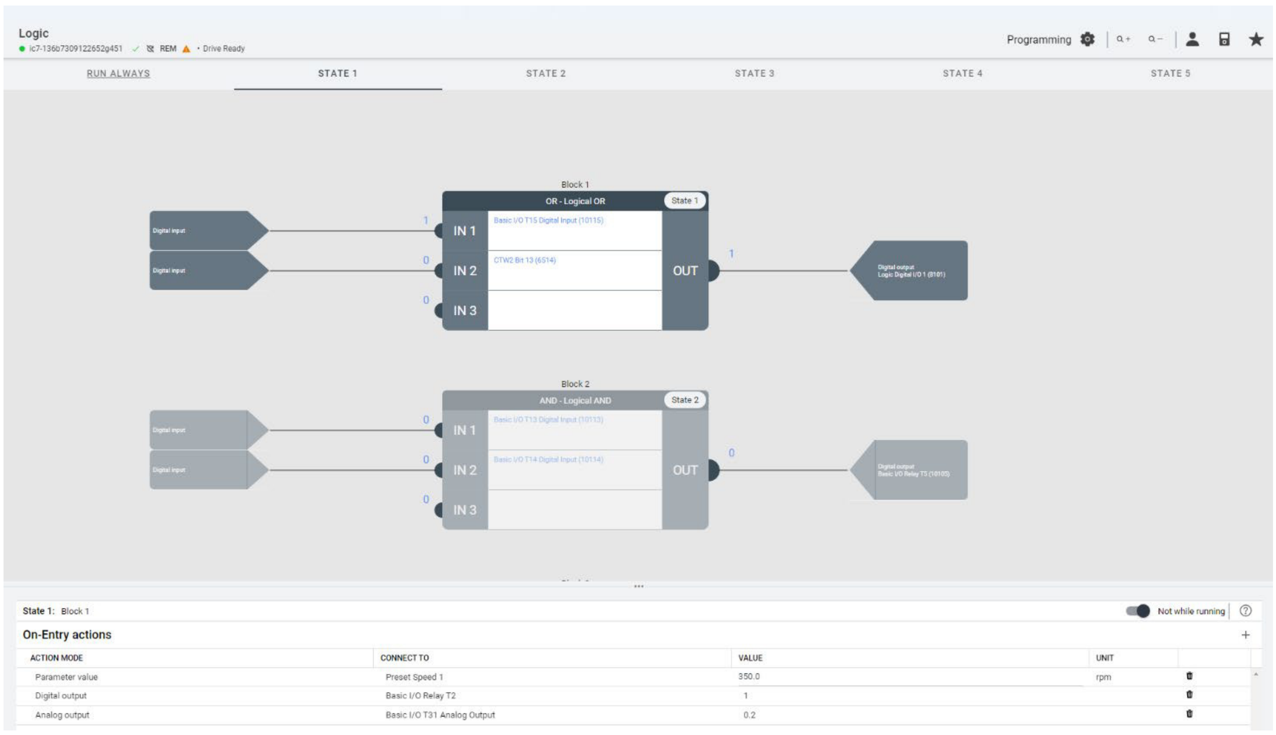
It is optional to use states alongside function blocks in Logic. Blocks that are not assigned to run in a state will always be executed.

4.1 State Definition

Logic can be configured to include a state machine with mutually exclusive states.
State transitions occur in two phases:

1. execution of an optional On-entry action list (for example, parameter writes, output settings), and
2. continuous execution of assigned function blocks.

Function blocks can request state changes based on various conditions. A state remains active until a new state is requested.



State 1: Block 1

On-Entry actions

ACTION MODE	CONNECT TO	VALUE	UNIT
Parameter value	Preset Speed 1	350.0	rpm
Digital output	Basic I/O Relay T2	1	
Analog output	Basic I/O T31 Analog Output	0.2	

Illustration 5: Logic State 1 being edited. Function Block 1 is assigned to State 1 and only executes when State 1 is active.

4.2 State Change

State changes are initiated by a function block requesting a state transition.
This request, available as a block output mode, specifies the target state. A true output triggers the state change. If multiple simultaneous requests occur, the request from the highest-numbered block takes precedence.

State information is available as function block inputs.
The input modes:

- State: Reads the requested state information - current state or previous state as a numeric value:
  1 = 'State 1', 2 = 'State 2' … and 0 = 'No State' (no state active)
- Time in State: Provides a method to get the time spent in the current state in seconds with the same resolution as the application (Industry = 5 ms)

In the following example, a rising flank on Basic I/O terminal 14 is used to request the activation of State 2.

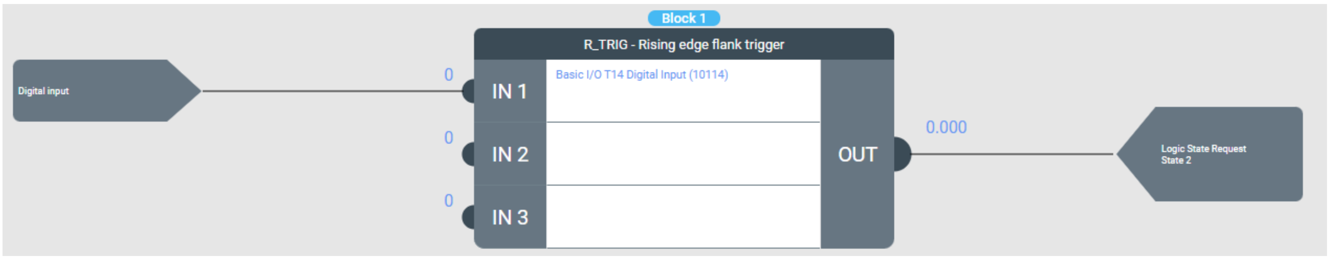
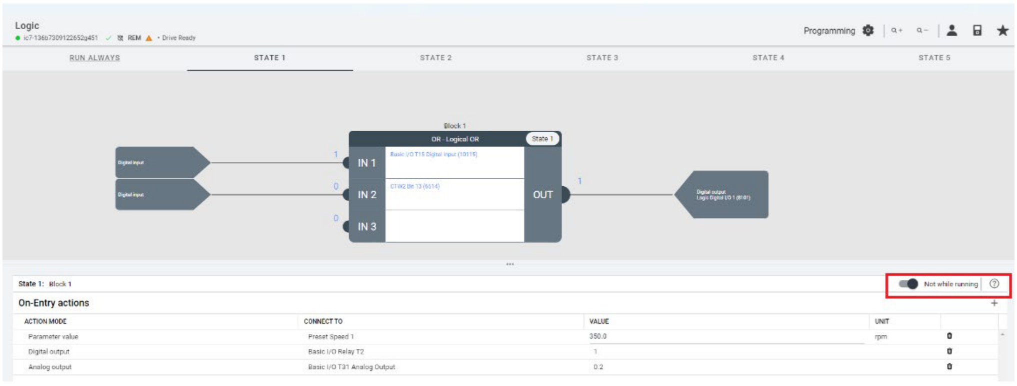

Block 1
R_TRIG - Rising edge flank trigger
Digital input
IN 1
IN 2
IN 3
Basic I/O T14 Digital Input (10114)
OUT
0.000
Logic State Request State 2

Illustration 6: Example of state change request.

As function blocks can be assigned to only run within a specific state, it can be useful to place the state change logic within the states.

Each state can be configured to prevent activation while the motor is running. Since Logic can reconfigure the drive, including motor control parameters, preventing state activation during motor operation is recommended when motor parameters are to be changed in a state.



State 1: Block 1

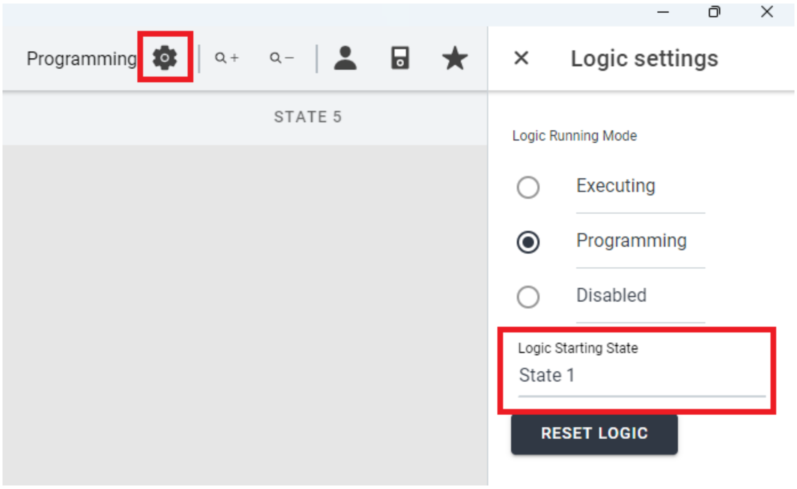
Not while running

On-Entry actions

ACTION MODE	CONNECT TO	VALUE	UNIT
Parameter value	Preset Speed 1	350.0	rpm
Digital output	Basic I/O Relay T2	1	
Analog output	Basic I/O T31 Analog Output	0.2	

Illustration 7: State setting: *Not while running*

At power-up or when Logic's *Running Mode* changes, a *Starting state* (or 'No State') is set. The starting state can be configured in the Logic GUI Settings



Logic Running Mode

Executing [✓ Programming] Disabled

Logic Starting State: State 1

Illustration 8: Logic Setting: *Starting State*

4.3 Function block in state

Function blocks can be assigned to run only when a specific state is active, enabling state-dependent functionality and preventing resource conflicts. Each function block can be configured to run continuously *Run Always* or only within a designated state.

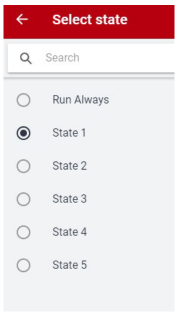


Illustration 9: Block State assignment.

If a function block is assigned to a state, it can be edited in the state that it is assigned to.

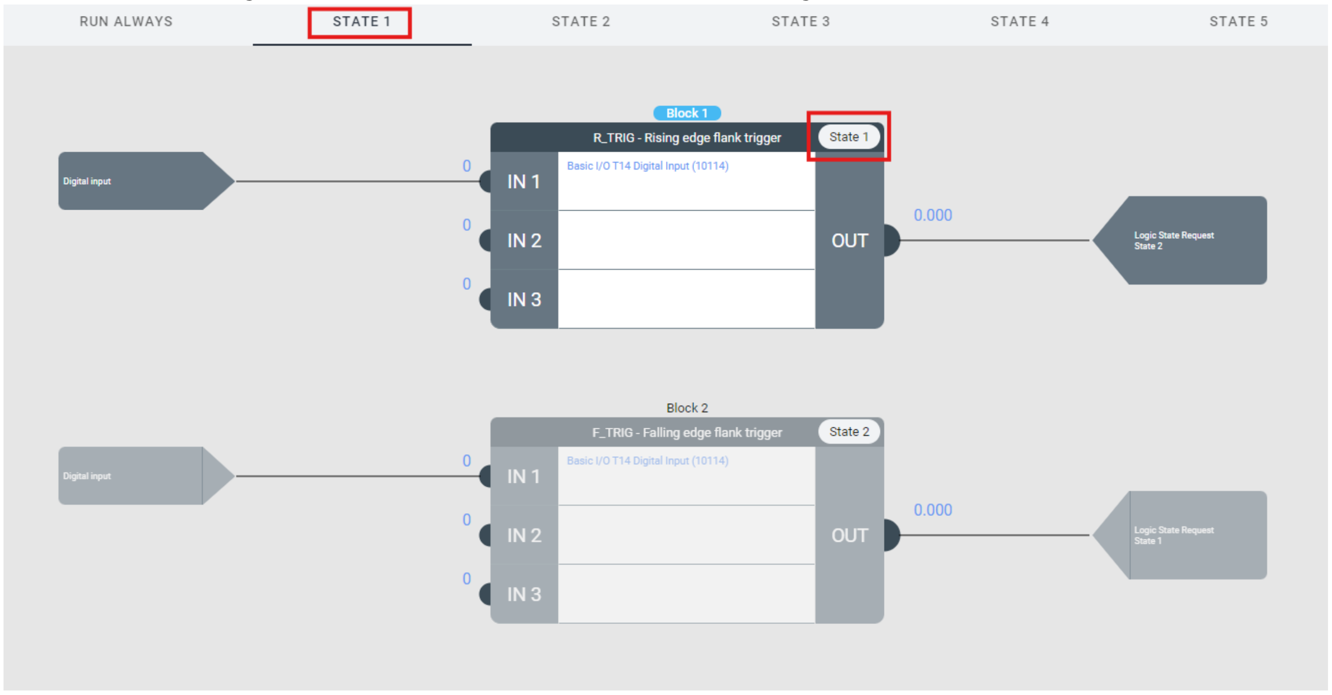
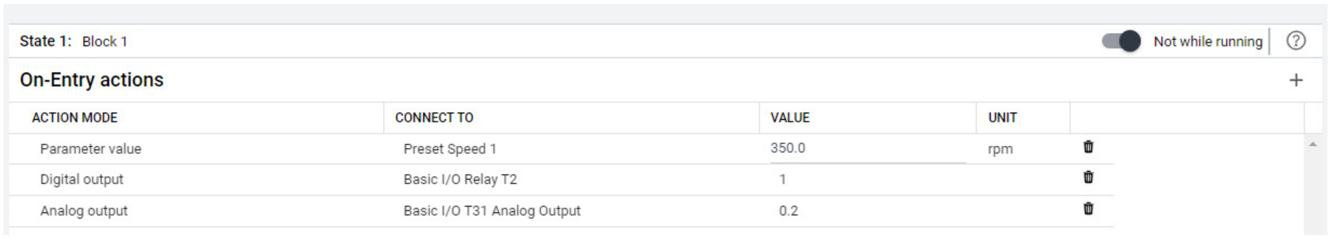

RUN ALWAYS
STATE 1
STATE 2
STATE 3
STATE 4
STATE 5
Block 1
R_TRIG - Rising edge flank trigger
State 1
Digital input
Basic I/O T14 Digital Input (10114)
0
IN 1
0
IN 2
0
IN 3
OUT
0.000
Logic State Request State 2
Block 2
F_TRIG - Falling edge flank trigger
State 2
Digital input
Basic I/O T14 Digital Input (10114)
0
IN 1
0
IN 2
0
IN 3
OUT
0.000
Logic State Request State 1

Illustration 10: Block State assignment. Block 1 is assigned to State 1. Block 1 can only be edited in the State 1 Tab.

4.4 State on-entry actions

State On-Entry actions consist of a user-defined list of actions that are executed when a state is activated.


State 1: Block 1 | Not while running

On-Entry actions

ACTION MODE	CONNECT TO	VALUE	UNIT
Parameter value	Preset Speed 1	350.0	rpm
Digital output	Basic I/O Relay T2	1	
Analog output	Basic I/O T31 Analog Output	0.2	

Illustration 11: Logic State 2 Action list example.

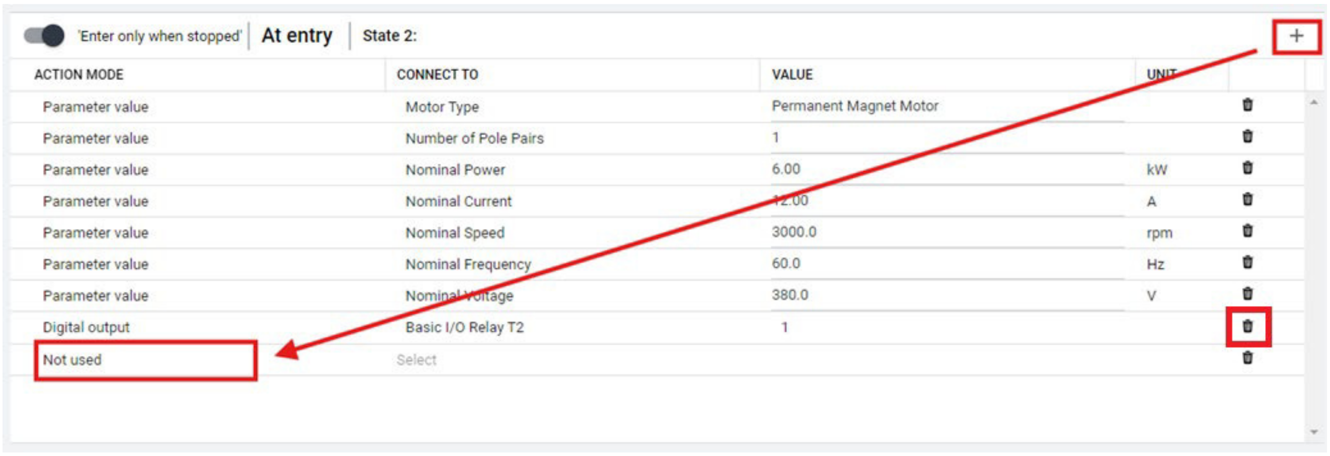
An action uses the same output mechanism as function block output and therefore is very similar. The value that must be written is added as a constant, where the format is presented based on the selected mode.

When an On-Entry action list is executed, all items are written at once as a transaction and therefore the sequence in which they are listed in does not have an impact on how they are written.

Table 13 : On-Entry Action Modes

Selection name	Description	Additional information
Not used	Does nothing.	
Digital output	Sets a digital output terminal to the selected value.	Digital output is a limited resource, with a limited number of instances.
Analog output	Sets an analog output terminal to the selected value.	Analog output is a limited resource, with a limited number of instances. The analog output is limited to values between 0.0 and 1.0 as it gives the internal reference for the analog output. The value is then converted to physical units based on the analog output mode and min. and max. value configurations in the application parameters.
Parameter value	Sets a parameter to the selected value.	Parameter write is a limited resource, with a limited number of instances. ReadOnly parameters cannot be written. Parameter values must be within the parameter's limits and a valid selection when limited by a selection list. Otherwise, a Logic Block Output Error event is triggered.
Logic speed reference	Sets the Logic speed reference to the selected value.	This mode provides a method for setting a speed reference directly from Logic. The Logic reference can be selected as a source in the control place configuration. Speed reference counts as a parameter write.
Logic torque reference	Sets the Logic torque reference to the selected value.	This mode provides a method for setting a torque reference directly from Logic. The Logic reference can be selected as a source in the control place configuration. Torque reference counts as a parameter write.
Logic process reference	Sets the Logic process reference to the selected value.	This mode provides a method for setting a process reference directly from Logic. The Logic reference can be selected as a source in the control place configuration. Process reference counts as a parameter write.
Logic position reference	Sets the Logic position reference to the selected value.	This mode provides a method for setting a position reference directly from Logic. The Logic reference can be selected as a source in the control place configuration. Position reference counts as a parameter write.

Adding items to the On-Entry action list is done by clicking the + symbol in the upper-right corner of the list.
A new item appears that can be configured.
Removing an item from the On-Entry list is done by clicking the trash bin-symbol on the right side of the respective entry.



'Enter only when stopped' | At entry | State 2: | +

ACTION MODE	CONNECT TO	VALUE	UNIT	
Parameter value	Motor Type	Permanent Magnet Motor		🗑
Parameter value	Number of Pole Pairs	1		🗑
Parameter value	Nominal Power	6.00	kW	🗑
Parameter value	Nominal Current	12.00	A	🗑
Parameter value	Nominal Speed	3000.0	rpm	🗑
Parameter value	Nominal Frequency	60.0	Hz	🗑
Parameter value	Nominal Voltage	380.0	V	🗑
Digital output	Basic I/O Relay T2	1		🗑
Not used	Select			🗑

Illustration 12: Logic On-Entry Action List, Selecting + to add new entries or Trash bin to delete entries.

4.5 Run Always

The *Run Always* tab contains all function blocks, including ones assigned to specific states and ones that execute regardless of the active state. Function blocks assigned to a specific state are marked accordingly.

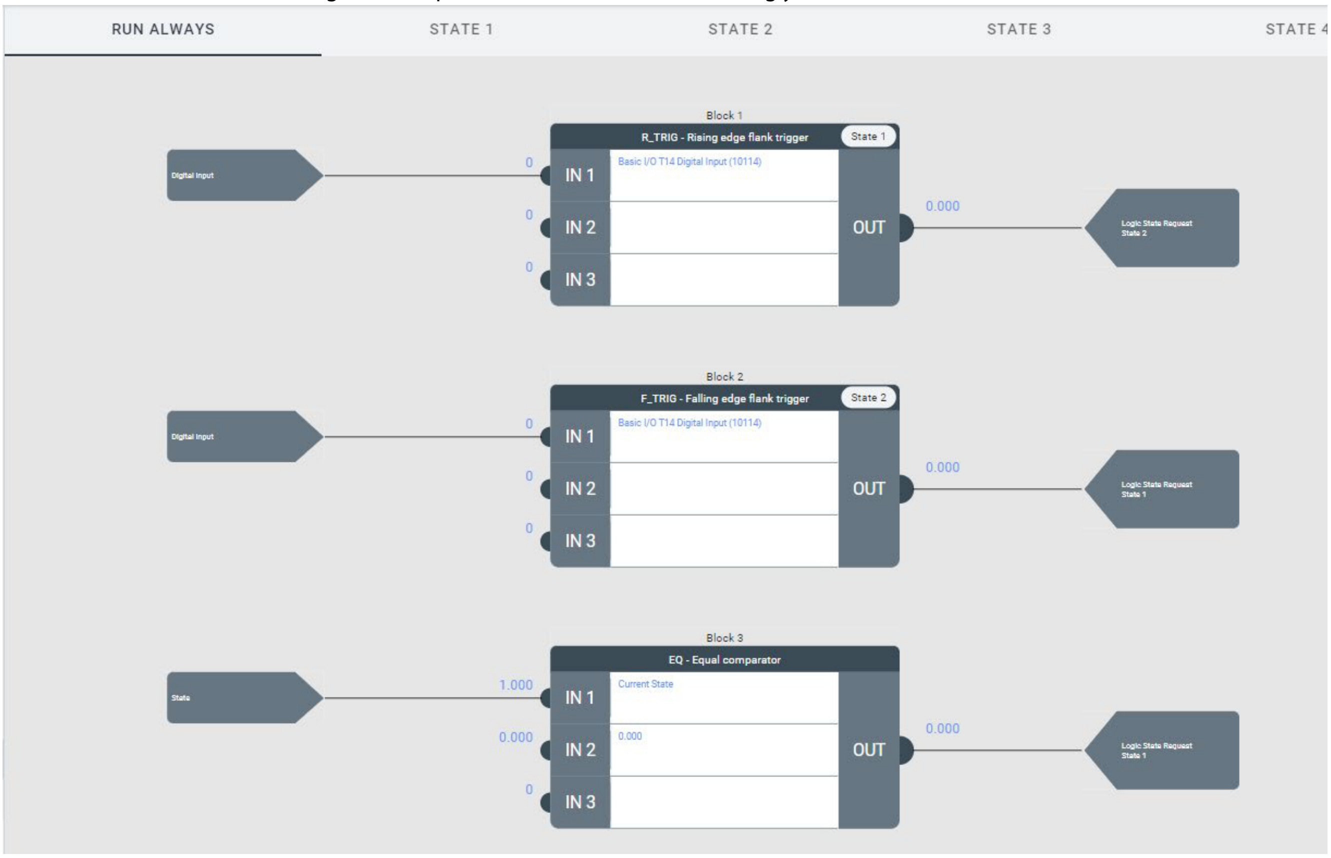

RUN ALWAYS
STATE 1
STATE 2
STATE 3
STATE 4
Digital Input
Block 1
R_TRIG - Rising edge flank trigger
State 1
Basic I/O T14 Digital Input (10114)
IN 1
IN 2
IN 3
OUT
0
0
0
0.000
Logic State Request State 2
Digital Input
Block 2
F_TRIG - Falling edge flank trigger
State 2
Basic I/O T14 Digital Input (10114)
IN 1
IN 2
IN 3
OUT
0
0
0
0.000
Logic State Request State 1
State
Block 3
EQ - Equal comparator
Current State
0.000
IN 1
IN 2
IN 3
OUT
1.000
0.000
0
0.000
Logic State Request State 1

Illustration 13: Run Always tab.

5 Error Handling

When executing Logic, it is important to check the active Events Log to see if any configuration errors have occurred. Logic can detect some configuration errors and issue a warning. Some errors will only occur after Logic's *Running Mode* is set to *Executing*.

NOTICE

LOGIC ERROR HANDLING

If a Logic Input Error or Logic Block Configuration Error is detected, the Logic outputs are not set and remain at their last value. This prevents incorrect or unintended outputs from being generated.

If more than one function block output is configured to drive the same output (DigOut, AnOut or Parameter). Logic sets the output to the last value assigned. Therefore, the output from the block with the highest number drives that output signal since the function blocks are executed sequentially.

Table 14 : Logic error handling

Group Number (Hex)	Number (Decimal)	Display Name	Description	Type of Event				Invert and Brake Chopper Action	
				I	W	F	PF	Inverter	Brake
0xFF06	5901	Logic Input Error	Configured input function reports an error.		X				
0xFF06	5902	Logic Output Error	Configured output function reports an error.		X				
0xFF06	5903	Logic Block Configuration Error	Logic block configuration is incorrect.		X				
0xFF06	5904	Logic State Error	Logic State Handling reports an error.		X				

6 Examples

To follow the examples, an open MyDrive® Insight instance with a connected iC7 drive that supports Logic is required.
All examples start with opening the Logic view in MyDrive® Insight and setting the *Running mode* to *Programming*. This allows the Logic configuration to be changed and stops the block program execution.
Once everything is configured and working as expected, check the debugging values in the GUI by setting Logic's *Running mode* to *Executing*. Now Logic drives the outputs and issues events if there are issues with the configuration.

6.1 Start based on Analog input T33

Description:

- The drive is controlled from I/O
- Frequency reference is given by analog input (T33)
- The drive is started when the T33 signal exceeds 50% and stopped when the signal goes below 40%.

In this example, the drive is controlled using I/O, and the frequency reference is provided by an analog input, specifically T33. The objective is to start the drive when the T33 signal exceeds a threshold level of 50% and stop it when the signal drops below a hysteresis level of 40%.

Analog Input T33 is already configured as the frequency reference in the drive's default settings. This example extends its functionality to include the start command based on the analog input level.

To implement this logic, use a GreaterThan (GT) function block to compare the analog input against the 50% threshold and a LessThan (LT) function block to compare it against the 40% hysteresis threshold. Also, an RS flip-flop can be used to latch the Start signal based on the results of these two comparisons.

Configure the function blocks and connect the inputs and outputs appropriately to create a logic configuration that enables the drive to start when the analog input exceeds the threshold and stop when it falls below the hysteresis level.

To configure both Logic and the drive's parameters, follow these steps:

1. Logic GUI Settings
   - Running mode = Programming
2. **Block1** handles the GreaterThan comparison.
   - Function = GT - GreaterThan
   - IN1 = Analog input - Basic I/O T33 Analog Input
   - IN2 = Numeric Constant - 0.5
3. **Block2** is used for LessThan function.
   - Function = *LT* - LessThan
   - IN1 = Analog input - Basic I/O T33 Analog Input
   - IN2 = Numeric Constant - 0.4
4. **Block3** contains the decision using RS – Set Reset flipflop.
   - Function = *RS* – Set Reset Flipflop
   - IN1 = Block output - Block 1
   - IN2 = Block output - Block 2
   - OUT = Digital Output - Logic Digital I/O 1
5. **Parameters view**, parameter group *5.5.6.1*
   - *Advanced Start Input Index 1* (*4722)* - Logic Digital I/O 1
6. Logic GUI Settings
   - Running mode = *Executing*

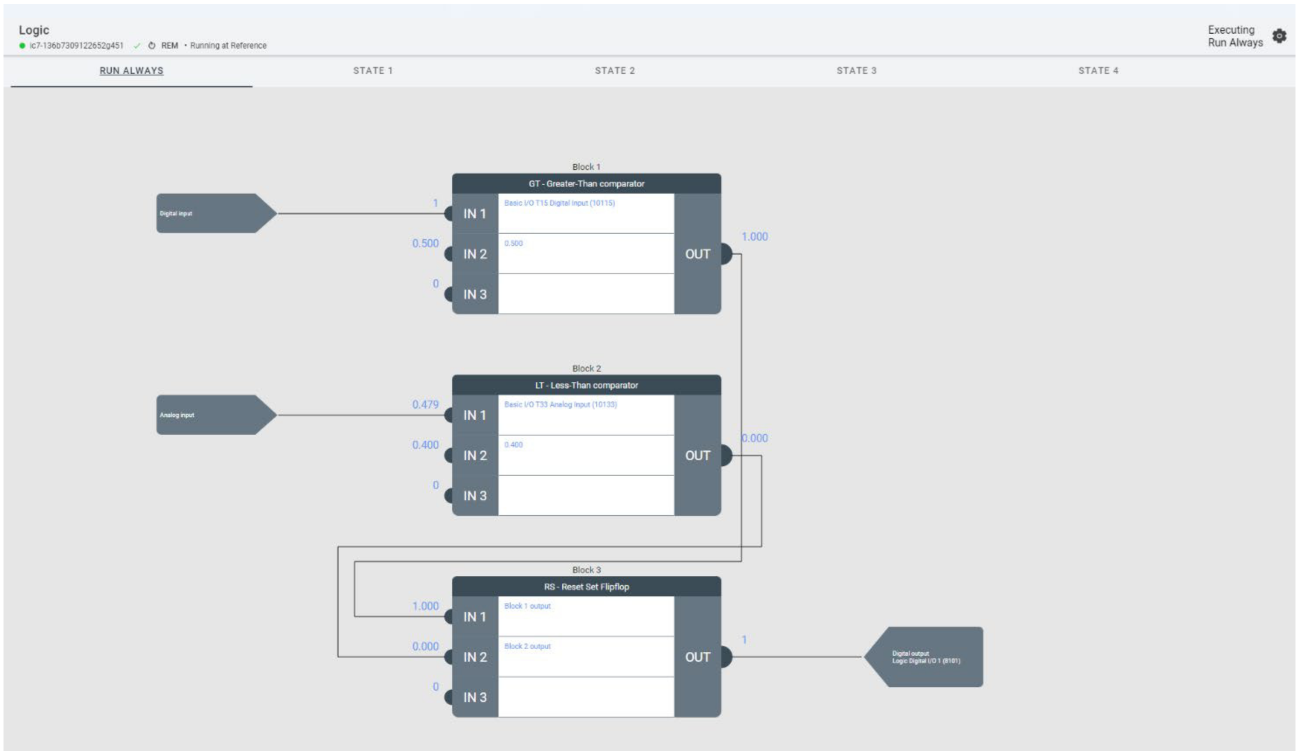


Illustration 14 : Example: Start based on Analog input T33

6.2 Scaling Motor Torque Limit by Analog input T34

Description:

- The drive is controlled from I/O
- Frequency reference is given by analog input (T33)
- Motor Torque Limit is changed linearly between 0…300% by an analog input (T34)

To change the value of parameter *Positive Torque Limit (1810)* using Logic, follow these steps:

1. Logic GUI Settings
   - Running mode = Programming
2. **Block1** for the multiplication function.
   - Function = MUL – Numeric multiply
   - IN1 = Analog input - Basic I/O T34 Analog Input
   - IN2 = Numeric constant - 300.0
   - OUT = Parameter value - *Positive Torque Limit* (1810)
3. Logic GUI Settings
   - Running mode = *Executing*

   Now, the *Positive Torque Limit* is set based on the analog input (T34), with a scaling of 0–300%.

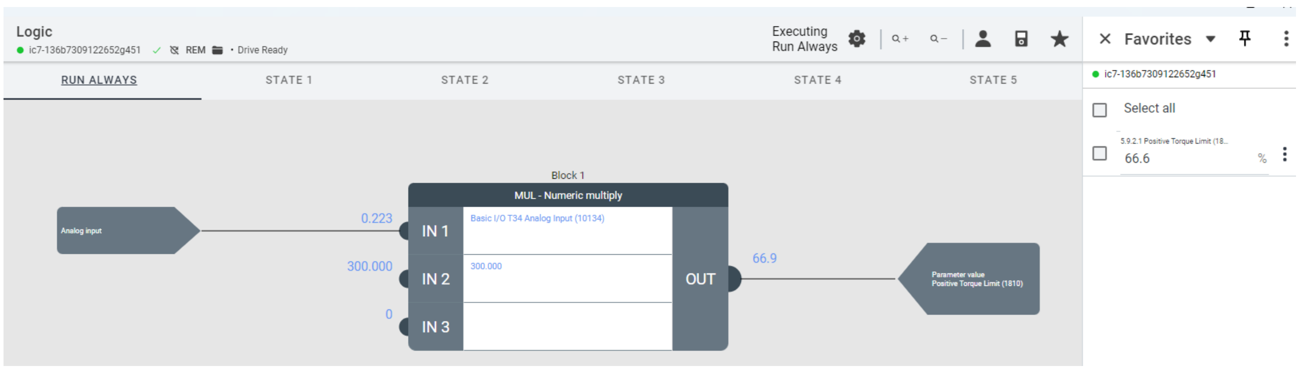


Illustration 15 : Example: Start based on Analog input T34

6.3 Delayed and conditional external fault

This example shows how to get some extra conditions added to the external fault triggering logic. By default, the external event is just a simple on/off type of logic connected, for example, to a digital input (T15). This example shows how to allow the fault triggering from digital input (T15) while the drive is in run mode and use a 2 second ON-Delay.

The best way to solve this issue is by stripping it down into two steps. The first step is to handle the conditional rule of only triggering when both the digital input T15 is active, and the drive is running. The second step is the ON-Delay handling which can be handled by the existing implementation in the application with parameter *4592 External Event 1 Delay*.

To implement delayed external fault with additional conditions, follow these steps:

1. Logic GUI Settings
   - Running mode = Programming
2. Use **Block1** for the conditional function.
   - Function = AND – Logical AND
   - IN1 = Digital input - Basic I/O T15 Digital Input
   - IN2 = Parameter Bit - *Motor Ctrl. Status Word* (1714) - Bit value = 1
   - OUT = Digital output - Logic Digital I/O 1
3. Go to the **Parameters view**, parameter group *5.2.2*.
   - *External Event 1 Input (4557)* - Logic Digital I/O 1
   - *External Event 1 Delay (4592)* - 2 s
   - Select the desired response. By default, parameter *External Event 1 Response (4559)* = Fault, ramp to coast
4. Return to the **Logic GUI Settings**
   - *Running mode = Executing*

   Now, the virtual terminal Logic Digital I/O 1 activates when both Digital input T15 is active and the drive is running, and an external event is triggered based on it with a 2 second delay.

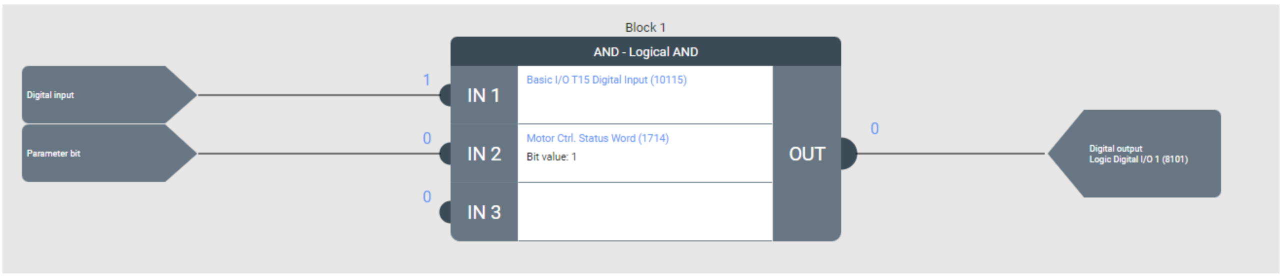


Illustration 16 : Example: Delayed external fault with condition

6.4 Custom scaling of status parameter to drive analog output

This example shows how to scale a signal and output it on an analog output. This is useful when the parameter or signal cannot be selected to be written to an analog output within the application or the application does not offer the desired scaling.
For example, the parameter *Motor Power Output* (*2305)* offers the possibility to select an output for the motor power signal. The scaling of the signal is 0–100% of the nominal power.

Logic can be used with customized scaling to handle overloads. In this example, the *Motor Power Output* is scaled as 0–300% of nominal power and output on Basic I/O T31 Analog Output.

1. Logic GUI Settings
   - Running mode = Programming
2. Use **Block1** for the scaling function: OUT = (Motor Shaft Power (kW) * 1/3) / (Nominal Power (kW)).
   - Function = *MULDIV – Combined numeric multiply and divide*
   - IN1 = Parameter value = *Motor Shaft Power* (*9008)*
   - IN2 = Numeric constant = 0.3333
   - IN3 = Parameter value = *Nominal Power* (*405)*
   - OUT = Analog output = Basic I/O T31 Analog Output
3. Make sure that the analog output T31 is configured as desired by configuring the parameters of parameter group *9.5.1 Output T31*.
4. Return to the **Logic GUI Settings**
   - *Running mode = Executing*

   Now, the analog terminal Basic I/O T31 Analog Output shows the motor power scaled as a range of 0–300% of the nominal power.

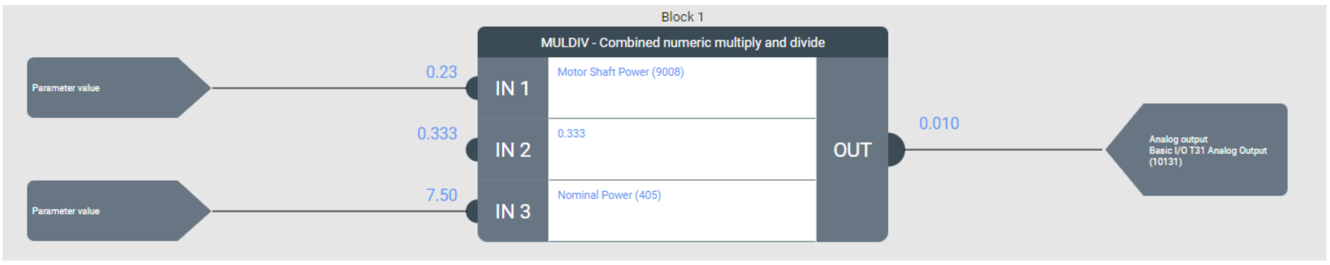


Illustration 17 : Example: Custom scaling of status parameters

6.5 Multi motor – advanced example

This example uses the Logic state handling feature added in the GR4 Automation Release and can thus only be used when the drive selected in MyDrive Insight contains Industry Application v.4.3.0 or Motion v.3.2.0 or newer.
This example shows one way of being able to run two different motors with one iC7 drive.

The goal is to use State 1 for a Permanent Magnet (PM) motor and State 2 for an induction motor. The hardware setup is the following: two motors are connected to the drive with a NO-contactor on each motor. We want to activate each contactor using the relays on the Basic I/O.
The selection of motor is to be done using bit15 in the *Fieldbus Control Word (1335)* where true selects the PM motor.
When the PM motor is selected, bit15 in the *Fieldbus Status Word (1307)* is set to True.

Step 1 of 3 – Setting up state change Logic
The logic for selecting the state and thereby the motor is always executed. The function blocks used for this are therefore all configured in the Run Always-tab.

1. Logic GUI Settings
   - Running mode = Programming
2. **Block 1** requests State 2, if bit15 in the fieldbus control word is true.
   - Function = EQ – Equal comparator
   - IN1 = Digital Input – CTW1 Bit15
   - IN2 = Boolean Constant – True
   - OUT = Logic State Request – State 2
3. **Block 2** requests State 1, if State 2 is NOT requested.
   - Function = NOT – Logical NOT
   - IN1 = Block Output – Block 1 Output
   - OUT = Logic State Request – State 1
4. **Block 3** sets bit15 in the fieldbus status word, if State 2 is active
   - Function= EQ – Equal comparator
   - IN1 = State – Current State
   - IN2 = Numeric Constant - 2
   - OUT = Digital Output – STW1 Bit 15

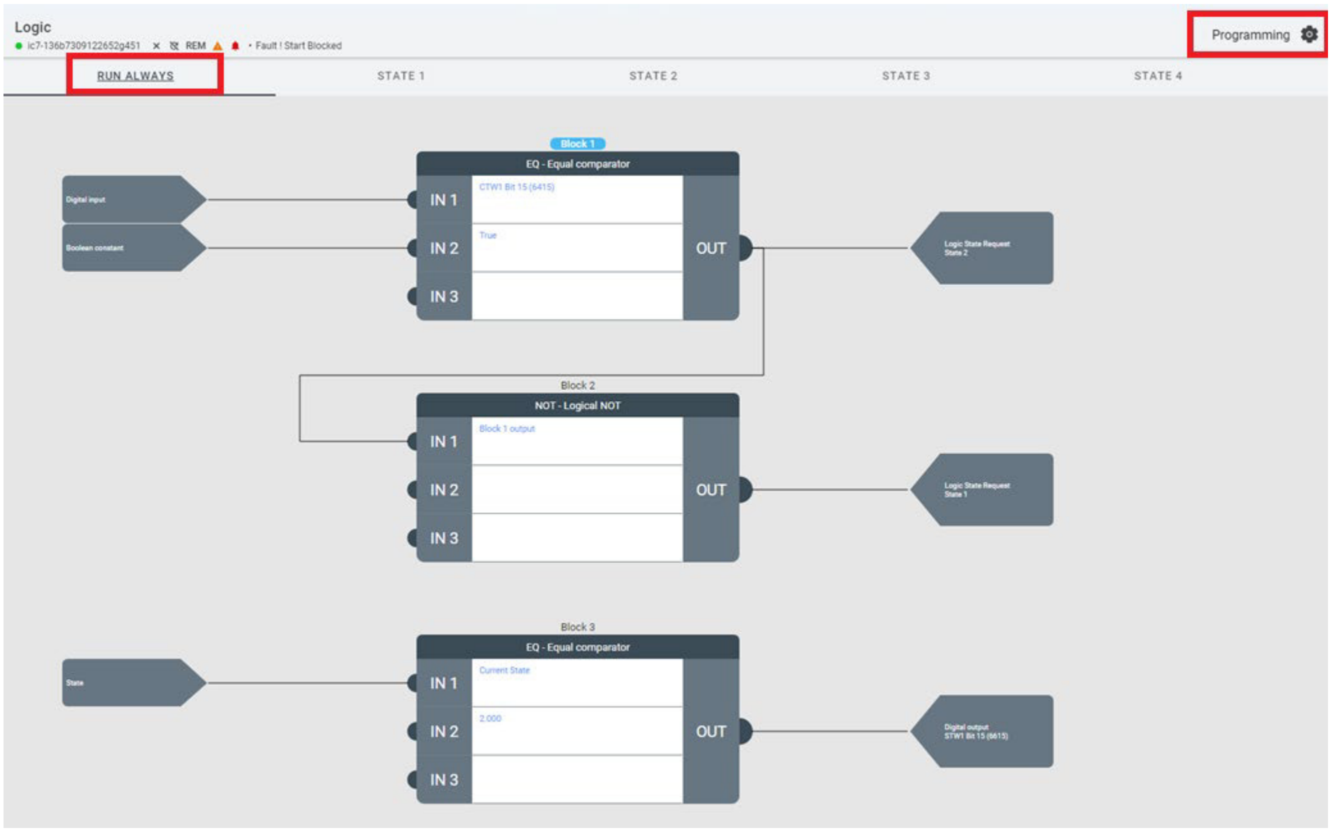
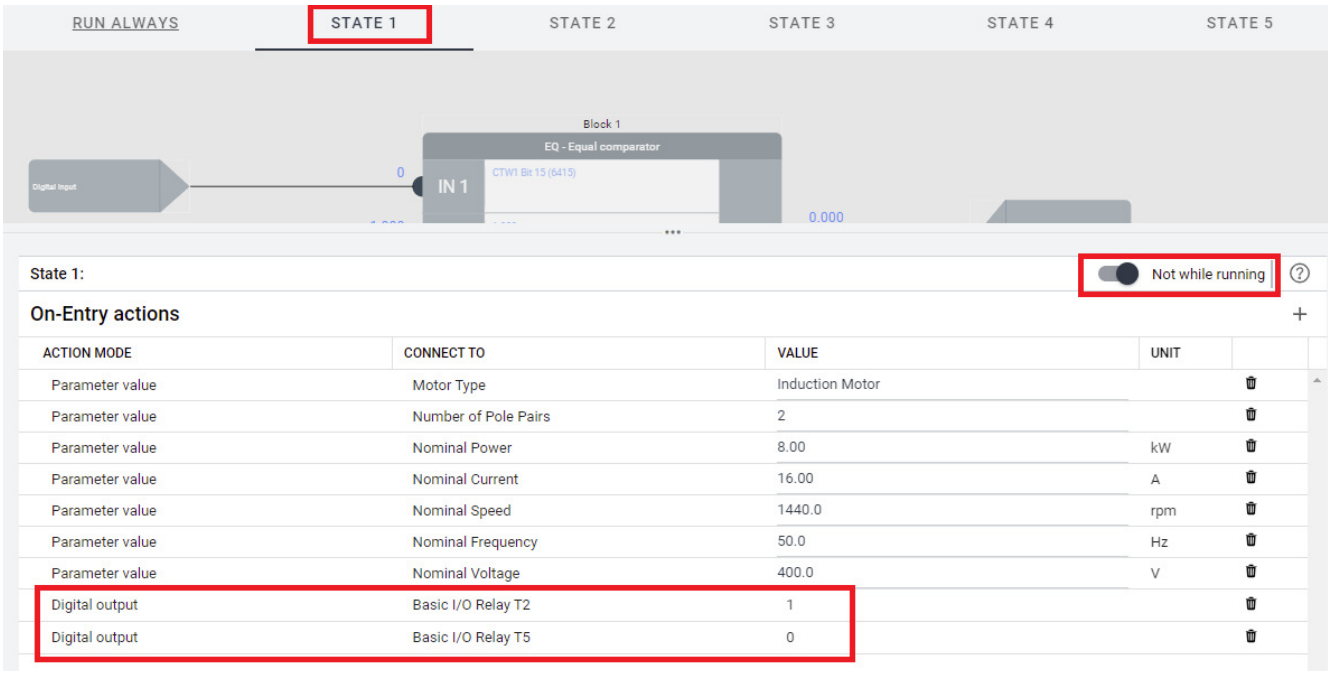


Illustration 18 : Example: Multi motor, State selection in Run Always-Tab

Step 2 of 3 – Setting up State 1 – Configure drive for induction motor

Because State 1 modifies parameters requiring a stopped motor, the state must be set to *Not while running*. Activating State 1 then energizes Basic I/O Relay T2 (activating the induction motor contactor) and de-energizes Basic I/O Relay T5 (deactivating the PM motor contactor).

5. To simplify the setup, first run the induction motor with the drive, noting down the Motor Nameplate Data.
   In this example only motor nameplate data is used for simplicity.
6. Select the State 1 tab in the Logic GUI
   - Activate the setting: *Not while running*
   - Action mode = Parameter value – Motor Type – Induction Motor
   - Add the Motor Nameplate parameters and values gathered in the previous step to the On-Entry action list
   - Action mode = Digital output – Basic I/O Relay T2 – 1
   - Action mode = Digital output - Basic I/O Relay T5 – 0



State 1:

Not while running

On-Entry actions

ACTION MODE	CONNECT TO	VALUE	UNIT
Parameter value	Motor Type	Induction Motor	
Parameter value	Number of Pole Pairs	2	
Parameter value	Nominal Power	8.00	kW
Parameter value	Nominal Current	16.00	A
Parameter value	Nominal Speed	1440.0	rpm
Parameter value	Nominal Frequency	50.0	Hz
Parameter value	Nominal Voltage	400.0	V
Digital output	Basic I/O Relay T2	1	
Digital output	Basic I/O Relay T5	0	

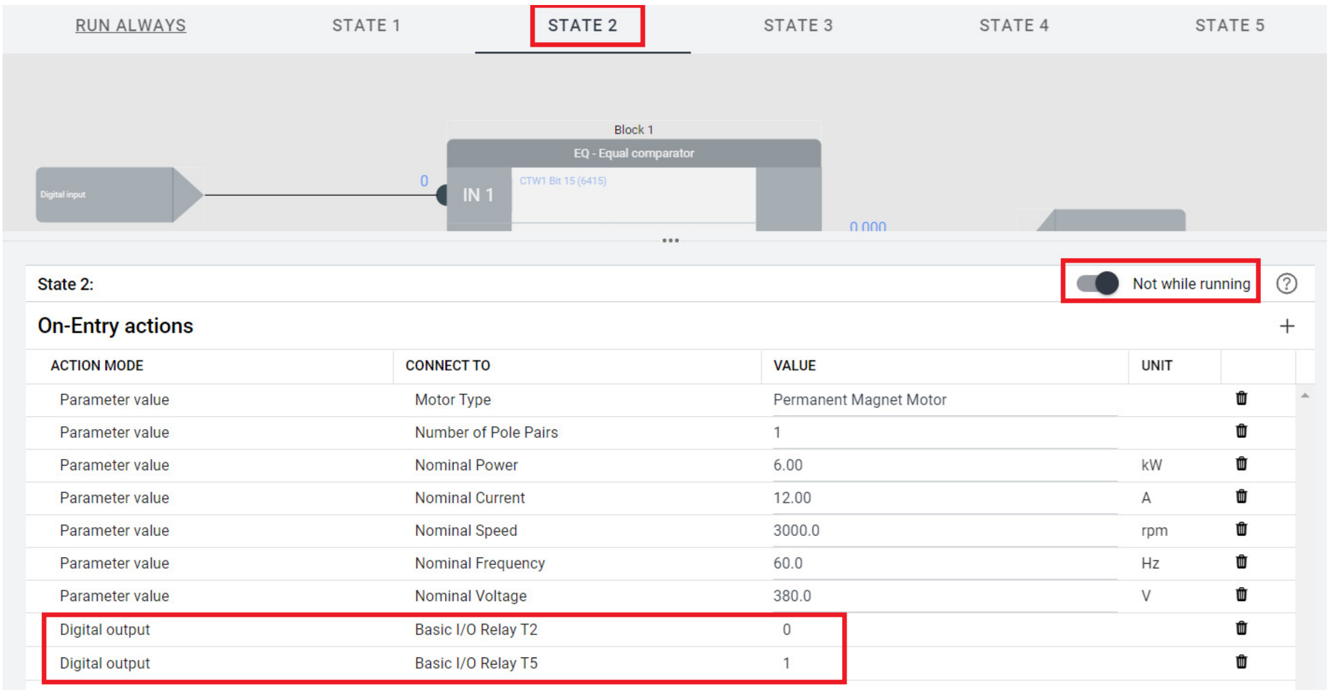
Illustration 19 : Example: Multi motor, State 1, Induction motor

Step 3 of 3 – Setting up State 2 – Configure drive for PM motor

Because State 2 also modifies parameters requiring a stopped motor, the state must be set to *Not while running*. Activating State 2 then de-energizes Basic I/O Relay T2 (deactivating the induction motor contactor) and energizes Basic I/O Relay T5 (activating the PM motor contactor).

7. To simplify the setup, first run the PM motor with the drive, noting down the Motor Nameplate Data.
8. Select the State 2 tab in the Logic GUI
   - Activate the setting: *Not while running*
   - Action mode = Parameter value – Motor Type – Permanent Magnet Motor
   - Add the Motor Nameplate parameters and values gathered in the previous step to the On-Entry action list
   - Action mode = Digital output – Basic I/O Relay T2 – 0
   - Action mode = Digital output - Basic I/O Relay T5 – 1
9. Logic GUI Settings
   - Running mode = Executing

   Now the *Fieldbus Control Word (1335)* bit15 can be used to switch between the connected induction and PM motor and the *Fieldbus Status Word (1307)* bit15 reflects the motor selection.



State 2: Not while running

On-Entry actions

ACTION MODE	CONNECT TO	VALUE	UNIT
Parameter value	Motor Type	Permanent Magnet Motor	
Parameter value	Number of Pole Pairs	1	
Parameter value	Nominal Power	6.00	kW
Parameter value	Nominal Current	12.00	A
Parameter value	Nominal Speed	3000.0	rpm
Parameter value	Nominal Frequency	60.0	Hz
Parameter value	Nominal Voltage	380.0	V
Digital output	Basic I/O Relay T2	0	
Digital output	Basic I/O Relay T5	1	

Illustration 20 : Example: Multi motor, State 2, PM motor

ENGINEERING
TOMORROW

Danfoss

Danfoss A/S
Ulsnaes 1
DK-6300 Graasten
drives.danfoss.com

M00450