VEICHI DRIVE FOR EVER 伟创电气

FBD Visual Programming Manual

VEICHI

Version Change Log

Date	Version	Content
2024.08	V1.0	First version issued

Catalog

Catalog 1
1.Overview 3
1.1 Background 3
1.2 Module Status 3
2 Free Module Application 3
2.1 Logic Operation 4
2.1.1 Logic AND 4
2.1.2 Logic NOT 4
2.1.3 Logic NOT 5
2.1.4 Logic XOR 6
2.1.5 Logic XNOR 7
2.1.6 Negate 8
2.2 Arithmetic Operation 8
2.2.1 Absolute Value 8
2.2.2 Additive Operations 9
2.2.3 Subtractive Operations 10
2.2.4 Multiplicative Operations 11
2.2.5 Division Operations 13
2.2.6 Square Root 13
2.3 Signal Processing 14
2.3.1 Limit 14
2.3.2 Differential 15
2.3.3 Hysteresis 16
2.3.4 Maximum 17
2.3.5 Minimum 18
2.3.6 Average 18
2.3.7 Filter 19
2.3.8 Level-Pulse Conversion 20
2.3.9 Pulse-Level Conversion 21
2.3.10 Delay 22
2.3.11 Pulse Generator 23
2.3.12 Ramp Processing 24
2.3.13 Transit 25
2.3.14 Opposite Number 25
2.3.15 Equals 26

2.3.16 Greater Than or Equal To 26
2.3.17 Greater Than(GT) 27
2.3.18 Less Than or Equal To 28
2.3.19 Less Than 28
2.3.20 Not Equal (NE) 29
2.3.21 Pulse Counter 30
2.3.22 RS or SR 31
2.3.23 Switching Signal (SEL) 32
2.3.24 Fault Code Monitoring 32
2.3.25 Fault Subcode Monitoring 33
2.3.26 Write Function Codes 34

1.Overview

1.1 Background

This document mainly introduces the function of the lower computer free programming module in the FBD visual programming project, aiming at facilitating the users to understand the internal logic, setting sequence, fault finding and other precautions at any time in the actual application. There are currently 10 free modules, each with 3 inputs, 1 output and 1 module status. The inputs, outputs and statuses of all free modules are displayed through the U07 connector and in the visual programming module of the host computer.

1.2 Module Status

The status values of all free modules are centrally displayed in the connector parameter U07.80. The bit0~bit9 of this parameter display the status values of 10 free modules respectively, and multiple module statuses can be displayed at the same time. In addition to the general display of the parameter list connector, the module status can also be indicated in the visual programming module by the module color.

For example, when the parameter U07.80 of the parameter list connector is 51 and the binary number is 0011 0011, i.e., the parameter bit0, bit1, bit4 and bit5 are all 1, it means that the status of module 1, module 2, module 5 and module 6 is 1, which means that the above 4 modules have an internal arithmetic error. At this time, the color of the above 4 modules in the visual programming of the host computer changes to red.

The meaning of the module status is 1 or the color turns red:

1. Illegal module input: The input type is not bit operation during logic operation, i.e., the input value is non-0 or non-1; or the fault code and fault sub-code over-limit.
2. Module setting error: Setting error in logic operation, limit function, logic delay, level-to-pulse function, slope processing, reset reset, switch function.
3. Data overflow: If a data overflow occurs inside the module during multiplication, the variables inside the module are all int32 bits.
4. Non-0 divisor: If the divisor (module inputs 2 and 3) is 0 during division, the module's internal variables are all int32 bits.
5. Radicand <0: The radicand is lower than 0 during square root calculation.

2 Free Module Application

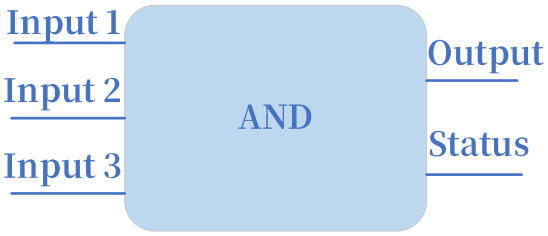
2.1 Logic Operation

Logic operations include application functions such as AND, NOT, XOR, XNOR, etc., to illustrate the logic of the module's application functions, input/output constraints, and module status indication.

2.1.1 Logic AND

(1) Function

Perform an AND logic on the 3 inputs; if both input conditions are true (or non-0 values), the result is true (or non-0 values), and if one or both two input conditions are false (or zero values), the result is false (or zero values).



Output= Input 1 && Input 2 Input 3

(2) Module

Module Interface	Type	Comment
Input 1	**0/1**	Bit operation
Input 2	0/1	Bit operation
Input 3	0/1	Bit operation
Output	0/1	-
Status	0/1	1(illegal input, setting error)

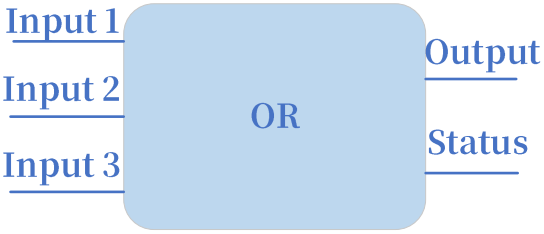
Note:

- When all 3 inputs of the module are valid, the module operates normally, when 1 input of the module is invalid, the remaining 2 inputs operate normally, when 2 inputs of the module are invalid, the module status is 1 and the output is 0. At this time, the module is set up incorrectly, and the host computer visual programming module turns red to indicate the module arithmetic abnormalities.
- When the input type of 3 inputs of the module is not bit operation, and the input value is non-0 and non-1, the module status is 1 and the output is 0. At this time, the input of the module is illegal, and the host computer visual programming module will turn red to indicate the module arithmetic abnormalities.

2.1.2 Logic NOT

(1) Function

Perform a NOT logic on the 3 inputs; one condition at least is true (or non-0 values), the result is true (or non-0 values), and if both of the two input conditions are false (or zero values), the result is false (or zero values).



Output= Input 1 || Input 2 || Input 3

(2) Module

Module Interface	Type	Comment
Input 1	**0/1**	Bit operation
Input 2	0/1	Bit operation
Input 3	0/1	Bit operation
Output	0/1	-
Status	0/1	1(illegal input, setting error)

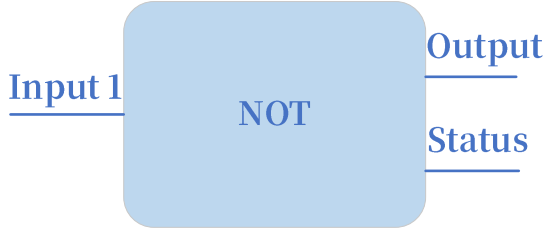
Note:

- When all 3 inputs of the module are valid, the module operates normally, when 1 input of the module is invalid, the remaining 2 inputs operate normally, when 2 inputs of the module are invalid, the module status is 1 and the output is 0. At this time, the module is set up incorrectly, and the host computer visual programming module turns red to indicate the module arithmetic abnormalities.
- When the input type of 3 inputs of the module is not bit operation, and the input value is non-0 and non-1, the module status is 1 and the output is 0. At this time, the input of the module is illegal, and the host computer visual programming module will turn red to indicate the module arithmetic abnormalities.

2.1.3 Logic NOT

(1) Function

Conduct a NOT logic on the input; if the input is true (or a non-0 value), the result is false (or zero). If the input is false (or zero), the result is false (or non-0 value).



The result =! Input 1

(2) Module

Module Interface	Type	Comment
Input 1	0\ non-0	Bit operation
Output	0/1	-
Status	0/1	-

Note:

- The actual application type of the module input is bool, i.e. true or false. However, the variable is of type int32 for the internal operation of the module, so users need to differentiate it from the inverse operation when using it.

2.1.4 Logic XOR

(1) Function

Conduct XOR on the 3 inputs, and the result is 0 if the two inputs are the same(both 0 or both 1), and the result is 1 if the two inputs are different (one is 0 and the other is 1).



Output= Input 1 ^ Input 2 ^ Input 3

(2) Module

Module Interface	Type	Comment
Input 1	**0/1**	Bit operation
Input 2	0/1	Bit operation
Input 3	0/1	Bit operation
Output	0/1	-
Status	0/1	1(illegal input, setting error)

Note:

- When all 3 inputs of the module are valid, the module operates normally, when 1 input of the module is invalid, the remaining 2 inputs operate normally, when 2 inputs of the module are invalid, the module status is 1 and the output is 0. At this time, the module is set up incorrectly, and the host computer visual programming

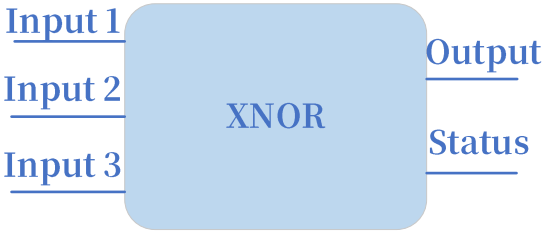
module turns red to indicate the module arithmetic abnormalities.

- When the input type of 3 inputs of the module is not bit operation, and the input value is non-0 and non-1, the module status is 1 and the output is 0. At this time, the input of the module is illegal, and the host computer visual programming module will turn red to indicate the module arithmetic abnormalities.

2.1.5 Logic XNOR

(1) Function

Perform XNOR on the 3 inputs; the output is true (or 1) if and only if the two inputs are the same; if the two inputs are different, the output is false (or 0).



Output= (Input 1 * Input 2 * Input 3)+（!Input1 * !Input2 * !Input3）

(2) Module

Module Interface	Type	Comment
Input 1	**0/1**	Bit operation
Input 2	0/1	Bit operation
Input 3	0/1	Bit operation
Output	0/1	
Status	0/1	1(illegal input, setting error)

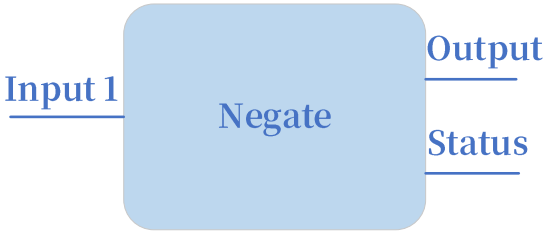
Note:

- When all 3 inputs of the module are valid, the module operates normally, when 1 input of the module is invalid, the remaining 2 inputs operate normally, when 2 inputs of the module are invalid, the module status is 1 and the output is 0. At this time, the module is set up incorrectly, and the host computer visual programming module turns red to indicate the module arithmetic abnormalities.
- When the input type of 3 inputs of the module is not bit operation, and the input value is non-0 and and non-1, the module status is 1 and the result is 0; At this time, the input of the module is illegal, and the host computer visual programming module will turn red to indicate the module arithmetic abnormalities.

2.1.6 Negate

(1) Function

Negate the input 1.



Output = ~Input 1

(2) Module

Module Interface	Type	Comment
Input 1	**0/1**	Bit operation
Output	0/1	-
Status	0/1	1(illegal input)

Note:

- The module input type must be a bitwise operation. This function performs a negate operation on one of the binary bits of the current variable. Note the distinction from logical AND and NOT.
- When the input type of of the module is not bit operation, and the input value is non-0 and non-1, the module status is 1 and the result is 0. At this time, the input of the module is illegal, and the host computer visual programming module will turn red to indicate the module arithmetic abnormalities.

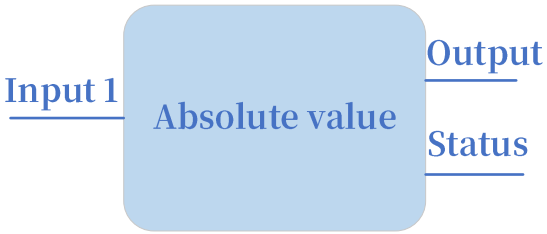
2.2 Arithmetic Operation

Arithmetic operations including addition, subtraction, multiplication, division, proportionality, absolute value, and others to illustrate the logic of the module's application functions, input and output limitations, and module status indications.

2.2.1 Absolute Value

（1） Description

Absolute value calculation makes the result to be positive and has the same amplitude as the input.



Output =| Input 1|

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Input type is constant/address number
Output	Uint32	
Status	/	/

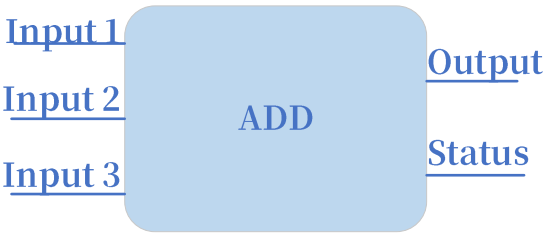
Note:

- The data types of module inputs and outputs are all unsigned variables, but since the variables are int32-bit for internal module arithmetic, negative arithmetic can be performed when the module input type is an address number.
- When the module input is higher than or equal to 0, the module output is equal to the input, and when the module input is lower than 0, the module output is equal to the opposite of the input.

2.2.2 Additive Operations

（1） Description

Get the sum of the three inputs added together.



Output= Input 1+ Input 2 + Input 3

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Input type is constant/address number
Input 2	Uint16	Input type is

		constant/address number
Input 3	Uint16	Input type is constant/address number
Output	Uint32	
Status	/	/

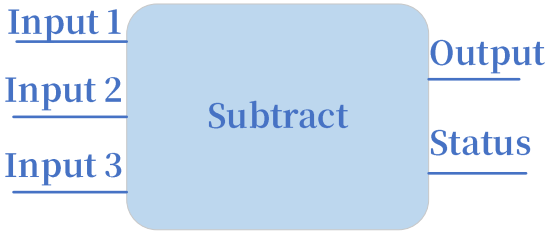
Note:

- The data types of module inputs and outputs are all unsigned variables, but since the variables are int32-bit for internal module arithmetic, negative arithmetic can be performed when the module input type is an address number.

2.2.3 Subtractive Operations

(1) Function

Get the difference of 3 inputs subtracted from each other.



Output= Input 1- Input 2 - Input 3

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Subtracted number and input type are constant/address numbers
Input 2	Uint16	Subtracted number and input type are constant/address numbers
Input 3	Uint16	Subtracted number and input type are constant/address numbers
Output	Uint32	
Status	/	/

Note:

- The data types of module inputs and outputs are all unsigned variables, but since the variables are int32-bit for internal module arithmetic, negative arithmetic can be performed when the module input type is an

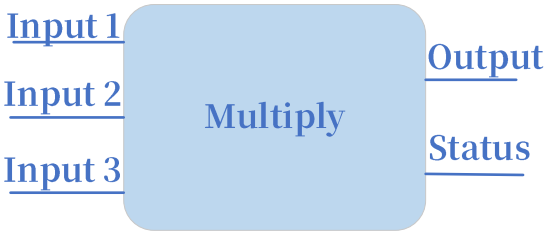
address number.

- Since the connector parameter type is Uint32 on the host computer, when a negative value is calculated within the module, both the parameter list connector monitoring value and the visual programming module display values are Uint32 bit data. For example, 4294967246 represents an actual value of -50.

2.2.4 Multiplicative Operations

(1) Function

Get the multiplication of the 3 inputs.



Output= Input 1 * Input 2 * Input 3

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Input type is constant/address number
Input 2	Uint16	Input type is constant/address number
Input 3	Uint16	Input type is constant/address number
Output	Uint32	
Status	0/1	1 (Data Overflow)

Note:

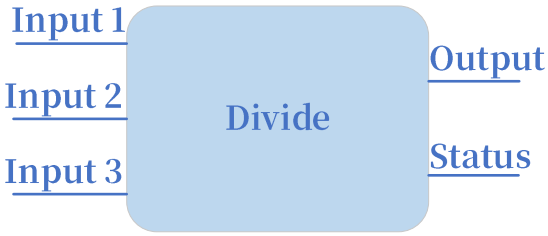
- The data types of module inputs and outputs are all unsigned variables, but since the variables are int32-bit for internal module arithmetic, negative arithmetic can be performed when the module input type is an address number.
- Since the connector parameter type is Uint32 on the host computer, when a negative value is calculated within the module, both the parameter list connector monitoring value and the visual programming module display values are Uint32 bit data. For example, 4294967246 represents an actual value of -50.
- Module function for the multiplication operation, if the module internal calculation result is greater than the

upper limit of Uint32, then the module state is 1, the output is 0, at this time, the upper computer visual programming module turns red to indicate that the module arithmetic abnormalities.

2.2.5 Division Operations

(1) Function

Get the quotient of the division of 3 inputs.



Output= Input 1 / Input 2 / Input 3, the

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Dividend and input type are constant/address numbers
Input 2	Uint16	Divisor and input type are constant/address numbers
Input 3	Uint16	Divisor and input type are constant/address numbers
Output	Uint32	
Status	0/1	1 (divisor is 0)

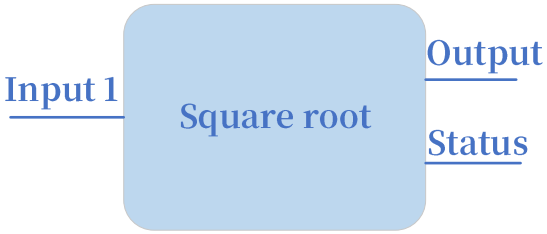
Note:

- The data types of module inputs and outputs are all unsigned variables, but since the variables are int32-bit for internal module arithmetic, negative arithmetic can be performed when the module input type is an address number.
- Since the connector parameter type is Uint32 on the host computer, when a negative value is calculated within the module, both the parameter list connector monitoring value and the visual programming module display values are Uint32 bit data. For example, 4294967246 represents an actual value of -50;
- Module function for the multiplication operation, if the module internal calculation result is greater than the upper limit of Uint32, then the module state is 1, the output is 0, at this time, the upper computer visual programming module turns red to indicate that the module arithmetic abnormalities.

2.2.6 Square Root

(1) Function

Get the arithmetic square root of the number being squared.



Output =$\sqrt{IN1}$, and the square number being squared cannot be less than 0.

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Dividend and input type are constant/address numbers
Output	Uint32	
Status	0/1	1 (Radicand less than 0)

Note:

- The data types of module inputs and outputs are all unsigned variables, but since the variables are int32-bit for internal module arithmetic, negative arithmetic can be performed when the module input type is an address number.
- Module function for the multiplication operation, if the module internal calculation result is greater than the upper limit of Uint32, then the module state is 1, the output is 0, at this time, the upper computer visual programming module turns red to indicate the module arithmetic abnormalities.

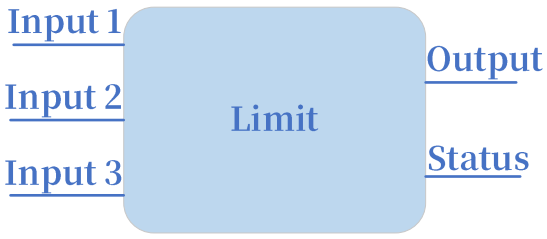
2.3 Signal Processing

Signal processing includes limiting, filtering, and logic delay to illustrate the logic of the functions, input and output limitations, and module status alarms.

2.3.1 Limit

(1) Function

The signal to be processed is limited, when the signal to be processed > upper limit, the output = upper limit value; when the signal to be processed < lower limit, the output = lower limit value; when lower limit <= signal to be processed <= upper limit, the output = value of the signal to be processed.



Input 1: Signal to be processed; Input 2: Upper limit; Input 3: Lower limit.

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Signal to be processed, input type are constant/address numbers
Input 2	Uint16	Upper limit and input type are constant/address numbers
Input 3	Uint16	Lower limit and input type are constant/address numbers
Output	Uint32	
Status	0/1	1(Setting error)

Note:

- When the module function is limit calculation, if the upper limit value is lower than the lower limit value, the module status is 1 and the output is 0. At this time, the host computer visual programming module turns red to indicate the module arithmetic abnormalities.
- When input 1 of the limit is invalid, the output of the module is always 0; when inputs 2 and 3 are invalid at the same time, the output of the module is always 0; when input 1 of the module is valid, and any one of input 2 and 3 of the module is valid, the module can operat normally.

2.3.2 Differential

(1) Function

The signal to be processed is differentiated, and its output is the current signal value minus the last cycle of the signal multiplied by the gain factor, and the cycle is the differentiation interval in units of 10ms.



Output = (Signal to be processed - Previous signal) * Gain factor

Input 1: Signal to be processed; Input 2: Gain factor; Input 3: Differential time interval/10ms.

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Signal to be processed, input type are constant/address numbers
Input 2	Uint16	Gain factor and input type are constant/address numbers
Input 3	Uint16	Differential interval/10ms and input type are constant/address numbers
Output	Uint32	
Status	/	/

Note:

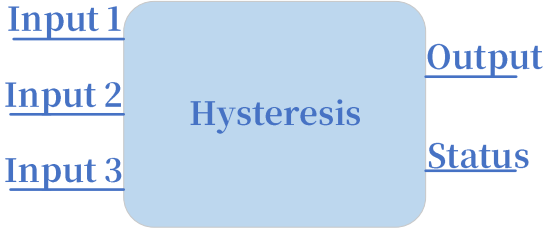
- When both input 3 and input 1 of the hysteresis function module are valid, the module can operate normally, otherwise the module does not operate, the output is always 0, and the module state is 0.

2.3.3 Hysteresis

(1) Function

The signal to be processed is compared with hysteresis values. When the signal to be processed is rising, if it is lower than threshold + hysteresis, the module output is 0; If it is higher than threshold + hysteresis, the module output is 1.

When the signal to be processed is falling, if it is higher than threshold - hysteresis, the module output is 1; If it is lower than threshold - hysteresis, the module output is 0.



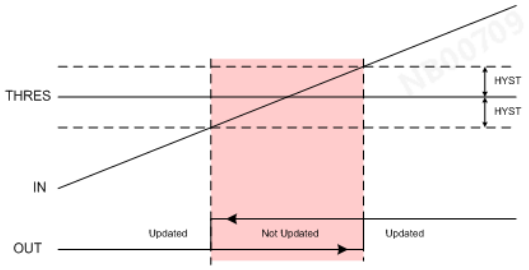
Input 1: Signal to be processed; Input 2: Threshold; Input; 3: Hysteresis

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Signal to be processed, input type are constant/address numbers
Input 2	Uint16	Threshold and input type are constant/address numbers
Input 3	Uint16	Hysteresis and input type are constant/address numbers
Output	Uint32	
Status	/	/

Note:

- When both input 1 and input 2 of the hysteresis function module are valid, the module can operate normally, otherwise the module does not operate, the output is always 0, and the module state is 0.



2.3.4 Maximum

(1) Function

Take the maximum value of input 1, input 2 and input 3.



(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Input type is constant/address number

Input 2	Uint16	Input type is constant/address number
Input 3	Uint16	Input type is constant/address number
Output	Uint32	
Status	/	/

Note:

- When all 3 inputs to the module are valid, the module takes the maximum of the 3 inputs; when 2 inputs to the module are valid, the module takes the larger one of the 2 inputs; when only one input to the module is valid, the output of the module is equal to the input.

2.3.5 Minimum

(1) Function

Take the minimum value of input 1, input 2 and input 3.



(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Input type is constant/address number
Input 2	Uint16	Input type is constant/address number
Input 3	Uint16	Input type is constant/address number
Output	Uint32	
Status	/	/

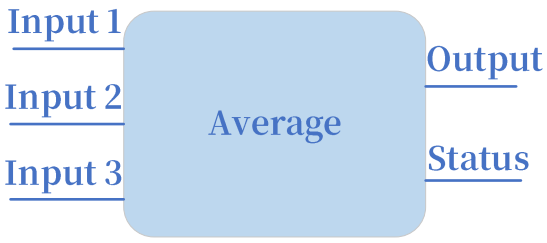
Note:

- When all 3 inputs to the module are valid, the module takes the lowest value of the 3 inputs; when 2 inputs to the module are valid, the module takes the smaller one of the 2 inputs; when only one input to the module is valid, the output of the module is equal to the input.

2.3.6 Average

(1) Function

Take the average value of input 1, input 2 and input 3.



Output = (Input 1 + Input 2 + Input 3) / X, X is the number of valid inputs to the module.

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Input type is constant/address number
Input 2	Uint16	Input type is constant/address number
Input 3	Uint16	Input type is constant/address number
Output	Uint32	
Status	/	/

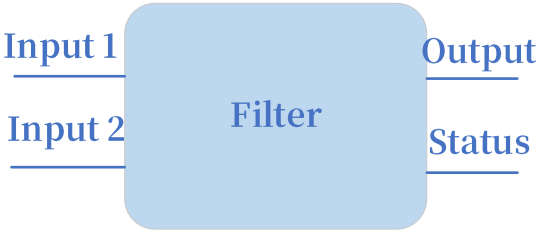
Note:

- When all 3 inputs to the module are valid, the module takes the average value of the 3 inputs; when 2 inputs to the module are valid, the module takes the average value of the 2 inputs; when only one input to the module is valid, the output of the module is equal to the input.

2.3.7 Filter

(1) Function

Conduct 1-order low-pass filter on the signal to be processed.



Input 1: Signal to be processed; Input 2: Filter time (10ms)

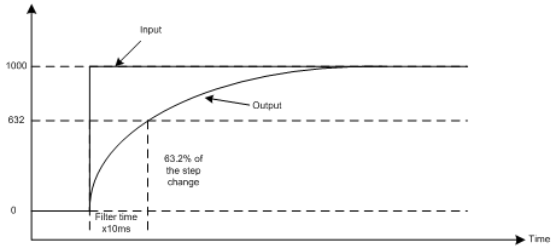
(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Signal to be processed and input type are constant/address numbers
Input 2	Uint16	Filter and input type are constant/address numbers

Output	Uint32	
Status	/	/

Note:

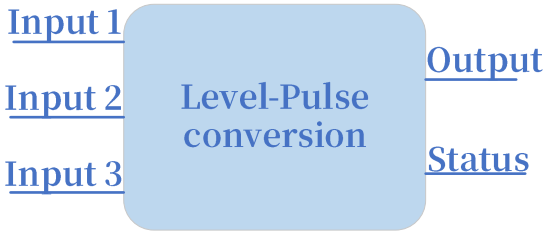
- When both input 2 of the module are valid, the filter function is normal, otherwise the module output is always 0.



2.3.8 Level-Pulse Conversion

(1) Function

Input 1: Level signal to be processed to converted into a pulse signal with a pulse period of input 2. Trigger modes are selected on the input 3, which are rising edge, falling edge and bidirectional triggering.



1: Level signal to be processed; Input 2: Pulses cycle/10ms; Input 3: Trigger modes; 0 means rising edge, 1 means falling edge, and 2 means bidirectional triggering.

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Level signal to be processed and input type are constant/address numbers
Input 2	Uint16	Pulse cycle and input type are constant/address numbers
Input 3	Uint16	Trigger method and input type are constant/address numbers
Output	Uint32	
Status	0/1	1(illegal input, setting error)

Note:



- When both input 3 of the module are valid, the filter function is normal, otherwise the module output is always 0.
- When module input 3 is 0, the low level turns to high level to trigger the rising edge; When module input 3 is 1, the high level turns to low level to trigger the falling edge; when module input 3 is 2, the level signal changes to trigger the pulse signal;
- When the value of module input 1 is non-0 and non-1, the module status is 1 and the output is 0. At this time, the module input is illegal, and the host computer visual programming module turns red to indicate the module abnormalities.
- When the value of module input 3 is greater than or equal to 3, the module status is 1 and the output is 0. At this time, the module input is incorrect, and the host computer visual programming module turns red to indicate the module abnormalities.

2.3.9 Pulse-Level Conversion

(1) Function

Convert input 1, the pulse signal to be processed to the level signal. Count the rising edge of the pulse signal and output a high level when the number of rising edges is odd and a low level when the number of rising edges is even.



Input 1: The pulse signal to be processed

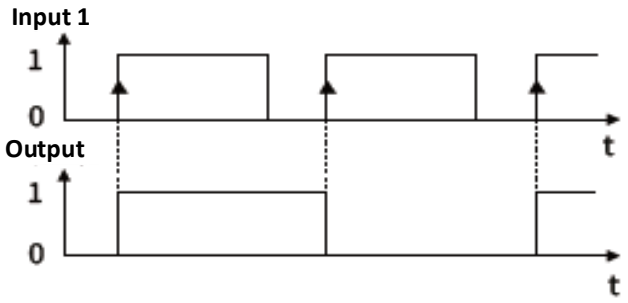
(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Pulse signal to be processed and input type are constant/address numbers
Output	Uint32	

Status	/	/

Note:

- When both input the module are valid, the filter function is normal, otherwise the module output is always 0.



2.3.10 Delay

(1) Function

Conduct delay on the input 1, the pulse signal to be processed. Logic delay includes 3 modes, ON delay, OFF delay and bidirectional delay. The module automatically filters out the pulse signals that are shorter than the delay time.



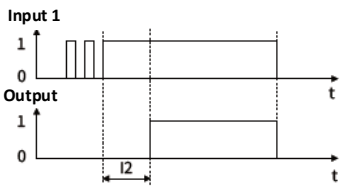
Input 1: Logic signal to be processed; Input 2 : Delay time; Input 3: Mode selection, 0 for ON delay, 1 for OFF delay, and 2 for bidirectional delay.

(2) Module

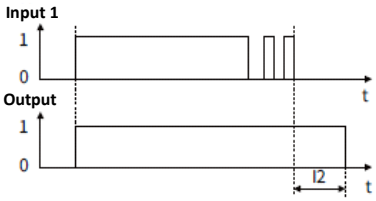
Module Interface	Type	Comment
Input 1	Uint16	Logic signal to be processed and input type are constant/address numbers
Input 2	Uint16	Delay time and input type are constant/address numbers
Input 3	Uint16	Mode and input type are constant/address numbers
Output	Uint32	
Status	0/1	1(illegal input, setting error)

Note:

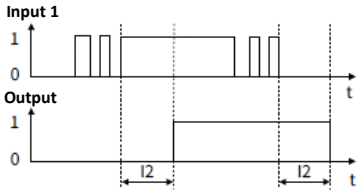
- When both Input 1 and 2 are valid, the filter function is normal, otherwise the module output is always 0.
- When the value of module input 1 is non-0 and non-1, the module status is 1 and the output is 0. At this time, the module input is illegal, and the host computer visual programming module turns red to indicate the module abnormalities.
- When the value of module input 3 is greater than or equal to 3, the module status is 1 and the output is 0. At this time, the module input is incorrect, and the host computer visual programming module turns red to indicate the module abnormalities.
- Upon the delay time end, if it is set to 0 at once, then it will enter the OFF delay processing; when the ON delay time has not yet reached but it is set to 0, this case is not taken as a valid processing.



ON Delay



OFF Delay

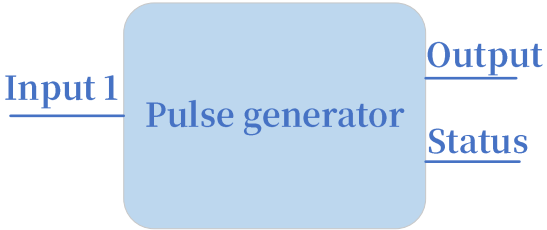


Bidirectional Delay

2.3.11 Pulse Generator

(1) Function

When the pulse generator cycle value is 0, the module continuously outputs low level signal, and when the pulse cycle value is not 0, the module outputs the pulse signal according to the pulse cycle.



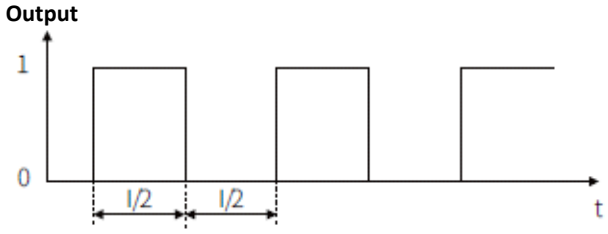
Input 1: Pulse cycle

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Pulse cycle and input type are constant/address numbers
Output	Uint32	
Status	/	/

Note:

- When input1 is valid, the filter function is normal, otherwise the module output is always 0.



2.3.12 Ramp Processing

(1) Function

Conduct slope on the signal to be processed. When Input 3 = 0, Output = Output + sign(Input 1 - Output)* Input 2; When Input 3 = 1, Output = Input 1.



Input 1: Signal to be processed; Input 2: Ramp rate/10ms; Input 3: Module enable.

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Signal to be processed and input type are constant/address numbers
Input 2	Uint16	Ramp rate and input type are constant/address numbers
Input 3	Uint16	Module enable and input type are constant/address numbers
Output	Uint32	
Status	0/1	1(Setting error)

Note:

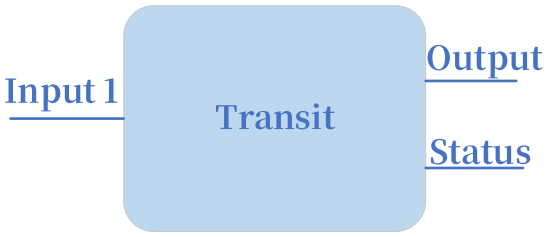
- When all 3 inputs of the module are valid, the module operates normally, otherwise the output of the module is always 0.
- When the value of module input 3 is greater than or equal to 3, the module status is 1 and the output is 0. At

this time, the module input setting is incorrect, and the host computer visual programming module turns red to indicate the module abnormalities.

2.3.13 Transit

(1) Function

Transit the signal to be processed directly to the output without any internal processing.



Input 1: Signal to be processed

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Signal to be processed and input type are constant/address numbers
Output	Uint32	
Status	/	/

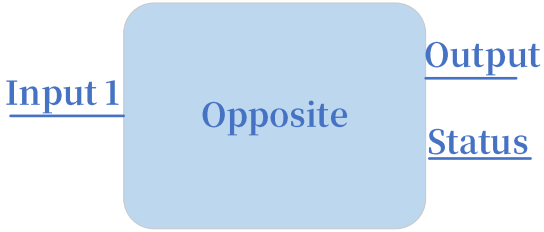
Note:

- When input1 is valid, the filter function is normal, otherwise the module output is always 0.

2.3.14 Opposite Number

(1) Function

Output by converting the signal to be processed to the opposite number.



Output = -Input 1.

Input 1: Signal to be processed

(2) Module

Module Interface	Type	Comment

Input 1	Uint16	Signal to be processed and input type are constant/address numbers
Output	Uint32	
Status	/	/

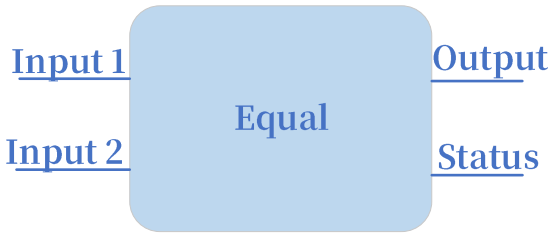
Note:

- When input1 is valid, the filter function is normal, otherwise the module output is always 0.

2.3.15 Equal (EQ)

(1) Function

Conducts equal comparison between the signals to be processed 1 and 2. The module outputs a high level when the signal 1 is equal to the signal 2, otherwise it outputs a low level.



When Input 1= Input 2, output = 1; When Input 1! = Input 2, output = 0;

Input 1: Signal to be processed; Input 2: Signal 2 to be processed.

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Signal 1 to be processed and input type are constant/address numbers
Input 2	Uint16	Signal 2 to be processed and input type are constant/address numbers
Output	Uint32	
Status	/	/

Note:

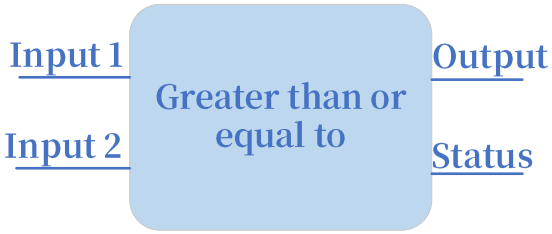
- When both Input 1 and 2 are valid, the module is normal, otherwise the module output is always 0.

2.3.16 Greater Than or Equal To (GE)

(1) Function

Conduct comparison between the signals to be processed 1 and 2 to see if signal 1 is greater than or equal to signal 2. The module outputs a high level when the signal 1 is equal to the signal 2, otherwise it outputs a low

level.



When Input 1= Input 2, output = 1; When Input 1< Input 2, output = 0;

Input 1: Signal to be processed; Input 2: Signal 2 to be processed.

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Signal 1 to be processed and input type are constant/address numbers
Input 2	Uint16	Signal 2 to be processed and input type are constant/address numbers
Output	Uint32	
Status	/	/

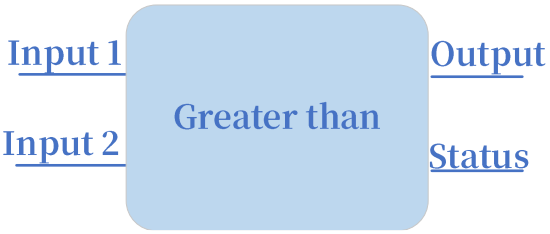
Note:

- When both Input 1 and 2 are valid, the module is normal, otherwise the module output is always 0.

2.3.17 Greater Than (GT)

(1) Function

Conduct comparison between the signals to be processed 1 and 2 to see if signal 1 is greater than signal 2. The module outputs a high level when the signal 1 is greater than the signal 2, otherwise it outputs a low level.



When Input 1> Input 2, output = 1; When Input 1≤ Input 2, output = 0;

Input 1: Signal to be processed; Input 2: Signal 2 to be processed.

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Signal 1 to be processed and input type are

		constant/address numbers
Input 2	Uint16	Signal 2 to be processed and input type are constant/address numbers
Output	Uint32	
Status	/	/

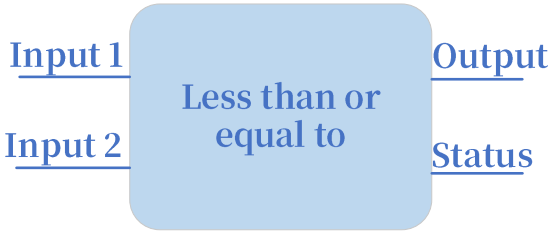
Note:

- When both Input 1 and 2 are valid, the module is normal, otherwise the module output is always 0.

2.3.18 Less Than or Equal To (LE)

(1) Function

Conduct comparison between the signals to be processed 1 and 2 to see if signal 1 is less than or equal to signal 2. The module outputs a high level when the signal 1 is less than or equal to the signal 2, otherwise it outputs a low level.



When Input 1≤ Input 2, output = 1; When Input 1> 2, output = 0;

Input 1: Signal to be processed; Input 2: Signal 2 to be processed.

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Signal 1 to be processed and input type are constant/address numbers
Input 2	Uint16	Signal 2 to be processed and input type are constant/address numbers
Output	Uint32	
Status	/	/

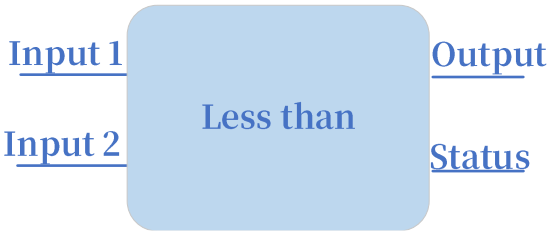
Note:

- When both Input 1 and 2 are valid, the module is normal, otherwise the module output is always 0.

2.3.19 Less Than (LT)

(1) Function

Conduct comparison between the signals to be processed 1 and 2 to see if signal 1 is less than signal 2. The module outputs a high level when the signal 1 is less than the signal 2, otherwise it outputs a low level.



When Input 1 < Input 2, output = 1; When Input 1 ≥ 2, output = 0;

Input 1: Signal to be processed; Input 2: Signal 2 to be processed.

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Signal 1 to be processed and input type are constant/address numbers
Input 2	Uint16	Signal 2 to be processed and input type are constant/address numbers
Output	Uint32	
Status	/	/

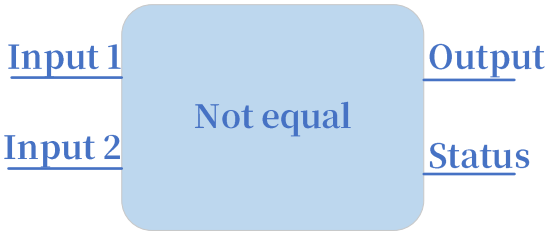
Note:

- When both Input 1 and 2 are valid, the module is normal, otherwise the module output is always 0.

2.3.20 Not Equal (NE)

(1) Function

Conduct comparison between the signals to be processed 1 and 2 to see if they are not equal. The module outputs a high level when the signal 1 is not equal to the signal 2, otherwise it outputs a low level.



When Input 1! = Input 2, output = 1; When Input 1 = Input 2, output = 0;

Input 1: Signal to be processed; Input 2: Signal 2 to be processed.

(2) Module

Module Interface	Type	Comment

Input 1	Uint16	Signal 1 to be processed and input type are constant/address numbers
Input 2	Uint16	Signal 2 to be processed and input type are constant/address numbers
Output	Uint32	
Status	/	/

Note:

- When both Input 1 and 2 are valid, the module is normal, otherwise the module output is always 0.

2.3.21 Pulse Counter

(1) Function

Count the pulse of Input 1 and Input 2. The counter registers 1 for each rising edge of the input 1 pulse signal, and decreases by 1 for each rising edge of the input 2 pulse signal and the counter is set to 0 when the input 3 reset flag is set to 1.



Input 1: CU (counter up); Input 2: CD (counter down); Input 3: RESET

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	CU and input type are constant/address numbers
Input 2	Uint16	CD and input type are constant/address numbers
Input 3	Uint16	RESET and input type are constant/address numbers
Output	Uint32	
Status	0/1	1 (Data overflow, illegal input)

Note:

- When all 3 inputs of the module are valid, the module operates normally, otherwise the output of the module is always 0.
- When the value of module input 1 is non-0 and non-1, the module status is 1 and the output is 0. At this time, the input is illegal, and the host computer visual programming module turns red to indicate the module

abnormalities.

- When the module is continuously counting CU or CD, and the output is greater than 2147483646 or less than -2147483646, the module state is 1. At this time, the module will have a data overflow, and the host computer visual programming module turns red to indicate the module abnormalities (theoretically).

2.3.22 RS or SR

(1) Function

Conduct the logic processing for input 1 (SET) and input 2 (RESET) signals. When the function is 0, it is RS, i.e. input 2 (RESET) has higher priority; when the function is 1, it is SR, i.e. input 1 (SET) has higher priority.



Input 1: SET; Input 2: RESET; Input 3: Function selection, 0 is RS,1 is SR.

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	SET and input type are constant/address numbers
Input 2	Uint16	RESET and input type are constant/address numbers
Input 3	Uint16	Function selection and input type are constant/address numbers
Output	Uint32	
Status	0/1	1(Setting error, illegal input)

Note:

- When all 3 inputs of the module are valid, the module operates normally, otherwise the output of the module is always 0.
- When the value of module input 1 is non-0 and non-1, the module status is 1 and the output is 0. At this time, the module input is illegal, and the host computer visual programming module turns red to indicate the module abnormalities.
- When the value of module input 3 is greater than or equal to 3, the module status is 1 and the output is 0. At this time, the module input setting is incorrect, and the host computer visual programming module turns red to indicate the module abnormalities.

2.3.23 Switching Signal (SEL)

(1) Function

Conduct logic processing on the switching signals. When input 3 = 0, output = Signal to be processed; Input 3=1, output = Signal 2 to be processed.



Input 1: Signal to be processed; Input 2: Signal 2 to be processed. 3: Function selection. 0 means Input 1 and 1 means Input 2.

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Signal 1 to be processed and input type are constant/address numbers
Input 2	Uint16	Signal 1 to be processed and input type are constant/address numbers
Input 3	Uint16	Function selection and input type are constant/address numbers
Output	Uint32	
Status	0/1	1(Setting error)

Note:

- When all 3 inputs of the module are valid, the module operates normally, otherwise the output of the module is always 0.
- When the value of module input 3 is greater than or equal to 3, the module status is 1 and the output is 0. At this time, the module input setting is incorrect, and the host computer visual programming module turns red to indicate the module abnormalities.

2.3.24 Fault Code Monitoring

(1) Function

Monitor Input 1: Fault status of the fault code. The output is 1 if the drive has a current fault, otherwise the output is 0.



Input 1: Fault code.

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Fault code and input type are constant/address numbers
Output	Uint32	
Status	0/1	1(illegal input)

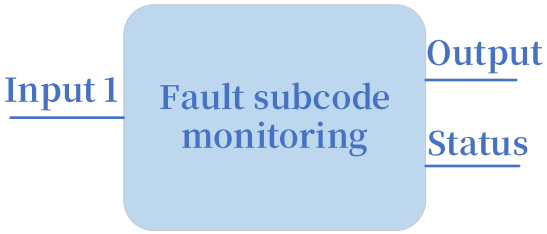
Note:

- When input 1 is valid, the function is normal, otherwise the module output is always 0.
- When the module input value exceeds the fault code value range, that is, the fault code is not in the range of 1~128, the module status is 1 and the output is 0. At this time, the module input is set incorrectly, and the host computer visual programming module turns red to indicate the module abnormalities.

2.3.25 Fault Subcode Monitoring

(1) Function

Monitor Input 1: Fault status of the fault subcode. The output is 1 if the drive has a current fault, otherwise the output is 0.



Input 1: Fault code.

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Fault subcode and input type are constant/address numbers
Output	Uint32	

Status	0/1	1(illegal input)

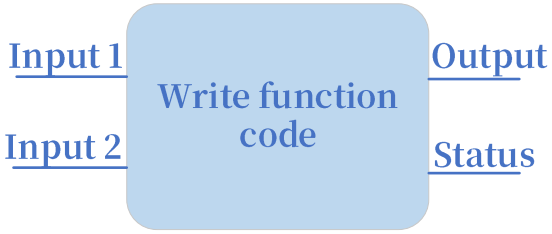
Note:

- When input 1 is valid, the function is normal, otherwise the module output is always 0.
- When the module input value exceeds the fault code value range, that is, the fault code is not in the range of 1~12899, the module status is 1 and the output is 0. At this time, the module input is set incorrectly, and the host computer visual programming module turns red to indicate the module abnormalities.

2.3.26 Write Function Codes

(1) Function

Write input 1 (value) to input 2 (address).



Input 1 (value to be written); input 2 (address to be written in).

(2) Module

Module Interface	Type	Comment
Input 1	Uint16	Value to be written and input type are constant/address numbers
Input 2	Uint16	Address to be written in and input type are constant/address numbers
Output	Uint32	
Status	0/1	1 (illegal address, illegal data, etc., see note below)

Note:

- When both Input 1 and 2 are valid, the module is normal, otherwise the module output is always 0.
- When the value is written successfully, the output of the module is 0 and the status is 0. When the value is not written, the status of the module is 1 and the output is 1~13. At this time, the visual programming module of the host computer turns red to indicate the module abnormalities.
- The meanings of the output values are as follows:

  Output = 0: Writing is completed

  1: Reserved

2: Transmission error

3: CRC parity error

4: Illegal address

5: Invalid data

6: Not for modification during operation

7: System locked

8: Saving to EEPROM

9: Parameter over range

10: Reserved

11: Reserved

12: Monitoring parameter read-only

13: Parameter protection