Logic Developer - PC: FBD Logic Instructions

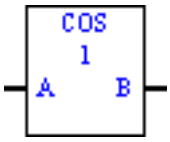
FBD instructions are the building blocks a  Function Block Diagram is composed of. You insert instructions onto the FBD editor to create simple executable units of Logic Developer - PC logic. Each instruction performs an operation on  variables defined for the  target the FBD is associated with.

FBD logic instructions are grouped functionally, that is, according to the type of operation performed. The instruction groups are:

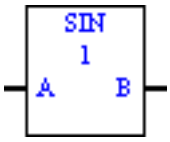
- Advanced Math
- Bitwise Logic
- Comment Block
- Comparison
- Copy
- Counters
- Math
- Program Flow
- Timers
- Type Conversion

FBD Advanced Math Instructions

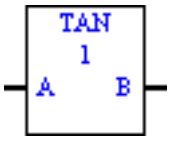
FBD advanced math instructions perform trigonometric and exponential operations on DINT and LREAL parameters.



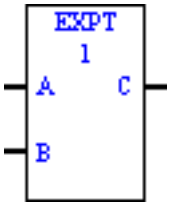
Cos (COS)



Sin (SIN)



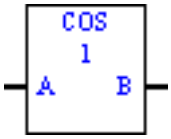
Tan (TAN)



X to the power of Y (EXPT)

See also advanced math instructions in ladder and ST.

Cosine (COS)



The FBD COS instruction calculates the cosine of input parameter **A** and stores the result (angle in radians) in output parameter **B**.

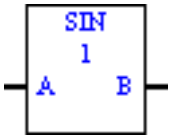
Parameter Data Types

Valid parameter data types for the COS instruction are:

If **A** is:	then **B** is:
LREAL LREAL constant LREAL expression	LREAL

See also COS instructions in ladder and ST.

Sine (SIN)



The FBD SIN instruction calculates the sine of input parameter **A** and stores the result in output parameter **B** (angle in radians).

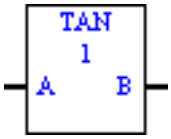
Parameter Data Types

Valid parameter data types for the SIN instruction are:

If **A** is:	then **B** is:
LREAL LREAL constant LREAL expression	LREAL

See also SIN instructions in ladder and ST.

Tangent (TAN)



The FBD TAN instruction calculates the tangent of input parameter **A** and stores the result in output parameter **B** (angle in radians).

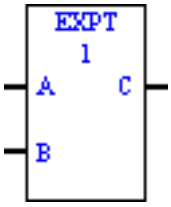
Parameter Data Types

Valid parameter data types for the TAN instruction are:

If **A** is:	then **B** is:
LREAL LREAL constant LREAL expression	LREAL

See also TAN instructions in ladder and ST.

Exponent (EXPT)



The FBD EXPT instruction calculates the exponent of input parameter **A** to the power of input parameter **B**, and stores the result in output parameter **C**.

Parameter Data Types

Valid parameter data types for the EXPT instruction are:

If **A**, **B** are:	then **C** is:
DINT DINT constant DINT expression	DINT
LREAL LREAL constant LREAL expression	LREAL

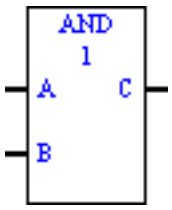
Notes

- **A**, **B**, and **C** must be the same data type.
- If the result is too large for C, C is undefined.

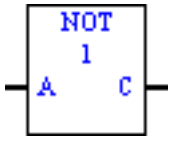
See also EXPT instructions in ladder and ST.

FBD Bitwise Logic Instructions

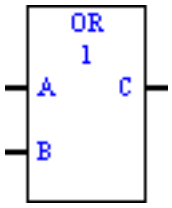
FBD bitwise logic instructions perform logical (BOOL) operations on BOOL and DINT parameters.



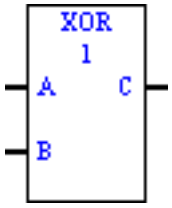
Bitwise AND



Bitwise NOT



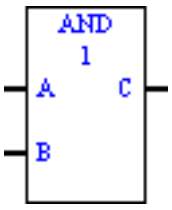
Bitwise OR



Bitwise XOR

See also bitwise logic instructions in ladder and ST.

Bitwise And (AND)



The FBD bitwise AND instruction sets each bit in output parameter **C** to ON(1) if the corresponding bit in input parameter **A** is set to ON(1) AND the corresponding bit in input parameter **B** is set to ON(1). Otherwise, the bit is set to OFF(0).

Truth Table

A AND	**B** =	**C**
1	1	1
1	0	0
0	1	0
0	0	0

Parameter Data Types

Valid parameter data types for the AND instruction are:

If **A**, **B** are:	then **C** is:
BOOL BOOL expression	BOOL
DINT DINT constant DINT expression	DINT

Tip: If all variables are DINTs, a 32-bit AND is performed.

Example

The following example shows the result of two DINTs being ANDed together.

0 = OFF, 1 = ON.

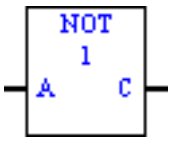
A = 0 1 1 0 . . . 0 1 0 0
B = 0 1 0 0 . . . 0 0 0 0
C = 0 1 0 0 . . . 0 0 0 0

Tips

- DINTs used in a bitwise AND should be displayed in binary format because other number formats may present misleading information.
- The PC system variables #ALW_OFF, #ALW_ON, #False, or #True can be used as input.

See also bitwise AND instructions in  ladder and  ST.

Bitwise Not (NOT)



The FBD bitwise NOT instruction sets each bit in output parameter **C** to ON(1) if the corresponding bit in input parameter **A** is set to OFF(0), and vice versa.

Truth Table

A NOT =	**C**
1	0
0	1

Parameter Data Types

Valid parameter data types for the NOT instruction are:

If **A**, **B** are:	then **C** is:
BOOL BOOL expression	BOOL
DINT DINT constant DINT expression	DINT

Tip: If both variables are DINTs, a 32-bit NOT is performed.

Example

The following example shows the result when a DINT is NOTed. 0=OFF, 1=ON

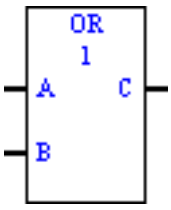
A = 0 1 1 0 ... 1 1 0 0
C = 1 0 0 1 ... 0 0 1 1

Tips

- DINTs used in a bitwise NOT should be displayed in binary format because other number formats may present misleading information.
- The PC system variables #ALW_OFF, #ALW_ON, #False, or #True can be used as input.

See also bitwise NOT instructions in  ladder and  ST.

Bitwise Or (OR)



The FBD bitwise OR instruction sets each bit in output parameter **C** to ON(1) if the corresponding bit in input parameter **A** is set to ON(1), OR, the corresponding bit in input parameter **B** is set to ON(1). Otherwise, the bit is set to OFF(0).

Truth Table

A OR **B** = **C**		
1	1	1
1	0	1
0	1	1
0	0	0

Parameter Data Types

Valid parameter data types for the bitwise OR instruction are:

If **A**, **B** are:	then **C** is:
BOOL BOOL expression	BOOL
DINT DINT constant DINT expression	DINT

Tip: If all variables are DINTs, a 32-bit OR is performed.

Example

The following example shows the result of two DINTs being ORed together. 1 = ON, 0 = OFF

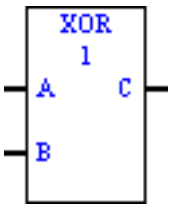
A = 0 1 1 0 ... 1 1 0 0
B = 1 1 0 0 ... 0 0 0 1
C = 1 1 1 0 ... 1 1 0 1

Tips

- DINTs used in a bitwise OR should be displayed in binary format because other number formats may present misleading information.
- The PC system variables #ALW_OFF, #ALW_ON, #False, or #True can be used as input.

See also Bitwise OR instructions in  ladder and  ST.

Bitwise Xor (XOR)



The FBD bitwise exclusive OR (XOR) instruction sets each bit in output parameter **C** to ON(1) if the corresponding bit in input parameter **A** is set to ON(1), or the corresponding bit in input parameter **B** is set to ON(1), but not both. Otherwise, the bit in **C** is set to OFF(0).

Truth Table

A XOR	**B** =	**C**
1	1	0
1	0	1
0	1	1
0	0	0

Tip: If all variables are DINTs, a 32-bit XOR is performed.

Parameter Data Types

Valid parameter data types for the bitwise XOR instruction are:

If **A**, **B** are:	then **C** is:
BOOL BOOL expression	BOOL
DINT DINT constant DINT expression	DINT

Example

The following example shows the result of two DINTs being XORed together. 0 = OFF, 1 = ON

A = 0 1 1 0 ... 1 1 0 0
B = 1 1 0 0 ... 1 0 0 1
C = 1 0 1 0 ... 0 1 0 1

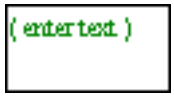
Tips

- DINTs used in a bitwise XOR should be displayed in binary format because other number formats may present misleading information.
- The PC system variables #ALW_OFF, #ALW_ON, #False, or #True can be used as input.

See also bitwise XOR instructions in  ladder and  ST.

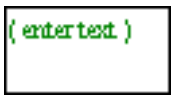
FBD Comment Blocks

You can add one or more FBD text blocks to your FBD. Text blocks are for documentation only. They have no inputs and no outputs.



Text

Text (TEXT)



An FBD text block:

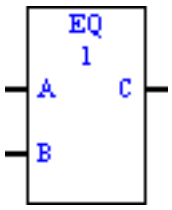
- Is used for documentation only.
- Can be placed anywhere on the FBD editor.
- Has no inputs and no outputs.
- Is ignored by the FBD compiler and by the run time Controller.

Notes

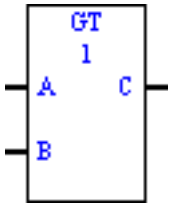
- Double-click a text box to edit its text.
- Single-click a text block to display its handles. To resize the text block, drag a handle with the mouse pointer.
- See also inserting a text box and resizing a text box.
- The number of text boxes that can be added to your FBD is limited only by the amount of memory on your computer.

FBD Comparison Instructions

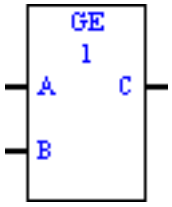
FBD comparison instructions compare the values of two DINT parameters or two LREAL parameters.



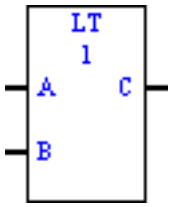
Equal (EQ)



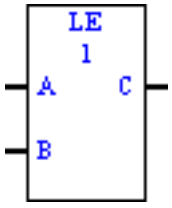
Greater than (GT)



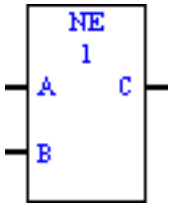
Greater than or Equal (GE)



Less than (LT)



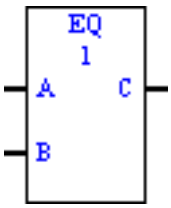
Less than or Equal (LE)



Not Equal (NE)

See also comparison instructions in ladder and relational instructions in ST.

Equal (EQ)



The FBD EQ instruction compares input parameter **A** to input parameter **B**. If they are exactly equal, output parameter **C** is set to ON(1). If **A** is not exactly equal to **B**, **C** is set to OFF(0).

Parameter Data Types

Valid parameter data types for the EQ instruction are:

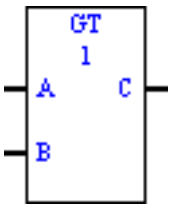
If **A**, **B** are:	then **C** is:
DINT DINT constant DINT expression LREAL LREAL constant LREAL expression	BOOL

Note: A and **B** must be the same data type.

Warning: Comparing LREAL values may produce unexpected results. For example, a calculation may result in 1.99999999999, which is not equal to 2.00000000000.

See also EQ instructions in ladder and relational instructions in ST.

Greater than (GT)



The FBD GT instruction compares input parameter **A** to input parameter **B**. If **A** is greater than **B**, output parameter **C** is set to ON(1). If **A** is less than or equal to **B**, **C** is set to OFF(0).

Parameter Data Types

Valid parameter data types for the GT instruction are:

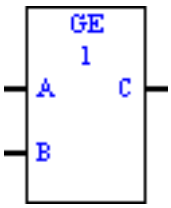
If **A**, **B** are:	then **C** is:
DINT DINT constant DINT expression LREAL LREAL constant LREAL expression	BOOL

Note: A and **B** must be the same data type.

Warning: Comparing LREAL values may produce unexpected results. For example, a calculation may result in 1.99999999999, which is not equal to 2.00000000000.

See also GT instructions in ladder and relational instructions in ST.

Greater than or Equal (GE)



The FBD GE instruction compares input parameter **A** to input parameter **B**. If **A** is greater than or equal to **B**, output parameter **C** is set to ON(1). If **A** is less than **B**, **C** is set to OFF(0).

Parameter Data Types

Valid parameter data types for the GE instruction are:

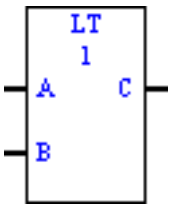
If **A**, **B** are:	then **C** is:
DINT DINT constant DINT expression LREAL LREAL constant LREAL expression	BOOL

Note: A and **B** must be the same data type.

Warning: Comparing LREAL values may produce unexpected results. For example, a calculation may result in 1.99999999999, which is not equal to 2.00000000000.

See also GE instructions in ladder and relational instructions in ST.

Less than (LT)



The FBD LT instruction compares input parameter **A** to input parameter **B**. If **A** is less than **B**, output parameter **C** is set to ON(1). If **A** is greater than or equal to **B**, **C** is set to OFF(0).

Parameter Data Types

Valid parameter data types for the LT instruction are:

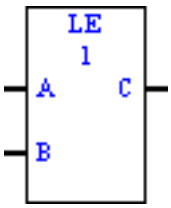
If **A**, **B** are:	then **C** is:
DINT DINT constant DINT expression LREAL LREAL constant LREAL expression	BOOL

Note: A and **B** must be the same data type.

Warning: Comparing LREAL values may produce unexpected results. For example, a calculation may result in 1.99999999999, which is not equal to 2.00000000000.

See also LT instructions in ladder and relational instructions in ST.

Less than or Equal (LE)



The FBD LE instruction compares input parameter **A** to input parameter **B**. If **A** is less than or equal to **B**, output parameter **C** is set to ON(1). If **A** is greater than **B**, **C** is set to OFF(0).

Parameter Data Types

Valid parameter data types for the LE instruction are:

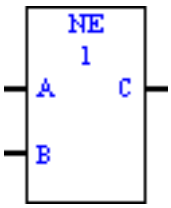
If **A**, **B** are:	then **C** is:
DINT DINT constant DINT expression LREAL LREAL constant LREAL expression	BOOL

Note: A and **B** must be the same data type.

Warning: Comparing LREAL values may produce unexpected results. For example, a calculation may result in 1.99999999999, which is not equal to 2.00000000000.

See also LE instructions in ladder and relational instructions in ST.

Not Equal (NE)



The FBD NE instruction compares input parameter **A** to input parameter **B**. If **A** is not exactly equal to **B**, output parameter **C** is set to ON(1). If **A** is exactly equal to **B**, **C** is set to OFF(0).

Parameter Data Types

Valid parameter data types for the NE instruction are:

If **A**, **B** are:	then **C** is:
DINT DINT constant DINT expression LREAL LREAL constant LREAL expression	BOOL

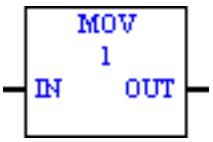
Note: A and **B** must be the same data type.

Warning: Comparing LREAL values may produce unexpected results. For example, a calculation may result in 1.99999999999, which is not equal to 2.00000000000.

See also NE instructions in ladder and relational instructions in ST.

FBD Move Instruction

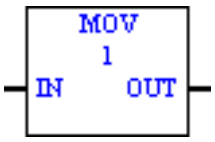
The FBD move instruction copies data from one parameter to another parameter.



Move (MOV)

See also move (copy) instructions in ladder.

Move (MOV)



The FBD MOV instruction copies input parameter **IN** to output parameter **OUT**.

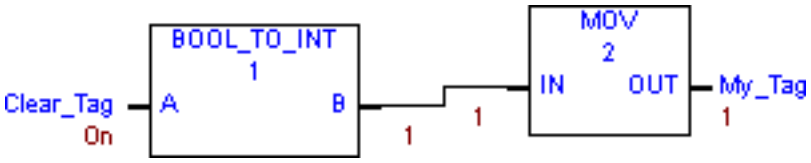
Parameter Data Types

Valid parameter data types for the MOV instruction are:

If **IN** is:	then **OUT** is:
BOOL, BOOL expression	BOOL
BOOL array-size n	BOOL array-size n
DINT, DINT constant, DINT expression	DINT
DINT array-size n	DINT array-size n
LREAL, LREAL constant, LREAL expression	LREAL
LREAL array-size n	LREAL array-size n

Example

A variable can be cleared by using the MOV instruction to move a zero or one into it.



When 'Clear_Tag' is set to ON(1), 'My_Tag' is set to one.

When 'Clear_Tag' is set to OFF(0), 'My_Tag' is set to zero.

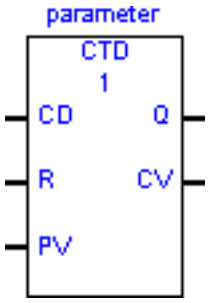
Tips

- The PC system variables #ALW_OFF, #ALW_ON, #False, or #True can also be used as input to the FBD MOV instruction.
- The number inside each rectangle of an FBD instruction displays the solve order of the instruction.

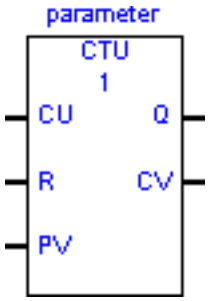
See also the MOV instruction in  ladder.

FBD Counter Instructions

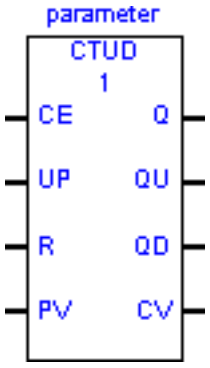
FBD counter instructions count events. They require a structure variable of data type COUNTER for the parameter above the instruction.



Down Counter (CTD)



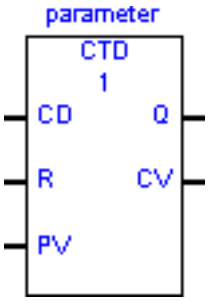
Up Counter (CTU)



Up/Down Counter (CTUD)

See also counter instructions in ladder.

Down Counter (CTD)



For every scan the FBD CTD instruction is executed, it decrements by one, starting from a preset value. The instruction is solved when it has decremented to zero or less.

Parameter Data Types

In the example below, the parameter assigned above the CTD instruction is a COUNTER structure variable named 'MyCounter'. The other parameter data types are:

Element	Description	Data type	Example
CD	Start Down Counter	BOOL	MyStartCounter
PV	Preset value	DINT	MyCounter.PV
CV	Current value	DINT	MyCounter.CV
R	Reset bit	BOOL	MyCounter.R
UP	Counting UP bit	BOOL	MyCounter.UP
QU (n/a)	Done UP bit	BOOL	MyCounter.QU
QD	Done DOWN bit	BOOL	MyCounter.QD
Q	Output enable bit	BOOL	MyCounter.Q

Tip: QU is not used for CTD.

The preset value 'MyCounter.PV' must be set manually or by another instruction solved before the CTD instruction.

Detailed Operation

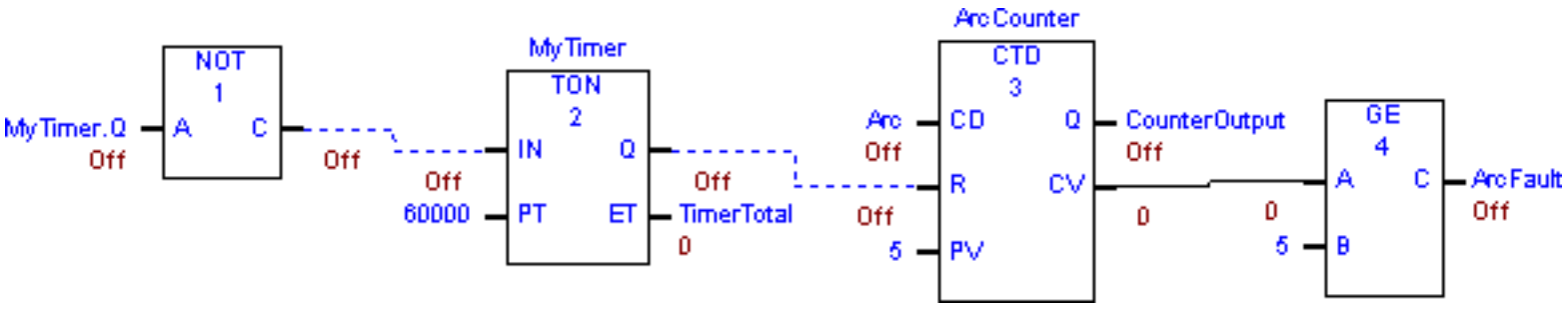
When the FBD CTD instruction executes, the current value 'MyCounter.CV' is decremented by one, and the reset enable bit 'MyCounter.R' is set to OFF(0).

When 'MyCounter.CV' is less than or equal to zero after decrementing, the output enable bit 'MyCounter.Q' is set to ON(1), and the instruction is solved.

When the reset enable bit 'MyCounter.R' is set to ON(1), 'MyCounter.CV' is set to 'MyCounter.PV'.

Example

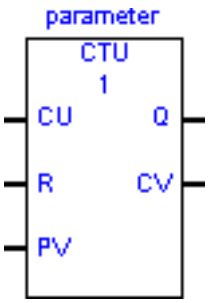
The following example shows how a CTD instruction is used to flag a fault after 5 arcs have been counted in a 1 minute period. 'MyTimer' resets the counter every minute.



Note: COUNTERs increment or decrement with every logic scan when executed. To count events, as in the example above, insert an instruction before the CTD instruction and draw a wire to the input Counter Down connection point (CD) of the FBD CTD instruction.

See also the CTD instruction in ladder.

Up Counter (CTU)



For every scan the FBD CTU instruction is executed, it increments by one. The instruction is solved when it has incremented up to a preset value or greater.

Parameter Data Types

In the example below, the parameter assigned above the CTU instruction is a COUNTER structure variable named 'MyCounter'. The other parameter data types are:

Element	Description	Data type	Example
CU	Start Up Counter	BOOL	MyUpCounter
PV	Preset value	DINT	MyCounter.PV
CV	Current value	DINT	MyCounter.CV
R	Reset bit	BOOL	MyCounter.R
UP	Counting UP bit	BOOL	MyCounter.UP
QU	Done UP bit	BOOL	MyCounter.QU
QD (n/a)	Done DOWN bit	BOOL	MyCounter.QD
Q	Output enable bit	BOOL	MyCounter.Q

Tip: QD is not used for CTU.

The preset value 'MyCounter.PV' must be set manually or by another instruction solved before the CTU instruction.

Detailed Operation

When the FBD CTD instruction executes, the current value 'MyCounter.CV' is incremented by one, and the reset enable bit 'MyCounter.R' is set to OFF(0).

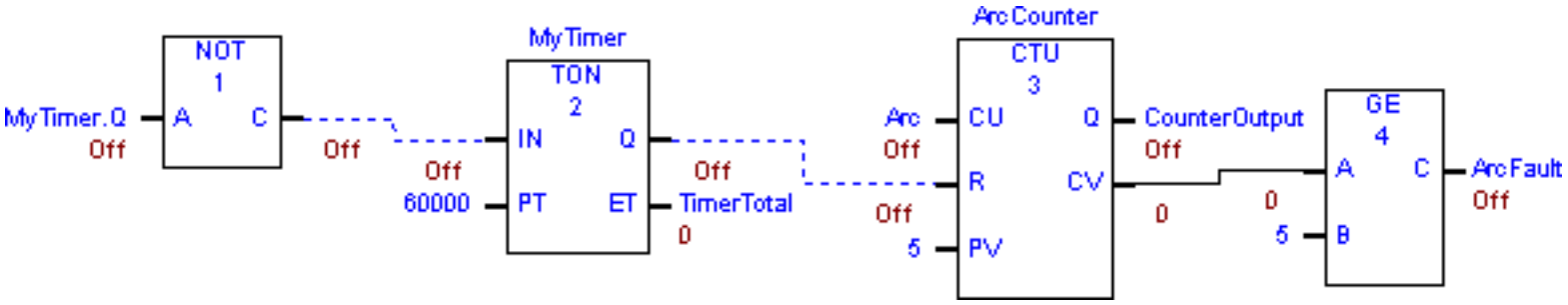
The CTU instruction is solved when the output enable bit 'MyCounter.Q' is set to ON(1) and 'MyCounter.CV' is equal to or greater than the preset value 'MyCounter.PV', after incrementing.

When the reset enable bit 'MyCounter.R' is set to ON(1), 'MyCounter.CV' is set to zero.

Example

In the following example, the CTU flags a fault after 5 arcs have been counted in a 1 minute (60,000 milliseconds)period.

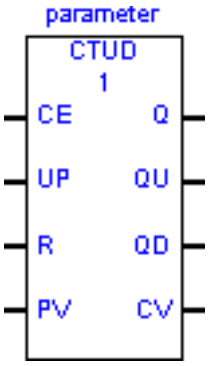
'MyTimer.Q' is set to ON(1) for one scan each minute, causing the reset bit 'ArcCounter.R' to be set to ON(1), thus zeroing 'ArcCounter.CV'. If an arc occurred during this scan, 'Arc' is set to ON(1) and the CTU instruction begins to be solved. If five arcs are counted within a minute, the CTU instruction is solved, causing 'ArcFault' to be set to ON(1).



Note: COUNTERs increment or decrement with every logic scan when being solved. To count events, as in the example above, insert an instruction before the CTU instruction and draw a wire to the input Counter Up connection point (CU) of the CTD instruction.

See also the CTU instruction in ladder.

Up/Down Counter (CTUD)



For every scan the FBD CTUD instruction is executed, it either increments or decrements by one. The CTUD instruction behaves like an FBD Up Counter when **UP** is set to ON(1), otherwise it behaves like an FBD Down Counter.

Parameter Data Types

In the example below, the parameter assigned above the CTUD instruction is a COUNTER structure variable named 'MyCounter'. The other parameter data types are:

Element	Description	Data type	Example
CE	Start Up Down Counter	BOOL	MyUpDownCounter
PV	Preset value	DINT	MyCounter.PV
CV	Current value	DINT	MyCounter.CV
R	Reset bit	BOOL	MyCounter.R
UP	Counting UP bit	BOOL	MyCounter.UP
QU	Done UP bit	BOOL	MyCounter.QU
QD	Done DOWN bit	BOOL	MyCounter.QD
Q	Output enable bit	BOOL	MyCounter.Q

The preset value 'MyCounter.PV' must be set manually or by another instruction solved before the CTUD instruction.

Detailed Operation

When the counting up bit 'MyCounter.UP' is set to ON(1):

- If the current value 'MyCounter.CV', after incrementing, is equal to or greater than the preset value 'MyCounter.PV', then 'MyCounter.Q' and 'MyCounter.QU' are set to ON(1).
- When the reset enable bit 'MyCounter.R' is set to ON(1), 'MyCounter.CV' is set to zero.

When 'MyCounter.UP' is set to OFF(0) (the CTUD instruction is in down mode):

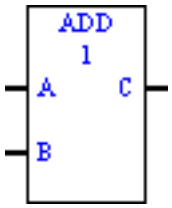
- If the current value 'MyCounter.CV', after decrementing, is equal to or less than zero, then 'MyCounter.Q' and 'MyCounter.QD' are set to ON(1).
- When the reset enable bit 'MyCounter.R' is set to ON(1), 'MyCounter.CV' is set to 'MyCounter.PV'.

Note: COUNTERs increment or decrement with every logic scan when being executed. To count events, as in the example above, insert an FBD instruction before the CTUD instruction and draw a wire to the input **UP** bit connection point of the CTUD instruction.

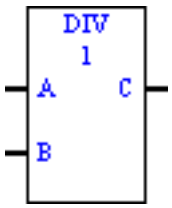
See also the CTUD instruction in  ladder.

FBD Basic Math Instructions

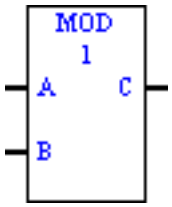
FBD basic math instructions perform basic arithmetic operations on DINT and LREAL parameters.



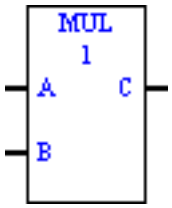
Add (ADD)



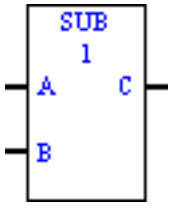
Divide (DIV)



Modulus (MOD)



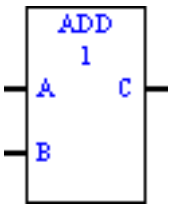
Multiply (MUL)



Subtract (SUB)

See also basic math instructions in ladder and ST.

Add (ADD)



The FBD ADD instruction adds input parameter **A** to input parameter **B**, placing the sum in output parameter **C**.

Parameter Data Types

Valid parameter data types for the ADD instruction are:

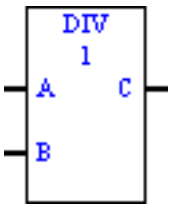
If A, B are:	**then C is:**
DINT DINT constant DINT expression	DINT
LREAL LREAL constant LREAL expression	LREAL

Notes

- **A**, **B**, and **C** must be the same data type.
- If the result is too large for **C** (the destination variable), #Overflow is set to ON(1), and the result is undefined.

See also ADD instructions in ladder and ST.

Divide (DIV)



The FBD DIV instruction divides input parameter **A** by input parameter **B**, placing the quotient in output parameter **C**.

Parameter Data Types

Valid parameter data types for the DIV instruction are:

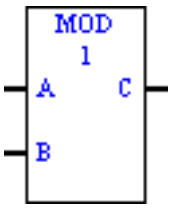
If **A**, **B** are:	then **C** is:
DINT DINT constant DINT expression	DINT
LREAL LREAL constant LREAL expression	LREAL

Notes

- **A**, **B**, and **C** must be the same data type.
- If **A** and **B** are DINT and the result is LREAL, the decimal portion of the result **C** is truncated.
  In this case, if the result of a divide is the LREAL value 2.99, the result becomes the DINT value 2.
- Upon division by zero, #Overflow is set to ON(1) and the result **C** (the destination variable) is undefined.

See also DIV instructions in ladder and ST.

Modulus (MOD)



The FBD MOD instruction divides input parameter **A** by input parameter **B**, placing the remainder in output parameter **C**. It performs a DINT mode of operation only.

Parameter Data Types

Valid parameter data types for the MOD instruction are:

If **A**, **B** are:	**C** is:
DINT DINT constant DINT expression	DINT

Example

In the following example, the DINT value 27 is divided by the DINT value 5; the DINT result 2 is placed in **C**.

A = 27
B = 5

$$5\overline{)27}\quad 5,\ 25,\ 2$$

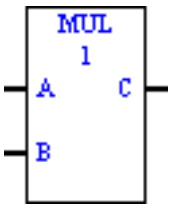
C = 2

Notes

- **A**, **B**, and **C** must be the same data type.
- Upon division by zero, #Overflow is set to ON(1) and the result **C** (the destination variable) is undefined.

See also MOD instructions in ladder and ST.

Multiply (MUL)



The FBD MUL instruction multiplies input parameter **A** by input parameter **B**, placing the product in output parameter **C**.

Variations

If **A** and **B** are DINTs, DINT constants, or DINT expressions, the instruction performs a DINT multiplication. Otherwise, it performs a floating-point (LREAL) multiplication.

Parameter Data Types

Valid parameter data types for the MUL instruction are:

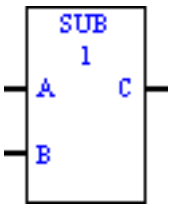
If **A**, **B** are:	then **C** is:
DINT DINT constant DINT expression	DINT
LREAL LREAL constant LREAL expression	LREAL

Notes

- **A**, **B**, and **C** must be the same data type.
- If the result is too large for **C** (the destination variable), #Overflow is set to ON(1), and the result is undefined.

See also MUL instructions in ladder and ST.

Subtract (SUB)



The FBD SUB instruction subtracts input parameter **B** from input parameter **A**, placing the difference in output parameter **C**.

Variations

If both **A** and **B** are DINTs (or DINT constants), the instruction performs a DINT subtraction. If either **A** or **B** are LREALs (or LREAL constants), both are converted to LREALs prior to performing a floating-point subtraction.

Parameter Data Types

Valid parameter data types for the SUB instruction are:

If **A**, **B** are:	then **C** is:
DINT DINT constant DINT expression	DINT
LREAL LREAL constant LREAL expression	LREAL

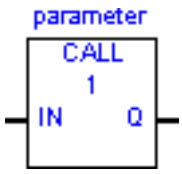
Notes

- **A**, **B**, and **C** must be the same data type.
- If the result is too large for **C** (the destination variable), #Overflow is set to ON(1), and the result is undefined.

See also SUB instructions in ladder and ST.

FBD Program Flow Instructions

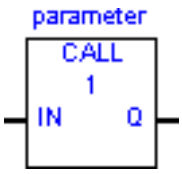
The FBD program flow instruction transfers program control (instruction execution) to another FBD, ladder program, or ST block.



Call

See also program flow instructions in ladder and calling function blocks in ST.

Call (CALL)



The FBD CALL instruction is a control instruction, which enables you to call and transfer control to other Logic Developer - PC logic blocks external to the currently open FBD, in this Machine Edition project. When the called PC logic block (this can be a ladder subroutine, ST block, or FBD) has been solved, control is transferred to the next FBD instruction in the solve order.

Parameter Data Types

Valid parameter data types for the CALL instruction are:

IN must be:	**Q** must be:
BOOL	BOOL

If IN is set to **ON (1)**, the FBD CALL instruction executes; that is, the PC logic block represented by the name above the CALL instruction is executed.

If the input parameter IN is set to **OFF (0)**, the FBD CALL instruction doesn't execute; that is, the PC logic block represented by the name above the CALL instruction isn't called. The next FBD instruction in the solve order is then solved.

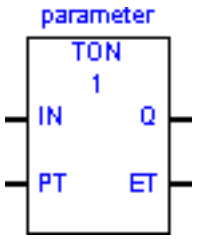
Tips

- When online to the Controller, the value of output parameter Q is set to the value of input parameter IN.
- Recursive calls are allowed; that is, an FBD can call itself.

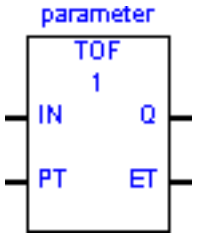
See also CALL instructions in ladder and ST.

FBD Timer Instructions

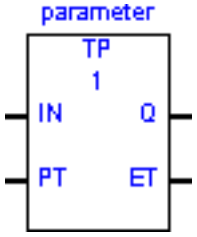
FBD timer instructions time events (in milliseconds). Each FBD timer instruction requires a structure variable of data type TIMER for the parameter above the instruction.



Timer On Delay (TON)



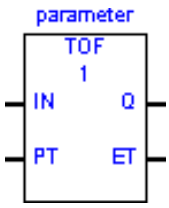
Timer Off Delay (TOF)



Timer Pulse (TP)

See also timer instructions in ladder and ST.

Timer Off Delay (TOF)



The FBD Timer Off Delay (TOF) instruction provides a delay of **PT** milliseconds after the input parameter **IN** is set to OFF(0). The TOF instruction begins to be solved as soon as **IN** is set to ON(1) and continues to be solved until **PT** milliseconds after **IN** has been set to OFF(0).

Parameter Data Types

In the example below, the parameter assigned above the TOF instruction is a TIMER structure variable named 'MyTimer'. The other parameter data types are:

Element	Description	Data type	Example
IN	Timer input	BOOL	#True
PT	Preset time	DINT	MyTimer.PT
ET	Elapsed time	DINT	MyTimer.ET
TI	Timing bit	BOOL	MyTimer.TI
Q	Output enable bit	BOOL	MyTimer.Q

The preset time 'MyTimer.PT' must be set manually or by another instruction solved before the TOF instruction.

Detailed Operation

While the FBD TOF instruction begins to be solved:

- The output elapsed time 'MyTimer.ET' is set to zero.
- The timing bit 'MyTimer.TI' is set to OFF(0).
- The output enable bit 'MyTimer.Q' is set to ON(1) (the instruction has been solved).

When the instruction has been solved:

- 'MyTimer.ET' begins counting upward (in milliseconds).
- 'MyTimer.TI' is set to ON(1).
- The output enable bit 'MyTimer.Q' remains set to ON(1).

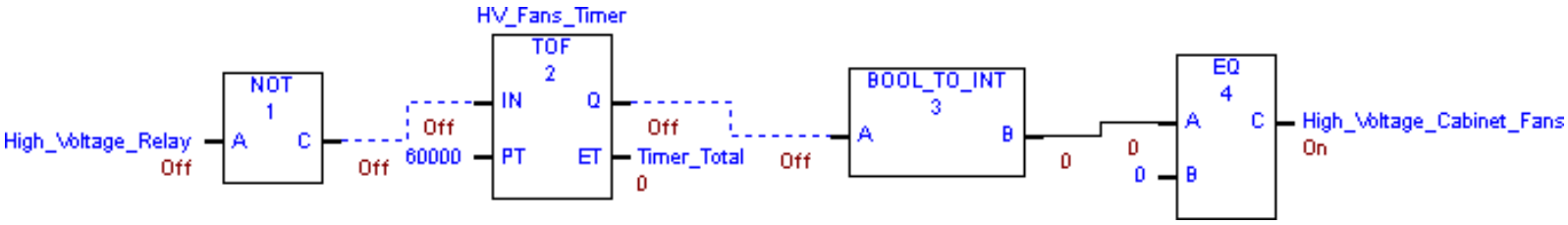
When the elapsed time 'MyTimer.ET' equals the preset time 'MyTimer.PT' after incrementing:

- 'MyTimer.ET' stays fixed at the preset value.
- 'MyTimer.TI' is set to OFF(0).
- 'MyTimer.Q' is set to OFF(0).

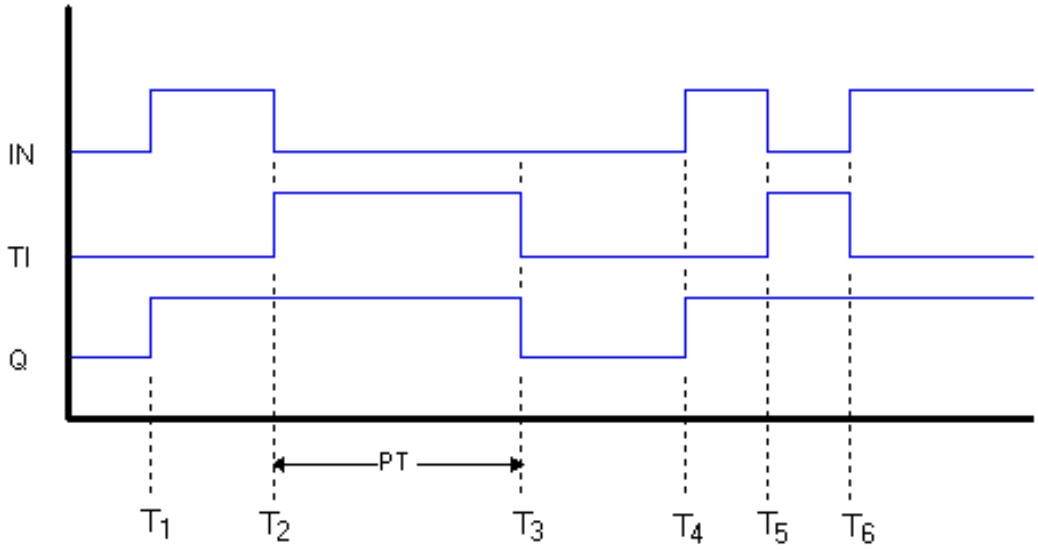
Example

The following FBD diagram illustrates how a Timer Off Delay can be used to keep the fans in a high voltage cabinet running for 1 minute (60,000 ms) after 'High_Voltage_Relay' is set to OFF(0) (as well as keep the fans set to ON(1) while 'High_Voltage_Relay' is set to ON(1)).

As long as 'High_Voltage_Relay' is set to ON(1), the TOF instruction has been solved and 'High_Voltage_Cabinet_Fans' is set to ON(1). When 'High_Voltage_Relay' is set to OFF(0), the TOF instruction begins counting down. One minute later, the TOF instruction has been solved and 'High_Voltage_Cabinet_Fans' is set to OFF(0).



The following timing diagram further illustrates the operation of the TOF instruction.



T_1: The timer input **IN** is set to ON(1), timing bit **TI** remains set to OFF(0), the output enable bit **Q** is set to ON(1), and the elapsed time **ET** is set to zero.

T_2: When the timer input **IN** is set to ON(1), the timer starts timing (**TI** is set to ON(1)), and the output enable bit **Q** remains set to ON(1).

T_3: After elapsed time **ET** equals preset time **PT**, the output enable bit **Q** is set to OFF(0), the timer stops timing (**TI** is set to OFF(0)), and the elapsed time stays fixed at the preset time (**ET**=**PT**).

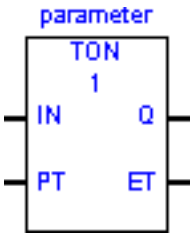
T_4: The timer input **IN** is set to ON(1), the timing bit **TI** remains set to OFF(0), the output enable bit **Q** is set to ON(1), and the elapsed time **ET** is set to zero.

T_5: When the timer input **IN** is set to OFF(0), the timer starts timing (**TI** is set to ON(1)), and the output enable bit **Q** remains set to ON(1).

T_6: The timer input **IN** is set to ON(1) before the elapsed time **ET** equals preset time **PT**, the timer stops timing (**TI** is set to OFF(0)), the output enable bit **Q** remains set to ON(1), and the elapsed time **ET** is set to zero.

See also the TOF instruction in  ladder.

Timer On Delay (TON)



The FBD Timer On Delay (TON) instruction adds a delay of input parameter **PT** (in milliseconds) from the time an event actually occurs; that is, from the time the input parameter **IN** is set to ON(1). TON is solved for **PT** milliseconds after **IN** is set to ON(1).

Parameter Data Types

In the example below, the parameter assigned above the TOF instruction is a TIMER structure variable named 'MyTimer'. The other parameter data types are:

Element	Description	Data type	Example
IN	Timer input	BOOL	#True
PT	Preset time	DINT	MyTimer.PT
ET	Elapsed time	DINT	MyTimer.ET
TI	Timing bit	BOOL	MyTimer.TI
Q	Output enable bit	BOOL	MyTimer.Q

A DINT constant data type can also be assigned to **PT**. The preset time 'MyTimer.PT' can also be set by another FBD instruction solved before the TON instruction.

Detailed Operation

When the FBD TON instruction begins to be solved; that is, **IN** is set to ON(1):

- The elapsed time 'MyTimer.ET' begins counting upward (in milliseconds).
- The timing bit 'MyTimer.TI' is set to ON(1).
- The output enable bit 'MyTimer.Q' is set to OFF(0).

When the elapsed time 'MyTimer.ET' equals the preset time 'MyTimer.PT' after incrementing:

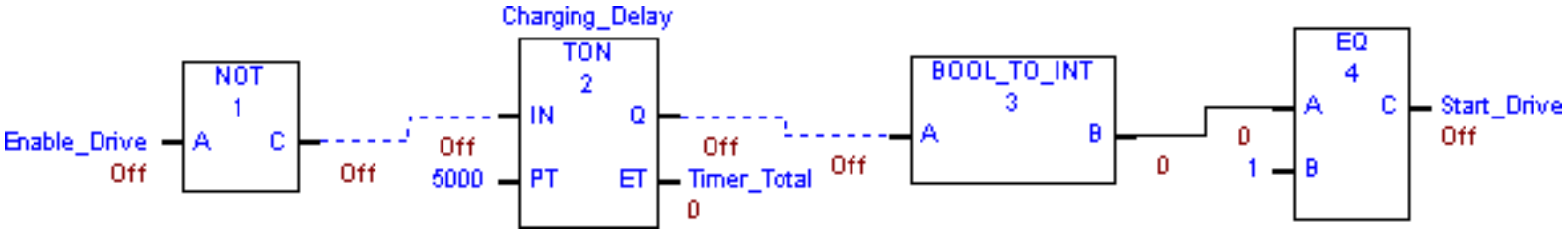
- 'MyTimer.ET' stays fixed at the preset value.
- 'MyTimer.TI' is set to OFF(0).
- 'MyTimer.Q' is set to ON(1); the instruction has been solved.

When the instruction has been solved:

- 'MyTimer.ET' is set to zero.
- 'MyTimer.TI' is set to OFF(0).
- 'MyTimer.Q' is set to OFF(0).

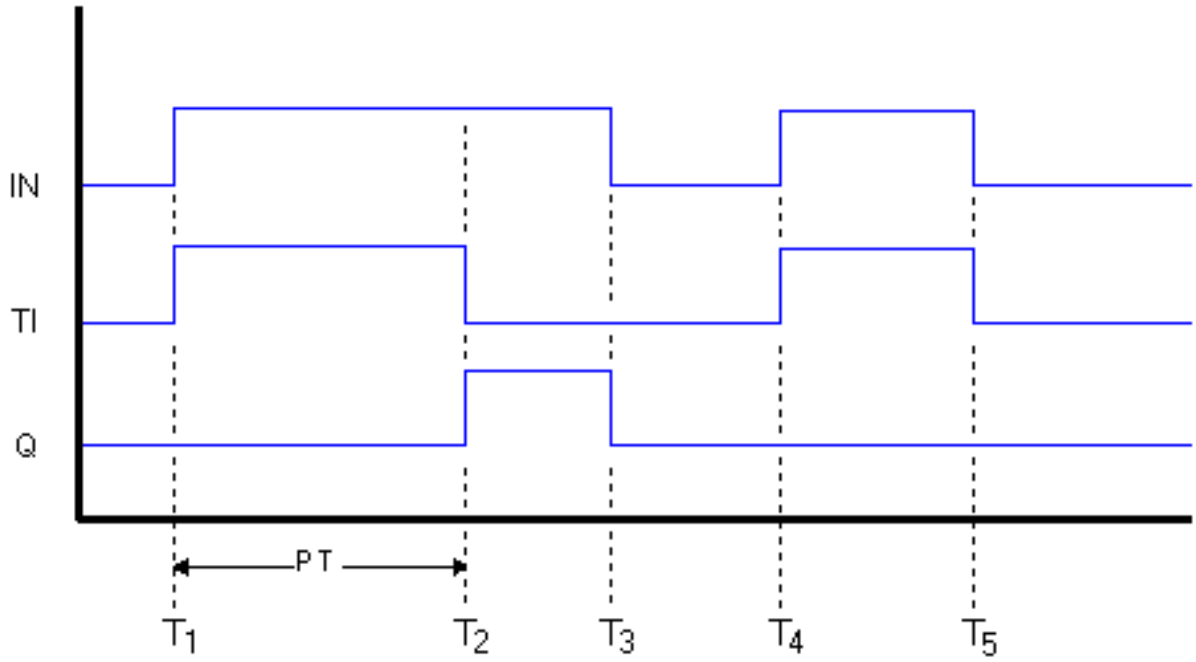
Example

The following FBD illustrates how a Timer On Delay would be used to delay a start signal. A drive needs 5 seconds after it is enabled (being solved) to charge up capacitors before starting.



When 'Enable_Drive' is set to ON(1), the TON instruction starts counting. Five seconds later the TON instruction is solved and 'Start_Drive' is set to ON(1).

The following timing diagram further illustrates the operation of the TON instruction.



T_1: When the FBD TON instruction is being solved, the timer input **IN** and the timing bit **TI** are set to ON(1), and the timer begins timing for **ET** milliseconds. **Q** remains set to OFF(0).

T_2: After time **PT** milliseconds, elapsed time **ET** equals preset time **PT**, the output enable bit **Q** is set to ON(1), and the elapsed time stays fixed at the preset time (**ET**=**PT**).

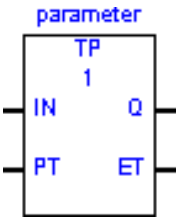
T_3: The timer input **IN** and the output enable bit **Q** are set to OFF(0), and the elapsed time **ET** is set to zero.

T_4: The timer input **IN** and the timing bit **TI** are set to ON(1), and the timer begins timing for **ET** milliseconds.

T_5: The timer input **IN** is set to OFF(0) before the elapsed time **ET** equals the preset time **PT**, the output enable bit **Q** remains set to OFF(0), and the elapsed time **ET** is set to zero.

See also TON instructions in ladder and ST.

Timer Pulse (TP)



The FBD Timer Pulse (TP) instruction outputs a pulse at the output parameter **Q** for a duration of the input parameter **PT** (in milliseconds) when the input parameter **IN** is set to ON(1).

Parameter Data Types

In the example below, the parameter assigned above the TOF instruction is a TIMER structure variable named 'MyTimer'. The other parameter data types are:

Element	Description	Data type	Example
IN	Timer input	BOOL	#True
PT	Preset time	DINT	MyTimer.PT
ET	Elapsed time	DINT	MyTimer.ET
TI	Timing bit	BOOL	MyTimer.TI
Q	Output enable bit	BOOL	MyTimer.Q

The preset time 'MyTimer.PT' must be set manually or by another FBD instruction solved before the TP instruction.

Detailed Operation

When the FBD TP instruction is being solved (**IN** is set to ON(1)):

- The elapsed time 'MyTimer.ET' begins counting upward (in milliseconds).
- The timing bit 'MyTimer.TI' is set to ON(1).
- The output enable bit 'MyTimer.Q' is set to ON(1) when the instruction has been solved.

When the elapsed time 'MyTimer.ET' equals the preset time 'MyTimer.PT' after timing:

- 'MyTimer.ET' stays fixed at the preset value if **IN** is still set to ON(1). If not, 'MyTimer.ET' is immediately set to zero.
- 'MyTimer.TI' is set to OFF(0).
- 'MyTimer.Q' is set to OFF(0); the instruction has been solved.

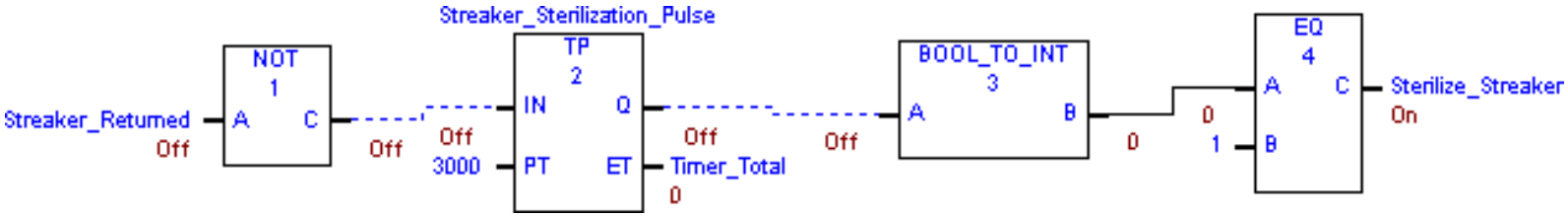
When the instruction has been solved:

- 'MyTimer.ET' is set to zero only if it has already reached the value of the preset time 'MyTimer.PT'. Otherwise, it keeps timing and 'MyTimer.Q' remains set to ON(1).

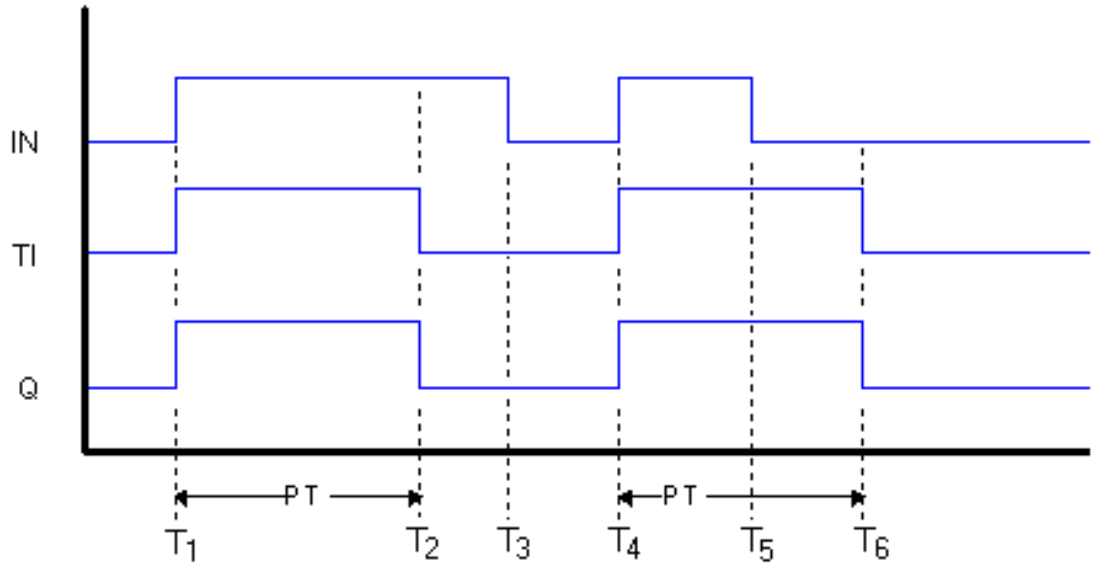
Example

The following example demonstrates how a Timer Pulse (TP) instruction is used to set an output to ON(1) for a three-second pulse. The Streaker is sterilized for three seconds after it returns to its home position.

When 'Streaker_Returned' is set to ON(1), the TP instruction has been solved, and 'Sterilize_Streaker' is set to ON(1). Three seconds later, 'MyTimer.Q' and 'Sterilize_Streaker' are set to OFF(0).



The following timing diagram further illustrates the operation of the TP instruction.



T_1: The timer input **IN** is set to ON(1), the timer starts timing (**TI** is set to ON(1)), and the output enable bit **Q** is set to ON(1).

T_2: After the elapsed time **ET** equals the preset time **PT**, the output enable bit **Q** is set to OFF(0), the timer stops timing (**TI** is set to OFF(0)), and the elapsed time stays fixed at the preset time (**ET**=**PT**).

T_3: The timer input **IN** is set to OFF(0), and the elapsed time **ET** is set to zero.

T_4: The timer input **IN** is set to ON(1), the timer starts timing (**TI** is set to ON(1)), and the output enable bit **Q** remains set to ON(1).

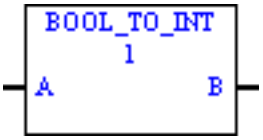
T_5: The timer input **IN** is set to OFF(0), the timer keeps timing (**TI** stays ON(1)), and the output enable bit **Q** remains set to ON(1).

T_6: After time **PT** elapsed time **ET** equals the preset time **PT**, the output enable bit **Q** is set to OFF(0), the TP timer stops timing (**TI** is set to OFF(0)), and the elapsed time **ET** is set to zero because the timer input **IN** is set to OFF(0).

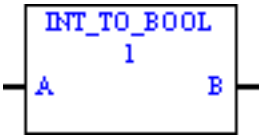
See also the TP instruction in  ladder.

FBD Type Conversion Instructions

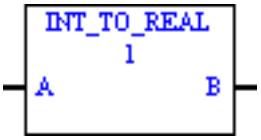
FBD type conversion instructions convert parameters to different data types.



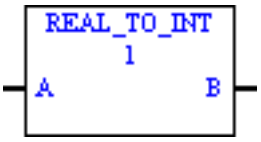
Bool to Dint (BOOL_TO_INT)



Dint to Bool (INT_TO_BOOL)



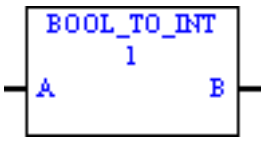
Dint to Lreal (INT_TO_REAL)



Lreal to Dint (REAL_TO_INT)

See also type conversion instructions in ladder and ST.

Bool to Dint (BOOL_TO_INT)



The FBD BOOL_TO_INT instruction converts a single BOOL value at input parameter **A** to a DINT value at output parameter **B**.

Parameter Data Types

Valid parameter data types for the BOOL_TO_INT instruction are:

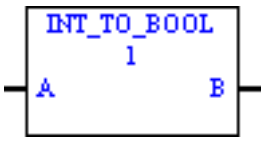
A	**B**
BOOL	DINT

Notes

- The PC system variables #ALW_OFF, #ALW_ON, #False, or #True can be used as input.
- The FBD INT TO BOOL instruction converts a DINT value to a BOOL value.

See also BOOL_TO_INT instructions in ladder and ST.

Dint to Bool (INT_TO_BOOL)



The FBD INT_TO_BOOL instruction converts a DINT value at input parameter **A** to a BOOL value at output parameter **B**.

Parameter Data Types

Valid parameter data types for the INT_TO_BOOL instruction are:

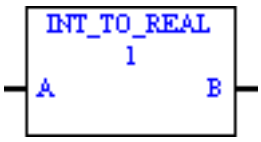
A	**B**
DINT	BOOL

Notes

- If **A** is zero, **B** is set to OFF(0); if **A** is not zero, **B** is set to ON(1).
- The FBD BOOL_TO_INT instruction converts a BOOL value to a DINT value.

See also INT_TO_BOOL instructions in ladder and ST.

Dint to Lreal (INT_TO_REAL)



The FBD INT_TO_REAL instruction converts a DINT value at input parameter **A** to an LREAL value at output parameter **B**.

Parameter Data Types

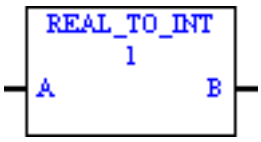
Valid parameter data types for the INT_TO_REAL instruction are:

A	**B**
DINT	LREAL

Note: The FBD REAL_TO_INT instruction converts an LREAL value to a DINT value.

See also the INT_TO_REAL instruction in ST.

Lreal to Dint (REAL_TO_INT)



The FBD REAL_TO_INT instruction converts an LREAL value at input parameter **A** to a DINT value at output parameter **B**.

All values to the right of the decimal point of the result **B** (the destination variable) are truncated. For example, the LREAL value 2.99 becomes the DINT value 2 after the conversion.

A	**B**
LREAL	DINT

Notes

- If the result is too large for **B**, the result **B** is undefined.
- The FBD INT_TO_REAL instruction converts a DINT value to an LREAL value.

See also the REAL_TO_INT instruction in ST.