Boolean Types

Original instructions
See: Related Topics

About us

At a Glance

There are three types of boolean:

- BOOL type, which contains only the value FALSE (=0) or TRUE (=1).
- EBOOL type, which contains the value FALSE (=0) or TRUE (=1) but also information concerning the management of falling or rising edges and forcing.
- ANY_BOOL type, only declared as a referenced data type that combines BOOL and EBOOL types.

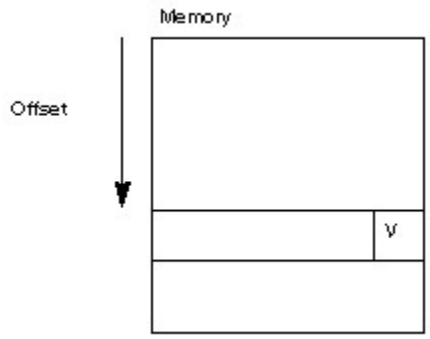
Principle of the BOOL Type

This type takes up one memory byte, but the value is only stored in 1 bit.

The default value for this type is FALSE (=0).

It is accessible via an address containing the offset of the corresponding byte:

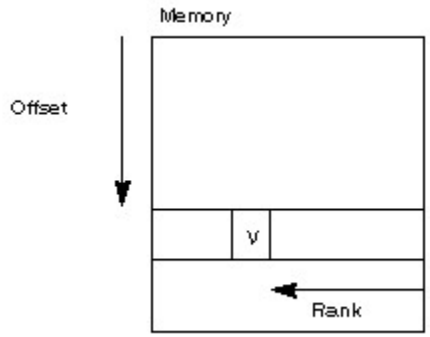
Address settings:



In the case **of the word extracted bit**, it is accessible via an address containing the following information:

- an offset of the corresponding byte
- the rank defining its position in the word

Address settings:



Principle of the EBOOL Type

This type takes up one memory byte which contains:

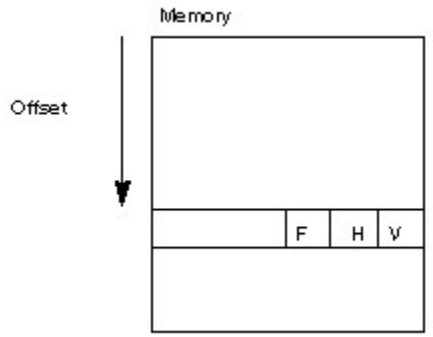
- the bit for the value (V),
- the history bit (H) for managing rising or falling edges. Each time the object status changes, the value is copied to this bit,
- the bit containing the forcing status (F). Equal to 0 if the object is not forced and equal to 1 if the object

is forced.

The default value for the bits associated with the EBOOL type is FALSE (=0).

It is accessible via an address specifying the offset of the corresponding byte:

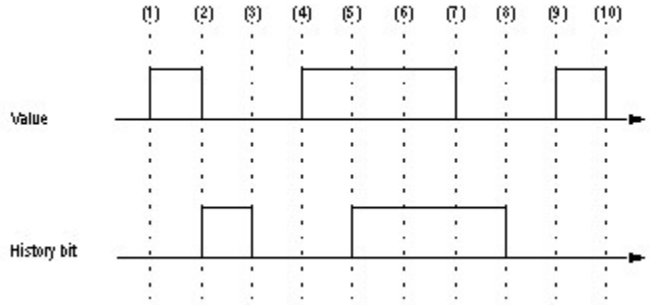
Address settings:



EBOOL Type Historical Trend Diagram

The trend diagram below shows the main statuses of the value and history bits associated with the EBOOL type.

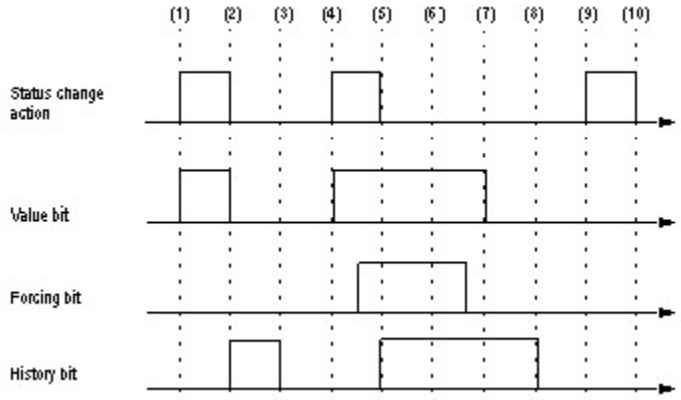
The rising edges of the value bit (1, 4) are copied to the history bit in the next PLC cycle (2, 5). The falling edges of the value bit (2, 7) are copied to the history bit of the next PLC cycle (3, 8).



EBOOL Type Trend Diagram and Forcing

The trend diagram below shows the main statuses of the value, history, and forcing bits associated with the EBOOL type.

The rising edges of the value bit (1, 4) are copied to the history bit in the next PLC cycle (2, 5). The falling edges of the value bit (2, 7) are copied to the history bit in the next PLC cycle (3, 8). Between (4 and 5), the forcing bit equals 1 while the value and history bits remain at 1.



Principle of the ANY_BOOL Type

The `ANY_BOOL` type can be used by supervision tools (a SCADA for example) to reserve variables declared

as generic data type. The generic data type is the element shared with Control Expert.
An ANY_BOOL type variable is declared as a reference, using the REF_TO keyword. More details on referencing and dereferencing are provided in the topic on Reference Data Type Declarations.

NOTE: Implicit conversion is allowed on dereferenced ANY_BOOL type variable (BOOL_TO_*).

Usage limitation of ANY_BOOL type:
- The ANY_BOOL type cannot be used to declare a variable in Control Expert application. A variable is declared using a reference to ANY_BOOL type with keyword REF_TO.
- Referencing REF_TO_ANY_BOOL is not allowed in program.
  MyRefToAnyBoolVar := REF(MyVar); is not allowed (whatever MyVar is: BOOL or EBOOL).
- In an EF or EFB, ANY_BOOL type cannot be used to declare a parameter or variable, even as a reference with keyword REF_TO.
- To reference an EBOOL, only the edge history is managed. The forcing functionality is not managed by the ANY_BOOL type when referencing an EBOOL.
- In a SCADA system, the ANY_BOOL type variable is the shared element, the data dictionary provides the final type of the ANY_BOOL reference (BOOL or EBOOL).
- A reference to a reference is not supported.
  Cascading dereference is not supported (for example, MyAnyBool1^MyAnyBool2^.xy is not supported).

Platform: ANY_BOOL type is used on the following platforms:
- Modicon M580 (OS version ≥ V2.00)
- Modicon Quantum 140CPU6•••• (OS version ≥ V3.30)
- Modicon M340 (OS version ≥ V2.70)

Time stamping: An ANY_BOOL reference variable can only be time stamped in *system time stamping* mode if the referenced variable is a constant (IsConstant attribute enabled). The referenced variable can be associated to:
- A BMX ERT 1604 T source.
- A BMX CRA 312 10 source.
- A BME CRA 312 10 source.
- A Modicon M580 CPU source (OS version ≥ V2.00).
- A topological variable (for example %M100).

PLC Variables Belonging to Boolean Types

List of variables

Variable	Type
Internal bit	EBOOL
System bit	BOOL
Word extracted bit	BOOL
%I inputs	
Module error bit	BOOL
Channel error bit	BOOL
Input bit	EBOOL
%Q outputs	
Output bit	EBOOL

Compatibility Between BOOL and EBOOL

The operations authorized between these two types of variables are:

- value copying
- address copying

Copies between types

	BOOL destination	EBOOL destination
BOOL source	Yes	Yes
EBOOL source	Yes	Yes

Compatibility between the parameters of elementary functions (EF)

Effective parameter (external to EF)	Formal BOOL parameter (internal to EF)	Formal EBOOL parameter (internal to EF)
BOOL	Yes	No
EBOOL	In ->Yes In-Out ->No Out ->Yes	Yes

Compatibility between the parameters of block functions (EFB\DFB)

Effective parameter (external to FB)	Formal BOOL parameter (internal to FB)	Formal EBOOL parameter (internal to FB)
BOOL	Yes	In ->Yes In-Out ->No Out -> Yes
EBOOL	In ->Yes In-Out ->No Out -> Yes	Yes

Compatibility between array variables

	ARRAY[i..j) OF BOOL destination	ARRAY[i..j) OF EBOOL destination
ARRAY[i..j) OF BOOL source	Yes	No
ARRAY[i..j) OF EBOOL source	No	Yes

Compatibility between static variables

	BOOL (%MW:xi) direct addressing	EBOOL (%Mi) direct addressing
BOOL (Var:BOOL) declared variable	Yes	No
EBOOL (Var:EBOOL) declared variable	No	Yes

Compatibility

EBOOL data types follow the rules below:

- An EBOOL type variable cannot be passed as a BOOL type input/output parameter.
- EBOOL arrays cannot be passed as ANY type parameters of an FFB.
- BOOL and EBOOL arrays are not compatible for instructing assignment (same rule as for FFB parameters).
- On Quantum:
  - EBOOL type located variables cannot be passed as EBOOL type input/output parameters.

- EBOOL arrays cannot be passed as parameters of a DFB.