

# Chapter 6

## Logical control

### 6.1 Introduction

Figure 6.1 shows a controller with input signals (or variables) and output signals (variables). The controller may be a PLC, or some similar control equipment, cf. Section 3.

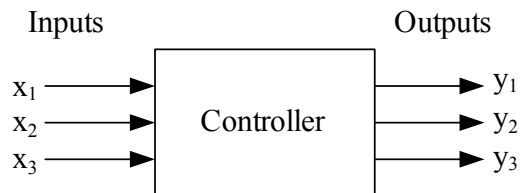


Figure 6.1: A controller with input signals and output signals

The values of the controller outputs may be generated by functions of controller inputs or as a result of e.g. some calculations or comparisons between variables. Here are few examples:

- A motor is started if a start button is pressed.
- A valve is closed if the level of the liquid in a tank is larger than a specific level.

In logical control the output variables have one of two possible binary values. And some of the inputs to the controller may also have binary

values. These binary values are described in the following section.

## 6.2 Logical variables

A logical variable can have one of two possible binary values. These values may be given various names or symbols, as shown in Table 6.1.

TRUE	FALSE
ON	OFF
HIGH	LOW
1	0

Table 6.1: Various names (symbols) on the two possible binary values

The physical representation of these values depends on the equipment or media. Here are some examples:

- **TTL** (transistor-transistor-logic): The binary value FALSE is represented by a voltage in the range [0V, 0.8V]. Value TRUE is any voltage above 2V.
- **A switch or a relay or a transistor:** Value FALSE (or OFF) means the element is open, i.e. does not conduct current. Value TRUE (ON) means the element is closed, i.e. it is conducting current.
- **A valve:** Value TRUE means the valve is open. Value FALSE means the valve is closed.

## 6.3 Logical functions

### 6.3.1 Introduction

In the following sections three basic logical functions operating on logical variables are defined. These functions are

- AND
- OR

- NOT

They are available in the programming tool of any PLC, in Matlab, LabVIEW, C++, Delphi, Visual Basic, etc. By combining these basic functions you can construct more complicated functions. In the following sections it is shown how the functions are defined and documented using a *truth table*, and it will be shown how the functions are represented in

- Boolean expression
- Function block diagram language (FB) defined in the IEC 61131-3 standard of PLC programming. The FB language is a graphical programming language. This means that you develop the program by connecting functions together in a diagram.
- Ladder diagram language (LD) defined in the IEC 61131-3 standard. The LD language is also a graphical programming language.
- Structured text language (ST) defined in the IEC 61131-3 standard. The ST language is a textual language similar to Pascal or Delphi. In addition to the basic boolean functions as AND, OR, NOT etc., you can use expressions as If..Then..Else, Case , While loops, For loops, etc.
- LabVIEW

### 6.3.2 AND function

#### Pseudo code

The AND function may be expressed as follows (in pseudo code) where  $x_1$  and  $x_2$  are inputs and  $y$  is output:

$$y = x_1 \text{ AND } x_2 \quad (6.1)$$

“AND” can also be regarded as an operator.

#### Boolean expression

The AND function is expressed as follows in a boolean expression:

$$y = x_1 * x_2 = x_1 x_2 \quad (6.2)$$

Note that in this context the asterisk operator  $*$  does not mean arithmetical multiplication. As indicated, the asterisk can be omitted.

### Truth table

Table 6.2 shows the *truth table* of the AND function. A truth table defines the output value for all possible combinations of the input values.

$x_1$	$x_2$	$y$
0	0	0
0	1	0
1	0	0
1	1	1

Table 6.2: Truth table of the AND function

### Function block

Figure 6.2 shows a *function block* representing the AND function. This symbol is according to the IEC 61131-3 standard of PLC programming. (There is also a US symbols standard, but US symbols are not shown here.)

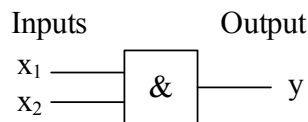


Figure 6.2: Function block symbol of the AND function

### Ladder diagram

Figure 6.3 shows the AND function in a *ladder diagram*. Ladder diagrams are one of the languages defined in the IEC 61131-3 standard. Ladder diagrams depicts a hypothetical electrical circuit diagram. A *contact* can conduct or not conduct current originating from the power rail. For the AND operation, current is flowing to the *coil*  $y$  only if both contacts  $x_1$  and  $x_2$  are conducting. In other words,  $y$  is ON or TRUE only if both  $x_1$  and  $x_2$  are ON or TRUE. “Contact” is used as a synonym for input, while “coil” is a synonym for output.

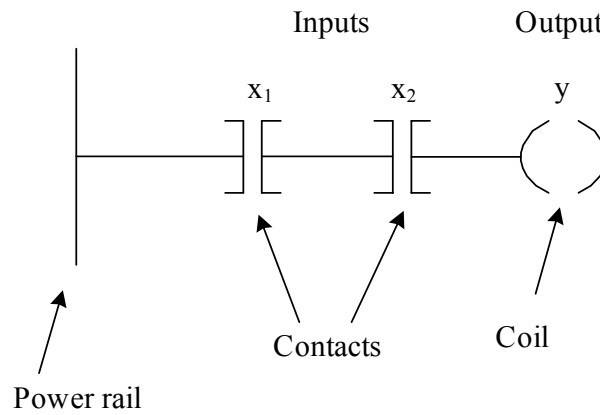


Figure 6.3: The AND function in a ladder diagram

### Structured text

In the Structured text language the AND function can be used as follows:

$$y := x_1 \text{ AND } x_2; \quad (6.3)$$

You can also write  $y := x_1 \& x_2;$ .

### LabVIEW

Figure 6.4 shows the AND function in LabVIEW.

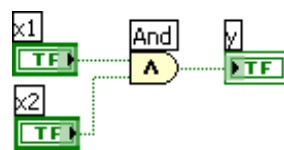


Figure 6.4: The AND function in LabVIEW

### 6.3.3 OR function

#### Pseudo code

$$y = x_1 \text{ OR } x_2 \quad (6.4)$$

**Boolean expression**

$$y = x_1 + x_2 \quad (6.5)$$

Note that the asterix operator  $+$  does not mean arithmetical addition, but logical OR.

**Truth table**

See Table 6.3.

$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	1

Table 6.3: Truth table of the OR function

**Function block**

See Figure 6.5.

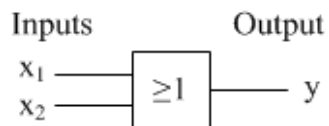


Figure 6.5: Function block symbol of the OR function

**Ladder diagram**

See Figure 6.6.

**Structured text**

In the Structured text language the OR function can be expressed as

$$y := x_1 \text{ OR } x_2; \quad (6.6)$$

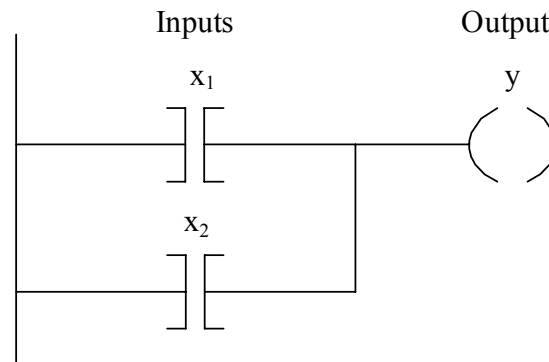


Figure 6.6: The OR function in a ladder diagram

You can also write  $y = x_1 + x_2$ .

### LabVIEW

See Figure 6.7.

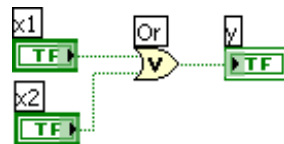


Figure 6.7: The OR function in LabVIEW

### 6.3.4 NOT function

The NOT function has only one input. It is also denoted the logical inverter function.

### Pseudo code

$$y = \text{NOT } x \quad (6.7)$$

**Boolean expression**

$$y = \bar{x} \quad (6.8)$$

**Truth table**

See Table 6.4.

$x$	$y$
0	1
1	0

Table 6.4: Truth table of the NOT function

**Function block**

See Figure 6.8.

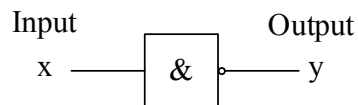


Figure 6.8: Function block symbol of the NOT function

**Ladder diagram**

Figure 6.9 shows how the NOT function is realized using a *normally closed* contact. “Normally closed” means that the contact is closed, i.e. it is not conducting current if you think of it as a real electrical contact, if it is in a “passive” state, i.e. if the variable has value OFF (or FALSE or 0). The contact used in Figure 6.3 is a *normally open* contact.

**Structured text**

In the Structured text language the NOT function is used as follows:

$$y = \text{NOT } x; \quad (6.9)$$



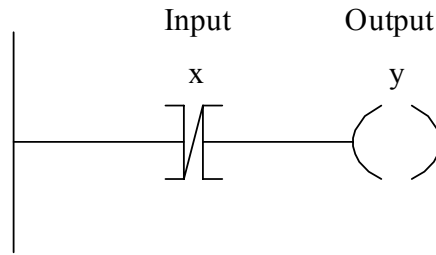


Figure 6.9: The NOT function in a ladder diagram

### LabVIEW

See Figure 6.10.

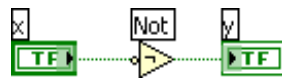


Figure 6.10: The NOT function in LabVIEW

## 6.4 Compound logical functions

Basic logical functions can be combined to realize compound logical functions. Assume as an example the following function:

$$y = (x_1 \text{ AND } x_2) \text{ OR } x_3 \quad (6.10)$$

This function can be realized in a function block diagram as shown in Figure 6.11. And Figure 6.12 shows how the function can be realized in a

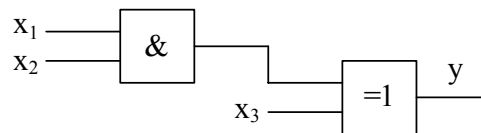


Figure 6.11: Function block diagram realizing the compound function  $y = (x_1 \text{ AND } x_2) \text{ OR } x_3$

ladder diagram.

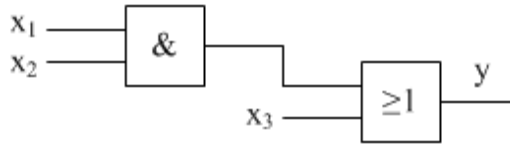


Figure 6.12: Ladder diagram realizing the compound function  $y = (x_1 \text{ AND } x_2) \text{ OR } x_3$

## 6.5 Various functions typically supported by PLCs

PLCs may have hundreds of functions in addition to the basic AND, OR and NOT functions described above. In the following sections a few of the most important additional functions are described.

### 6.5.1 Timers

A timer function is similar to a clock which is started (triggered) by a starting signal changing value from OFF to ON. When a preset time has elapsed, the timer output is set to ON. As an example, a timer can be used to implement a time-delayed start of a motor. Another example is to control the ON-time of an heater.

PLCs have a number of different timers in their functions library. Figure 6.13 shows an example of a timer. (The example can be found in a user’s manual of the GX IEC Developer programming tool of Mitsubishi PLCs.)

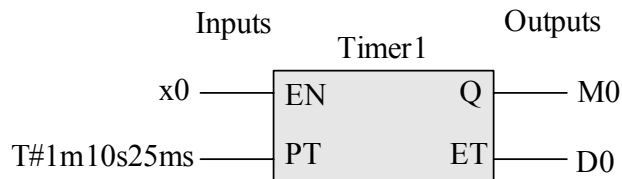


Figure 6.13: An example of a timer function

The timer parameters are as follows.

- Input **EN** (enable) [boolean] starts or triggers the timer.

- Input **PT** (preset time) [time] is the (elapsed) time or time delay before the timer output **Q** is set to ON. The time format is a special data format used to represent time. In Figure the PT value is 1 minute 10 second 25 milliseconds.
- Output **Q** (output<sup>1</sup>) [boolean] is the timer output. It gets value ON when the elapsed time is larger than the preset time.
- Output **ET** (elapsed time) [time] is the continuously running time.

Figure 6.14 shows a timing diagram which shows the behaviour of the timer along a time axis. The timer operates as follows:

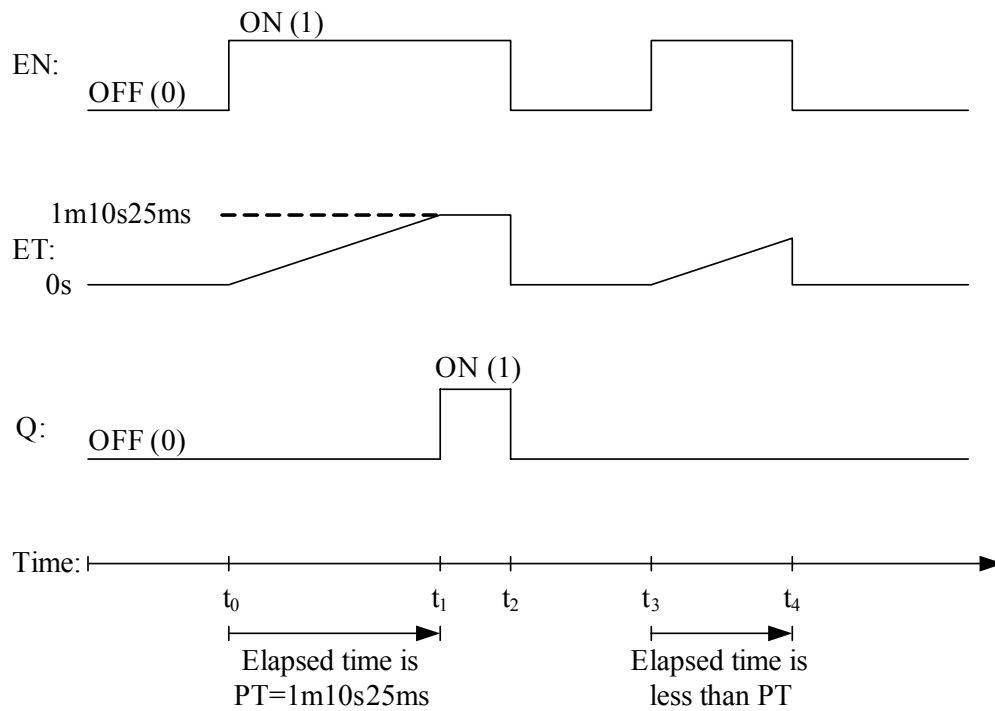


Figure 6.14: Typical timing diagram of a timer

- The timer starts when the input  $x_0$  at the EN input goes from OFF to ON. As the timer starts, the ET output increases continuously from the initial value of 0 seconds. In this example, the ET time value is stored in the data register D0 which is a general register or

<sup>1</sup>Q means output. The ideal symbol O for output is not used since it is too easily misinterpreted as zero.

memory cell where a value can be stored for use in subsequent programming expressions.

- When the ET time has become larger than the PT (preset time) value the boolean output Q is set to ON. In this example, the value of Q is stored in memory cell M0 which is a general memory cell where a boolean value can be stored for use in subsequent programming expressions.
- If EN goes from ON to OFF, ET is reset to zero.

### 6.5.2 Counters

A counter is a function which counts the number of times a boolean input variable has changed value either from OFF to ON, or from ON to OFF, or both. As an example, a counter can be used to count the number of bottles passing a photo diode.

In a given PLC there are various counters available in the library of functions. Figure 6.15 shows an example of a counter.

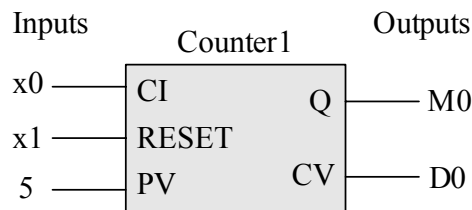


Figure 6.15: A example of a counter

Here is a description of the parameters and the operation of the counter:

- Input **CI** (counter input) [boolean] is the signal of which the changes from OFF to ON is to be counted. (The change from OFF to ON is denoted the positive edge of the signal.)
- Input **RESET** [boolean]: Value ON resets the counter value CV to zero.
- Input **PV** (preset value) [integer] is the preset “target” value of the counter. In 6.15 PV is 5, as an example.

- Output **Q** (output) [boolean] is the counter output. It gets value ON when the counted number, CV, is equal to or larger than the preset counter value, PV. Otherwise it has value OFF.
- Output **CV** (counter value) [integer] is the number of times the counter input, CI, has changed from OFF to ON.

Figure 6.16 illustrates how the counter works.

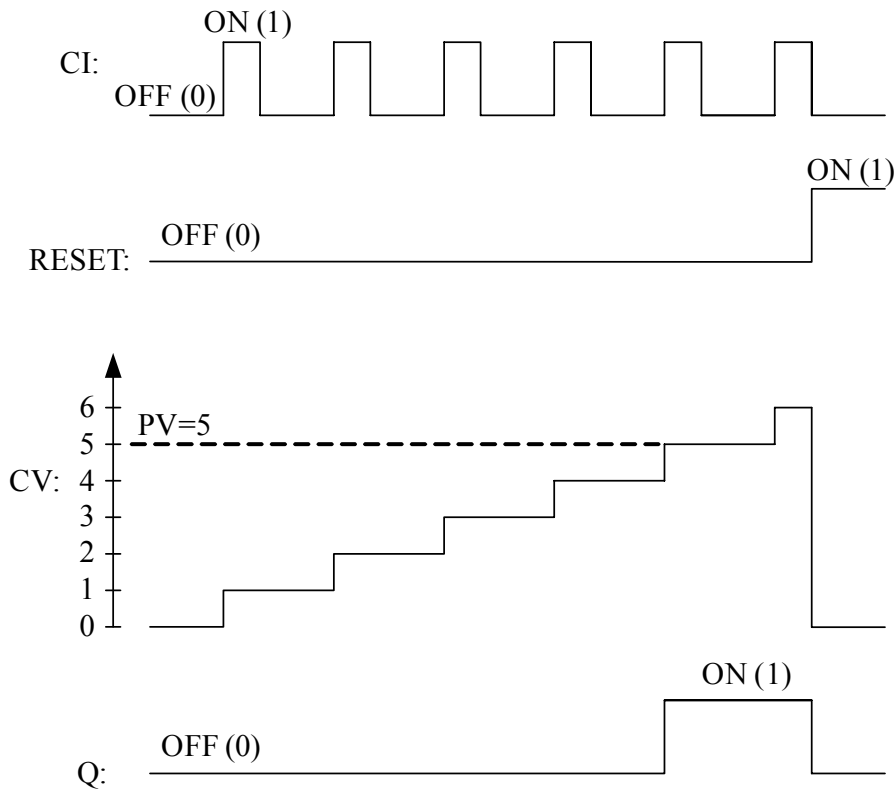


Figure 6.16: The operation of a counter

### 6.5.3 Latches

Not described yet.