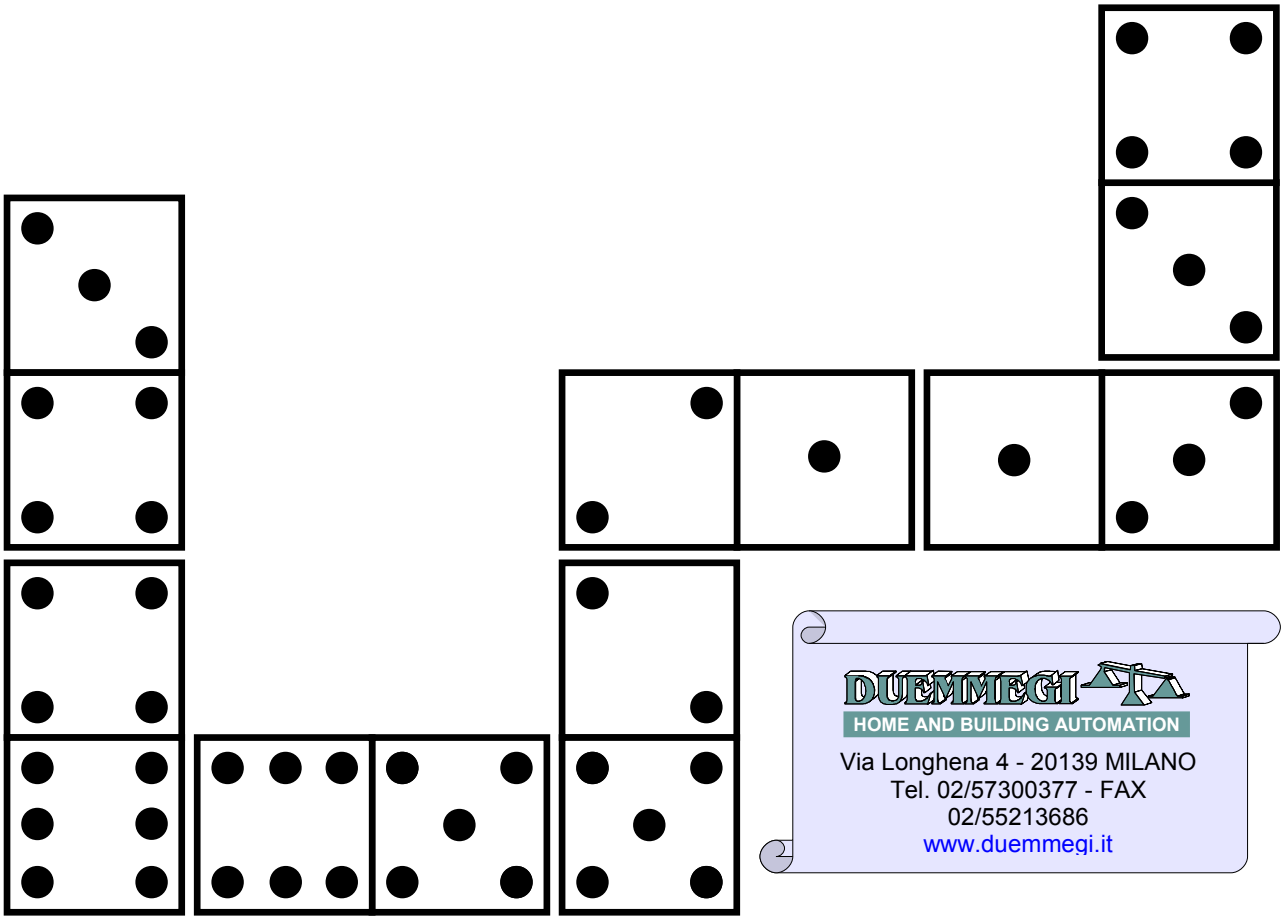
Domino

General Programming Manual

Release 2.2 – May 2015



DUEMMEGI
HOME AND BUILDING AUTOMATION
Via Longhena 4 - 20139 MILANO
Tel. 02/57300377 - FAX
02/55213686
www.duemmegi.it

INDEX

DUEMMEGI s.r.l. - Via Longhena, 4 – 20139 MILANO
Tel. 02/57300377 - Fax 02/55213686 – www.duemmegi.it

A1- CHANGES MADE TO THIS MANUAL IN RESPECT TO THE PREVIOUS RELEASE

Par. 5.1	Corrected last equation of example 8

A2- RECOMMENDATIONS

WARNING: ***This manual refers to Domino bus system and it provides a general guideline for programming the system. Do always refer to the data sheets of specific modules that are constantly kept up to date with the latest additions and changes. It is not therefore ensured the full consistency of this manual with the modules actually available.***
Also be always careful, in the data sheets of the modules, to the firmware version of the module which that sheet relates. Also make sure that the programs used for the programming and management of the system (eg BDTools, DCP Ide, etc..) are up to date; to check this, connect to the site www.duemmegi.it .

Correct disposal of this product

(Waste Electrical & Electronic Equipment)
(Applicable in the European Union and other European countries with separate collection systems). This marking on the product, accessories or literature indicates that the product should not be disposed of with other household waste at the end of their working life. To prevent possible harm to the environment or human health from uncontrolled waste disposal, please separate these items from other types of waste and recycle them responsibly to promote the sustainable reuse of material resources. Household users should contact either the retailer where they purchased this product, or their local government office, for details of where and how they can take these items for environmentally safe recycling. This product and its electronic accessories should not be mixed with other commercial wastes for disposal. Specifically about the battery, check local regulations for correct disposal. Never use municipal waste.

Installation and use restrictions

Standards and regulations
The design and the setting up of electrical systems must be performed according to the relevant standards, guidelines, specifications and regulations of the relevant country. The installation, configuration and programming of the devices must be carried out by trained personnel. The installation and the wiring of the bus line and the related devices must be performed according to the recommendations of the manufacturers (reported on the specific data sheet of the product) and according to the applicable standards. All the relevant safety regulations, e.g. accident prevention regulations, law on technical work equipment, must also be observed.

Safety instructions
Protect the unit against moisture, dirt and any kind of damage during transport, storage and operation. Do not operate the unit outside the specified technical data. Never open the housing. If not otherwise specified, install in closed housing (e.g. distribution cabinet). Earth the unit at the terminals provided, if existing, for this purpose. Do not obstruct cooling of the units. Keep out of the reach of children.

Setting up
The physical address assignment and the setting of parameters (if any) must be performed by the specific softwares provided together the device or by the specific programmer. For the first installation of the device proceed according to the following guidelines:

- Check that any voltage supplying the plant has been removed
- Assign the address to module (if any)
- Install and wire the device according to the schematic diagrams on the specific data sheet of the product
- Only then switch on the 230Vac supplying the bus power supply and the other related circuits

Applied standards
This device complies with the essential requirements of the following directives:
2004/108/CE (EMC)
2006/95/CE (Low Voltage)
2002/95/CE (RoHS)
EN 55022 Class B

Note
Technical characteristics and this data sheet are subject to change without notice.

1- ASSIGNING THE ADDRESSES TO MODULES

1.1- Introduction

The first operation to be accomplished for all modules of **Domino** family is the address setting; the address is a number which identify each module connected to the system.

The rules for address setting are the followings:

- Address of input modules must be a number in the range 1 to 255
- Address of output modules must be a number in the range 1 to 255
- Input modules and output modules can be set with the same address
- It is forbidden to assign the same address to different input modules
- It is forbidden to assign the same address to different output modules
- Consecutive addresses are not mandatory

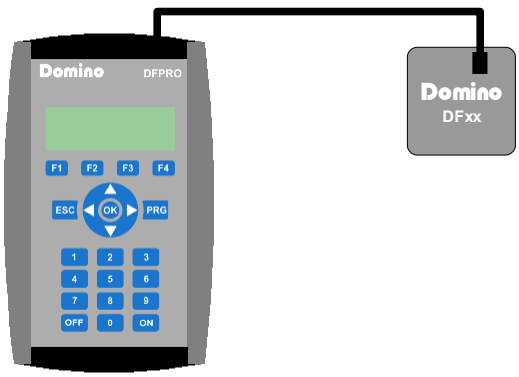
The address can be assigned to the modules before or after the installation, through **DFPRO** programmer, or via a PC connected to **Domino** bus through an interface (**DTRS** or **DFUSB** or **DFCP**). The PC must be equipped with the support program *BDTools* (if using DFRS or DFUSB) or DCP Ide (if using DFCP) provided by **DUEMMEGI**. These programs are also required for the system programming.

In almost all cases, the address assignment to a module requires the pressing the button on the module itself.

1.2- Assigning the address through DFPRO

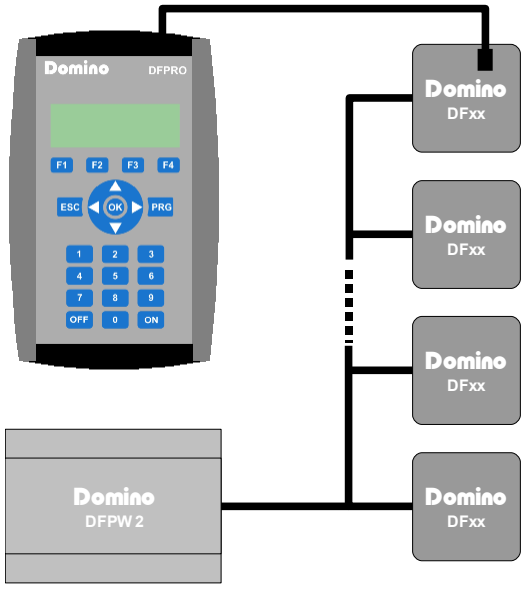
The address assigning to **Domino** bus modules can be executed at bench using a PC, equipped by BDTools, a DFRS or DFUSB module, a DFPW2 and some wirings (or by DCP Ide with DFCP).
This work become more difficult if the modules have been installed in the plant, because in this case two persons are required, one at the PC and another one going to push the programming button on the modules; in addition, the two operators have to communicate between them in order to synchronize the operation to be achieved for each module. **DFPRO** avoids all these uncomfortable operations, being a battery powered portable instrument that does not require the use of a PC.

DFPRO can also perform several diagnostic functions on the **Domino** system (for instance the reading of the inputs, the forcing ON/OFF of the outputs, the detection of fault modules) and it can also operate, through the proper provided cable, as interface between the RS232 port of a PC and the **Domino** bus, emulating exactly the DFRS module.

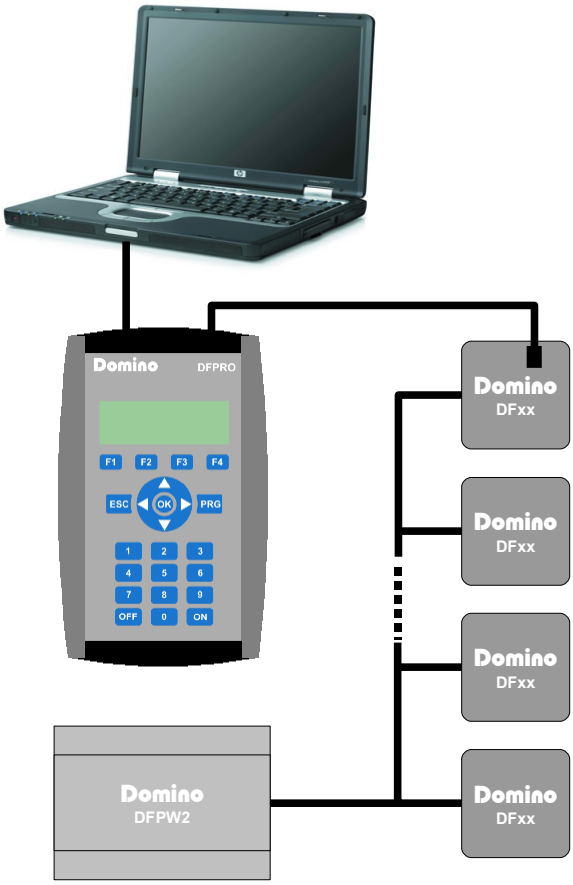


DFPRO can be directly connected to PRG connector (if available) of a single **Domino** not supplied by the bus. This kind of connection is typically useful for assigning and checking of the module address before its installation in the plant.

DFPRO can be directly linked to the PRG connector of a module inside a **Domino** bus system, supplied by one or more DFPW2; in this case, many diagnostic and configuration functions can be performed.

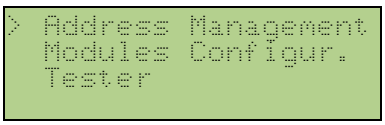


DFPRO can be finally be used as interface between a PC and the **Domino** bus (regardless of the activated menu). In this case DFPRO works in a way absolutely identical to the **Domino** DFRS interface.

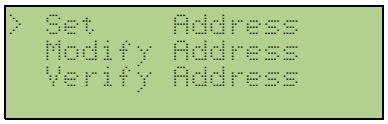


The use of **DFPRO** for address assignment will be described in the following.

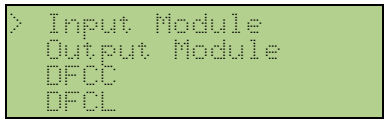
To switch on **DFPRO** (if it is not already ON) push the key ON. The display will show for 2 seconds a screen containing the firmware version, then it will show the main menu:



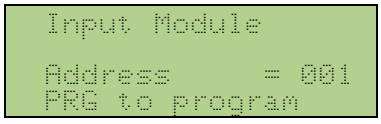
Choose the Address Management option and push OK; the following will be displayed:

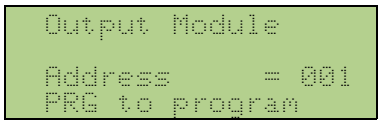


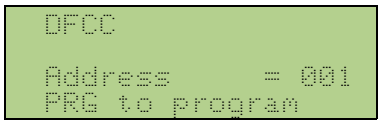
Choose Set Address option to assign the address to input or output module and to assign the ID tp special modules (DFCC, DFCL):

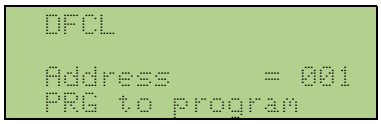


Choose the type of module (Input, Output, DFCC, DFCL):









Enter the desired address by the numerical keys, prepare the module to receive the address pressing the button on the module (the green LED will be constantly lighted) and push PRG on DFPRO within 10 seconds from the last pressing of the button on the module (the green LED will return to blink).

It is also possible to change or to check an address already assigned to a module; for more details on this and other possibilities featured by DFPRO refer to its manual.

DUEMMEGI s.r.l. - Via Longhena, 4 – 20139 MILANO
Tel. 02/57300377 - Fax 02/55213686 – www.duemmegi.it

1.3- Assigning the address through DFRS or DFUSB

After having connected the system to the power supply and launched BDTools program on the PC connectet to DFRS or DFUSB, proceed according to the following steps:

1. From *BDTools* menu select *Communication* and then *Enable Communication*
2. In the driver window press the button *Automatic Detection* to enable the communication between the PC and **Domino** system; close the driver window
3. From *BDTools* menu select *Programming* and then *Address Setting*
4. In the window *ADDRESS SETTING* choose the address to be assigned to the module (writing the address in the text box *New Address*)
5. By a small screwdriver or similar tools, press the small pushbutton of the module to which the address has to be assigned: the green LED of the module will continuously light
6. While the green LED of the module is ON, press the button *Set* in the window *ADDRESS SETTING* of *BDTools* (within 10 seconds from button pressing)
7. At this step, the green LED on the module will return to blink and the message “Completed” will appear in the window of the PC
8. Repeat steps 4 to 7 for all modules

Notes:

- Addresses will be set irrespective of input or output module type; in other words, it is possible to have an input module with same address of an output module
- Avoid to set the same address to more than one input module in the same system
- Avoid to set the same address to more than one output module in the same system
- It is not mandatory to set consecutive addresses, even if recommended to avoid mistakes

If for any reason the current address of a module must be changed, select *Modify Address* check box in the window *ADDRESS SETTING* of *BDTools* (the same window just described at previous step 4) and then specify the current address of the module (*Old Address*), if it is an input or output module and the new address to be assigned; finally, press button *Set*. In this case it is not required to press the pushbutton on the module; take attention that new address be not previously assigned to another module in the system, otherwise conflicts between

2- SYSTEM PROGRAMMING

The **Domino** bus system is essentially made by input modules and output modules distributed in the plant; the functions that must be executed by the system (called equations) normally reside in the internal memory of output modules, and these function are normally programmed using the software package BDTools.

Sometimes, the operating specifications required for the application need a very complex programming, and in some cases the complexity comes to the practical impossibility to satisfy this requirements through the standard programming tools. In this cases the programmable control module DFCP can surely solve the problem.
The DFCP programmable controller, therefore, was born from the exigence to satisfy some complex operating specifications in a Domino bus system; the required functions, as said above, should be impossible, or anyway very difficult, to be implemented through the standard programming of output modules. DFCP was born from the ten-year experience of DUEMMEGI in the world of bus systems, with its attested and well known reliability. The development tool DCP Ide allows both the programming of **DFCP** controller and the programming of output modules, thus replacing BDTools program.

This manual will describe the “base" programming of the modules, leaving to DFCP manual the description of the “advanced” programming.

To program all the functions of the system, a PC equipped with *BDTools* program has to be used; **DUEMMEGI** provides this program. The PC must be connected to DFRS or DFUSB interface.

Each module of **Domino** system has been specifically developed to perform special functions; for this reason the following paragraphs will specify the module type to which the described operator may be applied.

The writing of the program is performed specifying an output point according to the syntax **`Oi.p`** (or **`Vi.p`** for virtual outputs), where **`i`** is the output module address and **`p`** is the output point of that module; e.g., **`O4.2`** means the output point 2 of module which address is 4.
The output symbol described above must be followed by the equal sign (**`=`**); on the right side of this last sign can be placed input points only and other specific symbols called “operators”.
The input points must be specified according to the syntax **`Ii.p`** (or **`Vi.p`** for virtual inputs), where **`i`** is the input module address and **`p`** is the input point of that module; e.g., **`I7.3`** means the input point 3 of module which address is 7.

Each block made by the controlled output point, the = sign and input points that affect the specified output will be called “equation” in the following pages.

Equations may contain several operators that will be described in the following paragraphs.

In any cases, equations must be written according to the following rules:

- Spaces and TAB characters have no meaning and will be ignored. **It is however strongly recommended to use some space characters between the terms of an equation in order to improve the readability of the program**
- An equation can be broken on several lines using the symbol **** (backslash) at the line end to specify that the equation will continue on the next line
- The equation finishes at the end of the line (if the \ symbol is not specified)
- The **`//`** symbol (two slashes) declares that any following characters, until the line end, are comments, and so they will be ignored. **The comments are very useful for best readability and documentation of the program**. The use of the comment is strongly recommended to describe each equation in the program
- **Brackets cannot be used in any equations, unless otherwise specified**
- Both upper case and lower case characters can be used during the equation writing

The program (containing the equations) must be saved into a file with *.EQU* extension (e.g. MYHOME.*EQU*).

Instead of the input and output symbols **`Ij.k, Ox.y, Vn.m,`** it is possible to employ some variable names defined by the user through the **`define`** directive as here below described:

```
define      Pump1       O1.1  // Output definition
define      Command     I1.1  // Input definition

Pump1 = Command                 // Equation
```

The previous equation is fully equivalent to:

```
O1.1 = I1.1
```

but it is of course easier to be understood. The variable names defined through the **`define`** directive **cannot contains spaces**. In addition, the compiler will ignore upper or lower case.

DUEMMEGI s.r.l. - Via Longhena, 4 – 20139 MILANO
Tel. 02/57300377 - Fax 02/55213686 – www.duemmegi.it

The following example shows a possible and simple program:

```
//////////////////////
// Definitions
//////////////////////
define      StairLight        O1.1
define      Floor1Button      I1.1
define      Floor2Button      I1.2
define      Floor3Button      I1.3

// Stair light output
StairLight = TIMER( Floor1Button | Floor2Button | Floor3Button, 0, 180)
```

In the above example there are 3 buttons, one per each floor of a building; the pressing of a button switches on the stair light. This light will remain on during 180 seconds after the button release, then it will be automatically switched off. The same program may be written without using the definition of variable names as follows:

```
// Stair light output
O1.1 = TIMER ( I1.1 | I1.2 | I1.3, 0, 180)
```

Note that using the “define” directive, the program has a best and mnemonic readability.

Resuming, the programming of **Domino** system is carried out through 3 steps, all supported by *BDTools* program:

1. Building (or editing) of *filename*.EQU file, containing the operating program
2. compiling of *filename*.EQU, that is the conversion of the ASCII file in a format ready to be transferred into **Domino** modules
3. uploading of compiled program into the memory of **Domino** modules

Note: if one or more syntax errors will be detected during above step 2, the compiler will warn about type and number of line where the error occurs.

3- REAL POINTS AND VIRTUAL POINTS

Domino system allows the management of input and output “real” points, in other words “physically existing” points; in addition, some **virtual points** are available, and these ones are not related to any real point on the field; the virtual points are the result of the combination of inputs (real or virtual ones).

The virtual points allow defining of variables for some complex functions. The virtual points may be considered and handled as outputs that can be then used as inputs of other equations to control real outputs or other virtual points; for this reason, in the following pages, statements as **“virtual input”**, **“virtual output”** and **“virtual point”** will be used as equivalent terms.

To allows the using, in a **Domino** system, of virtual points, one or more **DF4I/V** modules must be installed; these modules are identical to DF4I input modules, but in addition to the 4 real inputs they provide 12 virtual points for each module. *If DFCP controller has been installed in the plant, this provides approximately 2000 virtual points, therefeore, generally, in this case it is not necessary to install DF4I / V modules.*

DF4I/V module **takes 4 consecutive input addresses and 4 consecutive output addresses**; to allow the module working, a **“base address”** has to be assigned to it. For instance, assigning to a DF4I/V module the base address 9, the same module will automatically take the addresses from 9 to 12 included, both for input and output section. Warning: **the base address must be multiple of 4 plus 1** (e.g. 1, 5, 9, 13, 17, etc.).

Said **n** the base address of a DF4I/V module, the points will be distributed as follows:

- points from `In.1` to `In.4` are real inputs (available on the terminal block)
- points from `On.1` to `On.4` are output points that cannot be used (reserved points)
- both input and output addresses `n+1`, `n+2` and `n+3` are related to the 12 virtual points (4 for each address)

If the system requires more than 12 virtual points, it is possible to install more DF4I/V modules on the same bus line, using different base addresses.

4- INPUT NEGATION

To reverse the logic of an input (real or virtual one) of **Domino** system, simply place the symbol **!** (exclamation mark) before the input itself. This rule, unless otherwise specified, is true for all the functions described in the following pages.

5- GENERAL PURPOSE RELAY OUTPUT MODULES AND VIRTUAL MODULES

Following paragraphs describe the programming functions that, ***unless otherwise specified***, may be applied to the following modules:

- **DF2R** (discontinued module): 2 relay outputs module
- **DF4R** (discontinued module): 4 relay outputs module
- **DF4RP** and **DF4RPR**: 4 power relay outputs module
- **DF4RP/I**: 4 power relay outputs and 4 digital inputs module
- **DFIGLASS**: "touch" keyboard 6 buttons and 6 LEDs
- **DF8IL**: 8 inputs and 8 LED outputs module
- **DF4IL**: 4 inputs and 4 LED outputs module
- **DFTR** (discontinued module): 1 general purpose and 1 shutter output module (applies to general purpose output only)
- **DFDV**: 1 general purpose and one 1-10V analog output for external dimmer control (applies to general purpose output only)
- **DF4I/V**: 12 virtual points and 4 real inputs module

When removing the power supply, all modules store the current status of the outputs; in this way, at the restoring of the power supply, the outputs will be forced to the same status as before the interruption (unless the program specify a different initial status).

5.1- Logical equations

Logical equations are the simplest and allow the combinations of several inputs to control an output. Allowed operators are the following:

- `&` (logical AND)
- `|` (logical OR)
- `S` (SET, run)
- `R` (RESET, stop)

`&` operator may be compared, in the electro-mechanical notation, to the series of contacts, while the `|` operator is equivalent to the parallel connection. The `S` and `R` operators allow to implement simple START/STOP sequences and must be placed before the related input. The SET input causes the activation

of the output when its related input goes to its active status and the output will be held ON even if the input itself goes back to its steady state. At the activation of the RESET input, the output will be de-activated.

The general equation defining a **START-STOP** sequence is:

```
Ox.y = SIj.k & RIn.m
```

where **Ij.k** is the input causing the activation of the output **Ox.y** and **In.m** is the input causing the switch-off of the same output. **&** symbol linking SET/RESET operators is mandatory. More SET and RESET commands can be combined as in the following examples. The operators S and R can also be combined with other "free" inputs using **&** operator (consent).
The SET and RESET commands work on the level and the RESET command has priority in respects to the SET one, so if the RESET term in an equation is active, the output will be always be off.

Example 1:
The following is a simple equation switching ON the output 1 of module which address is 6 when the contact connected to input 3 of module 1 closes:

```
O6.1 = I1.3
```

Example 2:
The following is an equation switching ON the output 2 of module 3 when at least one of inputs **I1.1**, **I2.1** and **I3.4** switch to active state; this equation is equivalent to the parallel connection of 3 contacts:

```
O3.2 = I1.1 | I2.1 | I3.4
```

Example 3:
The following is an equation switching ON the output 2 of module 3 when all inputs **I1.1**, **I2.1** and **I3.4** switch to active state; this equation is equivalent to the series connection of 3 contacts:

```
O3.2 = I1.1 & I2.1 & I3.4
```

Example 4:
The following is an equation switching ON the output 1 of module 5 at the activation of input **I1.3** or at the concomitant activation of both inputs **I4.1** and **I9.4** (or at activation of all inputs):

```
O5.1 = I1.3 | I4.1 & I9.4
```

Example 5:
The following is an equation switching ON the output **O6.1** at the opening of the contact connected to input **I1.3**:

```
O6.1 = !I1.3
```

Example 6:
The following is an equation setting output **O1.1** at the activation of input **I1.3**; the output will be held ON even if the input goes back to its steady state. The output will be reset at the activation of input **I4.1**:

```
O1.1 = SI1.3 & RI4.1
```

Note: SET and RESET inputs must be linked together by **&** operator.

Example 7:
The following is an equation setting output **O1.1** at the activation of input **I1.2**, but only if input **I1.1** is activated; the output will be held ON until the de-activation of input **I1.1** or input **I1.3** (in this case **I1.1** acts as consent):

```
O1.1 = I1.1 & SI1.2 & RI1.3
```

Example 8:
Equations with more SET and RESET terms:

`O1.1 = SI1.1 & SI1.2 & RI1.3` output is ON when `I1.1` and `I1.2` are both ON; the output is OFF when `I1.3` is ON

`O1.1 = SI1.1 & RI1.3 | SI1.2 & RI1.3` output is ON when activating `I1.1` or when activating `I1.2` and it is OFF activating `I1.3`

`O1.1 = SI1.1 & RI1.3 & RI1.4 | SI1.2 & RI1.3 & RI1.4` output is set by `I1.1` or by `I1.2` and reset by `I1.3` or `I1.4`

5.2- Toggle

The Toggle operator, identified by **`T`** character, inverts the output status at every OFF→ON variations of related input. The input status will be then ignored until new variations occur. This function simulates the bi-stable relay.

The Toggle operator optionally allows to define one input to force the switching ON and one input to force the switching OFF of the output (SET and RESET).

The general format of the toggle equation is the following:

`Ox.y = TIj.k | RIh.l | SIm.n`

where **`Ij.k`** is the input that invert the output, **`Ih.l`** is the reset input and **`Im.n`** is the set input.

It is also possible to invert, set and reset the output from more inputs using the | (OR) symbol as explained in the following examples; in addition, some inputs acting as consents can be included.
The terms TOGGLE work on the edge (in other words on the change of input, not on its level), while the terms SET and RESET work on the level with priority to RESET. The term **`T`** in a TOGGLE equation can be omitted, thus the result is an equation entirely similar to the SET/RESET function seen in the previous paragraph; see the examples in this same paragraph.

WARNING: free inputs (in other words inputs without **`T`**, **`R`** or **`S`** prefixes) in a Toggle equation require to be linked by AND operator to inputs having **`T`**, **`R`** or **`S`** prefixes; for instance, the following equation is allowed:

`O1.1 = I1.1 & TI1.2 | I1.3 & SI1.4`

but the following equation is not allowed:

`O1.1 = I1.1 & TI1.2 | I1.3`

because input **`I1.3`** cannot be included alone.

In addition, the consents must be placed before inputs having **`T`**, **`R`** or **`S`** prefixes.

DUEMMEGI s.r.l. - Via Longhena, 4 – 20139 MILANO
Tel. 02/57300377 - Fax 02/55213686 – www.duemmegi.it

Another feature of Toggle function is the defining of a timeout (evaluated from last switching ON of the related output) after which the output will be switched OFF; this timeout is called **"Actuation timeout"** and it must be specified, in **minutes**, inside round brackets before the = symbol in the equation. Allowed values for timeout are **0 to 255** minutes (equivalent to 4 hours and 15 minutes); zero value means an infinite timeout, that is the timeout function is disabled. In this last case the timeout value may be omitted. **Each activation of a SET input (if required) in a Toggle equation will reload the Actuation timeout (in other words this is a retriggerable timer)**.

WARNING: **the actuation timeout for all relay output modules and for DF4IL module is allowed for points 1, 2 and 3 only.**

Example 1:

The following is a simple equation inverting the output 1 of module 6 at each OFF⟶ON variation of input 3 of module 1:

```
O6.1 = TI1.3
```

Example 2:

The following is an equation inverting output **O6.1** at each OFF⟶ON variation of input **I1.3**; after 1 hour from last switching ON, the output will be switched OFF:

```
O6.1(60) = TI1.3
```

Example 3:

The following is an equation inverting output **O6.1** at each OFF⟶ON variation of input **I1.3**; in addition, at OFF⟶ON variation of **I6.1** the output is switched ON and at OFF⟶ON variation of **I6.2** the output is switched OFF:

```
O6.1 = TI1.3 | SI6.1 | RI6.2
```

Example 4:

In the following equation the output **O6.1** will be inverted at each OFF⟶ON variation of any inputs **I1.3**, **I6.1** e **I9.2**:

```
O6.1 = TI1.3 | TI6.1 | TI9.2
```

Example 5:

The following equation invert output **O1.1** at each OFF⟶ON variation of input **I1.1** but only if input **I1.2** is activated; in addition, **I1.3** switches ON the output and input **I1.4** switches OFF the same output:

```
O1.1 = I1.2 & TI1.1 | SI1.3 | RI1.4
```

Example 6:

Equation which set **O6.1** by **I1.3** and reset it by **I1.4**:

```
O6.1 = SI1.3 | RI1.4
```

Example 7:

Equation which set **O6.1** by **I1.1** or by **I1.2** and reset it by **I1.3** or by **I1.4**:

```
O6.1 = SI1.1 | SI1.2 | RI1.3 | RI1.4
```

5.3- Timer

Timer functions allow to delay an output point in respect of one or more input commands. This delay can be related to the activation, the de-activation or both ones.
The general format of the Timer function is the following:

```
Ox.y = TIMER(Ij.k, e, d)
```

where:

- `Ij.k` is the input which controls the delayed output
- `e` is the output activation delay, which value can be in the range 0 to 13107 seconds (3h:38':27") with 0.2s resolution
- `d` is the output de-activation delay, which value can be in the range 0 to 13107 seconds (3h:38':27") with 0,2s resolution

More timers may be combined **by OR operator only**, in order to control the same output by different inputs and with different times; the related general equation will have the following format:

```
Ox.y = TIMER(Ij.k, e1, d1) | TIMER(Ip.q, e2, d2) | ……
```

The max amount of timer allowed **for each module** (real or virtual) is 8.
If the output has to be controlled by several inputs, but with the same delays, the following format may to be used:

```
Ox.y = TIMER( Ij.k | Ip.q | ……, e, d)
```

In this case too, operator OR is the only one which can be used to combine inputs inside timer block.
Some consents, linked by AND operator to timer block, may be also included in the equation (see the examples in the following).
WARNING: **consents** can be combined to timer block **exclusively by AND operator**; other combinations have not any meanings. **Also, these consents must be always placed before the related timer block.**

In addition to above described standard timer function, it is possible to define a special timer generating a **pulse** at the activation (or de-activation) of an input; this function is called **monostable timer** and its general format is the following:

```
Ox.y = TIMERP(Ij.k, 0, d)
```

Character `P` (Pulse) identifies the monostable timer function; at input activation, the output will be switched ON until delay `d` elapses. The pulse duration `d` can be in the range 0 to 13107 seconds with 0.2s resolution; the activation delay, in this case, has no meaning and must be always zero.
The pulse on the output will have the specified duration even if the input will be de-activate before or after the delay `d` elapses; this timer type is also called **no-retriggerable monostable**.
Another timer type is the **retriggerable monostable**. In this last case the equation has the following general format:

```
Ox.y = TIMERPR(Ij.k, 0, d)
```

Characters `PR` identify the retriggerable monostable; in this case the pulse duration, even if it was already started, will be set to time `d` at each OFF to ON variation of the input (or at each ON to OFF variation if the input is preceded by NOT symbol `!`).

DUEMMEGI s.r.l. - Via Longhena, 4 – 20139 MILANO
Tel. 02/57300377 - Fax 02/55213686 – www.duemmegi.it

Example 1:
The following is a simple equation controlling an output (**O1.1**) by one input (**I1.1**); the output is 2 seconds delayed in respect to the input activation and 5.4 seconds delayed in respect to input de-activation:

```
O1.1 = TIMER(I1.1, 2, 5.4)
```

Example 2:
The following is an equation controlling an output (**O1.1**) by two inputs (**I1.1** and **I1.2**); delays are 0 seconds at the activation and 5 seconds at the de-activation:

```
O1.1 = TIMER(I1.1 | I1.2, 0, 5)
```

Example 3:
The following is an equation controlling an output (**O1.1**) by two inputs (**I1.1** and **I1.2**); activation and de-activation delays on output are 2 and 5 seconds in respect to the first input and 25 and 45 seconds in respect to the second one:

```
O1.1 = TIMER(I1.1, 2, 5) | TIMER(I1.2, 25, 45)
```

Example 4:
The following is an equation controlling an output (**O1.1**) by one input (**I1.1**). Activation delay is 3.2 seconds and de-activation delay is 8.6 seconds; in addition, the output can be controlled only if the consent **I9.1** is activated (the consent must be placed before the TIMER operator):

```
O1.1 = I9.1 & TIMER(I1.1, 3.2, 8.6)
```

Example 5:
The following equation implements a no-retriggerable monostable generating a 5 seconds pulse on output **O1.1**; the pulse will be generated at OFF to ON variation of input **I1.1**. Any variation of the input during the pulse does not affect the duration of the pulse itself.

```
O1.1 = TIMERP(I1.1, 0, 5.0)
```

Example 6:
In this case, each OFF to ON variation of input during the pulse will restart the pulse duration.

```
O1.1 = TIMERPR(I1.1, 0, 5.0)
```

5.4- Clock function (scheduler)

Programming CLOCK function allows the switching ON and OFF of the outputs as a function of one or more predetermined times. Programming can be daily or weekly, in which case, as well as the time, the day of the week must be specified.

Note: *Programming CLOCK function cannot be applied to DFTR and DFDV modules.*

To use this function, a **clock module DFCK3 (or a DFCP controller) must be installed in the system**. The general format of Programming Clock function is the following:

```
Ox.y = CLOCK( ON, OFF )
```

ON and **OFF** are the switching ON and the switching OFF time in the following format:

```
GG:HH:MM
```

where:

GG is the day of the week; allowed value are: **MON, TUE, WED, THU, FRI, SAT, SUN**. **If day of the week is omitted, programming will be daily**.
HH hours (00 ÷ 23); this parameter must be in the 24 hour notation: e.g., 7 means 7a.m., 19 means 7p.m..
MM minutes (0 ÷ 59).

More CLOCK functions may be combined together, **by OR operator only**, to control the same output, in order to have more output switchings inside the same day or the same week. Consents, linked by AND operator to CLOCK blocks, may be also included in the equation; **these consents must be placed before the related CLOCK block.**

Example 1:
The following is a simple equation controlling an output (**O1.1**) by a CLOCK function; the output will be switched ON everyday from 8:30 to 17:30:

```
O1.1 = CLOCK( 8:30, 17:30 )
```

Example 2:
The following equation controls an external lamp (**O1.1**); the lamp may be enabled everyday from 17:00 to 19:30, but switching ON may occur at sunset. Input **I1.1** will be connected to a light dependent relay as consent of Programming Clock function:

```
O1.1 = I1.1 & CLOCK( 17:00, 19:30 )
```

Example 3:
The following equation switches ON an output (**O1.1**) everyday from 8:00 to 12:00 and from 13:30 to 17:30:

```
O1.1 = CLOCK( 8:00, 12:00 ) | CLOCK( 13:30, 17:30 )
```

Example 4:
The following equation switches ON an output (**O1.1**) from Monday 9:00 to Friday 19:00:

```
O1.1 = CLOCK( MON:09:00, FRI:19:00 )
```


DUEMMEGI s.r.l. - Via Longhena, 4 – 20139 MILANO
Tel. 02/57300377 - Fax 02/55213686 – www.duemmegi.it

Example 5:
The following equation switches ON an output (**O1.1**) from 9:00 to 19:00 everyday from Monday to Friday included:

```
O1.1 =      CLOCK( MON:09:00, MON:19:00 ) | \
            CLOCK( TUE:09:00, TUE:19:00 ) | \
            CLOCK( WED:09:00, WED:19:00 ) | \
            CLOCK( THU:09:00, THU:19:00 ) | \
            CLOCK( FRI:09:00, FRI:19:00 )
```

5.5- Oscillator

The Oscillator function allows automatic and continuous switching of a virtual point, which frequency depends on the times fixed in the equation. The general format of Oscillator function is the following:

Vx.y = OSC(Ton, Toff)

where **Ton** and **Toff** are the ON and the OFF status duration respectively. Both times can be in the range 0.2 to 13107 seconds (3h:38':27"); the resolution is 0.2 seconds.

Notes: the Oscillator function applies **exclusively to a virtual point (DF4I/V or DF8IL)**. No combinations with other point is allowed. Avoid to define too short times (even if the minimum value is 0.2 seconds) because the traffic on the bus may be overloaded (since the module executes a transmission on the bus at every change of the point).

This function is useful in all cases in which there is the need, for example, to make a flashing lamp (warning lights or similar).

Example 1:
The following program control a lamp connected to **O1.1**; the lamp will be blinking at the activation of input **I3.2**. The blinking period is 2.8 seconds (**Ton=1.4 + Toff=1.4**).

```
V130.1 = OSC(1.4, 1.4)
O1.1 = I3.2 & V130.1
```

Example 2:
In the following program the output **O5.2** will be switched ON for 5 minutes every 2 hours (300"+6900" = 7200" = 120' = 2h).

```
V131.3 = OSC(300, 6900)
O5.2 = V131.3
```

5.6- Threshold

The Threshold function can be applied to digital output modules (e.g. DF4RP) and to virtual modules DF4I/V. The threshold function controls a digital output as function of the result of comparison between an analog value (for instance that returned by a DFAI or DFTA module) and a threshold, eventually with a hysteresis.
The general format of Threshold function is the following:

Ox.y = AIk >= T,H

Where:

- **Ox.y** is the output (real out in this case) controlled by the threshold function
- **AIk** represents the analog input which address is k
- **>=** is the comparison operator (in this case greater or equal to)
- **240** is the threshold
- **12** is the hysteresis (the comma symbol is mandatory)

Allowed comparison operators:

<	lower than
<=	lower or equal to
==	equal to
!=	not equal to
>	greater than
>=	greater or equal to

The hysteresis has a different meaning depending on the comparison operator as here described:

<	the output goes ON when **AI<T** and it returns OFF when **AI>=(T + H)**
<=	the output goes ON when **AI<=T** and it returns OFF when **AI>(T + H)**
==	the output goes ON when **AI=T** and it returns OFF when **AI>(T + H)** or when **AI<(T - H)**
!=	output goes OFF when **AI=T** and it returns ON when **AI>(T + H)** or when **AI<(T - H)**; this behavior is complementary to the previous case (**==**)
>	the output goes ON when **AI>T** and it returns OFF when **AI<=(T - H)**
>=	the output goes ON when **AI>=T** and it returns OFF when **AI<(T - H)**

Note: *if hysteresis has not been specified, then it will be assumed equal to zero.*

Of course, threshold and hysteresis values must be in the range of values allowed for the considered analog module (for DFAI module this is 0 to 1000).

By means of AND (**&**) and OR (**|**) operators, more threshold functions can be combined in the same equation as shown by the following examples.

Example 1:
The output goes ON when the analog value measured by module 1 is greater or equal to 730, and it goes OFF when it is lower than 728:

```
O4.1 = AI1 >= 730,2
```

Example 2:
The virtual output goes ON when the analog value AI1 is exactly equal to 240 or when AI2 is greater or equal to 30:

```
V130.1 = AI1 == 40 | AI2 >= 30
```

Example 3:
The output goes ON when the analog value AI1 is in the range 30 to 128 (greater than 30 and lower than 128):

```
O1.4 = AI1 < 128 & AI1 > 30
```

Example 4:
The output goes ON when the analog value AI9 is in the range 30 to 128 or when AI5 is greater than 600:

```
O3.2 = AI9 > 30 & AI9 < 128 | AI5 > 600
```


DUEMMEGI s.r.l. - Via Longhena, 4 – 20139 MILANO
Tel. 02/57300377 - Fax 02/55213686 – www.duemmegi.it

6- DIMMER MODULES

Dimmer function apply to **DFDM, DFDT, DFDI, DFDI2** and to dimmer section of **DFDV** modules to control the brightness of lamps; these modules have been specifically developed to perform some special functions; for this reason, all information in this paragraph apply to this module type only.

Generally, the dimmer functions may be applied to inputs connected to pushbuttons.
When removing the power supply, the dimmer module stores the current status of the output; in this way, at the restoring of the power supply, the output will be forced to the same status as before the interruption.
In a dimmer equation, the following operators can be used as a prefix of the input which must perform the related function:

- **U** to increase the brightness (Up)
- **D** to decrease the brightness (Down)
- **M** to increase and decrease the brightness (Mono-command)
- **P** to set the brightness at a well specified value (Preset)
- **A** for the automatic brightness regulation

For increasing and decreasing the brightness, dimmer function allows two operating mode:

1. by two separated pushbuttons to control the increment and the decrement of brightness respectively
2. by only one pushbutton (mono-command) to control both increment and decrement of brightness

These two modes of operation will be here bottom described.

Case 1: Up/Down pushbuttons
Pressing and holding Up pushbutton, the brightness will increase until the maximum value will be reached. Pressing and holding Down pushbutton, the brightness will decrease until the minimum value will be reached (this minimum value is not the completely OFF lamp condition).
When brightness reaches the wanted level, simply release the pushbutton to hold that level.
When the lamp is at any level, a short touch on one of the two pushbuttons causes the full switching OFF.
When the lamp is completely OFF, a short touch on one of the two pushbuttons causes the switching ON at the last level of brightness or at a programmable fixed value (see the dimmer module data sheet for more details).

Case 2: Single pushbutton
Pressing and holding the pushbutton (mono-command), the brightness will increase until the maximum value will be reached, then, after a pause of 1 second about, the brightness will decrease until the minimum value will be reached (this minimum value is not the completely OFF lamp condition); if pushbutton if held again, the cycle restart increasing the brightness and so on.
When brightness reaches the wanted level, simply release the pushbutton to hold that level.
When the lamp is at any level, a short touch on the pushbutton causes the full switching OFF.
When the lamp is completely OFF, a short touch on the pushbutton causes the switching ON at the last level of brightness or at a programmable fixed value (see the dimmer module data sheet for more details).

In both cases, the lamp switching ON and OFF occurs in a “soft” mode, because the brightness variation from the previous level and the OFF condition and vice-versa follows a predefined ramp; the ramp value can be changed as described in the following.

In addition, it is possible to define one or more Preset inputs allowing to set the brightness to one or more levels specified in the equation.

The brightness control by two pushbuttons and by mono-command may be both included in the same equation, with or without one or more Presets.

The general format of the equation controlling a dimmer module is the following:

```
Ox.y(Rd,MIN,MAX) = UIj.k | DIh.l | MIm.n | P(V,R)Is.t
```

where **(Rd,MIN,MAX)** are optional parameters specifying, in the same order, the default ramp, minimum value and maximum value of the output.
Ij.k is the Up input, **Ih.l** is the Down input, **Im.n** is the mono-command input and **Is.t** is the input for the PRESET of the brightness to value **V** with ramp **R** (both ones inside round brackets, the ramp value is an optional parameter and, if omitted, it will be assumed equal to the default value). The PRESET value is a number in the range 0 to 100 and it is the brightness level as percentage (%) of the maximum value. If **V** is a number greater than 100, then a well defined command will be executed, as specified in the technical sheets of the dimmer modules (for instance the value 124 here bottom described).

Note that all terms are linked by OR operator (**|**).

The **PRESET values 0 and 124** have a special meaning:

- **P(0)Is.t** switch off the lamp and store the previous brightness level
- **P(124)Is.t** switch on the lamp at last stored brightness level

In real applications, the just described preset values are useful for **centralized switching OFF and ON** of lamps controlled by more dimmer modules.

It is possible to include, in the same equation, more inputs U, D, M and P in order to control the brightness level from more points (locations). Some consents, linked by AND operator to inputs **U**, **D**, **M** and **P**, may be also included in the dimmer equation. These free inputs (in other words inputs without **U**, **D**, **M** and **P** prefixes) in dimmer equation **require to be linked exclusively by AND operator to inputs having U, D, M or P prefixes**; other combinations have not any meaning. **In addition, these consents must be placed before the inputs having the dimmer operators**.

Dimmer modules, in addition to the dimmer standard functions (Up, Down, Single Command and Preset), also feature an extended function allowing to implement the automatic regulation of the brightness on a room, comparing the value read from a light sensor connected on the **Domino** bus with a fixed setpoint. The keyword identifying this function is "**A**"; the following equation is a typical example:

```
O1.1 = UI1.1 | DI1.2 | V130.1 & A(650,20,2)AI18
```

The block **A(sp,h,p)AIx** identifies the automatic brightness regulation function, where:

- **sp** is the setpoint, that is the brightness level to be maintained; in the example, the setpoint is **650**
- **h** is the hysteresis (**20** in the example); the regulation function acts so that the light level will be maintained in the range from (setpoint-hysteresis) to (setpoint+hysteresis); therefore, in the example, the range is from 630 to 670; the hysteresis value must be ≤ 255
- **p** is the time period (**2** in the example): the module, every **p** seconds, compares light level read from the sensor and the setpoint (± hysteresis)
- **x** is the address (**18** in the example) of a light sensor module (e.g. DFLUX), or of a DFAI analog input module connected to a light sensor

The block **A(sp,h,p)AIx** must be preceded by a consent input (real or virtual point) activating and deactivating the automatic regulation function.
In the previous example, the consent is **V130.1**; activating this point, the automatic regulation will enabled, while deactivating it the automatic regulation will be disabled, but the output level of the dimmer module remains the last one reached.
For more details about the automatic brightness regulation feature, refer to the technical sheet of the chosen dimmer module.

DUEMMEGI s.r.l. - Via Longhena, 4 – 20139 MILANO
Tel. 02/57300377 - Fax 02/55213686 – www.duemmegi.it

Example 1:
The following is a simple equation controlling a dimmer output (**O1.1**) by Up and Down pushbuttons (inputs **I1.1** and **I1.2** respectively):

```
O1.1 = UI1.1 | DI1.2
```

Example 2:
The following is a simple equation controlling a dimmer output (**O1.1**) by a single pushbutton (Mono-command, input **I1.1**):

```
O1.1 = MI1.1
```

Example 3:
The following is a simple equation controlling a dimmer output (**O1.1**) by a single pushbutton (Mono-command, input **I1.1**); the pushbutton connected to **I1.2** set the brightness level to 50%:

```
O1.1 = MI1.1 | P(50)I1.2
```

Example 4:
The following is an equation controlling a dimmer output (**O1.1**) by Up and Down pushbuttons and by a mono-command; three additional inputs set the brightness level to zero, 40% and 80% respectively. **I99.1** recalls the last stored value (or a well defined value, depending on how the dimmer module has been set, see dimmer modules data sheet):

```
O1.1 = UI1.1 | DI1.2 | MI6.1 | P(0)I9.1 | P(40)I9.2 | P(80)I9.3 | P(124)I99.1
```

Example 5:
The following equation controls a dimmer output (**O1.1**) by a single pushbutton (Mono-command, input **I1.1**) having a series connected consent (input **I1.2**); if the consent is not active, it is not possible to change the brightness of the lamp connected to the dimmer. This consent may be, for example, a key switch that disables the button operation:

```
O1.1 = I1.2 & MI1.1
```

7- ROLLING SHUTTER MODULES

Rolling shutter functions apply to **DFTP** and **DFTP/I** modules and to the shutter section of **DFTR** modules (Warning: this module is discontinued). These modules have been specifically developed to perform some special functions; for this reason, what follows in this paragraph refer only to to this modules.

Generally, the Rolling Shutters functions may be applied to inputs connected to pushbuttons.
In a rolling shutter equation, the following operators can be used as a prefix of the input which must perform the related function:

- **O** to open the rolling shutter (Open)
- **C** to close the rolling shutter (Close)
- **OP** to open the rolling shutter by a centralized command (Priority Open)
- **CP** to close the rolling shutter by a centralized command (Priority Close)
- **H** to stop the shutter (Halt)
- **G** to partially close the shutter (GoTo)

The shutter modules execute automatically several functions as here described; assume that the module has been programmed for opening and closing by two pushbuttons (Open and Close) connected to an input module.
Pressing and holding Open or Close pushbutton, the rolling shutter will be driven in opening or closing direction; releasing the pushbutton, the rolling shutter will be stopped at the current position.

If the maximum opening or closing position has been reached before the releasing of pushbutton, the rolling shutter will be stopped by the upper or lower limit switch that break off the power supply to the motor; this **limit switches must be included by the shutter manufacturer** (all systems on the market generally have this device). Note that these limit switches have no any connection with **Domino** system.

When rolling shutter is not moving, a short touch on Open pushbutton or on Close pushbutton causes the movement of the motor until the limit switch is reached or until a programmable time out elapses (complete opening and closing function, called automatic mode). If during the automatic movement any Open or Close button is pushed again, the shutter stops at that position (this operation is called counter-command).

The centralized commands too work as described above for local commands; the difference is that **a centralized command is only automatic** and it is always executed regardless of the current shutter condition (moving or not moving). In other words, if the shutter was moving, a centralized command will cause the driving in the direction determined by the centralized command itself; on the contrary, a local command in the same conditions just described, will stops the shutter movement.

It is also possible to define additional commands performing the unconditional Halt, allowing to stop the motor regardless of the function currently in execution.

Finally, it is possible to add GoTo commands to perform partial movements, so as to close the shutter to a given percentage with respect to the total displacement. Since a position information is not generally available for standard shutter, this function is based on the timing of the command, after appropriate configuration (see related paragraph); please note that the time of opening and closing can vary with time and climatic conditions because of the variation of friction, therefore it is possible a certain error in the positioning. **To use the GoTo commands, it is necessary to properly configure the shutter module, so please consult the data sheet of the module itself.**

Notes:

- As said above, the shutter module does not know if the shutter reached the opening or closing limit; to avoid damages to the motor, check for the presence of limit switches in the chosen movement system.
- To avoid damages to the motor, the shutter module waits 2 seconds about before to invert the movement direction of the shutter.

During the automatic opening and closing functions, the relays driving the motor remain excited even after the shutter has reached the limit switch; to avoid unnecessary waste of current, the shutter module automatically turns off the relays after a predefined time (Actuation Timeout). This time, by default, is equal to 60 seconds, but it can have any value in the range 1 to 254; if the shutter requires a greater (or lower) time, in respect to the default one, to go from the fully closed to fully open position (and vice versa), it is possible to specify a different timeout value in the equation (the value must inside parenthesis), just before the equals sign. If the timeout value is omitted in the equation, the value of Actuation Timeout will be assumed by BDTools equal to the default value (60). **Setting the Actuation Timeout to 0 (zero), the automatic mode will be disabled (but not for centralized commands)**.

It is also possible to define a time, said “Delay from command”, which delays the motor start in respect of a centralized command; this delay may avoid the simultaneous start of all shutters controlled by the same centralized command. The default delay from command is zero, but may be modified by user up to 255 seconds (4 minute and 15 seconds).

The general format of the Rolling Shutter function is the following:

```
Ox.y(T1)= OIj.k | CIh.l | OP(T2)Im.n | CP(T3)Is.t | HIw.z | G(V)If.g
```

where `Ij.k` is the opening input, `Ih.l` is the closing input, `Im.n` is the centralized opening input and `Is.t` is the centralized closing input; `T1` is the actuation timeout in seconds, `T2` is the delay from centralized opening command in seconds and `T3` is the delay from centralized closing command in seconds. If `T2` and `T3` must be zero (default value), they can be omitted. `Iw.z` is the Halt input and `If.g` closes the shutter to V% (V must be in the range 0 to 100).

All terms in a Rolling Shutter equation must be linked by OR operator (|). ***As said above, setting T1=0, the automatic opening and closing functions will be disabled for local commands.***

More inputs O, C, OP, CP, H and G may be included in the same equation in order to control the shutter movement from more locations. Consents linked by AND operator to inputs with **O, C, OP, CP, H** and **G** prefixes may be also included in the equation.

***WARNING:* free inputs** (in other words inputs without **O, C, OP, CP, H** and **G** prefixes) in Rolling Shutter equation **require to be linked only by AND operator to inputs having O, C, OP, CP, H** and **G prefixes**; other combinations have not any meaning. **In addition, these consents must be placed before the inputs affected by the Rolling Shutter operators**.

Example 1:
The following equation controls a rolling shutter output (**O1.1**); the actuation timeout is set to 60 seconds. **I1.1** is connected to the opening pushbutton and **I1.2** is connected to closing pushbutton:

```
O1.1(60) = OI1.1 | CI1.2
```

Since the specified timeout is the default value, it may be omitted:

```
O1.1 = OI1.1 | CI1.2
```

Example 2:
The following equation controls a rolling shutter output (**O6.1**) which actuation timeout is set to 40 seconds; **I1.1** is the opening input, **I1.2** is the closing input and **I25.1** is the centralized closing input. The closing from centralized command occurs, for the shutter of this example, after 5 seconds from the command itself:

```
O6.1(40) = OI1.1 | CI1.2 | CP(5)I25.1
```

Example 3:
The following equation is identical to the previous one, but in this case the delay from the centralized command is set to 0:

```
O6.1(40) = OI1.1 | CI1.2 | CPI25.1
```

Example 4:
The following equation controls a rolling shutter output (**O6.1**) which actuation timeout is set to 60 seconds; **I1.1** and **I1.2** are opening and closing input, **I25.1** is the centralized closing input and **I25.2** is the centralized opening input.
The centralized commands are 2 seconds delayed. When input **I25.3** is not active, both centralized commands are disabled (e.g. to disable commands by unauthorized persons):

```
O6.1(60) =  OI1.1 | \
            CI1.2 | \
            I25.3 & CP(2)I25.1 | \
            I25.3 & OP(2)I25.2
```

Example 5:
The following equation controls a rolling shutter where, in addition to opening and closing commands, there are a Halt (**I5.1**) command and two GoTo commands closing the shutter to 50% and 80& respectively (**I7.1** e **I7.2**).

```
O1.1(40)= OI1.1 | CI1.2 | OP(5)I3.1 | \
          CP(10)I3.2 | HI5.1 | \
          G(50)I7.1 | G(80)I7.2
```

8- DFCK3 CLOCK MODULE

DFCK3 clock module allows to control outputs according to configurable schedules, both daily and weekly, in a **Domino** system.

As described in a previous paragraph, the CLOCK function allows to manage a virtually unlimited amount of outputs, each one with its proper switching ON and OFF times **fixed by program**; on the other hand, DFCK3 module allows to manage up to 15 different outputs (zones), with up to 4 different switching ON and 4 switching OFF times for each day of the week.
The advantage in respect to the CLOCK equation is that each scheduled time can be easily changed by the system supervisor acting as user interface (touch screen video terminal, DFTouch, WEBS, DFWEB or others).

During the setting up, DFCK3 module must be properly configured using BDTools, therefore please refer to the technical sheet of DFCK3.

DFCK3 module, from the **Domino** bus point of view, is fully equivalent to a digital input module; each point of the module corresponds to a zone and must be identified with the usual notation **`Ix.y`**, where **`y`** is in the range 1 to 15.
Each input point will be activated as function of the scheduled times and it must be used in proper equations inside output modules in order to execute the desired commands. In other words, the points of DFCK3 can be used as a usual input points of the system.

It is possible to install up to 8 DFCK3 modules on the same bus line. Only one of these ones must be set as MASTER (ID=1, see DFCK3 data sheet). Only the MASTER module will send on the bus the date (1 time per minute) and the hour (6 times per minute), in order to keep synchronized the other modules and to allow the use of the CLOCK equation in the output module allowing this function.
A change of the time on any DFCK3 module (regardless of the MASTER or SLAVE setting) will be automatically reported on all other modules.
If a DFCP controller is installed on the bus, its CLOCK will be always the MASTER in respect of the other DFCK3 modules, therefore all the DFCK3 modules must be set as SLAVE (thus with ID in the range 2 to 8).

9- EXAMPLES ABOUT USING OF VIRTUAL POINTS

As said before, each DF4I/V module provides to the system 12 virtual points and 4 real inputs. To avoid mistakes, it is recommended to assign “high” addresses to these modules and “low” addresses to real input and output modules. For instance, a good rule is to assign the base address of DF4I/V modules starting from 129. Also remember that DF4I/V module takes 4 input addresses and 4 output addresses and that the base address must be multiple of 4 plus 1; supposing to have assigned the base address 129 to a DF4I/V module, the points will be distributed as follows:

- from **`I129.1`** to **`I129.4`**: real inputs
- from **`O129.1`** to **`O129.4`**: reserved points (to not be used)
- from **`V130.1 to V130.4`**, from **`V131.1 to V131.4`** and from **`V132.1 to V132.4`**: 12 virtual points

Example 1:
The following program controls a lamp (**`O1.1`**) which must be in OFF state everyday from 08:00 to 18:30; outside of this range, the lamp has to be driven by a local pushbutton by toggle function (**`I1.1`**):

```
V130.1 = CLOCK( 08:00, 18:30 )
O1.1   = TI1.1 | RV130.1
```

Example 2:
The following program controls a lamp (**`O10.3`**) which brightness must be set to 50% everyday from 08:00 to 17:30; outside of this range (from 17:30 to 08:00) the lamp brightness must be set to 80% :

```
V130.1 = CLOCK( 08:00, 17:29 )
V130.2 = CLOCK( 17:30, 07:59 )
O10.3  = P(50)V130.1 | P(80)V130.2
```

Example 3:
The following program controls the opening and the closing of a rolling shutter (**O5.1**); the opening and the closing can be controlled manually by the pushbuttons connected to **I1.1** and **I1.2** respectively. In addition the rolling shutter must be automatically closed everyday at 22:00 and it must be automatically opened everyday at 07:00 :

```
V130.1 = CLOCK( 22:00, 22:01 )
V130.2 = CLOCK( 07:00, 07:01 )
O5.1   = OI1.1 | CI1.2 | CPV130.1 | OPV130.2
```

Example 4:
The following program controls a stair lamp (**O99.2**); three real inputs, connected to pushbuttons, switch ON the lamp, while the switching OFF occur automatically after 180 seconds. In addition, the switching ON must be allowed from 18:00 to 07:00 only:

```
V130.1 = CLOCK( 18:00, 07:00 )
O99.2  = V130.1 & TIMER(I1.1 | I2.1 | I3.1, 0, 180)
```

Example 5:
The following program controls two lamps by a single pushbuttons (**I129.1**): at the first pressing the lamp **O1.1** will be switched ON, at the second pressing the lamp **O1.2** will be switched ON too, at the third pressing both lamps will be switched OFF:

```
V130.1 = !V130.2 & !V130.1 & TI129.1 | V130.2 & V130.1 & TI129.1
V130.2 = !V130.2 &  V130.1 & TI129.1 | V130.2 & V130.1 & TI129.1
O1.1 = V130.1
O1.2 = V130.2
```

10- MAXIMUM ALLOWED NUMBER OF TERMS IN THE EQUATION

Since the operating program of an output module is loaded in the memory of the module itself, and since this memory is relatively limited, then the length of the equations loaded in a module cannot exceed fixed limits.

Let's define "term" or "weight" an input (real or virtual one) which control an output (real or virtual one); the input may be a "free" input or it may be applied to a simple operator (**S**, **R**, **T**, **!**). The maximum number of term allowed for the equations loaded into an output module depends on several factors.
Anyway, the compiler checks if the equations to be loaded in a module exceed the maximum memory size of the module itself; for this last reason, to dissipate any doubts, it is suggested to write the program and check if the compiler does not report the error message "Error: Equations too long for module x" or "Too much addresses for module x".

11- DXP SERIAL COMMUNICATION PROTOCOL

DFRS or DFUSB module allows to interface a PC or any other programmable device (called Host in the following) to the **Domino** bus system.
In this way it is possible to implement simple supervision systems to report the operating status of the system and for the direct control of the outputs.

The name of the proprietary protocol used by DFRS or DFUSB module is DXP; the communication occurs through a RS232 or USB port according to the following parameters:

Baud rate:19200 baud (fixed)
Data bits: 8
Parity: none
Stop bits: 1

Note: the numerical data in the following pages are intended in hexadecimal format (notation 0x) unless otherwise specified.

DFRS or DFUSB module acts as slave device; all messages between master unit (Host) and DFRS or DFUSB module have the following format:

START	FUNCTION	MESSAGE TYPE	MODULE ADDRESS	DATA 1	DATA 0	CHECKSUM

where:

START	1 fixed byte which value is 0x55, to identify the message start
FUNCTION	1 byte to identify the message function
MESSAGE TYPE	1 byte to identify the message type (command, request, ack)
MODULE ADDRESS	1 byte containing the address of **Domino** module
DATA 1, DATA 2	2 data bytes
CHECKSUM	1 check byte complemented sum of all transmitted bytes

In the following pages the allowed messages (request and answer) will be listed.

11.1- Status request to an input module

Message from Host to DFRS or DFUSB:

55	82	30	Address	33	33	CHECKSUM

Where Address is that of **Domino** input module which status has to be read.

Answer from DFRS or DFUSB to Host:

55	82	B0	Address	Data 1	Data 0	CHECKSUM

Where Address is that of **Domino** input module which answers to the request and Data 1-Data 0 have the following meaning:

- for **digital input modules** (e.g. DF4I), Data 1 is always 0x00 and Data 0 is the status of the input points, coded in binary format (1=input ON, 0=input OFF). Less significant bit of Data 0 is input point 1, most significant one is input point 8.
- for **analog input modules**, Data 1 and Data 0 have to be intended as 16-bit value (Data 0 is the LSB).

DUEMMEGI s.r.l. - Via Longhena, 4 – 20139 MILANO
Tel. 02/57300377 - Fax 02/55213686 – www.duemmegi.it

11.2- Status request to an output module

Message from Host to DFRS or DFUSB:

55	82	31	Address	33	33	CHECKSUM

Where Address is that of **Domino** output module which status has to be read.

Answer from DFRS or DFUSB to Host:

55	82	B1	Address	Data 1	Data 0	CHECKSUM

Where Address is that of **Domino** output module which answers to the request and Data 1-Data 0 have the following meaning:

- for **digital output modules** (e.g. DF4R), Data 1 is always 0x00 and Data 0 is the status of the output points, coded in binary format (1=output ON, 0=output OFF). Less significant bit of Data 0 is output point 1, most significant one is output point 8.
- for **rolling shutter output module** (e.g. DFTP), Data 1 and Data 0 depend on how the module has been configured; refer to paragraph 11.4 or to the data sheet of shutter modules for more details. For example, if the module has been set for "Motors Status", then Data 1 is always zero, bits 0 and 1 of Data 0 report the status of opening and closing command respectively of shutter 1, bits 2 and 3 of Data 0 report the status of opening and closing command respectively of shutter 2
- for **analog output modules**, Data 1 and Data 0 have to be intended as 16-bit value (Data 0 is the LSB).
- for **dimmer output modules**, Data 1 is always 0x00 and Data 0 is the percentage of the brightness level

11.3- Command of digital outputs

For modules having 4 digital output points for each address:
Message from Host to DFRS or DFUSB:

55	82	10	Address	Data 1	Data 0	CHECKSUM

Where Address is that of **Domino** output module which outputs have to be driven and Data 1-Data 0 have the following meaning:

- Data 1 is always 0x00
- Data 0 identifies the output (or the outputs) to be switched ON or OFF (1=output ON, 0=output OFF). Data 0 is binary coded as follows: the 4 most significant bits identifies which outputs have to be modified (mask), the 4 less significant bits are the commands to be transferred to the outputs. See next example for better understanding

Example:

To switch ON and then OFF outputs 1 and 3 of a module, the binary code of Data 0 is:

SWITCHING ON ⇒ **0 1 0 1 0 1 0 1**

SWITCHING OFF ⇒ **0 1 0 1 0 0 0 0**

In other words, first 4 bits select which output points have to be changed; non selected output points retain the previous status.

For modules having more than 4 digital output points for each address, points 1 to 8:
Message from Host to DFRS or DFUSB:

55	82	12	Address	Data 1	Data 0	CHECKSUM

Where Address is that of **Domino** output module which outputs have to be driven and Data 1-Data 0 have the following meaning:

- Data 1 is tha mask over 8 points (defined in a similar way to the 4 points case)
- Data 0 contains the commands to be sent to the output points selected by the mask

For modules having more than 4 digital output points for each address, points 9 to 16:
Message from Host to DFRS or DFUSB:

55	82	13	Address	Data 1	Data 0	CHECKSUM

Where Address is that of **Domino** output module which outputs have to be driven and Data 1-Data 0 have the following meaning:

- Data 1 is tha mask over 8 points (defined in a similar way to the 4 points case)
- Data 0 contains the commands to be sent to the output points selected by the mask

Answer from DFRS or DFUSB to Host:
For all the three above cases, the answer of the module is:

55	82	B1	Address	Data 1	Data 0	CHECKSUM

Where Address is that of **Domino** output module which answers to the request and Data 1-Data 0 have the following meaning:

- Data 1 and Data 0 report the status of the outputs of the module, encoded by binary code (1 = output active, 0 = inactive output). The least significant bit of Data 0 corresponds to the output point 1, the most significant bit of Data 1 to output point 6

11.4- Command of rolling shutter output

Open/Close command, message from Host to DFRS or DFUSB:

55	82	10	Address	Data 1	Data 0	CHECKSUM

Where Address is that of **Domino** shutter output module which outputs have to be driven and Data 1-Data 0 have the following meaning:

- Data 1 is always 0x00
- Data 0 identifies the output (or the outputs) to be switched ON or OFF (1=output ON, 0=output OFF). Data 0 is binary coded as follows: the 4 most significant bits identifies which outputs have to be modified (mask), the 4 less significant bits are the commands to be transferred to the outputs as here bottom listed:

BIT 3	BIT 2	BIT 1	BIT 0
CL 2	OP 2	CL 1	OP 1

(CL = closing, OP = opening). See next example for better understanding.

DUEMMEGI s.r.l. - Via Longhena, 4 – 20139 MILANO
Tel. 02/57300377 - Fax 02/55213686 – www.duemmegi.it

Example:

To open the shutter 1 (SH1) and to close the shutter 2 (SH2) the binary code of Data 0 is:

OP SH1, CL SH2 ⇒ **1 0 0 1 1 0 0 1**

GoTo command, message from Host to DFRS or DFUSB:

55	82	10	Address	Data 1	Data 0	CHECKSUM

Where Address is that of **Domino** shutter output module and Data 1-Data 0 have the following meaning:

- Data 1 must be 0x01 to control motor 1, 0x02 to control motor 2
- Data 0 is the percentage closing value to be set (0 and 100%, therefore it can assume values 0x00 to 0x64)

Answer from DFRS or DFUSB to Host:
In both cases, the answer of the module is:

55	82	B1	Address	Data 1	Data 0	CHECKSUM

Where Address is that of **Domino** shutter output module and Data 1-Data 0 have a meaning depending on the module configuration as follows:

"Motor Status":

Data 1								Data 0							
P	0	0	0	0	0	0	0	0	0	0	0	2C	2O	1C	1O

Where:
- P: waiting for address programming (it reports the fixed ON status of the PRG LED of the module)
- 2C: when this bit is "1", then the motor 2 is closing the shutter
- 2O: when this bit is "1", then the motor 2 is opening the shutter
- 1C: when this bit is "1", then the motor 1 is closing the shutter
- 1O: when this bit is "1", then the motor 1 is opening the shutter

"Final Position or Real Time Position WITHOUT Status of the Motors":

Data 1			Data 0		
0	0	Position M2 / 2	0	0	Position M1 / 2

Where:
- Position M2 / 2: this is a value in the range 0 to 50 that, multiplied by 2, reports the position of the shutter 2 as percentage 0...100% of the fully closed position
- Position M1 / 2: this is a value in the range 0 to 50 that, multiplied by 2, reports the position of the shutter 1 as percentage 0...100% of the fully closed position

"Final Position or Real Time Position WITH Status of the Motors":

Data 1			Data 0		
2C	2O	Position M2 / 2	1C	1O	Position M1 / 2

Where:
- 2C: when this bit is "1", then the motor 2 is closing the shutter
- 2O: when this bit is "1", then the motor 2 is opening the shutter
- Position M2 / 2: this is a value in the range 0 to 50 that, multiplied by 2, reports the position of the shutter 2 as percentage 0...100% of the fully closed position
- 1C: when this bit is "1", then the motor 1 is closing the shutter
- 1O: when this bit is "1", then the motor 1 is opening the shutter
- Position M1 / 2: this is a value in the range 0 to 50 that, multiplied by 2, reports the position of the shutter 1 as percentage 0...100% of the fully closed position

11.5- Command of dimmer output

Message from Host to DFRS or DFUSB:

55	82	10	Address	Data 1	Data 0	CHECKSUM

Where Address is that of **Domino** dimmer module which output has to be regulated and Data 1-Data 0 have the following meaning:

- Data 1 is always 0x00
- Data 0 is the percentage of the brightness level

Answer from DFRS or DFUSB to Host:

55	82	B1	Address	Data 1	Data 0	CHECKSUM

Where Address is that of **Domino** dimmer module which answers to the request and Data 1-Data 0 have the following meaning:

- Data 1 is always 0x00
- Data 0 is the current percentage of the brightness level

DUEMMEGI s.r.l. - Via Longhena, 4 – 20139 MILANO
Tel. 02/57300377 - Fax 02/55213686 – www.duemmegi.it

11.6- Writing of analog output modules

Message from Host to DFRS or DFUSB:

55	82	10	Address	Data 1	Data 0	CHECKSUM

Where Address is that of **Domino** analog module which output has to be written and Data 1-Data 0 have to be intended as 16-bit value (Data 0 is the LSB).

Answer from DFRS or DFUSB to Host:

55	82	B1	Address	Data 1	Data 0	CHECKSUM

Where Address is that of **Domino** dimmer module which answers to the request and Data 1-Data 0 have to be intended as 16-bit value (Data 0 is the LSB).

11.7- Error codes

If the requested **Domino** module is fault or it is not connected to the bus, DFRS or DFUSB answers to the Host as follows:

Answer in case of fault or not connected module:

55	82	00	00	00	F0	38

If DFRS or DFUSB interface, after a request from the Host, cannot access to the **Domino** bus, then DFRS or DFUSB answer to the Host as follows:

Answer in case of bus busy or bus failure:

55	82	00	00	00	FF	29

11.8- Examples

Example 1:

Status request to input module 1.

REQUEST: 0x55 0x82 0x30 0x01 0x33 0x33 0x91

ANSWER: 0x55 0x82 0xB0 0x01 0x00 0x05 0x72

In the answer, the binary code of the 6th Byte 0000 1001 shows that input points 1 and 4 are ON.

Example 2:

Status request to output module 75 (decimal).

REQUEST: 0x55 0x82 0x30 0x4B 0x33 0x33 0x47

ANSWER: 0x55 0x82 0xB0 0x4B 0x00 0x00 0x2D

In the answer, the binary code of the 6th Byte 0000 1001 shows that all output points are OFF.

Example 3:

Status request to dimmer module 10 (decimal).

REQUEST: 0x55 0x82 0x31 0x0A 0x33 0x33 0XF5

ANSWER: 0x55 0x82 0xB1 0x0A 0x00 0x2D 0xED

In the answer, the decimal value of 6th byte (0x2D) shows that the dimmer output is set to 45%.

Example 4:

Switching ON of points 2 and 4 of output module 30 (decimal).

REQUEST: 0x55 0x82 0x10 0x1E 0x00 0xAA 0X50

ANSWER: 0x55 0x82 0xB1 0x1E 0x00 0x0A 0x4F

In the answer, the binary code of the 6th Byte 0000 1010 confirm that output points 2 and 4 have been switched ON as requested.

DUEMMEGI s.r.l. - Via Longhena, 4 – 20139 MILANO
Tel. 02/57300377 - Fax 02/55213686 – www.duemmegi.it

Example 5:

Switching ON of point 1 and switching OFF of point 3 of output module 49 (decimal).

REQUEST: 0x55 0x82 0x10 0x31 0x00 0x51 0X96

ANSWER: 0x55 0x82 0xB1 0x31 0x00 0x01 0x45

In the answer, the binary code of the 6th Byte 0000 0001 confirm that output point 1 has been switched ON and output point 3 has been switched OFF as requested.

Example 6:

Opening of shutter 1 and 2 of shutter module 37 (decimal).

REQUEST: 0x55 0x82 0x10 0x25 0x00 0x55 0x9E

ANSWER: 0x55 0x82 0xB1 0x25 0x00 0x05 0x4D

In the answer, the binary code of the 6th Byte 0000 0101 confirm that module is performing the opening of shutter 1 and 2 as requested.

Example 7:

Command to set to 85% the output of dimmer module 10 (decimal).

REQUEST: 0x55 0x82 0x10 0x0A 0x00 0x55 0xB9

ANSWER: 0x55 0x82 0xB1 0x0A 0x00 0x55 0x18

In the answer, the decimal value of 6th byte (0x55) shows that the dimmer output has been set to 85% as requested.