Dear Sir,

Here is a description of main functions of Maestro product line.
Attached are also three appendixes:

a. List of Function blocks
b. List of Graphic buttons
c. List of expression language operators

On our web site you can find more information
General description of Maestro is on:
http://www.cdinnovation.com/page1.html?

Downloads are on:
http://www.cdinnovation.com/page28.html?
For example:

Video tutorials (Quick startup: learn basic operation):
Video_totorials_lesson_1_to_6.zip

Manual:
Maestro_Doc_1.5.pdf

Graphic library
Maestro Designer graphic library

Demo project -
http://www.cdinnovation.com/DownLoad/DEMO.zip

On the page are more downloads such as PPT presentation, high res images…

CDinnovation

Hardware:

**MTS4-10-2 (in wall):**

| | |
|---|---|
| LCD : | 10.1 inch TFT |
| LCD resolution: | 1280x800 |
| Colors: | 24 bit |
| Build: | No moving parts |
| Memory: | 1G RAM, 8 G flash |
| Power: | 12V DC, optimized for low power consumption. |
| Audio: | 2 speakers and microphone |
| I/O: | 1xLAN, 1xEIB/KNX (on USB), 1xUSB , 1xRS485 |
| Installation: | in wall |

**MTS4-4 (For DIN rail):**

| | |
|---|---|
| Build: | No moving parts |
| Memory: | 1G RAM, 8G flash |
| Power: | 5V DC, optimized for low power consumption. |
| I/O: | 1 x LAN, 1xEIB/KNX (on USB), 3xUSB, 1 x HDMI |
| Installation: | DIN rail |

CDinnovation

Software:

Maestro devices are running CDI's modified Android operation system on top of our Linux kernel.

Maestro Designer:
Maestro Designer is the PC software that creates Maestro applications.
It's made to maximize efficiency and user friendliness - Utilizing functions such as:

- Import of KNX information directly from ETS2, ETS3 and ETS4 and ETS5 databases
- 3, modular, programming levels:
    - Create graphic screens including buttons. Links the graphic elements to transmitters and receivers (such as KNX group addresses) using **Drag and drop**.
    - Create logic and control functions using graphic editor of **function blocks controller.**
    - Write your own function and communication protocols using **expression language.**
- Run mode - simulate the device on your PC and operate the graphic interface using you PC screen.
- Unlimited choice for screen Background pictures (any jpg or PNG files)
- Unlimited graphics design of buttons appearance and button size.
- No practical limitation – on number of Buttons, Group Addresses, data types, schedules, logged events, alarms, Macro/Scenes, function Blocks and pages.
- Output window – shows real time logger, simulation and debug information.
- Function blocks controller:
    - Event driven, made especially for KNX.
    - 10 different analog function blocks.
    - 14 different digital function blocks (some with up to 16 variations).
    - 28 miscellaneous function blocks: KNX functions and control functions.
    - Expression language function block – write your own functions, of any complexity, using script language.
    - Encode and decode non standard telegrams. Seamlessly transfer vales between different KNX data types and between systems (for example multimedia ⇔ KNX).
    - Variables.
- Graphic library:
    - More than 8,000 icons.
    - More than 200 other backgrounds images and images for buttons.
    - Add your own graphic using any .png or .jpg files and stretchable images on "9 patch" format.
- Pre-designed Drag and Drop Virtual Buttons
    - Buttons are Advanced Objects made to build beautiful and very effective user interface in minimum time.
    - More than 100 predestined buttons are supplied with the software
    - Enabling extremely flexible way to build the perfect interface for the user. Button Graphics structure is modular and (typically) includes:

- 3 eras: Up, Center, Low
- 5 layers: frame, text, icon, image color
- 2 states On, Off/Bar graph/analog value/text display.
- All of these graphic elements are fully editable, so you can easily modify the button's look, create customized buttons, and even save them for future use.

- Button types:
  - 1 x On Off.
  - 5 x Dimmer: 1 x dimmer button and 4 x dimmer bar graphs.
  - 1 x Shatter.
  - 5 x Analog: 3 x buttons and 2 x bar graphs.
  - 5 x any type: multi state, push release, toggle, short/long, push button.
  - 6 x system buttons: Macro, page flip, time, date, video stream.
  - 5 x background buttons.
  - 1 x Color button: select RGB color from a color wheel
- Button functionality:
  - Maestro's button range can control and monitor any EIB/KNX (and external systems ) device.
  - In order to significantly reduce programming time and prevent errors, we have predesigned several buttons to perform common EIB/KNX tasks. For example: Dimmer buttons are limited to send and receive only 1, 4, and 8 bit telegrams. Other buttons are more powerful and flexible and, for example, can send few telegrams simultaneously with different values to different KNX dada types/external devices on different situations (on push, on release...)

CDinnovation

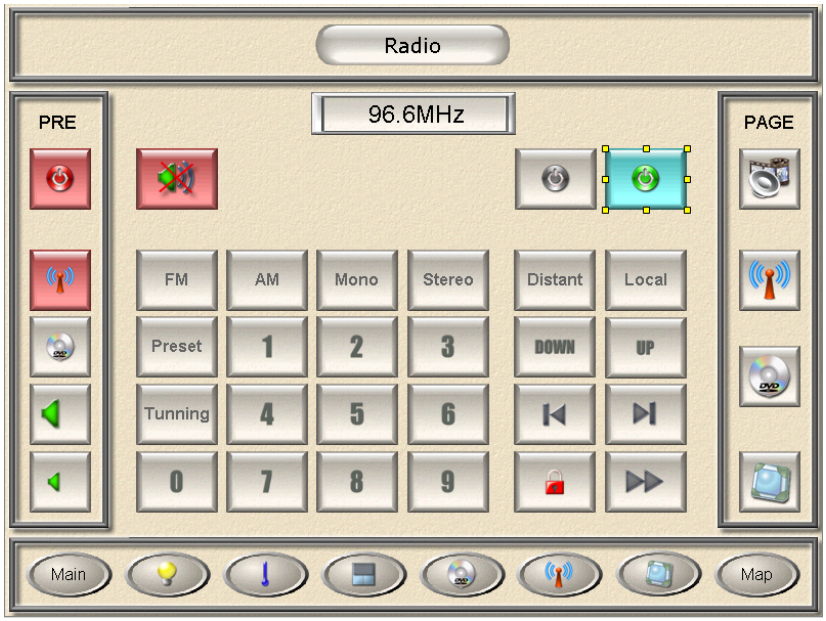
## DEVICE FUNCTIONALITY


- Schedule Manager:
  - Unlimited number of events.
  - Daily.
  - Weekly.
  - Date.
  - Sun rise and sun set related (astronomic clock).
  - List of coming events.
  - Editable by user.
- Macro/Scenario Manager:
  - Unlimited number of macros.
  - Unlimited number of loads.
  - Editable by user.
  - On line or off line tuning
  - Possibility to define delay between functions.
- Advanced Function Blocks Controller
  Maestro integrates powerful function block controller specially made to control EIB/KNX (event driven).
  The controller has over 50 different function blocks specifically designed to meet

the unique characteristics of the EIB/KNX system. There is no practical limitation on the number of function blocks used in a project.
The controller can act as the central control unit for the EIB/KNX system, perform highly complex logic operations and calculations, track the status of EIB/KNX loads/group of loads, and update external display and signal devices. On function block controller you can use/mix: KNX Group Addresses, KNX objects (Maestro knows all Group Addresses that are associated to them), IR commands, serial commands (e.g. "Receiver power on"), TCP commands, Maestro variables, function blocks and button blocks.
Maestro utilize automatic conversion algorithms so you can link different data types so for example you can compare temperature received in 4byte data type to temperature received in 2 byte format or use OR gate between 1 bit and 8 bit values.
For every individual link you make, to function block or to a button, you can define transformation. Transformations are very powerful including Boolean transform, linear transform and even LUT transform that lets you define an output value for every input value.

- <u>Audio/Video and other external systems Integration:</u>
  Maestro Server product line have direct connectivity to the
  - EIB/KNX system
  - IP
  - RS232
  - RS485 including modbus.
  - DMX

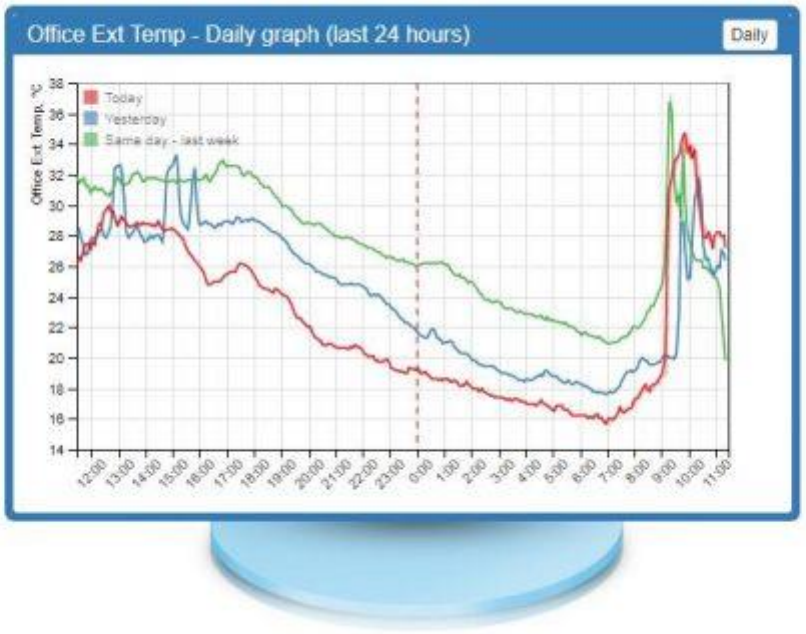
  The external network adepter GC100 enables to extend control over more RS232 ports and IR.
  The Maestro Designer software utilize an advanced protocol generator that enables you to implement the most complex, two way, protocols for controlling external devices/systems. This way you can use the Maestro to control and monitor not only the KNX but also other devices, moreover the Maestro integrate all of the system under it's control and act as gateway between them so, for example you can use KNX push button to control your audio system.




- Graphs

  Maestro is capable of collecting various data such as temperature, wind speed, power consumption, solar radiation etc' and presenting them on graphs. The source of such data can be received directly from equipment connected to Maestro using communication means such as: MQTT, MODBUS and KNX or the result of internal calculations performed by Maestro such as the schematic of various loads, calculation of electricity costs, water throughputs, etc'.

  Maestro locally collects and stores the required data. It can store tens of millions of samplings enabling to collect history data from up to 1000 different sources for up to 3 years each.




Data can be presented, monitored and analyzed by the end-user for maximum optimization, using various graph types: daily, weekly, monthly and annually. Comparative graphs may be used as well. e.g. electricity consumption this month as compared to the previous month or to the same month last year. The user can also export all data along any period of time as CSV file type utilizing for instance, Maestro as a KNX data logger, to process and present the logged data by other platforms regardless of what can be presented by Maestro.
data is saved on Maestro according to the following time intervals:
- every 1 minute for the last 48 hours,
- Every 5 minutes for the last 8 days
- Every 15 minutes for the last 1 year and one month
- And every 1 hour for the last 3 years

- KNXnetIP enabled
  Maestro supports KNX net IP tunneling. KNX system integrator can use it to program and monitor KNX devices of a remote site from his office using the internet. This function combined with 3rd party applications/products so they can use Maestro as KNXnetIP Gateway.

- IP SIP Video intercom
  Maestro touch server is equipped with microphone and 2 speakers and supports IP SIP H263/ H.263+ standard designed for IP video conferencing.
  Therefore maestro can be used as the house or even hole apartment building IP

video intercom system, SIP video phone and even presenting IP SIP video cameras. SIP H263 is worldwide standard therefore you can integrate the Maestro with 3rd party such as Mobotix™, Alphatech™ and Amroad™ 2n™ video intercoms and cameras.

- Hand held remote control:
  Maestro Remote App is a client application made by CDI. The app is installed on devices running iOS, Android and Windows, and use them as a remote control. The remote has the same graphics and functionality as of the Maestro server.


All communications between the Maestro and the remote App is SSH secured (encrypted and password protected).

- Remote shell
  communicate with the Maestro using maestro's simple ASCII protocol:
  - Send and receive all group addresses
  - Control/monitor all Maestro's objects, transmitters, receivers…

  3rd party controllers such as RTI Control4 AMX Crestron are using this interface to connect to KNX as it outperforms any existing KKNX gateway on the market.

- Display of IP video
  Maestro supports Motion JPEG (MJPEG) video streaming. No limitation on No of streams on the projects for example 12 different video streaming can be displayed on every graphic page.

- KNX to BACnet Gateway
  Maestro supports BACnet Server capability, enables to use Maestro as a KNX to BACnet Gateway making it possible for the BMS to control and monitor the KNX. With just simple Drag& Drop actions the integrator can create BACnet objects from any KNX Group Addresses as he likes. He can do so also for any other existing objects coming from Maestro such as data from Modbus, MQTT or any other connected equipment to Maestro, even internal variables that are used for Maestro's internal logic.
  The BACnet Client can initiate a 'Discovery' command to receive the BACnet Object list from Maestro and then Maestro enables the Client to take full control of the Bacnet Object including the value change of Polling and Subscription to 'Monitoring Value Changes'. No limit to the number of BACnet Objects that can be produced by Maestro.

- Amazon Alexa (echo)
  Maestro Server is capable of using Alexa to control all systems and devices connected to Maestro. By using simple voice commands users will be able to instantly activate devices connected to Maestro e.g. Lights, Shades, HVAC, Irrigation Systems, Alarms etc'.
  Maestro offers Alexa's voice control not only to KNX but to all devices using it's built in protocols such as Modbus, MQTT and DMX. Even devices using proprietary protocols supported by Maestro, like HVAC controllers and AV systems will now benefit from voice activation. More over; Maestro can translate Alexa's commands to ASCII strings thus providing a gateway between Alexa to any external controllers capable of handling simple ASCII commands.
  Maestro provides a large number of different voice commands like very specific ones: "Alexa, dim the chandelier to 50 percent" or: "Alexa, set the temperature to 24". But also group commands or a sequence of a number of actions: "Alexa turn on welcome Macro".

- MQTT protocol
  Maestro Server supports the well-known lightweight IoT MQTT protocol for Internet communication. Maestro can play any roll within the MQTT architecture: Broker, Agent (Publish and/or Subscribe) or even all at the same time.

Maestro can serve as a MQTT gateway for KNX providing valuable benefits when using it.
Moreover; Maestro can serve as a powerful central controller for MQTT components wherever they are, harnessing all Maestro's capabilities to the process such as: data collection and graphs presentation, scheduling and advanced logic settings, artificial intelligence for data processing etc'. These capabilities enable to securely retain the information at the user's local site and to choose if and what to be sent for processing on third party's cloud platform and what to retain in privacy and processed locally.

- DMX control
  Maestro provides optimal and seamless connectivity between KNX and DMX directly from Maestro's ports.
  This feature of DMX control brings many advantages like saving the extra cost for not requiring a dedicated DMX to KNX gateway, ease of integration and boosting performance. Maestro KNX server enables to control DMX by using any kind of KNX-data-type e.g. 4bit, 8bit. All 512 DMX channels are supported with Maestro providing 8 bit feedback object for every channel. On the fly additional special objects can be used for setting up of fade rates.
  The combination between DMX and Maestro's superb control capabilities enable the implementation of almost endless number of special effects. e.g. recording and playing of color sequences, continuous rotation of colors, changing colors by hours of the day, sunrise and sunset times etc.'

- Modbus protocol
  Maestro provides a built-in seamless Modbus RTU over RS485 / IP connectivity.
  Maestro can also be used as a gateway between Modbus supported devices to KNX devices or to any other external systems connected to Maestro.
  This feature offers control and possible integration between KNX to Modbus enabled VRF HVACs such as Daikin & Gree and to electric meters by ABB and Siemens etc'.
  In addition, Maestro can act as a serial to Ethernet converter enabling Modbus TCP master/s control serial (RS232/485) Modbus slaves.
  The new Maestro built-in Modbus capability eliminates the need for a dedicated KNX to Modbus gateway adapter, saving system integrators time and money.

- Sending Emails
  Sending different e-mail alerts for various events, the Email's text content can be dynamically composed by the artificial intelligence of the Maestro on the fly

- Sonos Gateway
  Maestro automatically discovers Sonos components in real time. It creates all user interface elements to control Sonos and add it to Maestro's graphic App. In addition Maestro provides the option to link Sonos commands and feedback to other Maestro objects such as KNX Group Addresses thus providing option to control Sonos from KNX switches and devices.

CDinnovation

**TRAINNING**

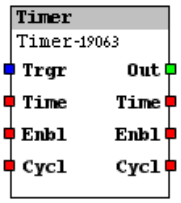
- Quick start: 6 video tutorial
- Demo / reference projects
- documentation

Sample screenshot:

CDinnovation

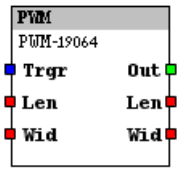
# Appendix A - Function blocks:

1. Timer Function-Block:



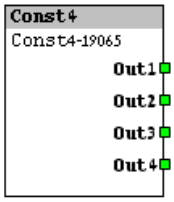

A Timer Function-Block is a count down timer. Once triggered, the timer starts measuring a predetermined time interval. When the time interval is complete, it will generate a True **Event** at its **Output**.

2. PWM Function-Block:



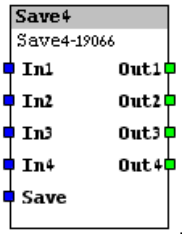

A PWM Function-Block generates a square wave form with variable length and cycle.

3. Const 1,2,..16 (Constant) Function-Blocks:



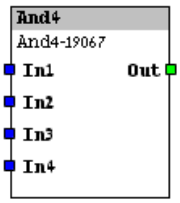

Constant Function-Block holds, on the device's flash memory, constant values.

4. Save 1,2..16 Function-Blocks:



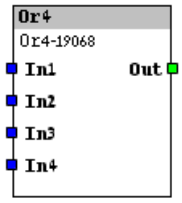

A Save Function-Block saves values in the **Device** flash memory.

5. And, AndN(2..16 inputs) Function-Blocks:



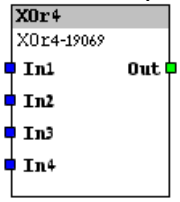

And Function-Block  performs Logic And operation over its **Input**s.

6. Or, OrN(2..16 inputs ) Function-Blocks:



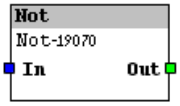

Or Function-Block  performs Logic Or operation over its **Input**s.

7. XOr, XOrN(2..16 inputs) Function-Blocks:



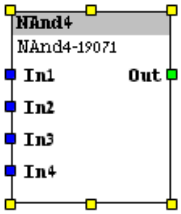

XOr Function-Block performs Logic XOr operation over its **Input**s.

8. Not Function-Block:



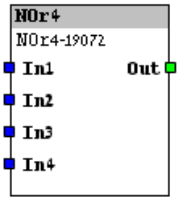

Not Function-Block performs Logic NOT operation over its **Input**s.

9. NAnd, NAndN(2-16 inputs) Function-Blocks:



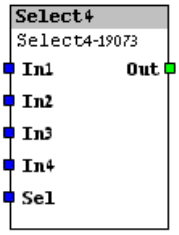

NAnd  Function-Block performs Logic NAnd operation over its **Input**s.

10. Nor, NOrN(2..16 inputs)  Function-Blocks:



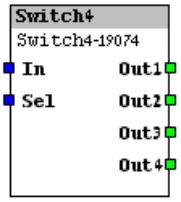

And Function-Block performs Logic NOr operation over all its **Input**s.

11. Sel2, SelN(2..16 inputs) (Select) Function-Blocks:



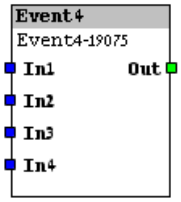

This Function-Block routs one of the **Input**s to the **Output**.

12. Switch2, SwitchN(2..16 outputs) Function-Blocks:



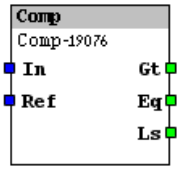

This function routs the **Input** to the selected **Output**.

13. Event2, EventN (2..16 inputs) Function-Blocks:



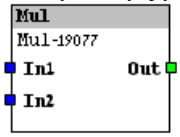

Event Function-Block generates an **Event** on its **Output** whenever it detects an **Event** on its **Input**s.

14. Comp (Comperator) Function-Block:



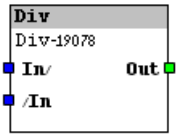

Comparator  Function-Block compares the value of its **Input**s.

15. Mul (Multiply) Function-Block:



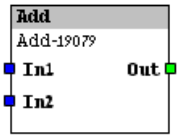

This Function-Block performs arithmetical multiplication operations.

16. Div (Divide) Function-Block:



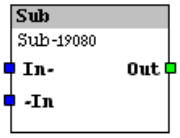

This Function-Block performs arithmetical Division operations.

17. Add Function-Block:



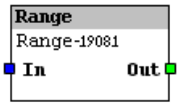

This Function-Block performs arithmetical sum operations.

18. Sub (Subtract) Function-Block:



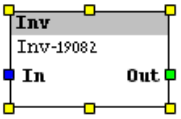

This Function-Block performs arithmetical Subtraction operations.

19. Range Function-Block:



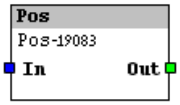

This function routs the **Input** to the **Output,** if the **Input** values are within a predefined range.

20. Inv (Inverter) Function-Block:



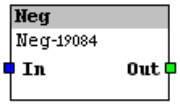

This function inverts the value of the **Input**.

21. Pos (1Positive) Function-Block:



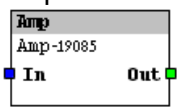

This function routs positive values and turns negative values to 0.

22. Neg (Negative) Function-Block:



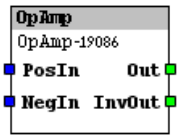

This function routs the absolute value of negative values and turns positive values to 0.

23. Amp Function-Block:



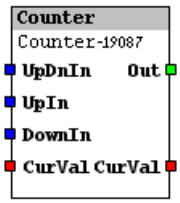

This function is an amplifier.

24. OpAmp Function-Block:



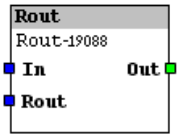

This function is an operational amplifier.

25. Counter Function-Block:




A Counter Function-Block is an up and down counter. It is possible to define the Delta of the counter, and to set an initial value to the count.

26. Rout Function-Block:




This function routs or blocks **Event**s coming from the **Input** to the **Output**.

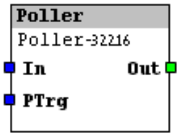
27. Time Function-Block:

This function receives and sends the time.

28. Date Function-Block:

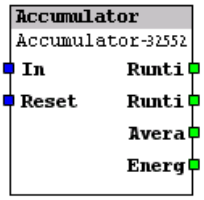
This function receives and sends the Date.

29. Poller Function-Block:



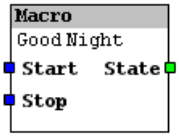

This **Function Block** enables the  polling of data.

30. Accum (Accumulator) Function-Block:




This **Function Block** calculates total run time, sum and average value of the incoming data on constant time intervals.

31. Macro **Function Block**:




A macro is list of **Event**s and delays that the **Device** will perform in a sequence. The Macro **Function Block** functionality:
- Activates a macro
- sends an indication during the time of the macro execution.
- Enables the stopping of the execution of the macro.

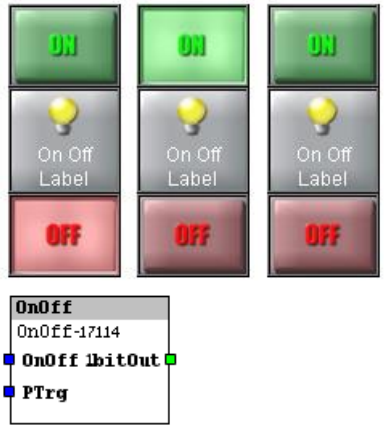
32. Script function block – **the most flexible and power full tool**: free programming using script language enables you to create your own algorithms functions and use the in future projects as well.

CDinnovation

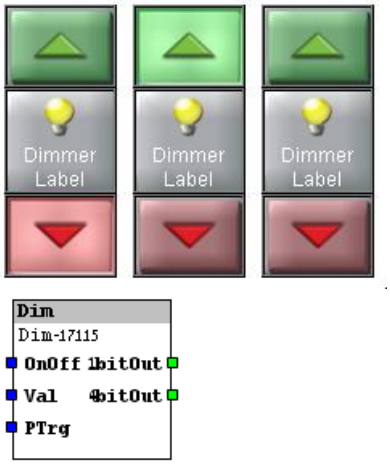
# Appendix B – Buttons

Remark: button design is editable; other, new, predesigned buttons are evaluable.

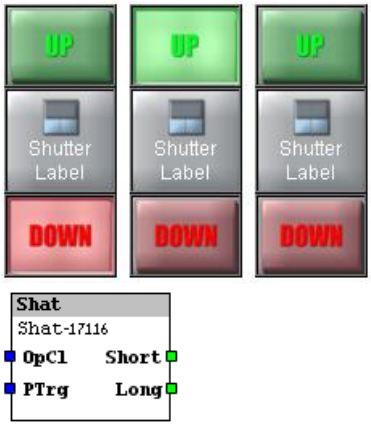
1 On/Off **Button**:
On/Off **Button** is designed especially to control and monitor 1bit data type objects:




2 Dimmer **Button**:
A Dimmer **Button** is designed especially to control and monitor Dimmer actors actions using 1bit, 4bit and 8bit data types. The **Button** has two states, so it can display the On/Off status of the load:

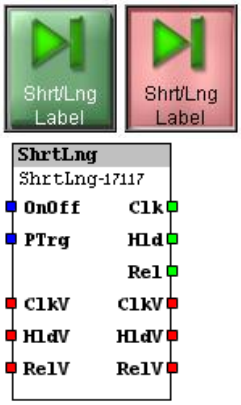



3 Shutter **Button**:
The shutter **Button** is designed especially to control and monitor shutter actors using 1bit telegrams. The **Button** has two states so, if the shutter actor supports the function, it can display the close/open status as well:



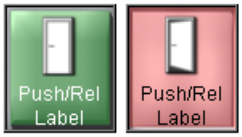

4 Short/Long (Click/Hold) **Button**:

A Short/Long **Button** is a general purpose **Button**. It reacts differently to long pushes and short pushes. It has three different **Output**s; *Hold* and *Release* for long pushes and *Click* for short pushes. You can use the **Output** to transmit any data type and value. The **Button** has two states so it can display On/Off status as well:

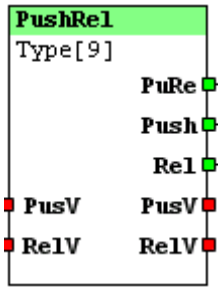



5 Push Release **Button**:

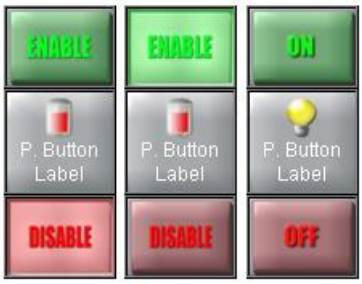
The Push Release **Button** is a general purpose **Button**. It reacts to the push event, and to the release event. You can use it to transmit any data type and value. This **Button** changes its state to the Push state only during the **User** push, and therefore is not capable of displaying status.

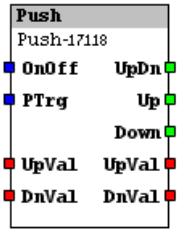



Functional description:

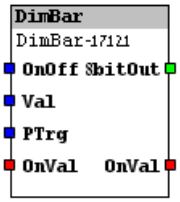



6 Push button **Button**:
The Push Button is a general purpose **Button**. It reacts **only** to push events. You can use it to transmit any data type and value. It has two states so it can display On/Off status as well.



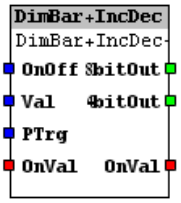




7 Dimmer vertical bar graph (Dimmer Bar Vertical) **Button**:
Dimmer Bar Vertical is designed especially to transmit absolute values used to control Dimmer actors using 8 bit telegrams. You can also use it to display 0 to 100 (%) values:



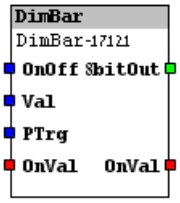

8 Dimmer bar graph + Increase/decrease Vertical (Dimmer Bar Inc/Dec Vertical) **Button**:

The Dimmer Bar Inc/Dec Vertical is designed especially to control and monitor dimmer actor sending 4 bit and 8 bit telegrams. Its central part has the functionality of Dimmer bar **Button**. Its top and bottom parts are push **Button**s made for sending 4 bit control telegrams.

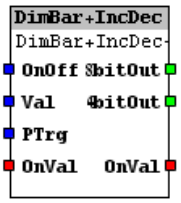




9 Dimmer Horizontal bar graph (Dimmer Bar Horizontal) **Button**:

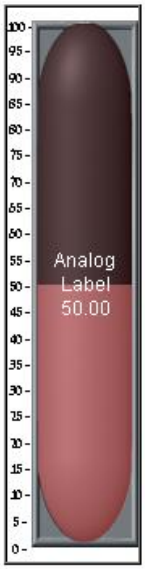
Dimmer Bar Horizontal is designed especially to transmit absolute values used to control Dimmer actors using 8 bit telegrams. You can use it also to display values 0 to 100 (%):

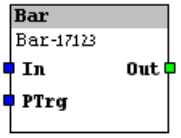



10 Dimmer bar graph + Increase/decrease Horizontal (Dimmer Bar Inc/Dec Horizontal) **Button**: Dimmer Bar Inc/Dec Horizontal is designed especially to control and monitor dimmer actor sending 4 bit and 8 bit telegrams. Its central part has the functionality of Dimmer bar **Button**. its left and right parts are push **Button**s sending 4 bit control telegrams.

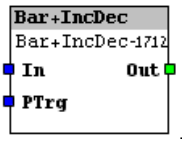

11 Bar Graph (Bar) **Button**:
A Bar is a general purpose **Button.** It is designed to transmit analog values. You can use it to transmit any numeric data type and value. You can use it also to display analog values:



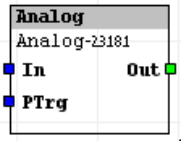




12 Bar Graph + Increase /Decrease (Bar+Inc/Dec) **Button**:

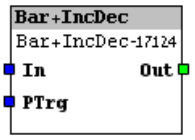
Bar + Inc/Dec Is a general purpose **Button.** It is designed to transmit analog values. **User** can enter new values by pressing on the bar as well as by pressing the upper And lower parts of the **Button**. You can use it to transmit any numeric data type and value. You can also use it to display analog values:

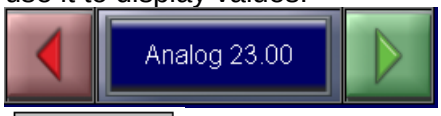

13 alphanumeric **Button**:
An Analog alphanumeric **Button** is a general purpose **Button.** It is designed to transmit analog values. You can use it to transmit any numeric data type and value. You can also use it to display analog values:

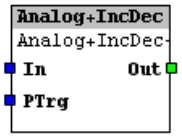




14 Analog Alpha numeric + Inc/Dec Vertical **Button**:
Analog Alphanumeric + Inc/Dec Vertical is a general purpose **Button.** It is designed to transmit analog values. You can use it to transmit any numeric data type and value. You can also use it to display values:



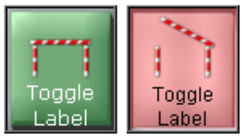

15 Analog Alpha numeric + Inc/Dec (horizontal) **Button**:
An Analog alphanumeric + Inc/Dec is a general purpose **Button.** It is designed to transmit analog values. You can use it to transmit any numeric data type and value. You can also use it to display values:

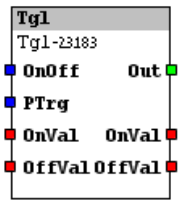



16 Toggle **Button**:

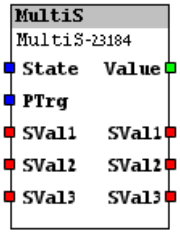
A Toggle **Button** is a general purpose **Button**. It has two states: On and Off, Any time the **User** pushes a Toggle **Button,** it toggles from whichever state it is in to the alternate state. You can use it to transmit any data type and value. It has two states so it can display On/Off status as well.




17 Multi state **Button**:


A Multi state **Button** is a general purpose **Button**. It has one **Output** and up to 10 states. For each state you can define a different output value. There is no limitation to the data type or value. Every state has its own graphic properties so you can use it to display the current state status as well. The **User** switch from one state to another by using the key pad.




18 Time **Button**:
This **Button** shows the time:


19 Date **Button**:
This **Button** shows the date:


20 Text **Button**:


The Text **button** is a background **Button,** therefore it is placed in the *background* layer behind the **Button**s layer. You can use it to create text titles and to mix them with dynamic texts.

21 Rounded Rectangle **Button**:


The Rounded Rectangle **Button** is a Background **Button,** therefore it is placed in the *Background* layer behind the **Button**s layer. It can also display text titles and mix them with dynamic texts.

22 Rectangle **Button**:
A Rectangle **Button** is Background **Button,** therefore it is placed in the *Background* layer, behind the **Button**s layer. It can also display text titles and mix them with dynamic texts.


23 Oval **Button**:
An Oval **Button** is Background **Button,** therefore it is placed in the *Background* layer,

behind the **Button**s layer. It can also display text titles and mix them with dynamic texts.


24 Line **Button**:
A Line **Button** is Background **Button,** therefore it is placed in the *Background* layer, behind the **Button**s layer. You can use it to create horizontal lines in the back ground. It can also display text titles and mix them with dynamic texts.


25 Macro **Button**:


A Macro **Button** is a system **Button**.
A macro is a list of **Event**s and delays that the **Device** will perform in a sequence. The Macro **Button** functionality is to:
- Activate a macro
- Display an indication during the time of the macro execution.
- Enable stopping the execution of the macro.
- open the macro editor.

26 Page flip **Button**:


When this **Button** is pushed it switches the graphic page on display.

CDinnovation

## Appendix C Expression language

### constructions:

- if then else
- do while (break, continue)
- while (break, continue)
- for (break, continue)

### Operators:

```
- BITAND      = "&";          // op1 & op2
- BITOR       = "|";          // op1 | op2
- BITXOR      = "^";          // op1 ^ op2
- AND         = "&&";         // op1 && op2
- OR          = "||";         // op1 || op2
- ADD         = "+";          // op1 + op2
- DIV         = "/";          // op1 / op2
- MUL         = "*";          // op1 * op2
- MOD         = "%";          // op1 % op2
- SHL         = "<<";         // op1 << op2
- SHR         = ">>";         // op1 >> op2
- SHRU        = ">>>";        // op1 >>> op2
- EQUAL       = "==";         // op1 == op2
- LESS        = "<";          // op1 < op2
- GREATER     = ">";          // op1 > op2
- LESSEQUAL   = "<=";         // op1 <= op2
- GREATEREQUAL = ">=";        // op1 >= op2
- INSTANCEOF  = "instanceof"; // op1 instanceof op2
- CAST        = "cast";       // (op2)op1 (without lose of precision)
- CONVERT     = "convert";    // (op2)op1 (possibly lose of precision)
- NOT         = "!";
- MINUS       = "-";
- COMPLEMENT  = "~";
- INC         = "++";
- DEC         = "--";
```