EZ-Red – Power I/O module

GUI (operator panel) editor User manual

Index

Introduction

Preamble

Starting with the version 1.4 of the TSMON application, it is possible to design and deploy user GUIs, or graphic panels, also called HMI, using a computer connected to the EZ-Red. This way, the system acts like a PLC with graphical user interface, suitable for many applications, even for inexpert users. The graphic panel, using specialized widgets, can show PLC internal data, write numeric values into variables and outputs, and start various operations.
Some TSMON command line parameters can bring up the graphic panel immediately; it is also possible to disable operations like stopping the PLC, modifying the cycle, operating the watch-dog, and exiting from the panel mode. This feature was designed in order to give the final user a simple, safe running system.

Employment

For every PLC cycle file (extension .ezCycle), a secondary file is created with extension .ini: this file contains, as well as preferences from last session, the graphic panel (HMI) data designed directly inside the program. The TSMON program is used as an *integrated development environment*, or IDE.

When the application is completely developed, the same TSMON program is used as a graphical user interface (GUI or HMI) instead of an IDE. In normal conditions TSMON starts up as a development tool and, to use the graphic panel, it is necessary to switch to **GUI Panel** tab and click the **RUN** button. By using the following parameters, several operations can be automatized.

/p parameter (switch)

This switch activates immediately the **GUI Panel** and puts the GUI in RUN mode, so there is no need to do those operations interactively. All the other operations remain possible. Note: it is a lowercase “p”.

/r parameter (switch)

This switch makes TSMON transmit the PLC cycle loaded from disk to the connected EZ-Red, and runs it. Normally this switch is accompanied by another parameter specifying the file to load (extension .ezCycle). If no file is specified, TSMON tries to load the file DEFAULT.ezCycle. Note: it is a lowercase “r”.

File name parameter

A parameter (*switch*) not starting with a slash (“/”) is assumed to be the name of a file to load; this name should also contain the extension .ezCycle.

By using all the tree parameters above on the same command line, it is possible to load a PLC cycle, compile it, send it to the EZ-Red, run it, and show the operator panel (OP Panel). For example, the following command:

TSMON /r /p quality.ezCycle

when executed from a batch file or a shortcut, starts an application to do a quality test.

Customer mode

If the application has to be used by an inexpert user, it is possible to simplify further the operations. By duplicating the TSMON program, and giving the duplicate the same name as the cycle to be executed, no further operations are to be carried out by the user, and no batch file or shortcut is needed; the program itself acts as a user interface without other operation allowed:

1. Create a copy of TSMON.EXE
2. Rename the copy to, for example, EXAMPLE.EXE.
3. Give the final user the following files:

    example.exe

    ezreddll.dll

    tsmon.inc

    example.ezCycle (optional)

    example.ini

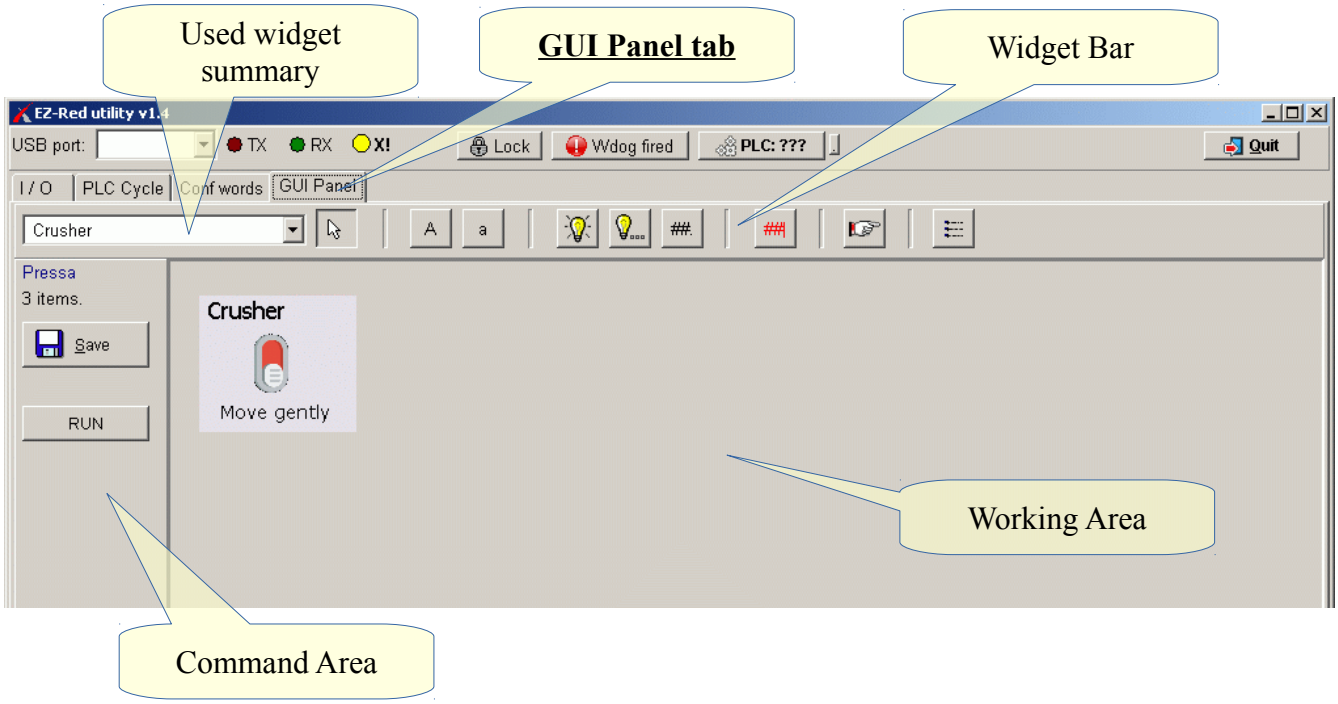
The user will be able to run the application by simply launching the program EXAMPLE.EXE with a pre-programmed EZ-Red connected to the computer.

Note: the **/r** switch works even if the TSMON program has been renamed, but in this case the optional source file (with extension .ezCycle) has to be present.

The operator panel editor

Introduction

The panel editor is the fourth tab (page) of the application, labeled "**OP Panel**":



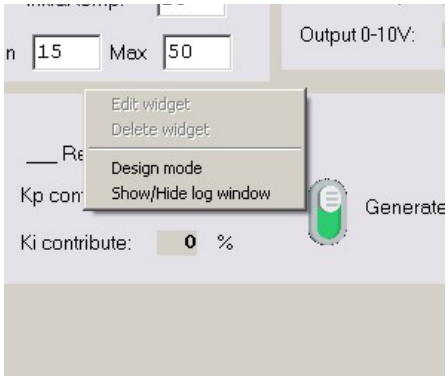
The panel editor is strictly bound to the cycle source in the "**PLC Cycle**" tab: any editing made to the cycle is reflected into the panel, logically. When setting options to the various widgets, a combo box summarizes all the PLC resources defined in the cycle; this directory is generated when the PLC cycle source is being compiled; to be sure to have it up to date, please compile the cycle before editing the panel widgets.

Panel data and PLC cycle are linked even in another aspect: saving one will save the other too: the **Save** button in the Command Area of the OP Panel does exactly the same thing as the **Save** button in PLC cycle tab, and vice-versa.

RUN mode and Design mode

The **GUI Panel** tab is always in one of two states: design state or RUN state. In the former it is possible to edit the panel data (add/edit/delete widgets); in RUN state the widgets interact with the user and the EZ-Red and can not be modified.

Panel context pop-up menu

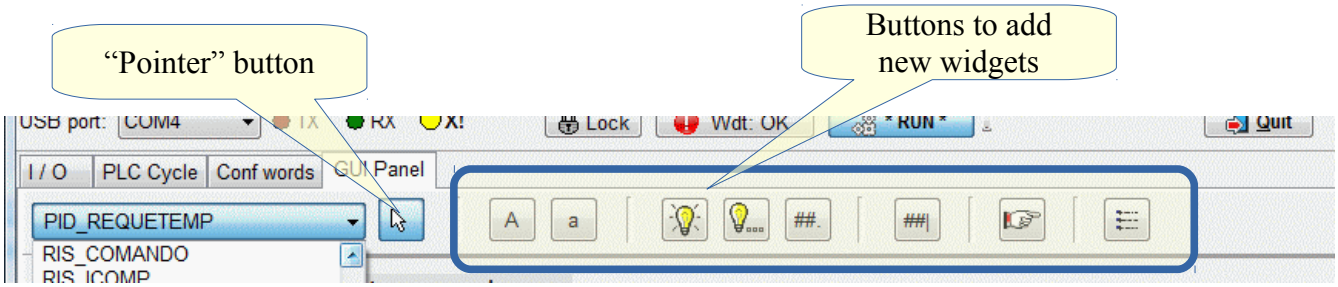


By right-clicking into the Working Area, on a widget, or on the widget directory, a context menu appears. The "**Edit widget**" item is to edit a widget, and "**Delete widget**" to delete a widget. The "**Design mode**" item exits from RUN mode (back to design mode), usefule especially when the Command Area is hidden.

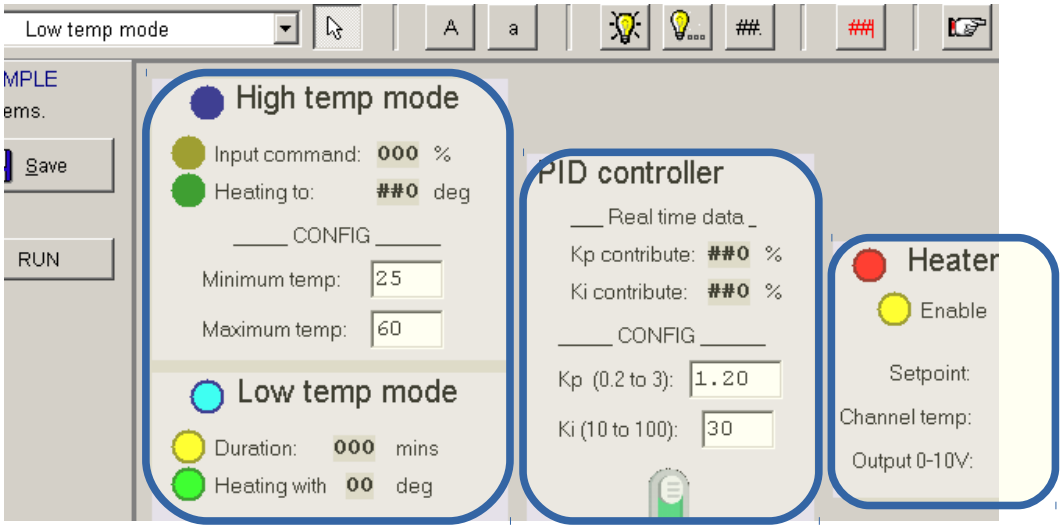
The "**Show/Hide log window**" item shows or hides the log pane, where debug messages and log entries are recorded.

Widget insertion

In order to add a widget to the panel, click one of the widget buttons to the right of the "pointer" button, to make it depressed, then click in a free zone of the Working Area. A dialog window opens to set the new widget properties. The "pointer" button serves to exclude this behavior, like a "null" widget.

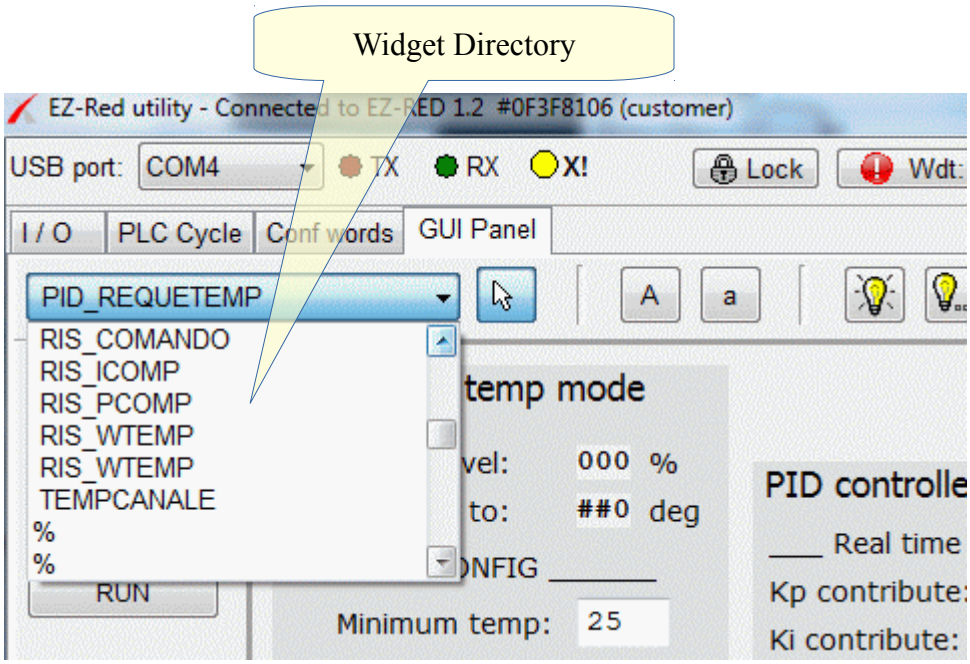


The widget addition is possible only if clicking in an empty/clear area of the Working Area; by clicking instead on a widget, the pointer button activates and no widget is added. This is especially important when dealing with *Panel labels*, which often occupy large areas on the screen. The image below shows an example of non-clear areas, occupied mostly by *Panel Labels*:



Widget editing

It is possible to edit a widget by operating directly on it in the Working Area; otherwise, it is possible to select it in the Widget Directory, and then right-click on the directory itself.:



The Widget Directory is necessary because there are widgets that are not visible in the Working Area, like the *Logger Entry* (custom log messages).

Mouse Right Button

By right-clicking on a widget in the Working Area, a pop-up menu appears; select "**Edit widget**" to edit the widget, or "**Delete widget**" to delete the it. The same thing is possible by right-clicking on the Widget Directory, to edit or delete the widget currently displayed in the directory itself.

Mouse dragging

Widget position and size can be modified by dragging the mouse in the Working Area. If the operation is made without keeping the **Ctrl** key pressed on the keyboard, the position of the widget is modified (classic drag operation). If the **Ctrl** key is pressed while dragging, the size of the widget is modified. In order to easen this second operation, it is best to press and keep down the **Ctrl** key, then click in the bottom-right corner of the widget; this corner can be dragged in the desired position.

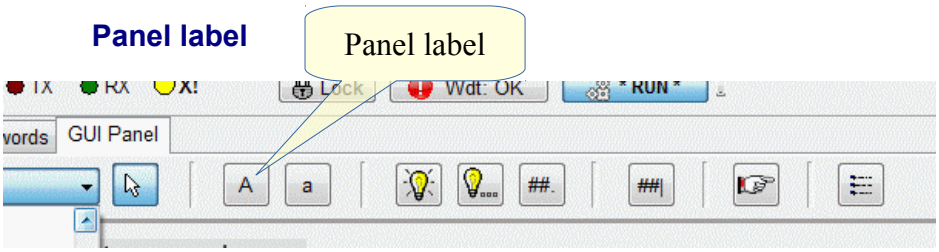
The *Panel label* widget is also special because, when dragged, it drags together all the other widget sitting inside it.

Double click

By simply double-clicking a widget in the Working Area, it is possible to edit its properties.

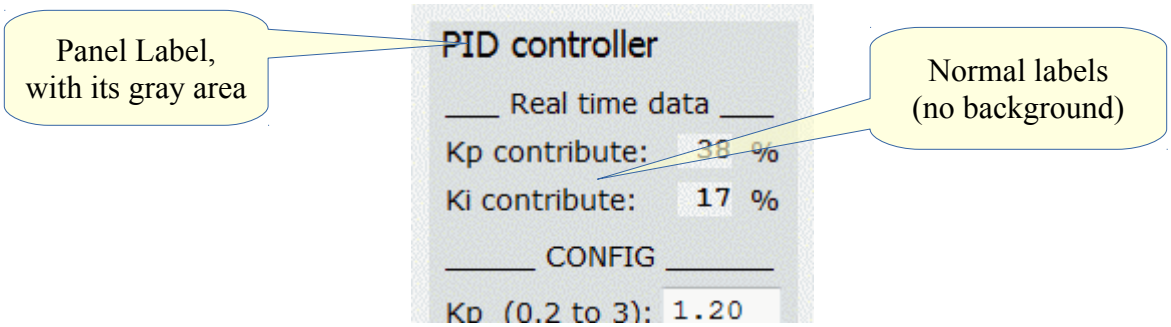
Available widgets

From left to right, the Widget Bar contains the following buttons:



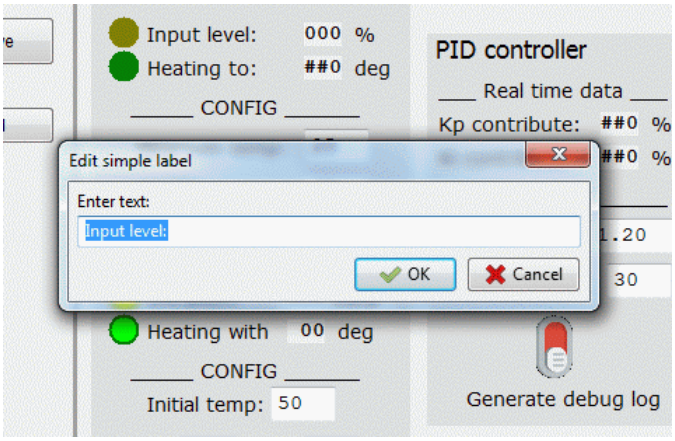
This is a label with a visible background, intended to group / contain other widgets; it is possible this way to create sub-panels to give a logical layout to the graphic panel. When a *Panel Label* is dragged, all the widgets inside is are dragged too.

The only property of this widget is its text, printed with bigger characters than the other widgets:



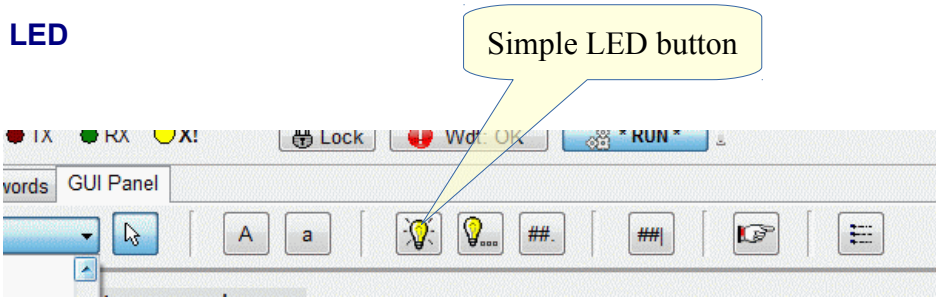
Simple Label

The next button, marked "**a**" instead of uppercase "**A**", creates a *Simple Label*; the only property is its text. For both the types of label, the following box appears when their text must be entered:

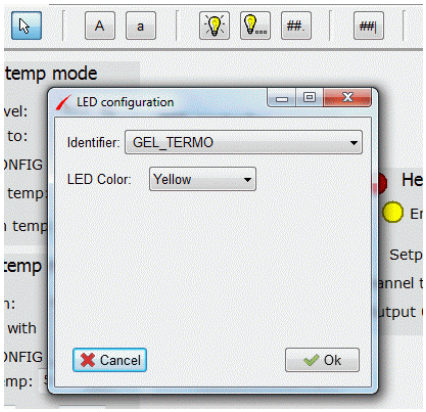


Click "**OK**" to confirm the text, or "**Cancel**" to leave it untouched..

Simple LED



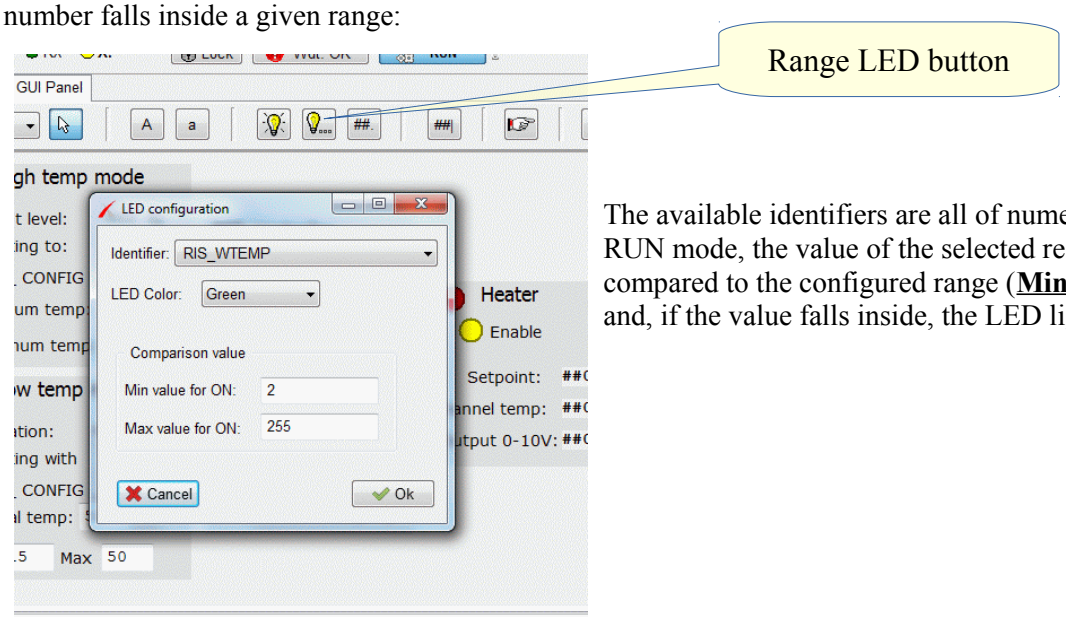
The *Simple LED* purpose is to show the content of a boolean value; it must refer to a bit resource, like a digital I/O or virtual relay; inserting or editing a *Simple LED* brings up the following window,



which shows a directory of resources to choose from and the desired color of the LED among Yellow, Red, Green and Blue. In RUN mode, the LED lights up when the referred resource has value equal to 1 or True/On. The "**Identifier**" directory contains, in this case, only bit/boolean resources. The identifiers are sorted, and the user defined ones come first.

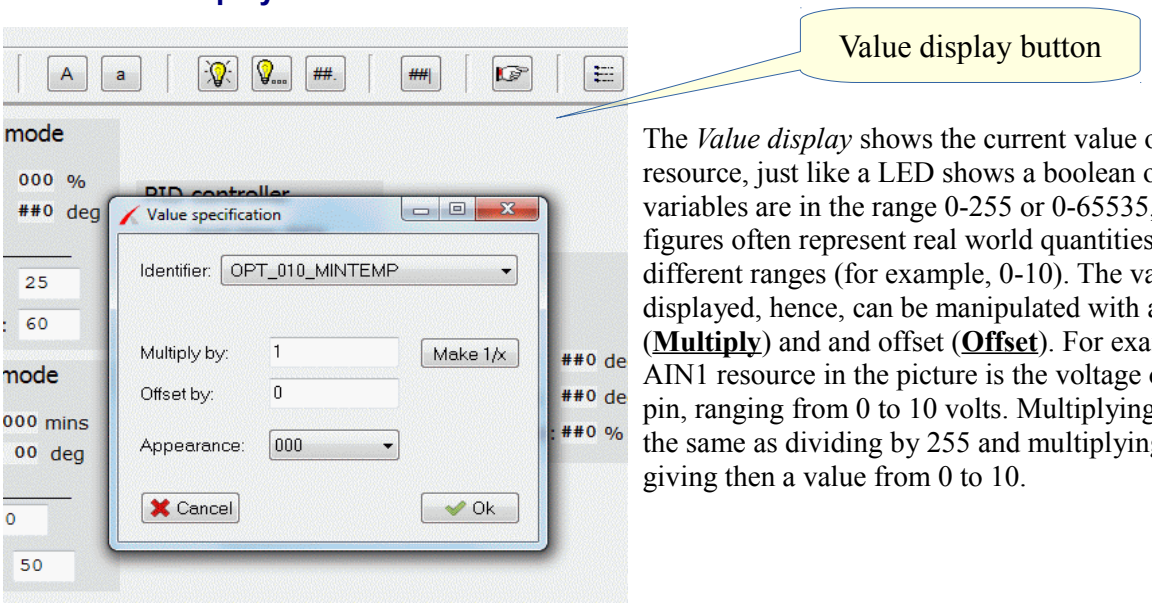
Range LED

The *Range LED* is similar to a normal LED, but it is associated to a numeric resource and it lights up when the number falls inside a given range:



The available identifiers are all of numeric kind; when in RUN mode, the value of the selected resource is continuosly compared to the configured range (**Min value** - **Max value**) and, if the value falls inside, the LED lights up.

Value display



The *Value display* shows the current value of a numeric resource, just like a LED shows a boolean one. EZ-Red variables are in the range 0-255 or 0-65535, but these figures often represent real world quantities which use different ranges (for example, 0-10). The value to be displayed, hence, can be manipulated with a multiply (**Multiply**) and and offset (**Offset**). For example, the AIN1 resource in the picture is the voltage of the AIN1 pin, ranging from 0 to 10 volts. Multiplying for 0.0391 is the same as dividing by 255 and multiplying for 10, giving then a value from 0 to 10.

For convenience there is a "**Make 1/x**" button, which calculates the reciprocal of the number in the field.

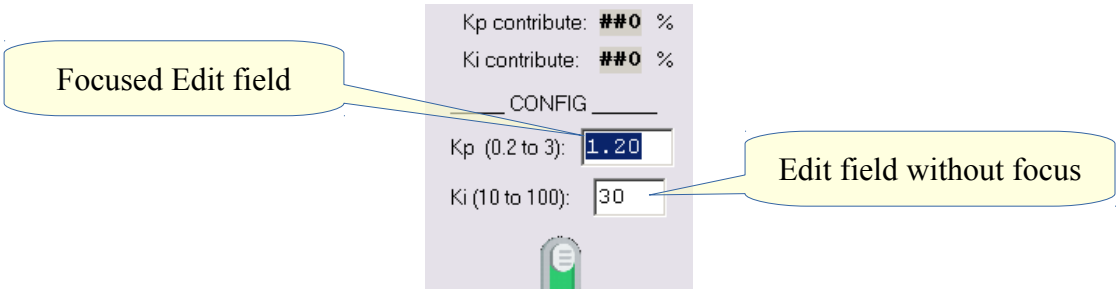
If scaling is not necessary, use neutral numbers for the two fields: 1 for "**Multiply by**" and 0 for "**Offset by**".

The **Appearance** field is to specify a format to display the number (how many integral and decimal digits). In this field, a "0" digit means that in that position a digit will always be displayed, even if not needed, while a hash "#" will display the 0 digit if only when needed. The number 12.3 can be displayed in different ways depending on the value of the **Appearance** field:

00.0	will display 12.3
000	will display 012, using always 3 digits and no decimals
##0	will display 12, omitting the first "0" and the decimals

Edit field

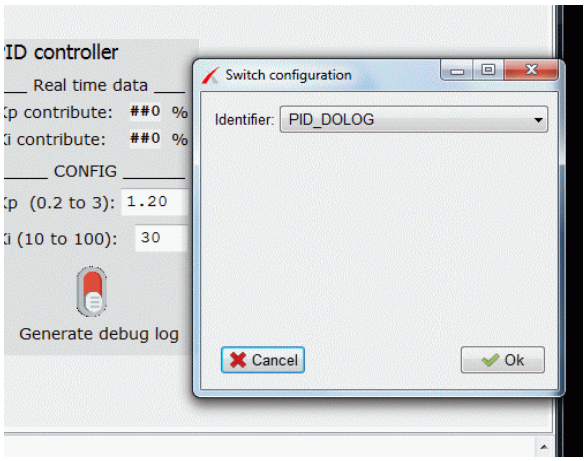
This widget displays a numeric value and allows to edit it:



To edit a numeric value in an *Edit field*, click it (or use the **Tab** key), write the new value, and press the **Enter** key. The decimal digits must be separated by a decimal dot, regardless of the country convention. The properties of this widget are identical to those of *Value display*: the number manipulation is performed in the opposite way to convert the user inputted number to the EZ-Red internal range of 0-255 or 0-65535.

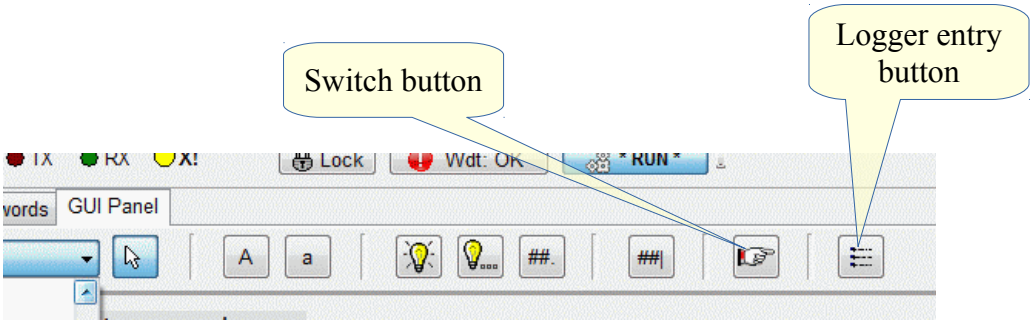
Switch

The *Switch* widget is the counterpart of the LED widget: in addition to display the boolean value of a resource, it can modify (negate) it:



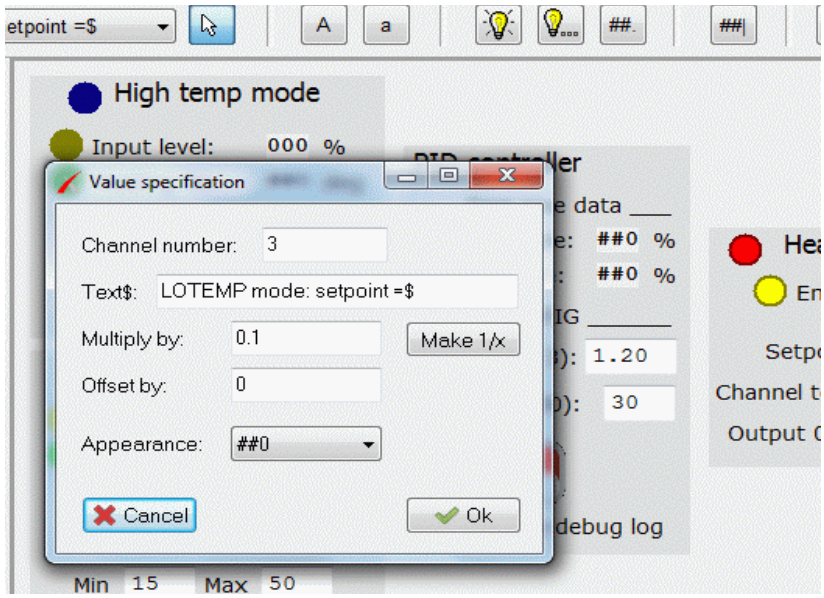
The properties dialog is the same of a LED, but it is not possible to choose a color for the *Switch*, it only shows the boolean identifier associated.

When, in RUN mode, the switch is clicked, its state gets inverted and the new state is sent to the connected EZ-Red. It is responsibility of the PLC cycle to manage this variation.



Logger entry

When the PLC cycle executes a LOG instruction (LOG *channel value*), the two numeric values are sent to the TSMON program. Using a *Logger entry* it is possible to associate every channel to a different text and a different conversion formula. So, it is possible to display meaningful messages even to inexpert users.



The configuration dialog is very similar to those of *Value display* and *Edit field*, with the same conversion values and format option. There are two other fields to select the channel ("**Channel number**") and the text to display ("**Text$**"). The **Text$** should contain a dollar sign ("$") in the position were the numeric value must be placed; if no dollar sign is present, the numeric value is discarded and the text will be displayed as is.

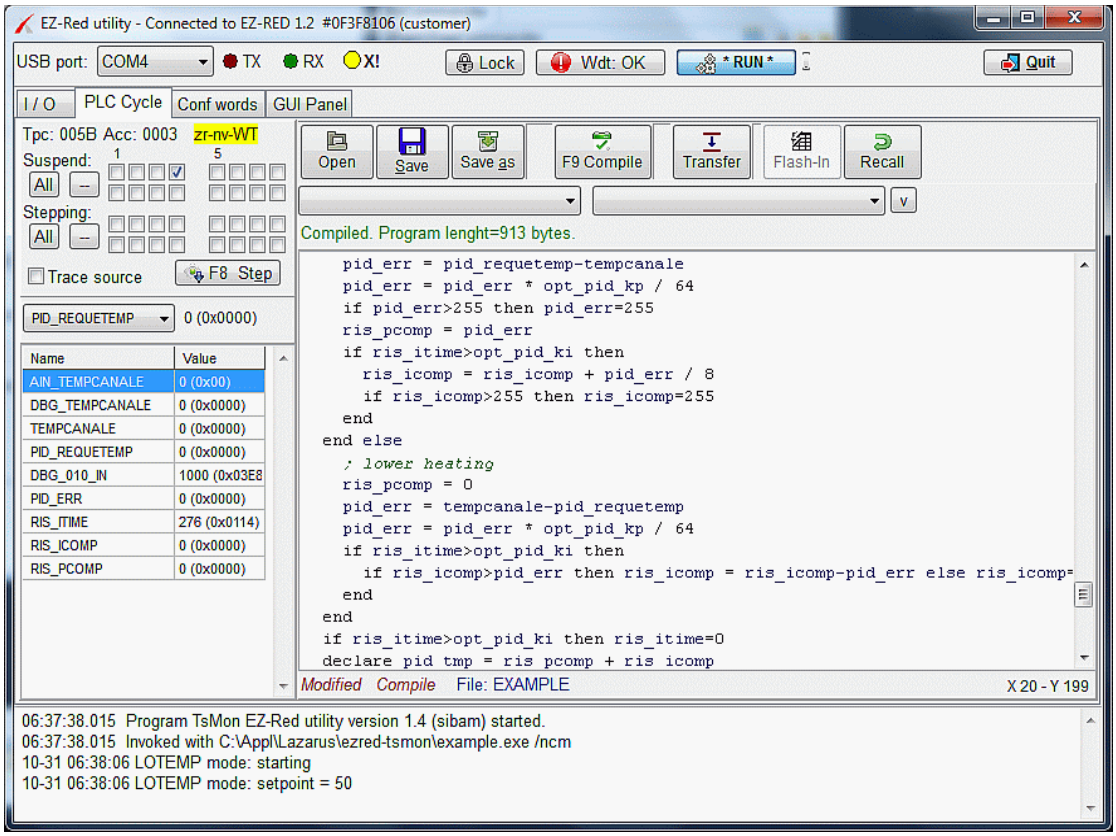
A complete example

The following example is a real world case of heating control for a special furnace. The furnace has two working modes, selectable by a switch connected to the X2 digital input. In the High Temperature mode, a potentiometer connected to AIN1 regulates the desired temperature; in the Low Temperature mode the temperature starts from a level and rises up to a maximum, then falls to a minimum. Both the modes select, in every moment, a desired temperature; a PID controller, implemented by the EZ-Red PLC cycle, modulates the heater to maintain the desired temperature in the furnace.

The first thing was to write the PLC cycle; a series of DEFINE and DECLARE was used to give meaningful names to the I/O pins (DEFINE), and to the variables (DECLARE). At the same time, for every identifier a suitable range of values was chosen; for example, the temperatures are expressed internally as tenths of degree: when a variable has a value of 200, its meaning is a temperature of 20 degrees. In other cases, a range of 0-255 has been mapped to 0-10 (volts).

A few variables (configuration) must not lose their content when the system is powered off. They are mapped, using the DEFINE statement, to resources such CONFIGCHR CONFIGBIT and CONFIGWRD, which are perfect for this task, and by using DEFINEs the designing of the operator panel was greatly simplified.

The following picture shows a fragment of the PLC source code:

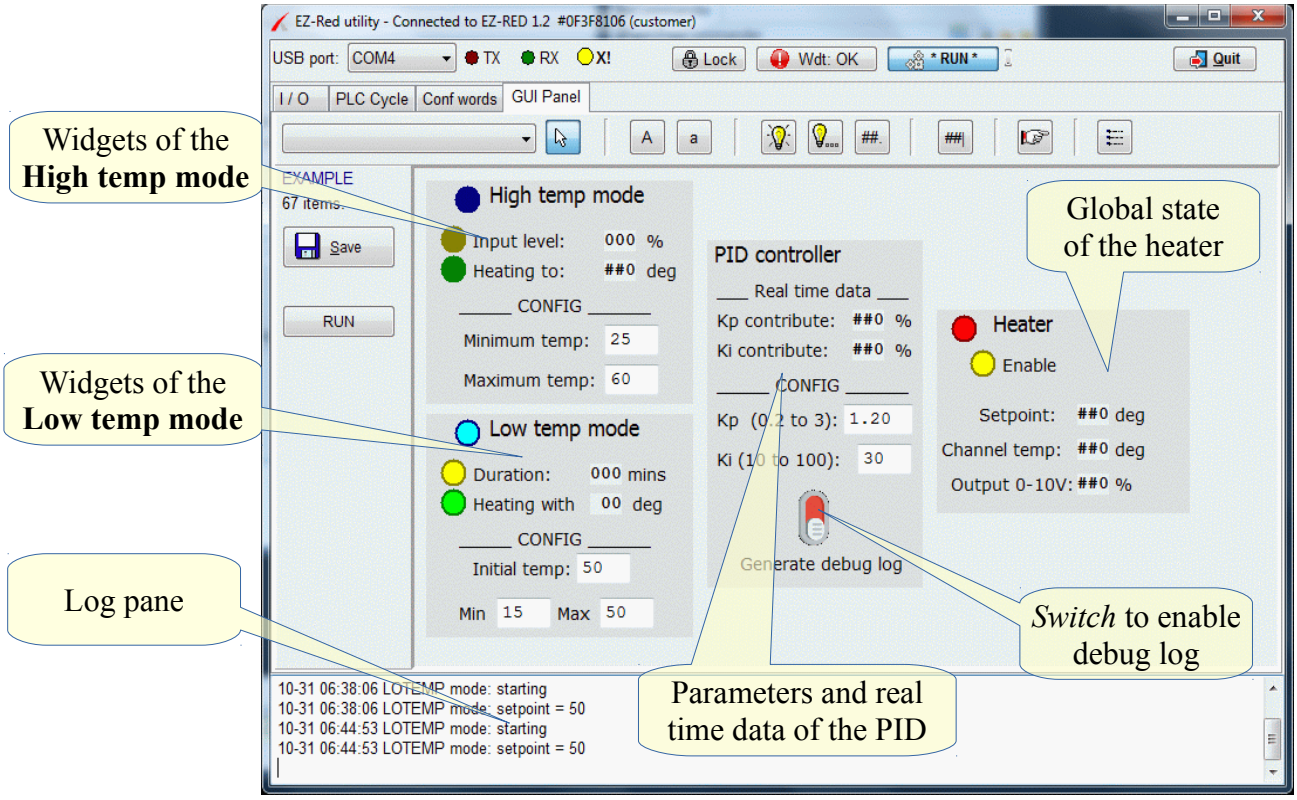


The PLC cycle is made up of 4 *Tasks*: the first is the general supervisor, which starts or stops the other three; the second implements the *High temp* mode, the third the *Low temp* mode, and the fourth the PID controller.

Designing the panel, all the widgets necessary to show and control the cycle have been created, and put into sub panels (*Panel labels*) to obtain a logical subdivision respecting the underlying logics.

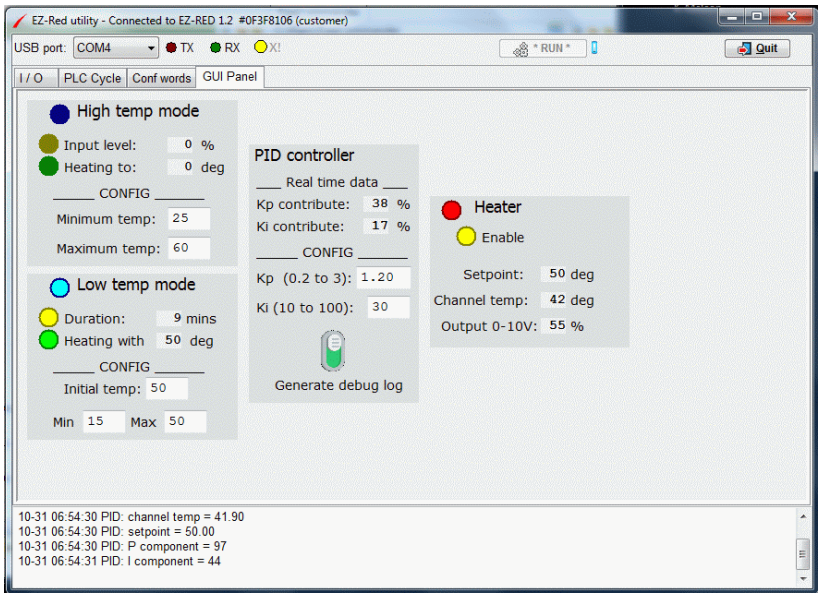
For every functional part of the PLC cycle the relevant widgets have been created and adjusted; then the widgets have been grouped into the sub panels.

The final work has the following look:



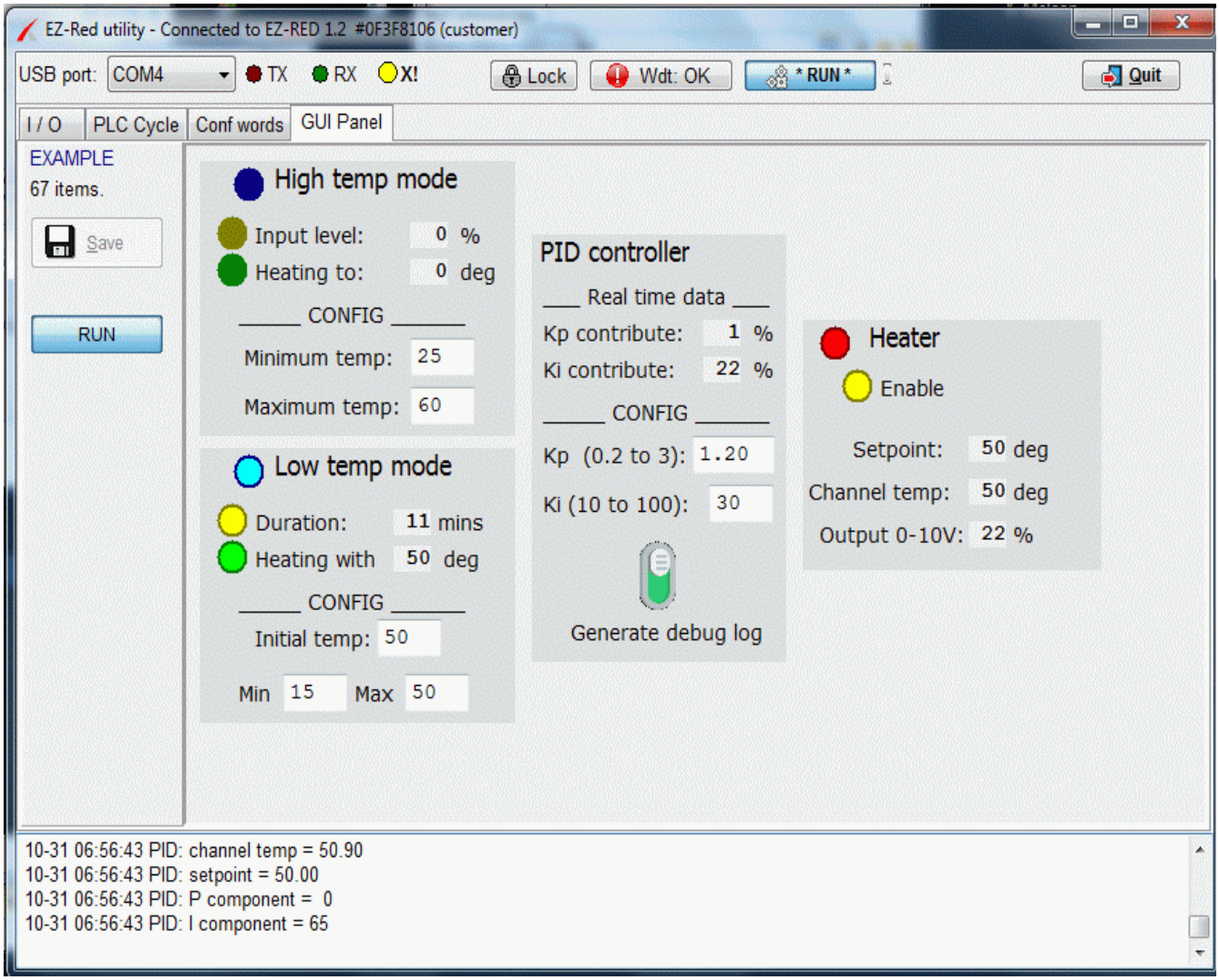
Two virtual relays, always opposite to each other, reflect the current state of X2 which selects the working mode. The two blue LEDs **High temp** and **Low temp** are associated to those relays. The yellow LED **Input command** is a *Range LED*, lighting up when the associated value, AIN1, has a value comprised in the range 2-255; the red LED **Heater** is similar, associated to the 0-10V output which controls the heater. The other widgets are mostly simple labels used to describe the other *Value display* and *Edit field*.

The system is operated mainly through discrete switches and potentiometers mounted on the rack, while the GUI is meant to only show data, so there is normally no keyboard or mouse connected. When installing, to help to determine the correct values for the **Kp** and **Ki** parameters, it is useful to turn on the **Generate debug log** switch; when active, it will log in the message pane, every 30 seconds, the actual internal data of the PID Controller, as seen in the picture below:



The picture shows a heating phase where the setpoint is 50 degrees while the temperature is 42 degrees.The ***P*** component has a value of 38%, while the ***I*** component has about 17%. The four log messages, emitted because of the **Generate debug** switch is turned on, show the same data (internal PID state).

After a short time, the system stabilizes itself to the desired setpoint:



In the log panel, the ***I*** component value reads 65 which, in the scale 0-255, results in the 22% displayed in the GUI panel. It would be possible to scale the log message value but, because this is a *debug* message, it was chosen to show simply the real internal values.

XON ELECTRONICS SRL
WWW.XONELECTRONICS.IT
INFO@XONELECTRONICS.IT

Please report any error or imprecision to web@xonelectronics.it