Scriptable LED panel

User Documentation for this variants:

Ethernet (HTTP GET API);

Ethernet (HTTP klient);

Ethernet (MODBUS/TCP Slave)

EGMedical, s.r.o. Filipinskeho 1534/55, 615 00, Brno CZ

www.egmenergo.cz

2023

Table of Contents

Version of the Manual as of 10/18/23, 16 pages.

Prepared by Bc. A. Hruškovský; Aleš Jílek; Bc. Martin Sekáč

1. Preface

Dear Customer,

We sincerely thank you for choosing EGMedical, s.r.o. products.

Our company's products stem from years of experience in the development and manufacturing of electronic devices across a wide range of fields, including electronic systems, voice applications, industrial control, robotics, automation, telecommunications, and communication technology. This guide will assist you in installing, using, and maintaining the product correctly. We are confident that EGMedical's product will serve you without any issues.

Before leaving our development laboratories, our products undergo comprehensive functionality and quality testing. However, if you encounter any issues with our goods, we are here to help you resolve them.

EGMedical provides a warranty for all its products, which applies only to products used in accordance with the instructions and safety guidelines. Interventions and repairs to the products should be performed by authorized EGMedical technicians, unless explicitly stated otherwise. Please note that adjustments to product settings or hardware system interventions may significantly affect its performance and lifespan.

This user manual has been prepared based on our knowledge and experience. Please be aware that our products are constantly evolving and improving, so you may encounter modifications in the future that are not described in this manual.

Sincerely,

Ing. Ivo Strašil, R&D manager | CEO

2. Web Interface

For control and configuration of the display, a simple web interface is provided. The web interface allows you to set the desired information for display, system details, and network settings. In the following text, individual settings will be discussed in more detail.

To access the web interface, you need to know the device's IP address and password. By default, these details are set as follows:

- IP address: ***192.168.1.206***
- Password: ***test***

2.1. Displaying Information on the Display

Information displayed on the display is entered using a LUA script. LUA scripts are written directly within the web interface, where an editor is provided to process these scripts. The editor also provides syntax highlighting. Scripts can be created, edited, and deleted.

Before reaching the page with the LUA script editor, click on the Display button and enter the name of the new LUA script. Choose Advanced script, which allows for more complex scripts and defining custom functions, or a Simple script, in which only a defined set of user functions can be used.

Creating a script will add a new entry to the script list, and you will be automatically redirected to the page with the LUA editor, where the script body will be pre-filled with sample code.

The user modifies the displayed information by editing the script using the available functions, along with their descriptions available on the web page. The list of functions for a Simple script, along with descriptions, is provided in the documentation in section 3.1. The list of functions for an Advanced script, along with descriptions, is also provided in the documentation in section 3.2.

After creating the script, click the Save button located in the top-right corner of the screen under the Script editor title, next to the name of your script. Return to the screen with the table of all scripts by clicking the Display button in the user menu. Select the newly created script and click Save.

If the code is valid, it will be executed immediately, and the display will render accordingly. If there is an error in the code, you will be informed through an error message located below the script table. An example of a possible error is shown in the following image.

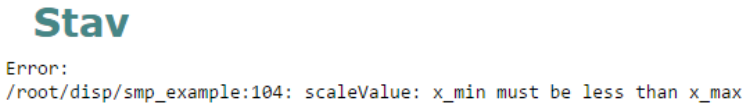

Fig. 2.1: Example of an Error Message

2.2. System Settings

To access the system settings, click on the "System" item in the user menu. The first setting is the Time Zone. If the device is connected to the internet, the date and time are automatically set using the NTP protocol for the selected time zone.

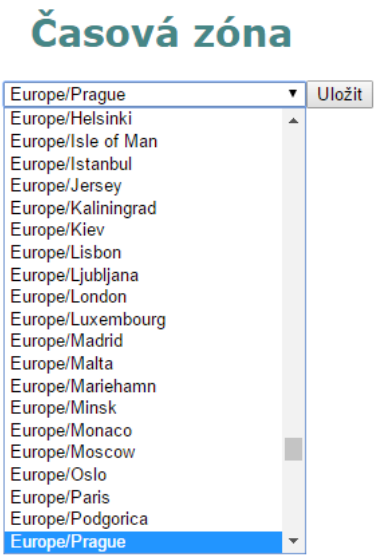

Fig. 2.2: Time zone selection

To apply the time zone change, it's necessary to restart the device. This can be done by clicking on the "Restart Device" button in the Reset section.

Fig. 2.3: Restart device button

In case you want to set the time manually, there is a form available for manual configuration. In the manual time setting, you can input the date in the format DD.MM.YYYY and the time in the format hh:mm. However, time set through NTP takes precedence and has priority over manual configuration.

Fig. 2.4: Form for Manual Date and Time Configuration

Access to the web interface requires password authentication. The password can be set or changed in the Login section. When entering a new password, a double check is used to prevent unintended typos.

Fig. 2.5: Form for Changing Password

2.3. Network Settings

To access the network settings, click on the "Network" button. At the top of the screen, you will see information about the network interface, assigned IP address, sub-net mask, etc. This is a display of the Linux command "ifconfig eth0."

Below this display, you will find a form for configuring the network interface. You can use this form to set your own IP address, sub-net mask, default gateway, and DNS server address. The default settings are shown in the image. 2.6.

Fig. 2.6: Default network interface settings

Changing the IP address, sub-net mask, and default gateway will take effect immediately. However, changing the DNS server requires a device restart to take effect.

3. List of functions

3.1. Simple Script

The table provides a list of all supported functions in a simple script along with a description of each function.

DispRed()	Setting the red color.
The text displayed after this function will be in red color	
DispGreen()	Setting the green color.
The text displayed after this function will be in green color	
DispYellow()	Setting the yellow color.
The text displayed after this function will be in yellow color.	
DispNextLine()	Set position to the beginning of the next line.
Text displayed after this function will be shown from the start of the next line.	

DispLinePosition(l)	Set the line position.
The function has one input parameter: l: line number, from the top. Zero is the first line.	
DispAdd(t)	Display user text.
The function has one input parameter: t: user text to be displayed on the display.	
DispSetX(x)	Set the X position on the line.
The function has one input parameter: x: position on the line. The value is limited by the display resolution.	
DispAddReg(reg)	Display the content of a register on the display.
The function has one input parameter: reg: register number. Value is limited to <0, 99>. Entering a number outside the allowed range will display 0.	
DispAddReg32s(reg, endian)	Display the content of a register as a 32-bit signed number on the display.
The function has two input parameters: reg: register number. Value is limited to <0, 98>. Entering a number outside the allowed range will display 0. endian: endianness selection. True – BigEndian, False – LittleEndian.	
DispAddReg32u(reg, endian)	Display the content of a register as a 32-bit unsigned number on the display.
The function has two input parameters: reg: register number. Value is limited to <0, 98>. Entering a number outside the allowed range will display 0. endian: endianness selection. True – BigEndian, False – LittleEndian.	
DispTime()	Display time on the display.
The function has no input parameters. It displays the current time in hours and minutes on the display (hh:mm).	
DispDate()	Display date on the display.
The function has no input parameters. It displays the current date on the display (DD.MM.YYYY).	
DispAddBit(reg, tt, tf)	Display the requested message on the display based on the value of a bit register.
The function has three input parameters: reg: register number. Value is limited to <0, 99>. Entering a number outside the allowed range will display the message stored in parameter **tf**. tt: message to be displayed if the register contains a True value. tf: message to be displayed if the register contains a False value.	
getBit(i)	The function returns the value of a specific bit register as a Boolean.
The input parameter is the register number <0, 99>. Entering a register address outside the specified range will return False. Example: bit = getBit(1) -- Reading the value of a bit	

getRegister(i[, b])	The function returns the value of a specific Modbus register.
The first, mandatory, parameter is the register number <0, 99>. Entering a register address outside the specified range will return 0. The second, optional, parameter is a Boolean: True: returns the number as signed False: returns the number as unsigned (default) Example: nez = getRegister(1) -- Reading the value of register 1 zna = getRegister(1, true) -- Reading the value of register 1	
getReg32u(i[, e])	The function returns the value of the specified and the following register as a 32-bit unsigned number.
The first mandatory parameter is the register number <0, 98>. Entering a register address outside the specified range will return 0. The second optional parameter is a Boolean: True: returns the value in BigEndian format False: returns the value in LittleEndian format (default) Example: nezn32le = getReg32u(1) -- Reading the combined value of registers 1 and 2 nezn32be = getReg32u(1, true) -- Reading the combined value of registers 1 and 2	
getReg32s(i[, e])	The function returns the value of the specified and the following register as a 32-bit signed number.
The first mandatory parameter is the register number <0, 98>. Entering a register address outside the specified range will return 0. The second optional parameter is a Boolean: True: returns the value in BigEndian format False: returns the value in LittleEndian format (default) Example: zn32le = getReg32s(1) -- Reading the combined value of registers 1 and 2 as signed zn32be = getReg32s(1, true) -- Reading the combined value of registers 1 and 2 as signed	
scaleValue(v, o1, o2, n1, n2)	The function returns a scaled value from the current range to a new range.
The function has five input parameters: v: numerical value to be scaled o1: numerical value, lower bound of the original range o2: numerical value, upper bound of the original range n1: numerical value, lower bound of the new range n2: numerical value, upper bound of the new range Example: value = scaleValue(5, 0, 10, 0, 100) -- Converts the value 5 from the range <0,10> to the range <0,100> (50)	
getDec(v, d)	The function returns the value of the specified decimal number with the desired number of decimal places.
The function has two input parameters: v: original numerical value d: number of decimal places, can also be added Example: value = getDec(666.456332, 3) -- Returns the value with 3 decimal places	

3.2. Advanced Script

Here is a list of supported functions for the advanced script. In an advanced script, the function update_text(counter) must be provided, which handles the display output.

update_text(counter)	Updating text on the display.
This function is automatically called with each overflow of the counter set by the setPeriod(t) function. It has one parameter: counter: a number that increments by one with each execution. Example: function update_text(counter) local text text = "\\x056P " text = text .. getDec(0.456332 + counter, 3) return text	

The following table contains the display and description of the supported functions:

setPeriod(t)	Setting the update text period.
The input parameter is an integer. The number ranges from 1 to 50, which corresponds to range from 100 ms to 5000 ms. Providing a number outside this range will result in an error. After the timeout expires, the update_text(counter) function is called. Example: setPeriod(10) -- Period set to 1 s	
getRegister(i[, b])	This function returns the value of a specific Modbus register.
The first mandatory parameter is the register number <0, 99>. Entering a register address outside this range will result in returning 0. The second optional parameter is a Boolean: True: returns the number as signed False: returns the number as unsigned (default value) Example: nez = getRegister(1) -- Read the value of register 1 zna = getRegister(1, true) -- Read the value of register 1	
getBit(i)	This function returns the value of a specific bit register as a Boolean.
The input parameter is the register number <0, 99>. Entering a register address outside this range will result in returning False. Example: bit = getBit(1) -- Read the value of the bit	
getReg32u(i[, e])	This function returns the value of the specified and the following register as a 32-bit unsigned number.
The first mandatory parameter is the register number <0, 98>. Entering a register address outside this range will result in returning 0. The second optional parameter is a Boolean: True: returns the value in BigEndian format False: returns the value in LittleEndian format (default value) Example: nezn32le = getReg32u(1) -- Read the combined value of registers 1 and 2 nezn32be = getReg32u(1, true) -- Read the combined value of registers 1 and 2	
getReg32s(i[, e])	This function returns the value of the specified and the following register as a 32-bit signed number.
The first mandatory parameter is the register number <0, 98>. Entering a register address outside this range will result in returning 0. The second optional parameter is a Boolean: True: returns the value in BigEndian format False: returns the value in LittleEndian format (default value) Example: zn32le = getReg32s(1) -- Read the combined value of registers 1 and 2 zn32be = getReg32s(1, true) -- Read the combined value of registers 1 and 2	

getTime()	This function returns three values in the order: hours, minutes, seconds.
The function has no input parameter. Example: h, m, s = getTime() -- Read hours, minutes, and seconds	
getDate()	This function returns 4 values in the order: day, month, year, day of the week.
The function has no input parameter. Days of the week are ordered: 0 – Sunday 1 – Monday 2 – Tuesday 3 – Wednesday 4 – Thursday 5 – Friday 6 – Saturday Example: d, m, r, w = getDate() -- Read day, month, year, and day of the week	
scaleValue(v, o1, o2, n1, n2)	This function returns a scaled value from the current range to the new range.
The function has five input parameters: v: numerical value to be scaled o1: numerical value, lower bound of the original range o2: numerical value, upper bound of the original range n1: numerical value, lower bound of the new range n2: numerical value, upper bound of the new range Example: value = scaleValue(5, 0, 10, 0, 100) -- Scale value 5 from range <0,10> to range <0,100> (result: 50)	
getDec(v, d)	This function returns the value of the given decimal number with the desired number of decimal places.
The function has two input parameters: v: original numerical value d: number of decimal places, can be extended Example: value = getDec(666.456332, 3) -- Returns value with 3 decimal places	

3.3. Remote Text Entry

The device enables direct text entry without the use of a scripts. This can be achieved using an HTTP GET request. Depending on the text encoding, one of the following variants can be used:

http://192.168.1.206/script/text1250?test%20text - Win1250 text encoding

- http://192.168.1.206/script/text1250 – returns the last manually sent text

http://192.168.1.206/script/textUTF8?test%20text - UTF8 text encoding

- http://192.168.1.206/script/textUTF8 – returns the last manually sent text

Both of these variants allow the brightness to be changed. Brightness changes can be achieved by adding the ASCII character 0x7F followed by a brightness value of 0-9 (0 = minimum brightness; 9 = maximum brightness). This combination of characters can only be used at the beginning of the text. Examples:

- http://192.168.1.206/script/text1250?%7F4test%20text - Win1250; brightness 4
- http://192.168.1.206/script/textUTF8?%7F9test%20text - UTF8; brightness 9 – max

3.4. Escape sequences that are specific to this display.

These sequences are used to change text properties. The following sequences are specific to this display. They can be used for Advanced Script and for Remote Text Entry.
Attention! When used in a script, it is necessary to enter the "\" character twice, e.g. "**\\g**"

\xNNN	Setting the X-coordinate of the text.
After the 'x' character (instead of NNN), three numbers must always follow. The value is limited by the display resolution. You can use a capital X that will double the specified coordinates in the case of bold text. Example: \x010TEXT -- Set the X-axis cursor position to coordinate 10 \x-010TEXT -- Set the X-axis cursor position to coordinate -10	
\r	Sets the text color to red.
The following text will be in red color.	
\g	Sets the text color to green.
The following text will be in green color.	
\a	Sets the text color to yellow.
The following text will be in yellow color.	
\0 \1 ...	Text positioning: range 0 to 7, depending on the panel.
The following text will be on the specified line, but the X-coordinate will remain unchanged.	
\i	Text size: normal text.
The following text will fit within one line, but the X-coordinate will remain unchanged.	
\b	Text size: tall text.
The following text will fit within two lines, but the X-coordinate will remain unchanged.	
\h	Text size: very tall text.
The following text will fit within three lines, but the X-coordinate will remain unchanged.	
\m	Text size: maxi tall text.
The following text will fit within four lines, but the X-coordinate will remain unchanged.	
\d	Text style: bold text.
The following text will be in bold font.	
\n	Text style: normal text.
The following text will be in normal font.	

3.5. Emergency mode

The device can be switched to emergency mode using the button inside on Linkit module. The button labeled "W_1F_1" needs to be held for at least 2 seconds. This will activate the emergency mode. Emergency mode temporarily sets the network to "IP 192.168.1.206 mask 255.255.255.0," and no login will be required in the web interface. This mode remains active until the device is restarted. Restart can be performed through the web interface.

4. Examples of LUA scripts

The display is equipped with sample scripts for text movement. The scripts are written in a way that allows easy modification.

4.1. Scrolling text across the entire panel

- To change the text, you need to modify the values of "text" and "sc0".

```
-- advanced script example:
function position(num)
        local ret
        if num < 0 then
                num = -num
                ret = "\\x-"
        else
                ret = "\\x"
        end
        if num < 10 then
                return ret .. "00" ..num
        elseif num < 100 then
                return ret .. "0" ..num
        elseif num > 999 then
                num = 999
        end
        return ret .. num
end
-- this function is called regularly and must return text for display
function update_text(counter)
        scp = scp – 2 -- here you can add speed, 2 = number of points
        if scp < sc0 then
                scp = sc1
        end
        return position(scp) .. text
end
-- this is the end where the text will stop (differs based on text length and effect)
sc0 = -256
-- this is the starting point from where the text will arrive (panel resolution-dependent)
sc1 = 96
-- this is the text that will move
text = "\\b\\aScroling text attempt etc etc The Sting etc etc ..."
-- starting position
scp = sc1
-- animation speed
setPeriod(1)
```

4.2. One static and one moving text.

- To change the moving text, you need to modify the values of "text" and "sc0".
- To change the static text, you need to modify the value of "txt0".

```
-- advanced script example:
function position(num)
        local ret
```

```
        if num < 0 then
                num = -num
                ret = "\\x-"
        else
                ret = "\\x"
        end
        if num < 10 then
                return ret .. "00" ..num
        elseif num < 100 then
                return ret .. "0" ..num
        elseif num > 999 then
                num = 999
        end
        return ret .. num
end
-- this function is called regularly and must return the text for display
function update_text(counter)
        scp = scp - 2 -- here you can add speed, 2 = number of points
        if scp < sc0 then
                scp = sc1
        end
        return txt0 .. "\\l" .. position(scp) .. text
end
-- this is the end of the text (it varies depending on the length of the text and the effect)
sc0 = -128
-- this is the beginning of where the text will come from (panel resolution)
sc1 = 96
-- this is the text that will not move
txt0 = "\\r\\x017Static text"
-- this is the text that will move
text = "\\gScrolling text on second line ..."
-- starting position
scp = sc1
-- speed animation
setPeriod(1)
```

4.3. Complex animations, two texts transitioning into one.

- The value "scl" must be exactly the length of the text in pixels.
- The value "text" is the text to be displayed.
- The value "wait0" is the duration of displaying two texts after collision.
- The value "wait1" is the duration of displaying a single large text.
  - After this duration, the animation will repeat.

```
-- advanced script example:
function position(num)
        local ret
        if num < 0 then
                num = -num
                ret = "\\x-"
        else
                ret = "\\x"
        end
```

```
        if num < 10 then
                return ret .. "00" ..num
        elseif num < 100 then
                return ret .. "0" ..num
        elseif num > 999 then
                num = 999
        end
        return ret .. num
end

function update_text_scroll(counter)
        scp = scp + 4 -- here you can add speed, 4 = number of points
        if scp >= sct then
                scp = sct
                tw = tw + 1
                if tw >= wait0 then
                        tw = 0
                        update_text = update_text_big
                end
        end
        return "\\g" .. position(scp-scl) .. text .. "\\l\\r" .. position(96-
scp) .. text
end

function update_text_big(counter)
        tw = tw + 1
        if tw >= wait1 then
                tw = 0
                scp = -16 -- here you can change the shift = time without display.
                update_text = update_text_scroll
        end
        return "\\a\\b" .. position(96-scp) .. text
end
-- length of the text
scl = 26
-- end of the animation (centered according to the text)
sct = 48 + (scl/2)
-- text to display
text = "testing"
-- waiting time after collision
wait0 = 2
-- waiting time after enlargement
wait1 = 30
-- process variables
scp = -10
tw = 0
-- animation speed
setPeriod(1)
-- this function is called regularly and must return the text to be displayed
update_text = update_text_scroll
```

4.4. Modbus – percentage of production

- The example shows the number of requested and the number of produced pieces on the first line, and the production status in percentage and the current date alternate on the second line
- Modbus register No. 0 – number of manufactured pieces
- Modbus register No. 1 – number of required pieces
- the example includes an image switching option

4.4.1. Simple script

```
DispAdd("Req.")
DispAddReg(1)
DispAdd("pcs")
DispSetX(64)
DispAdd("Man.")
DispAddReg(0)
DispAdd("pcs")

DispNextLine()
-- display different text every five seconds
if (counter % 10) < 5 then
   DispGreen()
   DispDate()
else
   if getRegister(1) == 0 then
         -- the required number of pieces is not set, cannot convert to
         percentages
         DispAdd("= N/A")
   else
         DispAdd("= ")
         -- conversion to percentages, limited to one decimal place
         DispAdd(getDec(scaleValue(getRegister(0), 0, getRegister(1),
0, 100), 1))
         DispAdd("%")
   end
   DispGreen()
end

DispSetX(96)
DispTime()
```

4.4.2. Advanced script

```
function timeToText()
   local h, m = getTime()
   if m < 10 then
         m = "0" .. m
   end
   if h < 10 then
         h = "0" .. h
   end
   return h .. ":" .. m
end
```

```
function dateToText()
   local d, m, y = getDate()
   return d .. "." .. m .. "." .. y
end

function update_text(counter)
   -- modbus register No. 1 – number of required pieces
   line0 = "Req. " .. getRegister(1) .. " pcs"

   -- modbus register No. 0 – number of manufactured pieces
   line1 = "\\x064Man. " .. getRegister(0) .. " pcs"

   -- display different text every five seconds
   if (counter % 10) < 5 then
         extra = "\\g" .. dateToText()
   else
         if getRegister(1) == 0 then
           -- the required number of pieces is not set, cannot convert
               to percentages
               extra = "= N/A"
         else
           -- conversion to percentages, limited to one decimal place
               perc = getDec(scaleValue(getRegister(0), 0,
getRegister(1), 0, 100), 1)
               extra = "= " .. perc .. "%"
         end
         extra = extra .. "\\g"
   end

   time = "\\x096" .. timeToText()

   return line0 .. line1 .. "\\x000\\1" .. extra .. time
end

-- animation speed
setPeriod(10)
```

Scripts can be copied as text within the web interface. When editing the source script, select everything with CTRL+A, copy it to the clipboard with CTRL+C, and then paste it into another location, for example, a newly created one (be cautious about the "advanced" option), using CTRL+A and then CTRL+V.

5. Contact Information

For any issues, feedback, or possible commendations, please contact the address provided below. Thank you.

EGMedical, s.r.o.

Filipinskeho 1534/55
615 00 Brno
Czech republic/ EU

tel.: +420 537 014 211
email: vyvojari@egmenergo.cz
web: www.egmenergo.cz