eZi-View-APP-W and eZi-View-APP-L

An eZi way of displaying control data from IoT edge devices, BMS or industrial controllers with Modbus RTU or IP client communications on a device running Windows device.

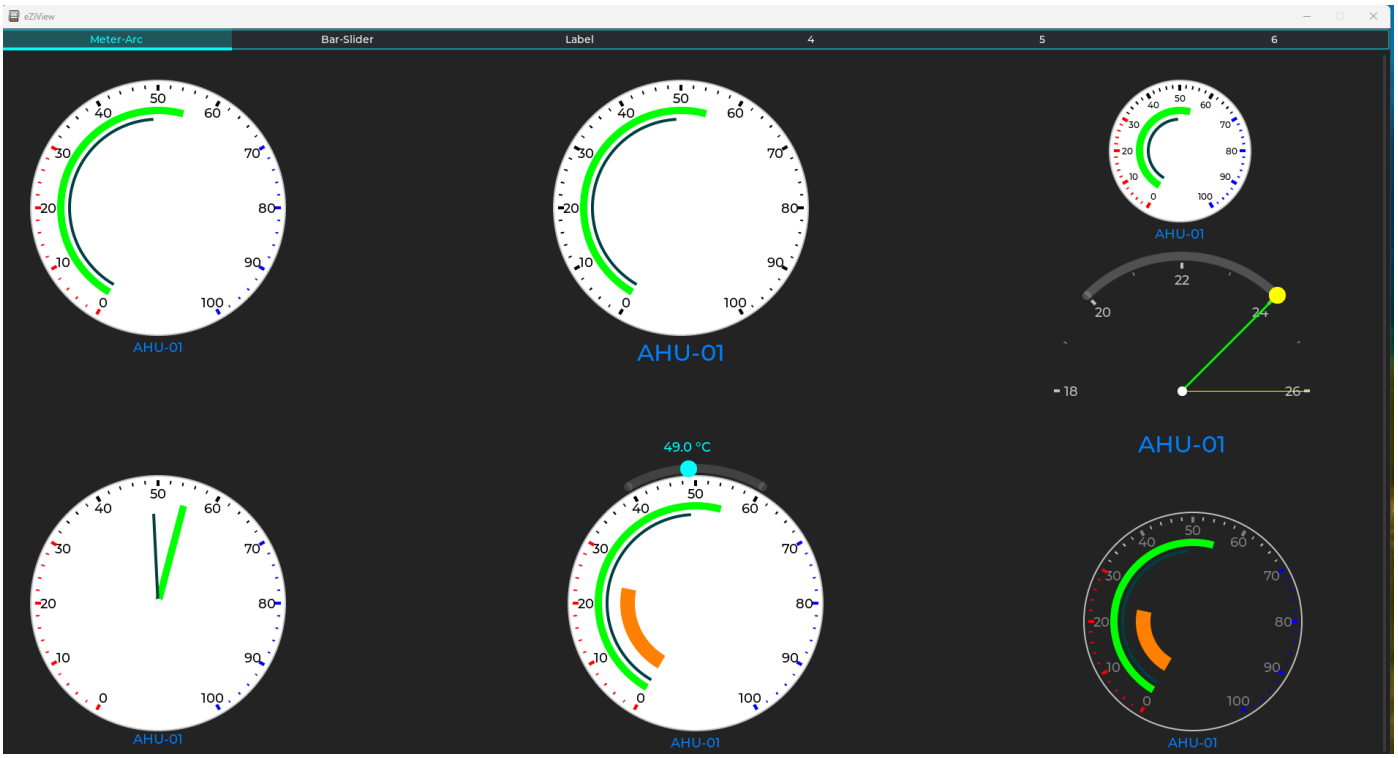
eZi-View-APP belongs to the eZi-View family with the following part numbers

1. eZi-View-APP-W Windows app.
2. eZi-View-APP-W Linux app. Not yet released

The app has 1 x Modbus IP server connection and 1 x Modbus RTU server connection to communicate between a client device and the app's Modbus Server Table. The Modbus Table is then linked to user designed widgets configured to visualize data in an intuitive way. Both display and command of registers are possible.

The app is licensed to the ID of the machine it is intended to run on. The license is hardware dependent and not transferable. Licenses are grouped into 2 categories – Touch and Display. The touch licenses provide user control of widgets with the option of password protection. Each category has license options that determine the number of widgets and tab screens allowed.

Only 1 instance of the app will run on a single machine.



Document Revision Control

1.0	Pre-Release	IP	21 Jan 2025
1.1	Update for ezicontrol.com	IP	25 Jan 2025
1.2	Updated endianness and byte swap	IP	7 March 2025
1.3	Updated for ezicontrol.com	IP	12 March 2025

App revision Control

1.0	Pre-Release	IP	21 Jan 2025
1.1	Update for ezicontrol.com. Compatible with edgeUP Rev1-2-0 and above	IP	21 March 2025

Index

TOC Heading

Overview

All references to app's and documentation are available on the URL shown in the QR code. (Knowledge - eZiControl)
Visit our YouTube channel for tutorials that explain from beginner to expert.

Navigate to Visualization and Display.

A new approach to graphical user interface (GUI) design that supports Modbus RTU (BACnet to be added Q2 2025) to visually display and set the values in a register table. Simply use text to design visually impactful interfaces. Drag and drop online design tool will be available Q2 2025

The GUI contains visualization objects referred to as "widgets". Each widget has configuration options that are referred to as the widget's "personality". A widget is linked to one or more registers in the register table to show real-time value changes. Personality settings are saved in a text file and loaded using the free desktop app edgeUP.

The display supports up to 16 tabbed pages facilitating grouping of widgets to a specific task. Example Tab1 used for Air Handling Unit 1, Tab2 used for Air Handling Unit 2 and Tab3 for Generator.

Widgets can be either D for Display or T for Touch. Display widgets do not allow user modification of register values (read only). Touch widgets display and can be used to change the register values (read write).

Many useful features are built in – such as page flip with user control via communication (Modbus). Password protection can be enabled on Touch widgets.

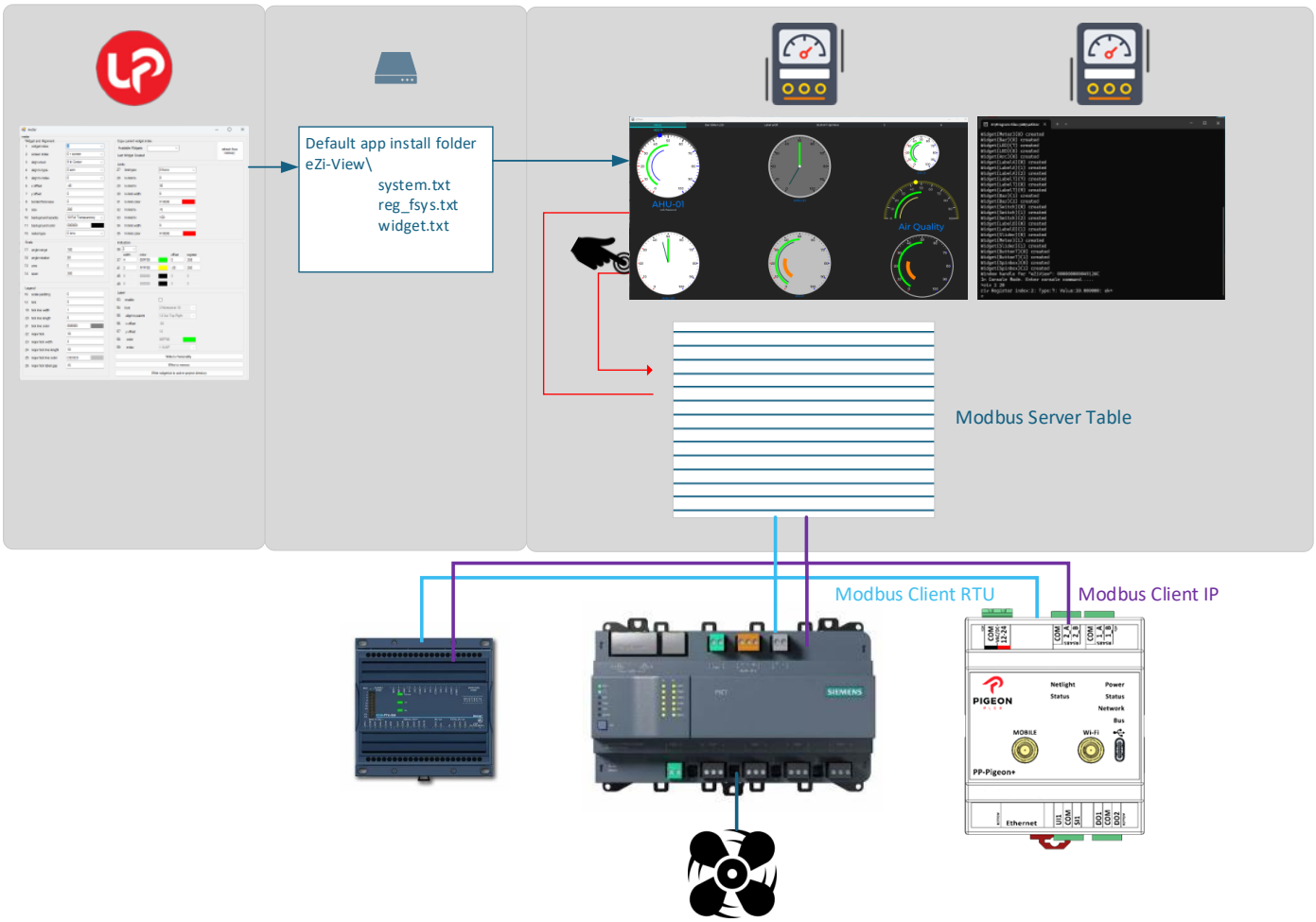
All communication channels are interfaced through a common register table. This approach allows the flexibility in communications and execution by changing registers – a simple method with immense power and configurability.

A flexible once off licensing model allows "pay for features" to keep the price in line with the field application in other words you only license the features required.

Install

Download the app from Knowledge - eZiControl. Unzip and run setup.

System Architecture



Licensing

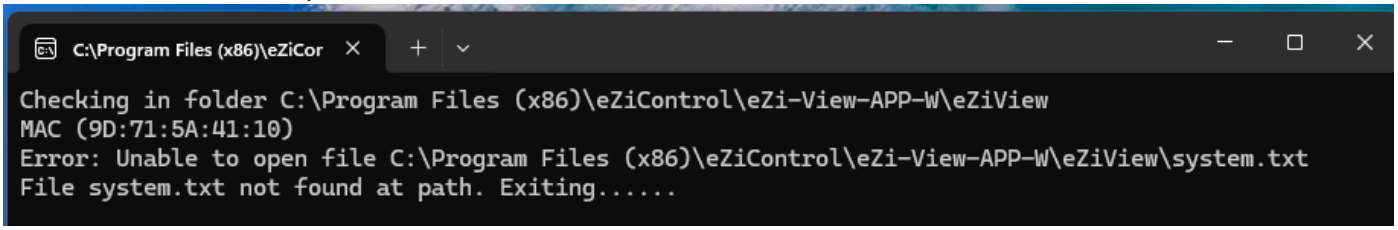
Once off licensing is used to match application complexity to price. Why pay for a device that can display hundreds of values when only a few widgets are required. The same goes if only display is required, why pay for control and password protection. The license plan is a once off purchase and can be upgraded to add additional widgets/screens and touch control at a later stage. Note that an unlicensed device will not display any widgets – it will remain on a splash screen showing the devices MAC address.

Once off licensing is used to match application complexity to price. Why pay for a device that can display hundreds of values when only a few widgets are required. The same goes if only display is required, why pay for control and password protection. The license plan is a once off purchase and can be upgraded to add additional widgets/screens and touch control at a later stage.

The license is ordered online using the license menu. The device MAC and features are required.

The computer identification MAC is used to generate a unique license. To get the MAC, install eZi-View-APP-W and run the application (refer image below). Note the MAC address in the command window. Example MAC (9D:71:5A:41:10). On receipt of the license code, paste this into system.txt file on a new line directly after the file name line. Refer to tutorials.

If the system.txt file is not found or license does not match the MAC address, the device will halt at the command screen with an error message on the startup splash screen. Please check the license code in system.txt and ensure the correct license for the MAC address is in line 2 of system.txt.



If there is no license file or the license is incorrect for the MAC, the application will enter Demo mode. This will provide full functionality on Modbus IP and RTU as well as display widgets and allow touch control based on a T3 license. After 15 minutes the application will close.

Maximum Personality Model Widgets

The license determines the personality model. Personality models are grouped into T for touch and D for display only. The touch models have widgets that respond to touch and allow user input, the display only models do not have these widgets. The maximum screens and widgets per model is shown in the table below. Each model has a maximum number of widgets and screens/tabs as defined below.

If the widget index is above the maximum allowed for a model, then the widget is ignored.

The License flag (refer to system volatile register table 403) indicates the state of registration.

Item	T1	T2	T3	T4	D1	D2	D3	D3
Screen	1	3	6	16	1	3	6	16
Label A	4	12	24	100	4	12	24	64
Label T	6	18	36	100	6	18	36	100
Label D	4	12	24	100	4	12	24	100
Meter	2	6	12	100	2	6	12	100
Bar	4	12	24	100	4	12	24	100
LED	4	12	24	100	4	12	24	100
Slider	2	6	12	100	0	0	0	0
Switch	2	6	12	100	0	0	0	0
ButtonT	4	12	24	100	0	0	0	0
Spinbox	1	3	6	100	0	0	0	0
Arc	1	3	6	100	0	0	0	0
Total	35	105	210	1116	25	75	150	580

Register Table

A common register table is used to provide a flexible modular product that is easily adapted to each use case. Figure1 shows the register table positioned as the central repository for data – all operations read from this table and write the result to the table.

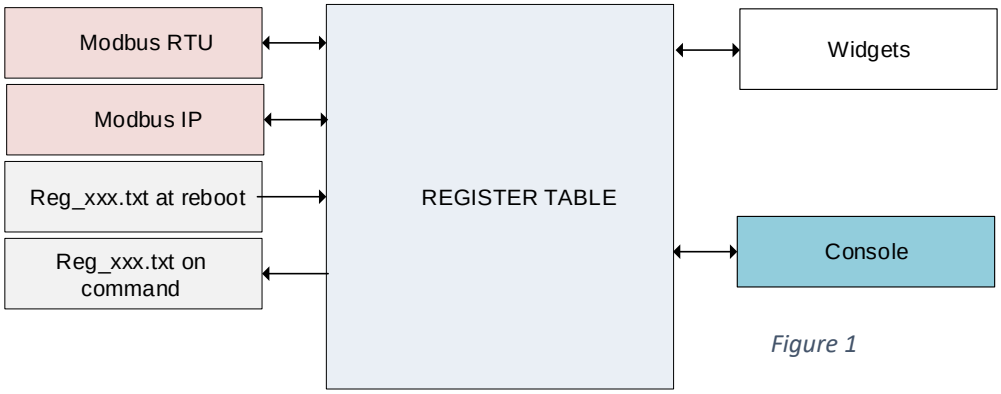


Figure 1

Register Table - Logical areas.

The register table is divided into *user* and *system* logical areas. Each area is further divided by memory type into volatile and nonvolatile with non-volatile divided into flash config and flash variables. This is in line with eZi-COM-DSP to have a common register table across all eZi-View family of products.

Each register is 16 bits and can be assigned a data type to identify how it is represented. Each register therefore has 2 tables, the *register table* and the *register type table*.

Register Section	Note	Address from	Address to
Volatile User	Use for variables display variables such as IO	0	199
Flash User Config	Use for configuration variables such as baud rates and addresses. These values are read from Flash on restart and NOT written to flash on change. This allows for default values. If the value is to be saved on write, then use the Variables section.	200	349
Flash User Variables	Use for variables such as setpoints. These values are read from Flash on restart and written to flash on change. Note write not implemented on rev1-0-0	250	349
Volatile system	Predefined use. Refer to User Selected System Values.	350	449
Flash System Config	Predefined use. Refer to User Selected System Values. Flash Config	450	459
Flash System User	Predefined use. Refer to User Selected System Values. Flash User	550	697
Special Registers	Reboot – Write 0x55AA. Not written to NV	698	
	Factory Reset – Write 0x1928. Not written to NV	699	

User Volatile - Vuser

User			
Index	**Description**	**Default Type**	**Reference**
0-118	User defined register	float	
120-199	User defined register	uint16	

User Flash Config- Fuser

User			
Index	**Description**	**Default Type**	**Reference**
200-228	User defined register	float	
230-249	User defined register	uint16	

User Flash Variable - Euser

User			
Index	**Description**	**Default Type**	**Reference**
250-318	User defined register	float	
320-349	User defined register	uint16	

System Volatile - Vsys

System			
Index	**Description**	**Default Type**	**Reference**
350	counter 0 to 100 (increments 5/500mSec)	float	
352	counter 18 to 26 (increments 0.2/500mSec)	float	
354…	Spare float		
394	Spare Float		
395	counter 0 to 5 (increments 1 per 1 Sec)	uint16	
396	counter 0 to 10 (increments 1 per 1 Sec	uint16	
397	counter 0 to 100 (increments 5/500mSec)	uint16	
398	counter -50 to + 50 (increments 5/500mSec	int16	
399	1 Second Pulse Train	uint8_t	**1 Second Pulse Train**
403	License	unit16	
404	Spare	unit16	
405	Spare	unit16	
406	Spare	unit16	
407	Spare	unit16	
408	spare	uint16	
409	Spare	unit16	
410	Page Flip	uint16	**Page Flip**
..449	Spare		

System Flash Config - Fsys

System				
Index	**Description**	**Default Type**	**Reference**	**Default**
450	Modbus RTU Baud Rate.	uint16		9600
451	Modbus RTU Server Address	uint16		1
452	Modbus RTU Turnaround Time.	uint16		1000
453	Modbus RTU COM port	uint16		1
454…		uint16		
460	Modbus IP Port Server Address	uint16		1
461	Modbus IP Port	uint16		1502
462…				

System Flash Variable - Esys

System				
Index	**Description**	**Default Type**	**Reference**	**Default**
550	Window top left x	uint16		10
551	Window top left y	uint16		10
555	Password	uint16	**Password**	1352
556	Password Timer in seconds	uint16		60
…696				

Data Type

The register table is made up of 16-bit registers. The following data types are supported by using 16 bit registers as a base.

Data Type	Data Type Enum	Number of Registers	Note
uint8	1	1	0 to 255
uint16	2	1	0 to 65 535
uint32	3	2	0 to 4 294 967 295
int8	4	1	-127 to +127
int16	5	1	-32 767 to +32 767
int32	6	2	-2 147 483 647 to + 2 147 483 647
float	7	2	-3.4E+38 to +3.4E+38. Little Endian byte swap
char	8	Depends on char application	String. Char length is dependent on application

Register allocation and data types are set up as default – no user intervention is required. For advanced users, remember to allocate the correct number of registers when creating the Index. If a data type uses 2 registers, then allocate register x for the data type and register x+2 for the next.

When writing to registers where the value exceeds the max value of a data type, the results will be as follows.

Example write value 300 to an uint8. The result will be 44. This is explained by looking at the bytes

300 decimal = 0x12C. If we look at the size of an uint8 it is 1 byte. take the 0x2C and convert to decimal. The result is 44. This means that type casting shows the bytes available in the registers at the correct byte position and does not do range checking.

The data is stored as little endian – example for 32bit float MSWord is 0x41C8 LSWord 0000. Note that Modbus packets are all converted to Little Endian Byte Swop.

Page Flip

Register 410
Write the tab number to register 410 to flip the display. This only works on Tabbed displays. Only valid tab numbers will be acted on. If the value is below 1 then tab1 will display. Any value above the maximum tabs will always display the last page.

Password

Register 555 and 556
Register 555 holds a 4-digit password. and register 556 the password timer in seconds. If no screen activity occurs for the period specified by the display timer, the password is killed, and the code will have to be re-entered on the next widget press. The minimum password timer allowed is 30 seconds.
Default Password 1352.
If a mistake is made entering password, then press OK button. A password error message will display. Press OK again to clear the password keypad. Press the widget again to show the password box.

A small eye symbol displays in the bottom left corner when password mode is active.

1 Second Pulse Train

Register 399 is a 1 second pulse train. 1 Second ON and 1 Second OFF. This can be used to create flashers.

Special System Registers

Simulator

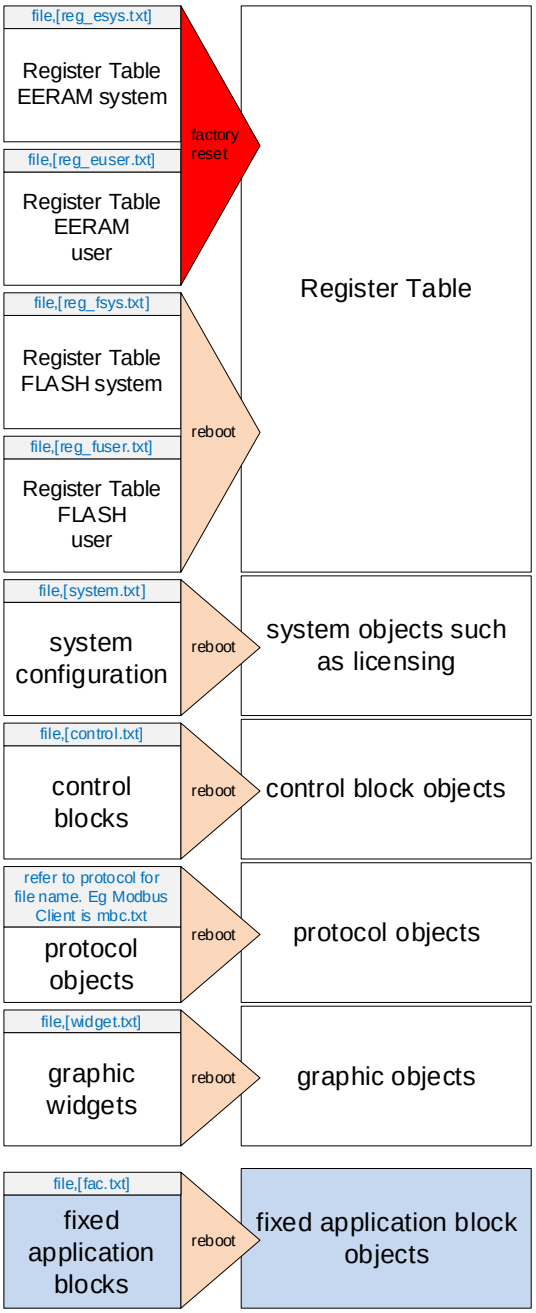
Registers 350-399 as defined in **System Volatile - Vsys** are free running counters that can be used to animate widgets for demonstration.

Personality Text files

The personality – or how the device has been configured is determined by text files with very specific names and functions as defined below. The text files are loaded into the installed directory eZiView folder.

Text files contain ASCII JSON strings that embed configuration information. There are multiple files each with a specific file name that identifies the use case.

All files are shown in Figure 2. The files required for correct minimum operation are detailed in Data is represented in the register table in little endian.
Modbus data on Server Port1 is little endian byte swap – data format 6.



JSON Strings.
Summary of files required.

1. system.txt
    a. License
    b. tabs titles
    c. screen configuration
2. widget.txt
    a. text,[]
    b. label,[]
    c. engu,[]
    d. state[]
    e. widget definition – 1 widget per line
3. reg_fsys.txt
    a. Port1 communication parameters
4. reg_esys.txt
    a. Log verbosity
    b. Backlight settings
    c. Password settings

Some of the files are license dependent. As an example, if the device is not licensed for control blocks, the control.txt file will be ignored at reboot. The same goes for bacnet.txt if the device is not registered for BACnet.

Files reg_euser.txt and reg_esys.txt are only read by special register command. Refer

reg_esys.txt is loaded at reboot.

If reg_fsys.txt is not found at reboot, the system creates the file and writes the defaults as per the register table. Both reg_fsys.txt and reg_fuser.txt are loaded at reboot.

widget.txt is loaded at reboot if licensed.

Files not mentioned above are related to the eZi-COM-IO and eZi-COM-FCU families.

Figure 2

File location

Files are located in the default install directory eZiView sub directory.
Copy the project files and save in the folder before opening eZi-View-APP-W.
Use edgeUP app to create the widgets. To interact with the installed directory, set the Active Project to the default install directory eZiView sub directory.

The default install directory is displayed when app is first opened.

File maximum and minimum parameters

These are the limit definitions defined in firmware

1. Maximum number of lines is 1200.
2. Maximum file name length including .txt extension 20
3. Maximum file line length 300
4. Maximum number of digits in a number 12

Text Strings

Text strings are defined in the following categories – each category is assigned to its own storage location and is referenced by index

1. **Tab**. Maximum25 characters. Used to name the screen tab. Index 0-19

2. **Text**. Maximum 40 characters. Used to store general text. Index 0-99
3. **Label**. Maximum 20 characters. Used to store labels. Index 0-99
4. **Engineering Units(Engu).** 8 characters. Used to store engineering units. Prepopulated with common units. Index 0-99
5. **State**. 12 characters. Used to store digital or multistate values. Prepopulated with common units. Index 0-99

When reference is made to these text strings in system.txt and widget.txt, the category is implied by the widget parameter – refer to the notes of each widget. The index points to the text in that category. This method removes long text strings from the widget and makes the widget definition more compact. The text string model allows for language independence. Simply change the text in widget.txt to the required language and download. The widget will display the new referenced text.

Console

A text console for direct commands is available on the USB port. Command echo is on meaning the character typed is echoed back. If a command is not completed. The console will time out after 5 seconds. To check if the console is connected, press enter.
Get help on the commands by entering ?

Console commands.

Command	Description
?	Help - prints out the list of commands.
riv	register index value. Used to set the register value. riv index value. Example riv 20 12.5 will set register 20 to 12.5. Note that the sign is ignored if the register type is uint8, uint16, uint32
rir	register index read. rir 20 will reply *rir Register index:20: Type:7: Value:12.50000: ok>*
ref	Refresh screen after changing widget.txt personality file

Modbus RTU Server

Communication parameters are stored in flash system register table (Refer **System Flash**). Default values are COM1, 9600 baud with address 1 for the server. The data bits are fixed at 8 and stop bit 1.
COM port options are 1 to 255. (Use system manager to determine the USB to RS485 converter COM port).
Baud Rate options are 1200 to 57600. No range checking is done between these 2 values. Ensure full baud number is entered Do not use 96 for 9600 as an example.

The Server address range must be between 1 and 255. Values outside this range will not be accepted.

The turnaround time is in uSeconds and is the time from receipt of the last byte in a packet to the transmit of reply.

Modbus commands are:

- 0x03 Read Holding registers (Max 16 registers or 32 bytes).
- 0x06 Write single register.
- 0x10 Write multiple registers (Max 16 registers or 32 bytes)

To disable the RTU service set COM port and Server Address to 0. Register 450 and 451 in reg_fsys.txt

Modbus IP Server

Parameters are stored in non-volatile flash system register table (Refer **System Flash**). Default values are Port 1502 with address 1 for the server. The Server address range must be between 1 and 255. Values outside this range will not be accepted.

Modbus commands are:

- 0x03 Read Holding registers (Max 16 registers or 32 bytes).
- 0x06 Write single register.
- 0x10 Write multiple registers (Max 16 registers or 32 bytes)

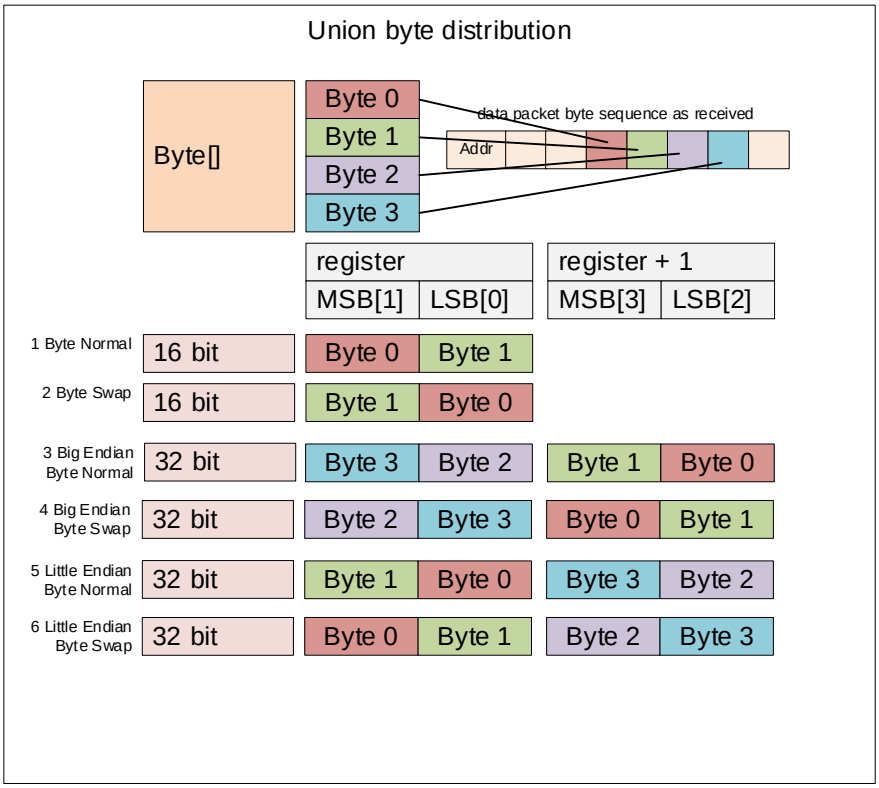
To disable the IP service set IP port and Server Address to 0. Register 460 and 461 in reg_fsys.txt

Endianness and byte swap

Endianness refers to the order in which bytes (8 bits) are arranged in a multi-byte data value. There are two types of endianness: big-endian and little-endian.

In big-endian (also known as network byte order), the most significant word is stored first in memory (register index), followed by the less significant word (register Index+1). This means that the MS word has the lowest memory address, while the LS word has the highest memory address. In Little Endian the order is reversed.

Byte swap refers to the order of the bytes when transmitted in a word. For a 16-bit word made of 2 x bytes, the normal order is Byte 0 occupies the lower position and Byte 1 the higher. Example the value of a 16-bit word is 0x1234. If the MS Byte is transmitted first 0x12 followed by 0x34 this is the normal order. In byte swap the bytes are transmitted 0x34 then 0x12.



As an example, the float value of 25.0 is 0x41C8 0x0000 in big endian and 0x0000 0x41C8 in little endian when stored in the register table will have 16 bit register address [x] the value 0x0000 and register [x+1] 41C8. This is little endian.

When this is transmitted as little endian the byte order starting from byte 0 is 00 00 C8 41. When transmitting as little endian byte swap the byte order starting from byte 0 is 00 00 41 C8

An online converter [Floating Point to Hex Converter (gregstoll.com)]

Modbus Data Formats

data format value	data format type
1	16 bit
2	16 bit byte swap
3	32 bit big endian
4	32 bit big endian byte swap
5	32 bit little endian
6	32 bit little endian byte swap

Data is represented in the register table in little endian.

Modbus data on Server Port1 is little endian byte swap – data format 6.

JSON Strings

Each line of the text file follows a common format.
<param0>,[<param1>,<param2>,....<paramX>]
line is terminated with CR or LF or CRLF. No other control characters allowed.

<param0> defines the use, from <param1>..<paramX> define the use case.

Only JSON strings relevant to the file will be processed. As an example, a cb function in the reg_esys.txt will not be processed. An empty line indicates the end of a file. Ensure no blank lines are in any of the files.

<param0> Each file is listed with the allowable param0 types.

1. **system.txt**
   - file,[] (must be on line 1)
   - lic,[] (must be on line 2)
   - tab,[]
   - scrn,[]

2. **reg_esys.txt, reg_euser.txt**
   - file,[] (must be on line 1)
   - reg,[]
   - riv,[]

3. **reg_fsys.txt, reg_fuser.txt**
   - file,[] (must be on line 1)
   - reg[]
   - riv,[]

4. **widget.txt**
   - file,[] (must be on line 1)
   - text,[]
   - label,[]
   - engu,[]
   - state[]
   - label_a,[]
   - label_d,[]
   - label_t,[]
   - meter_l,[]
   - bar,[]
   - led,[]
   - slider,[]
   - button_t,[]
   - Spinbox,[]
   - arc,[]
   - switch,[]

<file>

file,[filename.txt]

The first line in system.txt identifies the file name to provide a check that the correct file is processed. No white spaces are allowed in the file name and will be removed. Example file,[file name] is not allowed as there is a white space and the extension .txt is left out.
Example for system.txt file,[system.txt]

<lic>

lic,[license number]

The second line in system.txt is the supplier issued license number. This license number identifies the additional features licensed. If this line is excluded then the device will operate in default mode with no additional features such as control block capability.
Example lic,[xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx] where xxxx is the license supplied for the MAC address

<tab>

Index 0 to 16
Defines a Screen Tab name to be referenced by respective screen tab. Tab0 refers to the screen name of a non-tabbed screen. The maximum length of a tab string is 25 characters. The tab string must be preceded by the index number. Only 1 tab index per line is allowed. Refer to **Licensing** for number of tabs.
Keep the tab names short to ensure all the tabs configured are fully displayed on the screen size set. No special characters or comma's are allowed as these are used as JSON indexes when decoding the string.

<scrn>

scrn,[Refer below for parameter interpretation]
Note – the window position co-ordinates are stored in esys.txt register 550 and 551 and not in the <scrn>. This allows the window position to be by the user by changing reg_esys.txt.

The screen definition has numbered parameters. These must be entered in the sequence described below.

Key	Function	Value	Descriptor
1	horizontal res	uint16	Horizontal pixels. Max 2000
2	vertical res	uint16	Vertical pixels. Max 2000
3	orientation	uint8	Not used – leave as 1
4	theme dark primary color	uint32	Dark Mode color for system items (example Tab text). RRGGBB color in HEX. Refer **Colors**
5	theme light primary color	uint32	Light Mode color for system items (example Tab text). RRGGBB color in HEX. Refer **Colors**
6	dark background color	uint32	RRGGBB color in HEX. Refer **Colors**
7	light background color	uint32	RRGGBB color in HEX. Refer **Colors**
8	dark mode	uint8_t	0 for light theme >0 for dark theme. Refer Error! Reference source not found.
9	border dark mode color	uint32	border color in dark mode. RRGGBB color in HEX. Refer **Colors**
10	border light mode color	uint32	border color in light mode. RRGGBB color in HEX. Refer **Colors**
11	widget label custom color enable	uint8_t	0 all labels use dark/light colors below >0 to allow label color selection for each label.
12	widget label dark mode color	uint32	widget label color in dark mode. RRGGBB color in HEX. Refer **Colors**
13	widget label light mode color	uint32	widget label color in light mode. RRGGBB color in HEX. Refer **Colors**
14	text custom color enable	uint8_t	>0 to allow label color selection for each text widget
15	text dark mode color	uint32	text color in dark mode. RRGGBB color in HEX. Refer **Colors**
16	text light mode color	uint32	text color in light mode. RRGGBB color in HEX. Refer **Colors**
17	analog value custom color enable	uint8_t	>0 to allow analog value color selection for analog value widgets
18	analog value dark mode color	uint32	analog value text color in dark mode. RRGGBB color in HEX. Refer **Colors**
19	analog value light mode color	uint32	analog value text color in light mode. RRGGBB color in HEX. Refer **Colors**
20	screen type	uint8	Refer **Screen Type**
21	screen scroll	uint8	0 to disable screen scroll and scroll bars. 1 to enable
22	tab location	uint8	Refer **Screen Tab Location**
23	tab height	uint8	In pixels. Default 30. Max 100;
24	tabs	uint8	Limited to maximum licensed screens allowed. Tab title defined in system.txt **<tab>**

<scrn> dark mode and dark/light colors

Two color themes or modes are available – Dark and Light. Each theme has a color palette that defines the colors for each mode. Example border color dark mode and border color light mode. The 2 color modes provide a quick method of changing color options. In dark mode the background of a tab button is black and in light mode it is white.

<scrn> global settings for password

To enable password on touch widgets, the global password must be enabled in **Screen Type**. If the widget password is enabled and the global password is enabled then a password is required to change the register value using the widget. These 2 settings provide an override function where passwords can be globally enabled/disabled or locally enabled or disabled. An example of this is allowing a user to adjust setpoint without a password but preventing access to the PID loop settings such as proportional band or Integral time.

<scrn> global settings for label and text

If custom color enable is set (>0) for either widget label, text label or analog value text, then each widget must have the color defined per widget. If enable is clear (value 0) then all widget labels, text labels and analog values will use the light and dark colors as defined per widget or text type. This provides flexibility by allowing a common theme color that will automatically change when the theme is changed. If this option is set (>0) then widget and text color settings will require widget file modification when switching from dark to light mode.

<scrn> status area

A 20-pixel area covering the full width of the screen is used to show status symbols. An example is when password mode is on. If a widget overlaps the status area, the status icon will be on top of the widget.

<reg>

reg,[<register table index>,<register table type>,<register table default value>]

Defines a register in the register table.
<register table index>. Refer to register table.
<register table type>. Refer to Data Type
<register table value>. The value written to the register table. Must be consistent with the Data Type range.

Examples

1. reg,[1,2,50]
    a. param1 – register table index (register 1)
    b. param2 – register number type (uint16)
    c. param3 – value. 50
2. reg,[2,7,50.0]
    a. param1 – register table index (register 3)
    b. param2 – register number type (float)
    c. param3 – value. 50.0
    d. NOTE this register is 32 bits wide. Takes 2 register locations so the next available register is 4

<riv>

riv,[<register table index>, register table default value>]

Defines a register in the register table.
<register table index>. Refer to register table

<register table value>. The value written to the register table.

Examples
1. riv,[1,50]
    a. param1 – register table index (register 1)
    b. param2 – value. 50
2. riv,[2,50.0]
    a. param1 – register table index (register 3)
    b. param2– value. 50.0
    c. NOTE this register is 32 bits wide. Takes 2 register locations so the next available register is 4

<text>

Index 0 to 199
Defines a text string in the text array to be referenced by widgets. The maximum length of a text string is 40 characters. The text string must be preceded by the index number. More than 1 text string is allowed on a line up to the maximum file line length. Refer **File maximum and minimum parameters.**

Examples
1. text,[0,”Hello There”]
2. text,[0,”Hello There”,1,”Test Text”]
3. text,[2,”Next Test Text”]
4. text,[3,No Quotes] will use the comma as the text starting point, ignore all spaces and non-text characters up to the next quote or closing bracket or maximum length of text exceeded.

<label>

Index 0 to 199
Defines a text string in the label array to be referenced by widgets. The maximum length of a label string is 20 characters. The label string must be preceded by the index number. More than 1 label string is allowed on a line up to the maximum file line length. Refer **File maximum and minimum parameters.**

Examples
1. text,[0,”Hello There”]
2. text,[0,”Hello There”,1,”Test Text”]
3. text,[2,”Next Test Text”]
4. text,[3,No Quotes] will use the comma as the text starting point, ignore all spaces and non-text characters up to the next quote or closing bracket or maximum length of text exceeded.

All widgets except for a text label (label_t) can have a name label associated with the widget.. The label is aligned to the parent widget.

<engu>

Index 0 to 199
Defines an engineering unit string in the engu array to be referenced by widgets. The maximum length of an engu string is 8 characters. The engu string must be preceded by the index number. More than 1 engu string is allowed on a line up to the maximum file line length. Refer **File maximum and minimum parameters.**

Examples

1. engu,[0,"Pa"]
2. engu,[0,"Pa",1," °C"]
3. text,[2,"H Pa"] – shows spaces are allowed
4. text,[3,No Quotes] will use the comma as the engu starting point, ignore all spaces and non-text characters up to the next quote or closing bracket or maximum length of engu exceeded.

<state>

Index 0 to 199
Defines a state or multistate string in the state array to be referenced by widgets. The maximum length of a state string is 12 characters. The engu state must be preceded by the index number. More than 1 state string is allowed on a line up to the maximum file line length. Refer **File maximum and minimum parameters.**

Examples

1. state,[0,"Open"]
2. engu,[0,"Open",1," Closed"]
3. text,[2,"Door In"] – shows spaces are allowed
4. text,[3,Door out] will use the comma as the engu starting point, ignore all spaces and non-text characters up to the next quote or closing bracket or maximum length of engu exceeded.

Widgets

Widget Type	Widget Name	Note	Type
0	Screen	Screen 0 for screen type 1. Screen 1 to 6 for type 2	Display
1	Label_A	Label to display analog value	Display
2	Label_D	Label to display up to 5 digital states	Display
3	Label_T	Label to display Text only	Display
4	Meter	Meter with indicator lines or arc's	Display
6	Bar	Bar to display analog value	Display
9	LED	Led to display up to 5 digital states	Display
10	Slider	Slider to display/adjust analog value in integer steps	Touch
11	Switch	Switch to set a register	Touch
12	Button_T	Button for multi-state. Text label shows state	Touch
13	Spinbox	Spinbox to display and set analog value	Touch
14	Arc	Arc to display and set analog value	Touch

Display type

<label_a> Label for display of analog value with engineering unit linked to a register

[widget index 1]

Display an analog value with engineering units. An optional text label can be added.

Key	Function	Type	Descriptor
1	widget index	uint8	My widget Index. **Maximum Personality Model Widgets**
2	screen index	uint8	Screen to place widget on. Refer **Screen Type and Maximum Personality Model Widgets**
3	align enum	uint8	Refer **Aligning Widgets**
4	align to type	uint8	Refer **Aligning Widgets**
5	align to index	uint16	Refer **Aligning Widgets**
6	x	int16	X offset. Refer **Aligning Widgets.** Refer Note1 below
7	y	int16	Y offset. Refer **Aligning Widgets.** Refer Note1 below
8	analog value font	uint8	Font size. Refer **Font**
9	analog value border thickness	uint8	Border thickness in pixels
10	analog value text color	uint32	RRGGBB color in HEX. Refer **Colors**
11	engu index	int16	-1 not used. Engineering Units index to add after analog value. Refer **<engu>**
12	register	uint16	Register index for analog value. Refer **Register Table**
13	label enable	uint8	0 no label display. 1 to add a label. Refer **<label>**
14	label font	uint8	Font. Refer **Font**
15	label align to parent enum	uint8	Widget Type to align to. Refer **Aligning Widgets**
16	label position x offset	int16	X offset. Refer **Aligning Widgets**
17	label position y offset	int16	Y offset. Refer **Aligning Widgets**
18	label color	uint32	Label color. RRGGBB color in HEX. Refer **Colors.**
19	label index	uint8	Label index. Refer **<label>**

Notes

1. Alignment is with respect to the analog value portion. Keep alignment to left side of analog value. The label is excluded from the alignment and must be calculated manually. This allows for alignment of values rather than the lablels.

<label_d> Label for display of text linked to a register

[widget index 2]

Display state text linked to a multi-state or binary register value. Text and and colors are user defined for each state. Example register value 0 and 1 can be linked to 2 states on the button. Or value 0,1 and 2.

For any unused states, fill in 0 as a placeholder for text, text color and back color.

An optional text label can be added.

Key	Function	Type	Descriptor
1	widget index	uint8	My widget Index. **Maximum Personality Model Widgets**
2	screen index	uint8	Screen to place widget on. Refer **Screen Type**
3	align enum	uint8	Refer **Aligning Widgets**
4	align to type	uint8	Refer **Aligning Widgets**
5	align to index	uint16	Refer **Aligning Widgets**
6	x offset	int16	X offset. Refer **Aligning Widgets**
7	y offset	int16	Y offset. Refer **Aligning Widgets**
8	font	uint8	Font size. Refer **Font**
9	border thickness	uint8	Border thickness in pixels
10	size x	uint16	x axis size in pixels
11	size y	uint16	y axis size in pixels
12	register	uint16	Register index for digital or multistate value. Refer **Register Table**
13	states	uint8	Number of states. Minimum 2 Maximum 5. Zero index (0 to 4)
14	state 0 text color	uint32	Text RRGGBB color in HEX. Refer **Colors**
15	state 0 back color	uint32	Background RRGGBB color in HEX. Refer **Colors**
16	state 0 text	uint8	State text index. Refer **<state>**
17	state 1 text color	uint32	Text RRGGBB color in HEX. Refer **Colors**
18	state 1 back color	uint32	Background RRGGBB color in HEX. Refer **Colors**
19	state 1 text	uint8	State text index. Refer **<state>**
20	state 2 text color	uint32	Text RRGGBB color in HEX. Refer **Colors**
21	state 2 back color	uint32	Background RRGGBB color in HEX. Refer **Colors**
22	state 2 text	uint8	State text index. Refer **<state>**
23	state 3 text color	uint32	Text RRGGBB color in HEX. Refer **Colors**
24	state 3 back color	uint32	Background RRGGBB color in HEX. Refer **Colors**
25	state 3 text	uint8	State text index. Refer **<state>**
26	state 4 text color	uint32	Text RRGGBB color in HEX. Refer **Colors**
27	state 4 back color	uint32	Background RRGGBB color in HEX. Refer **Colors**
28	state 4 text	uint8	State text index. Refer **<state>**
29	label enable	uint8	0 no label display. 1 to add a label. Refer **<label>**
30	label font	uint8	Font. Refer **Font**
31	label align to parent enum	uint8	Widget Type to align to. Refer **Aligning Widgets**
32	label position x offset	int16	X offset. Refer **Aligning Widgets**
33	label position y offset	int16	Y offset. Refer **Aligning Widgets**
34	label color	uint32	Label color. RRGGBB color in HEX. Refer **Colors.**
35	label index	uint8	Label index. Refer **<label>**

<label_t> Label to display static text

[widget index 3]

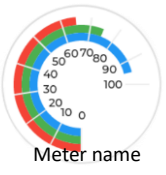
Display text. There is no border on this widget as it is intended for naming screens or groups of widgets..

Key	Function	Type	Descriptor
1	widget index	uint8	My widget Index. **Maximum Personality Model Widgets**
2	screen index	uint8	Screen to place widget on. Refer **Screen Type and Maximum Personality Model Widgets**
3	align enum	uint8	Refer **Aligning Widgets**
4	align to type	uint8	Refer **Aligning Widgets**
5	align to index	uint16	Refer **Aligning Widgets**
6	x offset	int16	X offset. Refer **Aligning Widgets**
7	y offset	int16	Y offset. Refer **Aligning Widgets**
8	font	uint8	Font size. Refer **Font**
9	align text	uint8	Refer **Text** Alignment
10	text color	uint32	RRGGBB color in HEX. Refer **Colors**
11	text index	uint8	Text index. Refer **<text>**

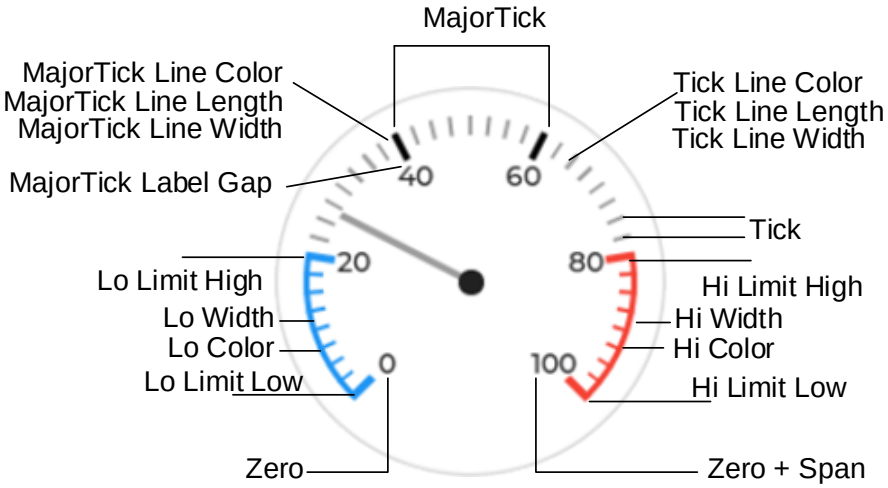
<meter>

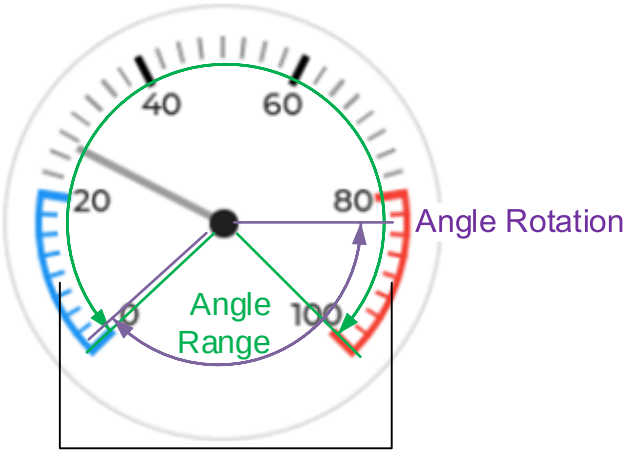
[widget index 4]

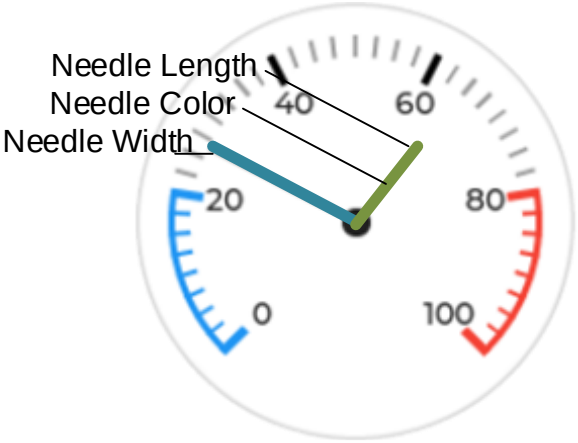
Display integer values on a meter with lines or arcs. Up to 5 indicators can be assigned to registers. The gauge is fully customized and allows for flexibility of display. An optional text label can be added. The guage text font is determined by meter size as follows: size <200 Montserrat12. >=200<300 Montserrat16. >=300 Montserrat 18



Line	Key	Value	Descriptor
1	widget index	uint8	My widget Index. **Maximum Personality Model Widgets**
2	screen index	uint8	Screen to place widget on. Refer **Screen Type**
3	align enum	uint8	Refer **Aligning Widgets**
4	align to type	uint8	Refer **Aligning Widgets**
5	align to index	uint16	Refer **Aligning Widgets**
6	x offset	int16	X offset. Refer **Aligning Widgets**
7	y offset	int16	Y offset. Refer **Aligning Widgets**
8	border thickness	uint8	Border thickness in pixels
9	size	uint16	size in pixels.
10	back opacity	uint16	Background opacity
11	back color	uint32	Background RRGGBB color in HEX. Refer **Colors**
12	Tick	uint16	value between minor ticks
13	Zero	int32	meter zero value
14	Span	int32	meter span. The meter will display from zero to (zero + span). Value must be > 2
15	meter type	uint8	0 for Line meter, >0 for Arc meter
16	scale padding	int16	padding of scale to border
17	angle range	uint16	display angle.
18	angle rotation	uint16	display angle from vertical going anti clockwise.
19	tick line width	uint16	width of minor tick line
20	tick line length	uint16	length of minor tick line
21	tick line color	uint32	Text RRGGBB color in HEX. Refer **Colors**
22	major tick	uint16	major tick every xx minor ticks
23	major tick width	uint16	width of major tick line
24	major tick line length	uint16	length of major tick line
25	major tick line/scale color	uint32	Major Tick and Text RRGGBB color in HEX. Refer **Colors**
26	major tick label gap	uint16	gap between major tick and label text
27	meter limit type	uint8	what limits are to be displayed. Refer to **Meter Limits**
28	lo limit lo	int32	lo limit lo value
29	lo limit hi	int32	lo limit hi value
30	lo limit width	uint8	lo limit line width
31	lo limit color	uint32	Text RRGGBB color in HEX. Refer **Colors**
32	hi limit lo	int32	lo limit lo value
33	hi limit hi	int32	lo limit hi value
34	hi limit width	uint16	lo limit line width
35	hi limit color	uint32	Text RRGGBB color in HEX. Refer **Colors**
36	indicators	uint8	number of indicators. Max 4
37	indicator[0] width	uint16	width of indicator
38	indicator[0] color	uint32	Text RRGGBB color in HEX. Refer **Colors**
39	indicator[0] offset from scale	int16	length of needle (+ value) or arc offset from meter outer (- value)
40	register[0]	uint16	Register index for analog value. Refer **Register Table**
41	indicator[1] width	uint16	width of indicator
42	indicator[1] color	uint32	Text RRGGBB color in HEX. Refer **Colors**
43	indicator[1] offset from scale	int16	length of indicator
44	register[1]	uint16	Register index for analog value. Refer **Register Table**
45	indicator[2] width	uint16	width of indicator
46	indicator[2] color	uint32	Text RRGGBB color in HEX. Refer **Colors**
47	indicator[2] offset from scale	int16	length of indicator
48	register[2]	uint16	Register index for analog value. Refer **Register Table**
49	indicator[3] width	uint16	width of indicator
50	indicator[3] color	uint32	Text RRGGBB color in HEX. Refer **Colors**
51	indicator[3] offset from scale	int16	length of indicator
52	register[3]	uint16	Register index for analog value. Refer **Register Table**
53	label enable	uint8	0 no label display. 1 to add a label. Refer **<label>**
54	label font	uint8	Font. Refer **Font**
55	label align to parent enum	uint8	Widget Type to align to. Refer **Aligning Widgets**
56	label position x offset	int16	X offset. Refer **Aligning Widgets**
57	label position y offset	int16	Y offset. Refer **Aligning Widgets**
58	label color	uint32	Label color. RRGGBB color in HEX. Refer **Colors**
59	label index	uint8	Label index. Refer **<label>**


MajorTick
MajorTick Line Color
MajorTick Line Length
MajorTick Line Width
MajorTick Label Gap
Tick Line Color
Tick Line Length
Tick Line Width
Tick
Lo Limit High
Lo Width
Lo Color
Lo Limit Low
Hi Limit High
Hi Width
Hi Color
Hi Limit Low
Zero
Zero + Span
40
60
20
80
0
100


40
60
20
80
0
100
Angle
Range
Angle Rotation


Needle Length
Needle Color
Needle Width
40
60
20
80
0
100

<bar>

[widget index 6]

A simple display of a value using a bar. A color gradient can be applied for improved visualization. To make the bar vertical size x must be smaller than size y. An optional text label can be added. The X10 option allows this widget to be used for small range views such span ranges – example where the temperature display is limited to between 18 and 26. To improve resolution, the range is increased by 10 to improve resolution of the bar display. No effect is made on the source register.

Line	Key	Value	Descriptor
1	widget index	uint8	My widget Index. **Maximum Personality Model Widgets**
2	screen index	uint8	Screen to place widget on. Refer **Screen Type**
3	align enum	uint8	Refer **Aligning Widgets**
4	align to type	uint8	Refer **Aligning Widgets**
5	align to index	uint16	Refer **Aligning Widgets**
6	x offset	int16	X offset. Refer **Aligning Widgets**
7	y offset	int16	Y offset. Refer **Aligning Widgets**
8	border thickness	uint8	Border thickness in pixels
9	size x	uint16	x axis size in pixels
10	size y	uint16	y axis size in pixels
11	zero	int32	bar zero value
12	span	int32	bar span. The bar will display from zero to (zero + span)
13	color back max	uint32	Background color at max value. RRGGBB color in HEX. Refer **Colors**
14	color back min	uint32	Background color at min value. RRGGBB color in HEX. Refer **Colors**
15	gradient direction	uint8	Direction of gradient if selected. Refer **Gradient Direction**
16	bar color	uint32	bar color. RRGGBB color in HEX. Refer **Colors**
17	analog value enable	uint8	0 no text display. 1 to add a text display with engineering units
18	analog value font	uint8	Font. Refer **Font**
19	analog value align to parent enum	uint8	Widget Type to align to. Refer **Aligning Widgets** and Error! Reference source not found.
20	analog value position x	int16	X offset. Refer **Aligning Widgets**
21	analog value position y	int16	Y offset. Refer **Aligning Widgets**
22	analog value text color	uint32	text color. RRGGBB color in HEX. Refer **Colors**
23	engu index	int16	-1 not used. Engineering Units index to add after analog value. Refer **<engu>**
24	register	uint16	Register index for analog value. Refer **Register Table**
25	label enable	uint8	0 no label display. 1 to add a label. Refer **<label>**
26	label font	uint8	Font. Refer **Font**
27	label align to parent enum	uint8	Widget Type to align to. Refer **Aligning Widgets**
28	label position x offset	int16	X offset. Refer **Aligning Widgets**
29	label position y offset	int16	Y offset. Refer **Aligning Widgets**
30	label color	uint32	Label color. RRGGBB color in HEX. Refer **Colors**
31	label index	uint8	Label index. Refer **<label>**
32	X10	uint8	If set then the scale is multiplied by 10. This is used for small range settings to improve increments

<led>

[widget index 9]
Indication LED that toggles between off and on (register value0 or >0)
An optional text label can be added.

Line	Key	Value	Descriptor
1	widget index	uint8	My widget Index. **Maximum Personality Model Widgets**
2	screen index	uint8	Screen to place widget on. Refer **Screen Type**
3	align enum	uint8	Refer **Aligning Widgets**
4	align to type	uint8	Refer **Aligning Widgets**
5	align to index	uint16	Refer **Aligning Widgets**
6	x offset	int16	X offset. Refer **Aligning Widgets**
7	y offset	int16	Y offset. Refer **Aligning Widgets**
8	border	uint8	Border thickness in pixels
9	size x	uint16	x axis size in pixels
10	size y	uint16	y axis size in pixels
11	border color	uint32	Border Color. RRGGBB color in HEX. Refer **Colors.** Note – border color is fixed at black\
12	on color	uint32	On Color. RRGGBB color in HEX. Refer **Colors**
13	off color	uint32	Off Color. RRGGBB color in HEX. Refer **Colors**
14	on/off	uint8	On/Off. Value options 0 or 1 only. Value of 1 means the LED shade is dimmed and the brightness border is set to 0. The LED looks Off.
15	register	uint16	Register index for analog value. Refer **Register Table**
16	label enable	uint8	0 no label display. 1 to add a label. Refer **<label>**
17	label font	uint8	Font. Refer **Font**
18	label align to parent enum	uint8	Widget Type to align to. Refer **Aligning Widgets**
19	label position x offset	int16	X offset. Refer **Aligning Widgets**
20	label position y offset	int16	Y offset. Refer **Aligning Widgets**
21	label color	uint32	Label color. RRGGBB color in HEX. Refer **Colors**
22	label index	uint8	Label index. Refer **<label>**

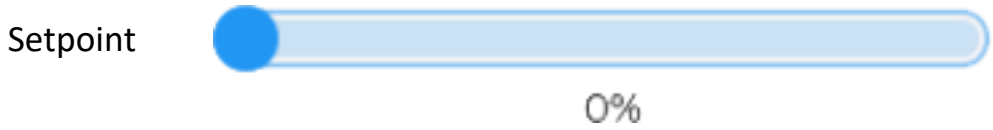
Widgets

Touch type

<slider>

[widget index 10]

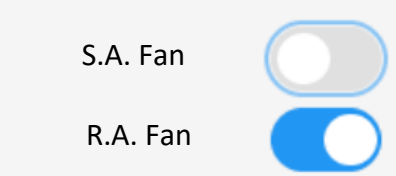
User input by moving a slider to determine the value of a register. The display shows one decimal point. Range is limited to between a maximum and minimum value. Background gradient and color selection as well as slider opacity and knob configuration. An optional text label can be added. The X10 option allows this widget to be used for small range views such span ranges – example where the temperature display is limited to between 18 and 26. To improve resolution, the range is increased by 10 to improve resolution of the bar display. No effect is made on the source register.



Line	Key	Value	Descriptor
1	widget index	uint8	My widget Index. **Maximum Personality Model Widgets**
2	screen index	uint8	Screen to place widget on. Refer **Screen Type**
3	align enum	uint8	Refer **Aligning Widgets**
4	align to type	uint8	Refer **Aligning Widgets**
5	align to index	uint16	Refer **Aligning Widgets**
6	x offset	int16	X offset. Refer **Aligning Widgets**
7	y offset	int16	Y offset. Refer **Aligning Widgets**
8	border	uint8	Border thickness in pixels
9	size x	uint16	X axis size in pixels
10	size y	uint16	Y axis size in pixels
11	zero	int32	Slider zero value
12	span	int32	Slider span. The meter will display from zero to (zero + span)
13	color back max	uint32	Slider background max color. RRGGBB color in HEX. Refer **Colors**
14	color back min	uint32	Slider background min color. RRGGBB color in HEX. Refer **Colors**
15	gradient direction	uint8	Direction of gradient if selected. Refer **Gradient Direction**
16	slider color	uint32	Slider color. RRGGBB color in HEX. Refer **Colors**
17	slider opacity	uint8	Opacity to hide slider 0-255. 0 = full opacity (slider not visible). 255 slider full visibility.
18	knob color	uint32	Knob color. RRGGBB color in HEX. Refer **Colors**
19	knob-x	int16	Deviation size from standard knob on x axis (-20 to +60)
20	knob-y	int16	Deviation size from standard knob on x axis (-20 to +60)
21	analog value enable	uint8	0 no text display. 1 to add a text display with engineering units
22	analog value font	uint8	Font. Refer **Font**
23	analog value align to parent enum	uint8	Refer **Aligning Widgets**
24	analog value position x	int16	X offset. Refer **Aligning Widgets**
25	analog value position y	int16	Y offset. Refer **Aligning Widgets**
26	analog value text color	uint32	Text color. RRGGBB color in HEX. Refer **Colors**
27	engu index	int16	-1 not used. Engineering Units index to add after analog value. Refer **<engu>**
28	register	uint16	Register index for analog value. Refer **Register Table**
29	label enable	uint8	0 no label display. 1 to add a label. Refer **<label>**
30	label font	uint8	Font. Refer **Font**
31	label align to parent enum	uint8	Widget Type to align to. Refer **Aligning Widgets**
32	label position x offset	int16	X offset. Refer **Aligning Widgets**
33	label position y offset	int16	Y offset. Refer **Aligning Widgets**
34	label color	uint32	Label color. RRGGBB color in HEX. Refer **Colors.**
35	label index	uint8	Label index. Refer **<label>**
36	widget password	uint8	Password enable > 0. Refer **Password**
37	X10	uint8	If set then the scale is multiplied by 10. This is used for small range settings to improve increments

<switch>

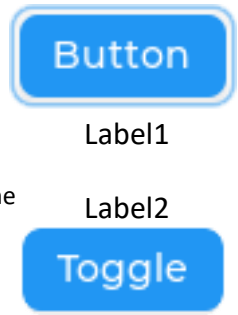
[widget index 11]
A slide switch to set the value of a digital point off or on. A value of 0 is off >0 is on. The on, off states and their respective background colors are configurable. An optional text label can be added.



Line	Key	Value	Descriptor
1	widget index	uint8	My widget Index. **Maximum Personality Model Widgets**
2	screen index	uint8	Screen to place widget on. Refer **Screen Type**
3	align enum	uint8	Refer **Aligning Widgets**
4	align to type	uint8	Refer **Aligning Widgets**
5	align to index	uint16	Refer **Aligning Widgets**
6	x offset	int16	X offset. Refer **Aligning Widgets**
7	y offset	int16	Y offset. Refer **Aligning Widgets**
8	border	uint8	Border thickness in pixels
9	size x	uint16	x axis size in pixels
10	size y	uint16	y axis size in pixels
11	on switch color	uint32	On color of switch. RRGGBB color in HEX. Refer **Colors**
12	on switch background	uint32	On color of background. RRGGBB color in HEX. Refer **Colors**
13	off switch color	uint32	Off color of switch. RRGGBB color in HEX. Refer **Colors**
14	off switch background	uint32	Off color of background
15	register	uint16	Register index for analog value. Refer **Register Table**
16	label enable	uint8	0 no label display. 1 to add a label. Refer **<label>**
17	label font	uint8	Font. Refer **Font**
18	label align to parent enum	uint8	Widget Type to align to. Refer **Aligning Widgets**
19	label position x offset	int16	X offset. Refer **Aligning Widgets**
20	label position y offset	int16	Y offset. Refer **Aligning Widgets**
21	label color	uint32	Label color. RRGGBB color in HEX. Refer **Colors**
22	label index	uint8	Label index. Refer **<label>**
23	widget password	uint8	Password enable > 0. Refer **Password**

<button_t>

[widget index 12]



A button to set digital or multistate value. Can function as a toggle (2 state) or multistate where the rotate can be set to either increase then decrease the value or increase and roll over to 0 when the number of states is reached.
The current state from 0 to number of states is written to the register. The button color scheme follows the current theme and cannot be set in the personality file. An optional text label can be added.

Line	Key	Value	Descriptor
1	widget index	uint8	My widget Index. **Maximum Personality Model Widgets**
2	screen index	uint8	Screen to place widget on. Refer **Screen Type**
3	align enum	uint8	Refer **Aligning Widgets**
4	align to type	uint8	Refer **Aligning Widgets**
5	align to index	uint16	Refer **Aligning Widgets**
6	x offset	int16	X offset. Refer **Aligning Widgets**
7	y offset	int16	Y offset. Refer **Aligning Widgets**
8	border	uint8	Border thickness in pixels
9	size x	uint16	x axis size in pixels
10	size y	uint16	y axis size in pixels
11	background color	uint32	Background color. RRGGBB color in HEX. Refer **Colors.**
12	rotate	uint8	>0 rotate when max states reached. 0 roll from max states to 0
13	states	uint8	how many states. Example 3 states represent 0/1/2. Maximum 5
14	text font	uint8	Font. Refer **Font**
15	text color	uint32	Text Color. RRGGBB color in HEX. Refer **Colors**
16	state 0 text	uint8	State 0 text index. Refer **<state>**
17	state 1 text	uint8	State 1 text index. Refer **<state>**
18	state 2 text	uint8	State 2 text index. Refer **<state>**
19	state 3 text	uint8	State 3 text index. Refer **<state>**
20	state 4 text	uint8	State 4 text index. Refer **<state>**
21	register	uint16	Register index for analog value. Refer **Register Table**
22	label enable	uint8	0 no label display. 1 to add a label. Refer **<label>**
23	label font	uint8	Font. Refer **Font**
24	label align to parent enum	uint8	Widget Type to align to. Refer **Aligning Widgets**
25	label position x offset	int16	X offset. Refer **Aligning Widgets**
26	label position y offset	int16	Y offset. Refer **Aligning Widgets**
27	label color	uint32	Label color. RRGGBB color in HEX. Refer **Colors.**
28	label index	uint8	Label index. Refer **<label>**
29	widget password	uint8	Password enable > 0. Refer **Password**

<spinbox>

[widget index 13]

User input of values. Spinbox has 3 parts – display, button + and button -. The relative location of the buttons to the display is configurable. The number of digits, precision and range of values are all configurable. If the register value is outside of the range, the register value will not be changed but the display will show the range limits (zero or zero + span). The digit that is under the cursor is incremented/decremented. Moving the cursor changes the resolution of the change. An optional text label can be added.

Line	Key	Value	Descriptor
1	widget index	uint8	My widget Index. **Maximum Personality Model Widgets**
2	screen index	uint8	Screen to place widget on. Refer **Screen Type**
3	align enum	uint8	Refer **Aligning Widgets**
4	align to type	uint8	Refer **Aligning Widgets**
5	align to index	uint16	Refer **Aligning Widgets**
6	x offset	int16	X offset. Refer **Aligning Widgets. Note on Spinbox Alignment.**
7	y offset	int16	Y offset. Refer **Aligning Widgets. Note on Spinbox Alignment.**
8	border	uint8	Border thickness in pixels
9	size x	uint16	x axis size in pixels. The height is automatically set to fit the font
10	background color button	uint32	Background color of buttons and selected text
11	text color button	uint32	Text Color of buttons (+/-)
12	background color spin	uint32	Background color of spinbox
13	text color spin	uint32	Text color of spinbox
14	text font	uint8	Font. Refer **Font**
15	button align	uint8	Position of buttons relative to Spinbox display. Refer **Spinbox Button Align**
16	zero	int32	Spinbox zero value
17	span	int32	Spinbox span. The meter will display from zero to (zero + span)
18	digits	uint8	Number of digits including decimal place digits.
19	decimal	uint8	Number of digits before the decimal point. Counted from least significant digit.
20	cursor position	uint8	Default position of cursor.
21	register	uint16	Register index for analog value. Refer **Register Table**
22	label enable	uint8	0 no label display. 1 to add a label. Refer **<label>**
23	label font	uint8	Font. Refer **Font**
24	label align to parent enum	uint8	Widget Type to align to. Refer **Aligning Widgets**
25	label position x offset	int16	X offset. Refer **Aligning Widgets**
26	label position y offset	int16	Y offset. Refer **Aligning Widgets**
27	label color	uint32	Label color. RRGGBB color in HEX. Refer **Colors.**
28	label index	uint8	Label index. Refer **<label>**
29	widget password	uint8	Password enable > 0. Refer **Password**

Note on Spinbox Alignment.

The size of spinbox is automatically calculated based on number of digits and decimal places. The alignment is made with reference to the text box and does not take into account the buttons as these can be defined on top/bottom/left/right in various configurations. Use the x and y offset to get the alignment correct.

<arc>

[widget index 14]

User input of integer values. The display shows a decimal point and can be assigned a float register however the value will only adjust by integer amounts between the set low limit and high limit. Can be placed around a meter to have value adjust and display on the same screen. An optional text label can be added.

Room Setpoint

Line	Key	Value	Descriptor
1	widget index	uint8	My widget Index. **Maximum Personality Model Widgets**
2	screen index	uint8	Screen to place widget on. Refer **Screen Type**
3	align enum	uint8	Refer **Aligning Widgets**
4	align to type	uint8	Refer **Aligning Widgets**
5	align to index	uint16	Refer **Aligning Widgets**
6	x offset	int16	X offset. Refer **Aligning Widgets**
7	y offset	int16	Y offset. Refer **Aligning Widgets**
8	border	uint8	Border thickness in pixels
9	size x	uint16	x axis size in pixels
10	size y	uint16	y axis size in pixels
11	zero	int32	meter zero value
12	span	int32	meter span. The meter will display from zero to (zero + span)
13	angle range	uint16	display angle.
14	angle rotation	uint16	display angle from vertical going anti clockwise.
15	Arc color	uint32	RRGGBB color in HEX. Refer **Colors**
16	Arc opacity	uint8	Opacity 0-255. 0 = full opacity arc background not visible. 255 arc background full visibility.
17	Indicator color	uint32	RRGGBB color in HEX. Refer **Colors**
18	indicator opacity	uint8	Opacity 0-255. 0 = full opacity arc background not visible. 255 arc background full visibility.
19	Knob color	uint32	RRGGBB color in HEX. Refer **Colors**
20	analog value follow	uint8	0 for text to be static. 1 for text to follow knob. Note – y offset not used
21	analog value enable	uint8	0 no text display. 1 to add a text display with engineering units
22	analog value font	uint8	Font. Refer **Font**
23	analog value align to parent enum	uint8	Widget Type to align to. Refer **Aligning Widgets**
24	analog value position x	int16	X offset. Refer **Aligning Widgets**
25	analog value position y	int16	Y offset. Refer **Aligning Widgets**
26	analog value text color	uint32	text color. RRGGBB color in HEX. Refer **Colors**
27	engu index	int16	-1 not used. Engineering Units index to add after analog value. Refer **<engu>**
28	register	uint16	Register index for analog value. Refer **Register Table**
29	label enable	uint8	0 no label display. 1 to add a label. Refer **<label>**
30	label font	uint8	Font. Refer **Font**
31	label align to parent enum	uint8	Widget Type to align to. Refer **Aligning Widgets**
32	label position x offset	int16	X offset. Refer **Aligning Widgets**
33	label position y offset	int16	Y offset. Refer **Aligning Widgets**
34	label color	uint32	Label color. RRGGBB color in HEX. Refer **Colors.**
35	label index	uint8	Label index. Refer **<label>**
36	widget password	uint8	Password enable > 0. Refer **Password**

Widget Enums

Widget configurations use numbers to define the option required. The explanation of these enum's or pre-defined constants are listed in this section.

Screen Type

Defines how the screen is set up. The 2 variants are single or tabbed. For single screens the index is always 0. For tabbed types, the index is from 1 which is the first tab on the left (or top) increasing by 1 for each subsequent tab. Note that the password option MUST be enabled to use passwords on a widget. Each touch widget has a password flag that is only checked if the screen is a password type. This allows for both global password enable/disable and widget level password control.

Number	Enum	Note
0	None	Not used
1	Single	Single Screen. Example Thermostat. No Password
2	Tab	Multiple Screen. Example Display. No Password
3	Single Password	Single Screen. Example Thermostat. With Global Password Enabled
4	Tab Password	Multiple Screen. Example Display. With Global Password Enabled

Screen Orientation

Not used

Screen Tab Location

Number	Enum	Note
0	Left	
1	Right	
2	Top	Used as default as this is easier to read
3	Bottom	

Font

The UTF8 font set is used. To use the degrees symbol **°,** combine the degree ascii code 176 (0XB0) then the unit.
Example for Degrees Celsius is **°C. This is 2 separate characters as opposed to the extended ascii of 1 character.**

Number	Enum	Note
0	Montserrat 12	
1	Montserrat 16	
2	Montserrat 18	
3	Montserrat 24	
4	Montserrat 32	

Text Alignment

Defines the horizontal alignment of text – in most instances this has no effect because the text box is created to the size of the text pixels. Vertical alignment is handled by setting the size of a widget to fit the text correctly if applicable. The vertical position of text within a widget cannot be adjusted.

Number	Enum	Note
0	Auto	Align Text based on widgets
1	Left	Align text to left
2	Center	Align text to center
3	Right	Align text to right

Widget Alignment

Number	Enum	Note
1	In Top Left	
2	In Top Mid	
3	In Top Right	
4	In Bottom Left	
5	In Bottom Mid	

6	In Bottom Right	
7	In Mid Left	
8	In Mid Right	
9	In Center	
10	Out Top Left	
11	Out Top Mid	
12	Out Top Right	
13	Out Bottom Left	
14	Out Bottom Mid	
15	Out Bottom Right	
16	Out Left Top	
17	Out Left Mid	
18	Out Left Bottom	
19	Out Right Top	
20	Out Right Mid	
21	Out Right Bottom	

Colors

Colors are represented as a hexadecimal number of RRGGBB where RR is the red hex value, GG the green and BB the blue.
Examples

- FF0000 Red
- 00FF00 Green
- 0000FF Blue
- 000000 Black
- FFFFFF White

To select a color from a palette, use a color picker – example Color Picker — HTML Color Codes

Gradient Direction

Number	Enum	Note
0	None	No Gradient
1	Vertical	
2	Horizontal	

Spinbox Button Align

Number	Enum	Note
0	LeftRight	Button- on left of display and Button+ on right
1	Top	Button + and – on top of display. Each button half of display width
2	Bottom	Button + and – on bottom of display. Each button half of display width
3	TopBottom	Button + on top of display. Button - on bottom of display

Meter Limits

Defines the limits on a meter widget.

Number	Enum	Note
0	None	
1	Lo Limit Only	
2	Hi Limit Only	
3	Both Hi and Lo Limits	

Aligning Widgets

Using the powerful alignment concept results in visually appealing screens without having to count pixels. Another benefit is not having to adjust all widgets if one widget is moved. If the subsequent widgets are aligned to this widget and each other then only 1 widget needs to be moved and all other widgets on the page will follow.

Widgets have a type and an index. The index is the sequence number of the widget. If we have 3 widgets, they will be numbered 0,1 and 2. The Error! Reference source not found. is defined in widget.txt and is zero indexed. If a widget is not a ligned to another widget then it must be aligned to the screen on which it must be displayed. In this case the widget type will be 0 and the screen will be 0 for a single screen type or the tab number of the screen for the widget.

Widgets can be placed by setting the *align* value of the widget to Default 0. The object will be placed at the absolute co-ordinates of x and y on the screen selected. To simplify placing of multiple widgets, the align value must be set as shown to the value in Figure Alignment Enum below. Example to align a widget to the **out bottom right**, select align value as **15.**

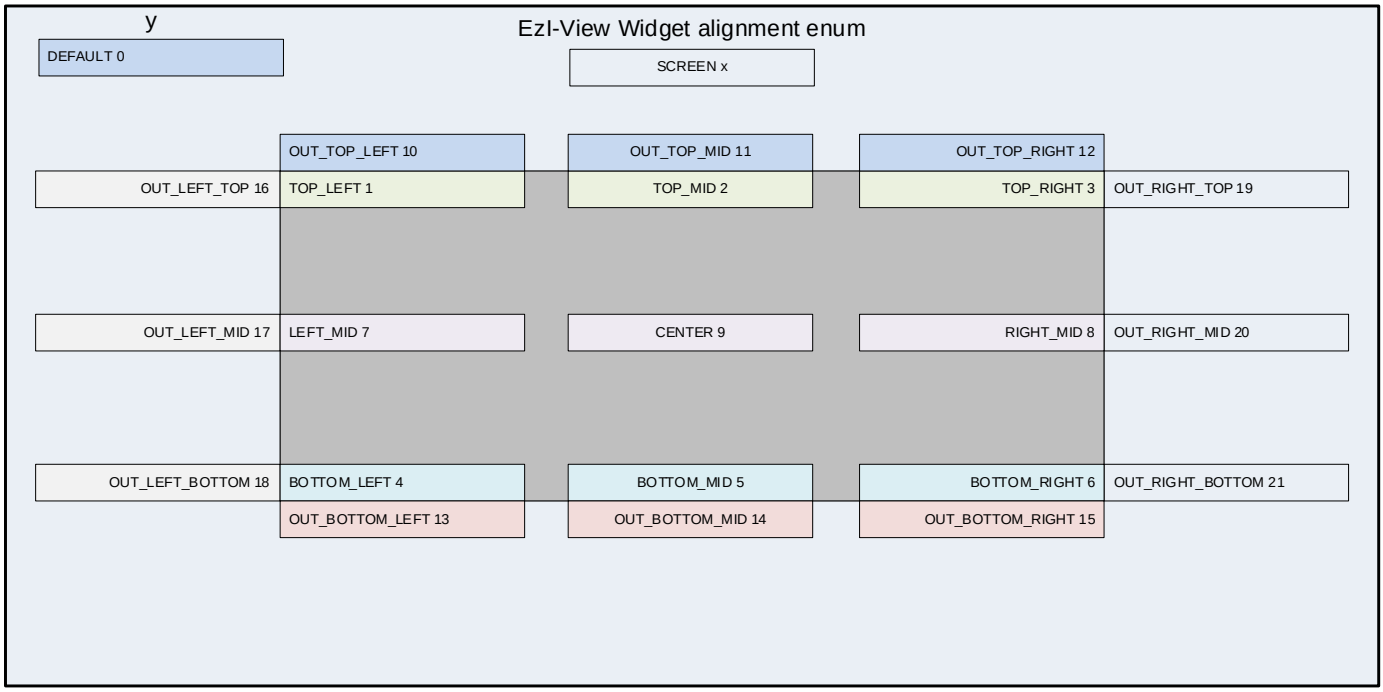


Figure 3 Alignment Enum

The Figure below shows an example of aligning widgets. The meter widget is placed on the screen at absolute co-ordinates x 12 y 40 by setting the align to and align type values to NULL. The text label widget is then placed and aligned to the meter widget type 4 index 2.

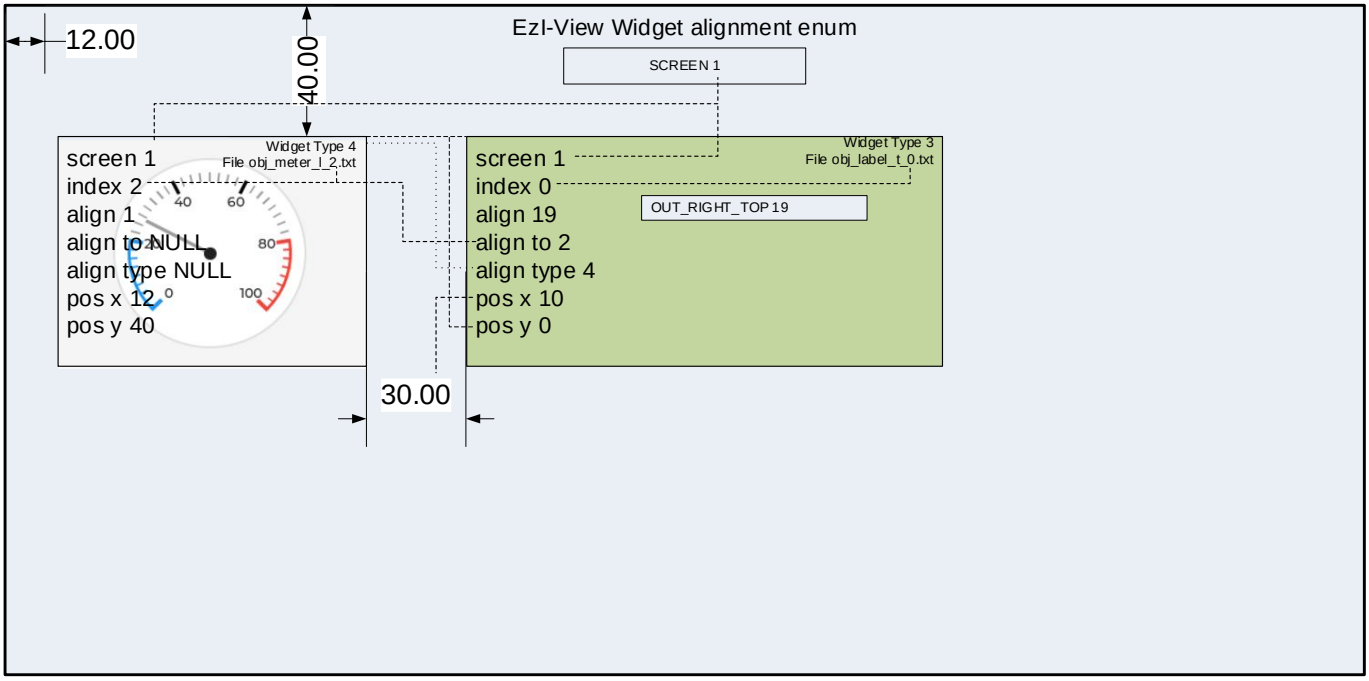


Figure 4 Alignment Example