FX-RP multiDISPLAY programming manual

Getting started with the FX-RP multiDISPLAY

General overview

The FX-RP multiDISPLAY's interface is freely programmable. It consists of "pages" that are made using the Fidelix graphics editor. Links can be made between pages, and the user will navigate in a similar way through the pages. The graphics editor generates .htm pages that have to be converted using the "HTML to Multi Display / Room Display Converter", which generates binary files that are suitable for the FX-RP multiDISPLAY.

Up to 768 kB of data (256 kB for generation 1 displays) can be stored into the internal memory of the FX-RP multiDISPLAY. This is sufficient for most projects, but when more capacity is needed (heavy graphics, lots of pages, …), a µSD card can be used. You can order high quality µSD cards from us, and we strongly advise you to use these µSD cards, as a (cheaper) µSD card breakdown will result in a freezing of the user interface, because new pages are always loaded from the µSD card when this is used. (see Using graphics from a µSD card)

The most common way the display is used, is as Modbus slave, in which case it can hold 250 variables, that are called and referenced to as POINTS.

Each user interface created for the FX-RP multiDISPLAY resides in its own folder. This folder will further be referenced to as the "project folder" in this manual. This folder needs to contain at least 1 .htm file, and a folder called "Symbols" with all symbols used in the project. When no symbols are used, the folder still needs to be present to be able to run the conversion from .htm files to binary files for the FX-RP multiDISPLAY.

The communication between the FX-RP multiDISPLAY as a slave, and a master device uses the Modbus RTU protocol. The FX-RP multiDISPLAY has 250 predefined points (POINTxxx) that can be freely used. Each point has 6 registers containing its value, unit, divider etc. (see Input data and Output data)
When working with other Fidelix devices, the programmer does not need to know about the Modbus registers; for the multi24, all necessary code is generated by the converter, and when working with an FX-controller, the first 50 points can be directly selected as the physical point of any DI, DI, AI, AO or alarm point.
If you are not using a Fidelix controller as your Modbus master, the linking of the FX-RP multiDISPLAY points happens in the same way, only you'll need to program your master to read and write the correct registers. The register structure used by the FX-RP multiDISPLAY is explained later in this document.

Device properties

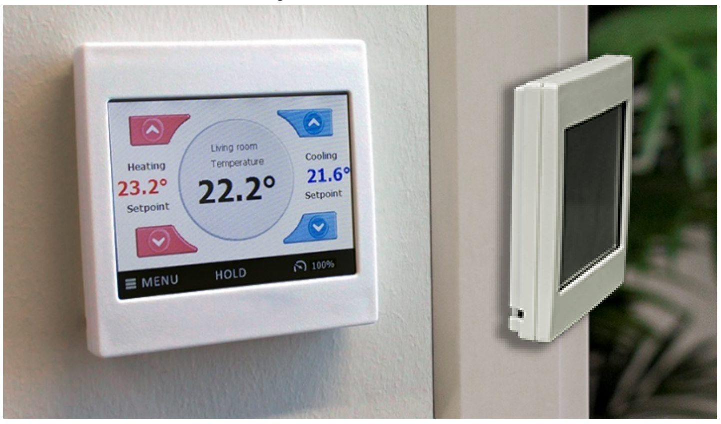
Physical size	3.5" (+/- 9 cm) diameter (72 x 54 mm screen, 85 x 85mm encasing)
Power supply	12-45 VDC or 16-32 VAC (V1 = 12-26VDC or 16-26 VAC)
Resolution	320x240 pixels
Colours	16 bit depth
Images	All standard supported image types. No dynamic gifs
Modbus speed	<u>Slave:</u> auto detect from 4 800 to 115 200 bps <u>Master:</u> freely configurable in ModbusMasterSettings.txt file (see Modbus Master functionality) from 9 600 to 115 200 bps
Modbus settings	<u>Slave:</u> Default settings: 8 Data bits, no parity, 1 stop bit Parity can be changed by placing a file "serial.txt" with content "Parity=Even", "Parity=Odd" or "Parity=No" in the project folder <u>Master:</u> freely configurable in ModbusMasterSettings.txt file (see Modbus Master functionality)
Display "points"	<u>Slave:</u> 250 points total, maximum 40 per page <u>Master:</u> 40 points / page, total depends on the distribution of the registers of the slave device(s) (see the explanation about ModbusDevices at the end of the Modbus Master functionality section)
Available memory	768 kB (V1 = 256 kB)

Different versions

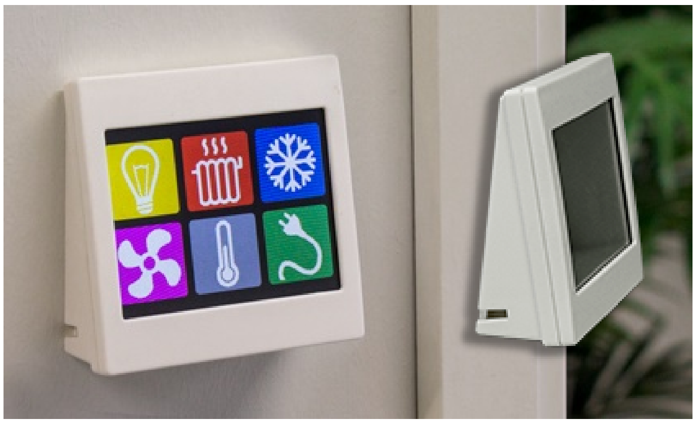
Encasings

The FX-RP (Fidelix Room Panel) multiDISPLAY is available in two different encasings. Both encasings fit in a standard European pattress box (electrical socket or light switch size). Both have external dimensions of 85x85mm.

The FX-RP-A is straight:



The FX-RP-B is +/- 7° tilted upwards:



The FX-RP-C is without encasing:

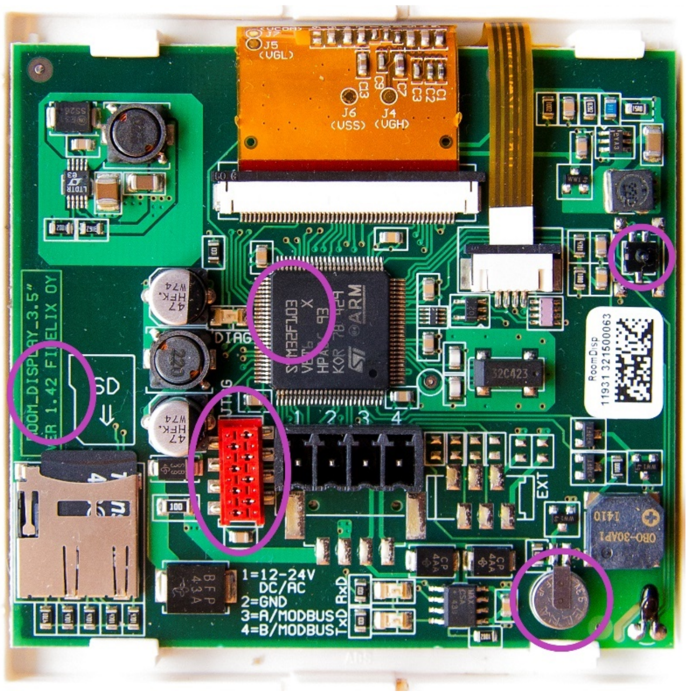
The FX-HP (Fidelix Hand Panel):

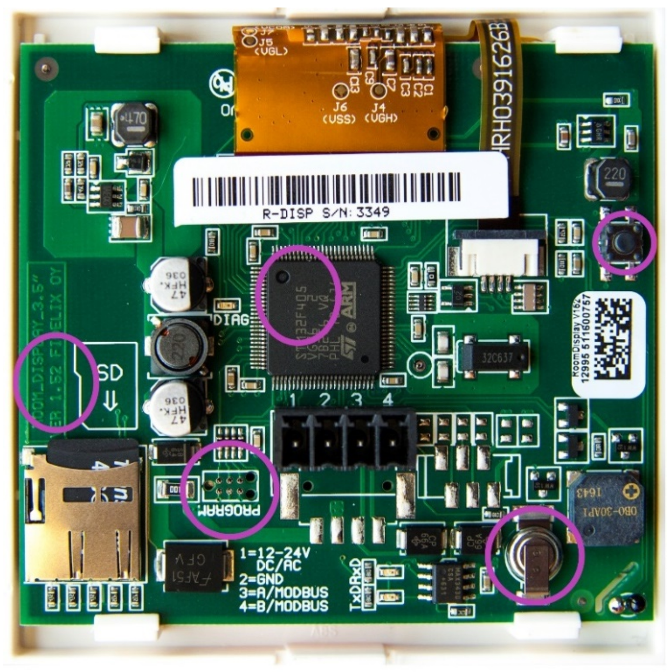
Though the PCB for the Room Panel and Hand Panel have a different shape, they have in fact the same hardware specifications and can thus be used with the exact same software.

Different chip versions

Two different chips have been used for the FX-RP multiDISPLAY; deliveries before 2017 were made with chips with 256 kB (STM32F103), and deliveries from January 2017 onwards are made with chips with 768 kB of memory available for project pages and graphics (STM32F405). This difference is important when selecting the correct version of the firmware for your FX-RP multiDISPLAY. The “first generation” displays use a processor with product code STM32F103 and require firmware with version numbers 0.xx or 1.xx. The second generation displays use a processor with product code STM32F405 and require firmware with version number 2.xx.

You can also recognise the different generation displays by their different programming connector (the red plastic connector, used during production), capacitor (the part looking like a small button battery, holding time schedule settings and the clock for 10-50 hours), and tamper switch, or by the PCB version; 1.52 and upwards = 2nd generation multiDISPLAY.





multiDISPLAY 1st generation
PCB version < 1.52
Processor = STM32F103
Power supply: 12-26 VDC / 16-26 VAC

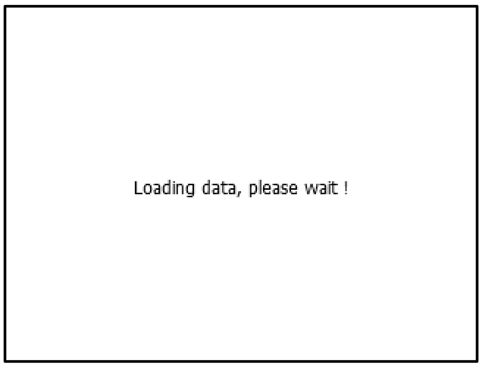
multiDISPLAY 2nd generation
PCB version >= 1.52
Processor = STM32F405
Power supply: 12-45 VDC / 16-32 VAC

Changed components
different capacitor and holder
different programming connector for production
different tamper switch

Using the multiDISPLAY

Startup

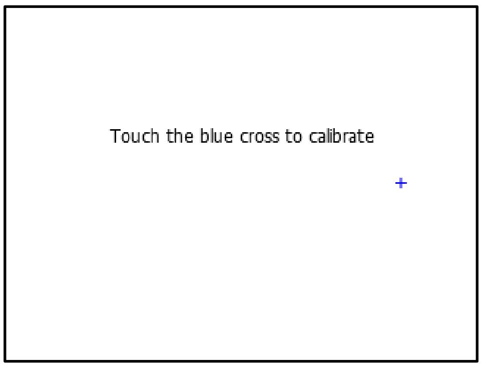
The FX-RP multiDISPLAY is started by connecting the power supply. When a project is detected, either on the internal memory or on the µSD card, this will be loaded into working memory and a message will be displayed. After a few seconds the "home" page of the project will be shown.



Loading data, please wait !

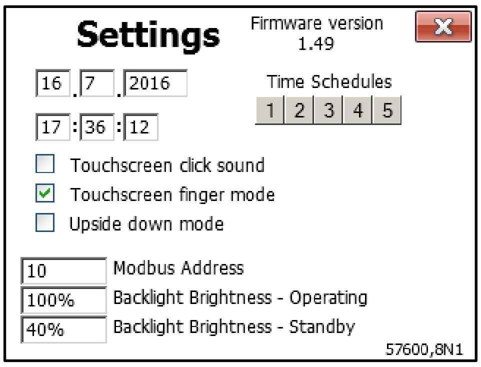
Recalibrating the touch screen and changing settings

By pressing the touch screen of the FX-RP multiDISPLAY for about 10 seconds, the multiDISPLAY's calibration is activated. Follow the on-screen instructions and click each of the three blue crosses one by one with a pointy device. Fingers can also be used, but tend to get less precise results.
Calibration can also be called from a link on the multiDISPLAY.



Touch the blue cross to calibrate

+



Settings

Firmware version 1.49

16 . 7 . 2016

17 : 36 : 12

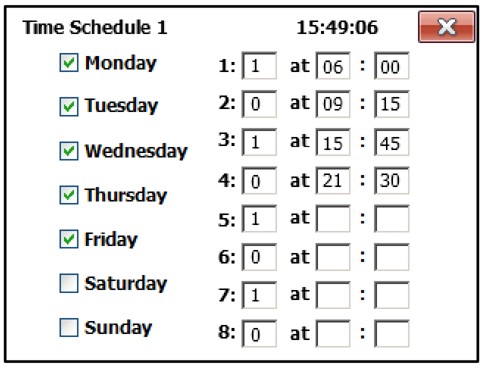
Time Schedules: 1 2 3 4 5

- [ ] Touchscreen click sound
- [x] Touchscreen finger mode
- [ ] Upside down mode

10	Modbus Address
100%	Backlight Brightness - Operating
40%	Backlight Brightness - Standby

57600,8N1

After calibration, the settings screen is opened. Time, Date, Modbus address, display brightness and the five internal time schedules can be adjusted from this page. Also the firmware version is show here, and click-sounds can be en-/disabled, the orientation of the display can be changed and the precision of the screen can be adjusted by selecting "finger mode".



Time Schedule 1 15:49:06

- [x] Monday
- [x] Tuesday
- [x] Wednesday
- [x] Thursday
- [x] Friday
- [ ] Saturday
- [ ] Sunday

1:	1	at	06	:	00
2:	0	at	09	:	15
3:	1	at	15	:	45
4:	0	at	21	:	30
5:	1	at		:	
6:	0	at		:	
7:	1	at		:	
8:	0	at		:	

Here the five internal time schedules can also be accesses and modified.

Note that you can also create your own graphics for the internal time schedules (see the Internal Time Schedules section).

It is possible to disable the settings page or certain parameters through a Modbus register. It is also possible to change settings through Modbus registers. Registers are explained in the "Display parameters" paragraph.

Opening the settings page and starting the calibration can also be done by creating a link on your pages. See the "Links" section in this document.

Updating the firmware of the FX-RP multiDISPLAY

The firmware of the FX-RP multiDISPLAY can be updated using the module update feature (Programming → Module versions) on an FX-controller. The firmware updating procedure is like uploading graphics to the internal memory of the FX-RP multiDISPLAY except that new firmware files should be copied to the \hdisk\fidelix\bin\hex folder instead of the \hdisk\fidelix\data folder. The firmware can also be uploaded to the FX-RP multiDISPLAYs connected to the sub-bus of a multi24 module. The file name for a pass-through updating should be for example PASSTH-12-MULTI-DISP.hex-0173, where 12 is the Modbus address of the FX-RP multiDISPLAY.

The firmware can also be updated by the display itself. Copy the file “RDFW.BIN” you can download from our partner page to the root of the µSD-card. With power off, insert the µSD-card into the display. Power on the display. The “DIAG” LED on the back of the display will blink rapidly during the time the firmware is being updated, and the screen will light up but stay blank during the updating process. The file “RDFW.BIN” will be renamed to “RDFW_LOADED.BIN” to avoid reloading the firmware every time the display is powered on again. If there already is a file called “RDFW_LOADED.BIN”, the renaming of the original file will fail without error message. (This means the next time a display is powered on with that µSD-card inserted, that display will update the firmware, regardless of the fact that it might have done so on the previous boot.)

You can purposely keep the “RDFW.BIN” file to update several displays consecutively, by naming the file “RDFWKEEP.BIN”. (Basically, this has the same effect as having both the “RDFW.BIN” and the “RDFW_LOADED.BIN” file on the µSD-card.)

Using the FX-RP multiDISPLAY with the Fidelix multi24 room controller module

The FX-RP multiDISPLAY can be connected to the multi24’s external bus either using the RJ12 connector or via the P2, G0, EA and EB contacts. The converter will generate a sample IEC-code to be used on the multi24. It is highly recommended to use this sample code at least as a starting point. Sample IEC-files suitable to use directly with your project are stored in the \UserFiles\IEC folder in the same folder the converter is located.

Demo projects

Fidelix provides a file with demo projects for the FX-RP multiDISPLAY. These demo projects are a good starting point for learning how to use the FX-RP multiDISPLAY. Note that these are sometimes incomplete, and in no way cover the totality of functionalities the FX-RP multiDISPLAY offers. They are, however, a valuable source of inspiration and are very useful as examples.

Software

Overview

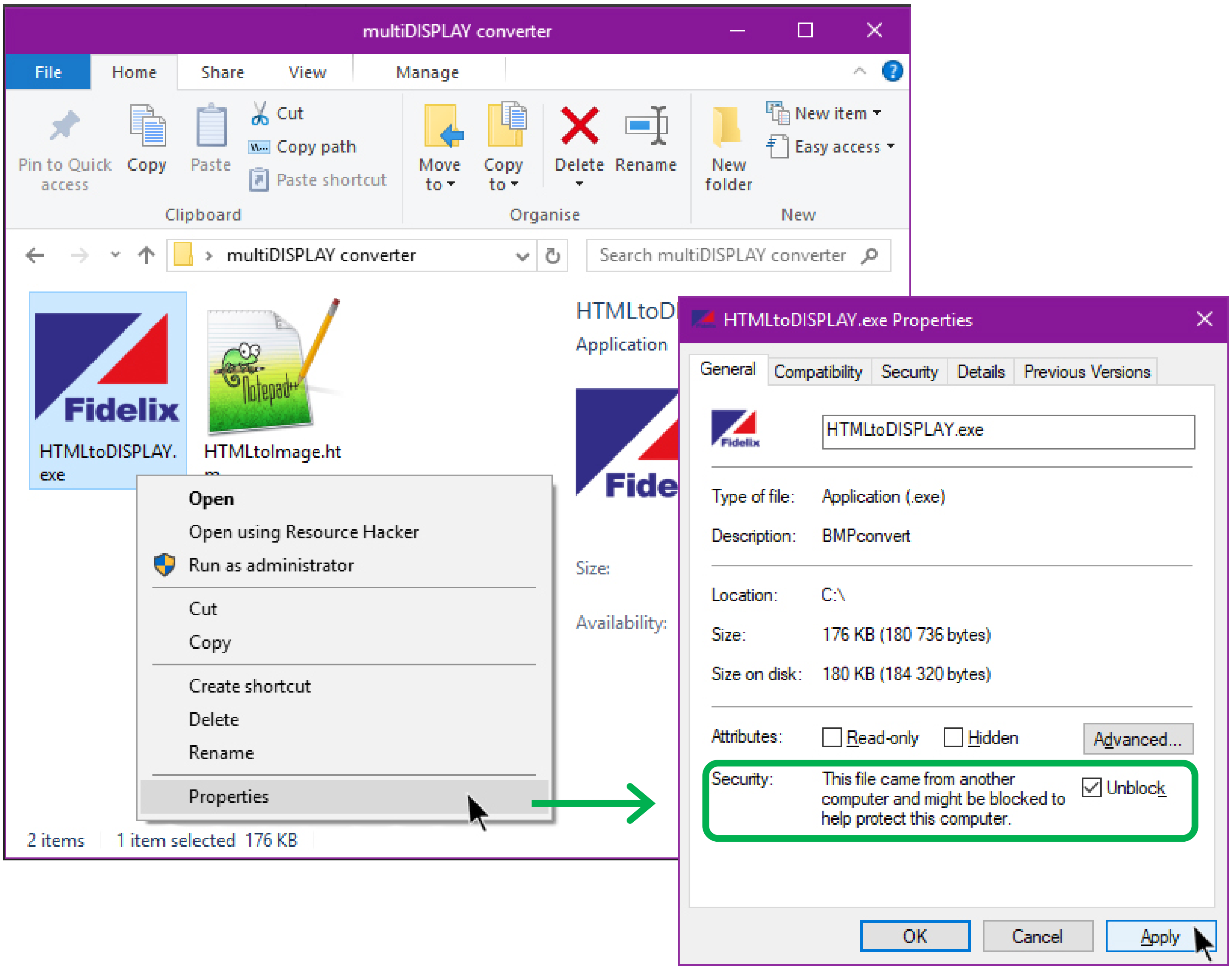
To make projects for the FX-RP multiDISPLAY, two programs are needed, that can both be downloaded from our partner page. The usual work flow will be as follows:

- Create pages to be loaded onto the FX-RP multiDISPLAY with the Fidelix Graphics Editor (.htm file format)
- Convert those pages with the HTML to Display converter program
- Copy the “UserFiles” folder generated by the Converter program to a µSD card
- Start up the FX-RP multiDISPLAY with the µSD card in it and wait for the project to be loaded into the internal memory
- Power off the FX-RP multiDISPLAY, remove the µSD memory card and power the display up again; your project is now loaded into the display and ready to be used

Installation

Our software is designed for Windows, and uses a lot of its embedded features. It also doesn't have an installation wizard where a User Account Control check could be done to give administrative rights to the programs, which is why all our programs need to be granted sufficient rights in another way. There are two ways to do this:

The first is by unblocking the downloaded files (in the file properties dialog box) in Windows file explorer as follows:



HTMLtoDISPLAY.exe

Type of file: Application (.exe)

Description: BMPconvert

Location: C:\

Size: 176 KB (180 736 bytes)

Size on disk: 180 KB (184 320 bytes)

Attributes: ☐ Read-only ☐ Hidden Advanced...

Security: This file came from another computer and might be blocked to help protect this computer. ☑ Unblock

NOTE: This needs to be done for all programs, as well as for all files!

The second way to achieve the same goal is by following the steps below carefully; sometimes, the unblocking doesn't seem to be enough, and then the following method needs to be used:

1. Unzip the program (preserving the subfolder structure!) and save the folder on your local hard drive.
2. Delete the downloaded zip-file.
3. Re-zip the whole folder you just unzipped locally and save the new zip-file on your local hard drive
4. Delete the previously unzipped folder (the folder unpacked from the downloaded zip-file).
5. Now, from the locally generated zip-file, unzip again the same folder (again, preserving the subfolder structure!) and save the folder on your local hard drive.
6. Delete the locally created zip-file.
7. Do the same for all software AND the <u>example folders</u>. (**IMPORTANT!**)

Fidelix Graphics Editor

What you should know before you edit pages

The Fidelix Graphics editor is not only used for making graphics for the multiDISPLAY. It is also used to make pages for our larger controllers; the FX-line and Spider, for our multiLINK protocol converter, and our webVision SCADA software. This means that some of the features in the graphics editor are not used for the multiDISPLAY (and that is also why we first make HTML pages to subsequently convert them to a format that the multiDISPLAY can read).

To open pages with the graphics editor, make sure your project folder structure is similar to this: all .htm files in 1 folder where there is also a folder called “Symbols”.

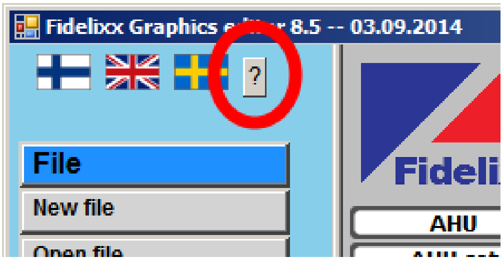
In the “EditPoint” dialog box, following features are NOT recognised by the multiDISPLAY: CSS file, CSS Class, User level, Show info, Image rotation, Image Front colour and Symbol Front colour. All other features can be used as with the other products.

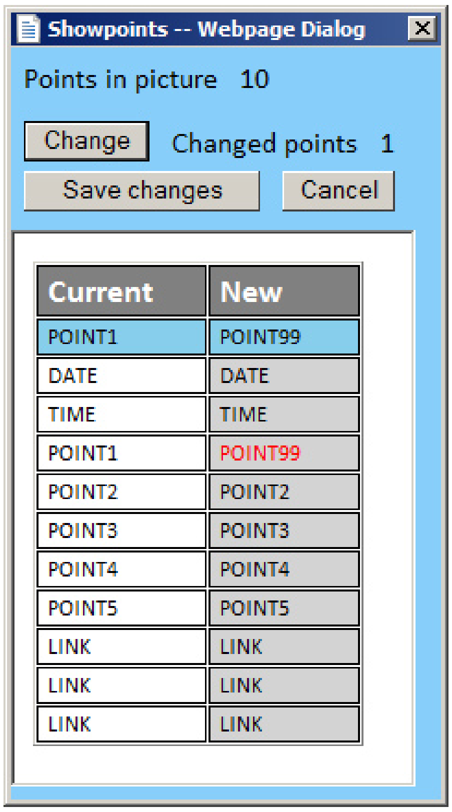
The border-lines you see changing at different editing sizes (640, 800, 1024) are not used for the multiDISPLAY. For the multiDISPLAY, the easiest way to mark the available graphics area is to put an element without pointID at 320 (X) and 240 (Y) pixels like in the example files.

Important to know is also that the project folder and the graphics editor program folder have to be on the same logical Windows drive (e.g. C:\) because of the same Windows rights reasons the editor needed to be unblocked.

With the editor, a set of .htm pages can be made. Links can be made between pages. Any number of elements can be placed on the plane, keeping in mind the 320x240 pixel size of the display itself. This means that any element positioned outside these boundaries will be ignored by the converter.

Make sure to read through the cheat sheet page that pops up when you click the question mark next to the three flags. It contains useful tips for working with the graphics editor.




Showpoints -- Webpage Dialog

Points in picture 10

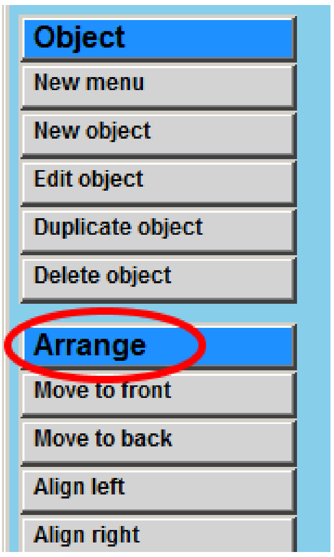
Change Changed points 1

Save changes Cancel

Current	New
POINT1	POINT99
DATE	DATE
TIME	TIME
POINT1	POINT99
POINT2	POINT2
POINT3	POINT3
POINT4	POINT4
POINT5	POINT5
LINK	LINK
LINK	LINK
LINK	LINK

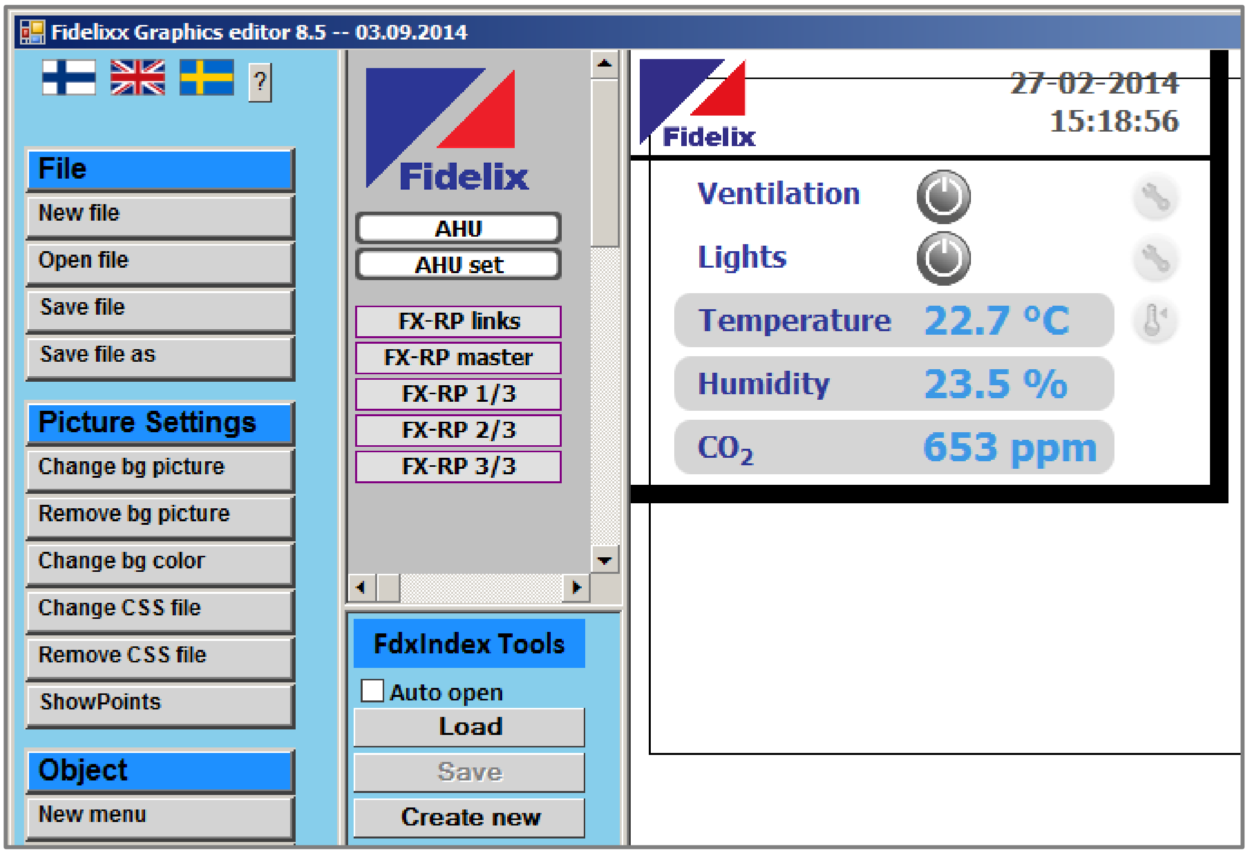
Choose “ShowPoints” in the left menu or click right on an empty spot in your page with no element selected and select “Show Points” for a fast way to batch change multiple points in your page; for instance, if you need to make 5 identical pages for 5 devices, you can very quickly rename the points for device 1 as 11, 12, 13 … 19, for device 2 as 21, 22, 23 … 29 etc.

Make sure to click the “Arrange” button in the bottom left corner of the screen to have access to some other very useful features of the graphics editor.



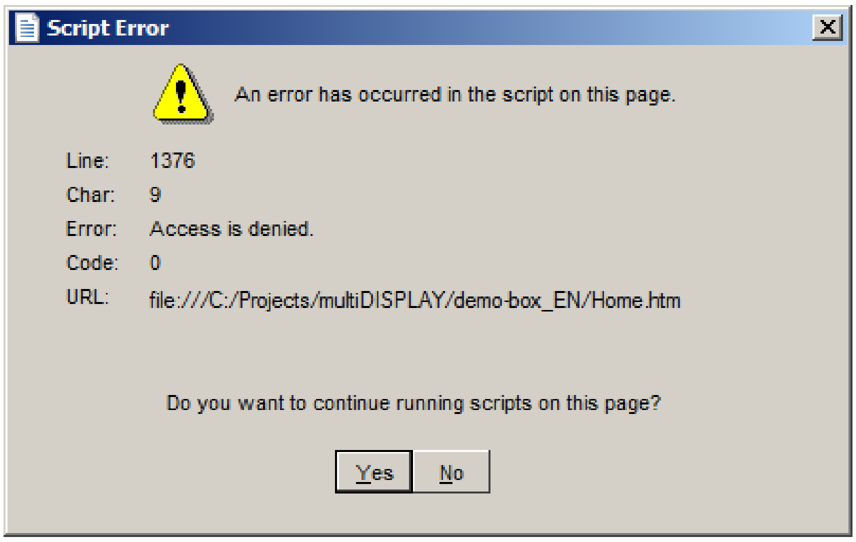
The FXINDEX page

After you have opened at least one page from the project folder, you can click the “Load” button in the middle column of the program. This will load the file called “FDXINDEX.HTM” to the centre column. This page is loaded from your project folder. You can also open this page to easily add links to other pages you make yourself. This menu is nowhere visible on the multiDISPLAY, it is just a faster way to navigate through the pages in your project folder.



Be careful when switching from page to page through the menu; any unsaved changes will be lost, and there is NO popup box to remind you to save changes!

Caution! Trying to load the FXINDEX page without it being present in your working folder will cause an error like this to pop up, and will cause the editor to completely freeze or get stuck in an endless loop of error report popups.


Script Error

An error has occurred in the script on this page.

```
Line:   1376
Char:   9
Error:  Access is denied.
Code:   0
URL:    file:///C:/Projects/multiDISPLAY/demo-box_EN/Home.htm
```

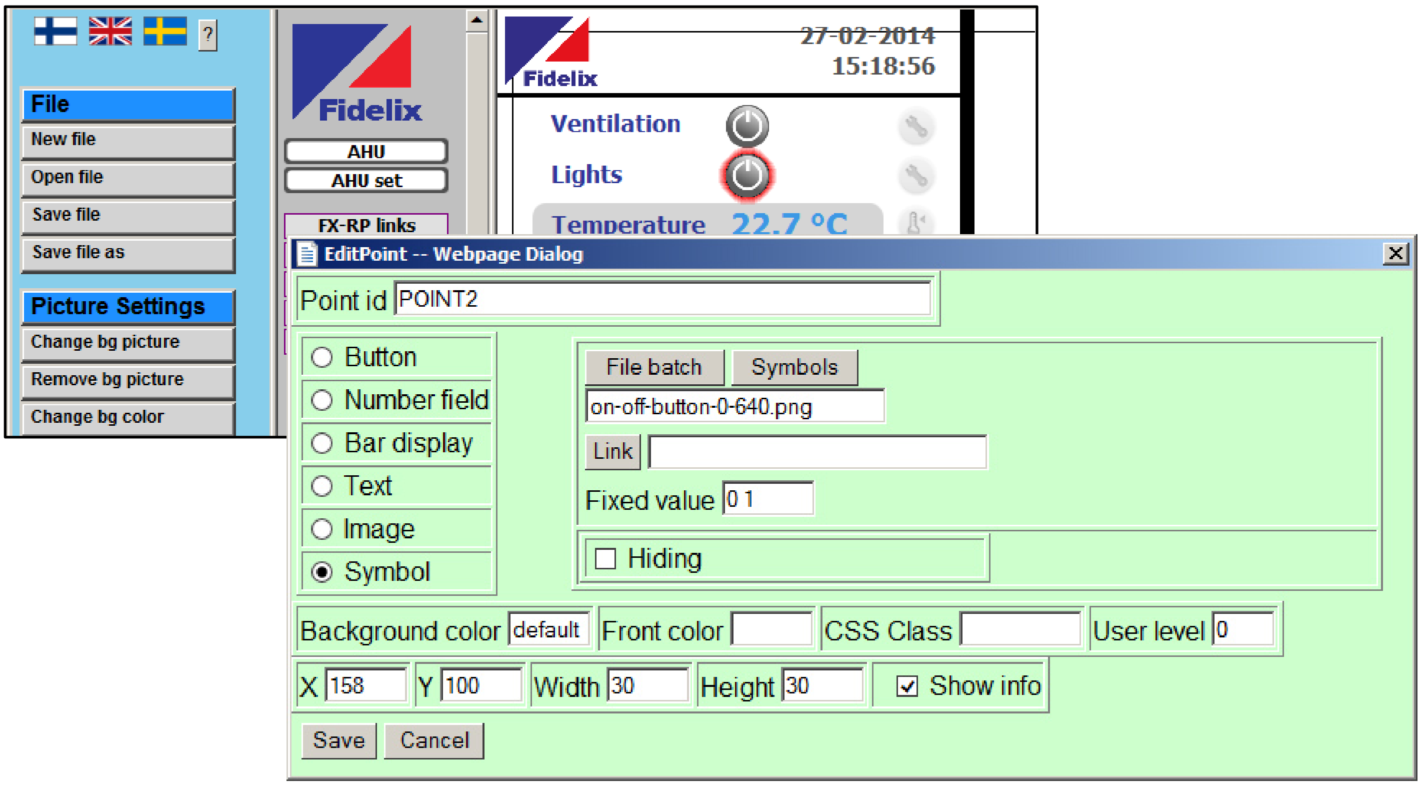
Do you want to continue running scripts on this page?

The reason for this is because the graphics editor uses Internet Explorer scripts for its base functionality (that is why it is so small) that cannot be stopped from outside of Internet Explorer,

So, make sure you have the file present in your working folder before loading it!

Editing pages (a few pointers)

- Once a page is opened inside the Graphics editor, you can drag elements around with the mouse or with the keyboard as you like (shift+arrow = 10px movements).
- To open the EditPoint window, double click an element or press Ctrl+E.
- Note the "border" images at X=320 and Y=240 in many of the example files to indicate how big the multiDISPLAY is. As they are outside the 320x240 drawing area, the converter will simply ignore them.
- Any graphical element can be used, but the naming should follow the conventions like in the "symbols" folder inside your project folder. (e.g. Filename-0-640.jpg, Filename-1-640.jpg, Filename-2-640.jpg, …)
  Graphical elements have to be either .bmp, .gif (dynamic gifs don't work on the display), .jpg, .jpeg or .png.
- Display points are referenced by their pointID; POINT1 through to POINT250.
  Modbus master points are defined by using "MODBUS:" followed by the configuration of the point in the pointID input field.
- If you want to use a multi-stage image (like the power buttons in the example pages), enter the desired values through which to toggle separated by spaces in the "Fixed value" field of the EditPoint dialog box:



Point id: POINT2

[✓ Symbol]

on-off-button-0-640.png

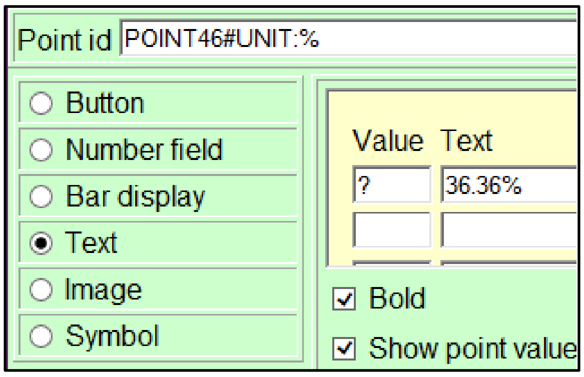
Fixed value: 0 1

☐ Hiding

Background color: default | Front color: | CSS Class: | User level: 0

X: 158 | Y: 100 | Width: 30 | Height: 30 | ☑ Show info

- Showing the value of a point is done by either selecting a "Number field" type element, or by ticking the "Show point value" tick box of a "Text" type element.



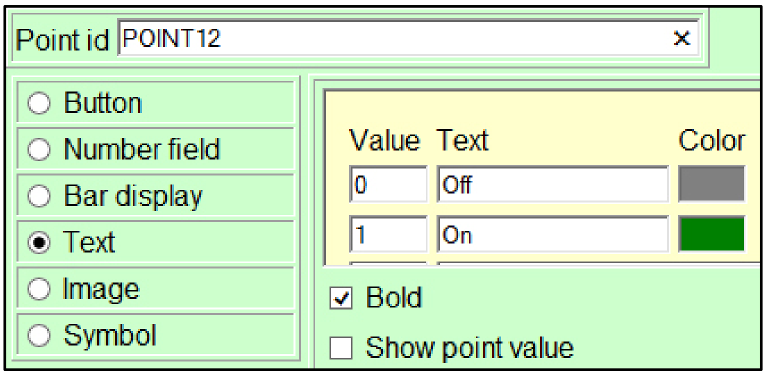
Point id: POINT46#UNIT:%

[✓ Text]

Value	Text
?	36.36%

☑ Bold

☑ Show point value



Point id: POINT12

[✓ Text]

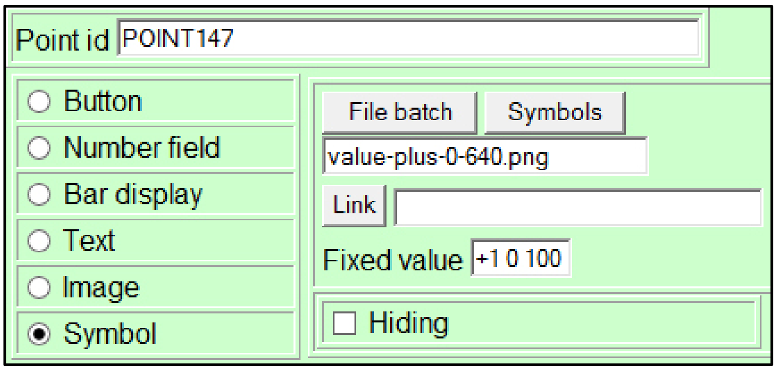
Value	Text	Color
0	Off	⬜
1	On	🟩

☑ Bold

☐ Show point value

- Showing text based on the value of a point (like for instance on/off, day/night or off / startup / slow / fast / error) is done by selecting a text type element and writing the desired values and corresponding texts in the yellow box inside the EditPoint dialog box.

- Making "+" or "-" buttons can be done using symbols and putting the step value, minimum and maximum value in the "Fixed value" field


Point id: POINT147

Button ( ) Number field ( ) Bar display ( ) Text ( ) Image ( ) [✓ Symbol]

File batch | Symbols

value-plus-0-640.png

Link:

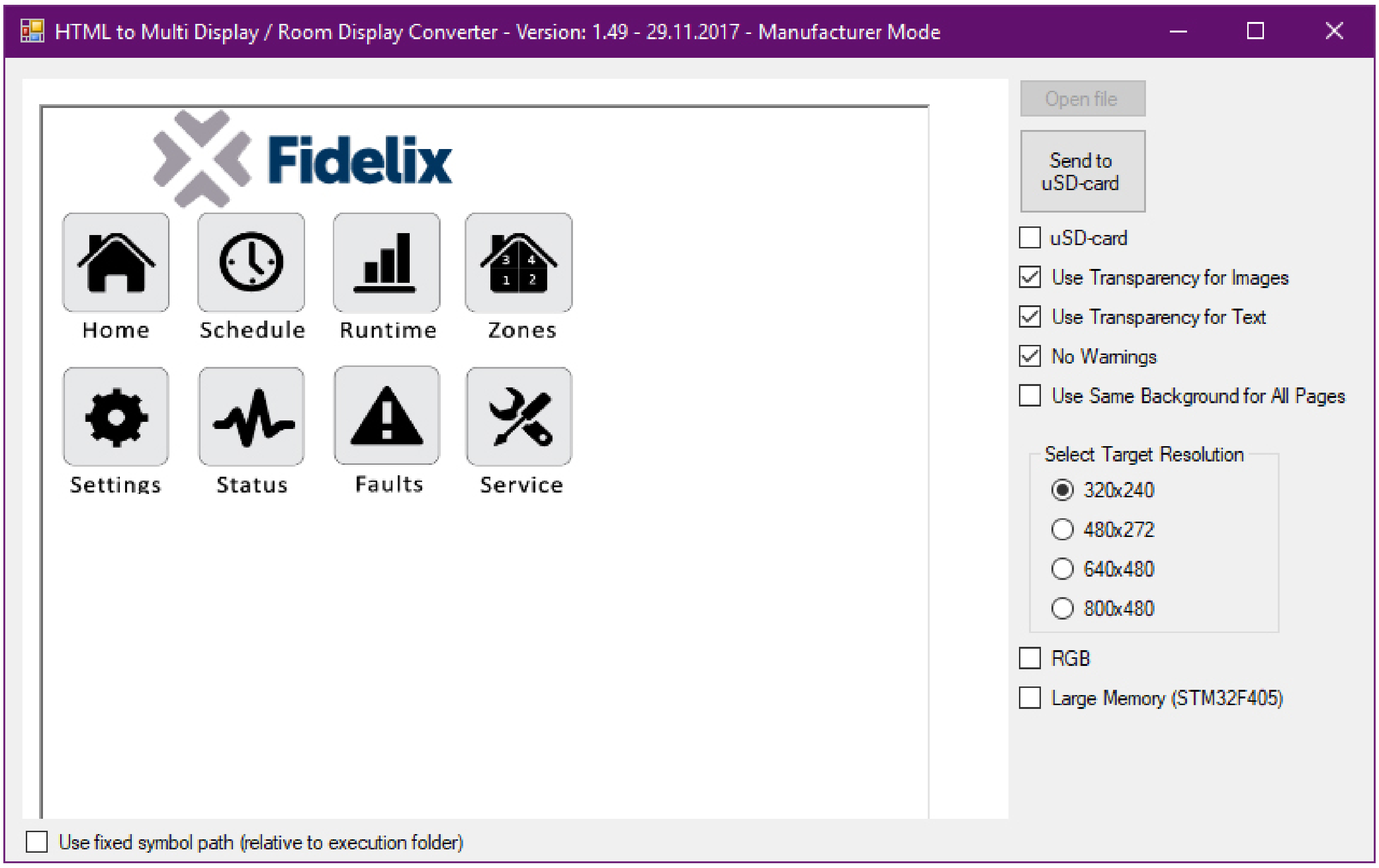
Fixed value: +1 0 100

☐ Hiding

For more detailed instructions on how to use the Fidelix Graphics editor, consult the graphics editor's own reference manual on our partner webpage, and read further in this manual on the FX-RP multiDISPLAY specific syntax in the detailed programming section.

HTML to Multi Display / Room Display Converter

The program's UI


HTML to Multi Display / Room Display Converter - Version: 1.49 - 29.11.2017 - Manufacturer Mode

☐ uSD-card
☑ Use Transparency for Images
☑ Use Transparency for Text
☑ No Warnings
☐ Use Same Background for All Pages

Select Target Resolution: [✓ 320x240] ( ) 480x272 ( ) 640x480 ( ) 800x480

☐ RGB
☐ Large Memory (STM32F405)
☐ Use fixed symbol path (relative to execution folder)

The "Open file" button is used to start the process. Select the .htm page that you want to use as a start page each time the display is powered on. Also, any links inside your project to "CLOSE", will direct the user to this page.

The converter can also be used to check how much memory a single image will take after conversion. Instead of opening an .htm file you simply open the image and the converter will check how much memory will be needed using different packing methods. The best packing method is automatically selected during the conversion.

- The “Send to uSD-card” button copies the conversion result to a removable memory device containing a “UserFiles” folder.
  NOTE: When using an empty µSD card for the first time, a folder called “UserFiles” must be created manually, or the “UserFiles” folder, generated by the converter must be copied manually.
  NOTE: The converter program searches for a Windows removable memory drive that has a folder called “UserFiles” in its root alphabetically. This means that, for instance, if there is a drive “G:” that contains another memory card which also has a “UserFiles” folder, this folder will be overwritten.

- The “uSD-card” tick box should only be selected (ticked) if you want to use the project from a µSD memory card. When not ticked, the project size is limited to 768 kB (256 kB for V1 displays) and all project files will be copied into the FX-RP multiDISPLAY’s internal memory upon startup of the display. A µSD-card can be used for projects that are bigger than the internal memory, in which case only the font files need to fit into the internal memory of the FX-RP multiDISPLAY. By checking the µSD-Card checkbox, the converter will not make one file of the project, to be loaded into the internal memory, but instead, will generate a file containing only the used fonts.

  During development, it is recommended to use the “µSD-card” feature. This will speed up the starting of the display as only fonts need to be loaded into the memory.

  During commissioning the tick box should not be ticked. Make a final conversion and copy the project onto a µSD card. Now, simply put the µSD card in the display, power up the display and wait for a few seconds while the project is loaded into the internal memory. Now, take the power off the display, remove the µSD card, and power up the display. You are now running your project from the internal memory.

- The “Use Transparency for Images” selection defines if you want to have transparent background for images. The loading of images takes more time if a transparent background is used. If not selected then instead of transparency, the page’s background colour is used.

- The “Use Transparency for Text” selection defines if you want to have transparent background for text. The loading of text takes more time if a transparent background is used. If not selected then instead of transparency, the page’s background colour is used.

- The “No Warnings” selection discards some not so important warning messages during the conversion process.

- The “Use Same Background for All Pages” selection is used if you want to have the same background image for all pages. This saves memory and helps to keep project size under 768kB. When using this feature, you have to define all objects as active object, because all static objects are discarded in this case. This means that for example you may define the page title as text object having pointID “Title”, in which case it is generated as an active object, but not linked to any physical register.

- The “Large Memory (STM32F405)” tick box should be ticked when you are converting a project for a V2 display (so, purchased later then 2016). This will allow for the full memory to be used. If you are converting a project to be used with a V1 display, which only has 256 kB of memory available for projects, make sure the box is unticked.

Converting HTML files to the DISPLAY format

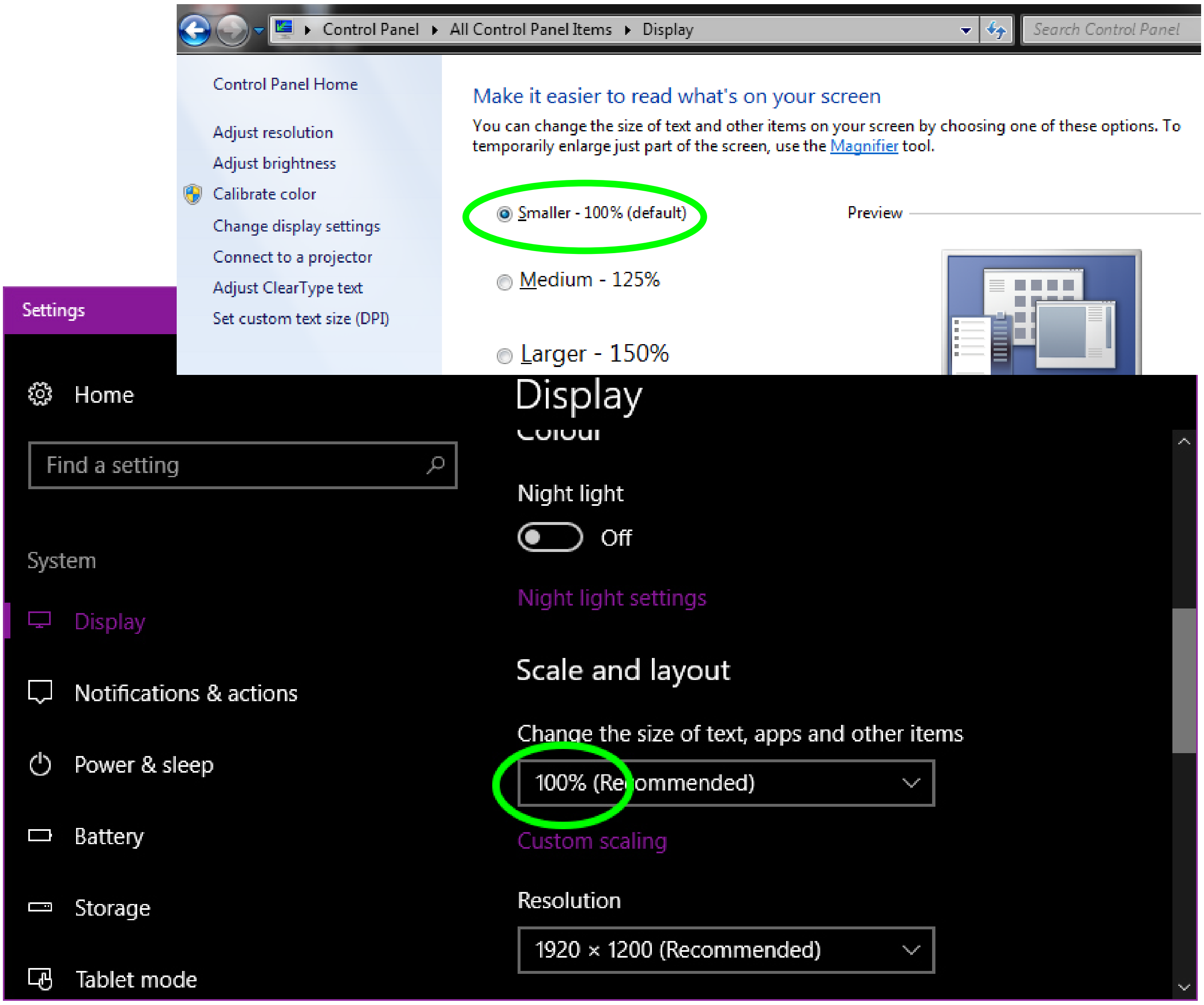
The HTMLtoDisplay program is used to convert HTML files to the format supported by the FX-RP multiDISPLAY. Conversion is started by opening the .htm file of the project’s main page. This is the page that will show up when the display is power up, or when a link pointing to “CLOSE” is used. The converter will go through the links in the page and any linked page. This means the converter needs to be started only once. If any dead links are found, the process will be aborted. During conversion, the results will be stored into \UserFiles folder.

This means that if you want to have pages that are forced to show up by the Modbus master, but that cannot be accessed through the normal UI, you have to include a “hidden” link somewhere in the project. This hiding can be done -for instance- by having a password protected settings page with on it a 1x1 pixel link that uses the "Hiding" feature. This way the chances of accidentally opening that page are reduced to almost none.

Windows zoom settings and the converter

The converter works in such a way, that it loads each page, and then takes a screenshot of your actual computer screen. This means that during conversion, no other windows are allowed inside or on top of the converter window, as whatever is placed on top, will be visible on the display.

Make sure to set the Microsoft Windows zoom or scale factor to 100% in the Windows settings:



Make it easier to read what's on your screen

You can change the size of text and other items on your screen by choosing one of these options. To temporarily enlarge just part of the screen, use the Magnifier tool.

[✓ Smaller - 100% (default)] Medium - 125% Larger - 150%

Display

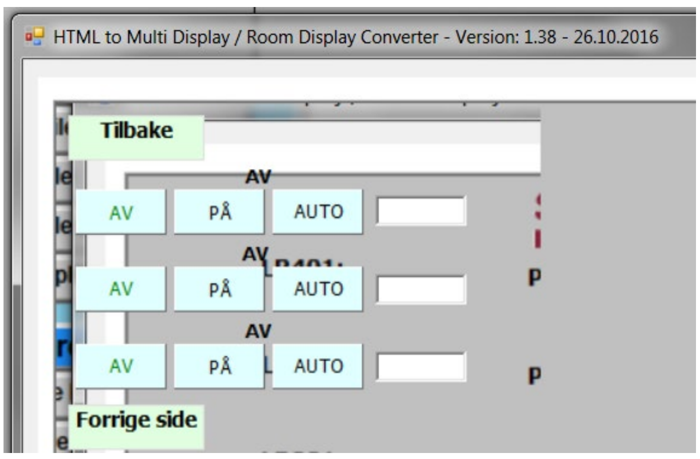
Night light: Off

Scale and layout

Change the size of text, apps and other items: [✓ 100% (Recommended)]

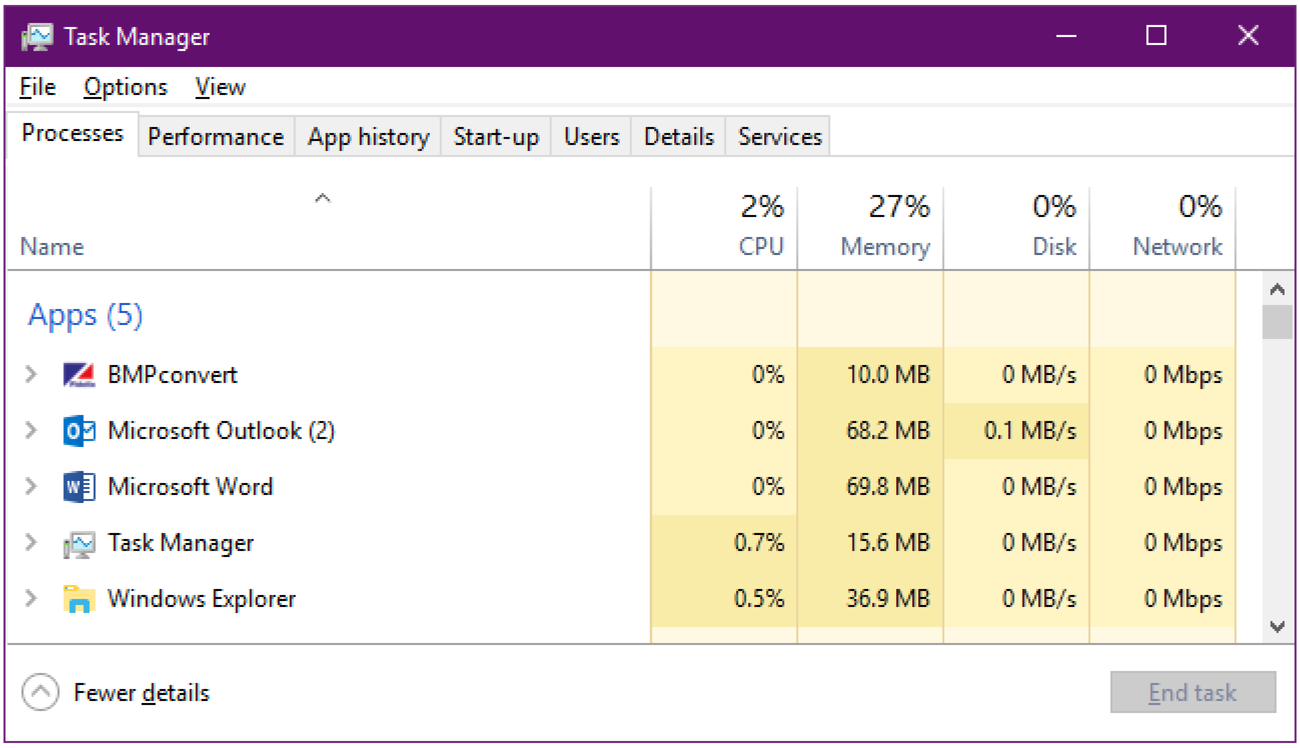
Resolution: 1920 × 1200 (Recommended)

Having the wrong zoom factor will result in a converter window looking like this:


HTML to Multi Display / Room Display Converter - Version: 1.38 - 26.10.2016

Troubleshooting the converter

The converter works using a lot of embedded Windows and Internet Explorer features. This unfortunately also means that it is recommended to do only 1 single conversion, and then close the program. You will easily encounter program crashes when doing multiple very similar conversions. IN that case, it might happen that an instance of the program stays in the computer memory, even while visually closed. In such a case, start the Windows task manager and manually stop the "BMPconvert" process:


Task Manager
File Options View
Processes Performance App history Start-up Users Details Services
Fewer details
End task

Name	2% CPU	27% Memory	0% Disk	0% Network
Apps (5)				
BMPconvert	0%	10.0 MB	0 MB/s	0 Mbps
Microsoft Outlook (2)	0%	68.2 MB	0.1 MB/s	0 Mbps
Microsoft Word	0%	69.8 MB	0 MB/s	0 Mbps
Task Manager	0.7%	15.6 MB	0 MB/s	0 Mbps
Windows Explorer	0.5%	36.9 MB	0 MB/s	0 Mbps

Getting projects on the multiDISPLAY

Projects can either be loaded into the internal memory of the multiDISPLAY, or be stored on a µSD card. The advantage of the internal memory is its speed, the advantage of the µSD card is its size. Most projects will easily fit into the internal memory of the FX-RP multiDISPLAY, but for complex, extensive graphics with many pages, if might be necessary to use a µSD memory card.

We recommend using the internal memory as long as this is possible, because this will limit the number of read operation from the µSD card and thus guarantee a longer life span of your project.

Loading projects into the FX-RP multiDISPLAY's internal memory

Make sure that when you want to use the graphics from the FX-RP multiDISPLAY's internal memory, the "uSD-card" tick box is NOT ticked. When the box is not ticked, the converter will generate 1 file that contains the whole project and that will be copied into the display's internal memory. When the box is ticked, the same file will only contain the used fonts of the project, and links to the files on the µSD card.

With an FX-controller

From an FX-Controller it is possible to load the generated binary file to the FX-RP multiDISPLAY through the module update feature on the controller (> Programming > Module versions). In order to see the graphics in the list, copy the generated "MULTI-DISP.dat-xxxx" file you will find inside the "UserFiles" folder on your PC to the \HDisk\Fidelix\DATA folder on the FX. Note that this folder is different than the one used for firmware updates. The module update feature can also be used to update graphics of FX-RP multiDISPLAYs connected to a multi24's external bus. This feature is called "pass-through". To enable the pass-through feature, the "MULTI-DISP.dat-xxxx" file name has to be changed to have the PASSTH-XX- prefix, where XX is FX-RP multiDISPLAY's Modbus address on the external bus. For example, when uploading a project to an FX-RP multiDISPLAY, connected to the external bus of a multi24 on address 10, the file name should be "PASSTH-10-MULTI-DISP.dat-xxxx". New files copied to the \hdisk\fidelix\data folder will automatically show in the firmware selection frame.

With a multi24 module and a PC

Using the multi24 programming tool, the same ".dat-xxx" files can be selected in the "Display graphics" section. The program will then ask the Modbus address of the display and the multi24 will be used as pass-through handler.

With a µSD memory card

It is possible to update the internal memory from a µSD-card. During the power up sequence, the FX-RP multiDISPLAY will check if a "MULTI-DISP.dat-xxxx" file is found on the µSD-card and if found the file is copied into the internal memory. After that, the µSD-card can be removed, and the display restarted (power off and back on). From then on, the internally stored pages are used. The advantage over using the graphics from the µSD card is the obvious lower number of reads from the µSD card, but mainly the speed.

Make sure to verify if your FX-RP multiDISPLAY is of the first generation or the second, as the available memory is tripled in the V2 displays. In the converter, you can tick the "Large Memory (STM32F405)" tick box when you have a V2 display so the converter will allow you to convert larger projects.

Using graphics from a µSD-Card

During the development stage of graphics or when your project doesn't fit the internal memory, you can use a µSD-card to store graphics. To do so, tick the "uSD-card" tick box in the converter and copy the generated "UserFiles" folder to the µSD-card using a memory card reader. Insert the card into the memory card slot of the FX-RP multiDISPLAY and reboot the FX-RP multiDISPLAY.

By using a µSD-card you can quickly test if graphics are working as expected without the need to load images into the internal memory of the FX-RP multiDISPLAY. A µSD-card also needs to be used if more than 768 kB storage space is needed.

When using a µSD memory card, please make sure you are using cards of good quality from a trustworthy manufacturer. SDXC cards are not readable by the FX-RP multiDISPLAY.
A good reference list for memory cards' quality can be found here: https://elinux.org/RPi_SD_cards.

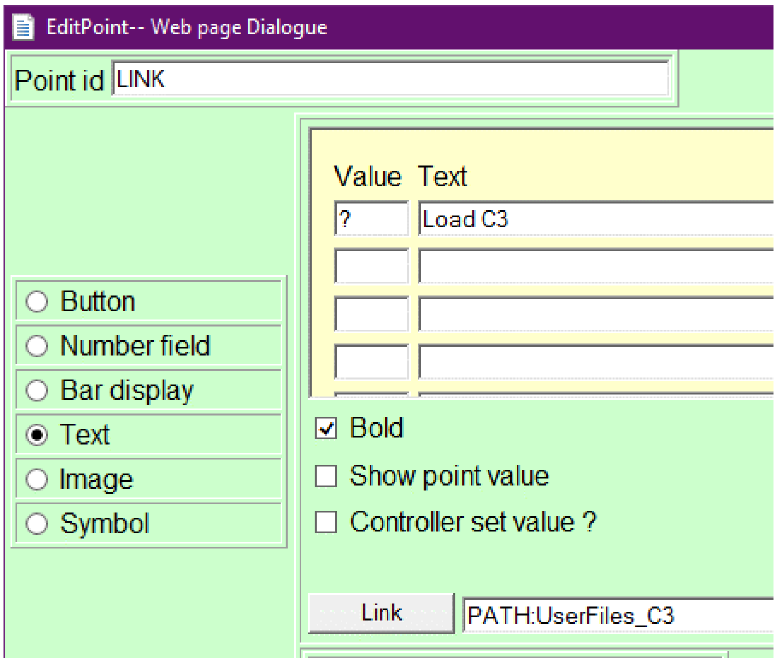
Minimizing binary file size

If your project does not fit into the 768kB (or 256 for V1 displays) reserved for internal data storage, you may either use a µSD-card for data storage or try to optimise your project to use less memory. Here are some tips to keep your total project size as small as possible:

- Do not use more than 255 colours in images. The easiest way to accomplish this, is to convert your images. The embedded compression of the converter only works with < 255 colours. Any images using more than 255 colours will not be compressed at all.
- Do not change the image size using object properties. Instead use the real height and width of the image.
- Minimise the colour count in pictures. Large areas using a single colour will not take much memory. Even if the colour count is less than 255, more memory is needed if more colours are used.
- For dynamic objects, use Text size 12. The "bold" option can be used. Font size 12 is embedded in the firmware, all other font sizes will take up space from the internal memory.
- If a larger font size is needed, keep in mind that a larger font size takes more memory than a smaller. So do not use larger fonts than needed.
- Minimise different font sizes used. Every different font size uses some space of the memory.
- Try to have only "active" elements; this will make the background image easier and thus smaller. You can make an element active by giving it a PointID. Make sure not to use a significant one already in use, like "POINTxx", "LINK", but rather go for something like "UNUSED_ELEMENT" or "IMAGE".

Multiple projects on one µSD-Card

Multiple graphic projects can be stored on a single µSD-Card. This feature is useful if you have many language versions of the same project or for demonstration purposes. Links to a different project folder can be defined with “PATH:UserFiles_xxx” syntax in the link field as shown on the right.



Point id: LINK

Value	Text
?	Load C3

[● Text]

[✓ Bold]

☐ Show point value

☐ Controller set value ?

Link: PATH:UserFiles_C3

The folder name should be “UserFiles_xxx” where xxx can be anything you want. The last character of the folder name is loaded into register 3013, so the Modbus master can detect which project is loaded. The FX-RP multiDISPLAY demo projects folder contains an example project folder named “multi language example” which will help you on your way.

Additionally, instead of using “PATH:UserFiles_xxx”, “LOCKPATH:UserFiles_xxx” can be used. The difference is that with “LOCKPATH”, the folders on the µSD card get renamed. The original “UserFiles” folder will be named “UserFiles_orig”, and the folder in the link will be renamed “UserFiles”. Note that because of this, you can only use “LOCKPATH” once with each µSD card.

The main use of the “LOCKPATH” feature is to be able to send out displays with µSD cards and only choose on site what program wil be loaded. If the selectable projects are compiled to fit in the internal memory of the FX-RP multiDISPLAY, selecting them will load the desired project into the memory, and the µSD card can then be removed, making this a very useful feature for situations where you don’t know in advance with what equipment the FX-RP multiDISPLAY is going to be used.

Uploading a project via Modbus

Graphical pages stored on the FX-RP multiDISPLAY’s internal memory or on the µSD can be updated through Modbus registers 65278 – 65343 by any Modbus master:

65278 - 65341		64 data registers
65342	statusCPU	status from substation
65343	statusDISP	status to substation

1)	To start a transfer, the master sets **statusCPU** register to **0xAAAA** (upload request)
2)	**statusDISP** receives **0x0000** from the display and the display will read "Loading data..." (ready for download)
3)	Modbus master sends the first 128 bytes of the bin file and sets **statusCPU** to **0x0001** (uploading part 1)
4)	**statusDISP** receives **0x0001** from the display (download part 1 completed)
5)	Modbus master sends the next 128 bytes of the bin file and sets **statusCPU** to **0x0002** (uploading part 2)
6)	**statusDISP** receives **0x0002** from the display (download part 2 completed)
…)	... (repeat until the file is sent)
n-1)	Modbus master sets **statusCPU** to **0xBBBB** to signal that file is sent (upload completed)
n)	**statusDISP** receives **0x2222** from the display (download completed)

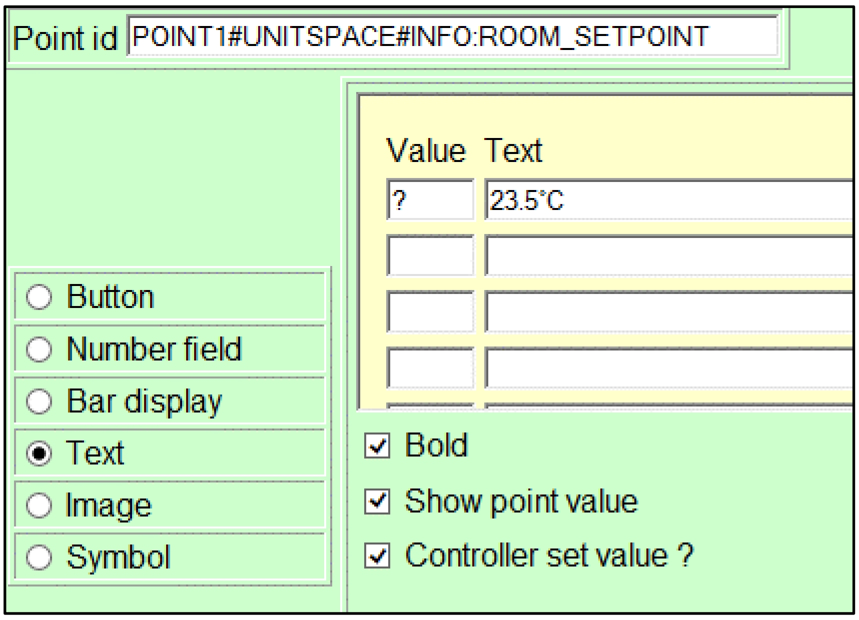
statusCPU	(written by master)
0xAAAA 0x00** 0xBBBB	Initiate a file transfer (requested upload) Counter for which part we are sending (uploading part x) Last part of the file was sent (upload completed)
statusDISP	(written by Display)
0x0000 0x00** 0x2222	ready to receive the file (ready for download) counter for which part was received completely (downloading part x) finished receiving the file (download completed)

Detailed programming

Introduction

The .htm pages that will constitute your FX-RP multiDISPLAY project consist of so called "**active elements**" and background images. Any element (text or image) you place inside the 320 by 240 pixels available space that doesn't have a PointID (= "Unknown") is considered and treated as background image.

To see the pointID of any element in the Fidelix Graphics Editor, either double click the element, or press CTRL+E.



Point id: POINT1#UNITSPACE#INFO:ROOM_SETPOINT

Value	Text
?	23.5°C

Button [✓ Text] Number field Bar display Image Symbol

☑ Bold
☑ Show point value
☑ Controller set value ?

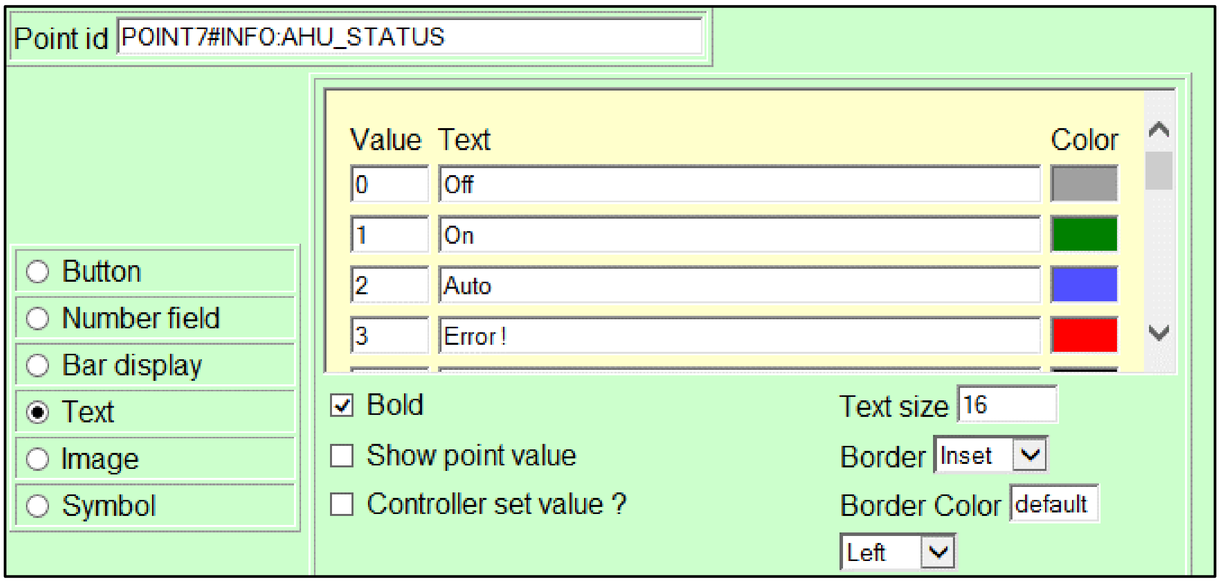
Any element that does have a PointID (keywords mentioned later, or freely chosen by yourself) will be placed on top of that generated background image.

When used as Modbus slave, each of the 250 available points will look something like the image on the right.

The text used in the yellow part is to 1) define the number of decimals and the unit displayed, 2) help to get the graphics correct during design.

Ticking "Show point value" will enable the displaying of the actual value of the point or register mentioned in the PointID field.

When not ticked, the middle column needs to contain the texts that will be shown, when the point has the corresponding value specified in the left column (see "Status texts").



Point id: POINT7#INFO:AHU_STATUS

Value	Text	Color
0	Off	⬜
1	On	🟩
2	Auto	🟦
3	Error !	🟥

Button [✓ Text] Number field Bar display Image Symbol

☑ Bold
☐ Show point value
☐ Controller set value ?

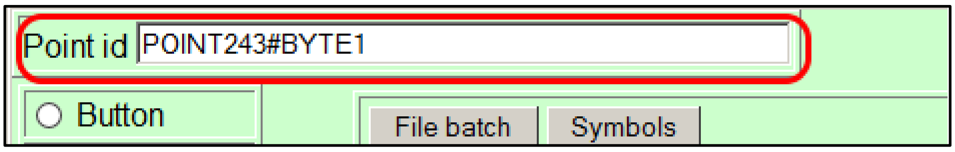
Text size: 16
Border: Inset
Border Color: default
Left

Many projects will have more than 1 page. A link can then be made to other pages (see "Links"). Only "text" or "Symbol" elements can be used to link to other pages by adding the name of the target page. In the "LINK" box of an active element. A link can only be used from an active element, so if your element has no other functionality, you can simply use "LINK" as PointID.

When used as Modbus master, the pointID will be used as the field to enter the correct parameters for each of the slave registers. The syntax is described later in this document.

Visualisation customisation

Any field of “text” or “Number field” type can be used to set values. In order to do so, “Controller set value” has to be selected in the EditPoint dialog box to make a point’s value editable from the FX-RP multiDISPLAY.
A number keypad will appear when the field is clicked, and the selected value is then attributed to the point.



Point id POINT243#BYTE1

Button

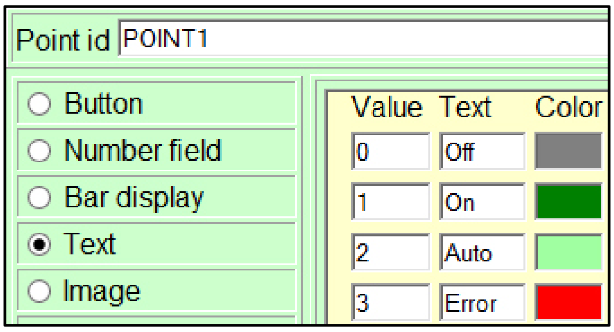
File batch | Symbols

The functionalities described hereunder are to be used by appending the according tag behind the “POINTxx” identifier in the pointID field of the EditPoint dialog box (as shown on the image). Features (and thus the ‘tags’) can be freely combined in any order.

NOTE: The content of any register can also be shown by using the point ID “REGxxxx”.

- You can choose to show the value of a point, or to display a text according to the point’s value. Showing the actual value is done by either selecting a “Number field” type element, or by ticking the “Show point value” tick box of a “Text” type element. Ticking the “Controller set value ?” tick box will generate a popup on the display when the value is clicked. Note that the user can now enter any desired value, so it is up to the connected controller to evaluate this user entered value.

  Showing text based on the value of a point (like for instance on/off, day/night or off / startup / slow / fast / error) is done by selecting a text type element and writing the desired values and corresponding texts in the yellow box inside the EditPoint dialog box.
  More details can be found in the “Status texts” section later in this document.

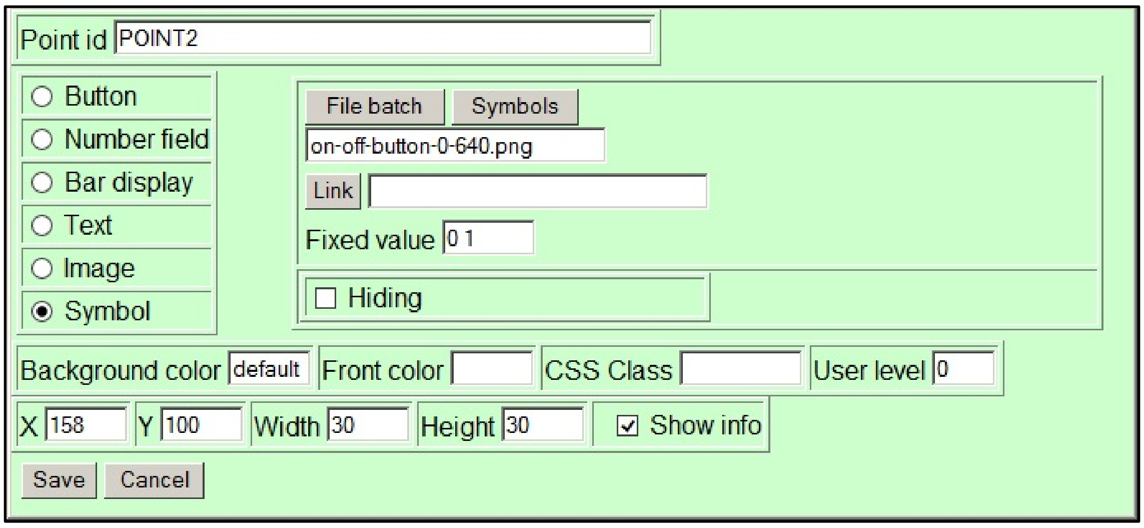


Point id POINT1

Button | Number field | Bar display | [✓ Text] | Image

Value	Text	Color
0	Off	⬛
1	On	🟩
2	Auto	🟩
3	Error	🟥

- To use a multi-stage image (like an “on/off” power button), enter the desired values through which to toggle separated by spaces in the “Fixed value” field of the EditPoint dialog box:



Point id POINT2

Button | Number field | Bar display | Text | Image | [✓ Symbol]

File batch | Symbols

on-off-button-0-640.png

Link

Fixed value 0 1

☐ Hiding

Background color default | Front color | CSS Class | User level 0

X 158 | Y 100 | Width 30 | Height 30 | ☑ Show info

Save | Cancel

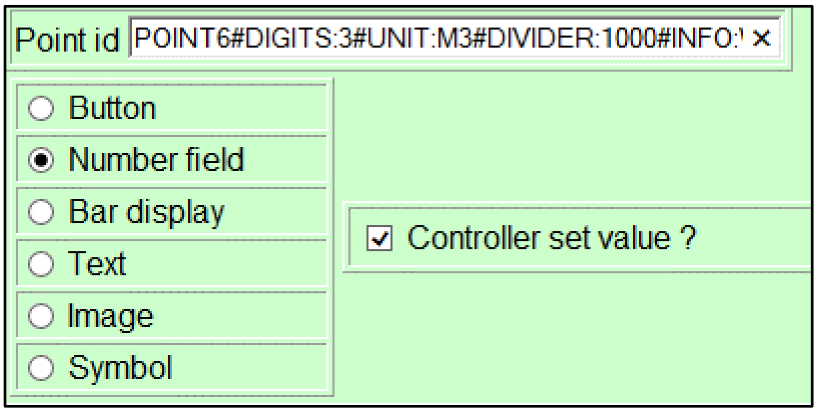
- #UNIT:xx where xx is the unit that is used to show the point. These units can either be programmed by the Modbus master (like it is done if the FX-RP multiDISPLAY is used with an FX controller or a multi24), or can be defined in the EditPoint dialog box in the graphics editor.
  For text type fields, the unit can also be defined in the “Text” field, in which case the unit is parsed from the end of the text. For example, “21.0°C” will define one decimal precision and degrees Celsius as unit.

  NOTE: A unit cannot be freely selected. The FX-RP multiDISPLAY firmware has a predefined list of units that can be used. If you need a unit that is not listed please contact us, and the needed unit can be added to our next firmware release. List of currently available units: "°C", "Pa", "bar", "V", "l/s", "m^3/h", "%", "m^3", "l", "mA", "Wh", "kWh", "MWh", "ppm", "K", "s", "min", "h", "Hz", "W", "kW", "MW", "Lx", "km/h", "°", "°/s", "l/h", "l/100km",

"%Rh", "ohm", "N", "kg", "ms", "hPa", "W/m²", "mm", "cm", "km", "m", "€", "€/kWh", "A", "°F", "CFM", "GPM", "%LIE", "%LEL", "%vol", "m³/s", "rpm", "m/s"

- #UNITSPACE can be added to put a space between the value and the unit. The default setting has the unit directly concatenated behind the point's value.
- #DIVIDER:xx where xx is used divider. 10 for 1 decimal, 100 for 2 decimals, 1000 for 3 decimals.

  REMINDER: For text type fields, the unit can also be defined in the "Text" field, in which case the divider is parsed from the end of the text. For example, "21.0°C" will define one decimal precision and degrees Celsius as unit.



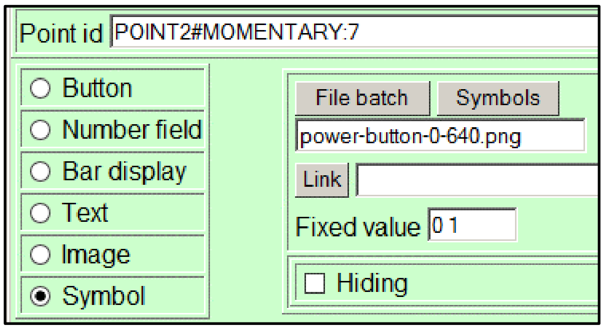
Point id POINT6#DIGITS:3#UNIT:M3#DIVIDER:1000#INFO:

( ) Button
(●) Number field
( ) Bar display
( ) Text
( ) Image
( ) Symbol

☑ Controller set value ?

- #DIGITS:xx, where xx is maximum 15. It defines the number of digits before the decimal sign. If the value is bigger than the number of digits specified, the full value will be displayed.
- #MIN:xx where xx defines the minimum allowed value for the point. It is useful if for example a set point's minimum value needs to be defined. Note that this value is before division, for example if one decimal is in use and you need to limit the minimum value to 10.0, use parameter "#MIN:100".
- #MAX:xx where xx defines maximum allowed value for the point. It is useful if for example set point's maximum value needs to be defined. Note that this value is before division. For example, if one decimal is in use and you need to limit the maximum value to 100.0, use parameter "#MAX:1000".
- #NOSCALE hides the low, middle and high values for a "Bar Display" point. You can use for example text objects to define the scale, or leave the scaling out completely. The hiding is particularly useful when your value has decimals, as the Low, Middle and High values are before division, for example if one decimal is in use and you need to limit minimum value to 10.0 use parameter "#MIN:100". The Middle value can be omitted, and even HAS to be omitted when using this combined with the "#NOEDIT" tag.
- #NOEDIT makes a Bar Display element only show a value, whereas by default, a bar display element can also be used as a slider to get a set value from the user. When this tag is used, the middle value of the Bar Display element needs to stay empty, as it is being used as container for the returned value, so #NOEDIT should always be used in combination with #NOSCALE.
- #INFO:xx, where xx is free text that will be added as a comment into the IEC code file that the converter generates. This will only make the IEC code file easier to read, there is no real functionality connected.
- #BYTE1, #BYTE2, #BYTE3, #BYTE4 gives access to the different bytes of the point's value. As each point has two registers for its value (see below for more detailed register structure), BYTE1 represents the least significant byte, BYTE4 the most significant. Values are displayed as decimal values from 0 to 255.
- #BIT0, …, #BIT31 gives direct access to each bit of the point's value. BIT0 represents the least significant bit, BIT31 the most significant.
- #BITGROUPxx:yy, where xx is the number of bits (2..6) you want to visualise and yy is the first bit of that group (0..30). The groups are allowed to overlap
  Example: POINT1#BITGROUP3:0 will show the three lowest bits of POINT1 (bit2, bit1 and bit0)
  Example: POINT2#BITGROUP5:14 will show five bits, beginning at bit 14 of POINT2 (bit18, bit17, bit16, bit15 and bit14)

- #MOMENTARY:xx can be used to send out 'impulses' of 'xx' seconds from a Modbus slave display. This should be used in combination with the 'fixed value' box in the EditPoint dialog box. Pushing the button in the example will set POINT2 to '1' for 7 seconds, after which the first value, '0' will be set back to POINT2. Be aware that there is no checking done if the master has received the impulse; the display merely sets the value for the defined time. This function is only available during slave mode operation.


Point id: POINT2#MOMENTARY:7

Button / Number field / Bar display / Text / Image / [✓ Symbol]

power-button-0-640.png

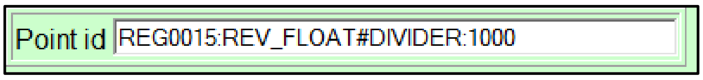
Link:

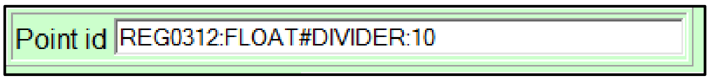
Fixed value: 0 1

☐ Hiding

IEEE-754 Floating Point format

Normal points on the display are represented as 32-bit values covering two consecutive registers. If the connected Modbus master is however sending out values as floating point values, the aforementioned "REGxxxx" syntax can be used with addition of ":FLOAT" or "REV_FLOAT", where "REV_FLOAT" is used when the registers are in reversed order. The register number in the pointID is the first of two consecutive registers used for the floating-point value.


Point id: REG0015:REV_FLOAT#DIVIDER:1000

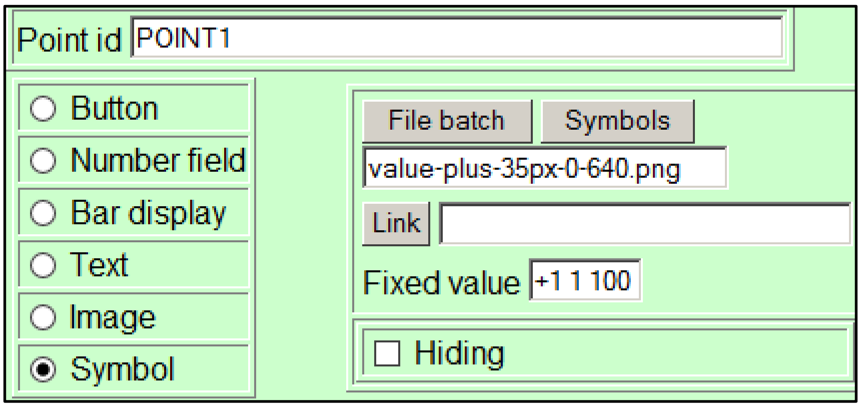

Point id: REG0312:FLOAT#DIVIDER:10

Pay close attention to the registers attributed to POINTS in the display; you can use a mix of POINTx and REGyyyy pointID's, but each POINT uses three registers (see the register structure later in this document).

NOTE: when using floating point values, the values shown are READ-ONLY and no other graphical elements can be attributed to it. So only the actual received value can be visualised.

Increment, Decrease, Minimum & Maximum

When a symbol or button is selected, this can be used to increment or decrease the value of a point. For this, the pointID just contains the point name (e.g. POINT145), and in the "Fixed value" field, 3 numbers, separated by spaces, define the incrementation or decrementation step, the minimum value and the maximum value a push on the symbol will trigger (e.g. writing "+1 0 100" means you will increment the point's value with steps of 1 to a maximum of 100). #MIN and #MAX in the pointID field have no influence here, and are ignored, so it best to omit them completely. Note however that the minimum and maximum value you define in the "fixed value" field must not only be adjusted to the point's #DIVIDER (so it is the value before division), but will also overrule any #MIN and #MAX value you define in the actual displaying of the point's value.


Point id: POINT1

Button / Number field / Bar display / Text / Image / [✓ Symbol]

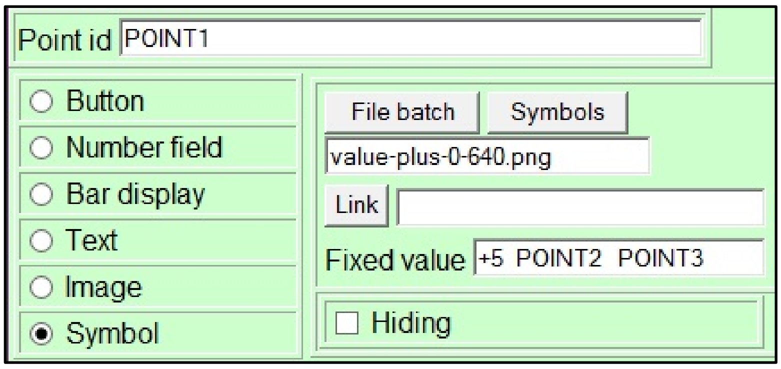
value-plus-35px-0-640.png

Link:

Fixed value: +1 1 100

☐ Hiding

Special symbols can be defined for points with #MIN and/or #MAX values defined in the pointID field of the EditPoint dialog box: the symbol with name formatted as "NameOfTheSymbol-min-640.gif" will be displayed when the point's value = #MIN, the symbol with name formatted as "NameOfTheSymbol-max-640.gif" will be displayed when the point's value = #MAX, the symbol with name formatted as "NameOfTheSymbol-0-640.gif" will be displayed when the point's value = zero (this image will be shown if #MIN or #MAX = zero), and the symbol with name formatted as "NameOfTheSymbol-mid-640.gif" will be displayed when the point has any value other than #MIN, #MAX or zero.



Point id POINT1

( ) Button
( ) Number field
( ) Bar display
( ) Text
( ) Image
[✓ Symbol]

File batch | Symbols

value-plus-0-640.png

Link

Fixed value +5 POINT2 POINT3

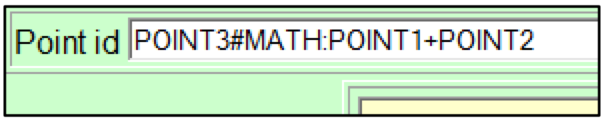
☐ Hiding

You can also use other points as minimum and maximum. In that case, both minimum and maximum need to be points, meaning it is not allowed to have a fixed upper limit and a variable lower limit. When using the minimum and maximum like that, make sure you don't allow the user to enter values that are not possible; often you'll want to set these dynamic minimum and maximum values from the Modbus master and not show them on screen at all.

Math operators

Display points can also contain a locally calculated value based on 2 other points. Know that all values are considered as integers, so divisions shown on the screen are not taken into account, and any value for a division smaller than "1", will give you the result of zero. You can use math operators for instance to calculate the local setpoint for a building-wide general, dynamic setpoint with local offset, or to let the end-user enter the water or energy price they pay, and, by simply sending the actual consumption to the screen, show the price that needs to be paid.

To use math operators, simply add a suffix to the pointID field:
POINTxx#MATH:POINTyy+POINTzz
POINTxx#MATH:POINTyy-POINTzz
POINTxx#MATH:POINTyy*POINTzz
POINTxx#MATH:POINTyy/POINTzz

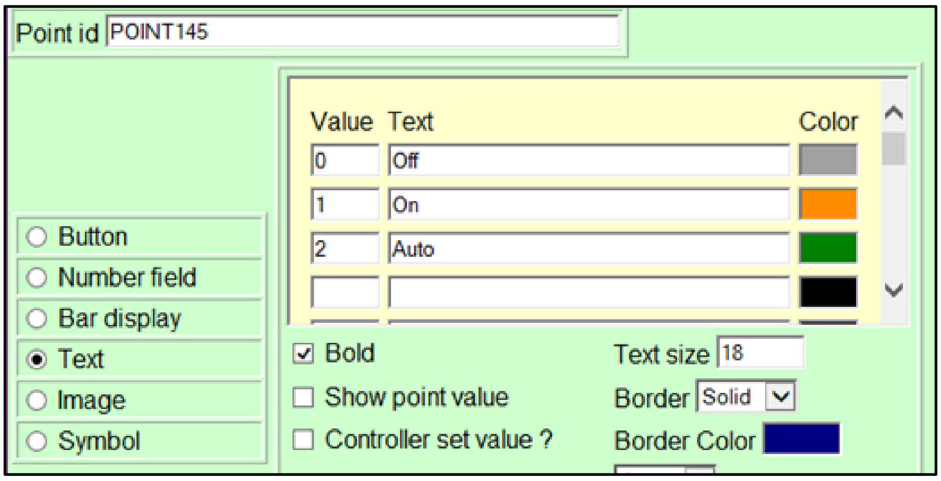


Point id POINT3#MATH:POINT1+POINT2

This function is only available during slave mode operation.

Status texts

By default, the number of different texts for fixed values (so called "Status Texts") is 20. These are texts like "On, Slow, Fast" that you can define in the yellow box of the EditPoint dialog box when editing a Text type element. This is useful when you for instance make a button with three statuses (0, 1 and 2) and you want to have a text changing accordingly to show the end user what status the button is currently in. Make sure you untick the "Show point value" tick box to show the entered texts.



Point id POINT145

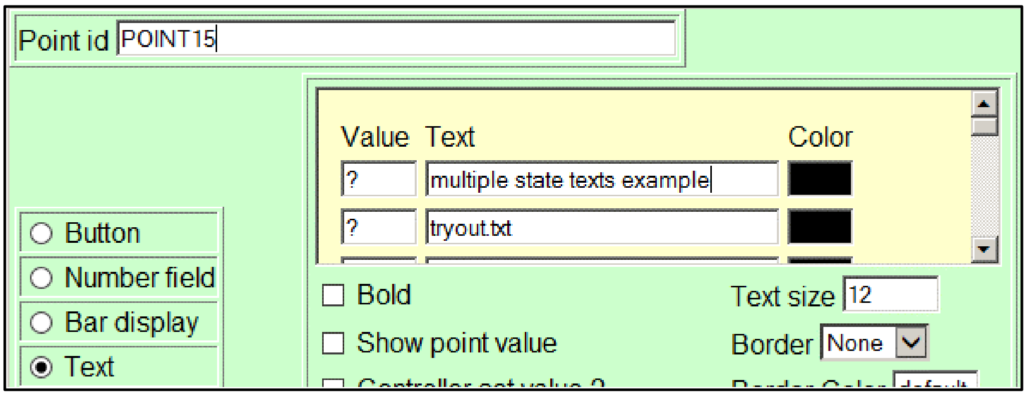
Value	Text	Color
0	Off	▨
1	On	🟧
2	Auto	🟩
		⬛

( ) Button
( ) Number field
( ) Bar display
[✓ Text]
( ) Image
( ) Symbol

☑ Bold
☐ Show point value
☐ Controller set value ?

Text size 18
Border Solid
Border Color 🟦

The Graphics editor limits the number of these statuses to 20, but you can add more by creating a file inside your project folder containing the texts for each status. The file is linked by writing its name in the second "Text" box like this:



Point id POINT15

Value	Text	Color
?	multiple state texts example	⬛
?	tryout.txt	⬛

( ) Button
( ) Number field
( ) Bar display
[✓ Text]

☐ Bold
☐ Show point value

Text size 12
Border None

The content of the linked .txt file should be formatted as follows:
0, value zero, #000000
1, first text, #000000
2, second text, green
3, third text, #000000
4, fourth text, #000000
…

where you first write the value, then the texts to show on the display and then the HTML colour code.

Status texts take up memory upon the loading of a page. All texts are loaded into memory when the page is loaded. That is why you can share status texts between points to save memory space. This is done by using 100000000 as the first status value for all the points with which you want to share the status texts. One of those points will then be stored into the memory, and all others only referenced. Make sure to copy all status texts into all points you want to use them in, as you cannot know of which point the status texts will be loaded into the FX-RP multiDISPLAY's memory.

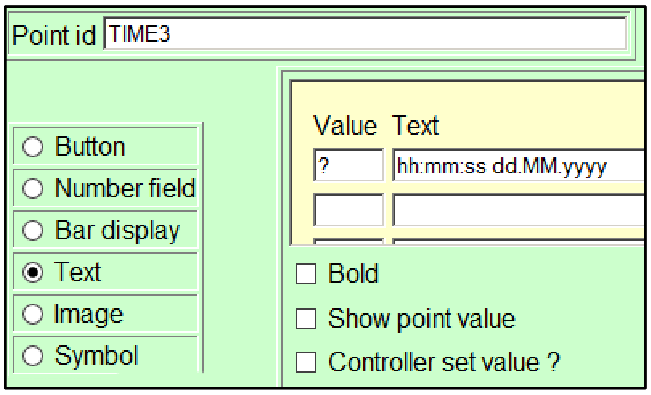
Using this feature is only necessary if you notice the page loads very slowly, or if you actually get an "out of memory" message when the page is loaded. It is possible to combine this feature with the previous one to have multiple points with more than 20 possible status texts, like, for example, when you are making a text input page.

Time and date, temperature, firmware version and Modbus address

Time, date, locally measured temperature and the display's firmware version and Modbus address can be displayed in various ways:

pointID	Further customisation in "Text" field of the "EditPoint" dialog box		Result
TIME	n/a		16:23:45
TIME2	n/a		16:23
TIME3	hh / HH	24h / 12h (**without** leading zero)	
	mm	minutes (always with leading zero)	
	ss	seconds (always with leading zero)	
	tt	AM / PM	
	dd	day (**without** leading zero)	
	MM	month (**without** leading zero)	
	yy / yyyy	Year with 2 or 4 digits	
	other characters	displayed as written (h, m, s, t, d, M or y not allowed)	
TIME4	hh / HH	24h / 12h (**with** leading zero)	
	mm	minutes (always with leading zero)	
	ss	seconds (always with leading zero)	
	tt	AM / PM	
	dd	day (**with** leading zero)	
	MM	month (**with** leading zero)	
	yy / yyyy	Year with 2 or 4 digits	
	other characters	displayed as written (h, m, s, t, d, M or y not allowed)	
DATE	n/a		03.09.2014
VERSION	n/a		1.32
TEMPERATURE	n/a		20.0°C
TEMPERATURE_C0	n/a		20°C
TEMPERATURE_C1	n/a		20.0°C
TEMPERATURE_C2	n/a		20.00°C
TEMPERATURE_F	n/a		68.0°F
TEMPERATURE_F0	n/a		68°F
TEMPERATURE_F1	n/a		68.0°F
TEMPERATURE_F2	n/a		68.00°F
MODBUS_ADDRESS	n/a		10

Example displaying date and time as "16:26:45 30.9.2014":


Point id TIME3

Value Text
? hh:mm:ss dd.MM.yyyy

Internal time schedules

The FX-RP multiDISPLAY has 5 internal time schedules, which can be accessed either from the settings page, or by making links to them yourself (see below, “Links”). Time schedules cannot be edited by an external Modbus master or slave, only from the local user interface. Time schedules can have any status from 0 to 15, though mostly only 0 and 1 are being used. You can of course attribute any action on your Modbus slave or master to any of the 16 available values.

NOTE: Time schedules hold their value, as long as no event is encountered that will change their value. This means that for instance, when you only select weekdays, but leave the time schedule at value 7 as your last entry on Friday, the time schedule will stay at 7 throughout the weekend.

You can change this behaviour and have the display reset the internal time schedules to zero at midnight for the unselected days by including a file named “timeschsetup.txt” into your project, containing a single line “SET_UNUSED_TIMESCH_DAYS_TO_ZERO”. Save this file in the root folder of your project and run it through the converter (version 1.39 or higher).

You can make your own graphics for the time schedules, to make them correspond better to the rest of the project layout and graphical style. For this, use following pointIDs:

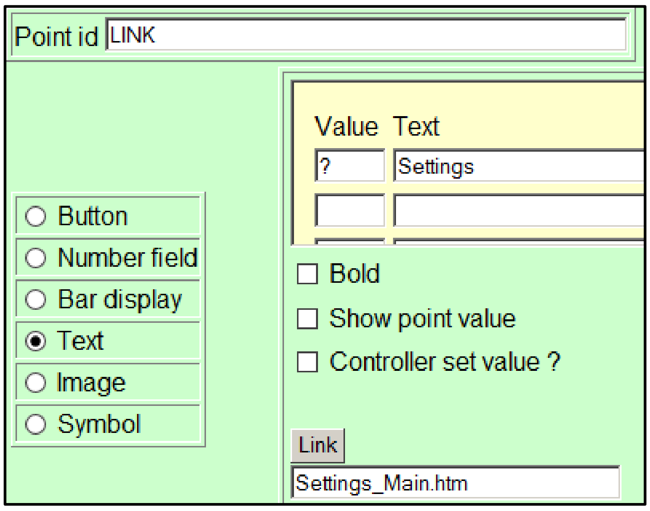
Day selection: 0 = not active 1 = active	TIMESCH1_MON TIMESCH1_TUE TIMESCH1_WED TIMESCH1_THU TIMESCH1_FRI TIMESCH1_SAT TIMESCH1_SUN	… … … … … … …	TIMESCH5_MON TIMESCH5_TUE TIMESCH5_WED TIMESCH5_THU TIMESCH5_FRI TIMESCH5_SAT TIMESCH5_SUN
Time schedule status: Any integer value from 0..15	TIMESCH1_STATE1 TIMESCH1_STATE5 TIMESCH1_STATE2 TIMESCH1_STATE6 TIMESCH1_STATE3 TIMESCH1_STATE7 TIMESCH1_STATE4 TIMESCH1_STATE8	… … … … … … … …	TIMESCH5_STATE1 TIMESCH5_STATE2 TIMESCH5_STATE3 TIMESCH5_STATE4 TIMESCH5_STATE5 TIMESCH5_STATE6 TIMESCH5_STATE7 TIMESCH5_STATE8
Time: Hours are in 24-hour format	TIMESCH1_HOUR1 TIMESCH1_MIN1 TIMESCH1_HOUR2 TIMESCH1_MIN2 TIMESCH1_HOUR3 TIMESCH1_MIN3 TIMESCH1_HOUR4 TIMESCH1_MIN4 TIMESCH1_HOUR5 TIMESCH1_MIN5 TIMESCH1_HOUR6 TIMESCH1_MIN6 TIMESCH1_HOUR7 TIMESCH1_MIN7 TIMESCH1_HOUR8 TIMESCH1_MIN8	… … … … … … … … … … … … … … … …	TIMESCH5_HOUR1 TIMESCH5_MIN1 TIMESCH5_HOUR2 TIMESCH5_MIN2 TIMESCH5_HOUR3 TIMESCH5_MIN3 TIMESCH5_HOUR4 TIMESCH5_MIN4 TIMESCH5_HOUR5 TIMESCH5_MIN5 TIMESCH5_HOUR6 TIMESCH5_MIN6 TIMESCH5_HOUR7 TIMESCH5_MIN7 TIMESCH5_HOUR8 TIMESCH5_MIN8

To show the current value of a time schedule on the screen, use REG3007 through to REG3011 as your pointID to show the value of time schedules 1 through to 5 (see display parameters).

Only this current value (or “output status”) of the time schedule is available over Modbus. All setup mentioned in the above table is only available on the local screen of the FX-RP multiDISPLAY.

Links

Links can be specified from any active point, by selecting the targeted page in the link field of an object. If there is no point attributed, but you just want to make a link, use "LINK" as the pointID. Note that some of the visualisation features may not work when adding a link to an element that also has other uses. (e.g. when using "REG3007" as pointID and a symbol to represent the value of the register, adding a link to the symbol, will disable the changing of the symbol.



Point id LINK

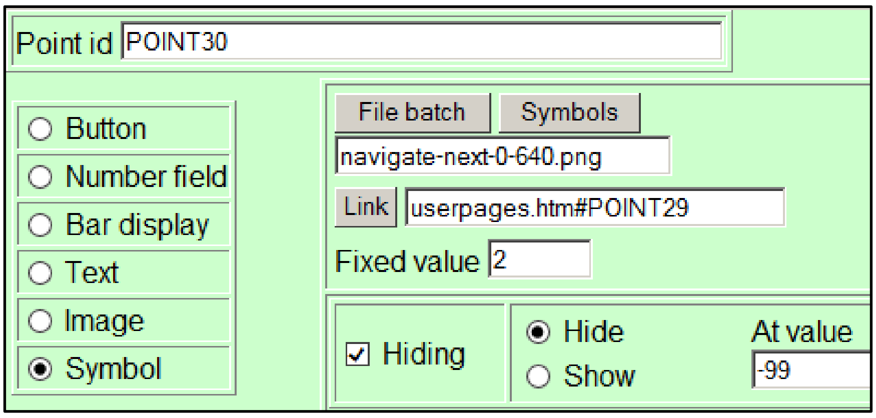
Value Text
? Settings

( ) Button
( ) Number field
( ) Bar display
(●) Text
( ) Image
( ) Symbol

☐ Bold
☐ Show point value
☐ Controller set value ?

Link: Settings_Main.htm

- If "CLOSE" is specified in the link field of an object, it will automatically link to start page of the project.
- If "TIMESCH1", "TIMESCH2", "TIMESCH3", "TIMESCH4" or "TIMESCH5" is specified in the link field of the object, the corresponding time schedule page will be opened.
- If "NAVIGATEBACK" is specified in the link field of the object, the link will go to previous page. Navigate back will list up to 4 previous pages. Pressing a "CLOSE" link will clear the navigation history.
- If "MODBUS" is specified in the link field of the object, a Modbus master status page is opened. Do not use this when the display operates in slave mode, as this will open a page with only zero values, since there is no Modbus master communication happening.
- If "CALIBRATION" is specified in the link field of the object, the calibration of the screen is triggered. After the calibration through this link, the display will go back to the page the link was triggered from, and NOT go into the settings page (like it does when you trigger the calibration by pressing in the same spot on the display for 10 seconds).
- If "DISPLAY_SETTINGS" is specified in the link field of the object, the link will go to settings page immediately (a page otherwise brought up by long pressing on the same place on the multiDISPLAY).
- Dynamic links can be created by using the value of a display POINT to select the page to navigate to.

  Point id POINT30

  ( ) Button
  ( ) Number field
  ( ) Bar display
  ( ) Text
  ( ) Image
  (●) Symbol

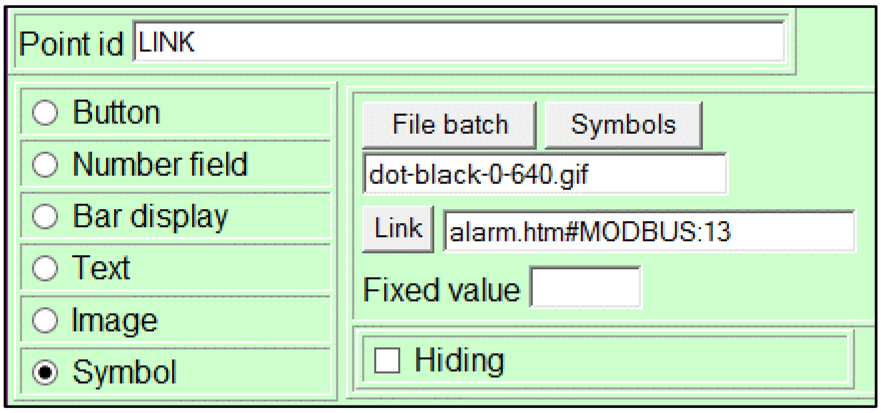
  File batch | Symbols
  navigate-next-0-640.png
  Link: userpages.htm#POINT29
  Fixed value 2

  ☑ Hiding
  (●) Hide
  ( ) Show
  At value -99

  Writing "#POINTxx" after the page name in the link field will direct the user to the specified page_Value-Of-The-Point.htm (e.g. when the value of POINT29 = 5, a link made by using "linkedpage.htm#POINT29" will navigate the user to linkedpage_5.htm). The converter will look for all pages that might be targetable in your project folder and include them. If the used POINT has a value for which there is no page available, a standard white error page with "Invalid file: M24_pagename.bin" message will be displayed. Upon closing that error page, the whole project will be reloaded and the user thus redirected to the first page of the project. It is therefore VERY IMPORTANT you don't make the points used for dynamic linking freely changeable, and you are very careful in programming your external Modbus master to only write values for which there are pages into the displaypoint you are using for the dynamic links. Having display native error messages pop up is obviously not advisable for end-user applications. This function is only available during slave mode operation.
- Writing "::xx" after the page name in the link field (e.g. Startpage.htm::120) will trigger a timer upon the loading of the page after which the link is followed automatically. "xx" is the number of seconds to wait before changing the page.

- If “#PASSWD:xx” is suffixed to the pointID from which the link is called (whether this be an actual display point or just “LINK”), a popup is shown before the link is followed. The password can be a value between 1 and 65535, but is a fixed value, and can only be changed in the graphics editor, before conversion. Entering a wrong password will keep the “Enter Password” dialog box open with a message: “Incorrect Password!”, upon which the user can try again. To close the dialog box, press the “C” button.

- Writing “#MODBUS:xx” after the page name in the link field allows following that link by setting Modbus register 3044 or 3046 from the Modbus master. xx is the value used for the corresponding link, each link should have a separate value.
  For example, the link field can contain “alarm.htm#MODBUS:13”, which means that the alarm.htm page will be opened if “13” is written to “open page register” 3044 or 3046.



Point id: LINK

Button / Number field / Bar display / Text / Image / [✓ Symbol]

File batch | Symbols

dot-black-0-640.gif

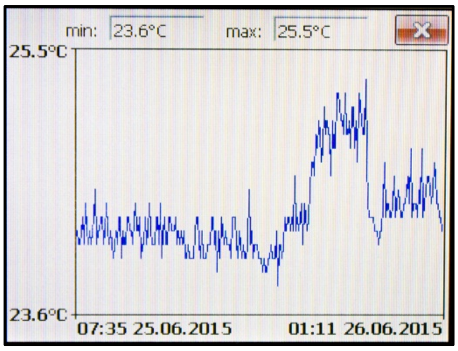
Link: alarm.htm#MODBUS:13

Fixed value:

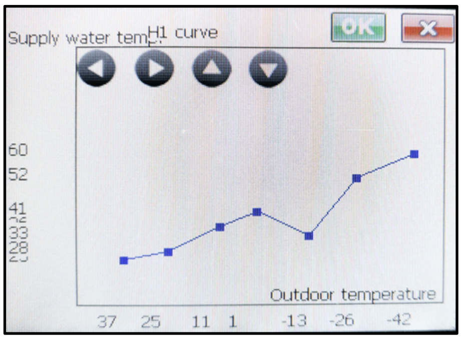
☐ Hiding

  This feature is used to “force” pages to appear. The links will most likely be hidden from the user, by putting them in a small “invisible” box in a corner of the screen (10px X 10px in the background colour) or by putting them on a (password protected) “settings page” in your project, and are thus only activated from the Modbus master side when user action is required. The pages can be “force shown” at any given time, meaning the user doesn’t have to be on the page containing the link for the linked page to “pop up” when instructed so by the Modbus master. This function is only available during slave mode operation.

- If “#HISTORY” is suffixed to the pointID from which the link is called (in this case, the linked must be called from one of the 250 points), the history graph window is opened when the link is clicked, and the latest values (history, trend) is requested from the Modbus master. The history functionality is explained later in this document.
  This function is only available during slave mode operation.



- If “#LUT” is suffixed to the pointID from which the link is called (in this case, the linked must be called from one of the 250 points), a graphical conversion table (look-up table) is opened and data is requested from the Modbus master device. The Modbus master has to send the conversion table data for the requested point. The look-up table functionality is explained later in this document.
  This function is only available during slave mode operation.



Strings / texts

It is possible to define a text object that loads a string from certain registers by selecting the “Show Point Value” selection box and entering “STRING” as a text. String selection is done by changing the object value. The object’s value can be preselected using format “STRING_XXYY” where XX is the string section number and YY is the selected string. See the section “String Variables” later in this document for more details.

Entrance control user panel

If “PIN” is specified in the link field of the object, PIN-Code entering dialog will be opened. This function is useable only if DU-10 mode is activated (RFID reader mounted).

The FX-RP multiDISPLAY can be equipped with a RFID reader which activates DU-10 mode. If DU-10 mode is activated the DISPLAY will communicate also as a DU-10 device using next Modbus address (If MULTI DISPLAY has address 10, DU-10 will have address 11). If DU-10 mode is activated the FX-RP multiDISPLAY can be used as an intruder alarm user panel.

Customising keypad buttons

Keypad buttons can be customised to make them match the project design. Custom buttons have to be exactly 65x35 pixels, except the OK button which has to be 135x35 pixels. Buttons should be saved as .png files and located in a subfolder called “Buttons” in your project folder. The converter will add the custom buttons automatically if files are found from Buttons folder. The file names indicate the function of the button and the following names should be used:

FdxButton0.png
FdxButton1.png
FdxButton2.png
FdxButton3.png
FdxButton4.png
FdxButton5.png
FdxButton6.png
FdxButton7.png
FdxButton8.png
FdxButton9.png
FdxButtonBackSpace.png
FdxButtonPlusMinus.png
FdxButtonCancel.png
FdxButtonDot.png
FdxButtonOk.png

The multiDISPLAY demo projects folder contains an example project folder named “Keypad example” that will help you on your way.

Extended UTF-8-character support

The FX-RP multiDISPLAY supports UTF-8 characters, but only Latin characters are loaded by default. The default character set includes following characters: “ ! \ " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\ \ ] ^ _ ` a b c d e f g h I j k I m n o p q r s t u v w x y z { | } ~ ä ö å Ä Ö Å ° ”. If you need characters that are not included by default you can define a custom character set by adding a “CharacterSet.txt” file to you project folder. Type all the required characters to CharacterSet.txt file, including also the default characters if those are needed (save the file using UTF-8 format). Note that every character takes some memory, so it is not recommended to define more characters than you need.

The multiDISPLAY demo projects folder contains an example project folder named “multi language example” inside which the Chinese, Finnish and Russian project folders each contain an example file that will help you on your way.

Modbus master functionality

To use the FX-RP multiDISPLAY as a Modbus master, the correct type of pointID must be used in the graphics editor. Point names starting with "MODBUS" are detected as Modbus master definition. The pointID should also contain fields for Modbus address, register number, Modbus device type and register format. A pointID containing all needed information looks something like this:

"MODBUS:ADD=1:REG=0:TYPE=INPUT:FORMAT=INT16".

field	**description**	**options**	
ADD	Modbus device address	value between **1-247**	
REG	Register number (starting from 0)	value between **0-65535**	
TYPE	Modbus device type	**INPUT** **HOLDING** **SINGLEREG** **DISCRETE** **COIL**	read-only register read-write register write single register read-only Boolean read-write Boolean
FORMAT	Register format defines how the value is presented in registers	**INT16** **UINT16** **INT32** **UINT32** **REV_INT32** **REV_UINT32** **FLOAT** **REV_FLOAT** **BITMASK;xxxx**	signed single register unsigned single register signed dual register unsigned dual register signed dual register with reversed register order unsigned dual register with reversed register order IEEE 754 dual register single precision floating point IEEE 754 dual register single precision floating point with reversed register order bitmask operation for register, bitmask is presented in hex format and it can contain one or more bits set Example: BITMASK;0020 means the 6th bit

In addition to these settings some serial settings and other communication related settings are needed. Those are defined in a text file named "ModbusMasterSettings.txt" which should be located in your project folder. If the file is not found or the definitions file is invalid, default settings are used (57600, 8N1, 50ms send delay, 100ms timeout, AUTO generated combined modbusdevices). There are example projects with the correct syntax used for the Modbus master feature available in the Demo projects folder. The settings in the file are explained in the following table. The term "modbusdevice" is used as the name for a zone of registers towards which a communication socket is opened by the Modbus master display.

field	**Options (or example)**	**description**
BAUD	9600 19200 38400 57600 115200	communication baud rate
BITS	8n1 8e1 8o1 8n2 8e2 8o2	data bits, parity and stop bits
SENDDELAY	numeric value	Send delay before new request in milliseconds. The timer is started when a reply has been received or when the timeout timer has ran out.

field	**Options (or example)**	**description**
MODBUSDEVICES	AUTO MANUAL	Defines if modbusdevices are generated automatically or manually. If AUTO is selected, defined modbusdevices are generated by the information given in point names. Also modbusdevices needed to write registers defined by SENDREGISTERS will be created. However, if the read registers from a slave are not available on the UI of the master display, make sure you still manually add the corresponding modbusdevice for those registers.
COMBINE	FALSE or numeric value	Affects only if AUTO mode defined. Defines if Modbus registers should be combined as a one larger modbusdevices containing multiple registers. This can speed up the communication by reducing the number of different Modbus messages. Maximum number of registers to combine can be defined here (< 100), or if FALSE then registers are not combined.
TIMEOUT	numeric value	Affects only if AUTO mode defined. Defines timeout for auto generated modbusdevices. Same timeout is used for every device. Makes sure not to select a value that is too small, especially during development!
MODBUSDEVICE	ADDRESS:1, STARTREG:0, COUNT:1, TYPE:COIL, TIMEOUT:180	Affects only if MANUAL mode defined. Modbusdevices are defined by using the same format as for point names. Multiple modbusdevices can be defined. Also needed for reading values from one slave and writing them to a second, while the read values are not shown on the multiDISPLAY. In this case, it can be used in combination with MODBUSDEVICES=AUTO.
MODBUSDEVICE	ADDRESS:10, READREG:33, WRITEREG:4, COUNT:1, TYPE:HOLDING, TIMEOUT:250	Affects only if MANUAL mode defined. Use this to make a field that shows the value of register 33, but that writes to register 4. The slave will most likely handle the writing of the value you write into register 4 into register 33. Don't forget to manually create the modbusdevice for register 4, as this will only generate the device for register 33.
SENDREGISTERS	ADDRESS:1, WRITEREG:9057, FIXEDVALUE:1, COUNT:1, TIMEOUT:500	Send a fixed value to a certain register on a slave device (for instance for a "communication is active" bit detection that will be set to 0 again by the slave). **COUNT as a number uses Modbus code 16, COUNT:SINGLE uses Modbus code 06.**
SENDREGISTERS	ADDRESS:10, WRITEREG:11, COUNT:1, FIXEDVALUE:45, TIMEOUT:220, SENDIF:TIMESCH1=12	Send a fixed value to a certain register on a slave device, based on the value of an internal time schedule on the display.
SENDREGISTERS	ADDRESS:8, WRITEREG:125, SOURCEREG:3007, COUNT:5, TIMEOUT:300	Send the display's internal registers to a slave. The example here will send the status of the five internal time schedules to registers 0-4 of slave at address 8
SENDREGISTERS	ADDRESS:4, WRITEREG:6, SOURCEREG: 12.2458.HOLDING, COUNT:1, TIMEOUT:300	Send the register value from one slave to another register on another slave. Don't forget to define the slave address that is being read as MODBUSDEVICE, if it is not visible on the display's interface, even when MODBUSDEVICES = AUTO is selected! Target registers are always Holding registers.

The number of Modbusdevices is not a fixed limit as such, but there is limited amount of memory reserved for the definition of Modbus communication sockets. The available memory for defining communication sockets (Modbusdevices) is 10240 bytes (4644 for V1 displays). Each Modbusdevice can have 99 registers maximum. Each separate Modbusdevice takes up 40 bytes of memory + 2 bytes per register in its definition.

So; a Modbusdevice that consists of 15 registers will use 70 bytes. Similarly, when defining Modbusdevices per 1 register, they will each use 42 bytes, making the maximum number of Modbusdevices you can define 243 (or 110 for V1 displays).

It thus depends on the setup how many registers or register ranges (Modbusdevices) you can read out on one FX-RP multiDISPLAY. If, while using the "MODBUSDEVICES=AUTO" setting, the available memory is insufficient, you can try to define the devices manually.

NOTE: Copying registers from 1 slave to another always requires 2 independent Modbusdevices; 1 register section that is being read and 1 register section that is being written. This needs to be considered when calculating the available memory for Modbusdevices.

NOTE: The converter doesn't do any checking of the number of Modbus communication sockets defined, so make sure that, when using a lot of Modbusdevices, you calculate the actual memory used.

The multiDISPLAY demo projects folder contains an example project folder named "Modbus master example" which will help you on your way. Also, the folders "text editing example", "basic example" and "time editing example" contain example files for the master functionality.

An example of each of the features available in the "ModbusMasterSettings.txt" file and the correct syntax of how to use them:

```
:: This is an example file to be used as reference to create your own custom file
:: Note that comment lines start with "::"
:: Comments are always a full line; comments behind parameters render this file invalid


:: == Set Modbus master baud rate
:: BAUD=9600
:: BAUD=19200
:: BAUD=38400
BAUD=57600
:: BAUD=115200

:: == Set bit count, parity and number of stop bits
BITS=8n1
:: BITS=8e1
:: BITS=8o1
:: BITS=8n2
:: BITS=8e2
:: BITS=8o2


:: == Set Modbus send delay in milliseconds (a delay before next Modbus message is sent)
:: == The delay timer starts running after a reply has been received
:: == or the timeout timer has ran out
SENDDELAY=500


:: == Select if modbusdevices (communication sockets) should be made automatically or manually
MODBUSDEVICES=AUTO
:: MODBUSDEVICES=MANUAL


:: == If MODBUSDEVICES=MANUAL is selected, these settings do not have any effect
:: == If MODBUSDEVICES=AUTO is selected, then you can select if Modbusdevices should be
:: == combined making one larger Modbusdevice with multiple registers
:: == set to FALSE if combining is not allowed, or enter the maximum number of registers that should
:: == be combined per query
:: COMBINE=FALSE
COMBINE=10
```

```
:: == Set Modbus timeout in milliseconds (only for autogenerated Modbusdevices)
TIMEOUT=250


:: == If MODBUSDEVICES=MANUAL selected Modbusdevices have to be defined manually
:: MODBUSDEVICE=ADDRESS:5,  STARTREG:0,    COUNT:1,  TYPE:HOLDING,  TIMEOUT:100
:: MODBUSDEVICE=ADDRESS:9,  STARTREG:2,    COUNT:10, TYPE:HOLDING,  TIMEOUT:120
:: MODBUSDEVICE=ADDRESS:33, STARTREG:0,    COUNT:1,  TYPE:INPUT,    TIMEOUT:100
:: MODBUSDEVICE=ADDRESS:12, STARTREG:0,    COUNT:1,  TYPE:DISCRETE, TIMEOUT:100
:: MODBUSDEVICE=ADDRESS:1,  STARTREG:2,    COUNT:1,  TYPE:COIL,     TIMEOUT:100


:: == Use this to make an element that shows the value of register 33, but writes into register 4
:: == this will generate a Modbusdevice for register 33,
:: == so don't forget to make sure you then have the "write" register also defined as Modbusdevice
:: MODBUSDEVICE=ADDRESS:10, READREG:33, WRITEREG:4, COUNT:1,  TYPE:HOLDING, TIMEOUT:100
:: MODBUSDEVICE=ADDRESS:10, STARTREG:4, COUNT:1, TYPE:HOLDING, TIMEOUT:120


:: == Send a fixed value to a slave register
:: SENDREGISTERS=ADDRESS:10, WRITEREG:29, COUNT:1,      FIXEDVALUE:1,    TIMEOUT:100,
:: SENDREGISTERS=ADDRESS:1,  WRITEREG:6,  COUNT:SINGLE, FIXEDVALUE:2048, TIMEOUT:200


:: == Send a fixed value to a slave register,
:: == based on the value of an internal time schedule of the master display
:: SENDREGISTERS=ADDRESS:10, WRITEREG:11, COUNT:1, FIXEDVALUE:3,  TIMEOUT:100, SENDIF:TIMESCH1=0
:: SENDREGISTERS=ADDRESS:10, WRITEREG:11, COUNT:1, FIXEDVALUE:6,  TIMEOUT:100, SENDIF:TIMESCH1=1
:: SENDREGISTERS=ADDRESS:10, WRITEREG:11, COUNT:1, FIXEDVALUE:9,  TIMEOUT:100, SENDIF:TIMESCH1=2
:: ...
:: SENDREGISTERS=ADDRESS:10, WRITEREG:11, COUNT:1, FIXEDVALUE:45, TIMEOUT:100, SENDIF:TIMESCH1=14
:: SENDREGISTERS=ADDRESS:10, WRITEREG:11, COUNT:1, FIXEDVALUE:48, TIMEOUT:100, SENDIF:TIMESCH1=15


:: == Send internal registers (such as time schedules, temperature, …) to slave(s)
:: == local TE
:: SENDREGISTERS=ADDRESS:10, WRITEREG:1002, COUNT:1,      SOURCEREG:3000, TIMEOUT:150
:: SENDREGISTERS=ADDRESS:10, WRITEREG:2,    COUNT:1,      SOURCEREG:3001, TIMEOUT:300
:: SENDREGISTERS=ADDRESS:10, WRITEREG:5,    COUNT:1,      SOURCEREG:3002, TIMEOUT:300
:: SENDREGISTERS=ADDRESS:10, WRITEREG:8,    COUNT:1,      SOURCEREG:3003, TIMEOUT:300
:: SENDREGISTERS=ADDRESS:11, WRITEREG:2000, COUNT:1,      SOURCEREG:3004, TIMEOUT:300
:: SENDREGISTERS=ADDRESS:29, WRITEREG:456,  COUNT:5,      SOURCEREG:3007, TIMEOUT:300
:: == local minute (0-59)
:: SENDREGISTERS=ADDRESS:12, WRITEREG:1044, COUNT:1,      SOURCEREG:3005, TIMEOUT:300
:: == local Time Schedule 1 value
:: SENDREGISTERS=ADDRESS:13, WRITEREG:1044, COUNT:1,      SOURCEREG:3007, TIMEOUT:300
:: == local Time Schedule 2 value
:: SENDREGISTERS=ADDRESS:10, WRITEREG:1008, COUNT:1,      SOURCEREG:3008, TIMEOUT:150
:: using "COUNT:SINGLE" will use Modbus function code 06 instead of 16 for writing to the slave
:: SENDREGISTERS=ADDRESS:2,  WRITEREG:12,   COUNT:SINGLE, SOURCEREG:3000, TIMEOUT:200


:: == Send the value of a register from one slave to another.
:: == Remember to add the read-registers to the Modbusdevice list
:: == (even when you are using the MODBUSDEVICES=AUTO setting)
:: == if you don't have those registers visible on the display.
:: == Only the write-registers are automatically created as Modbusdevice.
:: == When using the MODBUSDEVICES=MANUAL parameter,
:: == remember to create both the read and write registers as Modbusdevices.
:: == writing can only be done into Holding registers.
:: == If no type is specified, Holding register is assumed.
:: SENDREGISTERS=ADDRESS:20, WRITEREG:17,   COUNT:1,      SOURCEREG:1.2.COIL,          TIMEOUT:250
:: SENDREGISTERS=ADDRESS:20, WRITEREG:480,  COUNT:7,      SOURCEREG:3.20480.HOLDING,   TIMEOUT:250
:: SENDREGISTERS=ADDRESS:10, WRITEREG:1017, COUNT:1,      SOURCEREG:11.1017,           TIMEOUT:150
:: SENDREGISTERS=ADDRESS:10, WRITEREG:1020, COUNT:1,      SOURCEREG:11.1020.DISCRETE,  TIMEOUT:150
:: SENDREGISTERS=ADDRESS:10, WRITEREG:1023, COUNT:1,      SOURCEREG:11.1023.INPUT,     TIMEOUT:150
:: SENDREGISTERS=ADDRESS:11, WRITEREG:2,    COUNT:SINGLE, SOURCEREG:2.0,               TIMEOUT:200
```

Note that the FX-RP multiDISPLAY -obviously- cannot operate as a Modbus master and slave at the same time. However, a display acting as a Modbus master can be changed to be a Modbus slave by inserting a µSD-Card containing a slave project and rebooting the display (= taking power off).

Complete Display register structure overview

All internal registers used by the FX-RP multiDISPLAY are Holding registers.

Register sections

start	end	function
0	749	Input data
1000	1749	Output data
2000	2309	Trend
2310	2373	Graphical look-up table editor
2400	2911	String data
3000	3063	Special functions
65278	65343	Data download

Input data (input from an external Modbus master)

The external Modbus Master writes to these slave display registers:

POINT1	Reg00	Parameters, Divider, Unit
	Reg01	Value (16 highest bits)
	Reg02	Value (16 lowest bits)
POINT2	Reg03	
	Reg04	
	Reg05	
POINT3	Reg06	
	Reg07	
	Reg08	
…		
POINT250	Reg747	Parameters, Divider, Unit
	Reg748	Value (16 highest bits)
	Reg749	Value (16 lowest bits)

The “Parameters, Divider, Unit” register is divided into two:

Parameters & Divider (most significant byte of the “Parameters, Divider, Unit” register), and Unit (least significant byte of the “Parameters, Divider, Unit” register).

For the Parameters & Divider part, these are the meaning of bits from most significant to least significant:

8:	reset manual override	
7:	do not show value set from display	
6:	reserved	
5:	reserved	
4-1:	divider for the value (# of decimals)	
	0000	1 (0)
	0001	10 (1)
	0010	100 (2)
	0011	1000 (3)

The unit part contains the chosen value as follows:

0	no unit	15	K	30	ohm	45	g/kg
1	°C	16	s	31	N	46	°F
2	Pa	17	min	32	kg	47	CFM
3	bar	18	h	33	ms	48	GPM
4	V	19	Hz	34	hPa	49	%LIE
5	l/s	20	W	35	W/m^2	50	%LEL
6	m3/h	21	kW	36	mm	51	%vol
7	%	22	MW	37	cm	52	m^3/s
8	m3	23	Lx	38	km	53	rpm
9	l	24	km/h	39	m	54	m/s
10	mA	25	°	40	€		
11	Wh	26	°/s	41	€/kWh		
12	kWh	27	l/h	42	CFM		
13	MWh	28	l/100km	43	GPM		
14	ppm	29	%Rh	44	g/m^3		

The "Parameters, Divider, Unit" register contains 0x0F99 (3993) if the register is not set by the Modbus master.

Output data (output read by an external Modbus master)

The external Modbus master reads from these slave display registers:

POINT1	Reg1000	Info
	Reg1001	Value (16 highest bits)
	Reg1002	Value (16 lowest bits)
POINT2	Reg1003	
	Reg1004	
	Reg1005	
POINT3	Reg1006	
	Reg1007	
	Reg1008	
…		
POINT250	Reg1747	Info
	Reg1748	Value (16 highest bits)
	Reg1749	Value (16 lowest bits)

The "Info" register contains 0 if the value is not set from the display and 1 if the value set from the display. The FX-RP multiDISPLAY resets this back to 0 if the value is equal in both input and output register sections. This register also works as a "lock" to the point value registers, meaning that as long as the info register contains "1", the local value will be shown, and the value the Modbus master may be writing into registers 1+2, 4+5, … is ignored, unless of course you are setting the seventh bit of the "Parameters, Divider, Unit" register.

How to work with input/output registers

Each display point (POINT1, POINT2, ..., POINT250) has 6 registers attributed to it (3 input and 3 output).

When making graphics, you can assign a unit and a divider (the number of decimals) to each point. You can also set these attributes from your master, if that is desirable, into register 0, 3, ..., 747. This feature is currently no longer frequently used. Its original purpose (during development there were only 50 display points) was to set the point type dynamically; having only 1 page, "looping through" different values. Currently that register is set at 0F99x16 (3993) if it has not been written over by the Modbus master.

Each point has an "info" register (1000, 1003, ..., 1747) which indicates if the point's value has been set from the display (=1). This is useful when you allow a user to enter a value (you ticked the "controller set value" box in the EditPoint dialog box in the graphics editor) and you have not defined #MIN and/or #MAX on the graphics, but want to validate the input, or simply monitor this register to detect changes in setpoints. In this case you can use the "Parameters, Divider, Unit" registers (0, 3, ..., 747) to initially hide the user input value, validate it in your master and only then write it to the display (bits 7 and 8 of the most significant byte, see Input Data).

A value set from the display will be saved into registers 1001+1002, 1004+1005, ..., 1748+1749. At the same time, it will set register 1000, 1003, ..., 1747 to "1", indicating a point's value has been changed locally. If the master reads these values, it can "confirm" them by writing the same value into registers 1+2, 4+5, ..., 748+749. Once that has been done, the display will set the info register (1000, 1003, ..., 1747) back to 0, again notifying the Modbus master that all points and values have been properly synchronized.

A 4-and-a-half-minute video is available on Youtube to clarify this process: https://youtu.be/tEushV8ugsA

This rather complex process is however not always necessary. Actually, in most cases, we don't use it when not working with the Fidelix multi24 controller (where all of these functionalities are embedded into two functions GetDisplayPointF and SetDisplayPointF). Using a third party controller, the easiest way is to just write directly into registers 1001+1002, 1004+1005, ..., 1748+1749, overwriting any value from the display when the point value is editable from both the multiDISPLAY and the Modbus master.

TL;DR: In most cases, you will have a very distinct separation between the points you want to read and the ones you want to write and only select "controller set value" for those points you will set from the multiDISPLAY. Read those from registers 1001+1002, 1004+1005 etc and write "read only" values on the multiDISPLAY that are being written by the Modbus master into registers 1+2, 4+5 etc.

Trends (history)

The FX-RP multiDISPLAY can show trends with up to 300 points. The data is requested by the FX-RP multiDISPLAY from the Modbus master. The data is presented as 16bit signed integers, the divider is taken from the point definition. Instead of polling registers 2000 and 2001, the Modbus master may poll register 3012 which also contains information about requested trend.

Reg2000	0=Trend ready (cleared by Modbus master), 1=Trend request (set by display)
Reg2001	Point number of requested point
Reg2002	Minimum value (used for y-axis scaling)
Reg2003	Maximum value (used for y-axis scaling)
Reg2004	Sample interval (seconds, used for x-axis scaling)
Reg2005	Number of points (max 300)
Reg2006	Last unsaved measurement
Reg2007	seconds from last unsaved measurement (0=not used)
Reg2008	Update interval (seconds, 0=not in use)
Reg2009	reserved
Reg2010	Trend data start (300 registers)
Reg2011	
…	
Reg2309	Trend data end

Graphical look-up table editor

The FX-RP multiDISPLAY can handle graphical look-up table editing with up to 10 points. The actual look-up table must be implemented inside the Modbus master. Use the #LUT definition as a part of point name in the graphics editor to use the look-up table function. The LUT operation is started by the FX-RP multiDISPLAY if a point with #LUT definition is clicked. The FX-RP multiDISPLAY writes the point number of the requested LUT to register 2310. The Modbus master needs to poll register 2310 to notice the LUT request. After the request, the Modbus master should write the correct data to registers 2311-2339 and set register 2310 to “0xAAAA” indicating that the data is uploaded to the FX-RP multiDISPLAY. After the user has finished the modification of the LUT points, the FX-RP multiDISPLAY saves the (new) values to registers 2320-2339 and register 2310 is set to value “0xBB00 + point number” indicating to the Modbus master that the process has finished. The Modbus master can then read out the registers again to update the actual lookup table which resides inside the Modbus master

Reg2310	Status register
Reg2311	Minimum value of x-axis
Reg2312	Maximum value of x-axis
Reg2313	Minimum value of y-axis
Reg2314	Maximum value of y-axis
Reg2315	Divider
Reg2316	Point count
Reg2317	Hairline (0=do not draw, 1=draw)
Reg2318	adjust buttons (0=no buttons, 1=upper left corner, 2 = upper right corner, 3=lower left corner, 4=lower right corner)
Reg2319	reserved
Reg2320	X value of point 1
…	
Reg2329	X value of point 10
Reg2330	Y value of point 1
…	
Reg2339	Y value of Point 10

String Variables

The string section (registers 2400-2911) is divided into 8 blocks with 64 registers in each block (blocks 0 to 7; block 0: registers 2400-2463, block 1: registers 2464-2527, block 2: registers 2528-2591, block 3: registers 2592-2655, block 4: registers 2556-2719, block 5: registers 2720-2783, block 6: registers 2784-2847, block 7: registers 2848-2911). Each block may contain one or more strings. The first register of each block contains the count of strings in that block. The following X registers contain the start register and length of the corresponding string where X = the number of strings in that block. As an example the register section containing strings "Hello,", "Fidelix" and "rules!" should be configured as follows:

Register	HEX value	Decimal value		
2400	0x0003	3	number of strings = 3	
2401	0x0406	1030	Start register of first string = 4, length = 6 bytes	
2402	0x0707	1799	Start register of second string = 7, length = 7 bytes	
2403	0x0B06	2822	Start register of third string = 11, length = 6 bytes	
2404	0x4865	18533	‘H’	‘e’
2405	0x6C6C	27756	‘l’	‘l’
2406	0x6F2C	28460	‘o’	‘,’
2407	0x4669	18025	‘F’	‘i’
2408	0x6465	25701	‘d’	‘e’
2409	0x6C69	27753	‘l’	‘i’
2410	0x7800	30720	‘x’	
2411	0x7275	29301	‘r’	‘u’
2412	0x6C65	27749	‘l’	‘e’
2413	0x7321	29473	‘s’	‘!’

If you choose to have strings of 20 characters, this means you need 10 registers per string + 1 register per string for the string description (=11 registers per string). This means you can have 5 strings per text block (=55 registers), or 40 strings of each 20 characters in total on the display. Of course each string can have its own length and you can choose to have as many and as long (or short) strings as is needed.

Addressing strings on the slave display is done by using pointID: STRING and writing in the “Text” attribute: STRING_XXYY (four decimal characters), in which XX is the block number from 00 to 07, and YY is the number of the string inside that block. "Show point Value" should be selected. In the example above, the string “rules!” is referenced like this: “STRING_0003”. It is also possible to reference strings by point value. The point name should be as normal (POINTxx) and you should write "STRING" in the text field. "Show point Value" must be selected. By then setting the point’s value to 103, you reference block 1 string 3.

The excel tool “multiDISPLAY register value and character cheat sheet.xlsx” which you can find on our partner download page, can be used to easily find the right values for the characters you need.

Display parameters

The sample IEC-code for use with Info Team’s program “OpenPCS” generated by the converter will handle some of these special purpose registers in the end of the file “DisplayInterface.st”. Modify that file if you need more functions to be used in the program you run on your multi-24 module. If the FX-RP multiDISPLAY is connected directly to an FX controller, these registers should be handled by using Modbus devices.

Reg#	Function	R/W	Notes
3000	Internal Temperature Measurement in °C	R	Multiplied by 10, for example 255 = 25.5°C
3001	Current Day	R	Current Day (1-31)
3002	Current Month	R	Current Month (1-12)
3003	Current Year	R	Current Year (e.g. 2018)
3004	Current Hour	R	Current Hour (0-23)
3005	Current Minute	R	Current Minute (0-59)
3006	Tamper Status	R	0=OK, 1023=Tamper detected (read-only copy of register 3062)
3007	Time Schedule 1 Status	R	Possible values from 0..15
3008	Time Schedule 2 Status	R	Possible values from 0..15
3009	Time Schedule 3 Status	R	Possible values from 0..15
3010	Time Schedule 4 Status	R	Possible values from 0..15
3011	Time Schedule 5 Status	R	Possible values from 0..15
3012	History request	R	0=No request, 1-250=point number for which history is requested
3013	Project folder	R	Last character of the currently active project folder name. Used to detect which project is loaded
3014	Internal Temperature Measurement °F	R	Multiplied by 100, so 6883->68.83°F
3015	Internal Temperature Measurement °C	R	Multiplied by 100, so 2555->25.55°C
3016	Proof of “user action”	R/W	Master can write 0 to this register once all registers have been read. Contains the first changed point number after “0” reset. Used for monitoring only one register to see when and if user has changed anything on the display.
3017	Current day of the week	R	0=Sat, 1=Sun, 2=Mon, …, 6=Fri
3018	Not in use	R	Reserved
	…	R	Reserved
3031	Not in use	R	Reserved
3032	Operating Mode Display Brightness	R/W	10%-100%, multiplied by 10
3033	Standby Mode Display Brightness	R/W	0%-100%, multiplied by 10
3034	Apply Brightness Settings	R/W	If 1, apply values from registers 3032 and 3033

Reg#	Function	R/W	Notes
3035	Set VCOML	R/W	0-21, 0=VLCD63x0.60, 1=VLCD63x0.63, etc.
3036	Set VCOMH	R/W	0-63, 0=VLCD63x0.36, 1=VLCD63x0.37, etc.
3037	Set VLCD63	R/W	0-15, 0=VREFx1.780, 1=VREFx1.850, etc.
3038	Set Frequency	R/W	0-6, 0=50Hz, 1=55Hz, etc...
3039	Apply Display Settings	R/W	If 1, apply values from registers 3035 – 3038. These registers contain values to change internal voltage levels, affecting the viewing angle and other viewing related settings. Typically the standard settings are tested and optimal. Change them on your own responsibility.
3040	Enable Click Sound	R/W	0=disabled, 1=enabled, 2=user selectable
3041	Enable Finger Mode	R/W	0=disabled, 1=enabled, 2=user selectable
3042	Enable Upside Down Mode	R/W	0=disabled, 1=enabled, 2=user selectable
3043	Settings Page Access through long press	R/W	0=enabled, 1=disabled, 2=partially enabled (Modbus address disabled)
3044	Open Page	R/W	Open corresponding page if value found from link list (cleared automatically). See "Links" section.
3045	Language	R/W	Set language of time schedule page 0=English, 1=Finnish
3046	Open Page 2	R/W	Open corresponding page if value found from link list (not cleared automatically). See "Links" section.
3047	Calibration Page	R/W	0=normal, 1=disabled (also the settings page will be disabled), 2=skip calibration and jump directly to settings
3048	Calibration delay	R/W	0=normal (7 sec), all other values = seconds
3049	Startup sound	R/W	0=normal ("Positive action" in Quiet mode 1 or higher=startup sound disabled
3050	En-/Disable Daylight Saving Hour Change	R/W	0=enabled (default), 1=disabled (no automatic change from or to daylight saving time.) When disabled, visible (though not editable) on the settings page.
3051	Not in use	R/W	Reserved
3052	Disable temperature measurement compensation	R/W	0=normal (= use compensation), 1=disable compensation (only use this when an external temperature sensor is used; the compensation accounts for the heating of the PCB and the heat generated by the display depending on its brightness)
3053	Internal Temperature Measurement Adjustment	R/W	This value in °C is added to the internal temperature measurement value. (e.g. -23 means -2.3 °C)
3054	New Day Value	R/W	New Day Value (1-31)
3055	New Month Value	R/W	New Month Value (1-12)
3056	New Year Value	R/W	New Year Value
3057	New Hour Value	R/W	New Hour Value (0-23)
3058	New Minute Value	R/W	New Minute Value (0-59)
3059	Apply Values from Registers 3054-3058	R/W	Changing from 0 to 1 will apply new date and time from registers 3054 - 3058
3060	Play Quiet Sound	R/W	Play a quiet sound (1-12=play tune, 13=stop playing. Tunes listed at register 3061 on the next page)

Reg#	Function	R/W	Notes
3061	Play Loud Sound	R/W	Play a loud sound. Possible sounds / tunes are: 1: Positive Action 2: Für Elise 3: Turkey March 4: Minuet 5: Solveig's song 6: Siren 1 7: Siren 2 8: Whistle 9: Tone Scale 10: Positive Beep 11: Negative Beep 12: Disaster Beep 13: Stop Playing
3062	Tamper State	R/W	1023=Tamper detected. Master writes 0 to clear
3063	Boot Loader Startup Register	R/W	Used to start bootloader mode, do not change this value
3064	Non-volatile register	R/W	Registers for values that need to be saved upon power supply interruption. These values are written to the local memory, and read from there upon power-up. Can be written to up to 100 000 times.
3065	Non-volatile register	R/W	
3066	Non-volatile register	R/W	
3067	Non-volatile register	R/W	
3068	Non-volatile register	R/W	

Change log

This section describes the changes made forward from firmware version 1.47, and converter version 1.16 released on 10 November 2014. Mentioned numbers are always firmware version number / converter version number.

1.48 / 1.16: Calibration timeout defined in register 3048 changed to multiples of 1 second.

1.49 / 1.17: Converter now looks in the working folder for the presence of the "ModbusMasterSettings.txt" file. When this is found, master communication is generated. Link to calibration page added.

1.54 / 1.21: Calibration requires two pushes to be far enough from each other, so erroneous clicks are no longer validated. #MOMENTARY:x for slave added. "READREG:xx, WRITEREG:yy," for master added. Added feature to send fixed values from the display master depending on the internal time schedule status.

1.58 / 1.23: Added "SINGLEREG" for reading/writing single holding registers from the multiDISPLAY as Modbus master

1.59 / 1.25: Register 3053 can now be set from the local display itself also.

1.60 / 1.27: Added dynamic links.

1.63: Added free graphical interface possibility for internal time schedules. Removed necessity to separate Modbus master and slave projects in different folders. Changed order of loading point values and graphics to remove delay in graphics update on page load.

1.66 / 1.32: Ability to read from one Modbus slave and write to another (passing values).

1.71 / 1.34: Some small bug fixes to previously added functionalities.

1.73 / 1.36: Register 3017 now contains the day of the week. Added direct link to display settings page. Added visualisation of the current Modbus address. Added start-up sound disabling possibility. Added ability to write to µSD card when uploading graphics through Modbus.

1.75 / 1.37: Increased read timeout from µSD card. Added UINT to Modbus master register value formatting.

1.76 / 1.37: Small bug fix on the #BYTE values

1.78 / 1.38: Added math operators. Small bug fix on the summer- to wintertime clock change. Time changes at 04:00 to 03:00 for the change to wintertime, and at 03:00 to 04:00 for the change to summertime.

1.81 / 1.39: Added possibility to set unused days in time schedules to zero. Signed integers now used for displaying internal registers. A few minor bug fixes concerning +/- buttons and constant / momentary values.

1.82 or 2.85 / 1.40: These two versions are now equivalents for displays from generation 1 and 2.

Following this point, “x.yy” will refer to: x=1 or 2, depending on the hardware version, yy=the actual firmware version number. Both versions will continue to be compiled.

x.88 / 1.41: Improved Modbus master communication. Fix in V2 summer time shift. Added new units.

x.89 / 1.41: Added support for Modbus communication at 4800 bps in slave mode.

x.90 / 1.43: Added #BITx. Changed visualisation of timeout value on Modbus master communication overview page.

x.90 / 1.44: Length of file names is no longer taking up (valuable) memory space when the “µSD” has not been ticked in the converter as the files are all being renamed. A few other converter memory space improvements.

x.91 / 1.44: Small fix; elements of 1 by 1 pixel no longer cause the display to restart itself

x.95 / 1.51: Minimum and maximum in the “Fixed value” field of symbols can now be other points (making a setpoint value even more dynamic). Added #NOEDIT tag to bar display element. Internal display registers (3000+) are now also editable with +/- buttons. Increased available memory for the definition of ModbusDevices in Modbus master mode. Increased available memory for projects in V2 displays. File table is now dynamic; when more than 2 kB are needed for the internal file allocation table, the converter will dynamically attribute more memory to the file table. Added “LOCKPATH” feature to make another project start up by default. Added possibility to hide on #BIT or #BYTE value. Added #UNITSPACE PointID suffix.

x.96 / 1.51: Small fix; links to time schedules on settings page all pointed to time schedule 5 in version x.95.

x.96 / 1.52: Small fix; Chinese comma character (，) caused the converter to crash.

x.97 / 1.53: Added #BITGROUP feature to visualisation options (visualise 2..6 bits together as a group)

x.98 / 1.54: Added possibility to disable daylight savings time (register 3050). Several small bug fixes.

x.99 / 1.55: Added FLOAT and REV_FLOAT visualisation to REGxxxx pointID

Following this point, “x.yy” will refer to: x=2 or 3, depending on the hardware version, where 2.yy is suitable for hardware version 1, and 3.yy for hardware version 2.

x.00 / 1.56: Added SINGLE as an option to write to Modbus slaves (using function code 06). Several other bug fixes, tweaks and improvements, notably on the MATH functionality. READREG / WRITEREG usage in the Modbusmastersettings.txt file optimised.

x.00 / 1.57: 2020-03-18 - Bug fix on including the ampersand in the “CharacterSet.txt” file.