The Nextion Editor Guide

Introduction

This document goes through various features of the Nextion Editor. The Nextion Editor is used to rapidly create Human Machine Interface GUIs for Nextion HMI devices. As such the GUI can be created within Hours instead of Weeks, and Days instead of Months. So while we wont be covering basics such as opening a file, we will point out somethings that might prove helpful to know, or reminders need be made.

Requirements

- Windows Operating System (XP or higher). Users must know and be able to use their Windows OS. Windows support is beyond the scope of Nextion.
- .NET 3.5 Assemblies installed. When needed, download and install the .NET 3.5 Assembles from the Microsoft Website [here].
- Basic programming skills are prerequisite. The Nextion Instruction Set is made up of ASCII text based commands inbound, and binary Return Data. A component's Touch Event "Send Component ID" can be used to defer programming tasks to the user's MCU.
- As such, quickly creating an HMI GUI for Nextion does not demand extreme skills – but basic programming skill are expected. When programming logic Nextion side, then users should have a foundation in programming.

Note: Installations on other Operating Systems may have been accomplished successfully, but is not officially supported and beyond scope of any manual.

Downloading the Nextion Editor

The latest version of the Nextion Editor can be downloaded from [here].

There are two versions of the nextion-setup available for download.

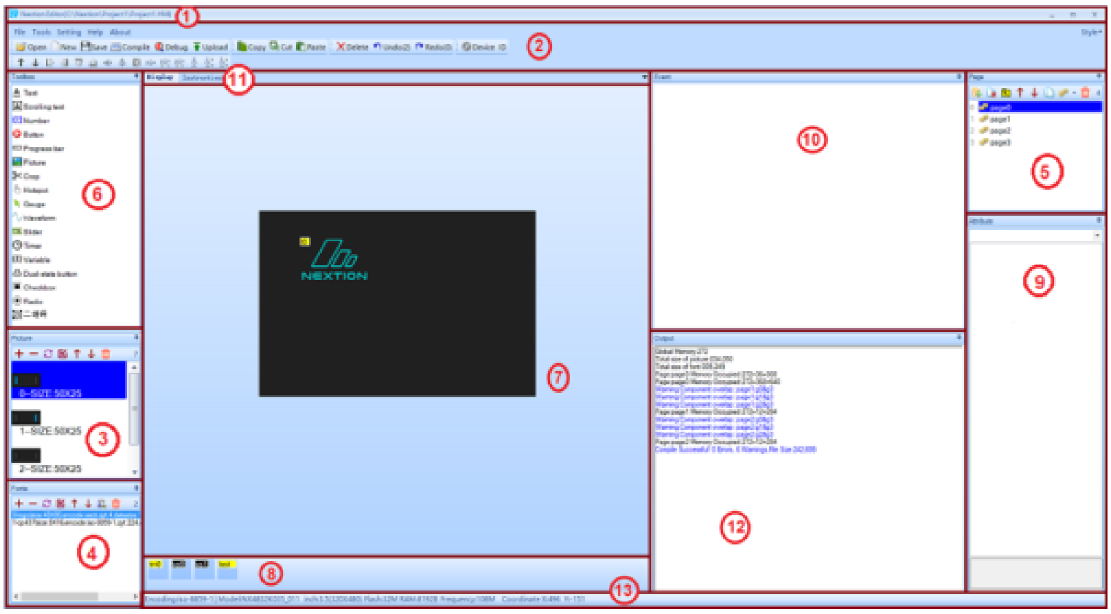
1. The EXE version is installed through the Windows MSI for a more automated installation. Only one version of the Nextion Editor may be registered at a time via the EXE version. When updating within the Nextion Editor, Auto Update will install the EXE version
2. The ZIP version can be unzipped into a user chosen folder and run directly from that folder. For maintaining multiple versions of the Nextion Editor, the ZIP version is recommended. When updating within the Nextion Editor, Manual Update will launch your web browser to the download page so you may download the ZIP version

Parts to this document

1. Nextion Editor Main Interface ... <goto>
2. Debug Simulator Overview ... <goto>

Want the shortened version: try The Nextion Editor: the Quick Review

Nextion Editor Main Interface



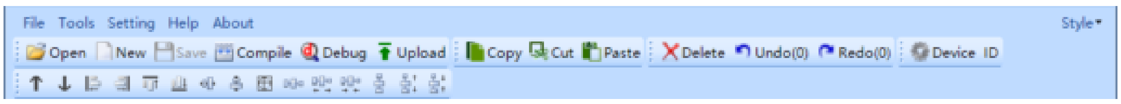
Styles

The Nextion Editor can be set to a Blue or Black themed style (in the upper right corner). Many of the panes can also be adjusted in both size and their location. When needed, you can reset these settings by selecting the ***Reset layout*** under the ***Setting*** menu.

1. Title Bar

The Title Bar contains the path and filename of the HMI project file when an HMI project is loaded. When an HMI project is not currently loaded, a TFT file can be loaded into the Debug Simulator by pressing Debug.

2.a) Main Menu



File Menu

Here, Users can create a ***New*** project, ***Open*** an existing project, ***Save*** the current project, ***Close*** their current project, and ***Exit*** the Nextion Editor. ***Import Project*** will append an existing project into the current project – usually with resulting naming and renumbering issues. As such, it is recommended to import individual pages if required. ***Clear Recent Projects*** will clear the Project filenames in the ***Recent projects*** pane – it will not delete any of the files from the hard drive.

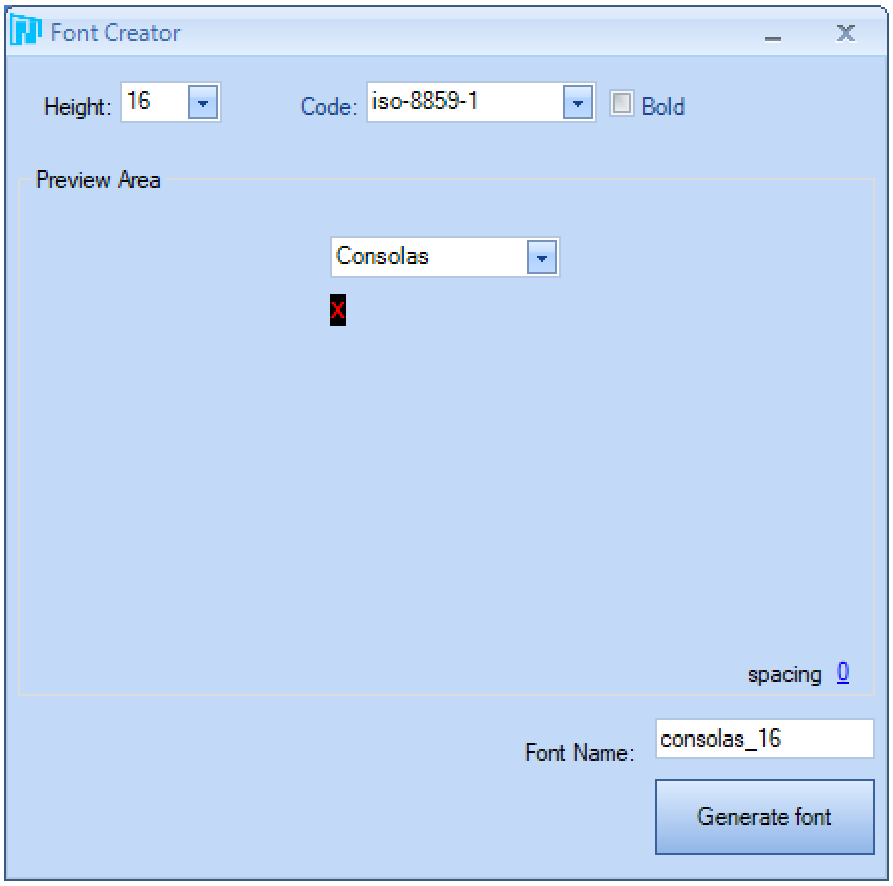
Two other useful folder shortcuts are included here: Open Build Folder and Backup Dir. All compiled projects create a *.TFT file in the C:\Users\Username\AppData\Roaming\Nextion

Editor\bianyi folder. ***Open Build Folder*** will open this folder in Windows Explorer. The Backup Directory keeps a copy of older HMI projects opened with a new version of the Nextion Editor in the C:\Users\Username\AppData\Roaming\Nextion Editor\backup folder. ***Backup Dir*** will open this folder in Windows Explorer.

Tool Menu

The Nextion Editor built-in *.zi font tool "Font Creator" can be launched by selecting ***Font Generator***. Any project containing components that output a .txt or .val attribute in a visual manner will require at least one font to be included in the user's HMI project. The Font Creator is included as a convenient way to generate a font quickly, though arguably not the best quality.

Font Creator



Height: 16

Code: iso-8859-1

Consolas

spacing 0

Font Name: consolas_16

+ Character Height is in pixels, a multiple of 8 from 16 to 160.
+ Character Width will be 1/2 of this Height.
+ Select a Height, encoding, Font from installed Fonts, additional spacing.
+ Give your font a name and then click ***Generate Font***.
+ Save your *.zi font file to your chosen zi font folder
+ You will be given an opportunity to add this to your HMI project.

Settings Menu

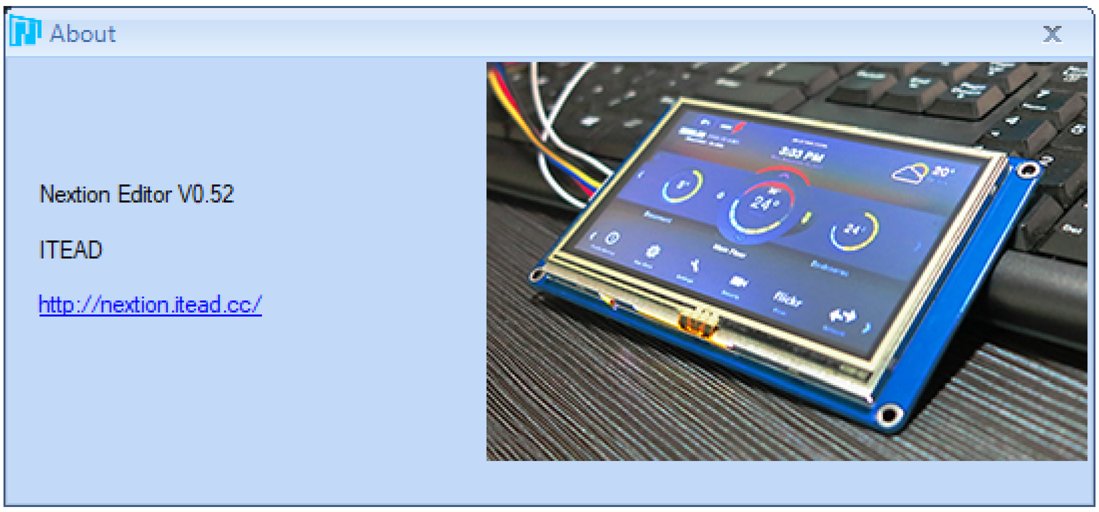
Here, a user can change a few Editor Setting by selecting ***Setting***,
or reset their Editor style layout selecting ***Reset layout***.

Help Menu

Selecting the ***Help*** *m*enu-item will launch Nextion Instruction Set in the user's External Web Browser.

About Menu

About will show the current Version of the Nextion Editor and Nextion site link.

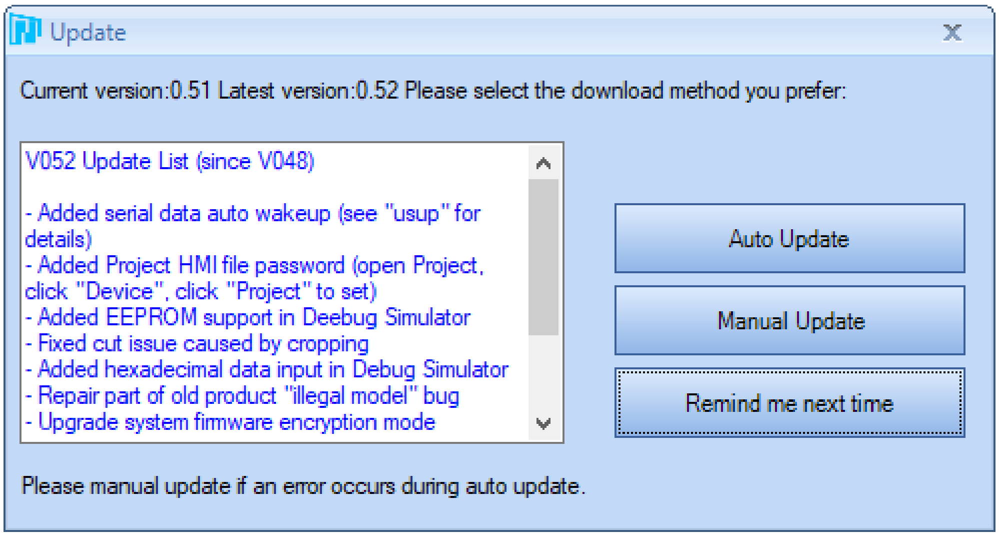


Nextion Editor V0.52

ITEAD

http://nextion.itead.cc/

Check for new version will use the Internet to check for a new version. If a new version is available, you will get an Update dialog. There are two versions of the Nextion Editor: an EXE version and a ZIP version. The EXE version will install in the Program Files (x86) folder and take care automatically for file associations. Selecting ***Auto Update*** will install the EXE version. The ZIP version can be downloaded and extracted into any folder of choice and then run directly from this folder. Selecting ***Manual Update*** will launch the user's external Web Browser to the Nextion Editor download page. Here the ZIP version can be downloaded. Selecting ***Remind me next time*** will close the Update dialog without any updates and continue with the current version.


Update

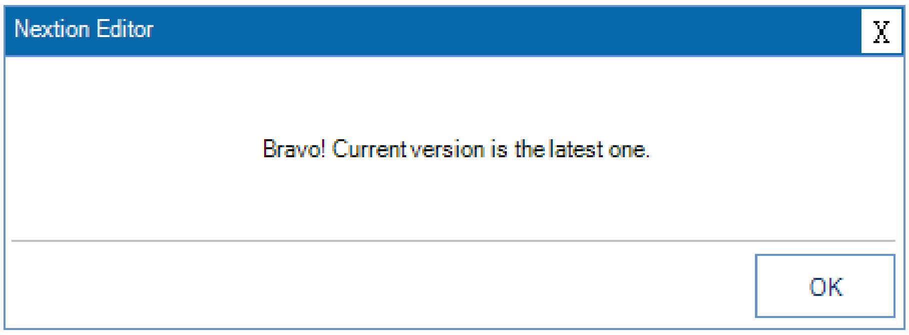
Current version:0.51 Latest version:0.52 Please select the download method you prefer:

V052 Update List (since V048)

- Added serial data auto wakeup (see "usup" for details)
- Added Project HMI file password (open Project, click "Device", click "Project" to set)
- Added EEPROM support in Deebug Simulator
- Fixed cut issue caused by cropping
- Added hexadecimal data input in Debug Simulator
- Repair part of old product "illegal model" bug
- Upgrade system firmware encryption mode

Please manual update if an error occurs during auto update.

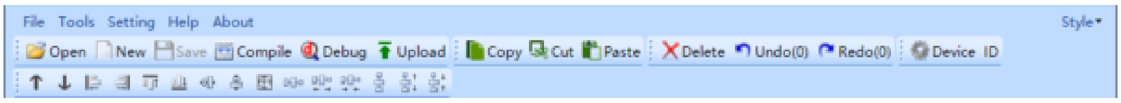
When the version is the latest and most current version, users will see:


Nextion Editor

Bravo! Current version is the latest one.

New Message menuitem has no current function at the moment.

2.b) Toolbars


File Tools Setting Help About
Style
Open New Save Compile Debug Upload
Copy Cut Paste
Delete Undo(0) Redo(0)
Device ID

Open, New and Save

These will ***Open*** an existing project, create a ***New*** project, or ***Save*** the current project.

Compile

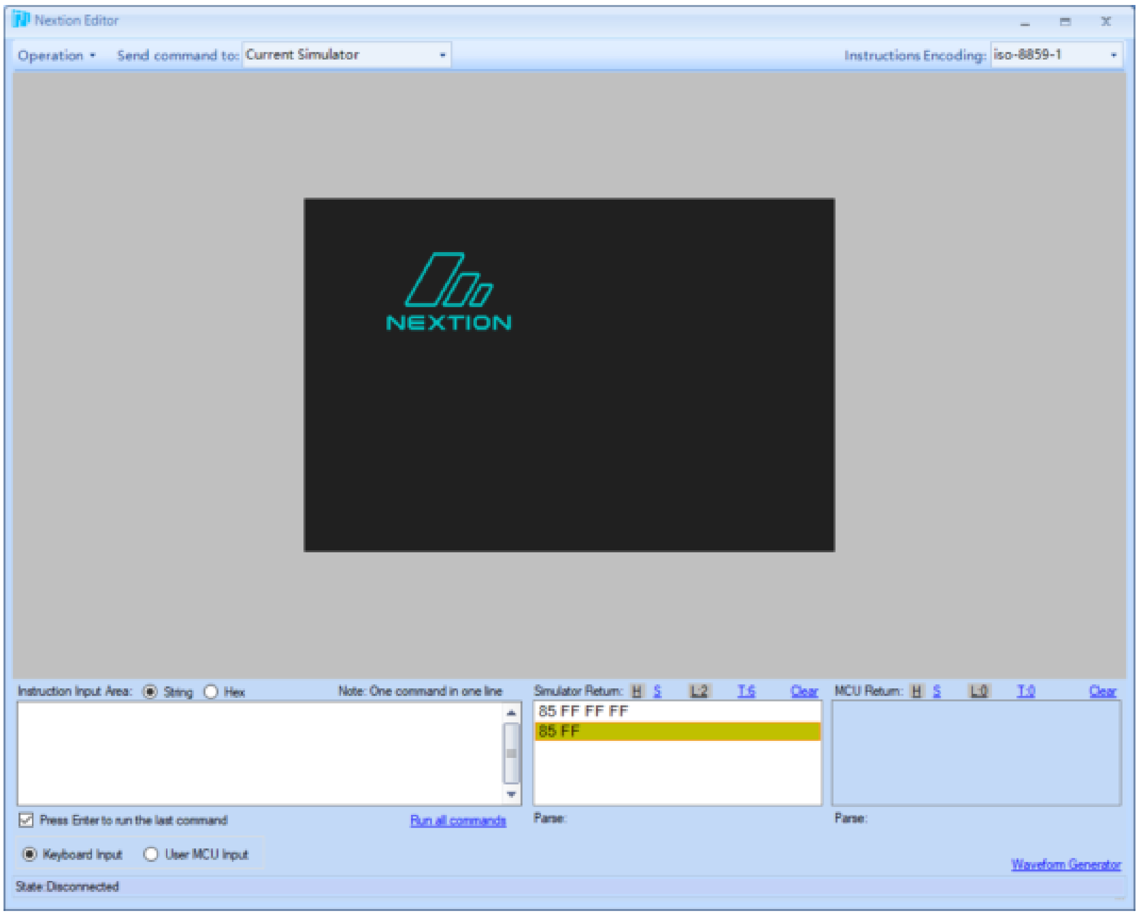
Use Compile to rebuild the currently loaded project. Any Warnings or Error Messages will be placed in Output Pane. If there is no error, a *.TFT file will be placed in the bianyi folder, but if there is an error then this *.TFT file will be zero bytes. Do not upload (over serial or via microSD) a zero byte *.TFT to the Nextion.

Pay attention to any warnings as these will mean your project may not run as you expect. Pay attention to any error messages as they will need to be corrected before continuing. Error messages are descriptive, and if it is a code error then the user can click to jump directly to the coding error location.

Compile is more of a build and assembly process. This is only stated so that users do not make the wrong expectations of native ARM code when making feature requests and/or Bug Reports. Nextion remains closed source.

Debug

The Nextion Editor contains a built-in Simulator that can be accessed via the toolbar ***Debug***. To be clear this is not a precision emulator and is intended to be sufficient to assist in debugging a users project. It in no way is meant to replicate the Nextion device exactly. (Any Windows OS is sufficient to make such precision unattainable.) This will be covered in a future section.



Send command to: Current Simulator

Instructions Encoding: iso-8859-1

Instruction Input Area: [✓ String] Hex

Note: One command in one line

Simulator Return:

```
85 FF FF FF
85 FF
```

MCU Return:

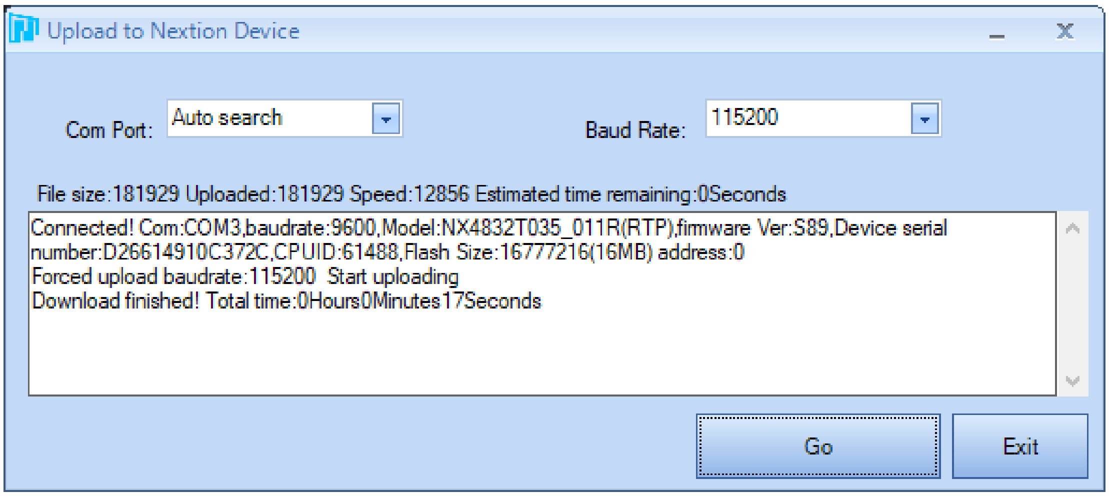
[✓ Press Enter to run the last command]

[✓ Keyboard Input] User MCU Input

State: Disconnected

If a project is not currently loaded in the Nextion Editor, ***Debug*** will open a dialog to open a compiled *.TFT file directly. This is handy for loading demos or sharing ideas without surrendering your code. Although the Simulator can run a *.TFT file from any Nextion series or model, it is important that the same version is used to simulate that was used to create the *.TFT file.

Upload



```
File size:181929 Uploaded:181929 Speed:12856 Estimated time remaining:0Seconds
Connected! Com:COM3,baudrate:9600,Model:NX4832T035_011R(RTP),firmware Ver:S89,Device serial
number:D26614910C372C,CPUID:61488,Flash Size:16777216(16MB) address:0
Forced upload baudrate:115200  Start uploading
Download finished! Total time:0Hours0Minutes17Seconds
```

Selecting Upload will launch the upload dialog. Ensure the Nextion is connected via serial (typically via USB to TTL adapter) before upload or the Port may not be available to select. Auto search feature will look for your Nextion's reply to the connect command, but realize that data is being sent on all serial ports that are searched (and may interfere with the other connected serial devices). A better choice is to select the correct Port and Baud Rate. Proper configuration of Serial adapters, Windows drivers, device conflicts, etc is beyond the scope of Nextion support and the domain of user responsibility to know their used Operating System and devices.

Once Nextion has responded to the connect command, the upload process will begin. Do not interrupt this process until completed. If the process has been interrupted, resetting the serial port may be required. When a partial *.TFT file has been uploaded and uploading over serial is no longer an option, then the user will need to upload via the microSD method.

Copy, Cut, Paste and Delete

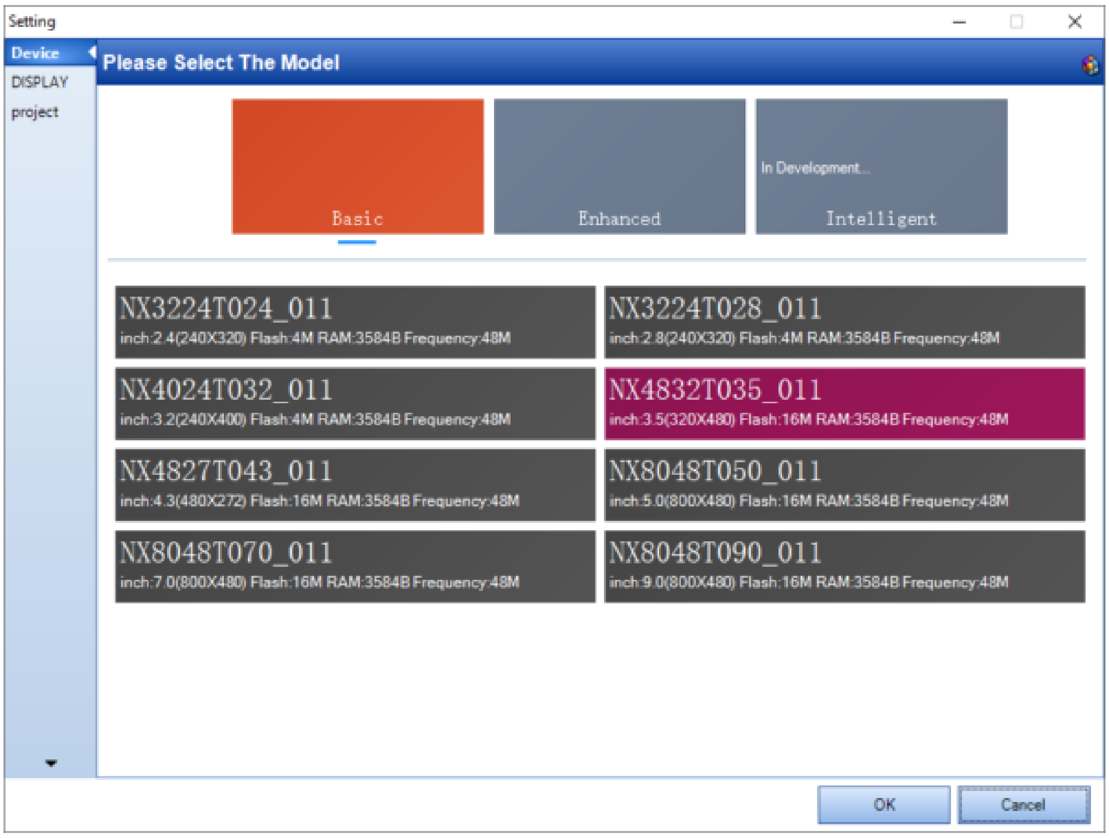
Users can select components or Multiple components and then ***Copy***, ***Cut***, ***Paste*** or ***Delete*** as required.

Undo, Redo

Use Undo and Redo to ***Undo*** the last operation, or to ***Redo*** the last operation undone.

Device

The steps to configure your HMI project for your Nextion Series and Model is usually done at the time of creating a New project. When you need to make changes, ***Device*** will launch the following window with the ***Device*** tab selected. First select the Nextion Series: ***Basic*** for the T models, and ***Enhanced*** for the K models. Then select the Nextion Model. For the Multi-touch Capacitive *Nextion* NX8048K070_011C: Select the ***Enhanced*** series and then the **NX8048K070_011** Nextion Model.



NX3224T024_011
inch:2.4(240X320) Flash:4M RAM:3584B Frequency:48M

NX3224T028_011
inch:2.8(240X320) Flash:4M RAM:3584B Frequency:48M

NX4024T032_011
inch:3.2(240X400) Flash:4M RAM:3584B Frequency:48M

NX4832T035_011
inch:3.5(320X480) Flash:16M RAM:3584B Frequency:48M

NX4827T043_011
inch:4.3(480X272) Flash:16M RAM:3584B Frequency:48M

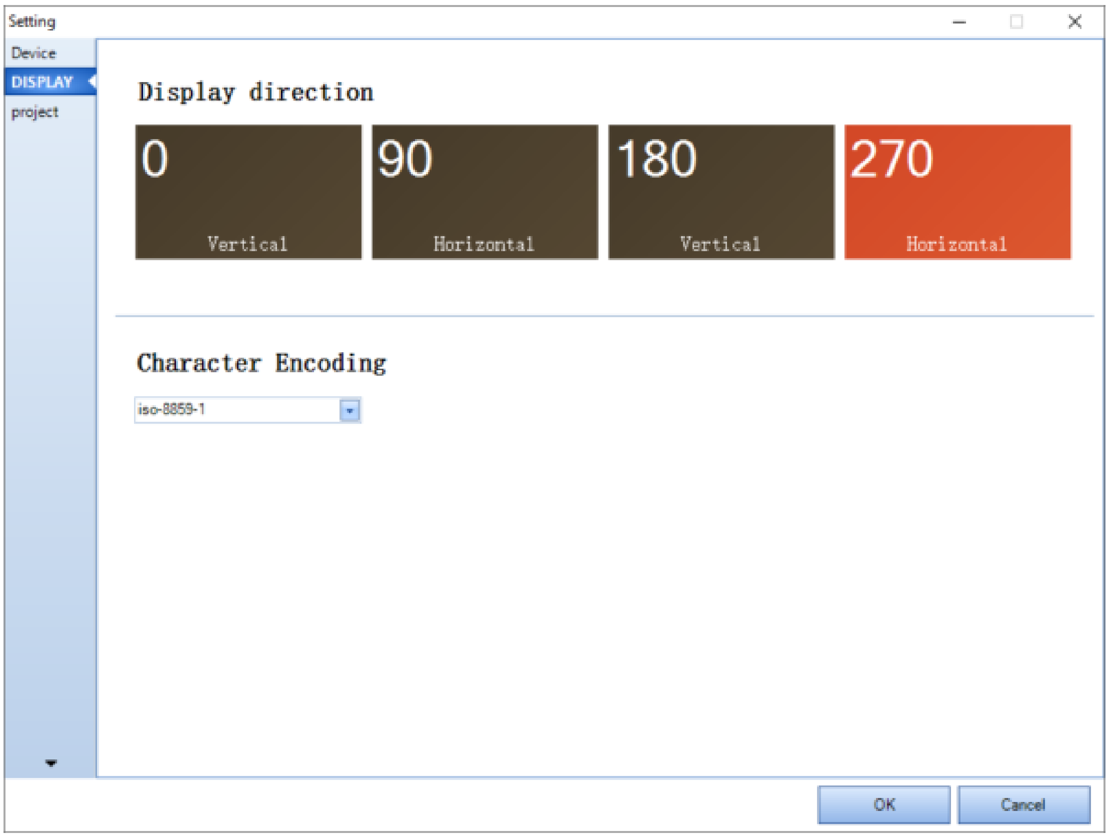
NX8048T050_011
inch:5.0(800X480) Flash:16M RAM:3584B Frequency:48M

NX8048T070_011
inch:7.0(800X480) Flash:16M RAM:3584B Frequency:48M

NX8048T090_011
inch:9.0(800X480) Flash:16M RAM:3584B Frequency:48M

Selecting the ***DISPLAY*** tab, the user can select the orientation and the Character Encoding. 0° is the native viewing angle for the selected model. Users can choose alternative orientations (90°, 180° or 270°) but this will not be the native viewing angle. Character Encoding is default iso-8859-1. Select from the character encodings that make sense for your HMI project to best display your local character sets. There are a selection of single byte and double byte character sets available. Unicode and UTF will not be among the supported encodings.

Currently supported Character encodings include:
ASCII, ISO-8859 (1,2,3,4,5,6,7,8,9,11,13,15),
GB2312, BIG5, KS_C_5601_1987,
Windows 874, 1255, 1256, 1257, and 1258


Setting

Device

DISPLAY

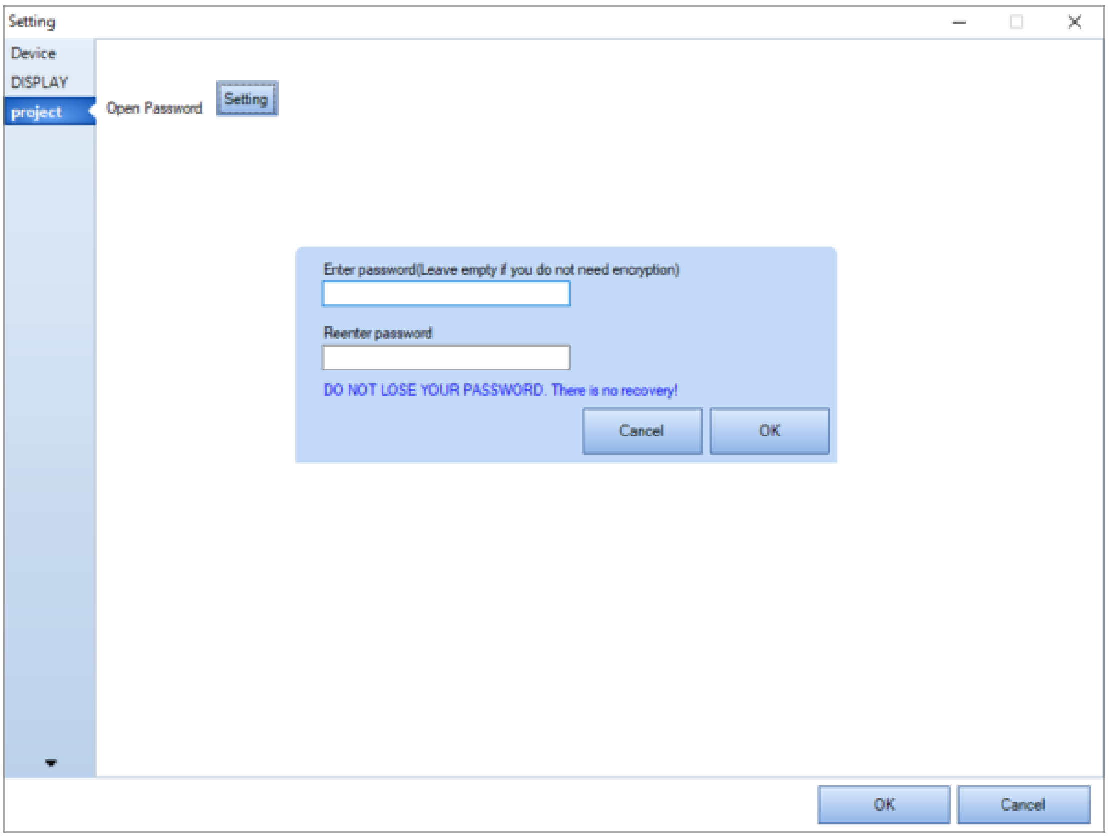
project

Display direction

Character Encoding

iso-8859-1

If you desire to password protect your entire HMI project, selecting the ***project*** tab will bring up the Open Password Setting button. If an existing password exists, it will need be entered before a new password can be set. When a Password is lost, it is not retrievable. Fair warning is given: DO NOT LOSE YOUR PASSWORD. There is no recovery! A project with a lost password would need to be rebuilt – So, do not lose – or do not use.


Setting

Device

DISPLAY

project

Open Password Setting

Enter password(Leave empty if you do not need encryption)

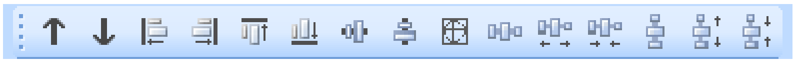
Reenter password

DO NOT LOSE YOUR PASSWORD. There is no recovery!

ID

Selecting ***ID*** to will toggle if the component .objnames are displayed in the upper left region of the component space. Yellow labelled components have a .vscope local, while black labelled components have a .vscope of global. (Hint: Event code is never global). When selecting multiple components, green labelled components indicate multiple components have been selected, while the one blue labelled component will be used as the baseline component. To change the baseline component while the group is still active, simply click on the already selected component you want to become the baseline component.

The Component Layout Toolbar



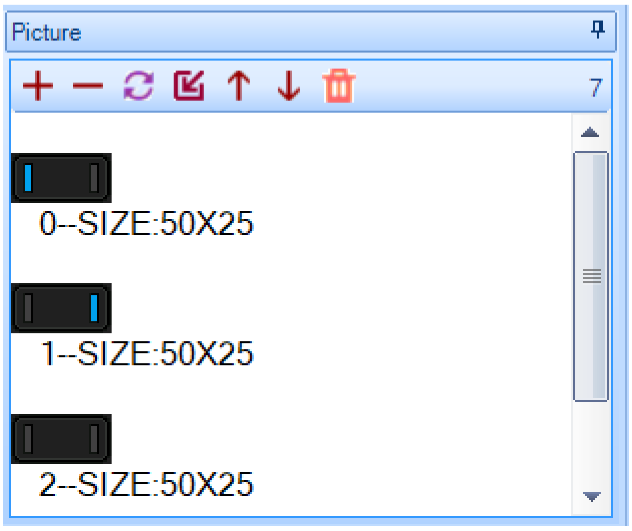
For Renumbering components: ***Bring Top*** (Arrow Up) will take the selected component(s) and renumber to the highest .id on the page. ***Bring Bottom*** (Arrow Down) will take the selected component(s) and renumber to the lowest (starting at 1, page component is always 0) .id on the page.

For Aligning components: ***Align Left***, ***Align Right***, ***Align Top*** and ***Align Bottom*** will take a group of selected components (green ID labels) and bring the alignment to match the component with the blue ID label.

For Resizing components: ***Same Width***, ***Same Height*** and ***Same Size*** will take a group of selected components (green ID labels) and set the size (width, height or both) to match the component with the blue ID label.

For Spacing components: ***Equal Horizontal***, ***Increase Horizontal spacing***, and ***Decrease Horizontal spacing*** will take a group of selected components (green ID labels) and adjust the horizontal spacing between components using the component with the blue ID label as the baseline component. Likewise: ***Equal Vertical***, ***Increase Vertical spacing***, and ***Decrease Vertical spacing*** will take a group of selected components (green ID labels) and adjust the vertical spacing between components using the component with the blue ID label as the baseline component.

3. Picture Resource Pane



Pictures are imported into your HMI project through the Picture Resource Pane. A picture is added with ***Add***, deleted with ***Delete***, and swapped with ***Replace***. ***Insert*** will add the imported pictures before the highlighted picture. Use the ***Arrow Up*** and ***Arrow Down*** to renumber the picture resource number within the Picture Resource Pane. Using ***Trash*** will delete all pictures within the project. Note that Delete will not delete a picture resource if it is being used with a component.

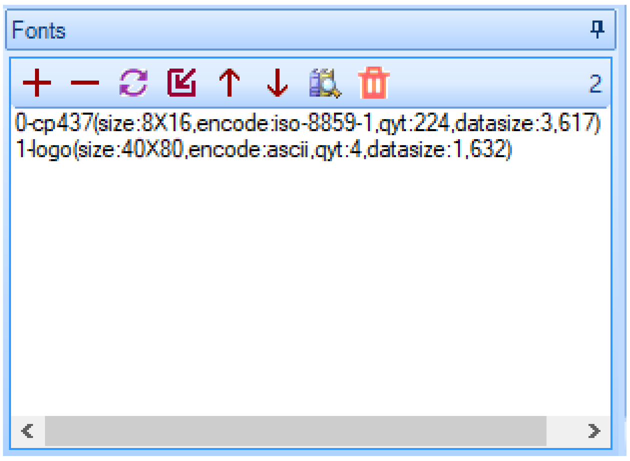
The accepted picture types to import are *.jpg, *.png, non animated *.gif and *.bmp files. When importing a picture, the picture is converted into the 565 16 bit color format used by Nextion. Nextion is not a graphics card, as such transparency and in picture animation is not supported. In native 16-bit color, picture resources consume 16 bits per pixel, or width x height x 2 bytes.

Exporting a picture resource is accessed by highlighting a the picture resource, right clicking and selecting Export. The available picture formats for export are *.jpg, *.png and *.bmp. Not that *.jpg may save space but is a lossy format. Once the data is lost, it is lost.

When using cropping, a full screen image is strongly recommended for the background. This avoids pulling data from non-existent space – which will resemble randomized colors (data from other locations). Cropping can consume more cycles than using a picture, using a picture will consume more cycles than solid colors. When cycle time becomes to high to render, tearing and flickering will be the sign. Nextion is an HMI device and not designed for multi-media and streaming. That said, amazing effects can still be achieved with purposeful programming.

Assignment of the picture by code at runtime would look like:
p0.pic=1ÿÿÿ over serial, or as p0.pic=1 inside an event as code.

4. ZI Font Resource Pane



```
0-cp437(size:8X16,encode:iso-8859-1,qyt:224,datasize:3,617)
1-logo(size:40X80,encode:ascii,qyt:4,datasize:1,632)
```

The ZI Font Resource is very important! If any visual component used in the user's project contains a .txt or .val attribute, then at least one font must exist in the project. If a user wants to use various size of styled fonts in their HMI project, they will need to generate a ZI Font file for each style and size they will use. For most users this will mean starting with a ZI Font generated with the built-in Font Creator (covered in the Menu section above).

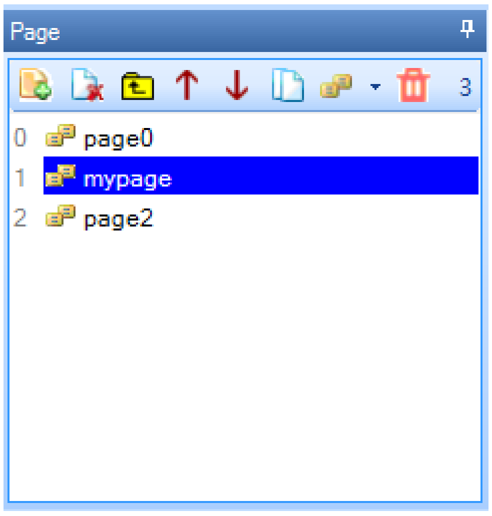
These *.zi font files will need to be added into the ZI Font Resource Pane. Each ZI Font Resource will show the font resource number, it's given name user gave at creation time, it's size in pixels Width X Height, it's character encoding, number of characters in the set and the size in bytes the font contributes to the HMI project's overall filesize. The appropriate *.zi font can then be selected by its number and assigned in a visual components .font attribute. Note that the design size of the visual component is set at design time and can not be changed at runtime.

Assignment of the font by code at runtime would look like:
t0.font=1ÿÿÿ over serial, or as t0.font=1 inside an event as code.

Properly formed ZI font resources will have a height that is either equal to the width or a height that is double the width. Height will always be a multiple of 8 from 16 to 192. ZI fonts are monobit (pixel is either on or off) and Fixed Width. ***Hint***: Generating a proportional font like Arial where W and @ are 24×24 and squeezing into a 12×24 space is like pouring 2 litres into a one litre bottle and expecting all 2 litres have been saved. Fonts that do not conform to the ZI size formula just stated will indeed have issues when it is rendered on the Nextion device.

A ZI font resource can be viewed by selecting the font and pressing ***Preview***. Trash will ***Delete all*** loaded ZI Font Resources. Multiple *.zi fonts can be imported with ***Add***. A font must not be associated to a component to remove with ***Delete***. A font can be swapped out with a different font using ***Replace***. ***Insert*** will import the font before the highlighted font. Use the ***Arrow Up*** and ***Arrow Down*** to renumber the font resource number within the ZI Font Resource Pane.

5. Page Pane

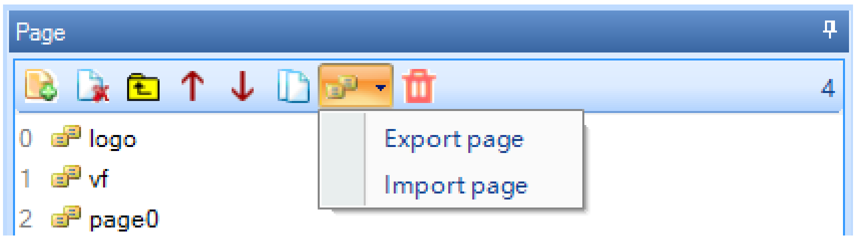


Every HMI project needs to have at least one Page. Pages are created or imported into your HMI project through the Page Pane. A Page is created with ***Add***, deleted with ***Delete***, and copied with ***Copy***. ***Insert*** will create a new page before the highlighted page. Use the ***Arrow Up*** and ***Arrow Down*** to renumber the page number within the Page Pane. Using ***Trash*** will delete all pages within the project.

Pages can be renamed to a maximum of 14 characters and the page names are case sensitive. To rename your page highlight the page, right click. and select ***Rename***. Then enter your new name (it is recommended to press Enter to ensure the change takes place). Double clicking a Highlighted page name will also trigger the page renaming function.

The page ***Lock*** and ***Unlock*** functions are only accessed by right clicking the highlighted page name and selecting Lock or Unlock. If the page has been locked with a password, the password must be entered to access the components and event code. There is no password recovery should the password becomes lost, so don't use or don't use. As an example, the keyboard pages are imported as locked, but do not use passwords (the keyboard pages are also a good coding example to review).

Pages can be exported from one project to another project with ***Export page***. This is the preferred way to share components and partial projects. Page files *.page can not be compiled on their own. If you want to export a page as locked or locked with a password, this must done before exporting.



To import a page, use *Import page*. Imported pages can be used independent if they are locked with a password, locked without a password, or unlocked. Locking is to protect the code. So as long as proper documentation about the page variables and functions accompanies the *.page file, an end user can use even a locked page. When a page is imported with naming conflicts, the affected conflicting names will be renamed. It is therefore relevant to perhaps select meaningful names.   To import a copy of one of the keyboard pages, one can either import the *.page file directly or use the .key attribute of a Text or Number component.

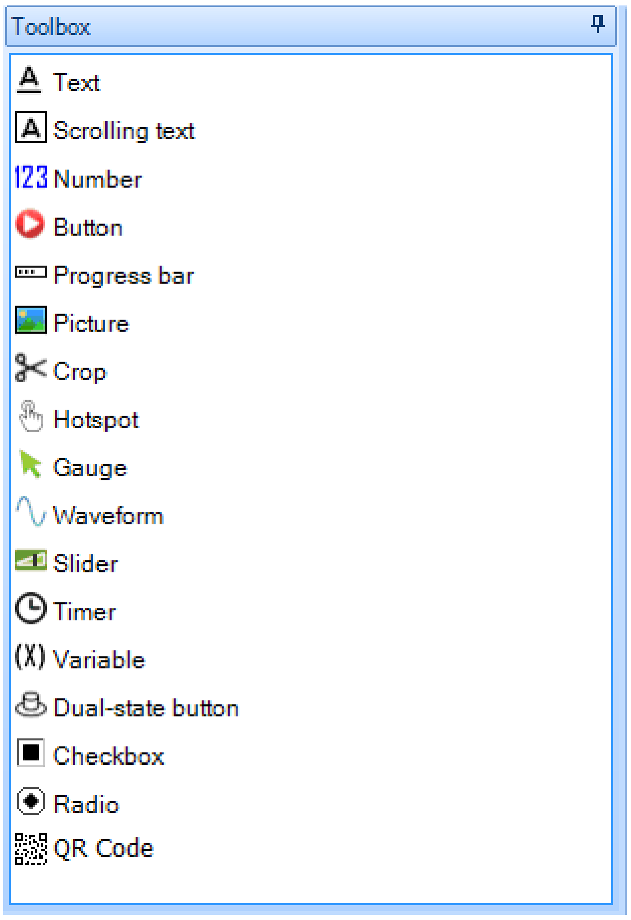
In an HMI project a page is a localized unit. When changing pages, the existing page is removed from memory and the requested page is then loaded into memory. As such components with a variable scope of local are only accessible while the page they are in is currently loaded. Components within a page that have a variable scope of global are accessible by prefixing the page name to the global component .objname.

As an Example: A global Number component n0 on page1 is accessed by page1.n0. A local Number component n0 on page1 can be accessed by page1.n0 or n0, but there is little sense to try access a local component if the page is not loaded. Only the component attributes of a global component are kept in memory. Event code is never global in nature.

A Page always contains a page component, and this page component will always have an .id of 0. Making the page component global, does not make the components within a Page global – just the very few page attributes of the page component (of which only .pic or .bco are changeable at runtime). The page component .id used with the b[.id] component array is not the page index number used with the p[index] page array.

A Page component can have a background of either : Solid color, image background, or no background. An image background should use a fill screen image to avoid calling non-existent data. No background will show the current page components over top the last unloaded page, this must be used with caution. No background on a no background, on a no background soon causes unwanted side effects. There is a hard limit of 250 components allowed per page, and 254 pages per project.

6. Components Pane



The ***page*** component not listed above is always created when the new Page is added to the Page pane. The page component will always have an .id of 0 and is always the bottommost layer. There is a hard limit of 250 components allowed per page.

Many components have multiple .sta choices of
– crop image (pulls background from .x and .y location of a user chosen picture resource)
– solid color (sets background to be a user chosen 565 color)
– image (text if any will be drawn over the user chosen image)

Many components have multiple .style choices of
– flat (no lines will be added around edges)
– border (lines will be drawn around edges)
– 3D_Down (lines will be drawn to yield impression of lowered)
– 3D_Up (lines will be drawn to yield impression of raised)
– 3D_Auto (lines will be drawn Up/Down according to state)

Many components have multiple text alignment and placement options
– .xcen of Left, Center or Right
– .ycen of Top, Center or Bottom
– .spax will add extra blank pixel spacing to right of each character
– .spay will add extra blank pixel spacing to bottom of each character

– .isbr for multi-lined (set to true) or single line (set to false)
– .pw for masking (Character is off, Password will mask with asterisk)

The ***Text*** component is a highly customizable component. The Text component has the .pw attribute for masking (Character is off, Password will mask with asterisk) and the .key attribute for integrating one of the included example keyboards (must be set to .vscope global before use).

The ***Scrolling text*** component combines an integrated timer component with a text component. The .pw option is not available with this component. The .key attribute allows for integrating one of the included example keyboards (must be set to .vscope global before use). There is a hard limit of 6 timer components per page within your project.

The ***Number*** component is used for signed 32-bit integer values. The .lenth (as spelled) sets the number of digits shown (useful for leading zeros). The new .format attribute allows for a choice of integer, currency (comma separated every three digits, not floats), or hexadecimal. Input should be in integer or hexadecimal. The .key attribute allows for integrating one of the included example keyboards (must be set to .vscope global before use).

The ***Button*** component is again highly customizable and integrates text in a momentary manner. Use images or event code to suit tastes.

The ***Progress bar*** component is for progress, thus a valid range of 0 to 100 to represent the percentage of progress. (Please no more requests to extend the range, even if many may give 110% effort). Best effects for progress are attained using images.

The ***Picture*** component will allow any picture resource to display in the Picture component. Example p0.pic=3. It is important that the picture resource is the user defined size in .w and .h or the picture resource will over draw the picture component boundaries, or incorrectly insert adjoining data. The Picture component is useful to represent multi-states and animation sequences.

The ***Crop*** component will replace its boundaries with the same location and boundaries from the picture resource pointed to with .picc. It is highly recommended that the picture resource being used is a full screen image to avoid errors (must be fullscreen image). The Crop component is useful to represent states.

The ***Hotspot*** component is a user defined touch spot to its overlaying region. At a 2 pixel by 2 pixel region, it makes for a useful code holder to be later called by the click command – thereby creating a user defined function. As a Hotspot, it turns any image area into a button, such as in creating a customized keyboard.

The ***Gauge*** component is a full circular component with value in degrees. This means a range of 0 to 360. Gauge components are not useful for stacking (example: a three handed clock), as the redrawn gauge will overwrite any lower gauge. The gauge component is always a square in nature. Semi-circular gauges at the screen's edge are not achieved with the gauge component.

The ***Waveform*** component is used to plot y axis data points on up to 4 channels. Waveforms are never global. Up to 4 waveforms can be used on a single page.   The Waveform component is limited to a y axis data range of 0 to 255 or 0 to waveform height -1. As a data point is added, it will consume one column, with the next data point using the next column. Recent changes now allow a variable to be used in the add command. Example add 1,0,h0.val. As the waveform data points are not global, changing pages away and back will revert the waveform to an emptied state. The addt command becomes useful to refill the waveforms on page load (such coding remains within the user domain).

The ***Slider*** component can be horizontal or vertical. The slider has the added event code for Touch Move, useful for providing updates to the sliders current position. Best results are attained with images. Slider length includes the size of the thumb as well as the range (often overlooked in calculations).

The ***Timer*** component is not expected to be a high precision interrupt driven component. It is however useful for queueing reoccurring event code after elapsed .tim has expired. As code is sequentially processed, it is very easy for the time to process the requested user event code to exceed the .tim intervals and therefore not interrupt driven (to avoid such stack overflows) and not high precision. There is a hard limit on the maximum number of timers running in a single page, this limit is 6. Beware that the scrolling text component integrates 1 timer. Timer attributes can have a variable scope of global, event code is never global. As such timer code can only be triggered within the current page they are designed in. As the timer is a non visual component, they are added below the Design Canvas.

The ***Variable*** component is a non visual component and also added below the Design Canvas. Variables are either 32 bit signed numeric or string content.

The ***Dual-state button*** component is an expanded Button maintaining its state between toggles.

The ***Checkbox*** component is another example of a lightweight dual-state component with less customization and lower memory usage.

The ***Radio*** component is yet another example of a lightweight dual-state component with little customization and lower memory usage. Obtaining grouping is achieved via user code (remains in the user domain).

The ***QR Code*** component is used to generate a 2D scan able QR. It is limited to a byte maximum for the .txt attribute of 84 on Basic T models and 192 on the Enhanced K models.

To add any of the above components to the currently design page, simply click on the component and it will be added with its .id set to the number of components on the page. All components within a page are listed in .id ascending order in the Component Drop down in the Attributes Pane. Then continue with placement and adjustment of .attributes as desired.

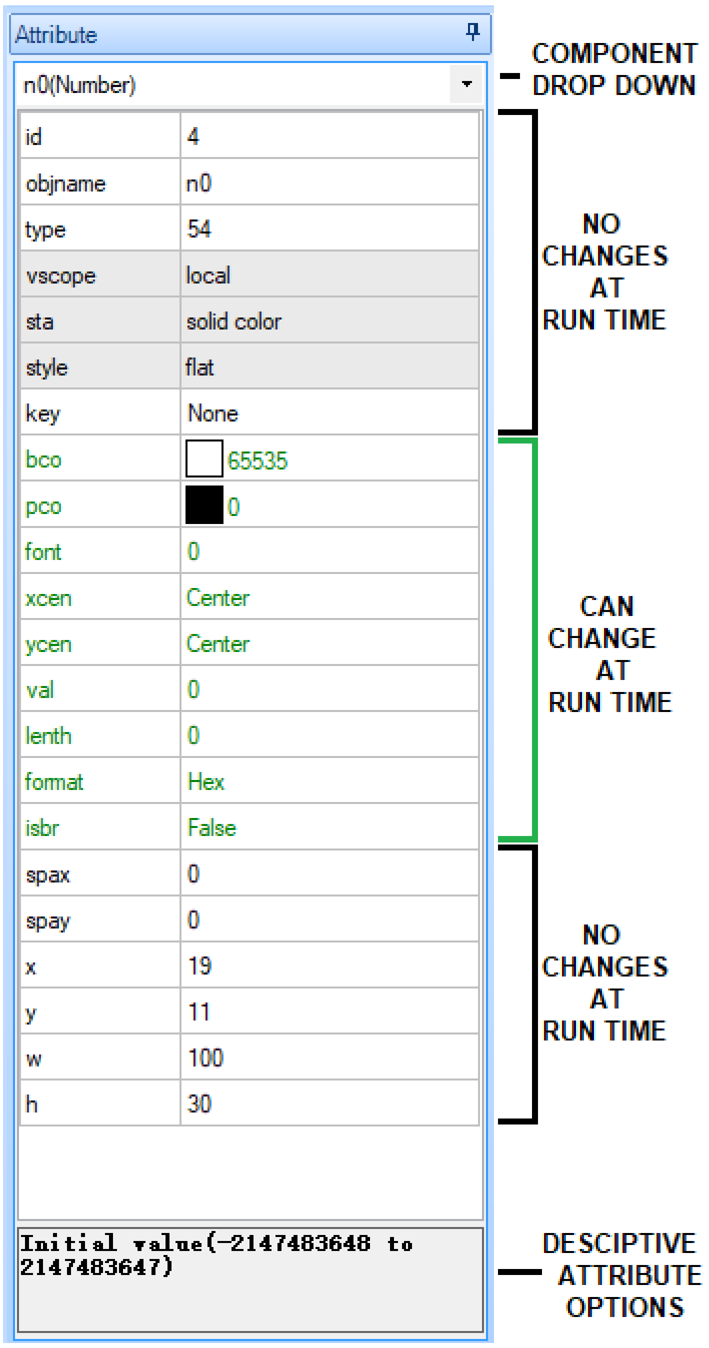
7. Design Canvas Visual Components

This is the main design space for the visual/touch components for the current Page. The Page's page component is always .id 0 and always the most back layer. The mouse coordinates are displayed on the Status Bar aiding in precision placement. Selected components can be moved in one pixel offsets using the keyboard arrow keys. For best precision, use the component's .x and .y attributes. Components selected with the left mouse click can be moved by dragging. As such using the right mouse click to select a component will not accidently move the component. Resizing a component via edge dragging can only be achieved on the bottom and right edges. There is a limit of 250 components (visual and non visual) allowed per page.

8. Design Non Visual Components

A Page's non visual components (Variables and Timers) will be listed in this area. This area is not displayed if there are no Variable or Timer components used in the page. There is a limit of 250 components (visual and non visual) allowed per page.

9. Attributes Pane



Attribute

n0(Number)

id	4
objname	n0
type	54
vscope	local
sta	solid color
style	flat
key	None
bco	⬜ 65535
pco	⬛ 0
font	0
xcen	Center
ycen	Center
val	0
lenth	0
format	Hex
isbr	False
spax	0
spay	0
x	19
y	11
w	100
h	30

Initial value(-2147483648 to 2147483647)

The Attribute Pane contains the list of components included within the current design page in the Component drop down. Clicking on a component, or selecting it from the drop down will display the component's available attributes. The left side contains the attribute name, the right side contains the attributes current value. Clicking on an attribute will display the attributes meaning and valid range/options at the bottom of the Attribute Pane. Double clicking a field with bring up resource editor for the attribute if attribute has such (ie: .pco opens color picker, .pic opens picture chooser).

Any attribute in black is read only at runtime (with the exception of .objname and .vscope) The .objname is inaccessible, and the .vscope does not report correctly. Any attribute in green can be both read and changed by user code at runtime. Empty unassigned attributes values or invalid attribute values will need to be resolved before a successful compile can be achieved.

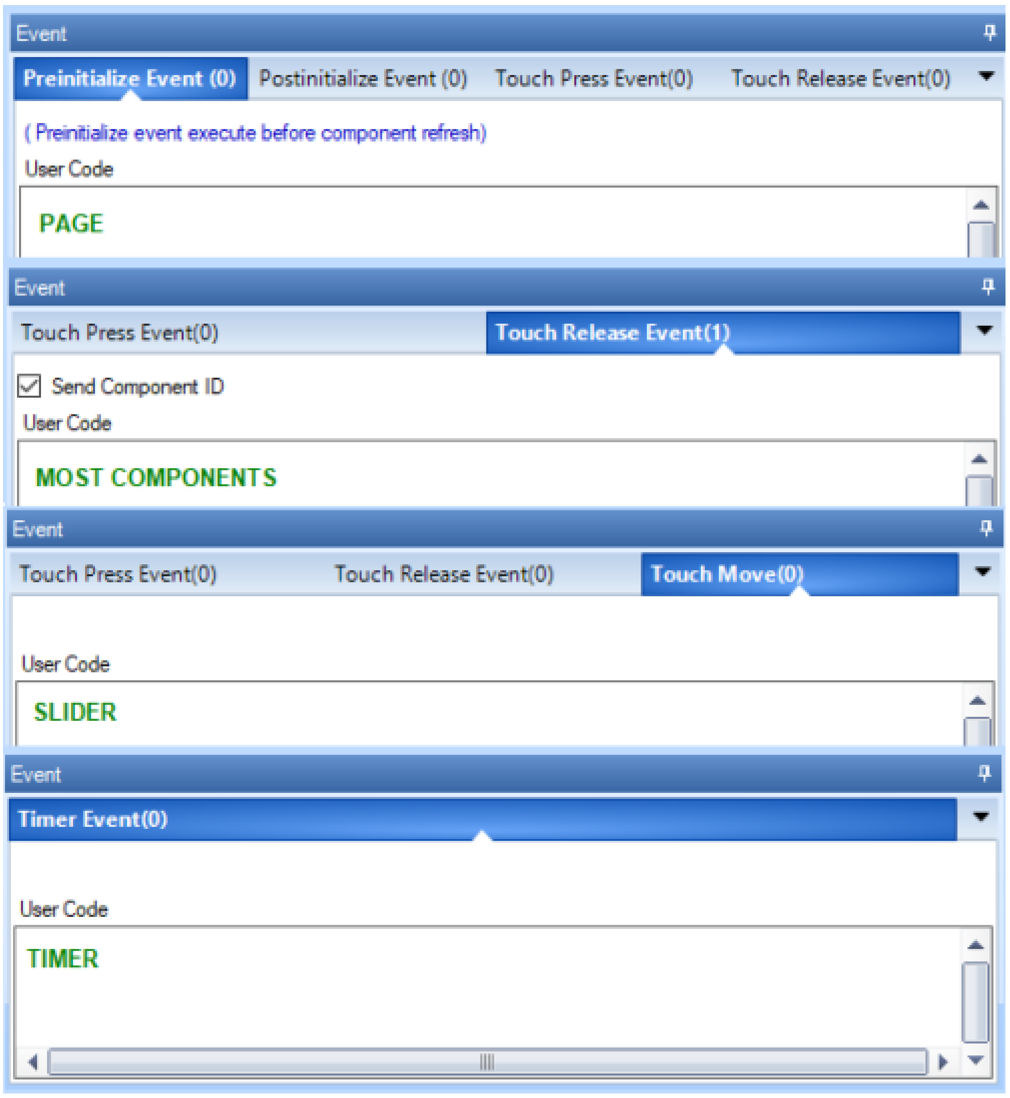
Page name prefixing is suggested to access a global component's attributes on another page.
– example: page0.va0.val
Page name prefixing is not required to access a local component's attribute on the current page
– example: va0.val

Attributes that have ranges are evaluated in full during Nextion's parsing of a complex expression and as such care is required. Nextion is stated as simplex expression, although these rules are often bent. Use care.
– example: gauge z0 with a .val range of 0 to 360.
z0.val=va0.val+step.val%360
Should va0.val+step.val exceed 360 the assignment fails before arriving at modulo 360.
– this is the nature of simplex expressions (think assembly language), and bending the rules has side effects.

The various combinations of attribute choices provides a wide range of expected behaviours with too many combinations to cover in any manual(s). This combined with the Nextion Instruction Set creates the opportunity for very powerful HMIs.

There is a hard limit for a combined tally of attributes and user code of 65534.

10. User Event Code


Event

Preinitialize Event (0) | Postinitialize Event (0) | Touch Press Event(0) | Touch Release Event(0)

(Preinitialize event execute before component refresh)

User Code

PAGE

Event

Touch Press Event(0) | Touch Release Event(1)

☑ Send Component ID

User Code

MOST COMPONENTS

Event

Touch Press Event(0) | Touch Release Event(0) | Touch Move(0)

User Code

SLIDER

Event

Timer Event(0)

User Code

TIMER

User Event Code can contain any valid Nextion Instruction. This section will not teach programming, but will quickly give an overview of the various types of Events where inserting user code is available. Event code is always local to page and never global.

Almost every component has the ***Touch Press*** and ***Touch Release*** events.
– The Touch Press Event includes a Send Component ID checkbox that when checked sends the 0x65 Return Data over serial on physical press. User code is run on either physical press or via the click command.
– The Touch Release Event includes a Send Component ID checkbox that when checked sends the 0x65 Return Data over serial on physical release.   User code is run on either physical press or via the click command.
– The ***Send Component ID*** 0x65 Return Data over serial action can not be triggered by the click command. This is reserved for an end-user physical action received through the touch sensor. The click command will only trigger running the user event code.
– The Nextion Instruction printh command can simulate 0x65 Return Data (Actual or spoofed)

The page component contains both Touch Press and Touch Release as well as

– The ***Preinitialize Event*** user code is run before the loading of the HMI designed page.

– The ***Postinitalize Event*** user code is run after the loading of the HMI designed page.

Note: global values will persist, will local values return to their designed state.

The slider component contains b0th Touch Press and Touch Release as well as

– The ***Touch Move Event*** user code is run during drag of thumb when slider changes values.

– The Touch Move Event does not have a Send Component ID to avoid serial overflow.

The timer component only contains the ***Timer Event*** for user code.

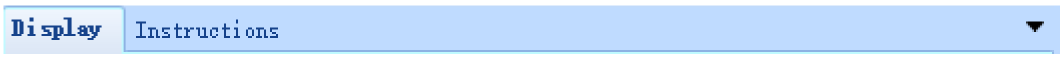
– Refer to the Timer component in the Component Pane section for more details

Nextion Return Data is returned at the end of command execution

– it would otherwise not be wise to predict an outcome before the end.

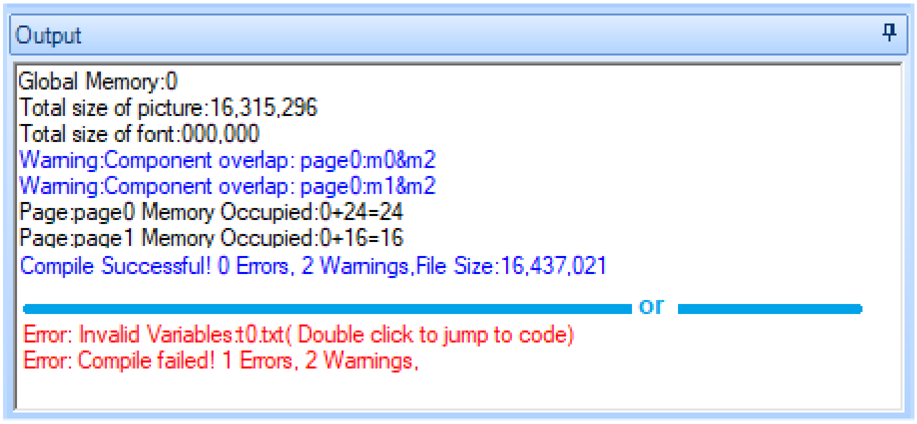
There is a hard limit for a combined tally of attributes and user code of 65534.

11. Display/Instruction Tabs



Selecting the ***Instruction*** Tab will load the Nextion Instruction Set in the Nextion Editor's built in web browser. (Internet connection is required). Selecting the ***Display*** tab will take you back to the Nextion Editor Main Interface.

12. Output



```
Global Memory:0
Total size of picture:16,315,296
Total size of font:000,000
Warning:Component overlap: page0:m0&m2
Warning:Component overlap: page0:m1&m2
Page:page0 Memory Occupied:0+24=24
Page:page1 Memory Occupied:0+16=16
Compile Successful! 0 Errors, 2 Warnings,File Size:16,437,021
```

```
Error: Invalid Variables:t0.txt( Double click to jump to code)
Error: Compile failed! 1 Errors, 2 Warnings,
```

The Output Pane contains details on the build process when Compile/Debug/Upload is selected. Compile needing to occur first, the user HMI is assembled into a usable TFT file for the selected Nextion Model. The first three lines of the output will list the total amount of global SRAM memory consumed by the HMI project, and then statistics for the total amount of Flash space the picture resources consumes, followed by the total amount of Flash space the ZI Font resources consumes.

The build process then goes through the project sequentially page by page. At the end of a successful page build, the page Memory stats are listed Global+Local=Total. Should a page not build successfully, the offending page is the last listed+1.

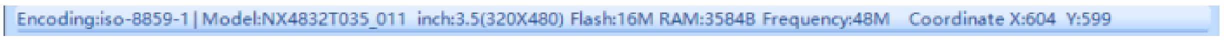
Warnings listed in blue (such as when using the not recommended layering techniques, it will compile, but warn of potential unexpected behaviours), Errors listed in red (this will not compile, and the build process halts). Note: Do not upload a zero byte file.

File Size must be small enough to fit in your Model's Flash size. See the Status Bar or your Datasheet for your model's Flash size. All pages' Total Memory usage must be small enough to fit your model's HMI allotted SRAM. See the Status Bar or your Datasheet for your model's HMI allotted SRAM.

Due to the nature of flash, it is possible that a compiled filesize may be under the MB size (ie: 1677216 bytes) and still be shy of the available and usable Flash on the Nextion Device. Nextion may report on upload the File is too big – this is not a hardware error but the working nature of Flash. Some allowance for unusable flash pages have to made, and even more over time.

Successfully compiled Project's *.tft file is in Nextion Editor/bianyi folder.

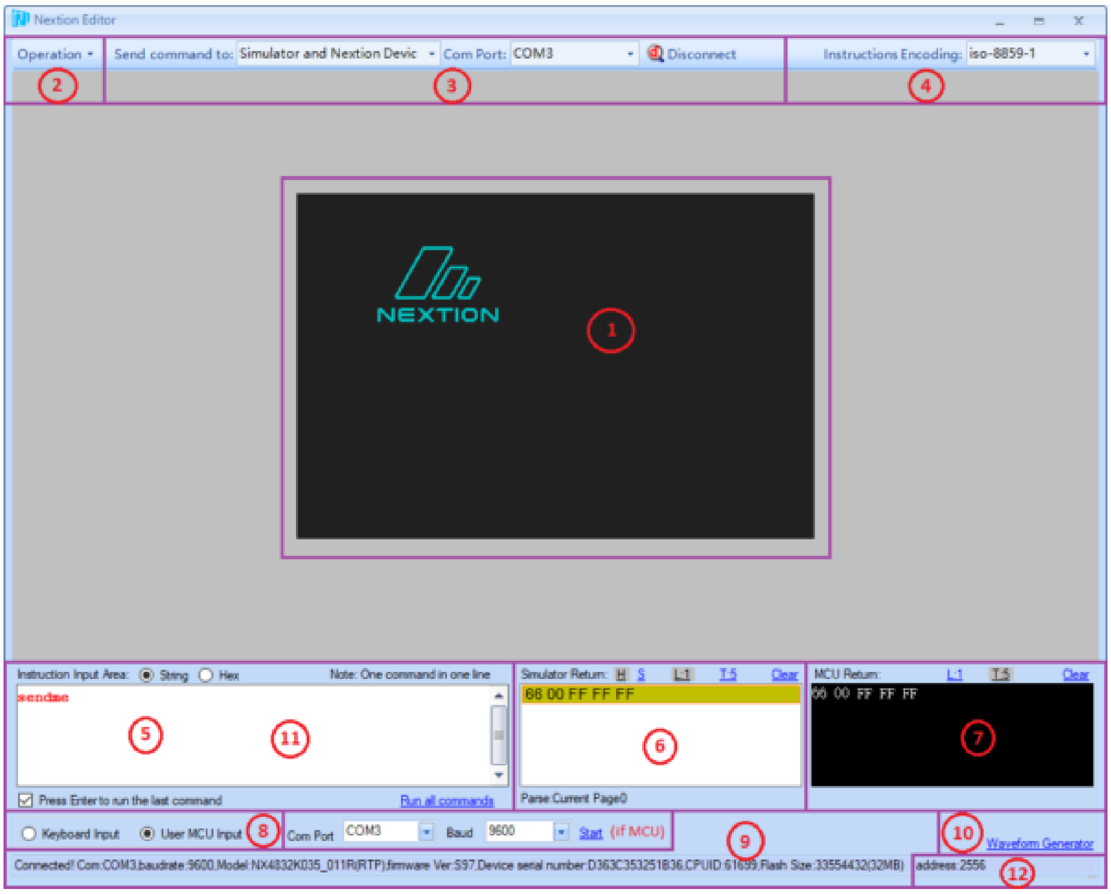
13. Status Bar



Encoding:iso-8859-1 | Model:NX4832T035_011 inch:3.5(320X480) Flash:16M RAM:3584B Frequency:48M   Coordinate X:604 Y:599

The Status Bar at the bottom contains three segments. The character encoding, the quick details of the Nextion Model selected, and the Design Canvas mouse coordinates. Clicking on ***Encoding*** will launch the DISPLAY tab of Device Settings. The Nextion Model details provide quick design specifications – limits of Project SRAM and Flash is useful to be mindful of.

Debug / Simulator Overview



```
sendme
```

```
66 00 FF FF FF
```

```
66 00 FF FF FF
```

```
Connected! Com:COM3,baudrate:9600,Model:NX4832K035_011R(RTP),firmware Ver:597,Device serial number:D363C353251836,CPUID:61699,Flash Size:33554432(32MB)
address:2556
```

1. Simulated Nextion … <goto>
2. Operations Menu … <goto>
3. Command Redirection … <goto>
4. Input Encoding … <goto>
5. Instruction Input … <goto>
6. Simulator Return … <goto>
7. User MCU Return … <goto>
8. Input Selection … <goto>
9. Connected Nextion Identification … <goto>
10. Waveform Generator … <goto>
11. Waveform Generator Options … <goto>
12. Nextion Address … <goto>

1. Simulated Nextion

- This is a simulation – this is not an emulation.
- Purpose of the simulation is to aid in debugging commands
- Timers will not be precision under any Windows OS
- RTC is simulated, GPIO is not simulated
- EEPROM is simulated with file located Nextion Editor/eeprom.bin

- Note: EEPROM adheres to 4 byte boundaries when writing
  Users may write to any address, but must care for data until boundary.

2. Operations Menu

- Upload, Synchronize RTC time and Reboot device.

3. Command Redirection

- Simulator only, Nextion device only, or both.
- Only one (Nextion device or User MCU) can be connected at a time
  When one is selected, the other is not available.

4. Input Encoding

- Character Encoding to use for input

5. Instruction Input

- text command input of any non-block Instructions (excludes if, for and while)
- Toggle hex mode to ensure byte precision in input.
  (Answers the question of how to input less frequently used characters)

6. Simulator Return

- Any Return Data or print commands from Simulator will display here
- Expected to be sufficient to aid in the debugging process
- Not expected to be full trace logs (Simulator – not emulator)
- Options added in v0.53 of the Nextion Editor
  View in Hex or String format
  View Line by Line (ÿÿÿ terminations), or by Total bytes in Hex
  Context Menu (right click) to: Copy HEX, Copy STR, Copy HEX+STR

7. User MCU Return

- Any Return Data or print commands from Nextion/MCU will display here
- Expected to be sufficient to aid in the debugging process
- Not expected to be full trace logs (Simulator – not emulator)
- Options added in v0.53 of the Nextion Editor
  View in Hex or String format
  View Line by Line (ÿÿÿ terminations), or by Total bytes in Hex
  Context Menu (right click) to: Copy HEX, Copy STR, Copy HEX+STR

8. Input Selection

- Keyboard input is the default simulator input method.
- Toggle to User MCU input to have user MCU interact with Simulator
  Select Com Port of user MCU, MCU operating baud rate and then Start
- Only one (Nextion device or User MCU) can be connected at a time
  When one is selected, the other is not available.
- Cut, Copy, Paste and Delete added to right click context menu.

9. Connected Nextion Identification

- When Nextion is connected to Simulator, portions of connect string is shown here.

10. Waveform Generator

- Click here to show the Waveform Generator

11. Waveform Generator Options

- Sinewave or random waveform input at selected intervals.
- Expected to be sufficient to aid in the debugging process
- Not expected to be an oscilloscope (Simulator – not emulator)

12. Nextion Address

- Default is 0 – Nextion normal mode, 256 to 2815 in address mode.
- Address will only apply to specialized advanced applications.