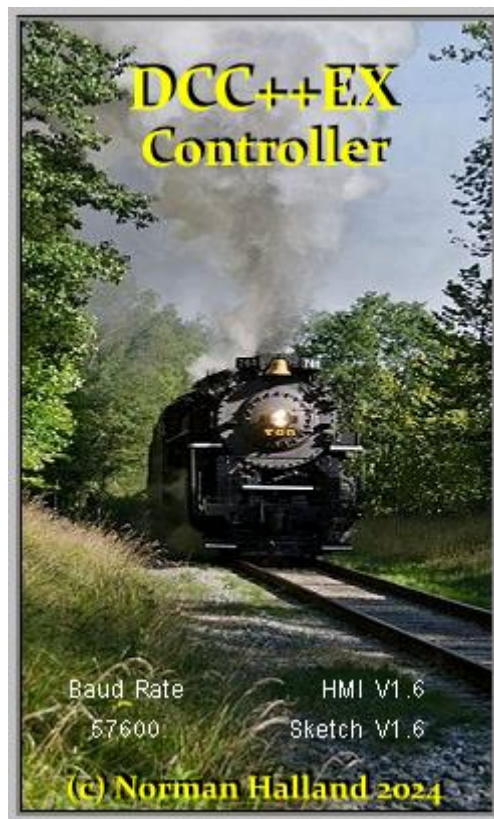


Nextion Based DCC++EX Controller



User Guide V1.6

Foreword and Credits

The idea for the Nextion based DCC++EX Controller came from Dave Bodnar's original Nextion Throttle. That implementation used an Arduino UNO as its processor and a Nextion 3.2in Basic display. Dave's concept spawned the desire to expand on what he had done, and subsequently grew to be "More than a Throttle" – hence the description as a "Controller".

As development progressed, the resources of the "humble" Arduino UNO reached a limit, thus needing to split the development into two closely related, but different devices – An UNO only "MK1" version, and an expanded "MK2". The latter would have the capacity to control a greater number of devices, add Wireless connectivity, facilitate future development, and yet retain a similar interface.

No project gets completed without the help of others. This one is no exception. What started for me as an opportunity to see if I could finally get to understand a little "C" and maybe "C++", has grown to be quite a complex endeavour – for me at least ☺. The help and encouragement from others has been a major motivator – their contribution cannot go un-mentioned. Here are some of them:

Dave Bodnar of course – his Nextion Throttle for an UNO was what got me going

Steve Feldman (esfeld) – probably the earliest adopter pre-2016?

Mike Dunston (atani) – exposed me to the error of my ways and the mysteries of C++ and OOP. I wasn't a good learner ☺

Francisco (jbsorocoba) – a Brazilian with whom we shared many discussions and thoughts – especially while using "Google Translate"

Erik Arckens (erik84750) – meeting someone over the 'net seems remote, but we seem to have known each other for years. Erik's enthusiasm and contribution especially the 3D printable enclosures demonstrate his invaluable involvement.

Did I mention Steve? From beginning to present, Steve always greets us (after his first?) cup of morning coffee and is there 'till late. Always new ideas, and he's already disclosed some new ones. New contributors Lee and Keith have already shown me many of the considerations needed when sharing a project. I look forward to many more hours testing and suggesting. It's been a ton of fun ☺

Last, but certainly not least is the selfless work done by the DCC-EX Team and in particular, Gregg Berman!

I'm sure all the guys will agree with me when I say I hope you enjoy using the Controller as much as we have had, while developing it!

Norm Halland

Table of Contents

<u>Section</u>	<u>Page</u>
Foreword	2
Introduction and Versions	4
Features	5
The Controller Case	6
Getting Started	7
Setting Up Arduino IDE	7
Preparing the Nextion	8
Individual Page Details	10
Cover	10
Menu	10
Throttle	11
Loco Inventory	12
Edit Loco	12
Edit Function	13
Accessories	14
Edit Accessories	15
Routes	16
Configuration	17
WiFi Status	18
Programming	19
Updating	20
Additional Pages	21
Appendix	22
Utility Programs and Libraries	22
- UNO Upload Utility	22
- Nextion Upload Utility	23
- NextionUpload Library	23
- ArduinoOTA Library	23
Pin Assignments and Connections	24
- UNO	24
- ESP32 Pin-out Assignments	25
- ESP8266 Pin-out Assignments	26
- RaspBerry Pi Pico and Pico W	27
Fault Reporting	28

Introduction and Versions

The design goal for the DCC++EX Controller was to use the concept of the modified version of Dave Bodnar's initial Throttle and add features which would make it more of an interface to the DCC++ and DCC-EX Command Stations.

If you're new to either this, or Dave's project, you might want to refer to the suppliers of the Nextion range of "HMI" displays:- <https://nextion.tech/> A download of the free-to-use Nextion Editor will give access to extensive reference material and is an excellent information source.

In summary, the Nextion Displays are a range of "Intelligent" LCD-TFT displays which have their own built in graphics processor and firmware, and are supported by a very useful design tool. Good news is that they are 3.3V and 5V compatible. You'll soon be familiar with two of their file types:- An HMI file (FileName.HMI) is the Nextion Source File, and gets compiled by the Nextion Editor. A TFT file (FileName.TFT) is the compiled HMI file and needs to be flashed to the Nextion itself.

Whilst the 3.2in Basic Nextion display had been the visual and operational component of Dave's early device, its limited size prompted the expansion of support to include versions up to the 5.0in model. This opened the door to ideas of supporting a Roster, a library of accessories, and perhaps some very simple "Routes" or accessory state setting lists. Very importantly, the ability to edit, delete and add devices without having to configure and compile a new device every time, was considered vital.

To keep things simple, the initial idea was to avoid any WiFi or wireless connections, and to keep the device as a "hard-wired"/directly-attached input device. Using a pair of HC12s was considered an acceptable exception.

Playing with an ESP32 changed all that, but the original idea remained. The development path progressed while maintaining a "One-Size-Fits-All" Arduino software sketch which could be compiled for anything from an Arduino Pro-mini, to an ESP32 Dev Module. Features could be enabled or disabled depending on the microcontroller used, while at the same time maintaining the same identical HMI Nextion interface for 4 of the Nextion sizes – 3.2, 3.5, 4.3 and 5.0in, and both Basic and Enhanced options.

Just before version 1.3.0 of the project, we reached what I consider was the peak of that philosophy. We were finding too many limitations were being imposed on both the UNO and ESP32 versions, simply to keep the "One-Size-Fits-All" thinking alive.

The decision to have two development paths was made, so the "UNO-Only" or MK1 was capped with the features available at the time – even the Nextion HMIs received their own version numbers, while all further development continued for the more advanced microcontrollers. This has now been dubbed the MK2 range. Currently the supported processors for the MK2 are ESP8266, ESP32, Raspberry Pi Pico, and Pico W. Others are no doubt possible, but not investigated as yet.

The future? Steve is already brimming with ideas so we might need to tie him down... But seriously, a very early desire I had was to make the Controller use a bi-directional communication link with the Command Station. In other words and as an example, when initiating a command for a Loco on the controller, the visual display should only get updated when the Command Station acknowledges receipt and execution of that instruction. At the time the code to achieve this was well above my pay grade – so was left for Version 2.0... (and DCCExProtocol?)

Features of the MK1 and MK2

Their common heritage has facilitated a smooth transition from the MK1 to the MK2. Common elements are the Nextion HMI Display, Enclosure, Printed Circuit Board, and the rechargeable Battery Pack. Header sockets are suggested to allow for the easy replacement of the processor – an Arduino Pro-mini as the supported UNO variant, or an ESP8266 mounted on a carrier PCB, or one of the two variants of the ubiquitous ESP32 Dev Module.

Apart from the Basic 3.5in Nextion, all Basic and Enhanced 3.2, 4.3 and 5.0in Nextions can be used – only the relevant Enclosure size differs.

We currently have users of both MK1 and MK2 Controllers – the MK1 being adequate for a small layout where untethered walk-around is not required. Wireless connection can easily be added by adding two HC12 serial communication boards – one connected to DCC++ or DCC-EX and one to the Controller.

If you already have an ESP32 Dev Module, the decision is an easy one – go for the MK2☺. Here are the major specifications of both variants:-

Features of the “UNO” based MK1

- Connection to DCC++ or DCC-EX via direct Serial “tether”, or a pair of HC12 serial extenders
- Uses an Arduino UNO, Nano, or Pro-Mini processor (Pro-mini required for Erik’s PCB)
- Supports Nextion 3.2, 3.5, 4.3 and 5.0in Basic or Enhanced display models
- Battery operation in an available 3D printable Enclosure (See the “Case for the Controller)
- A “Roster” or “Inventory” with a capacity of 10 Locos, each of which:
 - Can have its own Name, Type, Road Number and Address
 - Can have up to 10 Functions selected from the full range supported by the Command Station
 - Images for Functions can be Traditional “Fx” buttons or a choice of Graphic pictures
 - All Functions support Pulsed or Latched operation
- The 10 Loco Slots can be loaded with any of the locos from the Roster and in any vacant position thus representing a page of up to 10 Favourite Locos
- Facility is made for a “Guest” loco – a quick way to control a non-Roster-resident Loco
- All ten Loco slots can have active locos – their Speed, Direction and Function states are maintained and instantly available.
- Up to 36 Accessories (Turnouts) with individual images, names and addresses are supported and their location can be spread over 3 “pages” containing up to 12 each
- Up to 6 rudimentary Routes each containing up to 6 accessories can be defined
- Programming on the Program or Main track

Features of the MK2 Controller

- Includes all the MK1 Features
- Support for ESP32, ESP8266, and Raspberry Pi Pico W (Plain Pico also supported without WiFi). UNO is NOT supported.
- WiFi Connectivity in addition to Direct Connection ability
- Interactive configuration of Network and DCC-EX connection credentials
- Arduino IDE based OTA upload for the ESP32, ESP8266 or Pico W
- Remote wireless Nextion firmware upgrading via an HTTP server
- Up to 50 Locos in the Roster each with their own Name, Type, Road Number, Address and set of up to 10 configurable Functions

- Up to 96 Accessories (Turnouts) each with their own image, name and address
- Up to 48 rudimentary Routes containing up to 6 accessories each
- Enhanced Programming features

A Case for the Controller

Erik Arckens (erik84750) has designed and shared his 3D printing files of a range of Enclosures for the DCC++EX Controller and its Nextion Display. Four unique designs are available for the 3.2, 3.5, 4.3 and 5.0in Nextion models.

These designs, together with component requirements and assembly instructions, are available on Erik's GitHub site:-

https://github.com/Erik84750/Nextion_DCC-EX_enclosure

and a PCB supporting a range of Micro Controllers:-

https://github.com/Erik84750/Nextion_DCC-EX_master-pcb

The PCB is optional, but a handy way to keep everything mounted and organised.

If you choose to use the ESP8266 on a "Carrier" board, the PCB is required to provide the needed 3.3V for the ESP.

Other boards such as an Arduino Nano, or the Pico and Pico W will only require a limited amount of wiring to up. No PCB necessary.

Getting Started

Things you'll need:-

1. A Nextion Display (3.2, 3.5, 4.3 and 5.0in) – MK1 and MK2
 - All Enhanced models are supported, and all Basic versions except the 3.5in with the MK2 due to memory constraints
 - Nextion “TFT” files for 3.2, 3.5, 4.3 and 5.0in sizes are available (see Downloads in the Appendix)
2. Processor of your choice.
 - MK1:-
 - Any UNO Variant – Pro-mini required for Erik’s PCB
 - MK2:-
 - ESP32 Dev Modules offer an excellent price/feature package
 - The Raspberry Pi Pico W is hot on the heels of the ESP32
 - ESP8266 is catered for, and well suited for WiFi connection only – Direct Connection uses Software Serial which has proved to be unpredictable.
3. Enclosure and associated components to suit your choice of Nextion. Have I mentioned Erik’s version is an excellent choice😊?
4. A Computer running the Arduino IDE. Refer to the Appendix for setup considerations and requirements

First thing to do after ensuring your Computer is set up properly, is to download the Arduino Sketch, Libraries, Utility Programs and ready built Nextion images (.tfts).

All are available for download on my GitHub Repository:

<https://github.com/normhal/DCCppEX-Nextion-Controller>

Preparing the Arduino Sketch

Unzip the downloaded compressed file and copy the “DCC-EX_Controller_Vx.x” folder to your Arduino Sketch location.

Make a copy of the Credentials_Example.h file and rename it to “Credentials.h”. Complete the details for your particular Network and DCC-EX configuration, and keep another copy for safe keeping. Subsequent bug-fix or New Feature releases will be able to use your Credentials.h without over-writing them.

If desired, do the same for the “Hard_Coded_Values_Example.h” file. Apart from being an example of how to pre-load the Controller, it is a convenient way to keep a backup of your Controller configuration. Its use is not absolutely necessary however.

If you have configured your Arduino environment as per the details in the Appendix, the compilation should complete and you should be able to flash your choice of microcontroller.

The next step is to copy the Nextion Image file (Filename.TFT) to the Nextion itself.

Preparing the Nextion

Using Erik's beautifully designed enclosure is a Great way to keep everything together, however I recommend you do a test "dry-run" with the Microcontroller board connected directly to the Nextion to get familiar with the components and make sure everything works☺

For a new installation there are three possible methods to configure the Nextion:-

1. Uploading with the Nextion Editor

- Instructions can be found in the Nextion Editor Help text

I strongly recommend you download and install the Nextion Editor even if you choose not to use this option. The Editor provides a very comprehensive Help system and Reference for everything Nextion☺

2. Using an SD Card

- Copy the correct TFT file for your Nextion model to a blank, formatted SD card and insert it in the Nextion before powering it on.
- Wait for the upload to complete then remove the SD card and power

3. Using the "Nextion Upload Utility"

- The Utility is provided to enable an ESP32 to download and flash the Nextion TFT image without having to remove the display from an enclosure
- A free to use HTTP Server called "HFS" is included in the download or can preferably be downloaded from its original location below
<https://www.rejetto.com/hfs/download>
- Follow the instructions to install HFS and run it. You will need to give the program access rights to your network
- Once it is running, make a note of its IP Address displayed at the top of the Window as well as the Port used to connect to it.
- Install two Libraries in your Arduino IDE:
 - ArduinoOTA – Install from Library Manager
 - NextionUpload – Install the zip file again with the Library Manager
- Edit the NextionUploadUtility sketch before you compile it.
 - Complete the required Network Credentials, IP address and Port of the HFS Server
- Connect the Nextion to the Microcontroller
For the ESP32:-
 - Red to 5V
 - Black to GND
 - Blue(TX) to GPIO16(RX)
 - Yellow(RX) to GPIO17(TX)

For the ESP8266:-

- Red to 5V
- Black to GND
- Blue(TX) to GPIO3 (RX)
- Yellow(X) to GPIO1(TX)

For the Pico W:-

- Red to 5V
- Black to GND
- Blue(TX) to GPIO8(RX)
- Yellow(RX) to GPIO9(TX)

As mentioned previously, the Nextions are 3.3v and 5v compatible, so no need for level shifters.

- It helps to open a Debug Console window before compiling and running the NextionUploadUtility. The upload progress can thus be observed☺
- Compile and Flash the Sketch

The sketch will cause the Microcontroller to connect to your network, locate the HFS server, search for the “.tft” file you entered in the sketch, and start the download process. The Progress can be monitored in the HFS window as well as on the Debug Console and takes around 10 minutes to complete.

If all goes well, after the upload is complete the Nextion will reset and power up. You should see the DCC++EX Controller’s Cover page for the first time. By touching anywhere on the display, the displayed image will change to the Main Menu. Without the Microcontroller attached, four of the Menu buttons are “alive” :-

- Version – will present the Cover Page
- Info – will give a short background of the Controller Project
- Tips – provides a few handy “un-documented” features☺
- Credits – lists all those who have played a part in the development of the Controller. Without their questions, suggestions, ideas and testing, the Project would not be where it is today!

The next step is to compile and flash the Arduino Sketch for the Controller software itself. The two libraries mentioned above are the only libraries required to complete the build.

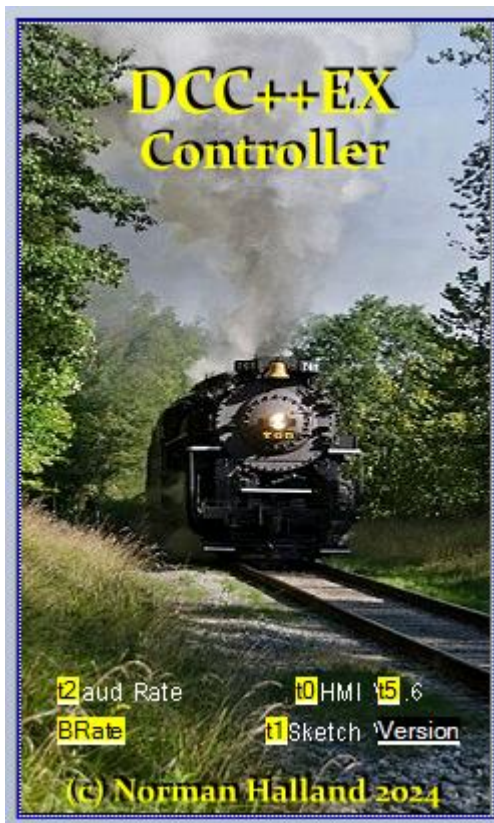
Before starting the compilation, refer to the Appendix for specific Microcontroller settings. Also select the “Com” port your computer assigned to the microcontroller board, and I suggest you open the Debug Console in order to view the startup messages once the Compiling and Flashing is done. Baud rate for the console is 115200.

Fill in your WiFi credentials in your copy of the “Credentials.h” file. Even though these details can be entered and edited via the Nextion, it’s more convenient to do so up front.

If all goes according to plan, the Controller will start up and you can observe the start up process on the Debug Console. Once started, the Menu Page becomes “Live” and all the buttons are active.

The next section details each of the “Pages” which can be accessed from the Main Menu as well as how to navigate around the Controller.

Cover Page



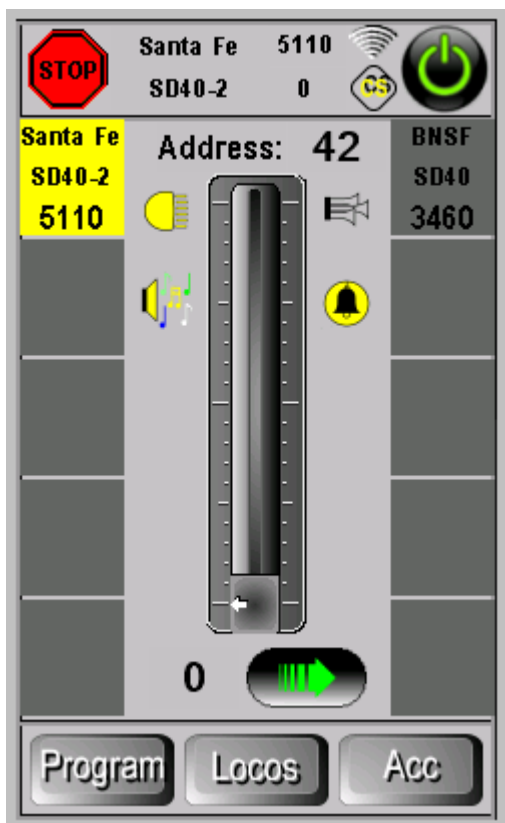
- Displayed for a few seconds at every power-on
 - Shows the Nextion Firmware and Sketch Versions
 - Under normal circumstances the two version numbers must match
- The configured Baud Rate used to communicate with the Controller is displayed for information and possible fault finding
- If the Controller does not automatically proceed to either the Menu or the WiFi Status Page it indicates a fault condition.
- Touching anywhere on the Cover will force the Menu Page to be displayed.
- The “Version” button on the Main Menu will display the Cover image if required

Main Menu Page



- The “Stop” and “Power” buttons appear on all functional pages
- The Stop Button can be configured to Stop either only the currently active Loco, or all 10 favourites on the Throttle Page (See Config Page)
- The Power Button turns power On or Off. The “On” state can issue either a Program and Main Track “Join” command, or simply a Global “On” (Refer the DCC-EX documentation and the Config Page)
- The WiFi signal strength Icon is updated every (programmable) few seconds (See Config Page) together with the DCC-EX Connection Status
- The bulk of the remaining buttons are simply “Navigation” images to access all the features of the Controller.

The Main Throttle Page



The Top section of the Throttle Page provides:-

- A common Top row with the “Stop” and “Power” buttons as on all Pages
 - Press the Red “Stop” to Stop either a **Single** or **All** running Locos (**Config Option**)
 - Press the Green “Go” to resume the speed for All Locos previously “Stopped”
 - Pressing the Red “Power” button will turn track power OFF
 - Pressing the Green “Power” button turns power ON (refer Config Page)
- The WiFi and DCC-EX status icons show
 - Continuous WiFi Strength
 - DCC-EX Connection Status
 - Update interval is a Config Page option
- The Name, Road Number, Type, and current Speed of the currently active Loco (Highlighted in Yellow).

If the “Hard Coded Values” option is compiled, first go to the “Config” Page and press the “Load” button. This will copy the values you have edited in the “Hard_Coded_Values.h” into the Controller. This can save you the trouble of manually entering and customising your preferences.

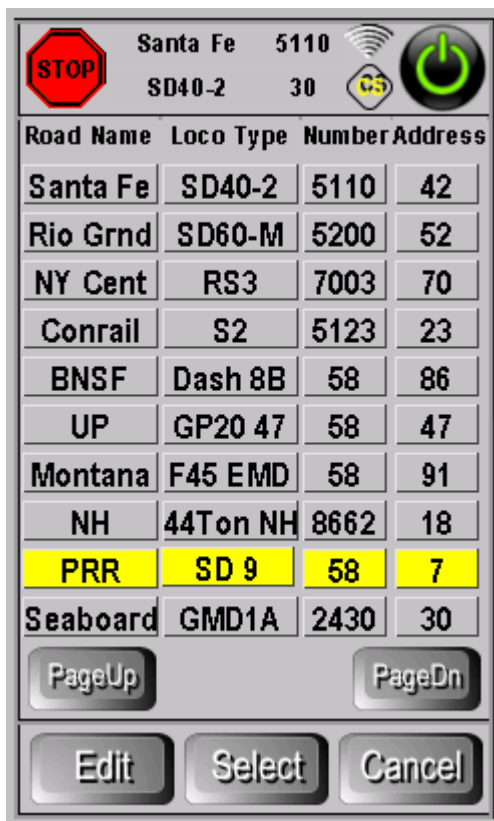
Below the Top section you have the “Throttle” itself:-

- 5 Loco “Slot” tabs on the Left and 5 on the Right – Active Tab in Yellow. The example above shows two configured Locos, a Santa Fe SD40-2 in Slot 0 and a BNSF SD40 in Slot 5.
- Each Loco “Slot” has 10 Function “Slots” – 5 on each side of the Throttle Slider. The example shows the active Loco’s functions in slots 0 and 1, as well as 5 and 6.
- The Direction button has the option to set the speed to zero above the value set in the “Config” page.
- Pressing the active loco’s actual address allows you to enter a “Guest” loco address for temporary control.
- Pressing the “Address” text takes you to the “Edit Loco” Page (see below)

The lowest section allows you to instantly jump to the Program, Loco Inventory or Accessory Pages. The currently active Loco’s details will be displayed in the top section of those pages. When you have used any of the three Pages above, you will be automatically returned to the “Throttle” Page.

The Main Menu can be accessed by touching the middle of the Top section.

The Loco Inventory



The MK1 Controller only has 1 page of up to 10 Locos, while the MK2 has 5 pages and up to 50.

Page Up and Page Down buttons allow navigation through the MK2's pages

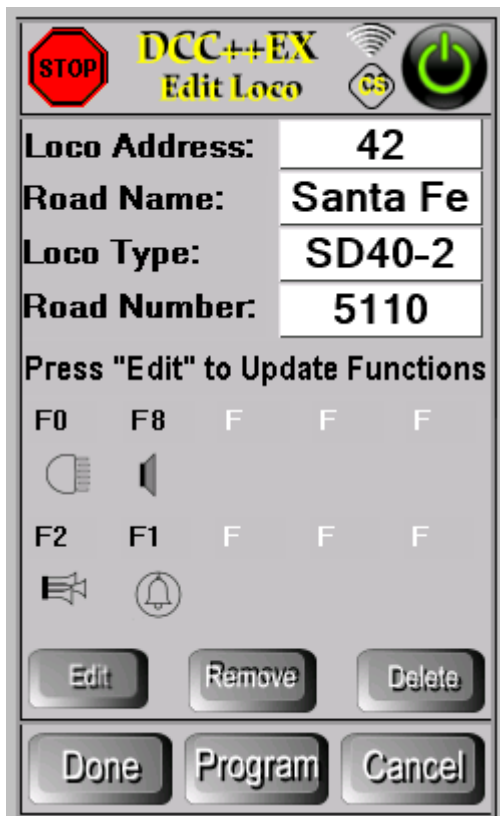
The example shows the Santa Fe is the currently active Loco (displayed in the top central section) and the PRR Loco Row has been Touched (Highlighted in Yellow).

It will replace the Santa Fe in the Throttle Page and become the new Active Loco if the "Select" button is pressed,

The "Edit" button will take you to the "Edit Loco" Page to edit the PRR Loco details and you will be returned to the Inventory Page when done editing.

The MK2 Controller allows for Loco "Types" to be Grouped on different inventory pages

Edit Loco Page



The Edit Loco Page can be called from **either**

- The main Throttle Page (by pressing the "Address" text **or**
- The Loco Roster Page by highlighting the Loco row and pressing the Edit Button

Access to all the Loco details is provided by this page:

- Loco Address (except 0)
- Loco Name
- Loco Type
- Loco Road Number
- Up to 10 Functions can be defined for each Loco. The example shows:
 - F0 must appear in function slot 0 and has an Image as its identifier
 - F8 appears in slot 2 with an image
 - F2 occupies slot 5 with an image
 - F1 also uses an image to show its purpose and will use slot 6

The Loco details on the Edit Loco page can all be edited. An address of “0” has the effect of deleting a loco, so take care (can be instantly recovered by entering a valid address). Note that there is a limit of 8 characters each for Loco Name and Type, and numerical values up to 65535 for Road Numbers.

Up to 10 Functions can be added, deleted or edited by first pressing the “Edit” button, and then pressing the required function slot. This opens the “Edit Function” Page (see below). Function numbers ranging from 0 to 68 are supported.

- Function Number buttons are available for functions 0 to 28. Each Button has two distinct images – one for Off and one for On.
- A limited number of Image pairs (On or Off) are available. More will become available – memory space permitting
- Each Function has the option of being either “Latched” On or Off, or “Pulsed” On when being touched.

Two additional buttons are the Remove and Delete emblems.

- The “Remove” button will remove the Loco being edited from the list of “Favourites” on the Throttle Page
- The “Delete” button totally deletes the Loco being edited and all its characteristics from the Roster

Pressing the “Done” navigation button saves all changes and returns you to the Page you came from (Throttle or Inventory/Roster).

Edit Function Page



The Usual Top and Bottom sections are presented, this time showing the Edited Locos details as reminders.

Note that the Edit Function Page can only be called from the Edit Loco Page.

In this example, the following details are shown:-

- Function 8 has been allocated for this slot
- An image for Sound has been chosen from the Function Images Page
- It is NOT a Pulsed function
- The function can be tested (Live) via the Test button

When “Done” is pressed with an Active “Delete” state the function’s details will be deleted, and an Inactive state will save them in the Loco’s record.

The “Cancel” button will abandon any selections or changes and return to the Edit Loco Page.

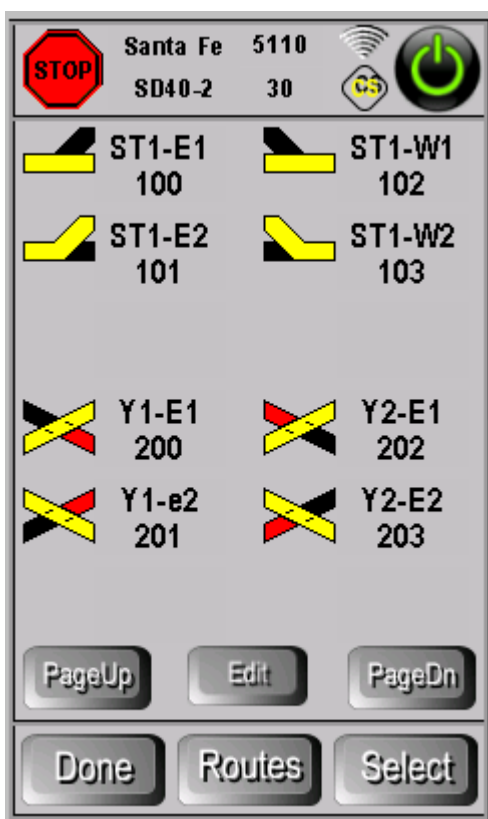
All editing relates to the Function “Slot” pressed on the Edit Loco Page. This allows you to place the Function you’re editing in any of the 10 Function positions on the Throttle.

The steps to add or edit a function as shown in the example are very straightforward:

- Enter the Function Number for the selected slot next to the “F” (0 to 68) - 8 in the example
- Press the Image buttons to open one of two pages – Function Number buttons, or Function Images. The example chose the image from the 2nd page of images
- Once the desired button image is selected (pressed) on the Image Page, you are returned to the Edit Function Page with the pair of function images displayed. The function can then be tested.
- If a “Pulsed” (horn) function is required, select that option as desired
- Pressing the “Delete” button prepares the “Done” button to remove the function you’re editing
- You can continue to edit and test until satisfied, then press the “Done” navigation button to return to the “Edit Loco” page with the new function in place.

When you return to the Edit Loco page, “Edit” mode stays active allowing for further Function additions or editing.

Accessories Page



This example shows a total of 8 accessories leaving 4 slots empty

Pressing anywhere in the area occupied by an accessory toggles its state

PageUp and PageDn buttons allow for scrolling through all the possible pages - 3 for MK1 and 8 for MK2

The “Edit” button is a Toggle button which allows the next accessory pressed to be edited via the Edit Accessory Page – automatically launched

The Navigation Buttons are self explanatory with the “Select” button only being relevant when creating “Routes”

As a reminder which Loco is active, its details are shown together with the usual “Stop” and “Power” buttons in the Page’s upper section, as well as the current Connection Status.

The Accessory Page can be called from three sources – the Main Menu, the Throttle, or the Routes Pages. As at Version 1.6 of the DCC++EX Controller software, only DCC accessories are supported. These are typically Dual-State devices – either On or Off, Closed or Thrown.

For the MK1 Controller, a total of 3 accessory pages are available for use, each with up to 12 individual accessories giving a total of 36.

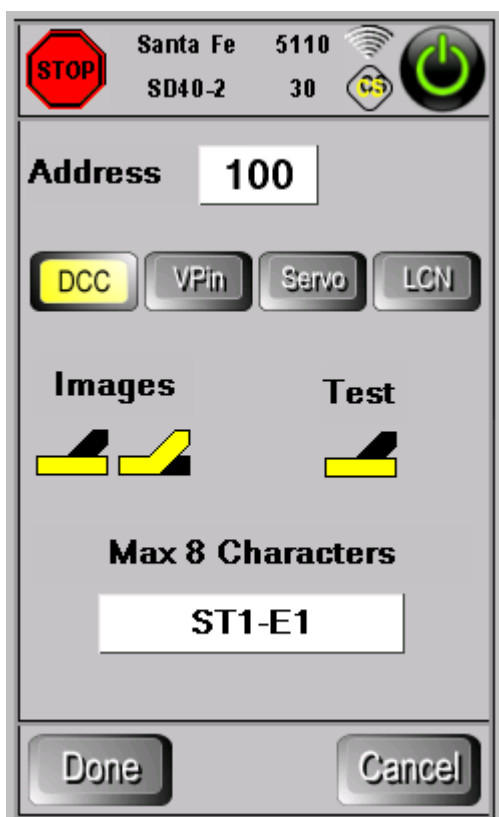
For the MK2 Controller, a total of 8 accessory pages are available, each with up to 12 individual accessories giving a total of 96. For convenience, accessories can be “grouped” in different pages if desired.

The “Hard_Coded_Values_Examples.h” file pre-loads a range of accessories as examples and can be edited to suit your requirements.

A typical Accessory has three editable characteristics – Name, Address and Image. As mentioned, Type is only included for a future Version.

To Edit or Add an accessory, first press the “Edit” button, then press one of the 12 accessory slots on the working area.

Edit Accessory Page



The now familiar Active Loco is displayed in the Top Section.

Accessory addresses can be any within the range supported by the Command Station

Images are available for Left and Right Hand turnouts in East-West, West-East, North-South and South-North orientations

Images are also available for other styles – Wye and Double Slip, also in the same choice of orientations

Up to 8 character Names can be defined. The example could be for “Station 1, East 1”

The “Test” button will physically activate the accessory as a confirmation when editing or creating.

The “Done” button will persist the change or addition.

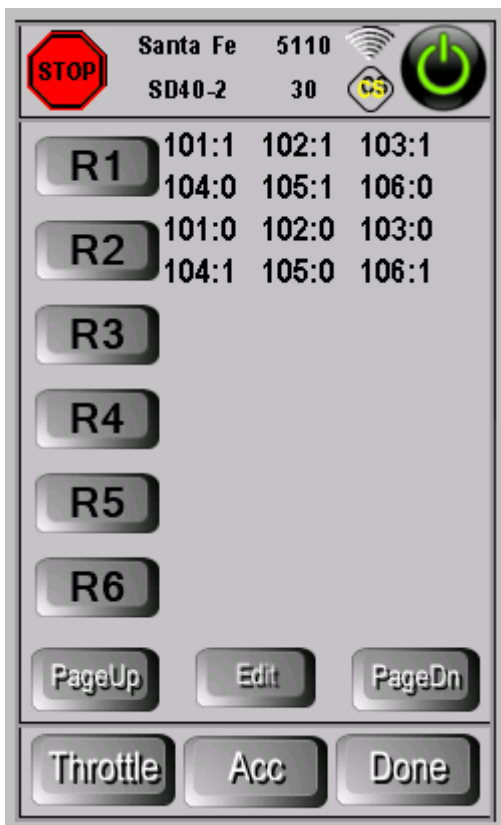
Can only be called from the Accessory Page and always relates to the Accessory Slot selected after pressing the “Edit” button.

To Create or Edit an accessory:

- Go to the Accessories Page☺
- Page Up or Down to the page containing the slot you wish to edit.

- Press the “Edit” button
- Press the Accessory Slot to Edit – This opens the Edit Accessory Page
- Enter the address in the standard DCC supported range first
- The “Type” defaults to “DCC”
- Press the area below the “Images:” heading
- Select an image for that Accessory
- A Name of up to 8 characters in length can be added
- Test if required – This is a “Live” test to give positive confirmation
- Press “Done” to save and navigate back to the Accessory Page

The Routes Page



The MK1 version supports 12 Routes and the MK2 can have up to 48.

Each Route can have up to 6 Accessory “States”

In this example, “Route 1” will set accessories with the addresses 101 to 106 to Thrown, Thrown, Thrown, Closed, Thrown, Closed respectively

“Route 2” will set the same 6 accessories to Closed, Closed, Closed, Thrown, Closed, Thrown

A delay between each activation can be set on the “Config Page”

Routes can be Activated, Created, or Deleted, but NOT edited

See Text description for Creation procedure

The DCC++EX Controller includes an elementary method to group up to 6 accessories to be set in a pre-defined state. The MK1 Controller supports 6 Routes (Each having up to 6 Accessory states) and the MK2 can have up to 48.

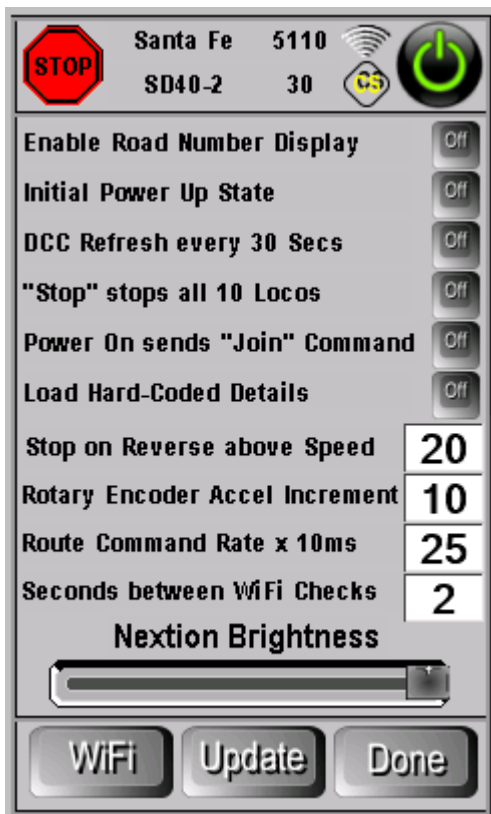
Creating a Route

Press the “Edit” button, then press the Route number to work with. This will open the “Accessory Page in a special “Route Creation” mode with the “Select” button activated.

- Set up to six accessories to be included in the Route to the state they must be set to when the Route is activated (Closed or Thrown).
- Select the required accessories by touching their Name or Address fields (The background will change to Yellow).
- Press the “Done” button to return to the Route Page with the new Route visible.
- De-Selecting the Select button resets the Route creation process and can be re-started.

In summary, creating a route is limited to selecting a route, setting the desired turnouts and their intended states, and pressing “Done”.

Config Page



A number of operational parameters can be changed from their compilation default and permanently saved.

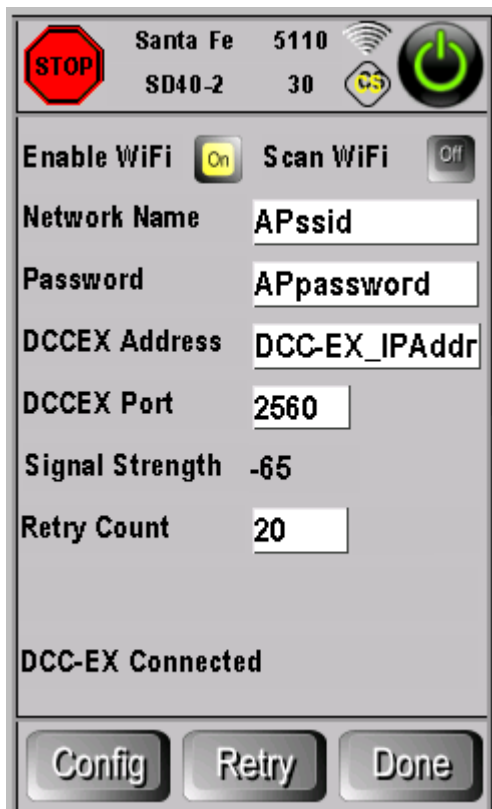
Most options are self-explanatory with some needing some clarification:-

- The Throttle Page Tabs can be set to display Road Number or Address values
- The Stop Button can stop (and resume) either the Active Loco only or All Favourites
- The “Stop on Reverse” value sets the speed below which the Active Loco will do an immediate direction change without stopping
- WiFi Status updates set to 5 seconds gives good results with minimal WiFi impact
- The “Update” navigation button opens the OTA and Nextion Upload page.
- The “Done” button writes all changes to permanent memory

In addition to the familiar Top and Bottom rows with their Navigation buttons, the “Config” page provides the ability to tailor Controller characteristics to personal preferences. The options are:

- Enable Road Number Display (default is Loco Address) on the Throttle Page
- Initial Power Up State (On or Off)
- DCC Refresh every 30 seconds (On or Off)
- Action taken by the “Stop” button
 - Default is to stop all 10 locos on the Throttle Page
 - Can be changed to Stop ONLY the active Loco
 - Pressing the “Go” image resumes speed and direction of one or all Locos
- Configuring the Power button to do a DCC-EX “Join” of Program and Mainline operation
- A Button to re-load predefined Loco, Accessory and Route settings from a “Hard_Coded_Values.h” file
- A Setting which sets the Loco speed above which a direction change sets the Loco speed to zero.
- The optional Rotary Encoder can be set to have an accelerated pulse rate to quickly increase or decrease speed if rotated rapidly
- Executing a route can have the rate at which accessory commands are sent – handy for some Capacitive discharge circuits
- A variable rate can be set for the Controller to monitor WiFi and DCC-EX connectivity states. A shorter time tends to interfere with Nextion data transmission and reception.
- The Nextion Brightness can be set and saved with this last setting

WiFi Status Page



This is the main WiFi configuration and Connection Page.

The example shows:-

- WiFi is Enabled
- A scan for local SSIDs can be initiated
- The Access Point SSID (Either your Network or DCC-EX)
- And its password
- Unless changed in DCC_EX, always 2560
- The selected Network's Signal Strength
- Number of retries before timing out
- Two lines of Progress and Status information

Note:

To minimise disruption to the current connection, the Status Information relates to the present connection activity and is not always updated when this page is initially called!

The Network Credentials presented on the WiFi Status page can either be specified in your own "Credentials.h" file or manually entered or edited on this page. A differentiation between a DCC-EX Command Station operating in STA and one in AP mode should be noted here.

For a DCC-EX Command Station operating in STA mode complete as follows:-

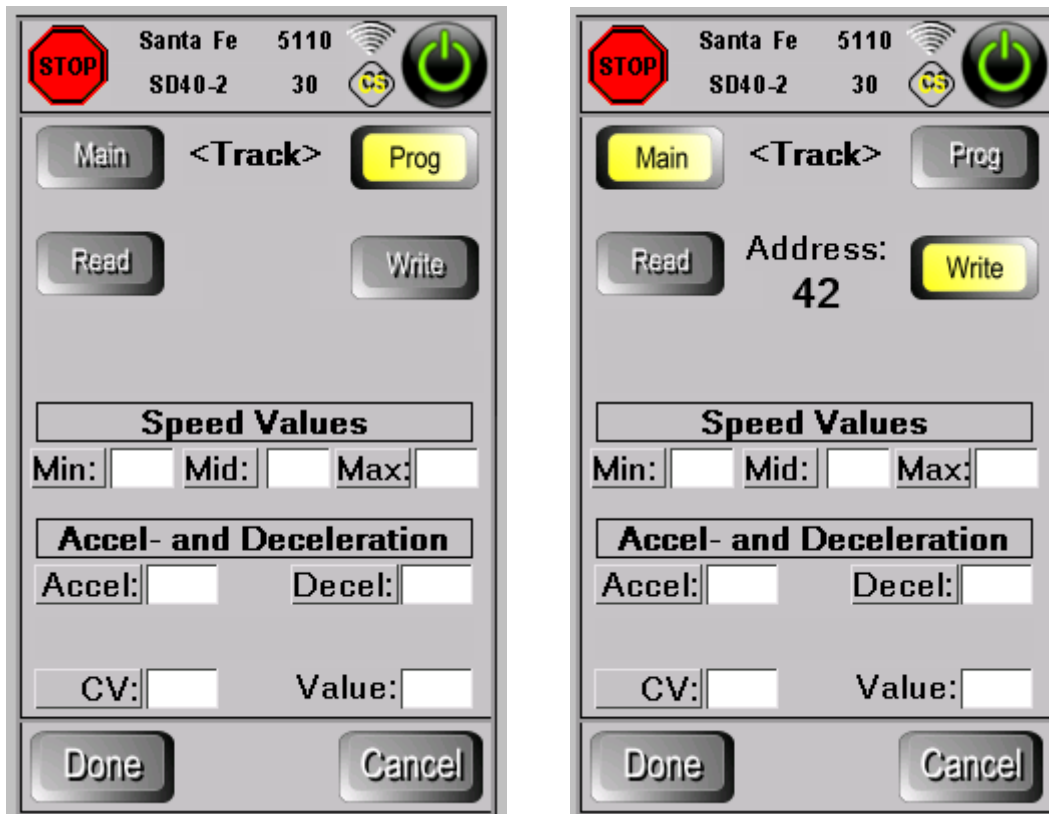
- The SSID is that of the Network the DCC-EX Command Station has connected to
- The Password to Connect to that network
- The IP Address given to the DCC-EX Command Station by the Network, or its Static IP address if so configured
- The Port used by DCC-EX – usually 2560

For a DCC-EX Command Station operating in AP mode:-

- The SSID is the name of the DCC-EX Command Station on the Network. The default is DCC-EX_XXXXX. Refer to the DCC-EX documentation
- The Password used to connect to DCC-EX. Refer to the DCC-EX documentation if needed
- The IP address is that of the DCC-EX Access Point – default is 192.168.4.1. These details can be confirmed by looking at the console log of a DCC-EX Command Station
- The Port used is again usually 2560

Other buttons of interest are the WiFi "Enable/Disable" button, a WiFi "Scan" button (can be used to discover either a DCC-EX Command Station or the Local Network. In addition a "Retries" button is available in the event of a lost connection.

Programming Page



The first thing to do is to select either the “Main” or “Prog” (Programming) track. The two examples above show their respective images.

Main Line Programming

Pressing the “Main” button automatically disables “Read” mode, selects “Write” mode and enables Power to the Main Line. If the “Join” option has been enabled (Config Page), Power will also be applied to the Programming track and the Loco can also reside there. The “Address” field will be displayed where you will need to enter the address of the Loco or device being programmed.

For convenience, 5 commonly used CV numbers are automatically selected for Minimum, Maximum, and Middle speed, as well as Acceleration and Deceleration rates. Simply enter the value for the required CV and press the appropriate name (Min, Mid, Max, Accel or Decel).

Other CVs can be programmed by completing both the CV number and its new value and pressing the “CV” field

Program Track Programming

Pressing the “Prog” button will turn Off the Power to the Programming Track and enable the selection of either Read or Write.

To Read a CV value, either press the Name of the value to read (Min, Mid, Max, Accel or Decel) or enter the CV to read in the “CV” field and press the “CV” name text field. Writing to a CV follows the same principle.

Updating

The left screenshot shows the 'DCC++EX Updating' interface. It includes a 'STOP' icon, a 'DCC++EX' logo, and a 'CS' icon. The 'SSID Scan for OTA or Upload' is set to 'Off'. The 'Network SSID' field contains 'uploadSSID' and the 'Password' field contains 'uploadPassword'. The 'HTTP Server Details for Upload' section includes 'IP Address' (HFS_IP_Addr), 'Server Port' (80), and 'Select TFT' (/Nex32b16.tft). At the bottom, there are buttons for 'Upload', 'OTA', and 'Cancel'. A message at the bottom says 'Press "Upload" or "OTA" as needed'.

The right screenshot shows the 'DCC++EX Selection' interface. It includes the same header as the left. Below the header is a 'List Complete!' section with a table of detected networks:

Network Name	Signal Strength
HomeNetwork	-72
Printer	-45
DCCEX ab123	-65

At the bottom of the right screenshot are buttons for 'Done' and 'Cancel'.

When the DCC++EX Controller is fully packaged in an enclosure, updating can be done without having to open or disassemble the device in any way. The Controller firmware is enabled for Over-The-Air (OTA) updating of the sketch, and the use of an HTTP file server to download and flash an updated Nextion TFT file.

Two scenarios are catered for by the Updating process:-

1. A Controller connected to a DCC-EX Command Station operating in Access Point (AP) Mode ie a different network than where the Arduino IDE resides
2. A Controller connected to a DCC-EX Command Station in Station (STA) Mode on the same network where the Arduino IDE resides

In order to facilitate these configurations, the Controller can store and switch between two different network SSIDs with their individual passwords. The HTTP Server (HFS) will typically reside on the Arduino IDE machine, but has its own unique Port and IP Address.

The “Updating” Page (called from the “Config” Page) provides the means to do these updates.

- The “Scan” button does a scan for nearby Networks and presents what is found on a “Selection Page”.
- The example shows a DCC_EX Command Station was found operating in AP mode as well as a Home Network and a Wireless Printer
- Pressing on the required SSID will return you to the Updating Page with the SSID space filled.
- The password is self explanatory☺

The above two fields (SSID and Password) are all that is required to prepare the Controller for an OTA update. See the Appendix for OTA details.

When upgrading the Nextion, in addition to having filled the SSID and Password fields, the IP Address and Port of the HFS HTTP server are also required to set up the transfer.

- The HFS IP address is displayed at the top of the HFS window
- The Port is usually 80 – also displayed in the HFS window
- Pressing the “Select TFT” button will present a list of compatible TFT files for the current version of the Arduino sketch. Select the one for your particular Nextion Model

The Naming convention for Nextion TFTs is:-

- /Nex (for Nextion☺)
- XX for the Nextion size (32, 35, 43 or 50)
- “b” or “e” for Basic or Enhanced
- NN for the Version number (currently 16)
- And finally “.tft” for the filename extension

The above convention results in the name “/Nex32b16.tft” for the TFT file for a Version 1.6 3.2in Basic Nextion.

One thing I find handy when doing these updates is to have the Debug Console open during the updating and transfer processes. There are a few considerations to do this:-

- OTA transfers use an IP address and a special port to do the update. Using this prevents the Serial Console from being opened via the Arduino IDE.
- A simple solution is to use a serial terminal such as “Putty” or “TeraTerm” (I like TeraTerm) connected to the Serial Debug Port while the update or transfer takes place.
- This gives the assurance that the transfers are indeed taking place and also can reveal any errors which may occur.

The same applies for the HFS Nextion transfer, even though the HFS window does show the progress. Sometimes however, the connection gets “Lost” and the debug console can reveal why☺.

Even though OTA updates for the Arduino Sketch are fast and efficient, the same can’t be said for the TFT transfer to the Nextion. Typical times are Model dependent – with the time for updating a 3.2in Nextion taking between 5 and 6 minutes, and close to 12 minutes for a 5.0in version.

Additional Pages

Three additional Pages are available on the Nextion Controller pages

- Information I quick summary of the Project
- Useful Tips Some “un-documented” features
- Credits How can we leave this out!:-)

Appendix

Downloads

The main source for everything DCC++EX Controller related is my GitHub repository

<https://github.com/normhal/DCCppEX-Nextion-UNO-Controller/> for the MK1 files

and

<https://github.com/normhal/DCCppEX-Nextion-Controller/> for the MK2 Version

For the MK1 Sketch and if you have “Git” installed copy the following command to a Windows Command Line to create a “DCCppEX-Nextion-UNO-Controller” Folder

“git clone <https://github.com/normhal/DCCppEX-Nextion-UNO-Controller/>”

The resultant folder will contain three additional directories:-

- DCCEX_Controller_For_UNO_V1_0_6 – Ready for the Arduino IDE to compile
- UNO_Upload_Utility_V1_0_6 – used to set up the UNO Controller
- Version 106 HMIs – all the HMI files for the Nextion variants

For the MK2 Sketch and if you have “Git” installed copy the following command to a Windows Command Line to create a “DCCppEX-Nextion-Controller” Folder

“git clone <https://github.com/normhal/DCCppEX-Nextion-Controller/>”

The resultant folder will contain four additional directories:-

- DCCEX-Controller-V1.6 – Ready for the Arduino IDE to compile
- DCCEX_Controller-V1.6-TFTs – all the TFT files for the Nextion variants
- NextionUpload library - in ”zip” form ready for Arduino to install
- Nextion Upload Utility – Used when Wireless upload doesn’t work☺

One final download you will need in order to use the MK2 Controller is the HFS http server. In view of the danger of viruses being installed when using an http server, I prefer that you download it from the Author’s original download link:-

<https://www.rejetto.com/hfs/download>

Follow his instructions to install and configure accordingly. Once done, “drag and drop” the .tft files downloaded above onto the HFS window.

Utility Programs and Libraries

UNO Upload Utility

This utility is exclusively for the UNO-ONLY version of the Controller, and is used as the first step to “pre-load” the UNO with Hard Coded Details which can be edited prior to compiling and flashin.

The second step is to flash the UNO ONLY version of the Nextion Controller sketch, and all the pre-configured will be available. This process was unfortunately required as the UNO came close to its full capacity.

NextionUpload Library

This library is little more than a slightly modified version of the official “ESPNextUpload” library. Two modifications have been made:

- Support for the RaspBerry Pi Pico W
- Support to use the Hardware Serial port of the ESP8266 instead of Software Serial

Install the library using the Arduino IDE Library Manager’s “zip” file installation method

ArduinoOTA Library

Install this library when using the ESP32, ESP8266 or RaspBerry Pi Pico W. No modifications necessary.

Arduino IDE Configuration, Considerations and Requirements

The Arduino IDE can be downloaded and installed from the official Arduino site:

<https://www.arduino.cc>

Go to Software, read the Terms and Conditions, and download the version of your choice. The DCC++EX Controller Sketch has been successfully compiled the both Versions 2.x.x and 1.8.19 of the IDE. Personally I still prefer the earlier 1.8.19 version because it has some features not available in the newer release

Once installed, you’ll need to install support for your choice of microcontroller. I won’t duplicate any of the excellent sites which describe in great length how to go about installing the necessary support, so to keep it short and sweet, if you add the following three lines to your “Preferences” in the Arduino IDE, you will be able to select the correct processor support:-

- https://espressif.github.io/arduino-esp32/package_esp32_index.json for ESP32
- http://arduino.esp8266.com/stable/package_esp8266com_index.json for ESP8266
- https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json for Pico and Pico W

For the ESP32, I use Version 2.0.14 of the Boards package. I find any of the Version 3.0.x versions don’t work☹

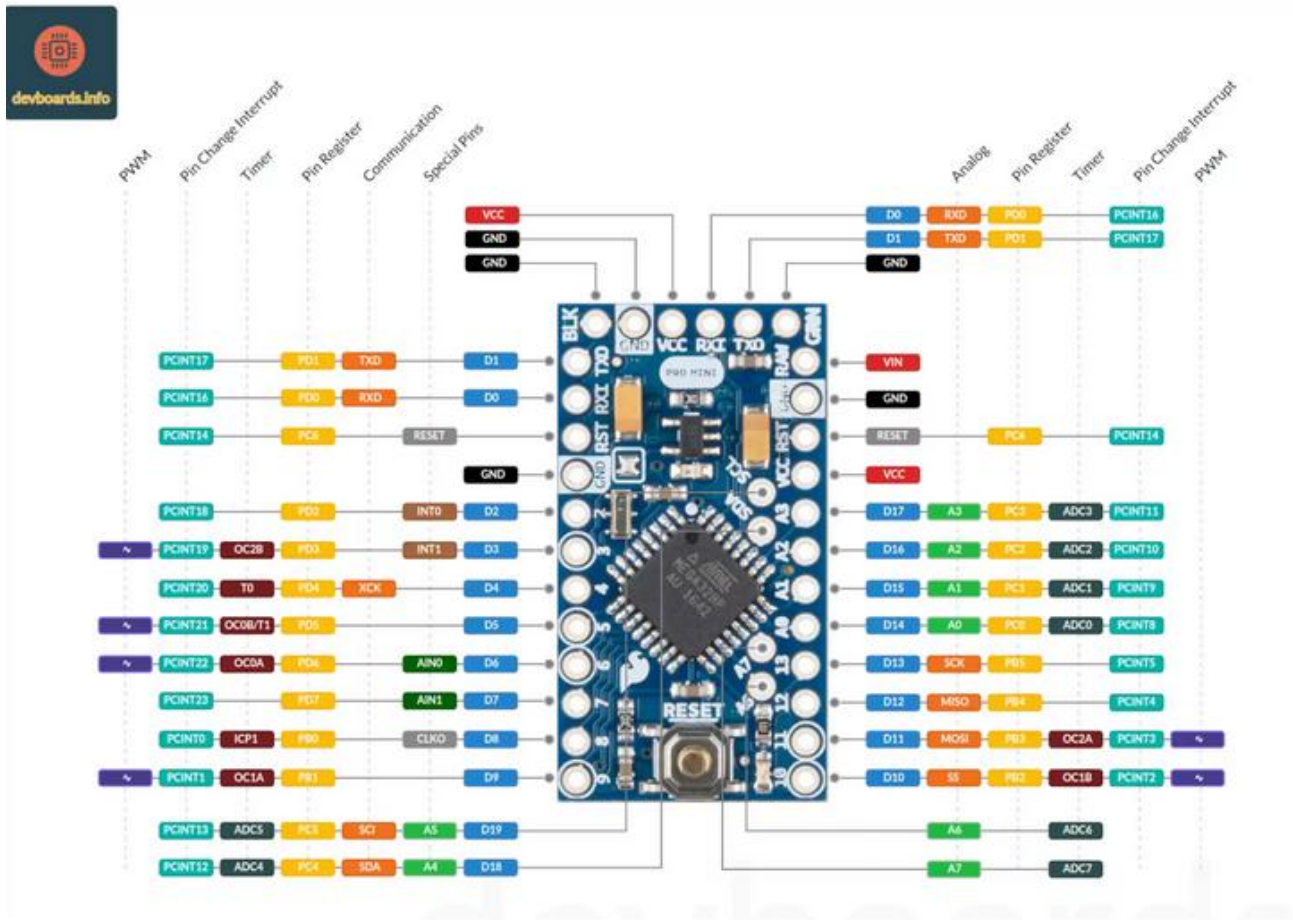
For the ESP8266 the version I use is 3.1.2

For the Pico W the version is 3.7.2

Be sure to Review the section Specific to the Microcontroller you’re using – UNO, ESP8266, ESP32 or Pico later on in this Appendix, and be sure to set the Controller options in the “Tools” section of the IDE correctly.

Pin Assignments and Connections

UNO Pro-Mini



Arduino Pro Mini Pinout and Specification - devboards.info

[Visit >](#)

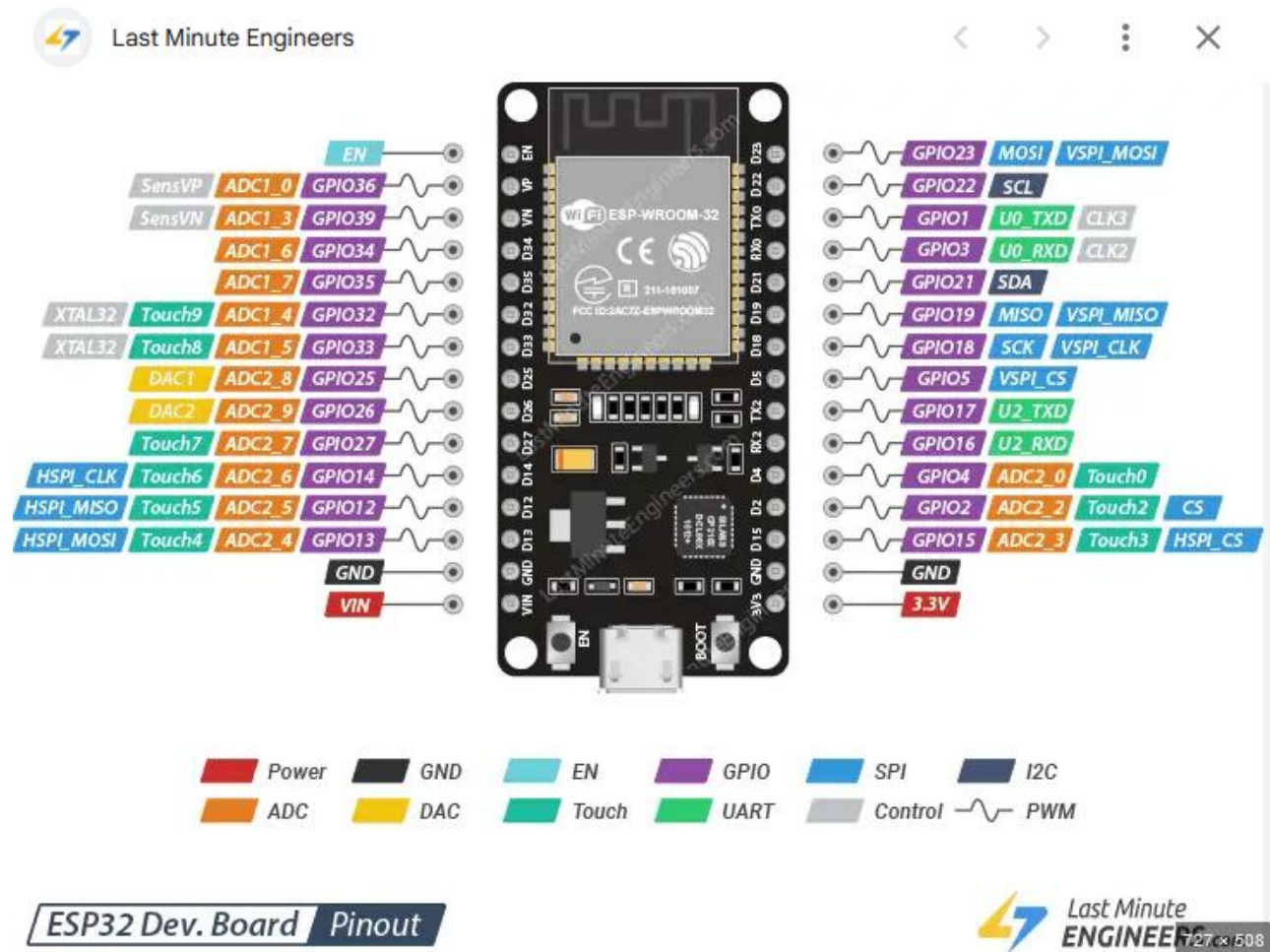
Pro-Mini GPIO Pins Used by the DCC++EX Controller

D4	Nextion Blue Wire (TX)
D5	Nextion Yellow Wire (RX)
D2	Rotary Encoder Pulse A
D3	Rotary Encoder Pulse B
D8	Rotary Encoder Switch
D0	Direct Connection RX
D1	Direct Connection TX

Arduino IDE Tools Settings

Port Comx as assigned by your computer

ESP32 Dev Module Pinouts



ESP32 Pinout Reference - Last Minute Engineers

[Visit >](#)

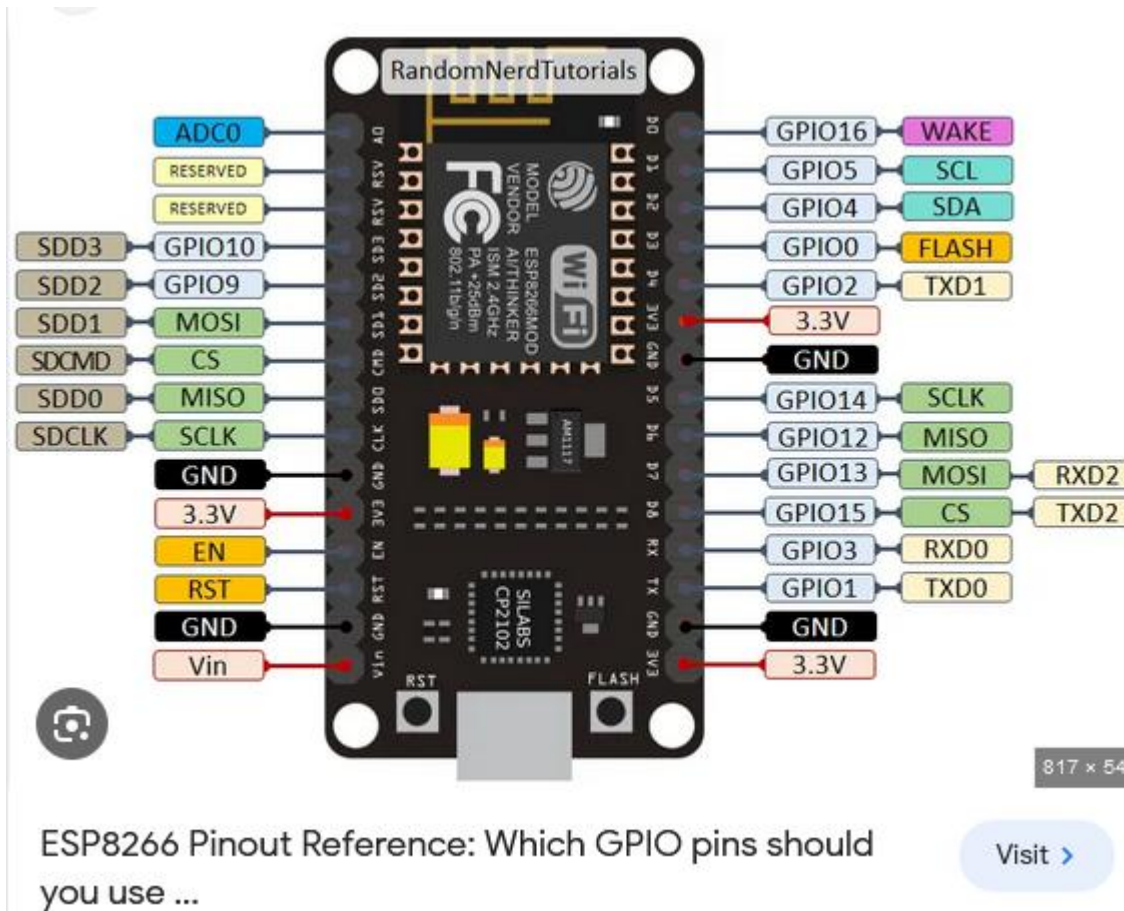
ESP32 GPIO Pins Used by the DCC++EX Controller

GPIO16	Nextion Blue Wire (TX)
GPIO17	Nextion Yellow Wire (RX)
GPIO4	Rotary Encoder Pulse A
GPIO5	Rotary Encoder Pulse B
GPIO13	Rotary Encoder Switch
GPIO3	Direct Connection RX
GPIO1	Direct Connection TX

Arduino IDE Tools Settings

Port	Comx as assigned by your computer esp32-xxxxxx at nnn.nnn.nnn.nnn IP Address if OTA previously loaded and available
Partition Scheme	Minimal SPIFFS (1.9MB APP with OTA/190KB SPIFFS)
Erase Flash	All Flash Contents (Only needed if ESP32 used previously)

ESP8266 Pin-out Assignments



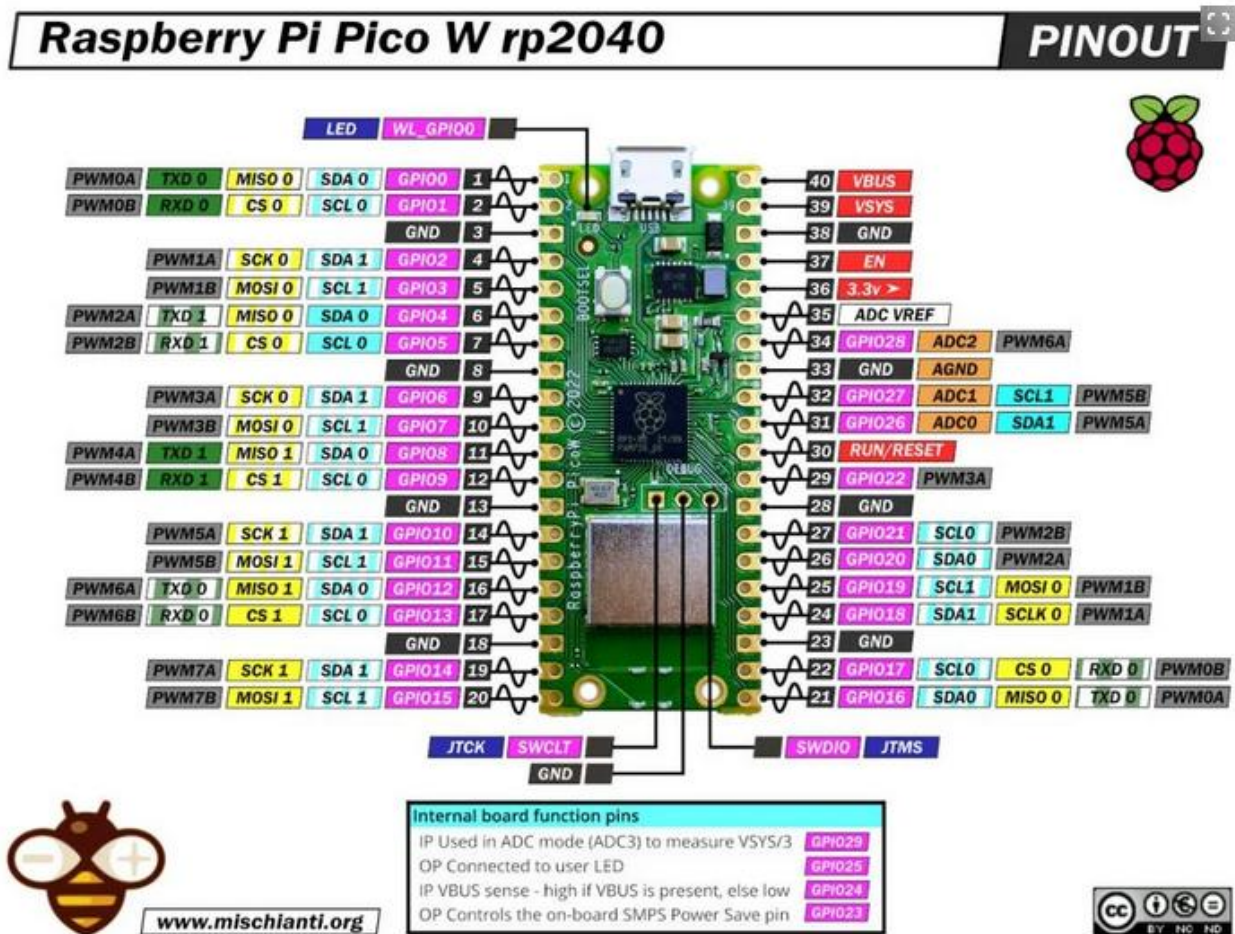
ESP8266 GPIO Pins Used by the DCC++EX Controller

GPIO3	Nextion Blue Wire (TX)
GPIO1	Nextion Yellow Wire (RX)
GPIO12	Rotary Encoder Pulse A
GPIO13	Rotary Encoder Pulse B
GPIO5	Rotary Encoder Switch
GPIO4	Direct Connection RX
GPIO14	Direct Connection TX

Arduino IDE Tools Settings

Port	Comx	as assigned by your computer	
	esp8266-xxxxxx	at nnn.nnn.nnn.nnn IP Address	If OTA previously loaded and available
Flash Size	4MB (FS:none OTA:~1019KB)		
Erase Flash	All Flash Contents (Only needed if ESP32 used previously)		

Raspberry Pi Pico and Pico W Pinouts



Pi Pico GPIO Pins Used by the DCC++EX Controller

GPIO8	Nextion Blue Wire (TX)
GPIO9	Nextion Yellow Wire (RX)
GPIO4	Rotary Encoder Pulse A
GPIO5	Rotary Encoder Pulse B
GPIO13	Rotary Encoder Switch
GPIO1	Direct Connection RX
GPIO3	Direct Connection TX

Arduino IDE Tools Settings

Port	Comx	as assigned by your computer	
	rp2040-xxxxxx	at nnn.nnn.nnn.nnn IP Address	If OTA previously loaded and available
Flash Size	2MB	(Sketch 1536 FS: 512KB)	
Erase Flash	All Flash Contents (Only needed if used previously)		

Fault Reporting

If you encounter any problems, please do not hesitate to contact me – either via TrainBoard or Discord – normhal.

Alternatively, send me an email – normhal@gmail.com

In view of the variety of configurations possible when using the DCC++EX Controller, it is important that you always include the following details:-

- Sketch Version Number
- HMI Version Number installed
- Processor being used
- Nextion Display Size and model
- DCC-EX Configuration (AP or STA mode)

It would also help if you could report the steps taken to repeat the problem.

If I can't repeat the problem you're encountering, I will most likely come back to you and ask for further details such as the versions of Libraries and/or Board Manager files being used. In any event any problem you might encounter (and we fix😊) will be for the benefit of everyone, so please don't hesitate to drop me a message😊