Nextion Displays and Great Cow BASIC

This step by step tutorial will show you:

- How to use a Nextion display and Great Cow BASIC with Nextion displays.
- How to use the Great Cow BASIC language with Nextion diskplays woyh Microchip PICs and AVRs 8bit microcontrollers including the Arduino range microcontrollers, with a common approach with a cross microcontroller solution.

The solution you will create will show you.

- How the Nextion display responds to Nextion Button Component release event to send instructions the microcontroller and the microcontroller will instruct the Nextion display to change page(s)
- How the Nextion display on-screen Button Component press/release event send instructions to the microcontroller to control an LED
- How the microcontroller update status Text Component on the Nextion display
- How the microcontroller increments a variable and updates a Number Component on the Nextion display
- How the Nextion display evaluates and update color status of a Text Component based on value of another Number Component
- The Nextion screen when created, following the instructions below, look like this.

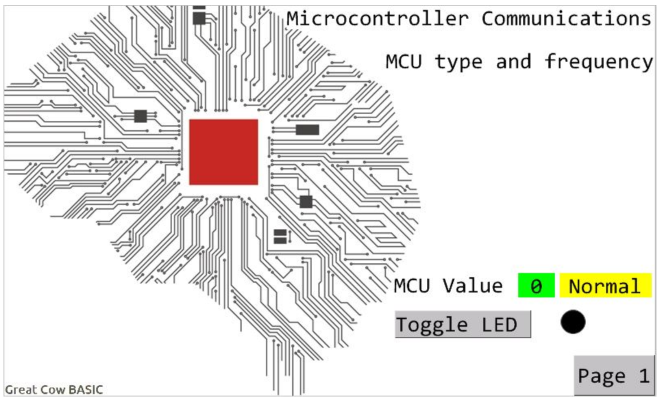
When the Nextion display is operation you can be able to control an LED attached to the microcontroller (via a suitable resisitor) and the microcontroller will update the Nextion display.

So, enjoy this tutorial and thank you to Patrick for the inspiration.

Evan

Nextion Displays and Great Cow BASIC: step by step

You will create a Nextion screen similar to this and integrate with a microcontroller.



This tutorial will explain:

- How to use a Nextion display with Great Cow BASIC.
- How Great Cow BASIC can support many Microchip PICs and AVRs with a portable langauge that is simply to use with the Nextion display.
- How to ensure the demonstration works for you by hilighting the critical activities that are mandated to make the tuturial work for you.
- How you adapt for your needs.

What you will understand following the tutorial

- How the Nextion display responds to Nextion Button Component release event to send instructions the microcontroller and the microcontroller will instruct the Nextion display to change page(s)
- How the Nextion display on-screen Button Component press/release event send instructions to the microcontroller to control an LED
- How the microcontroller update status Text Component on the Nextion display
- How the microcontroller increments a variable and updates a Number Component on the Nextion display
- How the Nextion display evaluates and update color status of a Text Component based on value of another Number Component
- How to ensure the Nextion display and the microcontroller communicate correctly

---

Step 1 - Install the software required.

	Activity	**Mandated**	**Comment**
1	Download and install Great Cow BASIC toolchain for your specific operating system	Yes	We need this software installed and operational before we start. See https://sourceforge.net/projects/gcbasic/
2	Download and install the Nextion Editor	Yes	See https://nextion.tech/ and then select the download

			options(s).

Step 2 - Choose your microcontroller and ensure you can program from Great Cow BASIC. There are many resources but the simple choices are to use an Arduino UNO or one of the many Microchip Xpress boards. The tutorial will use an Arduino UNO and an Microchip Xpress Evaluation Board which uses a PIC6F18855.

	Activity	Mandated	Comment
1	Test your Great Cow BASIC installation.	Do this, or download the latest from GitHub, see activity 3.2	The following Help page has a GIF that shows how to set up the programmers for the Arduino UNO and other programmers. http://gcbasic.sourceforge.net/help/_using_great_cow_basic.html If you need help on this - the https://sourceforge.net/projects/gcbasic/ forum is a great friendly resource.

Step 3 - Use an existing resource to start your project

	Activity	Mandated	Comment
1	Open the tutorial code within your Great Cow BASIC installatio	Do this, or download the latest from GitHub, see activity 3.2	The tutorial in located in your .. GreatCowBasic\Demos\glcd_solutions\glcd_nextion_solutions\tutorial sub folder
2	**Download the tutorials from GitHub.** Save the files to a location you will remember on your hard drive.	Do this, or use the stock demos, see activity 3.1	See https://nextion.tech/ and then select the download options(s).

Step 4 - The Nextion HMI

In this step you will create the Nextion solution using the Nextion Editor. We will be using a Nextion NX8048P070 but you can you use many of the Nextion displays - you will see how to adapt with each activity.

	Activity	Mandated	Comment
1	Gather the resources will be using. You may use a picture resource for the Nextion display background. The picture resource shoould match the size of your Nextion display	No	The picture is optional as you can set the bacground colo
2	Start the Nextion Editor, Create a new project. Name the project 'Tutorial' and select `Save`.	Yes	:You will be presented with a setting dialog to choose your device.
3	Select BASIC, Enhangced or Intelligent, Then, select your specific device type, Then, select the `Display` tab (if you press OK, you will be presented with the `Display` tab). Select the "Display Direction". Then, press `OK`	Yes	This activity is specific to your device type. ***Page 0*** is created as the default page Component. All Components discussed will be in BOLD and Italics.
4	We need another resource. A font resource. Create a font resource. An 8×16 ASCII ZI font file. Create this font using the Nextion Editor. Select Tools/Font Generator. Follow the dialog to create the font resource, save and finally select `Yes` when requested "Add the generated font?". C;lose the dialog,	Yes	You will require this font. If you have a suitable Nextion font then skip this activity but simply add the font to the project using the To review the font installed - select the resource interface (below the `Toolbox`, selecting the `Fonts` tab at the bottom.
5	Add the picture resource for ***Page 0*** background. Select the resource interface and then select the `Picture` tab. a) in the Picture tab, click `+` b) Select your picture file and click Open The selected picture is now added as "Picture resource	No	This is optional.

	ID:0"		
6	Add the picture to the page. Setting the Page ***Page0*** background to "Picture resource ID:0" a) Select the ***page0*** attribute ".sta" value dropdown, it will be the default value of "solid color". Select the dropdown option "image". b) Select attribute ".pic" then "browse" and select Picture resource "ID:0" and press `OK`. The picture will be inserted as the page background.	No	This is optional.`r0.bco=0` `r0.pco=0`

7	In the Event Pane, Preinitialize Event add the Nextion event instuctions.
	
	`bauds=9600`
	
	The value of the baud rate will need to match the microcontroller.
	
	Also, in the Event Pane, Postnitialize Event add the Nextion event instuctions.
	
	`r0.bco=0`
	`r0.pco=0`
		Yes	If the baud rate does not match there will be no communication, so, this is a critical step.

8	Add the font, if you have not added when you created the font ZI file. a) In the Font Pane click + b) select your 8×16 ZI Font file (*.zi) and click Open The Selected`r0.bco=0` `r0.pco=0` ZI Font is now added as "Font Resource ID:0"	No	'Page 1 events... there is only one, so, just change page
9	Add a Button Component. Use the Toolbox Pane to create the Button Component.. Edit the attibutes as follows: a) Ensure the attribute "objname" is "***b0***" b) Change the attribute "txt" to "Page1" c) Set attribute "x" to "700" d) Set attribute "y" to "430" then, in the Event Pane e) Select the Touch Press Event tab and check "Send Component ID" checkbox.	Yes	The "x" and "y" attributes are specific to your Nextion display. A quick way of setting this is to drag the Button Component to bottom right position.
You now have three Components that need to aligned - to look nice.			
10	Add a Text Component. Use the Toolbox Pane to create the Text Component.. Edit the attibutes as follows: a) Ensure the attribute "objname" is "***t0***" b) Set attribute "txt_maxl" to "15"... Set this before the next attribute c) Change the attribute "txt" to "MCU Value" d) Set attribute "x" to "470" e) Set attribute "y" to "330" f) Set attribute "w" to "150" g) Set attribute "h" to "30"	Yes	The "x","y","w" and "h" attributes are specific to your Nextion display. A quick way of setting this is to drag the Text Component to bottom right position and then resize the Text Component.
11	Add a Number Component. Use the Toolbox Pane to create the Number Component.. Edit the attibutes as follows: a) Ensure the attribute "objname" is "***n0***" b) Set attribute "x" to "630" c) Set attribute "y" to "330" d)Set attribute "w" to "60" e) Set attribute "h" to "30"		The "x","y","w" and "h" attributes are specific to your Nextion display. A quick way of setting this is to drag the Number Component and then resize the Number Component.
12	Add Text Component. Use the Toolbox Pane to create the Text Component.. Edit the attibutes as follows:	Yes	The "x","y","w" and "h" attributes are specific to your Nextion display. A quick way of setting this is to drag the Text Component and then resize the Text Component.

	a) Ensure the attribute "objname" is "***t2***". Get this attribure correct. b) Ensure the attribute "xcen" is "Right" c) Set attribute "txt_maxl" to "10"... Set this before the next attribute d) The attribute "txt" will be "nextxt". Leave as is. e) Set attribute "x" to "700" f) Set attribute "y" to "330" g) Set attribute "w" to "100" h) Set attribute "h" to "30"		The attribute "objname" is critical at this step.
You now have three Components that need to aligned - to look nice.			

13	Add a Timer Component. Use the Toolbox Pane to create the Timer Component..
	
	When creating a Timer Component is added in the middle of the UI. But, a quick way to locate the attributes of the timer is to use the Attribute Pane and select the timer.
	`r0.bco=0`
	`r0.pco=0`
	Edit the attibutes as follows:
	
	a) Ensure the attribute "objname" is "***tm0***"
	b) Set the attribure "tim" is "50"
	
	Then,
	
	c) In the Event Pane add the following Nextion code.
	
	`if(n0.val>50)`
	`{`
	`  n0.bco=63488`
	`  t2.txt="Too Hot!"`
	`}else`
	`{`
	`  n0.bco=2016`
	`  t2.txt="Normal"`
	`}`
		Yes	This critical timer will update the text Component created at step 12.
			
			63488 equates to RED
			2016 equates to GREEN

You can, optionally, "Debug" the project at this step. You should have no errors. If you enter "n0.val=100" in the `Instruction Input Area" the obejct ***n0*** will change to RED!! Exit the debugger.			

14	Add a Button Component. Use the Toolbox Pane to create the Button Component..
	
	Edit the attibutes as follows:
	
	a) Ensure the attribute "objname" is "***b1***". Get this attribute correct!
	b) Set attribute "txt_maxl" to "10"... Set this before the next attribute
	c) The attribute "txt" will be "Toggle LED".
	d) Set attribute "x" to "470"
	e) Set attribute "y" to "380"
	f)Set attribute "w" to "170"
	g) Set attribute "h" to "30"
	
	Then, In the Event Pane select the following:
	
	h) in Touch Press Event tab. Check the "Send Component ID" checkbox
	i) In the Touch Release Event tab. Check Send Component ID checkbox
	j) Add the following Nextion code, to always set a Radio Component to black. The Radio Component will be created at step 16.
	
	`r0.bco=0`
	`r0.pco=0`
		Yes	The "x","y","w" and "h" attributes are specific to your Nextion display. A quick way of setting this is to drag the Button Component and then resize the Button Component.
			
			The Event Pane actions are critical. Selecting the check boxes will send key information to the microcontroller when you Press and Release the button.

15	Note the Button Component "id" created in activity 14. This tutorial assumes the Button Component "id" equals 6.	Yes	This Button Component "id" value will be needed in our microcontroller program.

	But, you will need this value later!		
16	Add a Radio Component. Use the Toolbox Pane to create the Radio Component.. Edit the attibutes as follows: a) Ensure the attribute "objname" is "***r0***". <u>Get this attribute correct!</u> b) Set attribute "bco" to "63488" c) Set attribute "pco" to "0" d) Set attribute "x" to "670" e) Set attribute "y" to "380" f)Set attribute "w" to "30" g) Set attribute "h" to "30"	Yes	This Radio Component will be updated from the microcontroller.
17	Add Text Component. Use the Toolbox Pane to create the Text Component.. Edit the attibutes as follows: a) Ensure the attribute "objname" is "***t1"***. <u>Get this attribure correct.</u> b) Ensure the attribute "xcen" is "Right" c) Set attribute "txt_maxl" to "40"... Set this before the next attribute d) The attribute "txt" will be "Microcontroller Communications" e) Using drag and drop place the Text Component at a position below the top right corner and adjust the width.	Optional	
18	Add Text Component. Use the Toolbox Pane to create the Text Component.. Edit the attibutes as follows: a) Ensure the attribute "objname" is "***t3"***. <u>Get this attribure correct.</u> b) Ensure the attribute "xcen" is "Right" c) Set attribute "txt_maxl" to "30"... Set this before the next attribute d) The attribute "txt" will be "MCU type and frequency" e) Using drag and drop place the Text Component at a position below the top right corner and adjust the width.	Yes	This Text Component will be updated from the microcontroller.
Now create a second page and one component.			
19	Add a New Page. Use the Page Pane to create a second page. Add new Page "page1" a) In the Page pane click Add icon a) Set the attribute ".bco" as 0	Yes	
20	Add a Button Component. Use the Toolbox Pane to create the Button Component.. Edit the attibutes as follows: a) Ensure the attribute "objname" is "***b0***" b) Change the attribute "txt" to "Back" c) Using drap and drop place the Button Component at the botton right position. Then, In the Event Pane select the following: h) in Touch Press Event tab. Check the "Send Component ID" checkbox	Yes	
To return to review ***"page0***" double click ***"page0***" in the Page Pane. Save the project.			
21	Preparing your microSD card a) ensure the microSD card is embedded compatible A Kingston Class 10 HC 8GB microSDs has been proven	Yes	

	to work. b) ensure the microSD is format under Windows as FAT32 c) ensure there are no *.tft files on the microSD card		
22	Install the project to the Nextion device In the Nextion Editor File Menu a) Click Open Build Folder b) copy Tutorial.tft to your prepared embedded microSD card c) Turn power to the Nextion off. d) Insert the microSD card with Tutorial.tft into Nextion device e) Turn power to the Nextion on f) Wait for Nextion to begin uploading and complete uploading When Nextion completes uploading and states success g) Turn power to the Nextion off h) Remove the microSD card from the Nextion device i) Turn the power to the Nextion on j) Wait a few moments to ensure all firmware upgrades complete At this point the Tutorial should be running on the Nextion display.	Yes	

Step 5 - Great COW BASIC program

In this step you will create the Great Cow BASIC solution using the Great Cow BASIC IDE. We will be using an Ardunio (really an ATMEL328p microcontroler) and an Microchip Xpress 16F18855 board but you can you use many of the supported microcontrollers - you will see how to adapt with each activity.

The solution approach is relatively simple from a microcontroller programming point of view. The main programmng functionality is as shown below in the psuedo code.

```
Do a loop forever
|
|   Increment a value to be sent to the Nextion
|   Send the value to the Nextion display
|
|   Do this code segement if the microcontroller has received a byte from the Nextion
|   |
|   |   IF a Nextion TouchEvent
|   |   |
|   |   |   Select which Page was the source of the TouchEvent
|   |   |   |
|   |   |   |   Case Page 0
|   |   |   |
|   |   |   |      Select which Component Raise Event
|   |   |   |      |
|   |   |   |      |   Case ToggleLEDEvent
|   |   |   |      |
|   |   |   |      |       Set LED on
|   |   |   |      |       Instruct Nextion to change color of LED radio component
|   |   |   |      |
|   |   |   |      |   Case SelectPage1Event
|   |   |   |      |
|   |   |   |      |       Instruct Nextion to goto to page0
|   |   |   |      |
|   |   |   |      End Select
|   |   |   |
|   |   |   |  Case Page 1
|   |   |   |
|   |   |   |      Instruct Nextion to goto to page0
|   |   |   |
|   |   |   End Select
|   |   |
|   |   End if
|   |
|   End Do loop
|
End Do loop
```

This psuedo code represents the solution that we will create. We will add microcontroller section, communications setup, the creation of a buffer to handle the incoming stream of bytes from the Nextion and change some application constants to match the project you created in the Nextion Editor.

If you have downloaded the demonstration code from GitHub, or, opened the demonstration within your installation - the code may be easier.

	Code Segment	**Comment**
1	Specify the microcontroller type `#chip mega328p`	This could also be many of the 8-bit Microchip microcontrollers, like the 16F `#chip 16f18855`
2	If you are using an UNO. Include the hardware file to expose the UNO constants for the LED. `#include <uno_mega328p.h>`	
3	Tell the compiler to ensure all variables are defined. `#option explicit`	

4	Setup the Nextion. Change the width and height to match the rotation in the Nextion Editor
	
	`#include <glcd.h>`
	`#define GLCD_TYPE GLCD_TYPE_Nextion`
	`#define GLCD_WIDTH  800.`
	`#define GLCD_HEIGHT 480`
		This instructs the compiler to include the Nextion software library.

5	Setup the communications. We are using 9600. This will match our Nextion project.
	
	`#define USART_BAUD_RATE 9600`
	`#define USART_BLOCKING`
	`#define USART_DELAY 0`	

6	Define, adapt constants to match of Nextion project. Ensure this CONSTANTS SHOULD BE CORRECT and they match for Nextion project components in the tutorial.
	
	`#define RADIOCOMPONENT          "r0"`
	`#define NUMBERCOMPONENT         "n0"`
	`#define TEXTCOMPONENT`
	`"t3"`
	`#define TOGGLELEDBUTTON         6`
	`#define SELECTPAGE1BUTTON       1`
	
	`'Statics that should not change.`
	`#define PAGE0                   0`
	`#define PAGE1                   1`
	`#define TOUCHEVENT              0x65`
	
	`'   The microcontroller LED`
	`#define LED1            DIGITAL_13`
		These needs to match the Nextion project.
		
		The Nextion component `objname` and `id` are critical.
		
		If you have the LED attached to a different port then change the constant LE something like:
		
		`'   The microcontroller LED`
		`#define LED1            Porta.0`

7	Setup the serial buffer. This is a buffer ring where the incoming bytes are placed in the buffer.
	
	As a byte arrives we can test `bkbhit`. If this is true then a byte as arrived. After we know a byte as arrived, the function `bgetc` returns the byte.
	
	The buffer uses an interrupt to ensure the microcontroller places the byte in the buffer.
	
	`On Interrupt UsartRX1Ready Call`
	`readUSART`
	
	`'Constants required for Buffer Ring`
	`#define BUFFER_SIZE 255`
	`#define bkbhit (next_in <> next_out)`
	
	`'Declare the required variables`
	`Dim buffer( BUFFER_SIZE - 1 ) '`
	`Dim next_in as byte: next_in = 0`
	`Dim next_out as byte: next_out = 0`
		Some micrcontrollers, specificially the newer microcontrollers may be the `U adapting to the appropiate microcontroller event name.

8

Declare the variables that we need for the program. And, set the LED as an output

```
    ' ----- Declare the variables we need
    for this tutorial
        dim byteValueOutToNextion as byte
        dim inComingByteFromNextion as byte
        dim stringOutToNextion as string

    ' ----- Set the LED as an output
        dir LED1 out
```

9

Create a string for Nextion text component - this will state the type of microcontroller and the frequency of the microcontroller, and, call the method to initialise the Nextion component `***page0***`.

```
    #ifdef PIC
        stringOutToNextion = "PIC"
    #endif
    #ifdef AVR
        stringOutToNextion = ""
    #endif
    stringOutToNextion = stringOutToNextion
    + ChipNameStr + " @"+str(ChipMHz)+"Mhz"

    'Initialise page0
    displayPage0
```

As this program can support ATMEL and PIC we add a prefix only when a F

10

The main loop. This is eseentially the psuedo code.

```
'   Main loop, never exits
    do
        'Update the n0.val object
        byteValueOutToNextion = (byteValueOutToNextion + 1) % 101
        GLCDUpdateObject_Nextion( NUMBERCOMPONENT+".val",  byteValueOutToNextion )
        wait 50 ms

        'react to incoming data, if bkbhit is TRUE then we have some serial data!
        do while bkbhit

            'Has a Nextion Touch Event happended.  bgetc is the next serial byte, so, we can test
            if bgetc = TouchEvent then

                'The next bgetc byte in the buffer is the originating Nextion page
                select case bgetc

                    'Page 0 events... test the next buffer byte to set which component caused the
                    case PAGE0

                        'The next bgetc byte in the buffer is the  component ID that caused the e
                        select case bgetc

                            'Page change pressed
                            case SELECTPAGE1BUTTON
                                'Send an instruction to 'page' change the Nextion
                                GLCDSendOpInstruction_Nextion( "page",  "page"+str(PAGE1) )

                            'Toggle LED pressed or released
                            case TOGGLELEDBUTTON

                                'The next byte is the state of the TOGGLELEDBUTTON 0 or 1
                                inComingByteFromNextion  = bgetc

                                'Set the LED state to TOGGLELEDBUTTON state
                                LED1 = inComingByteFromNextion

                                'Tell the Nextion the LED is ON
                                if inComingByteFromNextion = 1 then
                                    Repeat 5
                                        GLCDUpdateObject_Nextion( RADIOCOMPONENT+".bco",  [long]6
                                        GLCDUpdateObject_Nextion( RADIOCOMPONENT+".pco",  [long]6
                                    end Repeat
                                end if

                        end select

                    'Page 1 events... there is only one, so, just change page
                    case PAGE1
                        displayPage0
                   end select

            end if

        loop

    loop
```

11 Create a method to initialise `***page0***`. This is created as the program can call this many times.

```
         sub displayPage0

             'Send an instruction to 'page' change the Nextion
             GLCDSendOpInstruction_Nextion( "page",  "page"+str(PAGE0) )
             'wait for display
             wait 750 ms
             'Update the page text
             GLCDUpdateObject_Nextion( TEXTCOMPONENT+".txt", stringOutToNextion  )

         End sub
```

12 Finally add some utility methods - these are required to support the Interrupt routines - no need to change

```
         Sub readUSART
             dim temppnt as byte
             buffer(next_in) = HSerReceive
             temppnt = next_in
             next_in = ( next_in + 1 ) % BUFFER_SIZE
             if ( next_in = next_out ) then  ' buffer is full so overflow
                 next_in = temppnt
             end if
         End Sub

         function bgetc
             wait while !(bkbhit)
             bgetc = buffer(next_out)
             next_out=(next_out+1) % BUFFER_SIZE
         end Function
```

If you are using a newer PIC with PPS you will be required to add the PPS instructions to setup the serial communications. See the GITHUB e
PPS.

Finally, compile and load the program.   When you load the program to an UNO.

Ensure to unplug the Nextion RX/TX so it does not interfere with the sketch upload. See connectivity section below.

Step 6

Connect the microcontroller to the Nextion display.

	Activity
1	For each of the four Nextion DuPont connector ends, insert a header pin into the end so can now be plugged in directly to the target board.
2	Do not connect the Nextion display directly to the microcontroller and the microcontroller power supply UNLESS you also make a similar confirmations (for your specific board and your model Nextion) via Datasheets. Nextion Datasheets are clear on the recommended supply current required. Failure to review and understand the recommended supply current required for the Nextion display will cause damage.
3	Connect as follow. Nextion RX (yellow) to Serial TX on the microcontroller. Nextion TX (blue) to UNO Serial RX on the microcontroller. Nextion GND (black) to the microcontroller 0V. Nextion 5V (red) to a suitable 5V supply**. This is probably NOT the microcontroller board as MANY of the microcontroller boards cannot provide sufficient current - and, if you do connect you may cause real damage.** **Connect the** Nextion 5V (red) to a suitable 5V supply and then connect a common 0V to the Nextion and the microcontroller.

Step 7

Enjoy the integrated solution of the Nextion display and a microcontroller. You can use the solution as follows

Action	**Response**
Press Nextion Button "Page1"	Nextion display notifies the microcontroller and the microcontroller instructs the Nextion to change the page to ***`page1`***
Press Nextion Button "Back"	Nextion display notifies the microcontroller and the microcontroller instructs the Nextion to change the page to ***`page0`***
Press Nextion Button "Toggle LED"	Nextion display notifies the microcontroller, the microcontroller set the state of the LED and the microcontroller instructs the Nextion to update the radio button on the Nextion display
Automatically the microconroller sends a value to the Nextion display	Nextion display is updated with the value. And, the Nextion display timer event evaluates this component and sets the number component color and the text compont appropiately.

Summary

It is simple to connect the Microchip range of 8-bit microcontrollers to the Nextion display. Great Cow BASIC makes the task very easy with very little code adapation for each microcontroller type.

Enjoy