DOI: 10.35598/mcfpga.2023.001

Study of the Effectiveness of Using Nextion Displays in Projects Based on STM32 Microcontrollers

Oleg Zubkov
ORCID 0000-0002-8528-6540
dept. Microprocessor Technologies and Systems
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
oleh.zubkov@nure.ua

Iryna Svyd
ORCID 0000-0002-4635-6542
dept. Microprocessor Technologies and Systems
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
iryna.svyd@nure.ua

Oleksandr Vorgul
ORCID 0000-0002-7659-8796
dept. Microprocessor Technologies and Systems
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
oleksandr.vorgul@nure.ua

***Abstract*—An analysis of the problems of developing graphic screens with an original design for STM32 microcontrollers without a built-in graphics accelerator was carried out. The analysis of the electronics development companies requirements for projects with built-in visualization in electronic devices has been carried out. A comparative analysis of the Nextion displays characteristics and industrial operator panels was carried out. The capabilities of Nextion displays were studied: the speed of loading images from flash memory and displaying these images on the screen, the time it took to develop screens, the speed of data exchange between the display and the microcontroller. A comparative analysis was carried out between identical projects created for the Nextion display in the Nextion Editor environment and a display for which there is no specialized GUI development environment. The effectiveness of using Nextion displays for high-quality visualization of the device state and systems designed on the STM32 microcontrollers has been proven.**

Keywords—display, microcontroller, widget, Nextion, STM32, interface UART.

I. INTRODUCTION

In many modern electronic devices and systems of multimedia, industrial, household, etc. applications use touch displays. They allow to display a device or system status in real-time and allow the user to control processes or configure device or system parameters [1, 2]. To compete in the modern electronics market, a number of requirements are imposed on new electronic devices [3]. The time to write code for the controller and display should be kept to a minimum. When changing programmers, new developers should quickly get to grips with the existing code and continue developing or maintaining it. All this requires the use of universal programming environments and standard approaches. Modern displays are divided into two groups: with and without an integrated controller [1, 4]. The presence of a built-in controller allows you to use any microcontroller to control the display. However, the exchange rate is limited by the speed of the controller built into the display, the capabilities of its software and the data transfer interface. The usage of displays without an integrated controller requires the use of microcontrollers with an integrated graphics accelerator. Among microcontrollers of the STM32 family, only the high performance F7, H7, L4 series have a built-in graphics accelerator [1]. For such series, there is a specialized software product Touc5GFX [4], using which, in a short time and at a high design level, you can create a set of graphic screens for visualization and control, and even provide video streaming. However, in the cheaper and more popular in practice series F0, F1, F2, F3, G0, L0, part of the F4, etc., is missing a graphics accelerator. In this case, the creation of beautiful and functional graphic screens requires the usage of third-party software environments, and the import of graphics into the microcontroller project requires significant time resources, which unacceptably increases the development time. Such solutions also require specific knowledge, which complicates the creation of a programmers team in large projects.

Since 2014, Nextion company has been producing 4 series of resistive and capacitive displays with screen sizes from 2.4' to 10'. The clock frequency of the controller built into the display is in the range from 48 to 200 MHz. Displays have built-in flash memory from 4 to 128 MB, built-in RAM from 3.5 to 512 kB, non-volatile memory up to 1 kB. The UART interface is used to communicate with the display. The main advantage of these displays is the intuitively simple environment for their configuration and programming - Nextion Editor. The displays are not designed to receive streaming video from the camera, but they successfully solve all other tasks of high-quality visualization and control [6, 7]. On the manufacturer's website and other resources, you can find many examples of interaction with Arduino devices [7], but there are no results from a study of performance when implementing animation and implementing complex graphic screens [6-9]. In [10-14] questions and examples of work that can be useful for studying the specified question are considered.

Therefore, the purpose of research and analysis was to evaluate the effectiveness of the Nextion displays usage when working together with STM32 microcontrollers.

II. Development of Graphic Displays and Data Exchange With STM32

The Nextion Editor environment has a built-in set of standard graphic widgets: screens, buttons, pictures, input fields, sliders, etc. For each widget, the user has access to settings: labels on widgets, their sizes, positions, colors, etc. However, it's not possible to change the design of the widget. For example, you can't make a standard button oval, round the edges, create a color gradient within the button, etc. It is only rectangular with a solid color fill. The solution to this problem is to overlay the graphic on the button widget. In this case, the creation of the image is carried out by the designer. In the same way, the problem associated with the limited number of widget types is solved. So, using pictures, you can create very complex scales, multi-position switches and other elements corresponding to electronic devices. However, each of the images that is superimposed on the widget must be read from the controller's flash memory and displayed. The display manufacturer does not provide information about the screen refresh rate. Therefore, it is necessary to conduct a study - what is the speed of the transfer images between flash memory to display and what is the relationship between the transfer and visualization time and the number of output files.

An equally important issue for developers of display screens is the required amount of RAM and flash memory to create one screen with original design. For each widget, at least one picture must be stored in flash memory. Although the number of pictures can be much larger. For example, for a button, it is necessary to save the images of the unpressed and pressed button. Information widgets that display errors or actuator states can have up to 10 or more graphical icons corresponding to their states. RAM stores information about the position of widgets, their sizes, available to change properties.

In addition to widgets, you can create variables in the Nextion Editor environment. The environment only supports 2 data types: 32-bit integers and string variables. This constraint requires all other microcontroller data types to be converted to screen data types before those values are sent to the screen. So, for example, float variables can be converted to text values. The UART interface is used to transfer data between the microcontroller and the screen. The transmission rate is adjustable from 2400bps to 921600bps. When display transmits data to the microcontroller, it is possible to form an arbitrary package structure or a structure that corresponds to the the Nextion display operating system. An arbitrary structure makes it possible to combine the values of a variables group into one package and reduce the data transfer time. When a structure corresponding to the Nextion environment is used, the value of each variable is passed as a separate package. Each byte of this packet is an ASCII character code. The package structure looks like

$$\text{Variable\_name.val} = \text{Value\_of\_variable 0xFF 0xFF 0xFF},$$

where val - shows access to the value of the variable; 0xff,0xff,0xff - three bytes that indicate the end of the packet.

When such a packet structure is used, the data transfer rate is determined by the expression

$$R_{inf} = \frac{N_{infinf\ char}}{N_{name\ var}+N_{infinf\ char}+7} \cdot 0.8 \cdot R_{interface},$$

where R_{inf} - is the rate of useful information transmission (values of variables), bit/s; $R_{interface}$ - UART interface data transfer rate, bps; $N_{infinf\ char}$ - the number of characters in the variable name; 7 - the number of bytes, which consists of the packet end and the group of characters '.val'. If we use variables with a name length of 2 characters, then the simplified formula for determining the information transfer rate is

$$R_{inf} \approx 0.286 \cdot R_{interface},$$

With a maximum screen baud rate of 921600 bps, up to 8200 variable values can be transferred in one second. In operator panels from companies such as Siemens, GE Fanuc, etc., when displaying the status of an automation system on the screen, the average number of variable values per screen is up to 100-150. The default value of the data update period on the screen is 1s. Less commonly, this value is set to 0.5s. Thus, the amount of data that can be transferred between the controller and the screen is sufficient to update the information on the display in a timely manner.

When values of string variables are passed, the frame format that is used in the operating environment of the display

$$\text{Variable}_{\text{name}}\text{.val} = \text{“Value}_{\text{of}_{\text{variable}}}\text{”}\ 0xFF\ 0xFF\ 0xFF\ .$$

The rate of useful information transmission is determined by the expression

$$R_{inf} = \frac{N_{infinf\ char}}{N_{name\ var}+N_{infinf\ char}+9} \cdot 0.8 \cdot R_{interface},$$

where $N_{infinf\ char}$ - is the number of characters in the string variable.

The transfer of variable values from the microcontroller to the display is carried out only in the format corresponding to the Nextion displays.

III. Development of ALGORITHMS and Research Results

The NX4832K035 display was chosen for research (Fig. 1).

Fig. 1. The appearance of the Nextion display.

The frequency of its built-in controller is 108MHz, the amount of flash memory is 32MB, the amount of RAM is 8kbytes, the screen resolution is 480x320, the amount of EEPROM memory is 1kbytes. The STM32F407 microcontroller was chosen to connect with the display. When studying the efficiency of information transfer between the display and the STM32 microcontroller, its UART interface was configured to transfer data at speeds of 460800 and 921600 bps. An array from 70 packets was formed in the microcontroller memory, which corresponded to 70 values of 32-bit internal display variables. In this case, the length of the array was 1050 bytes, and the transmission time was 10ms. Such a period for updating the values of variables made it possible to implement a check for missing received data in the interrupt handler from the timer in the display. To create a continuous stream of transmitted data, the array was transmitted in a cyclic mode using a DMA channel [9]. In each transmission cycle, the value of each variable was increased by 1. This made it possible, when processing the received information on the display, to detect a gap in the received value. The research results are presented in table 1.

TABLE I. PROBABILITY OF MISSING DATA DEPENDING ON THE TRANSMISSION RATE

Data transfer rate, bps	Probability of missing variable value
460800	0
921600	0.001

An unshielded cable 10 m long was used to connect the microcontroller and the display. As can be seen from Table 1, there are no gaps at speeds up to 460 kbps. At 921600 bps, no more than 1 pass per 960 packets is possible. Such gaps can be explained by the limited speed of the display processor, since the STM32F407 microcontroller operates in DMA mode. When a skip occurs, no errors occur inside the display, and work continues normally.

To estimate the refresh rate of graphic images on the screen, the following algorithm was developed:

1) The screen is loaded with an image resolution equal to the screen resolution.

2) A timer has been added to the project, and in the timer interrupt handler, the image is read and displayed on the screen.

3) The interval of the timer every 2s decreased by 1ms and a check was made - whether the display had time to display the image completely.

4) Test steps 1-3 were repeated for the case where 10 images are displayed. Each of the images has an area equal to 1/10 of the screen area.

In the screen refresh rate study, high-speed screen capture was performed at a frame rate of 120 fps. Further analysis of artifacts in the image made it possible to estimate the limiting value of the update rate without artifacts.

The results of the update time estimation are presented in the form of Table 2.

TABLE II. SCREEN REFRESH TIME

Number of images	Image size, pixels	Loading and display time, ms
1	480x320	324
10	48x320	352

Analysis of the data in Table 2 shows that the speed of updating images in the design of the screen meets the requirements of automated control systems.

To analyze the requirements for the amount of RAM and flash memory, a project was developed. It contained 7 screens of varying complexity. The simplest screen is the settings selection menu and contains only 3 widgets that allow you to go to the corresponding settings screens. The appearance of the simplest screen is presented in Fig. 2.

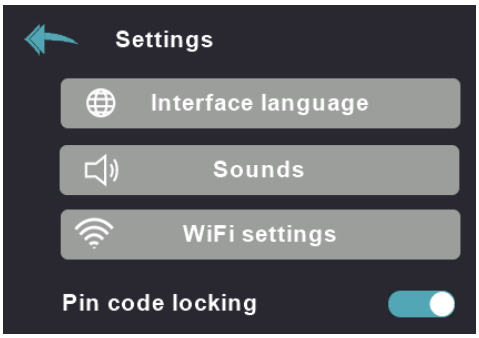

Fig. 2. The appearance of the simplest screen.

The most complex screen contained 12 widgets and 32 images corresponding to the states of these widgets. Most widgets had 4 images corresponding to their states. The project had multilingual support for up to 10 languages. The captions over the images were superimposed in accordance with the current interface language. The rest of the screens were marginally simpler than the main screen. Table 3 shows the data on the development time of the screens and the results of their compilation.

TABLE III. RESULTS OF DESIGN TIME AND COMPILATION OF PROJECT SCREENS

N screen	Flash memory size, kb	RAM size, bytes	Number of images	Number of widgets
1	71	400	1	2
2	106.6	712	26	15
3	32	508	29	12
4	46	680	10	7
5	16	608	4	8
6	28	756	8	14
7	30	672	6	6

The analysis of the data obtained shows that for storing images of one screen, the average value of the flash memory amount is 47 kbytes, and 612 bytes of RAM. Then the resources of the selected screen model are enough to create 13 full-fledged screens with design. Based on the characteristics of Nextion displays, the maximum number of screens is much more limited by the amount of RAM than flash memory.

This project is an updated and improved version of a previously developed project based on the demo board Makerbase MKS TFT35. Comparison of the screens

development time on Nextion and Makerbase MKS TFT35, shows that without the usage of development environments for graphic screens that are compatible with displays, it takes 5-6 times more time to create one screen and animate it.

IV. Conclusions

Nextion displays are great to visualize the status and control electronic devices in addition to displaying images from a video camera. In combination with STM32 microcontrollers that don't have an integrated graphics accelerator, they are an excellent low-cost solution. To obtain a high-quality screen design, you should use the original widget images loaded from flash memory. The time of loading images and displaying them on the screen does not exceed 352ms, which corresponds to the visualization standards for automated process control systems (APCS) operator panels. At UART speeds up to 460kbps, the display can process a continuous stream of packets with variable values. At a speed of 921600bps, the probability of missing a value does not exceed 10-3. At the maximum exchange rate, up to 8200 values of 32-bit variables can be transferred per second. This value is sufficient to visualize a system or control device of any complexity. In the operator panels of APCS, up to 100-150 variable values are displayed on one screen. It takes 1-2 days to learn the Nextion Editor environment. It takes 4-5 days to create one screen and related scripts, which is 5-6 times less than without using the interface development environment. The cost of Nextion displays is 5-8 times less compared to operator panels for process control systems from Siemens.

Thus, Nextion displays fully satisfy the requirements for visualization that are imposed on the development of new electronic devices by developers: development of visualization projects with original graphic design in a short time, quick acquaintance with the project when changing the programmer-developer.

References

[1] Gérard Bouvet, GeeseWare Dominique Jugnon "Introducing a Graphical User Interface to Your Embedded Application" STM32 Journal, vol.2, issue 2, 2022, pp.42-46.

[2] Yang Li, and Yunliang Wang "Design of control system of Smart Home based on embedded Linux" International Conference on Information Sciences, Machinery, Materials and Energy, 2015, pp.1402-1405.

[3] Nishu Patel, Ekata Mehul "Low End Human Machine Interface (HMI) Display using ARM Cortex M4 Based Controller" International Journal of Science Technology & Engineering, vol. 1, issue 12, 2015.

[4] Luyong Ren, Xiaoyu Yu "Hardware Implementation of STM32 Microcontroller-Based Indoor Environment Monitoring System" Open Journal of Applied Sciences, vol.11 no.9, 2021, pp.997-1008.

[5] G. İşnas and N. Şenyer, "Comparison of TouchGFX and LVGL Embedded Hardware GUI Libraries", Gazi University Journal of Science Part C: Design and Technology, 2021, vol. 9, no. 3, pp. 373-384.

[6] Dr. Antonio Carlos Bento "Nextion Tft Development an Experimental Survey for Internet of Things Projects" International Journal of Advance Research in Computer Science and Management Studies, vol. 8, issue 11, November 2020, pp. 1-9.

[7] Antonio Carlos Bento "An Experiment with Arduino Uno and Tft Nextion for Internet of Things" 2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE), 2018, pp. 1-5.

[8] Antonio Carlos Bento, Norberto dos Santos, José Carmino Gomes Júnior "An Experiment with 3 Layers Development forIoT with NodeMCU12e + Nextion" International Journal of Advanced Engineering Research and Science, vol.5, issue 11, November 2018, pp.183-189.

[9] Patrik JACKO, Dobroslav KOVÁČ "The DMA controller description and configuration of the STM32 microcontrollers" Journal of Industrial Electrical Engineering, vol.2, issue 1, 2018, pp.83-88.

[10] Програмування мікроконтролерів STM32 в середовищі STM32CubeIDE в прикладах і задачах: Навч. посіб. / О. В. Зубков, І. В. Свид, О. В. Воргуль, В. В. Семенець. Дніпро : ЛІРА ЛТД, 2022. 144 с.

[11] I. Svyd, V. Semenets, O. Vorgul, and I. Shevtsov, "Aspects of STEM education in the design of devices on microcontrollers and FPGAS," Theoretical and Applied Aspects of Device Development on Microcontrollers and FPGAs 2022, 2022. doi:10.35598/mcfpga.2022.018.

[12] O. Zubkov, I. Svyd and O. Vorgul, "Features of the Digital Filters Implementation on STM32 Microcontrollers", 2021 III International Scientific and Practical Conference Theoretical and Applied Aspects of Device Development on Microcontrollers and FPGAs, 2021. doi: 10.35598/mcfpga.2021.001.

[13] O. Zubkov, I. Svyd and O. Maltsev, "Features of the use of PID controllers when controlling evaporators", 2020 II International Scientific and Practical Conference Theoretical and Applied Aspects of Device Development on Microcontrollers and FPGAs, 2020. doi: 10.35598/mcfpga.2020.001.

[14] O. Zubkov, I. Svyd, and O. Vorgul, "Features of the implementation of an over/under voltage relay on STM32 microcontrollers," Theoretical and Applied Aspects of Device Development on Microcontrollers and FPGAs 2022, 2022. doi:10.35598/mcfpga.2022.001.