Security
of Infocommunication Systems
and Internet of Things

2024 Vol 2, No 2

https://doi.org/10.31861/sisiot2024.2.02011

Received 18 December 2024; revised 25 December 2024; accepted 28 December 2024; published 30 December 2024

SPI-FSMC Expander with DMA Mode Support

Andrii Yarmilko[*]

Department of Automated Systems Software, Bohdan Khmelnytsky National University of Cherkasy, Cherkasy, Ukraine

*Corresponding author (E-mail: a-ja@vu.cdu.edu.ua)

ABSTRACT The article provides further development of software and hardware solutions for building fully functional Industrial Internet of Things (IIoT) systems based on high-speed SPI. The possibility of overcoming the obstacles caused by the limitation of the set of input/output lines to a more complete use of the redundant computing resource of Wi-Fi modules outside the provision of "access point – station" information exchange for solving problems related to other aspects of the functioning of the embedded system has been investigated. It has been established that the current issue is to expand the basic capabilities of embedded system components, taking into account the provision of comprehensive access to the SPI bus for all the functionality of typical IIoT systems. The implementation of the proposed hardware and software solutions was carried out in the form of an expander module for the basic capabilities of the SPI bus, designed to intensify information exchange over the Wi-Fi channel between the SoC controller and peripherals: IO and LCD extensions, SD cards, DRAM and SPI-Flash file storage. The SPIExpander hardware driver and the MultySwitcher software driver for its control were developed. In software driver developing was implement the HAL paradigm and was used RTOS methods to prevent abnormal situations of multiple access on the bus. The advantages of introducing the high-speed SPI bus as opposed to the STM-FSMC interface are demonstrated in the example with the implementation of their HMI components into the structure of the embedded device. SPIExpander was used as a system proxy driver that implements the function of a network switch, with controlled by DrvLoadBlock software driver DrvSPI2LCD hardware adapter driver connecting on the one of its serial ports for converts DMA packets into a stream of 16-bit words. Hardware stability has been increased and the set of input lines for generating LCD control system signals and conducting DMA block operations has been minimized. The software model of the driver for the implementation of polycore and multithreaded information exchange is presented. Recommendations for using the SPI-FSMC expander are provided.

KEYWORDS host controller, SPI bus, embedded system, IIoT, human-machine interface.

I. INTRODUCTION

A typical solution in the development of embedded systems is the integration of Wi-Fi modules into their structure. They provide convenient communication of system devices at the level of Wireless LAN or remote access through a network router. However, the use of modern Wi-Fi modules exclusively for information exchange "access point – station" is not effective, since such a task requires only 10 – 15% of its computing power. Thus, the excess resource in the typical operation modes of Wi-Fi modules is more than 80%. A rational solution is to use this resource to solve problems related to other aspects of the functioning of the embedded system. In general, it can refer to the work with SSD devices, human-machine interaction support and utilitarian tasks (e.g., technological operations control of production equipment). In practice, a greater utilization of this Wi-Fi modules' resource is complicated by a limited set of input/output lines (GPIO). It is this fact that determines the relevance of further research and development.

We have already carried out research aimed at building two-level embedded control systems based on Wi-Fi modules with a limited pin resource. Achieving the set goals is carried out on the basis of SPI bus through the development of an external host-controller of the high-speed SPI-bus and the synthesis of the appropriate control software model [1].

The purpose of the research presented in this article was the synthesis of hardware and software solutions to obtain integrated or multi-level Industrial Internet of Things (IIoT) systems according to the concept [2]. The focus was on issues related to minimizing the complexity of both hardware and software drivers, while simultaneously maintaining the ability to perform streaming operations in DMA mode. The research was carried out, in particular, with an orientation towards the use of the aforementioned resource of Wi-Fi modules to ensure effective graphic output on the LCD display of IIoT systems.

II. BACKGROUND OF RESEARCH

The miniaturization of computing devices and their body combination with a set of peripherals led to the emergence and spread of embedded systems (EmSYS), which had a positive impact on the manufacturability, operational efficiency, and reliability of personal, home, and industrial devices [3]. In turn, with the increasing level of the global computer network coverage density and ramifications of local computer networks, EmSYS is intensifying the transition to the IIoT. In addition to the steady trend of decreasing the cost of electronic components, it is also facilitated by the intensification and hierarchical organization of management processes [4]. Today, even the embedded system of a separate machine is two-level. Usually, the lower-level subsystem is made on a microcontroller and works in hard real time (HRT) mode, directly supporting the production process within optimal limits. The upper-level subsystem solves the tasks of the machine-ICS operator-machine interaction, and in fact, implements the terminal function of the human-machine interface (HMI). At the same time, deepening the automation of modern productions distances the operator from the machine, thus, transforming them into IIoT technology [2].

These transformations are closely related to the evolution of human-computer interaction and were manifested in the change of HMI. Given that the visual channel of interaction is more significant, the development of the graphical user interface (GUI) software component is, without exaggeration, decisive [5]. Note that modern microcontrollers, in particular STM32, have hardware support for the Flexible Static Memory Controller (FSMC) interface [6]. Having such a component simplifies HMI implementation. However, connecting the LCD via a parallel bus requires a significant amount of pin resources, which negatively affects the ability to optimize the economic parameters of the proposed solutions.

As a rule, during the functioning of the embedded system, it is necessary to solve the task of group use of hardware resources in real time. In addition to the synchronization of application-level software processes using RTOS semaphores and mutexes, it is necessary to ensure the stability of the process flow at the driver level in conditions of comprehensive access to hardware resources. So, for the case of application as a tool of CNC devices control, the upper level embedded systems must implement the following functions: serve the LCD of the process monitor, access to SD and SPI-Flash storage devices, polling the keyboard and displaying binary states, communication via Wi-Fi, wired Ethernet or USB. It is not always possible or advisable to allocate a separate physical port to implement these functions, especially when implementing an embedded system based on a Wi-Fi module. For this reason, it is important to expand the basic capabilities of components, taking into account the provision of comprehensive access to the SPI bus for all the functionality of typical IIoT systems.

III. HARDWARE DRIVER SPIEXPANDER

In view of the above, the implementation of IIoT systems on the Wi-Fi module platform is impossible without the use of the SPI bus expansion hardware driver. Its main purpose should be the switching of peripheral devices. This development was carried out taking into account the functional features of the ESP12F module and the principles presented in [1].

The circuit of the SPIExpander driver is shown in Fig. 1. It's proposed to implement it in the form of a separate device – a motherboard with an SPI bus connecting the main functional components. The main elements of the SPIExpander driver and their purpose:

- DD1 decoder, which converts the binary code of the address bus A2, A1 and A0 into a linear one. It ensures unambiguity of peripheral device selection;
- A set of DD2 buffer elements, which serves as a switch

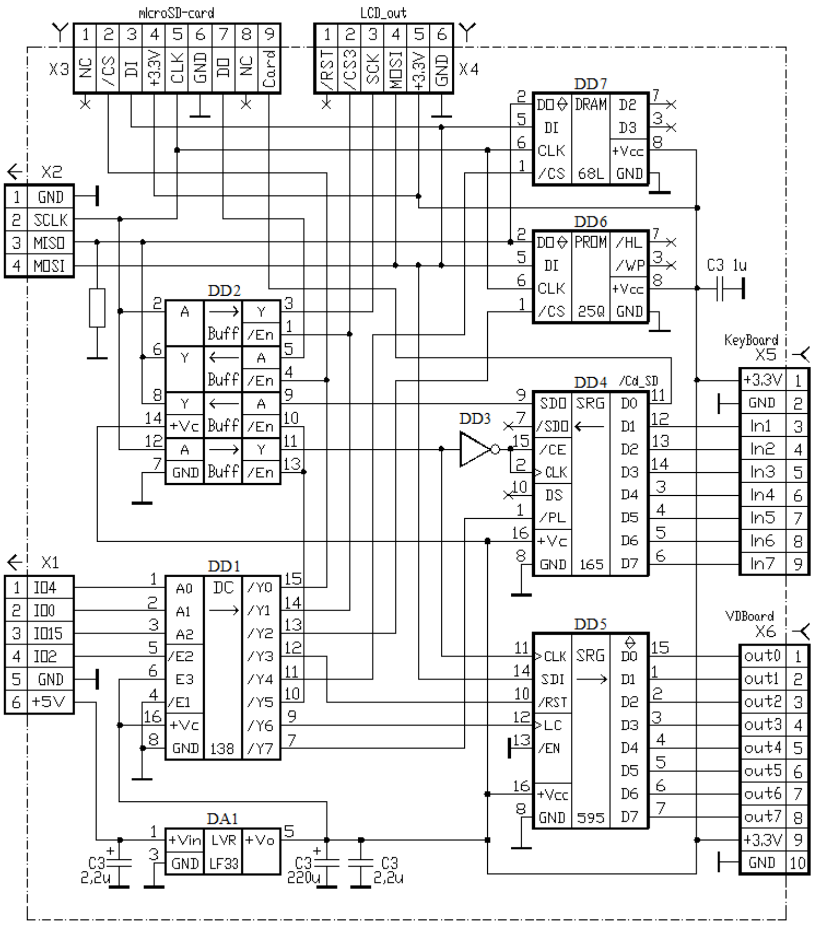

FIG. 1. Circuits of the SPIExpander.

for SCK and MISO signals of the SPI bus for peripherals that either do not have a Chip Select (CS) cascade, or their multimodality (SPI/DPI/QPI) does not guarantee the formation of a high-impedance state on the output lines;

- DD3 inverter, which converts the SCK signal from SPI0.0 mode to SPI1.0, which is caused by the SDO carry phase on the rising edge of CLK shift register. In SPI0.0 mode, data can only be modified by the falling edge of SCK;
- DD4 shift register, designed to convert the 8-bit word of the parallel input Dx to the serial signal SDO. It provides reading of the states of the on-board keyboard and the /CdSD (Card SD Insert) signal of the memory card;
- DD5 shift register with SDI serial input and 8-bit parallel output, which provides identification of the system components states, for example, performing file operations of an SD card connected to the X3 slot or an embedded SPI-Flash drive DD6, accessing to the DRAM of DD7 program memory buffering chip.

The functional multiplexity of GPIO lines of Wi-Fi modules, like all SoC and MCU devices, allows you to change their purpose at different stages and modes of operation, and not only by software. For example, the lines io15=0, io0=1 and io2=1 put the controller in the code loading mode from the on-board SPI-Flash – the standard launch of the embedded program, and with the values io15=0, io0=0 and io2=1 they load this system to the on-board SPI-Flash by the stream from UART port. In addition, in the ESP12F module, which is used as a research platform, the io2 line is hardware-connected to the on-board blue LED and can be used programmatically (activ.Low) to display the status of system processes.

The order of signals assignment is determined by the criterion of optimization of address sampling bit manipulations. The importance of using this approach is as follows. ARM7 SoC and MCU have an end-to-end addressing map, that is, program code memory, RAM and peripheral registers, which include GPIOs in particular, belong to the same physical address space. Registers are 32-bit (machine word). There are two ways to set bits in a word. First, programmatically of masks manipulating. Address formation is carried out according to the scheme: load the GPIO address into the register file (RF), read the value, then – masks to clear the corresponding bits, after that the AND operation is performed; the bit setting mask is loaded into the RF and a logical OR operation is performed. From the point of view of computing resources usage such a process is very expensive, but the main problem is the difficulty of ensuring the atomicity of the process. Handling of hardware interrupts or RTOS task switching can have unintended consequences because the masking sequence can be interrupted and intermediate values in the peripheral register will be modified. For such risks, preference should be given to the Bit-banding method, which is recommended for developers by the ARM-Limited company [7]. Programmatically, the method is usually implemented through a macro, which after the assembly preprocess is transformed into the code form of the direct addressing operation [7].

IV. SOFTWARE DRIVER MULTYSWITCHER

The MultySwitcher software driver, which manages the SPIExpander hardware driver, was developed according to the HAL paradigm, that is, it is abstracted from the hardware except for the macro definitions in the header file of the driver library. The hardware system resources to which SPIExpander belongs, such as a router of asynchronous information flows and data, is a heavily loaded and critical resource. Given that collision detection on the bus and elimination of their consequences is a complex task that requires significant processing power of the processor, it was decided to use RTOS methods to prevent abnormal situations of multiple access. This approach became decisive in obtaining the software model of the MultySwitcher driver (Fig. 2).

Calls to the SPIExpander driver from such processes as performing an operation with non-volatile storage devices, writing/reading DRAM, displaying data on the LCD display and binary states on VD-indicators, reading the state of physical buttons are asynchronous. Process priorities are different. It is the highest for DRAM access, the lower for operations with SD cards and SPI-Flash drives. The lowest priority is considered for operations with the LCD display, VD indicators and keyboard. A common feature of working with non-volatile drives and LCD display is the need to ensure the continuity of operations, that is, removing the /CS# signal automatically ends the operation cycle.

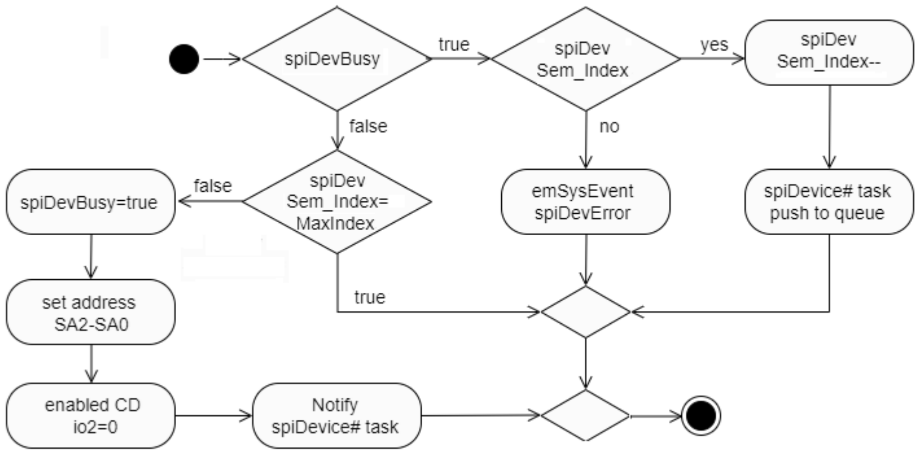

FIG. 2. The MultySwitcher software model as UML activity diagram.

Premature cycle termination, in most cases, violates the integrity of the data without the possibility of determining the resulting consequences programmatically. For this reason, receiving the io2=1 selection signal of the decoder and completion of the task is carried out in the body of the task itself.

The software model of the MultySwitcher driver consists of three branches: maintenance of the queue of access tasks to system peripheral devices (left part of access tasks to system peripheral devices (left part of the diagram), idle passage (in the center of the diagram) and providing access to the current tasks of the queue or the task that caused the activation of the driver to the corresponding hardware resource. In general, the algorithm of the MultySwitcher driver can be considered as a task manager, which suspends the execution of the task and manages the task renewal queue using the Notify mechanism. The priority of tasks is defined by the user by RTOS means, that is, MultySwitcher as a separate task – the access manager has a lower queue management priority than the RTOS core.

V. APPLICATION OF SPIEXPANDER TO PROVIDE HMI GRAPHICS OUTPUT

The advantages of introducing the SPI bus expander into the structure of the embedded device become significant with the implementation of their HMI components, namely in displaying data on the LCD display.

As practice shows, the applied implementation of embedded systems imposes additional requirements and requires more complex and comprehensive solutions. In particular, it was noted in [6] that in normal mode the embedded system functions autonomously, and access to services can be carried out through the on-board HMI, which includes an LCD monitor, a clock touch panel and buttons, several non-volatile drives provide storage for configuration files, control program, bitmap sprites and recipes. That is, we are talking about serial-multiple access to the SPI bus under the control of the host controller implemented according to the decoder topology. Instead, in the current study, a more progressive solution based on a controller – SPI expander is proposed as a system proxy driver.

A. Hardware driver DrvSPI2LCD.

SPIExpander was used as a system proxy driver that implements the function of a network switch. The DrvSPI2LCD hardware adapter driver, which converts DMA packets into a stream of 16-bit words, is connected to one of its serial ports. The resulting circuitry of the DrvSPI2LCD driver is shown in Fig. 3. The synthesis of control signals is presented on the basis of the time diagram (Fig. 4).

The adapter driver consists of two digital cascades: shift registers and a system signal synthesizer for controlling the LCD matrix and data buffering. Shift registers D3 and D4 are serially cascaded by the SQ7-SDI signal and implement the function of converting the serial MOSI signal of the SPI bus into a parallel bus of a 16-bit LCD control word and RGB data. The synthesizer of system signals forms strobes from SCK synchronous pulses: STCP – the control signal of data parallel transfer register from the cascades of the serial shift register to the latch register, which are components of microcircuits D3 and D4; D/C_LCD – LCD input word type selection signal: logical 1 indicates RGB data, and logical 0 switches the matrix to the command packet reading mode; WR_LCD is a timing signal for writing data to the control register of the controller or writing to the GRAM LCD.

In our previous research [1], the problem of data buffering was solved by a monovibrator and an RC circuit. However, there was instability due to the variability of the input capacitance of the R/C pin and the temperature drift of the timing resistor. According to the results of our research, a radical improvement of the synthesis of system signals was achieved due to the two-stage cycle of writing a word. In accordance with the tested concept, the formation of the buffering strobe of the 16-bit command word or STCP data is carried out on the leading edge of the 16th SCLK pulse with a delay of tpd= 4 – 10 ns on the D3.2 inverter. Loading the word into the LCD matrix is carried out already at the second stage according to the rising signal WR_LCD, which is formed on 8 SCLK pulses. Thus, the clock frequency of the LCD matrix may not exceed 10 MHz, even at the maximum SPI bus frequency of 80 MHz.

The DrvSPI2LCD driver works like this. The /CS3 line of the X1 connector receives log.0 from the SPIExpander driver – the signal for selecting the peripheral device of the LCD matrix. As a result of the inversion of the /CS3 signal by the D3.1 element, a logic unit appears at the R inputs, which transfers the triggers D2.1 and D2.2, the binary counter D1 and the registers D3 and D4 from the reset state to the normal mode of operation. The /CS3 signal must be held constant until the data transaction is complete. The following SCK pulses shift the logic values of the MOSI line sequentially: first through the internal stages of the D3 register, then through the SQ7-SDI connection and the internal stages of the D4 register. When 16 SCK pulses pass on the Q8 line, the counter D1 will fall from log.1 to log.0, this event is the formation of the STCP gate – loading of states from the internal shift cascades to the output register due to the inversion of the falling edge into the rising edge (LC line of registers dynamic), which causes the data to be latched at outputs Q0-Q7 of both registers. At the same time, the STCP signal will cause the transfer of log.1 from the D input to the Q output of the D2.1 flip-flop, which will put the 2I-Hi element into the active state of transmitting the falling edge on every subsequent eighth SCK pulse. Thus, when a signal with a rising edge is formed on the Q8 line of the binary counter, it, after passing through two inverters D3.3 and D3.4, will be fed to the D_WR line of the X2 connector – the connection of the LCD matrix as a WR_LCD signal. The first rising edge at the input C of the trigger D2.2, which is the 24 SCK pulse, will lead to the transfer of the line from log.0 – the data DB0-DB15 is the command word, on D/C_LCD=1 – the data word is the command parameters or writing to GRAM LCD.

B. The DrvSPI2LCD software driver DrvLoadBlock.

The DrvSPI2LCD hardware driver is controlled by the DrvLoadBlock software driver. The algorithm of its work is linear. When performing the basic operation of loading

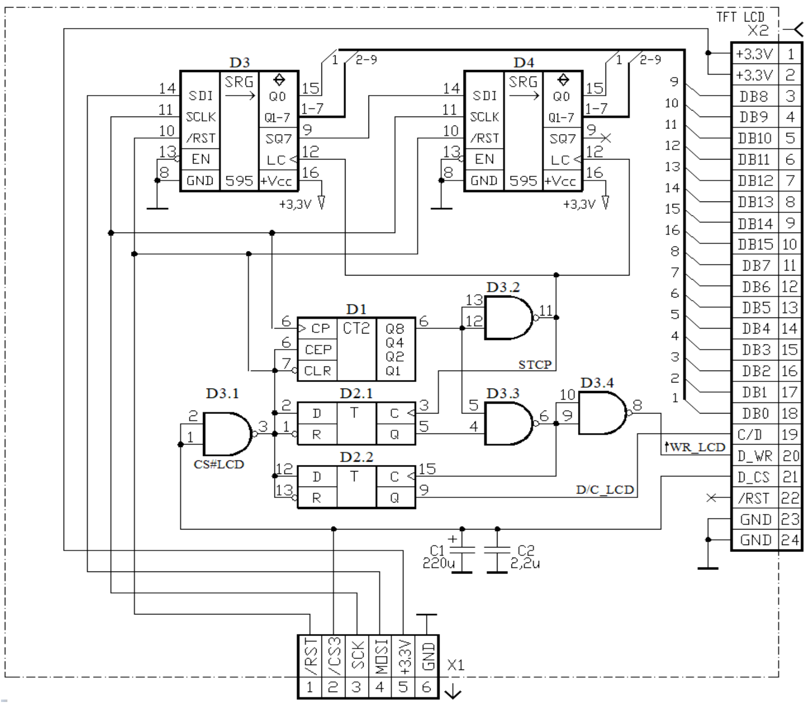

FIG. 3. Circuits of the DrvSPI2LCD.

a graphic sprite into the GRAM LCD, the transfer of the graphic block is carried out in six phases. Selection of the DrvSPI2LCD driver is achieved by setting the SA2-SA0 address bus bits and coloring the E2=0 line of the decoder, which causes the /CS3=0 line to shift. The 16-bit word 0x002A of the Column Address Set command is loaded into the SPI transmitter and the hardware transfer is carried out. In the same way, two words are transferred: the addresses of the start and end columns of the sprite. The decoder lines are programmatically transferred to the state E2=1, and after 10 system clocks they return to E2=0 again. The code 0x002B of the Page Address Set command and two words are transmitted: the addresses of the start and end lines of the sprite. The controller of the SPI module of the SoC is put into DMA mode and the cycle of loading the data block into the GRAM LCD is started.

Since the E2 line is programmatically switched and bytes are loaded into the SPI transmitter when forming the Column Address Set and Page Address Set commands, the sprite block transfer should be characterized as a quasi-DMA mode. However, this circumstance does not significantly reduce the efficiency of packet data transmission on the LCD. For example, when displaying only one current coordinate of a 3D printer extruder, which is equivalent to 6 digits and an area of 2860 pixels, only the

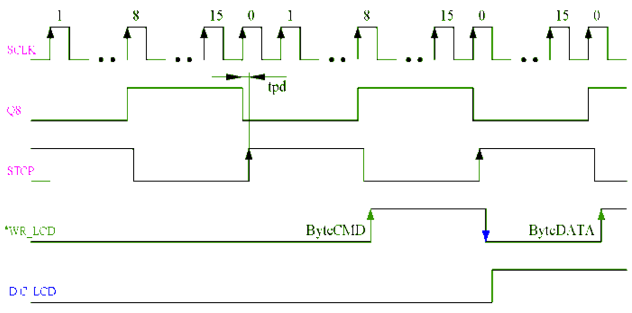

FIG. 4. The timing diagram of DrvSPI2LCD driver activity.

first 7 words are transmitted with software clocking, that is, the efficiency reaches 99.8 %.

VI. CONCLUSION

Hardware and software solutions for building fully functional IIoT systems based on high-speed SPI are offered. They implemented in the form of an expander module for the basic capabilities of the SPI bus, designed to intensify information exchange over the Wi-Fi channel between the SoC controller and peripherals: IO and LCD extensions, SD cards, DRAM and SPI-Flash file storage. The resulting circuitry is simple to implement in a discrete miniboard design. However, it is more expedient to implement the SPI expander on the basis of FPGA or in the form of specialized integrated circuits. This will allow MCU and SoC processors to acquire the characteristics of full-fledged terminal systems with combined control, both local and IIoT remote.

Representation of the SPI expander in the research as an FSMC device is due to the problem of conjugation of LCD matrices with a resolution of 640 × 480 and higher to obtain HMI components of embedded systems. Hardware conversion of the SPI mode video stream to RGB16 improves the display of screen widgets, since the built-in 3/4-line Display Serial Interface is limited to a clock frequency of 10 MHz.

The above research results are valuable for the creation of IIoT systems based on Wi-Fi-SoC modules, for example, ESP12F, ESP32-Wroom, Murata LBWA1KL1FX or similar.

AUTHOR CONTRIBUTIONS

A.Ya. – conceptualization, methodology, investigation; writing (original draft preparation), writing (review and editing).

COMPETING INTERESTS

The author declare no competing interests.

REFERENCES

[1] A. Yarmilko, "High-Speed SPI Bus Host Controller for Embedded Systems," in *Proc. 2022 IEEE 41st Int. Conf. Electron. Nanotechnol. (ELNANO)*, Kyiv, Ukraine, 2022, pp. 662–666, doi: 10.1109/ELNANO54667.2022.9927055.

[2] A. Karmakar, N. Dey, T. Baral, M. Chowdhury and M. Rehan, "Industrial Internet of Things: A Review," in *Proc. 2019 Int. Conf. Opto-Electron. Appl. Opt. (Optronix)*, Kolkata, India, 2019, pp. 1–6, doi: 10.1109/OPTRONIX.2019.8862436.

[3] R. Budihal, “Emerging trends in embedded systems and applications,” 2010. [Online]. Available: https://www.embedded.com/emerging-trends-in-embedded-systems-and-applications/. Accessed on: Nov. 4, 2024.

[4] A. J. Gapinski and Z. J. Czajkiewicz, “Automated Manufacturing: Processes and Technologies,” *Int. Conf. Eng. Comput. Educ.*, Buenos Aires, Argentina, 2009. [Online]. Available: https://www.researchgate.net/publication/319990699_Automated_Manufacturing_Processes_and_Technologies. Accessed on: Nov. 4, 2024.

[5] A. Kalnoskas, “Embedded GUI library for professional HMI development,” 2018. [Online]. Available: https://www.microcontrollertips.com/emwin-embedded-gui-library-for-professional-hmi-development/. Accessed on: Nov. 4, 2024.

[6] *AN2784 Application note. Using the high-density STM32F10xxx FSMC peripheral to drive external memories.* [Online]. Available: https://www.st.com/resource/en/application_note/cd00200423-using-the-highdensity-stm32f10xxx-fsmc-peripheral-to-drive-external-memories-stmicroelectronics.pdf. Accessed on: Nov. 10, 2024.

[7] *ARMDeveloping. Chapter 3. Programmers Model. Bit-banding.* [Online]. Available: https://developer.arm.com/documentation/ddi0439/b/Programmers-Model/Bit-banding. Accessed on: Nov. 10, 2024.

Andrii Yarmilko

PhD in Information Technologies, Docent, Associate Professor at Department of Automated Systems Software, Bohdan Khmelnytsky National University of Cherkasy. Research interests: computer vision and pattern recognition, industrial internet of things, human-machine interaction, information security and dependability of embedded systems. Author of more than 100 publications.

ORCID ID: 0000-0003-2062-2694

Експандер SPI-FSMC з підтримкою режиму DMA

Андрій Ярмілко[*]

Кафедра програмного забезпечення автоматизованих систем, Черкаський національний університет імені Богдана Хмельницького, Черкаси, Україна

*Автор-кореспондент (Електронна адреса: a-ja@vu.cdu.edu.ua)

АНОТАЦІЯ Надано подальшого розвитку апаратним та програмним рішенням для побудови повнофункціональних систем IIoT на базі швидкісного SPI. Досліджено можливість подолання перешкод, спричинених обмеженістю набору ліній введення/виведення (GPIO), до більш повного використання надлишкового поза забезпеченням інформаційного обміну «точка доступу – станція» обчислювального ресурсу WiFi-модулів для вирішення задач, пов’язаних з іншими аспектами функціонування вбудованої системи. Актуальним є розширення базових можливостей компонентів вбудованої системи із врахуванням забезпечення комплексного доступу до шини SPI за всім функціоналом типових IIoT-систем. Реалізацію запропонованих апаратних та програмних рішень виконано у форматі модуля-експандера базових можливостей шини SPI, призначеного для інтенсифікації інформаційного обміну по WiFi-каналу між SoC-контролером та периферією: IO та LCD розширеннями, SD-картами, DRAM і SPI-Flash файловими сховищами. Розроблено апаратний драйвер SPIExpander та програмний драйвер MultySwitcher для управління ним. В розробці програмного драйвера застосовано парадигму HAL та використано методи RTOS задля запобігання нештатних ситуацій множинного доступу до шини. Переваги застосування швидкісного SPI на противагу інтерфейсу STM-FSMC продемонстровано на прикладі реалізації HMI-компонентів вбудованих систем. SPIExpander застосовано в якості системного проксі-драйвера для реалізації функції мережевого комутатора, до одного з послідовних портів якого підключається керований програмним драйвером DrvLoadBlock апаратний драйвер-адаптер DrvSPI2LCD для перетворення DMA-пакетів у потік 16-розрядних слів. Апаратно підвищено стабільність та мінімізовано набір вхідних ліній генерації системних сигналів управління LCD і проведення DMA-блочних операцій. Представлено програмну модель драйвера для реалізації поліядерного та мультипотокового інформаційного обміну. Надано рекомендації щодо використання експандера SPI-FSMC.

КЛЮЧОВІ СЛОВА хост-контролер, шина SPI, вбудована система, IIoT, людино-машинний інтерфейс.