Omega Journal of Linguistics and Language Studies
ISSN: 3072-4678
Volume 01, Issue 03, June, 2026
Website: https://omegajournals.org

ANALYSIS OF THE ELECTRONIC DATA EXCHANGE PROCESS THROUGH THE USB PORTS OF A COMPUTER

R. J. Boymanov
Institute of Information Communication
Technologies and Military Signals

Abstract
This article is dedicated to the process of data exchange between a device and a host via the USB interface. It analyzes the protocols and their structure used in the data exchange process over the USB interface. The types of packets, frames, transactions, and their practical applications in data exchange are discussed.

Keywords: USB device, USB 2.0, USB 3.0, packet, frame, transaction, USB bus, PID, USB interface.

Introduction

Аннотация:
В данной статье рассматривается процесс обмена данными между устройством и хостом через интерфейс USB. В ней проанализированы протоколы и их структура, используемые в процессе обмена данными по USB-интерфейсу. Обсуждаются типы пакетов, кадров, транзакций и их практическое применение в обмене данными.

Ключевые слова: USB-устройство, USB 2.0, USB 3.0, пакет, кадр, транзакция, шина USB, PID, USB-интерфейс.

Annotatsiya:
Ushbu maqola USB interfeysi orqali qurilma va xost o‘rtasida ma’lumot almashish jarayoniga bag‘ishlangan. Bunda USB interfeysida ma’lumot almashish jarayoni hamda bunda foydalaniluvchi protokollari, ularning tuzilishi

Omega Journal of Linguistics and Language Studies
ISSN: 3072-4678
Volume 01, Issue 03, June, 2026
Website: https://omegajournals.org

tahlil qilingan. Ma'lumot almashinuvida paket, kadr, tranzaksiyalar turlari va ularning qo'llanilish amaliyoti ko'rib chiqilgan.

Kalit so'zlar: USB qurilma, USB 2.0, USB 3.0, paket, kadr, tranzaksiya, USB shina, PID, USB interfeysi.

Today, nearly every person possesses the skills required to use a personal computer. One of the most common methods of transferring electronic data from a personal computer is through external storage devices. In this process, USB (Universal Serial Bus) is considered one of the most efficient interfaces for data exchange. Understanding the stages of data transfer through a USB interface, as well as the protocols and processes involved, is of interest to every USB storage device user.

USB specifications have evolved through several generations. It all began with USB 1.0 and USB 1.1, followed by the introduction of the USB 2.0 standard, and later the latest USB 3.0 specification emerged [1]. First of all, several key performance indicators should be noted. For example, the USB 2.0 interface supports three operating modes:

High Speed – up to 480 Mbps

Full Speed – up to 12 Mbps

Low Speed – up to 1.5 Mbps

Control on the USB bus is performed by the host (for example, a personal computer), to which up to 127 USB devices can be connected. If this number is insufficient, an additional host must be added. Another important aspect is that a device cannot independently send or receive data from the host; instead, the host must initiate communication with the device.

The term "endpoint" is frequently used in discussions of the USB interface, although its meaning is often not clearly explained. An endpoint is a uniquely identifiable component of a USB device. Each device may have multiple endpoints. In general, an endpoint is a memory area (data buffer) within a USB device where data can be stored. Consequently, each device has a unique address on the USB bus, and each endpoint also has its own number.

Support for the USB protocol in personal computers is divided into three layers. The lowest layer includes the host controller driver, which provides the host with

Omega Journal of Linguistics and Language Studies
ISSN: 3072-4678
Volume 01, Issue 03, June, 2026
Website: https://omegajournals.org

capabilities for managing the connection process. It supports hardware initialization, transfer management, and processing of completed or interrupted transfers [2]. Any host controller driver implements a virtual root hub that provides hardware-independent access to the registers controlling the physical ports located on the computer.

The middle layer handles device connection and disconnection, device initialization, driver selection, communication channels, and resource management. These service layers also manage standard channels and requests transmitted through those channels.

The highest layer consists of device-specific (class) drivers for individual devices. These drivers implement protocols that operate on channels differing from the standard ones. In addition, they provide extra functionality that exposes the device to other kernel components or user-space applications. They utilize the USB Driver Interface (USBDI) provided by the service layer.

When the computer receives notification from the hub that a new device has been connected to a USB port, the service layer enables the port and supplies the device with 100 mA of power. At this stage, the device remains in its default state and listens on address 0. The service layer then continues by requesting various descriptors through the standard channel. Next, it sends a Set Address request to assign a new address to the device, moving it from the default address (address 0) to another unique address.

Several device drivers may support the same device. For example, a modem driver may support an ISDN terminal adapter through an AT-compatible interface. However, a dedicated ISDN adapter driver may provide better support for that device. To enable such flexibility, the recognition process returns priorities indicating the level of support. Support for a specific product version receives the highest priority, while a generic driver receives the lowest priority.

Furthermore, if a configuration contains multiple interfaces, several drivers may be attached to a single device. Each driver only needs to support a subset of those interfaces.

A device driver must anticipate errors during any data exchange with the device. The USB architecture supports and encourages the removal of devices at any time. Drivers must continue functioning correctly when a device is disconnected [3].

Omega Journal of Linguistics and Language Studies
ISSN: 3072-4678
Volume 01, Issue 03, June, 2026
Website: https://omegajournals.org

Moreover, a disconnected and reconnected device is not necessarily treated as the same device that was previously attached. This behavior may change in the future as more devices support serial numbers (see device descriptor information) or as more advanced device identification mechanisms are developed.

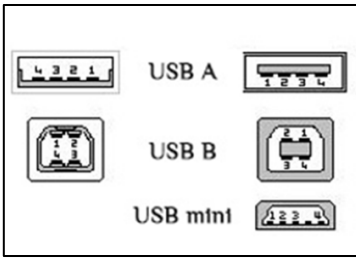
Device removal is reported through an interrupt packet sent by the hub to the driver. The status change information indicates on which port the change occurred. The device removal method is invoked for all devices connected to that port, and the associated data structures are cleaned up. If the port status indicates that a device is currently connected, the device detection and attachment process begins. Restarting a device causes a disconnect-and-reconnect sequence on the hub and proceeds as described above.

During the driver recognition process for a newly connected device, drivers designed for specific devices are checked first. If none are found, the recognition code iterates through all supported configurations until a driver successfully attaches to one of them. For devices supported by multiple drivers, the recognition process examines all interfaces within the configuration that are not yet occupied by a driver.

Configurations that exceed the power limits provided by the hub are ignored. During attachment, the driver should bring the device into its normal operating state but should not reset it, since resetting would disconnect the device from the bus and restart the recognition process.

To avoid excessive data transfers during attachment, communication channels do not need to be allocated immediately. Instead, channel allocation should be postponed until a file is opened and actual data transfer begins. When the file is closed, the channel should also be closed, even if the device remains connected [4].

Let us now briefly consider the hardware aspect of the USB interface. There are two primary types of connectors: Type A and Type B (Figure 1).



Pin	Signal	Color	Description
1	VCC		+5
2	D-		Data+
3	D+		Data-
4	GND		Ground

Figure 1. USB standard Type A, Type B, mini ports, and pin configurations.

Omega Journal of Linguistics and Language Studies
ISSN: 3072-4678
Volume 01, Issue 03, June, 2026
Website: https://omegajournals.org

The Type A connector always faces the host and is commonly found on personal computers. Type B connectors are intended for USB peripheral devices. A USB cable consists of four wires with different colors. The red wire carries power (+5 V), the black wire is ground, and the white and green wires are used for data transmission.

Although this color coding is the most common, some manufacturers may use different color schemes. There are even USB cables in which the red and black wires are reversed, meaning that the red wire is connected to ground. Therefore, if working with the internal wiring of a cable, caution should be exercised and wire colors should not be trusted blindly. It is advisable to verify the connections before use.

Regarding the data transmission method, the USB bus employs the NRZI (Non-Return-to-Zero Inverted) encoding principle. To transmit a logical "1", the voltage level on the D+ line must be raised above +2.8 V while the D− line is lowered below +0.3 V. When transmitting a logical "0", the situation is reversed: (D− > 2.8 V) and (D+ < 0.3 V).

The power supply of USB devices also deserves separate consideration. Several options are available. First, devices may be powered directly from the USB bus, which allows them to be divided into two categories:

Low-power devices

High-power devices

The difference between these categories is that low-power devices may not consume more than 100 mA of current. High-power devices must also remain below 100 mA during the configuration stage; however, after being configured by the host, their current consumption may increase up to 500 mA. In addition, some devices have their own external power source. In such cases, they may draw up to 100 mA from the USB bus while obtaining the remaining required power from their own supply [5].

Having discussed the structure of the USB port, let us now examine the structure of the data transmitted through it. Data exchange between a USB device and a computer (host) is carried out through packets, which together form transactions. A transaction typically consists of three packets: a token packet, a data packet, and a handshake packet. A transaction enables a single exchange of information

Omega Journal of Linguistics and Language Studies
ISSN: 3072-4678
Volume 01, Issue 03, June, 2026
Website: https://omegajournals.org

between the host and an endpoint of the device. Transactions, in turn, are organized into frames.

The beginning of a packet is identified by the SOP (Start of Packet) signal, which represents a transition from the Idle state to the K state. This is followed by the SYNC byte (80h), which, after NRZI encoding, appears as KJKJKJKK. The final two bits of the SYNC field (KK) serve as a marker indicating the beginning of the data block.

The data block consists of fields of varying lengths, and the total length of the data block must be divisible by 8 bits. Bytes are transmitted in least significant bit (LSB) first order, ending with the most significant bit (MSB) (Figure 2).

The packet is terminated by the EOP (End of Packet) signal, which lasts for three bit times and serves as a separator between consecutive packets.

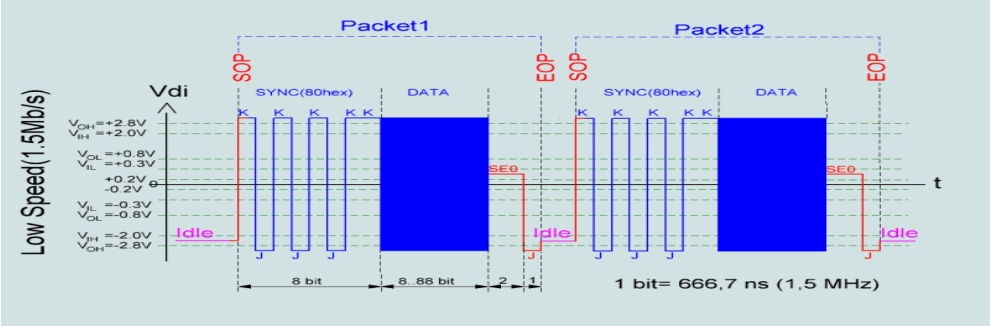


Figure 2. Packet structure of the Low-Speed USB 1.1 (1.5 Mbps) standard.

The beginning of a frame and the frame number are transmitted every 1 ms using a special packet called SOF (Start of Frame). All data is transmitted through frames that are sent at equal time intervals. A frame is a collection of transactions that begins with an SOF packet.

Frames are generated 1,000 times per second (every 1 ms). The SOF packet contains a frame counter. The SOF packet is received simultaneously by all connected devices, helping them synchronize with the host's clock [6].

All data transfer operations from devices must be completed before the start of the next frame. Devices that fail to comply with this requirement are automatically disconnected from the network. Each frame consists of SOF, data transactions, and EOF (End of Frame) transactions, as illustrated in Figure 3.

Omega Journal of Linguistics and Language Studies
ISSN: 3072-4678
Volume 01, Issue 03, June, 2026
Website: https://omegajournals.org

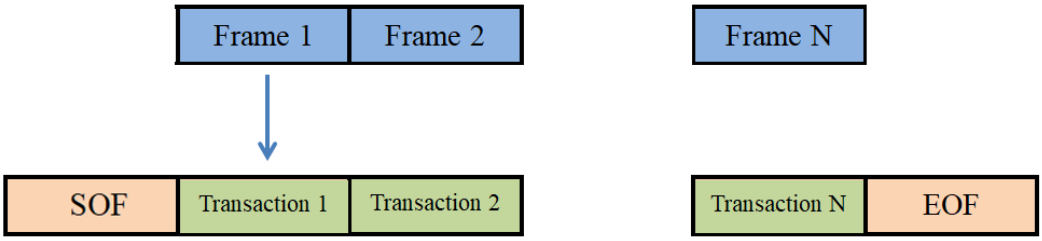


Figure 3. Transactions within a USB Frame

Each frame transaction begins with an SOF (Start of Frame) packet, followed by transactions intended for various endpoints, and concludes with an EOF (End of Frame) marker. More precisely, EOF is not actually a packet in the conventional sense; rather, it is a time interval during which data transmission is prohibited.

Each transaction consists of three main elements: a token, data, and status (handshake), as illustrated in Figure 4.

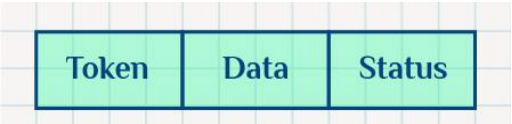


Figure 4. Structure of a USB Transaction

The first packet, known as the Token Packet, contains information about the address of the USB device and the endpoint number intended for the transaction. In addition, this packet includes information indicating the type of transaction being performed. The Data Packet primarily carries the actual data transmitted either by the host or by the endpoint, depending on the transaction type. The final packet, known as the Status (Handshake) Packet, is used to verify whether the transmitted data has been successfully received.

The term “packet” appears frequently in discussions of the USB interface; therefore, it is important to understand its role. Token packets are responsible for organizing the data exchange process and maintaining the communication session (Figure 5). Token packets are divided into three main categories:

IN/OUT

SETUP

Start of Frame (SOF)

An IN packet indicates that the USB device is prepared to send data to the host upon request. An OUT packet indicates that the host is ready and intends to

Omega Journal of Linguistics and Language Studies
ISSN: 3072-4678
Volume 01, Issue 03, June, 2026
Website: https://omegajournals.org

transmit data to the device. A SETUP packet is used for control transfers. The Start of Frame (SOF) packet is used to signal the beginning of a frame [7].

The value of the PID (Packet Identifier) field in a token packet depends on the packet type and may take the following values:

1. IN – Sent by the host to request data from the device. PID = 0001
2. OUT – Sent by the host to transmit data to the device. PID = 1001
3. SETUP – Used for control transfers. PID = 1101
4. SOF (Start of Frame) – Indicates the beginning of a frame. PID = 0101

These values play a crucial role in identifying packet types and managing the data exchange process within the USB interface.

Furthermore, the PID field consists of 4 bits. During transmission, it is supplemented by the bitwise inverse of these 4 bits, resulting in an 8-bit field. This mechanism provides an additional level of error detection and helps ensure the integrity of packet identification during communication.

Figure 5. Structure of a USB Interface Token Packet

Let us now move to the next important components of the Token Packet—the Address and Endpoint fields. These fields store the address of the USB device and the number of the endpoint designated for the transaction. The Address field uniquely identifies a device on the USB bus, while the Endpoint field specifies the particular endpoint involved in the communication.

Finally, the CRC (Cyclic Redundancy Check) field serves as a checksum mechanism. It is used to verify data integrity and detect transmission errors, ensuring that the packet has been received correctly and without corruption.

The next component of a USB transaction is the Data Packet, which carries the actual data being transmitted between the host and the USB device (Figure 6).

Figure 6. Structure of a USB Interface Data Packet

The structure of the Data Packet is generally similar to that of the Token Packet. However, instead of containing the device address and endpoint number, the Data

Omega Journal of Linguistics and Language Studies
ISSN: 3072-4678
Volume 01, Issue 03, June, 2026
Website: https://omegajournals.org

Packet carries the actual payload data being transmitted between the host and the USB device. Like other USB packets, it also includes synchronization and error-checking fields to ensure reliable communication.

Now let us consider the Status Packet. These packets are used to perform synchronization and confirmation functions during the data exchange process. A Status Packet consists of three main components:

SYNC – synchronization field;

PID – packet status identifier;

EOP (End of Packet) – indicates the end of the packet.

The structure of the Status Packet is illustrated in Figure 7.

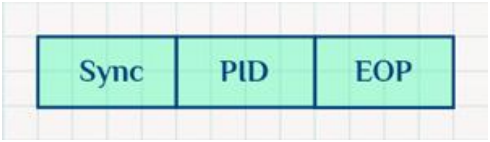


Figure 7. Structure of a Status Packet

In this packet, the PID (Packet Identifier) field can take only two values:

Packet received successfully – PID = 0010

Error occurred during packet reception – PID = 1010

These PID values enable the USB communication system to determine whether a data packet has been received correctly or whether a transmission error has occurred.

The Start of Frame (SOF) packet indicates the beginning of each frame and plays an important role in organizing data transmission on the USB bus [8]. This packet establishes timing intervals and provides the synchronization necessary for managing the transmission of other packets.

SOF packets help regulate data transfers and ensure that communication between the host and connected devices remains synchronized. As a result, they contribute to the overall efficiency and reliability of the USB data exchange process, as illustrated in Figure 8.

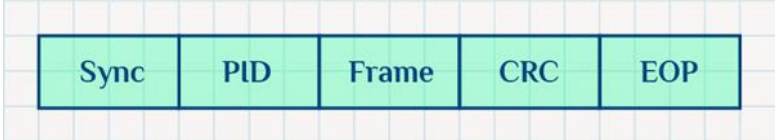


Figure 8. Structure of the SOF Packet

Let us now examine the process of writing data to a USB device, specifically the complete frame structure used during a write operation. As discussed earlier, a

Omega Journal of Linguistics and Language Studies
ISSN: 3072-4678
Volume 01, Issue 03, June, 2026
Website: https://omegajournals.org

frame consists of a sequence of transactions. These transactions are transmitted in a specific order to ensure reliable communication between the host and the USB device.

During a write operation, the host initiates the transfer by sending the appropriate token packets, followed by data packets containing the information to be written. The device then responds with the corresponding status (handshake) packets to indicate whether the data has been received successfully or if an error has occurred.

The sequence of transactions that form a complete frame during a data write operation is illustrated in Figure 9.

Figure 9. Sequence of Transaction Transmission Within a Frame

The transactions within a USB frame are transmitted in a predefined sequence to ensure proper synchronization and reliable data exchange between the host and the device. This sequence determines the order in which control, data, and status information are exchanged during communication.

The SETUP transaction is divided into two parts: a host-to-device transfer phase and a device-to-host transfer phase, as illustrated in Figure 10. These phases are essential for establishing and managing control transfers, allowing the host to configure the device, request information, or issue commands before the actual data transfer takes place.

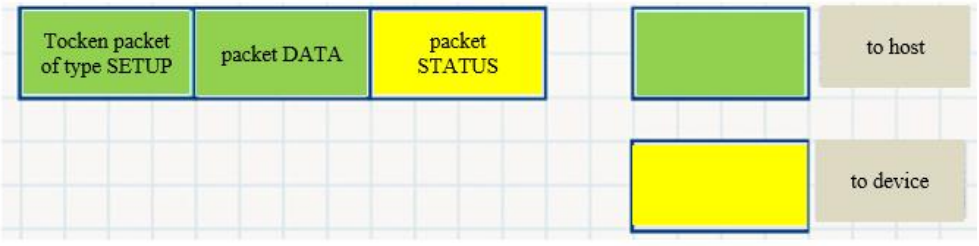


Figure 10. Structure of a SETUP Transaction

The SETUP transaction is used to initiate and control communication between the host and a USB device. It forms the first stage of a control transfer and contains the information required for device configuration, command execution, or data transfer requests. As shown in Figure 10, the SETUP transaction consists

Omega Journal of Linguistics and Language Studies
ISSN: 3072-4678
Volume 01, Issue 03, June, 2026
Website: https://omegajournals.org

of separate phases that support communication from the host to the device and vice versa.

During the process of reading data from a USB device, the frame is organized as illustrated in Figure 11. In this case, communication begins with a SETUP transaction, which establishes the parameters of the requested operation. This is followed by one or more IN transactions, through which the host requests data and the device returns the requested information. The sequence ensures that the data transfer process is properly initialized and synchronized before the actual data is transmitted.

Figure 11. Transaction Transmission Process

The IN transaction is used by the host to receive data from a USB device. It consists of the following stages:

1. Token Packet:This packet is sent from the host to the device and contains the device address as well as the endpoint number designated for the requested data. It indicates that the host is ready to receive data from the device.
2. Data Packet:The device transmits the requested data to the host. This packet contains the information being read from or provided by the device.
3. Status Packet:This packet indicates whether the data has been received successfully or whether an error occurred during transmission. It ensures proper coordination and synchronization between the host and the device.

The IN transaction is a fundamental component of the data retrieval process in the USB interface and is essential for the efficient operation of USB devices. The structure of an IN transaction is illustrated in Figure 12.

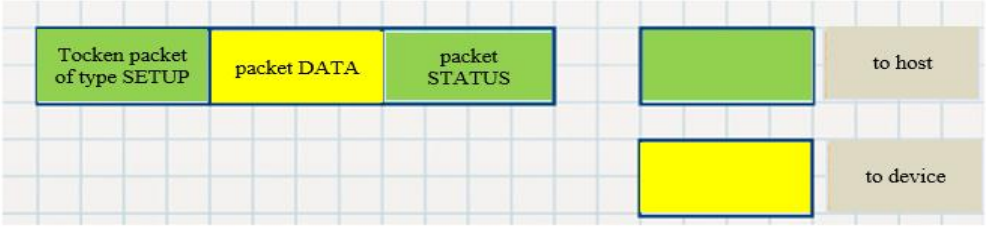


Figure 12. Structure of an IN Transaction

Omega Journal of Linguistics and Language Studies
ISSN: 3072-4678
Volume 01, Issue 03, June, 2026
Website: https://omegajournals.org

In addition, data exchange through the USB interface is carried out through the following stages:

A) Endpoints and Pipes

Endpoint: Each USB device contains its own data endpoints. These endpoints serve as logical addresses through which data is exchanged between the host and the device.

Pipe: Data flow between the host and endpoints is accomplished through communication channels known as pipes. Pipes are divided into two categories:

Control Pipe: Used for device management and the transmission of control information between the host and the device.

Data Pipe: Used for the transfer of application data between the host and the device.

B) Types of Data Transfer

The USB interface supports several data transfer modes. Each mode is designed for a specific purpose and aims to optimize communication efficiency:

Control Transfer: Used for transmitting control commands between the host and the device, such as device identification, configuration, and status information.

Bulk Transfer: Designed for transferring large volumes of data with high reliability and error checking, such as file transfers.

Interrupt Transfer: Used for small amounts of data that require rapid response times, such as signals from a keyboard or mouse.

Isochronous Transfer: Intended for real-time data streams that require continuous transmission, such as audio and video streaming applications.

C) Data Exchange Cycle

Data exchange in the USB interface is performed in a cyclical manner, where the host sends a request and the device responds accordingly. This cycle includes the following processes:

Host Request: The host sends a command or data request to the device.

Device Response: The device returns the requested information or status information to the host.

Data Transfer: If required, data is transmitted either from the host to the device or from the device to the host.

Omega Journal of Linguistics and Language Studies
ISSN: 3072-4678
Volume 01, Issue 03, June, 2026
Website: https://omegajournals.org

Transaction Completion: After successful data transmission, the transaction is terminated.

Conclusion

In conclusion, the process of data exchange through the USB interface involves a considerable level of complexity. In particular, different versions of the USB standard employ different frame and transaction sequences, which can make monitoring and analyzing the data transmission process more challenging. Therefore, when designing mechanisms for controlling devices connected via USB ports, it is essential to consider the characteristics described above, especially for collecting information related to logical device disconnection, connection events, and overall device activity monitoring.

REFERENCES

[1] Универсальная последовательная шина USB // Модернизация и ремонт ПК = Upgrading and Repairing PCs / Скотт Мюллер. — 17-е изд. — М. Вильямс, 2007. — Гл. 15 : Последовательный, параллельный и другие интерфейсы ввода-вывода. — С. 1016—1026.

[2] USB как альтернатива ISA интерфейсу в устройствах ввода-вывода. Колтун Василий, Шевердин Андрей. Компоненты и технологии. №5, 2023.М. Автор, 2023.с. 110-115.

[3] Review key features and advantages of USB. Здищук С. В. / Восточно-Европейский журнал передовых технологий. В.: 2023. С. 25-27.

[4] Axelson, Jan. Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems (Complete Guides series) [Текст] / Jan Axelson. — Видавництво «Lakeview Research», 2021.

[5] McDowell, Steven. USB Explained [Текст] / Steven McDowell, Martin Seyer. — Видавництво «Prentice Hall Ptr», 2023.

[6] Anderson, Don. Universal Serial Bus System Architecture (2-nd Edition) (PC System Architecture Series) [Текст] / Don Anderson. — Видавництво «Addison-Wesley Longman Publishing Co., Inc.», 2022.

[7] Axelson, Jan. USB Complete: Everything You Need to Develop Custom USB Peripherals (Complete Guides series) [Текст] / Jan Axelson. — Видавництво «Penram International Publishing Inc.», 2021.

Omega Journal of Linguistics and Language Studies
ISSN: 3072-4678
Volume 01, Issue 03, June, 2026
Website: https://omegajournals.org

[8] Агуров П. Интерфейс USB. Практика использования и программирования. СПб.: БХВ-Петербург, 2006. - 624 с. (Agurov P. Interface Usb. The practice of using and programming. SPb.: BHV-Petersburg, 2006. -624 p. - P. 146-150.)