JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

ENHANCING EMBEDDED SYSTEM INTEROPERABILITY: A REVIEW OF PROTOCOL CONVERSION TECHNIQUES AND REQUIREMENTS FOR APPLICATION DEVELOPMENTS

Dr.M.Kannan[1], V.Kousik Suriya[2], KV.Akshay[3], A.Chinnamani[4], B. Sandhya[5]

[1] Dean-R&D, United Institute of Technology, Coimbatore

[2,3,4,5] Final Year ECE, United Institute of Technology, Coimbatore

Abstract : Communication protocols in embedded microcontrollers play a crucial role in interfacing and networking with external devices. The most commonly utilized protocols include UART, I2C, SPI, CAN, LIN, as well as analog and digital communication interfaces. However, based on a comprehensive survey, it has been observed that these protocols are not universally integrated into all widely used 8-bit microcontrollers designed for embedded applications. Consequently, during the development of embedded systems, there is often a need for protocol conversion to ensure seamless communication and system functionality.

To address this requirement, hardware and software solutions have been developed using the ATMEGA 328 microcontroller. This system facilitates data exchange by receiving input from one communication protocol and converting it into another. Additionally, a keypad interface and a 2×16 LCD module are integrated to enable users to select the desired protocol conversion. The necessary peripheral devices are interfaced under the specified protocols, and the accuracy of the protocol conversion process has been validated through practical implementation. This conversion system serves as a valuable tool for embedded application development, enabling seamless interoperability between various communication protocols and enhancing system integration efficiency.

IndexTerms - Embedded Systems, Communication Protocols, Microcontrollers, Protocol Converters

1. INTRODUCTION

Embedded systems play a critical role in developing application-specific products using 8-bit, 16-bit, or 32-bit embedded microcontrollers. With the progress of the embedded technology [1-4], a complete embedded platform was used to make a protocol conversion device, which had a processor module, a communication module, and a memory module. To achieve cost-effective product design, many development companies prefer 8-bit microcontrollers. These microcontrollers are manufactured by several industry leaders, including Microchip, Analog Devices, Renesas, NXP Semiconductors, Infineon, Cypress Semiconductor, Silicon Laboratories, among others.

The cost of a microcontroller is influenced by factors such as the availability of integrated peripherals, memory, and communication interfaces. One of the most crucial aspects of a microcontroller is its support for communication protocols, which are essential for interfacing with external devices. Common protocols include Parallel Port, UART, I2C, SPI, LIN, CAN, Ethernet, and USB. Nowadays Arduino boards have been

among the top game-changers in the field of electronics and programming, improving the accessibility of these tools to many users across the world[5]. Many applications were developed based on the Arduino UNO boards and to name a few, the hardware system design for the measuring the Magnetoelastic Resonance were clearly outlined in the journal paper "Development of an Integrated Device Based on the Gain/Phase Detector and Arduino Platform for Measuring Magnetoelastic Resonance[6].

However, not all microcontrollers support every protocol. In many cases, a specific protocol like CAN may be required for application development, but the selected microcontroller might lack support for it. This mismatch can lead to delays in the product development cycle due to the unavailability of required communication interfaces.

To address this challenge, we propose the design of a configurable communication protocol converter. This system can accept data from one communication interface and convert it to another, bridging the gap between incompatible devices. The proposed solution enhances flexibility, accelerates system design and development, and mitigates delays caused by protocol limitations in microcontrollers.

2.OBJECTIVE

i. To conduct a comprehensive survey and comparative analysis of communication protocols employed in various 8-bit embedded
microcontroller architectures.
ii. To design and implement a robust hardware interface capable of converting and facilitating communication between different
protocol standards.
iii. To validate the functionality and reliability of the developed hardware through systematic testing and experimental
verification.
iv. To analyze the results, assess the performance, and outline potential avenues for future enhancements and scalability.

3. OVERVIEW OF COMMUNICATIONPROTOCOL SURVEY

Embedded microcontrollers frequently require communication with external devices, peripherals, or systems. This is achieved through a variety of communication protocol interfaces, which are hardware and software standards that govern data exchange. The following provides an overview of the most commonly used interfaces. A survey is taken to review the various standard communication protocols available in the most commonly used 8-bit Embedded Microcontroller.

Table-1: A survey of Standard Communication Protocols

Microcontroller Family/Mode	UART	SPI	I2C	1-Wire	CAN	USB	LIN	USART
Microchip PIC16F877A	✓	✓	✓	✗	✗	✗	✗	✓
Microchip PIC18F4550	✓	✓	✓	✗	✗	✓	✗	✓
Microchip ATmega328P	✓	✓	✓	✗	✗	✗	✗	✓
Microchip ATtiny85	✓	✓	✓	✗	✗	✗	✗	✓
Renesas 78K0/Kx2	✓	✓	✓	✗	✗	✗	✗	✓
Renesas RL78/G13	✓	✓	✓	✗	✗	✗	✗	✓
STMicroelectronics STM8S003	✓	✓	✓	✗	✗	✗	✗	✓
STMicroelectronics STM8AF6266	✓	✓	✓	✗	✓	✗	✓	✓
NXP 80C51	✓	✓	✓	✗	✗	✗	✗	✓
Silicon Labs EFM8BB10	✓	✓	✓	✗	✗	✗	✗	✓
Zilog Z8F082A	✓	✓	✓	✗	✗	✗	✗	✓
Intel 8051	✓	✓	✓	✗	✗	✗	✗	✓
Maxim MAXQ610	✓	✓	✓	✓	✗	✗	✗	✓
Cypress CY8C21x34	✓	✓	✓	✗	✗	✗	✗	✓
Holtek HT8F2020	✓	✓	✓	✗	✗	✗	✗	✓
Toshiba µPD78F0513	✓	✓	✓	✗	✗	✗	✗	✓
Freescale 68HC05B6	✓	✓	✓	✗	✗	✗	✗	✓
Rabbit R2000	✓	✓	✓	✗	✗	✗	✗	✓
National Semiconductor COP8780	✓	✓	✓	✗	✗	✗	✗	✓
Atmel AT89C51	✓	✓	✓	✗	✗	✗	✗	✓
Fairchild F3850	✓	✗	✗	✗	✗	✗	✗	✗

This survey has significantly helped us understand the availability of various protocols and has guided the development of the proposed system design to support data conversion from any communication protocol to the required one.

4. METHODOLOGY

Figure-1 illustrates the complete flow control process involved in protocol conversion. Various devices interfaced with the system utilize specific communication protocols. For instance, the DS1307 real-time clock employs the I2C protocol for configuration, as well as for reading and writing clock data.

Table-2: Devices and Protocol Used

Devices	Protocol Used	Devices	Protocol Used
DS1307	I2C	ENC28J60	SPI (Input) LAN(Output)
2x16 LCD	Parallel	SD Card	SPI
Keypad	Parallel	MCP2515	CAN
CP2102	UART(Input) USB(Output)	Internal PWM	Average Analog Output
10 k POT	Analog Input		

The data acquired via I2C is converted to UART format when the processor needs to transmit it to a computer through the RS-232 interface using a COM port. This necessitates protocol conversion from the I2C bus to UART. Similarly, all other connected devices exchange address, data, and control signals using their respective protocol interfaces. **Table 2** presents the various devices interfaced with the embedded microcontroller, along with their corresponding communication protocols.

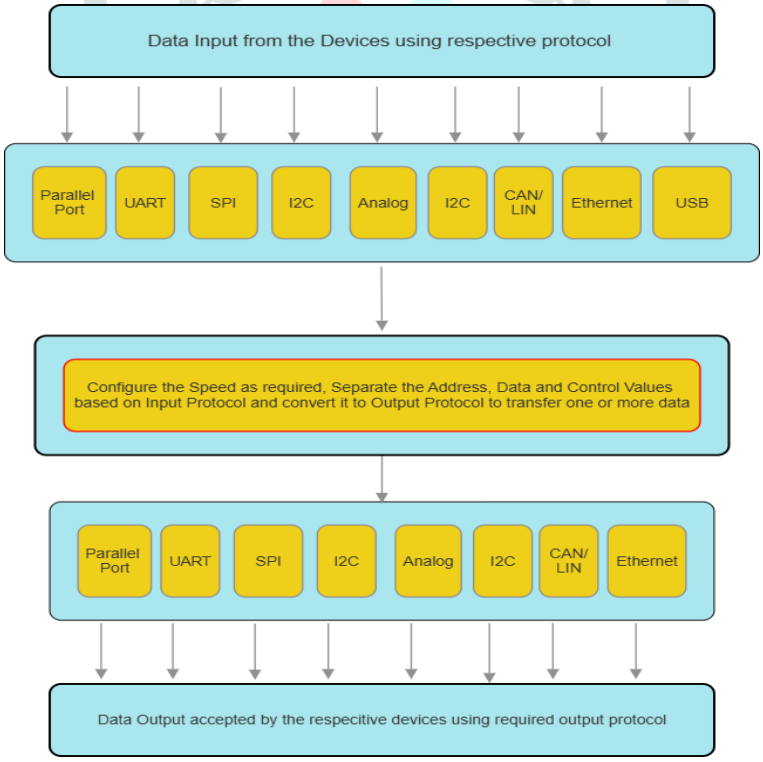

Figure-1: Complete process flow of protocol conversion

The communication protocol in Embedded Microcontrollers plays a vital role in the interface and networking of external devices. The commonly available protocols in Embedded Microcontrollers are UART, I2C, SPI, CAN, LIN, Analog, Digital and so on. Based on the survey, all these communication protocols are not integrated in all the most commonly used 8 bit microcontrollers used to build embedded applications. So, while building embedded applications, requirements may arise to convert from one communication protocol to another one to complete the system developments. The hardware and software are developed based on ATMEGA 328 Microcontroller, which all accepts data from one protocol and will convert it to another one. Necessary keypad interface and 2x16 LCD devices are interfaced to the Microcontroller to select the Input and output protocol conversion requirement. Necessary devices under specific protocol are interfaced and

the required protocol conversion data is verified practically. This device will be greatly considered as a tool for conversion for building embedded applications.

The Figure-2 shows the block diagram of the proposed system design for multi-protocol converter. It is designed and developed based on ATMEGA328 Microcontroller used in the Arduino UNO Board. It Integrates required memory, peripherals and interfaces to complete the system design. It is operated with 5V DC and with a speed of 20 MHz the processor integrates 32 kB of flash program memory, 2 kB of SRAM data memory and 1 kB Non volatile EEPROM. The configuration data are stored in EEPROM. The Flash and SRAM are used to store the program and data respectively.

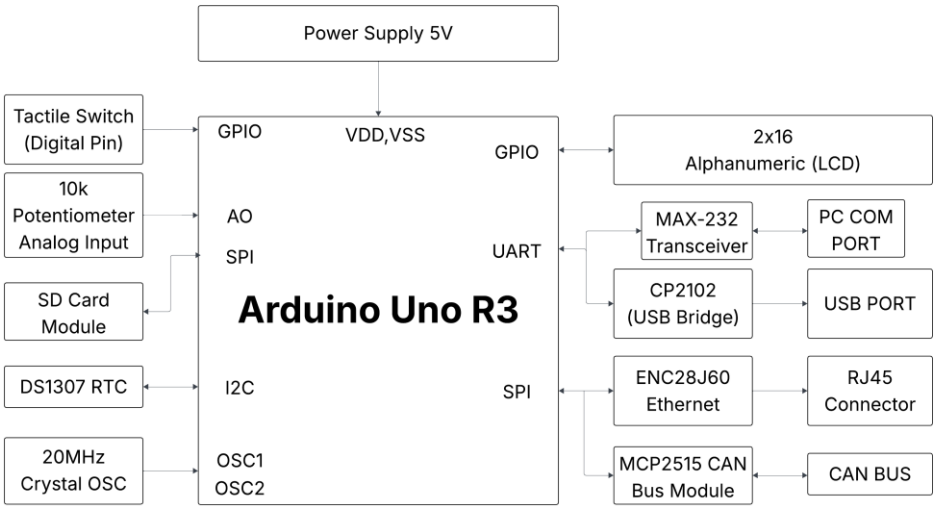

Fig 2: Block Diagram of the Communication Protocol Converter

Four tactile switches are connected to the microcontroller's parallel port to provide parallel data input. Additionally, a 2x16 alphanumeric display is interfaced in parallel with the processor. This display is used to indicate the selected input and output protocols, as well as to present the parallel output data

The ATmega328 microcontroller features a 10-bit Analog-to-Digital Converter (ADC) with a resolution denoted as 'R'. It converts analog input signals, such as those provided by a 10 kΩ potentiometer connected to the A0 pin, into corresponding digital values. The conversion follows the relation:

Digital Parallel Output = Vin / (2^R - 1)

The resulting digital output from the ADC can then be transmitted via various communication protocols, including parallel, I²C, SPI, and others.

The DS1307 is a real-time clock and calendar IC from Analog Devices, interfaced with the processor via the I2C bus. This communication bus requires only two lines: one for the clock (SCL) and the other for data (SDA). Each I²C device has a unique 7-bit address; the DS1307's address is 0x68. The processor uses this address to initiate communication and perform data transfers. Multiple compatible peripherals can be connected to a single I2C bus using their respective 7-bit addresses. The microcontroller retrieves time and date information from the DS1307, which can then be transmitted to other devices using various communication protocols. Additionally, the DS1307 can receive data via the I²C bus for non-volatile storage. The I2C bus supports only half-duplex data communication and can transmit or receive at a time[7].

The ATmega328 microcontroller includes a hardware serial port, designated as UART (Universal Asynchronous Receiver/Transmitter), which uses the Tx and Rx pins for serial data transmission and reception, respectively. A keypad and display connected to the microcontroller are used to configure the baud rate and standard serial communication settings. This UART interface enables data transmission to and from devices via Tx/Rx lines, USB, or a PC's COM port.

The UART pins of the microcontroller are also connected in parallel to a CP2102 USB-to-UART bridge. This bridge enables the conversion of UART signals to USB and vice versa. It requires minimal external components and integrates a full-speed USB 2.0 function controller, USB transceiver, oscillator, EEPROM, and an asynchronous serial data bus with full modem control signal support [8]. This bus facilitates data transmission and reception using the USB protocol, with the device connected via a USB port. The USB

communication protocol is implemented such that the microcontroller is configured as a device, while the system it connects to through the USB port functions as the host.

The Serial Peripheral Interface (SPI) supports full-duplex communication. The design and implementation of the SPI bus are clearly detailed in the paper "Design and Implementation of SPI Bus Communication in Embedded Systems" by Zhang, Y., and Wang, X. [9]. SPI is also referred to as a three-wire bus, utilizing the SCK (Serial Clock), MISO (Master in Slave Out), and MOSI (Master Out Slave In) pins. Multiple SPI-compatible devices can be interfaced with the microcontroller via the SPI bus using separate chip select (CS) pins. In this project, both the SD card and Ethernet controller are connected to the microcontroller through a single SPI bus. Only one peripheral can be selected at a time for data communication. The microcontroller receives data from various protocols and converts it into an SPI-compatible format. The speed of the SPI bus is configurable and can be adjusted through the configuration of special function registers and clock settings

The ENC28J60 is a standalone Ethernet controller equipped with an SPI interface, enabling network connectivity for microcontroller-based systems. It integrates a Media Access Control (MAC) module and a 10BASE-T Physical Layer (PHY), supports both full- and half-duplex communication modes, and includes advanced features such as automatic retransmission on collision and Cyclic Redundancy Check (CRC) generation. Reddy, L. S. N., & Vedanth, B. T. clearly outlined the Ethernet enabled digital I/O control in Embedded Systems [10]. The microcontroller will accept data from devices utilizing any protocol in to ethernet data frame through the SPI bus.

5. IMPLEMENTATION RESULT

The Arduino UNO board is interfaced with various external devices and peripherals using different communication protocols. Figure 3 shows the hardware circuit used for the protocol conversion application. The PIC development board from NSK Electronics was chosen for this project because it includes various interfaces such as the DS1307 RTC, LCD connectors, MAX232 interface, 10k potentiometer, keypads, LEDs, and more, which simplify system design and testing.
External components such as the SD card and Ethernet controllers, which use the SPI interface, are connected to the Arduino UNO board.

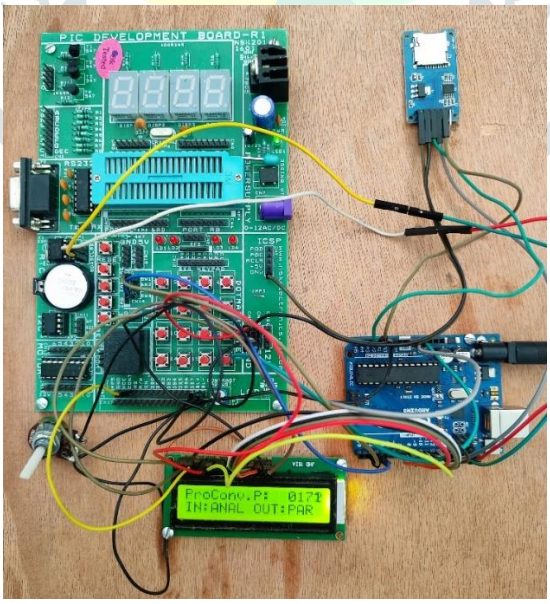

Fig.3 Hardware for Testing the Proposed System

A keypad interfaced with the microcontroller is used to select the desired input and output communication protocols. When the system is reset or powered on, all integrated and external peripherals are initialized with default configuration values. Provisions are provided to allow changes to these default settings. The 2x16 LCD, also interfaced with the microcontroller, displays the selected input and output communication protocols for conversion. It is also capable of displaying parallel data in 8-bit format.

For example, if I2C is selected as the input protocol and USB as the output protocol, the microcontroller reads data from the DS1307 real-time clock and transfers the data to the PCB via a USB bridge. Similarly, conversions between various communication protocols are tested using all connected devices to verify complete system functionality. The MCP2515 is used for transmitting and receiving data over the CAN bus and shares the SPI interface. Chip select signals are managed appropriately to enable communication with the required peripheral devices sharing the common SPI bus.

The 10k potentiometer interfaced to the analog pins of the microcontroller will convert the voltage across it in to digital value using integrated analog to digital converter which is of 10 -bit. This data considered to be parallel is displayed across 2x16 LCD. Also, the Microcontroller accepts data in parallel form and will convert it average analog value using integrated PWM controller in ATMEG328 microcontroller.

6. CONCLUSION

The complete hardware for communication protocol conversion is implemented using available Arduino UNO boards, a PIC development board, and various external modules—each serving specific functions and utilizing different communication protocols. For a more compact and reliable solution, it is recommended to design a custom schematic based on the project requirements, develop a dedicated PCB, and mount the necessary components to build an integrated system. This approach consolidates the entire setup onto a single board, improving both portability and performance.

Currently, the protocol conversion system relies solely on wired interfaces. To enhance flexibility and modernize the design, it is advisable to incorporate wireless communication protocols such as Wi-Fi, Zigbee, or Bluetooth. This would enable the system to receive and convert data wirelessly between different communication standards, as needed.

Furthermore, it is recommended to design the complete set of analog and digital devices, peripherals, -and interfaces using a PSoC (Programmable System-on-Chip). PSoC offers the advantage of hardware adaptability, allowing designers to modify and optimize system functions dynamically as project requirements evolve.

7. REFERENCES

1. Degada A, Savani V. Design and implementation of low cost, portable telemedicine system: An embedded technology and ICT
approach[C]Nirma University International Conference on Engineering. 2015.
2. Ruzaij M F, Neubert S, Stoll N, et al. Multi-sensor robotic-wheelchair controller for handicap and quadriplegia patients using
embedded technologies[C] International Conference on Human System Interactions. 2016.
3. Angulo I, García-Zubia J, Rodríguez-Gil L, et al. A new approach to conduct remote experimentation over embedded
technologies[C]// International Conference on Remote Engineering and Virtual Instrumentation. IEEE, 2016.
4. Ueki M, Akeuchi, K.T, Yamamoto T, et al. Low-Power embedded ReRAM technology for IoT applications[C]// Vlsi Circuits.
IEEE, 2 015:T108-109
5. Ibraheem Redhwi1 and Ahmad Fallatah, “The Versatile World of Arduino Boards”, A Comprehensive Review, Journal Material Science, Vol 9 no. 1, November 2024, 1-2
6. W. R. F. Silva, R. O. R. R. Cunha, and J. B. S. Mendes, "Development of an Integrated Device Based on the Gain/Phase Detector and Arduino Platform for Measuring Magnetoelastic Resonance," arXiv preprint arXiv:2406.19075, Jun. 2024.
7. Hoover, K., Impagliazzo, R., Mihajlin, I., & Smal, A. V. (2018). Half-Duplex Communication Complexity. In International Symposium on Algorithms and Computation (ISAAC 2018).
8. https://www.silabs.com/documents/public/data-sheets/CP2102-9.pdf
9. Zhang, Y., & Wang, X. (2021). Design and implementation of SPI bus communication in embedded systems. Journal of Embedded Systems and Applications, 15(3), 123–130.
10. Reddy, L. S. N., & Vedanth, B. T. (2013). Ethernet Enabled Digital I/O Control in Embedded Systems. IOSR Journal of Electronics and Communication Engineering, 7(6), 47–50.