International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

The Implementation of Multiple 89c51 Microcontrollers through RS232C Serial Communication Interface

Teleron, Jerry I.[1]
[1]Department of Computer Engineering, Surigao State College of Technology, Philippines
[1]Department of Graduate Studies, Surigao State College of Technology, Philippines
Email address: [1]jteleron@ssct.edu.ph

Abstract**—** *The paper presented the implementation of the RS232C serial communication interface to the intel 80c51 family microcontroller with circuit isolation of serial ports to avoid interferences during data transmission. It also presented the multiple serial interfaces or distributed system interfaces with simple serial communication programming codes and circuit diagrams. Also, a Visual Basic.net codes for the receiver/transmitter of the Windows Operating in the host computer/server. Finally, all the circuit diagrams presented were tested and functional, including program codes.*

Keywords**—** *89c51 Microcontrollers; Distributed Systems; Embedded Systems; RS232C Serial Communication; UART.*

I. INTRODUCTION

The Intel 8051 is a popular general-purpose microcontroller widely used for small-scale embedded systems. Many vendors such as Atmel, Philips, and Texas Instruments produce MCS-51 family microcontroller chips [1][3][11]. Some embedded application configures the 8051 UART (Universal Asynchronous Receiver Transmitter) port as read and writes to the serial port for software and hardware handshaking [1][10]

In our days, we can easily find embedded systems everywhere in our daily lives. Embedded Systems are quickly developing, particularly in remote and web applications. The Embedded Systems market is one of the quickest growing regions. By name, and installed embedded system device is a unique reason processing gadget intended to fill committed application [2][11][16][17]

An embedded system comprises its equipment and programming. The kit incorporates a microchip or microcontroller with extra connected outside memory, I/O, and different parts like sensors, keypad, LEDs, LCDs, and actuators. The embedded programs or code is the brain of an embedded system [3][11][16][17].

Increasingly more microcontrollers are embedded in an enormous area of items from modern to homegrown sites. A genuine model is an automobile, an advanced one containing several microcontrollers. As their number expanded, the communication between microcontroller to other microcontroller connectivity became important [2][3][16][17].

Therefore, the serial communication solution was preferred, and a lot of serial buses and protocols were developed with different optimizing parameters of the communication [3][11][4][12][16][17][18][19].

Serial communication is probably the most established instrument to communicate from one computer to another. Beginning with the IBM PC and viable PCs, practically all PCs are equipped with at least one serial port and one parallel port. As the name infers, a serial port sends and receives information sequentially, one bit at a time. Interestingly, in parallel port sends and receives data in eight bits all at once, utilizing eight separate wires [5][8][12]. One of the most common interfaces for communications specified is RS 232C standard. Though widely accepted, RS232 has limited transmission speed, range, and networking capabilities [6][8][9][16][17][18][19].

Asynchronous serial communication utilize for information exchange between the test system and the peripheral to be tried in the innovative work of automated test systems. Nonetheless, in many circumstances, a test system with a single CPU chip can't fulfill the need for multi-channel asynchronous serial communication [7][8][9][12][16][17]. In the modern automatic testing field, asynchronous serial communications play an essential role in data exchange between the test system and the tested peripheral because of flexibility, facility, and reliability. As a result, this technology is widely used in military and commercial products.[9][12][15][17][18].

To reduce the traffic congestion of serial communication transmission lines. A coding method implements to reduce the transmission energy of a serial communication medium by diminishing the number of transitions on the serial communication wire [10][12][16][17].

This method is essential to the developer because instead of the whole string traveling to the media, only the assigned codes and assigned comparable data of the received codes are in the receiving end if connected to the desktop computer USB port at the receiver end.

Utilizing the USB on PC connections and the serial monitor screen element of the application makes it incredibly simple to use the on-screen serial monitor for troubleshooting and to make fetching application programs with insignificant electronic interfacing [11][12][14][16][17][18].

In this paper, the researcher presented multiple functional microcontrollers connected in serial communication with

Teleron, Jerry I., "The Implementation of Multiple 89c51 Microcontrollers through RS232C Serial Communication Interface," *International Research Journal of Advanced Engineering and Science*, Volume 7, Issue 2, pp. 220-226, 2022.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

asynchronous data transfer, guiding future researchers in developing a distributed system application using multiple microcontrollers connected serially through UART ports. Figure 1 below shows the typical architecture of numerous microcontrollers installed in one system application.

Objective of the Study

The study's main objective is to give the students and practicing hardware developers a reference in data communication, particularly in serial communication using the 80c51 family microcontrollers. Furthermore, it aims explicitly to develop the following:

1. Implement a multiple 89c51 family microcontroller connected to its isolated UART ports and distribute the I/O programmable ports.
2. Creating running a program code to activate the Serial Transmit and Received function of the UART.
3. Making a functional Visual Basic.net code to communicate the microcontroller to the personal computer using Asynchronous serial communication.

.

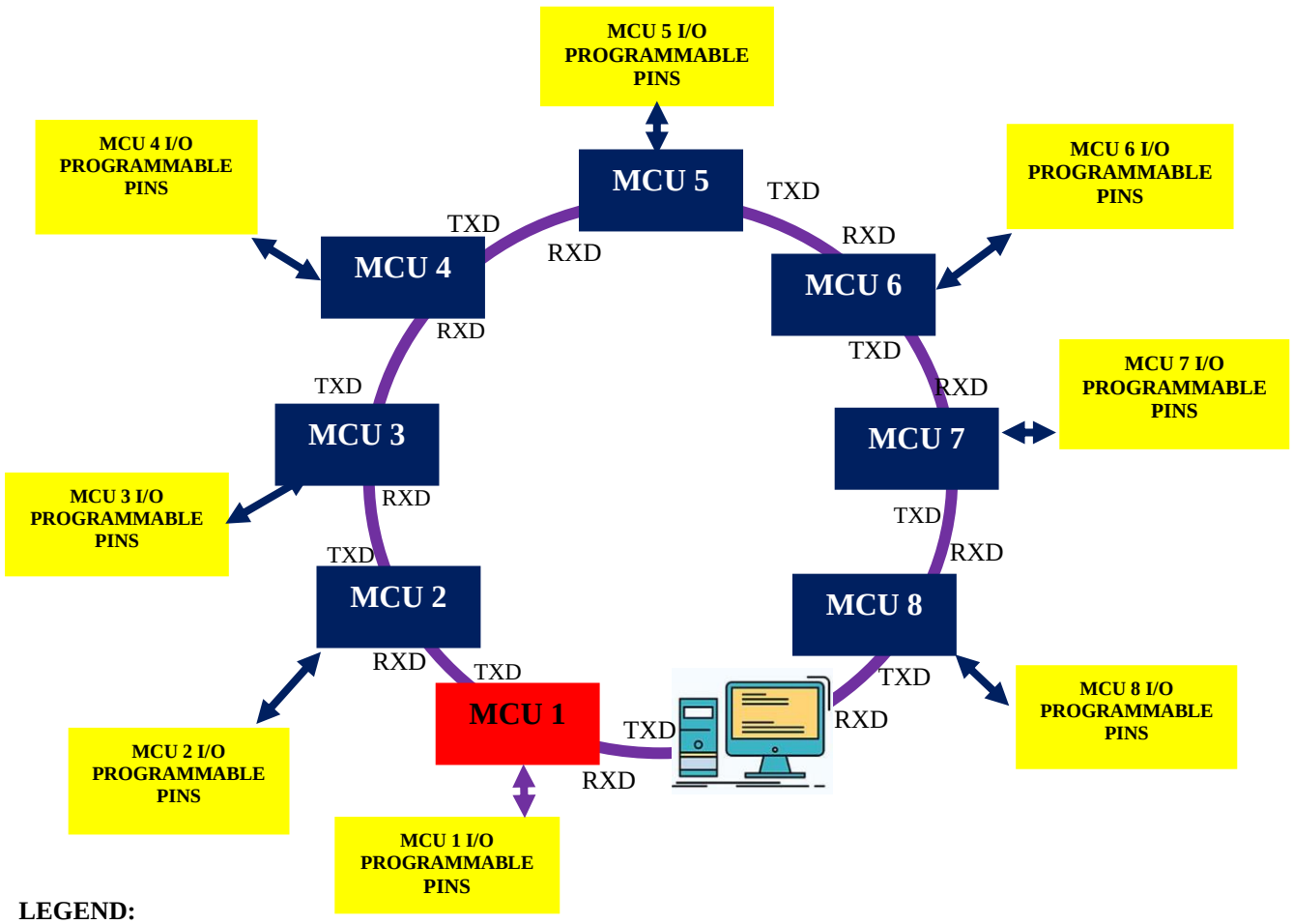

Fig. 1. Multiple Microcontrollers in a System

II. METHODOLOGY

The method used in this study is deductive because it is experimental by designing the circuit diagrams, simulating and implementing through the actual implementation using the specified microcontroller as shown below is the model as a guide in establishing a solution to the specific problem pointed out in this study.

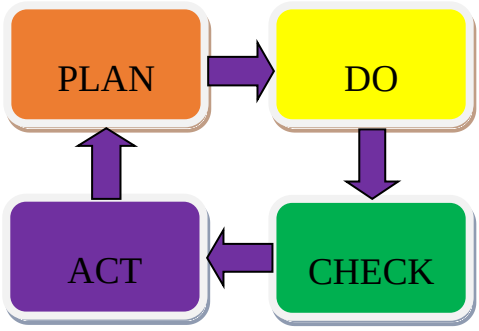

Fig. 2. Schema of the Study

Teleron, Jerry I., "The Implementation of Multiple 89c51 Microcontrollers through RS232C Serial Communication Interface," *International Research Journal of Advanced Engineering and Science*, Volume 7, Issue 2, pp. 220-226, 2022.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Plan

Design

The researcher used AT89C2051/AT89C4051 as a primary controller of the system. This microcontroller is twenty pins (20) with fifteen input/output (I/O) programmable ports that include two (2) universal asynchronous receiver transmitter (UART) pins [16][17][18][19]. This UART provides transmit data (TXD) and received data (RXD); traditionally, when serial communication is implemented, the RS232C is the reference because it is a standard communication protocol for communicating with computers and peripheral devices to enable serial data exchange. Fig 3 below shows the DCE and DTE pinout

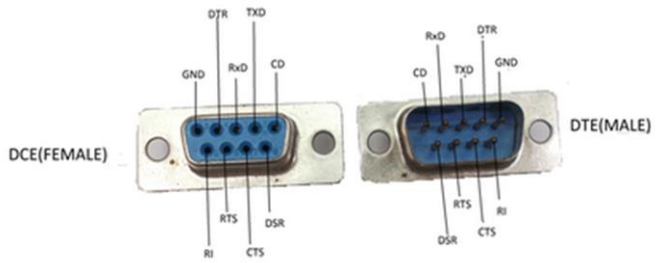

Fig. 3. DB9 Male and Female RS232C Pinout

In most basic asynchronous serial transmission, only three (3) pins are needed to communicate from device A to devise B, which are the TXD, RXD, and GND pins, respectively, for full-duplex transmission mode. In contrast, only two (2) pins are required in a half-duplex, either TXD and GND or RXD and GND or vice versa. These pins are suitable for implementing the microcontroller because the AT89C2051 has a built-in UART port, TXD, and RXD pins. However, during the commission in an industrial application, these UART pins TXD and RXD require an isolation circuit to avoid interferences during data transmission. This time a PC 817 optocoupler linear device is used to prevent the physical connection between one microcontroller to the other connected microcontrollers in the system even though it is connected to one system board to secure the data transmission. Fig 4 below shows the isolated RS232 connection that connects all microcontrollers in a system.

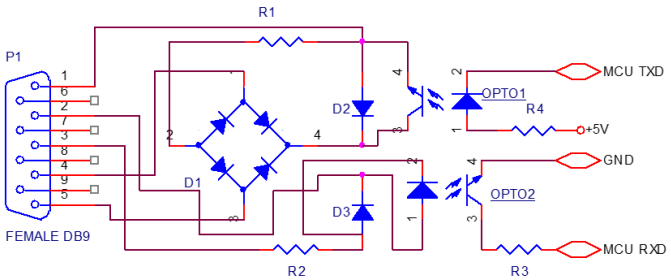

Fig. 4. Isolated RS232C from MCU

Figure 5 below represents the MCU1 or U1 of all MCUs in the system handling Input/Output programmable ports from IO01 to IO13 that can be used as both input and output programmable ports. For example, the eight (8) MCUs can produce one hundred four (IO104) input and output programmable ports, as reflected in figures 5 to 12, respectively.

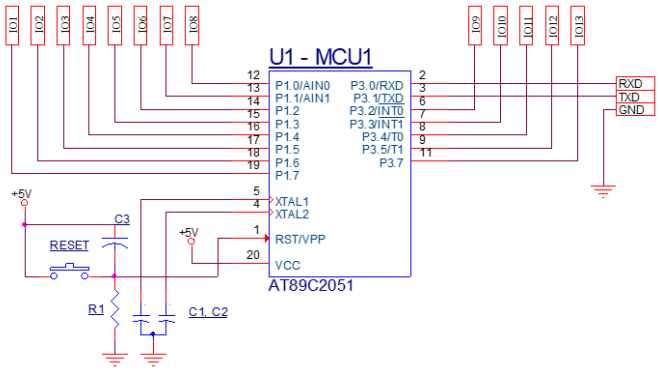

Fig. 5. MCU 1 circuit diagram I/O programmable port 01 to 13

Figure 6 below represents the MCU2 or U2 in the system that handles input and output programmable ports from IO14 to IO26 that can be used as input and output programmable ports.

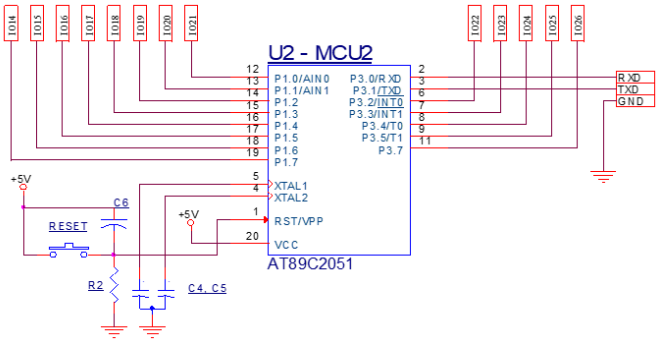

Fig. 6. MCU 2 circuit diagram I/O programmable port 14 to 26

Figure 7 below represents the MCU 3 or U3 of the systems that handle input and output programmable ports from IO27 to IO39 that can be used as input and output programmable ports.

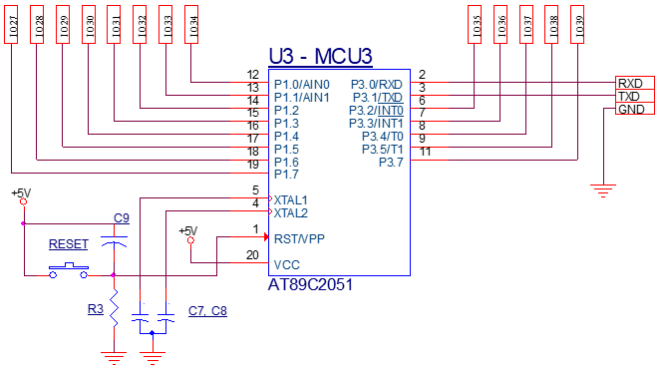

Fig. 7. MCU 3 circuit diagram I/O programmable port 27 to 39

Figure 8 below represents the MCU 4 or U4 of the systems that handle input and output programmable ports from IO40 to IO52 that can be used as input and output programmable ports.

Teleron, Jerry I., "The Implementation of Multiple 89c51 Microcontrollers through RS232C Serial Communication Interface," *International Research Journal of Advanced Engineering and Science*, Volume 7, Issue 2, pp. 220-226, 2022.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

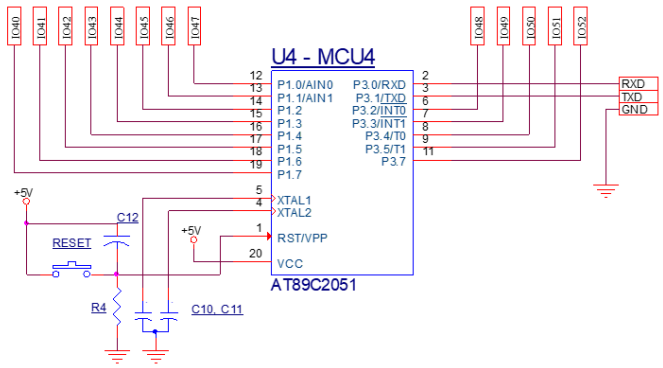

Fig. 8. MCU 4 circuit diagram I/O programmable port 40 to 52

Figure 9 below represents the MCU 5 or U5 of the systems that handle input and output programmable ports from IO53 to IO65 that can be used as input and output programmable ports.

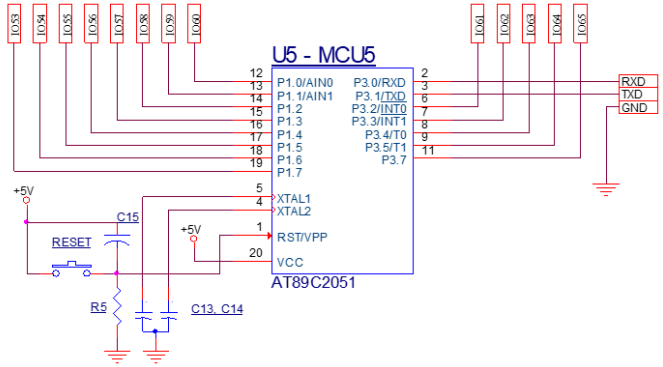

Fig. 9. MCU 4 circuit diagram I/O programmable port 53 to 65

Figure 10 below represents the MCU 6 or U6 of the systems that handle input and output programmable ports from IO53 to IO65 that can be used as input and output programmable ports.

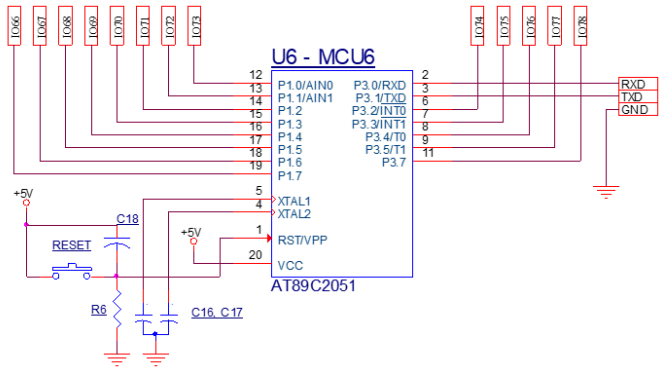

Fig. 10. MCU 6 circuit diagram I/O programmable port 66 to 78

Figure 11 below represents the MCU 7 or U7 of the systems that handle input and output programmable ports from IO79 to IO91 that can be used as input and output programmable ports.

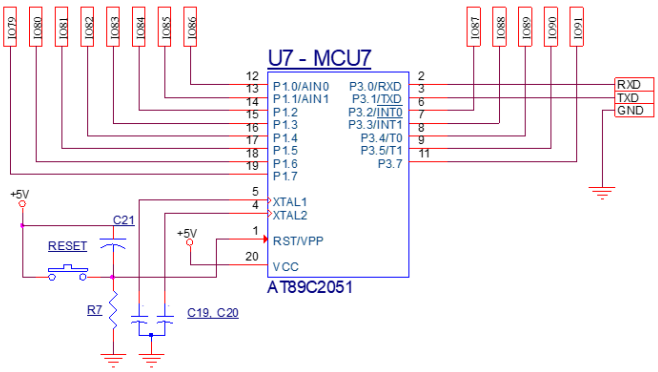

Fig. 11. MCU 7 circuit diagram I/O programmable port 79 to 91

Figure 12 below represents the MCU 7 or U7 of the systems that handle input and output programmable ports from IO92 to IO104, which can be used as input and output programmable ports.

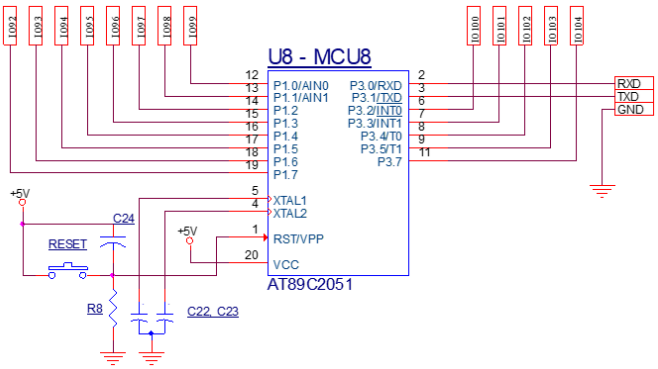

Fig. 12. MCU 8 circuit diagram I/O programmable port 92 to 104

Do

Layout

The researcher used EasyEDA software for the online layout of the PCB of the entire system so that it would be easy to order the PCB board online using the product of the software introduced by the PCB maker. The researcher simulated the circuit in the Proteus software before creating it on the EasyEDA for the system board's integrity because the PCB price is high depending on the actual size. Figure 13 below shows the entire circuit board of the multiple MCU in a system.

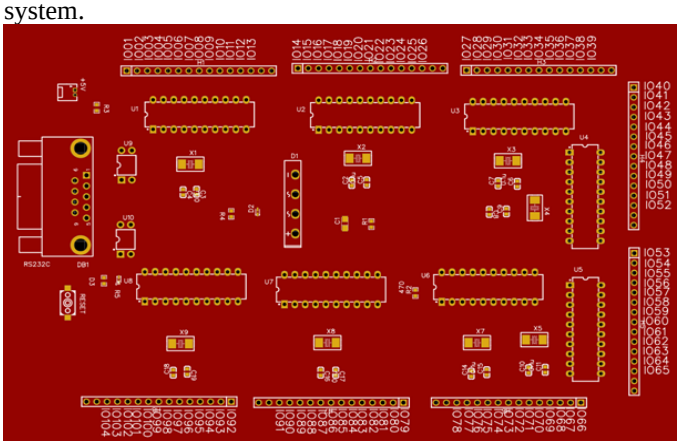

Fig. 13. PCB Layout of the System

Teleron, Jerry I., "The Implementation of Multiple 89c51 Microcontrollers through RS232C Serial Communication Interface," *International Research Journal of Advanced Engineering and Science*, Volume 7, Issue 2, pp. 220-226, 2022.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Check

Assembly of Components

In this study stage, the researcher assembled the parts for the PCB and placement the parts included in the system board. After soldering the system, the researcher starts making an embedded code to test the system's functionality. Figure 14 shows the PCB with a component in the 3D application of EasyEDA software.

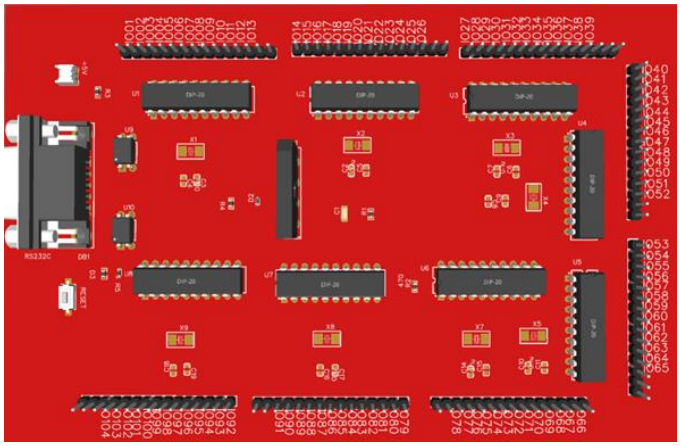

Fig. 14. PCB with Component

System Coding

The researcher makes a program code for the embedded microcontroller. Below is the principle transmitting routine code used in each microcontroller connected to the system.

This routine will transfer the content of register A that the researcher assigned values [16].

```
;;;;;;;;;;;;;TRASMIT DATA ROUTINE;;;;;;;;;;;;;;;;;;;
TRANSMIT:
          MOV   SBUF,A      ;REG A DATA STORE TO SBUF WAIT FOR TI
PROCEED:
          JNB   TI,PROCEED  ;CHECK IF TI READY
          CLR   TI          ;CLEAR TI READY FOR NEXT CHAT TXD
          RET
          END
```

Similarly, another sub-routine code below receives the data from the SBUF register coming from other microcontroller transmitted data [16].

```
;;;;;;;;;;;;RECEIVED ROUTINE;;;;;;;;;;;;;;;;;
RXD_DATA:
          JNB   RI,$        ;RECEIVING DATA READY
          MOV   A,SBUF      ;DATA IN BUFFER
          MOV   P1,A        ;SEND TRANSMITTED DATA TO PORT1
          CLR   RI          ;CLEAR RI READY FOR NEXT SIGNAL
          SJMP  RXD_DATA    ;RECEIVED ANOTHER CHAR
          END
```

Combining the two (2) routines, the transmit and received can accommodate a full-duplex transmission mode. The system uses for sending and receiving the data coming from the connected microcontroller in the system. Each microcontroller has a unique code that will automate the application, but the transmit and received routine are present in all embedded codes of eight microcontrollers connected serially.

Act

Testing

Testing its functionality, the researcher created a Visual Basic .net code for the testing process from the computer to the system developed. This testing program will transmit data from the computer, and the microcontroller that holds the address can receive the transmitted data. Also, a microcontroller connected serially can send the data that only a desktop PC can receive the shared information, as shown in the active form from the .net application in figure 15 below [17].

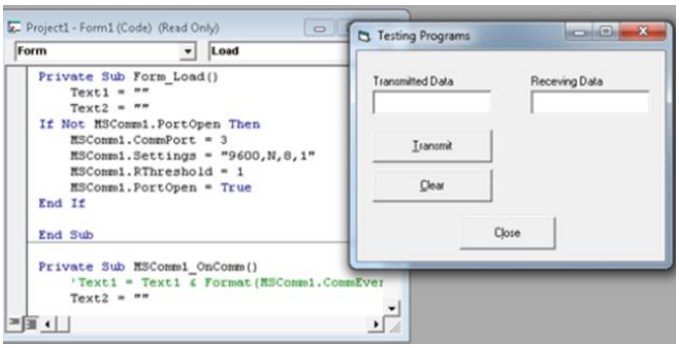

Fig. 15. Testing program form from .net environment

III. Results and Discussion

The developed system created by the researcher is a generic application that requires many inputs and outputs programmable ports application like car parking in a shopping mall wherein the system detects the vacant slot with a light indicator on it. Second, an elevator system also requires multiple sensors and actuators, including the display that connects all the microcontrollers serially in the elevator system. Last is the speech laboratory system, which features forty-five (45) students/cubicles or more significant users that use output ports to enable and disable speakers and an input port for the call button when the students call the teacher's attention, as shown in figure 16 below [17]. This application requires eight microcontrollers to complete the forty-five cubicles for the speech laboratory to control the headsets' enable and disable. They had developed a functional application using eight (8) microcontrollers connected serially, including the desktop PC as the central console of the speech laboratory.

Fig. 16. Application of Desktop PC for Speech Laboratory Console

Teleron, Jerry I., "The Implementation of Multiple 89c51 Microcontrollers through RS232C Serial Communication Interface," *International Research Journal of Advanced Engineering and Science*, Volume 7, Issue 2, pp. 220-226, 2022.

The researcher developed an assembly language program code shown below embedded in the microcontroller chip that transmits and receives information to or from the PC. At the same time, the PC is also equipped with a Visual Basic .net code for the receiver end that accepts the transmitted data from the microcontrollers. The code below may use for the rest of the connected microcontrollers, provided that the receiver may have a code to accept incoming data intended to be received.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; A CODE TO TRANSMIT AND RECEIVED CODES TO PC      ;
; BY: JERRY I. TELERON, Ph.D.,PCpE                 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
          ORG   00H
         NOP
         IE     EQU 0A8H       ;Address of Interrupt
                               ;Enable 0A8H-0AFH
         P1     EQU    090H
          P1_4  EQU    094H
         P1_5   EQU    095H
         P1_6   EQU    096H
         P1_7    EQU   097H

         TMOD   EQU    089H
         TH1    EQU    08DH
         SCON   EQU    098H
         SBUF   EQU    099H
         TR1    EQU    08EH
         RI     EQU    098H
         TI     EQU    099H
         P3_2   EQU    0B2H
          P3_3  EQU    0B3H
         SJMP   START
         ORG 013H            ;Select INT1 interrupt
                             ;switch button
         MOV R0,#255
LOOPS:
          DJNZ R0,LOOPS
          MOV P1,#00H       ;ON STATE FOR P1
          RETI
START:
          MOV P1,#0FFH      ; OFF STATE OF P1
          MOV IE,#10000100B ;Addressable bits setting
                            ;interrupts to configure
                            ;INT1
                            ;see IE: INTERRUPT ENABLE
                            ;REGISTER
                            ;BIT ADDRESSABLE
NEXT:
         MOV    TMOD,#21H   ;TIMER 1 8BIT AUTO RELOAD
         MOV    TH1,#0FDH   ;9,600 BIT/SEC BAUD
         MOV    SCON,#50H   ;1START, 8BIT, 1STOP, REN
                            ;ENABLE
                             ;8-N-1
          SETB TR1

;;;;;;;;;;;;;;RECEIVED ROUTINE;;;;;;;;;;;;;;;;;;;;;
RXD_DATA:
        JNB  RI,$           ;RECEIVING DATA READY
        MOV  A,SBUF         ;DATA IN BUFFER
        MOV  P1,A           ;SEND TRANSMITTED DATA TO
                            ; PORT1
        CLR  RI             ;CLEAR RI
        CALL TX_DATA
        CALL DELAY
        SJMP RXD_DATA       ;RECEIVED ANOTHER CHAR
TX_DATA:
        MOV  DPTR,#WELCOME_NOTE
        MOV     R7,#150
DISPLAY_WELCOME_NOTE:
        CLR     A
        MOVC    A,@A+DPTR
        MOV     SBUF,A
        JNB     TI,$
        CLR     TI
        INC     DPTR
        DJNZ    R7,DISPLAY_WELCOME_NOTE
        RET

;;;;;;;;;;;;;DELAY MATTERS;;;;;;;;;;;;;;;;
DELAY:
   MOV R4,#180            ; set middle loop with 180
 DLY2:
   MOV R3,#255            ; set inner loop with 255
 DLY1:
   DJNZ R3,DLY1           ; loop all values of delay
   DJNZ R4,DLY2
   RET
;;;;;;;;;;;;;;;;LOOK UP TABLES;;;;;;;;;;;;;;;;;;;;
WELCOME_NOTE: DB"COMMUNICATING WITH THE PC ",0AH,0DH
RECEIVING_NOTE:DB"PC RECEIVED CODE FROM MCU",0AH,0DH
END
```

The researcher developed a Visual Basic.net code to transmit and received data using the serial communication port using the Try...Catch...Finally block statement. Allows the application to close the serial port even if it generates an exception. All code that manipulates the serial port should appear within this block. As shown in the code below [26].

```
Dim returnStr As String = ""
Dim com1 As IO.Ports.SerialPort = Nothing
Try
    com1 = My.Computer.Ports.OpenSerialPort("COM1")
    com1.ReadTimeout = 10000
Catch ex As TimeoutException
    returnStr = "Error: Serial Port read timed out."
Finally
    If com1 IsNot Nothing Then com1.Close()
End Try
```

The code below is the function to receive serial data in the serial port COM1 only [26].

```
Function ReceiveSerialData() As String
    ' Receive strings from a serial port.
    Dim returnStr As String = ""

    Dim com1 As IO.Ports.SerialPort = Nothing
    Try
        com1 =
My.Computer.Ports.OpenSerialPort("COM1")
        com1.ReadTimeout = 10000
        Do
            Dim Incoming As String = com1.ReadLine()
            If Incoming Is Nothing Then
                Exit Do
            Else
                returnStr &= Incoming & vbCrLf
            End If
        Loop
    Catch ex As TimeoutException
        returnStr = "Error: Serial Port read timed
out."
    Finally
        If com1 IsNot Nothing Then com1.Close()
    End Try

    Return returnStr
End Function
```

To complete the code for serial, send and receive another code below for the transmit string routine. Again, the statement of the WriteLine method was used that sends the data to the serial port COM1 [26].

```
Sub SendSerialData(ByVal data As String)
    ' Send strings to a serial port.
    Using com1 As IO.Ports.SerialPort =
            My.Computer.Ports.OpenSerialPort("COM1")
        com1.WriteLine(data)
    End Using
End Sub
```

IV. CONCLUSION

Therefore, the author concluded that the study is essential to the aspiring computer engineer or hardware developer as their basis for implementing multiple microcontrollers in a system, particularly in the implementation that requires many inputs and output programmable ports to complete the design.

Specifically, it answers the objectives pointed out in this study:

1. Creating multiple microcontrollers in one (1) system application.
2. Making a code for the embedded application.
3. Making an application of Visual Basic .net for the desktop computer application that communicates between the developed system and the desktop PC.

The researcher answered all the queries by implementing the system with a hardware diagram of the microcontrollers connected serially through its UART ports.

Recommendation

The researchers suggest that an Internet of Things (IoT) may include the implementation of the multiple microcontrollers in a system for this study's future improvement. They were allowing future researchers and aspiring engineers to go on and implement the wireless fidelity application using its application without using the available module for WIFI applications.

ACKNOWLEDGEMENTS

The author would like to thank behind for making this research successful the author's family for unwavering support. And finally, our more powerful Lord Jesus Christ, who is on a high, always protects and gives good judgment when making this study successful.

REFERENCES

[1] Qian K., den Haring D., Cao L. (2009) Serial Communications. In: Embedded Software Development with C. Springer, Boston, MA. Retrieved from: https://bit.ly/3EOH945

[2] Y. Hendriana and R. Hardi, "Remote control system as serial communications mobile using a microcontroller," 2016 International

[3] Qian K., den Haring D., Cao L. (2009) Introduction to Embedded Systems. In: Embedded Software Development with C. Springer, Boston, MA. Retrieved from: https://bit.ly/36TXacu

[4] M. Popa, A. S. Popa, V. Cretu and M. Micea, "Monitoring Serial Communications in Microcontroller Based Embedded Systems," 2006 International Conference on Computer Engineering and Systems, 2006, pp. 56-61, doi: 10.1109/ICCES.2006.320425. Retrieve from: https://bit.ly/3rTh66k

[5] Serial Communications. In: Practical .NET 2.0 Networking Projects (2007). Apress. Retrieved from: https://bit.ly/3rQSfAc

[6] H. Al-amily, "RS-232c Serial Communications How to Use each Type of Connections," 2018.

[7] Piyush Saxena, K. K. Sharangpani, H. S. Vora, S. V. Nakhe, R. Jain, N. M. Shenoy, R. Bhatnagar, N. D. Shirke, "Plastic optical fiber serial communications link for distributed control system," Proc. SPIE 4417, Photonics 2000: International Conference on Fiber Optics and Photonics, (25 September 2001); Retrieved from: https://bit.ly/3OFYgcQ

[8] J. Machacek and J. Drapela, "Control of serial port (RS-232) communication in LabVIEW," 2008 International Conference - Modern

[9] W. Ji, L. Zhaoqing and Z. Yigang, "Design of 4-Channel Asynchronous Serial Communications Interface," 2007 8th International Conference on Electronic Measurement and Instruments, 2007, pp. 4-891-4-894, doi: 10.1109/ICEMI.2007.4351287.

[10] Kang Min Lee, Se-Joong Lee and Hoi-Jun Yoo, "Low energy transmission coding for on-chip serial communications," IEEE International SOC Conference, 2004. Proceedings., 2004, pp. 177-178, doi: 10.1109/SOCC.2004.1362398.

[11] Qian K., den Haring D., Cao L. (2009) 8051 Microcontroller. In: Embedded Software Development with C. Springer, Boston, MA. Retrieved from: https://bit.ly/3F6mxEJ

[12] Louis E. Frenzel Jr. "Handbook of Serial Communications Interfaces: A Comprehensive Compendium of Serial Digital Input/Output Standards" Copyright 2015 Retrieve from: https://bit.ly/3rU7f02

[13] Dukish B. (2018) Serial Communications. In: Coding the Arduino. A press, Berkeley, CA. Retrieved from: https://bit.ly/3MAuPa8

[14] V. Jindal, "PC-to-PC communication via RS-232 serial port using C," *Electron. World*, vol. 112, no. 1837, pp. 25–29, 2006.

[15] R. Steele, G. Little and W. Russell, "A Serial Communications Architecture for Real-Time Digital Control," Tomorrow's Television: 16th Annual SMPTE Television Conference, 1982, pp. 118-127, doi: 10.5594/M00629.

[16] J. I. Teleron, "The Implementation of Distributed System Application on Periodic Attendance of the Faculty," *Int. Res. J. Adv. Eng. Sci*., vol. 7, no. 1, pp. 196–202, 2022.

[17] J. I. Teleron, "Fabrication Of Fuzzy Logic Controller On Speech Laboratory System Using At89c2051 Microcontroller," *Int. J. Adv. Sci. Converg.*, vol. 2, no. 2, pp.15–23, 2020, [Online].Retrieved from: https://bit.ly/3EQ7FtI

[18] TutorialPoint, "What is RS 232C Standard?" TutorialPoint Copyright 2022. All Rights Reserved. Retrieve from: https://bit.ly/3rU7cRU

[19] Components101, "RS232 Connector" Copyright 2021 © Components101. All rights reserved Retrieve from: https://bit.ly/36Riu2h

[20] Technique and Technologies, 2008, pp. 36-40, doi:10.1109/SPCMTT.2008.4897488.

[21] Smythe R.J. (2021) Microcontrollers and Serial Communications. In: Arduino in Science. Apress, Berkeley, CA. Retrieved from: https://bit.ly/3MxYtgb

[22] X. Han and X. Kong, "The Designing of Serial Communication Based on RS232," 2010 First ACIS International Symposium on Cryptography, and Network Security, Data Mining and Knowledge Discovery, E-Commerce and Its Applications, and Embedded Systems, 2010, pp. 382-384, doi: 10.1109/CDEE.2010.80.

[23] X. Liu and Y. Liu, "Multi-functional serial communication interface design based on FPGA," 2017 3rd IEEE International Conference on Computer and Communications (ICCC), 2017, pp. 758-761, doi: 10.1109/CompComm.2017.8322645.

[24] Z. Hongfu, X. Xinyan and T. Yong, "Serial Communication Interface Design Based on Lab VIEW and VC Mix Programming," 2007 8th International Conference on Electronic Measurement and Instruments, 2007, pp. 3-44-3-49, doi: 10.1109/ICEMI.2007.4350950.

[25] H. -l. Peng and Y. -l. Nie, "Design of serial communication interface based on FPGA," 2011 IEEE International Conference on Computer Science and Automation Engineering, 2011, pp. 410-414, doi: 10.1109/CSAE.2011.5952879.

[26] Microsoft 2022, "How to: Receive Strings From Serial Ports in Visual Basic" Retrieved from: https://bit.ly/3F6ClHz