# Data Communication between Non Homogeneous Control Systems

[1]Swapnil Lanjewar, [2]Mrunal Shidore

[1]Student, [2]Assistant Professor
[1]Department of Electronics Engineering,
[1]Vishwakarma Institute of Technology, Pune, India

***Abstract:*** Communication is an important aspect in designing or developing any industrial automation setup. Mostly Programming Logic Controllers (PLCs) are used for the purpose of controlling and monitoring along with sensors specifically designed to work with them. Along with the outset of Industry 4.0 it is getting more important to connect these PLC based controllers to Internet technology. Arduino microcontrollers offer cost effective and reliable method to take the data online. Industrial plants are often geographically widespread. Successful automation of such industries depend on primarily controlling devices as well as powerful communication network. At a field level plc talk with field devices using proprietary protocols. This paper discuss the way to create a communication channel between PLC and Arduino microcontroller.

***IndexTerms - Arduino Uno, RSLogix, Allen Bradley (AB) Micrologix1200 PLC***

## I. INTRODUCTION

PLCs are one of the important integral part of automation systems. Also in small scale process control systems PLCs play a vital role. PLCs communicate with sensors, actuators at field level and with host controller at plant level. Industries apply automation to manufacture high quality products at faster production rate. Communication and networking is the heart of industrial automation. Communication is the data exchange between the controlling devices having common connectivity platform. If these system are non-homogenous we can use gateways to exchange data. For this data exchange the PLCs uses hardware specific protocols like MODBUS, DF-1 and PROFINET with the certain configurations. These hardware specific protocols limit the range and speed of communication. Moreover the hardware supporting wireless capabilities like ESP8266 Wi-Fi Modules, ZigBee Modules, GSM modules and Bluetooth Modules are difficult to interface with PLC.

Due to lack of these wireless capabilities the range of data transmission and reception got reduced and the scope of data was limited to factory premises only. With the advancement of Industry 4.0 it became necessary to incorporate new technologies so as to make things like wireless control possible. But replacing the earlier hardware and setup with the totally new embedded system based setup would be a costly affair. So there comes the need of intermediate device like Arduino Uno to send and receive data. Data from the Arduino can then be wirelessly transferred to the stations where actual control and monitoring by the Superior or person in charge.

Difficulties while communicating between these devices are: logic levels, communication protocol being used and the hardware that they use to transmit and receive data.
In this project we used AB PLC which has indigenous DF-1 protocol to communicate with programming workstation device (PC) which is one of the flavor of Serial Communication. The DF-1 protocol supports RS232 standard which works on -25V to +25V logic levels. On the other hand microcontrollers like Arduino Uno uses TTL/CMOS logic levels of 0 to +5V.
So to implement data exchange between AB PLC and Arduino Uno we used IC MAX232 based logic level shifter module which converts RS232 standard data to TTL/CMOS logic data.

## II. EXPERIMENTAL SETUP

The setup for data exchange project consists of

1. Hardware Setup
   - AB PLC,
   - Arduino Uno,
   - RS232 to TTL/CMOS logic converter based on MAX232 IC,
   - PS2 to RS232 cable
   - Conveyer belt with IR proximity sensor
2. Software Setup
   - RSLinx
   - RSLogix500
   - Arduino IDE.

### A. PLC CONFIGURATION

PLC programing is a crucial part of the overall setup. The PLC is programmed using typical ladder diagram. The critical part being able to transmit and receive data between the PLC and Arduino Uno.

The PLCs communicate with the programming PC with the help of DF-1 protocol which is indigenously developed by the Allen Bradley. DF-1 is byte oriented asynchronous protocol which uses RS232 interface module. This protocol belongs to link and application layer having both half duplex and full duplex modes of communication.

The DF-1 protocol utilizes Computer's Communications port to establish Serial Communication between the PLC and the PC. The configuration of which is described in Table 1.

Table 1. Configuration for Serial Communication done is RSLinx [2].

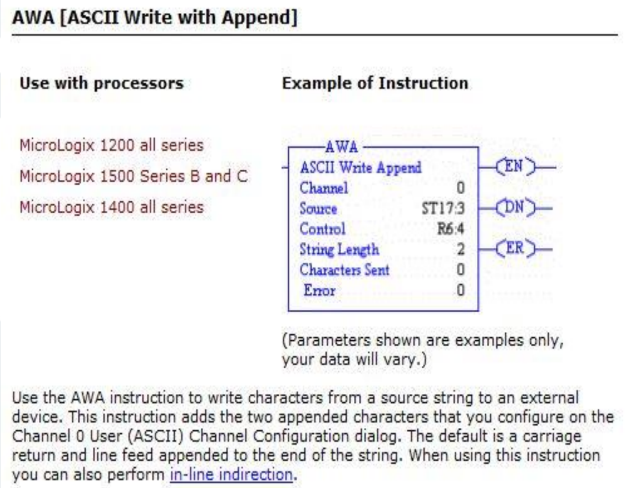
| | |
|---|---|
| Comm. Port | COM5 |
| Baud Rate | 19200 |
| Parity | None |
| Error Checking | CRC |
| Stop Bits | 1 |
| Protocol | Full Duplex |

Communication Port is a virtual port which is device specific. So it may be different for different devices. The Communication Port which is allotted by the PC can be known from the Device Manager.

Modern laptops don't come with DB9/RS232 port so a bridge device like the MAX232 based RS232 to TTL/CMOS convertor connector is used. Special Drivers are needed to be installed so as to make the hardware work with the PC.

After the connection with PLC and PC is done special PLC instruction are to be used for data transfer. The communication is done by sending and receiving ASCII characters.

The instruction for transmitting ASCII characters is described in the Fig 1[1].



AWA [ASCII Write with Append]

Use with processors

MicroLogix 1200 all series
MicroLogix 1500 Series B and C
MicroLogix 1400 all series

Example of Instruction

AWA
ASCII Write Append
Channel 0
Source ST17:3
Control R6:4
String Length 2
Characters Sent 0
Error 0

(Parameters shown are examples only, your data will vary.)

Use the AWA instruction to write characters from a source string to an external device. This instruction adds the two appended characters that you configure on the Channel 0 User (ASCII) Channel Configuration dialog. The default is a carriage return and line feed appended to the end of the string. When using this instruction you can also perform in-line indirection.

Fig 1. AWA instruction for data transmission

Source should have the address of data file (here ST17:3) which must be of string type and number of characters less than 82.

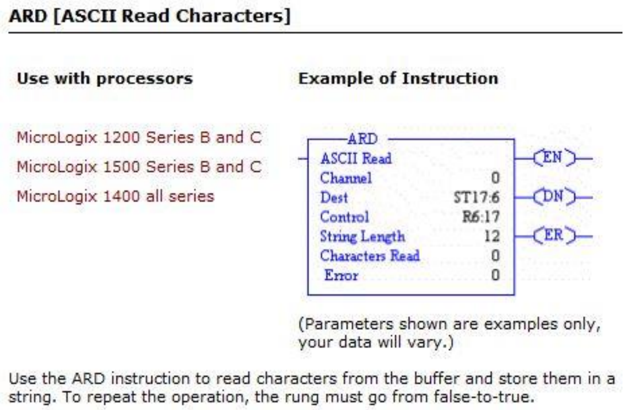
Control is the area that stores the control register required to operate the instruction. This direct address must be an R file type [1].

DN bit is set when the transfer is done with "Character Sent" showing number of characters sent.

Control is the area that stores the control register required to operate the instruction. This direct address must be an R file type [1].

DN bit is set when the transfer is done with "Character Sent" showing number of characters sent.

To receive the data the instruction and how to use it is described in Fig 2[1]



ARD [ASCII Read Characters]

Use with processors

MicroLogix 1200 Series B and C
MicroLogix 1500 Series B and C
MicroLogix 1400 all series

Example of Instruction

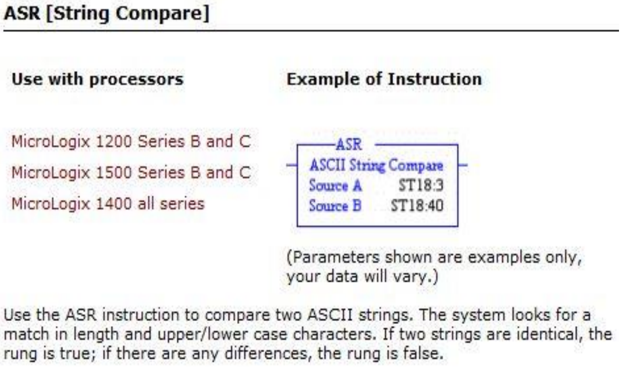
ARD
ASCII Read
Channel 0
Dest ST17:6
Control R6:17
String Length 12
Characters Read 0
Error 0

(Parameters shown are examples only, your data will vary.)

Use the ARD instruction to read characters from the buffer and store them in a string. To repeat the operation, the rung must go from false-to-true.

Fig 2. ARD instruction to receive data

When the data was transmitted from Arduino Uno it got stored in buffer memory in PLCs CPU and so as to compare retrieved data a special instruction “ASR” is used, which is described in Fig 3[1].



**ASR [String Compare]**

**Use with processors**

MicroLogix 1200 Series B and C
MicroLogix 1500 Series B and C
MicroLogix 1400 all series

**Example of Instruction**

ASR
ASCII String Compare
Source A ST18:3
Source B ST18:40

(Parameters shown are examples only, your data will vary.)

Use the ASR instruction to compare two ASCII strings. The system looks for a match in length and upper/lower case characters. If two strings are identical, the rung is true; if there are any differences, the rung is false.

Fig 3. ASCII compare instruction to compare received data from already stored data

### B. ARDUINO CONFIGURATION

Arduino Uno [8] being open source is featured for easy configuration and programming as well. Arduino Uno is has one Serial Port on pin 0 and 1. We used Software Serial library to send and receive data from one serial port to other so that we can verify the data received to Arduino Uno by connecting it to PC.

### C. INTERCONNECTION

The programmed Arduino and PLC is interconnected to each other with the MAX232 based connector as shown in the Fig 4

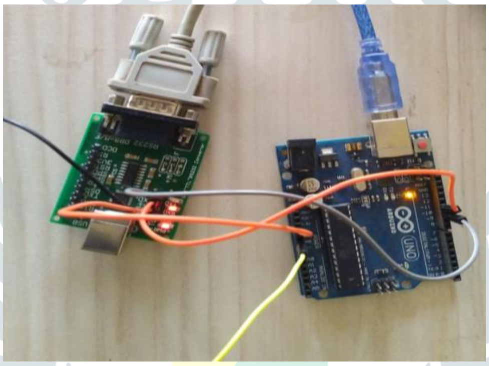


Fig 4. Interconnection between PLC and Arduino Uno via IC MAX232 based connector

MAX232 IC is used for data conversion from RS232 logic levels to TTL logic levels and vice versa.[3] The IC has capacitive voltage generator to supply for RS232 port which works on higher logic levels from 5V logic levels. And circuitry for supplying to 5V logic levels from 25V levels of RS232. The block diagram for setup is shown in Fig. 5

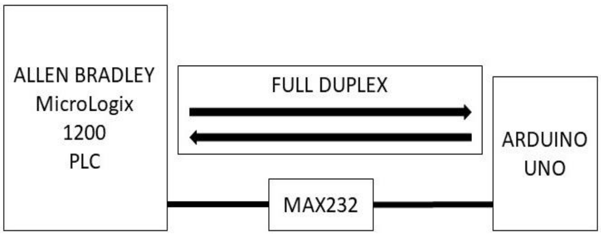


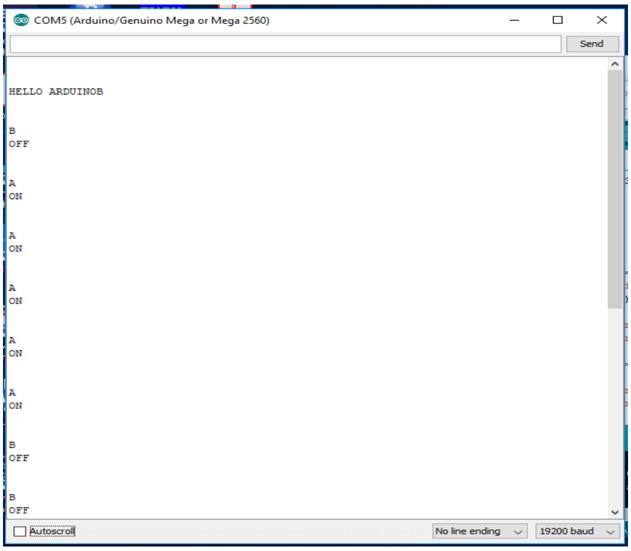
Fig 5. Block diagram of Experimental Setup

## III. RESULTS

Experimentation was done by sending data from PLC over the created connection in the form of Strings (that were already stored in PLC’s String Data file [1]) to the Arduino Uno. This data was send periodically with higher data rate by using PLC timer (TON) instruction [1]. In the implementation following procedure was followed-

1. PLC configuration for full duplex communication mode using RSLinx.
2. Ladder program.
3. Arduino configuration and program.

This project was developed for implementation of conveyer belt system for detecting the presence of object on the belt. IR proximity sensor was interfaced to the PLC for object detection. IR sensor detected objects which were passed on continuously with fixed time interval in between. PLC sent character “A” continuously which indicated that the objects were present on the conveyer. For missing object position PLC sent character “B” continuously. The data as received by Arduino Uno from PLC is shown in Fig 6.



```
HELLO ARDUINOB

B
OFF

A
ON

A
ON

A
ON

A
ON

A
ON

B
OFF

B
OFF
```

Fig 6. Data received from PLC.

## IV. CONCLUSION

In this project we implemented data transmission between two non-homogenous controlling devices. This setup was used to detect the presence of object on conveyer belt. We selected AB PLC MicroLogix1200, RSLogix500, Arduino Uno and MAX232 based level shifter. It was experimentally verified that the data transfer from PLC to Arduino Uno was successfully received.

## V. FUTURE SCOPE

The above described projects work on single master-slave setup wherein AB PLC is the master device and Arduino Uno being the slave device. We will develop a multiple slave setup in BUS topology.

Using this setup we will implement the data transfer to internet through Arduino's extension modules like ESP8266 based Wi-Fi module, HC-05 based Bluetooth module, ZigBee modules.

## VI. REFERENCES

[1] "MicroLogix 1200 Programmable Controllers"- Bulletin 1762 Controllers and 1762 Expansion I/O, Allen Bradley, User Manual

[2] RSLinx "Training Guide", RSLinx 2.20, Revision-2/27/2002

[3]Texas Instruments, "MAX232x Dual EIA-232 Drivers/Receivers", MAX232 datasheet, February 1989 (Revised November 2014).

[4]Zeebaree, S.R.M, and Yasin H.M. (2014). "Arduino based remote controlling for home power saving, security and protection". International Journal of Scientific and Engineering Research, 5(8), 266-272.

[5]Reneker, D. (2017). PLC vs. Arduino for industrial control. Retrieved June 28, 2018, from https://www.controldesign.com/articles/2017/arduino-vs-plc-for-industrial-control/.

[6]Mir, Arsheen & Rajguru, Swarnalatha. (2018). "Implementation of an industrial automation system model using an Arduino". Journal of Engineering Science and Technology. 13. 4131-4144.

[7]Yueqin Yin, Shaanxi Electric Power Company. "Application of PLC Control System and Communication Technology in Chemical Plant Equipment Renovation". Chemical Engineering Transactions.