Real-Time Simulation Of A Networked-Process Control System For A Chemical Industry

Isizoh A. N.[1]
Dept. of Electronic and Computer Engineering, Nnamdi Azikiwe University, Awka, Nigeria.

Anazia A.E.[2]
Dept. of Electrical Engineering, Nnamdi Azikiwe University, Awka, Nigeria.

Okide S. O.[3]
Dept. of Computer Science, Nnamdi Azikiwe University, Awka, Nigeria.

Okwaraoka C.A.P.[4]
Dept. of Electrical/Electronics Engineering, Federal Polytechnic , Nekede, Imo State, Nigeria

Abstract

This paper analyzes the real-time simulation of a networked-process control system for a chemical industry. The system has four Access layer servers connected to several loads via four Controller Area Networks (CANs) using Proteus software (Version 7.6). A typical industrial application used as a case study for this work is a chemical industry which produces Vinyl Chloride, a major raw material used in the production of Poly Vinyl Chloride (PVC) products. The system network was developed in a Proteus environment by interconnecting the Access layer servers, Ethernet Controller (which establishes a Local Area Network, LAN), and CAN Controllers. This study was implemented in the control of the industrial processes from the Access layer servers in a real-time environment. This was achieved by replacing the Ethernet controller with a programmed AT89C55 microcontroller to achieve the same switching logic, and this confirmed the workability of the system in a real-life situation.

Keywords: CAN controller, Proteus, Poly Vinyl Chloride, Local Area Network, Microcontroller.

1. Introduction

It is not new to say that there are many ways of controlling loads that are remotely distributed in a large industry or factory, such as using GSM handset, Internet-based method, etc. One of such ways that even appears to be the most effective method is by the use of an Ethernet controller to form a LAN network which connects many computers together. And these computers which form the nodes in that network are used to control several loads via different CAN controllers [1]. One good thing about this method is that each load in the network is linked to a particular CAN controller, and this enhances excellent control performance.
How this is achieved is that, each CAN controller is given an address which uniquely identifies that CAN, and makes it accessible at the Access layer terminal.

1.1. The Control System Block diagram

The block diagram of the system comprises of five distinct units/parts as shown in figure 1. The units are: the Access Terminal Unit, the Switching Unit, the Signal Indicator Panel Unit, the Controller Unit and Loads.

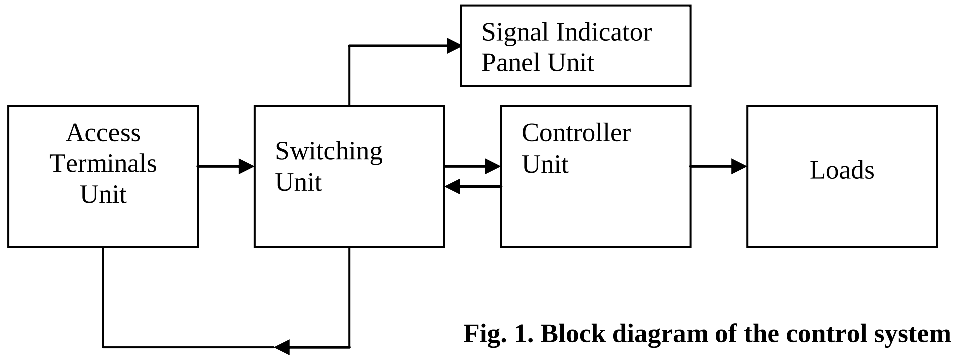


Fig. 1. Block diagram of the control system

Access Terminal Unit: This comprises of access layer servers. It can be of any number, depending on the user's request. Here, only four were used.
The primary function of this access terminal unit is to supply the needed address of each of the CAN controllers, based on their load priorities.
Thus in this work, the access terminals are logically connected to the virtual ports of the Ethernet in a Proteus interface.

Switching Unit: This unit principally consists of an Ethernet. It is distributive in the sense that it ensures that any address which emanates from any of the access terminal servers uniquely identifies a particular CAN controller. Since the typical industrial scenario used in this research is a chemical industry, the network to be provided is a Local Area Network (LAN) because the industry is assumed to be within 100 meters square.

Controller Unit: The main device in this unit is the Controller Area Network (CAN). It is a vehicle bus standard designed to allow microcontrollers and devices communicate with each other within a vehicle without a host computer. Each CAN bus has an address which uniquely identifies it. When an operator or user wants to control a device connected to any CAN,
the operator has to enter the address or identity of that CAN on any of the access terminals together with some appropriate codes.

Loads: These are the devices to be controlled. But in this paper, Light Emitting Diodes (LEDs) were used as loads to provide the necessary animations.

Signal Indicator Panel: All the control indicators like: error A, error B, error C, error D, receive and transmit messages are displayed on this panel. It is left for the programmer to determine the duration of each light.

2. The Layout model

The layout model of the entire system, known as Isizoinyiama layout model, is shown in figure 2. The layout involves four access terminals connected to an Ethernet. From this Ethernet, connections are made to different Controller Area Network (CAN) devices. With this arrangement, several loads can then be connected to the CAN devices.

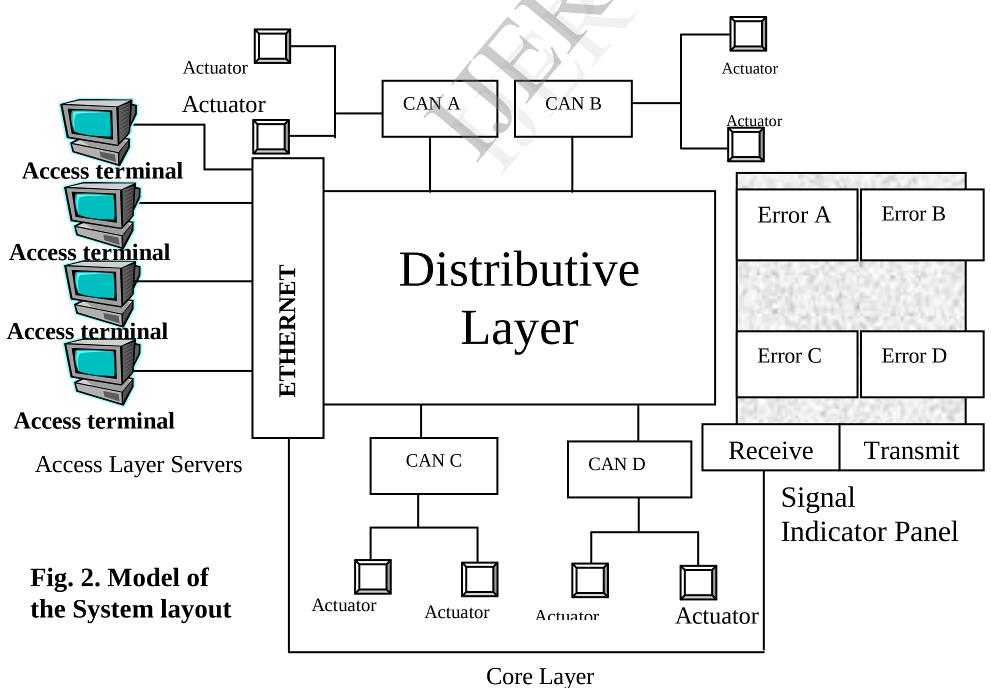


Fig. 2. Model of the System layout

A simplified model of this control system is shown in figure 3 below. There are two nodes. Node 1 has the Ethernet and the four Access terminals, while node 2 has four CANs and the loads. The model uses tree topology.

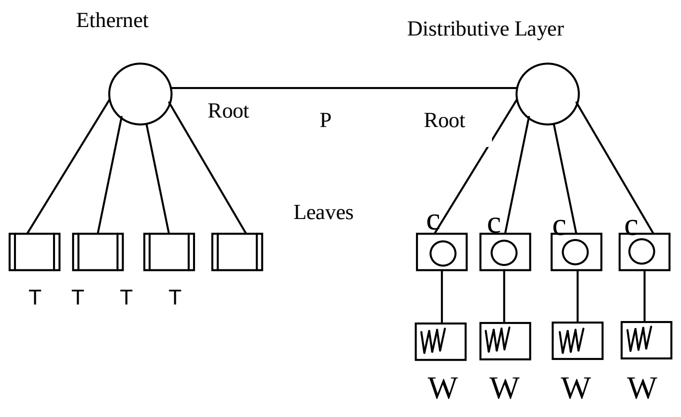


Fig. 3. Simplified model of the system

Where:

T = Access layer server
C = Controller Area Network
W = Loads

3. Development of the Switching Function

Every Ethernet controller is usually developed to provide a good switching function for its network. It is the part of the switch that stores or
carries the program for the switching function of the Ethernet.
In this paper, this controller was replaced with an AT89C55 microcontroller. Thus, since the microcontroller is a computer of itself, the design of the switching function must be a software-base [2].

In the Proteus interface, the developed switching system of the Ethernet is seen by placing cursor on the Ethernet controller in the diagram of figure 4, and pressing Ctrl C. When this is done, it will be seen that there are four AT89C55 microcontrollers labeled U1, U2, U3 and U4.
Each microcontroller pairs with one particular virtual terminal in order to control the distributed loads via CAN controllers. The analysis of each microcontroller arrangement here is the same with other three microcontrollers.

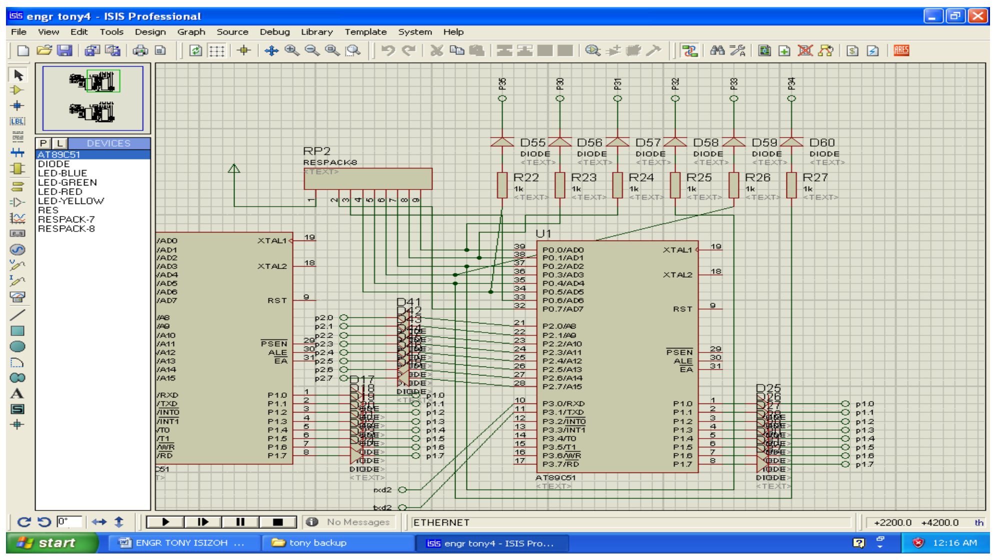


Fig. 4. Detailed connection of one microcontroller

By default, Port 0 (i.e. pin 32 to pin 39) is zero. Therefore to make it active (i.e. to have ones on pin 32 to pin 39), resistor pack-8 (labeled RP) called pull-up resistors are connected to the port (i.e. pins 32, 33, 34, 35, 36, 37, 38 and 39). Also in this Port 0, pin 34 to pin 39 are connected to the Signal Indicator Panel of the layout as listed below.

Pin 39	$\longrightarrow$	Error A (LED)
Pin 38	$\longrightarrow$	Error B (LED)
Pin 37	$\longrightarrow$	Error C (LED)
Pin 36	$\longrightarrow$	Error D (LED)
Pin 35	$\longrightarrow$	Received (LED)
Pin 34	$\longrightarrow$	Transmit (LED)

To enable the above connection be achieved, these pins are connected to resistors (of 10kΩ each). These resistors are to limit the currents entering into the Light Emitting Diodes (LEDs) in the Signal Indicator Panel through their individual diodes (D61-D66) which serve a special purpose of making sure that there is no reverse current.
The Port 1 and Port 2 of the microcontroller are connected to the distributed loads in the network.
The connections between the virtual ports of the microcontroller and the CAN controllers are as written below:
For Port 1,

P1.0	$\longrightarrow$	CAN A, P_0	$\longrightarrow$	D_1
P1.1	$\longrightarrow$	CAN A, P_1	$\longrightarrow$	D_2
P1.2	$\longrightarrow$	CAN A, P_2	$\longrightarrow$	D_3
P1.3	$\longrightarrow$	CAN A, P_3	$\longrightarrow$	D_4
P1.4	$\longrightarrow$	CAN B, P_4	$\longrightarrow$	D_5
P1.5	$\longrightarrow$	CAN B, P_5	$\longrightarrow$	D_6
P1.6	$\longrightarrow$	CAN B, P_6	$\longrightarrow$	D_7
P1.7	$\longrightarrow$	CAN B, P_7	$\longrightarrow$	D_8

For Port 2,

P2.0	$\longrightarrow$	CAN C, P_8	$\longrightarrow$	D_9
P2.1	$\longrightarrow$	CANC, P_9	$\longrightarrow$	D_{10}
P2.2	$\longrightarrow$	CANC, P_{10}	$\longrightarrow$	D_{11}
P2.3	$\longrightarrow$	CANC, P_{11}	$\longrightarrow$	D_{12}
P2.4	$\longrightarrow$	CAND, P_{12}	$\longrightarrow$	D_{13}
P2.5	$\longrightarrow$	CAND, P_{13}	$\longrightarrow$	D_{14}
P2.6	$\longrightarrow$	CAND, P_{14}	$\longrightarrow$	D_{15}
P2.7	$\longrightarrow$	CAND, P_{15}	$\longrightarrow$	D_{16}

The Port 3 of the microcontroller is not fully connected to any input/output (I/O) device except pin 10 and pin 11 (i.e. P3.0 and P3.1) which are connected to a virtual terminal (or access terminal). It should be noted that each access terminal is connected to a particular microcontroller.
One can use Ctrl X to get back to the normal interface or 'Home' environment.

4. Considerations

Each microcontroller is assumed to be a terminal of its own and must be paired with one access terminal. All the diodes used in this work are IN4001. They are all signal diodes used for directing signals.

Here, each resistor that was used in the biasing of each of these diodes is 1KΩ. This is because the AT89C55 microcontroller manufacturer's standard stipulates that the maximum allowable pin current for its 'sinking' is 5mA [3].
So since V = IR ……………… (1)
then R = V / I ……………... (2)
But V = 5V, and I = 5mA
Therefore, R = 5 / 0.005, which is 1000Ω or 1K Ω.

Each RP (Resistor Pack) used in this work has 8 resistors inside one pack. It is a packed pull-up resistors connected to port 0 of each microcontroller. The reason for this connection is that, by default, the Port 0 of this microcontroller is actively low, and to be able to use it, this port has to be made actively high by connecting pull-up resistors to the port [4].
Based on the AT89C55 manufacturer's standard, the value of each of these resistors in a pack is between 1KΩ to 10KΩ. But in this paper, 10KΩ was used.

It takes each microcontroller a default maximum time of 2mS to set up for operation from power-up. This is a reset time to enable it get set for its internal operation [5].
Thus it is a standard thing that an RC circuit is connected to pin 9 of each microcontroller, and this RC circuit is also called a reset circuit. This circuit generates the reset time, T for the microcontroller.
So the reset pin 9 of each microcontroller has a capacitor, C and a resistor, R. Based on AT89C55 manufacturer's standard, the choice of C and R are:
C = 10μF
R = 10KΩ.
This is because the period, T = 1.1 RC … (3).
Since C = 0.00001F, and R = 10000Ω, then,
T = 1.1 x 0.00001 x 10000 = 1.1mS. So these values of C and R are okay, since the value of T is not up to 2mS.
The AT89C55 has an on-chip oscillator but requires an external clock to run it. A quartz crystal oscillator is connected to inputs XTAL 1 (pin 19) and XTAL 2 (pin 18). The essence of using crystal oscillator is to control the speed of the microcontroller. Here, the speed of the microcontroller refers to as the maximum oscillator frequency connected in between XTAL 1 and XTAL 2. In this paper, a crystal oscillator with 12MHz frequency was used. The frequency can be observed on the XTAL 2 (pin 18) using an oscilloscope [6].

5. The Flowchart

The flowchart for the system operation is represented as below:

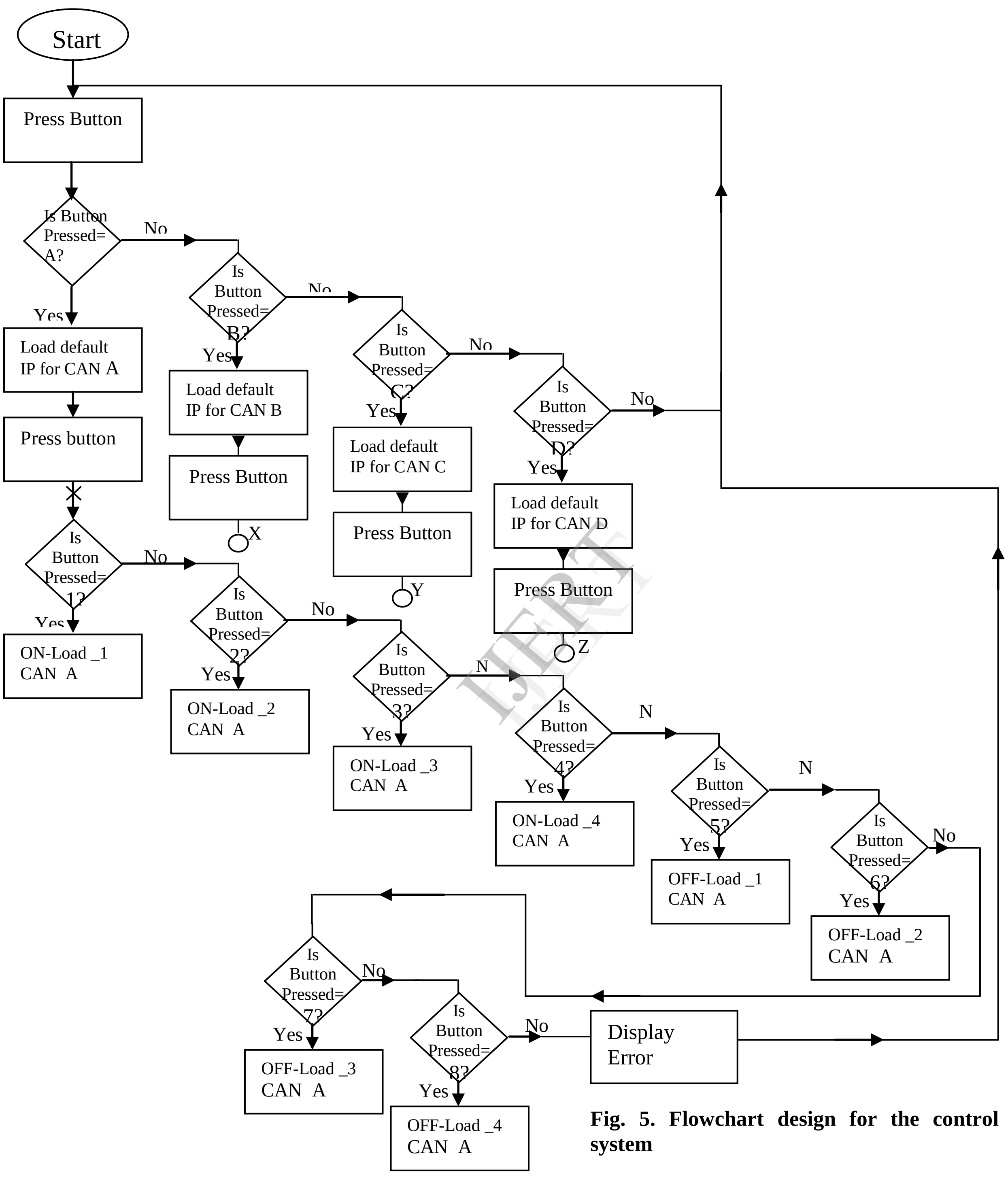


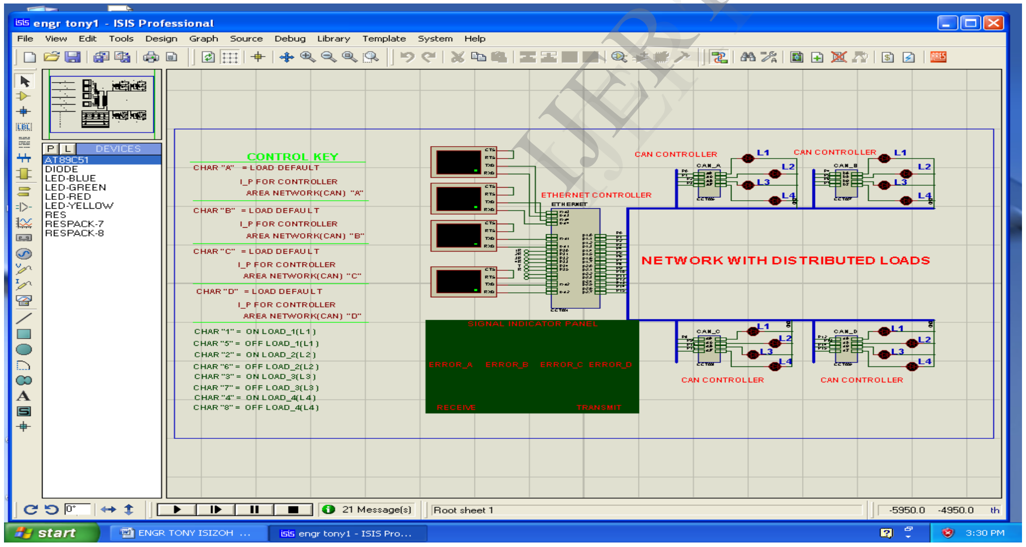
Fig. 5. Flowchart design for the control system

6. Software Development

Every microcontroller operation must be based on its control program. Without programming a microcontroller, it can never work. The pseudocode and flowchart above will enable software development be achieved.
A program for the control operation was developed in Assembly language using MIDE software. This original program is called the source program, while the numeric, microcontroller-compatible form of it is the object program. The assembler's job here is to convert the source program you can understand into an object program the microcontroller can understand [7].

7. System Implementation

The entire system can be set up for implementation as shown in figure 6. The Ethernet switch establishes the LAN network needed for this system implementation. The default setting of each virtual terminal is such that the CTS and RTS are connected together. With this provision, the four access terminals or nodes are connected to the Ethernet controller via the virtual ports of txd1, txd2, txd3, and txd4 for the four nodes respectively.
The output ports (P1 and P2) are connected to the CAN controllers, with each CAN port uniquely identified by its IP address. Each CAN has four loads connected to it. The loads are arranged as L1, L2, L3, and L4. In general, there are 16 loads connected to the network. For the purpose of this study, the loads are represented using LEDs.
Based on the above arrangement, the distributed loads can be controlled at any access terminal by typing the CAN controller address in capital letter, for example, A, B, C, or D. If what you entered was not any of these letters A to D, there would be a feedback error message, titled "Error". Also if the letters A, B, C, or D you entered was not in capital letter (example, a, b, c, or d), error message would be displayed. Thus, if an error message is displayed, no operation can take place.
When the correct CAN address has been entered, the default ID address is automatically loaded on the access terminal, and then one is expected to enter the load control address or number. The operation at this stage will determine whether one wants to put on a load or put off a load.



CONTROL KEY
CHAR "A" = LOAD DEFAULT I_P FOR CONTROLLER AREA NETWORK(CAN) "A"
CHAR "B" = LOAD DEFAULT I_P FOR CONTROLLER AREA NETWORK(CAN) "B"
CHAR "C" = LOAD DEFAULT I_P FOR CONTROLLER AREA NETWORK(CAN) "C"
CHAR "D" = LOAD DEFAULT I_P FOR CONTROLLER AREA NETWORK(CAN) "D"
CHAR "1" = ON LOAD_1(L1)
CHAR "5" = OFF LOAD_1(L1)
CHAR "2" = ON LOAD_2(L2)
CHAR "6" = OFF LOAD_2(L2)
CHAR "3" = ON LOAD_3(L3)
CHAR "7" = OFF LOAD_3(L3)
CHAR "4" = ON LOAD_4(L4)
CHAR "8" = OFF LOAD_4(L4)

Fig. 6. Implementation of the system

The system can be implemented in the control of processes going on in an industry. A typical industry used as a case study for this work is a chemical industry that produces vinyl chloride (a major raw material used in the production of Polyvinyl Chloride i.e. PVC materials). The diagram in figure 7 shows a typical layout of the chemical plants.

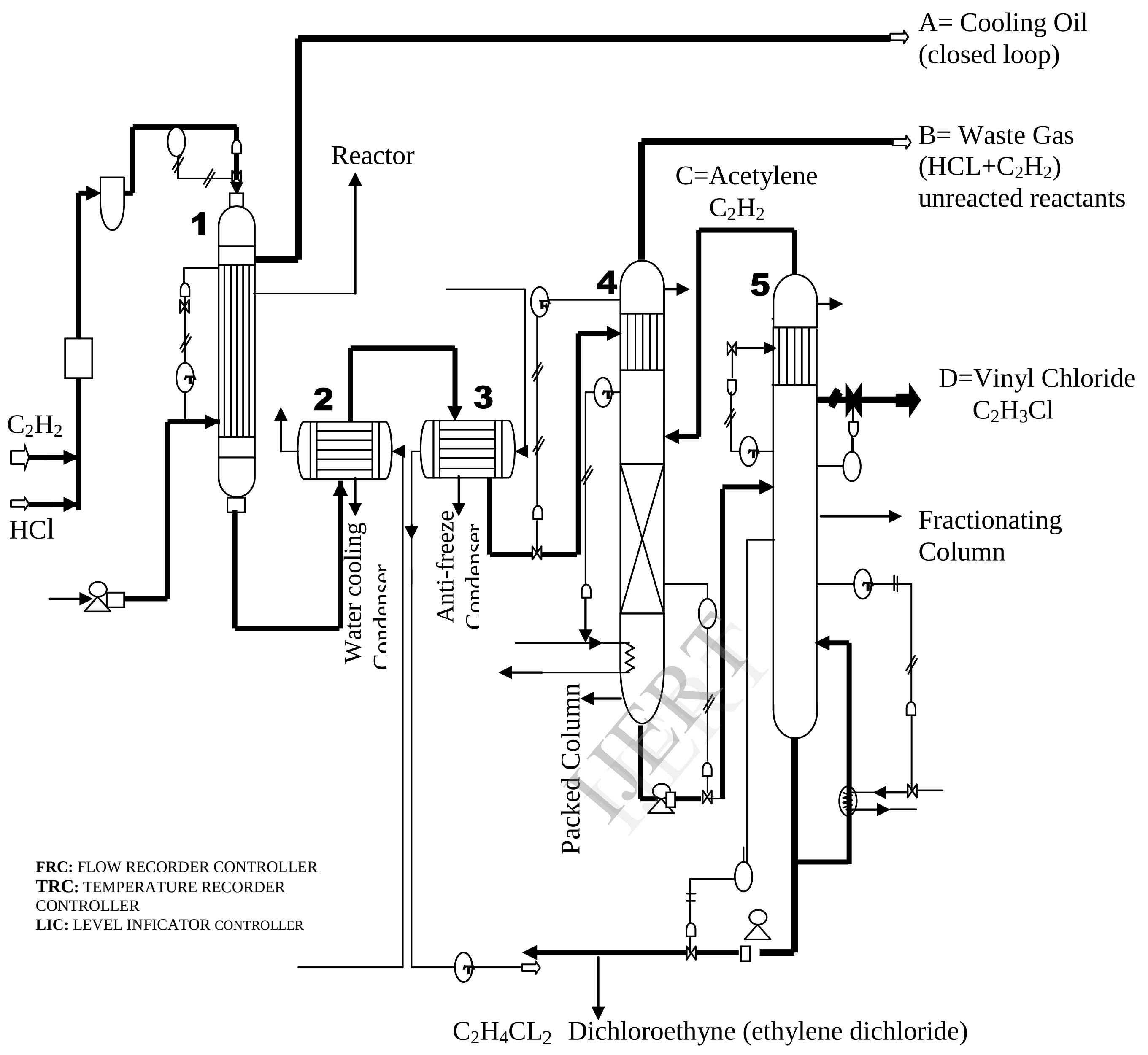

A= Cooling Oil (closed loop)
B= Waste Gas ($HCL+C_2H_2$) unreacted reactants
C=Acetylene C_2H_2
Reactor
1
2
3
4
5
D=Vinyl Chloride C_2H_3Cl
C_2H_2
HCl
Water cooling Condenser
Anti-freeze Condenser
Fractionating Column
Packed Column
FRC: FLOW RECORDER CONTROLLER
TRC: TEMPERATURE RECORDER CONTROLLER
LIC: LEVEL INFICATOR CONTROLLER
$C_2H_4CL_2$ Dichloroethyne (ethylene dichloride)

Fig.7. Chemical plants where the system was implemented

Unit 1: Reactor – This is a chamber where the two reactants (HCl and C_2H_2) mix or react together. One of the outlets (A) is where the cooling oil is recycled. The other outlet goes into the water cooling condenser.

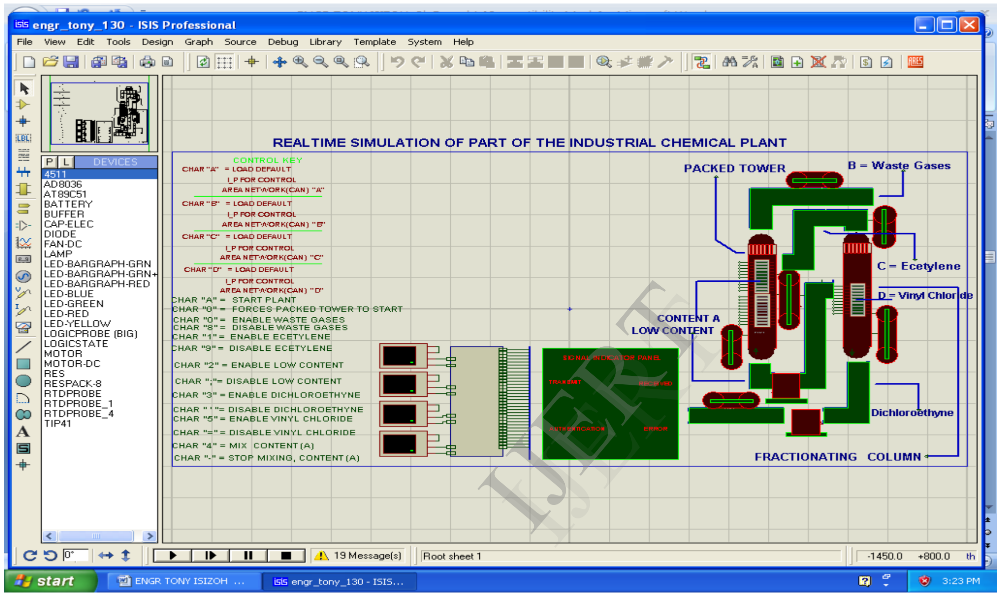
Unit 2: Water Cooling Condenser – The reaction that occurs inside the reactor above is exothermic, meaning that heat is given out during the process of reaction. Therefore this unit 2 serves as a cooling chamber after the reaction. It cools the affluent (ie. both the product and unreacted substances). The coolant used here is water.

Unit 3: Antifreeze condenser – This unit plant is also a heat exchanger like the unit 2 plant. The coolant here is not water, but a fluid with a very low freezing point.

Unit 4: Packed Column – This unit plant has packed beads and heater. It is also called stripping column. The inlet from unit 3 contains both liquid and gases. This plant is used for separation process. One of the process outlets contains waste gases (B), while the other contains the product that enters into the fractionating column.

Unit 5: Fractionating Column – This plant contains plates instead of beads. It is used for fractional distillation. When its content is heated, separation process goes on. Here liquid vinyl chloride (C_2H_3Cl) is collected at the (D) outlet. The bottom affluent is dichloroethyne or ethylene dichloride (C_2H_4Cl) which is collected at E outlet. The upper outlet, C contains Acetylene gas (C_2H_2) which is fed back into the packed tower for recycling [8].

The figure 7 above was connected to the developed work in figure 6 using Proteus software as shown in figure 8 below. Here the loads (used as LEDs) in figure 6 were replaced with valves in figure 7. However, due to the lengthy nature of the designed software (program), only units 4 and 5 were used.
One should understand that when Proteus software is used to implement a virtual system (ie. real-time simulation), not all the system components are placed on the layout. Some are placed in the sub sheet e.g. CAN controllers; some are not visible but are there by default e.g. reset circuit, crystal oscillator circuit, power, etc; while some are taken care of by the control program. In Proteus design, it is called “referencing”.


engr_tony_130 - ISIS Professional
File View Edit Tools Design Graph Source Debug Library Template System Help
DEVICES
REALTIME SIMULATION OF PART OF THE INDUSTRIAL CHEMICAL PLANT
PACKED TOWER
B = Waste Gases
C = Ecetylene
D = Vinyl Chloride
CONTENT A
LOW CONTENT
Dichloroethyne
FRACTIONATING COLUMN
19 Message(s)
Root sheet 1
start
3:23 PM

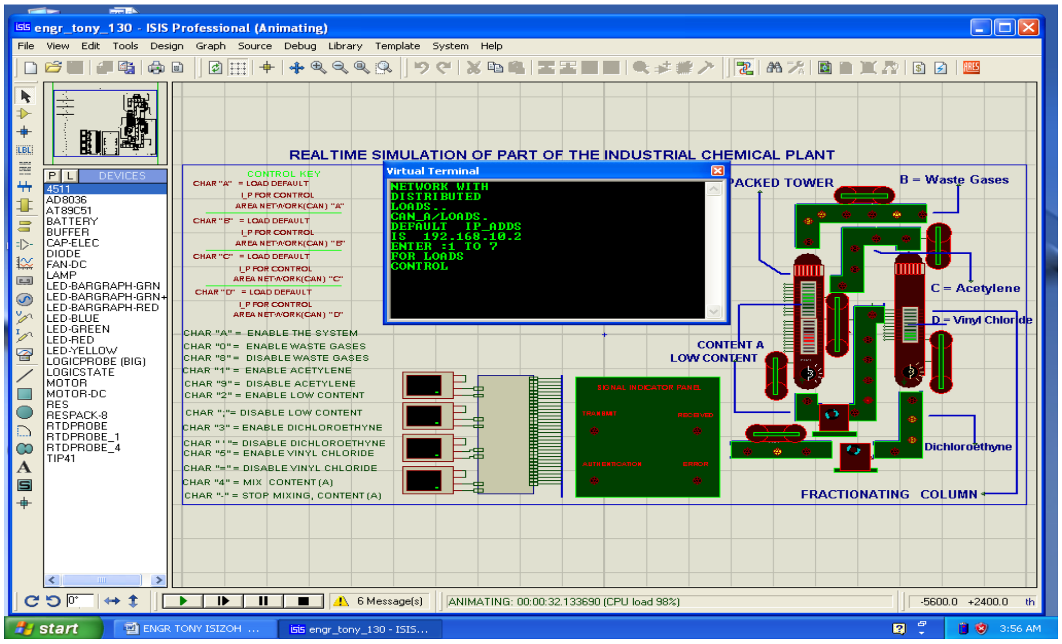
CONTROL KEY
CHAR "A" = LOAD DEFAULT I_P FOR CONTROL AREA NETWORK(CAN) "A"
CHAR "B" = LOAD DEFAULT I_P FOR CONTROL AREA NETWORK(CAN) "B"
CHAR "C" = LOAD DEFAULT I_P FOR CONTROL AREA NETWORK(CAN) "C"
CHAR "D" = LOAD DEFAULT I_P FOR CONTROL AREA NETWORK(CAN) "D"
CHAR "A" = START PLANT
CHAR "0" = FORCES PACKED TOWER TO START
CHAR "0" = ENABLE WASTE GASES
CHAR "8" = DISABLE WASTE GASES
CHAR "1" = ENABLE ECETYLENE
CHAR "9" = DISABLE ECETYLENE
CHAR "2" = ENABLE LOW CONTENT
CHAR ";"= DISABLE LOW CONTENT
CHAR "3" = ENABLE DICHLOROETHYNE
CHAR " ' "= DISABLE DICHLOROETHYNE
CHAR "5" = ENABLE VINYL CHLORIDE
CHAR "=" = DISABLE VINYL CHLORIDE
CHAR "4" = MIX CONTENT (A)
CHAR "-" = STOP MIXING, CONTENT (A)

Fig. 8. Implementation of the system for chemical process control

To implement this, the valves in figure 7 can either be opened or closed. If any valve is opened by using the control guide of any of the access terminals or nodes, the content will be flowing (i.e. by animation using Light Emitting Diodes). But if a valve is closed, the content of the pipe cannot come out.

8. Real-Time Simulation Results

It is always necessary to perform a real-time simulation of any microcontroller-based system you have developed. This is because a program for a microcontroller operation may be error-free, but when it is implemented, the desired goal may not be achieved. It has been noted that any system that works in a real-time simulation must work in a real-life situation.
The real-time simulation of the control system used as a typical industrial scenario is achieved by clicking the play button in the Proteus environment. When this is done, the four virtual terminals or nodes will be displayed. But here only one terminal is seen because the simulation was done for one virtual terminal. Click on that virtual terminal, and then load the default address by typing capital “A”. The control guides for the process control are shown below. Figure 9 shows the real-time simulation of the industrial chemical plant.



REALTIME SIMULATION OF PART OF THE INDUSTRIAL CHEMICAL PLANT

CONTROL KEY
CHAR "A" = LOAD DEFAULT I_P FOR CONTROL AREA NETWORK(CAN) "A"
CHAR "B" = LOAD DEFAULT I_P FOR CONTROL AREA NETWORK(CAN) "B"
CHAR "C" = LOAD DEFAULT I_P FOR CONTROL AREA NETWORK(CAN) "C"
CHAR "D" = LOAD DEFAULT I_P FOR CONTROL AREA NETWORK(CAN) "D"

CHAR "A" = ENABLE THE SYSTEM
CHAR "0" = ENABLE WASTE GASES
CHAR "8" = DISABLE WASTE GASES
CHAR "1" = ENABLE ACETYLENE
CHAR "9" = DISABLE ACETYLENE
CHAR "2" = ENABLE LOW CONTENT
CHAR ";"= DISABLE LOW CONTENT
CHAR "3" = ENABLE DICHLOROETHYNE
CHAR " ' "= DISABLE DICHLOROETHYNE
CHAR "5" = ENABLE VINYL CHLORIDE
CHAR "=" = DISABLE VINYL CHLORIDE
CHAR "4" = MIX CONTENT (A)
CHAR "-" = STOP MIXING, CONTENT (A)

Virtual Terminal
```
NETWORK WITH
DISTRIBUTED
LOADS..
CAN_A/LOADS.
DEFAULT  IP_ADDS
IS  192.168.10.2
ENTER :1 TO 7
FOR LOADS
CONTROL
```

Fig. 9. Real-time simulation for a typical chemical industry

The guides for the load control are:
CHAR “A”- Enable the system
CHAR “0”- Enable Waste gases
CHAR “8” -Disable Waste gases
CHAR “1”- Enable Acetylene
CHAR “9”- Disable Acetylene
CHAR “2”- Enable Low content
CHAR “;” - Disable Low content
CHAR “3” - Enable Dichloroethyne
CHAR “’” - Disable Dichloroethyne
CHAR “5” - Enable Vinyl chloride
CHAR “=” - Disable Vinyl chloride
CHAR “4” - Mix content “A”
CHAR “_” - Stop mixing content “B”

9. Conclusion

A very good process control system with distributed loads has been obtained. This was done by replacing the intelligent switch of an Ethernet controller with AT89C55 microcontroller which gives the same switching logic. In order to achieve this logic, a workable control program, written in Assembly Language was designed using MIDE software and TIME 8051 software. The program was later tested okay.

10. References

[1] Hassan Saeed S.,”Automatic Control Systems”, S.K. Kataria & Sons Publishers, New Delhi. 2008.S

[2] Ani C.O., ”Assembly Language Programming”, Immaculate Publications Ltd, Enugu, Nigeria, 2002.

[3] Douglas V.H., “Microcontrollers and Interfacing: “Programming Hardware” McGraw Hill Inc, New York, 2008.

[4] Schuster A., “Microcontroller Principles and Applications”, Maxon Press Ltd, Rochester, 2008.

[5] Holliday D. and Resmick Robert, “Fundamentals of Microcontrollers”, Don Peters Press Ltd, Fulmar, 2008.

[6] Kopetz H., “Real-Time Systems: Design Principles for Distributed Embedded Applications”, Kluwer Academic Publishers, Dordrecht, 2009.

[7] Wakerly John F., “Digital Designs: Principles and Practices”, 4th Edition, PHI Learning Private Limited, New Delhi, 2008.

[8] Luyben L. Willian, “Process Modeling, Simulation and Control for Chemical Engineers”, Second Edition, Mc Graw-Hill Publishing Company, India, 2006.