ASYNCHRONOUS COMMUNICATION PROTOCOL

PROJECT REPORT

Submitted in partial fulfillment of the requirements
For the award of Degree of

MASTER OF COMPUTER APPLICATIONS

MAHATMA GANDHI UNIVERSITY.

By

SALINY SANKAR. S
Reg No.42329

Under the guidance of Done at
MR.SURESH SABHAPATHY **GMRT,PUNE**

SCHOOL OF TECHNOLOGY AND APPLIED SCIENCES
REGIONAL CENTRE M.G.UNIVERSITY
PATHANAMTHITTA
2002-2005

CERTIFICATE

This is to certify that this project entitled "**Asynchronous communication protocol**" is a bona fide record of the original work done by Miss.**Saliny Sankar.S**(Reg.No.42329) submitted in the partial fulfillment of the requirements for the award of Master of Computer Application under the Mahatma Gandhi University, during the year 2003-2006

Submitted for the VIVA-VOCE **Examination held on**

Internal Examiner **External Examiner**

DECLARATION

I hereby declare that the project work entitled ‘Asynchronous communication protocol’ was done by me in **Giant Meterwave RadioTelescope,** Pune, under the guidance of **Mr. Suresh Sabhapathy,**Electronic Engineer, Giant Meterwave RadioTelescope, Pune and **Dr.P.R.Prince** internal guide, STAS, Pathanamthitta in the partial fulfillment of the requirement for the degree ‘Master Of Computer Applications’.

SALINY SANKAR.S

ACKNOWLEDGEMENT

First of all, I express my sincere and heartful thanks to **Almighty God** for all his blessings to complete the project work. I extend my sincere gratitude to **Dr. P.R.Prince, Regional Director, STAS, Pathanamthitta** for giving me the opportunity to do project in this esteemed institute. Secondly i would like to express my thanks to **Dr.Ishwar Chandra.**This project would not have been possible without his untiring effort.

Let me start by expressing my sincere thanks and gratitude to my project guid**e Mr.Suresh Sabhapathy**.I feel that I have been extremely lucky to encounter an advisor like him.Sir is not only an engineer with deep vision,but also,and most importantly,a very kind person.He was always there when I needed him,his numerous ideas,suggesions & criticism helped me in most important moments of making right decisions.All thanks to sir,for always believing in me and reminding me that the journey to a goal,is one step at a time.

I am also greatly indebted to all the members of the staff at GMRT,specially servo ,for being ever so helpful in all their interactions with me. A good share of gratitude is reserved for **my friends** who gave me such good company all though my stay.

CONTENTS

INTRODUCTION

1.INTRODUCTION

1.1 ABOUT THE ORGANISATION

The Giant Meter wave Radio Telescope (GMRT) is set up as a national facility for front line research in astronomy and astrophysics is located at Khodad,about 80 km north of Pune. It is set up and operated by the National Centre for Radio Astrophysics under the Tata Institute of Fundamental Research. This facility is one of the largest radio telescope in the world .It is an aperture synthesis array comprising 30 fully steerable parabolic dishes of 45km diameter each. In order to facilitate observation of both compact as well as ex tented sources,fourteen dishes are grouped together in a central square configuration while the rest sixteen are distributed along three arms in a Y-schemes.

Each of the 30 GMRT antennas is steered using an electronic servo system. The servo system is required to meet two main requirements.

1 It should be able to point the antenna anywhere in the sky.

2 It should be able to point to $\pm$ 10% of 3dB radio beamwidth for a given wavelength.

3 To counter the effect of the earth's rotation,the antenna must be able to track the object under observation.

The servo system is controlled by a station servo computer which carries out overall supervision of all subsystems. The station servo computer is built around bus structured processor and I/O boards. The bus master is 8086-2 processor running at 8MHZ.The field I/O is interfaced to the SSC through back-panel mounted D connectors.

It interfaces to the following equipments.

- AZ/EL position encoders and potentiometers.
- 2 no. of wind meters.
- Control switches,status indicators and display.
- Motor brakes and status outputs.
- Slow mechanisms
- Amplifiers
- Hand Held Terminal
- Antenna Base Computer
- Position Limit Switches

The SSC performs the following functions

- Closed loop position control in local and remote modes of operation
- Handling control and monitor commands from remote ABC.
- Generating demand angle trajectory every 100msec based on data received from ABC.
- Position measurement and display after off-set correction.
- Scanning contact inputs;performing operational and safety interlocking logic and driving relay outputs.
- Limit release operation.
- Stow,stow release and parking operation.
- Monitoring currents,speeds,wind speeds etc. and generating trip and inter_lock conditions.
- Handling user commands from HHT for set up,display and control.
- Watch dog trigger.
- Time of Day.
- Power on self test and diagnostics.

Circuit description of SSC

processor	8086-2 (8 MHZ)
memory	128 K EPROM (2X 27512)
	64K RAM (2X 62256)
	8K EEPROM (1X 2864A)
bus i/f	8/16 bit mono-master bus
serial i/f	channel 1-RS232/RS422(ABC)
	channel 2-RS232 (HHT)
timers	timer1- 10 m sec tick
	timer2-baud rate generator
watch dog	triggered every 100msec. On expiry
timer	resets and restarts the system
fail-safe	triggered every bus cycle by ALE SIGNAL
interrupts	total 8

1.2ABOUT THE PROJECT

The antenna-base computer and the servo computer are connected with an RS 422 link. The SSC-LINK unit is entrusted with the responsibility for handling the hardware and link layer functions. The part of the software handling hardware is small. It consists of one procedure called 'tx_rut'. This procedure reads from the serial communication port for reception and writes to the same address for transmission. The availability of a byte for reading is indicated by high D0 bit in the UART status register. Transmission is done after ensuring that the transmit buffer is free that is indicated by the status byte of the UART.

A part from the hardware layer,this asynchronous link protocol unit incorporates the code for the link layer. The job of the link layer is to serve as a bridge between the application and the hardware layer so as to make the details

of the hardware inconsequential to the application. To achieve this,this unit contains two state machines-the transmission state machine called 'tx_mc' and the reception state machine called 'rx_mc'. The ' tx_mc' has four possible states-TX-FREE when it has no job on hand to be transmitted;IDLE when it has completed a job and is deciding whether any other transmission request is pending. XMIT when it is in the process of transmitting a message;and AWAIT_RESP when it is waiting to verify the status of the last message sent.
The 'rx_mc' consists of seven states. State RXM_FREE is when no byte is being received. DEL_RXD_FREE is the state when a first DEL byte,which signifies a message to command changeover,is received and the machine is ready to read the next byte and make sense of the command. Similarly,DEL_RXD_ASMBL is the state when during the reception of a message a DEL byte is encountered and the machine is ready to identify the next command byte. The ASMBL state is when the machine,having received the starting signals of a message,is busy assembling the message byte by byte as they arrive. States GET_SRC_ID and GET_DES_ID ,as is obvious from the nomenclature,are when the state machine is waiting to read either the source or the destination id of the message. When it is in this state,the next byte that is received is deemed to be the source or the destination id. Finally the GET_BCC state deems the next received byte to be the binary checksum of the message. In this state,the received checksum is compared with the one calculated using the received message to ascertain if there has been faulty reception. In case of a fault,appropriate action is taken.

All messages start with two bytes DLE and STX. After these two bytes three bytes making up the header of the message containing source-id,destination-id and length of message is sent. This is followed by the actual message which can be a maximum of 255 bytes long. The tail of the message is made up of three bytes,DLE,ETX and the binary checksum value of the message. For every message that is sent,the sender expects a response from the receiver indicating whether the message was received in good form. If the receiver replies in the affirmative,then that message is considered serviced. Instead,if the receiver send a negative response(NAK),the message is retransmitted. There can be a maximum of three transmissions of a message and after which attempt to transmit that message is discarded. However,if the receiver does not send any response and a timeout occurs,the transmitting party sends an enquiry byte(ENQ) to the receiver to apprise it of its ignorance about the fate of the message. In case a timeout occurs again, enquiry byte is send again.A maximum of three enquiry bytes are send. If the response is still not available,the attempt is aborted.

SYSTEM ENGINEERING

2.SYSTEM ENGINEERING

System analysis is the study of an application for the purpose of understanding its essential features. It is characterized by obtaining information about the application. The system analysis process produces a description in terms of the requirements and objectives of the software. The result of system analysis comes as a graphical or textual,informal or formal,abstract model of the application. The purpose of an abstract model is to provide a description of the application. A top-down fashion for software development generally creates a system by successive refinements of software components. If the system is not trivially simple,it should be decomposed into large parts which may be further decomposed. Domain analysis plays a fundamental role in identifying potentially reusable components within an application domain. After identifying the information domain,making the feasibility studies and finally evaluating alternative design solutions until a preferred solution emerges before the system engineer.

2.1 IDENTIFICATION OF NEEDS

Since the analysis of the entire software is a major task,parts of the software,which make sense individually for analysis purpose,were evaluated. The criteria for selection of a part was that it should be individually executable,it should be an implementation of one or more finite state machines. As an example,procedure tx_mc can make up one sample to be analyzed. It can be individually compiled and contains a complete model of a finite state machine. Software updation is needed for better maintenance of the antenna systems and also for satisfying the requirements of scientists.

2.1.1EXISTING SYSTEM

The existing system is in PASCAL. It is very old language. Because of this, existing system has very difficult to update,very expensive for handling software,also it is very time-consuming .Also existing system suffers so many software bugs or errors.

2.1.2 PROPOSED SYSTEM

Study on Proposed System

Main advantage of the proposed system is

a. The program is self-documenting

b. More easily understood for testing,maintenance and future modification

c. Portability

d. Model provability(credibility)

2.2 FEASIBILITY STUDY

A feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that is spend on it. Feasibility study lets the developer foresee the future and its usefulness.

Feasibility study is a test of system proposed regarding its workability, impact on the organization, ability to meet the needs and effective use of resources. Thus when a new project is proposed, it normally goes through a feasibility study before it is approved for development.

This document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as:

Economical Feasibility

Technical Feasibility

Legal Feasibility

2.2.1 Economical Feasibility

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project. One of the factors,which affect the development of a new system , is the cost it would require. Since the system is developed as a part of project work , there is no manual cost to spend for the proposed system . Also if all of the resources are already available,it gives an indication that the system is economically possible for development. But since we are not going to market the software there is little concern about monetary benefits .

2.2.2 Technical Feasibility

Technical feasibility centers on the organization to what extend it can support the proposed system . The question to be answered is whether the organization is technically sound to operate the system. The necessary hardware and software must be installed in the organization. Again considering the company resources system is presently having sufficient software and hardware support. Though the technology may become obsolete after some period of time .Due to the fact that newer version of some software's support older version , the system may still be used .

2.2.3 Legal Feasibility

The developed system if it is to be marketed must be free of legal problems . If the system is not developed with the licensed versions of hardware and software , then that will be a breach of copyright law .

2.2.4 Alternatives

The selection of hardware and software tools for the system is an important aspect in the system development. Here the software is developed as Microsoft Windows 2000 Server platform because it possesses a large no of features that are suitable to attain the objective.

C programming is an approach to improve software quality through the use of elemental programming constructs. One of the advantage of programming is to reduce errors by reducing inter-segment and inter-construct branching. The editor Turbo-C is selected in this software development because of its performance enhancement, world class tool support, power and flexibility, simplicity, manageability, scalability, availability, customizability, and extensibility and security.

Also it take advantage of top-down development design

1.Coding can proceed without dependency on lower-level coding,

2.Interface assumptions are handled down from higher levels.

3.Testing can be done in actual environment,

4.No lengthy,end of project,integration phase,

5.A more manageable project,and

6.Earlier availability of a running program.

2.4 SYSTEM DEFINITION

The link layer incorporates **asynchronous transmission protocol** that decides how messages are sent .The messages exchanged between the two station can be classified into message codes which are send from the transmitter to the receiver,and response codes which are send from the receiver to the transmitter. DLE NAK and DLE ACK are the response codes whereas DLE STX,DLE ETX BCC,DLE ENQ and data codes are all message codes. Use of these control characters in message frame flow between two machines in full duplex mode.

Link layer message packets

A link layer message packet starts with a DLE STX, ends with a DLE ETX BCC,and includes all link layer data codes in between. Data codes can occur only inside a message packet. Response codes can also occur between a DLE STX and a DLE ETX BCC,but these response codes are not part of the message packet,they are called embedded responses.

DLE STX<-----HEADER+DATA(from application layer)-------> DLE ETX BCC

The HEADER is 3 bytes long arranged in the following order. Destination_id , source_id followed by the length of the data string.

BCC is two's complement of the sum of the all data bytes between DLE STX and DLE ETX BCC. It doesn't include any control codes.

objective

The main goal of the project is to make a Link Protocol in C language ,the protocol used for the communication between Station Servo Computer(SSC) and Antenna Base Computer(ABC) is a RS-422-C full duplex link protocol with acknowledgment,error detection and retry mechanisms built-in.It also provides task to task communication between the two stations.

This is a character oriented protocol using the following ASCII characters extended to 8 bits by adding a zero for bit 7.

Different modules are

1.The procedure **get_free_buff** returns TRUE and a pointer to the free buffer if available otherwise returns FALSE.

2.The procedure **put_free_buff** puts a user specified buffer passed as a pointer to link's free pool.

3.The procedure **put_in_req** puts a user specified buffer passed as a pointer to link's request queue.

4.The procedure **get_from_indq** returns TRUE and a pointer to a buffer provided a frame is received for the indicated task_id otherwise returns FALSE.

5.The procedure **get_from_confq** returns TRUE and a pointer to a buffer if link confirmation for the last transmission requests by the indicated task_id is ready (it returns the confirmation to the oldest frame that is not yet collected) otherwise returns FALSE.

6.The procedure **tx_mc** consist of the **tx_mc** has four possible states-**TX-FREE** when it has no job on hand to be transmitted;**IDLE** when it has completed a job and is deciding whether any other transmission request is pending. **XMIT** when it is in the process of transmitting a message;and **AWAIT_RESP** when it is waiting to verify the status of the last message sent.

7.The procedure **rx_mc** consists of seven states. State **RXM_FREE** is when no byte is being received. **DEL_RXD_FREE** is the state when a first **DEL** byte,which signifies a message to command changeover,is received and the machine is ready to read the next byte and make sense of the command. Similarly,**DEL_RXD_ASMBL** is the state when during the reception of a message a **DEL** byte is encountered and the machine is ready to identify the next command byte. The **ASMBL** state is when the machine,having received the starting signals of a message,is busy assembling the message byte by byte as they arrive. States **GET_SRC_ID** and **GET_DES_ID** ,as is obvious from the nomenclature,are when the state machine is waiting to read either the source or the destination id of the message. When it is in this state,the next byte that is received is deemed to be the source or the destination id. Finally the **GET_BCC** state deems the next received byte to be the binary checksum of the message. In this state,the received checksum is compared with the one calculated using the received message to ascertain if there has been faulty reception. In case of a fault,appropriate action is taken.

8.The procedure **compile** takes data from input device and convert this data into message packet starts with a **DLE STX**,ends with a **DLE ETX BCC**,and includes all link layer data codes in between. Data codes can occur only inside a message packet.

9.The procedure **tx_rut** takes message packet and write to the output port.

10. The procedure **start_tmr** count the timing of the transmission.

SYSTEM SPECIFICATION

PC configuration

Processor	- Pentium IV
Hard disk	- 40GB
RAM	- 256MB
Keyboard	-Standard Keyboard with 105 keys
Mouse	-2 button mouse
Monitor	-15 inch color monitor

Software Configuration

Operating System-Windows XP

Editor - Turbo C

Compiler - C compiler

REQUIREMENT ANALYSIS

3. REQUIREMENT ANALYSIS

The job of software requirements analysis is to understand the specific requirements that must be achieved to build the high quality software. In the development of this site the mode of elicitation of needs is the traditional interview. In this way requirements of the system to be developed can be gathered. Requirement analysis enables to specify the software function and performance indicate software interface with other system elements and establish constraints that software must meet . It also provides us with the models that can be translated into data,architectural,interface and procedural design.

3.1IDENTIFICATION OF INFORMATION DOMAIN

It is important to property identify and gather information required for the development of the system . This whole valid information comes under the information domain .

Main procedures of ASYNCHRONOUS COMMUNICATION PROTOCOL SYSTEM are

1.get_free_buf

2.put_free_buff

3.put_in_req

4.get_from_indq

5.get_from_confq

6.tx_mc

7.tx_rut

8.compile

9.rx_mc

10.start_tmr

3.2 MODELING

Modeling gives a detailed description of what are the functions that each component performs. We create functional models to gain a better understanding of the actual entity to be built. It must be capable of modeling the information that software transforms, the functions and sub-functions that enable the transformation to occur, and the behavior of the system as the transformation is taking place. Models focus on what the system must do, not on how it does it.

Models help in understanding the information, function and system behavior making requirement analysis task easier , models becomes the focal point for review and becomes the foundation for design.

3.2.1DATA FLOW DIAGRAM

The graphical description of the system's data and how the processes transform the data is known as data flow diagram . A graphical picture of the logical steps and sequence involved in a procedure or a program is called a flow chart. Unlike detailed flowchart , data flow diagram do not supply detailed description of the modules but graphically describes a system's data and how the data interact with the system .

Context Analysis Diagram(Zero Level DFD) is the top-level diagram. It represents the entire software element as a single bubble with input and output, data indicated by incoming and outgoing arrows.

To construct a data flow diagram , we use : Arrow, Circles , Open End Box and Squares

Arrow –: An arrow identifies the data flow in motion . It is a pipeline through which information is flown like the rectangle in the flow chart.

Circles -: Circles stands for process that converts data into information .

Open End Box -: An open end box represents a data store , data at rest or temporary reposition of data

Square -: A square identifies a source or destination of system data .

Six rules for constructing DFD :

1.Arrows should not cross each other .

2.Squares, circles and files must bear names .

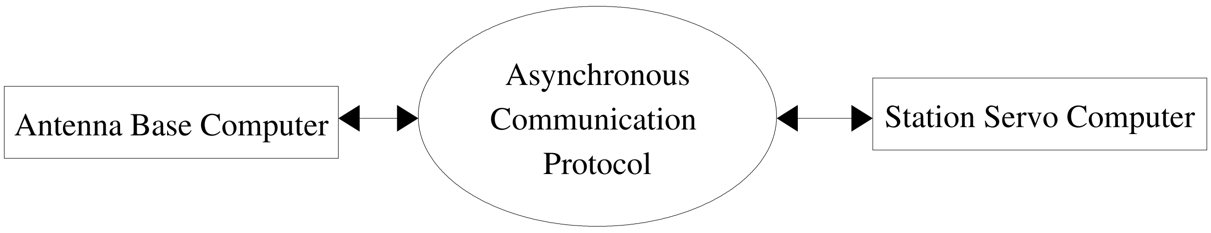
3.Decomposed data flow squares and circles can have same names .

4.Choose meaningful names for data flow .

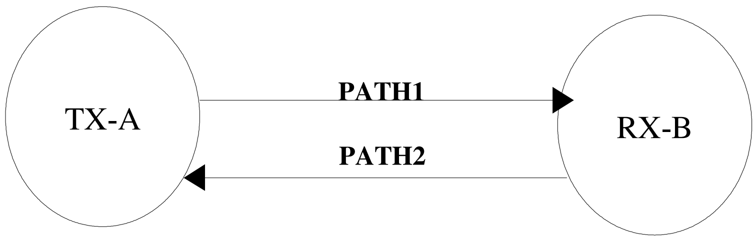
5.Draw all data flows around the outside of the diagram .

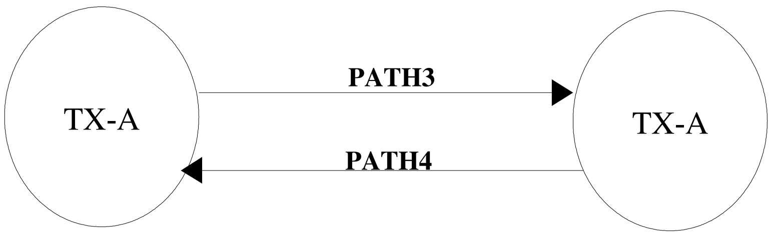
6.Control information such as record count , passwords and validation requirement are not pertinent to data flow diagram

CONTEXT ANALYSIS DIAGRAMS



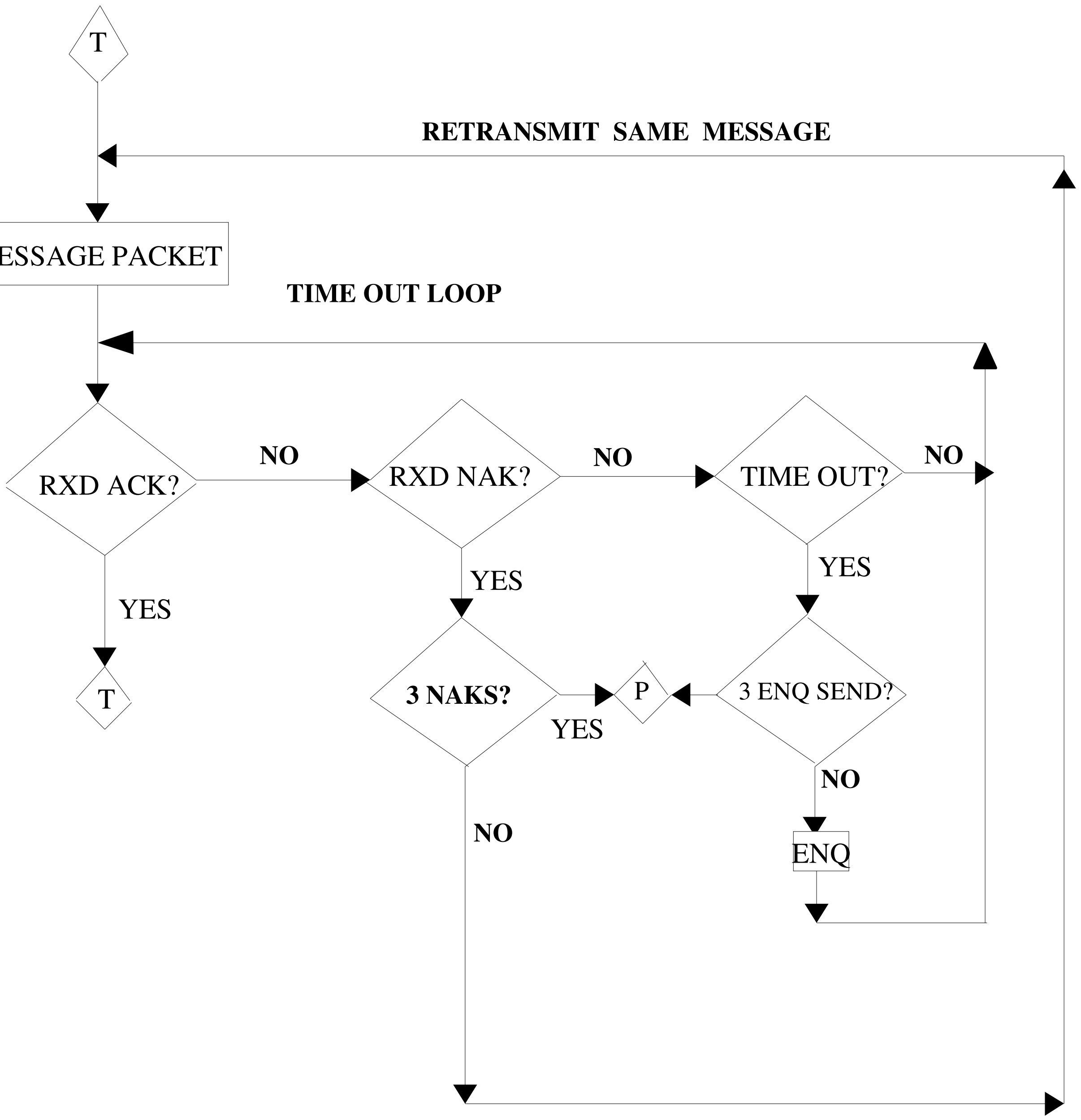
MESSAGE FLOW



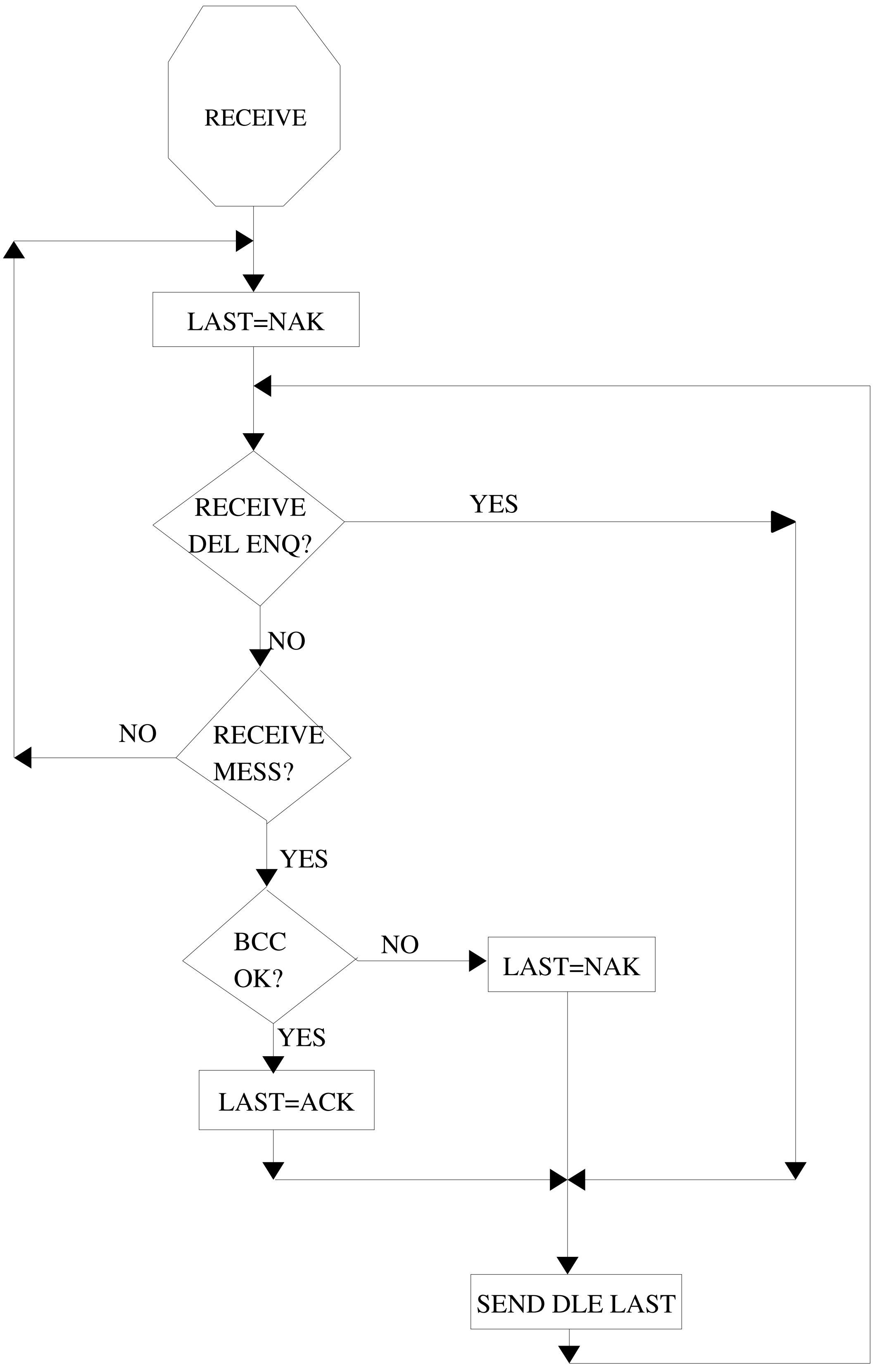


FLOW CHARTS

TRANSMITER STATE DIAGRAM



RECEIVER STATE DIAGRAM



DATA DICTIONARY

Function name	***Description***
get_free_buff	Check availability free buffer in buffer pool.
put_free_buff	Put given data into a free buffer .
put_in_req	Puts a user specified buffer passed as a pointer to link's request queue.
get_from_indq	Check the buffer provided a frame is received for the indicated taskid.
get_from_confq	check a pointer to a buffer if link conformation for the last transmission requests by the indicated task_id is ready
tx_mc	Transmission routine
rx_mc	Reception routine
tx_rut	Routine that data write to output port.
compile	Packet making routine
start_tmr	Check the time for transmission and reception responses
TXM_FREE	Transmission free state
IDLE	State reveals transmission settings become idle
XMIT	State in which transmission takes place
AWAIT_RESP	State in which waiting for response
GET_SRC_ID	State in which getting source id of the package
GET_DES_ID	State in which getting destination id of the package
ASSEMBEL	Assembling state of each bit into a frame
GET_BCC	Getting bcc of the transmission package

SYSTEM DESIGN

4 SYSTEM DESIGN

Structured programming,top down development,modular design and others are program development techniques that have obvious and reported benefits in improving the quality of software products.The aim of these methodologies is to structure the design,programming,and testing processes software development so that the final product will be
improved through such criteria as reliability,maintainability,generality,and performance.

Design models used for simulating computer systems have their own unique set of problems because of their intended purpose in representing a system description . Model development is not dissimilar to program development .

Program Design Analysis is the most demanding on representation. Other simulation applications have more to gain in using new development techniques because of less restriction on representation.

Models are software programs written in a simulation language and run on a computer,they must necessarily go through the familiar stages of design,design view,coding,parameter testing,and integration. Though the development of a model will involve much less effort over much less time than the development of the system being modeled. Model development must necessarily include coordinated efforts for several independent development processes. Also,models are often developed with reusability in mind;aspects of application independent design must therefore be considered in model development just as in system software development.

The processes involved in the system design are :

Study and analyze the existing manual procedures .

Modify and redefine those procedures to achieve precise and logical patterns and to establish computer input and output design .

Present the redefined system in a manner suitable for a translation into a computer system .

In system design high end decisions are taken regarding the basic system architecture , platforms and tools to be used the system design transforms a logical representation of a what a given system is required to be in the physical specification . Design starts with the system requirement specification and convert it into a physical reality during the development . Important design factors such as reliability , response time , throughput of the system , maintainability expandability etc should be taken into account .

Fundamental Design Concept

The fundamental design concept provide the necessary framework for “getting it right” . The fundamental design concepts such as abstraction , refinement , modularity , software architecture , control hierarchy , structural partitioning , data structure , software procedure and information hiding are applied in this project to getting it right as per the specification .

4.1DATA DESIGN

Database design is recognized as a standard of management information system and is available virtually for every computer system .

Software requires collection of tasks. These tasks need to communicate with one another. This is enabled to an extent by the members in the interface section of the unit implementing the task. The tasks located in various units need to pass messages between them. However,because of the fact that a variable loses the value contained in it once it is out of its declared scope,there are chances that the message contained in a variable in one unit is lost when accessed from another unit. To overcome this,all data items which serve to carry messages are declared global.

Some of the notable message-passing variables are status of timers,command bytes,response to a command,system mode and the status of the various state machines. This design around inter-process message passing technique brings the software one step closer to a real object-oriented architecture.

Header files

dos.h, stdlib.h, stdio.h, conio.h

Data Types

int,char,enum,typedef

Needed Data Structures

I. Structure

A structure provides a means of grouping variables under a single name for easier handling and identification. Complex hierarchies can be created by nesting structures. Structures may be copied to and assigned. They are also useful in passing groups of logically related data into functions .In this software has mainly 3 structures

1 struct frameq
It consist of two structure elements.

1a.com_buf_ptr hd:This points to the header field of the linked list.

1b.com_buf_ptr tl :This points to the tail of the linked list

II struct com_buf_type
It cosists of five structure elements.

IIa. data array   -Specifies data part of the frame.

IIb. length   -Specifies length of the data.
IIc. src_task_id -Specifies source task id.

IId. des_task_id -Specifies destination task id.

IIe. com_buf_typ *next-Pointer variable which points to the next linked list.
III struct ack_timer_type
It cosists of two structure variables.

IIIa. counts -Specifes the time count.
IIIb. stt   -Specifies the timer settings.

2. Arrays

Arrays are a data structure that is used to store a group of objects of the same type sequentially in memory. All the elements of an array must be the same data type, for example float, char, int, pointer to float, pointer to int, a structure or function. The elements of an array are stored sequentially in memory. This allows convenient and powerful manipulation of array elements using pointers. Mainly used arrays are

2a.received character array,rxd_char,of size 256.

2b.Transmitted character array,txd_char,of size 513.

3.Circular Buffers

These are basically of the array type but deserve a separate mention due to their important and extensive use. There is an RS232 serial communication link for communication between remote computers and the antenna base computer. Each buffer record can hold a message in an array,which is received one by one. For transmitting,the buffer holds the message compiled to add the start,stop,checksum and header bytes.

4.Queue

These are data structures in which access is in first-in-first-out sequence. The data item that joins the queue earliest is put at the head of the queue. This is because the queue is an implementation of a linked list. If the queue is a linear queue,the last member or the tail points to a null location. In this software ,for every task there are thee queues;one for the send request,another or the confirmation status of messages send,and a third for received messages.

Preprocessor Declarations

Variable	Hexa-DecimalValues
OK	0x00
NORESP	0x0F
ONLYNAK	0x11
DEL	0x10
ETX	0x03
STX	0x02
ENQ	0x05
ACK	0x06
NAK	0x15
REST	0x20

User defined Data type

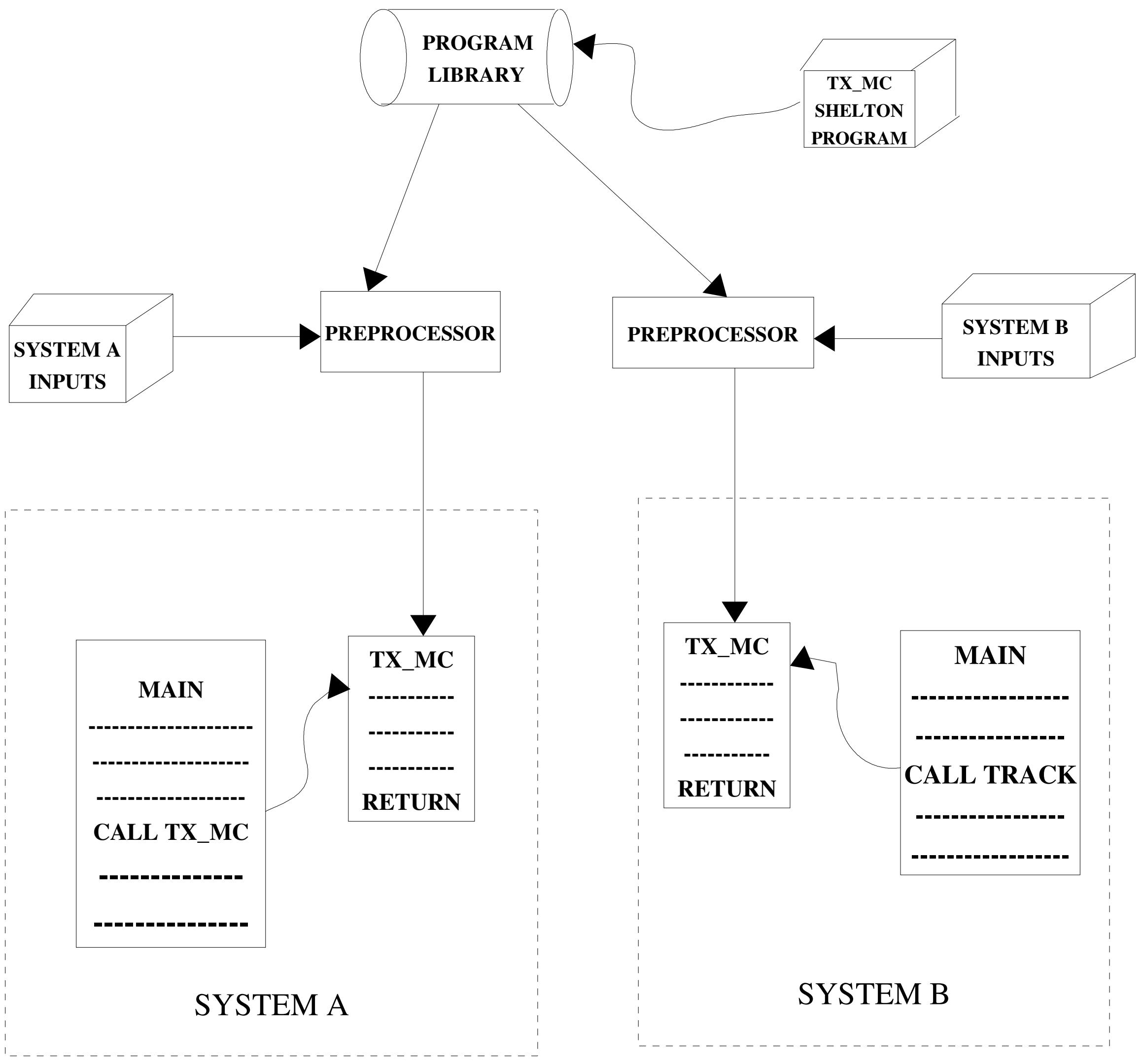
Data Type	Variable name	aliases
enum	txm_stts	TXM_FREE, IDLE, XMIT, AWAIT_RESP
enum	msg_stts	FREE, PENDING, DELX
enum	rxm_stts	RXM_FREE, ASSEMBEL, GET_SRC_ID , GET_DES_ID
enum	tmr_stts	ON,OFF,TIME_OUT

ARCHITECTURAL DESIGN

Top-down development is a methodology which structures the design processes of program development .The program is developed as “levels” of design in which modules are designed ,tested,and integrated before proceeding to the next lower design level. A program is a tree structure whose leaves are unique modules which implement the design requirements. The program is developed in a to-down manner by designing,testing and integrating each level from the top-most element in such a way that design decisions,as to how the requirements are implemented,are postponed to as low a level as possible. The postponing of design decisions is aided with the use of module stubs which serve only as a point of control acceptance,possible time consumption,and control return;these stubs become the top-most elements when the current level has been coded,tested,integrated,and verified correct.

Process construction is a development methodology in which commonly used program modules are saved in application-independent,skeletal form
and which,when required for a particular application,are preprocessed into an application-dependent,executable form.

Fig shows general architectural format of top-down design.



A PROCESS CONSTRUCTION SYSTEM

The skeleton library program is processed with application-dependent inputs to produce a reusable subroutine

Software architecture provides a holistic view of the system to be built. It depicts the structure and organization of software components, their properties and the connections between them .In this project modules are communicating on the basis of message passing techniques.

4.3 INTERFACE DESIGN

The interface is a packaging for computer software if the interface is easy to learn, simple to use, straight forward and forgiving, the user will be inclined to make good use of what is inside. If the interface has none of these characteristics, problem will invariably arise. If the interface is very good, the user will fall into an interactive software application.

Input Design

The input design is the process of converting the user-oriented inputs in to the computer based format . The data is fed into the system using simple interactive forms .Input design involves determining the record media, method of input , speed of capture and entry to the system.

In this software envoirnment ,data is taken from input and then it is converted into packets by compile routine and write to input port of the transmitter buffer by procedure tx_mc.

Input design involves determining the record media, method of input , speed of capture and entry to the system.

Input configuration

input port - port1

Address of input port – 0x2f8

Software configuration of port1

outportb(port1+1,0) – Turn off interrupts

outportb(port1+3,0x80)-Set DLAB on

outportb(port1+1,0x60)-Set baud rate

outportb(port1+3,0x03)-8 bits,No parity,1 Stop bit

outportb(port1+2,0xc7)-FIFO control register.

outportb(port1+4,0x0b)-Trun on DTR,RTS and OUT2.

outportb(0x21,(inportb(0x21)&0xef)-Set programmable interrupt controller.

outportb(port1+1,0x01)-Interrupt when data received.

outportb(port1+1,0)-Reset IRQ4.

outportb(0x21,(inportb(0x21)|0x10))-Masking(Disabling) IRQ4.

Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any systems results of processing are communicated to the user and to other systems through outputs. In the output design it is determined how the information is to be displayed for immediate need and also the hard copy output. It is the most important and direct source of information to the user. Efficient and intelligent output design improves the system's relationship with the user and helps in decision making .The output generally refers to the results and information that is generated from the system.

Output configuration

Output port-port2

Address of output port – 0x3f8

Software configuration of port2

outportb(port2+1,0) – Turn off interrupts

outportb(port2+3,0x80)-Set DLAB on

outportb(port2+1,0x60)-Set baud rate

outportb(port2+3,0x03)-8 bits,No parity,1 Stop bit

outportb(port2+2,0xc7)-FIFO control register.

outportb(port2+4,0x0b)-Trun on DTR,RTS and OUT2.

outportb(0x21,(inportb(0x21)&0xef)-Set programmable interrupt controller.

outportb(port2+1,0x01)-Interrupt when data received.

outportb(port2+1,0)-Reset IRQ4.

outportb(0x21,(inportb(0x21)| 0x10))-Masking(Disabling) IRQ4.

4.4 PROCEDURAL DESIGN

Procedural design occurs after data and program structure have been established . In an ideal world , the procedural specification required to define algorithmic details should be stated in natural language . Since the language used is common the people outside the software domain could also readily understand the specification , and no new learning would be required . Procedural design must specify procedural details unambiguously , and lack of ambiguity in a natural language is not natural . Procedures specify what tasks must be performed in using the system and who is responsible for carrying them out .

Psudo code of tx_mc

```
procedure tx_mc(var  inv_code:bye);
begin
{
done=false;
repeat if (! done) then
case txm_stt if
TXM_FREE: txm_stt=IDLE;
                    done=TRUE;
IDLE:   ack_retry_count=2;
          enq_retry_count=2;
```

```
            txm_stt=XMIT;
            compile();
XMIT:  txm_stt=AWAIT_RESP;
            start_tmr();
            tx_data_stt=PENDING;
            done=true;
            tx_rut();
AWAIT_RESP: if(inv_code==2)
           {
          if(rx_resp_byte==ACK)
           {
           ack_timer.stt=OFF;
           txm_stt=IDLE;
            }
           else
             {
             if(ack_retry_count!=0)
             {
             ack_retry_count--;
              txm_stt=XMIT;
                }
            }
            }
else if (inv_code ==3)
{
ack_timer.stt=OFF;
if(enq_retry_count !=0)
{
enq_retry_count--;
enq_tx_stt=PENDING;
if(txr_stt==FREE)
tx_rut();
start_tmr();
done=TRUE;
}
}
}
}
```

Psudo code of rx_mc

```
procedure  rx_mc
begin
```

```
{
var cx:real;
cx=inportb(port1);
repeat if(rxd_char_hd!=rxd_char_tl)
{
rxd_char_hd--;
if(rxm_stt==GET_SRC_ID)
{
rx_buff->src_task_id=cx;
rxm_stt=GET_DES_ID;
}
else if(rxm_stt==GET_DES_ID)
{
rx_buff->des_tas_id=cx;
rxm_stt=ASSEMBEL;
}
else if(rxm_stt==GET_BCC)
{
rx_bcc=rx_bcc+cx;
if(rx_bcc!=0)
{
```

```
last_resp=NAK;
else
{
last_resp=ACK;
}
res_tx_stt=PENDING;
if(txr_stt==FREE)
tx_rut();
rxm_stt=RXM_FREE;
}
}
else
{
 NAK:tx_mc(2);
STX:rxm_stt=GET_SRC_ID;
            last_resp=NAK;
ETX:rxm_stt=GET_BCC;
}
else if(rxm_stt ==ASSEMBEL)
{
rx_bcc=rx_bcc_cx;
```

```
last_resp=NAK;
}
}
}
}
```

TESTING

5. TESTING

Testing is a process of determining whether or not the module under development satisfies the requirements specify in the previous phases .

The aim of testing often is to demonstrate that a program works by showing that it has no errors . The basic purpose of testing phase is to detect the errors that may be present in the program . So the intent of testing should not be to show that the program works : but show that the program does not work . Therefore the definition of testing given is: testing is a process of executing a program with the intention of finding errors .

The testing strategies applied are :

1. Unit testing
2. Integration testing

5.1 UNIT TESTING

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

In this type of testing, the different modules are tested against the specifications produced during design for the module. Unit testing is essentially for the verification of the code produced during the coding face, and hence the goal is to test the internal logic of the module. The programmer of the module typically does it since it is done with the code of the module; structural testing is most suited to it. In project each modules are taken out seperately and compile and run seperately using C compiler and debugger.

The unit test is white-box oriented, and the step can be conducted in parallel for multiple components.

Black-box Testing

This testing technique is used to uncover the structural errors in the software . Although they are designed to uncover errors , black-box tests are used to demonstrate that software functions are operational , that input is properly accepted and output is correctly produced, and that the integrity of external information is maintained . A black-box test examines some fundamental aspect of a system with little regard for the internal logical structure of the software . In this software
given the input into a data format and check whether the given data is converted into packet also if it is transmitting correctly also if it is received by the other computer.

White-box Testing

White-box testing , sometimes called glass-box testing is a test case design method that uses the control structure of the procedural design to derive test cases . Using white-box testing methods , the software engineer can derive test cases that (1) guarantee that all independent paths within a module have been exercised at least once , (2) exercise all logical decisions on their true and false sides , (3) execute all loops at their boundaries and within their operational bounds , and (4) exercise internal data structures to ensure their validity.

This testing technique is used to uncover the logical errors in the program. For this the basis path testing method is used. Cyclomatic complexity of a small module is determined and keeping it as the upper bound the basis set of linearly independent paths are determined. Test cases are prepared and is executed on these paths. Thus every logical path in the module is executed and tested using the test cases. Thus all the modules in the project are unit tested.

Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

In this testing , many unit tested modules are combined in to subsystems and are then tested . The forming of subsystems are depend on the modules . The goal here is to see if the module can be integrated properly between the modules. Two type of Integration Testing are Top down Integration and Bottom up integration.

5.4VARIFICATION

Verification refers to the set of activities that ensure that software correctly implements a specific function. Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements. Verification and validation encompasses a wide array of software quality assurance activities that include formal technical reviews, quality and configuration audits, performance monitoring, simulation, feasibility study and documentation review, database review etc.

IMPLEMENTATION AND MAINTENANCE

6 IMPLEMENTATION AND MAINTENANCE

Implementation is one of the most important tasks in a project. Implementation is the phase,in which one has to be cautious, because all the efforts undertaken during this project will be fruitful only if the software is properly implemented according to the plans made.

Implementation is the stage in the project where the theoretical design is turned into a working system . The crucial stage is achieving successful new system and giving the users confidence in that the system will work effectively and efficiently. It involves careful planning, investigation of the current system and its constraints on implementation and design of methods to achieve changeover . Apart from these , the major task of preparing for implementation are education and training of users and system testing .

Post Implementation

Operational systems are quickly taken for granted . Every system requires periodic evaluation after implementation . A post implementation review measures the systems performance against predefined requirements . Unlike system testing , which determines where the system fails so that the necessary adjustments can be made , a post implementation review determines how well the system continues to meet performance specifications . It is after the fact-after design and conversion are complete . It also provides information to determine whether major redesign is necessary.

A post implementation review is an evaluation of a system in terms of the extend to which the system accomplishes stated objectives and actual project cost exceed initial estimates . It is usually a review of major problems that need converting and those that surfaced during the implementation phase .

User Training

An analysis of user training focuses on two factors:

. User capabilities

. The nature of the system

Users range from the native to the highly sophisticated. Developmental researches provide interesting in sights into how native computer users think about their first exposure to a new system. They approach it as concrete learners, learning how to use the system without trying to understand which abstract principles determine which function.

Task that require the user to follow a well defined, concrete, a step by step procedure require limited problem solving. This means that the training level and duration are basic and brief. Virtually all the thinking is performed by the computer. In contrast, require a training to analyze a given situation and translate it into a procedure for computer manipulation requires formal training for relatively long time.

SYSTEM MAINTENANCE

The maintenance phase of the software cycle is the time in which a Software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software product enhancements may involve providing new functional capabilities, improving user displays and mode of interaction, upgrading the performance characteristics of the system. So only through proper system maintenance procedures, the system can be adapted to cope up with these changes.

The first maintenance activity occurs because it is unreasonable to assume that software testing will uncover all latent errors in a large software system. The process that includes the diagnosis and correction of one or more errors is called corrective maintenance. The second activity that contributes to a definition of maintenance occurs because of the rapid change that is encountered in every aspect of computing. Therefore, adaptive maintenance- an activity that modifies software to properly interfere with a changing environment is necessary.

Out put of the software

1.Establish Communication between ABC and SSC through Serial Port by porting the PASCAL Source Code . This is done for easy maintenance and upgrade the future as the need may arise. This code does not have any OS (Standalone).

7 FUTURE ENHANCEMENT

2.Similar to step 1 part of the codes of other modules namely
 (a) Antenna Control
 (b) Encoder section
 (c)Data Acquisition
 (d)Compensator

3.Integrate all the modules and test the entire code.

8 APPENDIX

APPENDIX 1

Hardware Description of the Project

Asynchronous transmission allows data to be transmitted without the sender having to send a clock signal to the receiver. Instead, the sender and receiver must agree on timing parameters in advance and special bits are added to each word which are used to synchronize the sending and receiving units.

The concept of serial communication is ,the serial port sends and receives bytes of information one bit at a time. Advantage of serial communication is ,it is very simple,and can use it over longer distances about 1200m between any two devices. The important serial characteristics are baud rate,data bits,start bit,stop bits and parity. For 2 ports to communicate,these parameters must match.

Baud rate is a speed measurement for communication that indicates the number of bits per second which is similar to clock rate ,that is 4800 baudrate .The clock is running at 4800hz.In this software environment set the baud rate as 9600.

Start bit are used to indicate start of packet for communication. In this software for indication of start bit,hexa decimal value 0x02 is used.

Stop bits are used to signal the end of communication for a single packet. In this software 0x03 is used for end of transmission.

Parity is the simple form of error checking used in serial communication. In this software checksum calculation is done by adding all data bits,after that takes 2's complement of added values. The variable bcc contains all the added data values,then take 2's complement,by adding 1 to bcc and finally this added value is subtracted from 0xff.

Communication between two PC's are established using RS422 cable connection.RS422 is the serial connection found on IBM-compatible PC's.RS422 hardware permits communication at distances up to 50ft. RS422 id limited to point-to-point connections between PC serial ports and devices. There are two basic standards: one for transmitting equipment (DTE) and other for receiving equipment (DCE).Port is a term used to refer to any one of the possibly several communications devices available to the operating system.

PIN CONFIGURATION OF RS 232

1	DCD	------------	DCD	1
2	TxD	------------	RxD	3
3	RxD	------------	TxD	2
4	DTR	------------	DSR	6
5	Gnd	------------	Gnd	5
6	DSR	------------	DTR	4
7	RTS	------------	CTS	8
8	CTS	------------	RTS	79
9	RI	--------------	RI	9

PIN DESCRIPTION

PIN NO	DESCRIPTION
1	Data Carrier Detect
2	Received Data
3	Transmitted Data
4	Data Terminal Ready
5	Protective Ground
6	Data Set Ready
7	Request to send/Ready to receive
8	Clear to Send
9	Ring Indicator

Baud Rate

In addition to voltage ranges, RS-422 Protocol identifies how fast data is transmitted between two ports. This transmission speed is defined as Baud Rate roughly equivalent to the number of bits transmitted per second. Typically, the baud rate will vary between 1200 to 19200. Common baud rates are as follows: 1200, 2400, 4800, 9600, and 19200.

Cable Length Limitations

The length of the interconnecting cable between a control system and controlled equipment must decrease as the baud rate increases. Typically,at 1200 to 2400 baud, the cable length should be no more than approximately 100 feet. At 9600 baud, the distance should be no more than 50 feet and, at 19200 baud, the cable length should be no more than 20 feet.

Troubleshooting

Problems with RS-422 connections can typically be separated into two groups: those that are caused by hardware or physical hook-up conflicts and those that are caused by software conflicts.

Hardware

The majority of control systems require the connection of only two wires to the controlled device. The Transmit (XMT) and Ground (GND) pins on the control system are connected to the Receive (RCV) and Ground (GND) pins on the controlled device respectively, as shown below (Fig. 1).

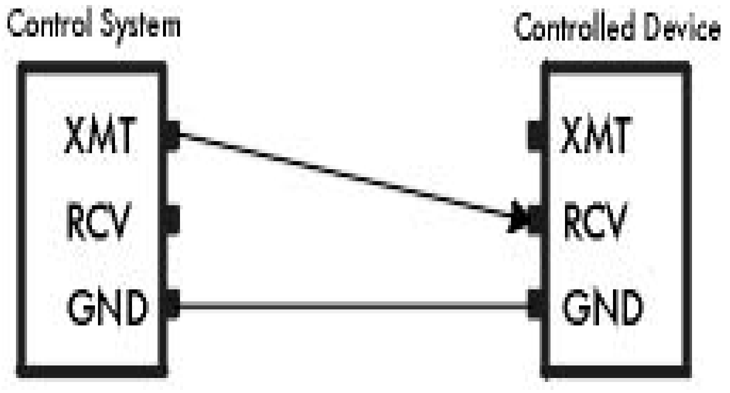


Figure 1

In situations in which the control system needs to receive some type of response from the controlled device, a third wire will also be connected (Fig. 2). When using a computer to control a device through its COM port,for instance, this is the recommended wiring.

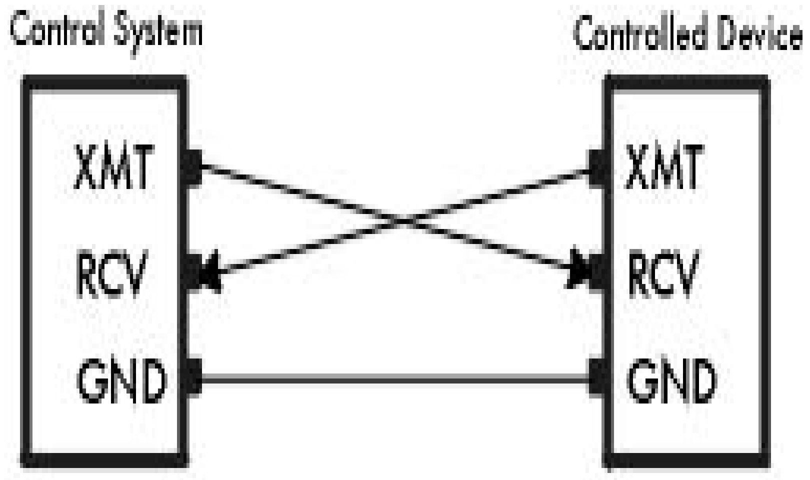


Figure 2

RS422 functions are sharply divided into data functions and control functions. The data functions are quite simply transmitter on pin2 and receiver pin3.All the remaining functions are control functions. Control functions of RS422 interface are controlled through hardware. Commonly used asynchronous functions are built into single controller IC known as UART (Universal Asynchronous Receiver/Transmitter).

UART

The Universal Asynchronous Receiver/Transmitter (UART) controller is the key component of the serial communications subsystem of a computer. The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes.

When a word is given to the UART for Asynchronous transmissions, a bit called the "Start Bit" is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter.

After the Start Bit, the individual bits of the word of data are sent, with the Least Significant Bit (LSB) being sent first. Each bit in the transmission is transmitted for exactly the same amount of time as all of the other bits, and the receiver “looks” at the wire at approximately halfway through the period assigned to each bit to determine if the bit is a 1 or a 0. For example, if it takes two seconds to send each bit, the receiver will examine the signal to determine if it is a 1 or a 0 after one second has passed, then it will wait two seconds and then examine the value of the next bit, and so on.

The sender does not know when the receiver has “looked” at the value of the bit. The sender only knows when the clock says to begin transmitting the next bit of the word.

When the entire data word has been sent, the transmitter may add a Parity Bit that the transmitter generates. The Parity Bit may be used by the receiver to perform simple error checking. Then at least one Stop Bit is sent by the transmitter.

When the receiver has received all of the bits in the data word, it may check for the Parity Bits (both sender and receiver must agree on whether a Parity Bit is to be used), and then the receiver looks for a Stop Bit. If the Stop Bit does not appear when it is supposed to, the UART considers the entire word to be garbled and will report a Framing Error to the host processor when the data word is read. The usual cause of a Framing Error is that the sender and receiver clocks were not running at the same speed, or that the signal was interrupted.

Regardless of whether the data was received correctly or not, the UART automatically discards the Start, Parity and Stop bits. If the sender and receiver are configured identically, these bits are not passed to the host.

If another word is ready for transmission, the Start Bit for the new word can be sent as soon as the Stop Bit for the previous word has been sent.

Because asynchronous data is “self synchronizing”, if there is no data to transmit, the transmission line can be idle.

When a word is given to the UART for Asynchronous transmissions,When two devices that are both DTE or both DCE must be connected together without a modem or a similar media translator between them, a NULL modem must be used. The NULL modem electrically re-arranges the cabling so that the transmitter output is connected to the receiver input on the other device, and vice versa.

The 8250/16450/16550 UART occupies eight contiguous I/O port addresses. In the IBM PC, there are two defined locations for these eight ports and they are known collectively as `COM1` and `COM2`. The makers of PC-clones and add-on cards have created two additional areas known as `COM3` and `COM4`, but these extra COM ports conflict with other hardware on some systems. The most common conflict is with video adapters that provide IBM 8514 emulation.

UART is able to have 12 register including scratch register.

Base Address	DLAB	Read/ write	Abbreviation	Register Name
+0	0	Write	___	Transmitter Holding Buffer
	0	Read	___	Receiver Buffer
	1	Read/Write	___	Divisor Latch Low Byte
+1	0	Read/Write	IER	Interrupt Enable Register
	1	Read/Write	___	Divisor Latch High Byte.
+2	__	Read	IIR	Interrupt Instruction Register
		Write	FCR	FIFO Control Register

Base Address	DLAB	Read/ write	Abbreviation	Register Name
+3	___	Read/Write	LCR	Line Control Register
+4	___	Read/Write	MCR	Modem Control Register
+5	___	Read	LSR	Line Status Register
+6	___	Read	MSR	Modem Status Register
+7	___	Read/Write	___	Scratch Register

Use of DLAB

UART is allowed only 8bit data bus,if we stored in 16bits,2 registers are used. The first register when DLAB=1 stores the Divisor Latch low byte, where as the second register (base +1 when DLAB=1) stores the "Divisor latch high byte.

APPENDIX 2

BASICS OF THE LANGUAGE USED.

C language is used for developing the software.C language was first implemented by Dennis Ritche on a DEC PDP-11 that used the Unix operating system.

Advantage of C language

Middle level language

Since it combines the best elements of high-level with the control and flexibility of assembly language.

Portability

Portability means that it is easy to adopt software written for one type of computer or operating system to another type.

Support the Concepts of DataTypes

C support common data types such as integer,character,and floating-point in addition to that C support several built-in data types, rather than Pascal and Ada. C permits almost all type of conversion.

No run-time error checking

Unlike most high-level languages,C specifies almost no run-time error checking.

No demand for type compatibility between a parameter and an argument

C does not demand strict compatibility between a parameter and an argument. Other high-level computer language will typically require that the type of an argument be exactly same type as the parameter that will receive the argument. While C allows an argument to be any type so long as it can be reasonably converted into the type of the parameter.

Suited for system-level programming

C allows direct manipulation of bits,bytes,words,and pointers. This makes well suited for system-level programming.

DEFAULT OF EXISTING LANGUAGE

PASCAL is the existing language. Default of PASCAL as follows:-

1.Types and Scopes

1.a The size of an array is part of its type

The bounds of an array are part of its type. So it is impossible to define a procedure or function which applies to arrays with differing bounds.

1.b There are no static variables and no initialization

A 'static' variable is one that is private to some routine and retains its value from one call of the routine to the next.

In pascal has no such storage class. This means that if a pascal function or procedure intends to remember a value from one call to another,the variable used must be external to the function or procedure, and its name must be unique in the larger scope.

Due to the lack of static variable ,the source code become bigger and thus it takes more time for compilation and running. The lack of initializers exacerbates the problem of too-large scope caused by the lack of a static storage class. The time to initialize things is at the beginning,so either the main routine itself begins with a lot of initialization code,or it calls one or more routines to do initializations. In either case, variables to be initialized must be visible,which means in effect at the highest level of hierarchy. The result is that any variable that is to be initialized has global scope.

The other difficulty is that there is no way for two routines to share a variable unless it is declared at or above their least common ancestor. In C extern static storage class both provide a way for two routines to cooperate privately,without sharing information with their ancestors.

2.Related program component must be kept separate

Since the original Pascal was implemented with a one-pass compiler,the language believes strongly in declaration before use. The result is that a typical Pascal program reads from the bottom-up-all the procedures and functions are displayed before any of the code that calls them,at all levels. This is essentially opposite to the order in which the functions are designed and used.

All the declaration of one kind must be grouped together for the convenience of the computer. The main disadvantage of PASCAL is the inability to make such groupings in structuring large programs.

3.There is no separate compilation

Theoretically,there is no need for separate compilation-if one's compiler is very fast,recompiling everything is equivalent. But in practice,compilers are never fast enough and source is often hidden and file inclusion is not part of the language,so changes are time-consuming.

4.There is no type casting mechanism

There is no way to override the type mechanism,nothing analogous to the “cast” mechanism in C. This means that it is not possible to write programs like storage allocators or I/O systems in Pascal,because there is no

way to talk about the type of object that they return,and no way to force such objects into an arbitrary type for another use.

6.Control Flow

The control flow deficiencies of Pascal are minor but numerous. There is no 'break' statement for exiting loops. This is consistent with the

one entry-one exit philosophy .

The index of a 'for' loop is undefined outside the loop,so it is not possible to figure out whether one went to the end or not. The increment of a 'for' loop can only be +1 or -1, a minor restriction.

There is no 'return' statement. This leads to contortions to make sure that all paths actually get to the end of the function with the proper value.

The 'case' statement is emasculated because there is no default clause.

7.The Environment

The Pascal run-time environment is relatively sparse,and there is no extension mechanism except perhaps source-level libraries in the "official"language.

The standard I/O is defective. There is no sensible provision for dealing with files or program arguments as part of the standard language, and no extension mechanism.

There is no notion of access to command-line arguments.

8.Cosmetic Issues

In Pascal uses the semicolon as a statement separator rather than a terminator. As a result one must have a reasonably sophisticated notion of what a statement is to put semicolons in properly. Perhaps more important,if one is serious about using them in proper places,a fair amount of nuisance editing is needed.

In general,semicolon as a separator is about ten times more prone to error than semicolon as terminator.

There is no way to put non-graphic symbols into strings. Concepts like newlines,tabs,and so on are handled on each system in an 'ad hoc' manner,usually by knowing something about the character set.

There is no macro processor. The 'const' mechanism for defining manifest constants takes care of about 95 percent of the uses of simple #define statements in C.

The main points in the case against Pascal.

1.Since the size of an array is part of its type,it is impossible to write general-purpose routines,that is,to deal with arrays of different sizes. In particular,string handling is very difficult.

2.The lack of static variables,initialization and a way to communicate non-hierarchically combine to destroy the “locality” of a program -variables require much more scope than they ought to.

3.The one-pass nature of the language forces procedures and functions to be presented in an unnatural order;the enforced separation of various scatters program components that logically belong together.

4.The lack of separate compilation impedes the development of large programs and makes the use of libraries impossible.

5.The order of logical expression evaluation cannot be controlled,which leads to convoluted code and extraneous variable.

6.The 'case ' statement is emasculated because there is no default clause.

7.The standard I/O id defective. There is no sensible provision for dealing with files or program arguments as part of the standard language,and no extension mechanism.

8The language lacks most of the tools needed for assembling large programs,most notably file inclusion.

9.There is no type casting.

9 BIBLIOGRAPHY

The Complete Reference C Fourth Edition	Herbert Schildit
Let Us C ,6 th Edition	Yashwant Kanetkar
C programmer's Guide to Serial Communications	Joe Campbell
Programming In ANSI C	Balagurusami
A Comparison of the Programming Languages C and Pascal	Alan R. Feuer and Narain H.Gehani
Software Engineering : A Practitioner's Approach	Roger S. Pressman

www.google.com

www.camiresearch.com

www.beyondlogic.org

www.quatech.com

www.taltech.com

www.developer.apple.com