DEVELOPMENT OF AN INTERFACE MONITOR TO OPERATE OVER A DATA TRANSMISSION LINE

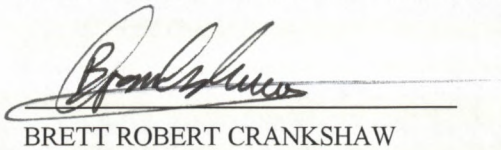
BRETT ROBERT CRANKSHAW

DEVELOPMENT OF AN INTERFACE MONITOR TO OPERATE OVER A DATA TRANSMISSION LINE

BRETT ROBERT CRANKSHAW
AUGUST 1994

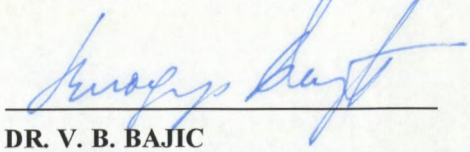
SUBMITTED IN COMPLIANCE WITH THE REQUIREMENTS FOR THE **MASTER'S DIPLOMA IN TECHNOLOGY**: ELECTRICAL ENGINEERING (LIGHT CURRENT), IN THE DEPARTMENT OF ELECTRONIC ENGINEERING, TECHNIKON NATAL, DURBAN, R.S.A.

I HEREBY DECLARE THAT THIS DISSERTATION REPRESENTS MY OWN WORK BOTH IN CONCEPTION AND EXECUTION.



BRETT ROBERT CRANKSHAW

APPROVED FOR FINAL SUBMISSION



16.08.1994

DR. V. B. BAJIC
Grad.E.E., M.E. Eng. Sc., D. Eng. Sc. (E.E.)
SUPERVISOR

DATE

MR. B. G. GALBRAITH
NTTD (TW), Dip Tech (Telecomm) (TP)
CO-SUPERVISOR

Acknowledgments

The author wishes to acknowledge, with sincere thanks and appreciation, the contribution made by the following individuals and institutions towards this dissertation.

- **Telkom S.A.** Ltd. for their positive response to the author's application for this study. This opportunity and the payment of the registration fees are very much appreciated.
A special thanks, in particular to the following men from Telkom, for their respective roles in releasing me for this project; Mr. F. Fedderke, Mr. E. Frenzel, Mr. S.G. King and Mr. C. Mathiesen.

- **Dr. Vladimir Bajic** the project supervisor at Technikon Natal for his supervision, guidance and support. His comments were instrumental in the formulation of this dissertation.

- **Mr. Bruce Galbraith** for the time sacrificed in the discussion of the project, together with the sharing of his experience and advice. His enthusiasm and dedication to the Electronic Engineering Department, is admired.

- **Altech Instruments**, (Johannesburg) for the efficient response to the inquiry with respect to the specifications of the Chameleon protocol analyser.
Mr. D. Crotz (**Alcatel Altech,** Johannesburg) for his facsimile regarding the location of the Quadruple Differential Line Receiver/Driver ICs.

- **Mr. Hans Bonde** (**Tekelec** INC., 26580 West Agoura Road, Calabasas, California, The United States of America) for the prompt reply facsimile with respect to the inquiry mentioned above.

- **Mr. L. Lovell** (Telkom S.A.) for his assistance in the operation of the software package, P-CAD, and with the manufacture of the PCB negatives.

Finally the people who have offered the most sacrifice towards the completion of this study, my family. I would like to express my sincere appreciation to my wife **Mary** and my four daughters, **Toni, Kim, Emma and Tiffani** for their understanding, love and support of the long hours that were necessary for the successful completion of this project.

ABSTRACT

One of the telecommunication facilities provided by Telkom S.A. is connection to the National and International X.25 data network. This network follows the international data protocol standard based upon the CCITT Recommendation X.25. The network allows data calls to be made from the X.25 user in very much the same fashion as a caller on the Public Switched Telephone Network (PSTN), in that the caller must specify the number of the called user on the network in order to be connected. It is in this environment, specifically in the testing and maintaining of these data circuits, that this study is based.

In the same manner that a caller on a PSTN would use a telephone line to make access to the nearest telephone exchange (switching point), the X.25 user requires a data circuit to provide access to the X.25 Node (switching point). This data circuit, connecting the X.25 user's equipment to the X.25 Node, is called the **access circuit**. Monitoring of the actual X.25 packetised data (**on the access circuit**) with the aid of data analyzers, plays a vital role in the maintenance and testing of X.25 circuits. A more helpful and sophisticated tool in the fault diagnosis of X.25 circuits however is a protocol analyzer. One of these protocol analyzers currently in use at the X.25 test and maintenance centre is the **"Chameleon"** by Tekelec INC.

The data/protocol analyzers used in fault diagnosis of these circuits, obviously require access to the data on the circuits. All data interfaces of X.25 access circuits pass through a Data Patch panel (Telenex Corporation 1991) before entering the Node. This panel provides a monitoring position for the analyzer to access any one of the data interfaces.

Telkom S.A. has basically 3 Node centres in the Natal region, namely

Durban, Newcastle and Pietermaritzburg. Durban is the largest of these and this is where the X.25 maintenance and test centre is situated (Saponet). This is the only fully fledged test centre in Natal (having staff who are specialised in X.25 problems) as the quantities of X.25 users connected to the other Nodes do not warrant their own test centres.

The **problem** experienced by Durban, and other major test centres in the country, is that the test technicians have no access to the data interfaces of a those X.25 users connected to the Nodes ***remote*** to the test centre. This problem initiated the quest for the development of a method to monitor remote data interfaces. **This study develops a prototype system that will facilitate remote interface monitoring using a data transmission line**. It will comprise of a Remote and Central Sub-system (see Developed Technology in Figure 1a) that are connected via an independent data communications line. The Remote Sub-system will monitor the remote interface and the Central Sub-system will reproduce the interface locally (for monitoring purposes, Figure 1a). There is theoretically no limit on the geographical separation of the two sub-systems.

EKSERP

Een van die telekommuniksie fasiliteite verskaf deur Telkom S.A. is die verbinding met die Nasionale en Internasionale X.25 data netwerk. Hierdie netwerk volg die internasionale data protokol standaard wat op die CCITT rekommendasie X.25 gebaseer is. Hierdie netwerk laat data oproepe van die X.25 gebruiker toe op ongeveer dieselfde manier as ñ oproeper op die Openbare Geskakelde Telefoon Netwerk (PSTN), in soverre dat die oproeper die nommer van die geskakelde gebruiker op die netwerk moet spesifeer, alvorens hy verbind kan word. Dit is teen hierdie agtergrond, spesifiek in die toetsing en instandhouding van hierdie data verbinding, dat dié navorsing plaasgevind het.

Op soortgelyke wyse waarop n oproeper op n PSTN n telefoonlyn sal gebruik om toegang tot die naaste telefoonsentrale te verkry, sal die X.25 gebruiker n data verbinding benodig om toegang tot die X.25 Nodus (aanskakel punt) te verkry. Hierdie data verbinding wat die X.25 gebruiker se toerusting met die X.25 nodus verbind, word die toegangsverbinding (access circuit) genoen. Die monitor van die eintlike X.25 gepakketeerde data (op die toegangsverbinding) met behulp van die analiseerders, speel n essentiële rol in die instandhouding en toetsing van die X.25 verbindings. n Meer handige en gesofistikeerde hulpmiddel in foutdiagnose van die X.25 verbindings is egter n protokol analiseerder. Een van hierdie protokol analiseerders huidiglik in gebruik by die X.25 toetsing- en instandhoudingsentrum is die "Chameleon" by Tekelec INC.

Die data/protokol analiseerders wat in foutdiagnose van hierdie verbindings gebruik word, benodig natuurlik toegang tot die data in die verbindings. Alle data tussenfase van die X.25 toegangsverbindings beweeg deur n Data Patch paneel (Telenex Korporasie 1991) voordat dit

die nodus binnedring. Hierdie paneel voorsien n monitor posisie vir die analiseerder om toegang tot enige van die data tussenfases te verkry.

Telkom S.A. het basies 3 nodus sentrums in die Natal gebied, naamlik Durban, Newcastle, en Pietermaritzburg. Durban is die grootste van hierdie drie en dit is waar die X.25 instandhouding- en toetsingsentrum geleë is (Saponet). Dit is die enigste volledig toegeruste toetsentrum in Natal, met personeel wat gespesialiseerd is in die X.25 se probleme. Daar is nie genoeg gebruikers in die Newcastle en Pietermaritzburg gebiede om n volledige toetsentrum te regverdig nie.

Die probleem wat deur Durban en ander hoof toetsentrums in die land ondervind word, is dat die toetstegnici geen toegang het tot die data tussenfases van die daärdie X.25 gebruikers wat gekoppel is aan die nodusse wat verafgeleë van die toetssentrum is. Hierdie probleem het aanleiding gegee tot die soeke na die ontwikkeling van n metode om die verafgeleë data tussenfases te monitor. Hierdie navorsing ontwikkel n stelsel wat die monitor van verafgeleë tussenfases moontlik maak deur die gebruik van n transmissielyn. Dit sal bestaan uit n Verafgeleë (Remote) en Sentrale (Central) Substelsel (sien ontwikkelde tegnoglie in Figuur 1a) wat gekoppel is deur middel van n onafhanklike data kommunikasielyn. Die Verafgeleë Substelsel sal die verafgeleë tussenfase monitor en die Sentrale Substelsel sal die tussenfase plaaslik reproduseer (vir monitor-doeleindes, Figuur 1a). Daar is teoreties geen perke op die geografiese skeiding van die twee substelsels nie.

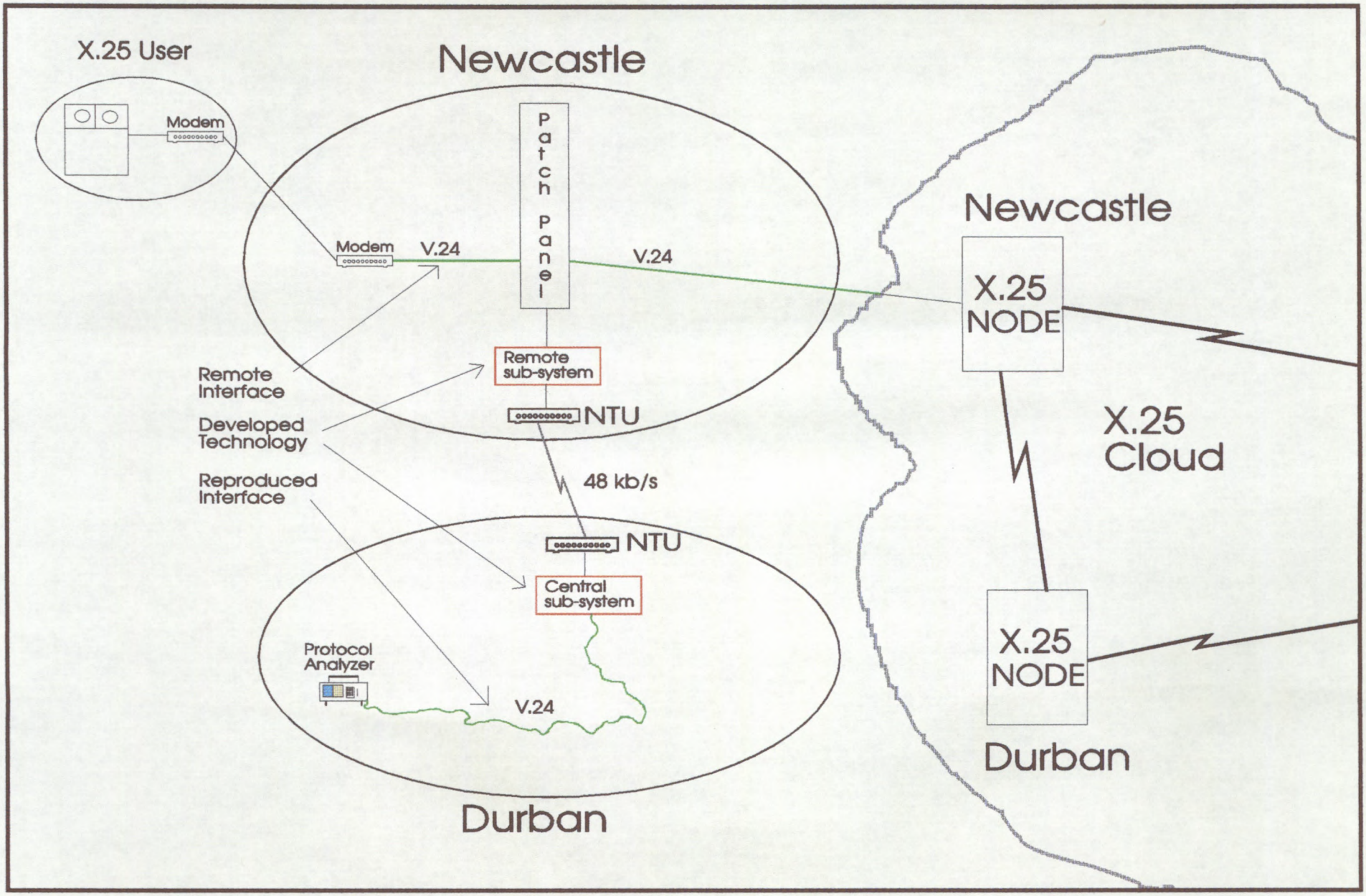

X.25 User
Modem
Newcastle
Patch Panel
Modem
V.24
V.24
Remote sub-system
NTU
Remote Interface
Developed Technology
Reproduced Interface
48 kb/s
NTU
Central sub-system
Protocol Analyzer
V.24
Durban
Newcastle
X.25 NODE
X.25 Cloud
X.25 NODE
Durban

Figure 1a

CONTENTS

CHAPTER ONE:

INTRODUCTION

This study is devoted to the development of a practical solution to a practical problem experienced by the test and maintenance staff of the X.25 packet switched data network of Telkom S.A. In order to precisely describe the technical nature of the problem, together with the development of the solution, it is necessary to list and define the specific terminology used in this dissertation.
These terms are listed in a glossary, located at the back of the appendix.

1.1 Introduction to the Problem

Users of the X.25 switched data network are scattered all around the Natal province and are connected to the network through the nearest X.25 switching node. There are two such nodes located in Natal other than at the main centre for Natal in Durban, i.e. Pietermaritzburg and Newcastle. The circuits cross the distance from the clients' premises to the actual node via Access circuits. These may make use of modems or, if using the digital data network of Telkom (Diginet), NTUs (Figure 1b). These modems or NTU's on Telkom's premises are each connected to the node via an interface cable which first passes through a patch panel. This panel is the point at which most of the testing is done and this is where the problem manifests itself. When fault locating, the test staff usually connect a protocol analyser to the monitor position of the patch panel and monitor the specific interface in question. Currently however the staff in Durban have no access to the interfaces at the patch panels of the remote nodes, i.e. Newcastle and Pietermaritzburg.
The aim of this study is to develop a practical solution to this problem.

The solution should provide the central test staff (Durban Saponet), with the signals from these remote interfaces in a manner that will assist in the testing and fault localisation of the X.25 circuits using these remote interfaces.

In order to explain the problem in detail an explanation of the functioning and architecture of the Switched Data Network provided by Telkom S.A. is required.

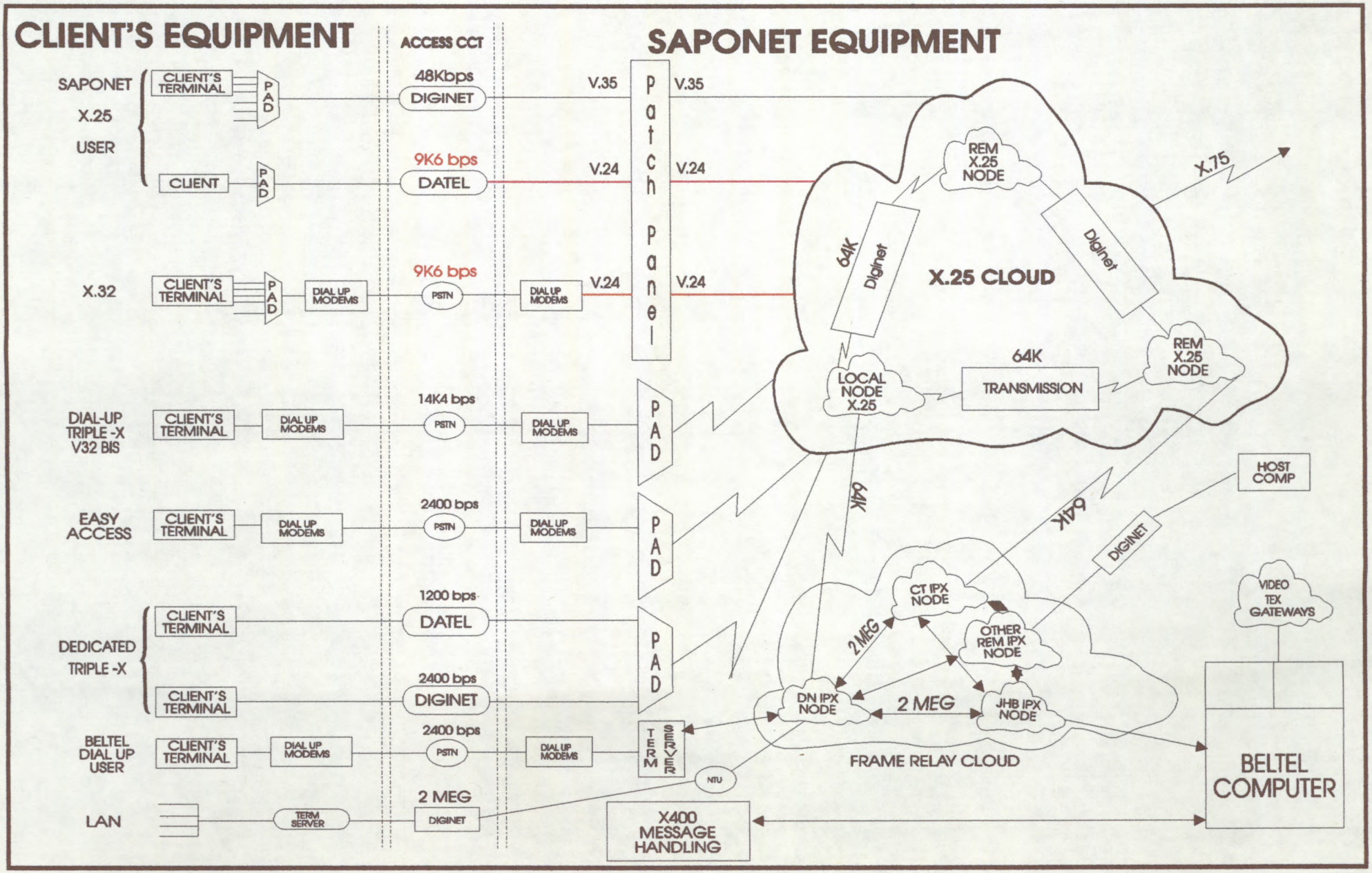

CLIENT'S EQUIPMENT
ACCESS CCT
SAPONET EQUIPMENT
SAPONET X.25 USER
CLIENT'S TERMINAL
PAD
48Kbps
DIGINET
V.35
V.35
CLIENT
9K6 bps
DATEL
V.24
V.24
Patch Panel
X.32
DIAL UP MODEMS
PSTN
9K6 bps
V.24
V.24
X.25 CLOUD
REM X.25 NODE
X.75
64K
Diginet
LOCAL NODE X.25
64K
TRANSMISSION
REM X.25 NODE
DIAL-UP TRIPLE -X V32 BIS
14K4 bps
EASY ACCESS
2400 bps
HOST COMP
DIGINET
VIDEO TEX GATEWAYS
DEDICATED TRIPLE -X
1200 bps
DATEL
2400 bps
DIGINET
CT IPX NODE
OTHER REM IPX NODE
2MEG
DN IPX NODE
2 MEG
JHB IPX NODE
BELTEL DIAL UP USER
2400 bps
TERM SERVER
NTU
FRAME RELAY CLOUD
BELTEL COMPUTER
LAN
TERM SERVER
2 MEG
DIGINET
X400 MESSAGE HANDLING

Figure 1b

1.2 Introduction to the Public Switched Data Network of Telkom S.A. Ltd.

There are various methods of accessing the Public Switched Data Network (Figure 1b). The large majority of circuits/users operating through the X.25 Switched Data Network have permanent synchronous links to connect their equipment (PAD on their premises) to Telkom's nearest Switched Data Network Node. These data circuits use either an analogue modem or an NTU (when making use of the Digital Data Network of Telkom, DIGINET) to provide the link between their premises and Telkom's.

The speed required by the user determines what method will be used to provide the synchronous link from their PAD to the Node. Since the analogue modems are largely limited to 9600 bps the Diginet service attracts the clients who require speeds of 48 kbps and above.

Provision is made for the smaller companies or less frequent users to access the network without having their own PAD. This provides a more cost effective alternative in accessing the network. This is possible via the Triple-X PAD. This PAD installed at Telkom's premises allows access to the X.25 network by terminals not working on the X.25 protocol. This access is governed by another set of standards (X.3, X.28 and X.29) (Telkom 1992 p. 1-10). Access to the Triple-X PAD can be made via modem lines operating over dedicated or dial-up (through the PSTN), synchronous or asynchronous links.

These users have a lower data traffic volume and it therefore would not be cost effective for the client to invest in their own PAD.

With X.400, messages can be sent from the X.25 network to the Telex/Teletex network and even to facsimile machines. Connection via X.25 is also possible to the Beltel computer (Figure 1b).

As indicated by the above few paragraphs the variety of combinations and protocol arrangements possible in accessing the X.25 network is fairly extensive. This causes a difficulty when searching for a solution to the problem of remote interface monitoring. For this reason the solution adopted was one which seemed most practical and most all inclusive.

Of importance is the number of X.25 clients using V.24 interfaces to access the network. These V.24 interfaces enter the Node via the Patch Panel. This group constitutes approximately 64 % of the users at the Pietermaritzburg node, and 90 % of those at the Newcastle node. Of these circuits using **dedicated analogue modems** on V.24 interfaces, the **vast majority** operate at a data transmission speed of **9600 bps** (M. Carle 1993).

For this reason the solution to the problem, mentioned in 1.1, was designed to cater for the monitoring of interfaces conforming to the V.24 and V.28 specifications and operating at a synchronous data transmission speed of **9600 bps** (9.6 kb/s).

CHAPTER TWO:

TESTING AND MONITORING REMOTE INTERFACES

2.1 Introduction

Significance of the Study

In the Public (packet) Switched Data Network provided by Telkom S.A., each service port enters the switching node via an interface patch panel. This patch panel provides a test and monitoring point used by the maintenance personnel. At this point the technicians connect the test equipment (amongst others, an X.25 protocol analyser) to the specific service's port (data communications interface). In this manner the data is monitored in order to answer client queries and to localise reported service faults.

This testing point is vitally important when monitoring for intermittent errors. At this point access is made to all the interface signals of the actual data interface itself (Telenex 1991 p 9). Without this testing point fault diagnosis would be very much more difficult, if not virtually impossible in certain circumstances.

Background Information to the Problem

All X.25 services in Natal R.S.A. are controlled and maintained by trained and specialised personnel at a central site in Durban. In other words, there are no specialised X.25 personnel at the remote nodes in Natal (Newcastle and Pietermaritzburg). Some of these personnel members have undergone a certain amount of training in X.25, however they do lack in experience. At present the staff in Durban are reliant on the remote staff for assistance

in the fault diagnosis of services working through those nodes. The remote personnel have other major job responsibilities and are consequently not always available to give assistance. The personnel at the remote nodes in Natal are equipped with data analysing equipment but have inferior testers to those at the controlling centre in Durban (with specific reference to the "Chameleon" Protocol Analyser manufactured by Tekelec). The Chameleon breaks down the monitored data into visible packets and completely decodes them. This function simplifies and streamlines the process of packet analysis. It has good storage facilities and has proved to be an invaluable tool in the fault localization of X.25 circuits.

The Need to Find a Method of Remote Interface Monitoring

In order for the centrally located test personnel (based in Durban) to monitor and investigate faults on remote circuits, a method of remote interface monitoring is required.

The technology presently used in this department has certain obvious short comings;

a) Circuit monitoring and fault localisation of circuits working to the remote nodes is complicated after normal office hours, as there is a need for personnel to be present at both the remote and the central nodes. This being necessary because the central controlling centre (Durban) dealing with the fault would require the remote personnel to monitor the circuit and then verbally relay the information via the telephone.
b) The verbal relaying of the packet (presented by the test equipment at the remote node) from the circuit in question is often less than reliable and is not desirable for this kind of application.

c) The remote test equipment does not completely analyse the packet (as does the Chameleon) and hence the delay from the occurrence of an error/special packet, to the time that it is interpreted by the remote personnel and then relayed to the central test personnel is longer than desired. This delay causes complications under certain circumstances, e.g., when interactively testing. Under certain circumstances a telephonic conversation is held with the client/user while the personnel watch for the packets sent on the circuit as they are sent. This cannot be solved by simply purchasing more Chameleons for the province as they each require an approximate investment of R120 000 (Altech Instruments).

The Benefits of Developing a Method of Remote Interface Monitoring

Bearing in mind the shortcomings of the present set of circumstances surrounding the testing of X.25 circuits at remote nodes, there are definite advantages envisaged from the development of remote interface monitoring. These benefits envisaged from the solution to this problem are outlined below.

a) Reduction in the Overall Maintenance Costs.

Under the present conditions specialised staff would be required at the remote nodes. An expensive data analyser would need to be purchased for each remote node site. Both the staff and the equipment at the remote nodes would now be under-utilised due to the fact that the remote nodes serve fewer data circuits and hence less manpower is required to serve in the testing thereof. The solution provided by this project would eliminate the need for both

the expensive equipment and specialised staff at all remote nodes. Overtime costs will be reduced for the company as the remote personnel will not be required to stay overtime and relay monitored information for those faults that last after office-hours.

b) Simplified Fault Finding

The central maintenance staff will now have direct access to the information on the interfaces at remote nodes. They will not have to rely on less experienced staff stationed at the remote nodes, who themselves (from a less sophisticated data analyser) gather information from the interface and then verbally relay it to the Test Centre via telephone. This process is complicated and always introduces delays.

The proposed system, if simple to operate, will thereby allow most technical staff available at the time to be able to assist and connect the monitoring system to the desired interface with very little instruction. This should eliminate delays incurred when the remote X.25 personnel are otherwise occupied.

c) Country Wide Usage

The technical solution obtained by this study can be reproduced by Telkom S.A. for the benefits it can provide, and be used in other regions/provinces of R.S.A. where similar situations have been confirmed to exist.

d) Longer Monitoring Hours

Problems (on services working through remote nodes) of an intermittent nature are generally placed on monitor for extended periods. Once the project's remote sub-system has been manually plugged into the correct interface at the remote patch panel, no

more assistance from the remote staff should be required. The automatic nature of the project will enable the central test staff to monitor the interface over however long a period they may require (including after office hours). This will result in a reduction in labour expense to Telkom S.A. as the staff at the remote node would not be required to be present.

d) Overall Improvement in Service

The improved testing and monitoring facilities together with the simplification of the process as mentioned in b) should result in a shorter repair time (for circuits operating into remote Nodes). This should thereby reduce the mean time to repair (MTTR) reported faults. This should positively affect the overall Grade of Service offered to X.25 users in the province and ultimately in the country.

2.2 Additional Details Regarding the Study

2.2.1 Defining the Purpose of this Developmental Study

The purpose of this research project is to design, build and test an electronic system to facilitate remote data interface monitoring that will enable an X.25 protocol analyser to monitor V.24 interfaces at remote switching nodes.

2.2.2 The Sub-Division of the Main Goal

The overall goal of the study can be sub-divided into the following tasks:

2.2.2.1 Sub-goal I

To establish the requirements and specifications the proposed system must meet and thereby establish parameters for the hardware and software design.

2.2.2.2 Sub-goal II

To design and build the suitable hardware that will facilitate remote interface monitoring using a data transmission line.

2.2.2.3 Sub-goal III

To develop the software that will:

a) control the hardware of the remote and central sub-systems developed in 2.2.2.2

b) process the monitored data at the remote sub-system (into a form suitable for transmission over the data transmission link)

c) process the data received at the central sub-system (in order to re-produce part of the monitored interface)

2.2.2.4 Sub-goal IV

To evaluate the results and performance of the system in terms of the criteria set in 2.2.2.1.

2.2.3 Framework Adopted in the Research

2.2.3.1 Design Specification

It was hypothesized that it would be possible to establish the requirements and criteria that the system should meet and thereby establish the design parameters.

2.2.3.2 Proposed Hardware

It was hypothesized that it would be possible to design and build the hardware that would be suitable to facilitate remote interface monitoring using a data transmission line

2.2.3.3 Proposed Software

It was hypothesized that it would be possible to develop software that would;

a) control the hardware of the remote and central sub-systems developed in sub-goal II

b) process the monitored data at the remote sub-system (into a form suitable for transmission over the data transmission link)

c) process the data received at the central sub-system (in order to re-produce part of the monitored interface)

2.2.3.4 Project Success

It was hypothesized that the developed system would meet the performance criteria established in sub-goal I and therefore be a success.

2.2.4 Assumptions Made Prior to the Development

In the study the following assumptions are made:

I

The necessary components and peripherals for development and construction of the proposed system are readily available.

II

The digital transmission link (Telkom S.A.'s Diginet service) used by the proposed system to cross the span between the central and remote nodes is of sufficiently high grade to be considered error free.

2.2.5 Restrictions Placed on The Study

a) The aim of the project is not a commercially marketable system but a prototype of a system that could be reproduced on a small scale, for the benefits it can provide.

b) The following circuits of the CCITT recommendation V.24 interface will be monitored:
   i) Circuit no. 103 (Transmit data)
   ii) Circuit no. 104 (Receive data)

   If feasible, other interface circuits could be included in the monitoring (CCITT (1985 b) Table 1/V.24 p. 102).

c) The interface signals presented at the reproduced interface (by the central sub-system) will not necessarily be received in real

time (as the process may involve buffering the information prior to it's transmission).

d) With respect to the signals on the monitored interface, this project will not be involved with protocol analysis nor fault diagnosis. The scope of the project will end at providing the signals to the equipment with such facilities (a protocol analyser).

e) The system developed will attempt to cope with monitoring interfaces with transmission speeds of up to 9600 bps since the majority of the interfaces at the remote nodes will be operating at 9600 bps (see 1.2).

f) The scope of the project will not include the development of a modem or any other data transmission device but will make use of a high grade data transmission line and associated NTU's.

CHAPTER THREE:

DEVELOPMENT OF THE REMOTE INTERFACE MONITOR

This chapter will take the reader through the various stages in the development of the Remote Interface Monitor.

3.1 Introduction

Firstly a set of design criteria and specifications were drawn up. These applied to the hardware and software of the two sub-systems of the proposed remote interface monitor. This was to provide an instrument of assessment by which the success of the hardware and software development was to be measured.

Secondly, the hardware was designed and constructed in terms of the above specifications. The hardware's basic functions were tested, modifications were performed and more tests conducted to conclude what hardware configuration would perform the required functions best.

Thirdly, the software was developed installed and tested. Minor modifications were made on the hardware and tests were again run. The software was developed in this way until the final version was complete. The system was then tested against the design criteria and specifications on site.

Finally the project hardware was developed using a CAD package and converted from bread board to a constructed prototype system on printed circuit boards.

3.2 Design Specifications and Criteria used in the Development of the Study

3.2.1 Design Criteria and Specifications of the Remote Sub-System

i) The monitoring circuitry should not affect the data on the working of the interface being monitored in any way. The interfaces monitored will be provided by the patch panel (monitor position). This interface now provided by the patch panel will not affect the monitored interface as it will still be in a through connected state.

ii) The remote system must be able to monitor synchronous data interfaces.

iii) The remote sub-system should be auto-sensing as far as the speed of the monitored interface is concerned. It should require as little operator attention and be as automatic as possible.

iv) The remote sub-system should be able to accept the "level" of the signals on the interface. RS232C is an Electronic Industries Association (EIA) standard for the electrical characteristics corresponding to the CCITT's V.24 international standard as confirmed in Chameleon 32 User's Guide p. B-2. V.24 is a list of definitions for interchange circuits between Data Terminal Equipment (DTE e.g. computer) and the Data Circuit-Terminating Equipment (DCE e.g. modem). The physical characteristics of the connector used are defined by the International Standards Organisation (ISO 2110). This 25 pin connector is commonly known as DB 25 and is most commonly found for V.24 interface connections although ISO 4902 can also be used (CCITT (1985 b) p. 101). The monitored V.24 interface will hence be operating on standard levels and the monitoring circuitry of the remote sub-system will require RS232C line receivers for monitoring.

v) Every single data bit and clock pulse on the monitored interface should be reproduced on the central side. The best possible design would result in a monitor that will reproduce the monitored interface bit for bit, identical to the original. This will mean that the remote monitoring sub-system with central sub-system should together work as a **transparent** monitoring system. This was decided because of the fact that this project is intended to be used as a fault diagnostic aid, and in order to successfully pin point errors, **all** the necessary information regarding the interface in question should be made available to the test staff and their equipment.

vi) The remote sub-system should be able to perform all of the tasks of the monitoring end, namely;

   a) monitoring all the information required from the monitored interface,
   b) transmit the information onto the data link in a form acceptable to the local sub-system,
   c) simultaneously control a data communications link to the central sub-system.

3.2.2 Design Criteria and Specifications of the Central Sub-System

i) The signals reproduced to recreate the interface being monitored should be on the level at which they are monitored. This means that the central sub-system will require RS232 line drivers to convert the signals to the level required on the reproduced V.24 interface. The electrical characteristics of the Chameleon's V.24 interface will be compatible to these signal levels.

ii) The central subsystem should be able to control the data communications link to the remote end.

iii) The central sub-system should be able to interpret the data received from the communications link into valid data for reproduction of the V.24 interface.

iv) The central sub-system must present the information to the interface in such a way that is acceptable to the protocol analyser, the Chameleon in particular, as this was the main tester that is used by the maintenance staff. Particular attention must be paid to the clocking of the data onto this interface.

v) The central sub-system should be designed to require as little operator attention an be as automatic as possible.

3.2.3 Other Design Considerations

3.2.3.1 Signal Element Timing (Data Clock)

Consideration had to be given to the clock sources for both the communications link and the clocking of the data to the reproduced interface. One must bear in mind that the monitored interface has its own clocking system and works totally independently of any monitor placed on it. This would mean that the clock used for timing the data bits on the reproduced interface is totally asynchronous (independent timing) with the monitored interface. For the purpose of illustrating the importance of correct handling of the data clocking used in the project the following situations are presented.

Typically the design tolerance of a modem data clock is ± 0.01%. Below a worst case scenario is used, where the two clocks of the two modems are at the worst possible extremes and combined to have largest mismatch.

Data transmission rate for this example is 9600 bps

Maximum frequency: f + 0.01% = 9600.96 Hz

Minimum frequency: f - 0.01% = 9599.04

Worst case difference:

Maximum frequency - Minimum frequency

= 9600.96 Hz - 9599.04 Hz = 1.92 Hz (or bps)

To Calculate the Number of bits difference in 8 hours

Seconds in 8 hours = 60 x 60 x 8 = 28800

Number of bits difference in 8 hours = 28800 × 1.92 bps

= 55296 bits

or (6912 Bytes)

a) Consider the clock at the remote end to be faster than the clock used at the central by the above amounts. What would happen is that more data would be monitored than what would be output at the central side and hence a buffer or storage facility would be required. This however would not remedy the problem as eventually over a few hours the buffer demands would become unreasonably large and data would be overwritten and lost.

b) Consider also that the clock at the remote end was slower than the clock at the central side. This would result in more data being put out than being put in to the system and unless the clock at the output (at the reproduced interface) is stopped periodically, there would be a problem with the clock running but no valid data to put out to the interface.

With all of this information in mind it would be advisable to choose the above option b), and run the **clock at the central side slightly**

faster than the anticipated data clock frequency on the remote side. The limiting factors to be considered would include;

a) the bit rate limitations placed by the specifications of the data interface, namely CCITT V.24,
b) the tolerance of the device connected to the reproduced interface to the periodic stopping of the data clock,
c) the practical design of such a circuit that would produce and control such a data clock signal as and when required.

3.2.3.2 The Choice of Micro-Processor Hardware

The Intel microcontroller 8751H was initially chosen to perform the control of the two sub-systems. The choice of the micro-processor was largely made on the following grounds;

a) The physical components.

   Many micro-processors require additional circuitry for their operation, mostly because of the clock source and program storage. The Intel 8751H micro-controller was chosen for its internal memory that will eliminate the need for external latches and ROM chips. This will result in a more compact final product as the PCB can be that much simpler and smaller. The internal ROM of two kilo-bytes was considered ample for this project.

b) Ease of accessibility

   Intel micro-processors are readily available from suppliers in the city where the development took place, Durban, R.S.A.

c) Technical Support

   Intel being the biggest semiconductor manufacturer in the United States of America and the second largest in the world (Intel (1992 b), preface) has resulted in an abundance of support available. This support ranges from hardware and software, to

documentation and development aids, such as In Circuit Emulators and Simulator packages.

d) Speed of microcontroller versus performance required.

Time frame of the Monitored interface:

Baud rate = 9600 bps

Therefore

1 bit period = (1/9600) s = 104,16 micro seconds (μs)

and the period of an 8 bit character = 8 x 104.16 μs = 833.3 μs

This shows that the interface monitor should be able to monitor a byte every 833.3 μs and gives an indication of the time frame that the monitor will be working under. This places certain restrictions on the speed of the oscillator clock used by the micro-processor and the amount of coded instructions it will be able to execute between monitored bytes. It must be born in mind here that there will be two bytes to be monitored every 833.3 μs (one for each direction of transmission).

Time frame of the Micro-controller

using 12 MHz crystal

Execution period of a single cycle instruction:

= 12 oscillator periods = (1/12 MHz) = 1 μs

Execution period of a double cycle instruction:

= 2 μs

(Intel (1992 b) p. 5-17)

Interrupt response latency 3 - 7 μs

The above calculations indicate the time frame under which the micro-controller will operate. From these it can be seen that while monitoring the interface, the micro-controller will be able to execute approximately 415 2-cycle instructions during the

period of one eight bit character. The 415 instructions were considered sufficient for the micro-controller to perform the anticipated control duties and other functions as the interface monitor during this period.

The same microcontroller was chosen for the central sub-system.

3.2.3.3 Communications link and Interface Standard.

The communication link chosen is one of the services offered by Telkom S.A. It forms part of the digital data network "DIGINET". The costs of the services provided by this network are very favourable when compared to the dedicated Analog data circuit options.

An example of a cost comparison between the two services is outlined below.

The example used is a data link from Newcastle to Durban (Central).

Analogue leased line:

Service Fee, per end:	Termination	R 180.00
	Modem	R 373.00
	VAT	R 77.42
	Total	**R 630.42**
Monthly Rental charges:	Modem @ R450	R 900.00
	Local Leads	R 76.80
	Trunk (258 km)	R 1019.10
	VAT	R 279.43
	Total	**R 2275.33**

For the above service one would receive a 24 hours per day, permanent and private data link between a site in Newcastle and one in Durban (central). The connection (using Telkom's modems) would operate at 9600 bps in a synchronous full-duplex mode. The cost would consist of an initial charge of R 1260.84 (2 × R 630.42) and then a monthly rental of R 2275.33.

High Speed Diginet circuit:

Service fee, per end:	Termination		R 180.00
	NTU		R 248.80
	VAT		R 60.03
		Total	**R 488.83**

Monthly Rental charges:	NTU's		R 269.00
	Local Leads		R 76.80
	Diginet Ports		R 240.00
	Line charge (258 km)		R 769.60
	VAT		R 189.76
		Total	**R 1545.16**

(SITES 1994)

For the above service one would receive a 24 hours a day permanent private data between a site in Newcastle and one in Durban (central). The connection (using Telkom's NTU's) would operate at 48000 bps in a synchronous, full-duplex mode. The cost would consist of an initial charge of R 977.66 (2 × R 488.83) and then a monthly rental of R 1545.16.

The Analogue circuits operating on two pairs of telephone cable are largely limited to 9600 baud. Faster modems using sophisticated data compression methods are now available but are not rented out by Telkom S.A. It would not have been cost effective to purchase

two such modems for this study when compared to the cost and transmission speed of the Diginet service.

The Diginet NTU can operate at speeds of 600 bps, 1.2 kbps, 2.4 kbps, 4.8 kbps, 9.6 kbps, 48 kbps or 64 kbps (Standard Telephone & Cables 1986).
Higher speeds of multiples of 64 kbps (n x 64 kbps) can be provided with a different, more expensive NTU. These higher speeds do cost a considerable amount more than the 48 kbps line.

The speed of the communications link required for this project is dependent on the speeds of the monitored interfaces. When monitoring the 9600 bps interfaces a speed of over 19.2 kbps is needed unless some sort of data compression is incorporated. This is explained below.

When monitoring a 9600 bps interface there will be 9600 bits per second that require monitoring on each of the two directions of data flow (full-duplex). In other words in one second there will be 9600 bits transmitted in one direction and 9600 bits transmitted in the other. This totals to 19200 bits per second. Now to transmit this information across a data link there will be some sort of data protocol involved which will control the data flow and maintain link synchronism. This would increase the number of bits required to be transmitted, pushing the transmission rate beyond the figure of 19.2 kbps.

The Diginet 48 kbps service would provide ample time on the link for all the necessary transmission of monitored data and supervisory signals.

After considering the project's design requirements and also the costs of the options available, the 48 kbps Diginet service was the option chosen to provide the data communication link between the proposed Central and Remote sub-systems.

There are various versions of NTU's available and the type used by this project is the Universal NTU manufactured by STC. This NTU is so called as it has three different interface sockets. It provides the option to the user for their interface standard preference. Together with this it is an advantage when different Data Terminals use the same NTU. The interface options given by the 4 wire Universal NTU are;

(a) The X.21bis/V.28 with D-25 pin connector

(b) The X.21bis/ V.35 with 34 pin connector

(c) The X.21/V.11, with D-15 pin socket connector (ISO 4903)

For function and simplicity the V.11 interface was chosen for use by this project.

3.3 Development of the Monitoring Circuitry (Remote sub-system)

3.3.1 Selection of an Appropriate Monitoring Method

A method of monitoring the synchronous data from the interface had to be devised. This method must also transform the serial data (monitored from the interface) into parallel data bytes, eight bits wide (a form usable to the microcontroller). Below are the various options investigated to perform this function. Each option is discussed and the weaknesses and strong points of each are given.

i) Shift register, Latch and Counter Method

A method that would perform such a function could involve a mod eight counter, an eight bit shift register and an eight bit tri-state latch.

Figure 2 outlines such a possible solution.

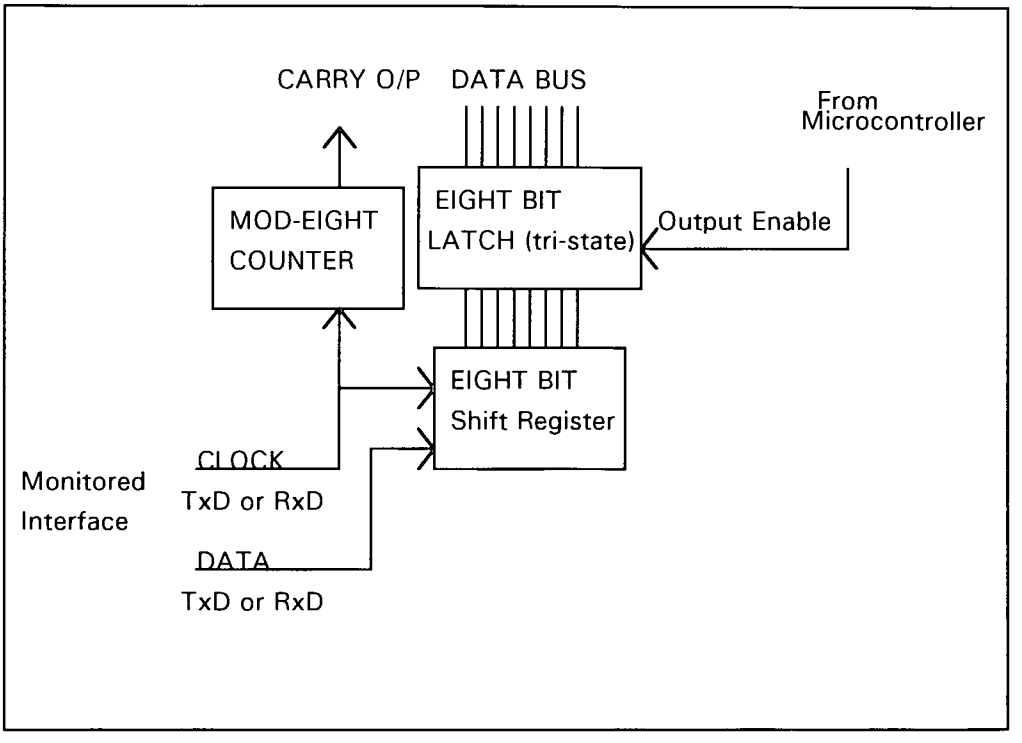


FIGURE 2

The configuration in Figure 2 would have to be duplicated as each of the data paths (namely Transmit Data and Receive Data) with their respective clock pins would require this circuit arrangement. The data would be shifted in to the shift register by the clock on the interface. This will then be converted to parallel and be presented to the latch. The modular-eight counter carry output (overflow) would indicate to the micro-controller that the next eight bits are ready to be read. The micro-controller would then enable the latch and read from it the value of the byte presented to it by the shift

register. The advantage given by this arrangement would be that the micro-controller has access to the actual data on the interface each and every bit thereof. This design tends to be very "clumsy" and inefficient and would result in a considerably higher chip count.

ii) Micro controller's Internal UART Method

Another possible alternative to the monitoring and serial to parallel conversion investigated was the use of the internal UART of the 8751 micro controller. The UART by definition only handles asynchronous data, although if configured in the micro-controller to "Mode Zero" can operate in the synchronous mode at a fixed speed of 1/12 the frequency of the micro-controller oscillator (Intel 1992 b p. 7-14).

One problem with this approach is that since there are two data streams to be monitored, two micro controllers would be needed using this method as opposed to one when using a more efficient method.

Another problem is that the data input/output rate would be restricted to 1/12 of the oscillator frequency. Not only does this actual value (1 MHz) cause a problem in this project but the monitoring device should be externally clocked. In other words the monitoring arrangement must be dependent on and synchronised by the clock from the interface, thereby allowing the data that is present on the monitored interface to be monitored according to its own rate. The monitoring device being externally clocked would provide the essential synchronism in the monitoring process.

iii) High-level Data Link Controller Method

Another alternative method of data monitoring was by the use of an HDLC chip. This would give an advantage of reducing the timing and control requirements of the micro-controller and thereby easing the demands placed on it (Intel (1984) pp. 7-34 - 7-52). However the microcontroller would not be given a true representation of all the data on the interface. The HDLC chip checks for frame integrity using the FCS (Frame Check Sequence) at the end of each frame. Any glitch or bit error would be identified as such by the HDLC chip.

Since the HDLC chip is a complex device and processes the data itself, certain single bit errors run the risk of not being presented to the microcontroller to be sent to the local sub-system. The project monitor should according to the design criteria in 3.2.1. v) perform in a data transparent manner not loosing any bits of information and is not required to perform any protocol management.

The micro-controller possibly not having direct access to every bit change on the monitored interface was the main drawback of using an HDLC chip in the project. This, together with the extra cost of the chip (approximately R127.00 + VAT), the added complexity of its operation and the under utilisation of an extremely powerful protocol link controller resulted in alternative methods of monitoring being sought.

iv) Microcontroller Port Pin Method

Direct monitoring of the interface data using a port pin of the micro-controller was another possible option in the monitoring process. The clock pin would now have to be scanned together with the data pin as there would be no other indication to the micro-controller of the timing of the centre of a data bit. This

would mean scanning approximately three times per data clock pulse in order to locate and not to miss the positive edge of the clock. In other words for each eight bit byte monitored, there would be required at least twenty four samples of the clock signal. To be more accurate the sampling rate and hence load on the microcontroller would be even higher. Remembering that there are two data streams to be monitored then doubles the load placed on the microcontroller. This is a very inefficient method of monitoring and as it would result in an absolute waste of **limited** processor time. This is very obvious when compared to option 3.3.1.1 which externally counts eight bits for the micro-controller, converts them to parallel and notifies the microcontroller when ready. Hence direct monitoring from the microcontroller port pins was not considered a viable solution to the problem.

v) Universal Synchronous/Asynchronous Receiver/Transmitter Method

The best solution found and the one adopted in the project was the one using an external USART, the Intel 8251A, to perform the required monitoring/converting functions.

The 8251A can operate in the synchronous or asynchronous mode and is designed for a wide range of Intel microcomputers. Its functional configuration is programmed by the micro-controller's software and this provides for maximum flexibility. In synchronous mode character synchronization can be internally or externally achieved. When the chip is programmed into external synchronisation mode then synchronisation is achieved by applying a High on SYNDET pin (Intel 1993 p. 2-12).

The chip has a Transmit Data and a Receive Data pin. In order to manipulate the device to monitor the data from the monitored

interface, it's "Receive Data" and "Receive Data Clock" pins would be used. In order to monitor both the Transmit data and Receive data from the interface, two USARTs in a similar arrangement are needed. The interface pin providing the signal element timing (data clocking of the respective data pin) will be connected to the Receive Clock input pin of the USART (RxC). The associated data pin from the interface will be connected to the Receive Data pin of the USART (RxD). The SYNDET pin will be permanently connected High thereby forcing the USART into synchronism at the next Receive data clock pulse (at RxC) (Intel 1993 pp. 2-11 - 2-12). The micro-controller is notified when eight monitored bits have been converted from serial to parallel and are ready to be read by the USART putting a High out on it's Receiver Ready (RxRDY) pin.

In this way the USART would perform the function of transparent data monitoring, serial to parallel conversion, providing indication to the micro-controller when the next eight bits are ready, buffer the information temporarily and interface to the data bus of the micro-controller.

3.3.2 Developing the Chosen method of Monitoring

Once the method of monitoring the synchronous data from the interface was chosen, the proposed configuration was developed on a project board for testing and practical confirmation of the theoretical design.

Fulfilling the Requirements of the USART (Intel 8251A)

The USART uses a separate Clock input (CLK) for internal device timing (Intel 1993 p. 2-3). This clock is not referenced to any other clock connected to the USART. There are however restrictions on the frequency of this clock.

a) CLK input frequency must be greater than 30 times the receiver or transmitter data bit rates.

   A check whether the ALE output from the microcontroller would fulfill the restrictions had to be conducted. Considering a 12 MHz crystal oscillator was to be used to drive the microcontroller, the ALE output from the microcontroller being activated twice per machine cycle (Intel 1992 b p. 5-19) would then be High for a minimum of two clock pulse periods (of the crystal), out of every 12, giving a frequency of 2 MHz. Assuming the data rate of the receiver being 9600 bps, the minimum CLK frequency would then be 30 x 9600 Hz = 288 kHz (0.288 MHz). The ALE would comply with this restriction

b) The minimum and maximum frequencies allowable for the CLK input can be calculated from the clock Tcy specifications of the 8251A (Intel 1993 p. 2-19).

   f_{max} =1/(320 ns) = 3.125 MHz

   f_{min} = 1/(1350 ns) = 740.74 kHz

   The ALE of 2 MHz will fulfill both the upper and lower limits placed on the CLK input to the USART.

c) There is a minimum mark period allowable on the CLK input.

Mark period of the ALE = 2 x the period of the oscillator

= 2 x 1/(12 MHz)

= 2 x 83 ns

= 166 ns

which is greater than the minimum allowed of 120 ns (Intel 1993 p. 2-18).

3.3.3 Monitoring of the Control Interface Signals

By far the most important function in the monitoring of the interface is the monitoring of the Transmit and Receive Data signals. Of secondary importance is the monitoring of the next five interface signals which are referred to in this study as the **Control** interface signals.

			V.24
They are;	Request to send	RTS	Pin 4
	Clear to send	CTS	Pin 5
	Data set ready	DSR	Pin 6
	Data terminal ready	DTR	Pin 20
	Data carrier detect	DCD	Pin 8

At the Remote Monitor these five signals will pass directly through an RS232 line receiver and be permanently connected to five input port lines of the microcontroller.

3.4 Development of the Interface Reproduction circuitry (Central sub-system)

3.4.1 Reproducing the Monitored Interface

First of all, a method had to be found that would convert the data already in parallel (from the micro-controller) to serial format for the interface. This had to be performed while providing the appropriate signal element timing signals. A shift register (74HCT165) would perform the conversion process. This register would be loaded (in parallel) with the next eight bit byte from the microcontroller. This register requires a clock to control the shifting out of the serial byte. The next sub-problem involved developing a method of providing this clock. A free running clock source would pose certain problems with regard to the synchronising of this clock with the data presented to the shift register (see 3.2.3.1 for envisaged timing problems).

This method of providing a clock pulse for each bit of data clocked out from the shift register was developed using a latch circuit for control.

LATCH CIRCUITRY

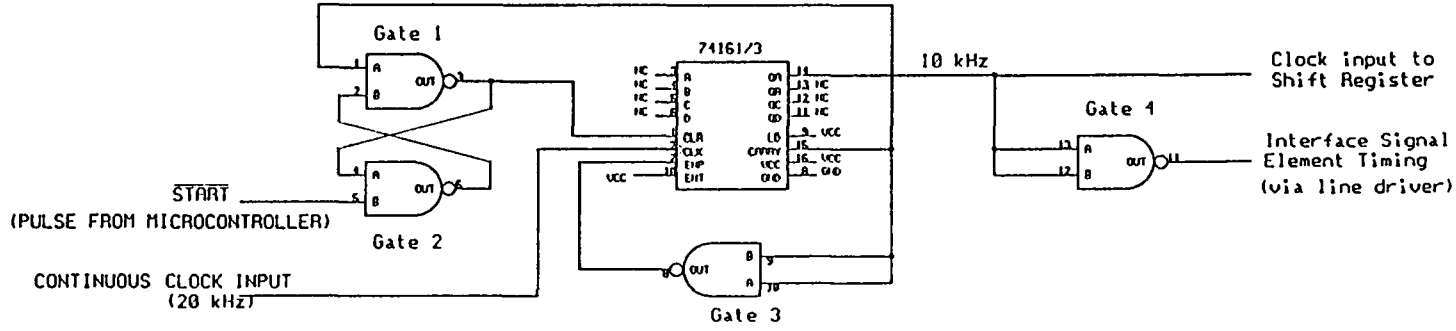
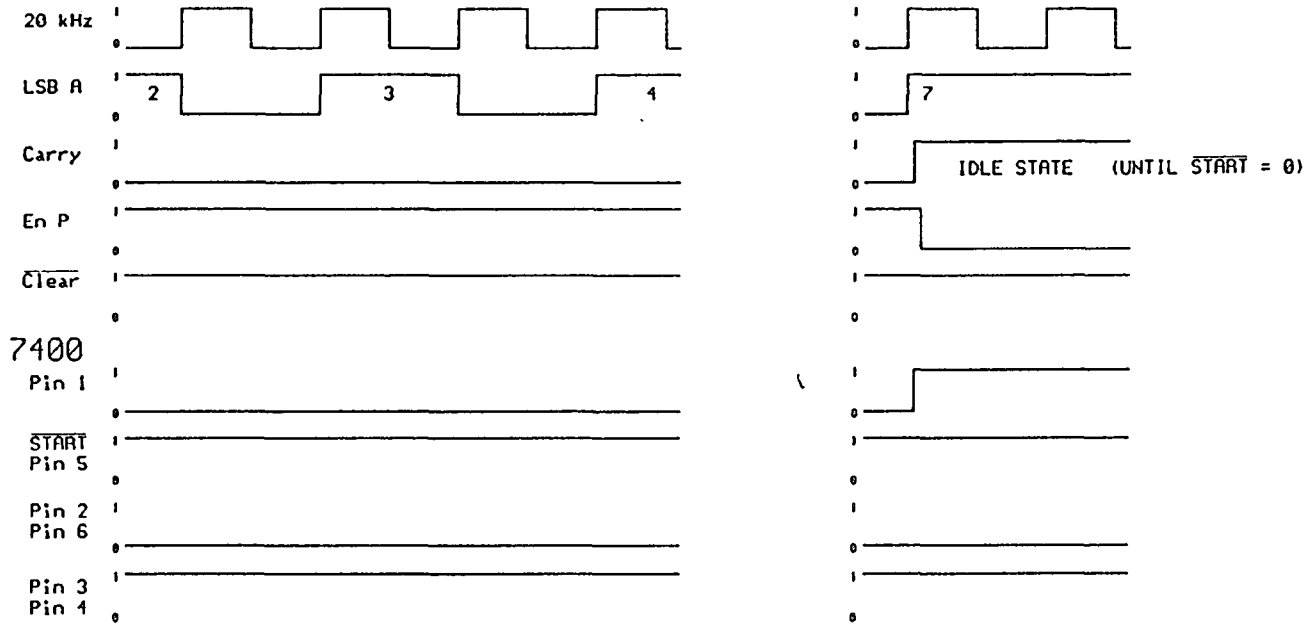


Figure 3a

POWER UP SEQUENCE



The signals output from the project must comply with the CCITT specifications of the V.24 interface for the Chameleon.
These specifications are met by the "Max 235" chip which receives and transmits signals in the RS232 level. This chip has internal voltage doublers and inverters and has no need of any external components (Maxim 1990).

3.4.2 Description of the Latch Circuitry for Data clocking on the Reproduced Interface.

Here follows the description of the operation of the circuitry in **Figure 3a** which defines the latch "**IDLE**" state.
The clock provided to the 74163 is a continuous 20 kHz. This continuous square/rectangular wave is derived from the ALE output from the microcontroller which is then frequency divided by 100.
- On power up;
    - We assume Ripple carry output from the counter (74163/1) to be Low
    - Clear input will then be High (not active)
    - Enable-P input will then also be High
    - Enable-T permanently tied High
    - Start to be High

Under these conditions the counter is enabled and will run to count "15" when Ripple carry will go High.
After this happens the Enable P line will go Low, disabling the counter and stopping it on that count. This change in Ripple carry (on Pin 1) will not affect the latch as a Low was already latched onto the other input of the same gate (Gate 1 Pin 2). The High from Gate 1 on Pin 3, and on Clear input is unaffected. No other

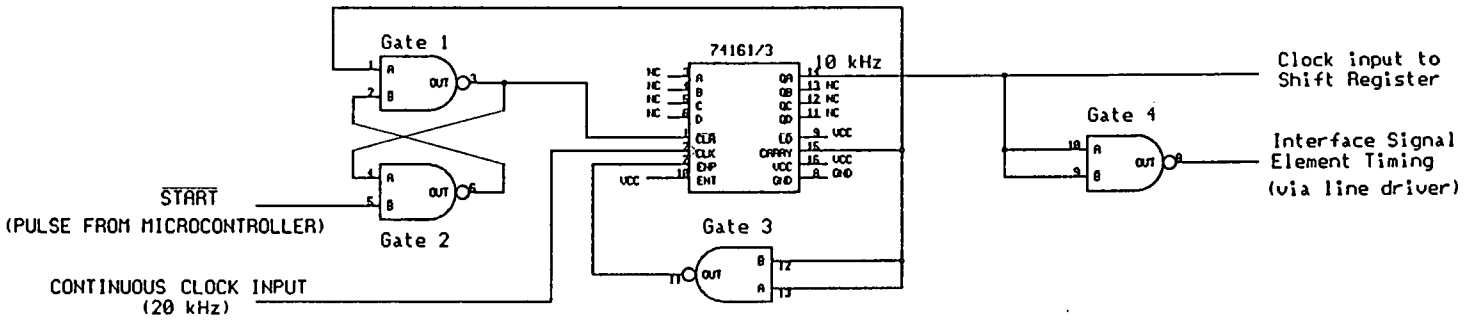
changes will occur in the circuit and all 20 kHz pulses will be ignored.

This stable condition of the clocking circuit will henceforth be referred to as the **"IDLE"** state.

In the "IDLE" state, all the counter outputs Q_A, Q_B, Q_C, and Q_D will be High. The Least Significant Bit (LSB "Q_A") from the counter is used as the clock input for the shift registers.

When a train of eight pulses is again required to clock the next data byte out to the reproduced interface, the microcontroller will start the sequence by it's port pin providing the not START signal.
The sequence of events detailed below use a counter with an asynchronous clear input (74161 see **Figure 3b**).
On the falling edge of this not START signal the following sequence of events will take place;
- Negative edge of not START on Gate 2 pin 5
- Logic High on Gate 2 Pin 6 and Gate 1 Pin 2
- Logic Low on Gate 1 Pin 3 and hence on Clear input of counter
- Counter is cleared and Carry goes Low
- Logic Low on Gate 1 Pin 1 causes High out on Gate 1 Pin 3
- Logic High on Clear input and clear is removed from counter
- Enable P goes High and counter is enabled
- Start can go High again with no effect on the counter latch
(the shortest possible period of a Low on not START is 1 μs [limited by the microcontroller])
- The counter will be clocked by the continuous clock provided at it's clock input until the terminal count (15) is reached, after which it will be in the stable "IDLE" state when carry is High

LATCH CIRCUITRY



LATCH CONTROL OF CLOCKING

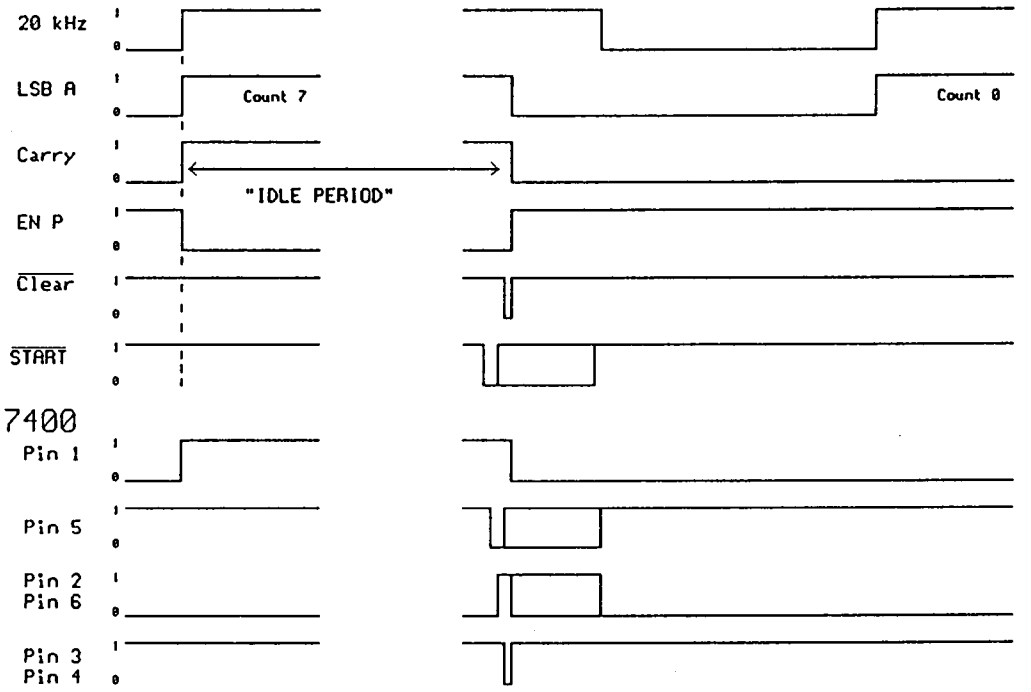


Figure 3b

In this way the counter counts from Zero to Fifteen. The LSB "Q_A" output from the counter will then give one pulse for every two counts of the counter. This will have the effect of a train of eight pulses at 10 kHz (half of the input frequency of 20 kHz).
The microcontroller can also control when a train of eight pulses is started. It must also be stated that the train cannot be restarted unless the counter/latch circuit is in the "IDLE" state.

The clock pulses from LSB "Q_A" will for further discussion be referred to as Clock A.
On the positive (rising) edge of Clock A the next bit is shifted out of the register onto the reproduced interface. In other words the positive edge of "Clock A" indicates the change of the data bit. According to the V.24 interface specifications (CCITT 1985 b p. 105) the positive edge of the data clock (on the interface) should indicate the change of the data bit. From this we arrive at the conclusion that we should use the inverted Clock A signal for the data clock on the interface (V.24 circuit 114 and circuit 115) because the signal will again be inverted when passing through the RS232 line driver. Since the circuit is duplicated in as much as the shift registers are concerned both data streams, from channel 1 and channel 2 (Transmit Data and Receive Data) will be clocked to the interface simultaneously, using the same clock. The RS232 specifications of one driver to one receiver will be met and hence the signal is split at the TTL level and driven by two separate drivers.

The above operation was however slightly modified during most of the development of the project as the integrated circuit 74HCT161 was unavailable. The next best, the 74HCT163 was used in it's

place. The difference between the two is that the 74163 is a fully synchronous eight bit counter, whereas the 74161 has an asynchronous clear input. The difference between the two had a substantial effect as far as clearing after the latching of the start pulse from the microcontroller is concerned. When the start pulse goes out from the microcontroller, it is latched almost immediately (a few nano-seconds later) and enables the counter to run (driven by the continuous 20 kHz). However if a synchronous counter such as the 74163 is used then the clearing of the counter only takes place after the first positive edge of the continuous clock at it's clock input (20 kHz). In other words, using a synchronous clear counter would cause a delay from the start pulse to the actual clearing of the counter and beginning of the 8 clock pulses on Clock A

Further development was done with this counter until when a 74161 could be located. As will be seen in the programs, the micro-controller waited for the counter to actually clear before continuing. This was done in order to avoid reloading the shift-registers and overwriting the bytes that had just been loaded into the registers. This was a possibility if the micro-controller was to check the carry from the counter and seeing it High could assume that the last pulse train was finished when in actual fact it had not yet started (because the carry had not reset **immediately** when the START signal was given to the latch {74163}). This delay was worked into the software system so as to minimize the delay of the program flow.

3.4.3 Clocking of Continuous Bytes to the Interface.

i) Counter latched in the "Idle" State

When the ripple carry goes High, the nand inverter connected to the Enable-P input will disable any further counts on the 74161/3.

ii) Shift Registers Loaded

The shift registers are loaded with the next 8 bits to be output to the interface. The operation of the register (74165) is such that once loaded the LSB (least significant bit) is already present at the serial output (Q_H).

iii) Clocking of the Data onto the Interface

After the shift registers are loaded the microcontroller can start the next pulse train which will clock the data bits out of the register at the required frequency (10 kHz). In order to start the train the "START" signal, from one of the micro controller's port pins, will go low. This will start the counter in it's count sequence by clearing the chip count (and Carry) and enabling it to be driven by the continuous 20 kHz in the manner described in 3.4.2.

Since the first bit was already at the output of the shift register, it would also appear on the interface (although having passed through the RS232 line drivers). Before the Start signal is given to the latch, the counter is in the "IDLE" state. This means that the Clock A (LSB "Q_A") will be a High. This High changes to a low immediately after the negative edge of the Start signal. This negative edge of Clock A is inverted by the NAND gate and fed to the RS232 line drivers onto the interface. This then will according to the V.24 specification be the correct logical transition (negative edge) of the clock signal to indicate the centre of the data bit on the interface. The same edge being directly fed from the Clock A will

have no effect on the shift register as it is positive edge triggered. When the counter produces the next edge at LSB "Q_A" it will be a positive edge. This edge will shift the next data bit to the serial output of the shift register and at the same time make it appear on the interface. The inverted Clock A appearing on the interface (now a positive edge) will have no effect on the interface. The next transition of LSB "Q_A" has no effect on the shift register but will indicate to the interface that this is the centre of the next data bit. The counter will continue to alternatively shift and clock the eight data bits from the register to the interface at a frequency of 10 kHz or 10 kbps.
(Texas 1985 p.7-182)

3.4.4 Identification and Elimination of Possible Problems

Once this arrangement had been designed on paper the components were purchased and the circuit was constructed on breadboard.

What needed to be confirmed next was the success of the circuit practically. A short test program was written to assist in the test operation. The serial output from the shift register was supplied to an RS232 line driver and presented to a data analyser (Interview 40A). The main function of the test was to prove that the circuit arrangement was successful in its task of parallel to serial conversion.

3.4.4.1 Random Byte Boundaries

A problem envisaged with the presentation of the data to the interface (ultimately to the protocol analyser) was that the random boundaries created (by the remote sub-system) in the monitoring

process at the end of each group of eight monitored bits, would not be tolerated by the protocol analyser. This was because the clock signal in actual fact would not have a mark to space ratio of 1:1. This is illustrated in Figure 3b by the clock of LSB A. This signal referred to above as Clock A will have a 1:1 mark to space ratio for all of the pulse train except for the "IDLE" period (between count 7 and 0). This is a variable period which is determined by the availability of more data bits in RAM to be clocked to the interface.

A cyclic test program, TEST 1, was written to investigate this method.
This test and the results thereof are documented in section 3.5.1.

3.5 Development and Test Results

3.5.1 Testing of the Parallel to Synchronous Serial Conversion Circuitry.

This test used the circuit illustrated by Figure 4 (on bread board) and the flow chart of the program TEST1 is found in Figure 5.1 and 5.2.
The main objectives of program TEST 1 are:

a) to ascertain whether there would be any **discrepancy with the manner in which the data clock was controlled** in the serial outputting of the data,
b) to provide externally controlled program branches to increase testing flexibility.

The built-in flexibility provided operator control of the program to branch and thereby control the blocks of bytes output to the interface. This control was all via the logic conditions on a few port pins. This saved time by not having to write an updated test program once the previous test was confirmed. Much time is wasted in accessing a UV eraser and reprogramming the ROM with small variations on a test program. Firstly tests were performed with the **Tekelec Interview 40 A, Data Analyser.** It was required that the display of the analyser be in Hexadecimal format for this test. For this reason the Chameleon was not initially used as it does not have the facility of only displaying the Hexadecimal characters received from the monitored interface (Instead it displays the more advanced interpretation of the X.25 protocol packets in ASCII format). The Chameleon is also a very well used piece of equipment, making it difficult to the find time when it is not being used by the maintenance staff.

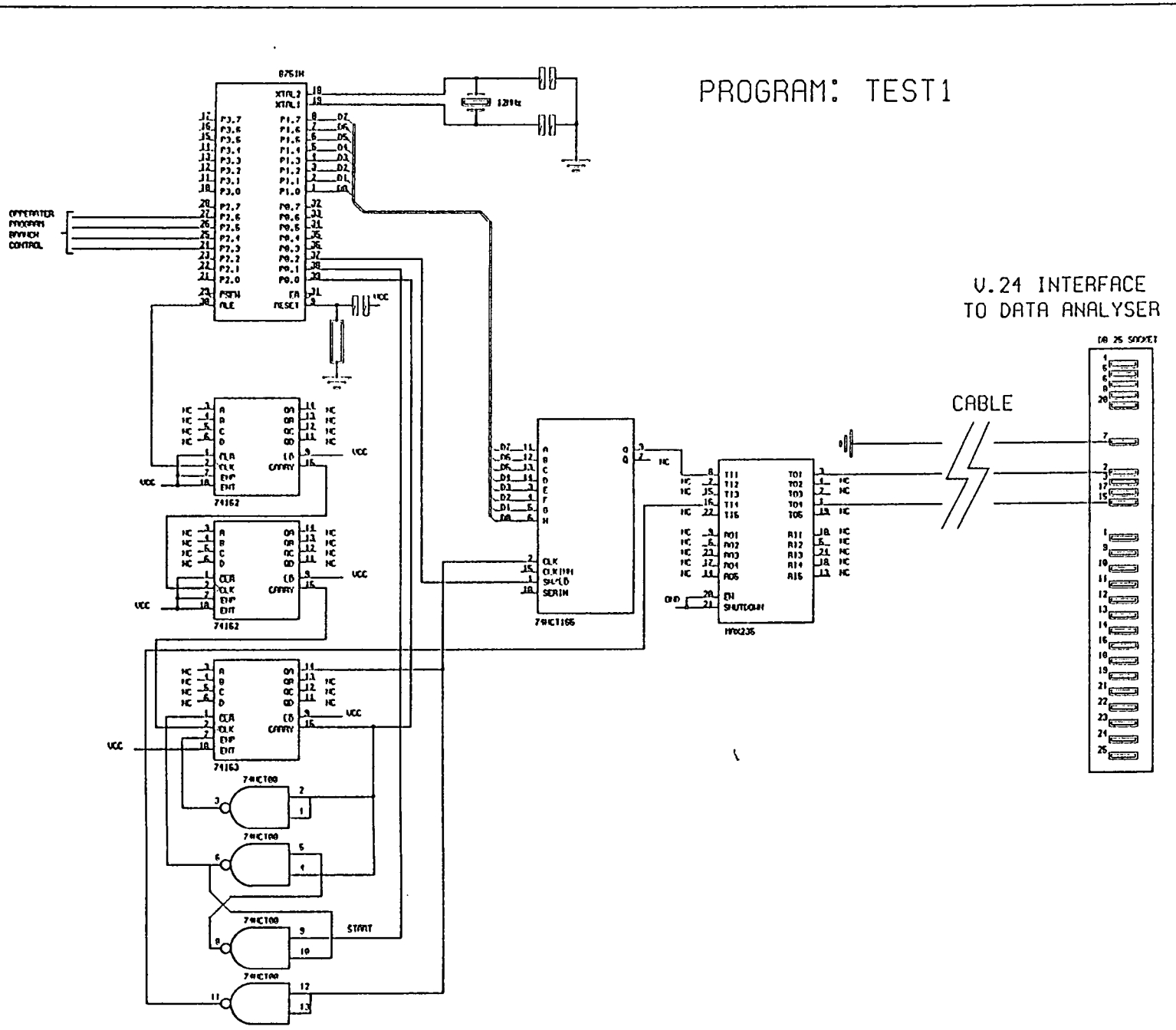


FIGURE 4

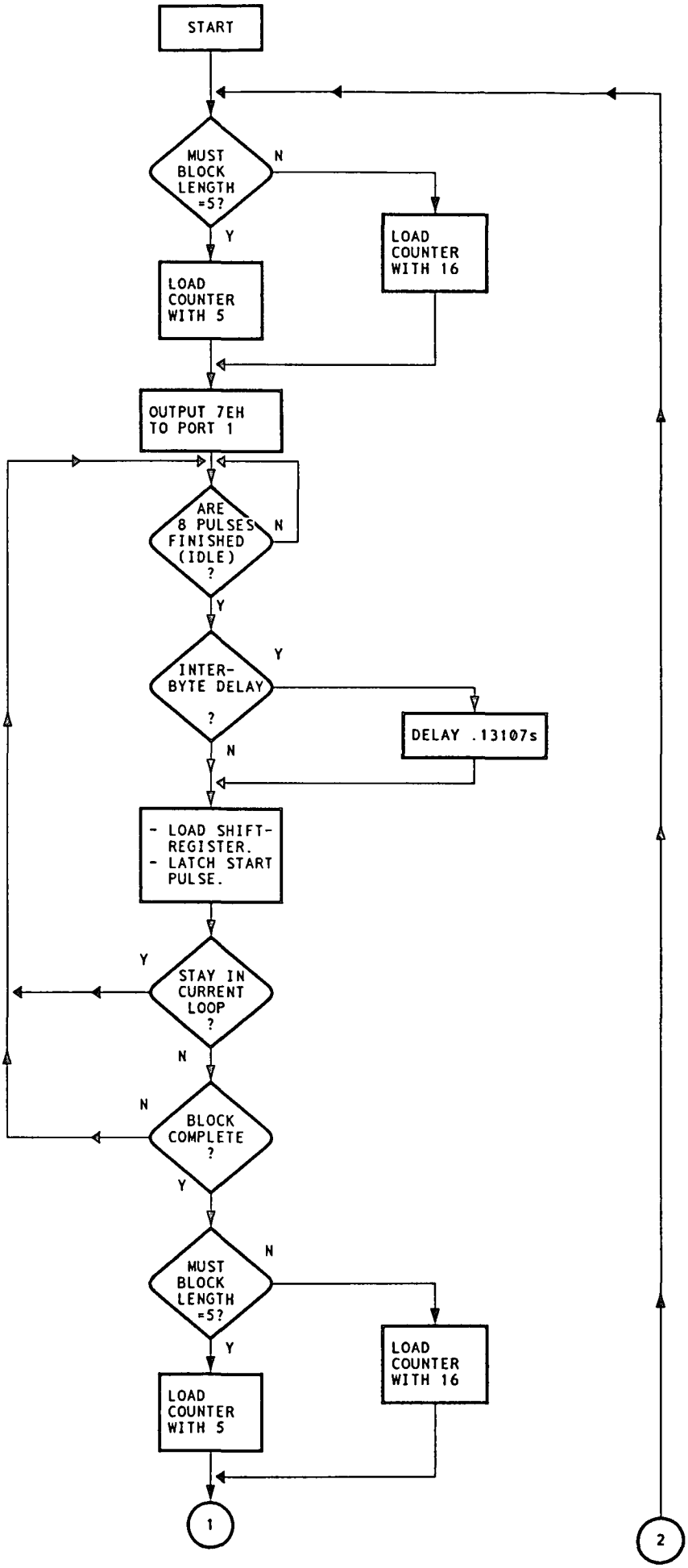

START
MUST BLOCK LENGTH =5?
N
Y
LOAD COUNTER WITH 16
LOAD COUNTER WITH 5
OUTPUT 7EH TO PORT 1
ARE 8 PULSES FINISHED (IDLE) ?
N
Y
INTER-BYTE DELAY ?
Y
DELAY .13107s
N
- LOAD SHIFT-REGISTER.
- LATCH START PULSE.
STAY IN CURRENT LOOP ?
Y
N
BLOCK COMPLETE ?
N
Y
MUST BLOCK LENGTH =5?
N
Y
LOAD COUNTER WITH 16
LOAD COUNTER WITH 5
1
2

FIGURE 5.1

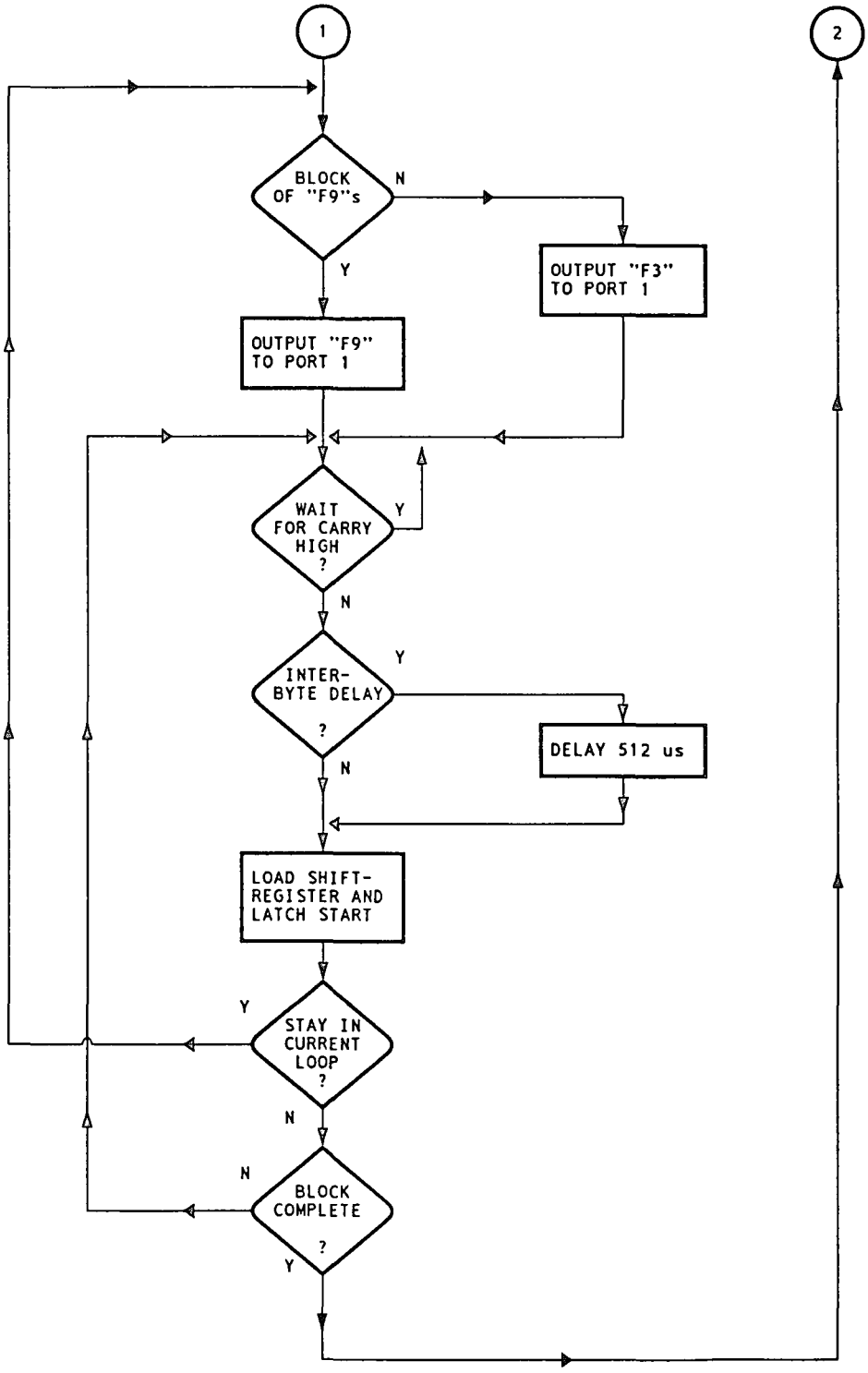

1
2
BLOCK OF "F9"s
N
Y
OUTPUT "F3" TO PORT 1
OUTPUT "F9" TO PORT 1
WAIT FOR CARRY HIGH ?
Y
N
INTER-BYTE DELAY ?
Y
N
DELAY 512 us
LOAD SHIFT-REGISTER AND LATCH START
STAY IN CURRENT LOOP ?
Y
N
BLOCK COMPLETE ?
N
Y

FIGURE 5.2

The Interview 40A Data Analyser is capable of monitoring and testing various data protocols operating on a DB 25 pin cable connector, V.24 interface. It was thought that if this analyser did not fail as a result of the clocking method used, then there would be a reasonable chance of the "Chameleon" or any other analyser also accepting the clocking from the project.

When the circuit was powered up the first time, the bytes were not displayed by the data analyser. Upon investigation with a logic probe, no clock activity was found at the LSB output from counter. After the circuit was thoroughly checked, it was decided to an ICE to assist in finding the fault. The problem manifested itself at the point where the start pulse is sent to the latch. The counter clear was changing state repeatedly at a high frequency for a short duration after the latch pulse was sent. It then became apparent that the differences between ENP and ENT lines of the counter are substantial (Texas 1985 pp. 7-177 - 7-191). The permanent high should have been connected to the ENT and not the ENP line. This was corrected and the following procedure was followed;

i)

Firstly the analyser was setup in the "synchronous data" monitoring mode with double "sync" characters where the "sync" character was set to 7EH. The character length set to 8 bits with no parity and external clocking. The external program control via the microcontroller port pins was set as to cause the program to send a continuous stream of 7EHs to be clocked from the shift-register, with no inter-byte delays involved. The analyser displayed the received bytes, 7EH without any problem.

ii)

The next task was to prove that each one of the bytes was accounted for and that none were missed or left out from the

display because of some or other timing error. The program was then allowed to output the alternate blocks of 7EHs and F9Hs. The screen display of the Interview 40A was examined over a period of 5 minutes and all transmitted bytes were accounted for. To confirm this however, the block length was changed (using the external program control) from 6, to 16 continuous bytes and no errors were found. Finally to prove without a doubt that one could change the bytes at random the F9H block was changed to F3Hs (via external pin P2.7) and the Data analyser remained in sync, displaying all expected outputted bytes.

The data analyser has a facility by which the user can enhance certain bytes on the display. In other words if the user would like the byte "7EH" highlighted (or enhanced) to make it easier to visually distinguish between sync characters and any other characters (possibly data bytes in a frame of data) he would set the analyser to enhance the character "7EH." This facility was used in the test and the byte constituting the one block was enhanced. The second block now was seen as data packets or frames in-between the sync characters of a synchronous data stream. The data bytes in this block were interpreted correctly by the data analyser and there were no errors in this regard.

Finally all the possible combinations of data bytes with different sync-character settings were tested and all tests performed were successful in this regard.

iii)

The next test was to see if the receive clocking of the analyser would accept bytes that have their bits separated by one small irregular clock pulse (produced between the bytes being output by the register/latch circuitry during the "IDLE" period).

This test was done by clocking the same byte continuously out from the shift register (7EH), but now with the data analyser having been setup with the sync character equal to F9H. The display indicated that the bytes being received were F9Hs although it was not receiving F9Hs but in fact 7EHs (see Figure 6). This test was successful too, showing that the receiver synchronizes into the data received and shifted it's own interpretation of where the byte borders ought to be. This test also confirmed that there is no problem with an irregular pulse width in the bits received. The **inter-byte delay** (0.131072s) was introduced under these test conditions with no disturbing effect apart from a slower scrolling of received data on the display as in the previous test. This further confirmed that the irregular clock pulse in the character did not concern the Analyser.

The **results** of the tests (i, ii and iii above) provided by this program, the circuit on project board together with the data analyser proved that the **analyser (Interview 40A) would only sample the data (on the V.24 interface) on the negative edge of the clock irrespective of when that edge arrived.** This result was considered positive to the development of the project, as it meant that it was probable that the Chameleon would also accept the same clocking method designed in the project, because it would probably have the same interface tolerances. This however still had to be proved through experimentation.

The only problem envisaged with the clocking tolerance of the Chameleon was that there could possibly be a sophisticated clock frequency monitor inside the Chameleon which would be a watchdog on the clocks (Transmit and Receive data). This would possibly monitor for irregularities and sudden changes in either

frequency or pulse length. It was expected that this function if existent would make use of a **Phase Locked Loop (PLL).** If this was the case, this internal circuit would pick up the irregular clock pules provided by the project (on the reproduced interface's data clock pins) and possibly cause a rejection of the data presented to the interface. Had this been the case, the clocking method proposed in the project would not be acceptable and either another method of clocking would need to be developed, or more refinement and modification would have to be made to the developed method.

This PLL monitoring of the clock was a more probable problem with the Chameleon than with the other data analysers because of the more specialised and advanced facilities provided by it. The operation manuals were studied at length and no specific information could be found save that there is a LED display indicating the reception or transmission of clocks (Tekelec 1990, p. E-1).

More information was then requested from the supplier in Johannesburg, Altech Instruments Ltd. (Randlehoff 1993).

A reply facsimile message (Bonde 1993) indicated that the synchronous receiver in the Chameleon recovers the clock and **does not monitor frequency**, nor any such facet which could pose a problem to the project, provided that the Data was in synchronism with the clock edge (according to the CCITT V.24 interface specifications).

The facsimile also indicated that there is **no phase locked loop monitor** of any sort on the Chameleon's V.24 interface.

Below in Figure 6 the results of one of the tests clearly indicate

how the receiver synchronised into the "shifted" data. In this particular test the analyser was set for "sync" character of 7EH and was therefore hunting for 7EHs but was only receiving F9Hs. The receiver hunted for the 7EHs and synchronised to the different character borders. The program was then externally controlled to no longer only output F9Hs but alternating blocks of F9H and 7EH. Because the character border of the receiver had been shifted, the block of 7EHs was not displayed as 7EHs but interpreted as shown in Figure 6 below. When interpreting this diagram one must bear in mind that the data is read from left to right and all bytes are transmitted with their Least Significant Bit (LSB) first.

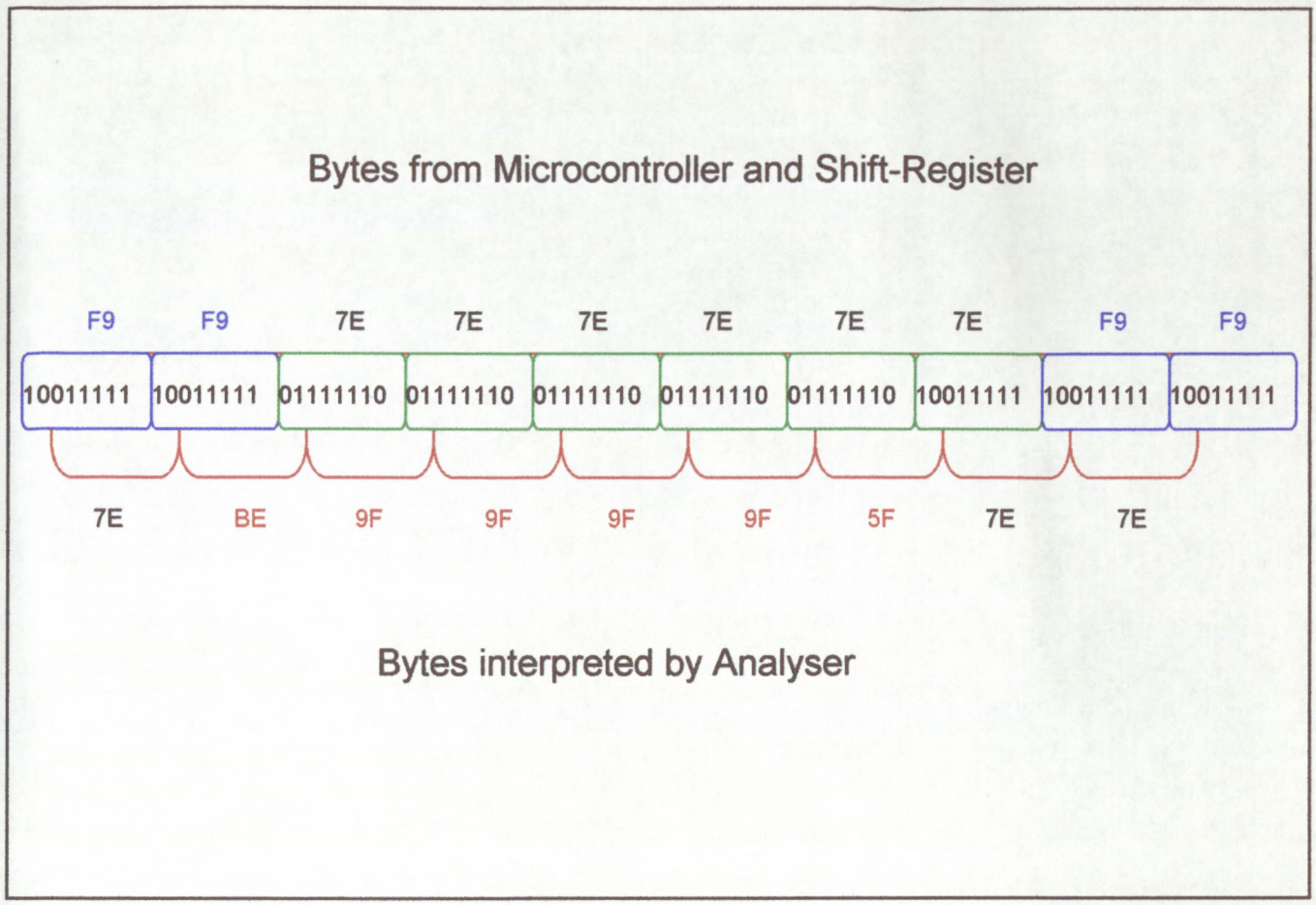

Bytes from Microcontroller and Shift-Register
F9
F9
7E
7E
7E
7E
7E
7E
F9
F9
10011111
10011111
01111110
01111110
01111110
01111110
01111110
10011111
10011111
10011111
7E
BE
9F
9F
9F
9F
5F
7E
7E
Bytes interpreted by Analyser

Figure 6

After using the Logic Analyser and the Interview 40 A and confirming this whole test with the Interview 80 it was concluded that the clocking method was satisfactory into the interface, but the data of a true X.25 packet would be needed to test these tolerances on the Chameleon. This is dealt with in 3.5.3 (TEST 3)

3.5.2 Testing of the Serial to Parallel Conversion Circuitry.

The main objectives of this test (Program TEST2) are to confirm:

a) that the USART initialisation sequence was correct,
b) that the USART could successfully be used by the project for serial to parallel conversion.

The first test run on the serial to parallel conversion using a USART was to output a serial synchronous byte and to receive that same byte using the USART. The serial byte was output using the pre-tested method above (3.5.1 using the shift-register and latch, explained in 3.4.2). The data from the serial output of the shift-register was connected to the Receive Data (RxD) pin of the USART. The clocking generated by the 4 bit binary counter and controlled by the latch/carry arrangement was connected to the Receive Clock (RxC) pin.
The circuit (on project board) used in this test is shown by Figure 7. It was not necessary to make use of any line drivers in the circuit as all signals remained on the same electrical level (TTL). The flow chart associated with program TEST 2 can be found in Figure 8. The shift-register was loaded with the byte in parallel and clocked out serially. This byte was then received by the USART on the same project board. The status byte of the USART was repeatedly

read and when the RxRDY bit was set, the data byte was read from the USART. This byte was then output to Port 1. The program would wait at this point for a signal on Port 0 pin 0 (P0.0) for an indication to continue with the next byte. In the mean time the byte at Port 1 was checked with a logic probe and recorded, and then compared with the byte output to the shift-registers. The cycle would run again from this point once the signal was given by means of a logic High on P0.0. The way in which track could be kept was that the byte was incremented each time it was output.

This test was successful and proved that the USART can be used for the purpose of serial to parallel conversion, with no regard to the sync characters. This test also confirmed that the initialisation sequence performed on the USART was correct for it's manipulation to perform this specific monitoring function.

PROGRAM: TEST 2

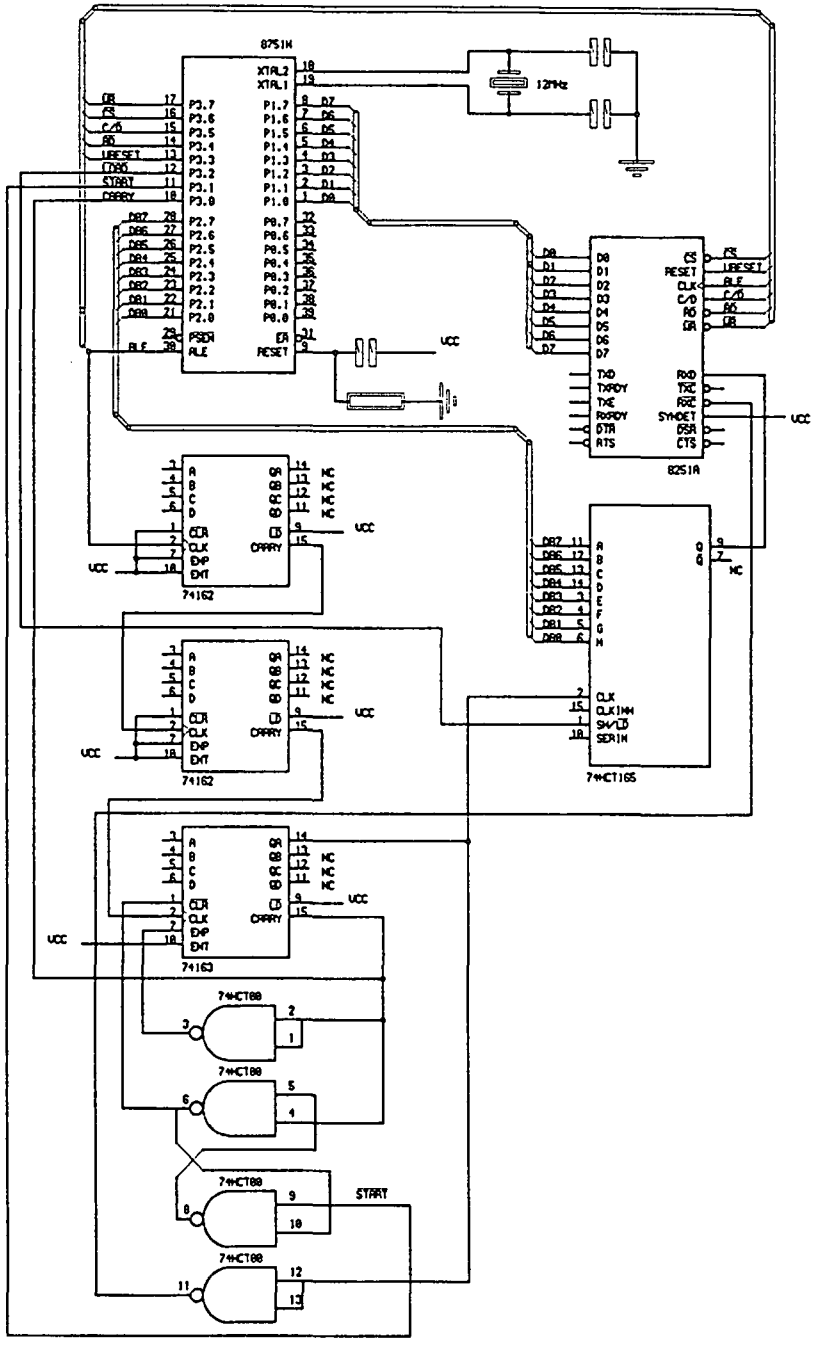


FIGURE 7

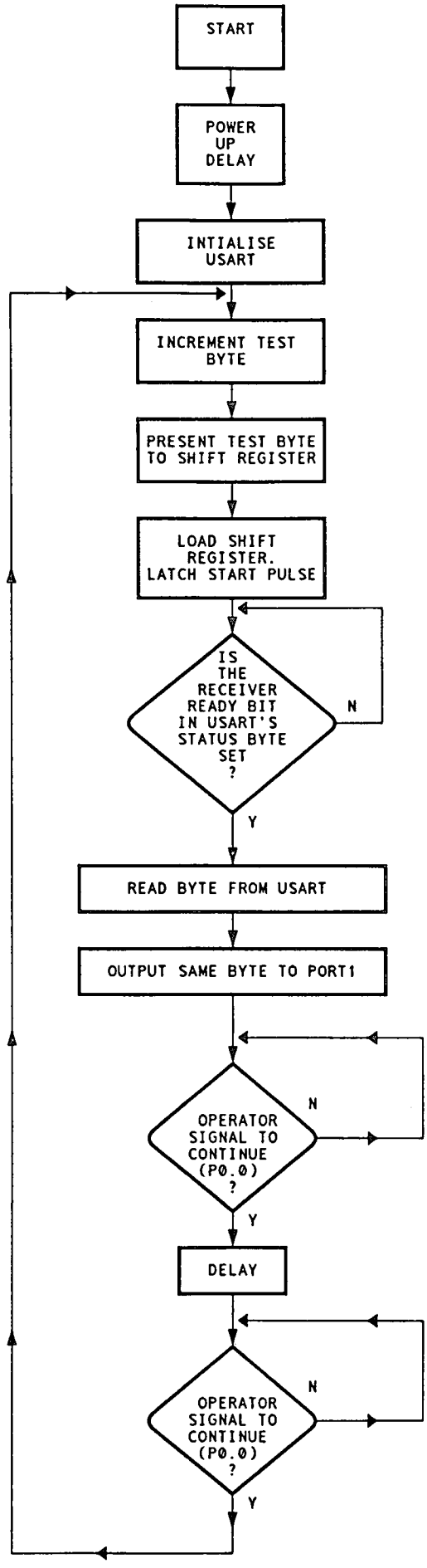

START
POWER UP DELAY
INTIALISE USART
INCREMENT TEST BYTE
PRESENT TEST BYTE TO SHIFT REGISTER
LOAD SHIFT REGISTER. LATCH START PULSE
IS THE RECEIVER READY BIT IN USART'S STATUS BYTE SET ?
N
Y
READ BYTE FROM USART
OUTPUT SAME BYTE TO PORT1
OPERATOR SIGNAL TO CONTINUE (P0.0) ?
N
Y
DELAY
OPERATOR SIGNAL TO CONTINUE (P0.0) ?
N
Y

FIGURE 8

3.5.3 Testing of Both Conversion Circuits with the Chameleon.

(Program TEST 3)

To perform a conclusive test in order to prove that there would be absolutely no rejection of the monitoring, nor the method of reproduction of the interface signals a test configuration outlined in Figure 9 was designed. The project board circuit schematic can be found in Figure 10.

The main objectives of this test were:

a) to investigate the tolerance of the Chameleon to the clocking presented by the shift-register/latch configuration,

b) to monitor a live interface (one operating at 9600 bps and using the X.25 data protocol).

This arrangement would create an environment with conditions very close to the actual operating conditions for which the circuit was ultimately designed. Testing in this way not only would prove that the clocking was acceptable to the Chameleon but would also allow for a lengthy test to be performed where any intermittent discrepancy, problem or glitch would be revealed.

The chameleon, when used normally to monitor a data interface operating on CCITT X.25 protocol, displays the information traveling in both directions (Transmit and Receive) simultaneously. The screen (a cathode ray tube type) is split into two halves (left and right) by an invisible vertical line down the center. This attribute of the Chameleon was used to best advantage in this test as the one half of the screen would display the pure data from the Transmit Data pin (and it's associated timing as indicated in Figure 9), while the other half of the screen would display the same data,

after having been processed through the project. This data from the project had been monitored and read into the microcontroller, then presented to the shift-register and clocked out according to the developed clocking method under latch control.

It was not considered to be essential to monitor more than the one channel in this test in order to investigate the tolerance of the Chameleon to the clock philosophy involved in the project.

Access to the signals on the monitored interface was made by means of a breakout box. All interface signals from the patch panel side of the interface were disconnected, with the exception of a) signal ground, b) transmit data and c) it's timing signal (transmit data signal element timing). The same signals were then tapped off (connected) to the monitoring system developed thus far, i.e., the serial to parallel conversion using the USART.

The program TEST 3 (with flow chart in Figure 11) caused the bytes monitored by the USART to be read into the micro controller, and then put out to the shift register/clocking circuitry in order to reproduce the serial data of the monitored pin. The reproduced signals (data and associated clocking) were now physically connected to the breakout box on the Chameleon side as Receive Data and it's timing signal (Figure 9). The purpose for this was that now both the original and reproduced X.25 data signals (of the data in only one direction on the monitored interface) could simultaneously be seen on the split screen of the Chameleon. This test was run successfully for 2 hours. The X.25 data frames interpreted and displayed by the chameleon were identical on both halves of the screen. The success of this test proved that the Chameleon had tolerance to the clocking method used by the project, particularly when monitoring an X.25 interface (bit oriented protocol).

The monitored interface provided the Receive Data, the Receive Data Clock and the Signal Ground to the monitoring circuitry as shown below in Figure 9.

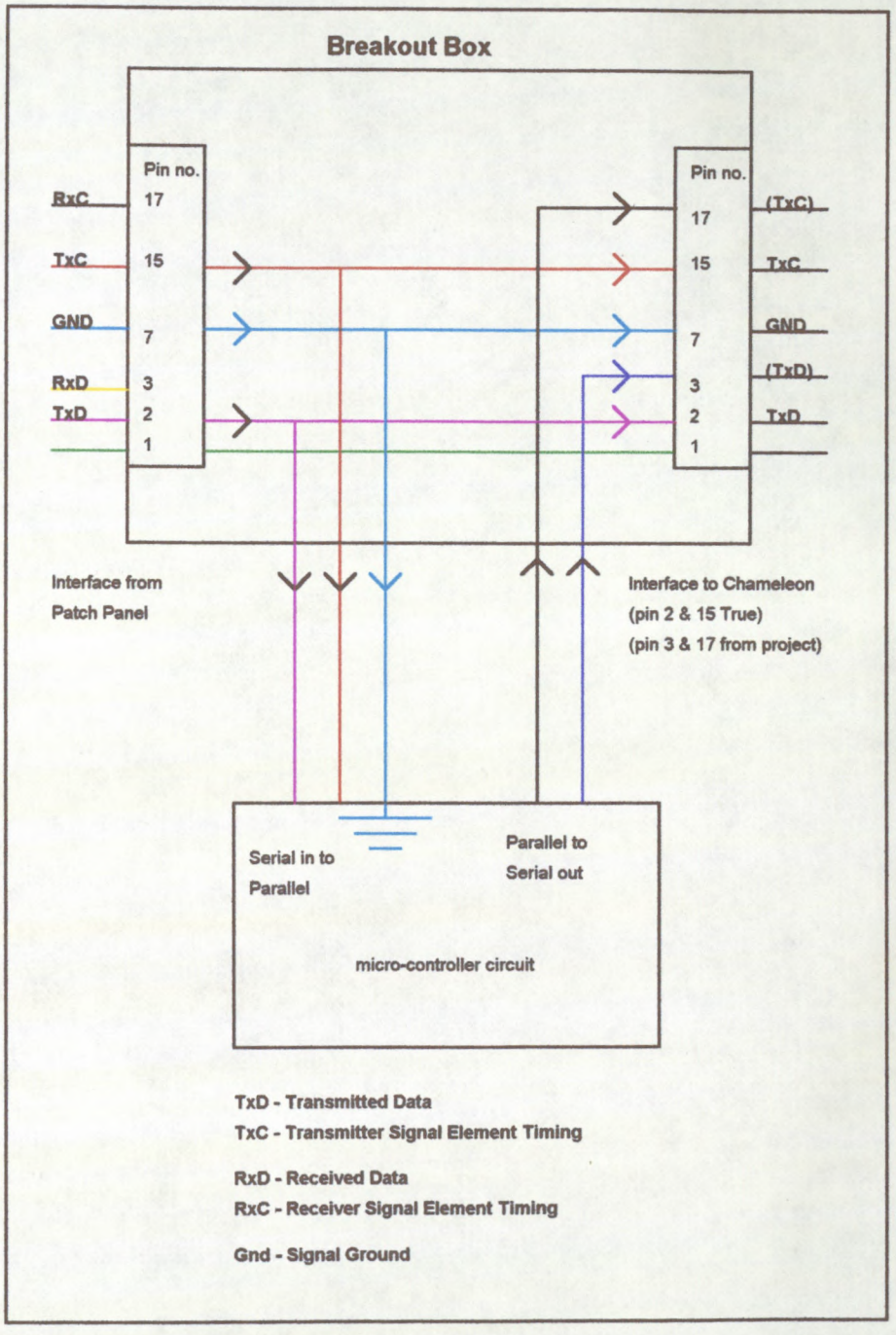
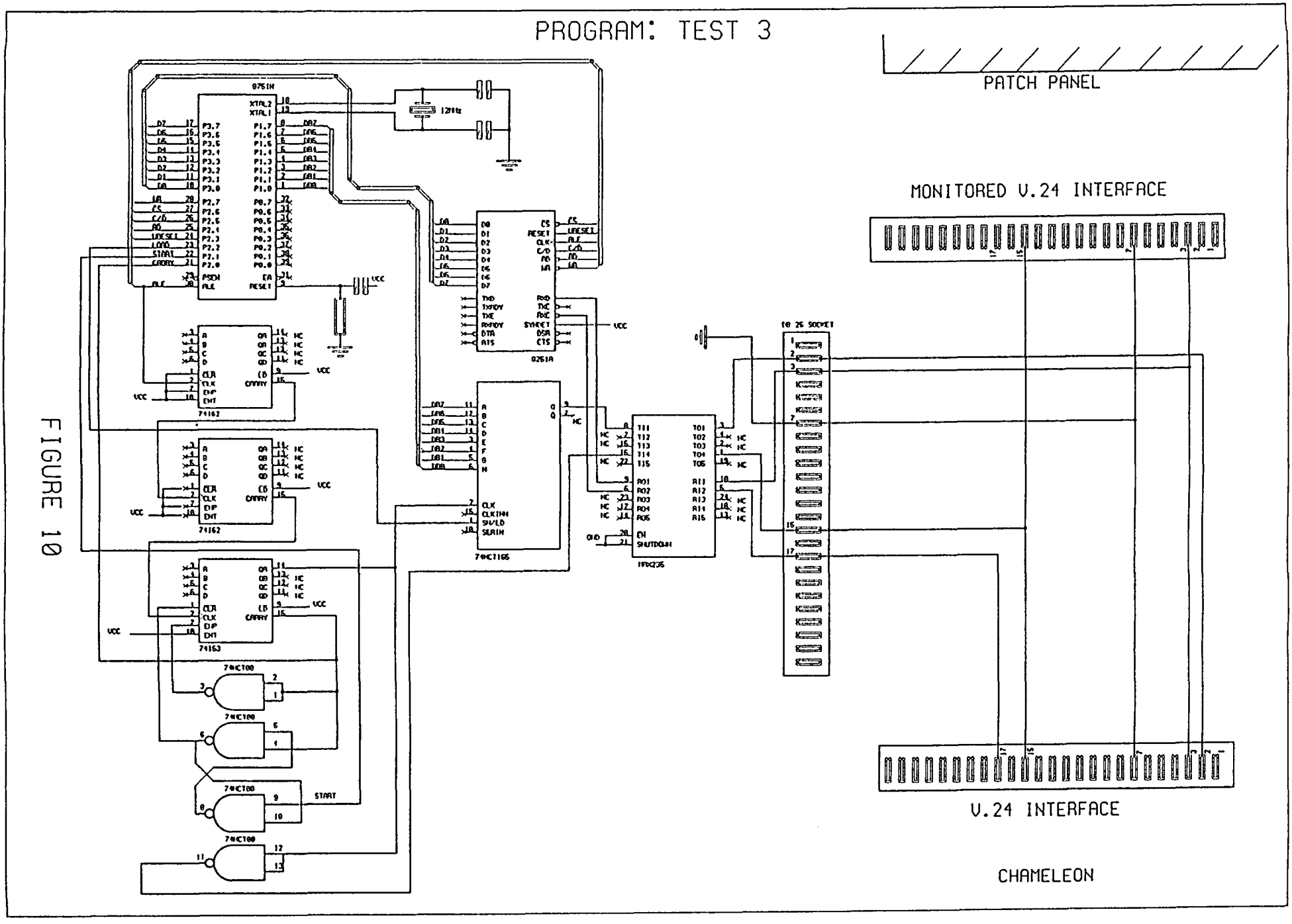


Figure 9

The test program, after reading the monitored byte from the USART, presented the same byte to the shift-register. When the latch/clocking circuitry reached the IDLE state the byte was loaded into the register and the start pulse given to the latch. The program waited there until the clocks started again and the carry reset. The cycle then repeated from polling the USART onwards.


PROGRAM: TEST 3
PATCH PANEL
MONITORED V.24 INTERFACE
V.24 INTERFACE
CHAMELEON
FIGURE 10
8751H
74162
74163
74HCT165
74HCT00
START
TO 25 SOCKET

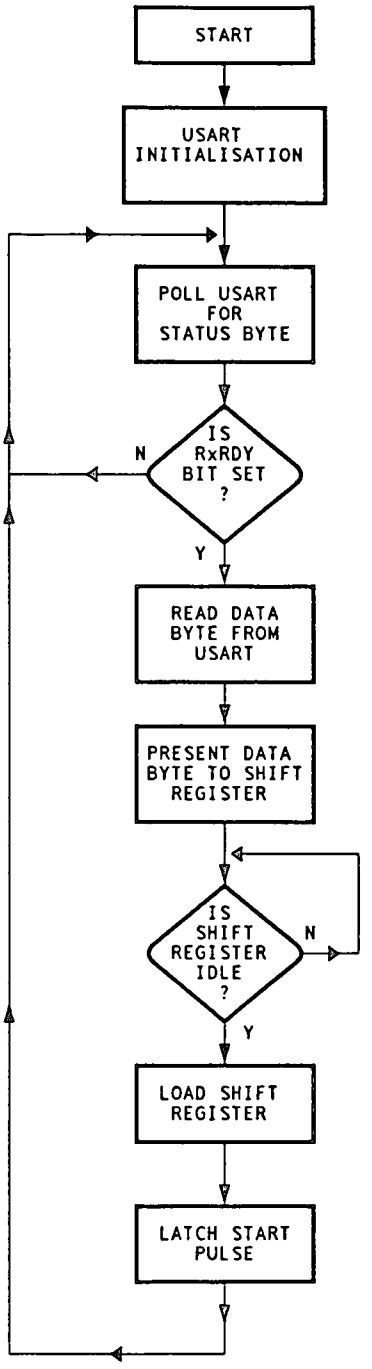

START
USART INITIALISATION
POLL USART FOR STATUS BYTE
IS RxRDY BIT SET ?
N
Y
READ DATA BYTE FROM USART
PRESENT DATA BYTE TO SHIFT REGISTER
IS SHIFT REGISTER IDLE ?
N
Y
LOAD SHIFT REGISTER
LATCH START PULSE

FIGURE 11

3.6 Testing of the Communications Link Control

3.6.1 Preliminary Test of USART Control over Communications Link

USART Intialisation

A critical part of any serial communications link is the initialisation of the link controlling device. The USART in this case the link controller, after being powered up may be placed in the Mode, Sync character or Command format. Before any commands are given to the device, it is recommended that the USART definitely be placed in the Command instruction format. It is recommended that the worst-case initialisation sequence be used. This would mean the USART being in Sync (synchronous) mode with two sync characters.

This initialisation sequence can easily be executed by writing three 00Hs to the USART with control/not data line held High. This would configure the USART into Sync mode with two dummy 00H sync characters. Now an Internal Reset command sent to the USART will be accepted as such by the USART, placing it into the "idle" mode. Once in the "idle" mode it is ready for the initialisation sequence for the particular communications application.

For the purpose of this project a synchronous communications link is used. It was decided to use double characters instead of single and the character being BCH.

The control words are split into two formats:

A) Mode Instruction

B) Command Instruction

A) The mode instruction byte sent to the USART was 0CH

This instructed the USART that;

a) double sync characters are to be used

b) SYNDET (sync detect) is an output

c) no parity is used

d) the character length is 8 bits (being a more time and control efficient option)

The sync characters (BCH) follow the Mode instruction to the USART, when initialising it in the synchronous mode.

B) The Command instruction byte used was A7H which formatted the USART for;

a) entering Hunt mode

b) permanently active Request To Send

c) receive enable

d) transmit enable

e) permanently active Data Terminal Ready

The Test (Program TEST4C and TEST4R)

Although the cables and the NTU's were tested prior to connecting to the project, the first test with regards the communications link was to confirm the following:

-true and straight connection of the interface cables and connectors

-correct polarity of the RS-422 line drivers and receivers

-correct programming and initialisation of the USARTs

-faultless Network Terminating Units (digital "modems")

-good line (4 wire) connection between the NTU's

-wiring of project board

-correct software control of the communications link

The test, using programs TEST 4C and TEST 4R, was successful and hence confirmed all of the above. The test involved transmission of a pair of two alternating bytes which when received at the other end of the line were read from the USART and displayed at a spare port of the microcontroller. The functioning of the system could easily be determined using a logic probe at the port pins. At the central system the bytes B8H and FAH were transmitted and the bytes 7EH and F8H were received from the remote. For a successful test the following conditions are expected at the display ports of the systems' micro controllers;

Central Micro-controller on Port 1

7EH 0111 1110 b

F8H 1111 1000 b

bit 0 of port remains LOW

bits 3,4,5 and 6 of port remain HIGH

bits 1,2 and 7 alternate.

Remote Micro-controller on Port 2

B8H 1011 1000 b

FAH 1111 1010 b

bits 0 and 2 of port remain LOW

bits 3,4,5 and 7 of port remain HIGH

bits 1 and 6 alternate.

With the Logic Probe the bytes at the spare port were checked according to the received bytes that were expected. This could be confirmed with a more thorough test however, by the use of a logic analyser placed at the Transmit Data pin of the USART and at the Received Data pin of the USART of the other sub-system. In this way the data leaving the one USART before reaching the line drivers was checked and compared to the data arriving at the USART of the other sub-system after the signals had passed through the other line driver, cable, NTU, line, NTU, cable and line receiver. The monitoring of the Transmit Data pin together with the Received Data of the other side, would assist in fault finding should the Received Data not correspond with the expected data (as there often are a few minor problems with project board circuits)

In order to investigate the success of the test more thoroughly, it was necessary to ascertain what data was actually transmitted from the one USART and what was being received at the other. At first glance a data analyser would seem to be a better choice of tester to assist in this regard. A logic analyser however was chosen for this purpose as it was the simpler option under the prevailing circumstances. The data on the communications link, one must remember is being transmitted at 48 kbps which is above the V.24/RS232C (V.28) data transmission rate specification limit of 20 kbps (CCITT 1985 b p. 199).

In order to monitor the data transmitted from the Local to the Central sub-system one had three options;

a) to gain access to a monitor with a V.11 interface
b) to violate the recommendation V.24 and drive the data signals to be monitored through RS232 drivers connect them to the V.24 interface of the analyser available,
c) to monitor the data using a Logic Analyser which could monitor the signals at TTL level (0-5 V) at the input to the RS422C line drivers (at the transmit side or at the output of the line receivers at the receive side).

The data analysers available had only V.24 interface connections for monitoring data. The use of the Logic Analyser would not require any extra components nor wiring to be done on the project board and was hence chosen as the quicker, simpler method.

The basic flow charts of both the central and remote sub-systems indicate the functions performed in this test. Figure 12 shows the flow chart of the central software and the circuit is outlined in Figure 13.

At the conclusion of this test, TEST4 it was confidently assumed that the method of controlling the communications line was correct.

Brief Description of the Flow of Program TEST4C

(Figure 12 and circuit Figure 13)

At power up the program initialises the USART as described above in "USART Initialisation". After this the carry flag in the micro-controller is used to alternately indicate which byte to be transmitted next in the sequence. The USART is continuously polled for it's status byte and only when able to transmit the next byte is the byte written to it.

After this the status byte is polled once to check if there is a received byte to be read. If there is, then it is read from the USART and displayed at the micro-controller's Port 1 for inspection. If not then the program prepares the alternate byte for transmission and returns to the cyclic polling of the status byte, waiting for the USART to be in a position to accept another byte for transmission.

Since the USART does not begin transmitting Sync characters until a data byte has been written to it's transmit buffer and been transmitted, this program initiated the transmission and did not look for synchronism to be achieved first. The USART transmitted data to the other end of the line in order for the remote USART to synchronise instead. A delay (SYNCDEL) was inserted in order to allow the remote system time to receive the sync characters and synchronise to them, before it started sending it's data.

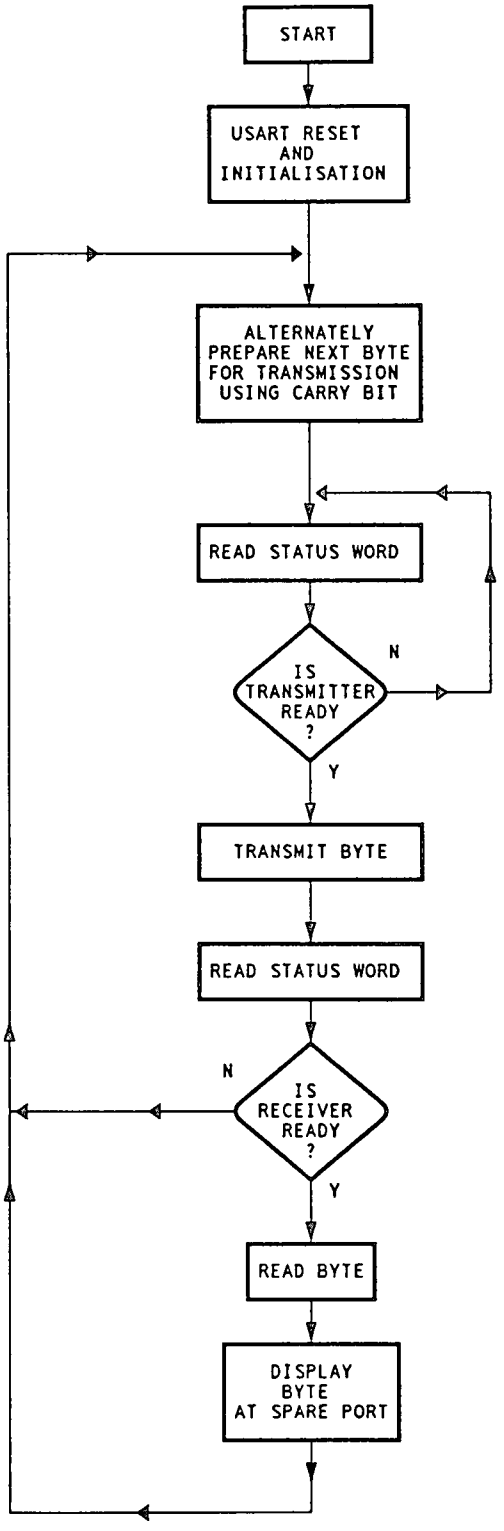

START
USART RESET AND INITIALISATION
ALTERNATELY PREPARE NEXT BYTE FOR TRANSMISSION USING CARRY BIT
READ STATUS WORD
IS TRANSMITTER READY ?
N
Y
TRANSMIT BYTE
READ STATUS WORD
IS RECEIVER READY ?
N
Y
READ BYTE
DISPLAY BYTE AT SPARE PORT

FIGURE 12

TEST4C

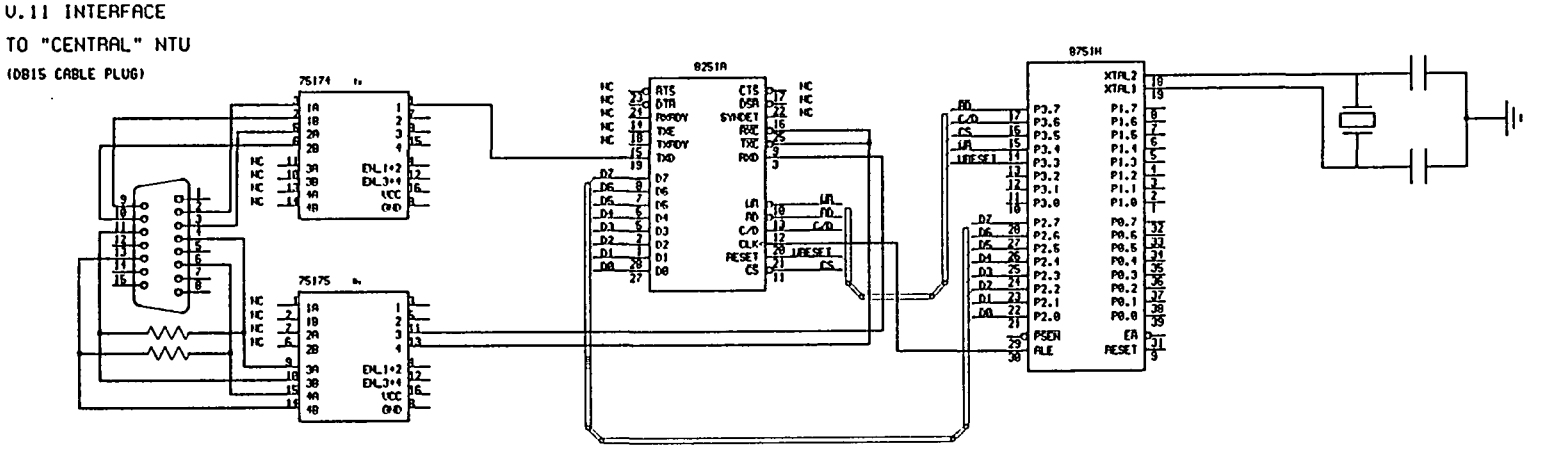


FIGURE 13

Brief Description of Program TEST4R Flow.

(Figure 14 and circuit in Figure 15)

This program differs from TEST4C in a few respects.

Firstly this sub-system will have three USARTs and they will eventually be configured for two different purposes, (namely to control the communications link and to monitor data from the serial interface). This program was used as a starting point upon which the following software for the Remote sub-system would be built. For this reason the initialisation of the two monitoring USARTs was included in this program.

Secondly the remote sub-system waited to achieve synchronism at the communications link before continuing with the program. This complemented the software of the central sub-system which first started a train of sync characters being transmitted before sending the alternating test bytes.

It is at this point appropriate to **note** the manner by which it was checked whether synchronism had been achieved at the communications link. The USART does provide an external pin to indicate when character synchronism has been achieved (SYNDET). It also provides indication by means of a bit in the Status byte (SYNDET/BRKDET).

When the Status word is read however the USART exits from the Enter Hunt mode (Intel 1993 p. 2-12). For this reason it would be simpler to just monitor the external pin (SYNDET) and get an indication of synchronism from there, as opposed to having to command the USART back into Enter Hunt Mode again. There is obviously a price to pay for this "simpler" method, and that is a sacrifice of an I/O line of the micro-controller. In this instance it

was decided to go for the polling of the Status Word method.

It is also important to note the insertion of a delay (GETSYNC). It would be incorrect to command Enter Hunt and then immediately read the status byte for the SYNDET bit, before giving the USART time to synchronise and to set the bit in the Status Word.

The delay (GETSYNC) was necessary to allow time to synchronise after each time the USART was put back into the hunt mode.

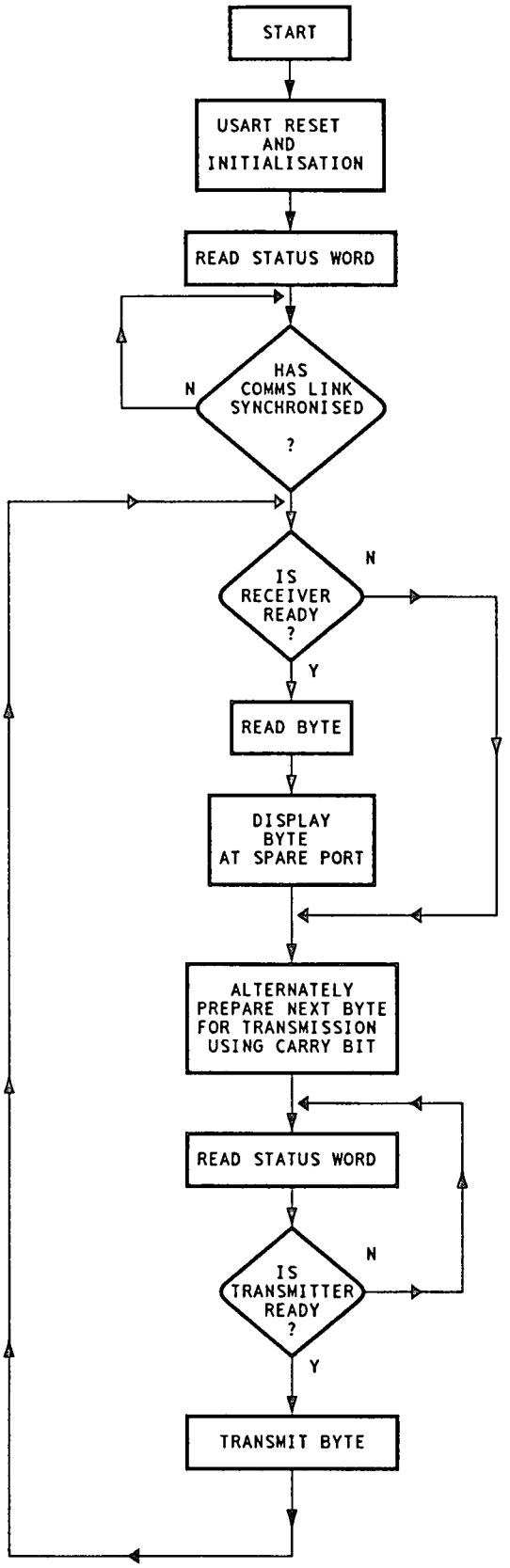

START
USART RESET AND INITIALISATION
READ STATUS WORD
HAS COMMS LINK SYNCHRONISED ?
N
IS RECEIVER READY ?
N
Y
READ BYTE
DISPLAY BYTE AT SPARE PORT
ALTERNATELY PREPARE NEXT BYTE FOR TRANSMISSION USING CARRY BIT
READ STATUS WORD
IS TRANSMITTER READY ?
N
Y
TRANSMIT BYTE

FIGURE 14

TEST4R

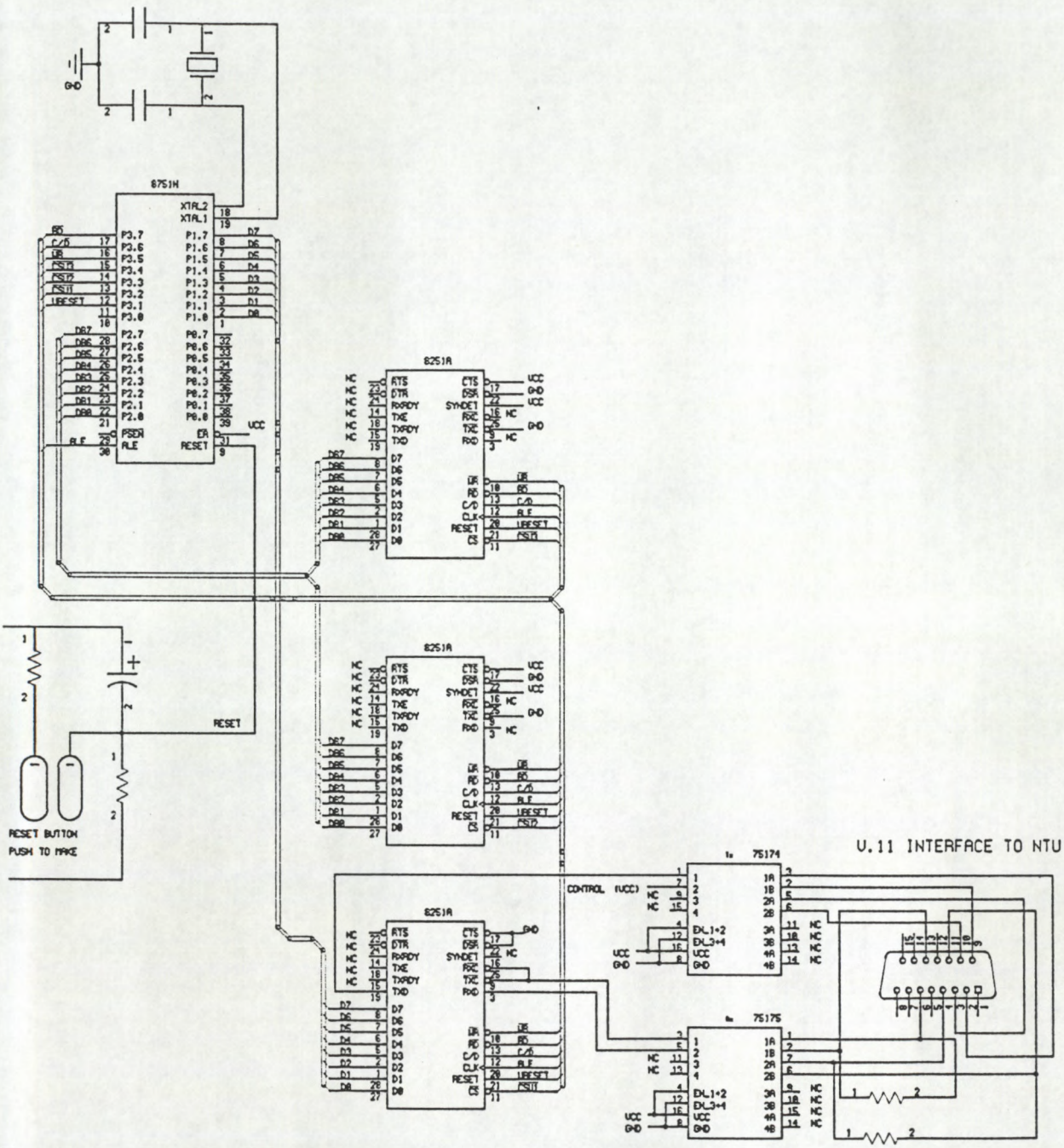


FIGURE 15

3.6.2 Testing of Communications Link Control While Monitoring Transmit Data only

Test Overview

(Programs TEST5R and TEST5C)

Once the basic control of the communications link was confirmed, the project was taken one step further involving interaction between the microcontroller, the communications link and the monitoring process. This test involved just one of the two channels of data (one of the two directions, i.e., Transmit Data) from the monitored interface.

The circuits used on the two project boards were modified to facilitate the monitoring of the one channel of data. The methods of synchronous data monitoring and clocking of the parallel data, to synchronous serial previously developed in the project were used. The circuit schematic diagram of the remote sub-system is outlined by Figure 17 and that of the central is shown by Figure 19.

A test program for each of the sub-systems was developed and programmed into the ROM of the micro-controllers for each circuit. These programs once successfully tested, were developed further adding slightly more complexity and detail. Following is a short description of the functions performed by the first pair of programs (TEST5C for the central sub-system and TEST5R for the remote sub-system software). The flow chart of the remote software is shown by Figure 16 and the chart of the central software is shown by Figure 18.1 and 18.2.

Description of Test Program TEST5R (Figure 16)

On power up the program executes the initialisation of all three USARTs. USART 1 and 2 are initialised and programmed for synchronous mode with external sync-detect (as before, not all the USARTs are used, in this case USART 2 is not).

These USARTs, 1 and 2, are intended to monitor the appropriate data from the monitored interface.

USART 3 is programmed for synchronous data, using the double sync character and internal sync detect.

After the initial programming of the USARTs the test program now achieves synchronism on the communications link to the central sub-system under control of USART 3. The status word of the USART is read into the microcontroller and when the transmitter is ready an arbitrary byte is written to the USART (in this case the Status Word) which begins the automatic sync. character transmission.

The USART is now commanded back into the Hunt Mode and the SYNDET pin is monitored. The program will wait at this point until a logic High appears at this pin, indicating that the communications link has synchronised in the receive direction.

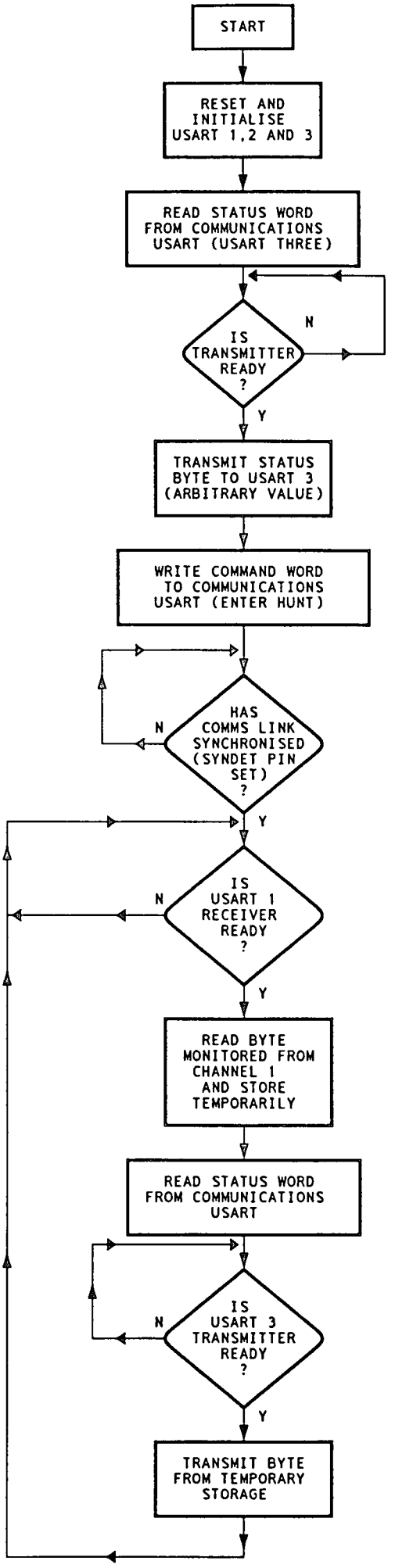
The next stage of the program cyclically polls USART 1, reading its Status word and waiting for the Receiver Ready (RxRDY) bit to be set. This stage of the program waits for an eight bit byte to be monitored and assembled by USART 1 from the Transmit Data (TxD) of the monitored interface. Once the status word indicates the Receiver Ready, then the data byte is read from USART 1 and stored temporarily.

The micro controller then cyclically reads the status word of USART 3 until the Transmitter Ready (TxRDY) bit is set

indicating that the USART can accept another data byte for transmission over the communications link.

After this has been done the program then loops in this sequence of reading the monitored bytes from USART 1 and transmitting them to the Central sub-system over the communications link.

TEST 5R



KEY:
USART ONE: MONITORS CHANNEL ONE
USART TWO: NOT USED
USART THREE: CONTROLS COMMS LINK

FIGURE 16

TEST 5R

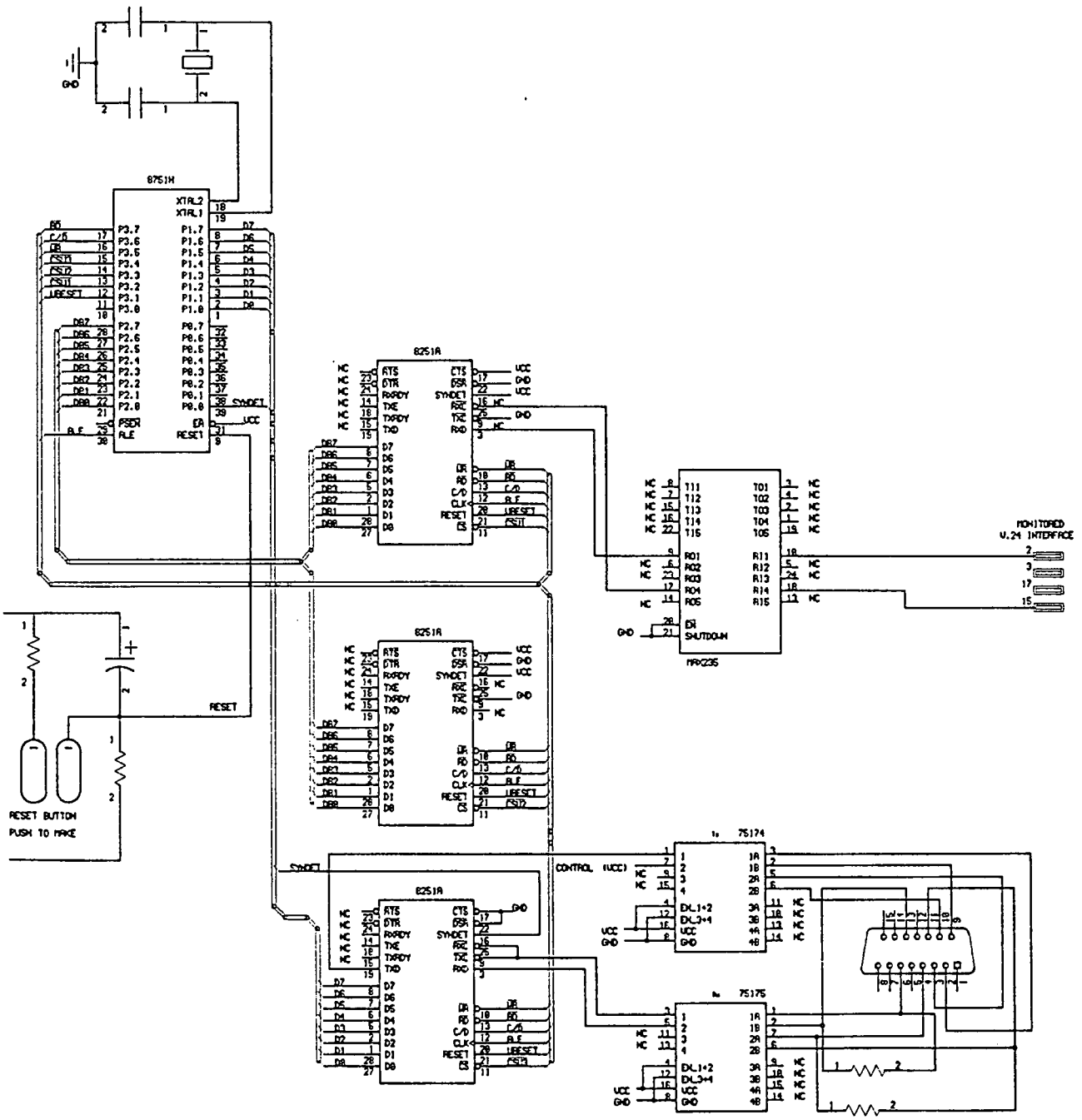


FIGURE 17

Description of TEST5C (Figure 18.1 and 18.2)

This program follows basically the same format as TEST5R on power up. The USART (only one in this circuit, Figure 19) controlling the communications link is initialised in the same manner. After synchronism is achieved on the communications link the status word of the USART is read to determine when the USART has a received byte for the micro-controller. The valid data bytes received from the communications link are written into a queue in the Random Access Memory (RAM). The queue is accessed and read for the next byte to be output to the reproduced interface.

One of the main tasks of this program is to decipher the difference between the sync characters and valid data bytes received from the Remote sub-system via the communications link. A very primitive (apparently unique) protocol was developed to assist in this task.

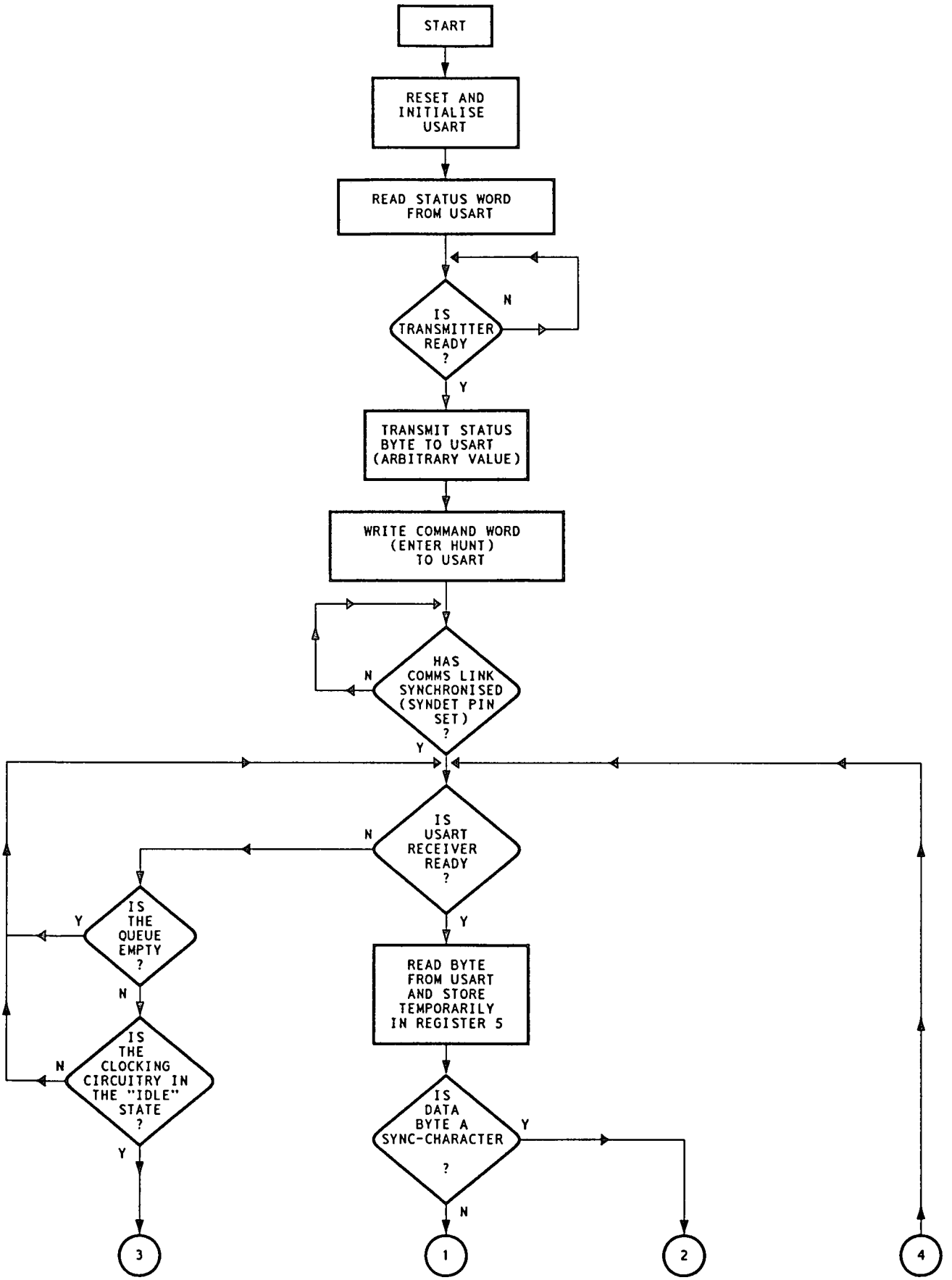

START
RESET AND INITIALISE USART
READ STATUS WORD FROM USART
IS TRANSMITTER READY ?
N
Y
TRANSMIT STATUS BYTE TO USART (ARBITRARY VALUE)
WRITE COMMAND WORD (ENTER HUNT) TO USART
HAS COMMS LINK SYNCHRONISED (SYNDET PIN SET) ?
N
Y
IS USART RECEIVER READY ?
N
Y
IS THE QUEUE EMPTY ?
Y
N
IS THE CLOCKING CIRCUITRY IN THE "IDLE" STATE ?
N
Y
READ BYTE FROM USART AND STORE TEMPORARILY IN REGISTER 5
IS DATA BYTE A SYNC-CHARACTER ?
Y
N
3
1
2
4

FIGURE 18.1

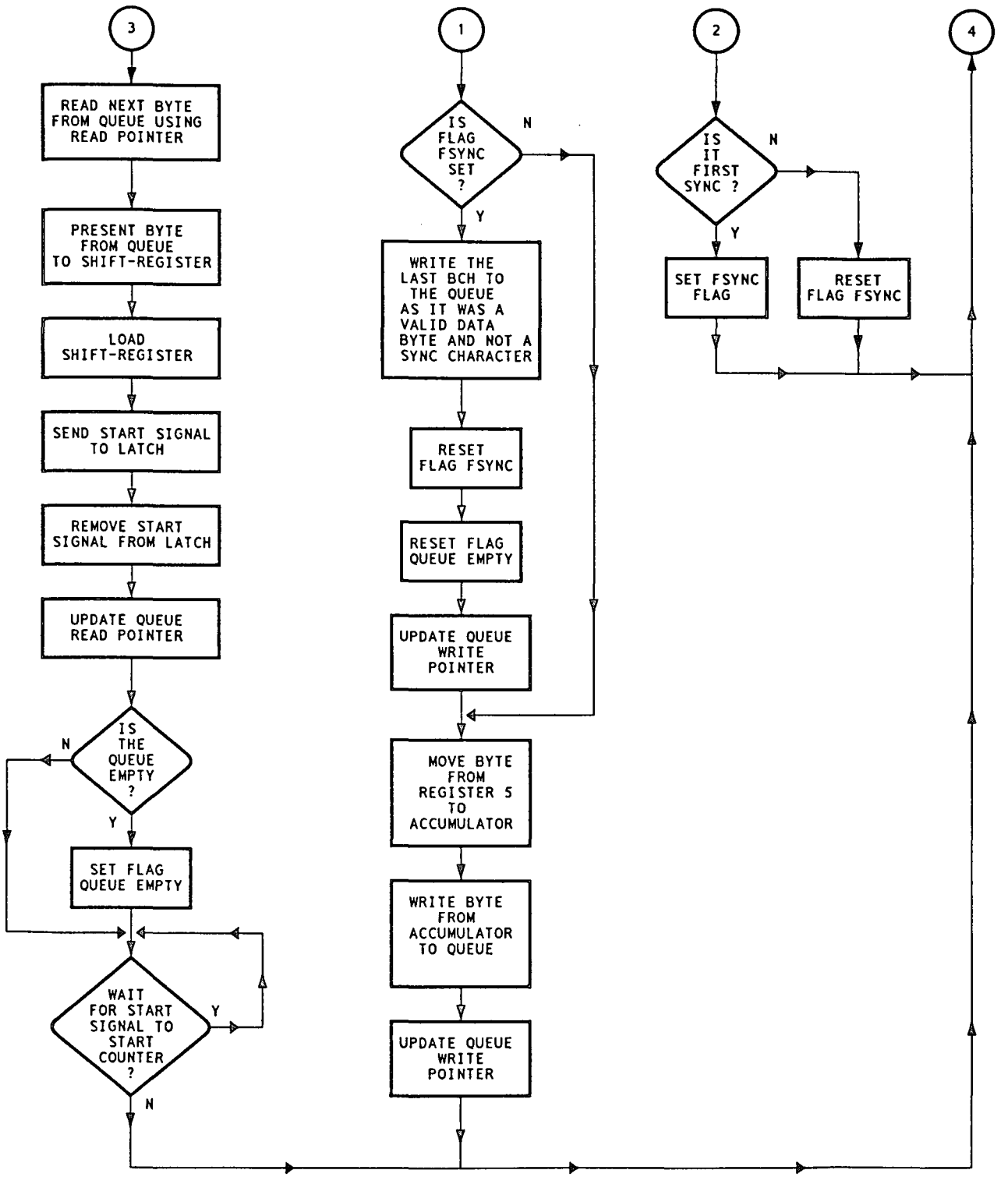

3
READ NEXT BYTE FROM QUEUE USING READ POINTER
PRESENT BYTE FROM QUEUE TO SHIFT-REGISTER
LOAD SHIFT-REGISTER
SEND START SIGNAL TO LATCH
REMOVE START SIGNAL FROM LATCH
UPDATE QUEUE READ POINTER
IS THE QUEUE EMPTY ?
N
Y
SET FLAG QUEUE EMPTY
WAIT FOR START SIGNAL TO START COUNTER ?
Y
N
1
IS FLAG FSYNC SET ?
N
Y
WRITE THE LAST BCH TO THE QUEUE AS IT WAS A VALID DATA BYTE AND NOT A SYNC CHARACTER
RESET FLAG FSYNC
RESET FLAG QUEUE EMPTY
UPDATE QUEUE WRITE POINTER
MOVE BYTE FROM REGISTER 5 TO ACCUMULATOR
WRITE BYTE FROM ACCUMULATOR TO QUEUE
UPDATE QUEUE WRITE POINTER
2
IS IT FIRST SYNC ?
N
Y
SET FSYNC FLAG
RESET FLAG FSYNC
4

FIGURE 18.2

TEST 5C

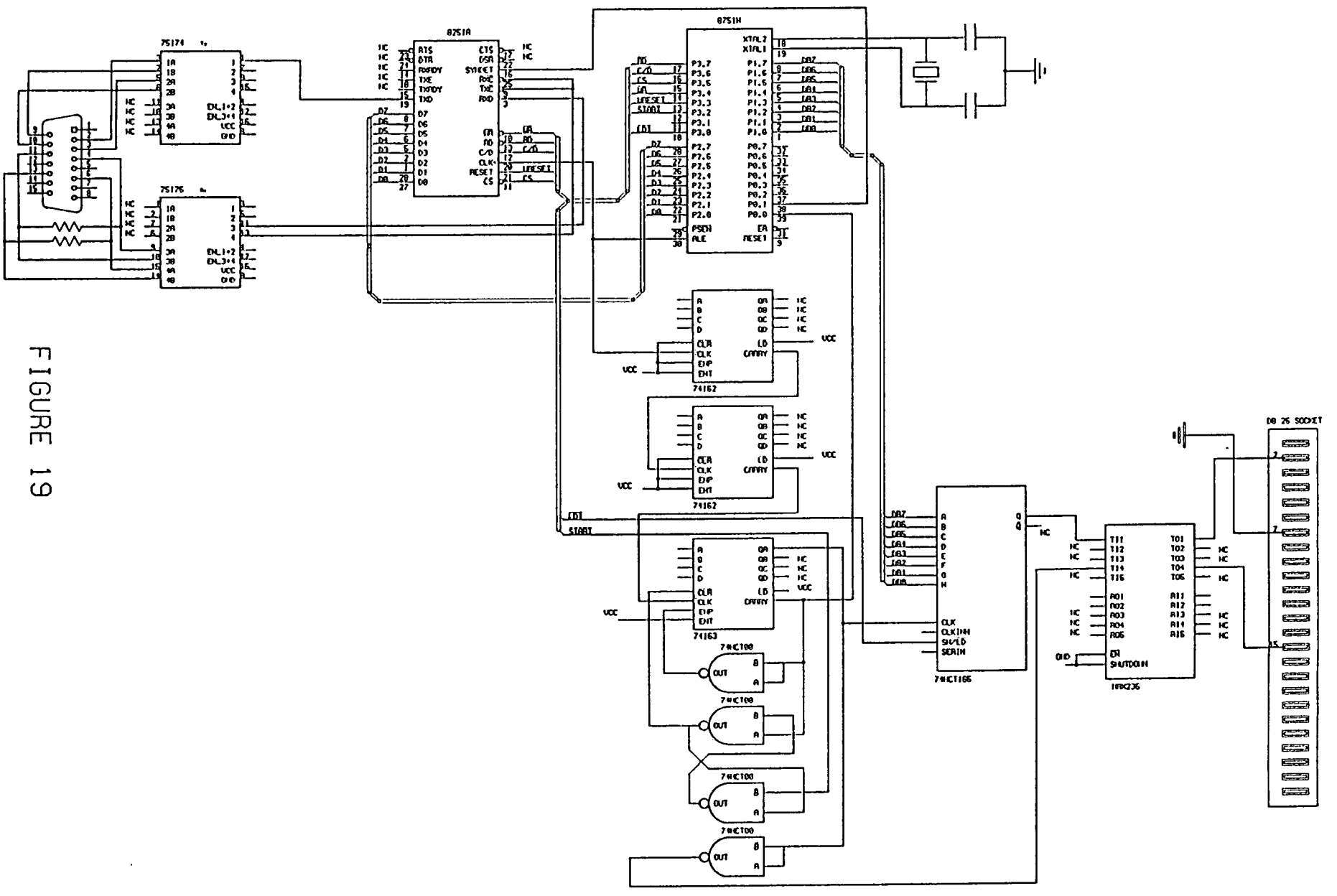


FIGURE 19

Description of Data Protocol Used in TEST5R and TEST5C)

A full blown protocol would not have been appropriate in this basic test. This developed protocol does not cater for all combinations of monitored data and consequently allows for potential errors in the interpretation of received data. The protocol however fulfilled the requirements for the purpose of this test (see Results of TEST5C and TEST5R below). For the simplification of explaining the flow of decisions by the program, a case of two successive received bytes, namely A and B will be discussed. The program causes the micro-controller to cyclically poll the USART and to read it's status byte. When a byte is received by the USART the micro-controller will notice the Receiver Ready bit of the status word being set and will then read and temporarily store the byte (A) from the USART. Received bytes are checked whether identical to the sync character. In this case let's say the byte A is identical. The program checks whether or not received bytes identical to the sync-character, are the first or second in the pair of sync-characters (transmitted by the remote USART, controlling the link). When the byte is identical to the sync-character the micro-controller makes a note of it (by setting a flag called FSYNC {first sync}). Should the next data character (B) received from the link also be identical to the sync-character, both consecutive characters will be ignored. This is because they will be accepted as a pair of sync characters (in the controlling of the communications link, maintaining synchronism) and not data or information bytes transmitted by the Remote micro-controller. Should the next character (B) received not be identical to the sync character, the previous byte (A) being equal to the sync character, would then be interpreted as a valid data byte. The byte (A) then is retrieved from storage and written into the queue. The following byte (B) is then also written into the queue. This

scenario is the one which is most often the case for the test run in this exercise.

There is a potential shortfall in the philosophy used in this link protocol. The problem however does not manifest itself in this test because of the specific conditions under which the test is executed. This is explained further in the following paragraphs.

Results of TEST5C and TEST5R

Problem with link protocol overcome

The inherent problem with the protocol is, that in order for no data bytes to be misinterpreted there is a limitation placed on the actual data allowed to be transmitted on the link. The limitation is placed on the data because two successive received data bytes identical to the sync character (whatever it may be) will be ignored and lost by the receiver. The communications link using this protocol shall not allow two successive bytes equal to the sync character, in the data to be transmitted without introducing errors in the interpretation thereof.

This limitation is not however restrictive in this particular case for the following reason.

The interface monitored is operating at a speed of 9600 bps.

The bit period therefore being $T = 1/f$

= 1/(9600 bits per second)

= 104.16 μs per bit

The period of eight bits = 8 x 104.16 μs

= 833.3 μs

The communications link operating at a speed of 48 kbps.

The bit period therefore being $T = 1/f$

= 1/(48000 bits per second)

= 20.833 μs per bit

The period of eight bits = 8 x 20.833 μs

= 166.666 μs

Period of 5 bytes on high speed link = 5 x period of 8 bits

= 5 x 166.666 μs

= 833 μs

Over a period of 833μs on the transmission link we can see that there is exactly enough time to transmit five, eight bit, characters. In the same time however only one group of eight bits will have been monitored from the monitored interface and be ready for transmission over the transmission link. It is therefore reasonable to assume that there will never be two successive data bytes for transmission to the local end before the USART automatically inserts the sync characters. In other words the data on the communications link would consist of four sync characters and one data byte for every 833 μs of transmission time. This would dovetail with the protocol used and result in successful management of the communications link.

Explanation of Wait Delay in Figure 18.2

After the START signal was sent to the Latch (in order to clear the counter and begin another cycle of 8 pulses), the program had to make sure that the pules had started before proceeding with program execution. This had to be done as it was possible for the

program to overwrite the byte already loaded in the Shift Register if the conditions were conducive. The conditions were; that the USART's Receiver Ready was not set, that the queue was not empty and that the counter had not reset as yet. The possibility was unlikely but the program delay was inserted to make certain that the possibility was ruled out completely.

The asynchronously cleared counter, the 74161, would eliminate this problem but was unavailable at the time of the test.

3.6.3 Advanced Testing of Communications Link Control, While Monitoring Transmit Data and Receive Data

Test Overview

(Programs TEST51R and TEST51C)

Description of TEST51R

The program TEST51R is a modification on the program TEST5R. The major improvements on this program are;

a) Both channels of data, namely in the Transmit and Receive directions are now monitored from the monitored interface (as opposed to only Transmit Data in the previous software version).
b) The USART 3 (communications controller) now interrupts the main program when the Transmitter is ready (as opposed to the repeated polling of the USART to read it's status byte)

The flow chart of this program is found in Figures 20.1, 20.2 and 20.3.

The schematic of this circuit is shown in Figure 21.

On power up the program execution is similar to the previous version (TEST5R). After synchronism is achieved at the communications link and the data byte is monitored from the Transmit Data interface signal. Now the second channel's information is also monitored by USART 2. Because the data transmission rate of the monitored interface is consistent for both Receive and Transmit directions (channels) the data bytes monitored will be assembled (by USART 1 & 2) almost, if not at exactly at the same time. The bytes will then be read by the microcontroller very soon after one another. These data bytes are monitored and stored temporarily, only then is any transmission of information to the central side possible.

After the two bytes are read from the two monitoring USARTs, a flag (MONFLAG) is cleared which prevents the program from reading the monitoring USARTs until both monitored bytes are transmitted.

The transmission of the bytes will take place under the control of the Interrupt (zero) program. In other words the two data bytes from the separate channels of the interface are transmitted to the central side as a pair. At the (monitored) data rate of 9600 bps, the microcontroller has approximately 833 μs before the USARTs will have compiled the next data bytes from the serial monitored interface. This time period is equal to the period required for the transmission of 5 data bytes on the communications link (working at 48 kbps) to the central side.

At the stage where the second monitored byte is read into the microcontroller, the monitoring is disabled by the use of a flag (MONFLAG already mentioned) and then the interrupt zero is enabled.

The main program now loops back to the position at the beginning of the monitoring and waits there until this flag is set. The flag is set by the interrupt zero routine **after** both monitored data bytes have been written to the communications USART (USART 3) for transmission.

The Transmitter Ready output pin (from USART 1) causes the program counter to change to the interrupt 0 vector address. The first vector to the interrupt program will result in the transmission of the first byte after which the interrupt routine will set a flag indicating that the first byte has been transmitted (CH1DONE). The interrupt routine then will end and processor control is returned to the main program.

The main program, remember, is waiting for the flag preventing it from monitoring (MONFLAG), to be set. At the next interrupt, caused by the next High at the Transmitter Ready the interrupt routine is called again. This time the interrupt program jumps at the inspection of the flag (CH1DONE) indicating that the first byte has been transmitted. The program now writes the second byte to the USART, and then resets the flag (CH1DONE) to indicate that the next byte to be transmitted is the channel one byte. The flag preventing the main program from monitoring (MONFLAG) is now set. The interrupt program now prevents the USART's Transmitter Ready output from re-interrupting the main program until after the next two bytes are prepared for transmission. It does this by clearing the EX0 bit in the Interrupt Enable Special Function Register, which disables interrupt zero from further triggering.

The introduction of interrupts was considered a good path to follow for the following reasons;

a) The interrupt would later be used for the final software as it simplifies the programming removing the need to continually

poll the USART to see when the Transmitter is ready

b) The two bytes transmitted should not have any chance of being separated on the communications link by sync character/s as this would result in a faulty interpretation of the data received at the central side. The interrupt would definitely see to it that the very next character transmitted is the second monitored byte.

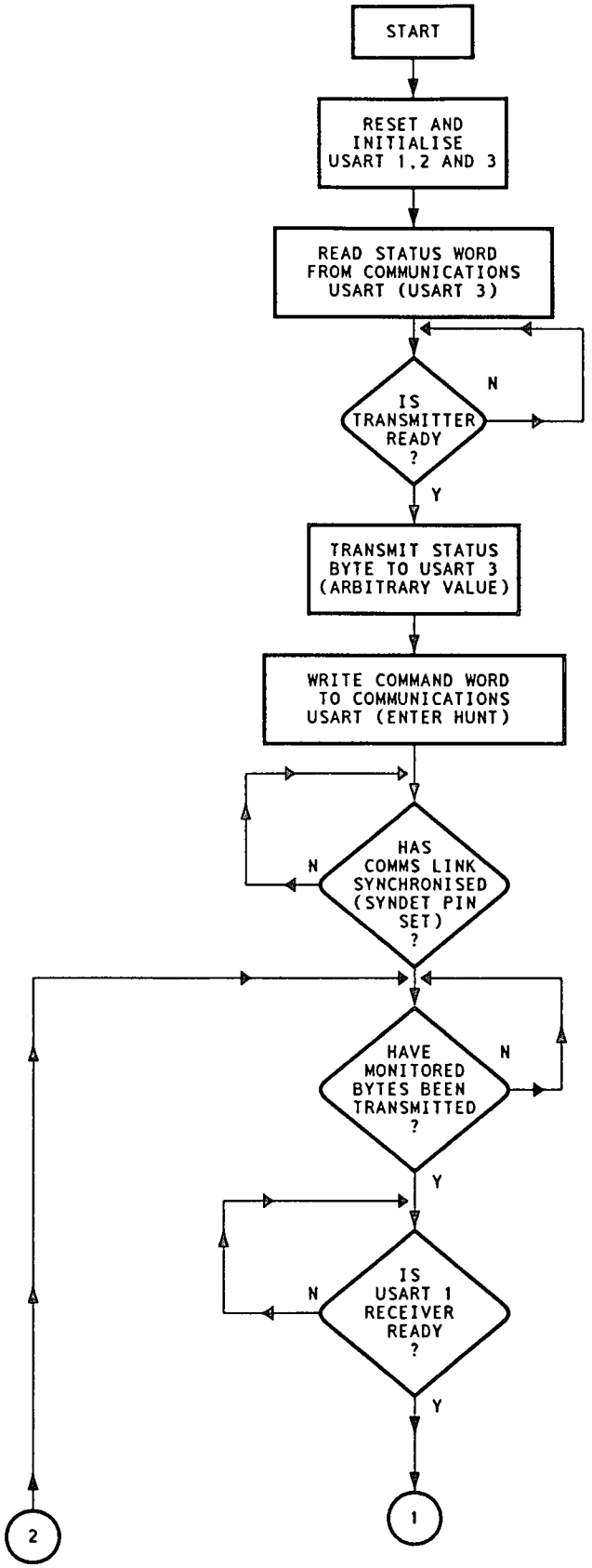
Point b) above is of importance to the protocol used for receiving data at the central side (see paragraph The Results below).

There is no chance of USART 1 and/or 2 loosing the next two monitored bytes (caused by overrun error) while the monitoring is disabled. The monitoring by the main program is re-enabled approximately 500 μs before any bytes are available for reading from the monitoring USARTs 1 and 2.

Period of monitored byte = 833 μs = "x"

Period of disabled monitoring ≈ (2 x 166) μs = "y"

Period of remaining monitor wait ≈ "x" - "y" = 501 μs



KEY:
USART 1: MONITORS CHANNEL ONE
USART 2: MONITORS CHANNEL TWO
USART 3: CONTROLS COMMS LINK

FIGURE 20.1

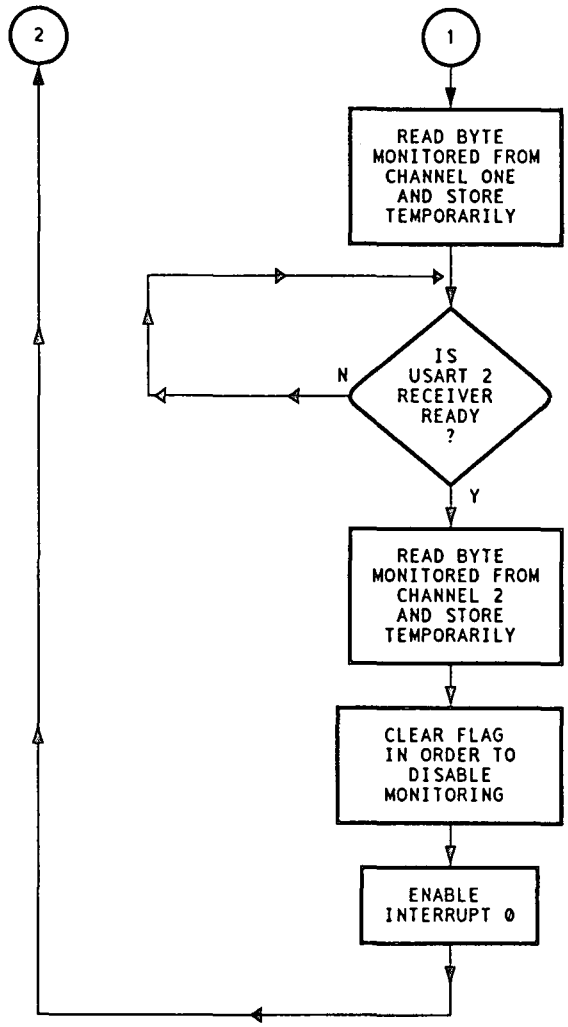

2
1
READ BYTE MONITORED FROM CHANNEL ONE AND STORE TEMPORARILY
IS USART 2 RECEIVER READY ?
N
Y
READ BYTE MONITORED FROM CHANNEL 2 AND STORE TEMPORARILY
CLEAR FLAG IN ORDER TO DISABLE MONITORING
ENABLE INTERRUPT 0

FIGURE 20.2

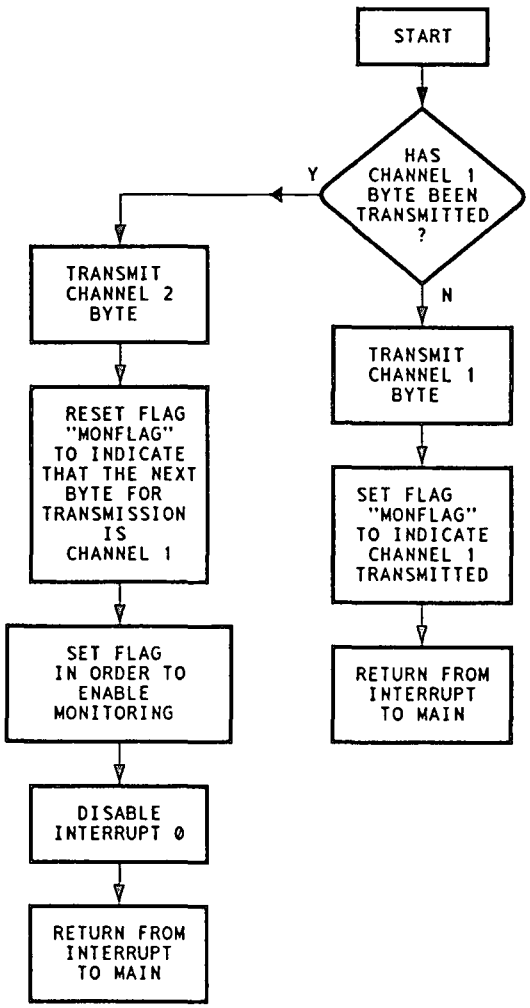

START
HAS CHANNEL 1 BYTE BEEN TRANSMITTED ?
Y
N
TRANSMIT CHANNEL 2 BYTE
RESET FLAG "MONFLAG" TO INDICATE THAT THE NEXT BYTE FOR TRANSMISSION IS CHANNEL 1
SET FLAG IN ORDER TO ENABLE MONITORING
DISABLE INTERRUPT 0
RETURN FROM INTERRUPT TO MAIN
TRANSMIT CHANNEL 1 BYTE
SET FLAG "MONFLAG" TO INDICATE CHANNEL 1 TRANSMITTED
RETURN FROM INTERRUPT TO MAIN

FIGURE 20.3

TEST 51R

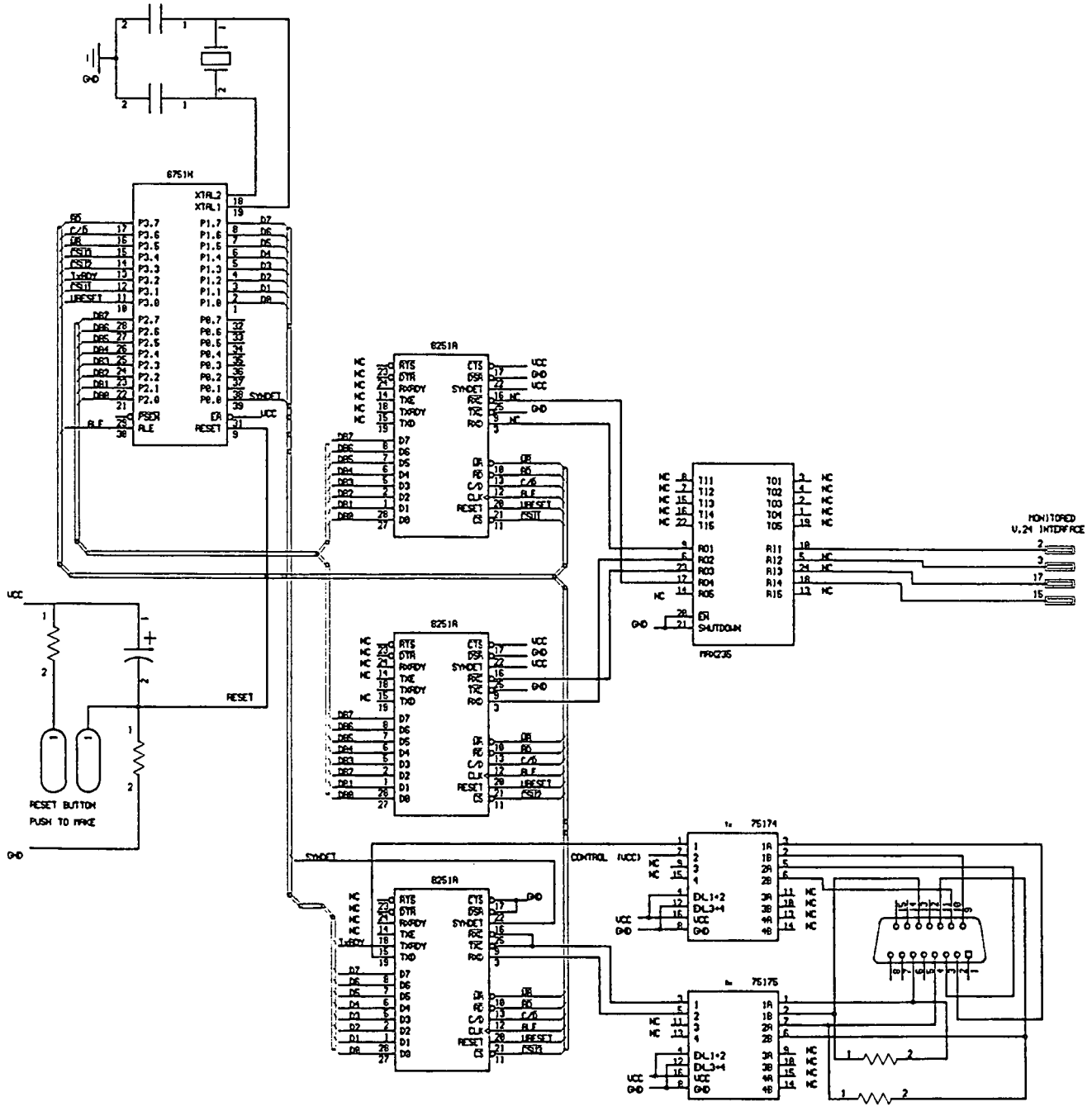


FIGURE 21

Description of TEST51C (Figure 22.1 & 22.2)

The program TEST51C is a modification on the program TEST5C. The major improvements on this program are;

a) two (instead of one) shift registers are controlled and hence reproduce the two data channels to the interface,
b) the protocol used in the reception of data from the communications link is adapted to cater for information frames of two bytes long.

The main objectives of this program are;

a) to confirm that the interrupt program of the remote software (TEST51R) was correctly implemented and error free,
b) to confirm that the transmission of both channels over the communications link is functional,
c) to test the additional software controlling the added channel of monitored data.

This added data byte, on the communications link, now changed the validity of the protocol once again and there was a new limitation placed on the data using that protocol. This is discussed further under "The Results" below.

This program, TEST51C, follows basically the same format as TEST5C on power up. The USART controlling the communications link is initialised in the same manner. After synchronism is achieved on the communications link the status word of the USART is read to determine whether the USART has a received byte for the micro-controller. The valid data bytes received from the communications link are firstly checked for equality to the sync characters used by the communications link. If

the character is found to be identical then it is ignored.

When a byte received is not found to be identical it is accepted as identifying the start of the data frame of length two bytes.
The first byte of the frame is stored and a flag (CH2) is set, indicating that the next byte received is the second and last byte of the frame.

Once the frame is complete (once the second byte is received) a flag (QE - Queue Empty) is cleared to indicate that there is information to be output to the shift-registers.
The program will branch to check whether the queue is empty whenever it checks that the Receiver is not ready. If the queue is **not** empty, then the status of the clocking circuitry is checked. If it is in the "IDLE" state (see 3.4.2) then the next data byte is read from the queue and presented and written to Shift-Register 1. The queue read pointer is then updated. The next data byte is read from the queue and presented and written to Shift-Register 2.

Once this has taken place the latch is triggered by the "START" signal out from the microcontroller.
The wait delay for the clear of the counter was left in from the previous version (TEST5C) for the same reasons outlined in paragraph "Results of TEST5C and TEST5R" and 3.4.2.

The queue is checked to see if it is empty and the queue empty flag is updated accordingly. This was to save time while waiting. The clearing of the counter is waited for and then the program jumps back to the polling of the Receiver Ready on the communications USART.

Once cleared, the two bytes loaded into Shift-Register 1 and 2 are clocked and driven onto the interface via RS232 drivers.

In this manner the two monitored bytes are simultaneously clocked onto the reproduced interface.

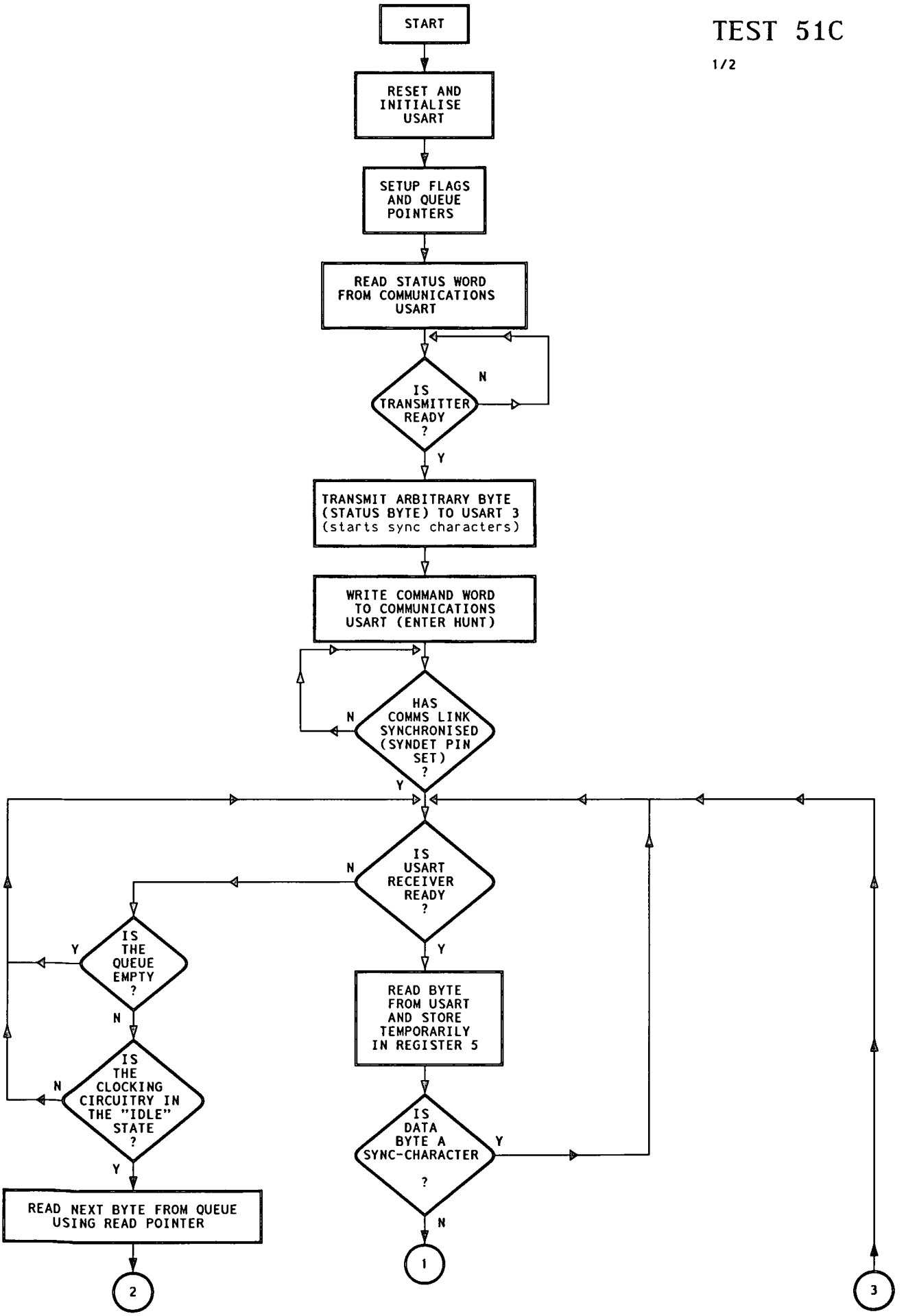

START
RESET AND INITIALISE USART
SETUP FLAGS AND QUEUE POINTERS
READ STATUS WORD FROM COMMUNICATIONS USART
IS TRANSMITTER READY ?
N
Y
TRANSMIT ARBITRARY BYTE (STATUS BYTE) TO USART 3 (starts sync characters)
WRITE COMMAND WORD TO COMMUNICATIONS USART (ENTER HUNT)
HAS COMMS LINK SYNCHRONISED (SYNDET PIN SET) ?
N
Y
IS USART RECEIVER READY ?
N
Y
READ BYTE FROM USART AND STORE TEMPORARILY IN REGISTER 5
IS DATA BYTE A SYNC-CHARACTER ?
Y
N
1
IS THE QUEUE EMPTY ?
Y
N
IS THE CLOCKING CIRCUITRY IN THE "IDLE" STATE ?
N
Y
READ NEXT BYTE FROM QUEUE USING READ POINTER
2
3

FIGURE 22.1

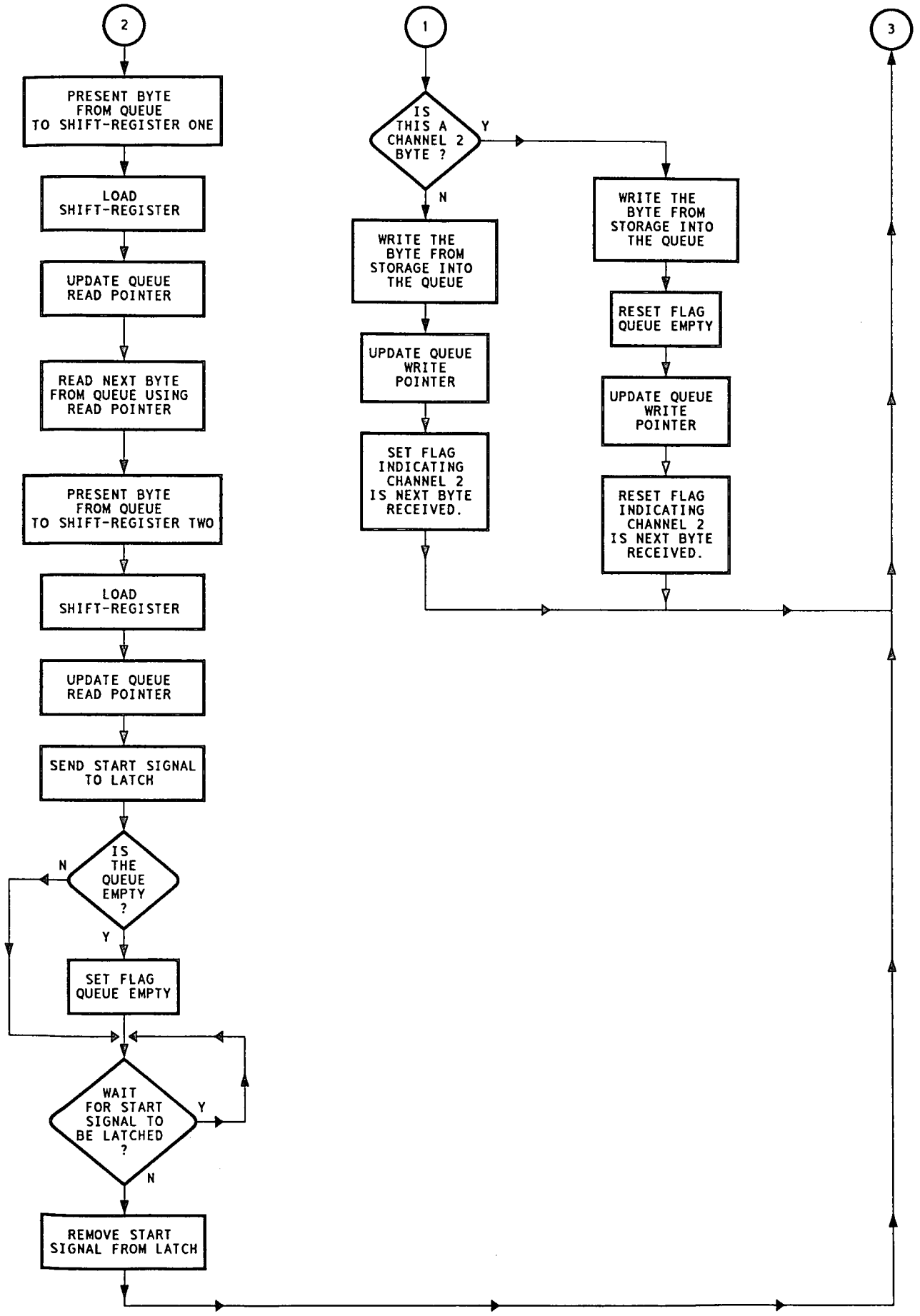

2
PRESENT BYTE FROM QUEUE TO SHIFT-REGISTER ONE
LOAD SHIFT-REGISTER
UPDATE QUEUE READ POINTER
READ NEXT BYTE FROM QUEUE USING READ POINTER
PRESENT BYTE FROM QUEUE TO SHIFT-REGISTER TWO
LOAD SHIFT-REGISTER
UPDATE QUEUE READ POINTER
SEND START SIGNAL TO LATCH
IS THE QUEUE EMPTY ?
N
Y
SET FLAG QUEUE EMPTY
WAIT FOR START SIGNAL TO BE LATCHED ?
Y
N
REMOVE START SIGNAL FROM LATCH
1
IS THIS A CHANNEL 2 BYTE ?
Y
N
WRITE THE BYTE FROM STORAGE INTO THE QUEUE
UPDATE QUEUE WRITE POINTER
SET FLAG INDICATING CHANNEL 2 IS NEXT BYTE RECEIVED.
WRITE THE BYTE FROM STORAGE INTO THE QUEUE
RESET FLAG QUEUE EMPTY
UPDATE QUEUE WRITE POINTER
RESET FLAG INDICATING CHANNEL 2 IS NEXT BYTE RECEIVED.
3

FIGURE 22.2

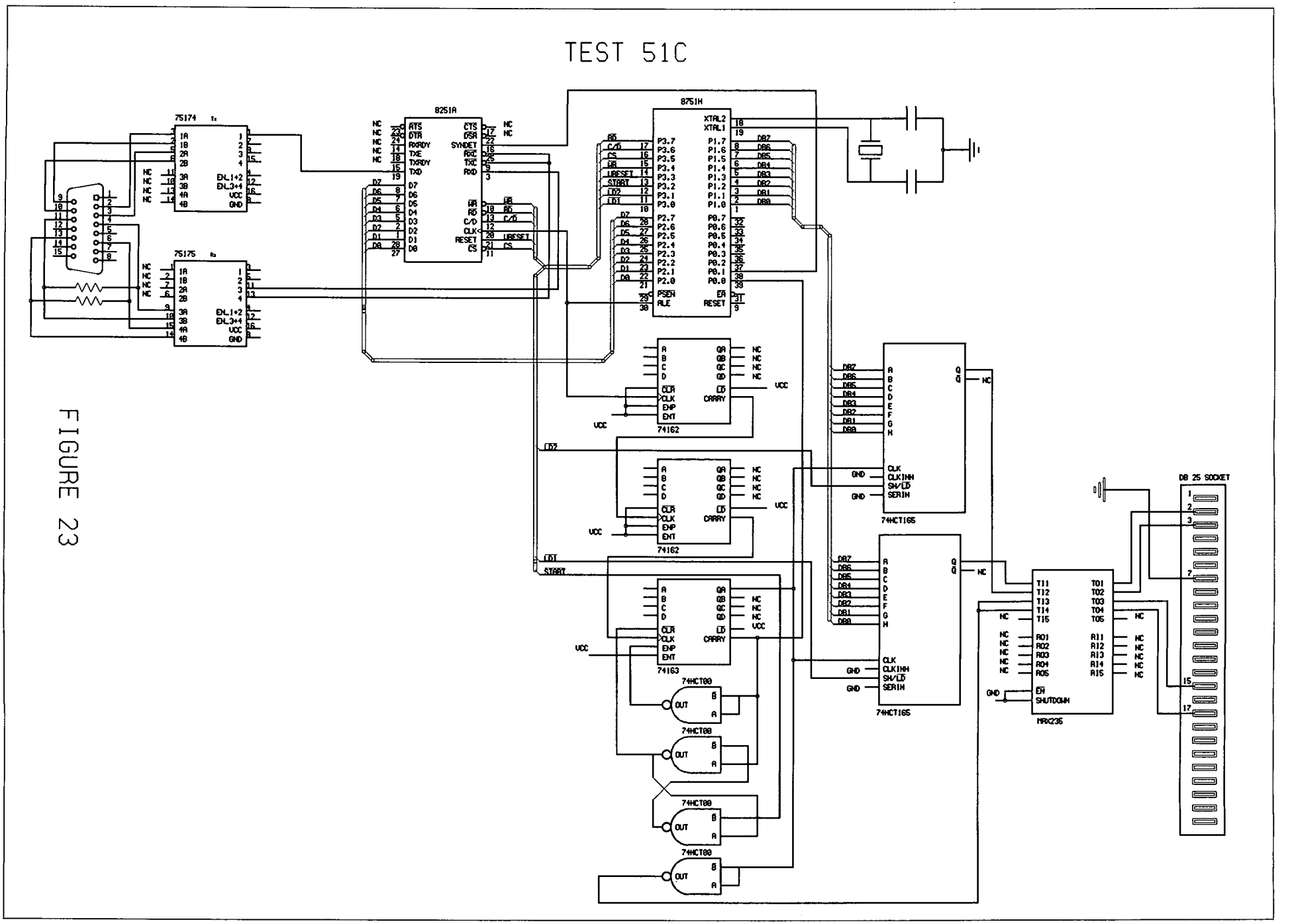

TEST 51C
75174
75175
8251A
8751H
74162
74162
74163
74HCT00
74HCT00
74HCT00
74HCT00
74HCT165
74HCT165
MAX235
DB 25 SOCKET
VCC
GND
CLK
CLKINH
SH/LD
SERIN
SHUTDOWN
CARRY
OUT

FIGURE 23

The Results (TEST51R and TEST51C)

The results of the first program (TEST51R and TEST51C) that remotely monitored both data channels, were largely positive. This was after superficial software bugs were removed with the help of an ICE. The MDS at Technikon Natal was used.

The data protocol used was not without fault, and problems would arise if certain combinations of data were to be transmitted over the communications link. In other words for the protocol to be 100% valid, restrictions would be required on the data permitted across the link, excluding the data combinations which posed a problem to the protocol. Since the data being transmitted over the link is monitored, the system has no control over it whatsoever and it is inevitable that the problematic data combinations will at one time or another be encountered.

The problem is caused by the basic protocol not catering for data equal to sync characters (BCH). The data, i.e., the monitored bytes, is ignored by the Central sub-system when identical to the sync character. This causes the actual data byte to be lost and the channel information to be corrupted for that particular byte pair. This error will cause the incorrect interpretation of the incoming data and the Chameleon rejecting the packet of data. These faulty packets were indicated as "incomplete event" by the Chameleon. When this happens the Chameleon will work at trying to synchronise back into the data being presented to it, which it eventually does (approximately 30 s delay). When the Chameleon re-synchronizes to the data presented to it the packets are displayed on both sides of the screen as normal (indicating the success of the remote data monitoring through the project and over a 48 kbps

communications link).

In retrospect an improvement could be made on the protocol used here merely by changing the sequence of checks made on the data bytes received from the data interface. If the data byte was first checked whether it was a first or second byte (in the monitored pair) then if being the second, the check for equality to a sync character could be skipped (see Figure 22.1 and 22.2). This would decrease the probability of errors on the data using this protocol by half.

The problem with the protocol did not cause too much concern at this stage of the development and the test was used upon which to build further development.

Description of test programs TEST52R and TEST52C

Description of test program TEST52R

The operation of this test program has no major changes from the TEST51R, apart from the following;

a) The USARTs were memory mapped and linearly addressed.

b) Delays were inserted during USART initialisation.

The circuit schematic is outlined in Figure 24.

The previous way of USART addressing was very time consuming (one microsecond per instruction). For every access to the USART a minimum of 4 processor instructions was required. In addition, the byte then read or written was required to be moved to/from the port using more processor time. Each time there was a change in direction from output to input the port had to be enabled before

reading, also taking more processor time. The USARTs now were operated on the conventional microcontroller data bus (Port 0) and addressed directly and controlled by the micro controller's READ and WRITE lines. The USART's control and data registers were memory mapped and addressed by the I/O of Port 2. The memory mapping shown below.

0000H - 7FFFH	PROGRAM MEMORY
6000H	USART 3 DATA TRANSFER
7000H	USART 3 CONTROL/STATUS TRANSFER
A000H	USART 2 DATA TRANSFER
B000H	USART 2 CONTROL/STATUS TRANSFER
C000H	USART 1 DATA TRANSFER
D000H	USART 1 CONTROL/STATUS TRANSFER

TEST 52R

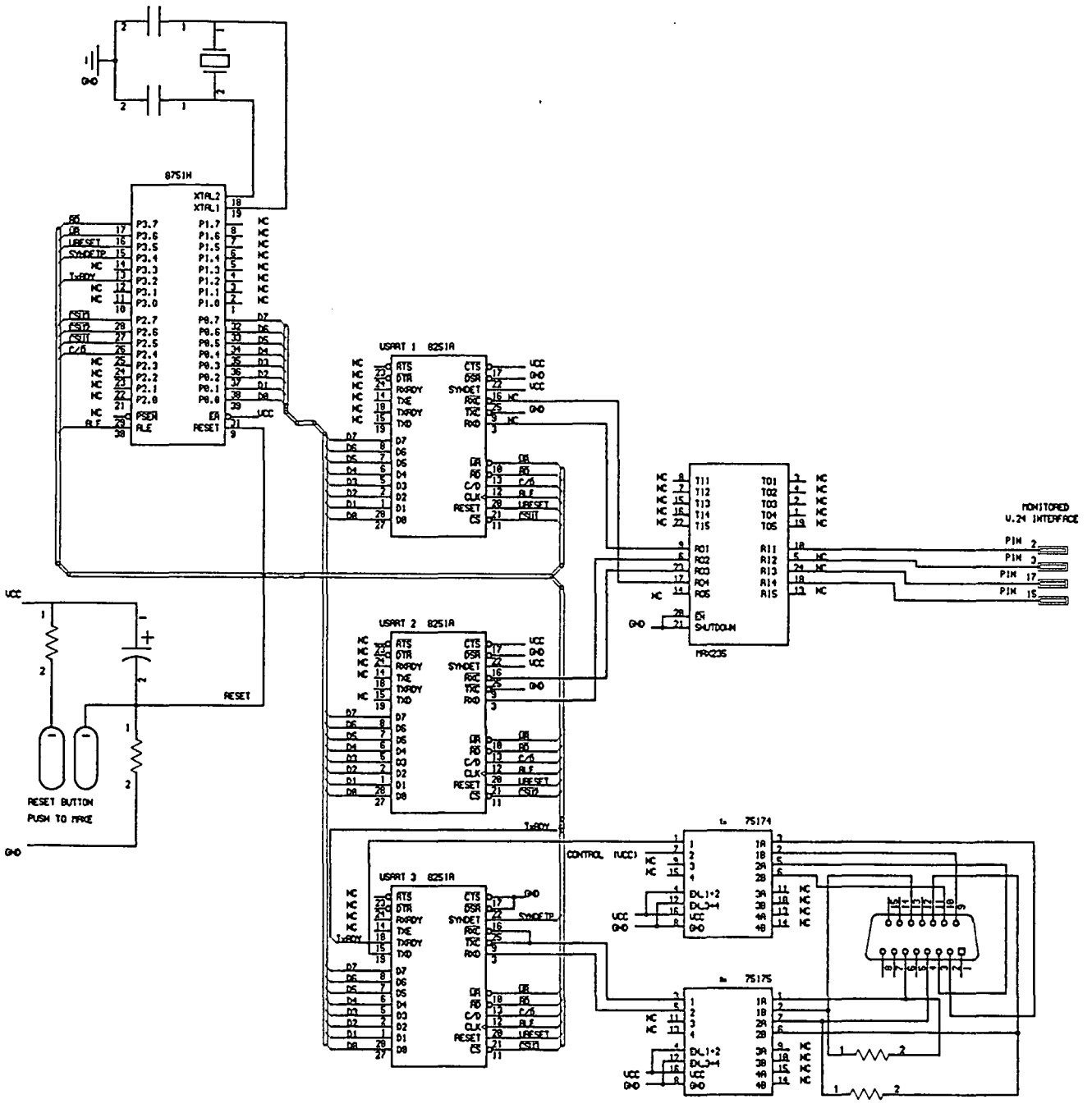


FIGURE 24

The basic **flow chart** being **identical** to that of program TEST51R is not displayed in this dissertation.

Advantages of this change in design include the following;

a) faster execution of the same process,

b) shorter code listing,

c) saving of micro-controller ROM

e.g. binary programs

TEST51R.BIN used 308 bytes

TEST52R.BIN used 162 bytes

TEST 52C

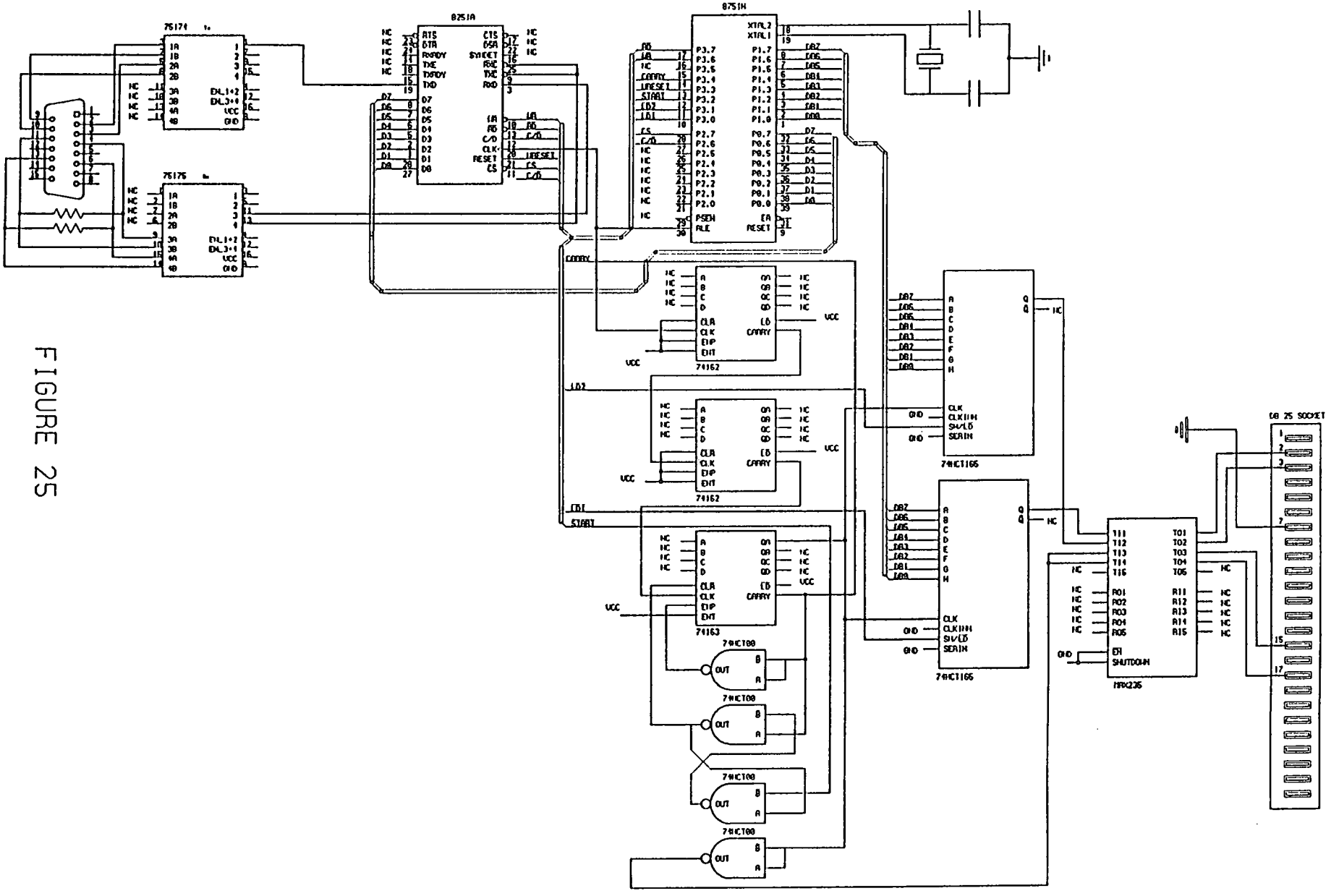


FIGURE 25

Description of test program TEST52C

The operation of this test program has no major changes from the program TEST51C, apart from the following;

a) the USART is linearly addressed as described above,

b) delays were inserted during USART initialisation

The circuit schematic is outlined in Figure 25

Results of the Test (TEST52R and TEST52C)

There was a problem when this test was first run. After investigation the problem was obvious that it involved the USARTs' initialisation. After power up the communications USARTs would only sometimes transmit sync characters, other times there would be a permanent High at their TxD pins. Since they were being programmed for synchronous transmission and a data byte had been written to them, it was expected that they would be transmitting continuous sync characters. After using the MDS in emulation mode (ICE) it was obvious that the USART initialisation was correct yet the problem remained in that area (that of USART initialisation). The insertion of the delays into the program at the initialisation of the USARTs remedied the problem.

The **first delay** was inserted after power up in order that the USART reset line is active for a sufficient period.

The **second delay** was inserted immediately after removing the reset from the USART.

The **third delay** was inserted immediately after the software reset (the internal Reset command 40H) given to the USART. This was found to be absolutely essential in this experiment. Tests were performed with it inserted, and again with the program modified to the original without it. It was found that the experiment was 100% successful with the delay there and less than 10% successful when

the delay was absent.

No specific reason for this problem was found save that there should be a period of recovery (t_{RV}) after Writes to the USART (Intel 1993, Note 4 and Write Cycle table p.2-16 and 2-17). This is conditional for Mode Initialisation only. The **minimum Recovery Time** between Writes for Synchronous Mode is sixteen Clock Periods (16 t_{CY}) which is calculated at 8μs when using the ALE (2 MHz) as the clock (CLK input pin) required by the USART for it's internal timing.

Recovery Time = 16 x CLK Period
= 16 x 1/(Clock frequency)
= 16 x 1/(2 MHz)
= 8μs.

Another modification was performed on the central circuit once TEST52 was completed. The shift registers were moved from Port 1 of the microcontroller, to Port 0 sharing with the USART. This meant that there were now three devices on the data bus. This was possible since the threat of more than one device talking at the same time is prevented because the shift registers are uni-directional devices (input only on the A-H lines). When not selected the input goes high impedance. Program TEST52C was modified to cater for this hardware modification and the test was successful. Since this was a very minor modification the test program is omitted from this dissertation.

3.6.4 Data Link Protocol and Final Software Development

3.6.4.1 Development of Link Protocol

It is necessary to define a set of rules and/or control procedures to be followed by both communicating parties if a reliable data link with valid meaningful data is required. These rules control the operation of the USARTs on either side of the link and will dictate to the controlling program the manner in which the received data is handled and interpreted. The data to be transmitted will also adhere to these rules so that it can be used as meaningful data when received and processed at the other end. There are various schemes and techniques used in this field, each having their own specific applications where best suited. When choosing a protocol for a specific application the advantages and disadvantages of each are compared. There are also various error control techniques that are accepted as standard. Some have error detection without the ability to correct errors while others have built in error-correction.

The field of data link protocols is fairly large and it is not the purpose of this dissertation to cover this subject in great detail.

Error Control.

The data link used by this project is of high enough grade to be considered error free. The Diginet Network is considered to have a bit error rate of 1:1000000 (P. Lochner 1994).

Two basic strategies used in error control are Echo-Checking and **A**utomatic **R**epeat re**Q**uest (ARQ). The former generally used in asynchronous transmission to/from

computer to terminal. In this method everything that is transmitted is re-transmitted back to the source to confirm the validity of the data received. Clearly this is inefficient in bandwidth usage and not suitable for this project, yet applicable for the terminals of remote computers.
There are various modifications on the latter (ARQ), some more complex than others and the decision of which to use usually determined by the buffer storage needed and utilisation of transmission bandwidth of each. Here the basic philosophy is where the receiver acknowledges correct frames with a short signal or message. When an incorrect frame is received the transmitting side will re-transmit the frame on receipt of another such a short message. The two most common variations of ARQ are Idle RQ (send and wait) and the more complex Continuous RQ (Halsall 1990, Chapter 4)

The very basic error control used in this project does not reply to each and every information frame received. Instead the only errors detected are when the received data frame does not comply with the protocol laid out for the frame structure. This violation has two consequences. Firstly in this case a short message is sent to the other side of the link informing the distant sub-system of the error received. Secondly the sub-system receiving the error then resets itself and the program runs right from the power up sequence again. The other sub-system will do the same on receipt of this short message.
Not too much emphasis was placed on error control and this method of control, in the event of an error on the

transmission line, was considered adequate.

Link Protocol.

Data transmission links working with synchronous transmission can either be bit or character oriented. The difference essentially being the method used in determining the start and end of a frame. The bit oriented method allows the receiver to detect the end of a frame at any bit instant whereas the character oriented method limits the end of frame to multiples of character length (Halsall 1990, pp. 85-86).

In this project the data link was character oriented with the character length being 8 bits. The data transmitted on the link will form data frames. These blocks of data characters are transmitted and the first byte of every frame will be used to identify that particular frame. This first byte of each frame will for further discussion be referred to as the Frame IDentifier byte (FID).

The Data to be transmitted by this project over the transmission link is **largely in the direction remote side to central side**. This is because the monitored interface is at the remote location and the monitored information needs to be displayed by the chameleon on the central side. Data transmission in the opposite direction is used purely for link control. Before a link protocol specific for this project could be developed an investigation into what data is required to be transmitted from the remote side had to be conducted. Firstly, the monitored data bytes from each direction (Transmit and Receive) of the monitored interface had to be

transmitted on a regular basis.

Secondly, an attempt must also be made to transmit the additional monitored information, regarding the other monitored interface signals (3.3.3). Since these monitored signals do not change on a frequent basis it was decided that the information regarding these signals only be transmitted on detection by the remote sub-system of a change.

With regards the data bytes from the monitored channels, it was decided against storing a large number or bytes. Because the monitoring system was live, the local side is only able to output data to the chameleon once it has received it. It would be undesirable for the local side to run out of data, as the synchronous clock presented to the chameleon would have to be stopped until the local side received more data. In other words, the period for which the clock would be stopped would be longer but less frequent according to the length of data frames used on the transmission link. One must remember that the clock used to produce the synchronous clocking of the data onto the reproduced interface (Central side) is of a frequency higher than that of the monitored data, i.e., 10 kHz (see 3.2.3). At the remote sub-system all monitoring bytes are first written into queues in RAM before being transmitted to the local side. There are two queues, one for each monitored data stream. When the remote sub-system is ready to transmit a new frame it first has to assemble the FID. Bit 7 of the FID is permanently reset (0). This serves as an identifier for the central receiver (Figure 26). Before the FID is assembled note must be taken of how full the two queues are and whether there has been a change in the other monitored

interface signals. These monitored signals consist of DTR, DSR, RTS, CTS and DCD. These five are combined in a predetermined order and are stored in a byte in RAM at the Remote side. The byte shall be referred to as SIGB (Signal Byte) henceforth.

The program while assembling the FID makes use of a flag (signal byte change SIGBCH) to determine whether the status of the five monitored signals has changed. If this flag has been set, a bit is set in the FID. This indicates to the receiver at the Central side on receipt of the FID that there is an updated SIGB attached at the end of the coming data frame. During FID assembly the remote program also uses counters (CH1QL Channel 1 queue length and CH2QL) to determine how many bytes are awaiting transmission from each queue. Although the queues have maximum storage capacity of 16 bytes each, the assumption is made that there will never be more than 7 bytes awaiting transmission in either queue. These values of queue lengths are placed in the FID in the predetermined positions.

The structure of the data frames in this project is illustrated by Figure 26.

The first byte of every frame is the FID. The next byte/s (up to seven) following the FID is/are that/those monitored from the channel 1 (Transmit Data) of the monitored interface. After these byte/s follow the byte/s (up to seven) from channel 2 (Receive Data) of the monitored interface. This would be the end of a valid data frame received at the central side if (at the time that the FID was assembled) there was no change in the status of the other monitored signals.

If however the SIGB had changed and SIGBCH flag was set, then at the end of the bytes from channel two would follow the new SIGB byte. The central receiver would be expecting this byte and would then update the conditions on the output port accordingly. The receiver at the central side will ignore all sync characters until the next FID is received.

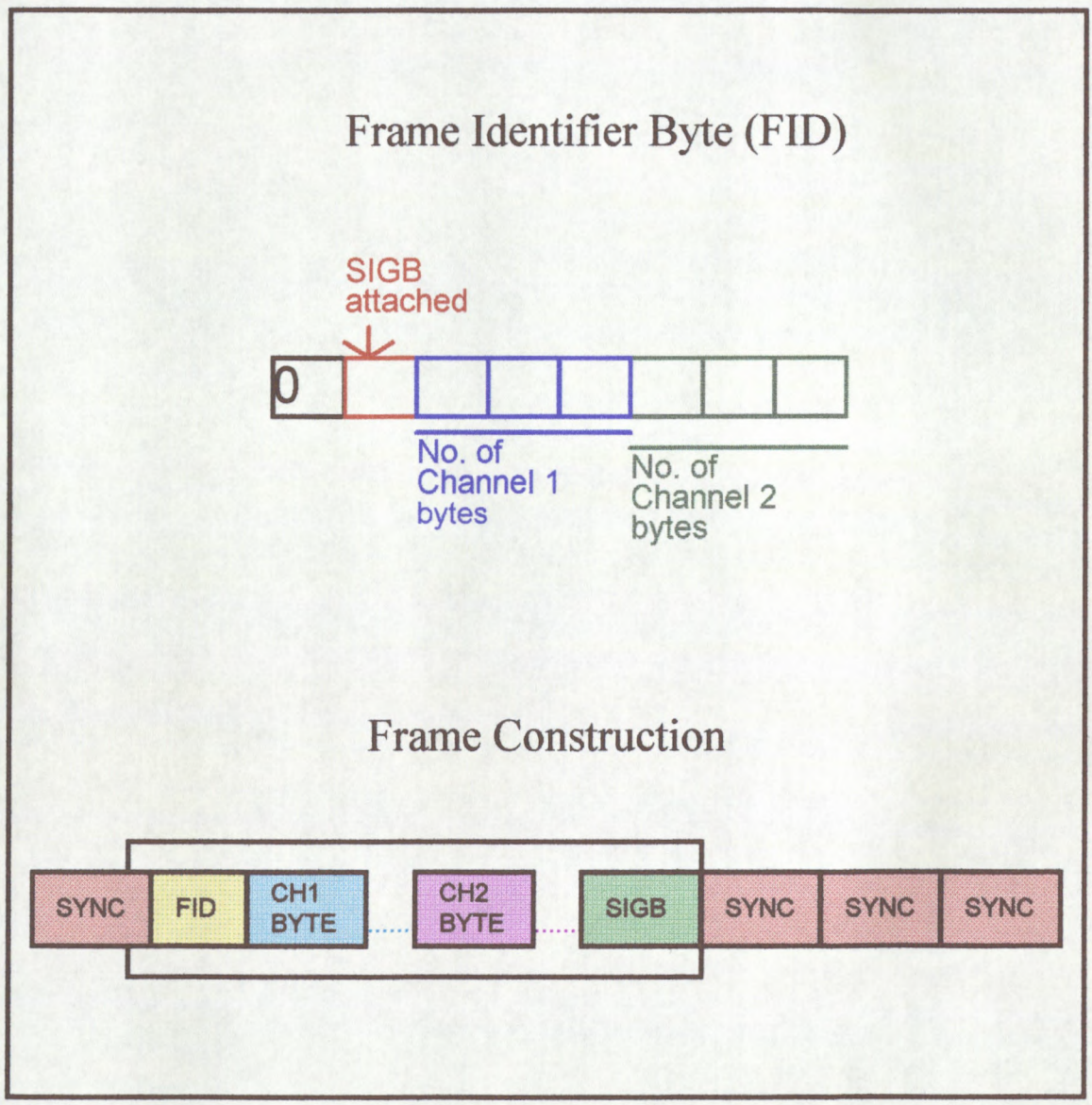


Figure 26

The protocol allows for a test byte to be transmitted from the Central side to the Remote. The Remote, in response to receiving this byte will return a Test Acknowledge byte to the Central side. The receiver must therefore be able to distinguish between the various bytes used in the developed protocol

Summary of Data Frame Construction

The data received by the central sub-system will be a continuous stream of synchronising characters interspersed with data frames (packets or envelopes). The data frame will always conform to a fixed basis. It will always start with a Frame Identifier byte (FID). This byte will give the receiver all the information it requires to interpret the coming frame including it's length.

The FID will then be followed by one or more monitored bytes from Channel 1 and then one or more bytes from Channel 2. Also indicated by the FID will be whether there is an extra byte attached to the end of the channel 2 byte/s which then is used to update the other interface signals (3.3.3). At the end of the data frame will be an unspecified number of synchronising characters until the next FID is received.

3.6.4.2 Development of the Final Software

The flow charts of the final programs indicate the overview of the functions performed. The programs are outlined by the set of flow charts and hence will not be discussed at length except for a few points of interest.

Description of Program TEST6R

TEST6R is the software used by the Remote sub-system's microcontroller in the final stages of the project. The circuit used by this program is outlined in Figure 30.

The main program's basic Flow Chart is found at Figure 27. The Main program performs the following main functions;

a) initialize the communications USART,
b) initialize the two monitoring USARTs,
c) wait for the communications link to synchronise,
d) assemble the Frame Identifier byte when necessary,
e) monitor the stipulated five interface control signals,
f) prepare the frame bytes for transmission in the correct sequence.

The Remote sub-system does not expect a large quantity of received data apart from continuous sync characters. There are only two valid bytes that are received while using this protocol. They are the valid sync character as mentioned and a test request from the central sub-system. Any other byte received will be interpreted as an error.
In the event of an error being detected in the data received from the Remote sub-system the software undergoes a complete restart. For this reason the first few lines of code take care to reset each and every flag including the value of stack pointer. Also one may note that the global interrupt is disabled until the initialisation sequence is complete.

In the program listing Program TEST6R register 3 and register 5 are given RAM addresses. This simplifies the movement of one register's contents to another e.g. at the beginning of the Main program (position ASMBL_FID:) where the contents of registers 3 and 5 are moved to registers 4 and 6 respectively by only 2 commands.

Also of interest is the explanation of the control of the byte sequence in the transmitted frame. In particular the manner in which the Signal Byte (SIGB) is sent at the end of the frame (when necessary). When the main program checks that the prepare next byte (PREPNXT) flag is set, it immediately checks whether there are any bytes of the current frame from Channel One not yet sent. If not, a similar check is conducted on Channel Two. If there are no bytes to be transmitted in this frame from Channel Two, then the signal byte is loaded into the transmit buffer in RAM (NEXTFBYTE).

This may at first seem incorrect as the SIGB should only be transmitted when there has been a change in it's value. However this has all been taken into account by the main program at point **CHK_LAST?**. Here a check is done when the last Channel Two byte is loaded into the transmit buffer (NEXTFBYTE), whether or not there is a new value for SIGB that requires transmission. If there is, then the flag indicating that the buffer contains the last byte in the frame (LASTFBYTE) is not set. Interrupt Zero controls whether the PREPNXT flag is either set or left cleared.

In the case where the last Channel Two byte has been transmitted, the check is made here (after TRANS, at JNB

LASTFBYTE,CHK_REC?) that if the LASTFBYTE is not set then the PREPNXT flag will be set for the sole purpose of the main program loading the SIGB into the transmit buffer at LOADSIGB through the process mentioned above.

TEST6R

BASIC FLOW OF MAIN PROGRAM

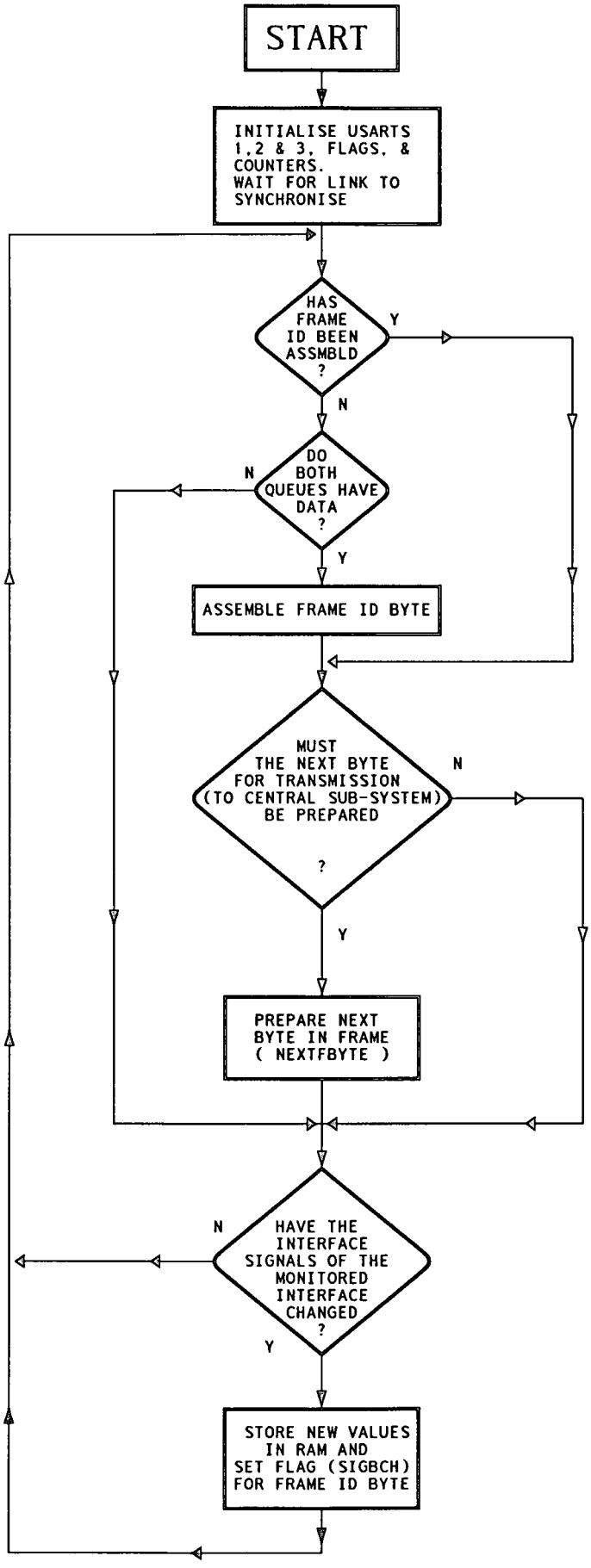


FIGURE 27

The Interrupt 0's basic Flow Chart is found at Figure 28.

This software performs the following main functions;

a) Responds to the communications USART Transmitter Ready (TxRDY) output and transmits either the next byte in the frame, the test acknowledge byte or a sync character.

b) Responds to the communications USART Receiver Ready (RxRDY) output and identifies the received byte as either a sync character, a test request or a reset request.

After a test has been received the test acknowledgment byte is sent when the transmitter ready interrupts and there are no bytes of a frame ready for transmission (at TRANS and at **TESTREP?)**. The test response is given the lowest priority in order to not interfere with the monitoring operation whatsoever. This is besides the fact that the operator at the central side will never notice the time delay in any case. The result is a **transparent test facility** which will prove to the operator that the link is good (relatively error free) and that the Remote sub-system is powered, connected and functional.

TEST6R

BASIC FLOW OF INTERUPT 0

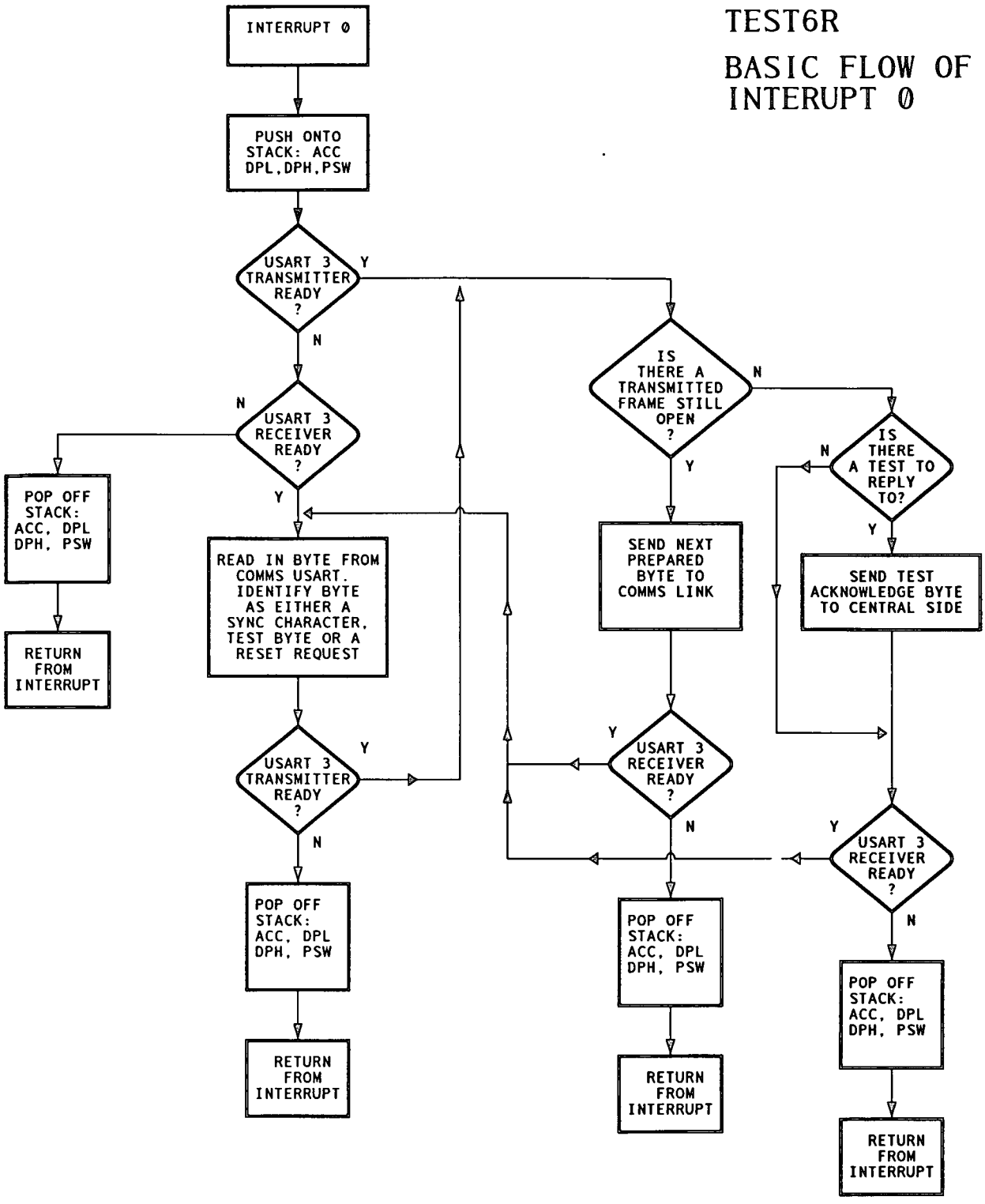


FIGURE 28

TEST6R

BASIC FLOW OF INTERRUPT 1

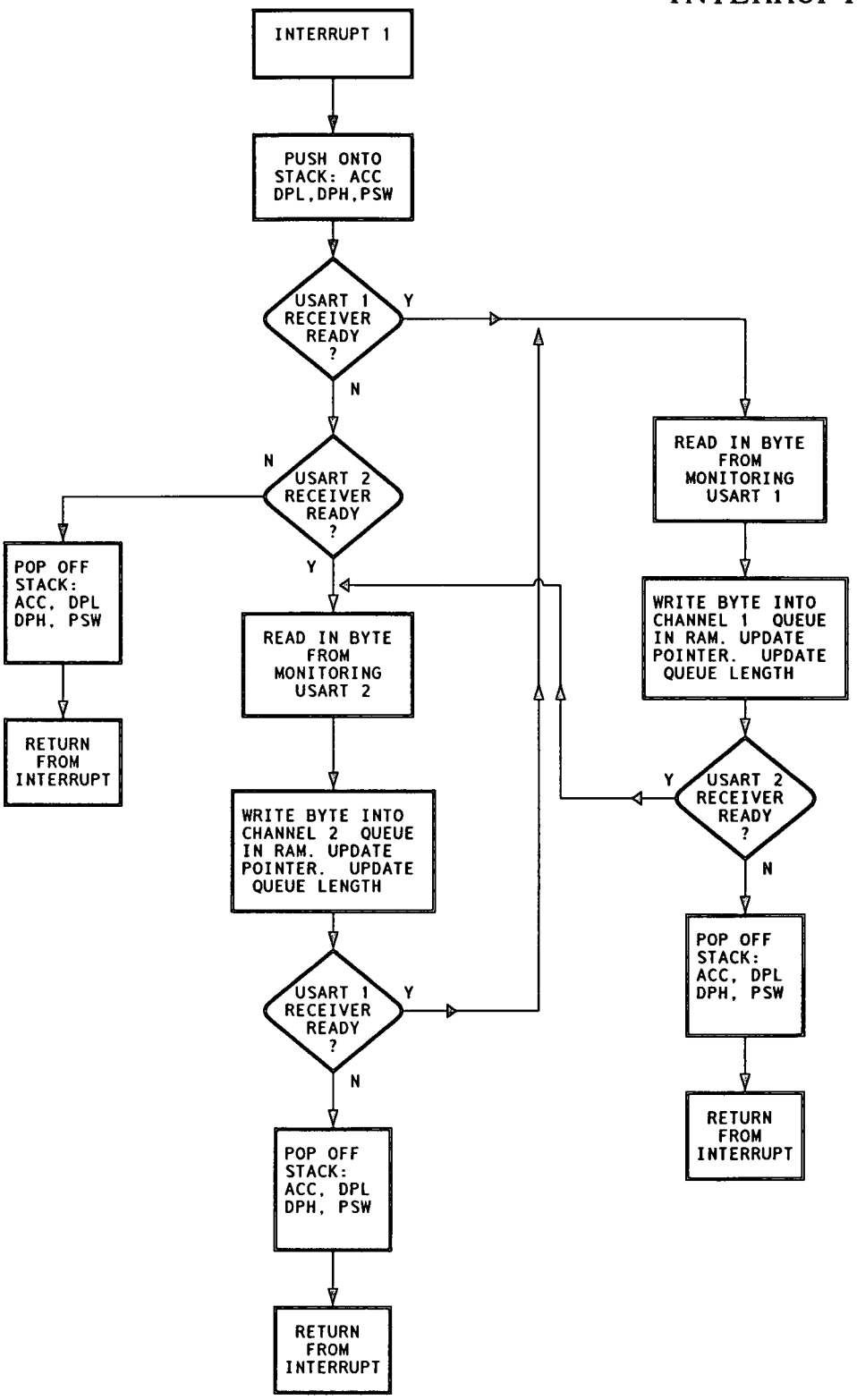


FIGURE 29

The Interrupt 1's basic Flow Chart is found at Figure 29. This software performs the following main functions;

a) Responds to the Receiver Ready outputs from the two monitoring USARTs and decides which USART caused the interrupt.

b) The monitored byte is read from the USART and written into the appropriate queue. This program maintains the write pointers and the queue lengths at their correct values.

REMOTE SUB-SYSTEM
TEST6R

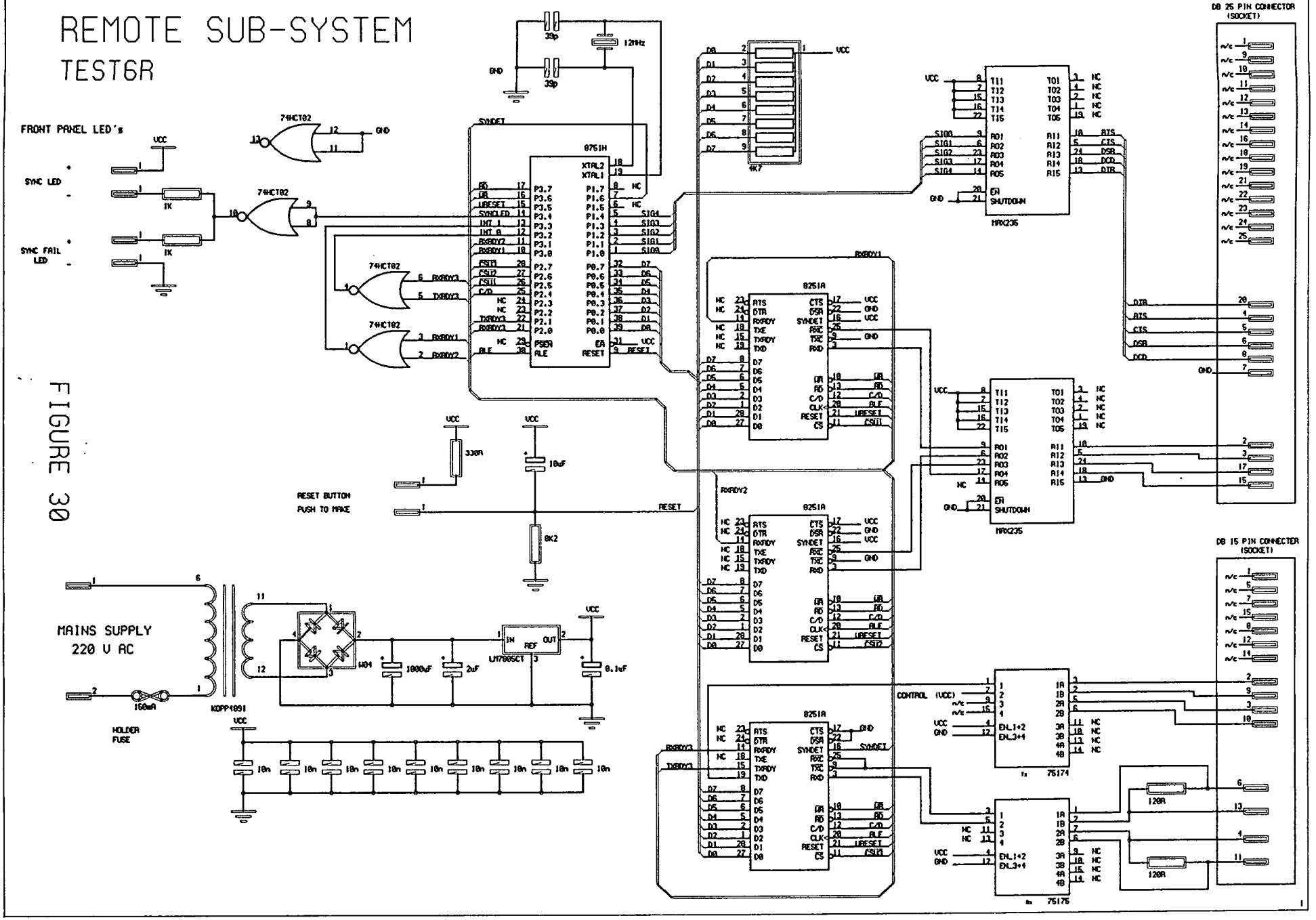


FIGURE 30

Description of program TEST6C

TEST6C is the software used by the central sub-system during TEST6. The circuit schematic used is shown by Figure 33.

The main program's basic Flow Chart is found at Figure 31.

The main program performs the following main functions;

a) initialize the communications USART,
b) read the bytes from the two queues and load them into the appropriate parallel shift-registers,
c) control the clock latch circuitry,
d) respond to the test button being depressed by transmitting a test byte to the remote side.

The Interrupt 0's basic Flow Chart is found at Figure 32. This program mainly responds to the Receiver Ready output from the USART, interprets the FID and prepares various counters for the reception of the coming frame.

Only Interrupt 0 is used in this program as only data in the receive direction is of high importance. The bytes received are interpreted by the interrupt routine. The bytes must fall into 4 categories; namely sync characters, FID bytes, frame data bytes (any byte value but distinguishable by the fact that the RFO, Receive Frame Open flag, is set) or test acknowledgment bytes. Any other byte is seen as a receive error and causes the software to restart from scratch.

TEST6C

BASIC FLOW OF MAIN PROGRAM

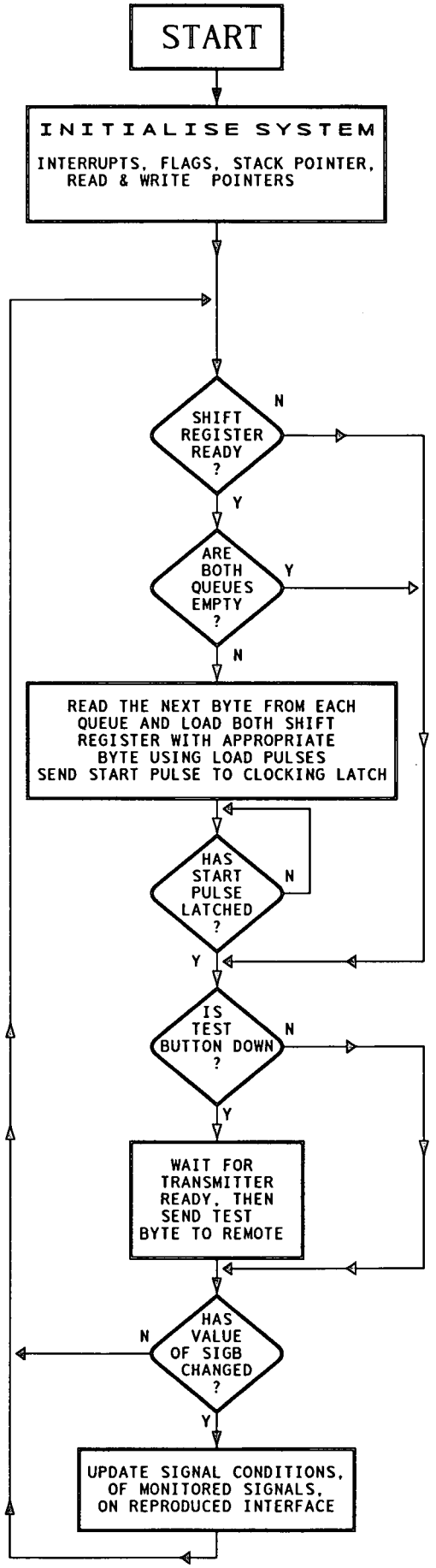


FIGURE 31

TEST6C

BASIC FLOW OF INTERRUPT PROGRAM

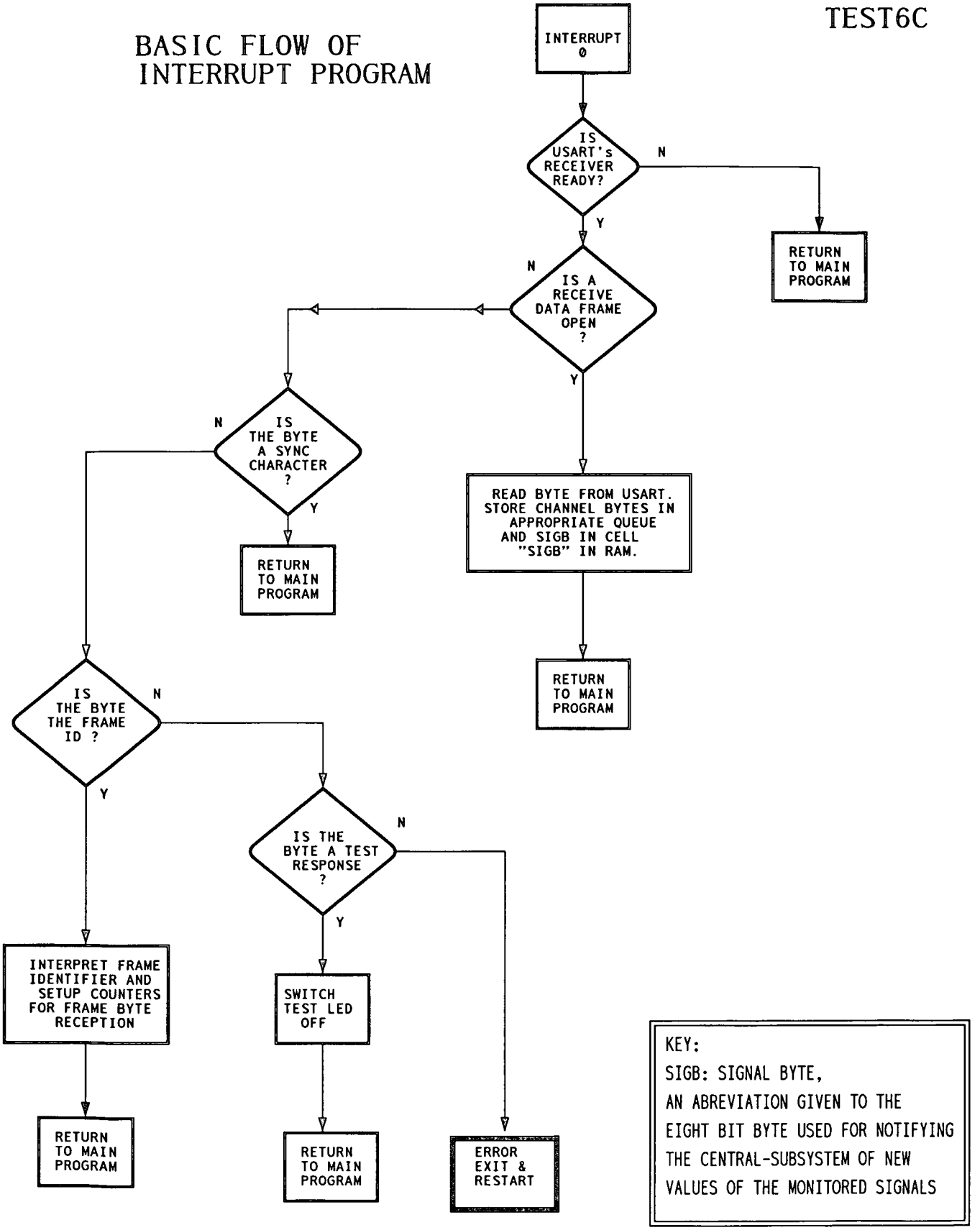
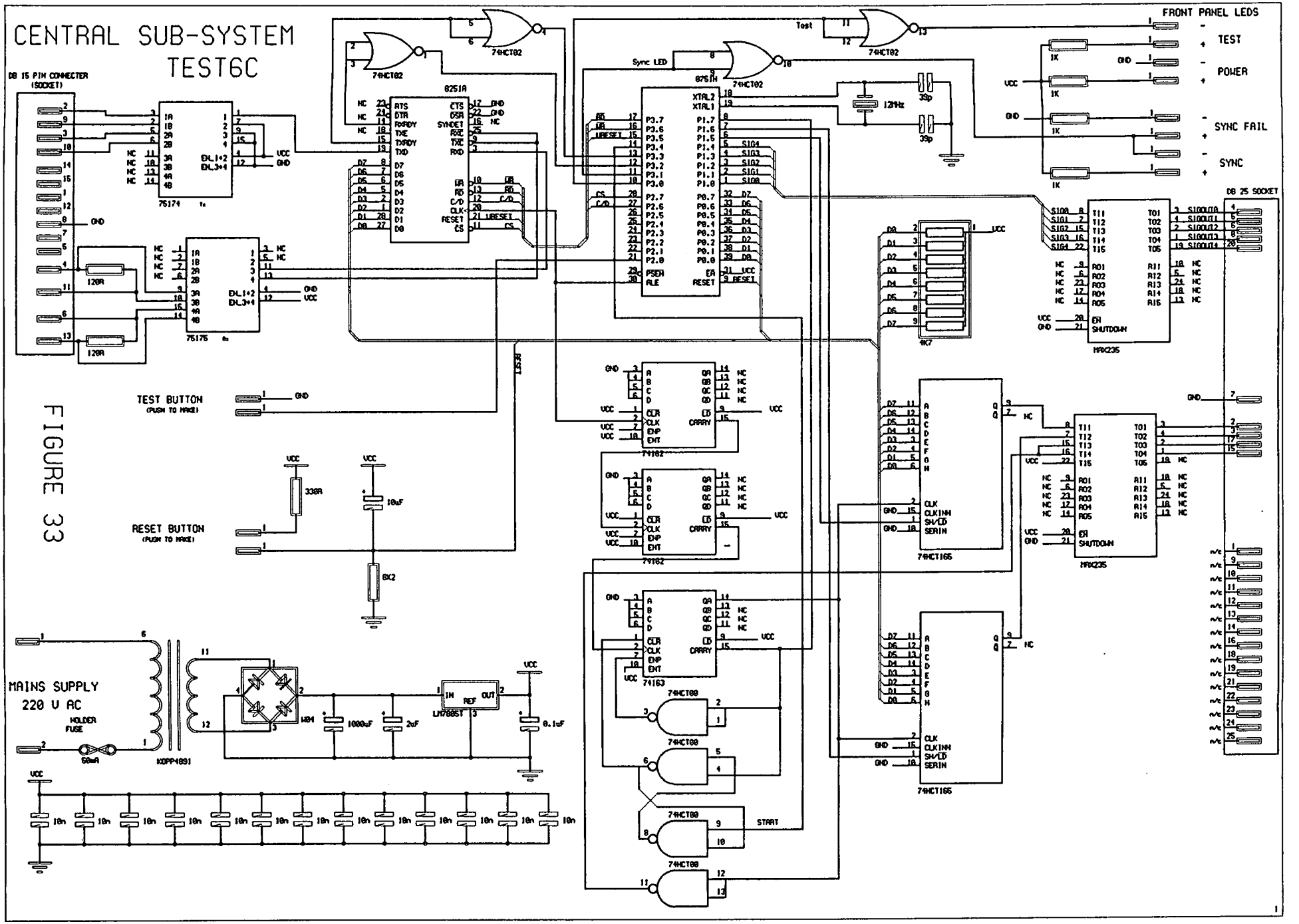


FIGURE 32


CENTRAL SUB-SYSTEM
TEST6C
DB 15 PIN CONNECTER
(SOCKET)
75174
75175
8251A
74HCT02
Sync LED
Test
12MHz
39p
FRONT PANEL LEDS
TEST
POWER
SYNC FAIL
SYNC
DB 25 SOCKET
MAX235
74HCT165
74HCT00
74163
74162
TEST BUTTON
(PUSH TO MAKE)
RESET BUTTON
(PUSH TO MAKE)
MAINS SUPPLY
220 V AC
HOLDER
FUSE
50mA
KBPP4B91
LM7805T
START
FIGURE 33

3.7 Construction of the System Prototype

After the successful testing of the project , the author transferred the final designed and tested circuits onto a permanent arrangement in the form of a Printed Circuit Board (PCB). This was done in fulfillment of sub-goal II (outlined in 2.2.2.2). The aim was to develop a **prototype** of a system for possible reproduction on a small scale (2.2.5. a). For details regarding the construction of the prototype PCB refer to **PCB Design and Construction** in Appendix A.

CHAPTER FOUR: PERFORMANCE TESTING

4.1 The Test Configuration and Conditions

The technology developed in this study, complete with all the final modifications, was tested in the field on site at the Durban Saponet Test Centre, Pine Street. The interface which simulated the remote interface being monitored, was synchronous at 9600 bps, and operated on the X.25 protocol. This interface was accessed via the Patch Panel. This was then connected to the Remote sub-system and also directly to Chameleon B which was used to verify the display on Chameleon A. The developed sub-systems were each connected to an NTU via the V.11 interface cable. The NTU's were connected "back to back" in order to simulate the communications link over the physical separation of the Central and Remote sub-systems. The "reproduced" interface was then connected from the output of the Central sub-system to the Chameleon (A).

4.2 The Test Procedure

In order to evaluate the performance and success of the system, the displays of the two Chameleons, (Chameleon A with the data interface from the central sub-system, and Chameleon B with the original interface) would have to be compared.

The Chameleons have a data capture and storage facility with a capacity of 10 Megabytes. This provides the Chameleon with an ability of storing approximately 2 hours of data traffic, depending on speed and density (Tekelec 1988, p. A-2). Since the data rate was slightly fast for the actual

packets displayed by the Chameleons to be carefully compared, the settings of the Chameleons' displays were altered in the "history" mode such that the displays were scrolled at the same pace. This meant that the Chameleons were actively monitoring the data interfaces but that the information being displayed was actually being retrieved from memory.

The test was run on three separate occasions for a period of between 3 and 5 hours. During this time neither Chameleon interpreted any **errors** in the data presented to them. The data packets were buffered in the Chameleons and were scrolled to the screen at a rate allowing thorough **visual** comparison. No discrepancies in the two displays were detected during this time and the developed system proved to be 100 % reliable under these test conditions.

To improve the validity of the testing of the developed technology, all testing of the developed system was conducted under the exact conditions for which the system was designed. For the data test patterns used the reader can contact the author at Telkom, 296 Pine Street, Durban, R.S.A. (Tel +27-31-3631863).

CHAPTER FIVE: CONCLUSIONS

With reference to Figure 1b, the developed system had to comply with the existing environment and therefore monitor only synchronous interfaces. The interfaces of the services; Dial-up Triple -X V.32bis, Easy Access and Dedicated Triple-X, are all asynchronous. However the data on those interfaces is not in the X.25 protocol, and hence it would be pointless to monitor them.

All the X.25 Saponet users' interfaces are **synchronous** and as mentioned in section 1.2 the vast majority of them operate at **9600 bps**. For this reason the design of the prototype was restricted to monitor these interfaces and not asynchronous interfaces.

From the results obtained from the system performance tests detailed in Chapter 4, it can be reasonably concluded that the system;

a) does not affect the data flow on the monitored interface in any way,
b) successfully monitors synchronous interfaces,
c) successfully monitors interfaces operating at a speed of 9600 bps (above which the operation is not guaranteed)
d) successfully complies with the CCITT V.24 and RS232C level at the monitoring interface as well as at the reproduced interface,
e) successfully operates as a data transparent monitor,
f) successfully controls the link and transmits the information in a form acceptable to the other side,
g) successfully presents the signals to the reproduced interface in a manner acceptable to the Chameleon protocol analyser.

The system specifications and design criteria were investigated and are fully documented in section 3.2 of this dissertation. This in turn fulfilled

sub-goal I (2.2.2.1) in that it concluded the task of investigating the requirements and specifications of the proposed system in order to establish the hardware and software design parameters.
The hypothesis made at the outset of the study (2.2.3.1), which stated; that it would be possible to establish the requirements and criteria that the system should meet and thereby establish the design parameters was validated through the successful conclusion of sub-goal I.

The system was developed through progressive test stages (Test 1 through Test 6, see 3.3, 3.4, 3.5 and 3.6) which resulted in circuit hardware (Figures 30 and 33) that does facilitate remote interface monitoring over a data transmission line. This resultant hardware from successive testing and development served to fulfill the sub-goal II (2.2.2.2) which required that the hardware to facilitate remote interface monitoring be designed and built. The second hypothesis made (2.2.3.2) which stated that; it would be possible to design and build the hardware that would be suitable to facilitate remote interface monitoring using a data transmission line, was validated through the fulfillment of sub-goal II (2.2.2.2).

The system's controlling software was developed in parallel with the hardware (Test 1 through Test 6, see 3.3, 3.4, 3.5 and 3.6). The developed software was tested successful in;

a) controlling the hardware of the developed remote and central sub-systems
b) processing the monitored data at the remote sub-system (into a form suitable for transmission over the data transmission link)
c) processing the data received at the central sub-system (in order to reproduce part of the monitored interface).

This can be stated from the results of the tests run in the field at the test centre for X.25 circuits in Durban (Chapter Four). Since the execution of

points a, b and c above were fulfilled by the developed software, sub-goal III laid down at the outset of the study (2.2.2.3) was also fulfilled.
This in turn validated the third hypothesis made, which stated that it would be possible to develop a method of processing the monitored data and to develop the software that would perform the points a, b and c above (see 2.2.3.3).

Bearing in mind that the study developed a **prototype** of a system from which the technology could be used for further development and/or minor modification, and not a commercially marketable product, the testing was not overly extensive and aimed to verify the concepts used in the design of the equipment. Therefore no formal specifications for the system were presented. A system with the same functionality can be reproduced from the details and data presented in this document.
There is a facility (refer to 3.6.4.2) provided for the operator to confirm that the remote interface monitor is powered and functional and that the link between the local and remote sub-systems is operational. This is a simple procedure which the local operator can execute by depressing a button and observing the response on the test LED.
When the performance of the system is evaluated in terms of the design criteria established in section 3.2, the developed technology and the physical system can be regarded successful. This is due to the fact that it fulfilled the design requirements and is functional to the end for which it was designed. The evaluation was done in fulfillment of the task set in sub-goal IV of paragraph 2.2.2.4.
The hypothesis (2.2.3.4) which stated that the developed system would meet the performance criteria (3.2) was validated through the successful tests performed in Chapter 4.

Significance of the Successful Prototype

The successful implementation of the technology developed in this study could have a marked impact on the testing of X.25 circuits connected to the remote nodes. These circuits can, through the use of a system based on this prototype, now be accurately monitored from remote locations.

Discussed in the following paragraphs are only some of the benefits gained from the use of a remote interface monitor system, based on this developed prototype.

Delays incurred when attempting to explain procedures and protocol interpretation telephonicaly to remote staff will be avoided and a faster connection to the protocol analyser will be one of the advantages as a result of the developed system.
A more accurate assessment by the test Technicians of the fault conditions will also result from the application of this system.
These benefits are in addition to those already outlined in 2.1.4.

The accumulative effect of all of the benefits gained, as a result of the development of a method of monitoring remote interfaces, should positively impact the service provided by Telkom S.A. to their X.25 clients.
This system of remote interface monitoring is planned to be implemented in other provinces in addition to Natal. This will increase the positive significance of the study further.

Due to the uniqueness of the problem there is limited documentation available on the subject. For this reason the reference listing is not extensive.

Appendix

A.1 PCB Design and Construction

Computer Aided Design

The software design package P-CAD, a registered trademark of Personal CAD Systems Inc, was used in this development. The circuit schematic diagrams were drawn using the package PC-Caps. This is the package with which all the circuit schematics in this dissertation are drawn. The circuit schematics were then processed using PC-Nodes which generated the nodes and net-listings with the parts information. PC-Pack was then used to load the part information and package it into a resultant PCB file with extension "pcb". The PCB design package used for the PCB design and component layout was PC-Cards. The "pcb" files were now manually routed. Attention was given to the routing of the board in view of noise particularly when routing the power. Each and every IC had a de-coupling cap as close to the power pins as possible (Intel 1992a, pp. 6-2 - 6-21) The schematic and PC Board layouts were designed by the author.

From these PCB layout files the top and bottom layers of each board was plotted onto plotting film. The scale 1:1 was used throughout the PCB construction.

Printed Circuit Board Construction

The plots of the top and bottom layers described above were used to expose the negative film using Ultra Violet light (period of 6 seconds). The negatives were developed in developer liquid and then cured in fixer solution. This produced the negatives of the track layouts of both layers of the two boards.

A drill plot was plotted and used to drill the holes in the bare double sided copper plated board. These boards were then laminated on both sides with riston. The negatives created above were then used to expose the riston laminated boards under Ultra Violet for 40 seconds. The riston on the boards was then soaked in a caustic soda solution which removed the riston from areas where the copper was to be etched. The boards were etched in a Ferric Chloride solution where unwanted copper was removed from the boards. The remaining riston was removed from the tracks on the boards by a Potassium Hydroxide solution.

The finished boards were then lightly smeared with solder flux and "tinned" for the following benefits

a) assisting in the elimination of "Dry joints" in the soldering process,
b) assisting in the inspection of the newly etched board in order to locate breaks in tracks,
c) assisting in reducing electrical noise by decreasing the PCB track impedance (through increasing the cross sectional area of the track).

Power Supply Design

During the testing and development of the circuits (while they were still on project or bread-board) portable power supplies were used to provide power to the sub-systems.
However when the PCBs were being designed consideration had to be given to the power requirements of the circuits and to design an appropriate on-board supply.

Measurements of the supply current to each sub-system were taken using a digital electronic multimeter (Fluke 77). A readily available transformer

was selected according to it's power specifications. The current readings were used in calculations (Boylestad 1982, p. 575) together with these specifications to investigate the feasibility of it's use in this power supply. It's rated specifications were;

Secondary Voltage: 9.5 V

@ Secondary Current: 580 mA. (Kopp 1993)

A.2 Selected Programs used in this Development

```
;*************************************************************************
;                    PROG 6R                                              *
;                    REMOTE                    ** BBR_61 **               *
;                                                                         *
;-SENDS A BYTE TO ENABLE SYNC AT BOTH ENDS BEFORE CONTINUING              *
;-DELAYS AFTER USART RESET PULSE AND COMMAND                              *
;-INTERUPT 0 USED FOR TxRDY AND RxRDY ON LINK (HIGHER PRIORITY)           *
;-INTERUPT 1 MONITORS USART 1 + 2 AND AND WRITE IN TWO SEPERATE           *
; QUEUES                                                                 *
;-MONITORS INTERFACE CONTROL SIGNALS (SIGB IF CHANGED INCL AT END OF      *
;FRAME)                                                                  *
;-RESPONDS TO TEST BYTE FROM LOCAL WITH ACKNOWLEDGMENT BYTE               *
;*************************************************************************


;PORT PINS

                    ;P1.0 MONITORED INTERFACE SIGNALS (SIGBYTE)
                    ;P1.1 MONITORED INTERFACE SIGNALS (SIGBYTE)
                    ;P1.2 MONITORED INTERFACE SIGNALS (SIGBYTE)
                    ;P1.3 MONITORED INTERFACE SIGNALS (SIGBYTE)
                    ;P1.4 MONITORED INTERFACE SIGNALS (SIGBYTE)
                    ;P1.5 NC
SYNDETP EQU 096H    ;P1.6 SYNDET PIN FROM COMMS USART 3
                    ;P1.7 NC
```

```
RXRDY3 EQU 0A0H     ;P2.0
TXRDY3 EQU 0A1H     ;P2.1
                    ;P2.2 NC
                    ;P2.3 NC
                    ;P2.4 CONTROL NOT DATA LINE  COMMON TO U1,2 AND 3
                    ;P2.5 USART 1 CHIP SELECT
                    ;P2.6 USART 2 CHIP SELECT
                    ;P2.7 USART 3 CHIP SELECT

RXRDY1 EQU 0B0H     ;P3.0 RXRDY PIN FROM USART 1
RXRDY2 EQU 0B1H     ;P3.1 RXRDY PIN FROM USART 2
                    ;P3.2 INTERRUPT 0 COMMS USART3 (RXRDY) NOR (TXRDY)
                    ;P3.3 INTERRUPT 1 (USART1 RXRDY) NOR (USART2 RXRDY)
SYNCLED EQU 0B4H    ;P3.4 THROUGH NOR GATE TO GIVE VISUAL LINK STATUS
URESET EQU 0B5H     ;P3.5 RESET LINE COMMON TO USART 1,2 AND 3
                    ;P3.6 WRITE. USED FOR USARTS
                    ;P3.7 READ. USED FOR USARTS

; REGISTERS
                    ;R0 USED FOR INDEXING THE POINTERS IN MAIN PROG
                    ;R1 USED FOR INDEXING THE PNTRS IN INTERRUPT 1 PROG
COUNTER EQU R2      ;R2
CH1QL EQU R3        ;No. OF BYTES IN CH 1 QUEUE   (CH1 QUEUE LENGTH)
CH1FL EQU R4        ;No. OF CH 1 BYTES IN FRAME  (CH1 FRAME LENGTH)
CH2QL EQU R5        ;No. OF BYTES IN CH 2 QUEUE   (CH2 QUEUE LENGTH)
CH2FL EQU R6        ;No. OF CH 2 BYTES IN FRAME   (CH2 FRAME LENGTH)
CH1QLRA EQU 03H     ;(RAM ADDRESS OF CH1QL USED FOR MOVING TO R4
CH2QLRA EQU 05H     ;(RAM ADDRESS OF CH2QL USED FOR MOVING TO R6
```

```
; RAM ADDRESSES

TESTRX EQU 00H        ;FLAG SET ONCE TEST IS RECEIVED FROM LOCAL
SIGBCH EQU 02H        ;FLAG SET IF SIGB CHANGES, ∴ ADD TO END OF FRAME
SIGBCHF EQU 03H       ;FLAG SHOWS SIGB CHANGE INCL IN FID ASSMBLY
FIDRDY EQU 04H        ;FLAG SHOWS FID IS ASSMBLD & RDY TO OPEN Tx FRAME
PREPNXT EQU 06H       ;FLAG SET IF NEXT FRAME BYTE SHOULD BE PREPARED
LASTFBYTE EQU 07H     ;FLAG TELLS TRANS ROUTINE LAST BYTE IN
                      ;FRAME AND HENCE MUST RESET ALL FRAME
                      ;TRANSMISSION FLAGS
SIGB EQU  050H        ;BYTE HOLDS MONIOTORED INTERFACE SIGNAL STATUS
NEXTFBYTE EQU 051H    ;NEXT FRAME BYTE FOR TRANSMISSION
WNXT1 EQU 052H        ;HOLDS VALUE OF WRITE NEXT POINTER FOR CH1 QUEUE
RNXT1 EQU 053H        ;HOLDS VALUE OF READ NEXT POINTER FOR CH1 QUEUE
WNXT2 EQU 054H        ;HOLDS VALUE OF WRITE NEXT POINTER FOR CH2 QUEUE
RNXT2 EQU 055H        ;HOLDS VALUE OF READ NEXT POINTER FOR CH2 QUEUE


;CONSTANTS

;USART CONTROL USING LINEAR ADDRESSING (BIT"0-3"="F" TO ENABLE PINS AS
;INPUTS)
U1D EQU 0CFH              ;USART 1 DATA ADDRESS
U1C EQU 0DFH              ;USART 1 CONTROL ADDRESS
U2D EQU 0AFH              ;USART 2 DATA ADDRESS
U2C EQU 0BFH              ;USART 2 CONTROL ADDRESS
U3D EQU 06FH              ;USART 3 DATA ADDRESS
U3C EQU 07FH              ;USART 3 CONTROL ADDRESS

BOUNDARY1 EQU 040H        ;FIRST BYTE OF RAM ABOVE CH1 QUEUE
```

```
BOUNDARY2 EQU 050H      ;FIRST BYTE OF RAM ABOVE CH2 QUEUE
Q1ST EQU 030H           ;CH 1 QUEUE START ADDRESS IN RAM
Q2ST EQU 040H           ;CH 2 QUEUE START ADDRESS IN RAM

TESTBYTE EQU 0EEH       ;TEST BYTE FROM LOCAL
TESTACK EQU 0ACH        ;TEST RESPONSE BYTE SENT TO LOCAL
;=========================================================================

            ORG 0000H
            AJMP BEGIN
            ORG 0003H               ;VECTOR FOR EX INT 0
INT_0:      AJMP TX_RX?

            ORG 0013H               ;VECTOR FOR EX INT 1
INT_1:      AJMP CH1_CH2?

;**** HARDWARE AND SOFTWARE  RESET OF USART *****

BEGIN:      CLR EA
            CLR EX0
            CLR EX1
            SETB URESET
            CLR TESTRX              ;INITIAL VALUE OF FLAG
            CLR SYNCLED             ;DISPLAY WAITING TO ACHIEVE
                                    ;SYNC
            CLR SIGBCH              ;INITIAL VALUE OF FLAG
            CLR SIGBCHF             ;INITIAL VALUE OF FLAG
            CLR PREPNXT
            MOV CH1QL,#00H          ;INITIALISE Q1 LENGTH COUNTER
            MOV CH1FL,#00H          ;INITIALISE FRAME 1 LENGTH
```

```
                MOV CH2QL,#00H          ;INITIALISE Q2 LENGTH COUNTER
                MOV CH2FL,#00H          ;INITIALISE FRAME 2 LENGTH
                                        ;COUNTER
                MOV SP,#007H            ;RESET STACK POINTER AFTER
                                        ;ERROR
                MOV WNXT1,#Q1ST         ;INITIALISE POINTERS
                MOV RNXT1,#Q1ST         ;INITIALISE POINTERS
                MOV WNXT2,#Q2ST         ;INITIALISE POINTERS
                MOV RNXT2,#Q2ST         ;INITIALISE POINTERS

RDELAY:         MOV COUNTER,#0FFH       ;
DEL1:           DJNZ COUNTER,DEL1       ;RESET PULSE MIN OF 6 CLK
                                        ;PERIODS
                CLR URESET              ;TAKE H/W RESET OFF USART  ** **
DEL2:           DJNZ COUNTER,DEL2       ;RESET RECOVERY DELAY

DUMMYRUN:       MOV COUNTER,#03H        ;LOAD COUNTER
IDLE:           MOV A,#00H              ;CONTROL BYTE
                MOV DPH,#U1C
                MOVX @DPTR,A
                MOV DPH,#U2C
                MOVX @DPTR,A
                MOV DPH,#U3C
                MOVX @DPTR,A
                DJNZ COUNTER,IDLE       ;WRITE 00H 3 TIMES, S/W
                                        ;INITIALISATION

                MOV A,#040H             ;INTRNL RST U1+2+3,  BACK TO
                                        ;MODE
                MOVX @DPTR,A            ;U3 WAS LOADED INTO DPTR
```

```
                MOV DPH,#U2C            ;U2
                MOVX @DPTR,A
                MOV DPH,#U1C            ;U1
                MOVX @DPTR,A

MODE:           MOV COUNTER,#00FH       ;RESET PULSE, MIN OF 6 CLK periods
DELAY1:         DJNZ COUNTER,DELAY1

                MOV A,#0CCH             ;MODE INSTRUCTION FORMAT
                                        ; U1 + U2
                MOV DPH,#U1C
                MOVX @DPTR,A            ;U1 WAS LOADED IN DPTR
                MOV DPH,#U2C            ;U2
                MOVX @DPTR,A

                MOV A,#00CH             ; MODE INSTRUCTION FORMAT U3
                MOV DPH,#U3C
                MOVX @DPTR,A

SYNC:           MOV A,#0BCH
                MOVX @DPTR,A            ;WRITE SYNC CHARACTER U3
                MOVX @DPTR,A            ; SECOND

                MOV DPH,#U1C            ;SYNC TO U1
                MOVX @DPTR,A
                MOV DPH,#U2C            ;SYNC TO U2
                MOVX @DPTR,A

COMMAND:        MOV A,#06H              ;COMMAND INSTRUCTION U1 + U2
                MOVX @DPTR,A            ;COMMAND TO U2,
```

```
                MOV DPH,#U1C            ;  "   " U1
                MOVX @DPTR,A

                MOV A,#0A7H             ; COMMAND   U3
                MOV DPH,#U3C
                MOVX @DPTR,A

;-------------------------------------------------------------------------------------------------------
;               END OF USART INITIALISATION
;       NOW ACHIEVE SYNC AT THE COMMS USART
;-------------------------------------------------------------------------------------------------------

WRITER:     MOVX A,@DPTR            ;READ STATUS OF COMMS LINK
            RRC A
            JNC WRITER              ;USART IS TAKEN OUT OF HUNT
            MOV DPH,#U3D            ;START SYNC CHARCTRS
            MOVX @DPTR,A            ;SYNC CHARACTERS WILL NOW go...
                                    ;...OUT AND THE LINK CAN
                                    ;SYNCHRONISE

GETSYNC:    JNB SYNDETP,GETSYNC     ;SYNC ACHIEVED, NOW CONTINUE
DISPLAY:    SETB SYNCLED            ;DISPLAY SYNC ACHIEVED

EN_INT:     SETB PX0                ;GIVE EXT INT 0 PRIORITY 1
            CLR PX1                 ;GIVE EXT INT 1 PRIORITY 0
            SETB EX0                ;ENABLE COMMS INTERRUPTS
            SETB EX1                ;EN USART 1+2 INTRPTS FOR
                                    ;MONITORING
            SETB EA             ;ENABLE  INTERPTS CLEARED AT BEGIN
```

```
;-------------------------------------------------------------------------------------------------
;                MAIN PROGRAM
;-------------------------------------------------------------------------------------------------

MAIN:          JB FIDRDY,PREP_NXT?
               MOV A,CH1QL              ;IF CH1QL=ZERO,  DONT CONTINUE
                                        ;WITH..
               JZ CHK_SIGB              ;.. FRAME IDENTIFIER ASSEMBLY
               MOV A,CH2QL              ;IF CH2QL=ZERO THEN DONT
                                        ;CONTINUE
               JZ CHK_SIGB              ;.. WITH FRAME IDENTIFIER
                                        ;ASSEMBLY

ASMBL_FID:     MOV CH1FL,CH1QLRA        ;SET THE LENGTH OF CH1 FRAME
                                        ;BYTES
               MOV CH2FL,CH2QLRA        ;SET THE LENGTH OF CH2 FRAME
                                        ;BYTES
               MOV A,CH2FL
               RL A
               RL A
               RL A
               ORL A,CH1FL
               JNB SIGBCH,SET_FIDRDY    ;IF NO CHANGE IN SIGB,END FID
                                        ;ASEMBLY
               ORL A,#040H              ;IF SIGBCH IS SET THEN SET BIT IN
                                        ;FID
               SETB SIGBCHF             ;INDICATE THAT FID INCLUDES
                                        ;SIGB AT end of frame...

SET_FIDRDY:    MOV NEXTFBYTE,A          ;FID IS NEXT BYTE FOR Tx
```

```
                SETB FIDRDY             ;SHOW THAT FID IS ASSMBLED
;IN THIS ORDER SO IF INTERRUPTED THEN THE FID WILL ALREADY BE
; LOADED

PREP_NXT?:      JNB PREPNXT,CHK_SIGB    ;IF NOT SET THEN SKIP PREP NXT

MORECH1?:       MOV A,CH1FL             ;CHECK IF MORE CH1 BYTES FOR...
                                        ;... THIS FRAME
                JZ MORECH2?             ;IF NOT, CHECK FOR CH2
                MOV R0,RNXT1            ;LOAD NEXTFBYTE WITH NEXT
                                        ;BYTE..
                MOV NEXTFBYTE,@R0       ;...FROM CHANNEL 1 QUEUE
                CLR PREPNXT             ;BYTE PREPARED, SO RESET FLAG
                DEC CH1QL               ;UPDATE CH 1 QUEUE LENGTH
                DEC CH1FL               ;DECREASE CH 1 FRAME BYTES left
                INC RNXT1               ;UPDATE CH1 READ POINTER
                MOV A,RNXT1             ;CHECK AND KEEP PNTR IN VALID
                                        ; RANGE
                CJNE A,#BOUNDARY1,CHK_SIGB
                MOV RNXT1,#Q1ST
                AJMP CHK_SIGB
MORECH2?:       MOV A,CH2FL             ;CHECK IF MORE CH2 BYTES FOR...
                                        ;... THIS FRAME
                JZ LOADSIGB             ;IF NO MORE THEN GOTO LOADSIGB
                MOV R0,RNXT2            ;LOAD NXT CH2 BYTE FOR Tx
                MOV NEXTFBYTE,@R0
                CLR PREPNXT             ;BYTE LOADED SO CLEAR FLAG
                DEC CH2QL               ;UPDATE CH 2 QUEUE LENGTH
                INC RNXT2               ;UPDATE CH2 READ POINTER
                MOV A,RNXT2             ;CHECK AND KEEP PNTR IN VALID
```

```
                                        ;RANGE
                CJNE A,#BOUNDARY2,CHK_LAST?
                MOV RNXT2,#Q2ST
CHK_LAST?:      DJNZ CH2FL,CHK_SIGB     ;DECREASE No. OF CH 2 FRAME
                                        ; BYTES
                                        ;...LEFT FOR TRANSMISSION. GO &
                                        ; CHECK.MONITORED SIGNALS'
                                        ;CONDITIONS
                JB SIGBCHF,CHK_SIGB     ;IF NO MORE CH2 FRAME BYTES
                                        ;AND .NO CHANGE IN SIGB THEN
                                        ;THIS
                SETB LASTFBYTE          ;.. IS LAST BYTE IN FRAME, SO SET
                                        ; FLAG
                AJMP CHK_SIGB

;
;***THERE ARE NO MORE CH1 NOR CH2 BYTES IN THIS FRAME YET THE
;(PREPNXT) FLAG WAS SET MEANING THE LAST BYTE IN THIS FRAME IS SIGB

LOADSIGB:       MOV NEXTFBYTE,SIGB      ;LOAD SIGB TO BE SENT AT
                                        ;END OF FRAME
                CLR PREPNXT
                SETB LASTFBYTE
                CLR SIGBCH
                CLR SIGBCHF
CHK_SIGB:       MOV A,P1                ;GET CURRENT SIGNAL conditions
                ORL A,#0E0H             ;MASK UNUSED BITS 5,6 & 7
                CJNE A,SIGB,SETFLAG     ;CHECK IF SIGB HAS CHANGED
                AJMP MAIN               ;IF SIGB IS UNCHANGED THEN
                                        ;GOTO MAIN
```

```
SETFLAG:    SETB SIGBCH             ;IF SIGB HAS CHANGED THEN
                                    ;SETFLAG
            MOV SIGB,A              ;STORE NEW VALUE OF SIGB IN ram
            AJMP MAIN               ;GOTO BEGINNING OF MAIN program

;-------------------------------------------------------------------------------------------------
;       INTERRUPT 0     ROUTINE FOR TxRDY3 AND RxRDY3
;-------------------------------------------------------------------------------------------------

TX_RX?:     PUSH ACC
            PUSH DPL
            PUSH DPH
            PUSH PSW
            JB TXRDY3,TRANS
            JB RXRDY3,REC
            POP PSW
            POP DPH
            POP DPL
            POP ACC
            RETI                    ;FALSE INTERRUPT REQUEST

REC:        MOV DPH,#U3D
            MOVX A,@DPTR
            CJNE A,#0BCH,NOTSYNC    ;CHECK IF BYTE IS SYNC character
            JB TXRDY3,TRANS         ;IF SYNC CHARACTER THEN IGNORE
                                    ;AND..CHECK IF TXRDY IS SET
            POP PSW
            POP DPH
```

```
                POP DPL
                POP ACC
                RETI
NOTSYNC:        CJNE A,#TESTBYTE,RESTART ;ERROR IF REC'D BYTE IS NOT A sync
                                         ;.CHARACTER nor a TESTBYTE (EEH)
                SETB TESTRX              ;SET FLAG BECAUSE WE HAVE Rx'd
                                         ;A TEST
                JB TXRDY3,TRANS
RETURN:         POP PSW
                POP DPH
                POP DPL
                POP ACC
                RETI                     ;RETURN FROM INTERRUPT
RESTART:        CLR EA                   ;ERROR (default RESTART REQUEST)
                AJMP BEGIN               ;BEGIN SUBSYSTEM SOFTWARE
                                         ; RESTART

;-------------------------------------------------------------------------------------------------------

TRANS:          JNB FIDRDY,TESTREP?      ;IF FID ISN'T ASSEMBLD THEN NO
                                         ;FRAME...
                                         ;..BYTES TO TRANSMIT ∴ CHECK IF
                                         ;THERE IS A RECEIVED TEST TO
                                         ;RESPOND TO.
                MOV DPH,#U3D
                MOV A,NEXTFBYTE          ;MOVE NEXT BYTE IN FRAME TO
                                         ;ACC
                MOVX @DPTR,A             ;XMIT NEXT BYTE IN FRAME TO
                                         ;CENTRAL
                SETB PREPNXT             ;SET FLAG TO PREPARE NEXT BYTE
```

```
                                    ;FOR Tx
                JNB LASTFBYTE,CHK_REC?
RESETFLAGS:     CLR PREPNXT         ;IF LAST FRAME BYTE  then reset flag
                CLR FIDRDY          ;END OF FRAME so reset ALL FLAGS
                CLR LASTFBYTE

CHK_REC?:       JB RXRDY3,REC
                POP PSW
                POP DPH
                POP DPL
                POP ACC
                RETI                ;RETURN TO MAIN PROGRAM

TESTREP?:       JNB TESTRX,TXSYNC   ;IF NO TEST Rx'd THEN GOTO
                                    ;TXSYNC
                MOV DPH,#U3D
                MOV A,#TESTACK      ;LOAD A WITH TEST ACKNOWLEGE
                                    ;BYTE
                MOVX @DPTR,A        ;SEND TEST ACKNOWLEDGE TO
                                    ;LOCAL
                CLR TESTRX
                JB RXRDY3,REC
                POP PSW
                POP DPH
                POP DPL
                POP ACC
                RETI                ;RETURN TO MAIN PROGRAM


TXSYNC:         MOV DPH,#U3D        ;NO DATA TO Tx SO Tx  SYNC
```

```
                                        ;CHARACTER...
                MOV A,#0BCH             ;.. TO RESET INTERUPT ON TxRDY3.
                MOVX @DPTR,A
                JB RXRDY3,REC
                POP PSW
                POP DPH
                POP DPL
                POP ACC
                RETI                    ;RETURN TO MAIN PROGRAM

;-------------------------------------------------------------------------------------------------------------
;       INTERRUPT 1     ROUTINE FOR RxRDY1 AND RxRDY2
;-------------------------------------------------------------------------------------------------------------

CH1_CH2?:       PUSH ACC
                PUSH DPL
                PUSH DPH
                PUSH PSW
                JB RXRDY1,READCH1
                JB RXRDY2,READCH2
                POP PSW
                POP DPH
                POP DPL
                POP ACC
                RETI                    ;RETURN TO MAIN PROGRAM


READCH1:        MOV DPH,#U1D
                MOVX A,@DPTR            ;READ DATA BYTE FROM USART 1
STOR_CH1:       MOV R1,WNXT1
```

```
                MOV @R1,A                   ;WRITE CH 1 BYTE TO QUEUE 1
INC_Q1L:        INC CH1QL
                INC WNXT1                   ;UPDATE WRITE POINTER
                MOV A,WNXT1
                CJNE A,#BOUNDARY1,CHK_CH2   ;IF STILL IN QUEUE AREA
                                            ;GOTO INCREMENT CH1QL
                MOV WNXT1,#Q1ST             ;START OF QUEUE ADRESS INTO
                                            ;WRT PNTR

CHK_CH2:        JB RXRDY2,READCH2
                POP PSW
                POP DPH
                POP DPL
                POP ACC
                RETI                        ;RETURN TO MAIN PROGRAM

READCH2:        MOV DPH,#U2D
                MOVX A,@DPTR                ;READ DATA BYTE FROM USART 2
STOR_CH2:       MOV R1,WNXT2
                MOV @R1,A                   ;WRITE CH 2 BYTE TO QUEUE 2
INC_Q2L:        INC CH2QL
                INC WNXT2                   ;UPDATE WRITE POINTER
                MOV A,WNXT2
                CJNE A,#BOUNDARY2,CHK_CH1
                                            ;IF STILL IN CUE AREA GOTO
                                            ;INCREMENT CH2QL
                MOV WNXT2,#Q2ST             ;START OF QUEUE ADRESS INTO
                                            ;WRT PNTR

CHK_CH1:        JB RXRDY1,READCH1
```

```
POP PSW
POP DPH
POP DPL
POP ACC
RETI                    ;RETURN TO MAIN PROGRAM

END                     ;END OF PROG TEST6R (BBR_61)
```

```
;*************************************************************************
;                          TEST6C                                         *
;          LOCAL SUB-SUBSYSTEM SOFTWARE          (BBL_61)                *
;                                                                         *
;- READS BYTES RECEIVED FROM REMOTE SIDE AND INTERPRETS EACH FRAME        *
; BY THE HEADER BYTE (FID) WRITES TO PISO 1 + PISO 2                      *
;- USART and PISO'S ON PORT 0 BUS                                        *
;- START SYNC CHARACTERS OUT BY TX ARBITARY BYTE                         *
;-INT 0 RxRDY                                                            *
;- INT 1 NOT USED.  COULD BE USED FOR TxRDY IN A LATER VERSION OF SW      *
;    IF DESIRED (NOT NECESSARY TO USE INTERRUPT FOR THE TRANSMITTER)     *
;-TEST BUTTON CONNECTED TO P2.0                                          *
;*************************************************************************


PISO EQU 080H              ;P0  DATA BUS SHARED BY USART AND PISO


;SIGB 0                    ;P1.0    INTERFACE SIGNAL
;SIGB 1                    ;P1.1    INTERFACE SIGNAL
;SIGB 2                    ;P1.2    INTERFACE SIGNAL
;SIGB 3                    ;P1.3    INTERFACE SIGNAL
;SIGB 4                    ;P1.4    INTERFACE SIGNAL
LD1 EQU 095H               ;P1.5  LD FOR PISO 1    (ACTIVE LOW)
LD2 EQU 096H               ;P1.6  LD FOR PISO 2    (ACTIVE LOW)
CARRY EQU 097H             ;P1.7    CARRY FROM MOD EIGHT COUNTER


TESTBUTTON EQU 0A0H        ;P2.0    PUSH-BUTTON ACTIVATES LINK TEST
                           ;P2.6    USART CONTROL/NOT DATA
                           ;P2.7    USART CHIP SELECT
```

```
TESTLED EQU 0B0H    ;P3.0    TEST NOT YET ACKNOWLEDGED, FRONT PANEL led
SYNCLED EQU 0B1H    ;P3.1    LINK IN SYNC, FRONT PANEL LED
RXNRDY EQU 0B2H     ;P3.2    EXTERNAL INTERRUPT0 [RxRDY INVERTED (NOR)]
TXNRDY EQU 0B3H     ;P3.3  USART TXRDY PIN (INVERTED "NOR")
ST EQU 0B4H         ;P3.4    START FOR 8 PULSES (ACTIVE LOW)
URESET EQU 0B5H     ;P3.5    RESET FOR USART      (ACTIVE HIGH)
;WRITE              ;P3.6    USART WRITE CONTROL
;READ               ;P3.7    USART READ CONTROL


; CONSTANTS

UC EQU 07FH                 ;LINEAR ADDRESS for USART CONTROL (Port 2, DPH)
UD EQU 03FH                 ;LINEAR ADDRESS FOR USART DATA   (Port 2, DPH)
Q1ST EQU 030H               ;030H   QUEUE 1 START ADDRESS IN RAM
Q2ST EQU 040H               ;040H   QUEUE 2 START ADDRESS IN RAM
BOUNDARY1 EQU 040H          ;040H   FIRST ADDRESS IN RAM ABOVE QUEUE 1
BOUNDARY2 EQU 050H          ;050H   FIRST ADDRESS IN RAM ABOVE QUEUE 2
TESTBYTE EQU 0EEH           ;BYTE SENT AS TEST
TESTACK EQU 0ACH            ;TEST REPLY BYTE


; INTERNAL RAM

SIGB EQU  050H
WNXT1 EQU 051H      ;HOLDS VALUE OF WRITE NEXT POINTER FOR CH1 QUEUE
RNXT1 EQU 052H      ;HOLDS VALUE OF READ NEXT POINTER FOR CH1 QUEUE
WNXT2 EQU 053H      ;HOLDS VALUE OF WRITE NEXT POINTER FOR CH2 QUEUE
RNXT2 EQU 054H      ;HOLDS VALUE OF READ NEXT POINTER FOR CH2 QUEUE
;FLAGS
SENDTEST EQU 00H            ;00     FLAG SHOWS TEST BYTE WAITING FOR Tx
```

```
RFO EQU 01H             ;01     FLAG SHOWS RECEIVE FRAME IS OPEN
SIGBCHF EQU 02H         ;02     FLAG SHOWS THAT SIGB IS ATTACHED TO
                        ;EOF
SIGBCH EQU 03H          ;03     FLAG SHOWS MAIN THAT NEW VALUE IN
                        ;SIGB


;  REGISTERS

                        ;R0 USED FOR INDEXING THE POINTERS IN MAIN PROG
                        ;R1 USED FOR INDEXING THE PNTRS IN INTERRUPT PROG
COUNTER EQU R2          ;R2     USED FOR LOOPS AND DELAYS
CH1QL EQU R3            ;NUMBER OF BYTES IN CHANNEL 1 QUEUE
CH1FL EQU R4            ;NUMBER OF CHANNEL 1 BYTES IN FRAME
CH2QL EQU R5            ;NUMBER OF BYTES IN CHANNEL 2 QUEUE
CH2FL EQU R6            ;NUMBER OF CHANNEL 2 BYTES IN FRAME

;=================================================================


                ORG 0000
                AJMP START
                ORG 0003
                AJMP RX_BUTT?           ;INT 0 FOR USART RxRDY & TEST
                                        ;BUTTON

START:          CLR EX0                 ;DISABLE INTERRUPT 0
                CLR EA                  ;GLOBAL INTERRUPT DISABLE

SETUP:          SETB URESET             ;FOR WHEN SW RESET OCCURS
```

```
                MOV SP,#007H            ;RESET STACK POINTER VALUE
                SETB SENDTEST           ;SEND TEST WHEN SYNC IS
                                        ;ACHIEVED
                SETB TESTLED            ;PANEL INDICATION OF SYSTEM...
                                        ;... INITIALISATION
                CLR SYNCLED             ;PANEL INDICATION WHEN LINK...
                                        ;... SYNCHRONISES
                MOV P1,#0E0H            ;SETB LD1,  SETB LD2,  ENABLE
                                        ;CARRY I/P,  CLEAR ALL INTERFACE
                                        ;SIGNAL OUTPUTS
                SETB ST                 ;SET START SIGNAL
                MOV RNXT1,#Q1ST         ;LOAD QUEUE 1 READ POINTER
                MOV WNXT1,#Q1ST         ;LOAD QUEUE 1 WRITE POINTER
                MOV RNXT2,#Q2ST         ;LOAD QUEUE 1 READ POINTER
                MOV WNXT2,#Q2ST         ;LOAD QUEUE 1 WRITE POINTER

;-------------------------------------------------------------------------------------------------------
;       USART INITIALISATION
;-------------------------------------------------------------------------------------------------------

                CLR URESET              ;TAKE H/W RESET OFF USART ** **
                MOV COUNTER,#008H
RDELAY1:        DJNZ COUNTER,RDELAY1    ;DELAY AFTER REMOVING USART
                                        ;RESET

DUMMYRUN:       MOV COUNTER,#03H        ;LOAD COUNTER
                MOV A,#00H              ;INITIALISATION BYTE FOR USART
                MOV DPH,#UC
INIT:           MOVX @DPTR,A
                DJNZ COUNTER,INIT       ;WRITE 00H THREE TIMES FOR INIT
```

```
U_RST:      MOV A,#040H              ;INTRNL RESET USART ,BACK TO
                                     ;MODE
            MOVX @DPTR,A
            MOV COUNTER,#0FH
RDELAY2:    DJNZ COUNTER,RDELAY2     ;DELAY AFTER SW RESET
MODE:       MOV A,#00CH              ;MODE INSTRUCTION FORMAT
            MOVX @DPTR,A
SYNC:       MOV A,#0BCH              ;WRITE SYNC CHARACTER
            MOVX @DPTR,A
            MOVX @DPTR,A             ;WRITE SECOND SYNC CHRCTR TO
                                     ;USART
COMMAND:    MOV A,#0A7H              ;COMMAND USART ** ENTER HUNT
            MOVX @DPTR,A

;-------------------------------------------------------------------------------------------------------
;   ACHIEVE SYNC AT THE COMMS USART
;-------------------------------------------------------------------------------------------------------

            MOV DPH,#UD              ;USART DATA ADDRESS
            MOVX @DPTR,A             ;TRANSMIT BYTE STARTS SYNCS Tx

EH:         MOV A,#0A7H              ;PREPARE ENTER HUNT MODE again
            MOV DPH,#UC
            MOVX @DPTR,A             ;COMMAND ENTER HUNT TO USART
DLY:        MOV COUNTER,#0FFH        ;
EH_DLY:     DJNZ COUNTER,EH_DLY      ;DELAY FOR USART TO ACHIEVE
                                     ; SYNC
GETSYNC:    MOVX A,@DPTR             ;READ USART STATUS BYTE
            RLC A
```

```
                RLC A                   ;SYNDET IS STATUS BIT 6
                JNC EH                  ;WAIT IN LOOP UNTIL
                                        ;SYNCHRONISED

                SETB SYNCLED            ;GIVE INDICATION OF SYNC
                                        ;ON LINK
                SETB EX0                ;ENABLE INTERRUPT  (TEST +
                                        ;RxRDY)
                SETB EA

;-------------------------------------------------------------------------------------------------------
;       MAIN PROGRAM
;-------------------------------------------------------------------------------------------------------

MAIN:           JNB CARRY,TX_TEST?      ;IF PISO'S BUSY THEN GOTO
                                        ;TX_TEST?
                MOV A,CH1QL
                JZ TX_TEST?             ;IF CHANNEL1 Q EMPTY THEN SKIP
                                        ; PISO'S
                MOV A,CH2QL
                JZ TX_TEST?             ;IF CHANNEL2 Q EMPTY THEN SKIP
                                        ;PISO'S

RD_Q1:          MOV R0,RNXT1
                MOV PISO,@R0            ;READ NEXT BYTE FROM QUEUE 1
LOAD1:          CLR LD1                 ;LOAD PISO 1
                SETB LD1

RD_Q2:          MOV R0,RNXT2
```

```
                MOV PISO,@R0            ;READ NEXT BYTE FROM QUEUE 2
LOAD2:          CLR LD2                 ;LOAD PISO 2
                SETB LD2

STRT:           CLR ST                  ;START THE 8 PULSES (TRIGRED BY
                                        ;FIRST PULSE FROM 20 kHz CLK
                SETB ST                 ;TAKE START SIGNAL FROM LATCH
UPDATE1:        DEC CH1QL               ;DECREASE CH1 QUEUE LENGTH
                INC RNXT1               ;UPDATE RNXT1
                MOV A,RNXT1
                CJNE A,#BOUNDARY1,UPDATE2
                                        ;CHECK IF RNXT IN Q AREA. THEN
                                        ;GOTO.UPDATE2
                MOV RNXT1,#Q1ST         ;LOAD Q START ADDRESS INTO
                                        ;READ POINTER
UPDATE2:        DEC CH2QL               ;DECREASE CH2 QUEUE LENGTH
                INC RNXT2               ;UPDATE RNXT
                MOV A,RNXT2
                CJNE A,#BOUNDARY2,SETBST ;CHECK IF RNXT IN Q AREA
                                        ;THENGOTO.SETBST
                MOV RNXT2,#Q2ST         ;LOAD Q START ADRES INTO READ
                                        ;PNTER

SETBST:         JB CARRY,SETBST         ;WAIT FOR COUNTER TO CLEAR
                                        ;.(0 <WAIT< 40 us)


TX_TEST?:       JB TESTBUTTON,SIGBCH?   ;SAMPLE THE TEST BUTTON ON
                                ;PORT 2. IF NO TEST BUTTON GOTO
                                ; SIGBCH?
```

```
                SETB TESTLED            ;VISIBLE LED, TEST PROCESS
                                        ;INITIATED
TxRDY?:         JB TXNRDY,TxRDY?        ;WAIT, USART TRANSMITTER
                                        ;READY PIN
                MOV A,#TESTBYTE         ;LOAD TEST BYTE INTO A
                MOV DPH,#UD
                MOVX @DPTR,A            ;SEND TEST BYTE

SIGBCH?:        JNB SIGBCH,MAIN
                MOV P1,SIGB
                CLR SIGBCH
                AJMP MAIN
;-------------------------------------------------------------------------------------------------
;               INTERRUPT ROUTINE USED BY RxRDY
;-------------------------------------------------------------------------------------------------

LINKRX:         PUSH P0                 ;STORE P0 TO PREVENT LOSING
                                        ;BYTE TO.PISO IF INT 0 OCCURS
                                        ;WHILE LOADING PISO
                PUSH ACC                ;..
                PUSH PSW
                PUSH DPH
                PUSH DPL
                PUSH 007H


RX?:            JNB RXNRDY,RDBYTE       ;CHECK IF RxRDY, IF NOT THEN
                                        ;RETURN.FROM FALSE INTERRUPT
FLS_INT:        POP 007H
                POP DPL
```

```
                POP DPH
                POP PSW
                POP ACC
                POP P0
                RETI

RDBYTE:         MOV DPH,#UD             ;SETUP ADDRESS
                MOVX A,@DPTR            ;READ DATABYTE FROM USART
                MOV R7,A                ;SAVE BYTE TEMPORARILY IN
                                        ;REGISTER 7

RxFO?:          JNB RFO,SYNC?           ;IF A RECEIVE FRAME ISN'T OPEN
                                        ;THEN.CHECK IF BYTE IS A SYNC
                                        ;CHARACTER

CH1BYTE?:       MOV A,CH1FL
                JZ CH2BYTE?
                MOV R1,WNXT1
                MOV A,R7
                MOV @R1,A               ;WRITE RECEIVED BYTE TO QUEUE 1
                INC WNXT1               ;UPDATE WRITE POINTER
                INC CH1QL               ;INCREASE THE NUMBER OF BYTES IN CH1
                                        ;QUEUE
                DEC CH1FL               ;DECREASE THE NUMBER OF CH1 BYTES TO
                                        ;COME
                MOV A,WNXT1
                CJNE A,#BOUNDARY1,CH1DONE       ;IF STILL IN CUE AREA GOTO
                                                ;WRITE ACC
                MOV WNXT1,#Q1ST         ;START OF QUEUE 1 ADDRESS INTO WRT 1
                                        ;PNTR
```

```
CH1DONE:    POP 007H
            POP DPL
            POP DPH
            POP PSW
            POP ACC
            POP P0
            RETI

CH2BYTE?:   MOV A,CH2FL
            JZ SIGBYTE
            MOV R1,WNXT2
            MOV A,R7
            MOV @R1,A           ;WRITE RECEIVED BYTE TO QUEUE 2
            INC WNXT2           ;UPDATE WRITE POINTER
            INC CH2QL           ;INCREASE THE NUMBER OF BYTES IN CH2
                                ;QUEUE
            DEC CH2FL           ;DECREASE THE NUMBER OF CH2 BYTES TO
                                ;COME
            MOV A,WNXT2
            CJNE A,#BOUNDARY2,CH2DONE
                                ;IF STILL IN CUE AREA GOTO WRITE ACC
            MOV WNXT2,#Q2ST     ;START OF QUEUE 2 ADDRESS INTO WRT 2
                                ;PNTR
CH2DONE:    MOV A,CH2FL
            JNZ MORE            ;CHECK IF MORE CHANNEL 2 BYTES TO
                                ;COME
            JB SIGBCHF,MORE         ;CHECK IF END OF FRAME
            CLR RFO                 ;RESET FRAME FLAG (INDICATE NO
                                    ;FRAME OPEN)
MORE:       POP 007H
```

```
                POP DPL
                POP DPH
                POP PSW
                POP ACC
                POP P0
                RETI

SIGBYTE:        MOV SIGB,R7
                SETB SIGBCH
                CLR SIGBCHF
                CLR RFO                 ;RESET FRAME FLAG (INDICATE NO
                                        ;FRAME OPEN)
                POP 007H
                POP DPL
                POP DPH
                POP PSW
                POP ACC
                POP P0
                RETI

SYNC?:          CJNE A,#0BCH,NOTSYNC    ;IF BYTE READ ISN'T "BC" GOTO
                                        ;NOTSYNC
                POP 007H                ; FRAME NOT OPEN AND BYTE
                                        ;EQUAL TO
                POP DPL                 ;...SYNC CHARACTER...
                POP DPH                 ;.. SO IGNORE AND RETURN
                POP PSW
                POP ACC
                POP P0
                RETI
```

```
NOTSYNC:    RLC A                   ;IF BIT 7 OF RECEIVED BYTE IS SET,
                                    ;THEN IT IS NOT A FRAME
                                    ;DENTIFIER BYTE
FID?:       JC TESTACK?             ;IF BIT 7 IS SET, GOTO CHECK
                                    ;IF = TESTACK
                                    ;BIT7 NOT=1 THEN BYTE IS TAKEN
                                    ;AS FID

; DISMANTLE FID INTO USEFUL INFO REGARDING COMING FRAME
            RLC A
SIGBCHF?:   JNC RESTORE             ;FIND IF SIGNAL BYTE IS CHANGED,
                                    ;IOW
            SETB SIGBCHF            ;...WHETHER IT WILL BE AT THE
                                    ;END OF FRAME
RESTORE:    RRC A
            RRC A                   ;RESTORE BYTE BECAUSE THE
                                    ;BYTE IS FID
            ANL A,#007H             ;GET VALUE OF CH1FL
            MOV CH1FL,A             ;SAVE DERIVED CH1FL
            MOV A,R7                ;RESTORE FID TO ACCUMULATOR
            ANL A,#038H             ;GET VALUE OF CH2FL
            RR A
            RR A
            RR A
            MOV CH2FL,A             ;SAVE DERIVED CH2FL
            SETB RFO                ;SET FLAG SHOWS RECEIVE FRAME
                                    ;OPEN
            POP 007H
            POP DPL
```

```
                POP DPH
                POP PSW
                POP ACC
                POP P0
                RETI

TESTACK?:       JNB TESTLED,RESTART
;IF IT IS'NT SYNC NOR FID AND we HAVN'T.SENT a TEST THEN it is a RECEIVE error.

                RRC A                   ;RESTORE RECEIVED BYTE
                CJNE A,#0ACH,RESTART    ;IS THE RX BYTE A TEST ACK FROM
                                        ;REM
TSTACK:         CLR TESTLED             ;YES, SO RESET TEST LED &
                                        ;RETURN
RETURN:         POP 007H
                POP DPL
                POP DPH
                POP PSW
                POP ACC
                POP P0
                RETI

RESTART:        CLR EA
                AJMP START

                END                     ;END OF PROGRAM TEST6C
```

Glossary of Specific Terminology

ASCII

American Standards Committee for International Interchange (Halsall 1990 p 160)

Asynchronous

asynchronous data transmission refers to the transmission mode in which the transmitter and receiver clocks are independent

bps

Data transmission rate expressed in bits per second (kbps - kilo bits per second)

Break-out Box

A device that can provide access to the interface signals from a data interface cable. It has a male connector on one end and a female connector on the other, and is used by serially plugging it in line with the interface cable. The break-out box has a switch for each interface pin which gives the user the option to either block each signal (by open circuiting the switch) or to allow the signal to pass through normally (by closing the switch). Because access is given to the actual pin, signal pinouts can modified through cross connection, signals can be monitored, and other signals can be placed onto the interface pin to replace the signal at the other side of the switch that has been blocked.
Manufacturer: Network Communications Corporation
Model no.: BM50

CAD

Computer Aided Design

CCITT

The International Telegraph and Telephone Consultative Committee

Central

Refers to the location where the protocol analyzer is located and where the project's "**central** sub-system" will be. This is where the X.25 trained and specialised maintenance staff are located.

Chameleon

A protocol analyser manufactured by Tekelec. This analyser is used by Telkom maintenance staff in the fault diagnosis of X.25 services. When monitoring a V24 or V.35 interface, it can analyze the X.25 protocol. It also has data storage facilities for data history displays. The traffic (data packets) on the interface (in both transmit and receive directions) is displayed on a cathode ray tube and clearly indicates the boundaries of packets and decodes their specific identification. **Chameleon** is a registered trademark of Tekelec INC.

CMOS

Complementary Metal-Oxide semiconductor (one of the important integrated circuit technologies (Floyd 1982, p.84))

Datel

Analogue Data services provided by Telkom S.A.

Diginet

Digital Data Network provided by Telkom S.A.

DTE

Data Terminal Equipment

EIA

Electronic Industries Association

FID

Frame Identifier Byte used in the data link protocol developed in this project. This byte transmitted at the beginning of each data frame identifies the frame to follow.

fixer

solution in which the photographic negative is bathed, in order to prevent it from any further exposure
(see A.1 in Appendix)

Fluke 77

Digital Multimeter, John Fluke MFG. CO. INC., Everett, Washington

Full-duplex

Communication simultaneously possible in Transmit and Receive directions.

HDLC

High Level Data Link Control

IC

Integrated Circuit

ICE

In Circuit Emulator

ISO

International Standards Organisation

Interview 40A

Data Analyser manufactured by Tekelec

kbps

see bps

Logic Analyser

This is basically a multichannel oscilloscope type of instrument (used in trouble shooting) with the ability to detect and display logic levels in several forms, i.e., timing diagram format, bit format, or hexadecimal code (Floyd 1982, p. 15). The logic analyser used in this study was the PM 3632 manufactured by Phillips

Logic Probe

Test instrument with which one can determine the voltage levels or pulses of a particular point in a TTL or CMOS logic circuit (Floyd 1982, p. 17).

LSB

Least Significant Bit

MDS

Microprocessor Development System. An Intel dedicated computer system with emulation facilities used in the development of microprocessor and microcontroller circuits

Nodes

Switching point or Packet Switch Exchange on the Public Switched Data Network (also known as X.25 exchange)

Noise

unwanted electrical signals that could possibly interfere with the reception and interpretation of valid signals

NTU

Network Terminating Unit. Part of the Digital Data Network of Telkom S.A. providing the interfacing between the DTE and the Digital Network. The NTU is to the digital network what a Modem is to an analogue data network.

PAD

Packet Assembler and Disassembler circuit. Used to interface terminals not using X.25 as a protocol to the X.25 network by converting data into X.25 protocol and vice versa (CCITT 1988, p .20)

Patch Panel

Test and monitoring position of data interfaces. This is positioned at the point where the data interface enters the X.25 node. This is the last physical point at which monitoring of the data interface can be done. The patch panel is a bay or rack, through which the interfaces pass and where testing can be done using one of the three positions for each interface. These test positions are namely;

a) break towards the modem (DCE),

b) break towards the terminal (DTE in this case the X.25 Node),

c) the monitor position.

(Telenex 1991, p. 9)

The remote sub-system of this project will be provided with the monitored interface from this monitor position of the patch panel. When plugged in here, the flow of data on the monitored interface will be uninterrupted.

P-CAD

Personal-CAD Systems, Inc. Computer software company who supply the CAD package P-CAD used in PCB design. The following are trademarks of the company; P-CAD, PC-CAPS, PC-CARDS, PC-PRINT, PC-COMP, PC-DRC, PC-DRILL, PC-ECO, PC-NODES, PC-PACK (P-CAD 1989)

PCB

Printed Circuit Board used to mount and connect electronic components of a circuit in a permanent arrangement.

Protocol Analyzer

a specialised piece of test equipment used to analyze the data on a communications link from the interface

PSTN

Public Switched Telephone Network

Remote

refers to the location where the remote X.25 node is situated. This is the location of the interfaces that need to be monitored and of the project's "**remote** sub-system".

Riston

Flexible adhesive layer used in dry film processing manufacture of PCBs

ROM

Read Only Memory, also known as Program Memory, where the program of a micro-processor will be stored.

SDLC

Synchronous Data Link Control (Intel 1984)

SIGB

Name given to the Byte stored in the RAM of the Remote sub-system. It is used to save the current status of the monitored interface signals.

Synchronous

synchronous data transmission refers to the transmission mode in which the transmitter and receiver clocks are synchronised (Halsall 1990, p.82)

TTL

Transistor-Transistor logic (one of the important integrated circuit technologies) (Floyd 1982, p. 84)

UART

Universal Asynchronous Receiver and Transmitter (Halsall p. 77)

USART

Universal Synchronous/Asynchronous Receiver/Transmitter (Intel 1993 p.2-1)

V.24

CCITT Recommendation
List of definitions for interchange circuits between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE)

V.28

CCITT Recommendation
Electrical characteristics for unbalanced double-current interchange circuits

X.3

CCITT Recommendation

Packet Assembly Disassembly Facility (PAD) in a Public Data Network

X.28

CCITT Recommendation

DTE/DCE interface for a start-stop mode data terminal equipment accessing the packet assembly/disassembly facility (PAD) in a public data network situated in the same country

X.25

CCITT Recommendation

Interface between data terminal equipment and data circuit-terminating equipment for terminals operating in the packet mode and connected to the Public Data Network by dedicated circuit.

X.29

CCITT Recommendation

Procedures for the exchange of control information and user data between a packet assembly/disassembly (PAD) facility and a packet mode DTE or another PAD

References

H. Bonde (1993), Facsimile message (July), Tekelec INC., West Agoura Road, Calabasas, California, U.S.A.

R.L. Boylestad and L. Nashelsky (1982) "**Electronic Devices and Circuit Theory**" third edition, Prentice-Hall INC, Engelwood Cliffs, NJ, U.S.A.

M. Carle (1993), Durban Saponet, Telkom S.A. (personal communication).

CCITT (1985 a), "**Electrical Characteristics for Balanced Double-Current Interchange Circuits for General use with Integrated Circuit Equipment in the Field of Data Communications** (Recommendation V.11 pp. 31-40)", in ITU, International Telecommunication Union, CCITT, The International Telegraph and Telephone Consultative Committee, Red Book, Vol. VIII, Fascicle VIII.1, **Data communication over the telephone Network,** Recommendations of the V series, VIII th Plenary Assembly, Malaga-Torremolinos, 8-19 October 1984, Geneva, ISBN 92-61-02301-0.

CCITT (1985 b), "**List of Definitions for Interchange Circuits Between Data Terminal Equipment and Data Circuit-Terminating Equipment** (Recommendation V.24 pp. 100-116)", in ITU, International Telecommunication Union, CCITT, The International Telegraph and Telephone Consultative Committee, Red Book, Vol. VIII, Fascicle VIII.1, **Data communication over the telephone Network,** Recommendations of the V series, VIII th Plenary Assembly, Malaga-Torremolinos, 8-19 October 1984, Geneva, ISBN 92-61-02301-0.

CCITT (1985 c), "**Electrical Characteristics for Unbalanced Double-Current Interchange Circuits** (Recommendation V.28 pp. 199-202)", in ITU, International Telecommunication Union, CCITT, The International Telegraph and Telephone Consultative Committee, Red Book, Vol. VIII, Fascicle VIII.1, **Data communication over the telephone Network,** Recommendations of the V series, VIII th Plenary Assembly, Malaga-Torremolinos, 8-19 October 1984, Geneva, ISBN 92-61-02301-0.

CCITT (1985 d), "**Interface Between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE) for synchronous operation on Public Data Networks** (Recommendation X.21 pp. 35-55)", in ITU, International Telecommunication Union, CCITT, The International Telegraph and Telephone Consultative Committee, Red Book, Vol. VIII, Fascicle VIII.3, **Data Communication Networks Interfaces,** Recommendations X.20-X.32, VIII th Plenary Assembly, Malaga-Torremolinos, 8-19 October 1984, Geneva, ISBN 92-61-02321-5.

CCITT (1985 e), "**Interface Between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE) for Terminals Operating in the Packet Mode and Connected to Public Data Networks by Dedicated Circuit** (Recommendation X.25 p. 108-242)", in ITU, International Telecommunication Union, CCITT, The International Telegraph and Telephone Consultative Committee, Red Book, Vol. VIII, Fascicle VIII.3, **Data Communication Networks Interfaces,** Recommendations X.20-X.32, VIII th Plenary Assembly, Malaga-Torremolinos, 8-19 October 1984, Geneva, ISBN 92-61-02321-5.

CCITT (1989), ITU, International Telecommunication Union, CCITT, The International Telegraph and Telephone Consultative Committee, Blue Book, Vol. VIII, Fascicle VIII.2, **Data Communication Networks: Services and Facilities, Interfaces,** Recommendations X.1-X.32, IXth Plenary Assembly, Melbourne, 14-25 November 1988, Geneva, ISBN 92-61-03671-6.

D.A.T.A. INC. (1986), "**D.A.T.A. Book, electronic information series**" Vol. 31, Cordura Publications INC., San Diego CA, U.S.A.

Dept. of Posts and Telecommunications (1989), "**Data Communication Maintenance**", Internal publication.

T.L. Floyd (1982), "**Digital Fundamentals**" 2 nd Edition, Charles E. Merrill Publishing Company, Columbus, Ohio, U.S.A.

F. Halsall (1990) "**Data Communications, Computer Networks and OSI**" second edition, Addison-Wesley Publishing Company, Avon. Great Britain.

B. Holsworth, G.R. Martin (1991), "**Digital Systems Reference Book**", Butterworth-Heinemann, LTD.

Intel (1993), "**Connectivity**", Intel Corporation, U.S.A., ISBN: 1-55512-174-8.

Intel (1992 a), "**Embedded Applications 1993/1994**", Intel Corporation, U.S.A., ISBN: 1-5512-179-9.

Intel (1992 b), "**Embedded Micro controllers and Processors Vol. 1**", Intel Corporation, Santa Clara, CA, U.S.A.

Intel (1981), "**Getting Started with the ICE -51 In-Circuit Emulator**", Intel Corporation, Santa Clara, CA, U.S.A., Manual order number 121595-001 Rev A.

Intel (1984), "**Microsystem Components Handbook**", Vol. 1, Intel Corporation, Santa Clara, CA, U.S.A.

Kopp Electronics LTD (1993), "**Kopp Electronics LTD Components Catalogue**", internal publication

P. Lochner - Telkom S.A. (1994), Engineer Diginet, Pretoria, (telephonic communication Tel. 012-3111632).

Marconi Communication Systems (1986), "**Technical manual CT3431C, Digital Data Network U3000**", STC, Boksburg.

Maxim Integrated Products Inc. (1990), "**Maxim 1990 New Releases Data Book**", Internal publication, Sunnyvale CA. U.S.A.

National Semiconductor Corporation (1988), "**Linear Data Book 1**", Internal publication, USA

Personal CAD Systems Inc (December 1989), " **P-CAD Illustrated User's Guide** Printed Circuit Board Design", Internal publication, San Jose CA. U.S.A.

K. Randlehoff (1993), Altech Instruments (agent for Tekelec), Johannesburg, Private communication (July).

SITES (1994), Telkom's software package used for calculating the charges and tariffs of various Telematic services.

Standard Telephones & Cables (1986), "**Network Terminating Unit - X.21 (V11) User Operating Instructions**" Issue 1, Internal publication, (April 1986), Boksburg.

Tekelec (1988), "**DDN X.25 Qualification Procedures for the Chameleon 32 User's Manual**", Tekelec, Calabasas, CA, U.S.A.

Tekelec (1990), "**Chameleon 32 User's Guide**", Vol. 1, Tekelec, Calabasas, CA, U.S.A.

Telenex Corporation (1991), "**Manual Fallback Switching and DATA-PATCH Systems** Technical Manual", Issue 4, Internal publication, Springfield, Virginia.

Telkom S.A. (1992), "**Principles of Packet Switching**" issue 1, Internal publication.

Texas Instruments (1985), "**The TTL Data Book for Design Engineers**" Vol.1, Internal publication Germany.