A PACKET RADIO SYSTEM FOR AN INDUSTRIAL DATA NETWORK

Gajadhar Sewnath

submitted in partial fulfilment of the requirements for the degree of Master of Science in Engineering in the Department of Electronic Engineering, University of Natal

1992

ABSTRACT

This project was undertaken for a commercial electronics company, CONTROL LOGIC (CONLOG) which is involved in the research, design, development and manufacture of data acquisition, control, energy management and automotive equipment. Currently CONLOG uses an inhouse token passing local area network CONET for industrial data communications.

The need had arisen to provide a means of data communication amongst widely geographically distributed remote terminal units (RTUs) generating demands at a very low duty cycle. A need for communications between RTUs and a centralised controller was also required. In addition to this, multihop communications between the RTUs was required. Packet switching using a broadcast radio network provides an efficient means of achieving this.

An investigation into to the various media access control protocols and contention techniques using packet radio was carried out. The various media access techniques were compared with respect to throughput and normalised delay. This led to the selection of a media access scheme for the packet radio network using RTUs.

A protocol specification for the packet radio network, in which control is centralised or distributed, was done. The architechure of the switching protocol specified adheres to the Open Systems Interconnect (OSI) model of the International Standards Organisation.

An experimental packet switching radio network was implemented using the protocol specification defined above. The packet radio network (PACNET) uses existing off the shelf radios and purpose built hardware for the remote terminal units.

The thesis describes methods of data communications suitable for widely dispersed industrial data communications, the selection of the packet switching media access methods and control protocols, and the design and implementation of the prototype system.

To my wife Thiloshni

PREFACE

This thesis, unless specifically stated to the contrary in the text, is my own work and has not been submitted in part, or in whole to any other University.

This project was done in conjunction with the "Pole Mount Remote Terminal Unit" (PMRTU) project at the electronics company, CONTROL LOGIC (PTY) LTD (CONLOG). The PMRTU project is based on a packet radio networking system for the transfer of data between remote terminal units (RTU)s in the system. The design and implementation of a packet radio network for such a system initiated this thesis. The research carried out was aimed at developing a packet radio network for geographically widespread RTUs.

The research was carried out at the University of Natal on a part time basis whilst the design and implementation of the packet radio network was carried out at CONLOG. Implementation of the packet radio network was done at CONLOG using hardware developed inhouse.

The supervisor for this project was Mr D.C. Levy (University of Natal).

ACKNOWLEDGEMENTS

I am indebted to the following persons for their assistance during the project and during the compilation of the thesis:

Ian McNielage for assisting in the specification of the packet radio network protocol and his assistance at the termination of the project,

D.C Levy for assisting in the collection of research material on packet radio, data communications literature and his support,

Joe Steyn for his assistance in the compilation of illustrations,

D. Celine for allowing me to use the project at CONLOG for my MSc thesis,

CONLOG for its sponsorship of this research,

and the following persons:

Vanesh Mudhanlall, Bruce Were, Angelo Borros, Owen Taverner Smith and Dave Bartlett, of CONLOG.

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

Tables:

Figures:

CHAPTER 1

1 INTRODUCTION

CONLOG recently undertook a project for the design and manufacture of a supervisory control system for pole-mounted devices on various reticulation power networks. The system consists basically of three subsystems that are discussed below. A block diagram representing an overview of the system is shown in Figure 1.

Remote Terminal Units (RTUs) are located on the same pole as the pole mounted devices and provide the immediate control and monitoring of the pole mounted devices. The pole mounted devices may be auto-reclosers, air breakers etc. The RTU consists of digital inputs and outputs, analog inputs, central processing unit with memory and software to provide the functions for monitoring and control of the pole-mounted devices, software for communications, Minimum Shift Keying (MSK) modem, radio unit and relevant mechanics to interface to the pole mounted devices. The RTUs communicate with each other and with a high site radio using MSK modulation in the UHF band. In addition to the above, the RTUs are required to behave as store and forward repeaters (digipeaters) for extending communication around obstacles such as mountains etc.

The Man Machine Interface (MMI) provides the functions required to enable an operator to interface to the system. It consists of a personal computer running a Supervisory Control And Data Acquisition (SCADA) software that has features such as graphic picture generation and editing, system and device status displays, system configuration, alarm and report generation, execution of controls etc.

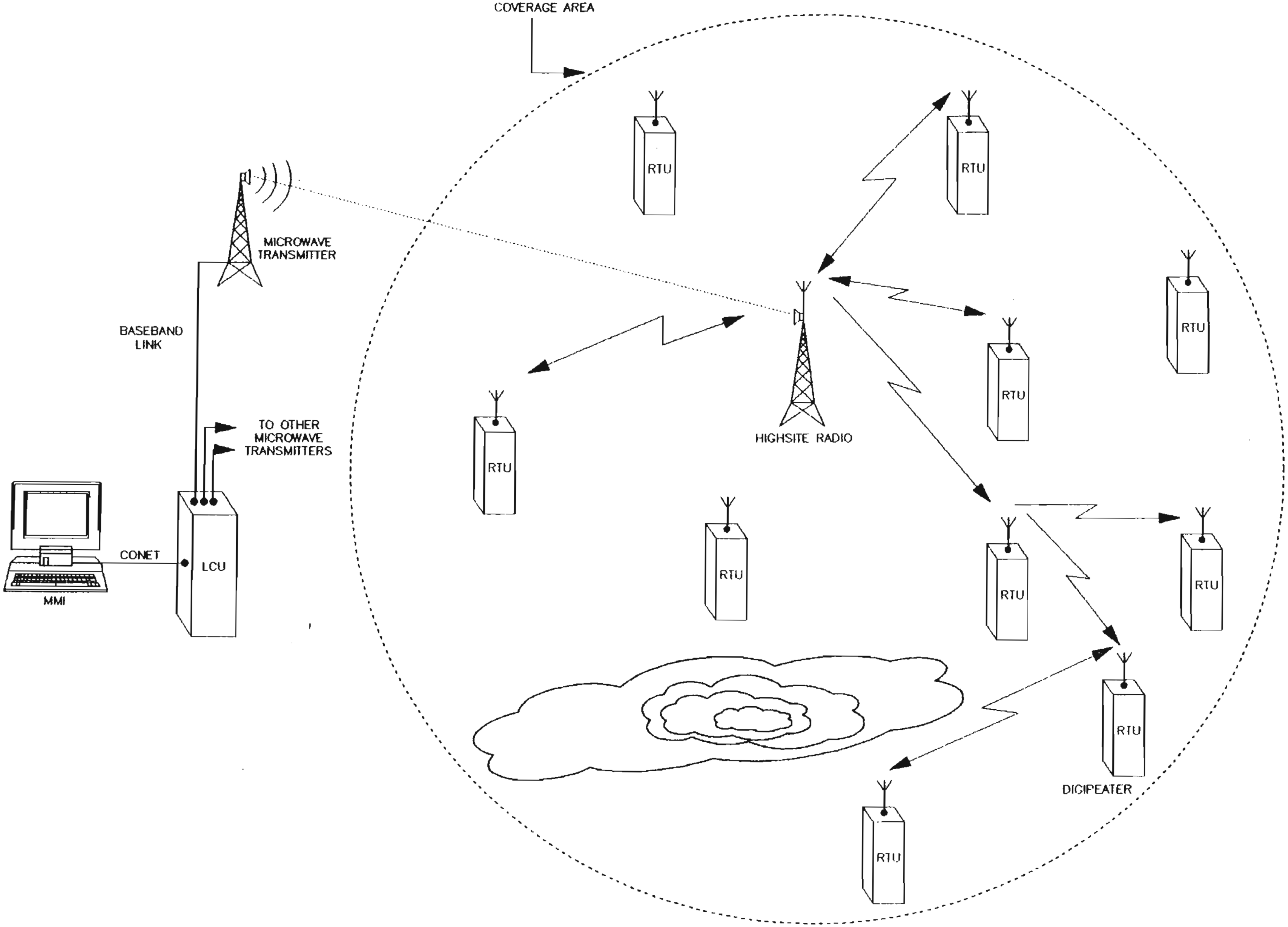


FIGURE : 1 PACKET RADIO SYSTEM OVERVIEW

The Line Control Unit (LCU) functions as the gateway between the MMI and the RTUs. It consists of hardware and software to handle communications between the MMI and the RTUs. The LCU can handle up to twelve coverage areas (only one is indicated in Figure 1).

The LCU and the MMI will henceforth be referred to as the central controller.

Data communications in the system is handled using two local area networks. The MMI and LCU communicate by means of the token passing local area network CONET. Currently the only nodes on this network is the MMI and the LCU, but the system may be expanded to incorporate multiple MMIs. The communications between the LCU and the RTUs in a certain coverage area is effected using the packet radio network PACNET. Due to the large distances involved between the central controller and the various coverage areas, a microwave transmitter is used to relay messages from the LCU to the high site radio in the coverage area. A baseband link exists between the LCU and its local microwave transmitter.

Messages emanating from the RTUs consists of self-initiated messages such as a change of state detected on an input, and responses to polls from the central controller. Messages leaving the central controller consist of polls for data from a specific RTU, execution of a control on an RTU and responses to self-initiated messages from RTUs.

The packet radio network PACNET was developed to provide communications between the LCU and the RTUs. Messages leaving an RTU will contend for the radio channel using the rules as laid down for

PACNET. On gaining access to the shared radio channel, the RTU will be able to make its transmission to the another RTU or the LCU. The microwave transmitters are transparent to the RTUs. At the end of a transmission, the RTU will wait for an acknowledgement. If an acknowledgement has not been received within a certain period, the RTU assumes that a collision has occurred and will delay for the back-off period before retransmitting the message. A similar procedure will be followed by messages leaving the LCU for the RTUs.

This thesis is involved with only one part of the system, i.e. the research, design and development of the packet radio network, named PACNET, that was used to provide communications between the LCU and RTUs. The thesis provides an overview of various standards used in the specification of the packet radio network, the more common techniques used in sharing a shared communications channel and finally the design of the packet radio network.

CHAPTER 2

2 DATA COMMUNICATIONS USING RADIO

2.1 Introduction

The provision of data communications connectivity between stations that are geographically dispersed, generating data traffic characterized by a low duty cycle, in the absence of existing communication links, such as telephone cabling, becomes a task when using conventional wire techniques. The cost of cabling and installation of such wire links amongst a large number of users, in the order of hundreds, makes such a method infeasible for the provision of data communications.

The use of packet switching using broadcast radio with a shared channel provides an efficient way of achieving connectivity amongst remotely distributed users. The users in the context of this project are RTUs that generate bursty traffic, with a long quiescent interval before transmitting again. Another user is the LCU that requires a prompt means of communication between the RTUs and itself on demand. In both cases, coverage area is in the order of fifty kilometers.

Numerous techniques have been devised for accessing a multiple access communications channel, such as polling, reservation techniques, random access, fixed assignment etc [Tobagi, 1980]. This chapter contains a study of the three stronger solutions for channel access in the shared medium environment, their advantages and disadvantages discussed with respect to the packet radio network that was developed for the RTUs.

Packet radio architectures may be classified as centralized or distributed [Stallings, 1985].

In packet radio systems using centralized architectures, a central station does the mediation between the various stations around it. The central station has the ability to communicate with all stations on the network, but the outlying stations may communicate with each other only through the central station. Such a system, although being inefficient in the passage of data between outlying stations, has its merits in providing rapid access to outlying stations when the need arises.

Distributed packet radio systems allow stations in a network to communicate with each other directly or indirectly through a repeater. Channel access is achieved using a protocol adhered to by all stations in the network. Such a system allows a greater degree of freedom between the various stations in the network and is especially useful in mobile applications [Kahn, 1977]. Distributed architectures allows deployment of stations within the network with ease and minimum reconfiguration of the network.

In the packet radio system to be designed, an attempt will be made to merge the characteristics of centralized and distributed architectures for a system that will take advantage of both architectures.

In the specification of the packet radio network protocol, two basic standards were used. The first is the Open Systems Interconnect (OSI) architecture of the International Standards Organization (ISO) on which the protocol was based. The second standard used is the

International Electrotechnical Commission (IEC) standard for the data link layer in the OSI reference model. Both standards are discussed briefly in section 2.2, below.

2.2 Standards

2.2.1 ISO Reference Model

The OSI reference model was established by the International Standards Organization (ISO) to provide a framework for defining standards for linking heterogeneous computer systems.

In the OSI model, the task of communication is partitioned into a set of smaller, more manageable tasks. The ISO used the layer structuring technique in the OSI model to provide for the subdivision of the communications tasks.

The OSI reference model is organized as a series of seven layers, with each layer performing a well defined function. Figure 2 indicates pictorially the OSI network architecture.

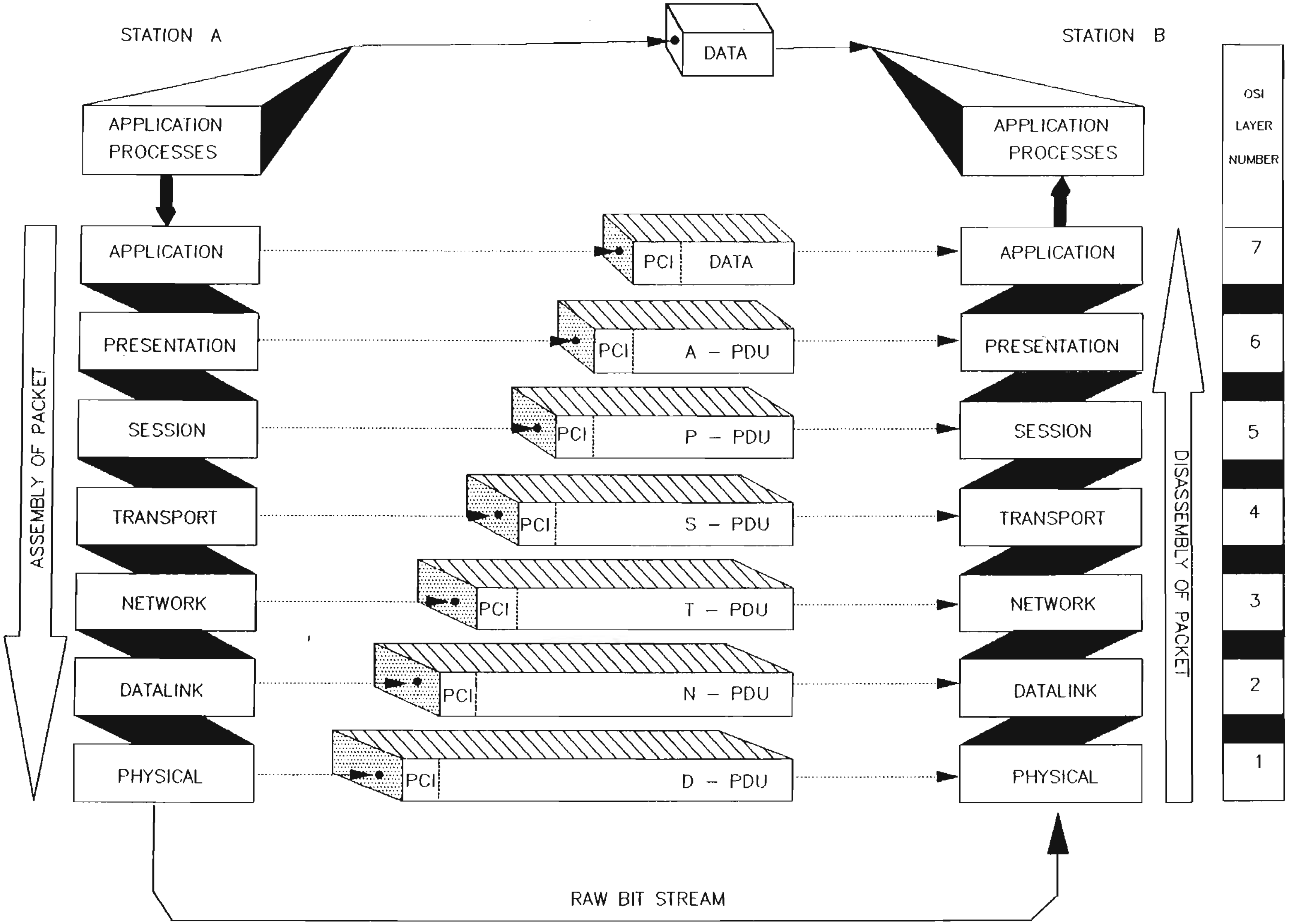


FIGURE : 2 LAYERING IN THE OSI REFERRENCE MODEL

Each layer has a well defined interface between itself and the layer above and below it. The purpose of each layer is to provide services to the layer above, and at the same time keep the implementation of the services transparent.

Communication between two host computers occur using the protocol for the respective peer layers. The dotted lines in Figure 2 indicate virtual communications between peer processes whilst the solid line indicates the physical communication between the layers. Peer layers communicate using the services offered by the lower layers.

Stallings, [1990] gives a good description of the functions of the various layers in the OSI model and are given below. An "upper" or "higher" layer is considered to be one that is higher than the described layer in the OSI model hierarchy. A "lower" layer is the inverse of the latter.

(a) PHYSICAL :

This layer deals with the transmission of the raw bit stream on the communication medium. It covers the electrical, mechanical, functional and procedural interface to the physical medium.

(b) DATA LINK :

This layer attempts to make the raw bit stream of the physical layer more reliable by providing error detection and control.

(c) NETWORK :

This layer is used to shield the upper layers of any knowledge of the underlying network technologies used to connect the systems. It provides for the transparent transfer of data across the network. It handles the end to end routing and ordering of packets from the source to the destination.

(d) TRANSPORT:

The transport layer basically handles end to end transfer of data. The data may be transferred from source to destination directly or through several networks using multiple paths. This layer ensures that upper layer data is delivered error free, in sequence and without duplications or losses. It is has the ability to split large packets sent from the upper layers into smaller units for transmission and will reassemble smaller units received from the network layer. It also manages connections established by the session layer. The transport layer may also have the ability to offer a quality of service to the upper layers.

(e) SESSION:

This layer provides services used in establishing, maintaining and releasing a dialogue between two or more hosts or presentation layer entities ie. allow sessions between two users.

(f) PRESENTATION:

This layer is involved with the representation (syntax) of the data during a transfer of packets between two application layer protocol entities. Another function of the presentation layer is data encryption.

(g) APPLICATION:

The application layer consists of a variety of protocols that are commonly used by various users. These may include file transfer protocols, terminal emulations, electronic mail etc.

Each layer contains one or more active elements known as **entities.** An entity is used to implement a certain function provided by that layer, eg. subroutine to determine the network path in the network layer. A layer N+1 user may request services in layer N through **Service Access Points (SAPs)** at the interface between the two layers. Figure 3 indicates pictorially the interaction between the layers using the Network and Data Link layer as an example.

Peer layers in the system communicate using Protocol Data Units (PDUs). PDUs contain control information and data. Though a PDU cannot be transferred directly between peer layers, PDUs are passed to the layer immediately below it, becoming the Service Data Units (SDUs) for the lower layer.

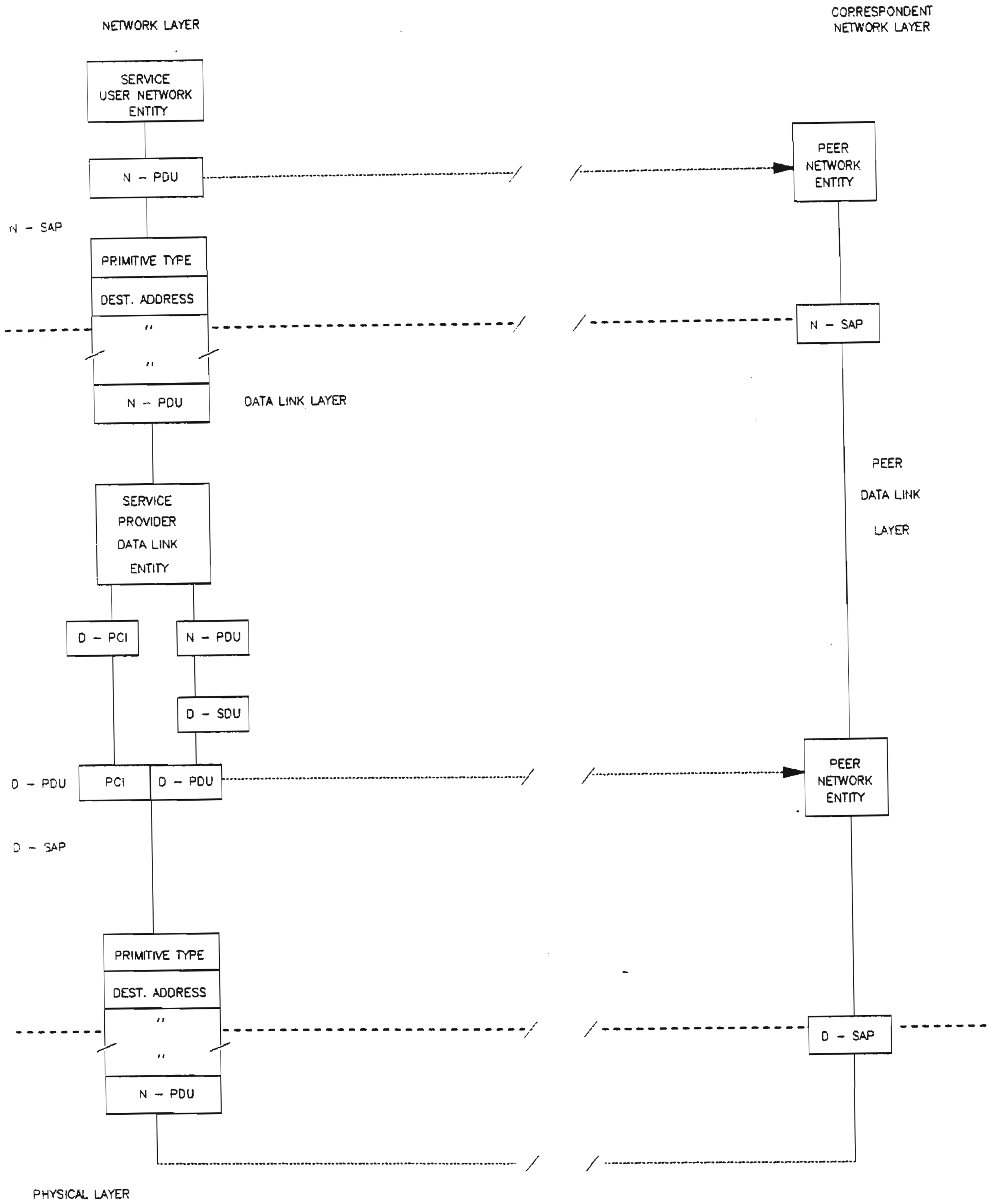


FIGURE : 3. INTERACTION BETWEEN LOCAL LAYERS AND BETWEEN PEER LAYERS.

The SAP may contain the type of primitive (as explained below), the destination address, the PDU to be transmitted and other control information. In the example in Figure 3, the Network Layer will communicate with a peer Network Layer using a PDU formed by the Network Layer. The broken lines indicate virtual communications whilst the solid lines indicate the physical communication path.

The Data Link Layer entity will add its own Data Link Layer Protocol Control Information (D-PCI) to the D-SDU and create a PDU for the Data Link Layer (D-PDU) which is transmitted to the peer Data Link Layer using the services of the Physical Layer.

On the correspondent side, the peer layers will disassemble the packet as it travels from the lowest layer to the peer layer, removing any control information used by a layer itself.

The services in a layer is invoked using primitives. The four types of primitives used in the OSI model are illustrated in Figure 4. The **REQUEST (REQ)** primitive is used to invoke a service and pass certain parameters required by the service.

The **INDICATION (IND)** primitive is used to indicate that a procedure has been invoked by a peer layer and will pass all required information with it. The **RESPONSE (RES)** primitive is used to acknowledge or complete a procedure that was invoked by an INDICATION. The **CONFIRM (CON)** primitive is used by the service provider layer to complete or acknowledge a procedure invoked by a corresponding REQUEST.

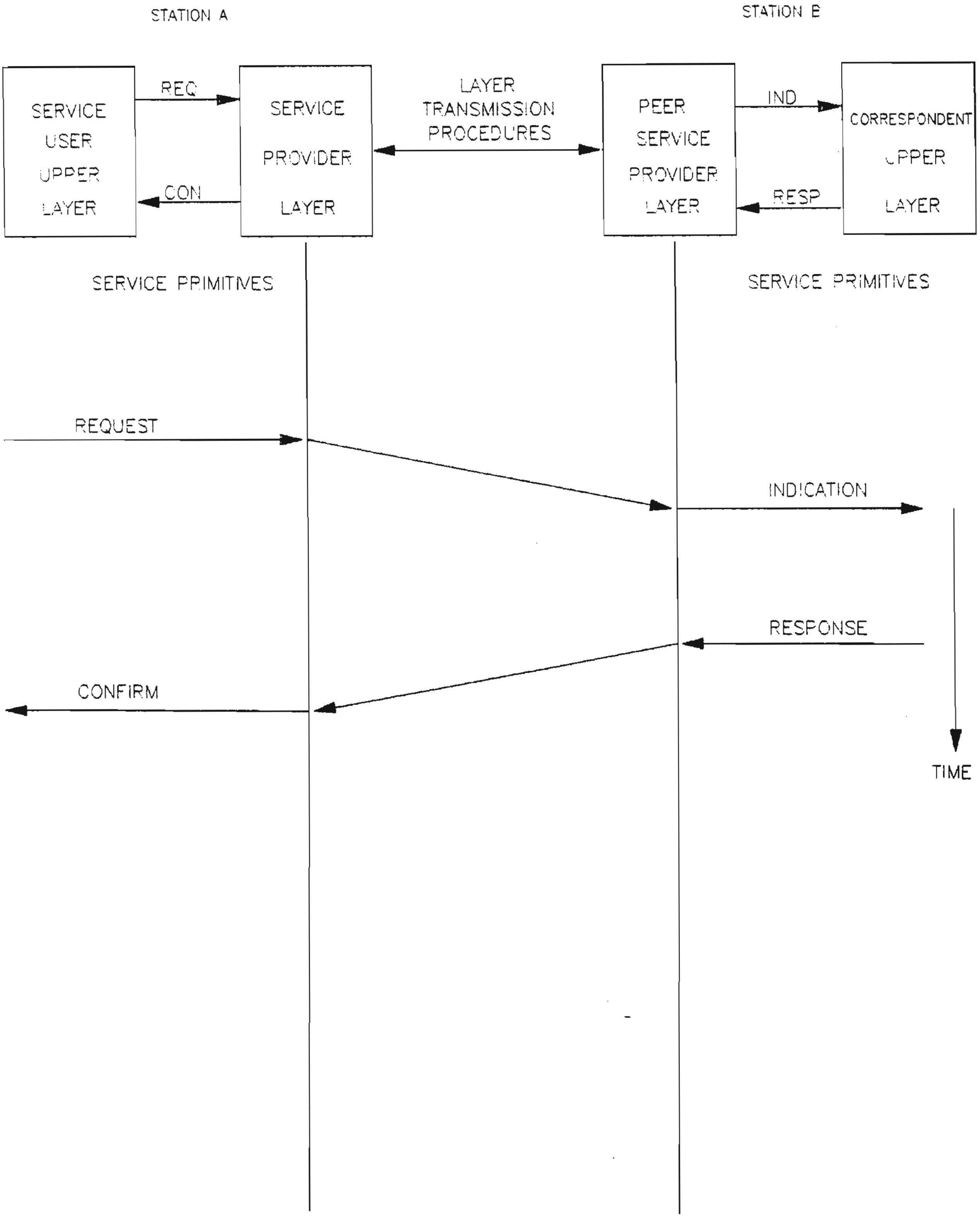


FIGURE : 4 INTRACTION OF SERVICE PRIMITIVES WITH SERVICES IN THE OSI MODEL.

In order to describe each layer in context with the other layers, it is only necessary to describe:

(a) the services that the layers offers to the one above it,

(b) the internal operation of the layer and ,

(c) the services that the layer uses from the layer below it in order to carry out its functions.

The protocol specification, described in Chapter 3, uses the nomenclature and methodology of protocol specification as dictated by the OSI reference model.

2.2.2 IEC Data Link Transmission

2.2.2.1 General

The data link layer in the packet radio network architecture is based on the IEC standard for transmission frame formats [IEC,1990] and link transmission procedures [IEC,1988]. These standards form sections one and two of **IEC Transmission Protocols for Telecontrol Equipment and Systems**. At the time of designing the packet radio network, section two of the standard was not published; hence revision eight of this document was used instead.

The IEC standard for transmission frame formats covers the specification of standard block codes and transmission rules for the transmission of bit serial information frames. This standard was used to determine a suitable frame format for the packet radio system that was designed.

The IEC standard for link transmission procedures specifies standard rules for transmission of link user data amongst geographically widespread users. This standard was used in the design and implementation of the data link layer in the OSI model.

2.2.2.2 **Transmission Frame Formats**

This standard is involved with specifying standards on coding, formatting and synchronizing data frames of variable and fixed lengths that meet specified data integrity requirements. Of concern to the packet radio network design is the standard block code formats used for transmission of bit serial frames over a memoryless channel.

Data integrity is evaluated by assessing the non-integrity of data in a system. The two causes of non-integrity of data are [IEC, 1990]:

(a) residual error rate =

$$\frac{\text{number of undetected incorrect messages}}{\text{total number of messages sent}} \tag{2-1}$$

(b) rate of residual information loss =

$$\frac{\text{number of undetected lost messages}}{\text{total number of messages sent}} \qquad (2\text{-}2)$$

The IEC [1990] standard has established three classes of data integrity: I1, I2 and I3. The classes set an upper limit on the residual error rate with respect to various bit error rates. Figure 5 shows graphically the upper limits for the three data integrity classes.

The data integrity requirements of the packet radio system designed fall into the I2 class. The frame format was chosen to suit this class is the FT1.2 frame format [IEC,1990]. Figure 6 is a pictorial representation of the FT1.2 frame format with variable data length.

The length byte specifies the number of bytes between the second start byte and the checksum byte, excluding the start and the checksum bytes.

The transmission rules using the FT1.2 frame format is summarized below:

(1) Line idle is binary 1.

(2) Line idles are not permitted between characters of a frame.

(3) A minimum line idle interval of 33 bits is required between frames when an error is detected.

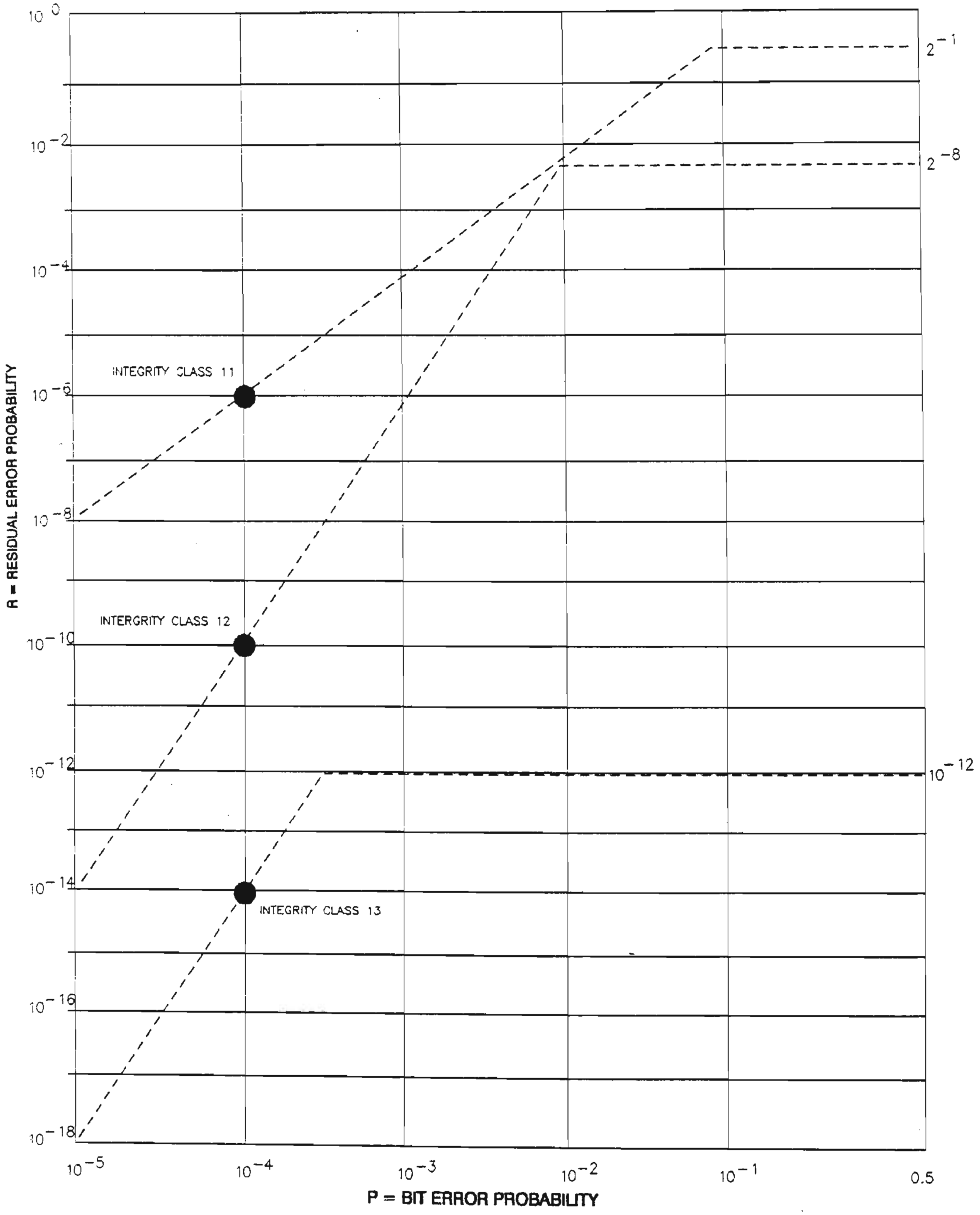
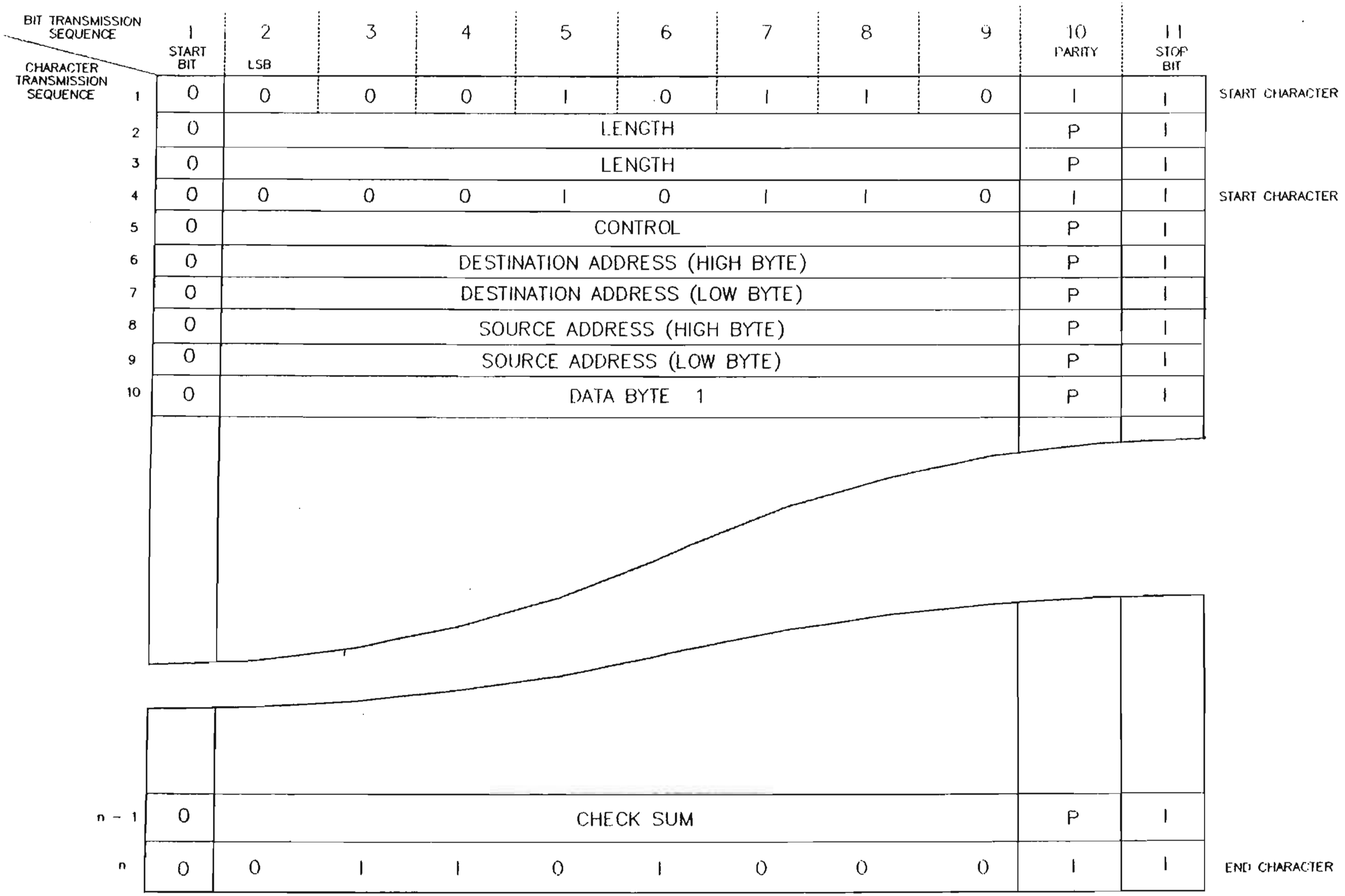


FIGURE : 5. DATA INTEGRITY CLASSES. (IEC . 870 - 5 - 1)



BIT TRANSMISSION SEQUENCE CHARACTER TRANSMISSION SEQUENCE	1 START BIT	2 LSB	3	4	5	6	7	8	9	10 PARITY	11 STOP BIT	
1	0	0	0	0	1	0	1	1	0	1	1	START CHARACTER
2	0	LENGTH								P	1	
3	0	LENGTH								P	1	
4	0	0	0	0	1	0	1	1	0	1	1	START CHARACTER
5	0	CONTROL								P	1	
6	0	DESTINATION ADDRESS (HIGH BYTE)								P	1	
7	0	DESTINATION ADDRESS (LOW BYTE)								P	1	
8	0	SOURCE ADDRESS (HIGH BYTE)								P	1	
9	0	SOURCE ADDRESS (LOW BYTE)								P	1	
10	0	DATA BYTE 1								P	1	
n – 1	0	CHECK SUM								P	1	
n	0	0	1	1	0	1	0	0	0	1	1	END CHARACTER

FIGURE : 6. FT 1,2 DATA LINK TRANSMISSION FRAME FORMAT

(4) The checksum is the arithmetic sum (modulo 256) of all octets starting at the control byte and ending at the byte before the checksum byte, disregarding overflows.

(5) On receipt of a character, the start, stop and parity bits must be checked.

(6) On receipt of an entire frame, the following must be checked:

i. the two start characters and the end character.
ii. validity of the checksum.
iii. equality of the two length bytes.
iv. number of bytes received equals the specified length plus six.

The frame is rejected if any of the checks above fails.

2.2.2.3 Link Transmission Procedures

This standard defines link dialogue procedures for acquiring data from and, controlling processes that are geographically distributed. The transfer of messages are limited to a window size of one and are applicable to balanced and unbalanced transmission with a half or full duplex transmission channel.

In the paragraphs to follow, a primary station is one that initiates a transmission whilst a station that receives the transmission is considered as the secondary station.

In the case of unbalanced transmissions, a station may behave as a primary or a secondary station only. In the unbalanced case, a station may be a primary or a secondary station depending on whether it is transmitting or receiving a message.

The transmission procedures support the following services:

(a) SEND/NO REPLY

(b) SEND/CONFIRM

(c) REQUEST/RESPOND

Figure 7 shows the interaction of the service primitives with these services in the data link layer.

SEND/NO REPLY

On receipt of the request(SEND/NO REPLY) primitive from the link user, the SEND/NO REPLY entity in the data link layer of the primary station is invoked. The transmission frame is compiled as shown in Figure 6 and transmitted. The indication primitive on the secondary station indicates to the correspondent link user of the arrival of the message and results in the termination of the transmission procedure. This service does not support protection against loss or duplication of messages.

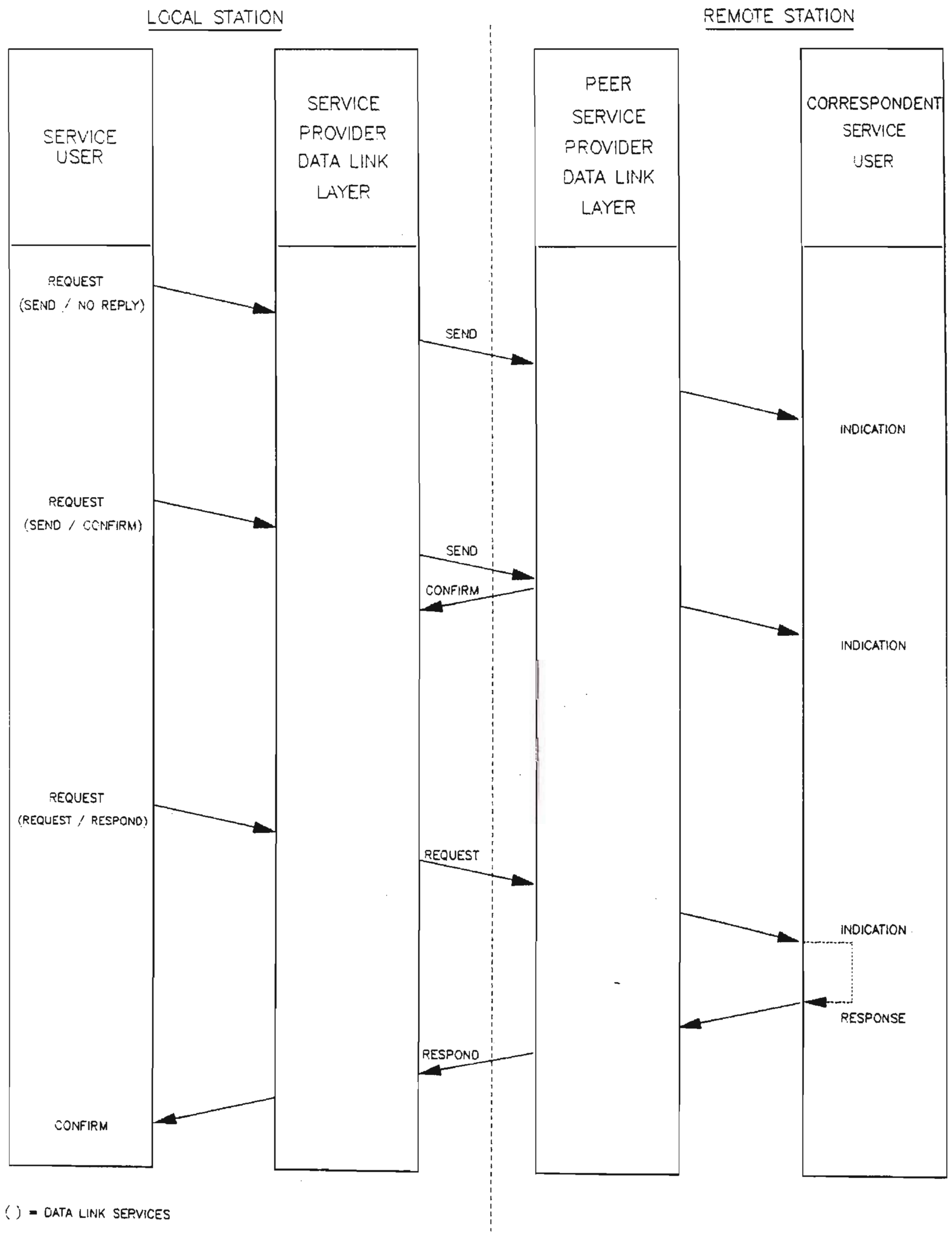


FIGURE : 7 INTERACTION BETWEEN SERVICE PRIMITIVES AND DATA LINK SERVICES IN THE IEC STANDARD.

SEND/CONFIRM

This service is initiated by a request(SEND/CONFIRM) primitive from the data link layer user. The SEND frame is constructed as shown in Figure 6 and transmitted. In the secondary station, the indication primitive signals the arrival of the message, error free. The secondary station will immediately generate a CONFIRM (positive acknowledgement - ACK) frame which is sent back to the primary station. Should an error be detected at the secondary station, such as duplication of a message, a CONFIRM (negative acknowledgement - NACK) with the appropriate error code is reported to the primary station. In either case, the receipt of the CONFIRM frame in the primary station terminates the transmission procedure. If the CONFIRM frame is not received at the primary station after an appropriate timeout, the data link layer will use a suitable retransmission procedure to resend the message. The data link layer in the primary station will report the failure of transmission to the link user if the maximum number of retransmissions have been gone through without success.

REQUEST/RESPOND

This service is invoked by the receipt of a request (REQUEST/RESPOND) primitive from the data link user. A transmission frame as shown in Figure 6, is generated and transmitted to the remote station. The indication primitive from the peer data link entity in the secondary stations heralds the arrival of the frame, error free, to the correspondent data link layer service user. If the requested data is available, the data link layer in the secondary station will transmit the requested data in a CONFIRM frame to the primary station. Should the data not be available at the secondary station, the peer data link layer will report this in a negative CONFIRM frame to the primary station. In the primary station, the issuing of the confirm primitive to the data link layer service user terminates the transmission procedure.

In the packet radio network designed, the balanced mode of transmission using a half duplex channel is used. The format of the transmission frame used is shown in Figure 6.

In the transmission frame the source and destination addresses are defined as 16 bit addresses, where the most significant byte is always transmitted first.

The control field in the transmission frame is used for control of data between the primary and secondary stations and has the format shown in Figure 8.

	MSB							LSB
Bit number	7	6	5	4	3	2	1	0
	DIR	PRM	FCB	FCV	FUNCTION NO			

Figure 8. Control Field Format for Balanced Transmission.

The definition of the various bits in the control field and their relationship to the packet radio network designed, is given below:

DIR : Direction bit specifies the direction of the packet on the network. In the packet radio network designed, the DIR bit is set to 0 for messages originating at the LCU and set to 1 for messages originating at an RTU. The DIR bit in an acknowledgement follow the one from the received packet. This bit is merely set or reset accordingly in this system but currently not been made use of. It could be useful with selection of transmission frequencies when a packet traverses from station to station.

PRM : Primary bit is set to 1 when a station initiates a message and to 0 when a station replies with an acknowledgement, ie. a secondary station.

FCB : Frame Count Bit is used for suppressing duplication of messages. In the primary station the FCB is alternated with each new SEND/CONFIRM message directed to the same station. If the secondary station receives a message from a station with unchanged FCB, then it will send a negative acknowledgement to the primary station with the appropriate function number.

In the case of the reset command (function 6), the secondary station will be set to expect the next FCB from the primary station to be opposite to the one received. The station will otherwise process the received message as normal.

FCV : Frame Count bit Valid is used for indicating validity of the FCB. It is set to 1 if suppression of duplicate messages is to be effected, or 0 otherwise.

FUNCTION NO : specifies the action to be performed in the secondary station from the primary station data link layer or the error reports to the primary station from the secondary station. IEC [1988] provides a list of function numbers used during data transfer between peer data link layers.

2.3 **Channel Access Schemes**

2.3.1 **Overview**

There exist several techniques for accessing a shared medium, but most are weak solutions to providing communications between users on the network. This sections discusses two of the more popular schemes of channel access in a shared medium. These are ALOHA and Carrier Sense Multiple Access (CSMA) respectively. The performance of these channel contention schemes are compared with respect to the following parameters [4]:

(a) **Throughput (S)** - is defined as the average number of successful transmissions during a packet transmission time.

(b) **Offered Traffic (G)** - is defined as the average number of attempted packet transmissions during a packet transmission time.

(c) **Average Transfer Delay** T - is defined as the normalized average time delay between a station initiating a packet and the end of a successful transmission after n retransmissions.

A part of this section deals with the effect of the use of non-ideal radios during channel contention. This effect is demonstrated using one of the CSMA channel access schemes. From this, one can conclude if the known results of Kleinrock & Tobagi [1975], assuming ideal radios, is suitable for comparison purposes.

2.3.2 **Aloha**

Alohanet, one of the first packet radio system, was developed in 1970 at the University of Hawaii [Stallings, 1985]. The radio channel contention protocol used in the system was named Aloha. This system is also generally referred to as "pure Aloha".

In the Aloha system, a common radio channel is used in which all nodes on the network transmit whenever they have data to send, at will and in a completely unsynchronized manner. After the node has made a transmission, it waits for an acknowledgement from the destination. If an acknowledgement is not received during a timeout period, the node assumes that a collision or corruption of the packet has occurred and will use a retransmission strategy to resend the previous packet.

For the purposes of simplicity, the derivation of the equations for throughput and normalized delays is left out. These are handled in detail by Hammond & 'O Reilly [1986].

For Aloha, the relationship between S and G is given by:

$$S = Ge^{-2} \tag{2-3}$$

and between T and G:

$$T = e^{2G} + (e^{2G} - 1)\, B/P \tag{2-4}$$

where:

B is the mean backoff delay and depends on the retransmission strategy used and,

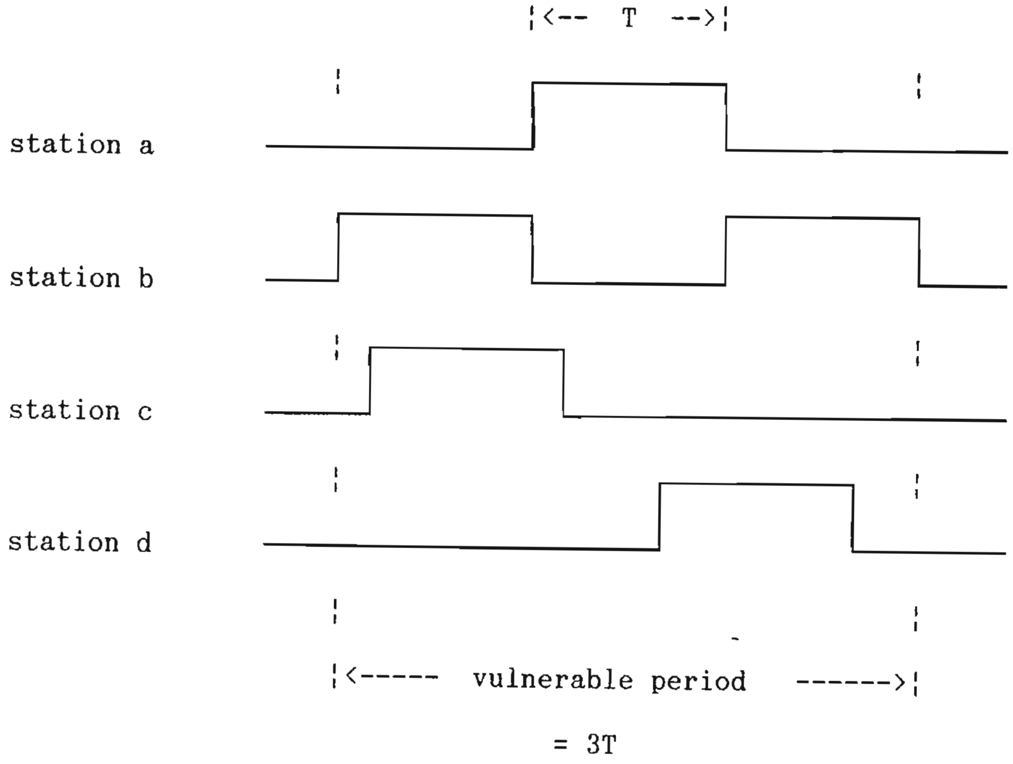
P is the transmission time of a packet.

Graphs of (2-3) and (2-4) are given in [Hammond & 'O Reilly, 1986] and are shown in Figures 10 and 11 below. From these, the instability of Aloha at increased offered traffic and a low maximum throughput (channel capacity) of 0.184 at an offered load of 0.5 can be easily seen. The situation worsens with increasing offered load. These results will be used later for analysis relating to the system to be designed.

A refinement to Aloha is Slotted Aloha. In this system, the time is quantized into slots equal to the maximum transmission time of a packet (P). Transmission from a station is synchronized to start only at the beginning of a slot. If packets were to collide, then collision will only be limited to a slot. A comparison of the transmission overlaps resulting in collisions between Aloha and Slotted is shown in Figure 9.

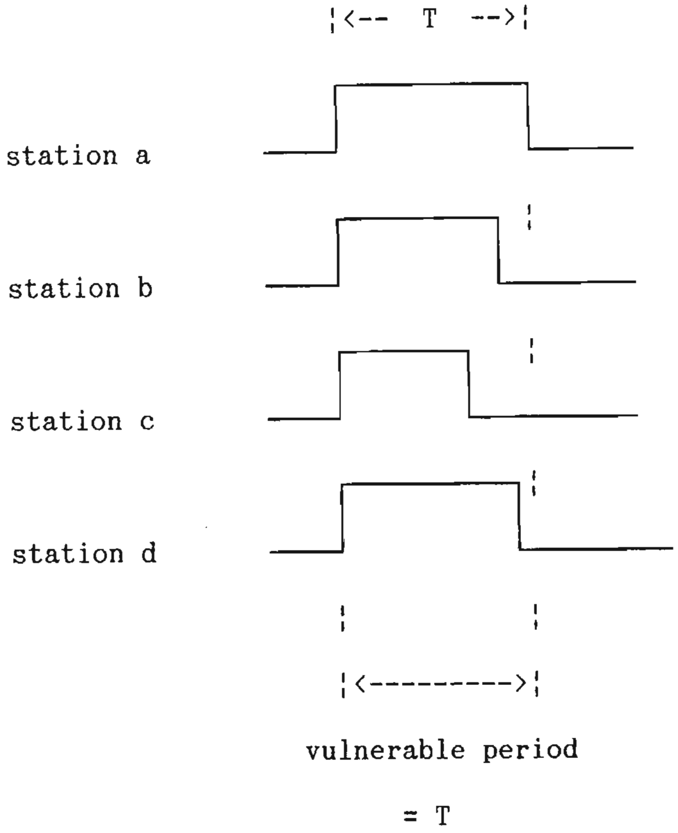
(a)

Pure Aloha:



(b)

Slotted Aloha:



where represents a transmission from a station.

Figure 9. Vulnerable periods for Pure and Slotted ALOHA.

The superiority of slotted ALOHA over pure ALOHA can be seen in Figure 9 which indicates a much shorter vulnerable period for slotted Aloha than for pure Aloha. The vulnerable period is defined as the period during which an overlap of transmission can occur.

A brief analysis of this system is done in Hammond & O' Reilly [1986] and the results are shown below for throughput and normalized delay:

$$S = Ge^{-G} \tag{2-3}$$

$$T = 1.5 + a/3 + ([1 - q_n]/q_t)(r + K/2 + 0.5) \tag{2-4}$$

where:

$$q_t = [(e^{-G/K} - e^{-G})/(1 - e^{-G})][e^{-G/K} + (G/K)e^{-G}]^{K-1}e^{-S} \tag{2-6}$$

K = maximum multiple of slot time used for determining the retransmission back-off delay.

Figure 10, below, indicates immediately the improvement of slotted Aloha over pure Aloha. The normalized delay, as seen in Figure 11, is also worse for pure Aloha than for slotted Aloha.

Slotted Aloha require synchronization between all stations, so that transmissions may only occur during a slot time. Some means of synchronization must be provided, such as transmission of a clock signal on a different frequency from a central controller. Such techniques may be difficult to implement in the situation that we have where multiple RTUs may be out of line of sight and may require some form of repeating in order to get access to them.

2.3.3 Carrier Sense Multiple Access

Carrier Sense Multiple Access (CSMA) is a further refinement over the ALOHA techniques and make use of the property that the propagation delay of packets in a ground radio environment is much smaller than the transmission time of that packet.

The three random access techniques discussed below differ by the action that a station takes after sensing the end of a transmission, or a free channel. A free channel is one in which no transmissions are currently occurring. If the propagation time of the radio transmission was large compared to the actual transmission time of a packet, then the information received at the outlying stations would be too late to be of any use. The channel could be put in further instability by the stations making use of this late information.

The three strategies for CSMA that will be discussed are non persistent CSMA, p-persistent CSMA and 1-persistent CSMA. A detailed analysis of these protocols is given by [Kleinrock and Tobagi, 1975]. In each case, when a station has a packet ready for transmission, it sense the channel and obeys a set of rules in carrying out the transmission. These are outlined below:

(a) non persistent CSMA

1. If the channel is sensed free, then the packet is transmitted.
2. If the channel is sensed busy then the station uses a random backoff algorithm to delay transmission. At the end of this delay, the station repeats 1. above.

(b) p persistent CSMA

1. If the channel is sensed busy, the stations continuously checks the channel until it is free. When the channel becomes free, rule 2. is applied.

2. If the channel is sensed free, then with probability p the station transmits the packet, or with probability (1-p) the station delays for t seconds where t is the end to end propagation delay in the radio channel. At the end of this delay rule 1 is re-applied.

(c) 1 persistent CSMA

1. If the channel is sensed busy, the station continuously checks the channel until it is free. Rule 2. is applied when the channel becomes free.

2. If the channel is detected free, the packet is transmitted with probability 1.

In the cases above, the station may wait for an acknowledgement using a suitable timeout. If after the timeout, an acknowledgement has not been received, the station will backoff a random delay and may retransmit the previous packet.

Carrier sense multiple access provides a much greater improvement over the Aloha systems, but still has its problems, particularly the hidden terminal problem. If a station A wishes to transmit a packet to station B and station C also wishes to transmit to station B, it may not be possible for C to hear A's transmission. There could an obstruction between A and C, such as a building or hill, or A could just be out of range of C. CSMA does avoids certain collisions from occurring but not all.

2.3.4 **Discussion of Random Access Techniques**

Kleinrock and Tobagi [1975] derived a set of equations and curves from their simulation studies of the various random access techniques and are useful in studying the comparative performance of the various systems discussed above. Their results are based on the following assumptions:

(a) The time involved in detection of a carrier due to packet transmissions is negligible.

(b) All packets are of constant length and transmitted over a noiseless channel.

(c) Any overlap of two transmission result in destructive interference and must be retransmitted.

(d) The propagation delay is small compared to the message transmission time.

(e) Ideal receivers and transmitters are used in the network, ie. zero receiver attack and zero transmitter rise times.

Figure 10 shows the relationship between throughput and offered load for the various access schemes with respect to "a". The parameter "a" is the ratio of propagation delay on the channel to the actual transmission time of a packet.

From the set of curves in Figure 10, it can be seen that the non-persistent scheme is much superior at large loads whilst the p-persistent method offers greater throughput at smaller loads. The persistency of the p-persistent scheme may be adjusted in order to

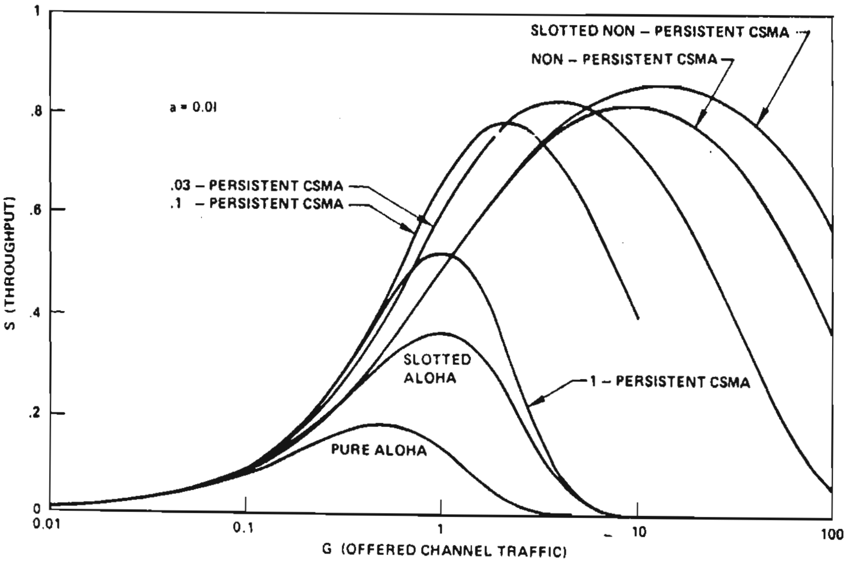


Figure 10. Throughput - Offered Load plots of the various access schemes. [Tobagi, 1980].

obtain the optimum performance for a given network. The channel capacity (C), defined as the maximum throughput for the respective channel access scheme, of the CSMA techniques is much higher than that of the ALOHA schemes.

It is self evident from the set of throughput-delay curves in Figure 11 that the p-persistent technique for channel access has the smallest delays for the respective throughput values.

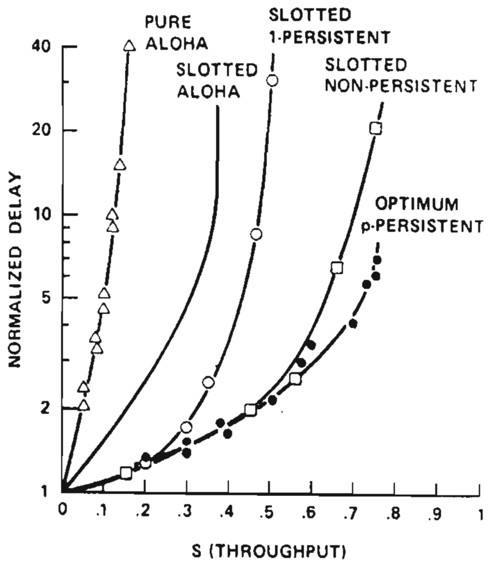


Figure 11. Simulated Throughput-Delay plots for the various access schemes. [Tobagi, 1980].

Figure 12 indicates the effect of the normalized propagation delay "a" on channel capacity. It is can be seen that only the CSMA systems are sensitive to the normalized propagation delay. At large values of a, the ALOHA access schemes are more superior. This proves that CSMA systems using carrier sense information that is late is detrimental to the stability of the network.

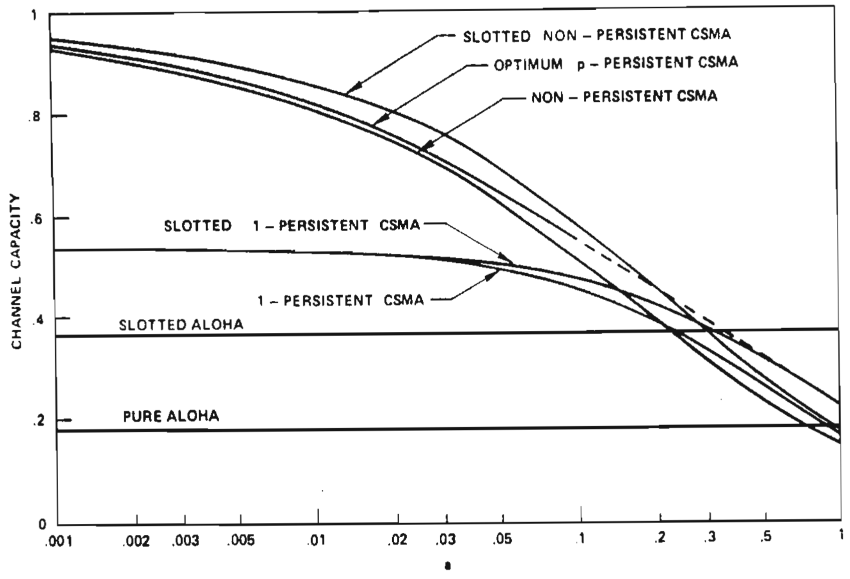


Figure 12. Effect of propagation delay "a" on various access schemes. [Tobagi, 1980].

2.3.5 Choice of Access Protocol

From the plots of offered traffic and throughput in Figure 10, it would seen obvious to select the non-persistent CSMA protocol for the packet radio system to be designed. However, certain characteristics of the system may lead one to select otherwise.

In the computations done by Kleinrock and Tobagi [1975], an ideal transmitter in the transmitting station was assumed , ie. zero transmitter rise time. With off the shelf radios, the transmitter rise times are in the order of 250 milliseconds, and is a substantial part of the total transmission time.

The transmitter rise time is important as it increases the window for collisions. When a station, in the designed packet radio network, is in the process of transmitting, it is not able to sense the channel.

With the packet radio system operating at a bit rate of 1200 bits/second, and an average packet duration of twenty bytes, transmitter rise time of 250 ms, in system with the maximum distance between stations being fifty kilometers a quick calculation gives:

Total transmission time = transmitter rise time + packet transmission time + propagation delay

= 433.5 ms

Ratio of propagation delay to total transmission time (a) = 0.00038

Ratio of transmitter rise time to total transmission time (Tr):

Tr = 0.5767

To indicate the effect of Tr on channel throughput, Appendix A gives the calculation of channel throughput using the additional delay Tr for the non-persistent transmission scheme. A plot with the Tr = 0.5767 and "a" = 0.00038 and a plot with Tr = 0 and "a" = 0.5767 is done in Figure A1. This was done in order to determine a relationship between the two results, to enable usage of the results obtained by Kleinrock and Tobagi [1970]. This was done as the analytical and simulation techniques used to obtain the results of the p-persistent scheme are exhaustive.

From the plot in Figure A1, it is clear that the results obtained with Tr = 0.5767 and a = 0.00038, compares closely with Tr = 0 and a = 0.5767. The latter case is Kleinrock and Tobagi's [1975] results. The channel capacity is however higher with a high Tr value than with a similar high "a" value. This indicates that the curves in Figure 10 may be used for purposes of comparison in system to be designed by making "a" equal to the Tr value of the packet radio system to be implemented. This will obviously give the worst case scenario.

Using the curves from the simulation of Kleinrock and Tobagi [1970], Figure 12 , it can be seen that for an "a" value of approximately 0.57, slotted ALOHA is much superior, followed by the p-persistent, non-persistent and pure ALOHA access schemes.

In the packet radio system to be implemented, the implementation of slotted ALOHA is not feasible. This stems from the fact that synchronization of slots from a central station is difficult with nodes operating in half duplex mode. The synchronization transmission may interfere with normal transmissions. The use of repeaters further complicate the task of synchronization as shown by Tobagi [1980].

A suitable comprise for the packet radio network to be designed, is the p-persistent technique of channel access. The "p" value was chosen to be 0.1 in order to provide a compromise on unnecessarily large delays on transmission of packets and loading of the network.

2.3.6 **Transmission Procedure**

This section covers the transmission rules obeyed by a station when it has a packet ready to send onto the network.

All stations will wait a minimum of two propagation delays before transmitting after sensing the channel free. This is done in order to allow stations with a wait of one propagation delay to have priority over the station with two propagation delays. A propagation delay comprises the rise time of a radio (press to talk delay - PTT) and the packet propagation delay through the medium.

The transmission of packets from a station uses 0.1 persistent CSMA to get access to the network initially. Thereafter, stations receiving data may gain access to the network by the "Master Of Network" rule. This rule is useful when gaining access in a heavily loaded network.

The rule uses the principle that enables a station to become master of the network once it has gained access to the medium. Using CSMA, all other stations will listen to the transmission from this station and will not transmit. Figure 13 illustrates the packet relay cycles using such a rule.

The transmission rules are best illustrated by an example as in Figure 13. Station 1 transmits a message to station 2 after getting access to the radio channel using 0.1 persistent CSMA. At the end of its transmission, the channel will become free. At this point in time, station 2 has received the whole packet and processes the acknowledgement for station 1, taking a small fraction of a PTT delay. Provided that all stations have been listening and will not transmit before the end of two propagation delays, station 2 may send its acknowledgement to station 1 immediately.

At the end of receiving the packet from station 1, station 2 becomes master of the network. It will send an acknowledgement, if one is required, and any one other message that it has to transmit. This message is "piggybacked" onto the acknowledgement. The piggybacking of another message onto the acknowledgement relieves station 2 from using CSMA to get access to the channel again.

Once station 2 completes it transmission, the whole process is repeated with the station that receives station 2's transmission (not the acknowledgement). Hence a receiving node becomes master of network if it has a message to transmit.

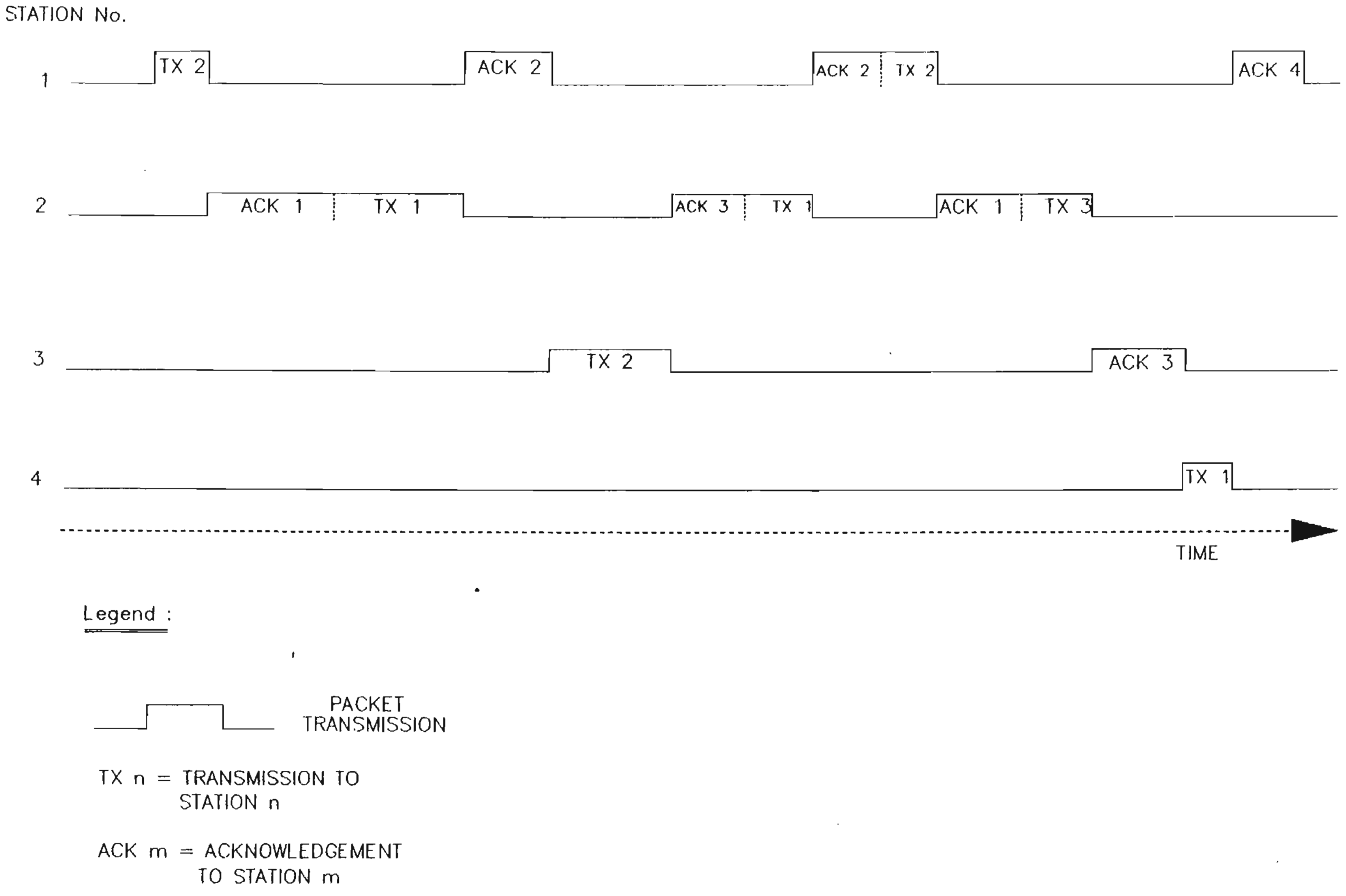


FIGURE : 13 PACKET RELAY TIMING DIAGRAM

This feature is useful when relaying packets through multiple stations, a phenomenon that is commonly referred to as digipeating.

The node that the message is destined for will reply with the acknowledgement almost immediately.

At the end of transmission of the message onto the network, the data link layer will wait a time period for the acknowledgement from the remote station. If an acknowledgement (ACK), or negative acknowledgement (NAK) has not been received after the timeout period, then the message is retransmitted after a delay.

The rule used in calculating the delay is known as the *truncated binary exponential backoff rule* [Stallings 1990, Volume 2]. The backoff delay is calculated as an integral number of slot times where a slot time is the worst case round trip propagation time for a packet and its acknowledgement. The following psuedo code shows the method of determining the delay before the nth retransmission:

```
while attempts < DL_MAXRETRY
        k == Minimum (n,10)
        r == Random (0,$2^k$)
    delay == r x slot_time
```

r is a uniformly distributed random integer in the range

$0 < r =< 2^k$.

If an ACK has not been received after the maximum number of retries, then the data link layer will alert the data link layer user of this.

The whole process is repeated when a new message becomes ready for transmission.

2.3.7 Retransmission Strategy

The concept of Automatic Repeat Request (ARQ) is used to provide a means of enabling the transmitting station to repeat packets that were unsuccessful. This can be the result of either the transmitted or received packet being corrupted.

The "stop-and-wait" ARQ retransmission protocol is used in the packet radio network. With this type of ARQ, the transmitting station will wait for the acknowledgement (ACK) or negative acknowledgement (NAK) to the message just sent before proceeding to send the next message.

In order to identify ACKs or NAKs to the correct REQUESTs, a modulo 2 sequence number is used in all packets. In the protocol, this corresponds to states 0 and 1 for sequencing. The sequence number is represented as a single bit in the control byte of the header.

The operation of the sequence numbers is as follows:

Assume that a station j wishes to transmit a message to station i. j's sequence number is initially 0. In the message sent to i, the sequence number will be set to 0. Upon receipt of an error free message from station j, station i will transmit an ACK to station j with sequence number 1, ie. the sequence number of the next message expected by station i. On receipt of an error free ACK from station i, station j will proceed to transmit the next message to station i with sequence number 1.

This technique of ARQ is simple to implement and is rugged, though not making efficient use of the channel.

3 EXPERIMENTAL PACKET RADIO NETWORK DESIGN

3.1 Introduction

The packet radio network designed is based on a system of five hundred Remote Terminal Units (RTUs), that are geographically dispersed, and a Line Control Unit (LCU).

Figure 1 represents a block diagram of the system. Each RTU and the LCU consists of a processor, buffer, modem and a radio unit. The LCU is linked to a personal computer using CONLOG's local area network CONET. The personal computer is used as the Man Machine Interface (MMI) to the system.

In the system being designed, the RTUs are used as controllers for the control of reticulation power networks. Each RTU as well as the LCU, will contain packet switching and routing software for the packet radio network specified below.

The architecture used to specify the packet radio network is based on the seven layer Open Systems Interconnect model of the International Standards Organization.

In the packet radio network designed, the following layers have been implemented:

	OSI Layer number
(a) Physical	1
(b) Data Link	2
(c) Network	3
(d) Presentation	6

This chapter covers the specification of the packet radio network protocol, PACNET, the hardware base used for the protocol software, the software design and the testing and performance of the packet radio network.

In the protocol specification below, the services provided by each layer and the services used by that layer from the layer below it, is defined. The protocol is specified within the framework of the OSI standard. The methodology of specifying the protocol layers is based on the methods used by Halsall[1988].

3.2 Protocol Specification

3.2.1 Physical Layer

3.2.1.1 Overview

The physical layer is concerned with the rules used in passing the raw bit stream from one station to another and it covers the electrical, functional, and procedural means of interfacing the station to the packet radio network. It covers signal element generation/detection, bit synchronization, signal quality supervision and transmission speeds.

Figure 14 shows a block diagram of the respective components of the physical layer.

The interface between the physical layer and the networking software is provided by the Serial Communications Interface (SCI) and is based on the RS232 standard. The Modem Interface is used in interfacing the SCI to the radio. The radio provides the interface between the actual medium and the serial bit stream from the modem.

3.2.1.2 Radio Interface

The communications media in this application is the air. Data is sent using UHF radio communications. The radio interface conforms to the following specifications:

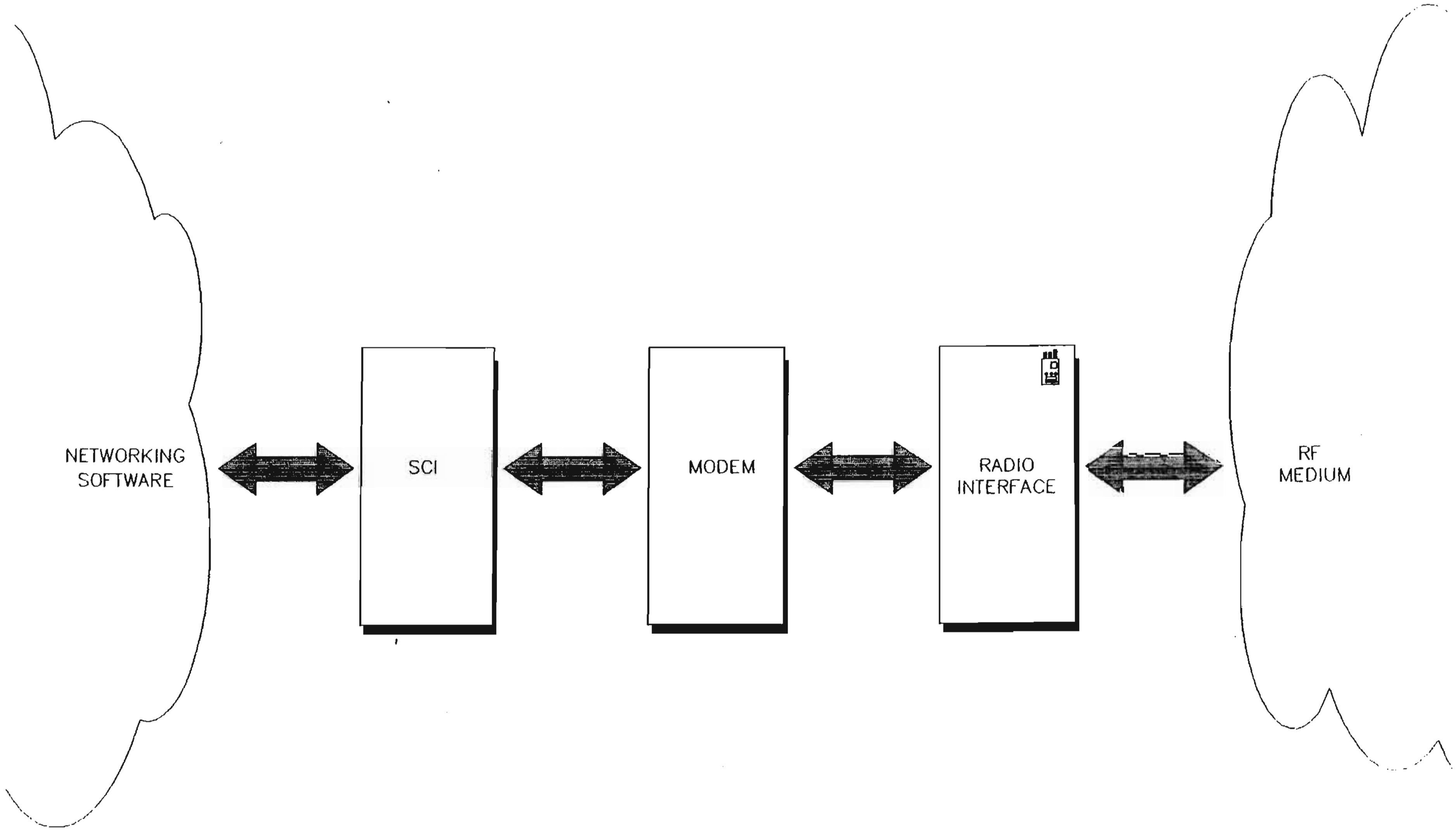


FIGURE : 14. PHYSICAL LAYER COMPONENTS

Frequency band : 406 to 420 MHz

Channel Spacing : 12.5 kHz

Modulation : FM with 2.5 kHz maximum deviation.

Mode of operation : Half-duplex ie. different receive and transmit frequencies.

A description of the communications network is necessary to determine certain parameters of the physical layer such as carrier detection and timing. Refer to figure 1 for an overview of the network.

Communications can take place between the central controller and the outlying RTUs in either direction. All stations receive and transmit on different frequencies but not simultaneously.

The radio interfaces to the modem via the following lines (refer to figure 15):

- Transmit analog pair
- Receive analog pair
- Press-to-talk (PTT)
- Carrier detect

The radio carrier detect is logical HIGH when a signal is present and is logical LOW otherwise. The PTT line is HIGH in receive mode and must be pulled LOW to transmit.

The radio will normally be in the receive mode with the audio output squelched. The reason for this is that it is undesirable for the modem to continuously receive noise out of which it could extract

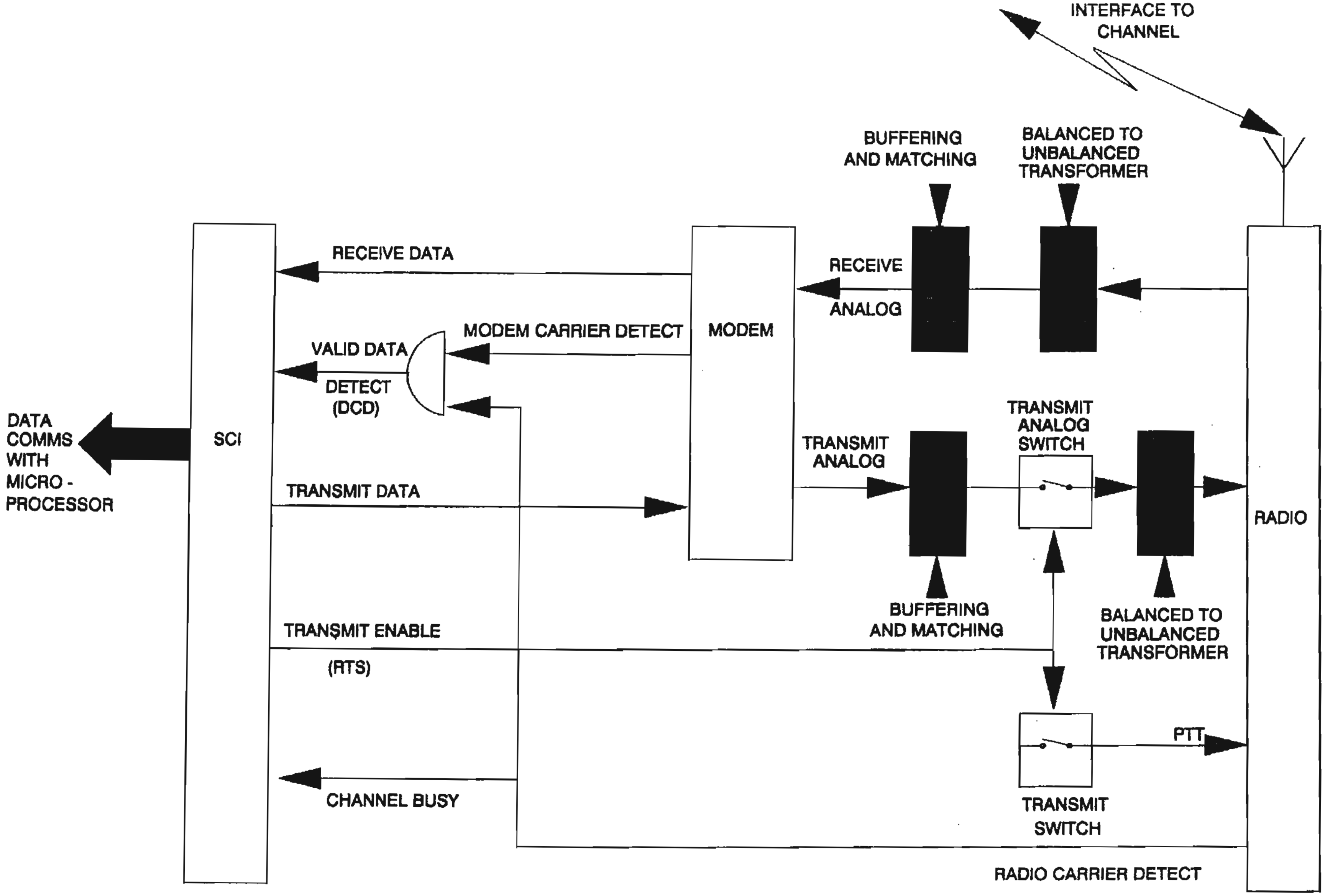


FIGURE 15 : SERIAL COMMUNICATIONS / MODEM / RADIO INTERFACE

spurious data. The squelch threshold will be setup to greater than or equal to the sensitivity of the radio (0.4 µV). Any signal that is received less than this would have a poor signal-to-noise ratio (<12 dB) and would be prone to error. The exact squelch threshold to be setup will be determined by site trials to determine the level of ambient noise in a typical installation. The threshold must always be less than the minimum expected signal strength of a valid received message. This figure will also be determined by site trials but is likely to be of the order of 0.8µV.

The method of squelch operation which will be employed in the radio used will have two levels of detection. The first will be normal carrier level detection. The second will look for noise reduction due to the quietening effect of a valid carrier. These techniques are likely to result in more reliable squelch operation.

Even with these precautions it is likely that in some areas the noise level will be above the squelch threshold and hence further measures must be taken to filter this noise. These methods are discussed in the next section on the modem.

3.2.1.3 **Modem**

The modem provides the interface between the radio and the serial communications interface (SCI). The SCI forms an integral part of the software interface.

In the transmit direction the main function of the modem is to convert the serial data to an analog signal that can be use to modulate the transmit carrier frequency of the radio. In the receive direction the modem must take the demodulated signal from the radio receiver and convert it back to serial data.

The modem used for this application conforms to the following specifications:

Modulation type	: Minimum Shift Keying (MSK)
Operating mode	: Transmit and receive in four wire mode. Full duplex operation.
Baud rate	: 1200 bits/s
Frequency assignments	
Mark or 'high' frequency	: 1200 Hz
Space or 'low' frequency	: 1800 Hz
Backward channel	: not required
Impedance of transmit pair	: 600 ohms balanced
Impedance of receive pair	: 600 ohms balanced
Return loss (300 to 3400Hz)	: >20 dB
Transmit signal level	: -20 dBm to +5 dBm
Receive signal level	: -20 dBm to +5 dBm

Minimum Shift Keying (MSK) was chosen as the modulation technique for the following reasons [OSTEN, 1980]:

(a) It provides a 3 dB improvement on the signal to noise ratio in the receiver. The bit error rate for a given signal to noise ratio is lower than that of FSK.

(b) Generates less significant sidebands which reduces distortion for tight bandwidths.

The other important function of the modem is to provide a carrier detect output to the microprocessor in order that it may determine whether the channel is busy is busy or not.

The modem interfaces with the radio via the lines described in the previous section (3.2.1.2).

It interfaces with the SCI via the following lines:

- Transmit Data
- Receive Data
- Transmit Enable
- Valid Data Detect
- Transmit Clock
- Receive Clock

Figure 15 shows the interfaces between the modem and its peripherals. It should be referred to in the following discussion.

Receive mode

If the radio squelch is "broken", the modem will receive an analog signal on its receive analog input. This may be either a valid signal or a high level of ambient noise. The received signal is passed through an anti-aliasing filter and then through an amplifier with automatic gain control (AGC). A receive bandpass filter then limits the bandwidth of the signal reducing out of band interference (such as noise). The signal then passes through a group delay equaliser and a limiter. A demodulator then performs frequency to voltage conversion. This signal is finally filtered and passed through a slicer to produce the final digital output signal on the modem receive data output.

A mark or high bit will be produced by an analog signal of 1200 Hz. A space or low bit will be produced by a frequency of 1800 Hz.

If there is no valid analog signal on the modem receive input the data output will be continuously HIGH. This is achieved by pulling this output to logic HIGH. The line idle state is therefore continuously HIGH.

A carrier detect output with some hysteresis (2.5 dB) and an adjustable threshold is also included (-45 dBm minimum). The carrier detect is HIGH for detected data and LOW otherwise. This carrier detect provides a further level of protection against noise. It is only set if there is sufficient in-band energy.

The data is finally received by the serial communications interface. This function is described in detail in the next section (3.2.1.4).

When valid data finally arrives at the SCI, The VALID DATA DETECT line is presented to the microprocessor. This line is generated by the logical AND of the RADIO CARRIER DETECT and MODEM CARRIER DETECT lines.

Transmit Mode

The data to be transmitted is passed from the SCI into the transmit data port of the modem. The data is then converted to analog frequencies by using the MSK modulator. A mark or high bit produces a 1200 Hz signal and a space or a low bit produces a 1800 Hz signal. The analog signal is then passed through a filter and amplifier and appears on the transmit analog output of the modem.

When no data is being sent, the transmit data input to the modem will be continuously HIGH by being tied to logic HIGH. No signal will appear on the transmit analog input to the radio because the analog signal will pass through a switch which will be activated by the TRANSMIT ENABLE (active HIGH) line from the microprocessor.

A HIGH on the TRANSMIT ENABLE line will also pull the PTT line to the radio LOW, via the transmit switch, which will cause the radio to start transmitting.

3.2.1.4 Software Interface

The software interface is used to interface the MSK modem to the networking software. It is primarily made up of a serial communications interface which allows for byte by byte transmission from the upper layers onto the network.

The SCI makes the following flags and registers available to the software (some of the flags are indicated in Figure 15):

(a) CHANNEL BUSY : This flag is set by the physical layer when it detects any radio frequency energy in the shared channel. This would imply activity in the shared radio channel. In the packet radio network design, this flag is set when the squelch in the radio is broken.

(b) RTS : Request To Send is a flag that must be set by the upper layers prior to transmission of a message. In the packet radio design, this flag will cause the Press To Talk (PTT) circuit of the radio to be activated.

(c) TDRE : Transmit Data Register Empty is a flag that is set the physical layer to indicate to the upper layers that the previous byte has been transmitted and the data register is empty. The upper layer may then place another byte in the transmit data register for transmission.

(d) RDRF : Receive Data Register Full is a flag that is set by the physical layer to indicate to the upper layers that a character has been received from the network and should be removed from the receive data register.

Although the physical layer can handle only byte by byte transmissions, the final transmitted packet has the structure shown in Figure 16.

PREAMBLE	DATA	INTERFRAME DELAY
44 bits	121 to 2871 bits	33 bits

Figure 16 . Structure of transmitted packet.

The 4 byte preamble, as discussed by Kahn et al [1978], are used in the radio section of the receiving station to provide the following functions:

(a) the first few bits are used to detect the carrier energy and to set the automatic gain control (AGC) to compensate for varying received signal strengths.

(b) the rest of the bits are used to provide bit timing in the modem circuit and to provide packet timing in the data link layer.

3.2.2 Data Link Layer

3.2.2.1 Overview

The Data Link Layer (DLL) is based on the IEC [1988] draft standard.

The standard has been modified sightly to suit the needs of the packet radio network designed. These include the allocation of certain new function numbers that are allocated for the use by the system. The IEC standard has reserved these function numbers for general use.

NOTE: A station is considered as a primary station if it initiates a message and station is considered a secondary station if it receives a message from the primary station. The DLL supports configurations in which there are multiple primary and secondary stations.

3.2.2.2 User Services

The DLL provides the upper layers with the following services:

(a) Error free transmission over the physical medium

(b) Error detection and control

(c) Handling loss and duplication of data

The services above are known as the SEND/CONFIRM service in IEC [1988].

Additionally, the SEND/NO REPLY service has been implemented. This service does not support handling of loss and duplication of data.

These services are invoked with the following primitives:

Primitive	**Parameters**
DL_DATA.request	Destination Address (REQ)
.indication	Source Address (IND)
	Maximum Retransmission Count (REQ)
	Data Link service user data (NL-PDU)
	Priority (REQ)
	Confirmation Required (REQ)

Table 1. Data Link layer service primitives.

The bracketed items in Table 1 relate the parameters to the respective primitive. If no bracketed items are used, then the parameters is used for all the primitives

The destination address passed with the request primitive is the physical address of the destination station to which the packet must be delivered.

The source address passed with the indication primitive is the physical address of the station that initiated the packet.

The maximum retransmission count parameter is passed in with the request primitive and is used to put an upper limit on the number of times the PDU must be retransmitted in the data link layer, when recovering from loss of data.

The priority parameter is used to pass priority 0 (P0) or priority 1 (P1) data to the DLL. Priority 1 data gets preference over priority 0 data for transmission.

The "confirmation required" parameter is used to invoke either the SEND/CONFIRM or SEND/NO REPLY service in the DLL.

Figure 17 is a time sequence diagram illustrating the interaction of the primitives between the network layer (NL) and the services in the DLL.

3.2.2.3 **Protocol Operation**

The Data Link Layer achieves peer to peer communications with the generation of two PDUs. These are the request and acknowledgement PDUs. The format of these PDUs is based on the IEC Link Transmission procedures standard [IEC,1988], and are shown below:

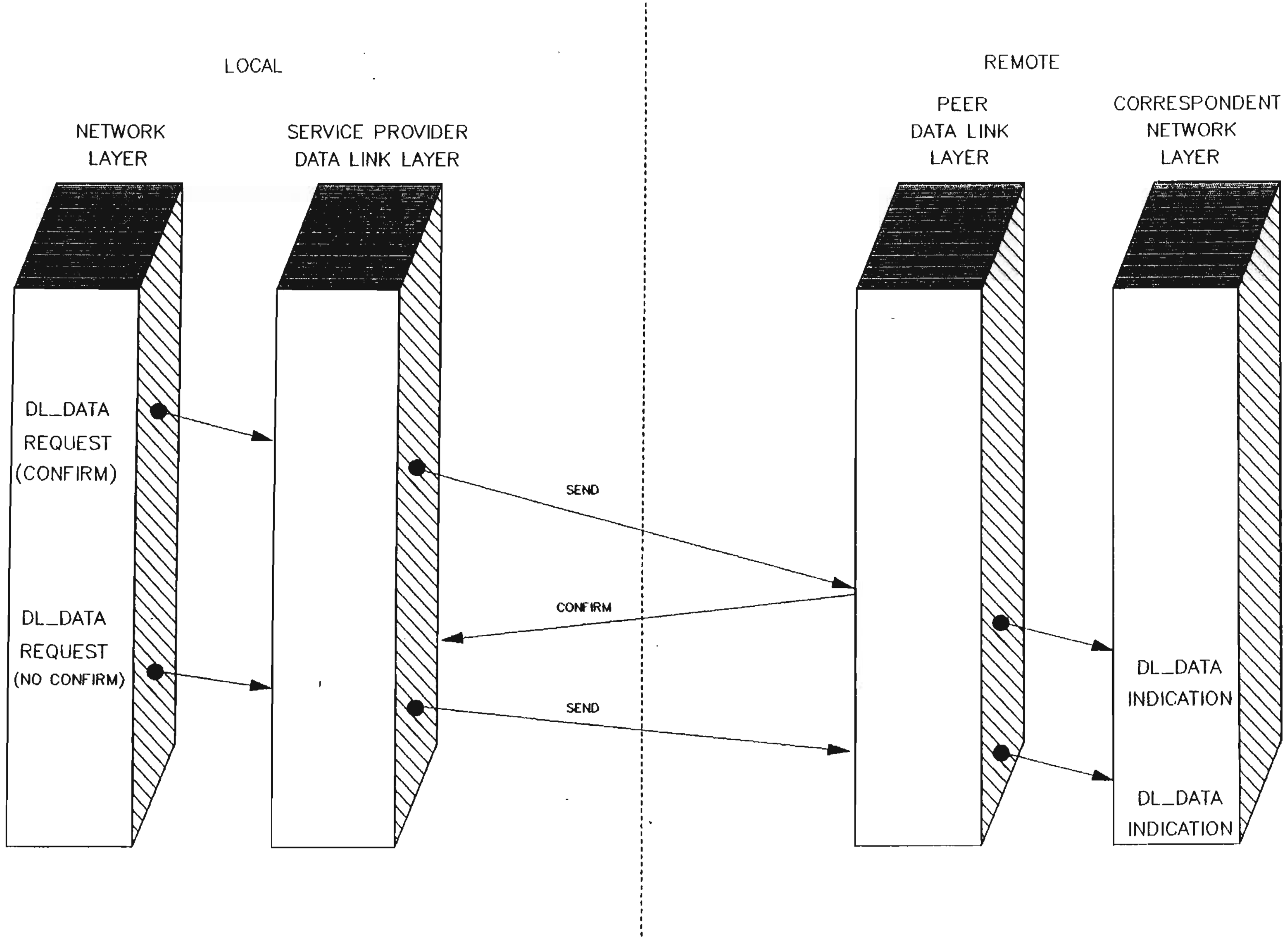

LOCAL
REMOTE
NETWORK LAYER
SERVICE PROVIDER DATA LINK LAYER
PEER DATA LINK LAYER
CORRESPONDENT NETWORK LAYER
DL_DATA REQUEST (CONFIRM)
DL_DATA REQUEST (NO CONFIRM)
SEND
CONFIRM
SEND
DL_DATA INDICATION
DL_DATA INDICATION

FIGURE : 17. DATA LINK LAYER PRIMITIVES.

Request Frame:

Byte sequence	Fields	VALUE (hexadecimal)
0	SYNC	55
1	SYNC	55
2	SYNC	55
3	SYNC	55
4	START	68
5	LENGTH	
6	LENGTH	
7	START	68
8	CONTROL	
9	DESTINATION ADDRESS (MSB)	
10	DESTINATION ADDRESS (LSB)	
11	SOURCE ADDRESS (MSB)	
12	SOURCE ADDRESS (LSB)	
13	USER DATA BYTE 1	
14	USER DATA BYTE 2	
15	USER DATA BYTE 3	
"	"	
"	"	
"	"	
(12+n)	USER DATA BYTE n	
(12+n)+1	CHECKSUM	
(12+n)+2	END	16

Acknowledgement Frame (reply):

0	:	SYNC	:	55
1	:	SYNC	:	55
2	:	SYNC	:	55
3	:	SYNC	:	55
4	:	START	:	68
5	:	LENGTH	:	5
6	:	LENGTH	:	5
7	:	START	:	68
8	:	CONTROL		
9	:	DESTINATION ADDRESS (MSB)		
10	:	DESTINATION ADDRESS (LSB)		
11	:	SOURCE ADDRESS (MSB)		
12	:	SOURCE ADDRESS (LSB)		
13	:	CHECKSUM		
14	:	END	:	16

where:

SYNC : is the modem synchronization characters.

START : is the start character according to IEC [1990].

LENGTH : specifies the number of octets from the CONTROL byte to the byte before the CHECKSUM, inclusive of these bytes.

CONTROL : is used during message parsing to enable the data link layer to carry out various functions, such as determining whether acknowledgements are required, sequencing of packets etc. A detailed description is given in Section 2.2.2.3.

DESTINATION ADDRESS : is the physical address of the destination station to which the PDU is to be delivered. MSB and LSB specify the bytes that contain the most significant and least significant bits of the 16 bit address.

SOURCE ADDRESS : is the physical address of the source station from which the PDU has been sent. MSB and LSB specify the bytes that contain the most significant and least significant bits of the 16 bit address.

USER DATA BYTES : is the user network layer PDU (NL-PDU).

CHECKSUM : is the arithmetic sum (modulo 256) of the bytes from the CONTROL byte to one byte before the CHECKSUM byte, inclusive of these bytes and disregarding overflow.

END : specifies the end of the frame.

The control byte in the PDU is used to control data flow between peer data link layers and contains the following information:

	MSB							LSB
Bit no:	7	6	5	4	3	2	1	0
	DIR	PRM	FCB	FCV	FUNCTION NUMBER			

Table 2 provides a list of function codes implemented in the design.

Messages sent from primary station (requests) :

FUNCTION CODE	SERVICE
3	User Data with no reset of FCB
6	User Data with reset of FCB

Messages sent from secondary station (replies) :

FUNCTION CODE	FRAME TYPE	SERVICE
0	CONFIRM	Positive Acknowledgement
1	CONFIRM	Negative Acknowledgement due to overflow in receiving buffers
2	CONFIRM	Negative Acknowledgement due to duplicate message received.

Table 2. Table of Function Codes for Requests and Replies.

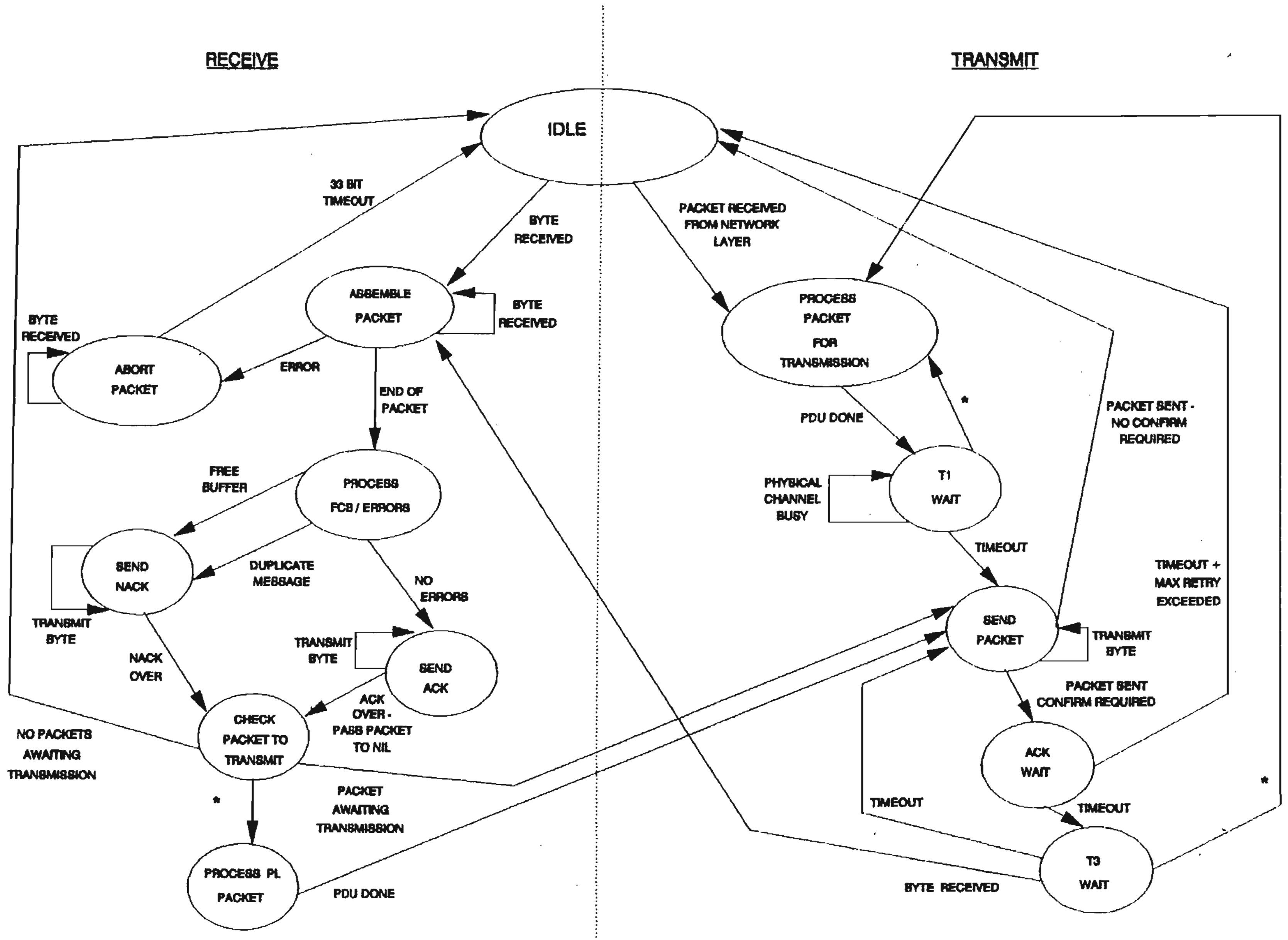
DL DATA.request primitive:

On receipt of a request primitive from the NL, the DLL will carry out the actions indicated in the state transition diagram of Figure 18 for the "transmit case".

With the receipt of a request primitive from the NL, the DLL entity will assemble a transmit PDU as shown above and enter a T1 wait state if the channel is free. In this state the DLL times out for the T1 timeout period before transmitting the PDU. T1 is a calculated as a random number in 10 PTT delays, to provide the 0.1 persistency of the CSMA protocol used. If the channel gets busy during the T1 timeout, then DLL will wait for the channel to become free. When the channel becomes free, the T1 timeout is recalculated and the whole process is repeated. The arrival of a P1 message for transmission will cause the DLL to process the P1 message first.

With the expiry of the T1 timeout, the DLL will attempt to transmit the frame byte by byte. At the end of transmission, if the SEND/NO REPLY service was used the DLL will revert to the IDLE state. If the SEND/CONFIRM service was invoked, the DLL will wait for the acknowledgement. With the receipt of an acknowledgement, the DLL will revert to the IDLE state. If no acknowledgement is received, the DLL will wait a T3 timeout period. The generation of T3 is based on the "truncated binary exponential backoff rule" as discussed in section 2.3.6.

If the DLL receives a request with a P1 message during a T3 timeout, it will process the P1 request first.



* P1 PACKET ARRIVED
WHILE P0 PACKET
AWAITING TRANSMISSION

FIGURE : 18 DATA LINK LAYER STATE TRANSITION DIAGRAM

While the DLL is in a T3 wait state, it may receive bytes from the network asynchronously.

At the end of a T3 wait, the DLL will attempt to retransmit the message as discussed above. If after n retransmissions, the acknowledgement has not been received, the DLL will report this to the NL and revert to the IDLE state.

DL DATA.indication primitive:

Figure 18 also represents the state transition diagram of the receive section in the DLL.

As each byte is received from the physical layer, the DLL will attempt to assemble a packet. If an error occurs prior to assembly of the entire packet, the DLL will go into a state in which all further bytes received are aborted. Errors may occur as a result of parity checks failing, packet not meant for the station etc.

When a packet is received completely and error free, it is checked for duplication (incorrect FCB) of messages and for availability of free buffer space. If any of these error occur, the DLL will transmit a negative acknowledgement (NAK) to the initiating station. If no errors are found, then an acknowledgement (ACK) is transmitted to the initiating station. The ACK or NAK is omitted in the case of the SEND/NO REPLY service.

If no errors have been found in the packet, the DLL will pass the message to the NL and will return to the IDLE state if no packets are awaiting transmission. If a packet is awaiting transmission, the DLL will attempt to transmit the packet immediately (piggy backed with the acknowledgement).

Figure 19 is a time sequence diagram showing the transfer of SEND/CONFIRM frames in the DLL with no disturbances on the network. Figure 20 shows a scenario where the CONFIRM frame from station A as well as the SEND frame from station B is corrupted as a result of a collision. After the T3 retransmission timeout, station A attempts a transmission and gets its frame through to station B. Since station B has a SEND frame to send to station A, it will piggy back this message with the acknowledgement frame to station A. Station A, on receipt of the acknowledgement and SEND frame from station B, will generate a CONFIRM frame to station B.

The duplication of messages in the case of SEND/CONFIRM procedures where the packet may be retransmitted n times is handled using a modulo 2 sequence number in the transmitted message. This corresponds to the Frame Count Bit (FCB) in the control byte.

Since each station has the capability of communicating with all other stations in the network, each stations needs to keep a record of FCBs of messages transmitted and received from other stations.

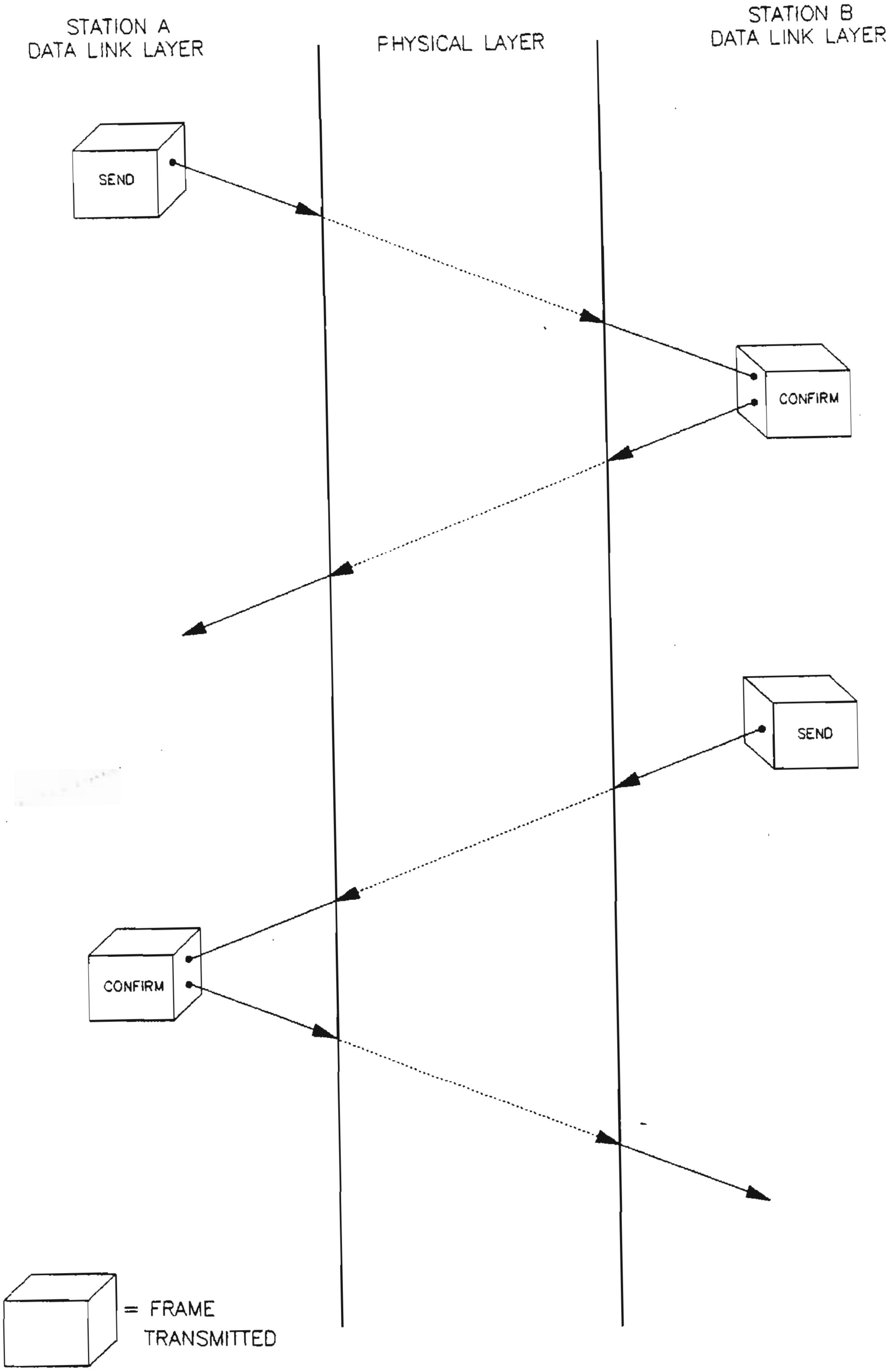


FIGURE : 19 UNDISTURBED SEND / CONFIRM SERVICE IN DATA LINK LAYER

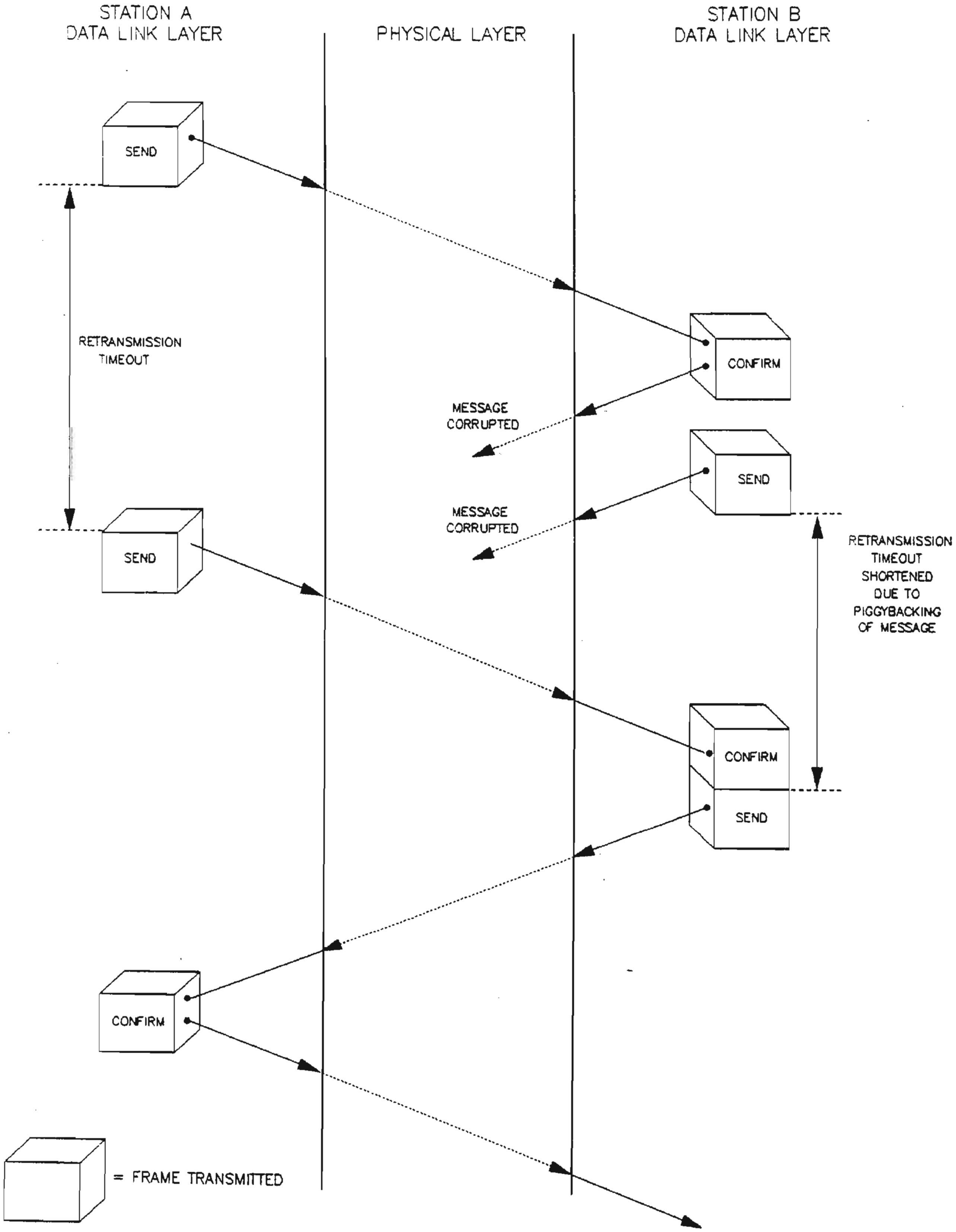


FIGURE : 20 DISTURBED SEND / CONFIRM SERVICE IN DATA LINK LAYER

Each station keeps a table of transmitted and received FCBs as shown in Figure 21.

Station no.	RXFCB	RXRES	TXFCB	TXRES
1				
2				
3				
511				
512				

Figure 21 Frame Count Bit Table.

On power-up all locations in the table is set to 0. The data link layer will use the source station address to index into the table.

The flowcharts in Figure 22 outlines the sequence of checking that occurs when receiving messages with a valid frame count bit and transmitting messages that require a valid frame count bit. In both cases, only messages that require acknowledgements follow the sequences shown in Figure 22.

3.2.2.4 Use of Physical Layer Services

The services used by the DLL to transmit and receive bytes from the network is outlined in section 3.2.1.4.

3.2.3 Network Layer

3.2.3.1 Overview

The primary function of the network layer is to route packets in the packet radio network.

The network layer will add the necessary control information and data to the PDU passed from the upper layers in order to transport the packet to its destination.

It will also determine whether messages received are for the station or for store and forward repeating.

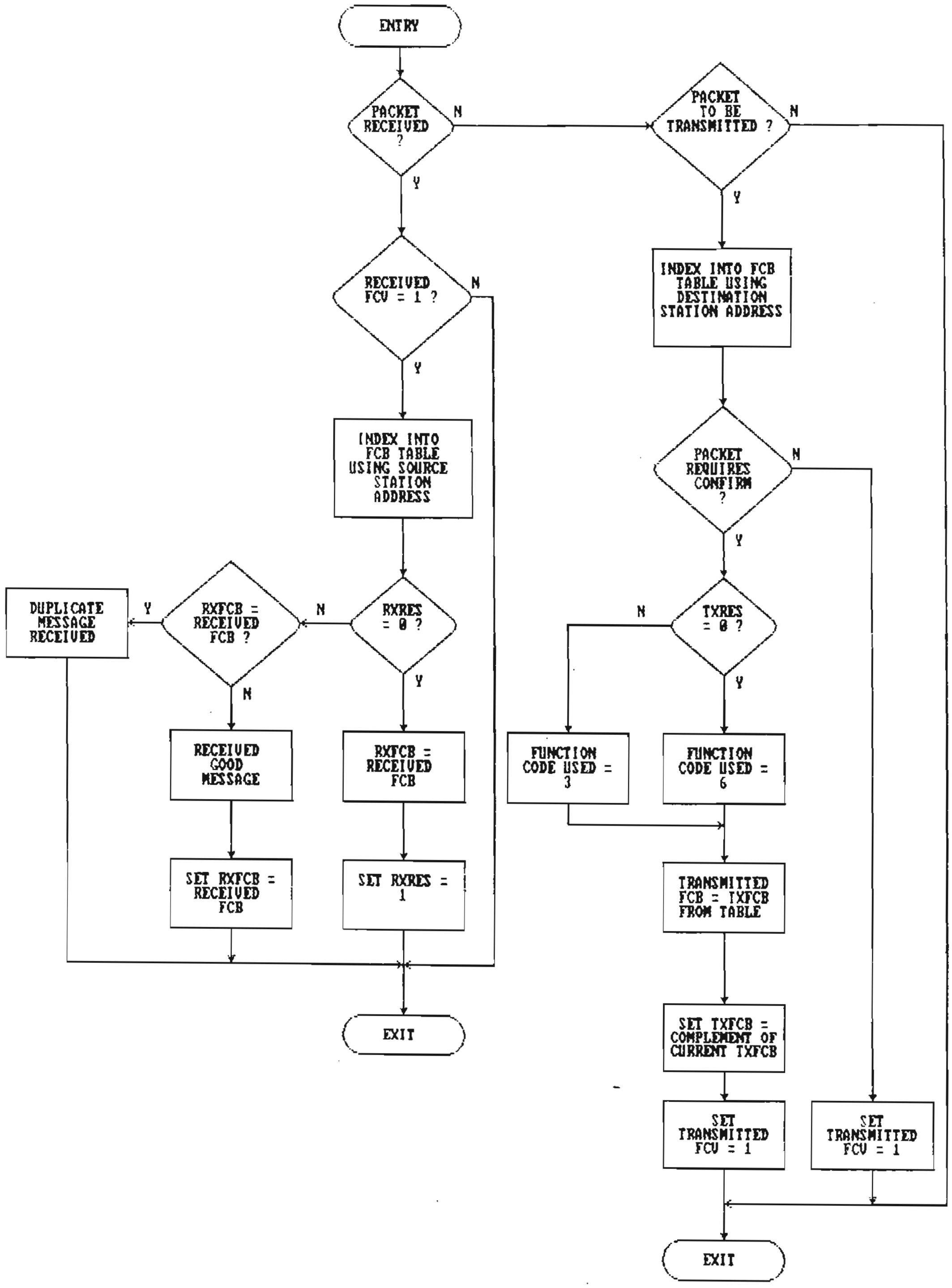


Figure 22. Frame Count Bit (FCB) Flow Chart.

3.2.3.2 **User Services**

The network layer has two basic services:

(a) allow routing of PDUs from higher layers and,

(b) provide store and forward facility for packets enroute to destination through one or more stations. This feature is commonly known as digipeating in a radio network environment.

The network layer will allow allocation of routing information for up to five destination stations and digipeating through a maximum of six stations. These services are accessed with the use of primitives as shown in Table 3.

Primitive	**Parameters**
DL_DATA.request	Destination Address (REQ)
.indication	Source Address (IND)
	Maximum Retransmission Count (REQ)
	Network layer service user data (PL-PDU)
	Priority (REQ)
	Confirmation Required (REQ)

Table 3. Network layer service primitives.

The 16 bit destination address is used with the request primitive and is used to specify the physical address of the destination station that the presentation layer PDU (PR-PDU) is to be delivered to.

The 16 bit source address is used with the indication primitive and is used to indicate the physical address of the originating station.

The definition of the maximum retransmission count, priority and confirmation required fields are explained in section 3.2.2.2.

Figure 23 is a time sequence diagram indicating the interaction between the various primitives.

3.2.3.3 Protocol Operation

Transmission to a peer network entity is initiated with the receipt of the NL_DATA.request primitive. The corresponding network layer entity will generate a PDU to be transmitted to the peer network layer or to an intermediate network layer.

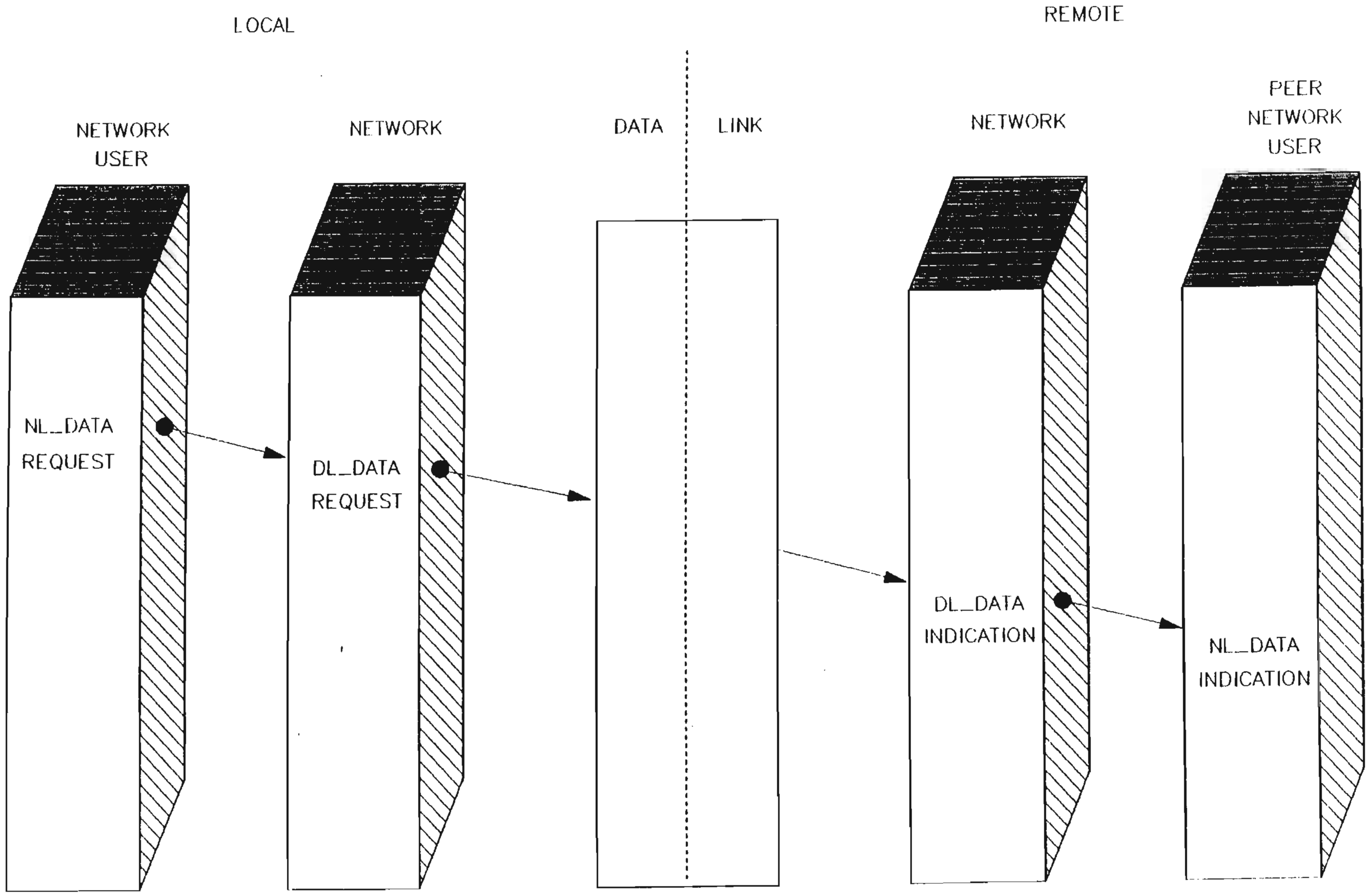


FIGURE : 23. INTERACTION OF PRIMATIVES DURING DATA TRANSFER BETWEEN PEER NETWORK LAYER USERS.

The format of the Network Layer PDU (NL-PDU) generated is given below:

Byte sequence	Fields
0	Count
1	Number of Digipeats (ND)
2	Source Address (MSB)
3	Source Address (LSB)
4	Digipeat Address 1 (MSB)
5	Digipeat Address 1 (LSB)
6	Digipeat Address 2 (MSB)
7	Digipeat Address 2 (LSB)
"	"
"	"
n-2	Digipeat Address m (MSB)
n-1	Digipeat Address m (LSB)
n = 4+(mx2)	Destination Address (MSB)
n+1	Destination Address (LSB)
n+2	PR-PDU Byte 1
n+3	PR-PDU Byte 2
"	"
"	"
n+1+t	PR-PDU Byte t

Fields:

Count : is the total number of bytes in the PDU excluding the Count Byte.

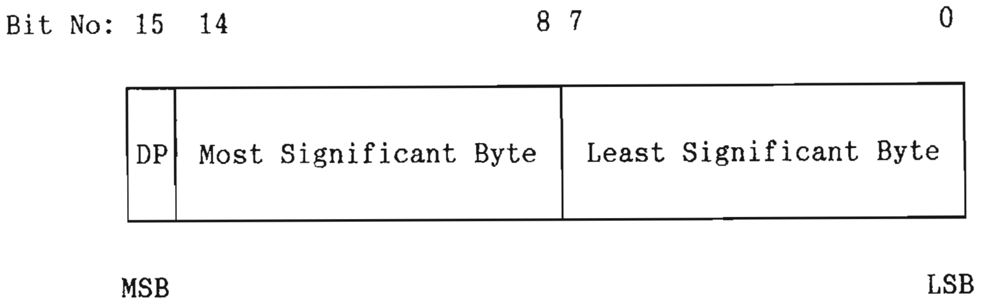
Source Address : is the physical sixteen bit address of the station initiating the message.

Destination Address : is the sixteen bit physical address of the destination station to which the PDU is to be finally delivered to. This is equivalent to the destination station address passed as a parameter in the request primitive.

Number of Digipeats : is the total number of times the PDU is to be stored and forwarded before reaching the destination ie. the total number of digipeating stations in the path of the PDU. If the PDU is not digipeated, then this field is set to zero and all fields from Source to Destination Address (inclusive) are omitted.

Digipeat Addresses : is the 16 bit address of each digipeat that the PDU has to traverse. A digipeat is a station that will behave as a store and forward repeater for the PDU.

Digipeat addresses take on the following format:



Bit number 15 is known as the digipeating bit (DP) and is set each time the PDU is stored and forwarded. The operation of this bit will be described in more detail below.

On receipt of the NL_DATA.request primitive, the network layer will use the destination address passed in, as a parameter, to check if a route has been allocated for that address.

Route allocations are stored in the NL and are programmed with an application process. Figure 24 shows the method of allocation of digipeat paths.

	ENTRY	DA	T	RL	DP1	DP2	DP3	DP4	DP5	DP6
SET BY APPLICATION	1		X							
		"	"	"	"	"	"	"	"	"
		"	"	"	"	"	"	"	"	"
	5		X							
SET BY NETWORK ENTITY	6									
		"	"	"	"	"	"	"	"	"
		"	"	"	"	"	"	"	"	"
	10									

X - not used

Figure 24. Network Addressing Table (NAT).

The network layers permits five route allocations by an application process and five allocations by the network layer itself. Each entry in the NAT is identified by its programmable sixteen bit Destination Address (DA). Associated with each entry is an eight bit Retry Length byte (RL). Bits 0 to 3 of RL is used to specify the number of digipeat addresses in the entry whilst bits 4 to 7 is used to specify the maximum number of retransmissions of the network PDU in the data link layer. The timer (T) is used only for network entries and is used to determine the age of a network entry. The NAT is capable of accepting up to six 16 bit digipeat addresses (DP1-6) for each entry.

The application process may overwrite entries 1 to 5 at will.

On receipt of the NL_DATA.request primitive, the network layer will search through entries 6 to 10 for a match with the destination address passed in with the primitive. If no match is found, then entries 1 to 5 is searched. In either case, if a match is found, the path in that entry is used in the generation of the NL-PDU. If a path is not found, then no path is added to the NL-PDU.

On receipt of the DL_DATA.indication primitive, the network layer will take the following actions on the NL-PDU received from the peer network entity:

(a) check the ND field. If it is 0, then the network layer will generate the respective PDU as shown under the presentation layer section. The generated PDU will then be passed to the presentation layer using the NL_DATA.indication primitive.

If the ND field is non-zero, then action (b) must be taken.

(b) Starting with the first digipeat address, check the DP bit (bit number 15) of each address. The first digipeat address with a DP bit value of 0 must be compared with the station address. If the digipeat address is equal to the station address, then the network layer must store and forward this PDU. The network layer will set the DP bit in the digipeat address to 1 and retransmit the PDU using the DL_DATA.request primitive.

If the digipeat address is not equal to the station address, then the PDU is discarded.

If all the digipeat address have been through without a zero DP bit, then action (c) must be taken.

(c) Check if the destination address is the station address. If the destination address in the PDU is equal to the station address, then the network layer will generate the respective PDU (as shown in the presentation layer) and will pass this up to the presentation layer using the NL_DATA.indication primitive. The source address used with the primitive is equal to the source address passed in with the DL_DATA.indication primitive.

If the destination address is not equal to the station address, then the PDU is discarded.

Figure 25 indicates pictorially the transfer of PDUs during digipeating.

The figure shows an example of a packet being digipeated through a two stations. The solid arrowed lines indicate the path that the packet takes when it is being digipeated. The broken arrowed lines indicate the communications between peer layers.

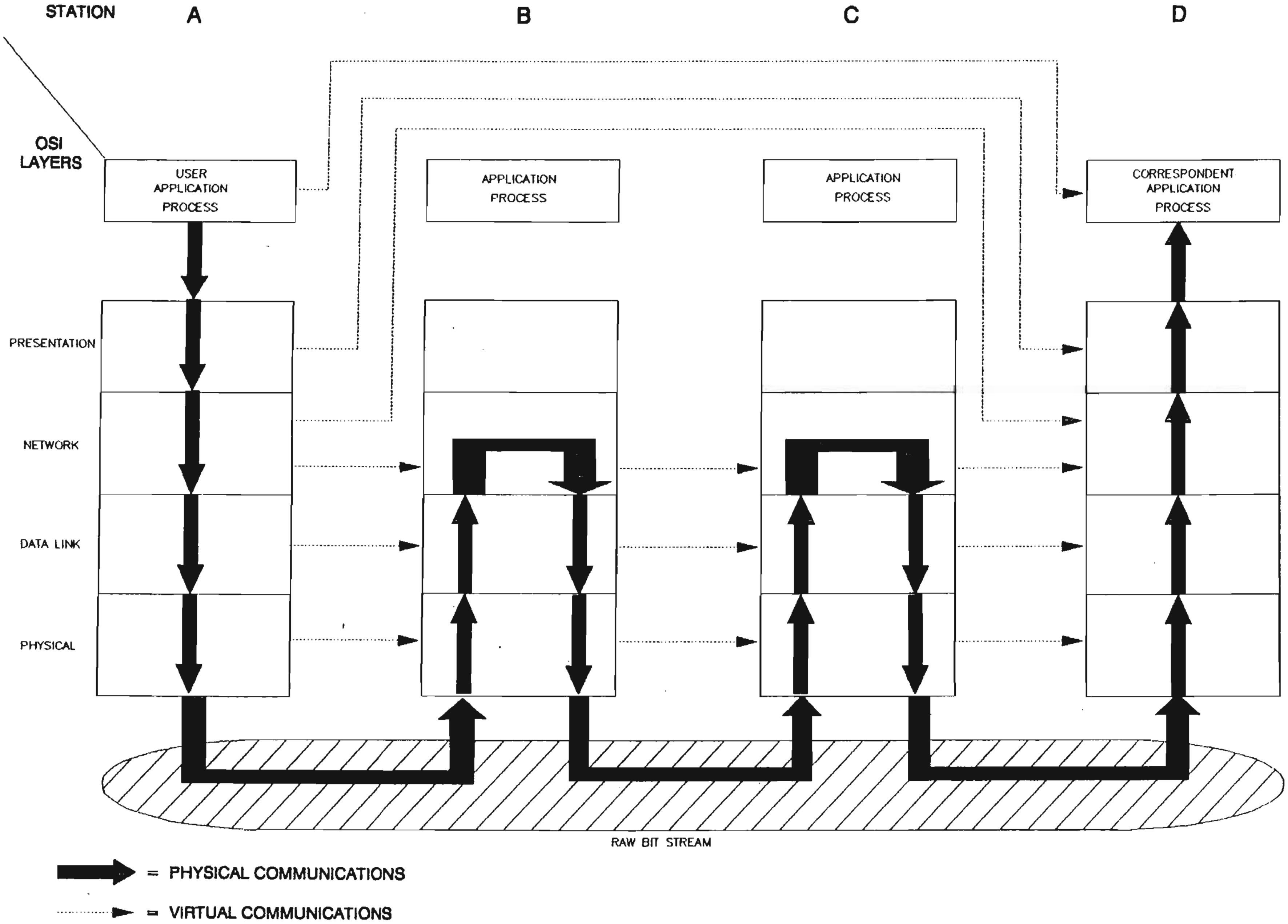


FIGURE : 25 LAYER INTERACTIONS DURING DIGIPEATING

3.2.3.4 Use of Data Link Services

The network layer uses the services of the Data Link Layer during the transfer of data between peer network entities. The primitives used are shown in Figure 17 and are tabulated in Table 2.

3.2.4 Presentation Layer

3.2.4.1 Overview

The presentation layer has as its main task the provision of a Data Interchange Table (DIT) and provides the required services to support this. The concept of the DIT is based on CONLOG's local area network, CONET, presentation layer.

The DIT is a set of sixteen bit read and/or write registers used for the interchange of data between heterogeneous application processes. The data in these registers may be read from or written to by a local application process or from m a remote application process.

In the presentation layer of the packet radio system, there are 256 DIT registers. Registers are numbered sequentially from 0 to 255. Registers 0 to 32 are reserved for use by the presentation layer.

3.2.4.2 **User Services**

The three services supported by the presentation layer are as follows:

(a) local DIT write/read,

(b) remote DIT write/read and,

(c) application messages.

The local DIT write/read service allows a local application process to write/read from the local DIT. The remote DIT write/read allows the DIT to be read from or written to by a remote application process.

The application messages service allows the application process to pass data and retrieve data from a remote application process, ie. the presentation layer simply acts as a pipe between the application processes and the NL for such messages.

Primitive	Parameters
PR_Local_DIT_READ.request	Register number
PR_Local_DIT_READ.confirm	Register value
PR_Local_DIT_WRITE.request	Register number Register value
PR_Remote_DIT_READ.request .confirm	Destination Address (REQ) Source Address (CON) Start Register number Range of registers to read Register values (CON)
PR_Remote_DIT_WRITE.request .confirm	Destination Address (REQ) Source Address (CON) Start Register number Range of registers to write Register values (REQ)
PR_APPDATA.request .indication	Destination Address (REQ) Source Address (IND) Application user data

Table 4. Table of parameters for the various primitives in the presentation layer.

The bracketed items in Table 4 indicate the primitive that is associated with the parameter. If no bracketed items are shown, the parameter is used by all respective primitives shown under the "Primitives" column.

Parameters:

Destination Address : is the physical address of station to which the message is sent. In Table 4 above, the destination address in a confirmation primitive is the same as the source address of the corresponding request primitive.

Source Address : is the physical address of the station initiating the request. In Table 4 above, the Source address in a confirmation primitive is the same as the destination address of the corresponding request primitive.

Register Number : is the number of the DIT register (16 bits) to be read from or written to. Although this is a 16 bit value, the presentation layer only supports register numbers from 0 to 255.

Register value(s) : is the 16 bit value(s) to be read from or written to the DIT.

Start Register Number : is the first DIT register (in a range of registers) to be read from or written to.

Range : is the range of DIT registers to read from or write to starting at the register number indicated by the "Start Register Number" above.

Figure 26 is a time sequence diagram representing the interaction between the primitives on both sides of the presentation layer. In this figure, the PR_APPDATA primitives have been omitted.

3.2.4.3 **Protocol Operation**

The presentation layer will generate a specific P-PDU when it receives a service primitive either from the adjacent higher or lower layer. The P-PDUs generated by the presentation layer and their formats are shown below. The PDUs below are shown in terms of bytes.

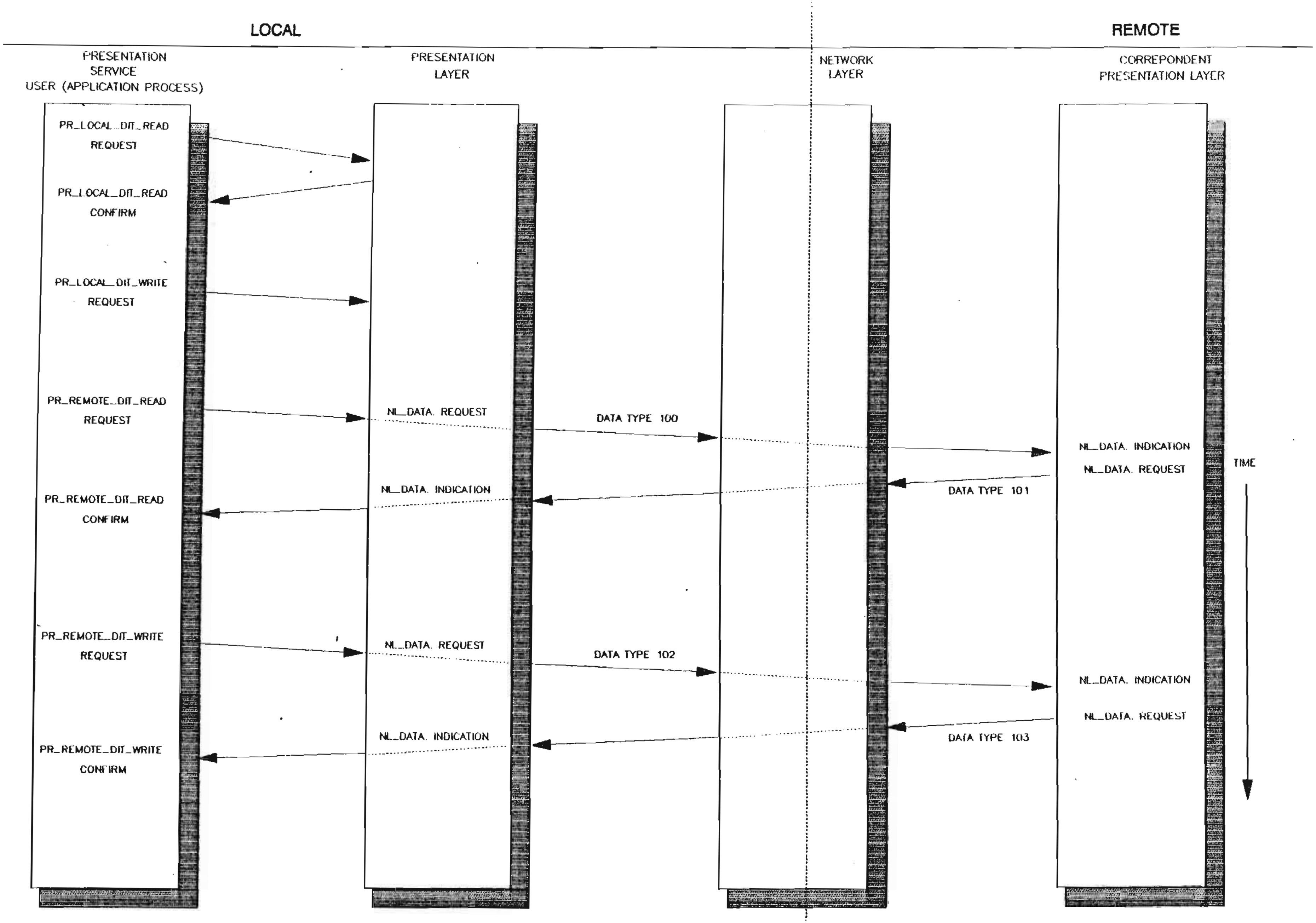
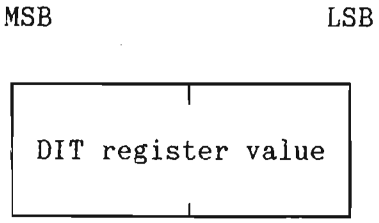

LOCAL
REMOTE
PRESENTATION SERVICE USER (APPLICATION PROCESS)
PRESENTATION LAYER
NETWORK LAYER
CORREPONDENT PRESENTATION LAYER
PR_LOCAL_DIT_READ REQUEST
PR_LOCAL_DIT_READ CONFIRM
PR_LOCAL_DIT_WRITE REQUEST
PR_REMOTE_DIT_READ REQUEST
NL_DATA. REQUEST
DATA TYPE 100
NL_DATA. INDICATION
NL_DATA. REQUEST
TIME
DATA TYPE 101
NL_DATA. INDICATION
PR_REMOTE_DIT_READ CONFIRM
PR_REMOTE_DIT_WRITE REQUEST
NL_DATA. REQUEST
DATA TYPE 102
NL_DATA. INDICATION
NL_DATA. REQUEST
DATA TYPE 103
NL_DATA. INDICATION
PR_REMOTE_DIT_WRITE CONFIRM

FIGURE : 26 TIME SEQUENCE DIAGRAM FOR INTRACTION OF PRIMITIVES IN PRESENTATION LAYER

(a) Local DIT read response

This two byte PDU contains the most significant and least significant bytes of the contents of a DIT register.



(b) Remote DIT read request

Byte sequence	Description	Value
0	COUNT	3
1	DATA TYPE	100
2	START REGISTER (MSB)	-
3	START REGISTER (LSB)	-
4	RANGE	-

(c) Remote DIT read reply

Byte sequence	Description	Value
0	COUNT	3
1	DATA TYPE	100
2	START REGISTER (MSB)	-
3	START REGISTER (LSB)	-
4	RANGE	- (m)
5	Register 1 (MSB)	-
6	Register 1 (LSB)	-
7	"	"
n-1	Register m (MSB)	-
n	Register m (LSB)	-

(d) Remote DIT write request

Byte sequence	Description	Value
0	COUNT	3 + No. of data bytes
1	DATA TYPE	102
2	START REGISTER (MSB)	-
3	START REGISTER (LSB)	-
4	RANGE	- (m)

5	Register 1 (MSB)	-
6	Register 1 (LSB)	-
7	"	"
n-1	Register m (MSB)	-
n	Register m (LSB)	-

(e) Remote DIT write reply

Byte sequence	Description	Value
0	COUNT	3
1	DATA TYPE	102
2	START REGISTER (MSB)	-
3	START REGISTER (LSB)	-
4	RANGE	- (m)

The START REGISTER and RANGE for (b) and (c) are equal. The same applies to (d) and (e).

Local DIT READ

On receipt of a Local_DIT_READ.request from the application processes, the presentation layer will read the respective register of the local DIT, generate a PDU as shown in (a) and pass this PDU to the higher layer using the Local_DIT_READ.confirm primitive.

Remote DIT READ/WRITE

The reading of a DIT register(s) in a remote presentation layer is initiated by the presentation layer (PL) receiving a Remote_DIT_READ/WRITE.request from the application process. The PL will generate the PDU shown in (b) and using the NL_DATA.request primitive to the layer below, pass the PDU to the remote PL.

The presentation layer in the remote station, on receipt of the NL_DATA.indication primitive from the network layer, will read the DIT register(s) required and generate the reply PDU as shown in (c)/(e). The DIT READ/WRITE reply PDU will be sent back to the initiating PL using the NL_DATA.request primitive to the network layer.

On receipt of the NL_DATA.indication primitive from the network layer in the local station, the presentation layer will pass the DIT READ/WRITE reply PDU to the respective application process using the PR_DIT_READ/WRITE.confirm primitive.

The receipt of a PR_APPDATA.request primitive will result in the PL simply transmitting the application PDU to the remote PL. The remote PL will announce the arrival of the PDU with a PR_APPDATA.indication primitive.

3.2.4.4 **Use of Network Services**

As can be observed from the time sequence diagram of Figure 26, that the local presentation layer makes use of the services of the network layer when communicating with the peer presentation layer. Table 3 shows the various primitives of the network layer that are used and their parameters.

3.3 **Hardware Design**

3.3.1 **Overview**

Of concern to the packet radio network design, is the RTU electronics. Figure 27 is a block diagram representing the components of the RTU electronics.

The RTU electronics basically consists of a power supply, central processing unit (CPU)/modem and two input/output modules. The modules are designed to interface electrically and mechanically to an in-house standard bus interface. All communications between the modules is done through the system bus (motherboard).

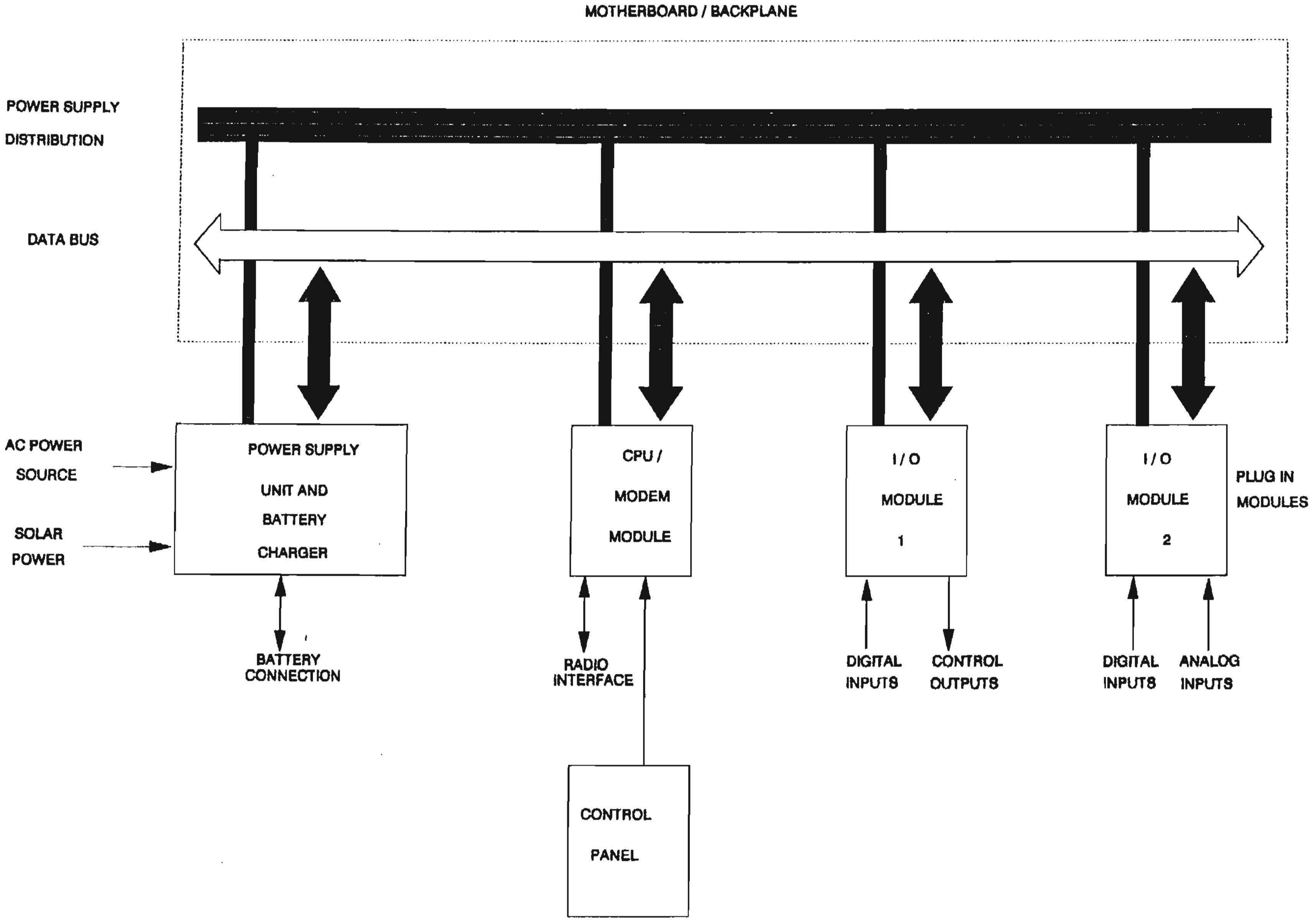


FIGURE 27 : RTU CONTROL MODULE ELECTRONICS

The power supply provides +5V DC and +12V DC on the motherboard. It also has the facility to provide battery backup and charging of the battery through the 220V ac mains input supply. The facility to accommodate a solar panel where mains is unavailable is also provided.

The CPU/Modem module, provide the intelligence required to make the RTU functional in the system. The CPU consists of an 8 bit microprocessor with 32k of ROM, 8 k of RAM, 128 bytes serial EEPROM and the necessary interface logic for the system bus and the modem. The modem provides the interface to the radio network. It employs Minimum Shift Keying (MSK) modulation to provide a narrow bandwidth modulating signal at 1200 baud, for transmission over a radio channel down to 12.5kHz channel spacing. It also uses standard E and M type signalling and allows the transmit signal level to be adjusted over a 25dB range. The receiver will also accept signals within a 25dB range.

The control panel provides the human interface in the RTU electronics. It consists of push-buttons to manually activate and deactivate control outputs and a rotary lockable switch to select the various modes of operation of the RTU.

The I/O modules provide the CPU with the necessary interface to the pole-mounted device to be controlled and various other transducers.

Of concern to this theses is the RTU CPU/Modem module design. The packet radio network software is based on this module. The following sections provide an overview of the design of this module. The sections to follow must be read in conjunction with the CPU and Modem circuit diagrams in APPENDIX B.

The microprocessor section of the RTU CPU/Modem module was designed by the author of this theses, whilst the modem section of the module was designed by a colleague, Mr Ian McNielage.

3.3.2 CPU Section

The microprocessor is the heart of the CPU module. It is used to address memory, run programs resident in EPROM, handle I/O and enable indicators to provide diagnostic indications.

The following paragraphs must be read in conjunction with the RTU CPU/Modem Board 1 circuit diagrams (sheets 1 and 2) found in Appendix B.

The microprocessor (CPU) used is an 8 bit CMOS 6303 microprocessor U4 operating at a bus speed of 1MHz. It operates in multiplexed mode. Address lines A0 to A7 are latched with U3.

The address decoding circuit associated with the CPU section (U7-A and U7-B) is used to select between the various types of I/O and memory. The address decoding is designed to select the following:

(a) 32/16k EPROM (U2)

(b) 2/8k RAM (U1)

(c) Slot Select lines $\overline{\text{SL1}}$ to $\overline{\text{SL8}}$ (U9)

(d) LED indicators (U10, U12)

(e) Asynchronous Communications Interface Adaptor (ACIA) on the modem board (via connector J4).

The EPROM and RAM sizes are link selectable using links LK1 and LK2 respectively.

The watchdog circuit consisting of U13 and U14 will toggle the RESET line of the CPU if the CPU is unable to reset the watchdog within one second. This circuit will enable the CPU to recover from a non-critical failure.

A power-on RESET signal is generated by U14.

U9, U11, U15 and U16 collectively form the bus interface circuitry to the system bus.

The LED indicators are memory mapped from the CPU and are turned on and off using latches U10 and U12.

An optional RS232 interface is provided through the port lines of the CPU and an RS232 line driver U5.

The port lines of the CPU are used to interface directly to a personality module (refer to the Personality Module circuit diagram in Appendix B). The personality module comprises of a 128 byte serial EEPROM used for storage of backed-up data.

3.3.3 Modem Section

The modem circuit consists of a serial interface adapter, a modem integrated circuit, transmit and receive analogue circuitry, E and M interface circuitry, and channel selection circuitry.

For the paragraphs to follow, refer to the RTU CPU/Modem Board 2 circuit diagrams (sheets 1 and 2) in Appendix B.

CPU/Modem Interface

An asynchronous communications interface adapter (ACIA) U1 is the interface between the microprocessor and the modem. Its functions include parallel-to-serial (from CPU to modem) and serial-to-parallel(from modem to CPU) conversion, addition and removal of framing bits, clocking of serial data and interfacing with the microprocessor. It interfaces to the modem IC via the transmit data/transmit clock and receive data/receive clock lines as well as via three control lines : RTS, CTS and DCD.

RTS is pulled low when U1 is instructed by the CPU. This causes the Modem IC, U2, to be put into transmit mode and also causes M line switching. DCD is a signal received from the modem IC to indicate data received.

MSK Modem circuit

U2 is the MSK modem IC. It converts data on its transmit data input (synchronized with the transmit clock) to MSK analog modulation on it transmit signal output. It also converts received MSK analog modulation on its receive signal input to data on its receive data output (synchronized with the receive clock). Transmit mode is enabled by the RTS signal from U1 and receive mode is enabled by the carrier detect signal derived from the E interface.

When a valid data carrier is detected the modem sets its carrier detect output high, which is connected to DCD on the ACIA (U1).

Transmit Analog circuitry

The transmit MSK signal is fed from the modem IC through U9-A and U9-B which permit the output level of the Modem card to be adjusted between its limits. This is achieved by adjusting VR1. T1 is a transformer which performs the task of matching the line impedance, converting to a balanced line and providing isolation from the line. D10 provides extra protection against line transients.

U10-D provides a low impedance dc reference for all the operational amplifiers in the transmit and receive circuits.

Receive Analog circuitry

Operational amplifiers U9-C, U10-A, U10-B, U10-C and U9-D perform the function of an automatic gain controlled amplifier so that the modem IC (U2) receives a constant, optimum signal level over a wide range of input levels from the line. T2 performs a balanced to unbalanced conversion, line impedance matching and line isolation. D11 provides additional protection against line transients.

U11 provides a loop-back function controllable by the microprocessor via the LOOP BACK signal. When this mode is selected, the transmitted signal is routed back into the modem IC and hence the microprocessor can compare the received data with the transmitted data to test the operation of the modem circuitry.

E & M Interface

U12 provides the interface to the E-line. When a dc potential is applied across the E interface the opto-isolator is enabled causing the modem IC (U2) to be enabled into receive mode. This line is also fed to the CPU as the CHANNEL BUSY signal.

Relay RLY1 provides the interface to the M-line. When transmit mode is selected and RTS is pulled low, the relay is enabled. This will cause the radio connected to the modem to go into transmit mode.

Channel Selection Circuitry

The channel select function of the modem is performed by U3 which is an eight bit latch and opto-isolators U7 and U8. U3 will hold the last data written to the channel select location on its outputs. A low on any output will turn the corresponding opto-isolator on. A high will turn it off. The radio will usually have a pullup resistor on each channel select line.

3.4 Software Design

3.4.1 Overview

The real-time software required to make each station in the packet radio network fully functional may be grouped as follows:

(a) Application process software and,

(b) Networking software.

The application process software is involved with the actual application tasks that the station is to perform in the system as a whole, and may change from system to system. In the system implemented, the application at each station allows it to behave as a data acquisition point as well as a controller for controlling various types of pole mounted devices.

The networking software deals entirely with the communications between the various stations in the network. It covers transmission and reception of frames between stations, contention in the medium, routing between stations etc. The networking software is designed to be fairly standard from system to system and may be tuned to a users need if it arises.

Figure 28 provides an overview of the software implemented in the RTU in the system designed. The application process software was written in the FORTH language. Although the application process software forms an integral part of the system software, it will not be discussed in this theses. However, the application software functional specification and design is included in APPENDIX D and E for reference.

The networking software was written in the 6303 microprocessor Assembler language and is based on the hardware, as discussed in section 3.3.2. The networking software basically implement the services and functions of the various layers of the protocol specification as discussed in section 3.2.

The detail design of the networking software would require may pages of documentation. The following sections attempts to provide an outline of the design of the networking software.

The design and implementation of the networking and application process software was carried out by the author of this theses.

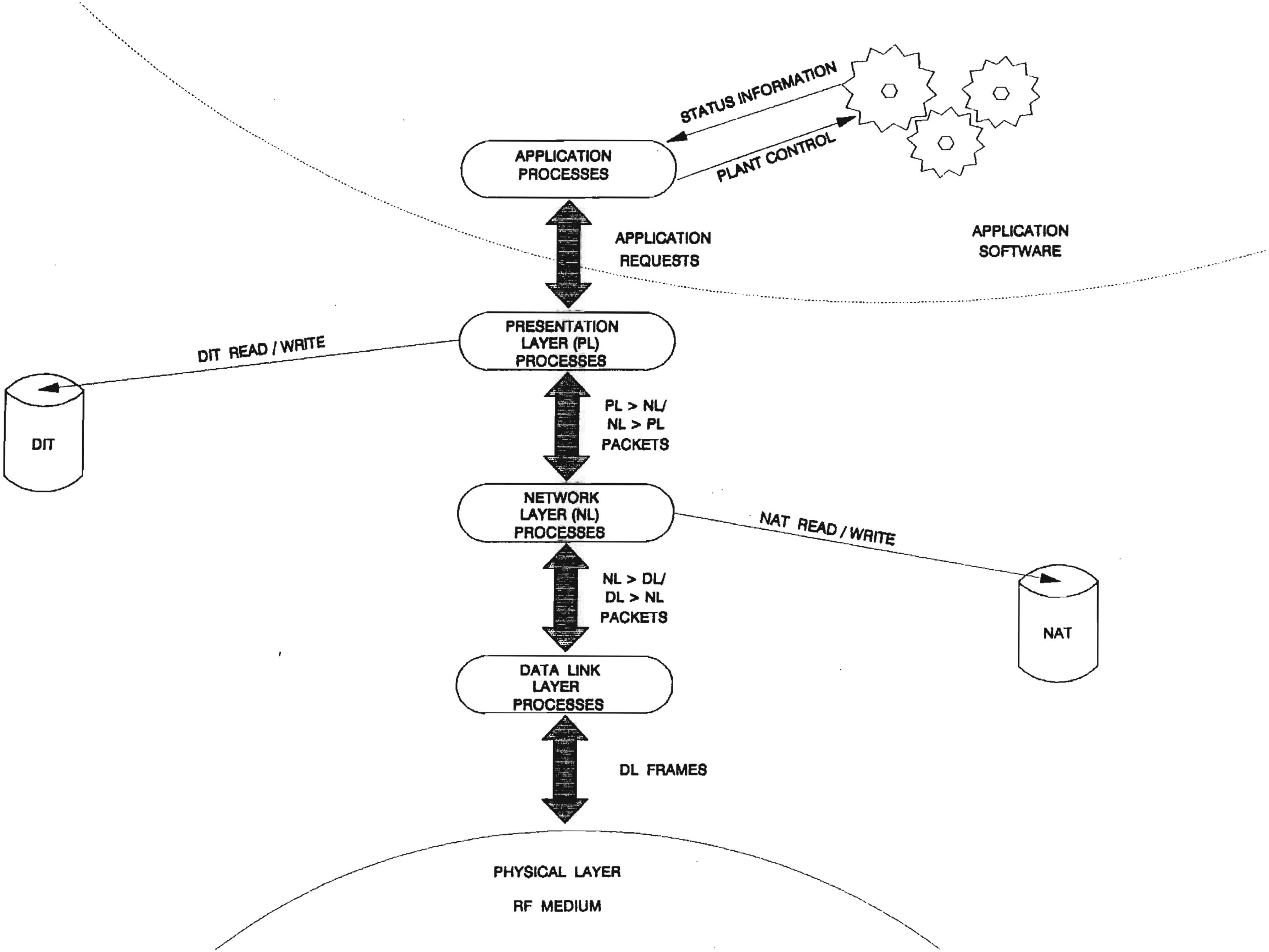


FIGURE : 28 RTU SOFTWARE OVERVIEW

3.4.2 Design Description

The basic requirement for the networking software is to implement the services and functions of the various layers of the protocol specification as discussed in section 3.2. The three protocol layers are each allocated a software module. Though the application processes are not discussed in this section, the interface between the application processes and the networking software will be given. Figure 29 is a detailed data flow diagram of these modules.

The FORTH kernel, which is off-the-shelf software, is written in FORTH and assembler and is used to run the application software. The FORTH kernel is proprietary software and is not discussed in this theses.

The application processes use FORTH calls to effect communications to and from the presentation layer. The FORTH calls trigger subroutines in the presentation layer that perform specific functions, eg. initiating a remote DIT write.

There are two sources of interrupts for the software. These are:

(a) a 10 millisecond timer interrupt and,

(b) a byte received or byte transmitted interrupt.

The first source of interrupt is generated internally to the microprocessor whilst the second source of interrupt is generated by the 6850 Asynchronous Communications Interface Adaptor (ACIA) in the

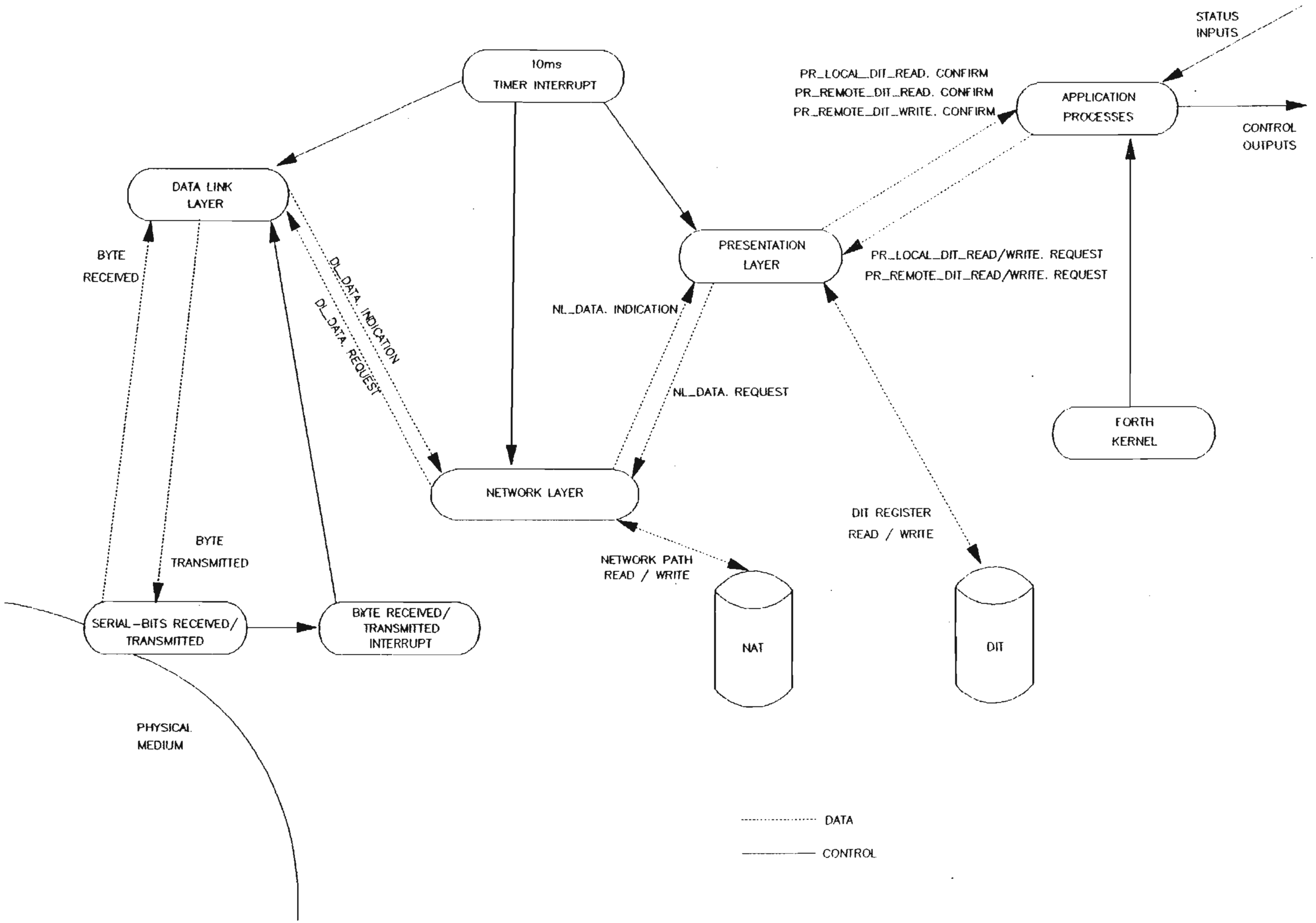

STATUS INPUTS
10ms TIMER INTERRUPT
PR_LOCAL_DIT_READ. CONFIRM
PR_REMOTE_DIT_READ. CONFIRM
PR_REMOTE_DIT_WRITE. CONFIRM
APPLICATION PROCESSES
CONTROL OUTPUTS
DATA LINK LAYER
PRESENTATION LAYER
PR_LOCAL_DIT_READ/WRITE. REQUEST
PR_REMOTE_DIT_READ/WRITE. REQUEST
BYTE RECEIVED
DL_DATA. INDICATION
DL_DATA. REQUEST
NL_DATA. INDICATION
NL_DATA. REQUEST
FORTH KERNEL
NETWORK LAYER
DIT REGISTER READ / WRITE
BYTE TRANSMITTED
NETWORK PATH READ / WRITE
SERIAL-BITS RECEIVED/ TRANSMITTED
BYTE RECEIVED/ TRANSMITTED INTERRUPT
NAT
DIT
PHYSICAL MEDIUM
DATA
CONTROL

FIGURE : 29 DETAILED RTU SOFTWARE DATA FLOW DIAGRAM

modem module. The receive byte interrupt occurs when the Receive Data Register (RDR) in the ACIA becomes full as a result of a byte being shifted in (RDRF interrupt). The transmit byte interrupt occurs as a result of the byte in the Transmit Data Register (TDR) being shifted out ie. emptied (TDRE interrupt).

The Data Link Layer (DLL), Network Layer(NL) and Presentation Layer (PL) modules are called every 10 milliseconds using the timer interrupt. Additionally, the Data Link module is called using the ACIA interrupts.

On power-up, all variables are initialized, the RDRF interrupt and the 10 ms timer interrupt is enabled, and the FORTH kernel is run. The FORTH kernel will run the application processes software continuously until interrupted by interrupts (a) or (b) above.

A layer is considered to be in transmission if it receives a message from a higher layer, processes it and (optionally) passes it to a lower layer. A layer is considered to be in reception if it receives a message from a lower layer, processes it and (optionally) passes it to a higher layer.

The transmission and reception of messages from layer to layer occur asynchronously and depend only on the availability of a service in a layer. This would indirectly depend on the availability of free buffer space in certain instances.

A byte received from the network will enable the Data Link module to start assembling a packet. The DLL module will process the packet and will either discard the packet, due to errors, or pass it to the NL module. The DLL module will also check for messages passed from the NL module. It will attach the necessary control information to the packet before transmitting the frame byte by byte onto the physical medium.

When the NL module is run, it will check for packets received from the DLL module. The NL module will distinguish between messages requiring digipeating or passing to the PL module. The NL module will also check for messages received from the PL module. It will perform the necessary routing of the message before passing it to the DLL module.

When the PL module is run, it will check for messages received from the NL module. It will distinguish between messages that need to be passed to the application processes or messages that require DIT processing. Messages that require DIT processing cause replies to be generated that are passed back to the NL module. The PL module will also check for application process messages that require transmission.

Each module interfaces with the adjacent module (as seen in Figure 29) with the use of primitives. Each primitive comprises of a semaphore register, relevant parameters and a data pointer. The data pointer is used to pass the relevant PDU between the modules.

Due to the restricted RAM area on the CPU, each module uses a "pool buffer" for storage of PDUs. There are 4 pool buffers (256 bytes per buffer) for use by the modules. Each buffer in the pool has the following format:

Byte Index Number	Description
0	Pool Buffer Number (0 - 3), bit 8 = Taken flag
1	Offset to "Length of Data" - (2+n)
2	no data
3	no data
4	no data
5	Length of data: m
6	Address - most significant byte
7	Address - least significant byte
8	Data byte 1
9	Data byte 2
"	"
"	"
4+n+m	Data byte m

The "Pool Buffer Number" is used by various subroutines to index into these buffers. The "Taken" flag is used to book a pool buffer. If it is set to 1, then the buffer may not be used by any module. A free buffer will have its "Taken" flag set to 0. The "Offset to Length of Data" field holds the offset to the start of data from the start of the pool buffer. The Length of Data field indicate the number of data bytes in the buffer.

The following sections provide an outline of each module and how each module interacts with the other.

3.4.3 **Modules**

3.4.3.1 **Data Link Layer Module**

The DLL module is the largest of the three modules and is run using five state machines. These are:

DLTMs_P0 : transmission of priority zero (P0) messages from the DLL onto the physical medium.

DLTMs_P1 : transmission of priority one (P1) messages from the DLL onto the physical medium.

DLTAs : transmission of acknowledgements (confirmations) from the DLL onto the physical medium.

DLTIs : transmission of bytes from the DLL onto the physical medium using the TDRE interrupt.

DLRIs : reception of bytes from the physical medium to the DLL using the RDRF interrupt.

where "s" in the acronyms represent the current state in the state machine.

DLTMs P0/P1 and DLTAs state machines

Figure 30 is a state transition diagram for the three state machines. The diagram also shows the relationship between the three state machines as well.

In the Idle State, the software continuously checks for transmission of acknowledgements, arrival of P1 messages from the NL module for transmission and arrival of P0 messages from the NL module for transmission, respectively.

With the arrival of a P1 message from the network layer module, the DLTM0_P1 state is entered. In this state, the software will wait for the radio channel to become free. This is achieved by monitoring the CHANNEL BUSY signal from the modem. The state machine will advance to DLTM1_P1 when the channel becomes free. The software will delay a time T1 before advancing to state DLTM2_P1. If during state DLTM1_P1 the channel becomes busy, the state machine is advanced to state DLTM0_P1.

T1 is a multiple of a PTT delay and is calculated such that the persistency of the protocol is 0.1. At the end of a T1 wait, with the channel being free, the state machine is advanced to DLTM2_P1. The PTT line to the radio is enable, putting the radio into transmit mode. In this state, the software waits for the PTT delay. When the PTT

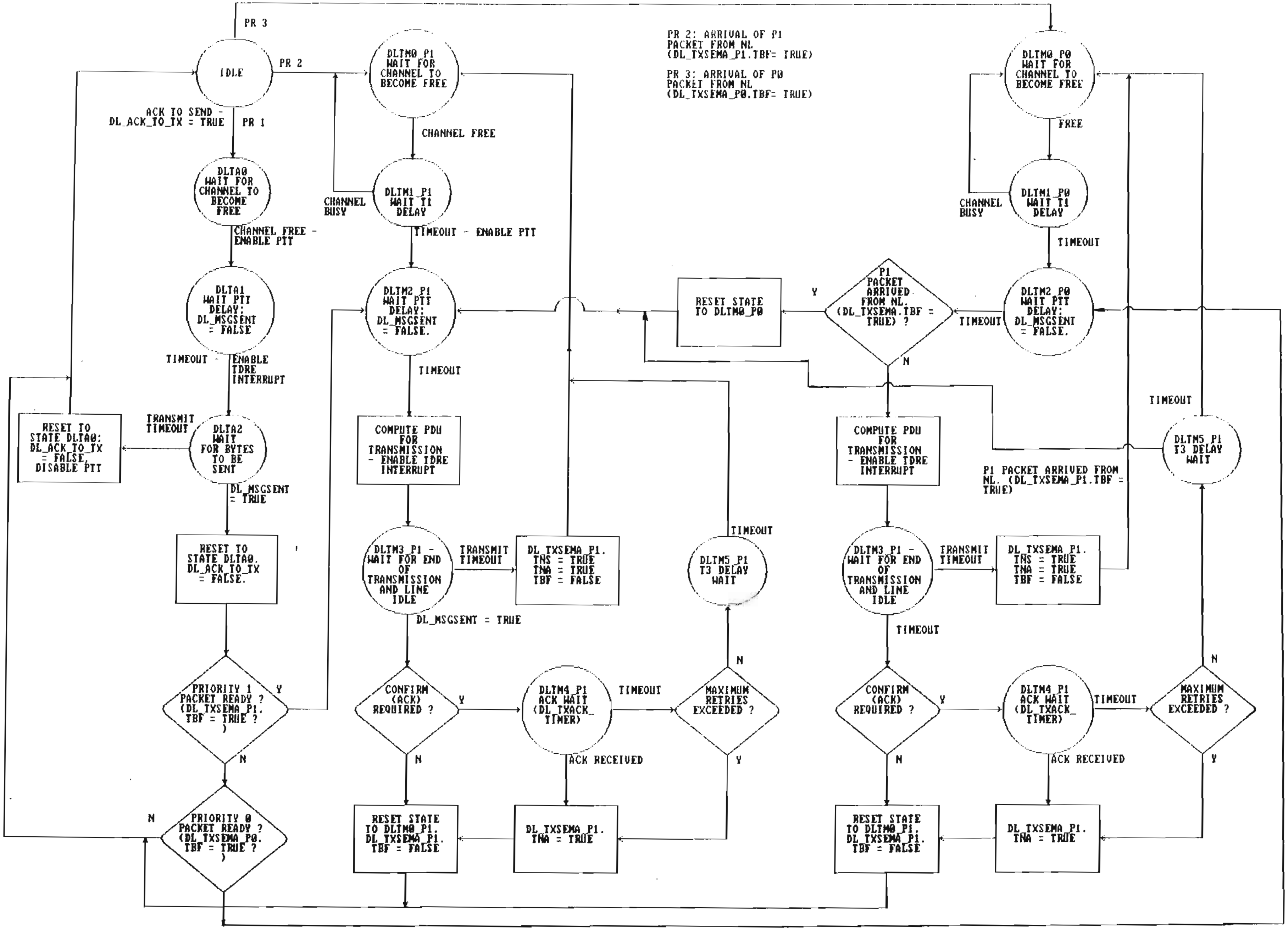

PR 3
IDLE
PR 2
PR 1
ACK TO SEND - DL_ACK_TO_TX = TRUE
DLTM0_P1 WAIT FOR CHANNEL TO BECOME FREE
CHANNEL FREE
PR 2: ARRIVAL OF P1 PACKET FROM NL (DL_TXSEMA_P1.TBF= TRUE)
PR 3: ARRIVAL OF P0 PACKET FROM NL (DL_TXSEMA_P0.TBF= TRUE)
DLTM0_P0 WAIT FOR CHANNEL TO BECOME FREE
FREE
DLTA0 WAIT FOR CHANNEL TO BECOME FREE
CHANNEL FREE - ENABLE PTT
CHANNEL BUSY
DLTM1_P1 WAIT T1 DELAY
TIMEOUT - ENABLE PTT
CHANNEL BUSY
DLTM1_P0 WAIT T1 DELAY
TIMEOUT
DLTA1 WAIT PTT DELAY: DL_MSGSENT = FALSE
TIMEOUT - ENABLE TDRE INTERRUPT
DLTM2_P1 WAIT PTT DELAY: DL_MSGSENT = FALSE.
RESET STATE TO DLTM0_P0
Y
P1 PACKET ARRIVED FROM NL. (DL_TXSEMA.TBF = TRUE) ?
N
TIMEOUT
DLTM2_P0 WAIT PTT DELAY: DL_MSGSENT = FALSE.
TIMEOUT
TIMEOUT
RESET TO STATE DLTA0: DL_ACK_TO_TX = FALSE, DISABLE PTT
TRANSMIT TIMEOUT
DLTA2 WAIT FOR BYTES TO BE SENT
COMPUTE PDU FOR TRANSMISSION - ENABLE TDRE INTERRUPT
COMPUTE PDU FOR TRANSMISSION - ENABLE TDRE INTERRUPT
DLTM5_P1 T3 DELAY WAIT
P1 PACKET ARRIVED FROM NL. (DL_TXSEMA_P1.TBF = TRUE)
DL_MSGSENT = TRUE
RESET TO STATE DLTA0. DL_ACK_TO_TX = FALSE.
DLTM3_P1 - WAIT FOR END OF TRANSMISSION AND LINE IDLE
TRANSMIT TIMEOUT
DL_TXSEMA_P1. TNS = TRUE TNA = TRUE TBF = FALSE
TIMEOUT
DLTM5_P1 T3 DELAY WAIT
DLTM3_P1 - WAIT FOR END OF TRANSMISSION AND LINE IDLE
TRANSMIT TIMEOUT
DL_TXSEMA_P1. TNS = TRUE TNA = TRUE TBF = FALSE
DL_MSGSENT = TRUE
TIMEOUT
PRIORITY 1 PACKET READY ? (DL_TXSEMA_P1. TBF = TRUE ?)
Y
N
CONFIRM (ACK) REQUIRED ?
Y
DLTM4_P1 ACK WAIT (DL_TXACK_ TIMER)
TIMEOUT
MAXIMUM RETRIES EXCEEDED ?
N
CONFIRM (ACK) REQUIRED ?
Y
DLTM4_P1 ACK WAIT (DL_TXACK_ TIMER)
TIMEOUT
MAXIMUM RETRIES EXCEEDED ?
N
N
ACK RECEIVED
Y
N
ACK RECEIVED
Y
N
PRIORITY 0 PACKET READY ? (DL_TXSEMA_P0. TBF = TRUE ?)
RESET STATE TO DLTM0_P1. DL_TXSEMA_P1. TBF = FALSE
DL_TXSEMA_P1. TNA = TRUE
RESET STATE TO DLTM0_P1. DL_TXSEMA_P1. TBF = FALSE
DL_TXSEMA_P1. TNA = TRUE

Figure 30. Data Link Layer State Transition diagram for transmission of acknowledgement frames, priority 0 and priority 1 messages.

delay times out, the DL_MSGSENT flag is set to false and the TDRE interrupt is enabled. This interrupt will wake up the Transmit Byte state machine (DLTIs) which will transmit the bytes in the frame. The DLTMs_P1 state machine is advanced to the DLTM3_P1 state. In this state the software waits for the end of transmission.

The DL_MSGSENT flag will be set by the DLTIs state machine to indicate the end of transmission. This flag being set will cause the DLTMs_P1 state machine to either advance to state DLTM4_P1 if an acknowledgement is required or the IDLE state otherwise. In either case the PTT line to the radio is disabled, removing it from the transmit mode. If no acknowledgement is received during the acknowledgement wait, the state machine is advanced to DLTM5_P1. This state is entered only if the maximum number of retransmissions have not been exceeded. In this state, the timer T3 is delayed for.

T3 is calculated based on the exponential binary back-off rule as discussed in section 2.3.6. At the end of the T3 timeout, state DLTM0_P1 is re-entered and the whole sequence is repeated.

The state machine for P1 messages is duplicated for priority 0 (P0) messages (DLTMs_P0). The only exceptions being that if a P1 message arrives in state DLTM2_P0 or DLTM5_P0, then the P1 message is processed. The P0 message will be processed once all P1 messages have been processed.

When an acknowledgement message becomes ready to send (with the DL_ACK_TO_TX flag being set to true in the DLRI state machine) the state machine advances to DLTA0. In this state the software waits for the transmission channel to become free. If the channel is not busy, the state machine advances to DLTA1. The PTT line of the radio is enabled, putting the radio into transmit mode. Also the DL_MSGSENT flag is cleared. In this state, the software will wait for the PTT timeout to expire before advancing to state DLTA2. On transition, the TDRE interrupt is enabled, causing the byte transmission state machine DLTIs to awake. This state machine is discussed later. At the end of transmission of all bytes, the DL_MSGSENT flag is set in the DLTIs state machine. This results in the acknowledgement state machine to advance to one of three states:

(a) DLTM2_P1 if a priority 1 message has arrived or is available for transmission or,

(b) DLTM2_P0 if a priority 0 message has arrived or is available for transmission or,

(c) IDLE if none of the above occurs.

(a) and (b) results in the "piggybacking" of messages with acknowledgements. With (a) and (b), the state machine advances as discussed for the cases when a P1 or P0 message arrives in the idle state, with the exception of no PTT and T1 delays. With the state machine advancing to the idle state, the PTT signal is disabled, removing the transmit mode of the radio.

These state machines are implemented by the subroutine DLTIRQ.

DLTIs state machine

This state machine , as shown in Figure 31, is responsible for the byte by byte transmission of a frame. It is initially in an IDLE state waiting for a TDRE interrupt to occur. Figure 31 indicates clearly the transition between states with the occurrence of TDRE interrupts. At the end of transmission, the DL_MSGSENT flag is set to true and the TDRE interrupt is disabled. This results in the state machine advancing to the IDLE state. If a transmit timeout occurs during any state, the state machine will abort the transmission of the message.

This state machine is implemented by the subroutine PROC_TXMSG.

DLRIs state machine

This state machine is responsible for assembling the bytes received from the network into a packet. Figure 32 is a state transition diagram of the DLRIs state machine.

The state machine is initially in the DLRI0 state. The states are advanced each time an RDRF interrupt occurs. In states 7 to 9, an error condition may cause the state machine to advance to state DLRI3

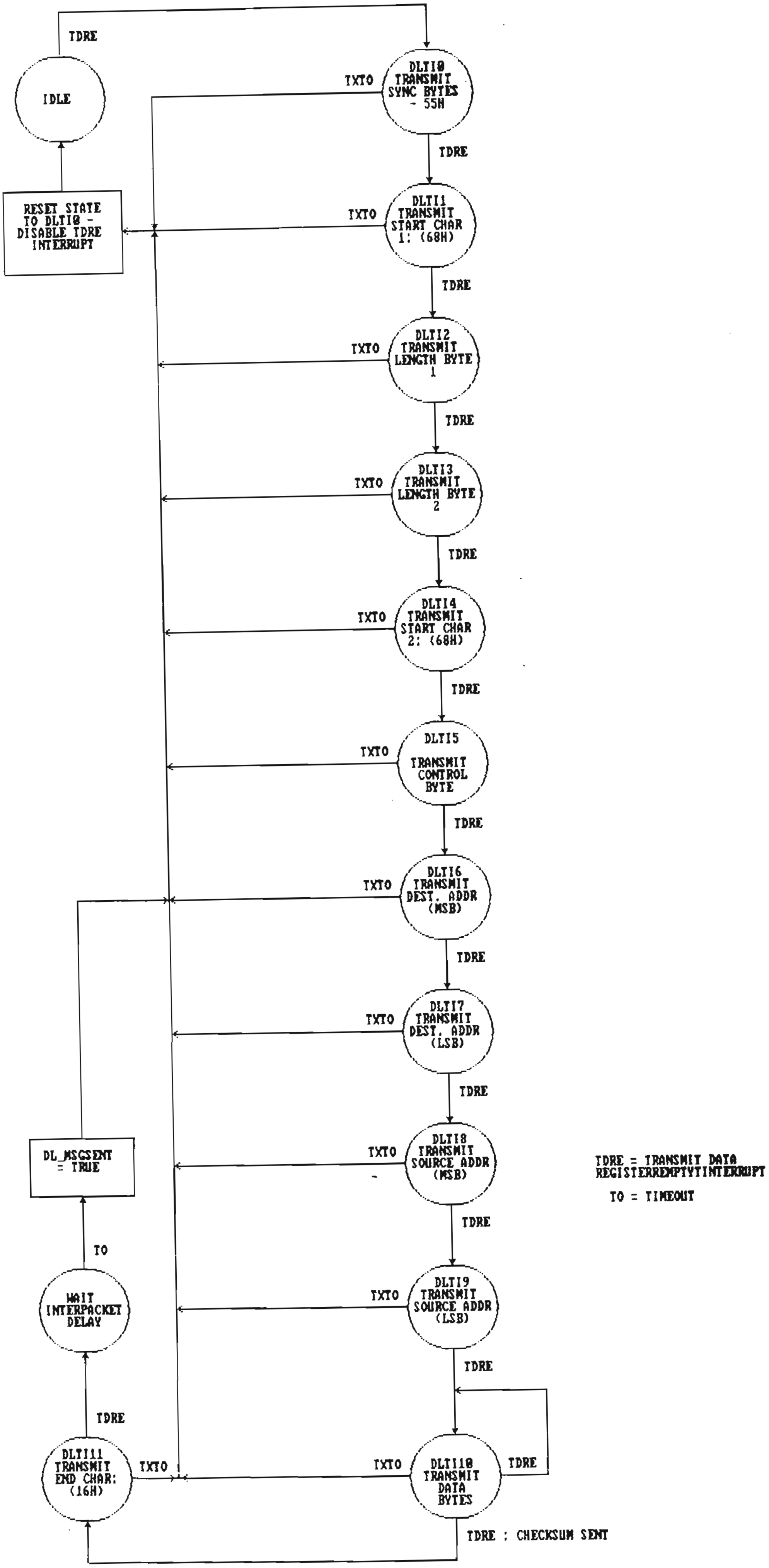

TDRE
IDLE
RESET STATE TO DLTI0 - DISABLE TDRE INTERRUPT
TXTO
DLTI0 TRANSMIT SYNC BYTES - 55H
TDRE
TXTO
DLTI1 TRANSMIT START CHAR 1: (68H)
TDRE
TXTO
DLTI2 TRANSMIT LENGTH BYTE 1
TDRE
TXTO
DLTI3 TRANSMIT LENGTH BYTE 2
TDRE
TXTO
DLTI4 TRANSMIT START CHAR 2: (68H)
TDRE
TXTO
DLTI5 TRANSMIT CONTROL BYTE
TDRE
TXTO
DLTI6 TRANSMIT DEST. ADDR (MSB)
TDRE
TXTO
DLTI7 TRANSMIT DEST. ADDR (LSB)
TDRE
TXTO
DLTI8 TRANSMIT SOURCE ADDR (MSB)
TDRE
TXTO
DLTI9 TRANSMIT SOURCE ADDR (LSB)
TDRE
DL_MSGSENT = TRUE
TO
WAIT INTERPACKET DELAY
TDRE
DLTI11 TRANSMIT END CHAR: (16H)
TXTO
TXTO
DLTI10 TRANSMIT DATA BYTES
TDRE
TDRE : CHECKSUM SENT
TDRE = TRANSMIT DATA REGISTERREMPTYTINTERRUPT
TO = TIMEOUT

Figure 31. Data Link Layer State Transition diagram for transmission of bytes using the TDRE interrupt.

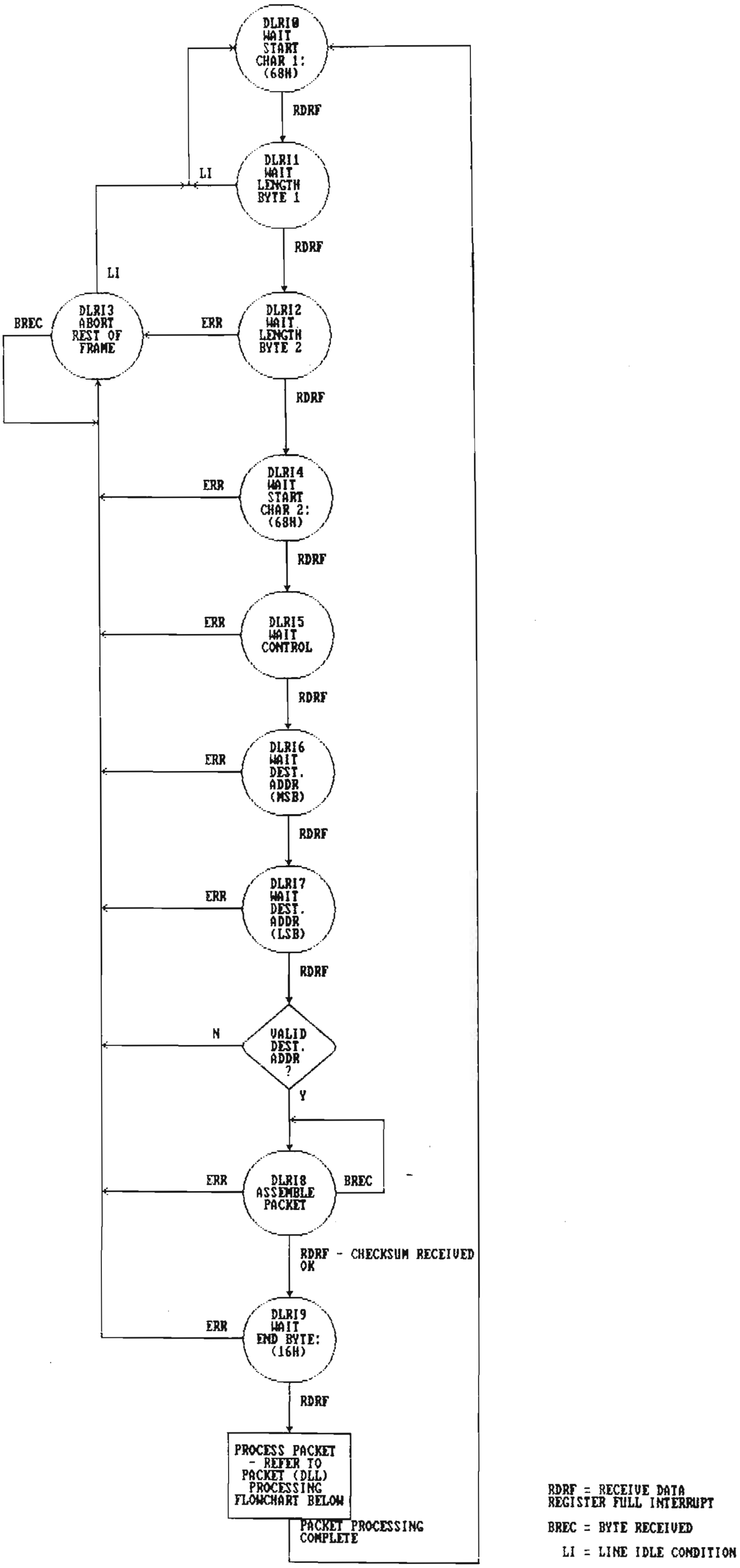

DLRI0 WAIT START CHAR 1: (68H)
RDRF
DLRI1 WAIT LENGTH BYTE 1
LI
RDRF
LI
DLRI3 ABORT REST OF FRAME
BREC
ERR
DLRI2 WAIT LENGTH BYTE 2
RDRF
ERR
DLRI4 WAIT START CHAR 2: (68H)
RDRF
ERR
DLRI5 WAIT CONTROL
RDRF
ERR
DLRI6 WAIT DEST. ADDR (MSB)
RDRF
ERR
DLRI7 WAIT DEST. ADDR (LSB)
RDRF
N
VALID DEST. ADDR ?
Y
ERR
DLRI8 ASSEMBLE PACKET
BREC
RDRF - CHECKSUM RECEIVED OK
ERR
DLRI9 WAIT END BYTE: (16H)
RDRF
PROCESS PACKET - REFER TO PACKET (DLL) PROCESSING FLOWCHART BELOW
PACKET PROCESSING COMPLETE
RDRF = RECEIVE DATA REGISTER FULL INTERRUPT
BREC = BYTE RECEIVED
LI = LINE IDLE CONDITION

Figure 32. Data Link Layer State Transition diagram for reception of a frame using the RDRF interrupt.

which will wait for a line idle timeout. Any bytes received in this state will be ignored. An error condition may result due to parity error, overrun, invalid byte, line idle timeout expired etc.

At the end of receipt of a good message, it is processed as shown in the flowchart of Figure 33. The software implemented for the processing of the packet is based on the IEC [1988] standard.

This state machine is implemented by the subroutine PROC_RXMSG.

Interface between Data Link Layer and Network Layer is as follows:

Transmission of Priority 1 messages (from the NL to DLL):

DL_TXSEMA_P1 -

is a semaphore register used to control transfer of data from the NL module to the DLL module. It has the following format:

Bit no:	7	6 5 4	3	2	1	0
	DIR	RESERVED	TNS	TNA	TAR	TBF

DIR : is the DIRection bit and is set by the NL. This bit determines directly the value of the direction bit in the control field of the DLL frame.

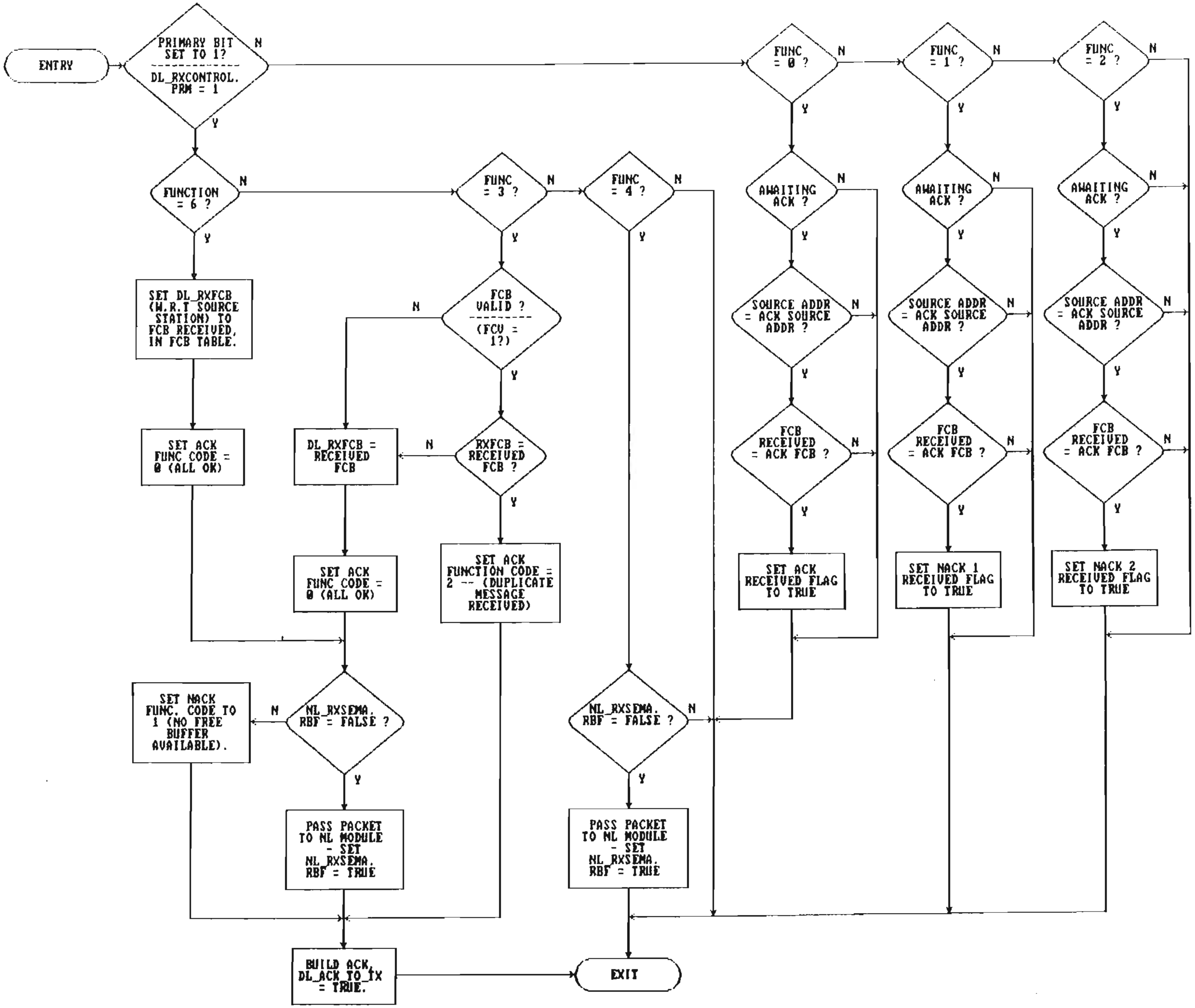

ENTRY
PRIMARY BIT SET TO 1?
DL_RXCONTROL. PRM = 1
N
Y
FUNCTION = 6 ?
N
Y
SET DL_RXFCB (W.R.T SOURCE STATION) TO FCB RECEIVED, IN FCB TABLE.
SET ACK FUNC CODE = 0 (ALL OK)
FUNC = 3 ?
N
Y
FCB VALID ?
(FCV = 1?)
N
Y
RXFCB = RECEIVED FCB ?
N
Y
DL_RXFCB = RECEIVED FCB
SET ACK FUNC CODE = 0 (ALL OK)
SET ACK FUNCTION CODE = 2 -- (DUPLICATE MESSAGE RECEIVED)
NL_RXSEMA. RBF = FALSE ?
N
Y
SET NACK FUNC. CODE TO 1 (NO FREE BUFFER AVAILABLE).
PASS PACKET TO NL MODULE - SET NL_RXSEMA. RBF = TRUE
BUILD ACK, DL_ACK_TO_TX = TRUE.
EXIT
FUNC = 4 ?
N
Y
NL_RXSEMA. RBF = FALSE ?
N
Y
PASS PACKET TO NL MODULE - SET NL_RXSEMA. RBF = TRUE
FUNC = 0 ?
N
Y
AWAITING ACK ?
N
Y
SOURCE ADDR = ACK SOURCE ADDR ?
N
Y
FCB RECEIVED = ACK FCB ?
N
Y
SET ACK RECEIVED FLAG TO TRUE
FUNC = 1 ?
N
Y
AWAITING ACK ?
N
Y
SOURCE ADDR = ACK SOURCE ADDR ?
N
Y
FCB RECEIVED = ACK FCB ?
N
Y
SET NACK 1 RECEIVED FLAG TO TRUE
FUNC = 2 ?
N
Y
AWAITING ACK ?
N
Y
SOURCE ADDR = ACK SOURCE ADDR ?
N
Y
FCB RECEIVED = ACK FCB ?
N
Y
SET NACK 2 RECEIVED FLAG TO TRUE

Figure 33. Data Link Layer Received packet processing flowchart.

TBF : is the Transmit Buffer Full flag and is set by the NL when it is passing a packet to the DLL. The DLL checks this flag to determine whether a packet has arrived for transmission or not. This is the last flag to be set when a packet is being passed from the NL. On completion of a transmission, either successfully or not, the DLL will set the TBF flag to 0.

TAR : is the Transmit Acknowledge Required flag. This flag will be set by the NL module to 1 if a confirmation is required and to 0 if no confirmation is required.

TNS : is the Transmitted message Not Sent flag. This is cleared by the NL when used. It is set to 1 by the DLL when the DLL is unable to transmit the message ie. the frame has not left the station. This flag should be checked by the NL only when the TBF flag is set to 0 by the DLL.

TNA : is the Transmitted message Not Acknowledged flag. This flag is set to 1 by the DLL when an acknowledgement to the transmitted message was not received. The NL will clear this flag prior to usage. This flag is checked by the NL only once the TBF flag is cleared by the DLL.

DL_TXBUFPTR_P1 -

is a double byte variable that holds the address of the pool buffer used by the NL module.

DL_MAXRETRY_P1 -

is a single byte variable that holds the maximum number of retransmissions that the message should undergo. This variable is set by the NL module.

Transmission of Priority 0 messages (from the NL to the DLL):

The following variables make up the interface between the NL and DLL modules for transmission of priority 0 (P0) messages.

DL_TXSEMA_P0	- single byte
DL_TXBUFPTR_P0	- double byte
DL_MAXRETRY_P0	- single byte

The function of these is the same as that of the priority 1 messages.

Reception of messages (between the DLL and NL):

The DLL module announces the arrival of messages to the NL module with the use of the following variables -

NL_RXSEMA -

is a semaphore register used to control transfer of data from the DLL module to the NL module. It has the following format:

Bit no:	7	6	5	4	3	2	1	0
	DIR							RBF

DIR : is the DIRection bit and is set by the DLL module. The value of this bit corresponds directly to the value of the DIR bit in the control field of the DLL frame.

RBF : is the Receive Buffer Full flag and is set by the DLL when it is passing a packet to the NL. The NL checks this flag to determine if a message has arrived for it from the data link layer.

NL_RXBUFPTR -

is a double byte variable that holds the address of the pool buffer used for transfer of the PDU from the DLL to the NL.

All interrupt handling tasks are serviced by the interrupt vector routine MIRQ. MIRQ is responsible for arbitrating interrupts generated as a result of a byte received or transmitted.

3.4.3.2 **Network Layer Module**

The NLL module is made up of two parts; a transmit and receive section.

Figure 34 is a flowchart indicating the sequence of events occurring during the receipt of a message from the DLL module.

The NL module constantly checks for messages arriving from the DLL module. A message is received when the RBF flag in the NL_RXSEMA register is set to 1. The received message is checked for a digipeat path. If one does not exist, then the message is passed to the PL module. If a digipeat path exists, then the digipeat path is checked as per the protocol specification for the NL. A digipeat message not being retransmitted is queued locally until the DLL is able to accept new messages.

Figure 35 is a flowchart illustrating the flow of events when a message is received from the presentation layer for transmission on the network.

The NL will check for messages just sent. If a P0 or a P1 message has just been sent, it will monitor the TBF flag in the respective DL_TXSEMA register. The TBF flag being cleared indicates end of transmission and the NL will transfer the status information in the respective DL_TXSEMA register to the respective NL_TXSEMA register.

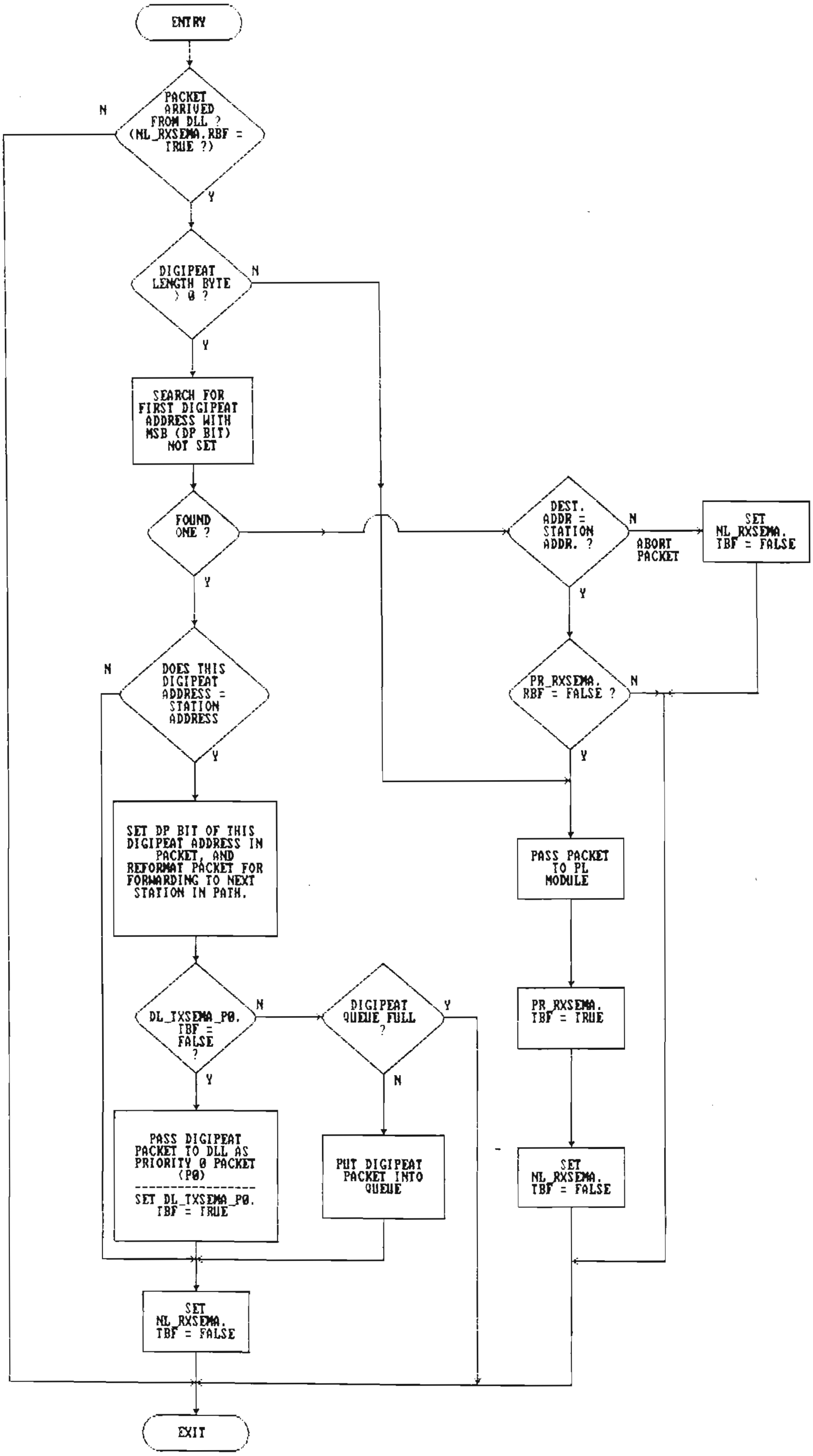

ENTRY
PACKET ARRIVED FROM DLL ? (NL_RXSEMA.RBF = TRUE ?)
N
Y
DIGIPEAT LENGTH BYTE > 0 ?
N
Y
SEARCH FOR FIRST DIGIPEAT ADDRESS WITH MSB (DP BIT) NOT SET
FOUND ONE ?
Y
DEST. ADDR = STATION ADDR. ?
N
ABORT PACKET
SET NL_RXSEMA. TBF = FALSE
Y
N
DOES THIS DIGIPEAT ADDRESS = STATION ADDRESS
Y
PR_RXSEMA. RBF = FALSE ?
N
Y
SET DP BIT OF THIS DIGIPEAT ADDRESS IN PACKET, AND REFORMAT PACKET FOR FORWARDING TO NEXT STATION IN PATH.
PASS PACKET TO PL MODULE
DL_TXSEMA_P0. TBF = FALSE ?
N
DIGIPEAT QUEUE FULL ?
Y
PR_RXSEMA. TBF = TRUE
Y
N
PASS DIGIPEAT PACKET TO DLL AS PRIORITY 0 PACKET (P0)
SET DL_TXSEMA_P0. TBF = TRUE
PUT DIGIPEAT PACKET INTO QUEUE
SET NL_RXSEMA. TBF = FALSE
SET NL_RXSEMA. TBF = FALSE
EXIT

Figure 34. Network Layer Receive flowchart.

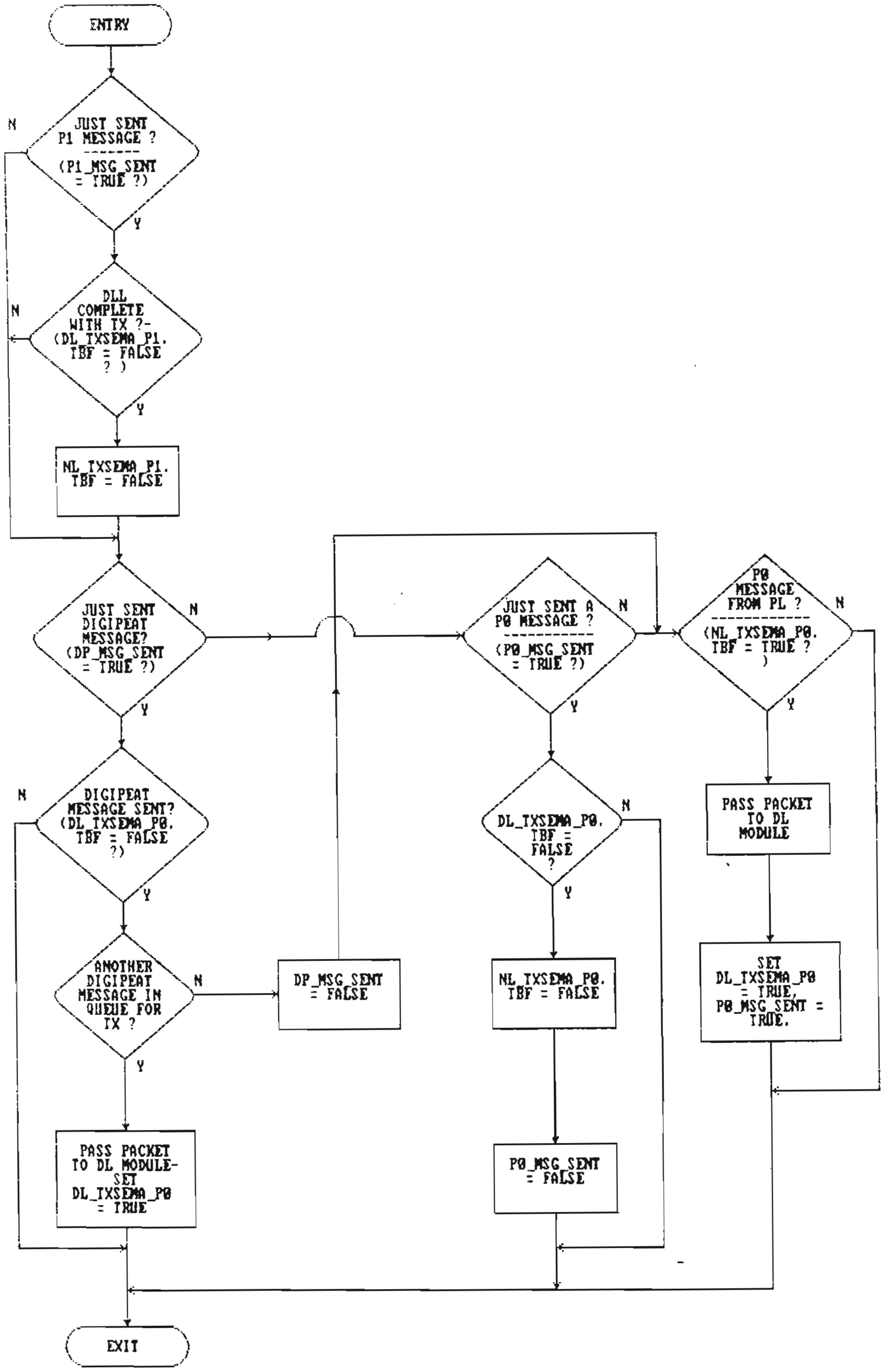

ENTRY
JUST SENT P1 MESSAGE ? ------- (P1_MSG_SENT = TRUE ?)
N
Y
DLL COMPLETE WITH TX ?- (DL_TXSEMA_P1. TBF = FALSE ?)
N
Y
NL_TXSEMA_P1. TBF = FALSE
JUST SENT DIGIPEAT MESSAGE? (DP_MSG_SENT = TRUE ?)
N
Y
DIGIPEAT MESSAGE SENT? (DL_TXSEMA_P0. TBF = FALSE ?)
N
Y
ANOTHER DIGIPEAT MESSAGE IN QUEUE FOR TX ?
N
Y
PASS PACKET TO DL MODULE- SET DL_TXSEMA_P0 = TRUE
DP_MSG_SENT = FALSE
JUST SENT A P0 MESSAGE ? ----------- (P0_MSG_SENT = TRUE ?)
N
Y
DL_TXSEMA_P0. TBF = FALSE ?
N
Y
NL_TXSEMA_P0. TBF = FALSE
P0_MSG_SENT = FALSE
P0 MESSAGE FROM PL ? ----------- (NL_TXSEMA_P0. TBF = TRUE ?)
N
Y
PASS PACKET TO DL MODULE
SET DL_TXSEMA_P0 = TRUE, P0_MSG_SENT = TRUE.
EXIT

Figure 35. Network Layer Transmit flowchart.

Transmission of priority 1 messages from the PL take priority over digipeat messages queued for transmission and these messages take priority over transmission of priority 0 messages from the PL module.

Interface between Network Layer and Presentation Layer is as follows:

Transmission of Priority 0 messages (from the PL to the NL):

The following variables make up the interface between the PL and NL modules for transmission of priority 0 messages.

NL_TXSEMA_P0	- single byte
NL_TXBUFPTR_P0	- double byte
NL_MAXRETRY_P0	- single byte

Transmission of Priority 1 messages (from the PL to the NL):

The following variables make up the interface between the PL and NL modules for transmission of priority 1 messages.

NL_TXSEMA_P1	- single byte
NL_TXBUFPTR_P1	- double byte
NL_MAXRETRY_P1	- single byte

Reception of messages (between the NL and PL):

PR_RXSEMA	- single byte
PR_RXBUFPTR	- double byte

The flags and registers above are defined in the same way as the corresponding variables in the DLL module.

The subroutine NLTIRQ implements the various functions and services of the NL module.

3.4.3.3 **Presentation Layer Module**

The PL module handles DIT read and writes and handles the passage of data to and from the application processes. Figure 36 is a flowchart of the sequence of events in the PL module.

The PL module constantly monitors the RBF flag in the PR_RXSEMA register. Data has arrived for the PL when the RBF flag is set to 1. The PL will check the data type in the packet passed.

Packets with data types of less than 29 are passed to the application processes if the application is able to accept messages (RBF flag in AP_RXSEMA must be false). If the application is unable to accept the

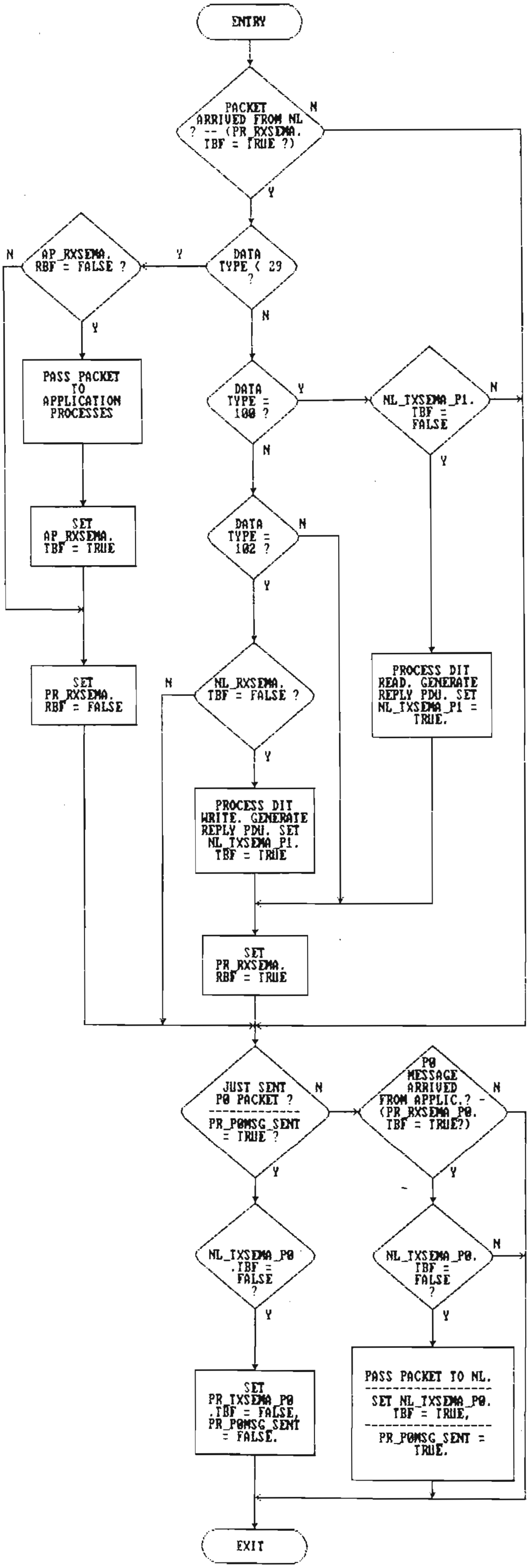

ENTRY
PACKET ARRIVED FROM NL ? -- (PR_RXSEMA. TBF = TRUE ?)
N
Y
DATA TYPE < 29 ?
Y
N
AP_RXSEMA. RBF = FALSE ?
N
Y
PASS PACKET TO APPLICATION PROCESSES
SET AP_RXSEMA. TBF = TRUE
SET PR_RXSEMA. RBF = FALSE
DATA TYPE = 100 ?
Y
N
NL_TXSEMA_P1. TBF = FALSE
N
Y
DATA TYPE = 102 ?
N
Y
PROCESS DIT READ. GENERATE REPLY PDU. SET NL_TXSEMA_P1 = TRUE.
NL_RXSEMA. TBF = FALSE ?
N
Y
PROCESS DIT WRITE. GENERATE REPLY PDU. SET NL_TXSEMA_P1. TBF = TRUE
SET PR_RXSEMA. RBF = TRUE
JUST SENT P0 PACKET ? PR_P0MSG_SENT = TRUE ?
N
Y
P0 MESSAGE ARRIVED FROM APPLIC.? - (PR_RXSEMA_P0. TBF = TRUE?)
N
Y
NL_TXSEMA_P0 .TBF = FALSE ?
Y
NL_TXSEMA_P0. TBF = FALSE ?
N
Y
SET PR_TXSEMA_P0 .TBF = FALSE, PR_P0MSG_SENT = FALSE.
PASS PACKET TO NL. SET NL_TXSEMA_P0. TBF = TRUE, PR_P0MSG_SENT = TRUE.
EXIT

Figure 36. Presentation Layer Module flowchart.

packet, the packet is abandoned by clearing the PL receive buffers for another message. This has been done to avoid the clogging of messages by a slow application process.

Packets with data types of 100 and 102 are processed as DIT read and writes respectively. All other messages are aborted. The respective replies are generated and placed for transmission to the NL module as a priority 1 message.

With the current PL layer, application messages are transmitted as priority 0 messages.

Application messages are passed to the PL module using the PR_TXSEMA_P0 register and the PR_TXBUFPTR_P0 data pointer. Before attempting to transmit any new application messages, the PL will check if any messages have been transmitted to the NL module. It will monitor the TBF flag of the NL_TXSEMA_P0 register. This flag will be cleared if the previous message is transmitted. The PL will transfer status bits in the NL_TXSEMA_P0 register to the PR_TXSEMA_P0 register on completion of a transmission.

The subroutine PRTIRQ handles the various processes described above, in the PL module.

Interface between Presentation Layer and application is as follows:

Transmission of Priority 0 messages (between PL and Application):

The following variables make up the interface between the PL and NL modules for transmission of priority 0 messages.

NL_TXSEMA_P0	- single byte
NL_TXBUFPTR_P0	- double byte
NL_MAXRETRY_P0	- single byte

Reception of messages (between NL and PL):

PR_RXSEMA	- single byte
PR_RXBUFPTR	- double byte

These variables are defined in the same way as the corresponding variables in the DLL module.

3.5 **Performance and Testing**

The testing of the protocol software was done using first a single stations with an in circuit emulator, followed by a setup with two stations.

In the first instance, the station was connected to a "Protocol Analyzer" (PA) with a baseband connection. The PA was developed at CONLOG and is used to "sniff" the traffic in a packet radio network. The PA program will display all messages on the network and was used to check the validity of messages generated from the station.

In the second test, the two stations were connected to each other and the PA with a baseband connection. Again the validity of messages were checked using the PA.

The application in each station was programmed to transmit and received the various types of messages discussed in the protocol specification, and their operation verified.

Finally, the baseband connection between the two stations and the PA was replaced with radios and the whole test repeated.

The protocol was also tested with five stations and performed as designed.

At the time of printing of this document, all known bugs in the protocol software had been eliminated.

The performance of the protocol as shown in Appendix A, is heavily dependent on the PTT rise times of the radios. The longer the rise time is, the worse the throughput of the network gets under heavy loading. Due to the non-tractability of the p persistent protocol, it is difficult to establish the performance of the packet radio network without a fair amount of simulation. The theoretical performance of the packet radio network may be derived from Kleinrock & Tobagi's [1975] simulation studies with respect to the performance of p persistent protocols.

Using radios with a PTT delay of 250 milliseconds, with no traffic on the network and an average packet size of 20 bytes, channel operating with a bit rate of 1200 bits/second, the best case transmission time between a station initiating its message and receiving the acknowledgement to the message is:

PTT delay at transmitting station	:	250 ms +
transmission time of message	:	183 ms +
PTT delay of remote station	:	250 ms +
transmission time of acknowledgement	:	128 ms
total transmission time	=	811 ms.

The network designed is not meant to be a high speed network. The speed at which events are reported are not crucial to the system as a whole. The performance of the network is sufficient for the type of application that the packet radio system was designed for.

4 CONCLUSION

Although the packet radio network design for PACNET proved to be very successful for the system intended for, it has a number of shortcomings. One such shortcoming is the inability of the packet radio protocol to broadcast messages in a multihop environment. Another is the ability to duplicate messages in the network layer, with messages using a multihop path. Although these shortcomings do not pose a problem with the current application, further research need to be done in solving these.

It has been proven that poor radios may seriously degrade the performance of the network. In order to achieve more superior channel capacity, the transmit rise times of the radio need to be kept as low as possible in a half duplex channel. Another solution to this would be to use a full duplex communications channel, in which each station may receive messages whilst simultaneously transmitting a message. This would enable stations that are busy enabling their transmitters to abort their transmissions if traffic is detected on the network. This, however, does impact the cost of the system.

From the design of the packet radio network, it has been noticed how the use of standards have simplified the design process. Although the data link layer has been standardized, the layering of the protocol permits future standardization of the other layers in the protocol, when suitable standards come about.

The use of piggybacking of messages on the network not only utilizes the communications channel more efficiently, but greatly speeds up the transport of messages in the network. This was especially noticeable for messages using a multihop path.

Due to the poor tractability of the p persistent protocol, performance of the protocol can only be derived using extensive simulation techniques as shown by Kleinrock and Tobagi [1975]. Further research needs to be done in devising a model to simulate the packet radio network protocol designed, to enable optimization of the performance of the network.

The packet radio network, PACNET, although not the ideal solution for the industrial system, provides a reasonable compromise between cost and performance of the system.

5 REFERENCES

[1] Fouad A. Tobagi, 1980, *Muliaccess Protocols in Packet Communication Systems*, IEEE Transactions on Communications, Volume COM-28, No. 4, pp 468 - 488.

[2] William Stallings, 1985, *Data and Computer Communications*, Chapter 10, U.S.A, Macmillan Publishing Company.

[3] Robert E. Kahn, 1977, *The Organisation of Computer Resources into a Packet Radio Network*, IEEE Transactions on Communications, Volume COM-25, No. 1, pp 169-178.

[4] Joseph L. Hammond and Peter J.P. O' Reilly, 1986, *Performance Analysis of Local Computer Networks*, Chapters 6 and 9, U.S.A, Addison-Wesley Publishing Company.

[5] Leonard Kleinrock and Fouad A. Tobagi, 1974, *Carrier Sense Multiple Access for Packet Switched Radio Channels*, ICC, pp 21B-1 to 21B-7.

[6] Leonard Kleinrock and Fouad A. Tobagi, 1975, *Packet Switching in Radio Channels: Part 1 - Carrier Sense Multiple Access Modes and Their Throughput-Delay Characteristics*, IEEE Transactions on Communications, Volume COM-23, No. 12, pp 1400-1416.

[7] Fouad A. Tobagi, 1980, *Analysis of a Two Hop Centralized Packet Radio Network - Part 1: Slotted Aloha*, IEEE Transactions on Communications, Volume COM-28, No. 2, pp 196-207.

[8] William Stallings, 1990, *Handbook of Computer Communications Standards Volume 1: The Open Systems (OSI) Model and OSI-Related Standards,* Second Edition, Volume 1, U.S.A, Howard W Sams & Company.

[9] International Electrotechnical Commission (IEC), 1990, *Telecontrol Equipment and Systems, PART 5: Transmission Protocols, Section One - Transmission frame formats*, IEC 870-5-1.

[8] William Stallings, 1990, *Handbook of Computer Communications Standards Volume 2: Local Area Network Standards,* Second Edition, Volume 2, Chapter 4, U.S.A, Howard W Sams & Company.

[9] Robert E. Kahn, Steven A. Gronemeyer, Jerry Burchfiel, Ronald C. Kunzelman, 1978, *Advances in Packet Radio Technology*, Proceedings of the IEEE, Volume 66, No. 11, pp 1468 - 1496.

[10] Osten Makitalo and Bertil Verri, 1980, *Signalling in the NMT System: Description of Methods and Results from Tests*, Radio Department, Swedish Telecommunications Administration.

[11] Andrew S. Tanenbaum, 1988, *Computer Networks*, Second Edition, U.S.A., Prentice Hall.

[12] Fred Halsall, 1988, *Data Communications, Computer Networks and the OSI*, Second Edition, Great Britain, Addison-Wesley.

[13] International Electrotechnical Commission (IEC), 1990, *Telecontrol Equipment and Systems Part 5: Transmission Protocols - Section One - Transmission Frame Formats (IEC 870-5-1)*, First Edition, Geneve, Central Bureau of the IEC.

[14] International Electrotechnical Commission (IEC), 1988, *Draft: Telecontrol Equipment and Systems Part 5: Transmission Protocols - Section Two - Link Transmission Procedures Revision 8 - (IEC 870-5-2)*, Geneve, Central Bureau of the IEC.

6 APPENDIX A : EFFECT OF NON IDEAL RADIOS ON THROUGHPUT

The analysis below shows the effect of a non ideal radio transmitter on the throughput of non persistent CSMA. Non persistent CSMA has been chosen for the analysis since it is more tractable than the other CSMA techniques and the calculation is less tedious.

Assumptions:

(a) traffic source is an infinite number of users generating traffic with a Poisson source. Packets are generated infrequently and the packets are transmitted in a time interval less than the period between generation of packets by the station.

(b) All packets generated are of constant length that require a transmission period of T seconds.

(c) The propagation delay between any two stations is "t" seconds where t is the one way propagation delay on the medium.

(d) Carrier sensing takes place immediately.

(e) Channel is noiseless.

(f) Overlap of any two packets is destructive and require retransmission.

(g) The non-ideal transmitter has a rise time (PTT delay) of Tr seconds.

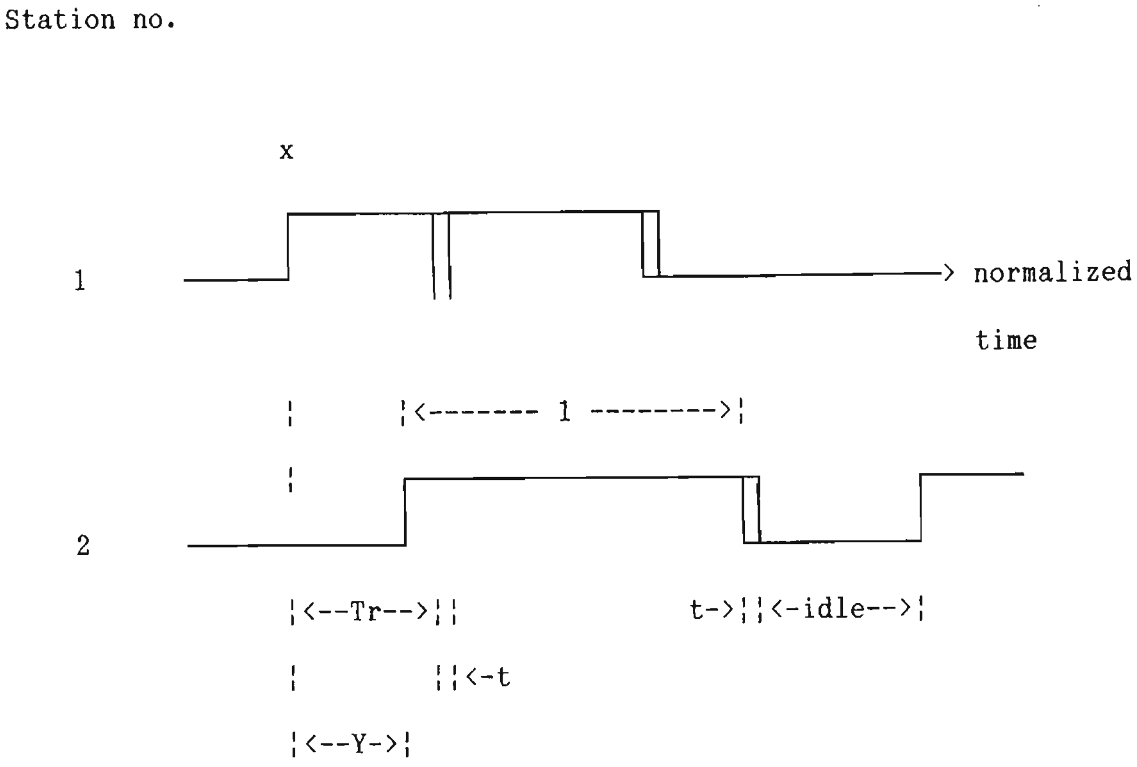


Figure A1 . Busy and idle periods for non persistent CSMA.

Consider a station sensing the channel idle at time x and transmitting. Any other station wishing to transmit in the period Tr+a will sense the channel idle and transmit. All packets arriving in period Y will be completed by Y+1+a.

Let I be the average duration of the idle period, B be the average duration of the busy period and U the average duration of the channel when utilized without conflicts.

Therefore throughput is:

$$S = U/(I+B) \qquad (A-1)$$

The probability that a transmission period is successful is the probability that no stations transmit in the period Tr+a, therefore

$$U = e^{-(Tr+a)G} \qquad \text{(A-2)}$$

The average duration of an idle period is simply 1/G and the average duration of a busy period is 1+Y+a+Tr. The distribution function for Y is:

$$F_Y(y) \tilde{=} \Pr\ \{Y<=y\} = \Pr\{\text{no arrivals occur in period } [(Tr+a)-y]\}$$

$$= e^{-G(Tr+a-y)} \qquad \text{(A-3)}$$

Therefore

$$Y = Tr + a - 1/G(1 - e^{-(Tr+a)G}) \qquad \text{(A-4)}$$

This gives

$$S = \quad Ge^{-(Tr+a)G}/(G[1+2a+Tr] + e^{-(Tr+a)}G) \qquad \text{(A-5)}$$

Two graphs have been plotted in Figure A2, one with an "a" value of 0 and a "Tr" value of 0,5767 and the other with an "a" value of 0.00038 and a "Tr" value of 0.

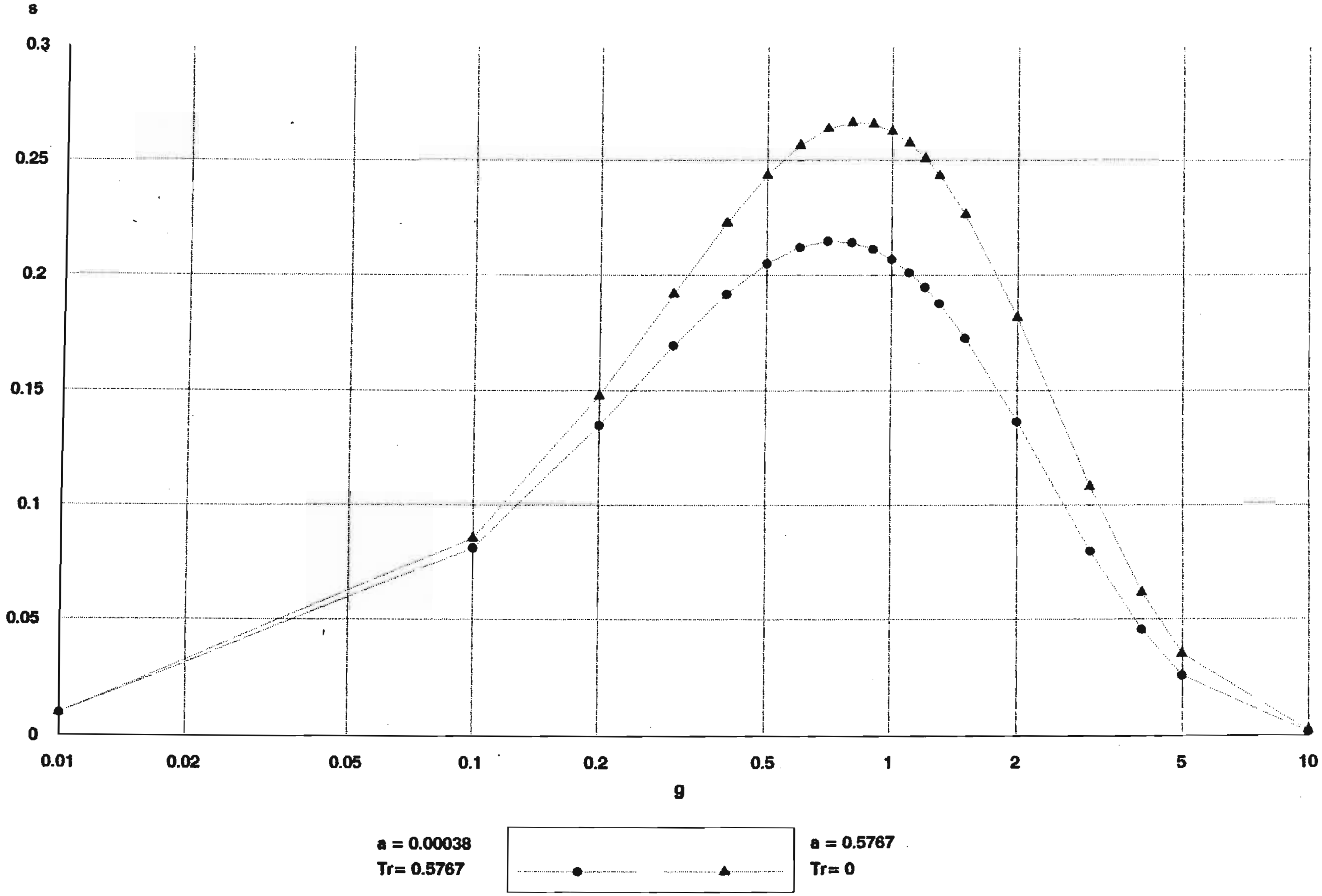


FIGURE:A2 GRAPH OF THROUGHPUT (S) VS OFFERED LOAD (G) FOR NON-PERSISTENT CSMA

From these curves it can be seen that the radio rise time has an effect on the throughput that is close to a similar effect with a large value of propagation delay "a". Hence known curves of throughput-load may be used to estimate the performance of a network by letting "a" equal the "Tr" value in the system.

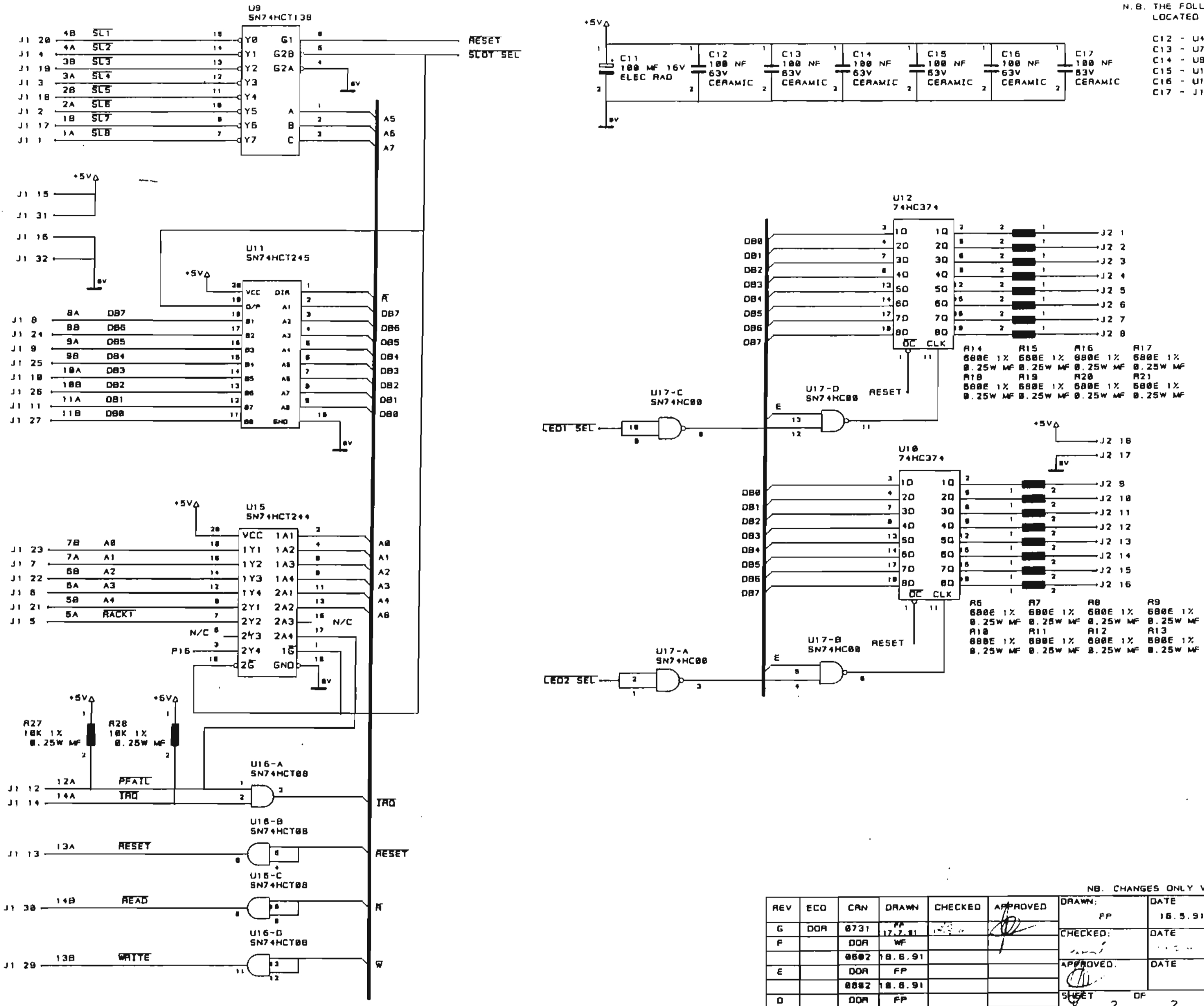
7 APPENDIX B : CIRCUIT DIAGRAMS

The circuit diagrams included in this appendix are those of the RTU CPU/Modem module. The following circuit diagrams are included:

(a) RTU CPU/Modem Board 1 sheets 1 and 2,

(b) RTU CPU/Modem Board 2 sheets 1 and 2 and,

(c) Personality Module.



N.B. THE FOLLOWING CAPACITORS MUST BE
LOCATED CLOSE TO THE RESPECTIVE ICS

C12 - U4 PIN 1&21
C13 - U7 PIN 8&16
C14 - U9 PIN 8&16
C15 - U11 PIN 10&20
C16 - U15 PIN 10&20
C17 - J1 PIN 15&16

NB. CHANGES ONLY VALID ONCE FILED ON COMPUTER

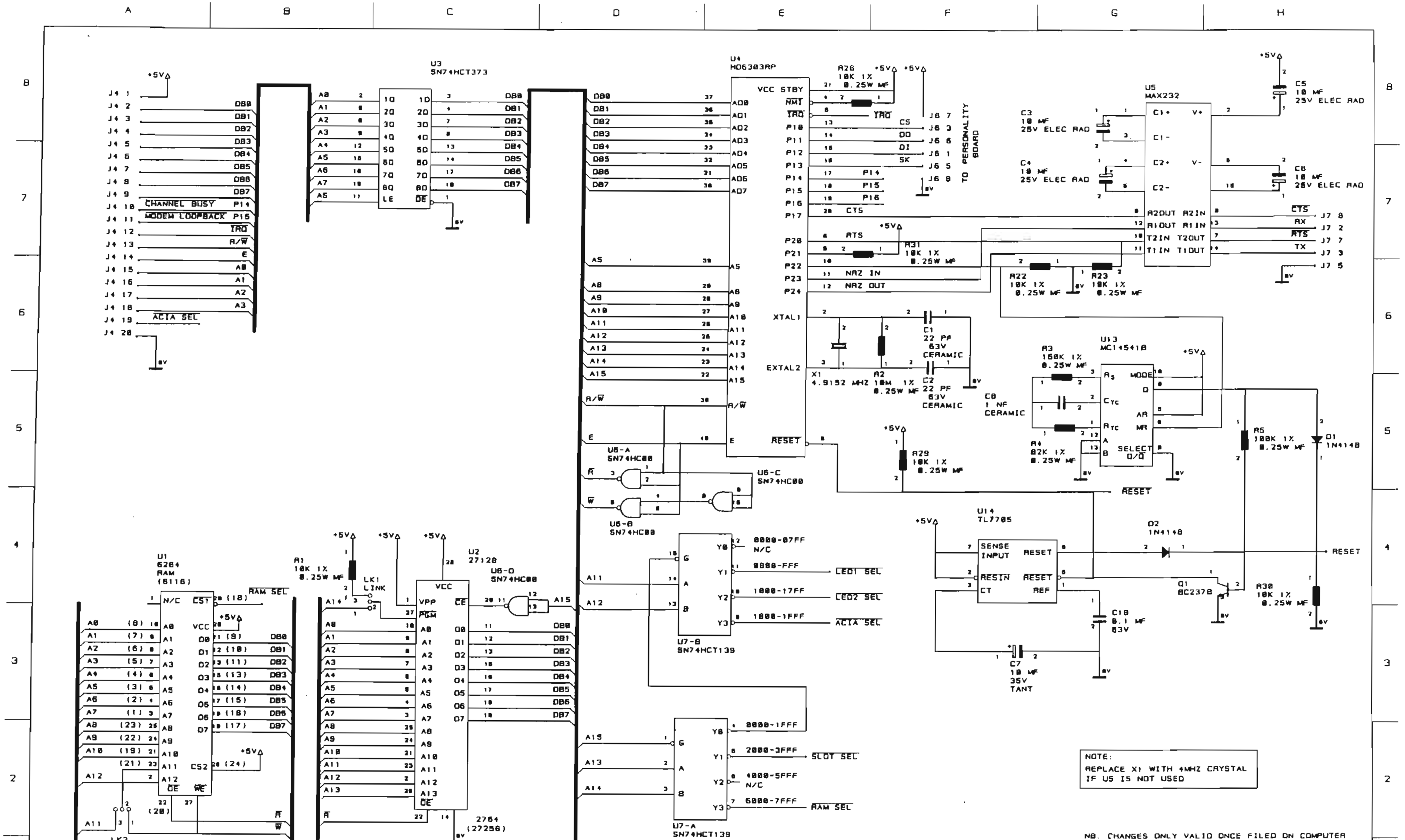
REV	ECO	CRN	DRAWN	CHECKED	APPROVED
G	DOR	0731	FP 17.7.91		
F		DOR	WF		
		0602	18.6.91		
E		DOR	FP		
		0602	18.6.91		
D		DOR	FP		
		0602	4.6.91		

DRAWN: FP	DATE 16.5.91
CHECKED:	DATE
APPROVED.	DATE
SHEET 2	OF 2

PMRTU
RTU/CPU MODEM
BOARD 1
CIRCUIT DIAGRAM

COMLOG

93 1019 112 | ISS G | A2



NOTE:
REPLACE X1 WITH 4MHZ CRYSTAL
IF U5 IS NOT USED

LK2. 1-2 - 6116
2-3 - 6264

LK1: 1-2 - 27256
2-3 - 27128 & 2764

NB. CHANGES ONLY VALID ONCE FILED ON COMPUTER

REV	ECO	CRN	DRAWN	CHECKED	APPROVED
G	DOR	0731	FP 17.7.91	D.S. 18.7.91	
F		DOR	WF		
		0602	18.6.91		
E		DOR	FP		
		0602	10.6.91		
D		DOR	FP		
		0602	4.6.91		

DRAWN. WF — DATE 16.5.91

CHECKED: — DATE 18.7.91

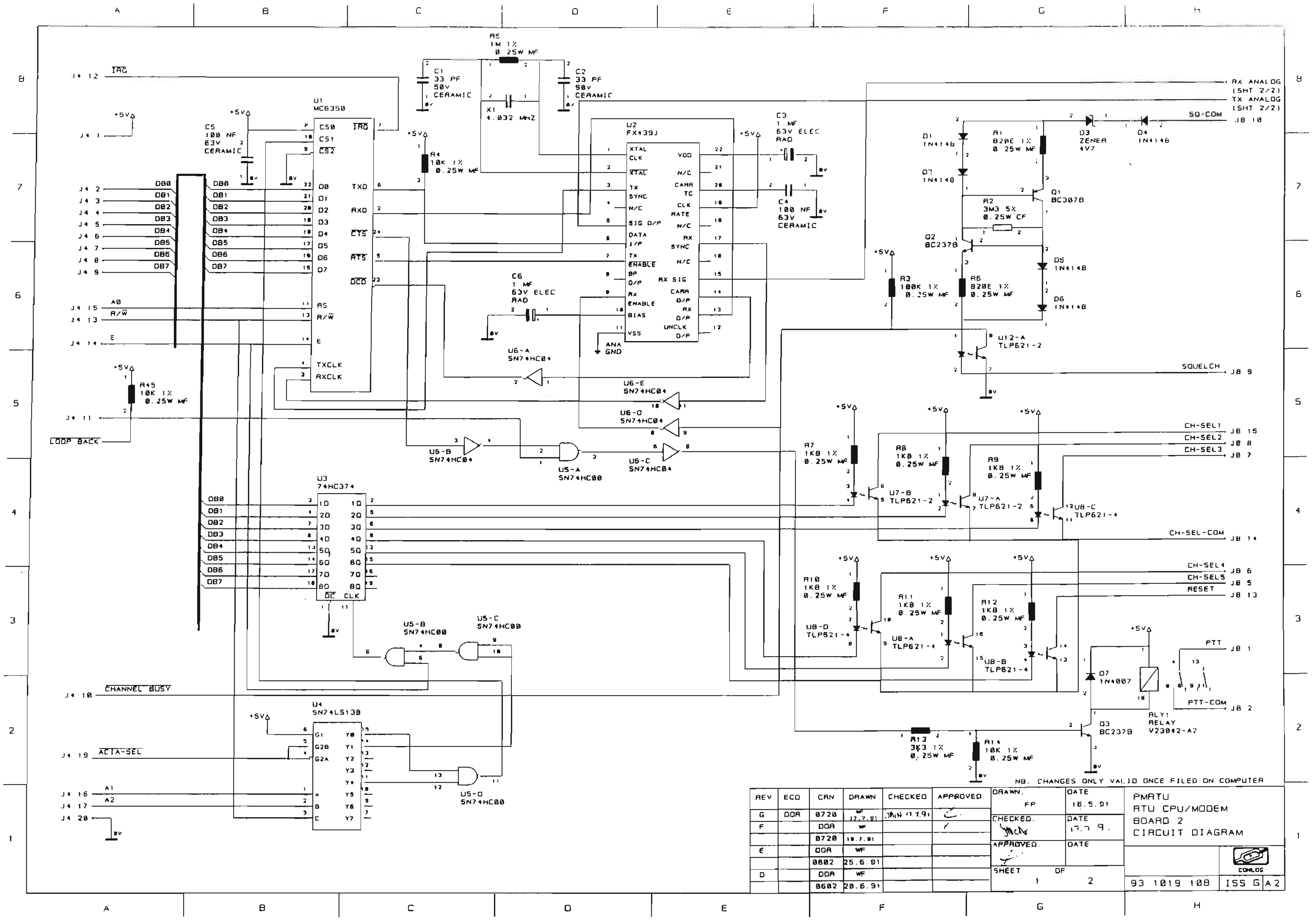
APPROVED. — DATE

SHEET 1 OF 2

PMRTU
RTU/CPU MODEM
BOARD 1
CIRCUIT DIAGRAM

COMLOG

93-1019-112 | ISS G | A2



NB. CHANGES ONLY VALID ONCE FILED ON COMPUTER

REV	ECO	CRN	DRAWN	CHECKED	APPROVED
G	DOR	0720	WF 17.7.91	JMcN 17.7.91	
F		DOR	WF		
		0720	18.7.91		
E		DOR	WF		
		0602	25.6.91		
D		DOR	WF		
		0602	20.6.91		

DRAWN. FP — DATE 16.5.91

CHECKED. JMcN — DATE 17.7.9.

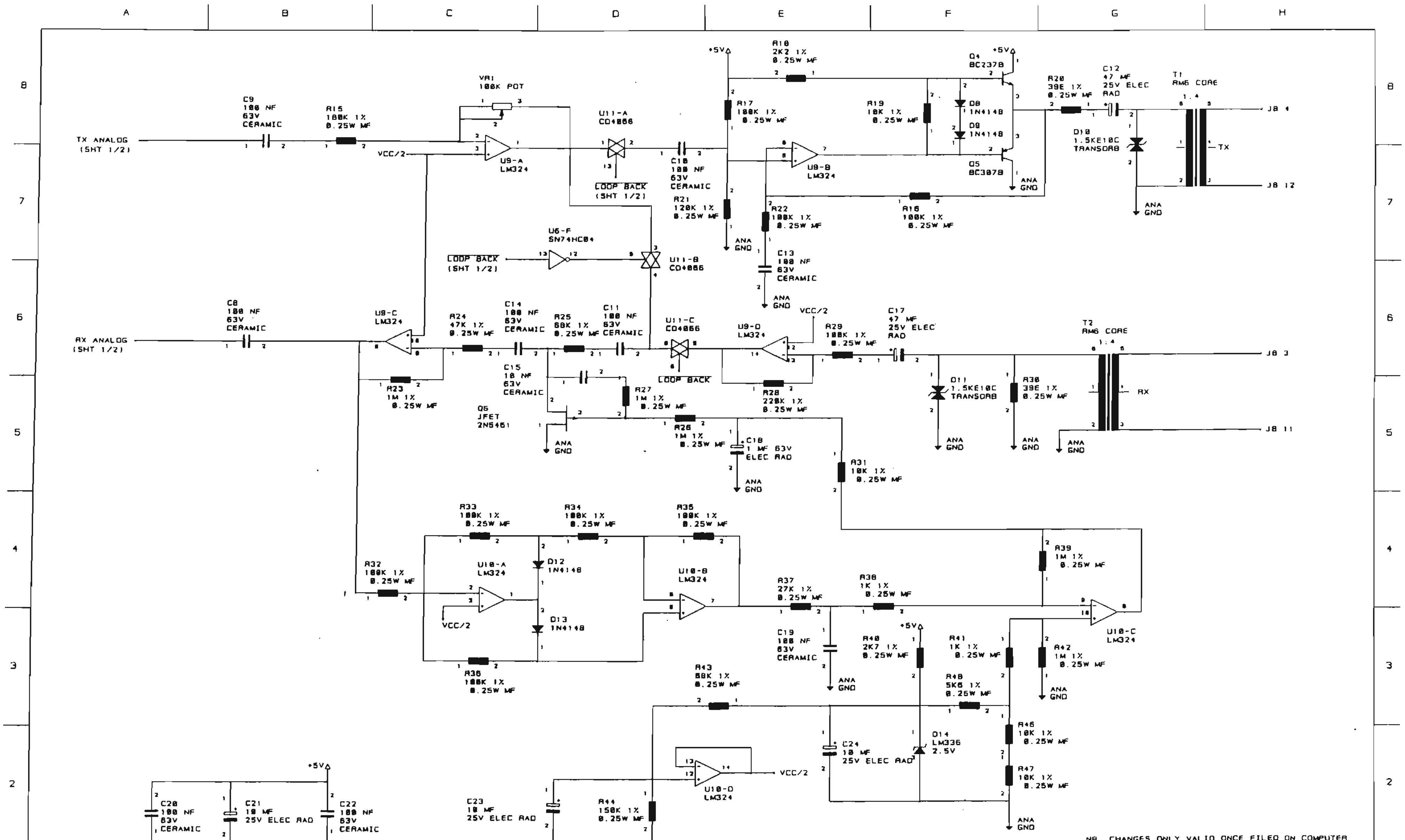
APPROVED. — DATE

SHEET 1 OF 2

PMRTU
RTU CPU/MODEM
BOARD 2
CIRCUIT DIAGRAM

COMLOG

93 1019 108 ISS G A2



NB. CHANGES ONLY VALID ONCE FILED ON COMPUTER

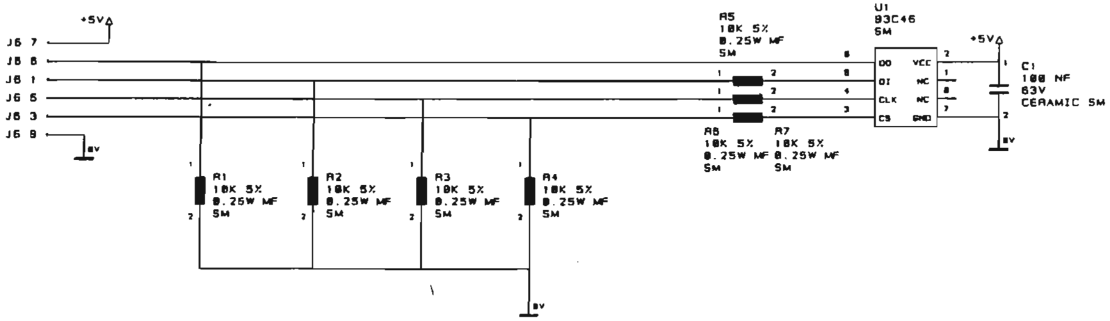
REV	ECO	CRN	DRAWN	CHECKED	APPROVED
G	DOR	0720	WF 17.7.91	17.7.91	
F		DOR	WF		
		0720	10.7.91		
E		DOR	WF		
		0602	25.6.91		
D		DOR	WF		
		0602	20.6.91		

DRAWN: WF	DATE 17.5.91
CHECKED.	DATE
APPROVED.	DATE
SHEET 2	OF 2

PMRTU
RTU CPU/MODEM
BOARD 2
CIRCUIT DIAGRAM

COMLOG

93 1019 108 | ISS G | A 2



NB. CHANGES ONLY VALID ONCE FILED ON COMPUTER

REV	ECO	CRN	DRAWN	CHECKED	APPROVED
D		DOR	WF	V.	
		0729	18.7.91	19.7.91	
C		DOR	WF		
		0729	17.7.91		
B		DOR	WF		
		0602	6.6.91		

DRAWN: WF	DATE 20.5.91
CHECKED:	DATE 19.7.9
APPROVED:	DATE
SHEET 1	OF 1

PMRTU
PERSONALITY MODULE
CIRCUIT DIAGRAM

COMLOG

93-1019-114 | ISS D | A2

8 APPENDIX C : SUMMARY OF PACNET PROTOCOL DRIVERS AND OTHER RELATED SOFTWARE

The PACNET protocol driver suite contain source code used in the implementation of the packet radio network PACNET.

The drivers are found in the floppy disk called "PACNET Drivers".

The protocol drivers associated with the Data Link Layer Module is found in the following files:

(a) INT.SRC - holds the communications interrupt handling routine MIRQ. This routine will distinguish between the various interrupts associated with the 6850 ACIA used as the communications interface to the network.

(b) DLINT.SRC - contains the subroutines PROC_TXMSG and PROC_RXMSG. These routines are used to implement the DLTIs and DLRIs state machines. These state machines are used to transmit and receive individual bytes from the physical medium.

(c) DLSERV.SRC - contains the Data Link Layer service routine. It consists of the subroutine DLTIRQ used to arbitrate between the DLTAs, DLTMs_P0 and DLTMs_P1 state machines. The DLTAs state machine is used to process the transmission of acknowledgements. The

DLTMs_P1/0 state machines are used to process the transmission of priority 1 and 0 messages respectively.

The protocol drivers associated with the Network Layer Module is found in the following file:

(a) NLSERV.SRC - contains the Network Layer Service routine NLTIRQ. This subroutine will process all Network Layer related tasks, such a digipeating, routing etc.

The protocol drivers associated with the Presentation Layer Module is found in the following file:

(a) PLSERV.SRC - contains the Presentation Layer Service routine PLTIRQ. This subroutine will process all Presentation Layer related tasks such as local DIT read/write, remote DIT read/write etc.

Apart from the protocol drivers above, other miscellaneous files used in the support of the protocol drivers is included in the floppy disk called "PACNET Support". The disk contain files that perform functions such as initialization of the hardware, power-up processing software, personality module read/write software, general I/O handling software for the microprocessor.

Of concern to this theses is the following files:

APPTICK.SRC - contains the 10 millisecond timer interrupt service routine. This routine is used to call the PLTIRQ, NLTIRQ and DLTIRQ subroutines. It is also used to handle messages from the application processes.

APPFUNC.SRC - contains a collection of routines used in the processing of FORTH calls from the application processes, eg subroutines to initiate message transfers between the application processes and the presentation layer etc.

9 APPENDIX D : RTU APPLICATION SOFTWARE FUNCTIONAL SPECIFICATION

This appendix covers the functional specification of the application processes used in the RTU. This appendix is port of the "RTU Application Functional Specification Rev. 5" document.

The original document was prepared by the author of this theses.

1 FUNCTIONALITY

1.1 GENERAL DESCRIPTION

The RTU application will be required to enable the RTU to operate as a remote data acquisition and control device in a network with a Line Control Unit (LCU). The LCU will be connected to the Man Machine Interface (MMI) with the CONET LAN.
The LCU and MMI is collectively referred to as the Central Controller below.

The RTU application will perform data acquisition by continually scanning the various inputs. If the RTU application detects the occurrence of an event or alarm condition, it will initiate the transmission of a message back to the Central Controller.

The RTU application will also be required to check for messages addressed to it. If the message is addressed to it, the RTU application will interpret it and respond by performing the required task. Typically, the message could be a command to operate the controlled device, or perform a hardware or software test in the RTU.

Where applicable, RTUs will be required to operate in other modes used mainly for unit maintenance. A mode switch will be used to alter the state of the RTU.

The RTU application will interface to the communications network through the services offerred by the network kernel software in the RTU. The network kernal together with its interface to the application, is detailed in the LCU-RTU Protocol Functional Specification.

The following paragraphs provide more specific operational detail.

1.2 ANALOG INPUTS

The RTU application will continually scan its analog inputs. Each analog input will have a set of hysteresis limits to prevent spurious alarms. The four setpoint values provided are high-high, high, low, and low-low limits.

An analog value will be considered to be in alarm if it goes above the high-high limit or goes below the low-low limit. An analog value would be considered to be returning to normal when it goes below the high limit or above the low limit ie. returning from a high or low alarm respectively.

The input values will be filtered using a suitable algorithm (discussed in the section on the Data Interchange Table below). The filtered analog values will be updated in the DIT and compared with their respective hysteresis limits. A register change of state is generated when the analog value either goes into alarm or returns to normal. The analog value is uploaded as a change of state message.

The generation of change of state messages may be inhibited by setting the respective inhibit bit in the Analog Inputs Inhibit register in the DIT.

The RTU application software will also upload the analog values to the central controller based on a programmed poll period for analogs in the DIT. Each of the four analog inputs will have its own poll period in the DIT. The uploading of analog values is also subjected to the restrictions of the analog inhibit bit. The uploading facility is inhibited if a poll period of 0 is programmed.

1.3 DIGITAL INPUTS

The RTU application will scan its digital inputs on a regular basis.

The RTU application will update the DIT with the latest digital input status information. The RTU will monitor the following digital inputs:

(a) digital inputs used to monitor a controlled device
(b) tamper input - RTU enclosure door
(c) operational mode
(d) plant actuator gas low
(e) battery level
(f) primary power supply fail

A change of state in any of the inputs above will result in a bit change of state transmission to the central controller. The bit change of state message will contain the bit number of the bit that changed state and the state of the bit.

All digital inputs, excluding (e) and (f), will be time desensitized to prevent spurious change of state messages being generated.

The generation of bit change of state messages may be inhibited by setting the respective inhibit bit in the Digital Inputs Inhibit register in the DIT.

1.4 OUTPUT CONTROL

On receipt of the control output command message, the RTU application will then internally select the respective output relay. It will then check the selection making sure that no other relays were selected. Upon internal confirmation the RTU application will latch power on to the selected output for a programmed duration. Upon completion of the operation, the RTU software will deselect all relays and send a message back to the Central Controller with the results of the operation. This would be in one of the following forms:

(a) bit change of state message to the central controller being generated as a result of a change in the status of the feedback switches on the plant actuator or,

(b) register change of state resulting from a change in the RTU status code register. This register will contain any errors that may have occurred in the operation.

A change of state of the input used to detect the closed position of the first controlled device will increment the accumulator DIT register. The input used in this case would be status input 1 in the Status Inputs DIT register.

1.5 OPERATING MODES

The RTU will be required to function in four different modes. Mostly, the RTU will function in the normal mode and will carry out the normal functions of data acquisition and control. However, primarily for maintenance purposes, there will be three other modes. Mode selection will be operated through a four position switch connected to the RTU inputs.

The three operating modes will be as follows:

(a) Normal Mode. All normal functions will apply as specified above.

(b) Manual Mode. This mode will allow for manual actuation of the controlled device. A push-button pair will be required to provide this function externally. In manual mode, the normal source of actuation, i.e. the control module digital outputs, will be completely isolated from the plant actuator. Likewise, in normal mode, the push-buttons will be completely isolated from the plant actuator. The push-buttons will be able to operate independently from the RTU electronics module.

(c) Lock-Out Mode. In this mode the plant actuator will be completely isolated from either actuation source.

(d) Test Mode. The Test mode will allow the Test Set access to the RTU to carry out a variety of test sequences and other functions. Control output operations are inhibited in this mode of operation.

When any mode change occurs, the RTU application will transmit this information to the Central Controller. Apart from the mode status, the operator at the Central Controller will still receive all input state change and alarm information from the RTU Independent of mode. Likewise he may request all the normal responses from the RTU, (the result of a select and execute command would be unsuccessful).

The mode switch will be sensed by the RTU control unit using digital inputs.

1.6 CONFIGURATION PARAMETERS

Configuration parameters for the RTU will reside in the RTU DIT as shown under the section on the RTU DIT below. Configuration parameters range from DIT registers 0 to 256. Refer to the section below on the RTU DIT.

1.7 RTU SELF-TESTS

The RTU control module will periodically perform hardware and software internal tests to ensure module integrity.

If the test shows a module failure, a register change of state message of the RTU Status Code register will be transmitted to the Central Controller. The Status Code register will be updated with the respective error codes.

The RTU will be able to detect faults on the following components:

(a) input/output modules.
(b) primary power supply.
(c) battery.

The tests will indicate correct operation or failure of any of the above.

The tests may also be invoked from the Test Set, and the results of which must be sent back to the Test Set. The Test Set will interface to this facility in the RTU application using the reserved portion of the dynamic part of the DIT.

1.8 RTU ALIVE POLL

The RTU will initiate an alive poll to the central controller based on the programmed alive poll period for the RTU. The purpose of this poll is to make the central controller aware of the RTU "aliveness". The timing for the alive poll will start at power-up of the RTU or from the last transmission initiated from the RTu, whichever is later.

The data uploaded in the alive poll will contain all the dynamic DIT registers. The registers uploaded to the central controller ranges from DIT register 80 to 87.

2 PERFORMANCE

All digital I/O will be scanned once every second.

Each analog input will be scanned with a frequency of between 1 and 4 Hz.

3 INTERFACES

3.1 I/O

All I/O will be accessed as memory mapped I/O. Refer to the CPU/Modem Card Target Specifications for more details.

3.2 POWER SUPPLY

The power supply will be accessed as memory mapped I/O. Refer to the CPU/Modem Card Target Specifications for more details.

3.3 NETWORK KERNEL

The application software will use the services of the network kernel by accessing various functions in the bios/presentation layer of the network kernel. Function numbers, together with various parameters, enable certain tasks to be carried out in the network kernel.

The following function numbers will be made available to the application:

FUNCTION No.	DESCRIPTION
1 - 3	Reserved
4	Write DIT Register Data
5	Read DIT Register Data
6 - 10	Reserved
11	Transmit message
12 - 14	Reserved
15	Set Timer
16	Read Timer
17 - 21	Reserved

All functions shown above will be implemented as detailed in the Eziforth Programmers Reference Guide.

Using the same format for the description of the Bios functions in the Eziforth Programmers Reference Guide, function 11 are described below:

Function 11 : receive control message

Stack : 22 >>> operation output_number n

Description : This function will check if a control output message has arrived for the application. If n = 0, then no control message has arrived and the operation and output_number is omitted from the stack. If n = 1, then a control output message has arrived. The operation (0 = open, 1 = close) and output_number (1-4) is also passed on the stack.

The application software will check for a control output operation request by regularly polling the network semaphore register using BIOS function 8. If the receive buffer is detected to be full, the BIOS function 9 will be executed in order to retrieve the contents of the incoming message. The format and usage ofthe BIOS calls are explained in the EziForth Programmers Reference Guide.

3.4 DATA INTERCHANGE TABLE (DIT)

3.4.1 DIT LAYOUT - RTU

DIT REGISTER NUMBER	DESCRIPTION
0000-0031	Reserved
0032	Digital Input Sense Time
0033	Control Output 1 (Device 1 Close) Activation Time
0034	Control Output 2 (Device 1 Open) Activation Time
0035	Control Output 3 (Device 2 Close) Activation Time
0036	Control Output 4 (Device 2 Open) Activation Time
0037	Control Output 5 Activation Time
0038	Control Output 6 Activation Time
0039	Control Output 7 Activation Time
0040	Control Output 8 Activation Time
0041	Status Inputs Inhibit
0041	Analog Inputs Inhibit
0042	Analog Smoothing Factor
0043	Analog Upload Period for Analog Input 1
0044	Analog Upload Period for Analog Input 2
0045	Analog Upload Period for Analog Input 3
0046	Analog Upload Period for Analog Input 4
0047	Analog Input 1 High High Hysteresis Limit
0048	Analog Input 1 High Hysteresis Limit
0049	Analog Input 1 Low Hysteresis Limit
0050	Analog Input 1 Low Low Hysteresis Limit
0051-0062	Analog Inputs 2 - 4 Hysteresis Limits
0063	T_1 Maximum Channel Access Time
0064	Retransmission Count (N)
0065	Propogation Delay
0066	RTU address
0067	LCU - RTU digipeat 1 address
0068	LCU - RTU digipeat 2 address
0069	LCU - RTU digipeat 3 address
0070	LCU - RTU digipeat 4 address
0071	LCU address
0072	ALIVE Poll Period
0073	Radio Channel setup
0074-0079	Reserved
0080	RTU Status Code
0081	Status Inputs
0082	Accumulator
0083	Analog Input 1
0084	Analog Input 2
0085	Analog Input 3
0086	Analog Input 4
0087	General Diagnostics
0088-0255	Reserved

The contents of the DIT is described in detail in the LCU-RTU Protocol Specification. The DIT forms part of the Presentation Layer in layer 6 of the protocol model for the RTU.

The DIT is accessed from the application using the functions as described in the "Network Kernel" interface above.

3.4.2 DIT CONFIGURATION DATA

3.4.2.1 DIGITAL INPUT SENSE TIME

This is the time period for which a digital input status change must remain in force before being relayed to the application software for processing. Should the digital input undergo another change of state within this time period, the previous change of state will be ignored and the new status will be subjected to this time constraint. All inputs will use the same time period.

The value in this register is a multiple of 100 milliseconds. eg

		time period
DIT register value = 4	:	400 milliseconds

Range: 2 to 600
Default on device power-up for first time: 5

3.4.2.2 CONTROL OUTPUT ACTIVATION TIME

The control output activation time is the activation period of a control output at the RTU. The duration of the open or close activation period is programmablable as shown in the DIT.

The value in this DIT is a multiple of 100 milliseconds eg a DIT value of 30 represents a control output activation time of 3 seconds (3000 milliseconds).

Range: 5 to 50
Default on device power-up for first time: 20

3.4.2.3 STATUS INPUTS INHIBIT

This register contains bits that are used to inhibit a message, due to a change of state of any of the "Status Inputs", being transmitted to the LCU. This excludes change of states occurring as a result of a control output request from the central controller. The "Status Inputs" register is discussed in the respective section below.

Bit 0 to 15 of this register corresponds to bit 0 to 15 of the Status Input register. The bits may have the following values:

Bit value :		effect
1	:	inhibit status input message transmitted to LCU
0	:	allow status input message to be transmitted to LCU

Default on device power-up for first time: 0

3.4.2.4 ANALOG INPUTS INHIBIT

This register contains bits that are used to inhibit a message due to a change of state of any of the "Analog Inputs", being transmitted to the LCU. These occur when the analog value traverses a setpoint and goes into alarm. The "Analog Input" registers are discussed in the respective section below.

Bits 0 to 3 of this register corresponds to analog inputs 1 to 4 respectively. Bits 4 to 15 are unused. The bits in this register may have the following values:

Bit value	:	effect
1	:	inhibit analog change of state message transmitted to LCU.
0	:	allow analog change of state messages to be sent to LCU.

Default on device power-up for first time: 0

3.4.2.5 ANALOG SMOOTHING FACTOR

The smoothing factor is used in the analog filtering algorithm. The values in this register range from 0 to 100 and represent smoothing factors of 0 to 1 as discussed below. Refer to section on analog input for more detail.

Default on device power-up for first time: 50

3.4.2.6 ANALOG UPLOAD PERIOD

These registers are used to set the period between sending of successive analog data, pertaining to the respective analog inputs, to the central controller by the RTU. The sending of analog data is invoked automatically by the RTU.

The value entered in this register must be a multiple of 10 seconds

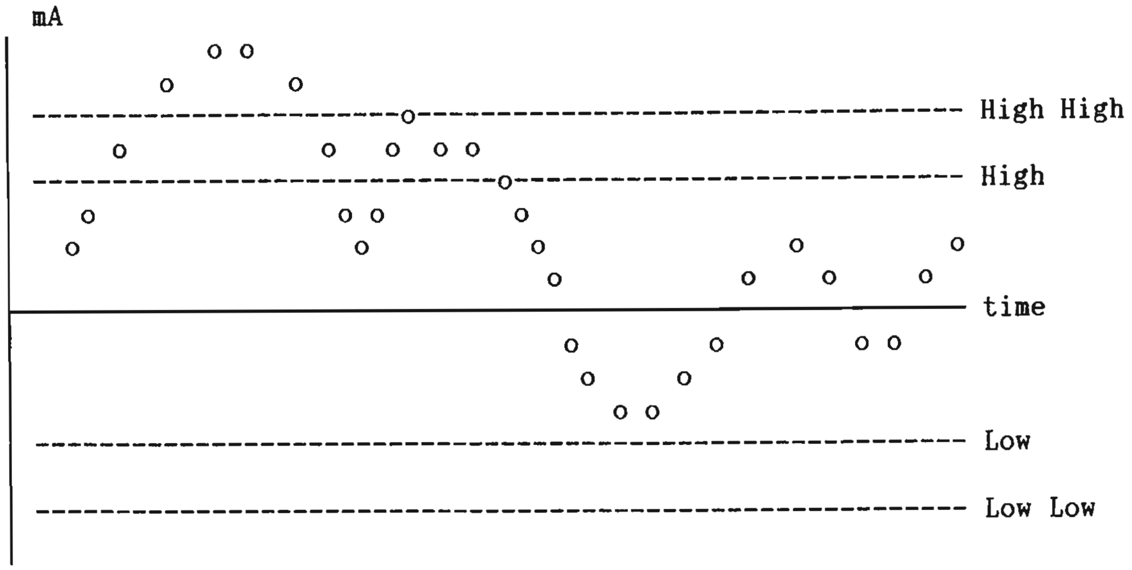
The value in this register will represent a multiples of 10 seconds. Valid range for this register is from 0 to 8640.

A value of 0 in this register will inhibit the RTU uploading analog data to the central controller. The uploading of analog data is also inhibited by setting of the respective inhibit bit in the Analog Inputs Inhibit DIT register.

3.4.2.7 ANALOG INPUT HYSTERESIS LIMITS

The hysteresis limits for each analog input will provide the analog input values with hysteresis so that spurious alarms may be curbed. The limits provided for are as illustrated below:

Current



The analog value will generate an alarm (change of state) message when it goes above the High High limit or goes below the Low Low limit. It will generate a back to normal (change of state) message when the analog value returns to below the Low High or above the High Low limit respectively.

Each analog value is filtered using the algorithm as discussed under ANALOG INPUTS below.

Range: 0 to FFFF hexadecimal
Default on device power-up for first time: FFFF Hex for High limits
: 0 for Low limits

3.4.2.8 T_1 MAXIMUM CHANNEL ACCESS TIME

This parameter is used in the channel access scheme as described in the LCU-RTU Protocol Specification. This is the maximum value that T_1 may have for a node.
The value in this register must be a multiple of the propogation delay as discussed below.

The value in the DIT register is entered as milli-seconds.

Range: (1 x propogation delay) to (n x propogation delay), provided that
(n x propogation delay) does not exceed 65535 milli-seconds.

n = 0, 1, 2, 3, 4 ....

Default on device power-up for first time: 4 x propogation delay

3.4.2.9 RETRANSMISSION COUNT (N)

The value in this register represents the maximum number of retransmissions that a message will undergo.

Range: 1 to 15

Default on device power-up for first time: 5

3.4.2.10 PROPAGATION DELAY

This DIT register represents the propagation delay incurred from the instance when an RTU has decided to make a transmission and another RTU being able to detect that transmission.

The value in the DIT is entered as milli-seconds.

Range: 1 to 10000 milli-seconds

Default on device power-up for first time: 250

3.4.2.11 LCU-RTU ADDRESSES

The LCU-RTU address DIT registers contain the digipeat address path from the LCU to the RTU in that order.

The RTU address represents the address of the RTU, whilst the LCU address represents the address of the LCU (line control unit).

3.4.2.12 ALIVE POLL PERIOD

The alive poll period is the time between consecutive uploads of the RTU dynamic DIT data to the central controller. This is used to make the central controller aware of the RTU's existence.

The value entered into the DIT must be in minutes.

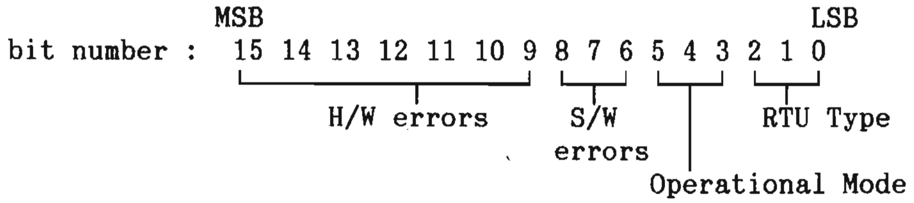
Range: 1 to 65535 minutes

Default on device power-up for first time: 1400

3.4.3 DIT DYNAMIC DATA

3.4.3.1 RTU STATUS CODE

The bits in this register are grouped to provide indication of RTU status. The bits are grouped as follows:



The RTU Type field is defined as follows:

Bit Number	2	1	0		RTU Type
	0	0	0	-	minimal
	0	0	1	-	basic
	0	1	0	-	expanded
	1	1	1	-	invalid

The RTU operational mode is defined as follows:

Bit Number	5	4	3		RTU Mode
	0	0	0	-	normal
	0	0	1	-	test
	0	1	0	-	manual
	0	1	1	-	lockout
	1	0	0	-	invalid
	1	1	1	-	reserved

The H/W (Hardware) and S/W (software) errors comprise of specific error codes used for diagnostic purposes.

ERROR LIST

Software errors:

Reserved for future use.

Hardware errors:

During control output activation:

1. Control output number invalid.
2. Output relay did not energise when selected.
3. Output relay contact did not close/supply relay contact is closed when output relay is energised.
4. Supply relay did not energise when selected.
5. Supply relay contacts did not close when relay was energised.
6. Supply relay did not de-energise when de-selected.
7. Supply relay contacts did not open when relay was de-energised.
8. Output relays did not de-energise when de-selected.
9. One of the output relays contact was closed when all output relays were de-energised.
10. Control operation successful in selection of supply and respective relays.

During normal processing:

11. One of the digital input circuitry in the digital input digital output card is faulty.
12. One of the digital input circuitry in the digital input analog input card is faulty.
13. Analog to Digital conversion circuitry on the digital input analog input card is faulty.
14. Modem circuitry faulty.
15. Radio faulty.
16. Power Supply Unit temperature high.
17. Personality module unplugged.

3.4.3.2 STATUS INPUTS

Each bit in this register represents the status of a specific function.
The bits are allocated as shown below.

Bit No.	function	Bit Value 1	Bit Value 0
0	I/O Card 1 dig. input 1 - control devl close	closed	open
1	I/O Card 1 dig. input 2 - control dev 1 open	closed	open
2	I/O Card 1 digital input 3 - tamper alarm	normal	door open alarm
3	I/O Card 1 digital input 4 - gas status	normal	low
4	I/O Card 1 digital input 5 - lockout mode	activated	not-active
5	I/O Card 1 digital input 6 - manual mode	activated	not-active
6	Primary power supply	normal	fail
7	Battery level	normal	low
8	I/O Card 2 digital input 1 - control dev2 close	closed	open
9	I/O Card 2 digital input 2 - control dev2 open	closed	open
10	I/O Card 2 digital input 3 - control dev3 close	closed	open
11	I/O Card 2 digital input 4 - control dev3 open	closed	open
12	I/O Card 2 digital input 5 - control dev4 close	closed	open
13	I/O Card 2 digital input 6 - control dev4 open	closed	open
14	I/O Card 2 digital input 7 - unuse	closed	open
15	I/O Card 2 digital input 8 - unused	closed	open

3.4.3.3 ACCUMULATOR

A change of state of the input used to detect the closed position of the first controlled device will increment the accumulator DIT register. The input used in this case would be status input 1 in the Status Inputs DIT register.

The accumulator will wrap around to 0 when the maximum value (65535) of the accumulator is reached. The accumulator is resettable from the network.

3.4.3.4 ANALOG INPUTS

These registers represent the filtered analog values as derived from the four analog inputs. The DIT will represent each analog as a 16 bit value.

Each analog value is filtered using the algorithm below:

$$SD_A = \frac{X_1 \; SD_C + (100 - X_1) \; SD_L}{100}$$

where SD_A = Smoothed analog value

X_1 = Smoothing Factor (range: 0 - 100)

SD_C = Current analog value

SD_L = Last smoothed analog value

The smoothing factor is derived from the DIT.

On an automatic download of the analog data, the RTU will send only these four registers to the central controller. The period between sending analog data is determined by the "Analog Poll Period" as discussed above.

3.4.3.5 GENERAL DIAGNOSTICS

This register will contain general diagnostic information.

4 ERROR HANDLING

The RTU software will check on a regular basis for corruption of software and hardware errors. The RTU will update the RTU Status Code register with the corresponding error number and upload the RTU Status Code register to the central controller.

All control outputs are disabled on detection of a software error.

5 TESTING

Testing can only be done when all software is present in the RTU, ie the application software and the network kernel. It is assumed that the network kernel software is fully debugged on linking with the application module.

Special software will be written to test the RTU application software. This may require slight modification of the existing "Chatterbox" program.

10 APPENDIX E : RTU APPLICATION SOFTWARE DETAILED DESIGN

This appendix covers the design of the application software used in the RTUs and is based on the RTU Application Functional Specification. This appendix is a port of the "RTU Application Software Detailed Design Rev. 3" document.

The generation of the original document as well as the design and coding of the software was carried out by the author of this theses.

The RTU application software was written in the FORTH programming language.

The FORTH RTU application software is included in the floopy disk marked "RTU Application Software".

1 INTRODUCTION

The RTU application software will provide the basic functionality of the RTU. It will enable the RTU to receive control messages from the central controller, perform controls, read and process status inputs, read and process analog inputs and handle hardware and software errors in the RTU.

The RTU appliction software will be written in the Eziforth programming language.

The RTU Application functionality is described in the Functional Specification of the RTU Application Software.

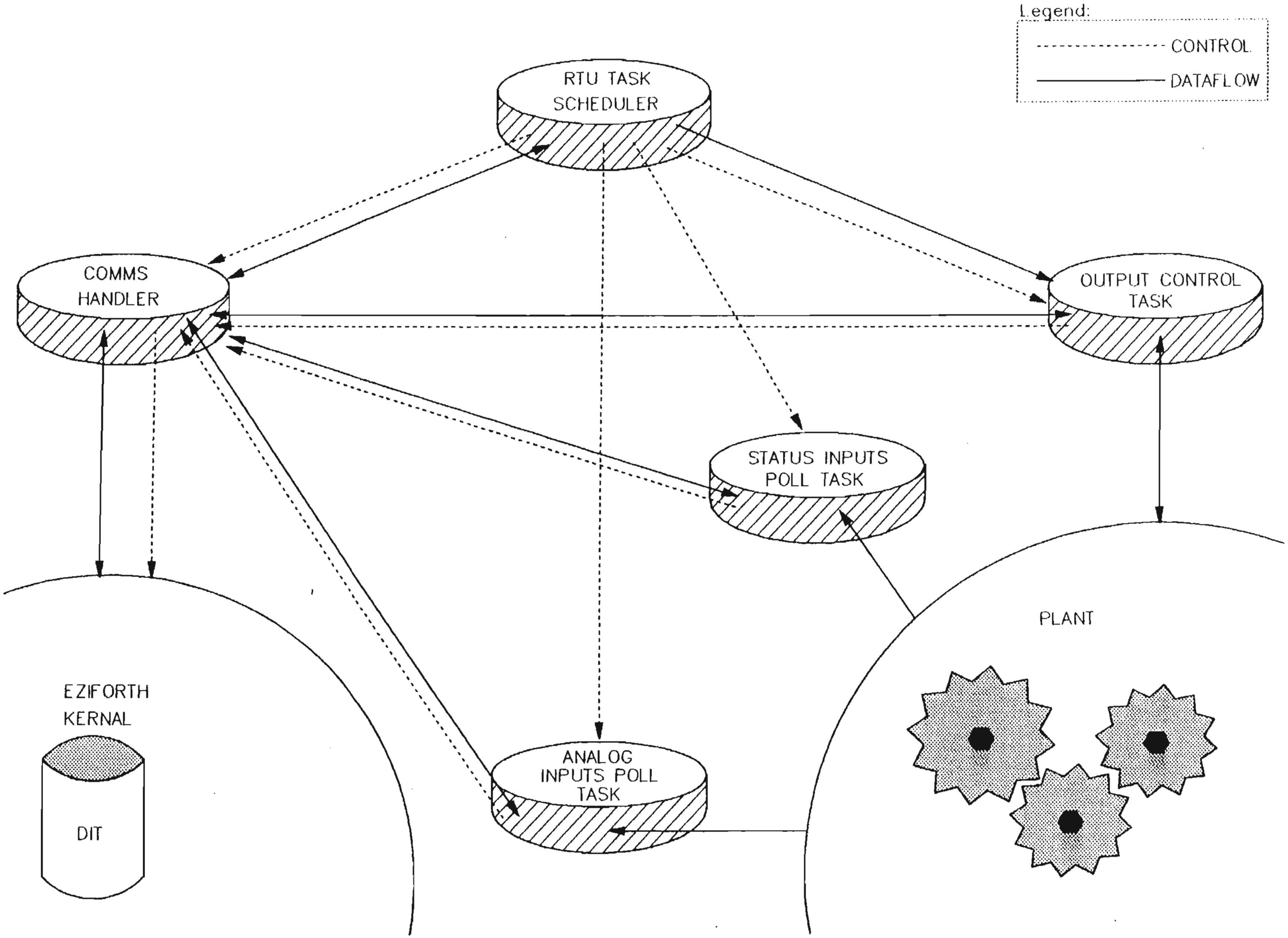
There are basically 5 modules that make up the RTU application software. These are as follows:

(a) RTU Task Scheduler
(b) Communications
(c) Output Control
(d) Status Inputs Poll
(e) Analog Inputs Poll

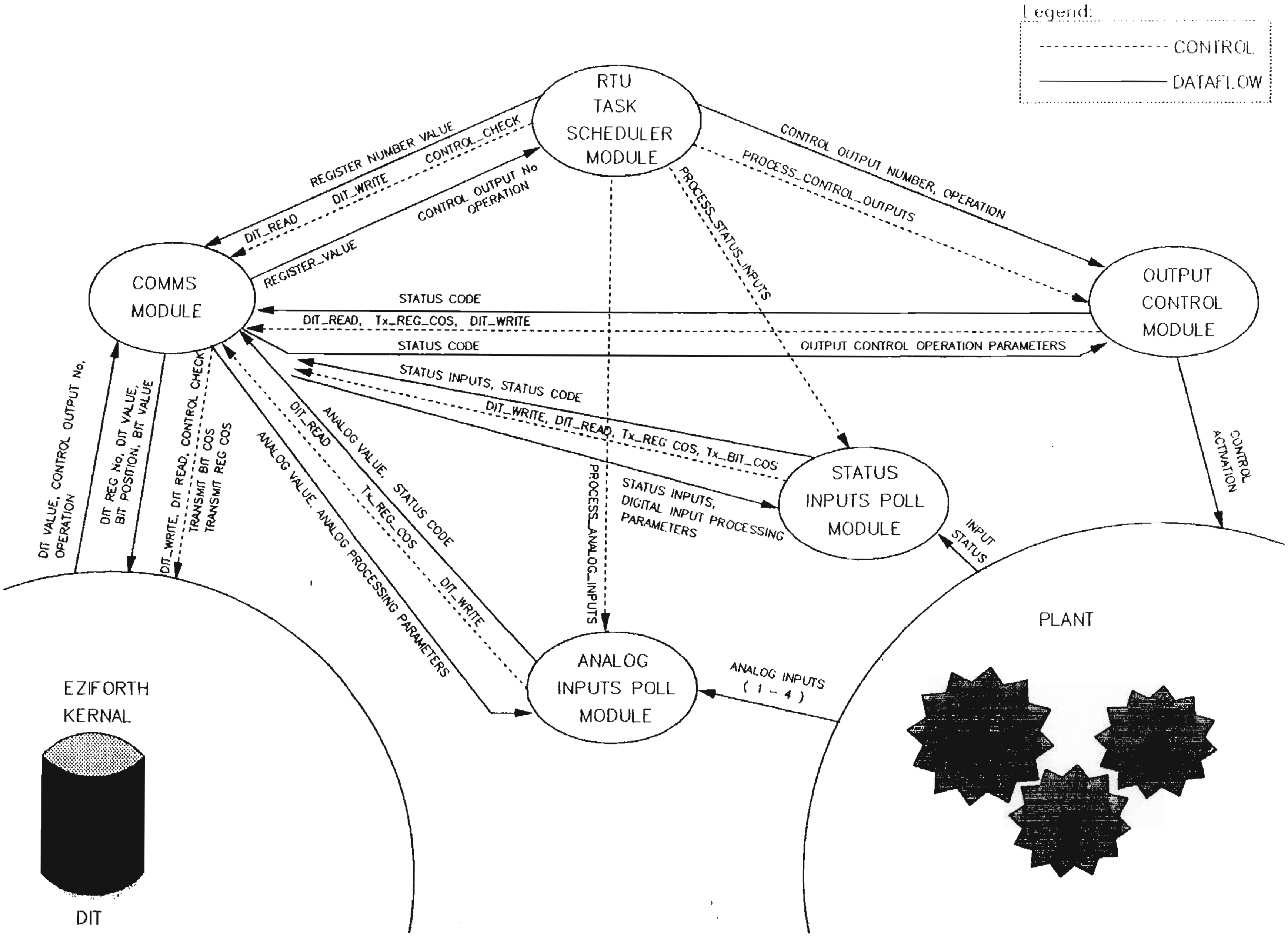
The data flow and control between the various modules is outlined in the RTU Application Software Architecture Diagram. A detailed dataflow of the application is given in the RTU Application Dataflow Diagram. This is followed by a corresponding data dictionary.

On power-up the Task Scheduler will run first and will call the other modules as required.

RTU APPLICATION SOFTWARE ARCHITECTURE DIAGRAM



RTU APPLICATION DATAFLOW DIAGRAM



DATA DICTIONARY

Control Output number : is the control output number used to carry out a control. (1 - 4)

Operation : 0 = open controlled device
1 = close controlled device

Analog value : is the filtered analog value to be written into the DIT or transmitted to the central controller

Status Inputs : Status inputs as generated by the Status Input poll module. Format of register is as per Status Inputs DIT register(Refer to Functional Specification of RTU Application).

Control Output Operation Parameters : control output activation time, Status Inputs Inhibit DIT register.

Analog Processing Parameters : Analog inputs inhibit DIT register, Analog input hysteresis limits, Analog smoothing factor, analog poll period.

Digital Input processing parms. : Digital input sense time, Status inputs inhibit register

2 DETAILED DESIGN

2.1 Overview

The main task in the RTU application is the Task Scheduler. This module will be executed on power-up of the RTU. This module will be responsible for initalising the RTU, transmitting alive polls and RTU self testing when required, call the Output Control Task when a control message arrives, call the Status inputs task to update the RTU status inputs, call the Analog inputs poll task to process the analog inputs.

Refer to the RTU Application Software Overview Flowchart.

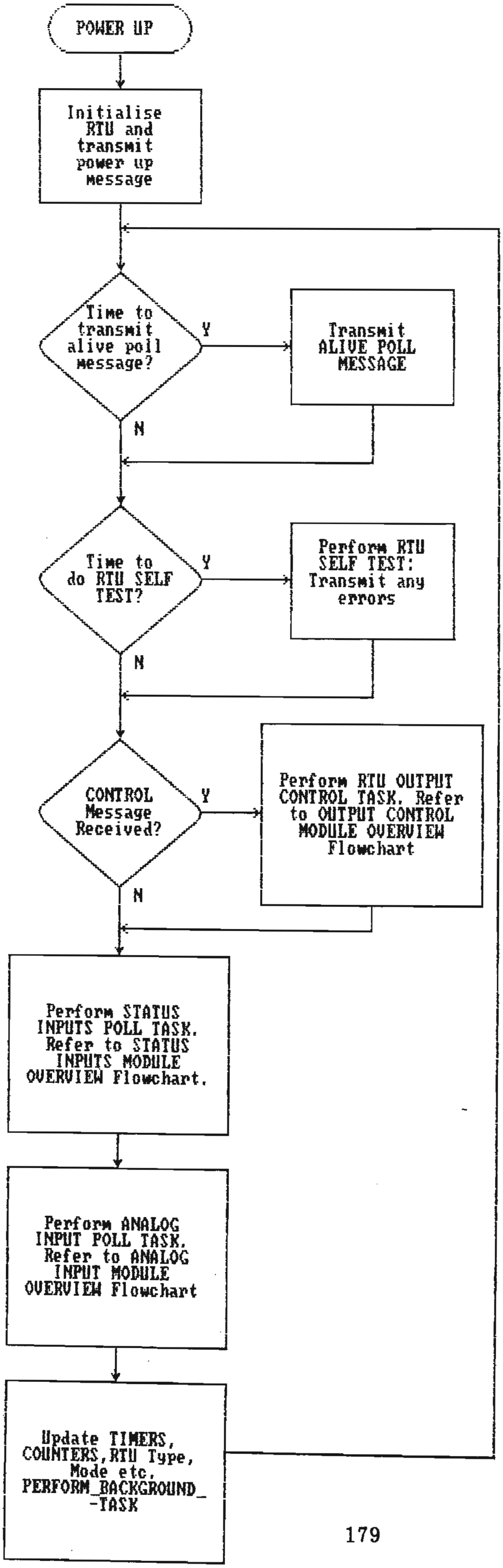
2.2 Nomenclature

The FORTH words are specified with the location and description of the stack contents before and after execution of the word. The stack contents are specified as follows:

eg. n1 n2 n3 >>> n4 n5 n6

The ">>>" symbol indicates execution of the associated word. The symbols to the left of ">>>" denote stack values that are used as operands and removed by execution, while the stack values to the right are the results of the execution. Where more than one symbol appears on either side, the rightmost symbol in a group is at the top of the stack.

RTU APPLICATION SOFTWARE OVERVIEW FLOWCHART (RTU TASK SCHEDULER MODULE Flowchart)


POWER UP
Initialise RTU and transmit power up message
Time to transmit alive poll message?
Y
Transmit ALIVE POLL MESSAGE
N
Time to do RTU SELF TEST?
Y
Perform RTU SELF TEST: Transmit any errors
N
CONTROL Message Received?
Y
Perform RTU OUTPUT CONTROL TASK. Refer to OUTPUT CONTROL MODULE OVERVIEW Flowchart
N
Perform STATUS INPUTS POLL TASK. Refer to STATUS INPUTS MODULE OVERVIEW Flowchart.
Perform ANALOG INPUT POLL TASK. Refer to ANALOG INPUT MODULE OVERVIEW Flowchart
Update TIMERS, COUNTERS,RTU Type, Mode etc. PERFORM_BACKGROUND_-TASK

2.3 Global Variables

Type	Variable	Description
integer	DIDO_CARD_SLOT_NO	digital input digital output (6DI8DO) card slot number. This is determined on power up and the background.
integer	DIAI_CARD_SLOT_NO	digital input analog input (8DI4AI) card slot number. This is determined on power up and in the background.
16 byte	DIG_IN_SENSE_TIMER_TAB	This is the digital input sense array timeout table. Each byte in the array represents the timeout status of the digital inputs. If a digital input has had a timeout, then the corresponding byte in the table is set to 1. It is up to the word using the timer to reset this bit. The allocation of the table is as follows:

array index	allocation
0	Digital input/output card - input 1
1	Digital input/output card - input 2
2	Digital input/output card - input 3
3	Digital input/output card - input 4
4	Digital input/output card - input 5
5	Digital input/output card - input 6
6 - 7	Not used
8	Digital input/Analog input card - input 1
9	Digital input/Analog input card - input 2
10	Digital input/Analog input card - input 3
11	Digital input/Analog input card - input 4
12	Digital input/Analog input card - input 5
13	Digital input/Analog input card - input 6
14	Digital input/Analog input card - input 7
15	Digital input/Analog input card - input 8

Type	Variable	Description
integer	CON_ACTIVE	flag to indicate that a control output operation is in progress. This flag is used in the RTU_TASK_SCHEDULER word to continue calling of the PROCESS_CONT_OUTPUTS word during the activation period. The flag is used and set in the PROCESS_CONT_OUTPUT word.
array of 4 integers	ANA_UPLOAD_TIME	used to hold the analog upload period for each of the 4 analog inputs. The values in this array is decremented very 10 seconds by the PERFORM_BACKGROUND_TASK word.

array of 16 integers : SENSE_TIMER_TAB :used to store the digital input sense timers. These timers (16) are decremented every 100 mS in the PERFORM_BACKGROUND_TASK [3.4.3.3] word.

integer : DIG_IN_MASK : the bits in this variable is used to represent the mask status of each digital. The mask bits for the inputs correspond to the inputs bits in the Status Inputs DIT register. A bit value of 0 means that the respective input is masked, whilst a bit value of 1 implies otherwise. When an input is masked, its status may not be checked or written into the DIT.

integer : BATT_TIMER : this variable will hold the battery settling time value.

integer : BATT_SAMPLE_TIMER : holds the battery sampling period timeout.

2.4 RTU Task Scheduler Module

2.4.1 Overview

The main word in this module is RTU TASK SCHEDULER. The functionality of this word is depicted in the RTU Application Overview Flowchart.

2.4.2 FORTH Word

2.4.2.1 RTU_TASK_SCHEDULER

Interface:

RTU_TASK_SCHEDULER

Stack: >>>

Description:

This is the main task in the RTU Application Software. This word will perform the following fuctions:

(a) initialisation of RTU
(b) RTU periodic self tests
(c) RTU alive poll
(d) update timers
(e) update configuration parameters in eeprom
(f) perform control outputs
(g) update input statuses
(h) update analogs
(i) perform test functions

2.4.2.2 RTU_INITIALISATION

Interface:

RTU_INITIALISATION

Stack: >>>

Description:

This routine will initialise application variables, DIT registers and transmit a power-up message to the central controller. The word will update the Status Code DIT register with the RTU Type (UPDATE_RTU_TYPE) and mode (UPDATE_RTU_MODE). The Status Inputs as well as the Analog Inputs will be updated.

The word will also check for the existance of a personality module and the transfer of configuration data from EEPROM to the DIT (EERAM_INIT).

DIT registers 80 to 87 (refer to the RTU Application Functional Specification) will be transmitted to the central controller as the power up message.

2.4.2.3 **PERFORM_BACKGROUND_TASK**

Interface:

PERFORM_BACKGROUND_TASK

Stack: >>>

Description:

Used to carry out background tasks such as updating of timers, RTU type and mode, updating LED indications etc.

The background task will keep the following timers:

- 10 second timer and at the end of every 10 seconds will decrement every value
  greater than 0 in the ANA_UPLOAD_TIME array.

- minute timer and will decrement the ALIVE_POLL_TIMEOUT and BATT_SAMPLE_TIMER
  variable every minute.

- 100 ms timer and will decrement every time value greater than 0 in the
  SENSE_TIMER_TAB every 100 ms.

- 1 second timer and will decrement BATT_TIMER every second if it has a value
  greater than 0.

This word will also check regularly for changes in the configuration data and will back these up in EEPROM whenever a change in configuration occurs.

2.4.2.4 **UPDATE_RTU_TYPE**

Interface:

UPDATE_RTU_TYPE

Stack: >>>

Description:

This word is used to update the RTU type field in the Status Code DIT register. This is done by reading the card ID for the cards in slots 1 and 2 of the RTU control module electronics. The RTU Type is determined as follows:

minimal - 6 digital input card (6DI) in slot 1, no card in slot 2.

basic - 6 digital input/2 digital output (6DI2DO) card in slot 1, no card in slot 2.

expanded - 6 digital input/8 digital output (6DI8DO) card in slot 1, 8 digital input/4 analog input (8DI4AI) card in slot 2.

invalid - any other combination.

2.4.2.5 **UPDATE_RTU_MODE**

Interface:

UPDATE_RTU_MODE

Stack: >>>

Description:

This word is used to update the RTU mode field in the RTU Status Code Register. This is achieved by reading digital inputs 5 and 6 on the digital input/output card for lockout mode and manual mode respectively. The determination of mode is achieved as follows:

normal - input 5 open , input 6 open and not in test mode

manual - input 5 open , input 6 closed

lockout - input 5 closed, input 6 open

test - input 5 closed, input 6 closed

2.4.2.6 **DI_CARD_TEST**

Interface:

DI_CARD_TEST

Stack: >>>

Description:

This word will check for the existance of a digital input/output card and a digital input/analog input card. If any of these cards are detected, the following action will be taken:

(a) the digital inputs on the card will be forced to the closed state and the software will verify that the all the inputs are closed.

(b) the digital inputs on the card will be forced to the open state and the software will verify that all the inputs on the respective card are open.

Should any of the checks in (a) or (b) fail, then a digital input error is flagged in the Status Code DIT register and a register change of state message of the Status Code register is generated using the word TX_REG_COS.

2.4.2.7 **ALIVE_POLL**

Interface:

ALIVE_POLL

Stack: >>>

Description:

Is used to transmit DIT registers 80 to 87 to the central controller based on a timeout after the last message sent to the central controller. The timeout used is derived from the ALIVE POLL PERIOD in the DIT.

2.4.2.8 **EERAM_INIT**

Interface:

EERAM_INIT

Stack: >>>

Description:

This word is used during the power-up initialisation of the RTU for verification of the personality module (EEPROM).

The following checks are carried out on the personality module:

(a) the personality module has to be plugged in. The CPU-OK LED will flash twice every second if a personality module is not plugged in.

(b) the checksum in the personality module must be correct. If the checksum cannot be verified, the CPU-OK LED will flash thrice every second.

(c) the RTU or LCU address in the personality module must be non-zero. If any of these addresses are zero, then the CPU-OK LED will flash four times every second.

The CPU-OK LED will be turned on permanantly once the conditions above have been fullfilled. The word will transfer all configuration data from the personality module to the RTU DIT. Also, the RTU application will not run until checks (a) to (c) have passed.

2.4.2.9 **EEUPDATE**

Interface:

EEUPDATE

Stack: >>>

Description:

This word will constantly monitor any changes in the RTU DIT configuration. This word will back any changes that occur in the configuration, in the RTU DIT.
The word will also check for removal of the personality module and will perform the same functions as indicated in EERAM_INIT above, if the personality module is re-inserted. The RTU Status code register will be uploaded to the central controller on removal and re-insertion of the personality module.

2.5 COMMS Module

2.5.1 Overview

This module consists of a suite of FORTH words to handle communications between the network kernel and the application. Included are words to read and write to the presentation layer DIT service in the network kernel.

2.5.2 FORTH Words

2.5.2.1 TX_BIT_COS

Interface:

TX_BIT_COS

Stack: State_of_bit BIT_number DIT_register_no >>>

Description:

This word is used to queue bit change of states (w.r.t DIT registers) in a circular queue that is 10 messages deep. The messages in the queue will be transmitted when the network becomes free. When the queue is full, the newest data will overwrite the oldest data. The word PROC_COMMS will be used to retrieve messages from the queue for transmission to the central controller.

The value of "bit position" ranges from 0 to 15 where 0 is the least significant bit position and 15 is the most significant bit position in the register.

"State of bit" will be the value of the bit ie. 1 or 0.

2.5.2.2 TX_REG_COS

Interface:

TX_REG_COS

Stack: DIT_register_value DIT_register_no >>>

Description:

This word is used to queue register change of states (w.r.t DIT registers) in a circular queue that is 10 messages deep. The messages in the queue will be transmitted when the network becomes free. When the queue is full, the newest data will overwrite the oldest data. The word PROC_COMMS will be used to retrieve messages from the queue for transmission to the central controller.

2.5.2.3 DIT_WRITE

Interface:

DIT_WRITE

Stack: DIT_register_value DIT_register_number >>>

Description:

This word is used to write a 16 bit value into an RTU DIT register.

2.5.2.4 DIT_READ

Interface:

DIT_WRITE

Stack: DIT_register_number >>> DIT_register_value

Description:

This word is used to read the value of an RTU DIT register.

2.5.2.5 CONTROL_OPER_CALL

Interface:

CONTROL_OPER_CALL

Stack: >>> n1 n2 n3

```
where n3 = 0 only : no control message
                      or
      n3 = 1 : control message arrived
                     and
      n2 =  Control output number
      n1 =  Type of Operation:  255 = close
                                  0 = open
```

Description:

This word is used to check if a control output command message has arrived at the RTU. After execution of this word, a 0 may be returned on the stack to indicate that no control message has arrived, or a 1 to indicate that a control message has arrived. If a 1 is returned, then the control output number and the type of operation is also returned.

2.5.2.6 **PROC_HW_ERROR**

Interface:

PROC_HW_ERROR

Stack: error_number >>>

Description:

This word will set the RTU Status Code register with the error number passed in and will generate a register change of state (TX_REG_COS) message of the Status Code Register to the central controller.

2.5.2.7 **PROC_COMMS**

Interface:

PROC_COMMS

Stack: >>>

Description:

This word will poll the network semaphore register and check if the network is free for transmission. If transmission can be made, the word will then transmit messages form the register and bit change of state queues alternately.

2.6 Status Inputs Poll Module

2.6.1 Overview

The Status Inputs Poll module contains a suite of words used in processing the digital inputs of the RTU and checking the state of the power supply unit.

The two main words in this module are : PROCESS_STATUS_INPUTS and PROCESS_BATTERY_STATUS. Refer to the "Status Inputs Module Overview Flowchart.

The PROCESS_STATUS_INPUTS (refer to the Read Status Inputs Overview Flowcharts A and B) word is responsible for updating the DIT with the latest digital input states and uploading any change of state messages to the central controller.

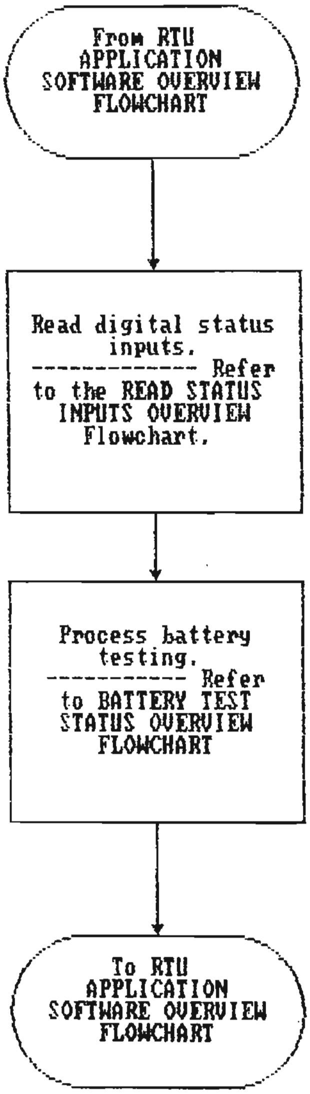
The PROCESS_BATTERY_STATUS (refer to the Battery Test Status Overview Flowcharts A and B) word is used to check the primary power supply status in the Status Inputs DIT register and to check the battery status.

The primary power status is retrieved by reading the primary power status from the power supply unit. The primary power status is a bit that will be set to 1 by the PSU if there is primary power and is reset to 0 if there is a primary power failure.

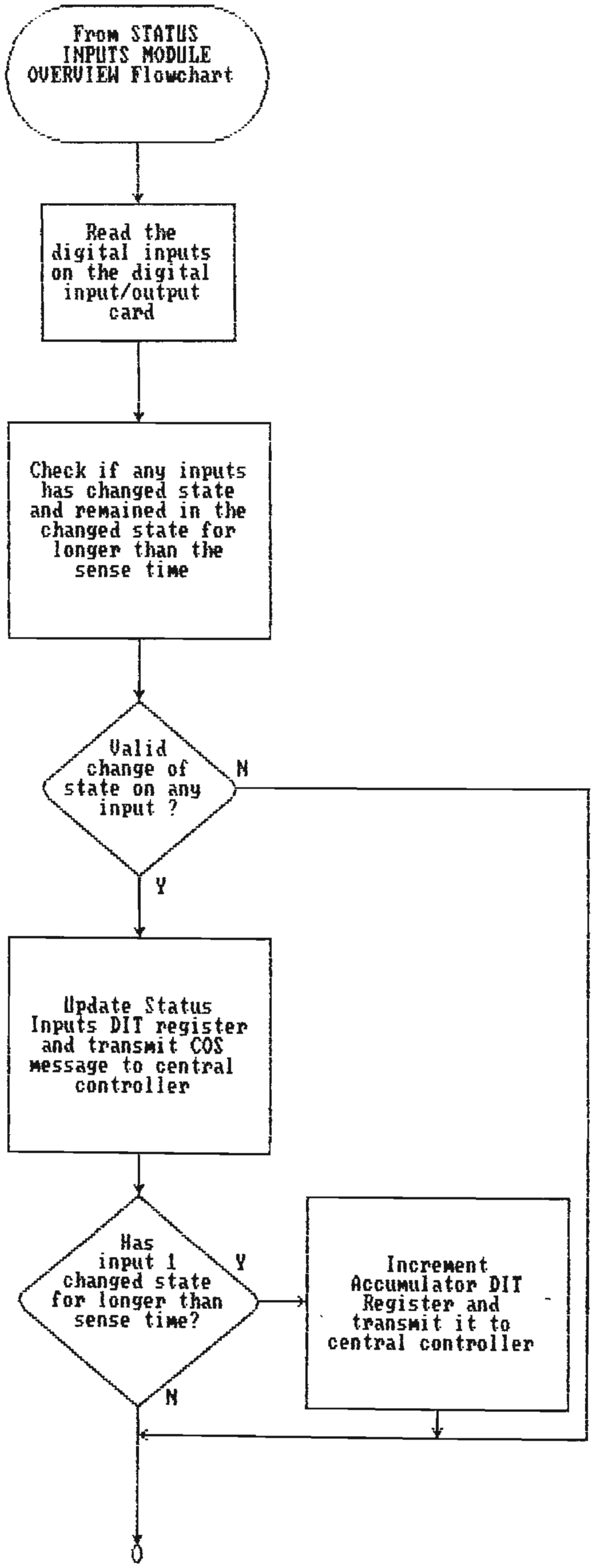
The battery status is retrieved by performing a battery test on a fixed time basis (sampling period). This period is reset to the sampling period timeout at the end of each battery test, so that a new battery test may be performed. At the termination of a sampling timeout period, the battery charger is turned off so that the RTU sources its power from the battery only. Due to the sudden removal of the charger, the battery requires a short period to stabilise its voltage. At the end of this timeout period (battery settling timeout) the battery voltage is retrieved. The battery voltage is retrieved by initiating an analog to digital conversion of the battery voltage, delaying the conversion timeout period and reading the A/D value. The A/D value of the battery voltage is compared with a threshold value. If it is lower than the threshold value, then the battery level bit in the Status Inputs DIT register is set to 0. If it is higher than the threshold value, then the bit is set to 1 to indicate a battery normal condition.

All flowcharts relating to the status inputs poll module start from the next page for easy reference. The flowcharts are arranged in hierachical order.

STATUS INPUTS MODULE OVERVIEW FLOWCHART

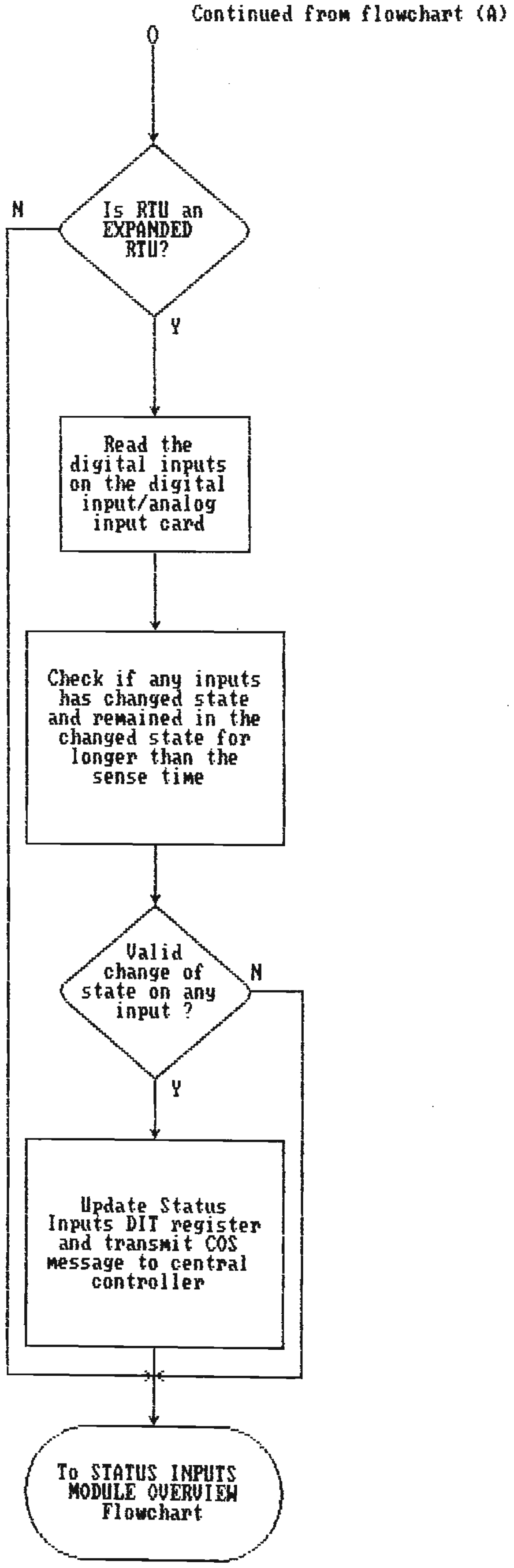

From RTU APPLICATION SOFTWARE OVERVIEW FLOWCHART
Read digital status inputs. ------------- Refer to the READ STATUS INPUTS OVERVIEW Flowchart.
Process battery testing. ----------- Refer to BATTERY TEST STATUS OVERVIEW FLOWCHART
To RTU APPLICATION SOFTWARE OVERVIEW FLOWCHART

READ STATUS INPUTS OVERVIEW FLOWCHART (A) (Refer to PROCESS_STATUS_INPUTS Flowchart)

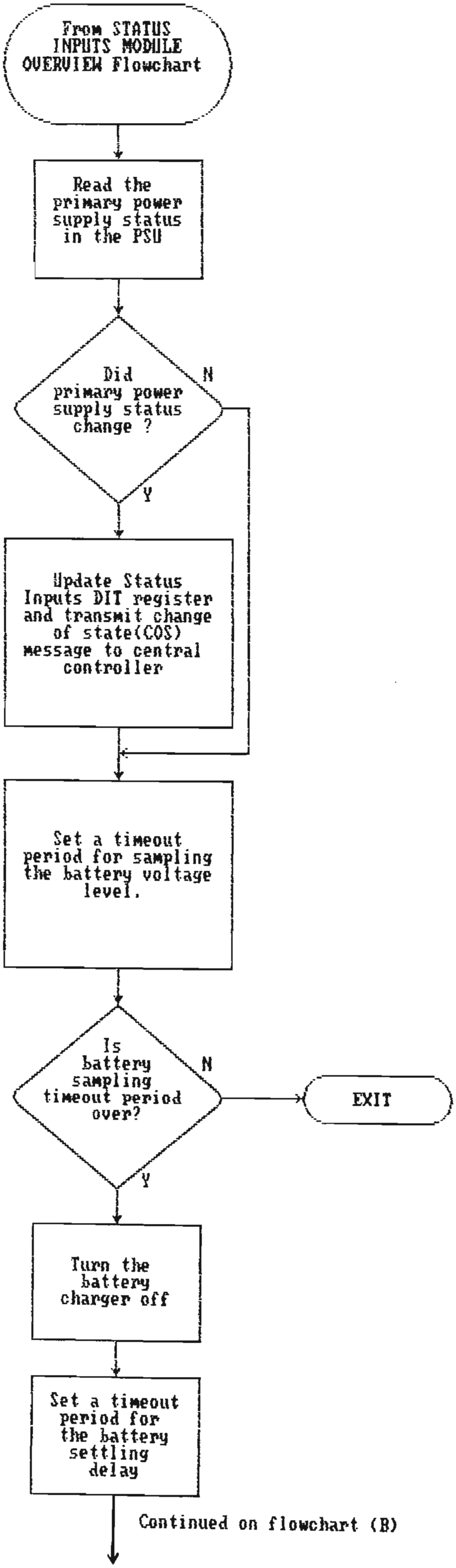

From STATUS INPUTS MODULE OVERVIEW Flowchart
Read the digital inputs on the digital input/output card
Check if any inputs has changed state and remained in the changed state for longer than the sense time
Valid change of state on any input ?
N
Y
Update Status Inputs DIT register and transmit COS message to central controller
Has input 1 changed state for longer than sense time?
Y
N
Increment Accumulator DIT Register and transmit it to central controller
0

Continued on flowchart (B)

READ STATUS INPUTS OVERVIEW FLOWCHART (B)


Continued from flowchart (A)
0
Is RTU an EXPANDED RTU?
N
Y
Read the digital inputs on the digital input/analog input card
Check if any inputs has changed state and remained in the changed state for longer than the sense time
Valid change of state on any input ?
N
Y
Update Status Inputs DIT register and transmit COS message to central controller
To STATUS INPUTS MODULE OVERVIEW Flowchart

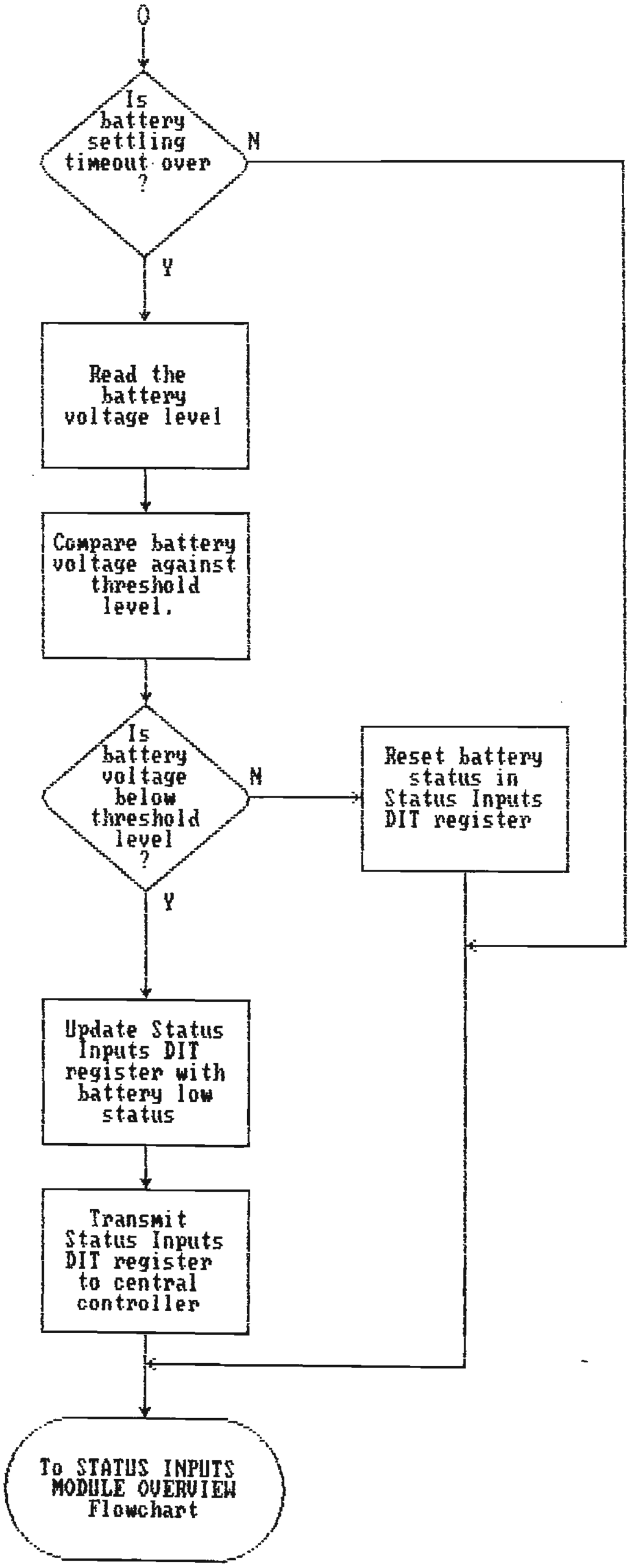
BATTERY TEST STATUS OVERVIEW FLOWCHART (A) (Refer to PROCESS_BATTERY_STATUS Flowchart)


From STATUS INPUTS MODULE OVERVIEW Flowchart
Read the primary power supply status in the PSU
Did primary power supply status change ?
N
Y
Update Status Inputs DIT register and transmit change of state(COS) message to central controller
Set a timeout period for sampling the battery voltage level.
Is battery sampling timeout period over?
N
EXIT
Y
Turn the battery charger off
Set a timeout period for the battery settling delay

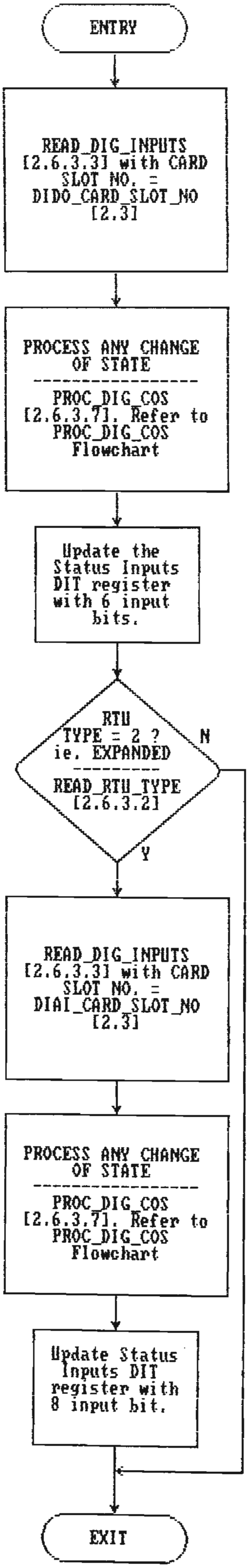
Continued on flowchart (B)

BATTERY TEST STATUS OVERVIEW FLOWCHART (B)

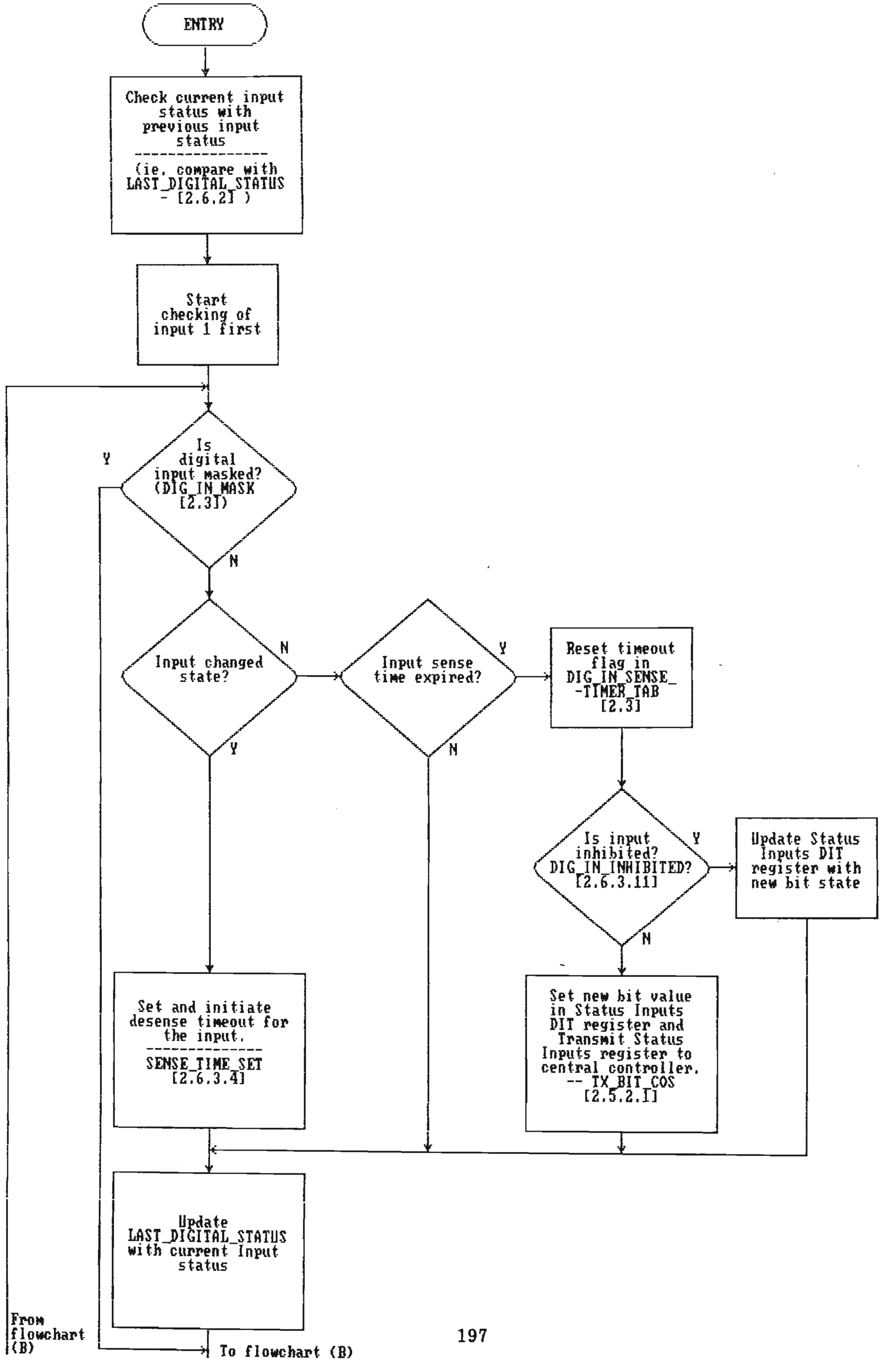
Continued from flowchart (A)


0
Is battery settling timeout over ?
N
Y
Read the battery voltage level
Compare battery voltage against threshold level.
Is battery voltage below threshold level ?
N
Y
Reset battery status in Status Inputs DIT register
Update Status Inputs DIT register with battery low status
Transmit Status Inputs DIT register to central controller
To STATUS INPUTS MODULE OVERVIEW Flowchart

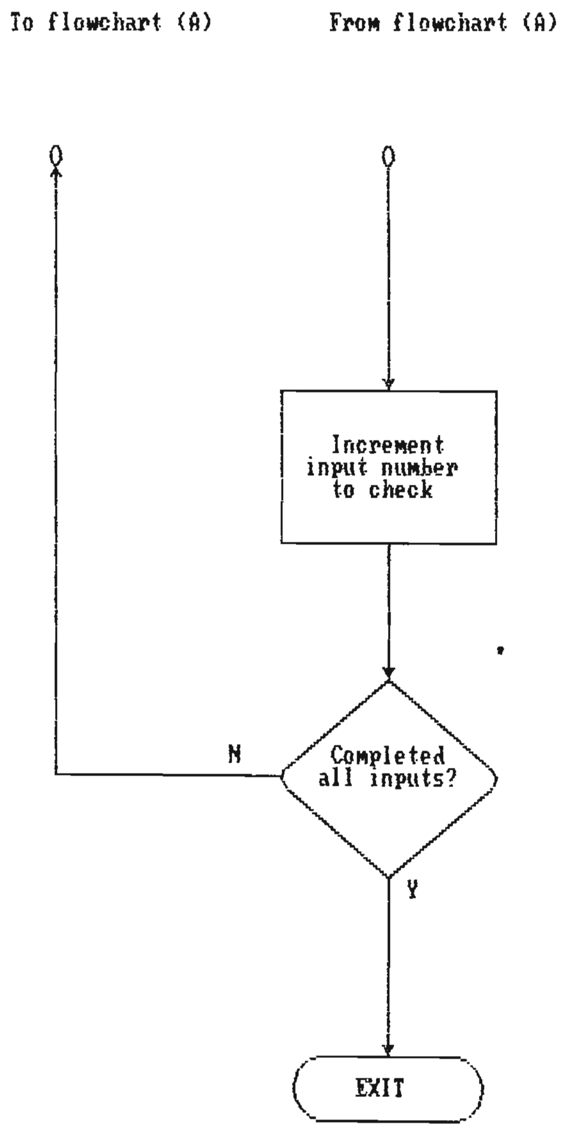
PROCESS_STATUS_INPUTS FLOWCHART


ENTRY
READ_DIG_INPUTS [2.6.3.3] with CARD SLOT NO. = DIDO_CARD_SLOT_NO [2.3]
PROCESS ANY CHANGE OF STATE
PROC_DIG_COS [2.6.3.7]. Refer to PROC_DIG_COS Flowchart
Update the Status Inputs DIT register with 6 input bits.
RTU TYPE = 2 ? ie. EXPANDED
READ_RTU_TYPE [2.6.3.2]
N
Y
READ_DIG_INPUTS [2.6.3.3] with CARD SLOT NO. = DIAI_CARD_SLOT_NO [2.3]
PROCESS ANY CHANGE OF STATE
PROC_DIG_COS [2.6.3.7]. Refer to PROC_DIG_COS Flowchart
Update Status Inputs DIT register with 8 input bit.
EXIT

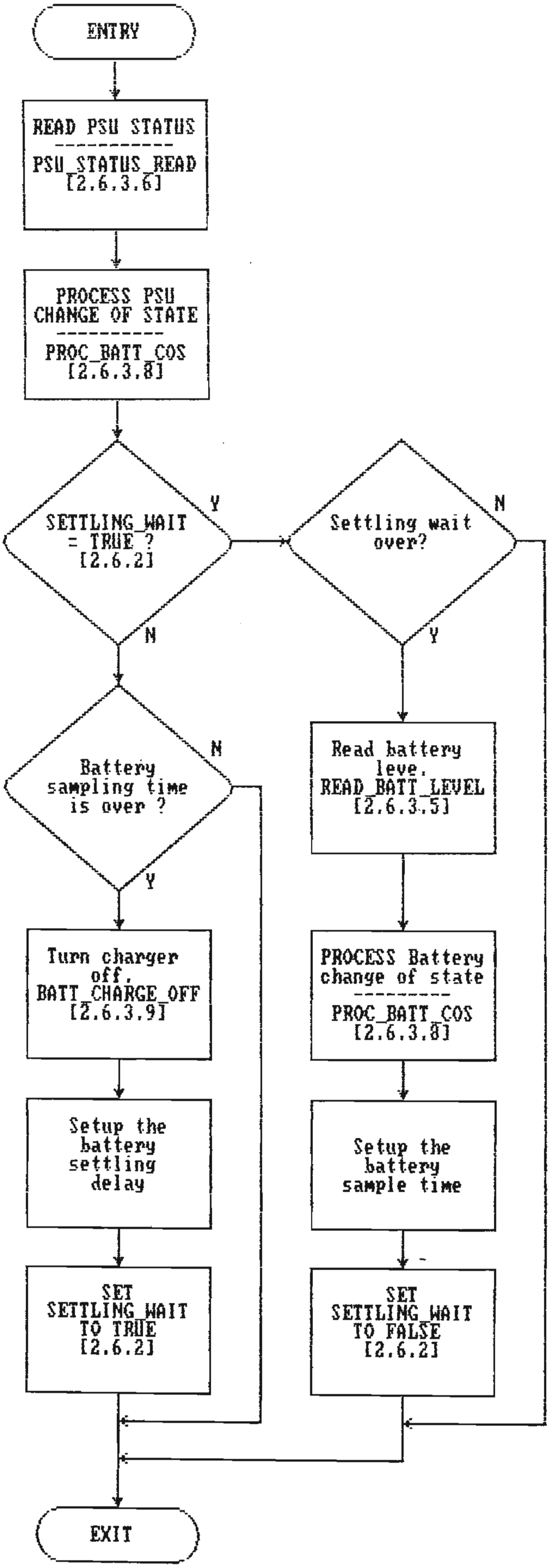
PROC_DIG_COS FLOWCHART (A) [2.6.3.7]


ENTRY
Check current input status with previous input status
(ie. compare with LAST_DIGITAL_STATUS - [2.6.2])
Start checking of input 1 first
Is digital input masked? (DIG_IN_MASK [2.3])
Y
N
Input changed state?
N
Y
Input sense time expired?
Y
N
Reset timeout flag in DIG_IN_SENSE_-TIMER_TAB [2.3]
Is input inhibited? DIG_IN_INHIBITED? [2.6.3.11]
Y
N
Update Status Inputs DIT register with new bit state
Set and initiate desense timeout for the input.
SENSE_TIME_SET [2.6.3.4]
Set new bit value in Status Inputs DIT register and Transmit Status Inputs register to central controller.
-- TX_BIT_COS [2.5.2.1]
Update LAST_DIGITAL_STATUS with current Input status
From flowchart (B)
To flowchart (B)

PROC_DIG_COS FLOWCHART (B) [2.6.3.7]



PROCESS_BATTERY_STATUS FLOWCHART



2.6.2 **Variables**

Type	Variable	Description
integers	LAST_DIGITAL_STATUS	holds digital status of last scan. Used in determining bit change of states.
integer	SETTLING_WAIT	flag to indicate that the module is in the process of waiting for the to settle after the charger is turned off.

2.6.3 **FORTH Words**

2.6.3.1 **PROCESS_STATUS_INPUTS**

Interface:

PROCESS_STATUS_INPUTS

Description:

This word will process all digital input statuses. It will update the DIT with the latest input status after subjecting all inputs to the desense period and generate bit change of state messages depending on whether a bit has changed state.

2.6.3.2 **READ_RTU_TYPE**

Interface:

READ_RTU_TYPE

Stack: >>> n1

where n1		
where n1	= 0	: minimal
	= 1	: basic
	= 2	: expanded
	= 7	: invalid type

Description:

This word is used to obtain the RTU type. This is achieved by performing a DIT read of the RTU Status Code register and extracting the relevent information.

2.6.3.3 **READ_DIG_INPUTS**

Interface:

READ_DIG_INPUTS

Stack: n1 >>> n2

where n1 = card slot number

n2 = input statuses

Description:

This word is used to retrieve the current statuses of the digital inputs on the digital input card at the specified slot. The bits in the number returned represent the status of the various inputs. A bit value of 1 represents a closed state whilst a bit value of 0 represents an open state. The least significant bit of the number returned represents input 1 on the input card addressed. Consecutive bits represents consecutive inputs.

2.6.3.4 **SENSE_TIME_SET**

Interface:

SENSE_TIME_SET

Stack: n1 >>>

where n1 = input_number 0 - 15

Description:

Used to set the digital input sensing time for a change of state of a digital input. The value for the digital input sense time is derived from the DIT.

The word will index the SENSE_TIMER_TAB array with the parameters passed in and store the respective sense time in the respective cell. The sense timeout values in this table will be automatically decremented in the PERFORM_BACKGROUND_TASK word.

2.6.3.5 READ_BATT_LEVEL

Interface:

READ_BATT_LEVEL

Stack: >>> n1

```
where  n1 = 0 : battery low
          = 1 : battery ok
```

Description:

This word is used to determine the status of the battery. The word will perform the following sequence of events:

(a) write any value to register 3 of the power supply to initiate an A/D conversion of the battery voltage.

(b) delay 150 µs for A/D conversion delay.

(b) read register 3 of power supply for battery voltage.

(c) if battery voltage is less than a predefined minimum, then return 0 to indicate a battery low condition else return 1 to indicate a battery ok condition.

Note that the power supply will always be located in slot 1.

2.6.3.6 PSU_STATUS_READ

Interface:

PSU_STATUS_READ

Stack: >>> n1

```
where n1 = 0 : primary power fail
         = 1 : primary power ok
```

Description:

Used to determine if the primary power supply has failed or not. The word will read the general status register (register 2) of the power supply in order to determine this. Bit 1 of this register when set to 1 will indicate "primary power supply ok" whilst a bit 1 value of 0 will indicate a "primary power supply fail" condition.

2.6.3.7 **PROC_DIG_COS**

Interface:

PROC_DIG_COS

Stack: n1 n2 n3 >>>

```
where n1 = digital input status
      n2 = 0 : digital input/output card
         = 1 : digital input/analog input card
      n3 = input number
```

Description:

This word is used to process a change of state of all digital inputs corresponding to the type of card passed in.

Refer to the PROC_DIG_COS Flowchart.

The word will go through each input in turn, checking for a change of state. It will also check if the input being checked is masked or not. If it is masked (checked using variable DIG_IN_MASK [3.3]), then the input is ignored and the next input checked.

The word will retrieve the old statuses from the variable LAST_DIGITAL_STATUS [3.6.2] and compare this with the current statuses passed in. If the word detects a change of state of an input, it will check the DIG_IN_SENSE_TIMER_TAB [3.3] table to determine if a timeout on a change of state has occurred. If a change of state on a digital input has gone through its sense time, then the Status Inputs DIT register is updated with the new change of state and the DIT register is uploaded to the central controller. At the same time, the sense timeout flag in the DIG_IN_SENSE_TIMER_TAB is reset.

If the word detects that an input has changed state, then it sets the sense timeout for that input in the SENSE_TIMER_TAB. The timers in this table is decremented every 100 ms. The sense timeout period for an input is derived from the DIT.

2.6.3.8 **PROC_BATT_COS**

Interface:

PROC_BATT_COS

Stack: n1 n2 >>>

```
where n1 = bit state
      n2 = bit number : 6 for primary power supply status
                      : 7 for battery level status
```

Description:

This word is used to compare the bit passed in with the corresponding bit status in the Status Inputs DIT register. If there is a discrepancy, then the Status Inputs register is updated with the new bit value and uploaded to the cenral controller.

2.6.3.9 BATT_CHARGE_OFF

Interface:

BATT_CHARGE_OFF

Stack: >>>

Description:

Used to turn the battery charger off in the power supply unit. This is achieved by writing any value to the register 1 in the PSU.

2.6.3.10 PROCESS_BATTERY_STATUS

Interface:

PROCESS_BATTERY_STATUS

Stack: >>>

Description:

This word will check the primary power supply status as well as the status of the battery. It will update the Status Inputs DIT register with the respective statuses. Should a change of state be detected, the word will upload the Status Inputs DIT register to the central controller, provided that the input is not inhibited or masked.

Refer to the PROCESS_BATTERY_STATUS flowchart.

2.6.3.11 DIG_IN_INHIBITED?

Interface:

DIG_IN_INHIBITED?

```
  Stack:  n1  >>> n2

  where n1 = digital input number

        n2 = 1 : digital input is inhibited
           = 0 : digital input not inhibited
```

Description:

This word will check whether the digital input as per digital input number passed in is inhibited or not. The word will determine this by checking the Status Inputs Inhibit DIT register.

2.7 Analog Input Poll Module

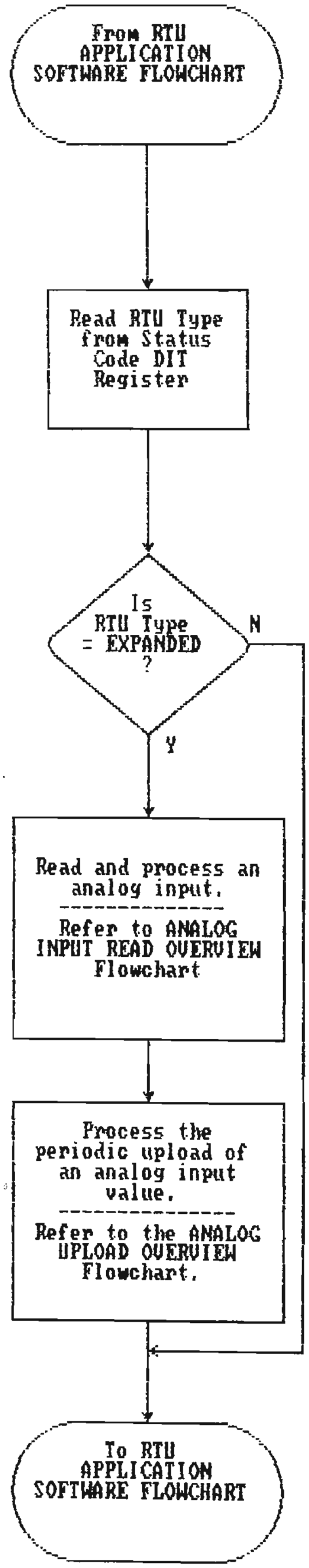
2.7.1 Overview

The analog input poll module contains a suite of words to process the acquisition of analog data, checking of out of limit conditions and return to normal conditions, invoking automatic upload (analog poll to central controller) of analog information to central controller and checking of malfunction in the analog to digital conversion circuitry.

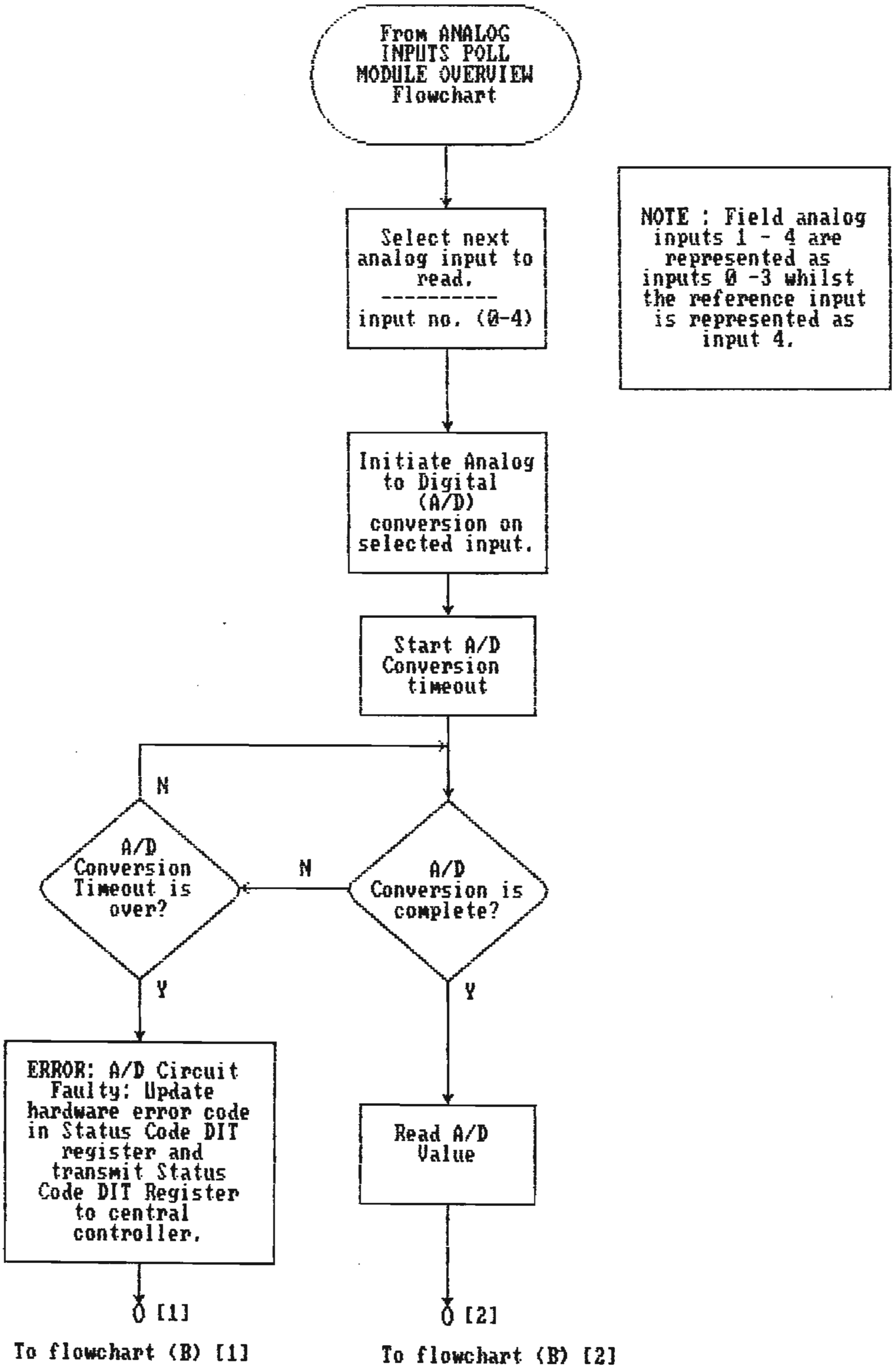
The main words in this module is PROCESS_ANALOG_INPUTS and PROCESS_ANALOG_UPLOAD. These word are called in every scan of the RTU Task Scheduler.

Refer to the Analog Inputs Poll Module Overview flowchart, the Analog Input Read Overview Flowchart and the Analog Upload Overview Flowchart. All flowcharts relating to the analog input poll module are grouped together starting on the next page, to make referencing easier. The flowcharts are grouped together in hierachical order.

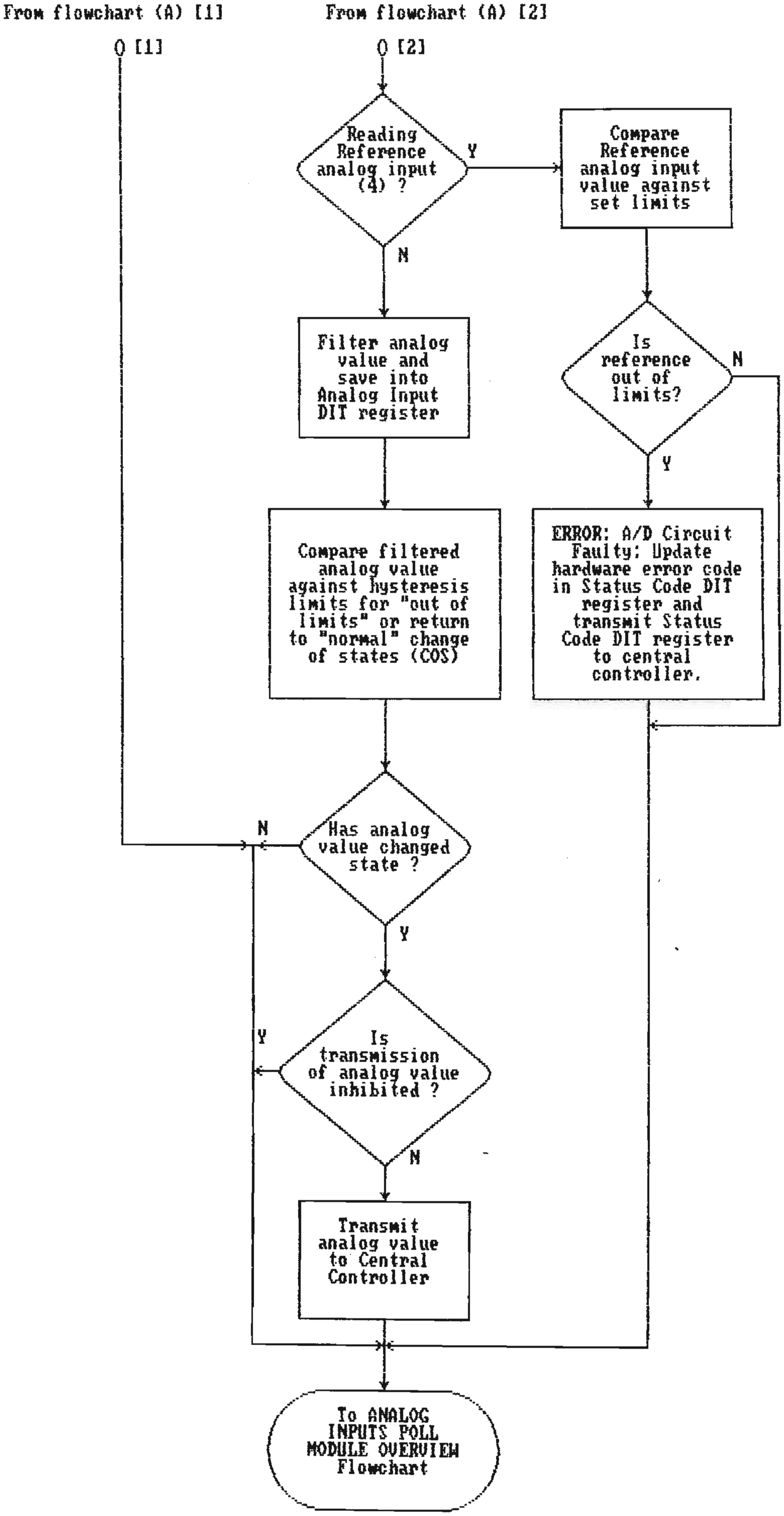
ANALOG INPUTS POLL MODULE OVERVIEW FLOWCHART


From RTU APPLICATION SOFTWARE FLOWCHART
Read RTU Type from Status Code DIT Register
Is RTU Type = EXPANDED ?
N
Y
Read and process an analog input.
Refer to ANALOG INPUT READ OVERVIEW Flowchart
Process the periodic upload of an analog input value.
Refer to the ANALOG UPLOAD OVERVIEW Flowchart.
To RTU APPLICATION SOFTWARE FLOWCHART

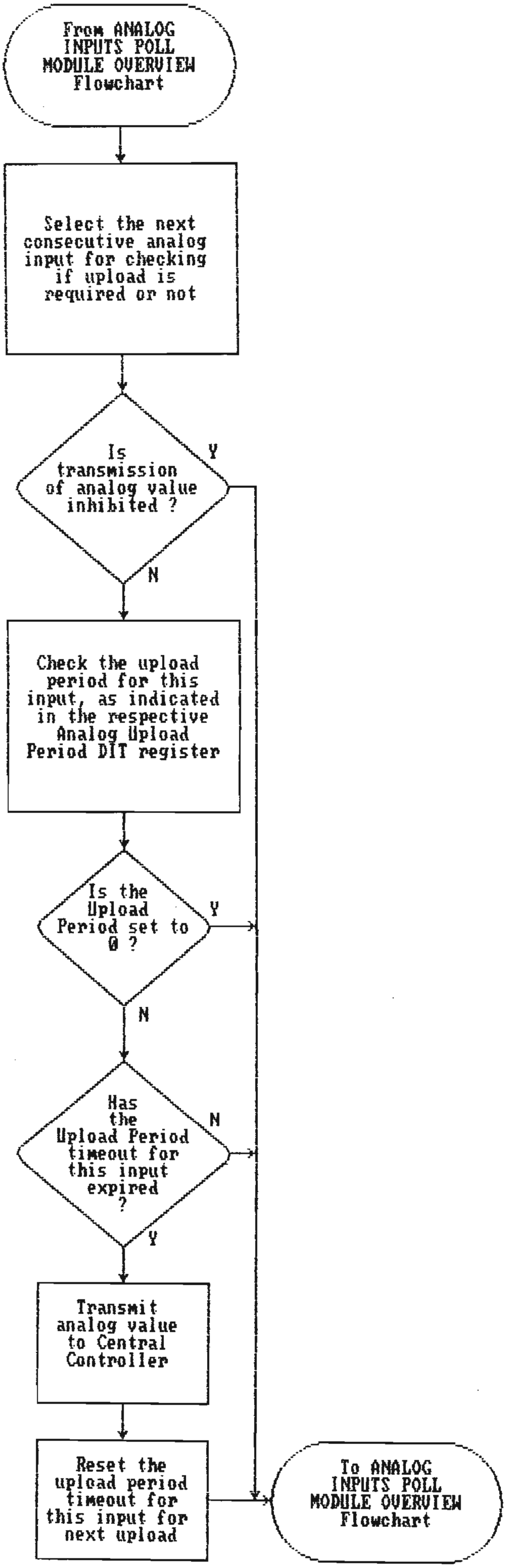
ANALOG INPUT READ OVERVIEW FLOWCHART (A) (Refer to PROCESS_ANALOG_INPUTS Flowchart)


From ANALOG INPUTS POLL MODULE OVERVIEW Flowchart
Select next analog input to read.
----
input no. (0-4)
NOTE : Field analog inputs 1 - 4 are represented as inputs 0 -3 whilst the reference input is represented as input 4.
Initiate Analog to Digital (A/D) conversion on selected input.
Start A/D Conversion timeout
N
A/D Conversion Timeout is over?
N
A/D Conversion is complete?
Y
Y
ERROR: A/D Circuit Faulty: Update hardware error code in Status Code DIT register and transmit Status Code DIT Register to central controller.
Read A/D Value
[1]
[2]
To flowchart (B) [1]
To flowchart (B) [2]

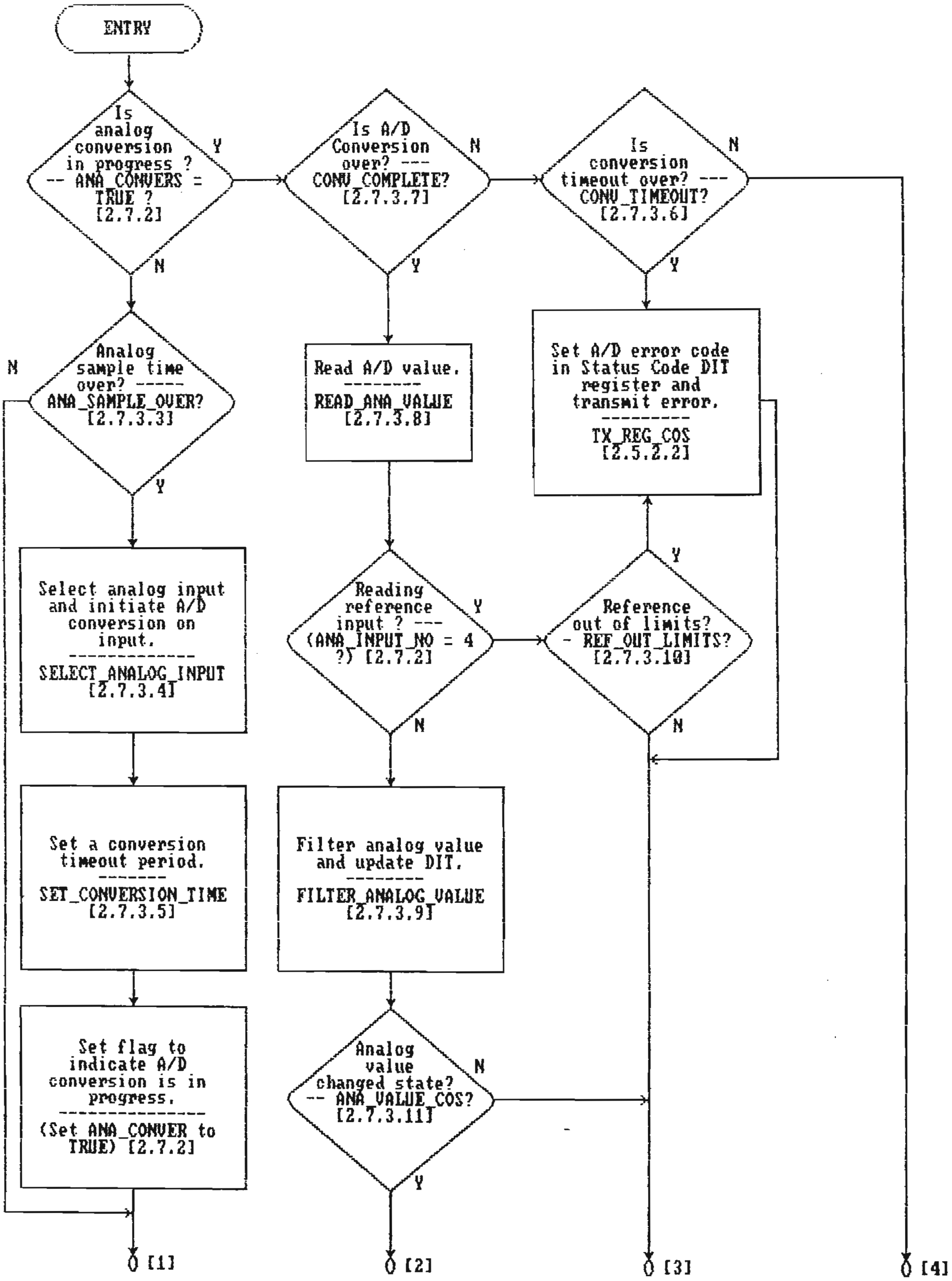
ANALOG INPUT READ OVERVIEW FLOWCHART (B)


From flowchart (A) [1]
From flowchart (A) [2]
O [1]
O [2]
Reading Reference analog input (4) ?
Y
N
Compare Reference analog input value against set limits
Filter analog value and save into Analog Input DIT register
Is reference out of limits?
N
Y
Compare filtered analog value against hysteresis limits for "out of limits" or return to "normal" change of states (COS)
ERROR: A/D Circuit Faulty: Update hardware error code in Status Code DIT register and transmit Status Code DIT register to central controller.
N
Has analog value changed state ?
Y
Y
Is transmission of analog value inhibited ?
N
Transmit analog value to Central Controller
To ANALOG INPUTS POLL MODULE OVERVIEW Flowchart

ANALOG UPLOAD OVERVIEW FLOWCHART (Refer to PROCESS_ANALOG_UPLOAD Flowchart)


From ANALOG INPUTS POLL MODULE OVERVIEW Flowchart
Select the next consecutive analog input for checking if upload is required or not
Is transmission of analog value inhibited ?
Y
N
Check the upload period for this input, as indicated in the respective Analog Upload Period DIT register
Is the Upload Period set to 0 ?
Y
N
Has the Upload Period timeout for this input expired ?
N
Y
Transmit analog value to Central Controller
Reset the upload period timeout for this input for next upload
To ANALOG INPUTS POLL MODULE OVERVIEW Flowchart

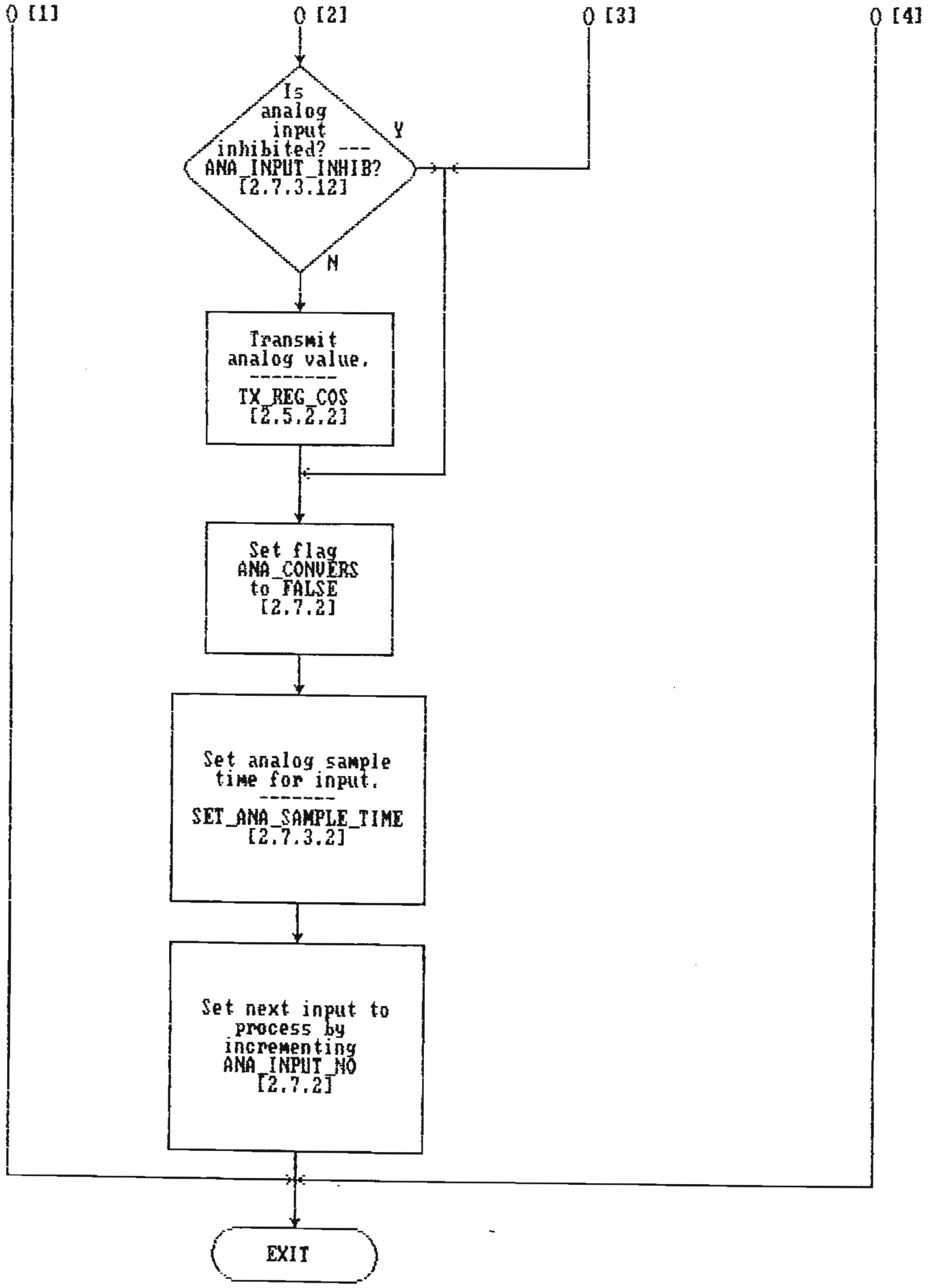
PROCESS_ANALOG_INPUTS FLOWCHART (A)


ENTRY
Is analog conversion in progress ? -- ANA_CONVERS = TRUE ? [2.7.2]
Y
N
Is A/D Conversion over? --- CONV_COMPLETE? [2.7.3.7]
N
Y
Is conversion timeout over? --- CONV_TIMEOUT? [2.7.3.6]
N
Y
Analog sample time over? ----- ANA_SAMPLE_OVER? [2.7.3.3]
N
Y
Read A/D value. --------- READ_ANA_VALUE [2.7.3.8]
Set A/D error code in Status Code DIT register and transmit error. --------- TX_REG_COS [2.5.2.2]
Select analog input and initiate A/D conversion on input. ------------- SELECT_ANALOG_INPUT [2.7.3.4]
Reading reference input ? --- (ANA_INPUT_NO = 4 ?) [2.7.2]
Y
N
Reference out of limits? - REF_OUT_LIMITS? [2.7.3.10]
Y
N
Set a conversion timeout period. -------- SET_CONVERSION_TIME [2.7.3.5]
Filter analog value and update DIT. --------- FILTER_ANALOG_VALUE [2.7.3.9]
Set flag to indicate A/D conversion is in progress. ---------------- (Set ANA_CONVER to TRUE) [2.7.2]
Analog value changed state? -- ANA_VALUE_COS? [2.7.3.11]
N
Y
[1]
[2]
[3]
[4]

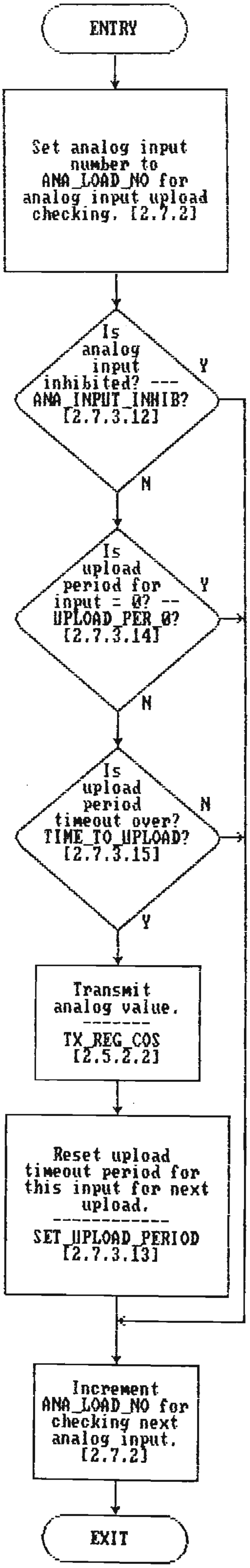
To respective points on flowchart (B)

PROCESS_ANALOG_INPUTS FLOWCHART (B)

From respective points on flowchart (A)


[1]
[2]
[3]
[4]
Is analog input inhibited? --- ANA_INPUT_INHIB? [2.7.3.12]
Y
N
Transmit analog value. -------- TX_REG_COS [2.5.2.2]
Set flag ANA_CONVERS to FALSE [2.7.2]
Set analog sample time for input. ------- SET_ANA_SAMPLE_TIME [2.7.3.2]
Set next input to process by incrementing ANA_INPUT_NO [2.7.2]
EXIT

PROCESS_ANALOG_UPLOAD FLOWCHART


ENTRY
Set analog input number to ANA_LOAD_NO for analog input upload checking. [2.7.2]
Is analog input inhibited? --- ANA_INPUT_INHIB? [2.7.3.12]
Y
N
Is upload period for input = 0? -- UPLOAD_PER_0? [2.7.3.14]
Y
N
Is upload period timeout over? TIME_TO_UPLOAD? [2.7.3.15]
N
Y
Transmit analog value. ------- TX_REG_COS [2.5.2.2]
Reset upload timeout period for this input for next upload. ------------ SET_UPLOAD_PERIOD [2.7.3.13]
Increment ANA_LOAD_NO for checking next analog input. [2.7.2]
EXIT

2.7.2 Variables

Type	Variable	Description
integer	ANA_INPUT_NO	current analog input selected for A/D conversion. This variable will take on values 0 to 4, representing analog inputs 1 to 4 and the reference input respectively.
integer	ANA_LOAD_NO	current analog input selected for checking if uploading of the respective analog input value is required or not. This variable will take on values of 0 to 3, representing analog inputs 1 to 4 respectively.
constant	ANA_SAMPLE_TIME	Used to setup the analog input sample time in multiples of 10 ms.
integer	ANA_READ	flag to indicate if an analog input read is in progress. A value of 0 = no read in progress, whilst a 1 = analog read in progress.
array of 4 integers	LAST_SMOOTHED_VALUE	holds the last smoothed value for each of the 4 analog inputs.
array of 4 integers	PREV_ANALOG_STATE	holds the last state of the 4 analog inputs. Each state may take on the following values: (a) 0 = return to normal (b) 1 = high high alarm (c) 2 = low low alarm
integer	ANA_CONVERS	flag to indicate status of A/D conversion of an input on the analog input card. If it is set to true (1), then an A/D conversion is still in progress. It is set to false when an A/D conversion has completed.
constant	CONVERSION_TIMEOUT	sets the conversion timeout period for the analog. The value in this constant is a multiple of 10 ms eg. a constant value of 2 provides a conversion timeout of 20 ms. The default value is 1.

2.7.3 **FORTH Words**

2.7.3.1 **PROCESS_ANALOG_INPUTS**

Interface:

PROCESS_ANALOG_INPUTS

Description:

Used to process the 4 analog inputs and the reference input (the reference input is used to determine the condition of the a/d conversion circuitry.

This word will exit on analog sample timeouts and analog to digital conversion time delays. When the word has completed the processing of an analog value, the next analog value is processed in the following scan. Hence the word will continually be processing analog inputs 1 to 4 and the reference input in that order.

Refer to the PROCESS_ANALOG_INPUTS flowchart.

2.7.3.2 **SET_ANA_SAMPLE_TIME**

Interface:

SET_ANA_SAMPLE_TIME

Stack: n1 >>>

where n1 = analog sample time in multiples of 10ms (ANA_SAMPLE_TIME)

Description:

Used to setup the analog sample timer that will time out with the time value passed on the stack.

The analog sample time is the time between reading of one analog input and another. For this application the constant ANA_SAMPLE_TIME is used when calling this word.

Timer 2 of the Eziforth kernel is used to set up this timer.

This word must be used with the word ANA_SAMPLE_OVER? in order to determine the end of this timeout period.

2.7.3.3 **ANA_SAMPLE_OVER?**

Interface:

ANA_SAMPLE_OVER?

```
  Stack:      >>>  n1

  where    n1 = 1 :  analog sample timeout over
              = 0 :  still timing
```

Description:

Used to check termination of the analog sample delay. This word will call Eziforth timer 1 in checking this.

2.7.3.4 **SELECT_ANALOG_INPUT**

Interface:

SELECT_ANALOG_INPUT

```
  Stack:  n1  >>>

  where  n1 = analog input number (0-3) or reference input number (4)
```

Description:

This word will select an analog input for analog to digital conversion and will initiate the conversion process. Selection of the input will occur by writing the respective analog input number into the Analog Input Selection register (register 4) of the digital input/analog input card. The A/D conversion is initated automatically once the input has been selected.

2.7.3.5 **SET_CONVERSION_TIME**

Interface:

SET_CONVERSION_TIME

```
  Stack:     >>>
```

Description:

Used to setup the conversion timeout period for the analog to digital conversion process. The constant CONVERSION_TIMEOUT [2.7.2] is used to set up the conversion timeout. This word must be used in conjunction with the word CONV_TIMEOUT? to determine the end of the timeout period.

2.7.3.6 CONV_TIMEOUT?

Interface:

CONV_TIMEOUT?

```
  Stack:      >>>  n1

  where n1 = 1 : conversion timeout period expired
           = 0 : still timing out
```

Description:

Used to establish the end of the timeout period set by the word SET_CONVERSION_TIME.

2.7.3.7 CONV_COMPLETE?

Interface:

CONV_COMPLETE?

```
  Stack:   >>> n1

  where n1 = 1 : Analog to digital conversion is complete
           = 0 : A/D conversion not yet complete
```

Description:

This word will poll the "Conversion Not Complete" bit in the CNC Check register (register 4) in the analog input card to determine if an A/D conversion has completed or not.

2.7.3.8 READ_ANA_VALUE

Interface:

READ_ANA_VALUE

```
  Stack:   >>>  n1


  where n1 =  A/D value   - 16 bits
```

Description:

This word is used to retrieve the analog to digital conversion value from the digital input/analog input card, after selection and conversion of an analog input.

2.7.3.9 FILTER_ANALOG_VALUE

Interface:

FILTER_ANALOG_VALUE

Stack: n1 n2 >>> n3

```
where  n1 =  current analog value
       n2 =  analog input number  ( 1 - 4 )
       n3 = smoothed analog value
```

Description:

This word will filter the analog value passed in using the algorithm shown below:

$$SD_A = \frac{X_1 \ SD_C + (100 - X_1) \ SD_L}{100}$$

where SD_A = Smoothed analog value

X_1 = Smoothing Factor (range: 0 - 100)

SD_C = Current analog value

SD_L = Last smoothed analog value

The last smoothed analog value is retrieved from the LAST_SMOOTHED_VALUE array. Before exiting this word, the new smoothed analog value is stored in the LAST_SMOOTHED_VALUE array in the respective position.

The smoothing factor is derived from the DIT.

2.7.3.10 REF_OUT_LIMITS?

Interface:

REF_OUT_LIMITS?

Stack: reference_input_value >>> n1

```
where  n1 = 1 : reference out of limits
          = 0 : reference within limits
```

Description:

This word is used to check if the reference input analog value passed in is out of tolerence or not.

2.7.3.11 **ANA_VALUE_COS?**

Interface:

ANA_VALUE_COS?

Stack: n1 n2 >>> n3

where n1 = analog value to be checked for change of state
      n2 = analog input number

n3 = 1 : analog value in alarm or has returned from an alarm state
   = 0 : analog value in normal state

Description:

This word is used to determine whether an analog value has changed state ie. if the analog value has entered an alarm state or is returning to the normal state.
The analog will be in the alarm state if it has gone above the respective high high hysteresis limit or has gone below the respective low low hysteresis limit. The analog value will be considered to be returning to the normal state if it returns to below the high limit (if it was in a high high alarm state), or if it returns to above the low limit (if it was in a low low alarm state).

The word will retrieve the previous analog state in order to determine if a change in state has occurred. The previous states are stored in the PREV_ANALOG_STATE array. Before exiting this word, the current analog state is stored in the PREV_ANALOG_STATE array. The array is indexed using the analog input number.

2.7.3.12 **ANA_INPUT_INHIB?**

Interface:

ANA_INPUT_INHIB?

Stack: n1 >>> n2

where n1 = analog input number

n2 = 1 : analog input inhibited
   = 0 : analog input not inhibited

Description:

This word will read the Analog Inputs Inhibit DIT register in order to determine whether the analog input is inhibited or not.

2.7.3.13 SET_UPLOAD_PERIOD

Interface:

SET_UPLOAD_PERIOD

Stack: analog_input_number >>>

Description:

This word will set the analog poll period for an input using the analog upload period programmed in the DIT.

The word will use the analog input number passed in as an index into the DIT table for the analog upload period time for that input.

The analog upload period is entered into the ANA_UPLOAD_TIME array in multiples of 10 secs. using the analog input number as the index. The values in the ANA_UPLOAD_TIME array are decremented in the PERFORM_BACKGROUND_TASK word in the RTU Task Scheduler module.

2.7.3.14 UPLOAD_PER_0?

Interface:

UPLOAD_PER_0?

Stack: analog_input_number >>> n1

where n1 = 1 if the analog upload period for an analog input is 0
= 0 otherwise.

and analog_input_number = 0 - 3.

Description:

This word will check the Analog Upload Period in the DIT corresponding to the input number passed it. It will return a 1 if the value in this DIT register is 0 (indicating inhibition of analog input value uploading). Else it will return a 0.

2.7.3.15 TIME_TO_UPLOAD?

Interface:

TIME_TO_UPLOAD?

Stack: analog_input_number >>> n1

where n1 = 1 if analog time for the input has decremented to 0
= 0 if the analog upload period timeout for that input is
still
timing out.

Description:

This word will check the analog upload period timeout in the ANA_UPLOAD_TIME array using the analog input number passed in, as an index.

If the timeout has decremented to 0, then a 1 is returned on the stack, else a 0 is returned.

2.7.3.16 PROCESS_ANALOG_UPLOAD

Interface:

PROCESS_ANALOG_UPLOAD

Stack: >>>

Description:

This word is used to upload an analog value corresponding to one of the four analog inputs, to the central controller. The word will check if uploading for an input is required. This is achieved by reading the corresponding Analog Upload Period DIT register. If the value in the DIT is 0, then the analog input value is not uploaded. This also applies if the analog input is inhibited.

The upload timeout period for each analog input is stored in the array ANA_UPLOAD_TIME [3.3]. Each time an analog value is uploaded, the corresponding entry in this array is reset with the Analog Upload Period time value from the DIT.

The word will contantly monitor the value in the ANA_UPLOAD_TIME array. When it decrements to 0, the word will transmit the respective analog value.

Refer to the PROCESS_ANALOG_UPLOAD Flowchart.

2.8 Output Control Module

2.8.1 Overview

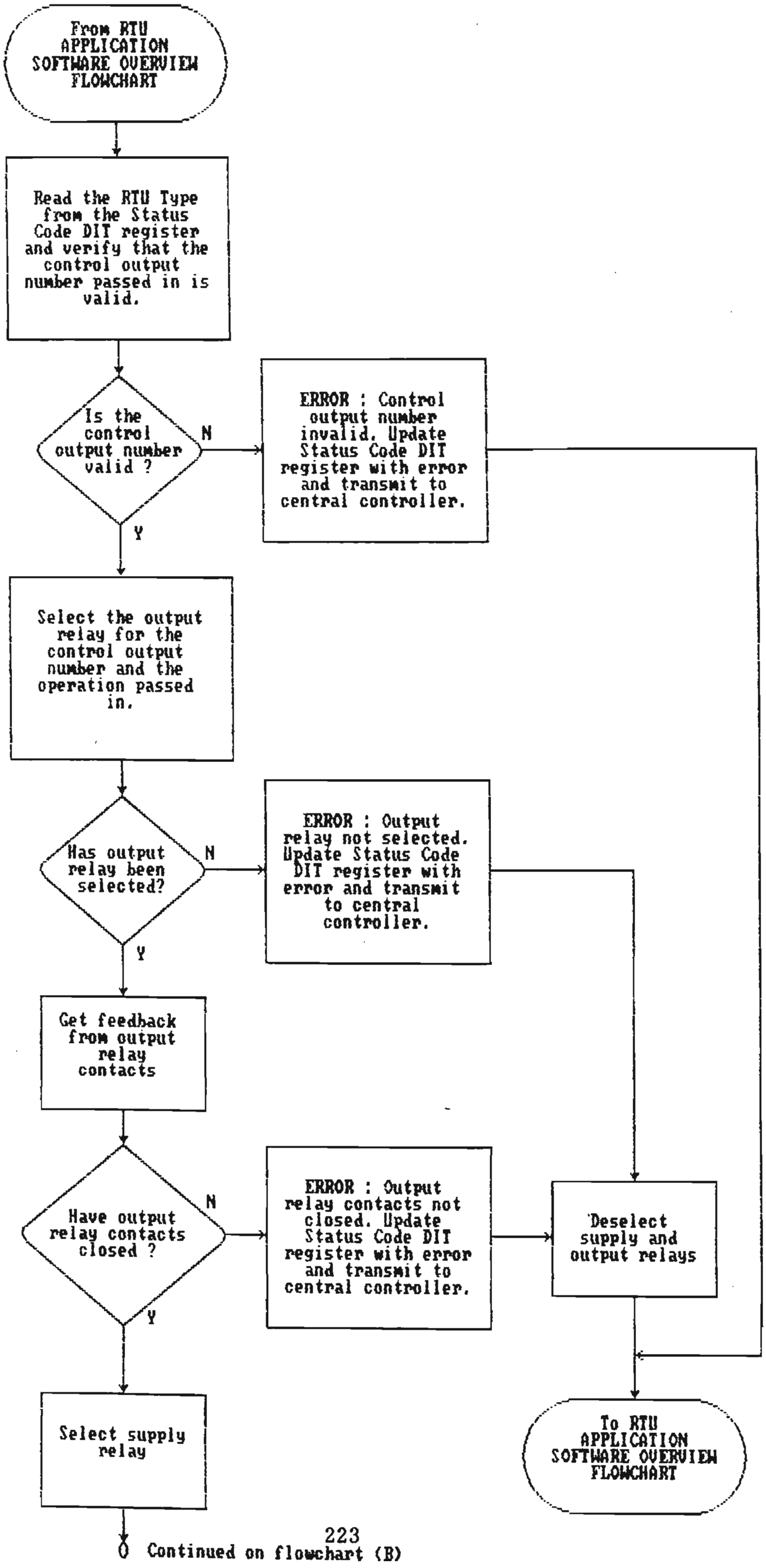
The output control module contains a suite of words to process a control output command message from the RTU Task scheduler.

The main task in this module is PROCESS_CONT_OUTPUTS that requires only the control output number and the operation to process an output.

The Task Scheduler will call PROCESS_CONT_OUTPUTS when a control message is received. The Task Scheduler will then check the variable CON_ACTIVE. If the variable is set to true, the Task Scheduler will continue calling PROCESS_CONT_OUTPUTS until the variable CON_ACTIVE is sensed false. Thereafter, the Task Scheduler will call PROCESS_CONT_OUTPUTS only when a control message is received.

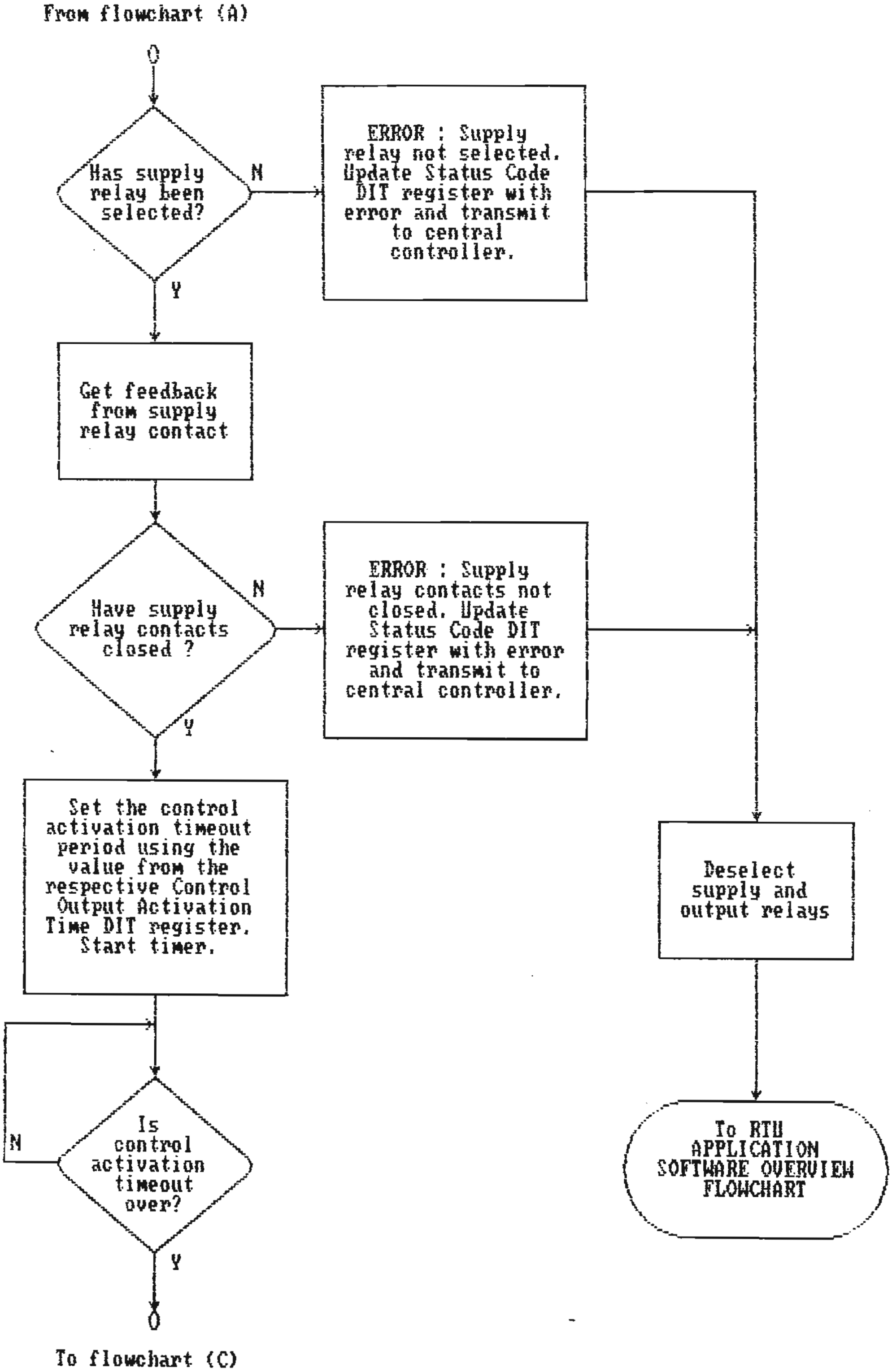
Refer to the CONTROL OUTPUT MODULE Flowchart (A), (B) and (C). All flowcharts relating to the output control module are grouped together starting from the next page. The flowcharts are grouped in hierachical order.

CONTROL OUTPUT MODULE OVERVIEW FLOWCHART (A) (Refer to PROCESS_CONT_OUTPUTS Flowchart)


From RTU APPLICATION SOFTWARE OVERVIEW FLOWCHART
Read the RTU Type from the Status Code DIT register and verify that the control output number passed in is valid.
Is the control output number valid ?
N
Y
ERROR : Control output number invalid. Update Status Code DIT register with error and transmit to central controller.
Select the output relay for the control output number and the operation passed in.
Has output relay been selected?
N
Y
ERROR : Output relay not selected. Update Status Code DIT register with error and transmit to central controller.
Get feedback from output relay contacts
Have output relay contacts closed ?
N
Y
ERROR : Output relay contacts not closed. Update Status Code DIT register with error and transmit to central controller.
Deselect supply and output relays
Select supply relay
To RTU APPLICATION SOFTWARE OVERVIEW FLOWCHART
O

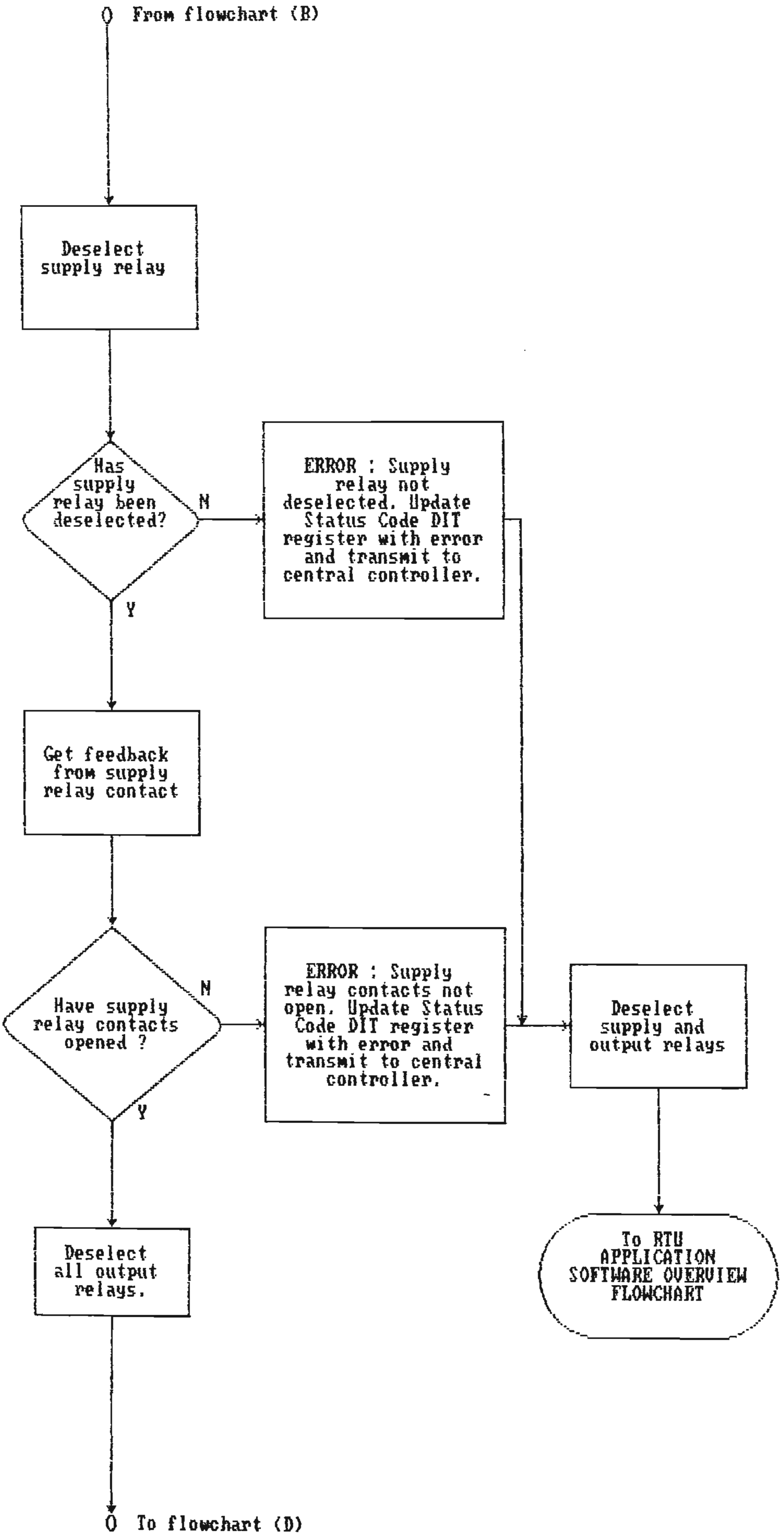
Continued on flowchart (B)

CONTROL OUTPUT MODULE OVERVIEW FLOWCHART (B)

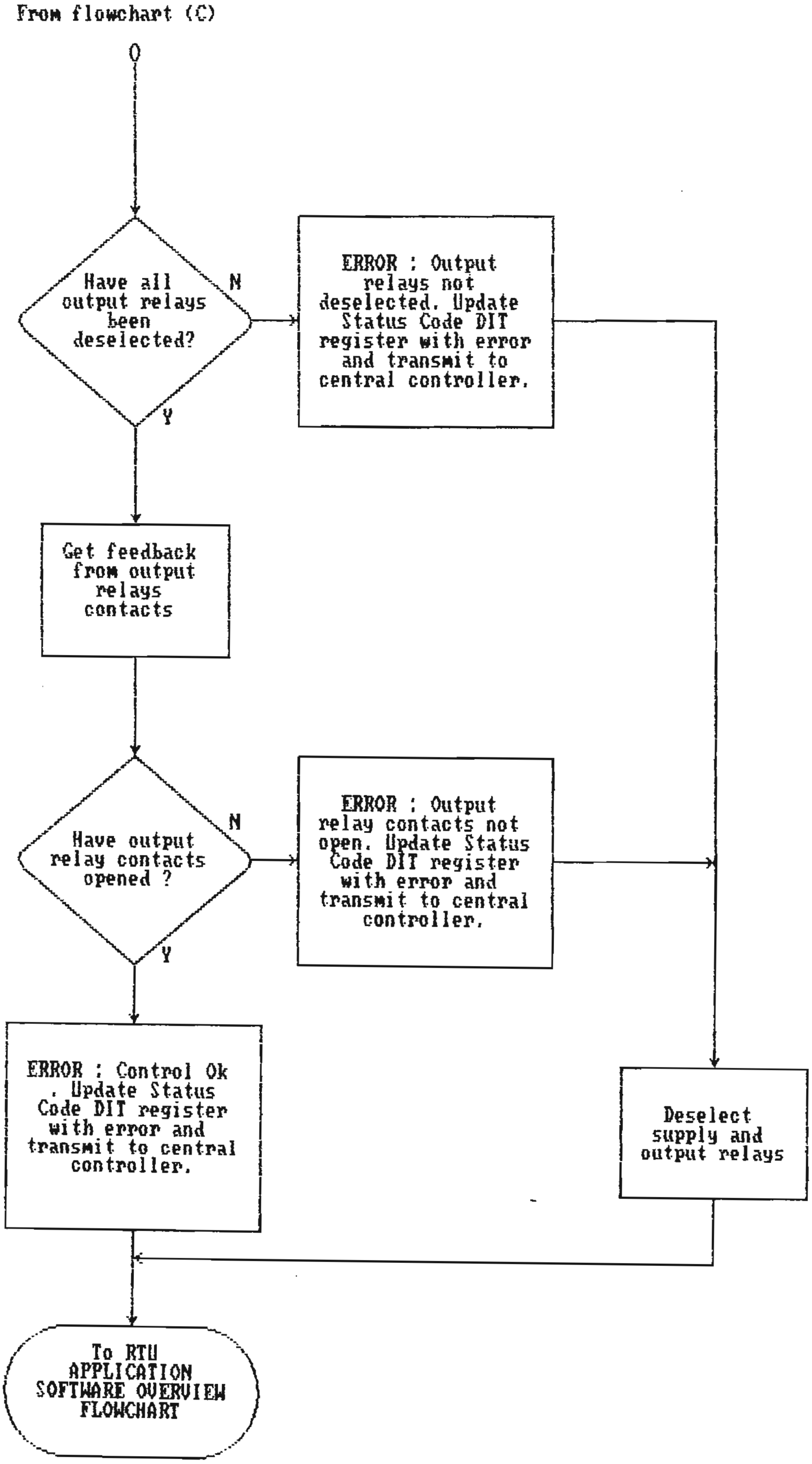


From flowchart (A)
Has supply relay been selected?
N
ERROR : Supply relay not selected. Update Status Code DIT register with error and transmit to central controller.
Y
Get feedback from supply relay contact
Have supply relay contacts closed ?
N
ERROR : Supply relay contacts not closed. Update Status Code DIT register with error and transmit to central controller.
Y
Set the control activation timeout period using the value from the respective Control Output Activation Time DIT register. Start timer.
Deselect supply and output relays
Is control activation timeout over?
N
Y
To RTU APPLICATION SOFTWARE OVERVIEW FLOWCHART
To flowchart (C)

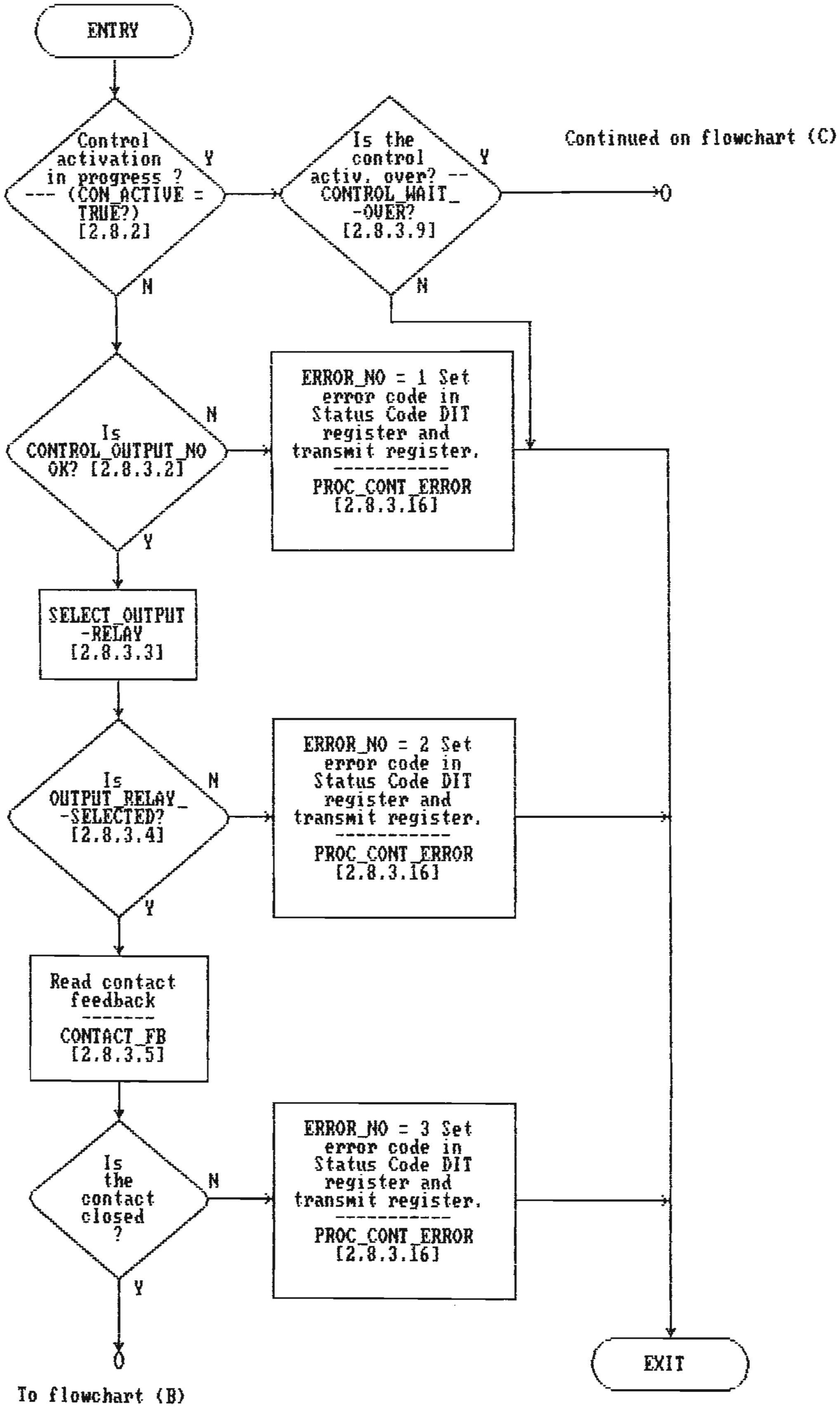
CONTROL OUTPUT MODULE OVERVIEW FLOWCHART (C)


From flowchart (B)
Deselect supply relay
Has supply relay been deselected?
N
Y
ERROR : Supply relay not deselected. Update Status Code DIT register with error and transmit to central controller.
Get feedback from supply relay contact
Have supply relay contacts opened ?
N
Y
ERROR : Supply relay contacts not open. Update Status Code DIT register with error and transmit to central controller.
Deselect supply and output relays
To RTU APPLICATION SOFTWARE OVERVIEW FLOWCHART
Deselect all output relays.
To flowchart (D)

CONTROL OUTPUT MODULE OVERVIEW FLOWCHART (D)


From flowchart (C)
0
Have all output relays been deselected?
N
Y
ERROR : Output relays not deselected. Update Status Code DIT register with error and transmit to central controller.
Get feedback from output relays contacts
Have output relay contacts opened ?
N
Y
ERROR : Output relay contacts not open. Update Status Code DIT register with error and transmit to central controller.
ERROR : Control Ok . Update Status Code DIT register with error and transmit to central controller.
Deselect supply and output relays
To RTU APPLICATION SOFTWARE OVERVIEW FLOWCHART

PROCESS_CONT_OUTPUTS FLOWCHART (A)


ENTRY
Control activation in progress ? --- (CON_ACTIVE = TRUE?) [2.8.2]
Y
N
Is the control activ. over? -- CONTROL_WAIT_ -OVER? [2.8.3.9]
Y
N
Continued on flowchart (C)
Is CONTROL_OUTPUT_NO OK? [2.8.3.2]
N
Y
ERROR_NO = 1 Set error code in Status Code DIT register and transmit register.

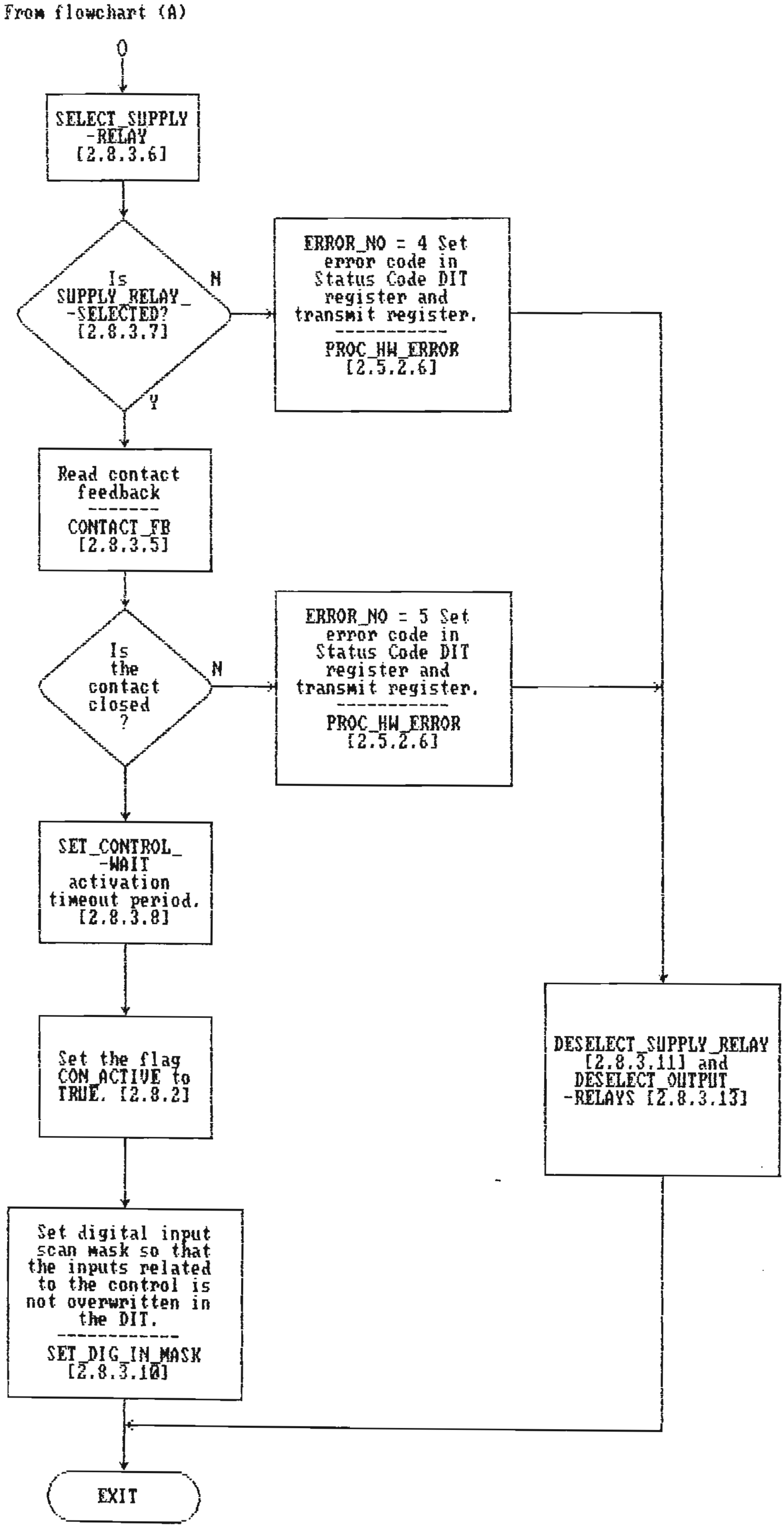
PROC_CONT_ERROR [2.8.3.16]
SELECT_OUTPUT -RELAY [2.8.3.3]
Is OUTPUT_RELAY_ -SELECTED? [2.8.3.4]
N
Y
ERROR_NO = 2 Set error code in Status Code DIT register and transmit register.

PROC_CONT_ERROR [2.8.3.16]
Read contact feedback

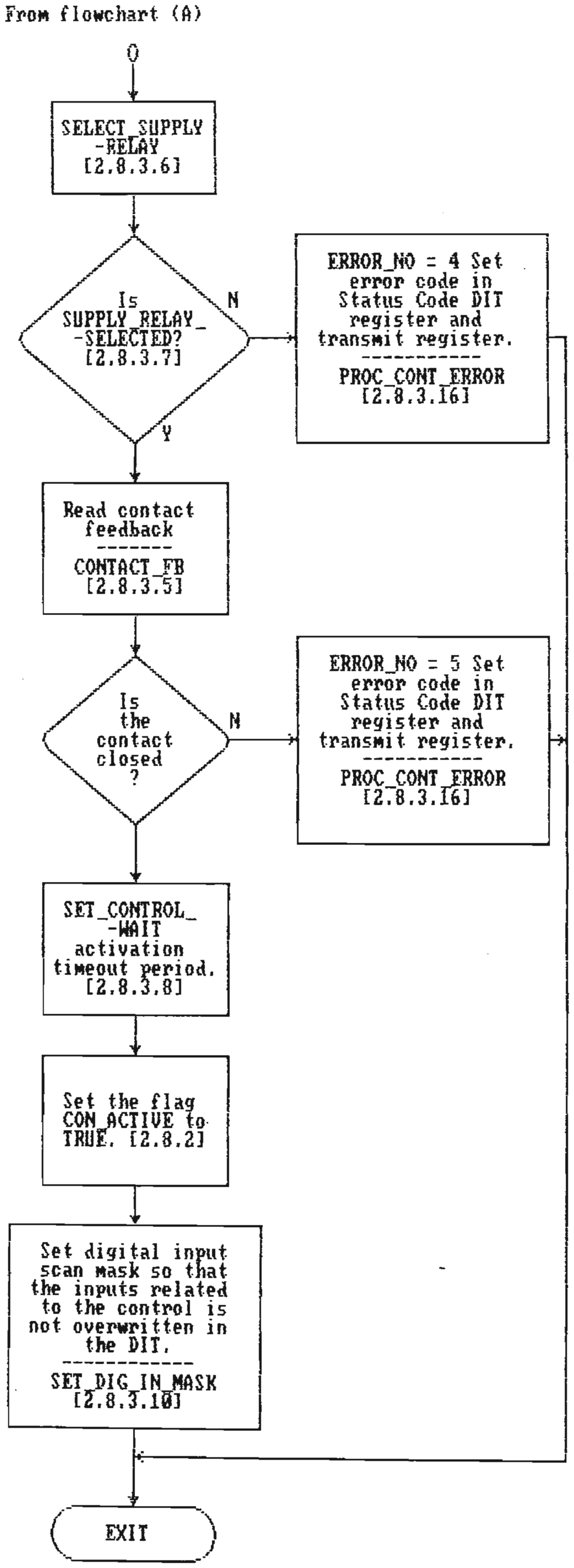
CONTACT_FB [2.8.3.5]
Is the contact closed ?
N
Y
ERROR_NO = 3 Set error code in Status Code DIT register and transmit register.

PROC_CONT_ERROR [2.8.3.16]
EXIT
To flowchart (B)

PROCESS_CONT_OUTPUTS FLOWCHART (B)

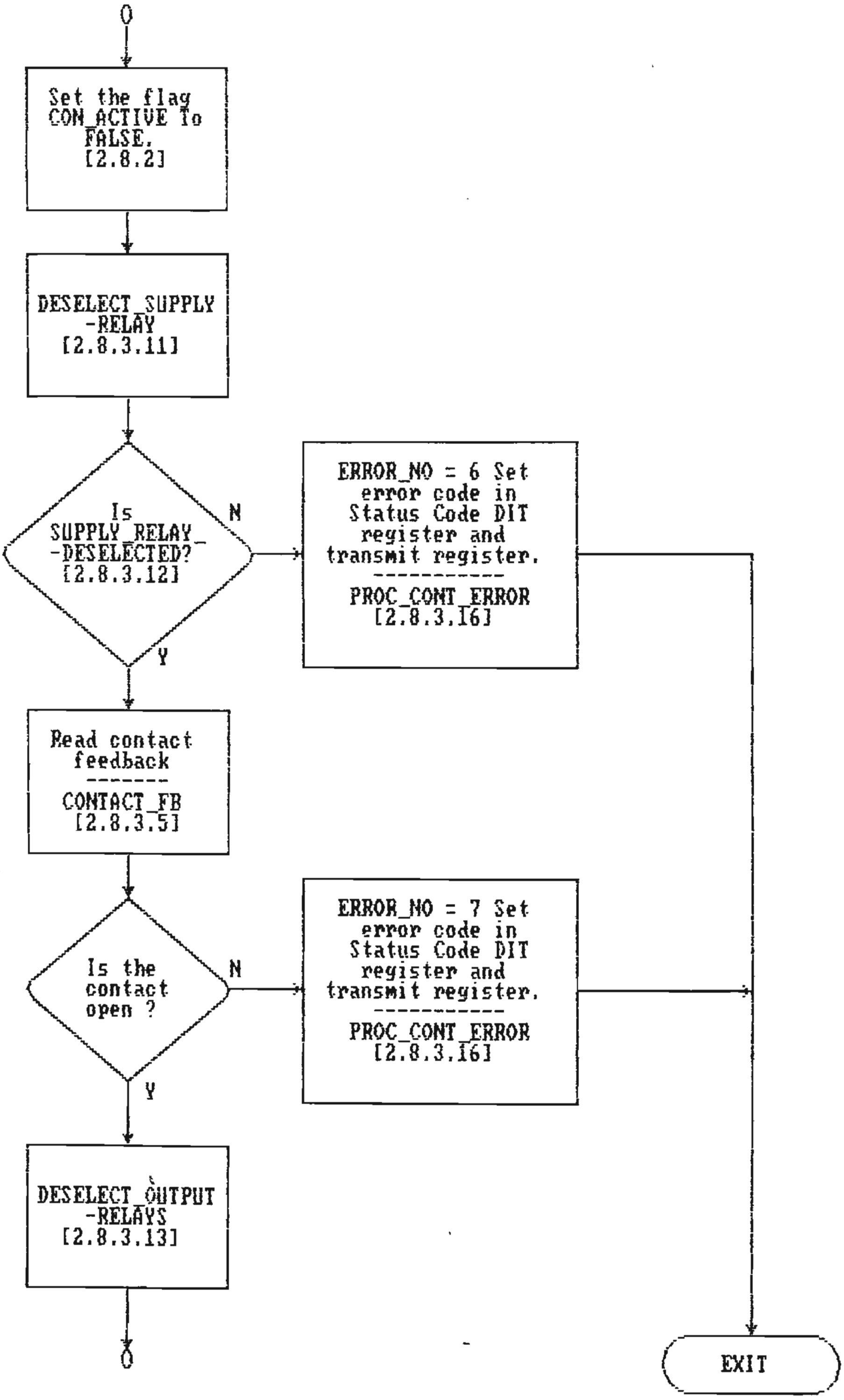

From flowchart (A)
O
SELECT_SUPPLY -RELAY [2.8.3.6]
Is SUPPLY_RELAY_ -SELECTED? [2.8.3.7]
N
Y
ERROR_NO = 4 Set error code in Status Code DIT register and transmit register.
PROC_HW_ERROR [2.5.2.6]
Read contact feedback
CONTACT_FB [2.8.3.5]
Is the contact closed ?
N
ERROR_NO = 5 Set error code in Status Code DIT register and transmit register.
PROC_HW_ERROR [2.5.2.6]
SET_CONTROL_ -WAIT activation timeout period. [2.8.3.8]
Set the flag CON_ACTIVE to TRUE. [2.8.2]
DESELECT_SUPPLY_RELAY [2.8.3.11] and DESELECT_OUTPUT -RELAYS [2.8.3.13]
Set digital input scan mask so that the inputs related to the control is not overwritten in the DIT.
SET_DIG_IN_MASK [2.8.3.10]
EXIT

PROCESS_CONT_OUTPUTS FLOWCHART (B)


From flowchart (A)
0
SELECT_SUPPLY -RELAY [2.8.3.6]
Is SUPPLY_RELAY_ -SELECTED? [2.8.3.7]
N
Y
ERROR_NO = 4 Set error code in Status Code DIT register and transmit register.
PROC_CONT_ERROR [2.8.3.16]
Read contact feedback
CONTACT_FB [2.8.3.5]
Is the contact closed ?
N
ERROR_NO = 5 Set error code in Status Code DIT register and transmit register.
PROC_CONT_ERROR [2.8.3.16]
SET_CONTROL_ -WAIT activation timeout period. [2.8.3.8]
Set the flag CON_ACTIVE to TRUE. [2.8.2]
Set digital input scan mask so that the inputs related to the control is not overwritten in the DIT.
SET_DIG_IN_MASK [2.8.3.10]
EXIT

PROCESS_CONT_OUTPUTS FLOWCHART (C)

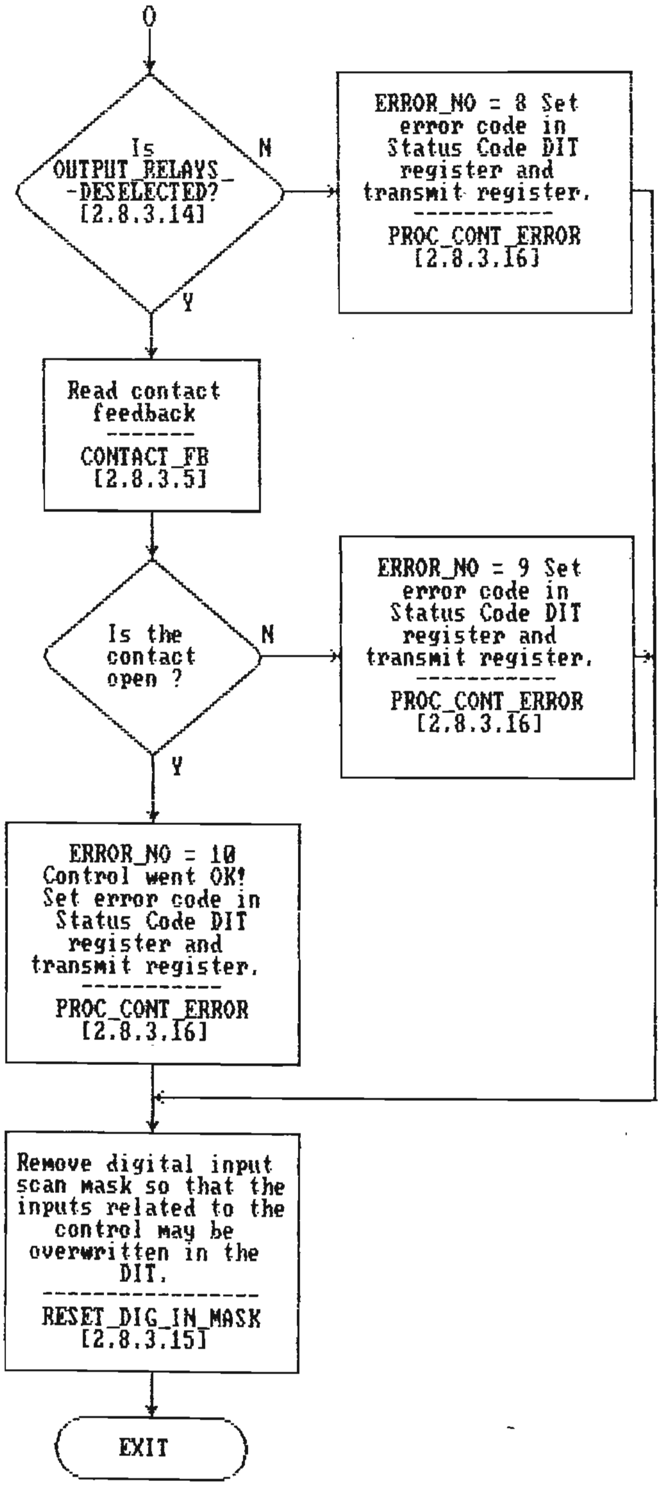
Continued from flowchart (A)


0
Set the flag CON_ACTIVE To FALSE. [2.8.2]
DESELECT_SUPPLY -RELAY [2.8.3.11]
Is SUPPLY_RELAY_ -DESELECTED? [2.8.3.12]
N
Y
ERROR_NO = 6 Set error code in Status Code DIT register and transmit register.
PROC_CONT_ERROR [2.8.3.16]
Read contact feedback
CONTACT_FB [2.8.3.5]
Is the contact open ?
N
Y
ERROR_NO = 7 Set error code in Status Code DIT register and transmit register.
PROC_CONT_ERROR [2.8.3.16]
DESELECT_OUTPUT -RELAYS [2.8.3.13]
0
EXIT

To flowchart (D)

PROCESS_CONT_OUTPUTS FLOWCHART (D)

Continued from flowchart (C)


0
Is OUTPUT_RELAYS_-DESELECTED? [2.8.3.14]
N
Y
ERROR_NO = 8 Set error code in Status Code DIT register and transmit register.
PROC_CONT_ERROR [2.8.3.16]
Read contact feedback
CONTACT_FB [2.8.3.5]
Is the contact open ?
N
Y
ERROR_NO = 9 Set error code in Status Code DIT register and transmit register.
PROC_CONT_ERROR [2.8.3.16]
ERROR_NO = 10 Control went OK! Set error code in Status Code DIT register and transmit register.
PROC_CONT_ERROR [2.8.3.16]
Remove digital input scan mask so that the inputs related to the control may be overwritten in the DIT.
RESET_DIG_IN_MASK [2.8.3.15]
EXIT

2.8.2 Variables

Type	Variable	Description
integer	CON_ACTIVE	flag to indicate that the control output activation period for a control is in progress. A furthur description is given in the section "Global variables".

2.8.3 FORTH Words

2.8.3.1 PROCESS_CONT_OUTPUTS

Interface:

PROCESS_CONT_OUTPUTS

Stack: operation output_number >>>

Description:

This word is used to perform a control on an output device with the parameters passed in.

The parameter "operation" will be set to 255 if a "CLOSE" operation is requested or to a 0 if an "OPEN" operation is requested. Refer to the "Control Output Module" flowchart. Output numbers range from 0 - 7.

2.8.3.2 CONTROL_OUTPUT_NO_OK?

Interface:

CONTROL_OUTPUT_NO_OK?

Stack: operation control_output_number >>> 1 = ok, 0 = not ok

Description:

This word is used to check if the cocontrol output number passed in is valid for the output card in the respective slot in the RTU electronic module.

The word will read the card type of the card in the respective slot to determine if the output number is valid or not.

2.8.3.3 **SELECT_OUTPUT_RELAY**

Interface:

SELECT_OUTPUT_RELAY

Stack: output_relay_number card_slot_number >>>

Description:

Used to energise an output relay on the control output card in the respective slot.

2.8.3.4 **OUTPUT_RELAY_SELECTED?**

Interface:

OUTPUT_RELAY_SELECTED?

Stack: output_relay_number card_slot_number >>> n1

```
where  n1 =   1 : relay selected
          =   0 : relay not selected
```

Description:

Used to confirm selection of an output relay.

2.8.3.5 **CONTACT_FB**

Interface:

CONTACT_FB

Stack: >>> n1

```
where n1 =   1 : contact feedback bit high
         =   0 : contact feedback bit low
```

Description:

This word is used to retrieve the status of the contact feedback bit from the the general status register in the control output card. Depending on the previous sequence of events the feedback bit will indicate closure of supply and/ closure of output relay contacts.

2.8.3.6 SELECT_SUPPLY_RELAY

Interface:

SELECT_SUPPLY_RELAY

Stack: card_slot_number >>>

Description:

Used to energise the supply relay on the control output card in the respective slot.

2.8.3.7 SUPPLY_RELAY_SELECTED?

Interface:

SUPPLY_RELAY_SELECTED?

Stack: card_slot_number >>> n1

```
where n1 =   1 : relay selected
         =   0 : relay not selected
```

Description:

Used to confirm selection of the supply relay on the output card.

2.8.3.8 SET_CONTROL_WAIT

Interface:

SET_CONTROL_WAIT

Stack: relay_output_number >>>

where relay_output_number = 0 - 7

Description:

This word is used to set the timeout period for the control output activation. It will use the parameters passed in to index into the DIT for the control activation period. The word CONTROL_WAIT_OVER? must be used in conjunction with this word to determine the end of the timeout period.

Timer 1 in the Eziforth kernel is used for this timer.

2.8.3.9 CONTROL_WAIT_OVER?

Interface:

CONTROL_WAIT_OVER?

Stack: >>> n1

where n1 = 1 : control wait period over
= 0 : still timing

Description:

This word is used to determine the end of the timeout period set by the word SET_CONTROL_WAIT.

2.8.3.10 SET_DIG_IN_MASK

Interface:

SET_DIG_IN_MASK

Stack: control_output_number >>>

Description:

This word is used to set the digital input scan mask so that change of state messages on digital inputs will not be generated while the control output activation is in progress. The variable used to set the mask is DIG_IN_MASK.
The word will set the mask as follows:

control output number	:	DIG_IN_MASK in binary
		MSB LSB
1	:	1111 1111 1111 1100
2	:	1111 1100 1111 1111
3	:	1111 0011 1111 1111
4	:	1100 1111 1111 1111

This mask corresponds to the control input bits in Status Inputs DIT register.

2.8.3.11 DESELECT_SUPPLY_RELAY

Interface:

DESELECT_SUPPLY_RELAY

Stack: card_slot_number >>>

Description:

Used to de-energise the supply relay on the control output card in the respective slot.

2.8.3.12 SUPPLY_RELAY_DESELECTED?

Interface:

SUPPLY_RELAY_DESELECTED?

```
  Stack:   card_slot_number >>> n1

  where n1 =   1 : relay deselected
           =   0 : relay not deselected
```

Description:

Used to confirm deselection of the supply relay on the output card.

2.8.3.13 DESELECT_OUTPUT_RELAYS

Interface:

DESELECT_OUTPUT_RELAYS

```
  Stack:   card_slot_number >>>

  where card_slot_number = 0 - 2
```

Description:

Used to de-energise an output relay on the control output card in the respective slot.

2.8.3.14 OUTPUT_RELAYS_DESELECTED?

Interface:

OUTPUT_RELAYS_DESELECTED?

```
  Stack:   card_slot_number >>>  n1

  where n1 =   1 : all output relays deselected
           =   0 : relays not deselected
```

Description:

Used to confirm deselection of the output relays.

2.8.3.15 **RESET_DIG_IN_MASK**

Interface:

RESET_DIG_IN_MASK

Stack: control_output_number >>>

Description:

This word is used to reset bits in the digital input scan mask that were set by the word SET_DIG_IN_MASK. The variable used to reset the mask is DIG_IN_MASK. The word will OR the existing mask with the following value depending on the control output number passed in.

The word will set the mask as follows:

control output number	:	ORing value
		MSB LSB
1	:	0000 0000 0000 0011
2	:	0000 0011 0000 0000
3	:	0000 1100 0000 0000
4	:	0011 0000 0000 0000

This mask corresponds to the control input bits in Status Inputs DIT register.

2.8.3.16 **PROC_CONT_ERROR**

Interface:

PROC_CONT_ERROR

Stack: control_output_number >>>

Description:

This word is used to transmit the RTU Status Code register to the central controller after being updated with the respective error code. Additionally, the word will de-activate all relays on the digital output card.