NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

A DSP-56001 SOFTWARE ARCHITECTURE FOR A RADIO AND DATA PACKET CONTROLLER

by

William Chee-Foon Chong

B.A.Sc., University of British Columbia, 1984

PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ENGINEERING
IN THE SCHOOL OF ENGINEERING SCIENCE

SIMON FRASER UNIVERSITY

ISBN 0-315-83656-3

APPROVAL

NAME: William Chee-Foon Chong

DEGREE: Master of Engineering

TITLE OF PROJECT: A DSP-56001 Software Architecture for a Radio and Data Packet Controller

EXAMINING COMMITTEE:

Chairman: Dr. John D. Jones

Dr. Shawn Stapleton
Assistant Professor

Dr. Jacques Vaisey
Assistant Professor

Mr. Peter McConnell
External Examiner

DATE APPROVED: December 11, 1991

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

A DSP-56001 Software Architecture for a Radio and Data Packet Controller

Author:

(signature)

William Chong

(name)

12 December 1991

(date)

ABSTRACT

The Advanced Radio and Data Packet Controller (ARDPC) is a prototype second generation radio and data packet controller for mobile data communications which is capable of full duplex 4800 bps operation using FM modulated bipolar waveforms and the Radio Network Communications Protocol (RNCP) signalling scheme. The ARDPC is implemented using a duo processor hardware/software architecture featuring a Motorola DSP-56001 digital signal processing (DSP) chip teamed with a Motorola 68000 microprocessor. The ARDPC represents an advancement in radio and data packet controller design through the use of DSP in its modem functions. This design feature allows the ARDPC to offer much improved intelligence, flexibility and functionality over its non-DSP predecessors. One notable functional improvement is the ability to perform message processing at the controller level to support radio channel monitoring operations.

This report focuses on the DSP-56001 software requirements, architecture, and DSP algorithms implemented for the ARDPC. The description of the software architecture is presented using a top-down module decomposition employing data and control flow (dcf) diagrams as tools. The DSP algorithms described include the modulation, demodulation and data detection algorithms for the ARDPC modem. The performance of these algorithms is measured using static and fading bit-error rate (BER) performance, modulation spectral occupancy and processor bandwidth utilization. The demodulator BER performance is measured against white Gaussian noise and is shown to be 2-3 dB better than a first generation controller. Spectral occupancy measurements of the ARDPC modulation scheme is shown to easily meet Federal Communications Commission (FCC) requirements for 25 kHz UHF/VHF radio channels while processor bandwidth measurements confirm that the software architecture is fully capable of supporting 4800 bps data communication.

TABLE of CONTENTS

LIST of TABLES

LIST of FIGURES

LIST of ACRONYMS and ABBREVIATIONS

AC	Alternating Current
ADC	Analog to Digital Convertor
ARDPC	Advanced Radio and Data Packet Controller
BCH	Bose-Chaudhuri-Hocquenghen
BER	bit-error rate
bps	bits per second
CC	Communications Controller
CCITT	Consultive Committee International Telegraph and Telephone
CM	Channel Multiplexer
CSMA	Carrier Sense Multiple Access
CWID	Continuous Wave IDentification
DAC	Digital to Analog Convertor
dB	decibels
DC	Direct Current
dcf	data and control flow
DISC	Discriminator
DOC	Department of Communications (Canada)
DSMA	Digital Sense Multiple Access
DSP	digital signal processor/processing
FCC	Federal Communications Commission (U.S.A.)
FEC	Forward Error Correction
FIR	Finite Impulse Response
FM	Frequency Modulation
HI	Host Interface port (DSP-56001)
HRDF	Host Receive Data Full (HI Port Interrupt)
Hz	Hertz
isr	Interrupt Service Routine
IIR	Infinite Impulse Response
kHz	kilohertz
LEDs	Light Emitting Diodes
MDTs	Mobile Data Terminals
MID	Mobile IDentifier
MOD	Modulator

modem	*modulator-demodulator*
msecs	milliseconds
NOVRAM	NOn-Volatile Random Access Memory
ns	nanosecond
POST	Power On Self Tests
RAM	Random Access Memory
RNCP	Radio Network Communications Protocol
ROM	Read Only Memory
RF	Radio Frequency
RS	Reed-Solomon
RSSI	Radio Signal Strength Indicator
SCC	Serial Communications Chip
SCI	Serial Communication Interface
SSI	Synchronous Serial Interface
SYNC	Frame SYNChronization codewords
UHF	Ultra High Frequency
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
VSWR	Voltage Standing Wave Ratio
UVEPROM	UltraViolet Erasable PROgrammable Memory
VHF	Very High Frequency
VLSI	Very Large Scale Integration

1. INTRODUCTION

1.1 PROJECT BACKGROUND

Radio and Data Packet Controllers (RDPCs) are a key component of an RNCP mobile data network. The main function of the RDPC is to perform modem functions at a transceiver site and to control radio channel access by mobile units. The first generation RDPCs produced by a local communications company were implemented using analog circuitry and custom Very Large Scale Integrated Circuits (VLSICs). These first generation controllers were essentially 'dumb' modems with specialized hardware functions. The principal disadvantage of the first generation design was that it was not protocol independent, meaning there was no means to upgrade or adapt the device to handle faster or more reliable radio channel protocols as they became available. This posed a serious problem for clients who became locked into a specific protocol upon purchasing the RDPCs and for the company, which potentially had to engineer a new RDPC each time a different protocol was introduced.

The ARDPC project was undertaken to prototype an advanced state-of-the-art controller which would be a candidate to replace the first generation design. The primary objective for the development of the ARDPC was to overcome the shortcomings of the first generation design by providing a universal hardware platform on which different radio channel protocols could be based. A critical requirement for the ARDPC was that its modem functions be protocol independent. This requirement led company designers to consider the use of digital signal processing architectures and techniques to satisfy the requirement. At the time of development, the latest trend in modem design involved the use of single chip DSP processors which could implement DSP algorithms in software. This technique seemed to be ideal for the ARDPC because of the advantages it offered in terms of system flexibility and the inherent ease and precision associated with processing digital rather than analog signals. The most attractive aspect of the technique was that a modem function implemented in a DSP processor could be updated without hardware changes, thus prolonging the useful life of a product. The eventual decision to adopt this technique was also facilitated by the remarkable increase in horsepower and decrease in unit costs of DSP semiconductors over the preceding years.

After careful consideration of available single-chip DSP processors it was decided that Motorola's DSP-56001 offered the best combination of cost, performance and capabilities for the ARDPC. The ARDPC was eventually implemented using a DSP based hardware/software architecture employing both a Motorola 68000 microprocessor and a DSP-56001 digital signal processing chip. This duo processor design is in sharp contrast to conventional modem designs which are usually implemented using discrete analog and digital components.

1.2 DOCUMENT OVERVIEW

The purpose of this report is to focus on the DSP-56001 software requirements, architecture and key processing algorithms implemented for the ARDPC. Details of the hardware and the 68000 microprocessor are presented in summary fashion. The initial sections of the report provides background information on the RNCP mobile data network. This information is necessary in order to understand the basis for the software architecture and some of the DSP algorithms implemented. After this background information has been presented, the functional requirements for the ARDPC are examined and the software architecture and implementation details used to satisfy the requirements are discussed. Finally, performance test results are presented to verify the operational capabilities of the ARDPC.

This report consists of seven sections:

Section 1 provides the introduction and document overview.

Section 2 presents an overview of the RNCP network physical and data link layers.

Section 3 provides an overview of the ARDPC functional requirements, hardware architecture and DSP-56001 software requirements.

Section 4 discusses the DSP-56001 software architecture.

Section 5 describes the key DSP algorithms implemented. The discussion includes details of the ARDPC demodulator and modulator algorithms. A data detection algorithm which is used by the ARDPC to detect the presence of a valid RNCP messages over the radio channel is also described.

Section 6 presents the performance test results. Included in the discussion are graphs showing the static and fading bit error rates (BERs), the spectral occupancy of the RNCP baseband waveforms and a measurement of the DSP-56001 processor bandwidth utilization.

Section 7 presents the conclusions and summary remarks.

2.0 RNCP NETWORK OVERVIEW

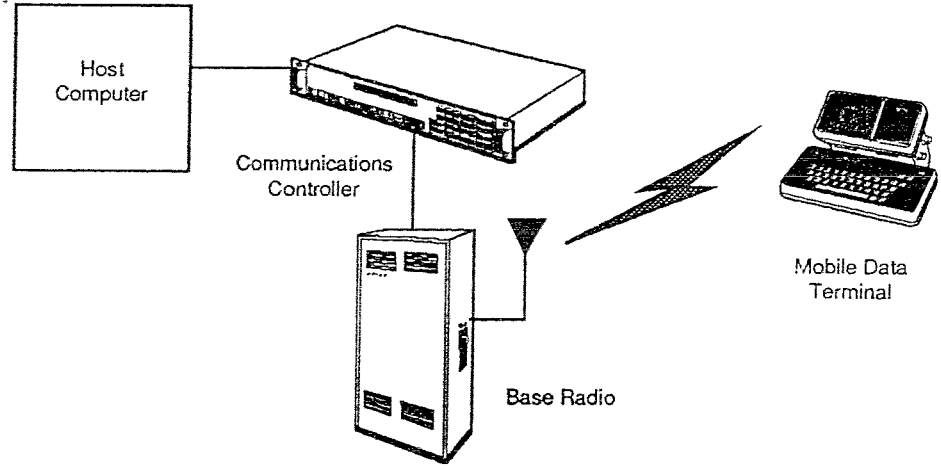
2.1 PHYSICAL LINK DESCRIPTION

An RNCP mobile data network is an example of a standalone UHF/VHF packet radio network. The network may be configured to operate with data only, or with shared data and voice. The network architecture may be organized in a simple manner to support a few mobile units or be complex enough to support several hundred mobile units using multiple transceiver sites. The network may also be configured to operate using ALOHA, CSMA or DSMA radio channel access. Only the latter two schemes provide a mechanism for inhibiting a mobile unit from transmitting until it senses that the radio channel is idle. In the CSMA configuration, a mobile unit uses the radio carrier frequency to sense channel activity while in the DSMA configuration a special channel activity bit is set and cleared in outbound messages to indicate the status of the inbound radio channel of the base site. (A message sent from the base site to a mobile unit is referred to as an "outbound" message while a message sent from a mobile unit to the base site is known as an "inbound" message). Further details of DSMA channel access control are described in section 2.2.2.5.

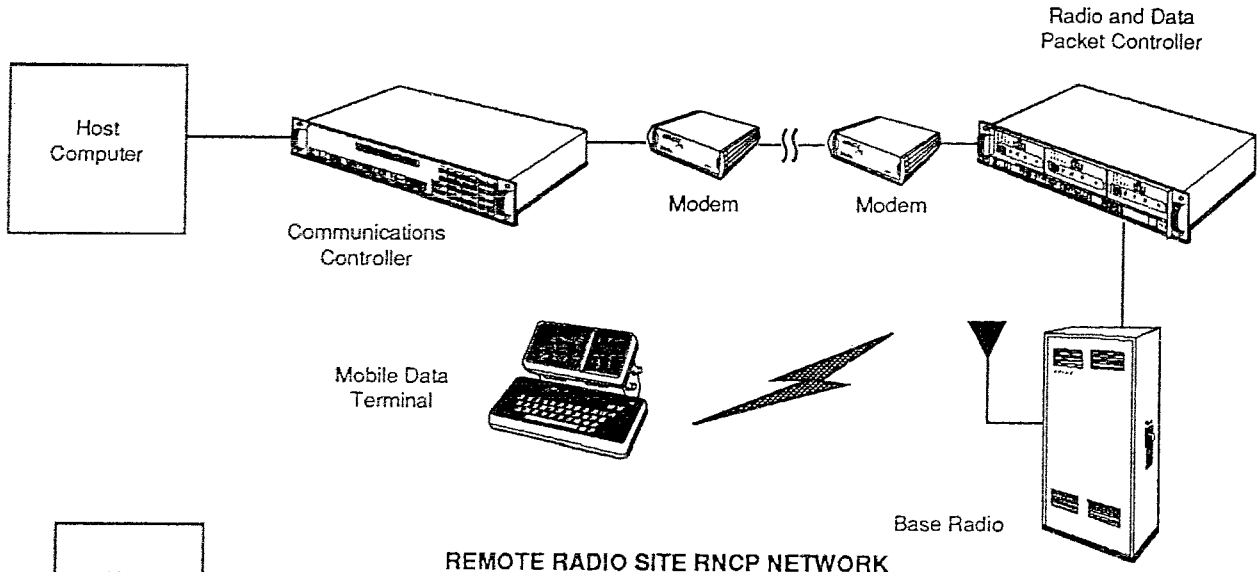
Base radio operation within the network can be configured for full duplex or half duplex operation with continuous, remote or local keying. The continuous keyed mode is most suited for full duplex operation. In this mode, the radio carrier signal is always present so that the radio never switches to an off state. In remote keying, the base radio is keyed on under control of the communications controller which is usually detached from the base site. Local keying retains radio control at the base site under the control of the RDPC.

A diagram of some typical RNCP network topologies is shown in Figure 2-1. Figure 2-1a shows the simplest configuration which involves the host computer and base radio being co-located at the same site. More complex configurations involving single or multiple remote transceiver sites are shown in Figures 2-1b and 2-1c. From the diagrams the RNCP network is shown to consist of the following physical components:

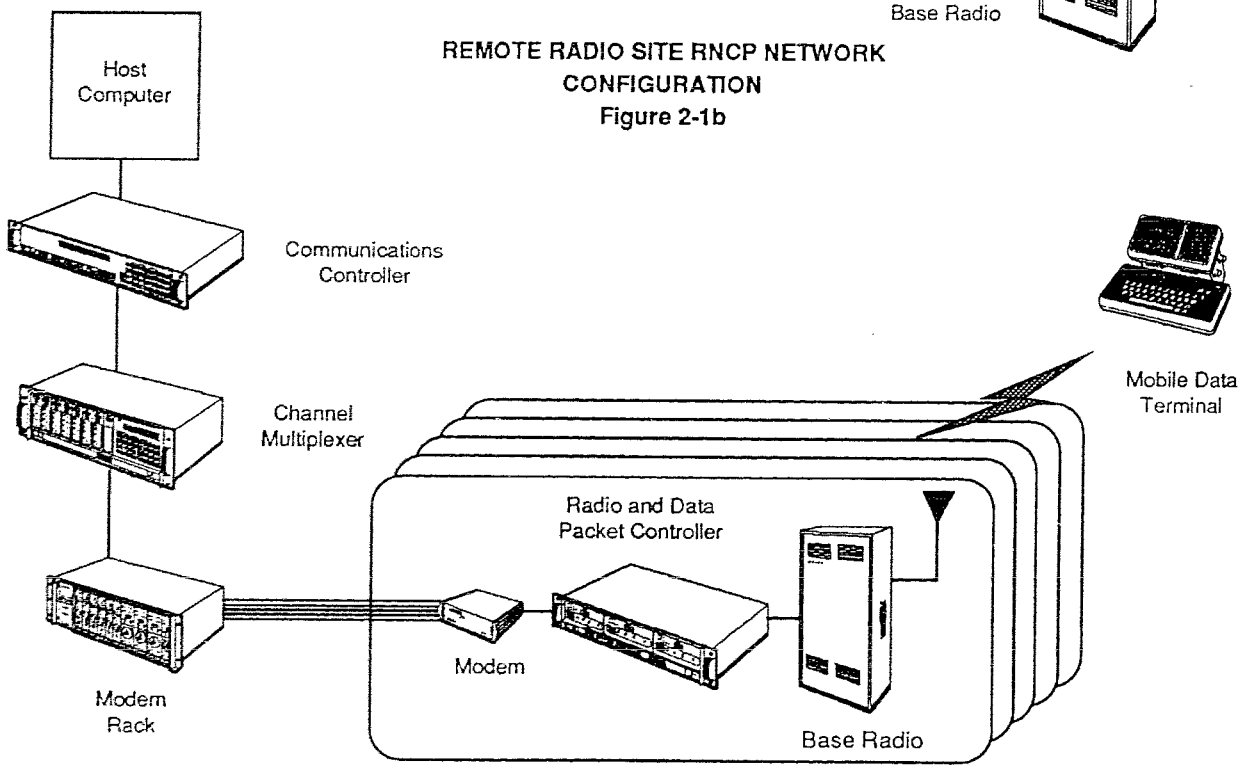
- Mobile Data Terminals (MDTs) (sec 2.1.1)
- host computer (sec 2.1.2)
- Communications Controller (CC) (sec 2.1.3)
- Radio and Data Packet Controller (RDPC) (sec 2.1.4)



CO-LOCATED COMPUTER AND RADIO SITE RNCP NETWORK CONFIGURATION
Figure 2-1a



REMOTE RADIO SITE RNCP NETWORK CONFIGURATION
Figure 2-1b



MULTIPLE TRANSCEIVER SITE RNCP NETWORK CONFIGURATION
Figure 2-1c

• Channel Multiplexer (CM) (sec 2.1.5)

The functions of each of the above units and their interactions with each other is discussed in the reference subsections listed.

2.1.1 MOBILE DATA TERMINALS (MDTs)

The MDTs are both the source of inbound messages to the host computer and the sink of outbound messages originating from the host computer. The MDTs may be configured to be mobile or portable. Each MDT is attached to a radio control unit capable of transmission and reception of data. The data is transmitted in the form of formatted data packets, which contain both text and control information. Included in the control information is an identification field which allows an MDT to react only to messages addressed to it. An acknowledgement is generated by an MDT to a base radio for each error free message received. All messages which are not acknowledged are re-transmitted by the communications controller at the host site.

2.1.2 HOST COMPUTER

The host computer provides the interface to the user at the host site and is the dedicated centralized system controller for the mobile data communications network. From the host computer, the user can format different classes of messages for specific mobile units. The host computer is directly connected to the communications controller in the mobile data network.

2.1.3 COMMUNICATIONS CONTROLLER (CC)

The communications controller serves as the interface for the MDTs to the host computer. Attached to one end of the communications controller is the host computer and at the other end the RDPC. The communications controller is responsible for controlling message re-transmission and for generating acknowledgements to inbound messages. The communications controller performs radio control by instructing the RDPC to initiate keying of radio transmitter sites closest to the mobile unit in a multi-transceiver installation. The communications controller is also responsible for message formatting by inserting forward error-correction (FEC) to messages destined for MDTs.

2.1.4 RADIO AND DATA PACKET CONTROLLER (RDPC)

The RDPC communicates via modem (leased line or dial-up) or microwave link to the CC. Its main function is to serve as the interface to the CC for the MDTs. The RDPC performs modulation and demodulation of inbound/outbound messages from host computer to MDT and vice-versa. The RDPC keys the radio transmitter at the base site under direct control from the CC. The RDPC is also responsible for multiple access control of the radio channel by asserting the channel busy condition in the RNCP message protocol when configured to operate using DSMA channel access.

2.1.5 CHANNEL MULTIPLEXER (CM)

In a complex network requiring extended coverage, there may be multiple remote transceiver sites which are linked together in a star configuration by a channel multiplexer to the host computer. The channel multiplexer, in addition to providing a multiplexing function also manages multiple copies of the same message received simultaneously from the multiple transceiver sites by forwarding only the first error-free copy to the CC.

2.2 DATA LINK DESCRIPTION

The mobile data signal environment is characterized not only by free space losses attributed to atmospheric propagation, but also by scattering and multi-path fading resulting from terrestrial interaction. The phenomena of multi-path fading is the cumulative result of reflected and scattered components of signals arriving at a receiver with varying angles of incidence, time delay and amplitude. These parameters vary with both the speed and location of a mobile unit and result in random phase and amplitude variations in the received signal. Multi-path fading can result in the amplitude of a received signal varying by up to 40 dB at a rate of 100 times per second for a vehicle travelling at 70 mph and at radio frequency of 900 MHz. The implications of multi-path fading are that a mobile voice or data signal suffers from high irreducible bit error rates and severe distortion in the mobile environment.

Nowhere are the effects of fading more evident than in the communication between a mobile unit and an RDPC, since signalling must be performed essentially at ground level where a multitude of local scatterers can exist. The design of a successful mobile data message protocol must address the effects of fading and signal distortion by employing techniques such as data interleaving, data redundancy and forward error-correcting codes to combat fading in the mobile environment.

The Radio Network Communications Protocol (RNCP) is an example of a successful 4800 bps mobile message protocol which has been in commercial use for several years. The RNCP protocol is used both for formatting and flow control of messages from a host computer and mobile units.

2.2.1 BIT LEVEL DESCRIPTION

The RNCP protocol transmits data at 4800 bps using differential-encoded bipolar-signalling of low pass filtered rectangular pulses. These pulses are frequency modulated (FM) by the base radio for transmission to mobile units. Differential encoded bipolar signalling is pseudo-ternary taking on the values -1, 0 and +1 at the symbol centre. A binary value of 0 is represented by a zero DC level while a binary value of 1 is represented by alternating positive and negative pulses. The use of alternating positive and negative pulses to represent a binary 1 provides a built in single bit-error detection capability since a receiver detection error which violates the alternating pulse rule can readily be detected.

The basic RNCP waveform is a low-pass rectangular pulse which satisfies the first Nyquist criterion for inter-symbol interference. A typical sequence of RNCP pulses is shown in Figure 2-2, while the eye diagram for pseudorandom RNCP data is shown in Figure 2-3. It can be seen that the noiseless RNCP pulse possesses high symmetry and is essentially sinusoidal (2400 Hz) for consecutive values of binary 1's.

2.2.2 PACKET STRUCTURE

The RNCP protocol is a packet-radio message protocol. Bits are packaged into message symbols, which are in turn arranged into RNCP messages. The RNCP protocol supplies network and channel control information via the packetized messages in addition to text information. The main highlights of the RNCP packet structure are listed below and are described in following sections:

- mandatory message header with variable number (0-8) of text blocks
- 7 bit message symbols using 6 bit ASCII characters
- Channel access control bit in the 7th bit of every symbol
- Triplicated header structure (14 RNCP symbols per copy)
- BCH encoded header symbols
- Reed Solomon (63,45) encoded text blocks
- Network synchronization provided via special messages and tones

The general RNCP Packet structure is shown in Figure 2-4.

2.2.2.1 MESSAGE SYMBOLS

The bits of the RNCP protocol are used to form message symbols. Each symbol is made up of 7 bits, of which the least significant 6 bits are used for the actual ASCII symbol. The 7th or most significant bit is used for channel access control and will be discussed later on in the description for channel contention control. Message symbols are transmitted and received starting with the least significant bit.

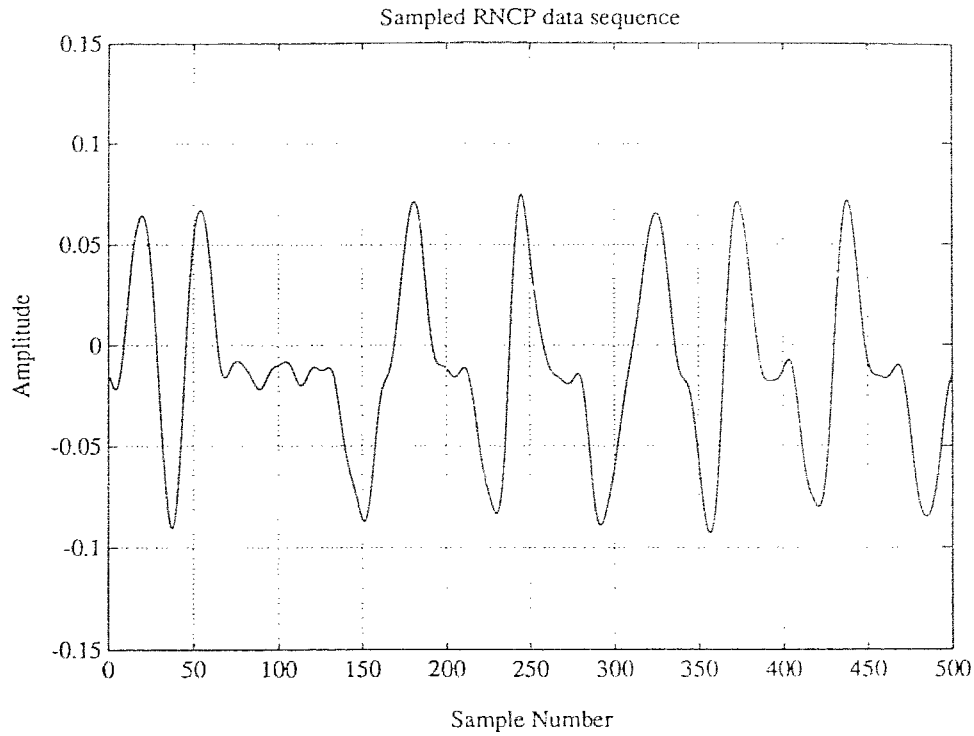


Figure 2-2 Typical Sampled RNCP Pulse Sequence

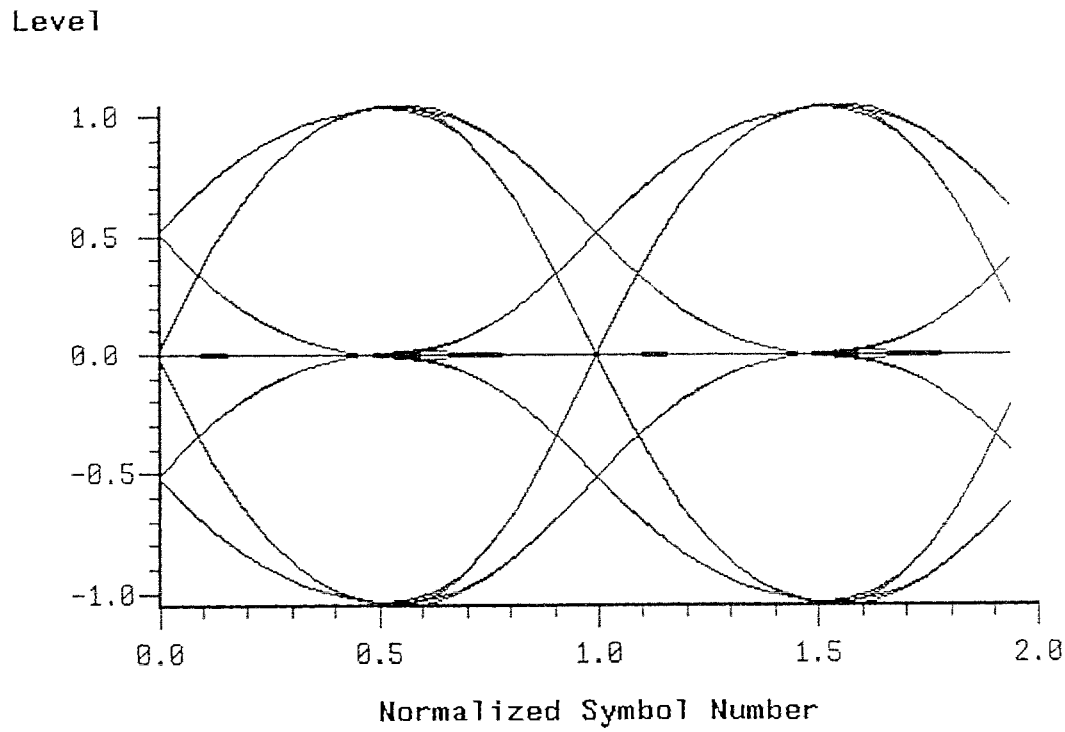


Figure 2-3 Eye Diagram for Pseudorandom RNCP Data

2.2.2.2 MESSAGE HEADER

Each RNCP message contains a mandatory triplicated header, which is majority voted by a receiver. The multiple copies of the header are used to provide protection during fading. A worst case fade duration of 20 milliseconds (one header duration) is tolerated as long as the remaining 2 header copies are received error free.

The RNCP message header is used to provide frame synchronization and control information to a target mobile unit. A copy of the RNCP message header is fixed length and is comprised of 4 symbols (28 bits) of frame SYNC codewords and 10 symbols (70 bits) of control information. The SYNC codeword symbols differ from the control information symbols in that they do not have a channel contention control bit (7th bit of a symbol). The control information in the header includes the mobile ID field, a message length, message priority field and BCH parity check bits.

The header mobile ID field allows a mobile unit to react only to messages addressed to it, while the message priority field allows the network to attach priority to messages. The parity check bits are used for error detection. Each copy of a RNCP header is encoded with a systematic BCH error-detection code, which can detect up to 7 bit errors. A receiver can compare received parity bits with the parity bits generated using the received header bits to determine if bit errors have occurred.

2.2.2.3 MESSAGE TEXT BLOCKS

An RNCP text block consists of 63 symbols of which 45 are actual text symbols and the remainder are Reed-Solomon (RS) parity check bits. An RNCP message can have anywhere from a minimum of 0 to a maximum of 8 text blocks. The symbols in a text block are normally 6 bit ASCII symbols which have been encrypted to provide a measure of security over the radio channel. Each text block is encoded with a (63,45) RS forward error correction code. The (63,45) RS code can detect v errors and s erasure fills which satisfy the equation $2v + s \leq 18$. The Reed-Solomon code is used since it is shown to be effective for burst errors which are characteristic of the fading mobile environment.

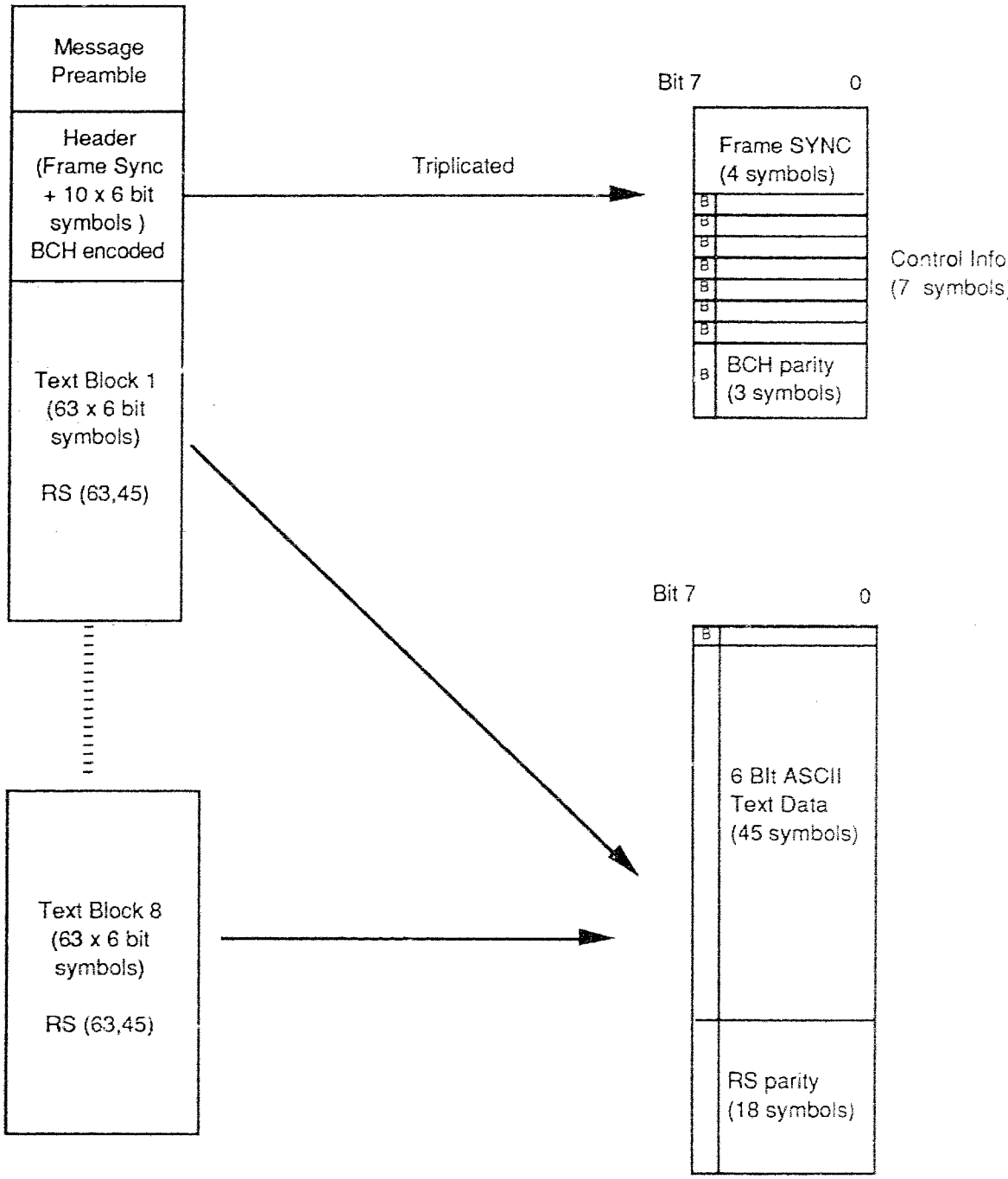


Figure 2-4 General RNCP Message Packet Structure

2.2.2.4 MESSAGE PREAMBLE

A sinusoidal preamble normally precedes the header of every RNCP message. The preamble is essentially a 2400 Hz sinusoid and is used to allow the timing recovery algorithm of a receiver to synchronize to an incoming message and to fill in the transient period on the radio channel when the base radio is first keyed up to when it achieves a steady state (the transient start-up time of a base radio is in the order of 10-20 msec). Message preamble is normally not required in systems where the radio is continuously keyed on.

2.2.2.5 CHANNEL CONTENTION CONTROL

The 7th bit of every header and text symbol, with the exception of the header SYNC codewords, contains a channel contention control bit or channel 'BUSY' bit. This bit is used for contention control for Digital-Sense-Multiple-Access (DSMA) channel protocols. The busy bit is ignored for ALOHA or CSMA channel radio channel access protocols.

The RDPC is responsible for setting and clearing the busy bit to control the radio channel. The busy bit is set in outbound messages if an inbound message transaction to the RDPC is in progress. All other mobiles in the field will sense that the busy bit is set and will be inhibited from transmitting until the current sender has completed its message transaction. Mobiles with messages to send while the busy bit is set will follow a random back-off algorithm before attempting to transmit. The RDPC clears the busy bit when the current message transaction has completed or when the channel is idle.

2.2.2.6 NETWORK SYNCHRONIZATION

The RDPC is responsible for providing synchronization information to mobiles in the field. The transceiver site transmits an idle tone when no inbound message traffic is present. This idle tone is shown in Figure 2-5. In addition to the idle tone, a network synchronization message is transmitted every 2 seconds to allow mobile units to re-synchronize their receive clocks to RDPC.

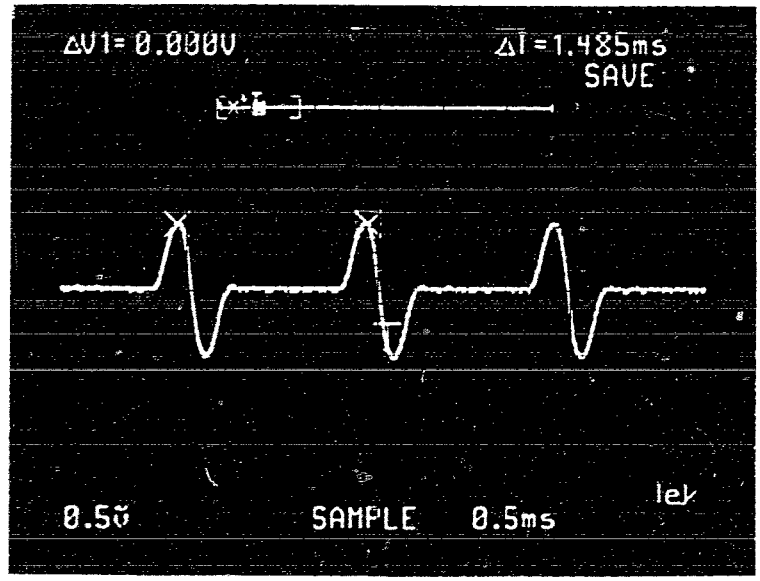


Figure 2-5 RNCP Network Idle Tone

3. ARDPC OVERVIEW

3.1 FUNCTIONAL REQUIREMENTS

The functional requirements for the ARDPC may be classified as being either core or extended functional requirements. The core functions are required in order to maintain compatibility with the first generation design and are considered to be essential for the operation of an RDPC. The extended functions represent desirable but non-essential features. The extended functions in general represent convenience improvements over the first generation design. Some extended functions are derived from first-hand field experience with the first generation units, while others are the result of the need to anticipate future requirements.

The list of functional requirements for the ARDPC are summarized below. The description for a listed requirement is given in the reference subsection indicated:

CORE FUNCTIONS

1. modulation/demodulation of RNCP message data to/from mobile units	(sec 3.1.1)
2. communications with the CC via modem link	(sec 3.1.2)
3. control of the transceiver site radio	(sec 3.1.3)
4. control of radio channel access for mobile units	(sec 3.1.4)

EXTENDED FUNCTIONS

1. VMEbus capability	(sec 3.1.5)
2. support for a user interface to a local console	(sec 3.1.6)
3. improved user diagnostics	(sec 3.1.7)
4. adjustment of modem signal levels from a local console	(sec 3.1.8)
5. FEC decoding of messages	(sec 3.1.9)
6. generation of radio channel error statistics	(sec 3.1.10)
7. RNCP message monitoring capability	(sec 3.1.11)
8. RF alarms and RSSI signal monitoring	(sec 3.1.12)
9. output of morse coded station id (CWID)	(sec 3.1.13)

3.1.1 MODEM FUNCTIONS

An RDPC's primary function is to provide modem capabilities at a transceiver site for packet-radio mobile data communications. The RDPC is responsible for demodulating inbound messages from mobile units and modulating outbound messages originating from the host computer or CC. Demodulation involves recovery of transmitted data clock, transmitted data and frame synchronization. Modulation involves converting a serial bit stream into analog pulses for transmission over the radio link.

3.1.2 CC COMMUNICATION

The CC is the source of acknowledgements to messages received at the host computer site and is the preprocessor for outbound messages generated from the host computer. In most cases, a transceiver site is situated away from the host computer site and the network may consist of many transceiver sites, each with its own base radio. In such a case, each RDPC is connected in a star network configuration to the CC via a CM. Each RDPC in the network must then communicate remotely with the host computer site via the CC using a leased-line or dial-up modem link.

3.1.3 TRANSCEIVER SITE RADIO CONTROL

The RDPC is responsible for radio control at the transceiver site. The controller keys the base radio under remote control from the host site or under local control at the transceiver site. The keying of the base radio coincides with outbound messages, which must be transmitted from a particular transceiver site.

3.1.4 RADIO CHANNEL ACCESS CONTROL

The RDPC is responsible for channel access control for DSMA based contention schemes. In DSMA based systems, the RDPC is responsible for setting and clearing the inbound channel busy bit in the RNCP message protocol. A mobile unit using DSMA channel access senses the state of the bit before attempting to transmit a message inbound to the host computer. A mobile unit is inhibited from transmitting until the channel access bit is cleared. Channel contention control is desirable for minimizing the number of corrupted messages that result when multiple mobile units access a common radio channel

simultaneously. Minimizing the number of corrupted messages increases the channel throughput by reducing the number of re-transmissions over the network.

3.1.5 VMEbus CAPABILITY

The ARDPC motherboard is required to have the capability to operate from an industry standard VMEbus. This function allows the ARDPC to be rack mounted and integrated with other communications devices using a VMEbus.

3.1.6 USER CONSOLE INTERFACE

The ARDPC is required to provide user access to all critical operating and channel status parameters from a local console. All critical ARDPC operating parameters should be adjustable from the user console. All operating information and channel status information should be available for a report display onto the user console. The user console interface function represents a significant improvement over the first generation RDPC, which could only pass operating state information via front panel LEDs.

3.1.7 IMPROVED DIAGNOSTICS

The first generation controller had only front panel LEDs to reflect hardware fault conditions. The ARDPC is required to provide better and more intelligent diagnostics for the hardware platform. The diagnostic tests should include loopback tests of the CC and RF data links and should be initiated on user demand from the console interface.

3.1.8 LEVEL ADJUSTMENTS FROM LOCAL CONSOLE

The first generation RDPC's input and output signal levels were adjusted by physically accessing potentiometers on the controller board. As a convenience feature the ARDPC MOD, DISC and MORSE code signal level adjustments are to be made from a local console rather than from the hardware. This avoids the chore of opening the chassis and accessing adjustment potentiometers to tweak the hardware. This feature is possible only because of DSP being used to process input and output signals.

3.1.9 FEC DECODING OF MESSAGES

The first generation controllers were essentially dumb modems which lacked the intelligence to perform message processing at the controller level. This resulted in all received inbound messages being passed to the CC regardless of condition, thus placing an unnecessary load on the CC. The ARDPC is required to perform message processing using FEC decoding to filter out corrupted messages and to generate error statistics. Message filtering helps to minimize unnecessary message traffic to the CC thus reducing the processing load on the CC.

3.1.10 RADIO CHANNEL MONITORING

The ARDPC is required to generate and log error statistics for RNCP messages transmitted and received. Generation of bit and packet error-statistics are useful in analyzing and tuning an RDPC within a network. The earlier RDPCs provided no indication of the number of messages rejected or received or the number of errors occurring in the inbound message text-blocks.

3.1.11 RNCP MESSAGE MONITORING

A means of echoing or dumping messages flowing through the ARDPC is a convenience feature which is valuable in analyzing a network. This feature is made possible by introducing message processing at the ARDPC level, which previously did not exist. The messages monitored must include all messages received from the CC for output onto the radio channel and all messages received demodulated from the radio link.

3.1.12 RF ALARMS AND RSSI SIGNAL MONITORING

The ARDPC is required to sample and log the RF alarm signals if available. A base station radio may have outputs for RF alarms, which provide an indication of the VSWR (Voltage Standing Wave Ratio) on the radio channel and hence provides information on the quality of the radio and antenna system

3.1.13 CWID OUTPUT

Each RDPC in field operation is required to transmit station identification for regulatory agencies such as the FCC in the U.S. or DOC in Canada. This station identification can be in the form of voice or Morse coded call letters which conform to FCC regulation 90.425. The ARDPC will use morse code of a single tone frequency to generate the station identification string.

3.2 HARDWARE ARCHITECTURE

The improved intelligence required for the ARDPC made it necessary to base a large degree of the functionality in software rather than hardware. A software design is inherently flexible and is better suited for the complex processing required for supporting the extended functions presented in section 3.1. The use of a microprocessor in the hardware design is fundamental for the implementation of an embedded software architecture. To address the need to support different modulation/demodulation schemes as detailed in the requirements and also facilitate the fast number crunching required for digital modem functions, it was decided to also incorporate a DSP processor for the modem functions.

A DSP processor with ADCs and DACs is able to convert functions previously implemented with analog circuitry, such as filtering, to be replaced with digital operations. The use of digital rather than analog operations retains the full benefits associated with digital circuitry, namely stability, precision and testability. A software based DSP modem provides a significant advantage because a change in the modulation/demodulation scheme can be implemented in software, without a change to the basic hardware. A single platform is thus able to support many different protocols, which provides a cost savings in design effort. DSP processing is also able to perform operations not possible in the analog domain. For example, analog level adjustments with a DSP processor can be made using software employing a user interface to a local console.

The final ARDPC hardware block diagram is shown in Figure 3-1. From the figure, it can be seen that the ARDPC features a duo-processor design utilizing a Motorola 68000 32 bit microprocessor and a Motorola DSP-56001 processor. The main function of the DSP processor is to perform the modem functions for the ARDPC. The DSP-56001 is slaved to the 68000 microprocessor which serves as the central control processor. Communication between the two processors is conducted via the 24-bit wide Host Interface port of the DSP-56001. The Host Interface port is a peripheral device within the DSP-56001 and appears as a memory mapped device to the 68000 microprocessor. The Host Interface port is equipped with a variety of control logic to allow the two processors to transfer data using interrupt or polling control.

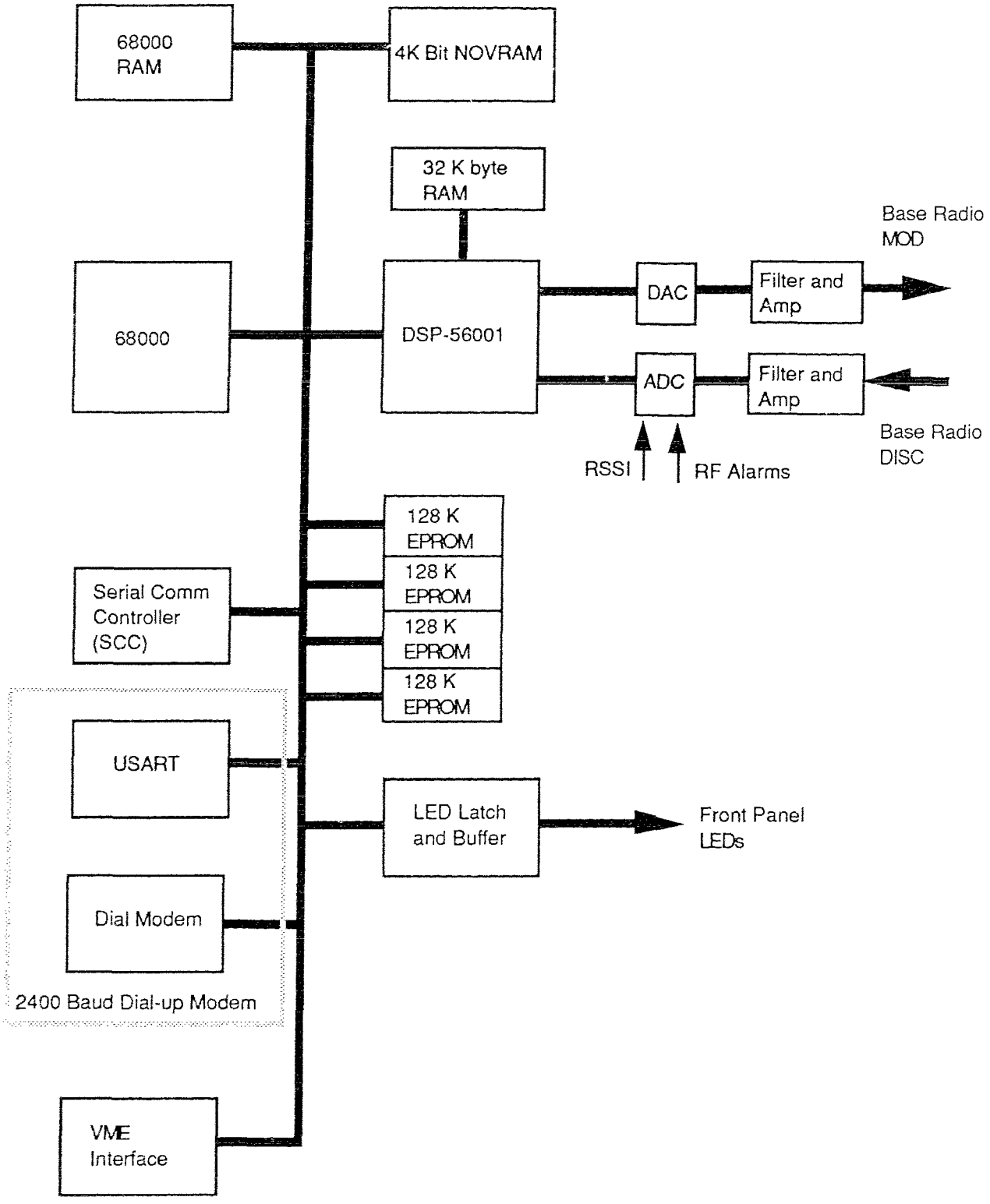


Figure 3-1 ARDPC Hardware Block Diagram

3.2.1 DEVICES UNDER 68000 CONTROL

The 68000 microprocessor controls 512K of UVEPROM which is used as storage for both 68000 and DSP-56001 code. It also accesses a battery backed RAM and 4K-bit non-volatile RAM (NOVRAM) which is used for storing data that must be involatile such as the ARDPC configuration or database parameters.

The 68000 communicates with the user console via a SCC chip or dial-up modem which allows remote access to the ARDPC. The 68000 microprocessor is interfaced to the dial-up modem via a USART chip which is under control of the 68000 control processor. The 68000 also controls a bank of front-panel Light-Emitting Diode (LEDs) indicators which may be used to signal different states or conditions of the ARDPC

A VME interface chip is provided to provide expansion capabilities for the circuit board or allow the card to be inserted into a VMEbus device. This chip is used to satisfy the auxiliary requirement for VMEbus capability.

3.2.2 DEVICES UNDER DSP-56001 CONTROL

The DSP-56001 processor is independently programmable and has access to 32K of external 1 wait state RAM for executing DSP programs. The actual DSP-56001 executable load is stored alongside the 68000 microprocessor software load in the 512K of UVEPROM. This memory is accessible only to the 68000. The DSP software must be transferred by the 68000 microprocessor from UVEPROM memory to the DSP-56001 processor RAM via the Host Interface port during a bootstrap load.

The DSP-56001 directly accesses one digital to analog converter (DAC) and two analog to digital converters (ADCs). One of the two ADCs is 8 bits wide and is multiplexed to sample the RF alarm channels and the RSSI signal from the base radio. The remaining ADC and the DAC form a set of 12 bit ports for the transfer of demodulation/modulation data samples to/from the DSP processor. The demodulation/modulation ADC/DAC pair is clocked by the DSP processor to input or output data at a sample rate of 76.8 kHz. This pair of convertors yields a dynamic range of 72 dB.

The analog demodulation/modulation signals entering or leaving the ARDPC are low passed filtered by a front-end 4 pole analog Butterworth filter with a cut-off frequency of 9.6 kHz. This cut-off frequency was chosen to accommodate potential upgrades to a 9.6 kbps data protocol. However, the analog cutoff frequency implies that the DSP processor must provide additional digital filtering for a 4.8 kHz data protocol such as the RNCP protocol.

3.3 DSP-56001 SOFTWARE REQUIREMENTS

The question as to which processor is responsible for what function naturally arises in a multi-processor design. The ARDPC software functionality that was eventually assigned to the 68000 microprocessor or the DSP-56001 processor was determined largely by considering which processor was best suited for a particular task. In some instances, both processors were equally suited for a particular function and the decision was arbitrary in assigning the function.

In the ARDPC, the 68000 microprocessor was assigned a master processor role simply because it was best suited for managing data and for making decisions. The 68000 was also best suited to implement a user interface from a local console because of the on-chip serial devices available. The DSP-56001 was assigned a slave-processor status in order to free it to become an arithmetic processing engine. This subordinate role allowed the ARDPC to take advantage of the superior arithmetic capabilities of the DSP-56001 processor. Consequently the DSP-56001 processor was assigned computationally intensive operations such as modulation and demodulation of data and generation of tones for continuous-wave identification (CWID) output.

The software requirements for the DSP-56001 are grouped using the following categories:

- General Operational Requirements
- Inbound message processing
- Outbound message processing
- Console adjustments
- RF alarm and RSSI signal monitoring.

The software requirements assigned to the DSP-56001 are listed below. A description for each requirement may be found in the listed sub-section.

GENERAL OPERATIONAL REQUIREMENTS

1. Inter-processor communication with the 68000 (sec 3.3.1)
2. POST diagnostics (sec 3.3.2)
3. Full duplex modem operation (sec 3.3.3)

INBOUND MESSAGE PROCESSING

4. Sampling of the DISC line (sec 3.3.4)
5 Low pass filtering of the sampled DISC data (sec 3.3.5)
6. Removal of DC offsets (sec 3.3.6)
7. Bit timing recovery (sec 3.3.7)
8. Data recovery (sec 3.3.8)
9. Frame and symbol synchronization (sec 3.3.9)
10. Message Header processing (sec 3.3.10)
11. Interprocessor communication with the 68000 (sec 3.3.11)

OUTBOUND MESSAGE PROCESSING

12. Pulse shaping (sec 3.3.12)
13. Busy bit insertion (sec 3.3.13)
14. Message preamble generation (sec 3.3.14)
15. Morse code station id (sec 3.3.15)
16. Idle tone generation (sec 3.3.16)
17. Interprocessor communication with the 68000 (sec 3.3.17)

CONSOLE ADJUSTMENTS (sec 3.3.18)

RF ALARMS AND RSSI SIGNAL MONITORING (sec 3.3.19)

3.3.1 INTERPROCESSOR COMMUNICATION

The DSP-56001 is required to maintain communications with the 68000. All commands for action will be initiated by the 68000 microprocessor which serves as the control processor. The control processor will communicate with the DSP processor using a command-response protocol based on the passing of information via the Host Interface port of the DSP-56001 processor. The Host Interface port provides for interrupt driven or polling transfer of 24 bits of information at any time between the control processor and DSP processor and vice-versa.

3.3.2 POST DIAGNOSTICS

The DSP-56001 must provide a basic set of "power on self test" (POST) diagnostics for detection of hardware faults on startup. The following standard tests should be included:

- RAM test
- checksum of DSP-56001 code
- Interrupt clock rate test
- ADC diagnostic
- Analog filter diagnostic

A read/write/verify diagnostic of all internal and external static RAM should be performed prior to downloading DSP-56001 code. A running checksum should be performed on all DSP-56001 instructions written to program memory to verify the integrity of the code. Diagnostics need to be provided for verifying the rate of all interrupt clocks, the accuracy of the various ADCs and the accuracy of the analog low pass filter.

The POST diagnostics should be initiated by the 68000 using the HI port message interface. Any failure of POST diagnostics should be reported to the 68000 using the same interface.

3.3.3 FULL DUPLEX MODEM OPERATIONS

The ARDPC must be able to perform full duplex 4800 bps operation using the RNCP message protocol. The flow of messages to and from the radio channel must be limited by the radio channel and not the RDPC in full duplex operation. The ARDPC will be configured for full or half duplex operation from the user console.

3.3.4 DISC DATA SAMPLING

The DISC line of the base radio supplies the baseband signal from the base radio discriminator. The DSP processor is responsible for sampling the DISC line of the base radio. The data is sampled at 76.8 kHz by the hardware to provide an oversampling ratio of 16 to 1. The sampling instances are determined by a sampling clock whose signal is used to generate interrupts to the DSP processor.

3.3.5 LOW PASS FILTERING

The sampled DISC signal must be low pass filtered by the DSP processor to reduce out of band noise in the frequency range 2.4 - 9.6 kHz. Frequencies above 9.6 kHz are filtered by the front-end analog Butterworth filter in the ARDPC hardware. The cutoff frequency of the analog filter is chosen so that a 9.6 kbps data rate can be supported by the ARDPC.

3.3.6 DC OFFSET REMOVAL

The ARDPC is AC coupled and therefore possesses a finite recovery time for a DC step-input. Such a step input is present under normal operating conditions when the base radio encounters the startup of a carrier signal on the radio channel. The presence of DC in the DISC signal has adverse effects on algorithms which rely on fixed DC thresholds for operation. Some form of DC-offset recovery must be implemented for such algorithms.

3.3.7 BIT TIMING RECOVERY

The DSP-56001 must self-synchronize to the data and the algorithm chosen must be able to adjust for jitter in the timing signal once a lock has been established on the symbol centers of the bit stream. The DSP-56001 must buffer samples of the bit stream in order for the timing recovery algorithm to search for the the best decision sample within the available samples.

3.3.8 BIT DECISION

The received data bits are elements in the domain {-1,0,1} because of the pseudo-ternary nature of the RNCP baseband signal. The data recovery algorithm must differentially decode the transmitted bit sequence and map the bits into the domain {0,1}

3.3.9 FRAME AND SYMBOL SYNCHRONIZATION

The DSP processor must determine message boundaries from the serial bit stream by scanning the received bit stream for the SYNC bit pattern in the RNCP message protocol. The SYNC pattern is triplicated in a RNCP message header and consists of 28 consecutive bits uniquely defined for the RNCP protocol. A condition of message SYNC is established if a majority vote of a sequence of scanned bits are equivalent to the SYNC

bits. Once the SYNC codewords are located, the symbol boundaries are known and the bits of the message header and text blocks may be grouped into 7 bit RNCP symbols. The extracted symbols must be buffered by the DSP-56001 for post-processing.

3.3.10 MESSAGE HEADER PROCESSING

The DSP-56001 is required to perform BCH decoding on each extracted message header to determine if the message header contains bit errors. The RNCP message header is received in triplicated form and must be de-triplicated by using majority voting before BCH decoding can be performed. A message header will be labelled 'good' if the operation of BCH decoding results in no errors otherwise the header will be marked 'bad'. The text portion of a message with a bad header will be discarded and only the header will be forwarded to the 68000 microprocessor.

3.3.11 68000 COMMUNICATION

The DSP-56001 is required to transmit a recovered RNCP message in header and text block segments to the 68000. Each header or text block will be framed with a sequence of bytes in the form of a 68000 message to identify the start and end of a header or text block segment. The extracted header of all messages will be transmitted to the 68000 but no text is to be sent with a message with a bad header.

3.3.12 PULSE SHAPING

The DSP-56001, upon command from the 68000, will receive outbound RNCP messages in header and text block segments for generation of pulses for output onto the radio channel. The messages appear to the DSP-56001 as a serial bit stream packetized on byte boundaries and on header and text block segments. A distinction must be made between header or text data because all symbols of a RNCP message have busy bit information with the exception of the SYNC codewords of the RNCP header.

Each output pulse for a bit will be oversampled by a factor of 16 (i.e. 16 modulation samples per pulse). Each RNCP symbol will be received on a byte boundary from the 68000 with only the least significant 7 bits being part of the symbol. The bits of each symbol will be modulated starting with the least significant bit and progressing towards the most significant bit.

3.3.13 BUSY BIT INSERTION

Busy bit insertion applies only for full-duplex base radio operations. The seventh bit or busy bit of a RNCP symbol is updated with the current status of inbound message processing. The busy bit of a RNCP symbol will be set to 1 if inbound message processing has detected SYNC and is actively receiving a message otherwise the busy bit will be set to 0 to indicate an idle inbound channel.

3.3.14 PREAMBLE GENERATION

The DSP-56001 will output the preamble pattern for the RNCP protocol when instructed to by the 68000. The RNCP preamble pattern consists of a serial pattern of consecutive 1's which resembles a 2400 Hz sine wave when modulated. The preamble pattern will be output by the DSP-56001 until the 68000 terminates the pattern. The duration of the preamble pattern will be a configurable parameter, which will be a part of the database for the 68000.

3.3.15 MORSE CODE STATION ID

The RDPC is required to output a morse code station identification string from each transceiver site as part of the radio requirements for the FCC. The morse code identification string will consist of a series of tone-on and tone-off signals. The DSP-56001 software must be able to generate tones of 600, 1200 and 1800 Hz to satisfy FCC regulation 90.425. The 68000 will be responsible for the scheduling and timing of the tone-on, tone-off commands to the DSP-56001.

3.3.16 IDLE TONE INSERTION

The DSP-56001 will broadcast the RNCP idle tone sequence in the absence of outbound RNCP messages. The RNCP idle tone contains busy bit information and can be terminated only on a idle tone boundary by a valid outbound RNCP message.

3.3.17 68000 COMMUNICATIONS

The DSP processor must communicate with 68000 to receive outbound RNCP messages in header and text segments. Each header or text segment will be framed with an

identification string to identify each segment. The DSP processor will acknowledge the completion of the modulation of message. The DSP processor will warn the 68000 of an modulation data underrun condition with an underrun message to the HI interface.

3.3.18 CONSOLE ADJUSTMENT OF SIGNAL LEVELS

A desired improvement of the ARDPC was to make the adjustment of critical levels possible from the user console instead of having to tweak a potentiometer on the actual hardware. This feature is made possible by the use DSP in the modem because analog output levels are directly related to the digital levels input to the DACs. The MOD and DISC levels can be adjusted by simply applying a gain value to the MOD and DISC processing algorithms. This gain can be specified by a user from the user console and can be transferred to the DSP processor by the 68000.

3.3.19 RF ALARM AND RSSI SIGNAL MONITORING

The DSP-56001 is responsible for periodically sampling and processing various RF alarms and the RSSI signal from the base radio. The RF alarms provide an indication of the voltage standing wave ratio on the radio channel and provides information on the quality of the radio and antenna system. The DSP-56001 must buffer the current samples taken and be able to provide the values on demand to the 68000.

The Radio Signal Strength Indicator (RSSI) provides information on the relative strength of the received radio signal. The level of this signal is useful for applying antenna diversity and for quantifying the quality of the received data.

4.0 DSP-56001 SOFTWARE ARCHITECTURE

4.1 CONTEXT OVERVIEW

The context diagram for the DSP-56001 processor is shown in Figure 4-1. From the diagram the following inputs and outputs are identified:

INPUTS

1. control signal from the sample rate (76.8 kHz) clock (sec 4.1.1)
2. control signal from the bit rate (4.8 kHz) clock (sec 4.1.2)
3. data and control information from the 68000 (sec 4.1.3)
4. data from RF alarms and RSSI signal (sec 4.1.4)
5. data from the DISC output of the base radio (sec 4.1.5)

OUTPUTS

6. data and control information to the 68000 (sec 4.1.6)
7. data to the MOD input of the base station radio. (sec 4.1.7)

A description of each of the above inputs and outputs is provided in the reference sub-section provided.

4.1.1 SAMPLE RATE CLOCK (76.8 kHz)

The sample rate clock signal is used by the DSP-56001 to control the sampling of inbound and outbound radio channel data. This clock signal provides interrupts to the DSP-56001 at a nominal rate of 76.8 kHz. The DSP-56001 interrupt service routine (isr) for the sample clock interrupt inputs samples from the DISC output of the base radio and/or outputs samples of the baseband modulation waveform to the MOD input of the base radio.

4.1.2 BIT RATE CLOCK (4.8 kHz)

The bit rate clock signal is provided to synchronize activities which must be completed at a symbol or bit time. This clock signal is used to generate interrupts in the DSP-56001 at a nominal rate of 4.8 kHz and is derived from the sample rate clock signal. The DSP-56001 interrupt service routine (isr) for the bit rate clock performs operations which are critical at the bit rate. These activities include generation of samples for the baseband modulation waveform and maintenance of the internal DSP software clock.

4.1.3 68000 TO DSP-56001 COMMUNICATION

The control processor (68000) supplies control and data information to the DSP processor. All communication between the two processors is asynchronously transacted via the Host Interface (HI) port of the DSP-56001. The HI port contains double-buffered full duplex data registers with different handshake protocols to facilitate the transfer of information between the 68000 and DSP processors.

Control information transacted from the 68000 is in the form of coded message directives formatted in packets of 24 bits. Data transferred from the 68000 includes header and text block segments of RNCP messages received by the 68000 from the communications controller.

4.1.4 RF ALARMS AND RSSI

The DSP-56001 is required to sample the RF alarms and RSSI signals periodically. The RF alarms provide an indication of the Voltage Standing Wave Ratio (VSWR) on the radio channel and hence provides information on the quality of the radio and antenna system. The RSSI provides information regarding the relative strength of an inbound radio signal.

4.1.5 FM DISCRIMINATOR

The FM discriminator (DISC) supplies the analog baseband RNCP data signal when data is being received. The DISC signal is filtered by an analog low pass filter before being sampled by the DSP-56001. Each received inbound pulse representing a bit is converted by sampling into a sequence of 16 digital samples by the ADC connected to the DSP-56001.

4.1.6 DSP-56001 TO 68000 COMMUNICATION

The DSP processor supplies data and control information to the 68000 via the HI port. The control information is in the form of processing status and response messages to commands initiated by the 68000. The data transferred includes demodulated RNCP messages in header and text block segments. The RNCP headers are left in triplicated format and are BCH decoded before being passed to the 68000.

4.1.7 MODULATOR

The DSP-56001 writes the MOD DAC with modulation samples which are converted to analog waveforms and are sent to the MOD input of the base site radio. The base radio then frequency shifts the spectrum of the data to the desired carrier frequency.

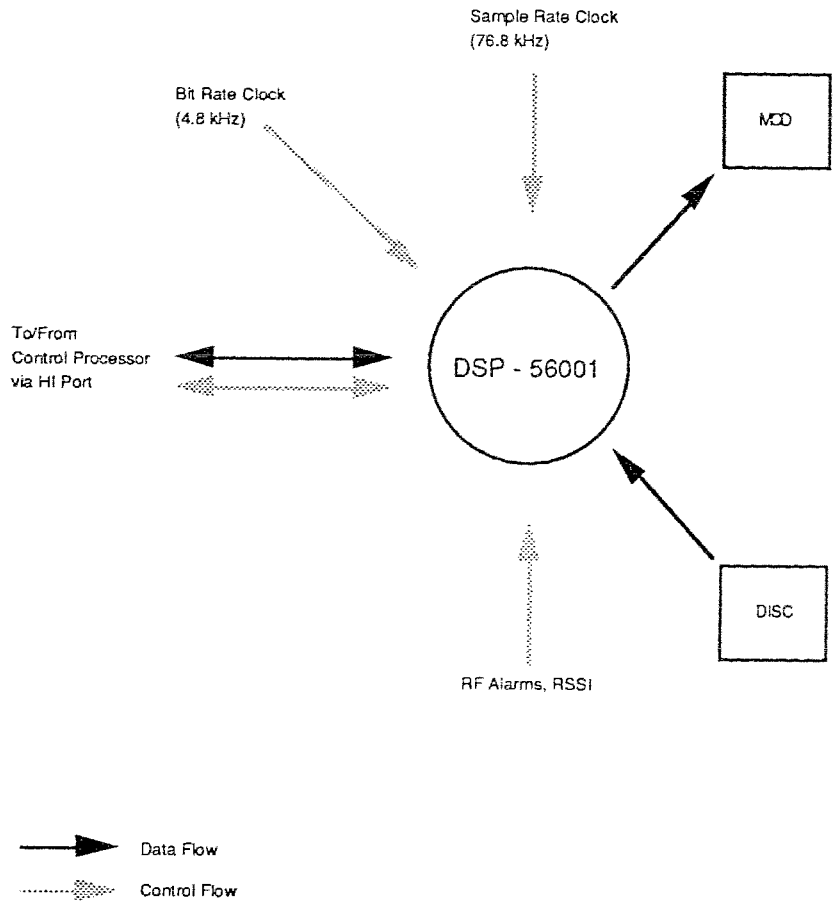


Figure 4-1 DSP-56001 Context Diagram

4.2 LEVEL ZERO SOFTWARE DECOMPOSITION

The DSP-56001 software at the highest level consists of 5 essentially independent routines; two of which are interrupt driven. The data and control flow diagram for the level zero modules is shown in Figure 4-2. The following modules are identified:

LEVEL ZERO MODULES

1. BOOT	(Bootstrap Loader)	(sec 4.3)
2. IO	(Sample Rate isr)	(sec 4.4)
3. SC	(Bit Rate isr)	(sec 4.5)
4. DIAGS	(Diagnostics Handler)	(sec 4.6)
5. MAIN	(Main Background Processor)	(sec 4.7)

BOOT is the bootstrap loader for the DSP-56001 software and executes only after a reset or power on condition to transfer code to the DSP-56001 memory. IO services the 76.8 kHz sample rate interrupt to transfer data to/from the modulator DAC and the discriminator ADC. SC services the 4.8 kHz bit rate interrupt and performs the processing which need to synchronized at the symbol rate. DIAGS performs a set of user specified hardware diagnostics for the ARDPC. MAIN is a main background processing task which performs the processing which is not time critical in the DSP-56001.

The formal decomposition of the level zero modules is provided in the reference sub-section indicated. A description of the data stores in Figure 4-2 is provided in section 4.8 which describes the data dictionary.

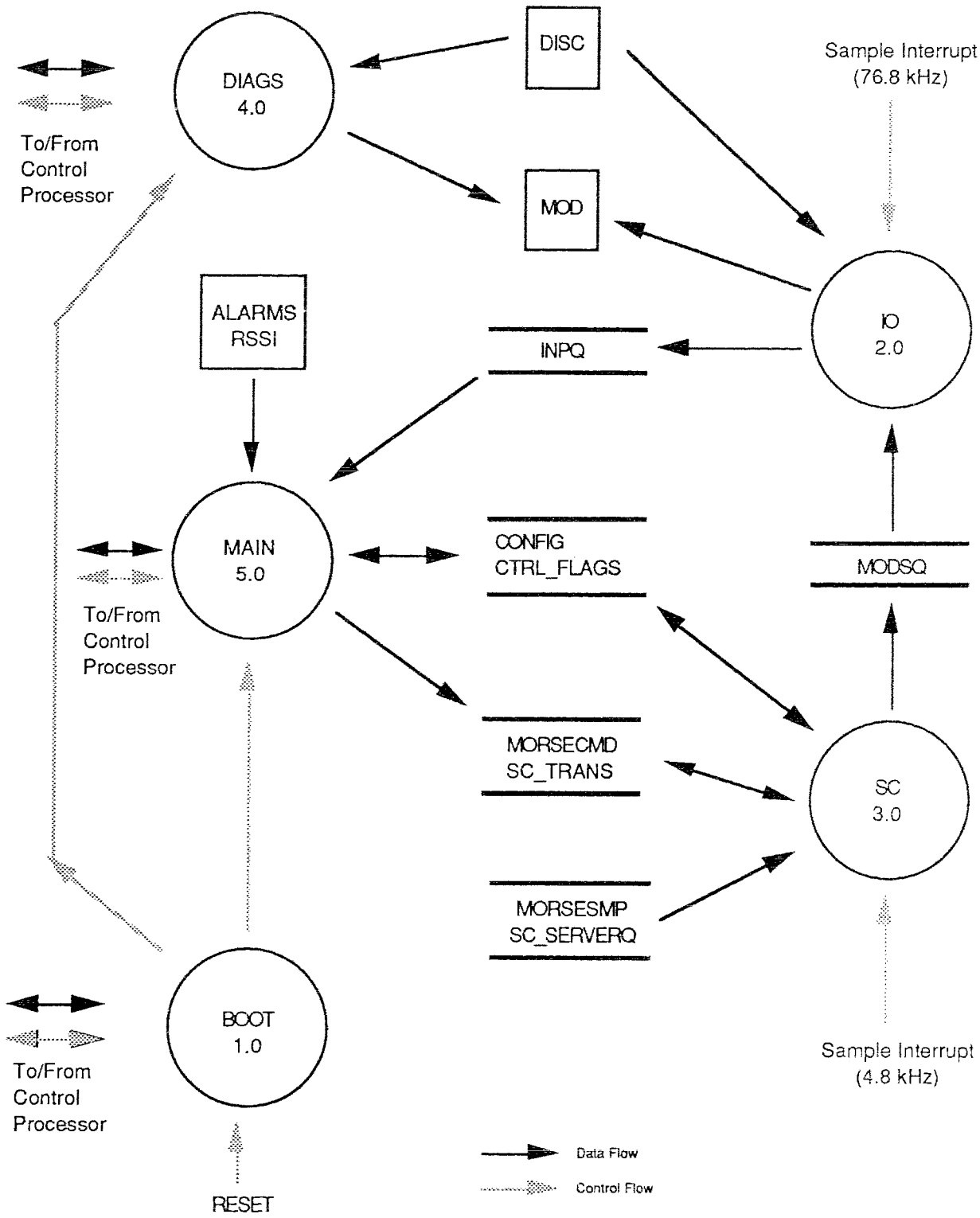


Figure 4-2 Level Zero Software Decomposition

4.3 DECOMPOSITION OF BOOT

The decomposition of BOOT consists of a single level only.

BOOT is an independent program which executes only after a power-on or hardware RESET condition. The main function of BOOT is to perform power on self-test (POST) diagnostics and the loading of the DSP-56001 main application code into external DSP RAM.

The DSP-56001 enters into a special bootstrap mode of operation upon power-up or reset. An intrinsic ROM based bootstrap program executes within the DSP-56001 while in the bootstrap mode. This resident program polls the HI port continuously for data from the 68000 processor and transfers any data received from the port into DSP-56001 internal memory. The 68000 processor is responsible in bootstrap mode for retrieving BOOT code from the UVEPROM and for transferring it to the DSP-56001 via the HI port using the intrinsic bootstrap loader. The transferred BOOT code automatically begins execution at the DSP-56001 starting address of $40 hex upon the completion of the BOOT code transfer.

BOOT executes by initially performing POST diagnostics to determine if it is safe to start the transfer of the DSP-56001 main application code from the 68000. The main application code is transferred from the 68000 into external DSP-56001 RAM by BOOT if the POST diagnostics pass otherwise a fatal error condition is communicated to the 68000. The final operation by BOOT is to branch to the main application code which never again executes BOOT.

4.4 DECOMPOSITION OF IO

The module IO consists of a single level only. The data and control flow diagram for IO is shown in Figure 4-3.

IO services the 76.8 kHz sample interrupt in the DSP-56001. IO takes advantage of the fast interrupt capability of the DSP-56001 and consists of 2 instructions. One of the instructions is used to read a sample from the DISC ADC and transfer it to the input data queue (INPQ) while the second instruction writes the MOD DAC with a sample from the modulation data queue (MODSQ). In order to facilitate the use of the DSP-56001 fast

interrupt, a pair of hardware registers are permanently dedicated to INPQ and MODSQ to serve as input and output pointers respectively.

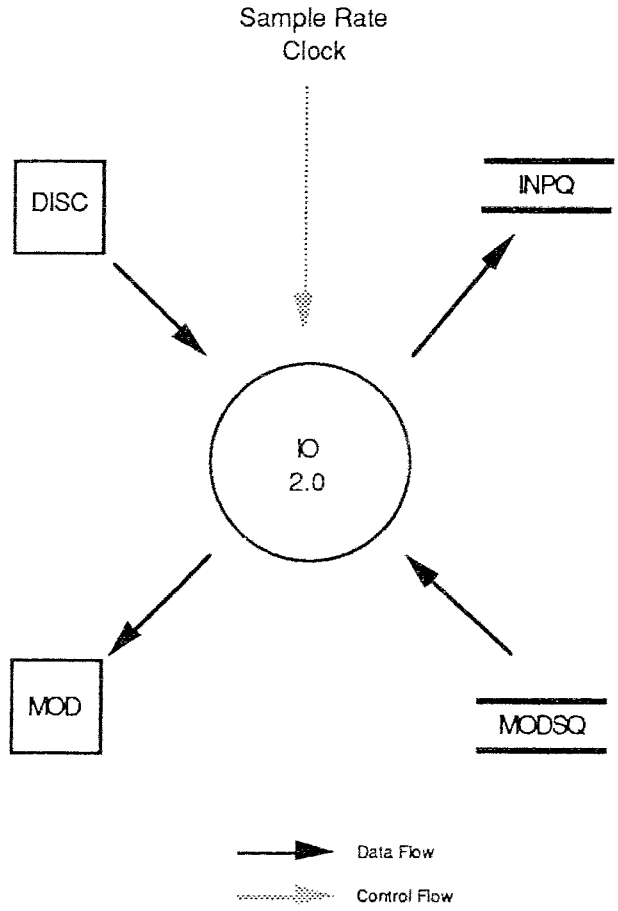


Figure 4-3 Data and Control Flow for IO

4.5 DECOMPOSITION OF SC

SC services the 4.8 kHz bit rate interrupt and is responsible for generating the modulation data samples for output to the MOD DAC. SC is decomposed into 2 routines which are specific to the type of modulation data being processed. The 2 routines are SC_msg_server and SC_morse_server. SC_msg_server is responsible for generating modulation samples for the idle tone, message preamble and RNCP message data. SC_morse_server is responsible for generating the pure tone signal for the morse code station id. Only one SC routine executes at any one time. The vectored address for the bit rate clock isr is written with the address of either SC_msg_server or SC_morse_server. Which address is written is determined by the module MAIN which interprets command messages and performs modulation data management. During normal processing SC_msg_server will be servicing the symbol clock interrupt. SC_morse_server will service the symbol isr when the 68000 schedules the output of the morse station id.

4.5.1 SC_msg_server

The data and control flow diagram for SC_msg_server is shown in Figure 4-4. SC_msg_server communicates with the background software via the SC_TRANS data store. This data store is under the control of the background routine MAIN_mword_ctrl which is responsible for scheduling and managing the data which is to be modulated by SC. SC_TRANS contains both data and control information for SC_msg_server.

The CONFIG and CTRL_FLAGS data stores contain the essential DSP-56001 configuration information for the ARDPC. The CONFIG and CTRL_FLAGS data stores supplies the modulation gain and the state of the channel busy bit for SC_msg_server.

Modulation samples generated by SC_msg_server are initially placed in the SC_SERVERQ which contains 16 samples (1 bit time) of data. The processing in SC_msg_server is pipelined so that it transfers a buffer of samples from SC_SERVERQ to the modulation data buffer (MODSQ) and also refreshes SC_SERVERQ each time it is emptied.

4.5.2 SC_morse_server

The data and control flow diagram for SC_morse_server is shown in Figure 4-5. SC_morse_server communicates with the background via the MORSECMD common section. MORSECMD is under the control of the background routine MAIN_morse_ctrl which is responsible for scheduling and controlling the morse code pattern to be output by SC_morse_server.

The MORSESMP data store is analogous to SC_SERVERQ in SC_msg_server while the CONFIG and CTRL_FLAGS serve the same purpose for SC_morse_server as they do for SC_msg_server.

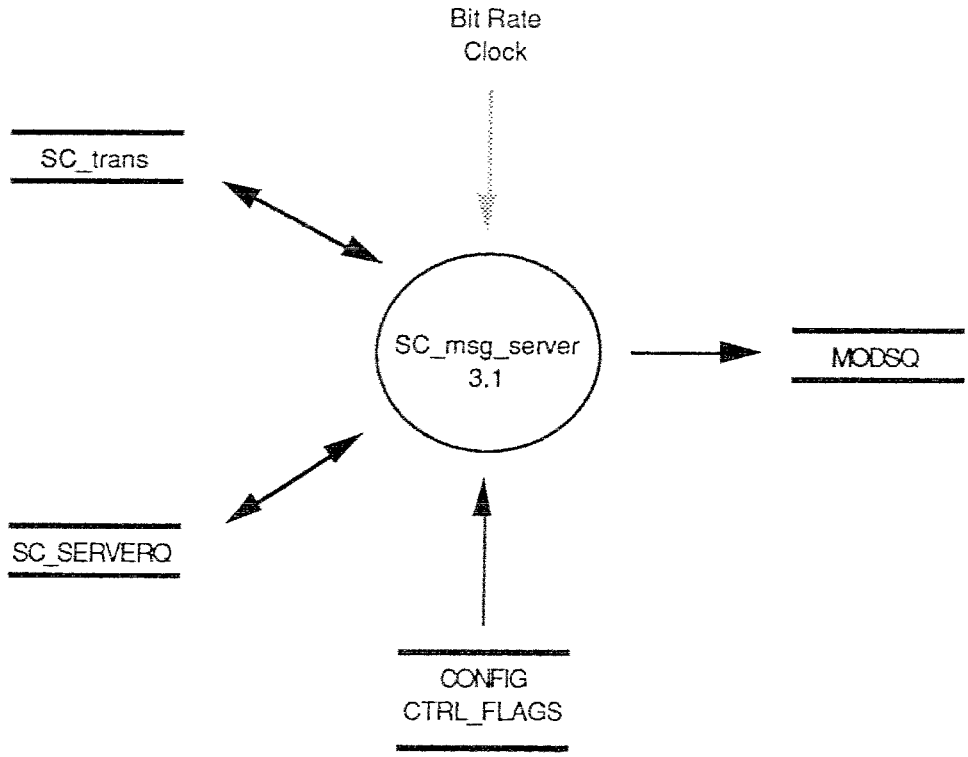


Figure 4-4 Data and Control Flow for SC_msg_server

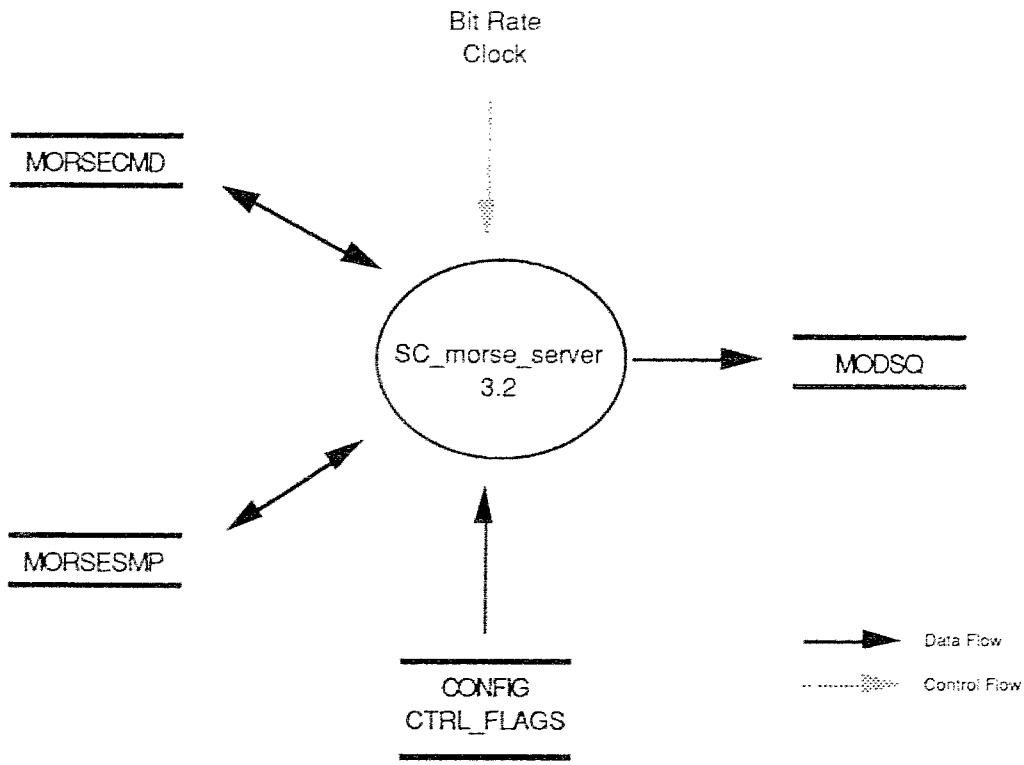


Figure 4-5 Data and Control Flow for SC_morse_server

4.6 DECOMPOSITION OF DIAGS

DIAGS performs a comprehensive set of diagnostics on user demand which supplements the POST diagnostics performed by BOOT. DIAGS executes independently of the steady state main application code and is loaded on demand by the 68000 over existing DSP program memory. DIAGS is normally intended to execute after BOOT which performs POST diagnostics.

The decomposition of DIAGS consists of a single level and the following routines:

• DIAGS_bus	(Memory overlap test)	(sec 4.6.1)
• DIAGS_ram	(Non-destructive RAM diagnostic)	(sec 4.6.2)
• DIAGS_checksum	(Program memory checksum)	(sec 4.6.3)
• DIAGS_filter	(Analog filter diagnostic)	(sec 4.6.4)
• DIAGS_adc8	(8 bit ADC diagnostic)	(sec 4.6.5)
• DIAGS_adc12	(12 bit ADC diagnostic)	(sec 4.6.6)

A general description of the above diagnostics is presented in the reference sub-section provided.

4.6.1 DIAGS_bus (External RAM overlap test)

DIAGS_bus performs a non-destructive read of all external RAM location with the address 2^k where k=(0...14). A random pattern is written into the selected memory location and is read back and compared with what was written.

4.6.2 DIAGS_ram (Non-destructive RAM test)

DIAGS_ram performs a non-destructive read/write test of external DSP ram using the following operations:

- an external RAM memory location is read and saved
- a random pattern is written into the same memory location
- the RAM address is re-read
- a comparison of the written and re-read values is compared
- the saved RAM value is restored

The above operations are repeated for all sequential external RAM locations.

4.6.3 DIAGS_checksum (Program checksum diagnostic)

DIAGS_checksum performs a checksum of DIAGS software by summing all program words modulo 24 bits. The grand total is then added to a value supplied via a command message from the 68000 and if the resultant sum is zero the test passes otherwise it fails.

4.6.4 DIAGS_filter (Analog filter diagnostic)

DIAGS_filter tests the quality of the analog low pass (9.6 kHz) filter of the ARDPC. A series of pure test tones are generated above and below the cut-off frequency of the analog low pass filter. These tones are generated with the aid of the internal SINE ROM table of the DSP-56001 and are nominally set at 6982, 7680, 85333, 9600, 10971, 12800 and 15360 Hz (based on SINE ROM table skip factors of 11, 10, 9, 8, 7, 6 and 5 respectively).

Samples of each tone are sent to MOD and the signal is looped back to the DISC line of the ARDPC. The peak value output from the filter for each tone is found and compared against reference values based on the filter components.

4.6.5 DIAGS_adc8 (8 bit ADC linearity test)

The 8 bit ADCs of the ARDPC are used for inputting analog RF and RSSI values into the DSP-56001. DIAGS_adc8 tests the integrity of the 8 bit ADC by looping back a reference tone through the ADC and comparing the read digital values with reference values. A pass/fail condition is returned to the 68000.

4.6.6 DIAGS_adc12 (12 bit ADC/DAC diagnostic)

The 12 bit ADC and DAC of the ARDPC are used for sampling MOD and DISC data for input and output data onto the radio channel. DIAGS_adc12 tests the integrity of the ADC and DAC pair by generating a 300 Hz waveform with the aid of the DSP-56001 SINE ROM, outputting the samples to the MOD DAC, looping back the MOD signal through DISC and reading back the samples from the DISC ADC of the ARDPC. The digital

values written to MOD DAC are compared against the digital values read back from DISC ADC and a pass/fail condition is returned to the 68000.

4.7 DECOMPOSITION OF MAIN

MAIN performs the processing which is not time critical in the DSP-56001. This includes the following:

- responding to 68000 commands via the HI port
- receiving and preparing RNCP data received from the 68000 for the outbound channel
- processing input DISC samples
- processing channel state information
- transfer of extracted inbound RNCP data to the 68000
- SYNC detection and majority voting of RNCP message data
- alignment of inbound RNCP data bit stream onto byte boundaries
- BCH decoding of inbound header data

At any time some or all of the above tasks will be waiting to be performed. The logic in MAIN polls the number of tasks waiting to be performed and distributes the processing time evenly amongst the tasks to avoid any single task from monopolizing the available processing bandwidth. Operations performed by MAIN are interrupted regularly by both SC and IO. Both isrs transfers control to MAIN upon completing execution.

The decomposition of MAIN consists of 3 levels which are discussed in sections 4.7.1 to 4.7.3.

4.7.1 LEVEL 1 DECOMPOSITION OF MAIN

The level 1 data and control flow diagram for MAIN is shown in Figure 4-6. From the diagram it can be seen that MAIN is decomposed into the following level 1 modules:

1. MAIN_ctrl	(MAIN control routine)	(sec 4.7.1.1)
2. MAIN_init	(DSP-56001 global initialization routine)	(sec 4.7.1.2)
3. MAIN_HI_rx	(HI port receive data manager)	(sec 4.7.1.3)

These modules are discussed in the reference subsections provided.

4.7.1.1 MAIN_ctrl (MAIN control routine)

MAIN_ctrl coordinates all background processing activities which execute when the sample and symbol isrs are not active. These activities include the following:

- DISC data processing
- 68000 command processing
- RF alarm and RSSI signal sampling
- Morse code and RNCP Modulation data management

All the above activities execute sequentially with equal priority in an infinite processing loop within MAIN_ctrl. Processing is arranged so that each activity is executed fully or partially in a single pass of the loop. No single activity is allowed to monopolize the processing in MAIN_ctrl.

4.7.1.2 MAIN_init (Initialization routine for the DSP-56001)

MAIN_init is responsible for initializing the DSP software in general and specifically performs the following tasks:

- Initialization of all global buffers and variables
- Initialization of vectored interrupts
- Initialization of the SCI, SSI and HI communication ports

The DSP-56001 begins execution at MAIN_init immediately after the main application DSP code has been successfully ported from the 68000 by the module BOOT. MAIN_init initializes the DSP software and enters an indefinite wait state after completion of the initialization. This wait state is exited only upon receiving a 'GO' message directive from the 68000. The 'GO' message allows the 68000 to synchronize the power-on start up activities and also contains configuration information which is extracted by MAIN_init and inserted into the CONFIG data store. MAIN_init transfers control to MAIN_ctrl after it finishes execution.

4.7.1.3 MAIN_HI_rx - (Receive RNCP data from the 68000 via the HI port)

MAIN_HI_rx is an isr which responds to the Host Receive Data Full (HRDF) HI port interrupt which is generated whenever the 68000 writes the HI port and the HRDF interrupt bit is enabled by the DSP-56001. The HI port contains either 68000 commands or RNCP message header or text block data when the interrupt is triggered. 68000 commands received at the HI port are left unread by MAIN_HI_rx and are processed at a later time by MAIN_ctrl. RNCP message data is stored by MAIN_HI_rx in the modulation data buffers MODBUF1 or MODBUF2. Which buffer is used is dependent on the buffer control information in the modulation data buffer control blocks MODCTRL1 and MODCTRL2. MAIN_HI_rx uses the information in the buffer control blocks to search for and determine whether a buffer is currently in use or not.

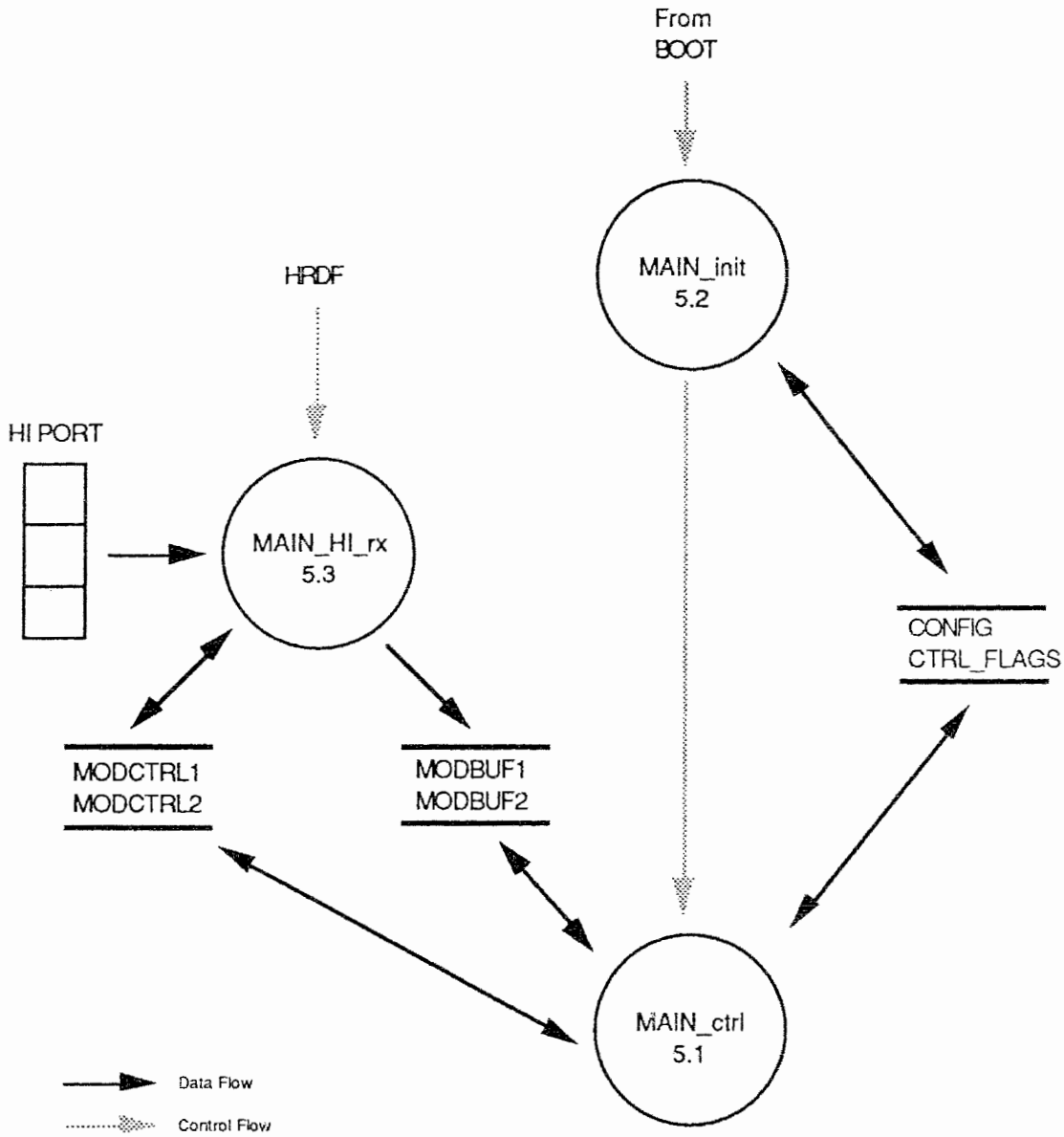


Figure 4-6 Level 1 Decomposition of MAIN

4.7.2 LEVEL 2 DECOMPOSITION OF MAIN

The level 2 decomposition of the module MAIN consists of the further decomposition of MAIN_ctrl. There is no further decomposition of the other level 1 routines MAIN_init and MAIN_HI_rx.

The data flow and control diagram for MAIN_ctrl is shown in Figure 4-7. From the diagram it can be seen that MAIN_ctrl is further decomposed into the following lower level routines:

• MAIN_cmd_ctrl	(68000 command handler)	(sec 4.7.2.1)
• MAIN_mword_ctrl	(Modulation data manager)	(sec 4.7.2.2)
• MAIN_morse_ctrl	(Morse code data manager)	(sec 4.7.2.3)
• MAIN_disc_proc	(DISC data processor)	(sec 4.7.2.4)
• MAIN_rfa_handler	(RF alarm manager)	(sec 4.7.2.5)

These routines are discussed in the reference subsections provided

4.7.2.1 MAIN_cmd_ctrl - (68000 command handler)

MAIN_cmd_ctrl is responsible for processing command messages to and from the 68000. MAIN_cmd_ctrl performs such tasks as responding to 68000 commands and transferring inbound message data to the 68000. Response messages to the 68000 which cannot be transmitted while MAIN_cmd_ctrl is executing is placed onto a backlog message queue for delayed transmission. MAIN_cmd_ctrl is responsible for maintenance of the backlog message queue.

4.7.2.2 MAIN_mword_ctrl - (Modulation data manager for idle tone, preamble and RNCP message)

MAIN_mword_ctrl is responsible for scheduling the output of all modulation data with the exception of morse code data. The data types handled by MAIN_mword_ctrl include the idle tone, message preamble and RNCP message data. The idle tone is the default data type and is always output in the absence of RNCP message data or morse code station id.

Message preamble precedes the actual message data in RF signalling schemes where the base radio is not constantly keyed. The preamble provides a synchronization sequence for receiving mobile units and its duration is dependent on the 'transmitter turn on' time of the base radio. Message preamble can be terminated directly by the 68000 or indirectly by the presence of outbound RNCP message data.

RNCP message data received by the DSP-56001 for output is sent 3 symbols at at time (i.e. 21 bits) to the modulation task SC_msg_server via the SC_TRANS data store. MAIN_mword_ctrl is responsible for retrieving the data from the modulation data buffers MODBUF1 and MODBUF2 using control information in the buffer control blocks MODCTRL1 and MODCTRL2. MAIN_mword_ctrl sends an acknowledgement message to the 68000 upon successful modulation of each RNCP message. The acknowledgement message contains a sequence number which identifies each message.

4.7.2.3 MAIN_morse_ctrl - (Morse code modulation data manager)

MAIN_morse_ctrl is responsible for switching the bit rate isr to produce pure tones for generation of the morse code station identification string. MAIN_morse_ctrl is invoked when the 68000 requests the output of morse code data. MAIN_morse_ctrl writes the address of SC_morse_server into the interrupt vector location for the bit rate interrupt in response to the 68000 request.

Morse code signal output is performed using a series of 'tone on' or 'tone off' commands from the 68000. These commands are processed by MAIN_cmd_ctrl and are communicated to MAIN_morse_ctrl which then forwards the information to SC_morse_server via the MORSECMD data store.

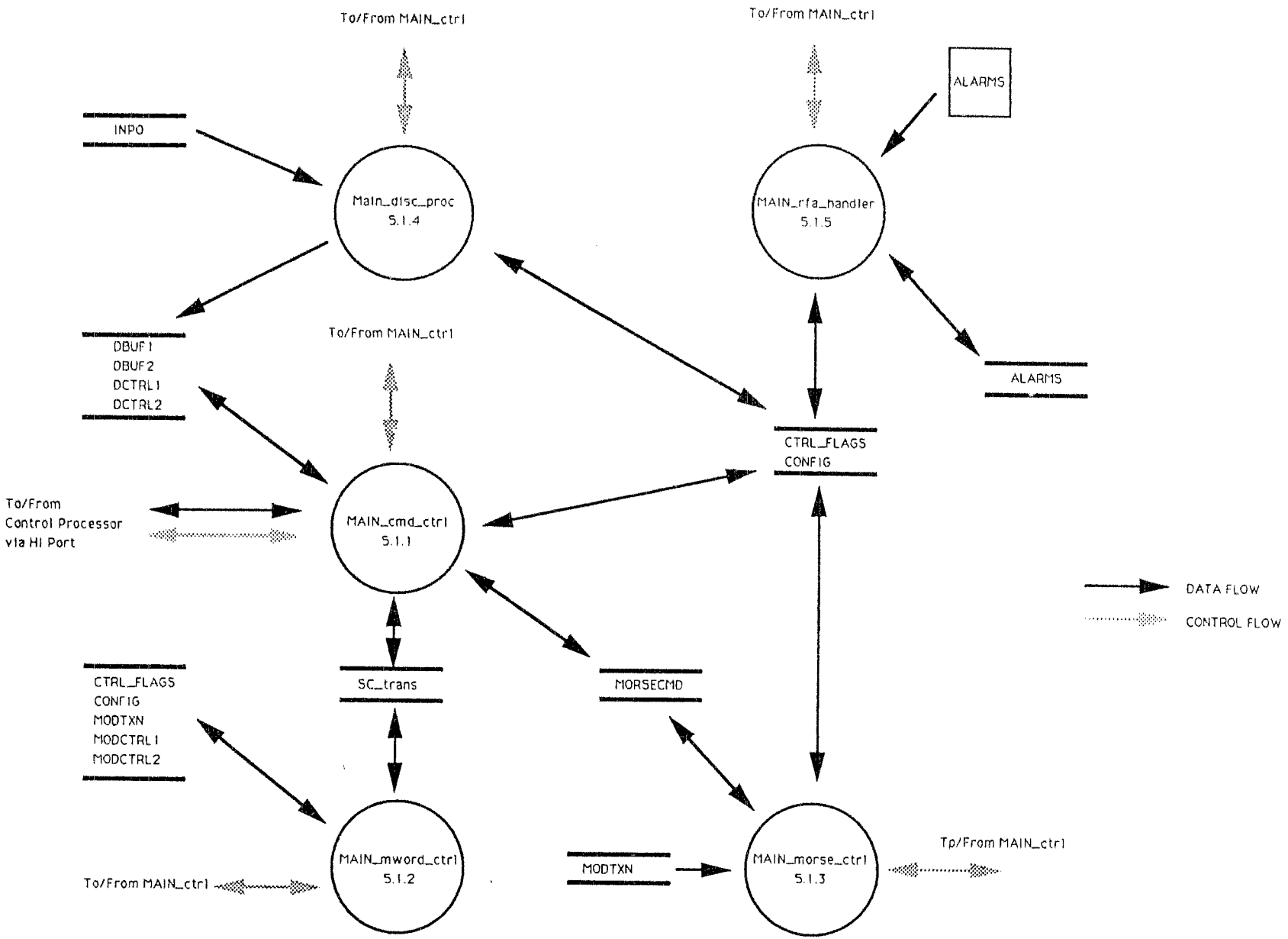


Figure 4-7 Level 2 Decomposition of MAIN

4.7.2.4 MAIN_disc_proc (DISC data processor)

MAIN_disc_proc is responsible for controlling the flow of processing for DISC data which includes BCH decoding of message headers and demodulation of received bits.

MAIN_disc_proc continuously monitors the DISC data buffers DBUF1 and DBUF2 for the presence of a newly demodulated RNCP header using the information in the DISC buffer control blocks DCTRL1 and DCTRL2. If a header is available it is BCH decoded by MAIN_disc_proc. MAIN_disc_proc will discard the text and forward only the header to the 68000 if errors are detected in the decoded header. The DISC bit alignment routine is allowed to return to a hunt for SYNC mode in such a case.

MAIN_disc_proc invokes the lower level routine DISC_PROC_dpll to demodulate received data. This routine is described in the level 3 decomposition of MAIN.

4.7.2.5 MAIN_rfa_handler (RF alarm and RSSI signal manager)

MAIN_rfa_handler is responsible for scheduling and obtaining samples for all RF alarms and the RSSI signal. RF alarm samples are obtained by having the DSP software read the SCI port of the DSP-56001. The RF alarm ADC is connected to this port and supplies 8 bit samples in bit reversed sequence. MAIN_rfa_handler obtains RF alarm values in a cyclical fashion starting with the first RF alarm. These samples are buffered into a common section and the values are communicated to the 68000 by MAIN_cmd_ctrl when requested to do so.

4.7.3 LEVEL 3 DECOMPOSITION OF MAIN

The level 3 decomposition of MAIN consists of further decomposition to the level 2 routines MAIN_cmd_ctrl and MAIN_disc_proc. The decompositions for these level 3 modules are described in sections 4.7.3.1 and 4.7.3.2 respectively.

MAIN_cmd_ctrl

The data flow and control diagram of MAIN_cmd_ctrl is shown in Figure 4-8. From this diagram it can be seen that MAIN_cmd_ctrl is further decomposed into the following lower level modules:

• CMD_exec_cp_cmd	(Execute 68000 commands)	(sec 4.7.3.1)
• CMD_send_queued_msg	(Send queued message to the 68000)	(sec 4.7.3.2)
• CMD_send_disc_data	(Fetch message data from DISC data buffers)	(sec 4.7.3.3)
• CMD_cp_xmit	(Transmit data or commands to the 68000)	(sec 4.7.3.4)

These modules are discussed in the reference subsections indicated.

MAIN_disc_proc (DISC data processor)

MAIN_disc_proc is responsible for coordinating the processing of DISC data. This processing involves the following operations:

- Detection of inbound data for asserting channel multiple access protocols
- Recovery of transmitted data clock
- Recovery of data bits
- Alignment of data bits into RNCP symbols
- Alignment of RNCP symbols into header and text blocks
- BCH decoding of message headers

Figure 4-9 shows the data flow and control diagram of MAIN_disc_proc which carries out the above processing. From the diagram it can be seen that MAIN_disc_proc is decomposed into the following secondary modules:

• DISC_PROC_dpll	(Phase locked loop for timing recovery)	(sec 4.7.3.5)
• DISC_PROC_align	(Bit alignment)	(sec 4.7.3.6)
• DISC_PROC_busy	(Data detection)	(sec 4.7.3.7)
• DISC_PROC_bch_dec	(BCH decoder for RNCP message headers)	(sec 4.7.3.8)

The lower level routines of MAIN_disc_proc are discussed in the subsections indicated.

4.7.3.1 CMD_exec_cp_cmd (Execute 68000 commands)

CMD_exec_cp_cmd serves as the gateway for all 68000 commands. CMD_exec_cp_cmd is responsible for updating processing conditions and information in response to commands received over the HI port. CMD_exec_cp_cmd initiates a desired 68000 action directly by performing the action or indirectly by setting up the appropriate global common areas for execution by other routines.

4.7.3.2 CMD_send_queued_msg (Send queued DSP to 68000 messages)

CMD_send_queued_msg is responsible for transmitting messages from the BACKLOG_MSG_Q to the 68000. The BACKLOG_MSG_Q contains messages which previously could not be transmitted to the 68000 due to blocking at the 68000 side of the HI port. A message is removed from the backlog message queue if CMD_send_queued_msg is able to transmit the queued message otherwise there is no change to BACKLOG_MSG_Q.

4.7.3.3 CMD_send_disc_data (Send DISC data to the 68000)

CMD_send_disc_data is responsible for transferring packetized demodulated inbound RNCP message data to the 68000. CMD_send_disc_data will transmit only the header portion of a received message if the message header has errors detected in it. The transfer of message data from a DISC data buffer begins upon successful reception of a message header. The transfer of data from a DISC data buffer may be pre-empted by the premature detection of a SYNC condition during the operation of bit alignment (i.e. another SYNC pattern detected before the current message has been completely transferred). In such a case the message transfer in progress will be terminated and a end of data message will be issued. Processing will continue with the new message.

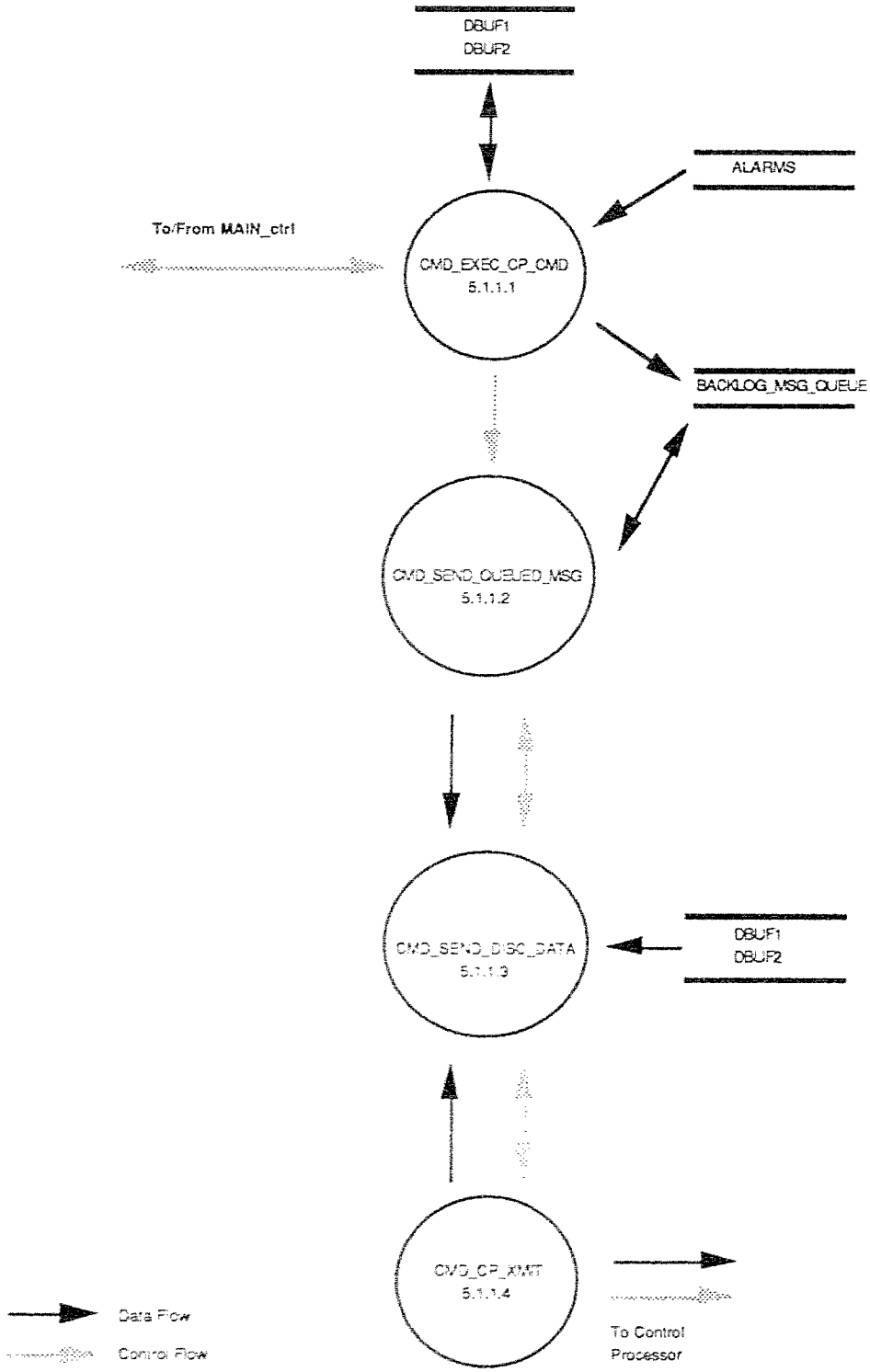


Figure 4-8 Decomposition of MAIN_cmd_ctrl

4.7.3.4 CMD_cp_xmit (Write data/commands to the HI port)

CMD_cp_xmit is responsible for writing data or commands to the HI port using the transfer protocols in the 68000 and DSP-56001 interface protocol. A message from the DSP-56001 to 68000 will be queued to BACKLOG_MSG_Q if the HI port contains unread data when CMD_cp_xmit is ready to write the HI port. All messages inserted into the backlog message queue are rescheduled for transmission at a later time.

4.7.3.5 DISC_PROC_dpll (Phase locked loop for timing recovery)

DISC_PROC_dpll is responsible for processing inbound DISC data and for recovering the received bit sequence. The demodulation algorithm used is a version of a self-synchronizing digital phase locked loop which is described in greater detail in section 5-1. The operations performed by the demodulation algorithm include:

- low pass digital filtering to reduce out of band noise
- high pass filtering to eliminate DC offsets
- early-late gate timing recovery
- data recovery using fixed decision thresholds.

DISC_PROC_dpll processes the received DISC data samples in blocks of 16 samples (i.e. 1 bit time). DISC_PROC_dpll does not execute if a window of 16 samples is not available and will execute more than once if more than 2 bit times of data is available. The additional processing is done to allow DISC_PROC_dpll to catch up with the input data while executing in the background.

Data for DISC_PROC_dpll is read from the input data buffer INPQ and are subsampled by a factor of 2 so that only 8 out of 16 available samples are processed. (This is done to reduce the processing load on the DSP-56001 since DISC_PROC_dpll is a critical routine which must execute every bit time. Simulation studies performed have shown that it is possible to retain the same level of demodulation performance with only 8 instead of 16 samples.). These samples are low passed filtered with a 4th order digital filter and then analyzed for timing error. Errors in timing are adjusted for by advancing or retarding the processing pointer to INPQ. Data decision is performed using a threshold comparison of the sample in the middle of the bit window against fixed thresholds. Each decisioned bit is

passed by DISC_PROC_dpll to the bit alignment routine DISC_PROC_align to search for frame synchronization or data packing.

4.7.3.6 DISC_PROC_align (Align bits into words)

DISC_PROC_align is responsible for detecting the start of valid RNCP messages using the decisioned bits from DISC_PROC_dpll. DISC_proc_align is also responsible for aligning bits onto the correct symbol boundaries once an RNCP message boundary has been located. Each aligned RNCP symbol is packaged into the least significant 7 bits of a byte. The individual RNCP symbols are then packaged 3 per 24 bit DSP-56001 word in preparation for transfer to the 68000.

DISC_PROC_align detects the start of a valid RNCP message by scanning the decisioned bit stream for the frame synchronization (SYNC) codewords of the RNCP message protocol. DISC_PROC_align performs majority voting on demodulated bits before testing for the presence of SYNC. Majority voting must be performed on all bits of a RNCP header even after SYNC has been detected in order to properly extract the message header contents. Majority voting is not required for processing the bits for the text blocks.

DISC_PROC_align extracts the number of text blocks for a message from the majority voted header. This information is used to determine the end of a message. A current message transaction will be aborted if a SYNC condition is detected prior to the end of a message as determined by the text block count field in the header. In such a case DISC_PROC_align is allowed to return to SYNC hunt mode to start the search for the next message boundary.

4.7.3.7 DISC_PROC_busy (Data detection)

DISC_PROC_busy is responsible for performing data detection for the purpose of asserting the channel busy bit in DSMA based multiple access protocols. The algorithm used by DISC_PROC_busy is described in detail in section 5-3. The algorithm works by selectively bandpass filtering a spectral component at 4800 Hz which is generated by performing a non-linear operation on the received data. A comparison of the energy level of the 4800 Hz spectral component is compared against fixed decision thresholds to determine whether valid data is present or not.

DISC_PROC_busy executes using the low pass filtered data processed by DISC_PROC_dpll. DISC_PROC_busy performs a further data reduction and processes only half the data (1/4 the sampling rate) that DISC_PROC_dpll processes in order to keep the processing in DISC_PROC_busy at a minimum.

4.7.3.8 DISC PROC_bch_dec (BCH decoder for message header)

DISC_PROC_bch_dec is responsible for performing error detection on extracted inbound message headers using BCH decoding and for determining the radio channel protocol when the ARDPC is configured to handle both standard and extended message headers

Standard headers of the RNCP protocol family have 12 bit MID fields and 24 BCH parity bits appended to the header while extended header has an 18 bit MID field and 18 BCH parity bits appended. When the ARDPC is configured to handle both standard and extended headers, an extracted header is BCH decoded twice using BCH routines specific for the standard and extended message headers. The BCH routine returning a successful decoding is declared to be the protocol type. A header with a detected error is labelled with a decoding failure and will be transferred by MAIN_ctrl to the 68000 but any text associated with a failed header is discarded.

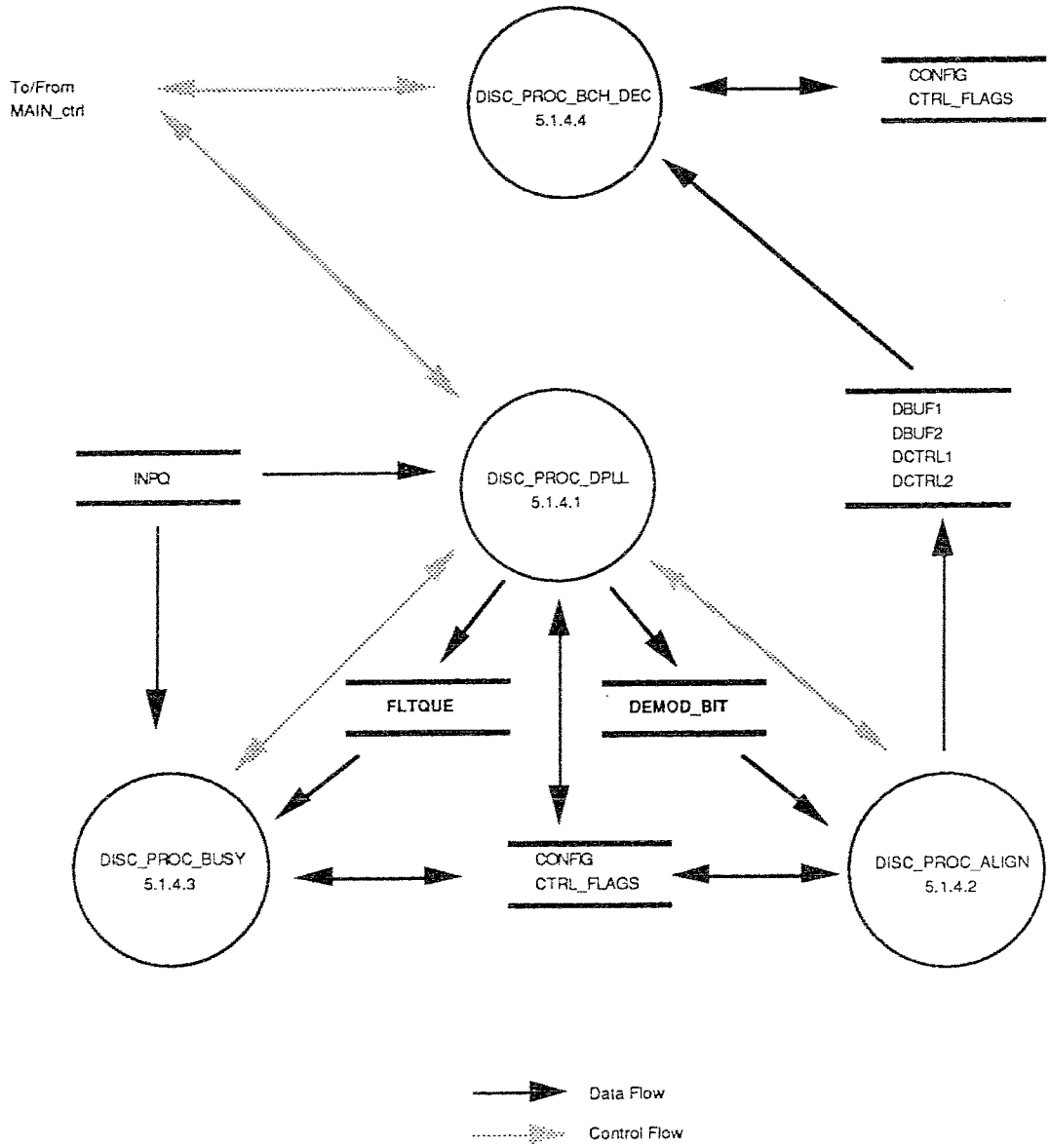


Figure 4-9 Decomposition of MAIN_disc_proc

4.8 DATA DICTIONARY

ALARMS - Buffer for RF alarms and RSSI samples

BACKLOG_MSG_QUEUE
- Stores 68000 response messages which initially could not be transmitted to the 68000 via the HI port

CONFIG - Contains DSP-56001 configuration information

CTRL_FLAGS - Contains current processor state information and busy bit information

DBUF1
DBUF2 - Buffers for demodulated and packetized DISC data

DCTRL1
DCTRL2 - Control block for DBUF1 and DBUF2

DEMOD_BIT - Data store for demodulated bits

FLTQUE - Buffer for filtered DISC data samples

INPQ - Buffer for raw DISC samples

MODBUF1
MODBUF2 - Buffers for packetized RNCP messages received from the 68000 for modulation on the outbound channel

MODCTRL1
MODCTRL2 - Buffer control blocks for MODBUF1 and MODBUF2

MORSECMD - Common section used for transfer of data and control between MAIN_morse_ctrl and SC_morse_server

MORSESMP - Temporary sample queue for SC_morse_server to store 1 bit time of data

MODSQ - Buffer for modulation data samples

MODTXN - Common section used to communicate between CMD_exec_cp_cmd and MAIN_mword_ctrl and MAIN_morse_ctrl

SC_SERVERQ - Storage for 1 bit time worth of modulation data samples

SC_TRANS - Common section used for transfer of data and control between MAIN_mword_ctrl and SC_msg_server

4.9 SOFTWARE TREE

BOOT (sec 4.3)

IO (sec 4.4)

DIAGS

- DIAGS_bus (sec 4.6.1)
- DIAGS_ram (sec 4.6.2)
- DIAGS_checksum (sec 4.6.3)
- DIAGS_filter (sec 4.6.4)
- DIAGS_adc8 (sec 4.6.5)
- DIAGS_adc12 (sec 4.6.6)

SC

- SC_msg_server (sec 4.5.1)
- SC_morse_server (sec 4.5.2)

MAIN

- MAIN_init (sec 4.7.1.2)
- MAIN_HI_rx (sec 4.7.1.3)
- MAIN_ctrl (sec 4.7.1.1)
  - MAIN_cmd_ctrl (sec 4.7.2.1)
    - CMD_exec_cp_cmd (sec 4.7.3.1)
    - CMD_send_queued_msg (sec 4.7.3.2)
    - CMD_send_disc_data (sec 4.7.3.3)
    - CMD_cp_xmit (sec 4.7.3.4)
  - MAIN_mword_ctrl (sec 4.7.2.2)
  - MAIN_morse_ctrl (sec 4.7.2.3)
  - MAIN_disc_proc (sec 4.7.2.4)
    - DISC_PROC_dpll (sec 4.7.3.5)
    - DISC_PROC_align (sec 4.7.3.6)
    - DISC_PROC_busy (sec 4.7.3.7)
    - DISC_PROC_bch_dec (sec 4.7.3.8)
  - MAIN_rfa_handler (sec 4.7.2.5)

5.0 DSP ALGORITHMS

This section discusses in detail the key DSP algorithms implemented in the ARDPC software architecture. The algorithms described include the demodulator, modulator and the data detection algorithms. The description of the demodulator algorithm provides the details for bit timing and data recovery of received RNCP data. The discussion of the modulator algorithm provides details for RNCP waveform generation, while the discussion of the data detection algorithm details the technique used by the DSP-56001 to sense the presence of inbound data over the radio channel.

5.1 DEMODULATOR ALGORITHM

The main purpose of the ARDPC demodulator is to recover received data from a noisy baseband signal supplied by the base station radio discriminator. The algorithm used is an implementation of a self-synchronizing digital phase locked loop which executes every bit time as previously discussed in the software architecture.

The block diagram for the demodulator algorithm is shown in Figure 5-1. The following key operations are identified from the figure:

- Subsampling of the sampled DISC signal
- Digital low pass filtering
- Digital high pass filtering
- Bit Timing Recovery
- Data Recovery

The details of these operations are provided in the remainder of section 5.1.

5.1.1 SUBSAMPLING/DIGITAL LOW PASS FILTERING

The analog DISC signal input to the ARDPC before digital conversion is low pass filtered using a 4 pole analog Butterworth filter with a cut-off frequency of 9.6 kHz. The cut-off frequency was chosen to provide the flexibility to accommodate future upgrades of the ARDPC to faster data protocols. Unfortunately the choice of cut-off frequency also places a requirement on the DSP-56001 software to digitally low pass filter the sampled DISC data to reduce out of band noise in the range of 2.4 kHz to 9.6 kHz.

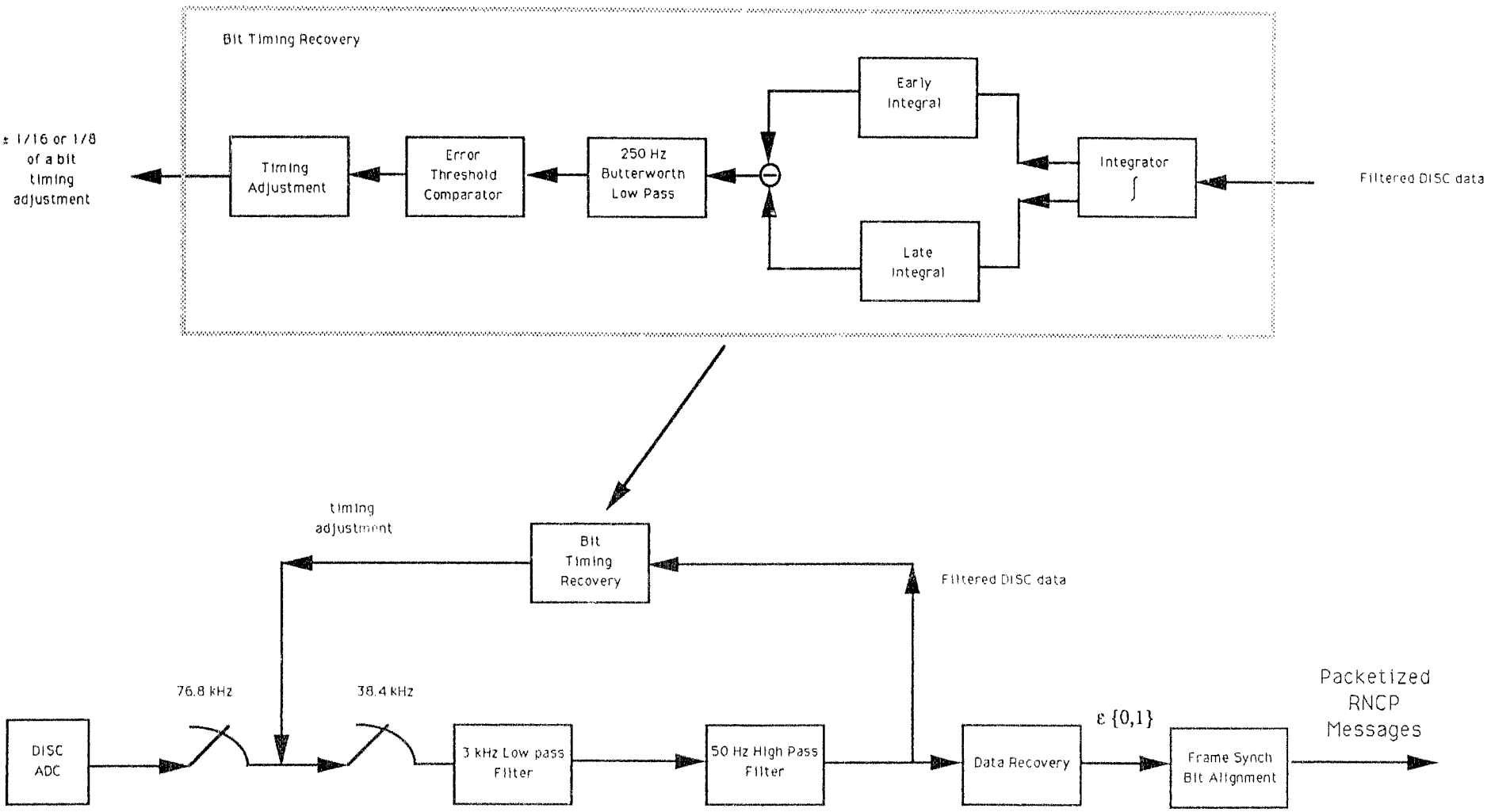


Figure 5-1 ARDPC Demodulator Algorithm Block Diagram

The DSP-56001 samples the filtered analog DISC data at 76.8 kHz, which provides 16 samples per bit for processing. The sampled data is further subsampled (by a factor of 2) by the demodulator algorithm so that only 8 of the 16 available samples per bit are actually processed. This subsampling is performed to halve the potential processing bandwidth consumed by the demodulation algorithm.

The subsampled DISC data is digitally low pass filtered to reduce out of band noise using a fourth order elliptic Infinite Impulse Response (IIR) filter. (The filter was designed using a cut-off frequency of 3 kHz and a sampling frequency of 38.4 kHz). An elliptic filter was chosen because this class of IIR filter provided the steepest roll-off and greatest stop-band attenuation for a given filter order. However, the superior magnitude response performance of the elliptic filter comes at the expense of greater non-linearity in the phase response.

The transfer function for the low pass elliptic IIR filter is shown in equation 5.1.

$$H(z) = \frac{0.123 + 0.171z^{-1} + 0.123z^{-2}}{1 - 1.098z^{-1} + 0.4624z^{-2}} \cdot \frac{0.117 + 0.119z^{-1} + 0.117z^{-2}}{1 - 0.928z^{-1} + 0.2049z^{-2}} \qquad (5.1)$$

Equation 5-1 was implemented using a pair of cascaded biquadratic sections with 24 bit arithmetic. These biquadratic sections are shown in Figure 5-2a. The equations used to implement the filter are shown in equations 5-2 through 5-5. The actual DSP-56001 code segment used to implement the filter is shown in Table 5-1. Figure 5-2b diagrams the register map of the DSP-56001 used in implementing the filter. The code in Table 5-1 can easily be converted to be general for an Nth order IIR filter.

$$w1(n) = x(n) - a_{11} \cdot w1(n-1) - a_{12} \cdot w1(n-2) \qquad (5\text{-}2)$$

$$y'(n) = b_{10} \cdot w1(n) + b_{11} \cdot w1(n-1) + b_{12} \cdot w1(n-2) \qquad (5\text{-}3)$$

$$w2(n) = y'(n) - a_{21} \cdot w2(n-1) - a_{22} \cdot w2(n-2) \qquad (5\text{-}4)$$

$$y(n) = b_{20} \cdot w2(n) + b_{21} \cdot w2(n-1) + b_{22} \cdot w2(n-2) \qquad (5\text{-}5)$$

The frequency and phase response of the implemented filter is shown in Figures 5.3a and 5.3b respectively. The group delay characteristic of the filter is shown in Figure 5.3c. It

can be seen from Figure 5.3b that with the design parameters chosen, the filter approximates a linear phase response in the critical region 0 to 2.4 kHz . (A linear phase response in this region is desirable to maintain constant group delay for the data.)

Table 5-1 DSP-56001 Assembler Code for a 4th Order Digital IIR Filter

```
move y:disc_gain,x1                                    ; Get the disc gain
do #TIME_SAMPLES,end_filter
  move y:(r7)+n,y0                                     ; Get an input sample
  mpy  y0,x1,a        x:(r3)+,x0     y:(r5)+,y0        ; scale input sample
                                                       ; x0 = a12/2, y0=w1(n-2)
  do #NSEC,end_cell
     mac -x0,y0,a     x:(r3)+,x0     y:(r5)-,y1        ; a = x(n) - ai2*wi(n-2)
                                                       ; x0=ai1/2, y1=wi(n-1)
     macr -x0,y1,a    x:(r3)+,x0     y1,y:(r5)+        ; a = x(n) - ai2*wi(n-2) - ai1*wi(n-1)
                                                       ; x0=bi2/2, push wi(n-1) to wi(n-2)
     mpy x0,y0,b      x:(r3)+,x0     a,y0              ; b=bi2*wi(n-2)
                                                       ; x0=bi1/2, y0=wi(n)
     mac x0,y1,b      x:(r3)+,x0     a,y:(r5)+         ; b = bi1*wi(n-1) + bi2*wi(n-2)

                                                       ; x0=b10/2, push w1(n) to w1(n-1)
     macr x0,y0,b     x:(r3)+,x0     y:(r5)+,y0        ; b=bi0*wi(n)+bi1*wi(n-1)+
                                                       ;    bi2*wi(n-2)
                                                       ; x0 = a22, y0 =w2(n-2)
     move b1,a                                         ; a=y(n)/2
  end_cell
  move a,x:(r4)+                                       ; save 2 * y(n)
  move x:(r5)-,a      y:(r3)-,b                        ; reset r3 and r5 for the next cycle
end_filter
```

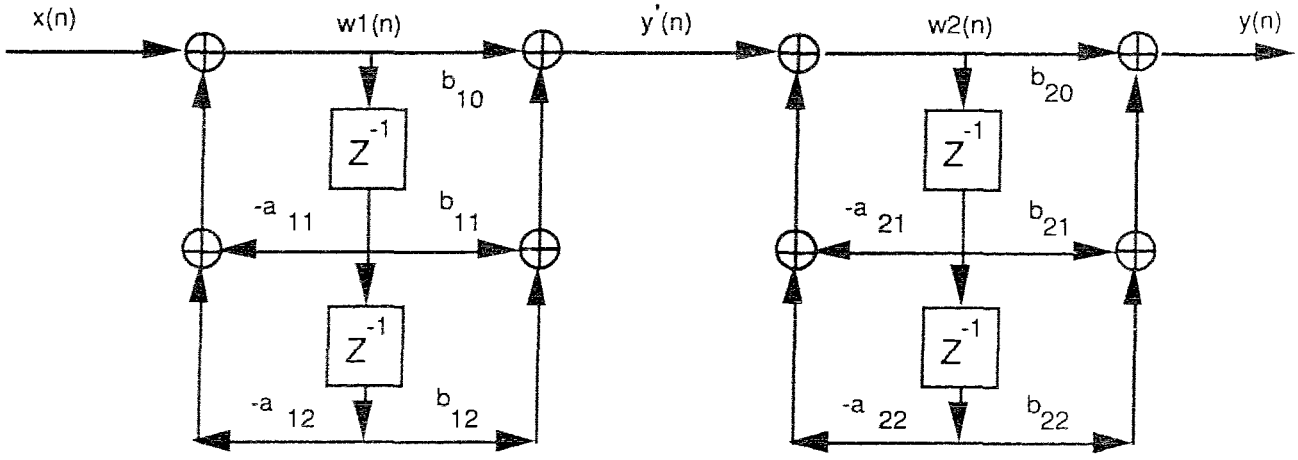


Figure 5-2a Demodulator Low Pass IIR Filter Biquadratic Sections

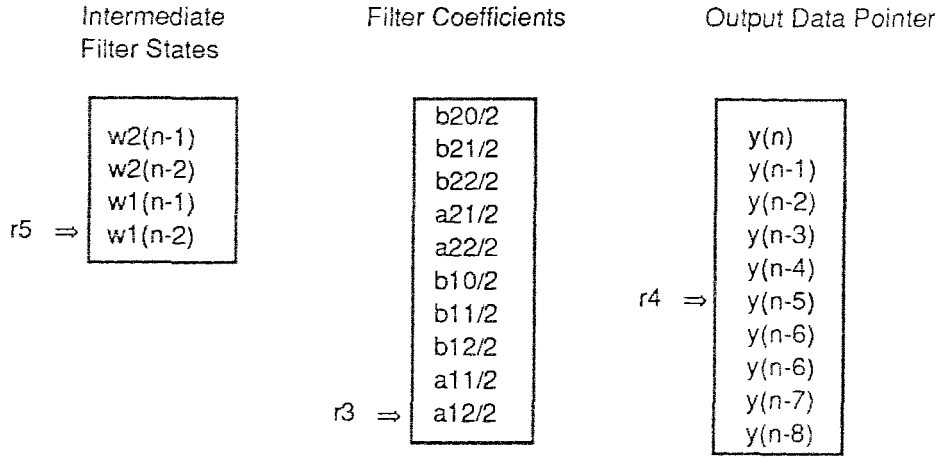


Figure 5-2b DSP-56001 Register Pointers for the Low Pass IIR FIlter

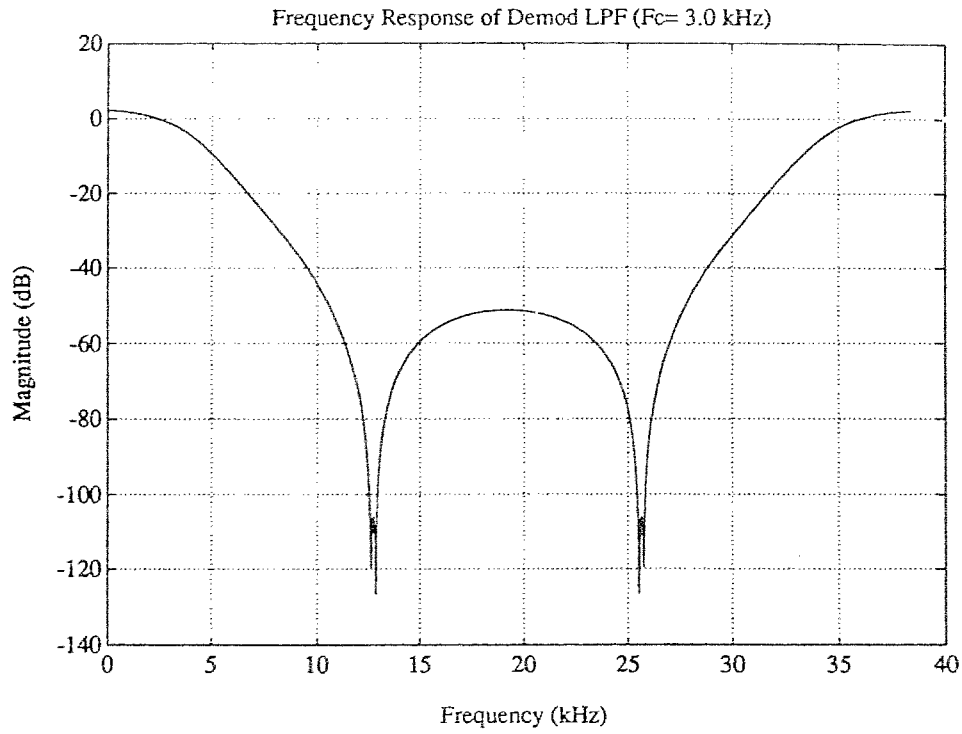


Figure 5-3a Frequency Response of Demodulator Low Pass IIR Filter

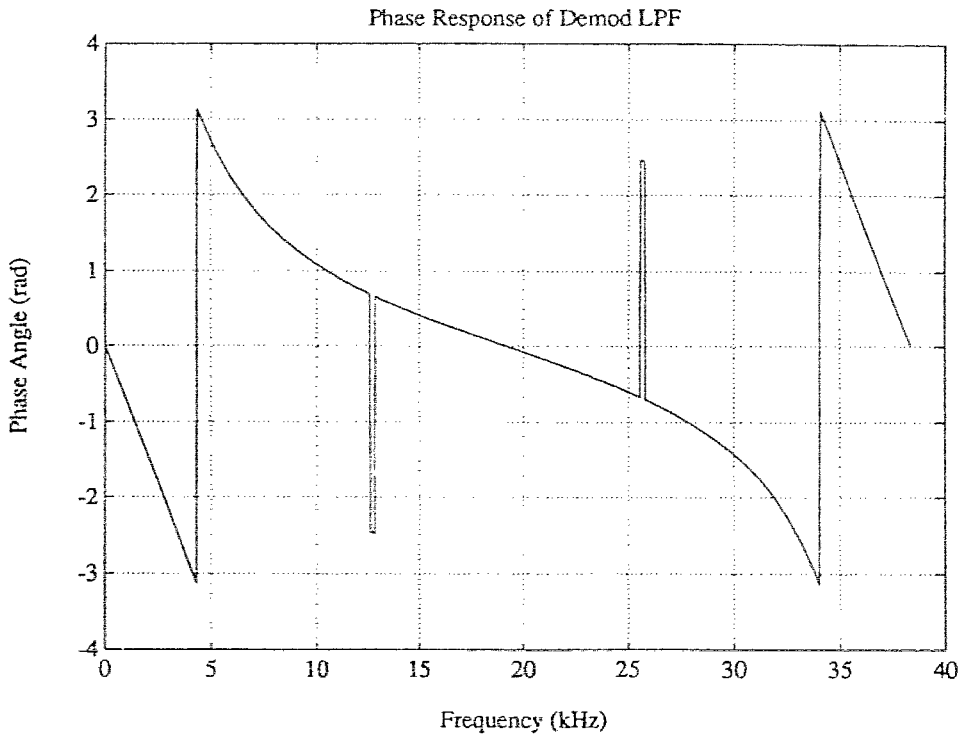


Figure 5-3b Phase Response of Demodulator Low Pass IIR Filter

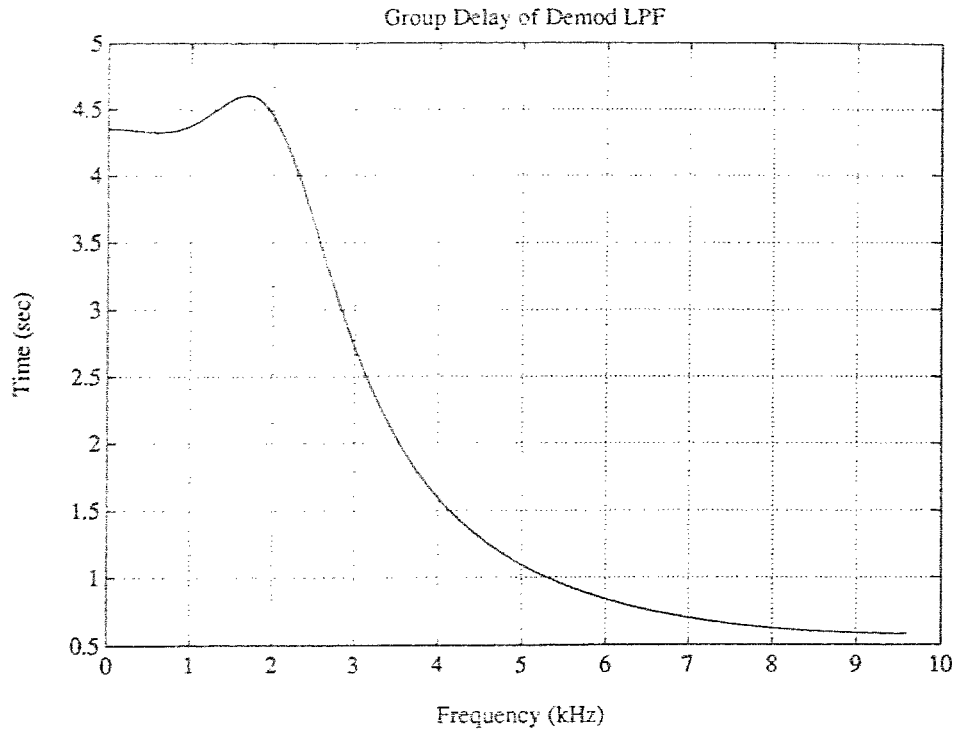


Figure 5-3c Group Delay Characteristics of Demodulator Low Pass Filter

5.1.2 HIGH PASS FILTERING

The output from the discriminator of a base station radio is sometimes characterized by an exponential decay from a DC driven transient step. This condition occurs during the start-up of a carrier signal from a mobile unit. The amount of DC offset is determined by the carrier frequency offset between the mobile and base station radios. The above situation is illustrated in Figure 5-4, which shows a pair of oscilloscope traces capturing the startup transient of the base station radio when preamble data is being received inbound. The bottom trace represents the DISC line leading into the ARDPC, while the top trace represents the state of the channel busy bit.

The ARDPC DISC input is AC coupled and its response to a DC step function is that of a decaying exponential with a time constant up to 1 second. This relatively long decay time affects the demodulation data decision process, since the decision thresholds for data recovery are based on fixed DC values. In order to reduce the effect of the startup transient and allow the modem to lock onto symbol centres sooner, the low pass filtered data is passed through a cascaded high pass filter with a 50 Hz corner frequency. This filter is implemented as a first order filter and its transfer function and frequency response is shown in equation 5.6 and Figure 5-5 respectively.

$$H(z) = \frac{0.9959 - 0.9959z^{-1}}{1 - 0.9918z^{-1}} \qquad (5.6)$$

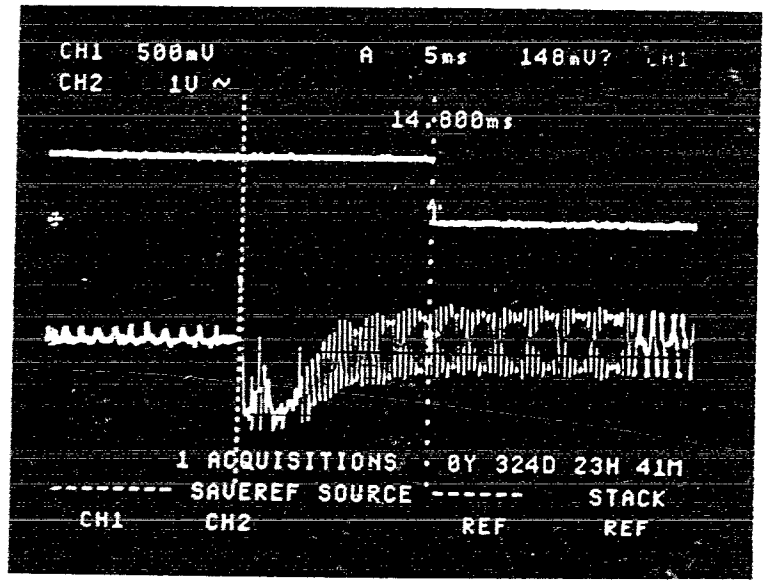


Figure 5-4 Transient Startup Condition of RNCP Data

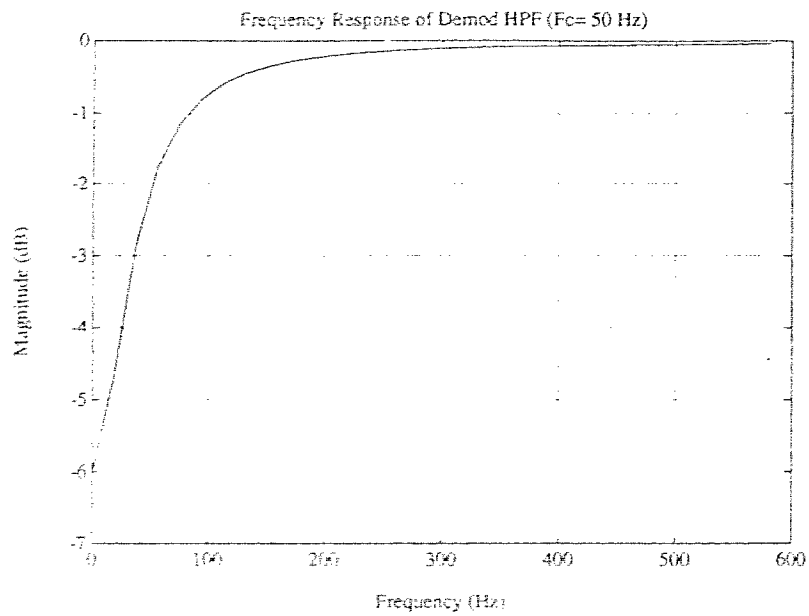


Figure 5-5 Frequency Response of Demodulator High Pass Filter

5.1.3 TIMING RECOVERY

The block diagram for the timing recovery algorithm can be seen in boxed area of Figure 5-1. The algorithm is based on an early-late gate algorithm which exploits the symmetry in the pulses of the baseband RNCP signal to lock onto symbol centres and adjust to timing jitter. The input to the timing recovery algorithm is the filtered data samples generated after low and high pass filtering. A window of 8 samples is available for use by the timing recovery algorithm every bit time.

The timing recovery algorithm is based on geometric arguments and a description of the basic RNCP pulse replica is useful in understanding the rationale behind it. A sequence of RNCP pulses are shown in Figure 2-2. The waveforms are pseudo-ternary with 3 valid levels at the symbol center namely -1,0 and +1. From Figure 2-2 it can be seen that the basic pulse replica is sinusoidal in nature and possesses high symmetry. The demodulator timing recovery algorithm, when locked, establishes its timing centre at the centre of a pulse and calculates an integral to the left (late gate) and to the right (early gate) of the timing centre based on the samples taken. This is shown in Figure 5-6 where the data from points C to D form the early gate and points D to E form the late gate. The difference between the early and late gate integrals is 0 when the pulse is symmetric and the timing centre is correct. A timing error or non-zero difference between early and late gate integrals results when the timing centre is incorrect. This timing error is used to determine the direction and amount of timing adjustment to perform in an attempt to lock onto the received symbol centres.

The timing error is filtered with a low pass filter (250 Hz Butterworth) to derive a smoothed signal which reflects both the magnitude and sign of the timing error. A phase kick in a direction to correct the timing error is applied when the filtered error signal exceeds established thresholds. A phase kick is implemented by adjusting the data pointer to the raw DISC data queue forwards or backwards by the appropriate number of data samples.

The above algorithm was implemented using two levels of phase corrections. A phase correction of ± 1 sample or 1/16 of a bit is applied when the timing error has exceeded the deadband threshold. A faster phase correction of ± 2 samples or 1/8 of a bit is provided when the timing error is excessively large. The phase correction made is applicable in the next bit time for the selection of the next window of 16 samples to filter. The intermediate

states of the timing error filter are zeroed whenever a phase correction is applied to prevent over correction due to filter delay. The action of the timing recovery filter is akin to that of an analog integrate and dump filter.

A drawback of the timing recovery algorithm is that it suffers from a timing ambiguity condition as shown in Figure 5-7. This figure depicts a situation where a zero early/late gate integral difference is possible with an incorrect timing centre. If the timing centre is established as shown, the algorithm will be fooled into believing that it has locked on to the correct timing centre. This is an undesirable condition because it will result in large number of bit errors. Fortunately the situation shown in Figure 5-7 is unstable because a phase kick in any direction will eventually migrate the timing centre back to its correct position. The simplest way to compensate for this timing ambiguity condition is to check the sign and magnitude of the early/late gate integrals when the timing error is small. If the integrals are large and of the same sign, then the correct timing centre has been found. If the integrals are large and of opposite sign then the timing ambiguity condition is found and a corrective phase kick must be applied to correct this situation.

5.1.4 DATA RECOVERY

The data is recovered by performing a threshold comparison on a bit window of data which has been filtered and adjusted for timing error. The centre sample of an 8 sample wide bit window is used for the decision sample to determine whether a +1,0 or -1 was received. The centre sample is compared against fixed thresholds to arrive at a decision. The thresholds for +1 and -1 are normally set to 50% of the peak positive and negative digital values since the occurrence of a 0 or 1 is assumed to be equally likely.

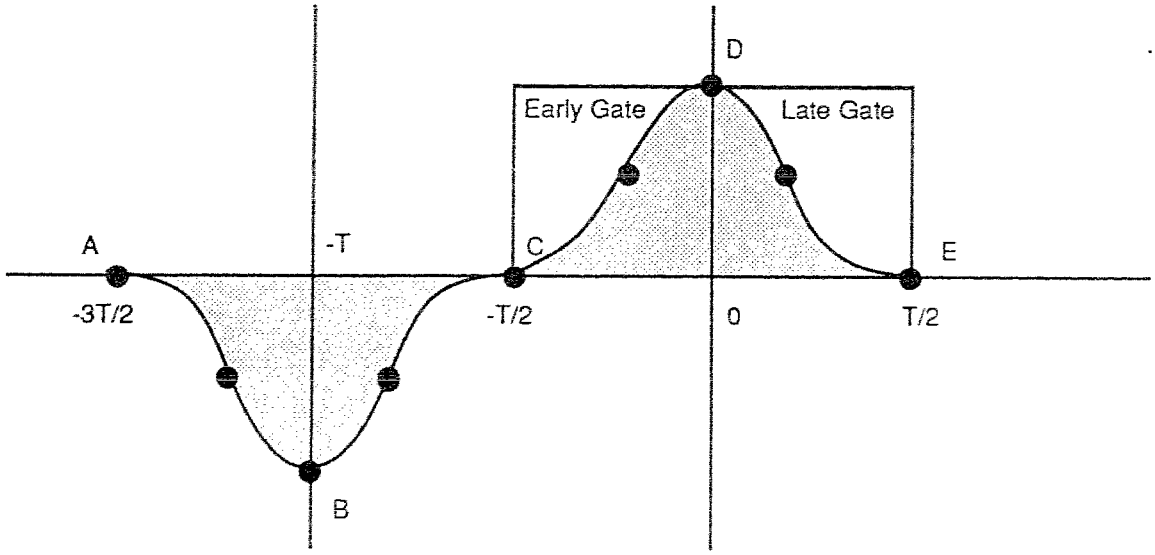


Figure 5-6 Early/Late Gate Timing Recovery

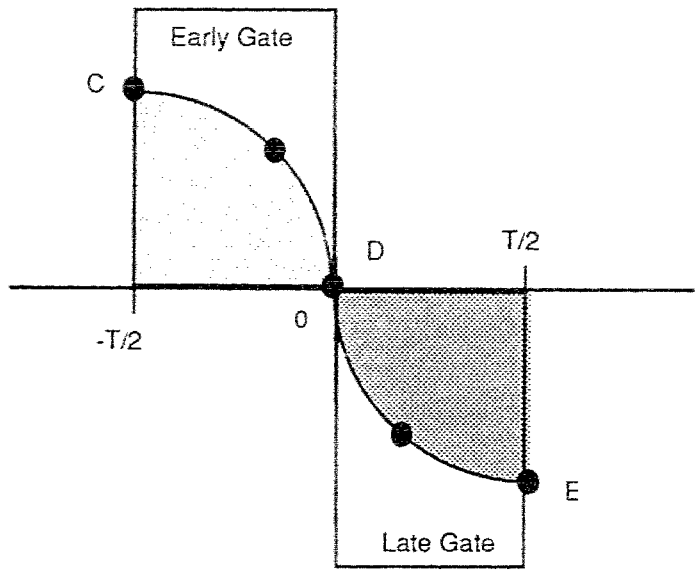


Figure 5-7 Early/Late Gate Timing Ambiguity

5.2 MODULATOR ALGORITHM

As discussed in section 2.2.1, RNCP modulation consists of differential encoded bipolar signalling of low pass filtered rectangular pulses. In bipolar signalling, a zero is represented by the zero dc level while 1's are represented as alternating positive and negative pulses. The effect of alternating positive and negative pulses to represent 1's is to remove the dc component in the transmission of binary data while conserving bandwidth.

The basic RNCP waveform satisfies the first Nyquist criterion for inter-symbol interference. The side lobes of the pulse decay at a rate of $1/t^3$ compared to 1/t for a sinc pulse. The low pass filter used to generate the pulse is a fourth order digital elliptic filter whose coefficients are given in equation 5.7 and are based on the switched capacitor filters used in the first generation design.

$$H(z) = \frac{6.86 - 0.0588z^{-1} + 6.57z^{-2} - 0.0588z^{-3} + 6.86z^{-4}}{1 - 2.623z^{-1} + 2.584z^{-2} - 1.113z^{-3} + 0.171z^{-4}} \times 10^3 \qquad (5.7)$$

The above filter coefficients are based on a sampling frequency of 76.8 kHz to yield an over-sampling ratio of 16:1 for RNCP data. The frequency and impulse response of the filter is shown in Figures 5-8 and 5-9 respectively.

The RNCP waveform may be implemented directly by using a rectangular pulse directly input to the filter given in equation 5.7. However, this technique becomes computationally expensive when 16 samples per bit need to be generated. This is the case even when the filter is implemented using biquadratic sections. To illustrate, a filter biquadratic section on average requires 10 DSP-56001 instructions to execute per section per sample. A fourth order filter would require up to 320 instructions or consume approximately 12% of the available processing bandwidth based on a bit time (2600 instructions available per bit with an 80 ns cycle time) for modulation. This bandwidth consumption figure was deemed to be excessive since data management overhead was not even included in the calculation.

An alternative scheme using a tapped delay line (FIR filter) to generate the pulse directly was implemented in the ARDPC in an attempt to reduce the computational load on the DSP-56001 processor during modulation. This technique initially required filtering a rectangular pulse (16 samples wide) and collecting the response. The coefficients for the

pulse shaping FIR filter were then obtained by truncating and scaling the pulse response over a period of 3T or 48 samples. Windowing of the FIR coefficients was not deemed necessary since the filtered response was essentially finite over a period of 3T. The final FIR coefficients are shown in Table 5-2.

During modulation, the input to the FIR filter consists of a 4800 Hz impulse train whose non-zero values are ± 1. Computationally speaking, a single sample for a RNCP pulse is then obtained by adding or subtracting FIR filter coefficients separated by 16 samples. A buffer consisting of the current bit and the previous 2 bits is necessary for the computation of the current sample point since the filter is 48 samples wide. In general, the FIR algorithm for modulation required only 3 instructions per output sample (i.e. 3 multiplications) compared to 40+ instructions for the direct filtered approach, thus realizing a substantial saving in processing bandwidth.

Table 5-2 Modulation FIR Filter Coefficients

h(1)	0.00631	h(26)	0.26257
h(2)	0.02283	h(27)	0.18089
h(3)	0.05589	h(28)	0.11524
h(4)	0.10691	h(29)	0.06557
h(5)	0.17893	h(30)	0.03049
h(6)	0.26997	h(31)	0.00776
h(7)	0.37379	h(32)	-0.00522
h(8)	0.48227	h(33)	-0.01108
h(9)	0.58753	h(34)	-0.01215
h(10)	0.68320	h(35)	-0.01039
h(11)	0.76513	h(36)	-0.007283
h(12)	0.83131	h(37)	-0.003884
h(13)	0.88164	h(38)	-0.000843
h(14)	0.91741	h(39)	0.00150
h(15)	0.94077	h(40)	0.00304
h(16)	0.95428	h(41)	0.003831
h(17)	0.95422	h(42)	0.004002
h(18)	0.93905	h(43)	0.003727
h(19)	0.90440	h(44)	0.003176
h(20)	0.85036	h(45)	0.002497
h(21)	0.77496	h(46)	0.001804
h(22)	0.68085	h(47)	0.001175
h(23)	0.57464	h(48)	0.000655
h(24)	0.46456		
h(25)	0.35847		

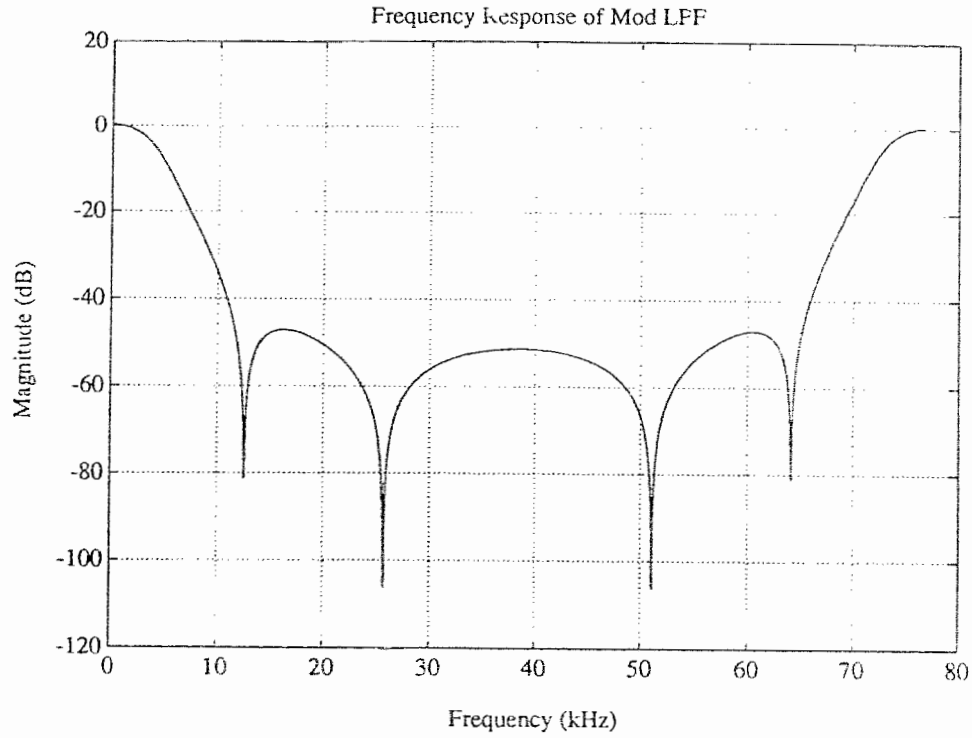


Figure 5-8 Frequency Response of Modulator Low Pass Filter

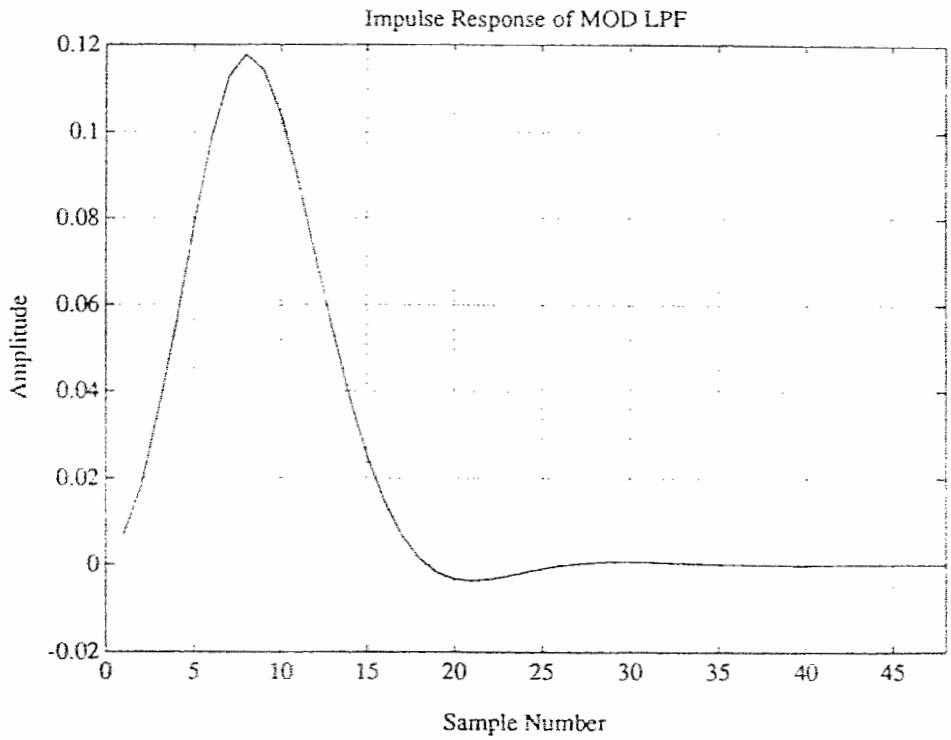


Figure 5-9 Impulse Response of Modulator Low Pass Filter

5.3 DATA DETECTION ALGORITHM

The data detection algorithm is used by the ARDPC to sense the presence of valid RNCP messages over the radio channel in a DSMA configuration. The ARDPC sets the busy bit (of the RNCP protocol) in subsequent outbound message symbols if a valid inbound message is detected. Mobile units other than the current sender with messages to send will sense that the busy bit is set and will follow a random back off algorithm before attempting to transmit. The ARDPC will clear the busy bit to indicate that the radio channel is free when the current sender has completed its inbound message.

There are some characteristics desirable in a data detection algorithm. A data detection algorithm should possess a fast attack time for detecting the transition from random noise to data and vice-versa on the radio channel. The former condition is known as the "busy attack time" and the latter is known as the "busy hang time". The throughput for the radio channel will be adversely affected if the busy hang time is excessively long since the available radio channel idle time will be reduced (i.e. mobile units waiting with messages to transmit will be blocked from doing so until the base station's data detection algorithm can make the transition from data to noise on the radio channel). The data detection algorithm must also show stability during the period that a message is being transmitted on the channel. If the algorithm is unstable then false windows of idle channel time will be opened, which will allow a waiting mobile to begin message transmission before the current sender has been completed, thus resulting in a collision and subsequent loss of both messages.

A data detection algorithm must also be insensitive to the amplitude of noise output by the discriminators of different base radios. Ideally, the discriminator noise characteristics of base radios should be the same, but this is usually not the case. On some base radios, the discriminator amplitude for noise reaching the ARDPC will be more than double the amplitude for signal, while on other radios the noise amplitude will be approximately equal the data amplitude. A data detection algorithm will not be portable to different FM base radios if it is not insensitive to discriminator noise level.

The data detection algorithm designed for the ARDPC satisfies all the above requirements and is simple in concept. The block diagram for the algorithm is shown in Figure 5-10. The steps involved in the algorithm are summarized as follows:

- the baseband RNCP signal is converted into a rectangular version of itself using hysteresis thresholding.

- a non-linear operation in the form of a T/2 delay and multiply operation is applied to the rectangular signal with the result that a strong spectral peak at the clock frequency is generated at the output of the delay and multiply block.

- the delay and multiply output is bandpassed filtered using a narrow digital bandpass filter to isolate the clock frequency spectral component.

- the absolute value of the bandpass filter output is taken, smoothed with a 250 Hz Butterworth low pass and then thresholded to determine the presence/absence of data. Hysteresis is applied to the decision process for debouncing purposes.

5.3.1 SPECTRAL COMPONENT GENERATION

The theoretical spectrum of a bipolar signal does not contain a discrete component at any frequency. However, a strong spectral component at the transmitter clock frequency can be generated with the application of a non-linear operation to the baseband signal. (The theory behind this result can be found be in references [2] and [11]). Such relevant non-linear operations include: absolute value; squaring or delay and multiply. The effects of these operations on simulated RNCP data were studied. The data sets used included data and preamble segments and were generated to be representative of the output of the demodulation process, which implies that the data has been filtered and subsampled so that there are 8 samples to a bit. The signal to noise ratio of the data sets varied from that corresponding to a BER of 10^{-2} to essentially noise-free data.

Typical results of non-linear operations on noise-free simulated data are shown in Figures 5-11 through 5-14. These figures show that a component of the clock frequency which is about 20 dB above the noise floor can be generated using any of the non-linear operations described in the previous paragraph. Figure 5-14 shows the spectrum of the data which has been converted to rectangular pulses using hysteresis thresholding and then processed

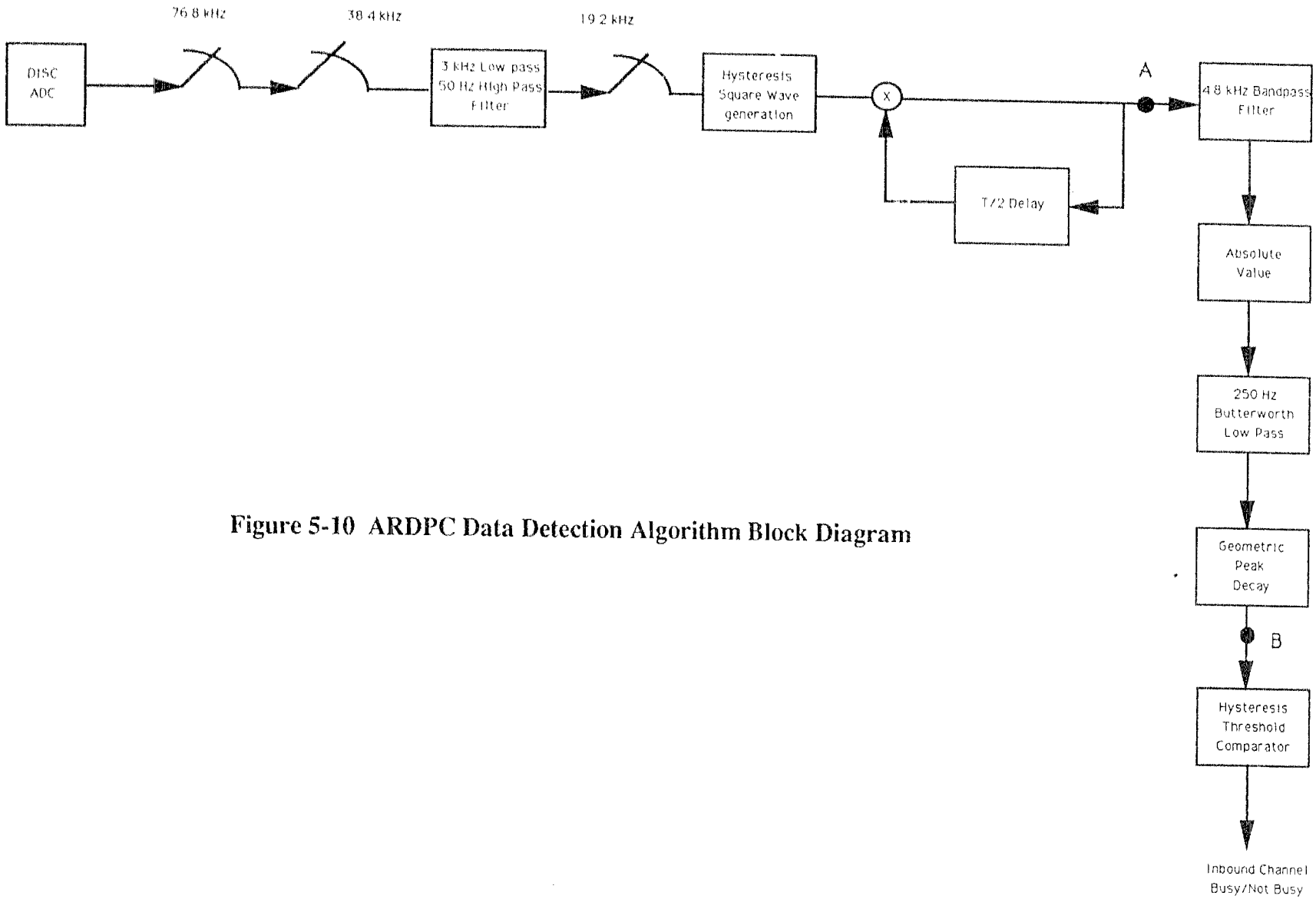


Figure 5-10 ARDPC Data Detection Algorithm Block Diagram

using a T/2 delay and multiply operation. (Delays other than T/2 were also used but resulted in a reduced clock frequency spectral component).

In the ARDPC, a window of 8 samples of digitally filtered DISC data is available every bit time. This data is generated by the demodulation algorithm for bit synchronization and data recovery. It was convenient to have the data detection algorithm execute at every bit time immediately after the demodulation algorithm, since both algorithms can share the same set of input data samples (i.e. raw DISC samples which have been low passed filtered to reduce noise and high passed filter to reduce effects of DC offset). However, this approach imposed an additional load on DISC data processing which, combined with demodulation, could consume more than 60% of the processor bandwidth at certain stages. This prompted an investigation in reducing the amount of data processed for data detection.

Another series of simulations were performed using data which was further subsampled by an additional factor of 2 to determine the effects of further data reduction. The data used to generate Figures 5-11 through 5-14 was subjected to further subsampling and the same non-linear operations corresponding to Figures 5-11 through 5-14 were reapplied and the results are shown in Figures 5-15 through 5-18. Overall, the simulations show that there is very little degradation in the final results by processing 1/4 rate data over 1/2 rate data. An interesting result is shown in Figure 5-18 where the data was converted to a rectangular pulse train before being operated on using a T/2 delay and multiply operation. Figure 5-18 shows that all the multiples of the the 2400 Hz spectral component are eliminated with the exception of the 4800 Hz component. This was a desireable property because it provided greater flexibility in the design of the bandpass filter, which must be used to extract the clock frequency component. Therefore this technique was adopted over the others to generate the clock frequency component for the data detection algorithm

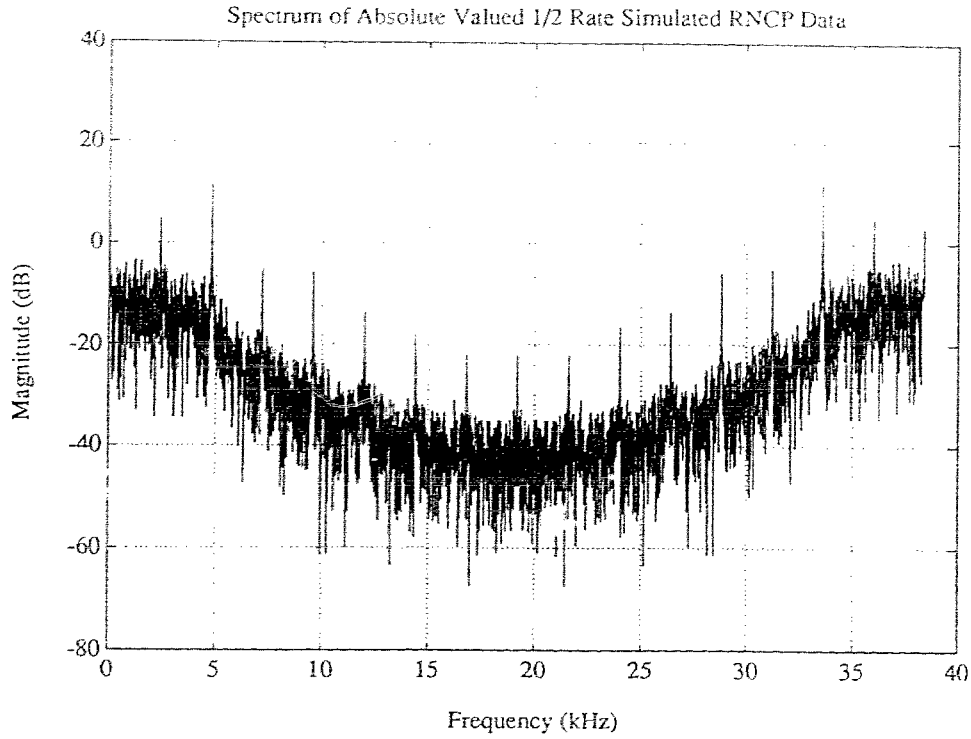


Figure 5-11

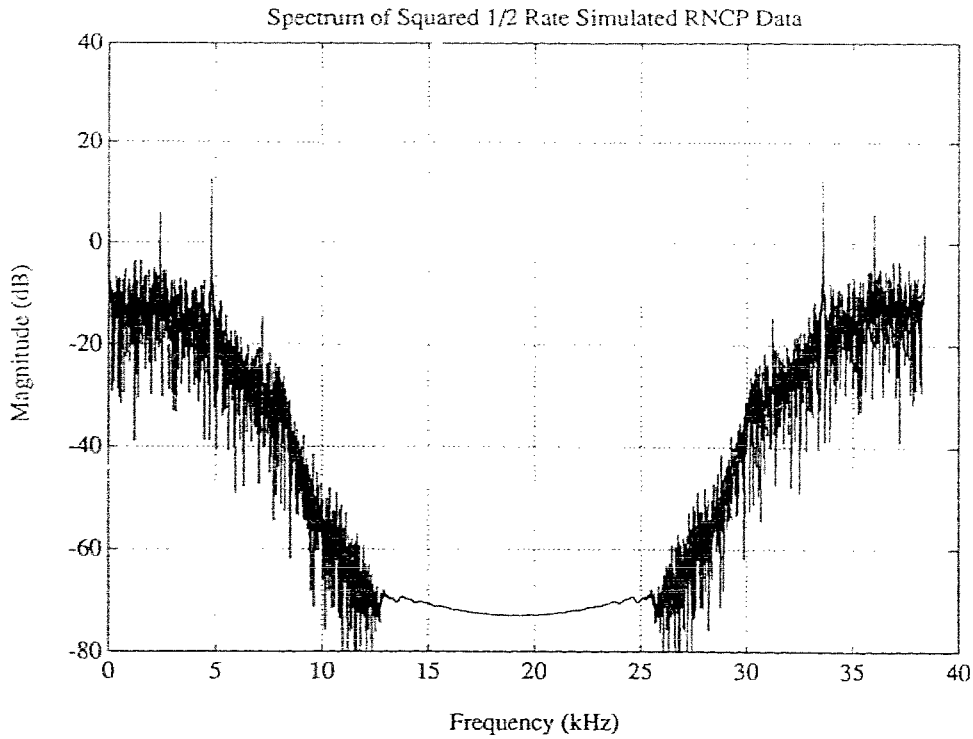


Figure 5-12

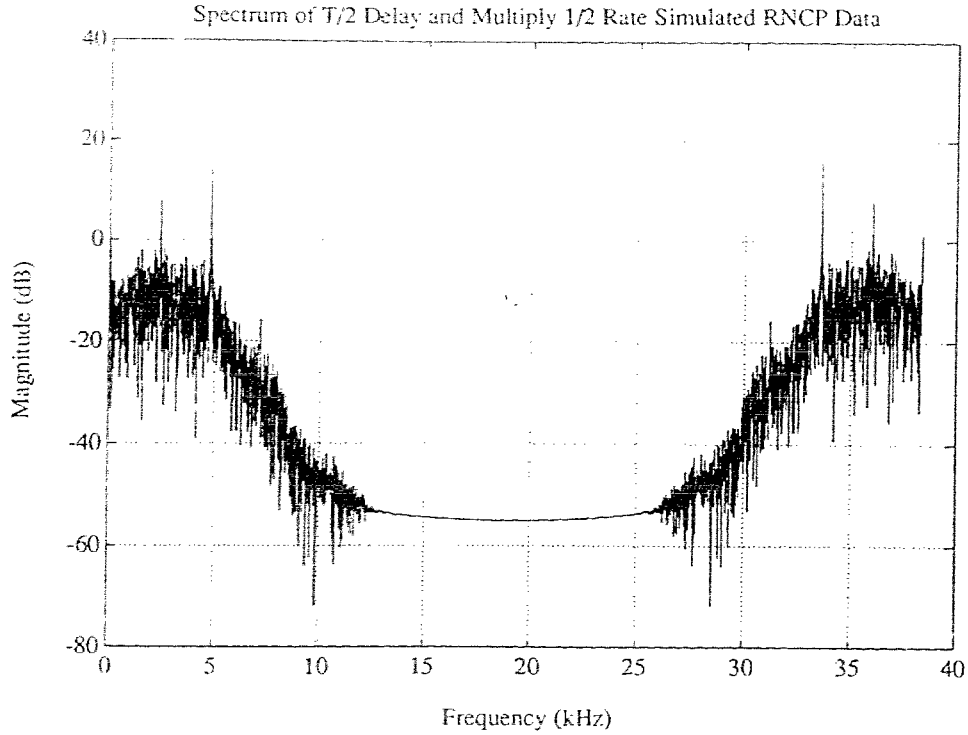


Figure 5-13

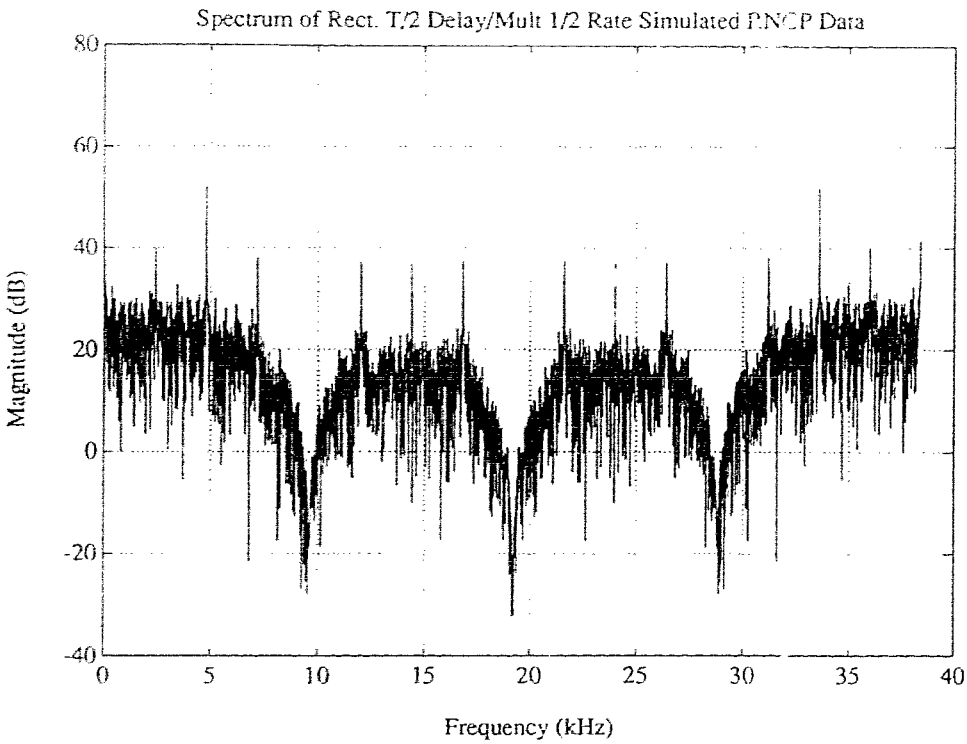


Figure 5-14

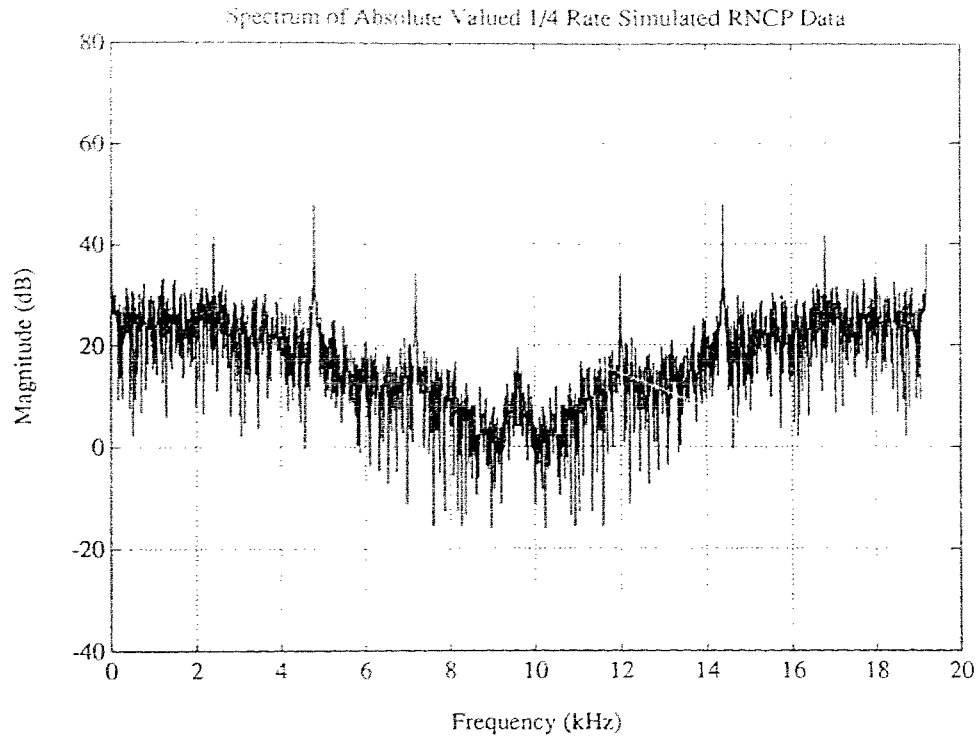


Figure 5-15

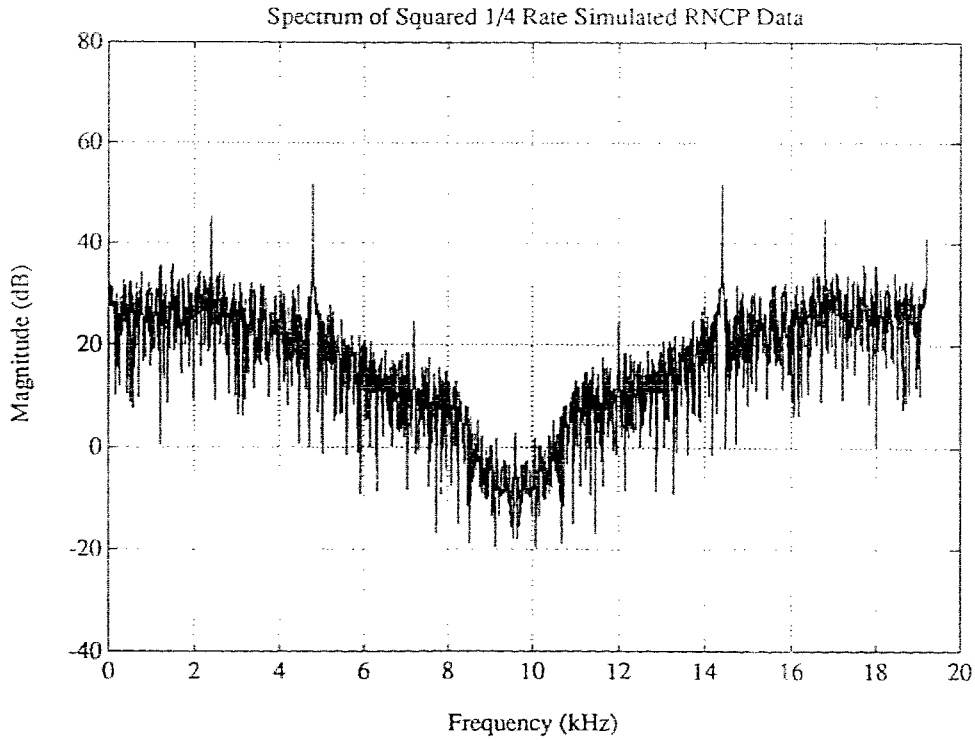


Figure 5-16

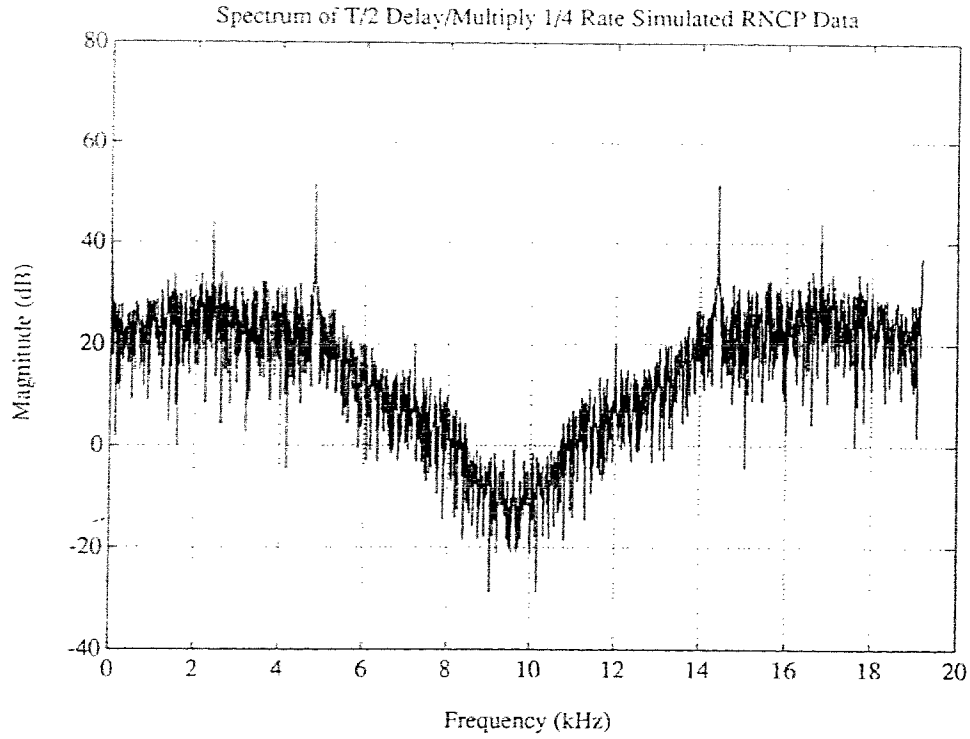


Figure 5-17

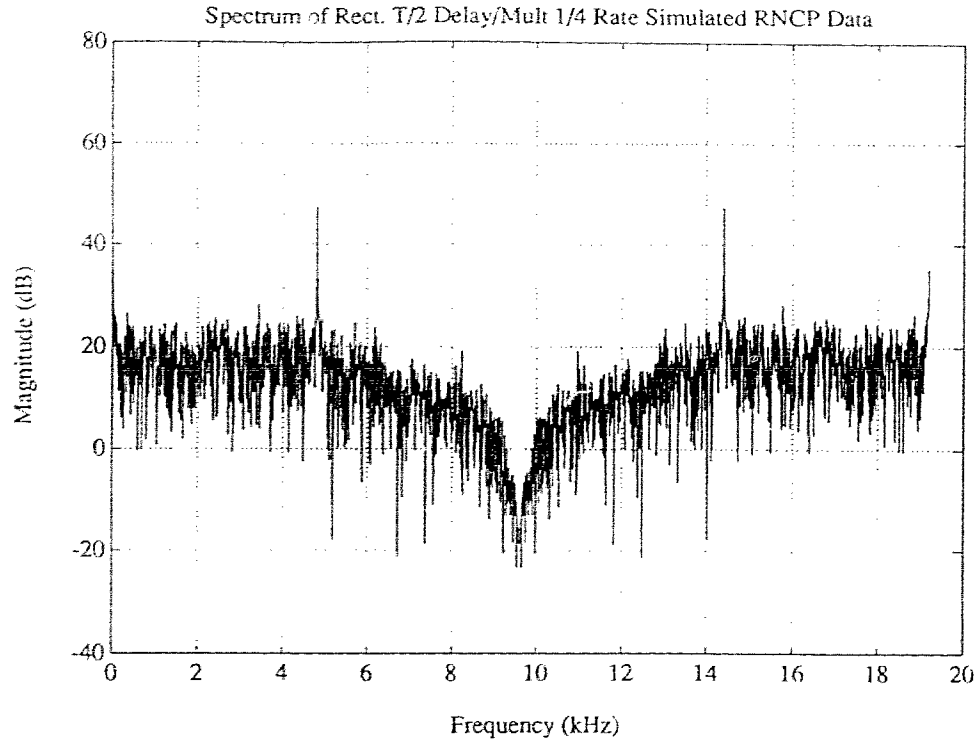


Figure 5-18

5.3.2 BANDPASS FILTERING

The spectrum of the output of the delay and multiply operation implemented contains a strong discrete component at the data clock frequency (4.8 kHz). The magnitude of this component is data dependent and is greatest when the 2.4 kHz RNCP message preamble is received and is weakest when a long string of 0's are being received. The clock frequency spectral component can be isolated using a narrow digital bandpass-filter centred at the clock frequency. The bandpass filter implemented was a second order all-pole filter whose poles were located on the radial line joining the clock frequency on the unit circle with the origin. Higher order filters were attempted and produced better results, but proved to be computationally expensive.

The Z transform for the bandpass filter used is given by :

$$H(z) = \frac{z^2}{(z - re^{-j\theta})(z - re^{j\theta})} \qquad (5.10)$$

Equation 5.10 reduces to the following form for 1/4 rate data:

$$H(z) = \frac{b}{1 + az^{-2}} \quad ; a = r^2 \; ; b = 1 - a^2 \qquad (5.11)$$

The parameter a determines the width of the filter passband. As with analog filters the narrower the filter passband (i.e. higher the Q value) the greater the delay of energy to pass through the filter. On the other hand increasing the filter passband width allows more energy from adjacent frequencies to leak through resulting in a less smooth result.

A series of simulations were performed with different values of a. These are shown in Figure 5-19. As a increases (i.e. approaches the unit circle) it can be seen that the bandpass results increase in smoothness due to greater selectivity in passing the clock frequency component but with the drawback of greater delay in the clock frequency component achieving its maximum value. From the simulations it was decided that the best combination of attack/hang time response and stability was for r = 0.994.

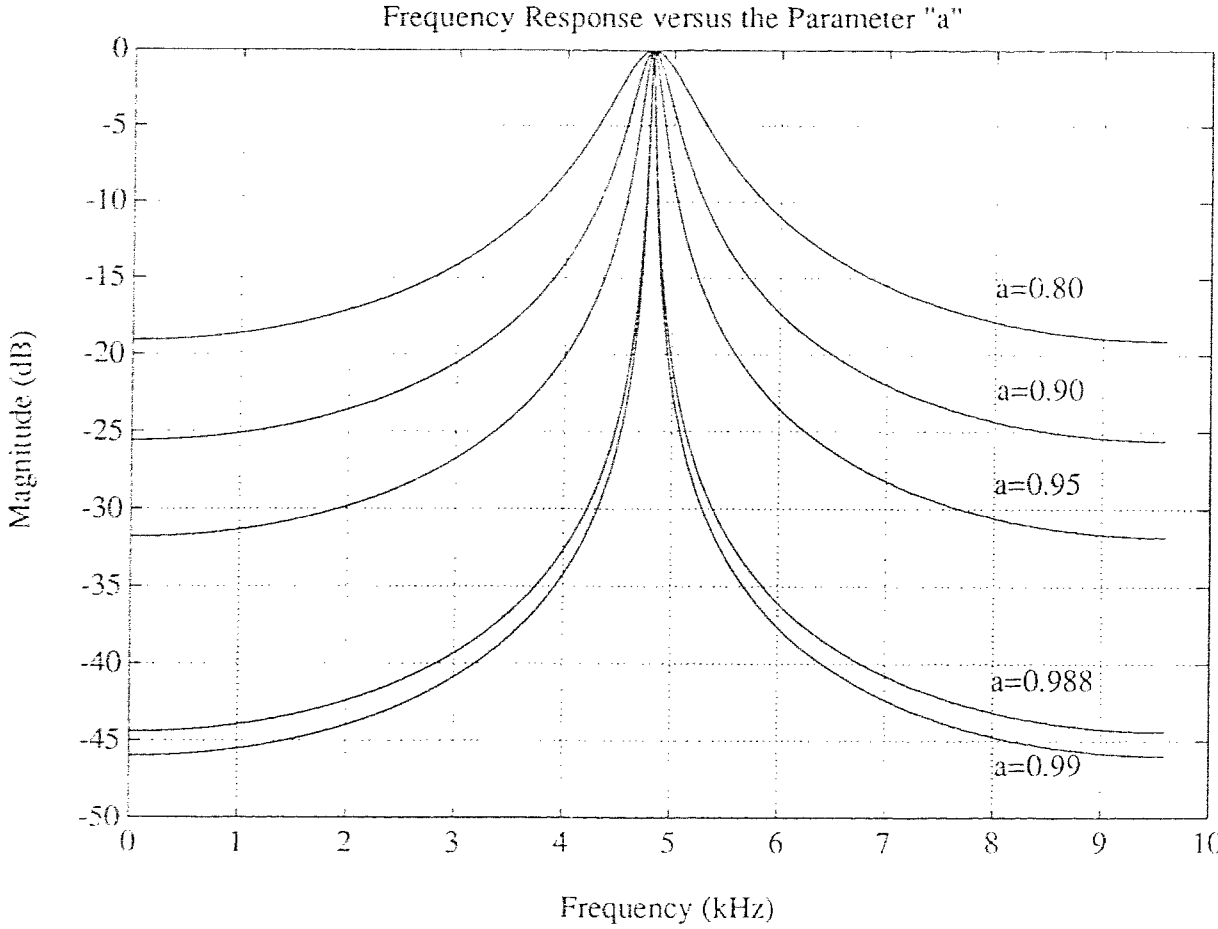


Figure 5-19 Variation of Bandpass Filter Bandwidth with Parameter 'a'

5.3.3 DATA CONDITIONING

Three operations were used to prepare the data for decision on whether data was present on the radio channel. These operations include the following:

- rectification
- smoothing
- geometric peak decay

The time domain output of the bandpass filter include sinusoids in the neighbourhood of 4800 Hz with a varying amplitude envelope. The sinusoids are strongest in the time domain when the 2.4 kHz RNCP message preamble is present and varies with the contents of the data stream. It was decided to rectify this signal in order to simply the decision process. This rectification was accomplished by taking the absolute value of the bandpass filter output.

The rectified filter output is further smoothed with a 250 Hz Butterworth low pass filter to remove rapid variation in the filtered rectified output amplitude.

The geometric decay operation is used as a computationally inexpensive peak holder to prevent rapid variation in the filtered rectified signal when a RNCP data stream with a majority of zeroes is present. The geometric decay serves as a debouncing mechanism, which will hold a peak level and then geometrically decay it over time when the peak level is falling. Without the geometric peak decay, a data stream sparse in 1's has the potential to cause data detect falsing (i.e. data mistaken for noise) by allowing the smoothed rectified output to rapidly decrease in amplitude.

5.3.4 DECISION

The decision block in Figure 5-10 decides on whether noise or data is present in the DISC data. The decision is made by comparing the output of the conditioned signal amplitude against hysteresis thresholds. Hysteresis is used because on occasion, the amplitude of the filtered rectified result can actually dip below the noise floor even with the geometric peak decay operation. These dropouts more or less rule out the use of single decision level, since excessive falsing may occur. Instead a two-level hysteresis decision scheme was implemented to debounce potential falsing which occurs when data is mistaken for noise

and vice-versa. In the hysteresis decision scheme, a data and noise floor was determined using a combination of simulation and experiments. A decision for "data present" was made if the conditioned signal amplitude exceeded the data floor. Once "data present" was decided, a decision for noise could not be made until the conditioned signal amplitude fell below the noise floor.

5.3.5 SIMULATIONS TO TEST ALGORITHM EFFECTIVENESS

A set of end to end simulations were performed to measure the effectiveness of the data detection algorithm once all the building blocks shown in Figure 5-10 were assembled. Three data sets distinguished only by the level of random noise preceding and following the RNCP preamble and data respectively were used in the tests. The data sets are shown in Figures 5-20a through 5-20c. The normally distributed random noise occupied sample locations 1 through 1024 and 4096 through 5144 in all data sets. The data in locations 1025 through 3072 contained 1024 samples of preamble. The RNCP data and preamble were simulated to be representative data for a worse case bit error rate of 10^{-2}. The spectrums appearing at node A in Figure 5-10 is shown in Figures 5-21a through 5-21c. The only real difference between the three spectral results is the magnitude of the spectrum for frequencies greater than 4800 Hz.

The result appearing at node B in Figure 5-10 is shown for all data sets in Figures 5-21a through 5-21c. The results at node B are used to decide whether the inbound channel is busy of not. Figures 5-21a through 5-21c show that a noise floor threshold of 0.02 and data floor of 0.04 is sufficient for the data detection algorithm to have a reasonable attack time and hang time without excessive data falsing. It should be noted that there is no definitive way to select the values for the hysteresis decision thresholds. The decision for the value of the thresholds must be based on the tradeoffs, which must be made between falsing and busy attack and hang times. Figures 5-21 show that if the data floor threshold is reduced then the busy attack time is improved but the risk of busy falsing in increased greatly since the threshold is closer to the average value of the noise. By the same token if the noise floor threshold is raised then the busy hang time is improved but risk of busy dropouts is increased.

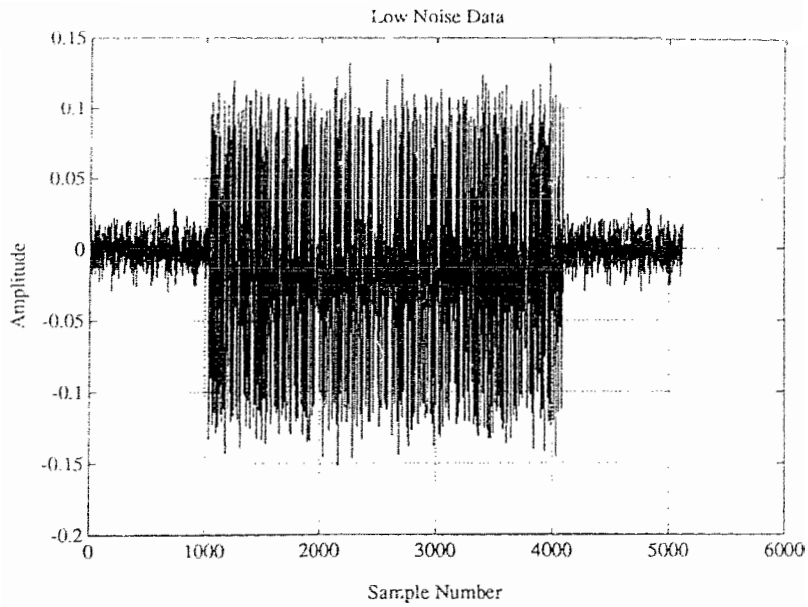


Figure 5-20a

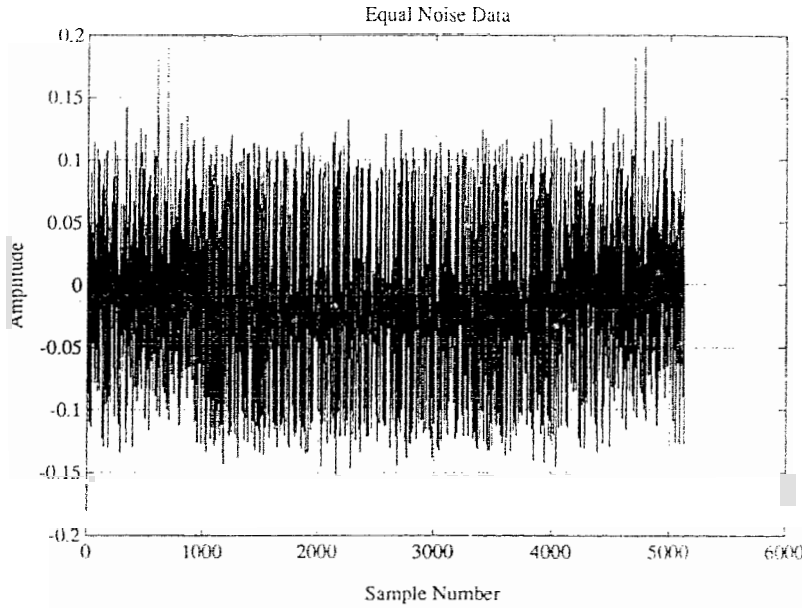


Figure 5-20b

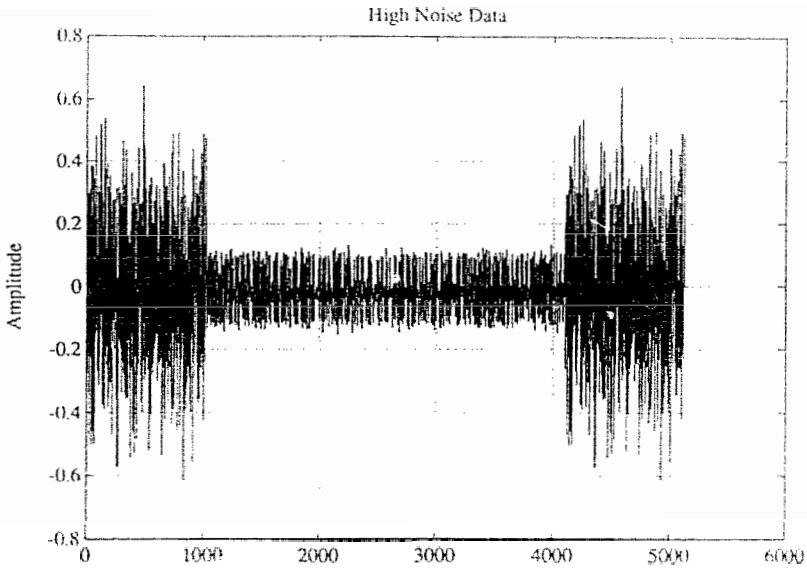


Figure 5-20c

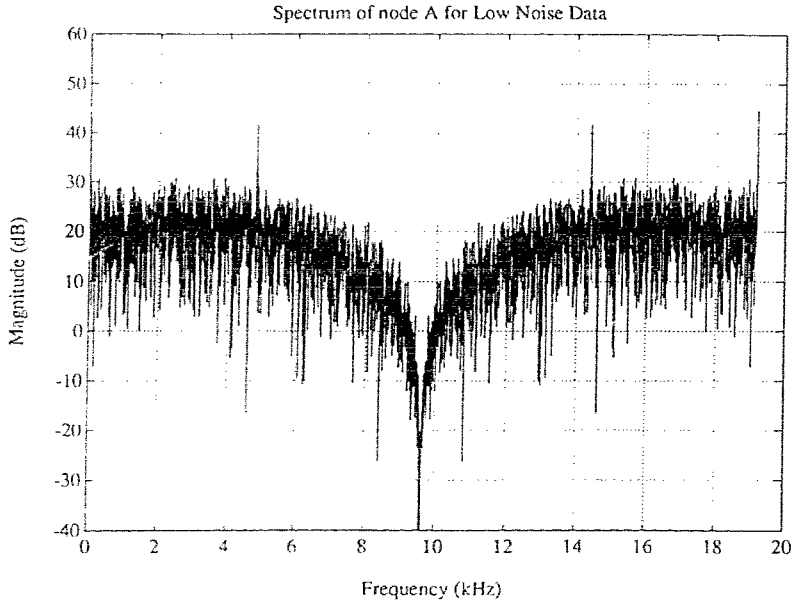


Figure 5-21a

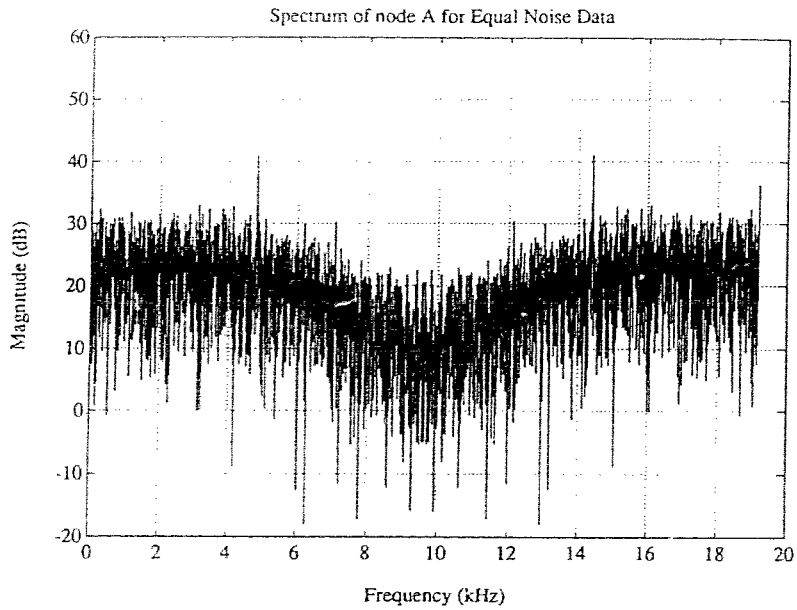


Figure 5-21b

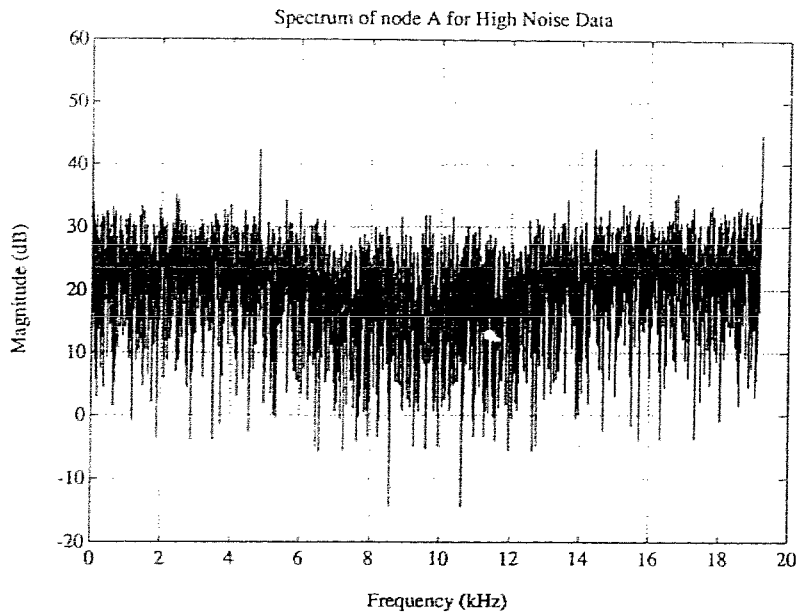


Figure 5-21c

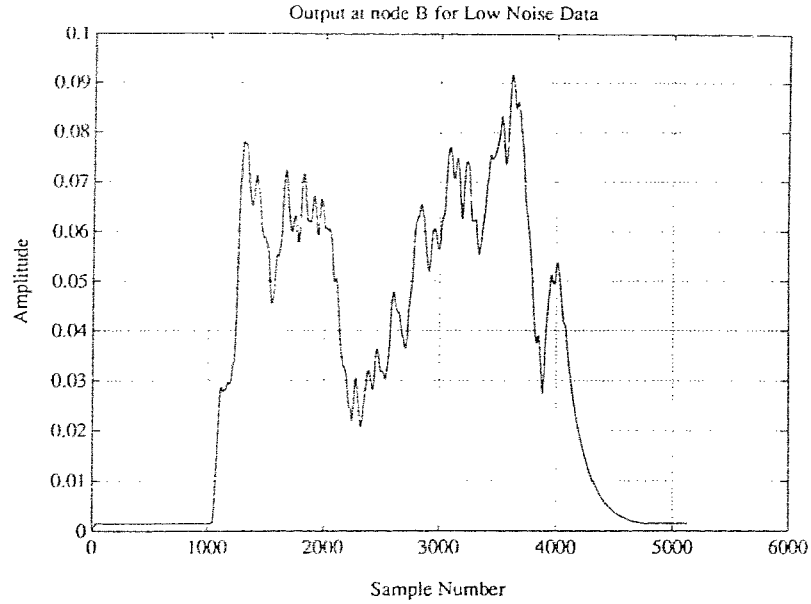


Figure 5-22a

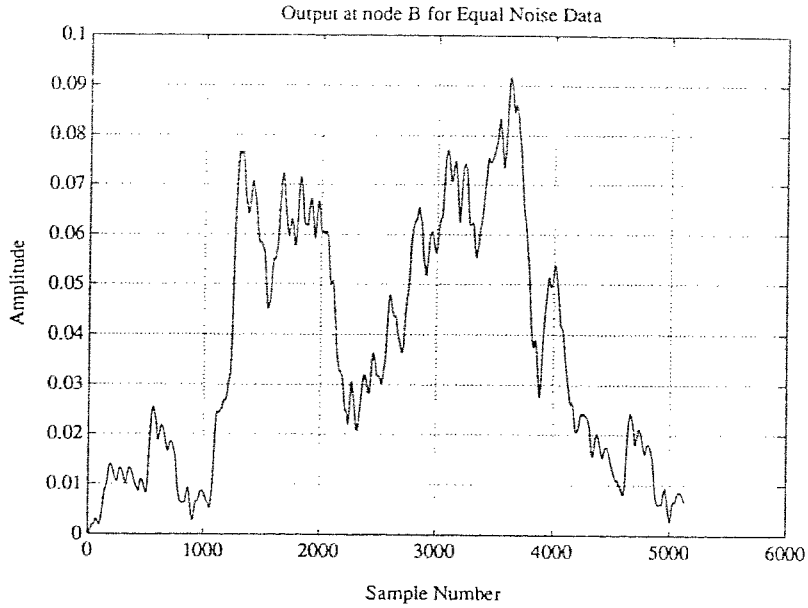


Figure 5-22b

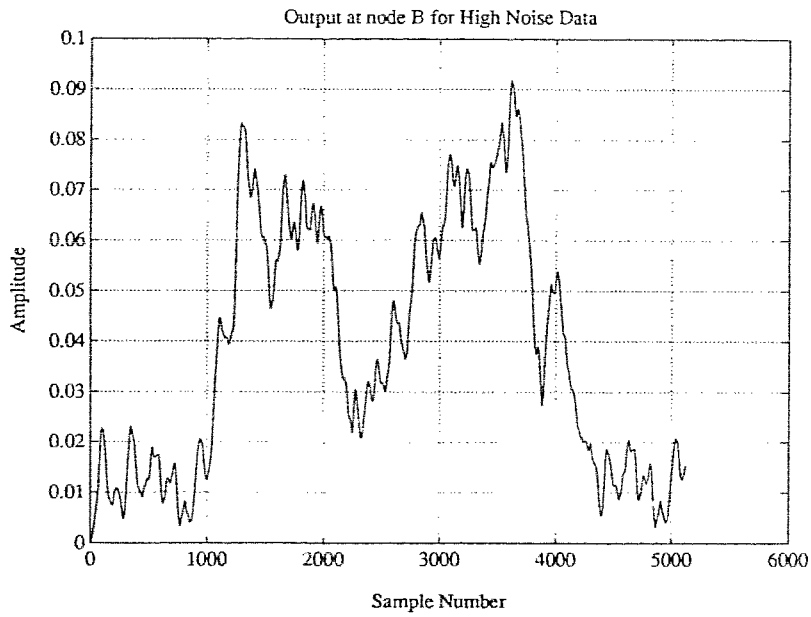


Figure 5-22c

6.0 PERFORMANCE RESULTS

The performance of the ARDPC modulation, demodulation and data detection algorithms were characterized using a series of tests with a typical transmitter and receiver combination. The emphasis was on the testing and characterization of the modem performance and not on the actual radio used. For example the static BER sensitivity of the modem is specified in terms of Eb/No rather than absolute carrier level, which implies that the tests will be repeatable using radios of different sensitivities. (Conversion from Eb/No to carrier level can be performed provided the noise figure and equivalent noise power bandwidth of the radio is known.)

The demodulator performance is characterized using both static and fading BER sensitivity tests and a bit timing recovery test, which measures the time to acquire a lock on a 2400 Hz tone. The modulator performance is characterized by measuring the spectral occupancy and comparing the bandwidth occupied against a standard FCC spectral emission mask. The data detection performance is characterized by the measurement of the busy attack and the busy hang times. The busy attack time is the time taken from the start of an inbound message packet to the time that the channel busy bit is actually set. The busy hang time is the time from the end of a message packet to the time that the channel busy bit is released. The processing bandwidth of the ARDPC is characterized by measuring the number of instructions used to perform functions.

6.1 BER SENSITIVITY TESTS

The BER sensitivity tests characterizes the raw bit-error performance of the modem demodulation algorithm. The block diagram for the test is shown in Figure 6-1. Two ARDPCs were used in the test; one to perform modulation and the other to perform demodulation. The DSP-56001 software was modified to work in conjunction with a protocol analyzer to generate RNCP data representative for a pseudorandom bit stream of pattern length 2047 and CCITT compatible block length. The output of the modulation ARDPC was upconverted to 800 MHz using a signal generator. The upconverted signal was converted to baseband using an 800 Mhz radio and the DISC output of the radio was fed to the demodulation ARDPC. A wideband noise source was used to supply additive white Gaussian noise. A Rayleigh fading simulator was used to simulate fading on the radio channel for Doppler frequency fade rates representative for vehicle speeds of 5, 20, 50 and 80 km /hr.

The graphs for the static and fading BER tests are shown in Figures 6-2 and 6-3. The static BER graph also includes a plot of the performance of the first generation RDPC under the same test conditions. This curve is provided to give a relative comparison of the performance of the demodulation algorithms for the first and second generation RDPCs. From Figure 6-2 it can be seen that the ARDPC with the DSP based software architecture yields a static BER performance 2-3 dB better than the first generation design. The differences in part can be attributed to the bandwidth of the low-pass filters used in the DSP based ARDPC. (The first generation design used switched capacitor filters with a low pass cut-off frequency of 5 kHz compared to the ARDPC which implemented a digital low-pass filter with a cut-off frequency of 3 kHz)

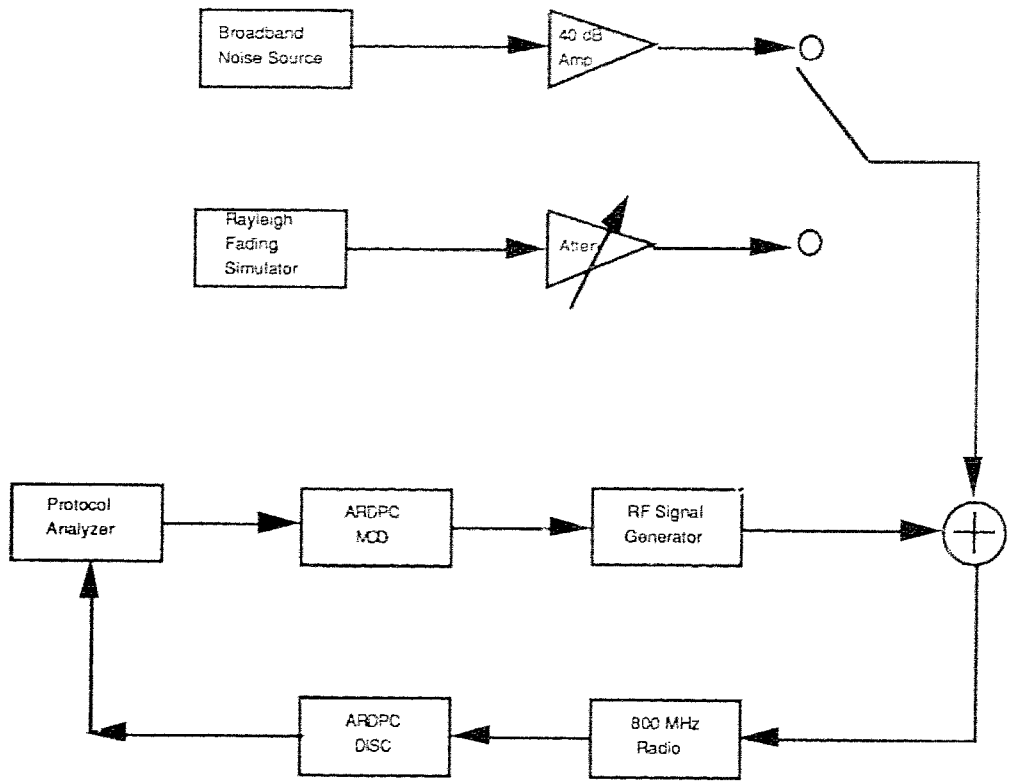


Figure 6-1 Block Diagram for BERT Sensitivity Tests

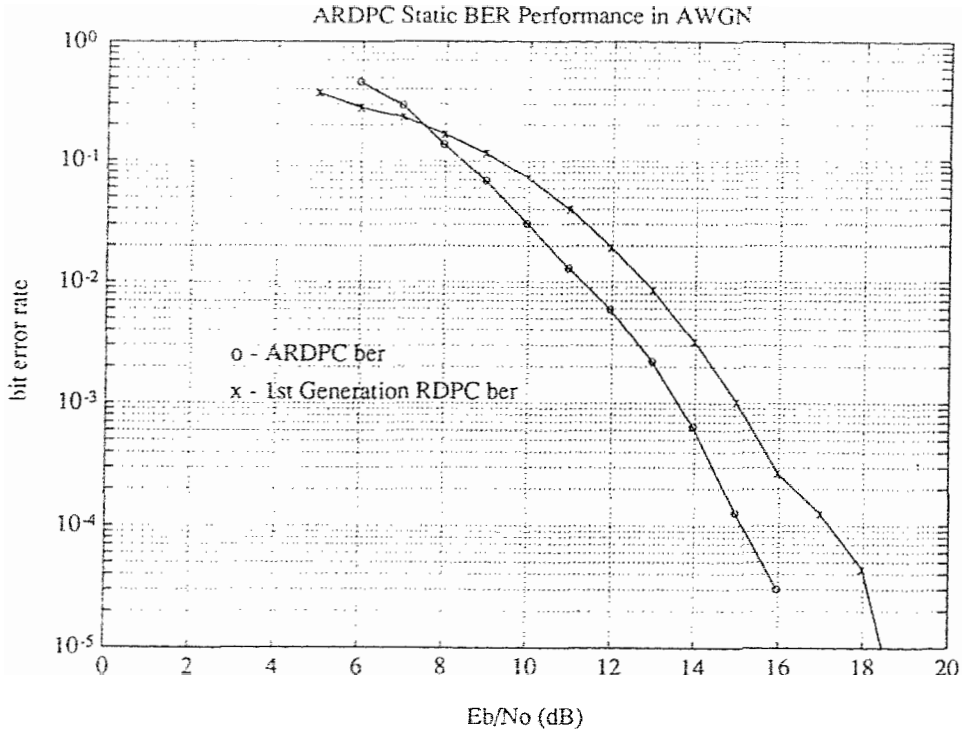


Figure 6-2 Static BER Performance of the ARDPC

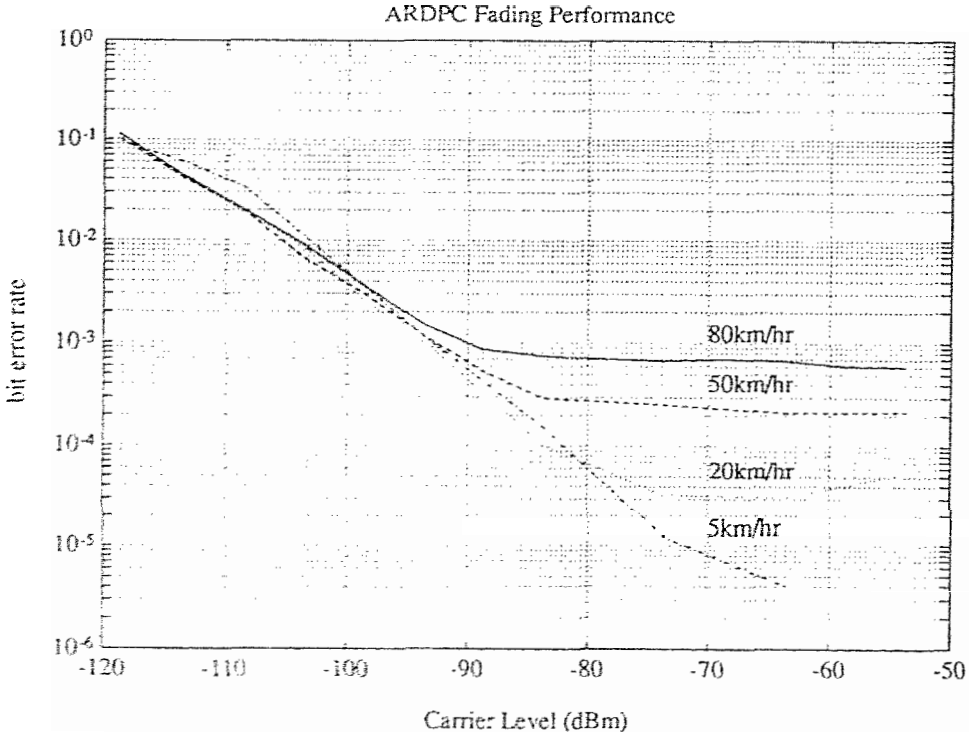


Figure 6-3 Fading BER Performance of the ARDPC

6-2 BIT TIMING RECOVERY

The bit timing recovery test measures the speed at which the ARDPC timing recovery algorithm is able to adjust its symbol sampling point to the centre of the received eye pattern. The block diagram for the test is shown in Figure 6-4.

The tests involved measuring the error in the timing recovery algorithm when receiving a 2400 Hz tone. Worst case signal to noise ratios of the tone signal were transmitted to an 800 MHz radio, whose DISC signal was connected to the DISC line of the ARDPC. The DSP-56001 software was modified to output the error in the timing recovery algorithm to hardware bits, which could be monitored by a digital storage oscilloscope (DSO). The time taken to lock (i.e. zero timing error) was determined directly from the oscilloscope trace of tone signal and the error signal.

The results for the test is below in Table 6-1

Table 6-1 Bit Timing Recovery Test Results

Mean time to lock @ < 10^{-5} static BER	4.2 msec
• 90% confidence level	5.7 msec
Mean time to lock @ 10^{-2} static BER	5.5 msec
• 90% confidence level	6.5 msec

These results in Table 6-1 are typically better then the first generation controller which averaged a mean lock time greater than 10 msec.

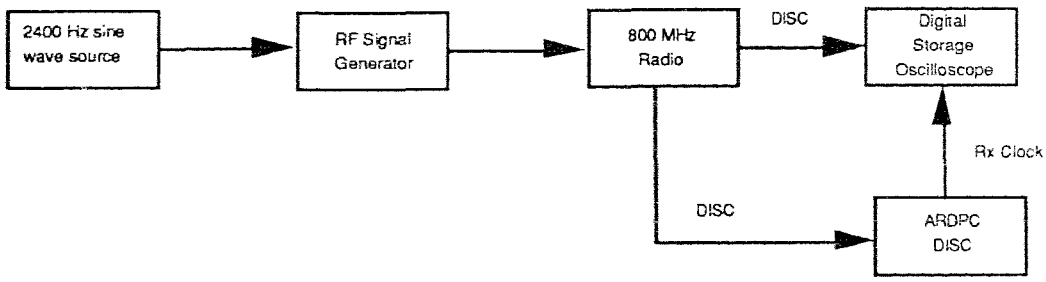


Figure 6-4 Block Diagram for Bit Timing Recovery Tests

6.3 SPECTRAL OCCUPANCY

The spectral occupancy tests were performed to determine if the modulated signal of the ARDPC falls within the spectral mask defined by the prevailing authorities such as the FCC for 25 kHz radio channel spacing with peak frequency deviation of 4 kHz.

The block diagram for the test setup is shown in Figure 6-5. The DSP-56001 software was modified to modulate pseudorandom data. Measurement of the modulated and unmodulated carrier signal was performed using a spectrum analyzer and a FCC spectral emission mask was overlaid over the measured spectrum. The measured spectrum for the modulated and unmodulated carrier is shown in Figure 6-6. It can be seen that the measured spectrum easily fits within the FCC mask.

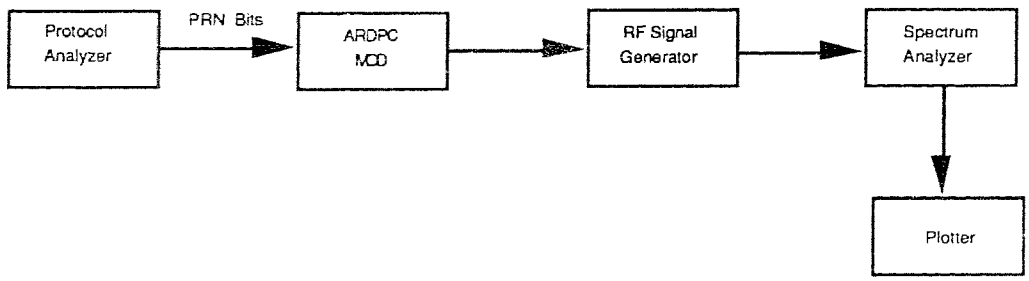


Figure 6-5 Block Diagram for Spectral Occupancy Tests

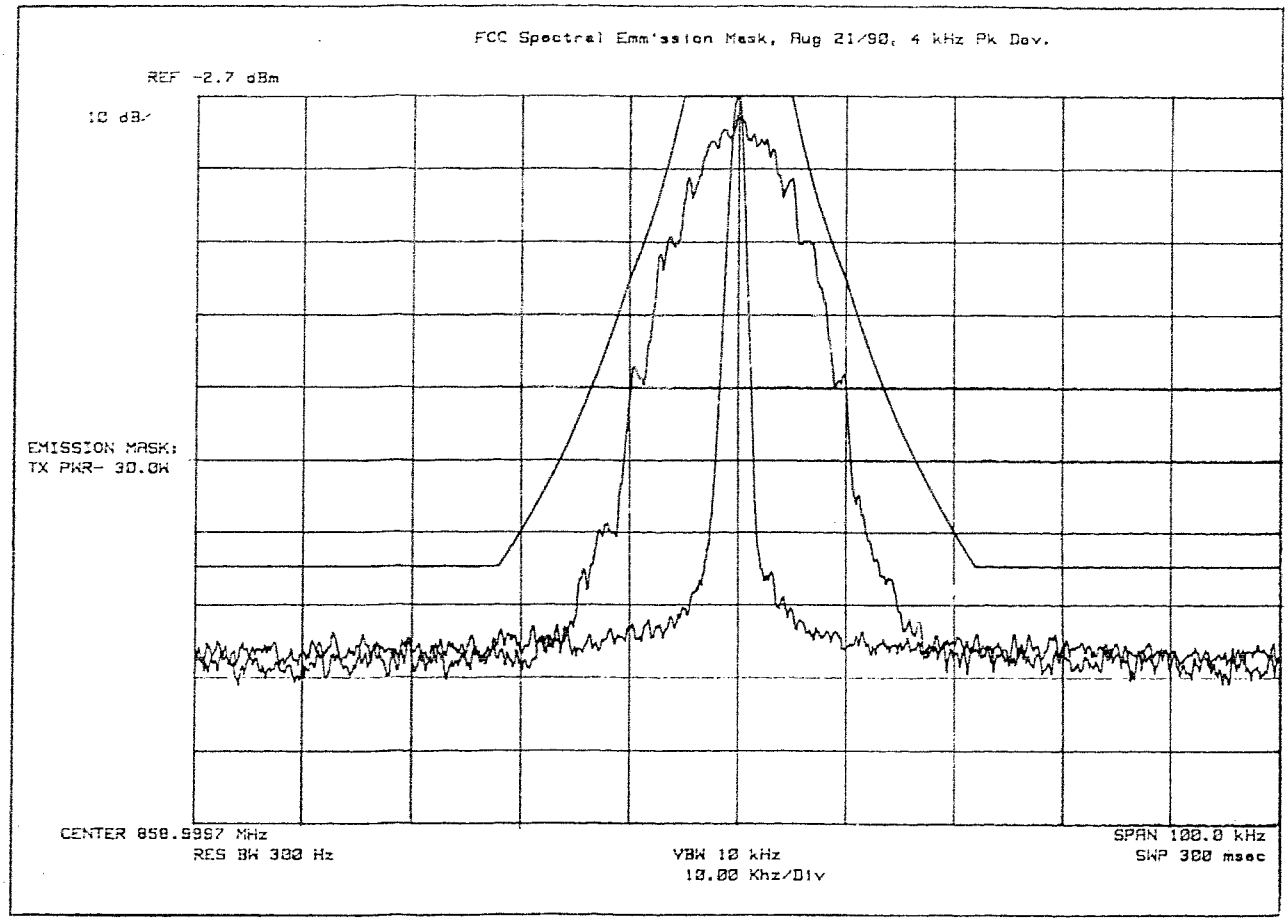


Figure 6-6 Spectral Occupancy Test Results

6.4 DATA DETECTION

The data detection tests characterizes the performance of the data detection algorithm under various conditions. The primary parameters measured are the busy attack time (i.e. time to detect the presence of data) and the busy hang time (i.e. lingering time from end of data). The busy attack time is measured from the start of a 2400 Hz tone to when the busy state is asserted in the DSP-56001. The hang time is measured from the end of actual message data to the time when the busy bit is de-asserted.

The block diagram for the test is shown in Figure 6-7. For the purposes of the test a 2400 Hz sine wave source was modulated at 4 kHz deviation and was received by an 800 MHz radio. The modulated signal was switched on or off using an RF switch. The busy attack time was measured directly off the DSO. One channel of the oscilloscope was attached to the data detection busy bit line while the other channel was attached to the DISC input line of the ARDPC.

The results of the test based on 50 trials is shown in Table 6-2.

Table 6-2 Data Detection Test Results

Mean busy attack time @ 10^{-2} BER	7.5 msec
Mean busy hang time @ 10^{-2} BER	28.5 msec

The results in Table 6-2 are typically better than the first generation controller which averaged 15 msec busy attack time and a busy hang time which could be much greater than 30 msec.

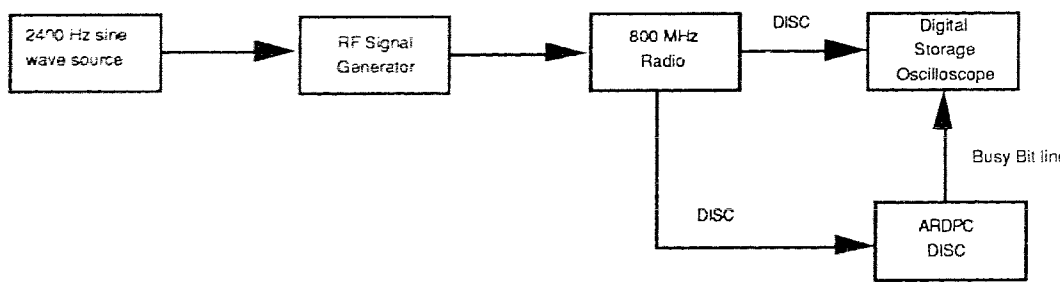


Figure 6-7 Block Diagram for Data Detection Tests

6-5 DSP-56001 PROCESSOR UTILIZATION

DSP-56001 processor utilization measures the average number of instructions used by the DSP-56001 software for performing various tasks. All measurements were referenced to a bit time. (The ARDPC provided a 25 MHz clock signal to the DSP-56001, which yielded an 80 ns instruction time. This provided approximately 2600 instructions available per bit time.) The instruction count was measured using a logic analyzer and is presented in Table 6-3.

Table 6-3 DSP-56001 Bandwidth Utilization

		% of bandwidth/bit time
Modulation		
• RNCP data	485	18.7
• Morse Code Station ID	954	36.7
Demodulation		
• Timing Recovery	397	15.2
• Data Detection	290	11.1
• Bit alignment		
- Hunt for SYNC	214	8.2
- Header Majority Vote/Extraction	1262	48.5
- Byte Alignment	260	10.0
• BCH decoding	357	13.7
Host Communication		
• Transfer message data to host	154	5.9
• Receive command from host	66	2.5
• Receive data from host	139	5.3
Average DISC processing Bandwidth Consumption		
• Timing Recovery	397	
• Data Detection	290	
• Byte Alignment of bits	260	
Total	947	36.4
Worst case DISC processing Bandwidth Consumption		
• Timing Recovery	397	
• Data Detection	290	
• Header Majority Vote/Extraction	1262	
• BCH header decoding	357	
• Transfer message data to host	154	
Total	2460	94.6
Average MOD/DISC processing Bandwidth Consumption		
• RNCP data modulation	485	
• Receive data from host	139	
• Data Detection	290	
• Timing Recovery	397	
• Hunt for SYNC	214	
Total	1525	58.7

7.0 CONCLUSIONS

In this thesis, a DSP-56001 software requirements analysis was performed and a proposed software architecture along with accompanying DSP algorithms was established for a 4800 bps mobile data communications radio and data packet controller. The performance of the software architecture and algorithms was proven to work successfully through various performance tests. Overall, the performance of the DSP based modem algorithms exceeded those of an earlier RDPC which was based on a custom VLSI communications chip. The static BER performance of the DSP based controller is better by 2-3 dB than the earlier bsc. The data detection attack and lingering times are better by up to 50% than the previous design.

The implemented DSP-56001 software architecture is general enough so that it can serve as the core for any low speed duo-processor modem design. However, the architecture still suffers from the constraints present in real-time applications such as processing bandwidth, memory availability and functional requirements. A close examination of these factors as well as a well-defined partitioning of functions is of utmost importance when working with the DSP-56001 in particular and with DSP chips in general. The functional partitioning in a duo-processor design such as the one used for the ARDPC must take into account the special capabilities of each processor. The DSP-56001, for example, can perform many microprocessor control functions but it would be a waste for it to do so since it is not optimized for the type of data manipulation that a microprocessor is capable of. In the development of the ARDPC, it was a design goal to minimize the control functions in the DSP-56001 and free it to become an arithmetic processing engine for the modem functions.

LIST OF REFERENCES

1. Lee, William C.Y., **Mobile Communications Engineering,** McGraw Hill Book Company, 1982.

2. Spilker Jr, James J., **Digital Communications by Satellite**, Prentice-Hall, 1977.

3. Lathi, B.P., **Modern Digital and Analog Communications System,** Holt, Rinehart and Winston, 1983.

4. Oppenheim, Alan V. and Scahfer Ronald W., **Digital Signal Processing,** Prentice- Hall, 1975.

5. **Handbook of Digital Signal Processing Engineering Applications,** Douglas F. Elliot editor, Academic Press Inc., 1987.

6. Michelson Arnold M. and Levesque Allen H., **Error-Control Techniques for Digital Communications**, John Wiley and Sons, 1985

7. Kloker, Kevin L., "The Motorola DSP56000 Digital Signal Processor", IEEE Micro, vol. 6, no. 6, pp 29-58, December 1986.

8. DSP 56000/56001 Digital Signal Processor User's Manual, DSP56000UM/AD Rev 2, Motorola.

9. Bateman A., and Martin P.M., "A DSP-based Generic Modem for Mobile Data Communications", Proceedings of Digital Signal Processing of Signals in Communications, Loughborough, vol 20-23, pp 85-87, September 1988.

10. Hughes, P.M., and Millar P.C., "Design of Digital Filters for Use in Frequency Shift Keyed Modems", Proceedings of Digital Signal Processing of Signals in Communications, Loughborough, vol 22-26, pp 269-276, April 1985.

11. Gill W.J. and Spilker Jr. J.J., "An Interesting Decomposition Property for the Self-Products of Random or Pseudorandom Binary Sequences", IEEE Transactions on Communications Systems, pp 246-247, June 1963