1

**PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT**

PRECISIONSM RESOLUTION TARGETS

**A Feature Suppressed Spread Spectrum
Communication System**

Wayne Patrick McPherson, B.Sc., B.Eng.

**CARLETON UNIVERSITY
Ottawa, Canada**

DOE-96-09

© copyright

2.2 Spread Spectrum Performance Parameters	8
2.3 Spread Spectrum System Architecture	11
2.4 Other Advantages of Spreading	12
2.5 Spread Spectrum Techniques	13
2.5.1 Direct Sequence	14
2.5.2 Frequency Hopping	16
2.5.3 Hybrid Systems	20
2.6 Pseudorandom Noise Sequences	24
2.6.1 PN Code Generation	25
2.6.2 Maximal Length Sequences	25
2.6.3 Additional Useful Properties	29
2.7 Exploitation Of Low Probability Of Intercept Signals	29
2.7.1 The Wide Band Radiometer	30
2.7.2 Carrier Detection	32
2.7.3 Chip-Rate Detection	34
2.7.3.1 CRD Figure of Merit	37
2.7.3.2 CRD Performance Analysis	39
 3.0 A THEORETICAL DEVELOPMENT OF CHIP-RATE HOPPING	41
3.1 Introduction	41
3.2 Conventional Autocorrelation and Power Spectrum	44
3.3 Autocorrelation and Power Spectrum of Chip-Rate Hopping	48

ABSTRACT

The novel technique of chip-rate hopping (CRH) was devised to suppress the chip rate feature of a Direct-Sequence-Binary-Phase-Shift-Keying (DS-BPSK) communication system. Chip-rate hopping deterministically varies the chip rate through the utilization of a second Pseudonoise Sequence (PNS) in addition to the conventional PNS used to spread the transmitted data. A second enhancement, burst data transmission, was added. Spread data is transmitted on every eighth chip-rate, while a third PNS is transmitted when data is absent.

A hardware implementation of a DS-BPSK transmitter is described. Experimental results demonstrate that the chip-rate feature has been suppressed when the signal is exposed to an envelope-type chip-rate-detector (CRD). In particular, the chip-rate feature was undetectable at an input signal-to-noise ratio (SNR) of -10 dB when either a swept-superheterodyne or an FFT spectrum analyser was used in a 10 minute examination of the known bandwidth. A software implementation of a matched filtering demodulation scheme was utilized to demonstrate acceptable performance is achievable at an input SNR of -15 dB.

The undersigned hereby recommend to
the faculty of Graduate Studies and Research
acceptance of the thesis

**A FEATURE SUPPRESSED SPREAD SPECTRUM
COMMUNICATION SYSTEM**

submitted by
Wayne Patrick McPherson, B.Sc., B.Eng.

In partial fulfilment of the requirements
for the degree of Master of Engineering.

Chair, Department of Electronics

Thesis Supervisor

Thesis Co-Supervisor

Carleton University

September 1996

ABSTRACT

The novel technique of chip-rate hopping (CRH) was devised to suppress the chip rate feature of a Direct-Sequence-Binary-Phase-Shift-Keying (DS-BPSK) communication system. Chip-rate hopping deterministically varies the chip rate through the utilization of a second Pseudonoise Sequence (PNS) in addition to the conventional PNS used to spread the transmitted data. A second enhancement, burst data transmission, was added. Spread data is transmitted on every eighth chip-rate, while a third PNS is transmitted when data is absent.

A hardware implementation of a DS-BPSK transmitter is described. Experimental results demonstrate that the chip-rate feature has been suppressed when the signal is exposed to an envelope-type chip-rate-detector (CRD). In particular, the chip-rate feature was undetectable at an input signal-to-noise ratio (SNR) of -10 dB when either a swept-superheterodyne or an FFT spectrum analyser was used in a 10 minute examination of the known bandwidth. A software implementation of a matched filtering demodulation scheme was utilized to demonstrate acceptable performance is achievable at an input SNR of -15 dB.

ACKNOWLEDGEMENTS

This work is dedicated to the memory of Raymond Bruce McPherson; a man who always had a positive attitude, an encouraging word, and demonstrated the value of hard work and a job well done every day. He was one of my best friends and is sorely missed.

I would also like to acknowledge the support and encouragement of my family, Marjorie and Ian, over the course of this work. A huge thank-you goes to Julie, who continued to help me over the rough spots, put things in perspective, and was instrumental in its completion. I couldn't have done it without her. Also, Basil and Heather Grace for providing me with a much needed and much appreciated home away from home.

I would like to thank all of the people at the Defence Research Establishment Ottawa for their help and technical expertise. In particular: Robert Fry and Andrew Mudry for their hardware and software debugging skills and timely suggestions. Graham Estey and Bob Humble for their network and PC expertise, and Lorne Gütz for his design help on the hardware.

Last but not least, a huge thanks goes to Jim Wight at Carleton University, Douglas Hill at DREO, and Hans Haugli at VISTAR Telecommunications, all of whom provided guidance, support, encouragement, technical expertise, advice, and understanding in allowing me to juggle my time between projects and finish this work.

TABLE OF CONTENTS

ACCEPTANCE SHEET	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
LIST OF SYMBOLS	ix
LIST OF ABBREVIATIONS	xii
LIST OF FIGURES	xiv
LIST OF TABLES	xvii
1.0 INTRODUCTION	1
1.1 Motivation	1
1.2 Objective and Outline	4
2.0 SPREAD SPECTRUM THEORY	6
2.1 Introduction	6

2.2 Spread Spectrum Performance Parameters	8
2.3 Spread Spectrum System Architecture	11
2.4 Other Advantages of Spreading	12
2.5 Spread Spectrum Techniques	13
2.5.1 Direct Sequence	14
2.5.2 Frequency Hopping	16
2.5.3 Hybrid Systems	20
2.6 Pseudorandom Noise Sequences	24
2.6.1 PN Code Generation	25
2.6.2 Maximal Length Sequences	25
2.6.3 Additional Useful Properties	29
2.7 Exploitation Of Low Probability Of Intercept Signals	29
2.7.1 The Wide Band Radiometer	30
2.7.2 Carrier Detection	32
2.7.3 Chip-Rate Detection	34
2.7.3.1 CRD Figure of Merit	37
2.7.3.2 CRD Performance Analysis	39
 3.0 A THEORETICAL DEVELOPMENT OF CHIP-RATE HOPPING	41
3.1 Introduction	41
3.2 Conventional Autocorrelation and Power Spectrum	44
3.3 Autocorrelation and Power Spectrum of Chip-Rate Hopping	48

3.3.1 A Specific Example of Chip-Rate Hopping	50
3.4 Conventional Matched Filtering	51
3.4.1 Digital Matched Filtering of Chip-Rate Hopping	56
3.4.2 A Specific Example of Digital Matched Filtering of a Chip-Rate Hopped Signal	56
 4.0 THE IMPLEMENTATION OF A CHIP-RATE SUPPRESSED DSSS COMMUNICATION SYSTEM	 58
4.1 Introduction	58
4.2 Transmitter Design Considerations	59
4.3 Selection Of Transmitter Design Parameters	60
4.4 High Level Transmitter Board Implementation	64
4.5 Low Level Transmitter Board Implementation	67
4.5.1 High Level Circuit Operation	67
4.5.2 STEL 1032 Setup Circuit	70
4.5.3 STEL 1272B Board Interface	75
4.5.4 Other Circuits	79
4.6 Matched Filter Hardware Design Considerations and Implementation	81
4.7 Matched Filter Software Design Considerations and Implementation	83
4.7.1 Program Description	84
 5.0 CHIP-RATE HOPPING TESTING AND RESULTS	 88

5.1 Introduction	88
5.2 Verification of Correct Operation	89
5.3 Chip-Rate Detection Test Parameters	94
5.4 Chip-Rate Detector Results	96
5.5 Dynamic Matched Filter Testing	101
 6.0 SUMMARY AND RECOMMENDATIONS	 108
6.1 Introduction	108
6.2 Summary	109
6.3 Future Work	111
 Appendix A: Circuit and Timing Diagrams	 113
 Appendix B: Software Listings For E ² PROM Data	 120
 Appendix C: Software Listing For Dynamic Matched Filter	 132
 REFERENCES	 143

LIST OF SYMBOLS

a_n	Coefficient of a pseudonoise sequence
b_m	Coefficient of a second pseudonoise sequence
B	Data bandwidth
B_α	Chip-rate detector bandwidth
B_h	Chip-rate hop bandwidth
B_l	Lower limit of hop band
B_u	Upper limit of hop band
C_n	Shift register generator output
$c(t)$	Pseudonoise sequence
d	Arbitrary time delay
$d(t)$	Second pseudonoise sequence
$E[\]$	Expected value
E_c / N_0	Chip energy to one-sided noise power density ratio
E_j	Jamming energy
E_s	Signal Energy
f_c	Chip rate
$\mathcal{F}[\]$	Fourier transform
$\mathcal{F}^{-1}[\]$	Inverse Fourier transform
$f(f_c)$	probability density function for chip rate, f_c
G_p	Processing gain

$\gamma(f)$	Power spectral density of carrier component, carrier detection
$H^*(f)$	Complex conjugate of the matched filter frequency response
$h(t)$	Matched filter impulse response
J	Jamming margin
L	Length of hopping maximal length sequence
μ	Mean hop frequency
N	Length of spreading maximal length sequence
$n(t)$	Noise component of received signal
$N_0/2$	Two-sided noise power spectral density
$o(t)$	Output of chip-rate detector
$p(t)$	Baseband pulse
$P^*(f)$	Complex conjugate of Fourier transform of $p(t)$
$r(t)$	Received signal
$R_s(\tau)$	Continuous autocorrelation function of a maximal length sequence
$R_{s,d}(\tau)$	Continuous cross-correlation function of two maximal length sequences
$R_{cm}(\tau)$	Autocorrelation function of chip-rate hopping
$s(t)$	Transmitted baseband signal
$S_s(f)$	Power spectral density of a maximal length sequence
$S_{cm}(f)$	Power spectral density of chip-rate hopping
σ^2	variance of hop frequency
$\text{sinc}(x)$	$\sin(x) / x$
S_{nn}	Power spectral density of noise spectra, chip-rate detection

S_{ω}	Power spectral density of signal spectra, chip-rate detection
t	Time
T	Integration period
T_c	Chip period
T_s	Duration of a finite length signal
$\theta_c(k)$	Discrete autocorrelation of a maximal length sequence
$\theta_{c,d}(k)$	Discrete cross-correlation of two maximal length sequence
V	Magnitude of radiometer output
W	Spreading bandwidth
$x(t)$	Output of baseband matched filter, continuous
x_k	Discrete output of baseband matched filter
x_{k_i}	Discrete output of baseband matched filter, chip-rate hopping
$y(t)$	Output of squaring circuit, carrier detection
Z	Chip-rate spacing

LIST OF ABBREVIATIONS

A/D	Analog to Digital
AWGN	Additive White Gaussian Noise
BPSK	Binary Phase Shift Keying
CDMA	Code Division Multiple Access
CRD	Chip-rate detector
CRH	Chip-rate hopping
DSSS	Direct Sequence Spread Spectrum
ECM	Electronic Counter Measure
ECCM	Electronic Counter-Counter Measure
ESM	Electronic Support Measure
FFT	Fast Fourier Transform
FFH	Fast Frequency Hopping
FH	Frequency Hopping
HF	High Frequency
Hz	Hertz
kb	kilobit
LFSR	Linear feedback shift register
LPE	Low Probability of Exploitation
LPI	Low Probability of Intercept
m-sequence	Maximal length sequence

Mb	Megabit
MF	Matched Filter
MSK	Minimum Shift Keying
OQPSK	Offset Quaternary Phase Shift Keying
PN	Pseudonoise
PNS	Pseudonoise Sequence
ppm	Parts per million
QPSK	Quaternary Phase Shift Keying
SFH	Slow Frequency Hopping
SNR	Signal to Noise Ratio
SQPSK	Staggered Quaternary Phase Shift Keying
SR	Stored Reference
SSRG	Simple Shift Register Generator
TR	Transmitted Reference

LIST OF FIGURES

Figure 2.1	Power spectrum of a digital signal before spreading	7
Figure 2.2	Power spectrum of a digital signal after spreading	7
Figure 2.3	Attempted jamming of a spread signal	10
Figure 2.4	Empirical processing gain curve	10
Figure 2.5	A simplified example of direct sequence spreading	15
Figure 2.6	Received signal plus jamming	15
Figure 2.7	Rejecting jamming by despreading	15
Figure 2.8	A typical frequency hopping time-frequency map	17
Figure 2.9	Frequency hopping system architecture	19
Figure 2.10	A frequency hopped-direct sequence hybrid system	22
Figure 2.11	Hybrid signal spectrum (logarithmic scale)	22
Figure 2.12	Code generation using a ssrg	26
Figure 2.13	Autocorrelation function of an m-sequence	28
Figure 2.14	Power spectrum of an m-sequence	28
Figure 2.15	Radiometer operation	31
Figure 2.16	Carrier detection operation	33
Figure 2.17	Chip-rate detection	36
Figure 3.1	Comparison of the autocorrelation functions of chip-rate hopping (9-10 MHz) with that of a single chip rate at 9.5 MHz	52
Figure 3.2	Comparison of the power spectra of chip-rate hopping (9-10 MHz) with that	

of a single chip rate at 9.5 MHz	52
Figure 3.3 Physical realization of a digital matched filter	55
Figure 4.1 Chip-rate hop pattern using 8 stage code [8, 4, 3, 2] with $2^8 - 1 = 255$ different chip rates	66
Figure 4.2 Time-frequency map of implemented transmitter	66
Figure 4.3 Implemented chip-rate hopping transmitter board	68
Figure 4.4 Overview of transmitter board operation	69
Figure 4.5 Overview of transmitter board timing	71
Figure 4.6 STEL 1032 setup circuit	73
Figure 4.7 STEL 1032 setup circuit timing	74
Figure 4.8 STEL 1272B board interface	76
Figure 4.9 STEL 1272B board interface timing	77
Figure 4.10 Flowchart of matched filter program	85
Figure 5.1 Verification of hopping band and chip rates	91
Figure 5.2 Verification of chip-rate spacing	92
Figure 5.3 Verification of frequency accuracy	93
Figure 5.4 Chip-rate detection experimental setup	95
Figure 5.5 Detected chip-rate lines with (upper) and without (lower) hopping; $E_c/N_0 = 0$ dB, cumulative sweep time = 10 minutes	98
Figure 5.6 Detected chip-rate lines with (upper) and without (lower) hopping; $E_c/N_0 = 0$ dB, cumulative sweep time = 30 seconds	98
Figure 5.7 Detected chip-rate lines with (upper) and without (lower) hopping; $E_c/N_0 =$	

-10 dB, cumulative sweep time = 10 minutes	99
Figure 5.8 Hopping mode detection results using an FFT analyser: $E_c/N_0 = 0$ dB, cumulative sweep time = 10 minutes	99
Figure 5.9 Hopping mode detection results using an FFT analyser: $E_c / N_0 = -10$ dB, cumulative sweep time = 10 minutes	100
Figure 5.10 Illustration of the 200 Hz bandwidth of the chip-rate spectra	100
Figure 5.11 Matched filter test setup	102
Figure 5.12 Despreading results; chip rate = 9.003922 MHz, $E_c/N_0 = \infty$	104
Figure 5.13 Despreading results; chip rate = 9.003922 MHz, $E_c/N_0 = 0$ dB	104
Figure 5.14 Despreading results; chip rate = 9.003922 MHz, $E_c/N_0 = -15$ dB	105
Figure 5.15 Despreading results; chip rate = 9.003922 MHz, $E_c/N_0 = -20$ dB	105
Figure A1 Control circuit schematic	113
Figure A2 Control circuit timing	114
Figure A3 Data circuit schematic	115
Figure A4 Data circuit timing	116
Figure A5 Chip clock circuit schematic	117
Figure A6 Chip clock circuit timing	118
Figure A7 Output circuit schematic	119

LIST OF TABLES

Table 4.1	Demonstration Of Equal Probability Of A Chip Rate Containing Data	62
Table 4.2	Operational Features of the Chip-Rate Hopping Transmitter	65

CHAPTER I

1.0 INTRODUCTION

1.1 Motivation

The Department of National Defence, in conjunction with Carleton University, has identified a requirement to investigate the feasibility of making Direct Sequence Spread Spectrum (DSSS) communications signals more covert by suppressing detectable features such as the chip rate. This thesis presents the theory, implementation, and experimental results of the investigation, performed at the Defence Research Establishment Ottawa under Supply and Services Canada contract number W7714-4-9826/01-SV.

Both covertness and channel security are of prime importance in applications such as tactical military communications. Covertness may be defined as having a low probability of detection or interception (LPI), while channel security may be defined as having a low probability of exploitation (LPE). DSSS communication systems are popular for many reasons including their low probability of intercept, multipath interference resistance, jamming resistance, and multiple access features. However, simple techniques exist which enable a potential foe to both intercept and extract features from a DSSS signal; thus the requirement for a highly reliable, robust, and even more covert DSSS communication

system.

A secure communication system is one which resists unintentional interference, jamming, detection, and unauthorized extraction of information by unauthorized users. This intelligence may include data modulation technique and range and bearing information. Historically, these specifications were primarily for the purpose of development of military communications and applications. In this discussion, the military aspect of these systems will be explored. Some or all of the principles may be extrapolated to civilian applications such as digital cellular telephony, personal communication systems (PCS), and wireless networks.

The designer of a tactical radio system or network is faced with a number of important requirements [1]. These include:

- i) ESM and ECM resistance (defined below)
- ii) integrated voice and data to the user
- iii) performance to meet the projected user demand, i.e. error detection/correction, quality, and delays.
- iv) effective use of the transmission medium
- v) interoperability with defined other networks
- vi) flexibility in deployment
- vii) survivability
- viii) provision of user mobility

It is difficult to achieve all of these requirements simultaneously in a particular application. As such, this study is only concerned with the first Electronic Support Measures (ESM) and Electronic Counter Measures (ECM) resistance. ESM involves searching, intercepting, locating, and analysing radiated electromagnetic energy for the purpose of exploitation in the support of military operations. ECM involves the prevention or reduction of an enemy's use of the RF spectrum [2].

The goals of the interceptor are four-fold [3, p.6]. First, the signal must be covered; i.e., the receiver must be tuned to the frequency at which the signal is being transmitted at the time which it is being transmitted. Secondly, the signal must be detected. The interceptor must make a decision regarding whether or not the received power contains a signal plus noise and interference, or if the received power consists of noise plus interference only. Thirdly, the interceptor must extract features from this signal to determine whether the signal intercepted is of interest or not. These features can include carrier frequency, chip rate, data rate, and modulation information. Finally, the interceptor may attempt to exploit the signal by extracting additional features and ultimately demodulating the baseband signal to generate a bit stream. Furthermore, the interceptor may wish to block the information flow (jamming) and/or insert false information (spoofing).

The communicator has two basic objectives [4, p.25]. First, a reliable communication link must be established and maintained. Secondly, any attempt by an interceptor to disrupt this link must be counteracted. Several methods exist to reduce the likelihood of signal

interception. First and foremost is the minimization of transmit power. When coupled with burstiness, the interceptor must endeavour to detect a low power, short duration signal. The utilization of high gain antennas results in a narrow radiation pattern which is difficult to intercept and jam. Long-range interception of communication signals is not possible when the communicator employs frequencies above about 30 MHz, as the radio waves will not reflect off the ionosphere. Communication using coaxial or optical cables will virtually eliminate the interception problem, but is a highly impractical solution on a dynamic platform.

A popular method of minimizing detectability and/or exploitation is the use of novel coding techniques, one of which is spread spectrum. However, the interceptor soon discovered how to exploit this waveform (ESM). As a result, the communicator must find a way to defeat or counter this exploitation (ECM). The interceptor is then challenged with the task of devising a method of defeating or countering the communicator's countermeasures, namely Electronic Counter-Counter Measures (ECCM). The covert communication arena is therefore a continually evolving and sophisticated battle of defeating the innovations of one's opponent while simultaneously maintaining acceptable performance.

1.2 Objective and Outline

The objective of this research was to design, implement, and evaluate a communication system which was difficult to detect when using a filter-and-square envelope-type chip rate

detector (CRD). The novel technique of Chip-Rate Hopping (CRH) involves the pseudorandom variation of the chip rate of a Direct Sequence (DS) signal. This is accomplished through the use of a second m-sequence in addition to the one which spreads the data. A third m-sequence is employed for transmission on non-data hops. Such a system therefore has both a low probability of interception (LPI) and a low probability of exploitation (LPE). The failure to find any references to such a technique in either an open literature search or a patent search suggests that this technique has not yet been publicly examined.

The present discussion will focus on spread spectrum theory and techniques. In particular Direct Sequence (DS) and methods for interception are presented in Chapter II. The theory of Chip-Rate Hopping is introduced and developed in Chapter III. The design and implementation of both a hardware CRH transmitter board employing DS-Binary-Phase-Shift-Keying (DS-BPSK) modulation, and a software digital matched filter are presented in Chapter IV. Supporting experimental evidence, found in Chapter V, demonstrates the feasibility of Chip-Rate Hopping as a valid technique to counter filter-and-square envelope-type chip-rate detectors. Demodulation results employing a software implementation of a matched filter may also be found in Chapter V and show that acceptable communication performance has been achieved. Chapter VI provides a summary of the work and gives general conclusions, including suggestions for future work.

CHAPTER II

2.0 SPREAD SPECTRUM THEORY

2.1 Introduction

Spread spectrum communication systems have been in use for close to 40 years. Initial development was for military purposes such as anti-jamming tactical communications and weapons guidance systems. The concept may be defined as follows: Spread spectrum is a means of transmission in which the signal occupies a bandwidth in excess of the minimum necessary to send the information. The band spread is accomplished by means of a code which is independent of the data, and a synchronized reception with the code at the receiver is used for despreading and subsequent data recovery [5]. The transmitted power, in each case, is identical. In effect, a narrow band, high amplitude signal (Figure 2.1), is spread such that a wide band, low amplitude signal (Figure 2.2) is transmitted over the channel. It would appear at first glance that spreading the desired information over a wide bandwidth is an inefficient method of communication. However, several desirable characteristics result. Some of these include resistance to jamming, interference rejection, low probability of intercept, multiple access, high resolution ranging, and accurate universal timing [5] - [10].

Figure 2.1 Power spectrum of a digital signal before spreading

Figure 2.2 Power spectrum of a digital signal after spreading

The most popular methods of achieving this spread are Direct Sequence (DS), Frequency Hopping (FH), and hybrids of the two. The fundamental concept of these techniques is such that the spreading is accomplished by encoding the data with a pseudorandom noise sequence (PNS) which is independent of the information itself [6]. The sequences themselves will be discussed in detail in Section 2.6. An additional spreading technique, Time Hopping (TH), exists whereby a signal is transmitted by pulses with varying time intervals between these pulses. These intervals are determined by the PNS. Time Hopping is neither as popular nor as widespread as the aforementioned spreading techniques and hence will not be addressed. It is first necessary to examine the performance parameters by which spread spectrum systems are evaluated to discover why spread spectrum is an attractive method of information exchange. Much of the following discussion is condensed from [5], unless otherwise noted.

2.2 Spread Spectrum Performance Parameters

The parameters most often cited in the evaluation of a spread spectrum system are processing gain (G_p) and jamming margin (J). The processing gain is a measure of the interference rejection capability and may be defined as the ratio of the spread bandwidth (W) to the data bandwidth (B) [7, p.89]:

$$G_p = \frac{W}{B} \quad (2.1)$$

Alternatively, the processing gain may be defined as the ratio of output signal to noise ratio (SNR) to input SNR at the despreader:

$$G_p = \frac{SNR_{output}}{SNR_{input}} \quad (2.2)$$

Jamming is simply an attempt to disrupt communication service by transmitting an interfering signal into the frequency band in which the communicator is attempting to transmit (Figure 2.3). Thus, in the presence of jamming, the signal to noise ratio (SNR) at the data demodulator may be described by:

$$SNR = \frac{E_s}{E_j} G_p \quad (2.3)$$

where E_s denotes the received signal energy, E_j denotes the received jamming energy, and G_p is the processing gain given in (2.1). Thus, it is observed that although SNR is reduced due to jamming, it is increased by the processing gain, since W , the spread bandwidth, is much greater than B , the data bandwidth.

As a direct consequence, the designer must allow for a jamming margin. In simple terms, if the jamming energy exceeds the processing gain, system performance will degrade, perhaps to the point where data recovery is not possible. Figure 2.4 is a typical processing gain curve showing ideal SNR and SNR degradation due to the presence of jamming. The jamming threshold, i.e., the point where communication is no longer possible, is typically taken to be the point where the two curves differ by 1 dB.

Figure 2.3 Attempted jamming of a spread signal

Figure 2.4 Empirical processing gain curve

To summarize, the requirements of digitally modulated spread spectrum communication in a jamming environment are threefold. Firstly, the spread bandwidth must be much larger than the signal bandwidth to give a high processing gain. Secondly, receivers must be designed such that orthogonal waveform jamming components are rejected. Finally, the waveforms used must be randomly or pseudorandomly generated, and be equally likely to occur anywhere in the available bandwidth.

2.3 Spread Spectrum System Architecture

It is evident that a spread spectrum system must have the capability of generating and receiving all of the possible waveforms of an orthogonal signal set. In addition, a random selection process, selecting one of M such signals, must be independently repeated each time a signal is transmitted. This is necessary, of course, to avoid jamming in the event that the signal set is predicted. As such, three possible system configurations may be proposed: Transmitted Reference (TR), Stored Reference (SR), and Matched Filter (MF) [9, vol. 1], [10].

In a TR implementation, two signals are generated: one modulated and one unmodulated. The receiver then recovers both signals and extracts the data. There are several problems with such an implementation. First, data may be easily corrupted when a jammer sends a signal which is acceptable to the receiver as the transmitted signal. Performance will be poor at low signal levels since noise and interference, present on both signals, are cross correlated.

Data may be easily extracted by a listener who has access to both waveforms. Finally, since two channels are required, extra bandwidth may be necessary.

An SR system requires an independent generation at both the transmitter and receiver of identical pseudonoise sequences (PNS). The receiver, therefore, must be synchronized with the incoming waveform. Demodulation is achieved by cross-correlation.

In a matched filter link, the spread signal is generated by pulsing a filter having a pseudorandom, wide impulse response. Data recovery is accomplished using an identical, synchronous filter at the receiver. Externally, matched filter systems have identical properties to stored reference systems. However, the systems differ internally in the detection process.

2.4 Other Advantages of Spreading

As mentioned previously, there are several advantages to spreading a signal. The antijamming capability has already been discussed. Other advantages include Low Probability of Intercept (LPI), multiple access, and ranging. LPI is achieved by spreading the signal uniformly over the frequency band [10]. This spreading causes two possible problems for a potentially unwanted listener. First, a larger frequency band must be monitored and secondly, if Direct Sequence is employed, the signal is of very low power. Hence, the spread signal is difficult to intercept, and will most likely be indistinguishable from noise. Even if

the signal is somewhat detectable, the listener still requires the correct despreading code in order to extract any useful information. It should be noted that since the probability of intercept is low, the probability of locating the source or identifying it is also minute. Multiple access is also possible due to the properties of the spreading/despreading sequences. These codes have very low cross correlation. Ideally, in a communication system, one desires no cross correlation. However, there are a finite number of orthogonal signal sets which limits the number of users. There are several ways of generating finite length sequences with low cross correlational properties [7, p. 92], [9, Chap. 5], [11, Chap. 8]. Thus, in theory, several spread spectrum systems may be simultaneously operated over the same bandwidth.

2.5 Spread Spectrum Techniques

In this section, the techniques of Direct Sequence (DS), Frequency Hopping (FH), and Hybrid (DS/FH) spread spectrum will be explored. The basic signal characteristics of modern spread spectrum systems are as follows [10]:

- i) The carrier is an unpredictable or pseudorandom wide band signal.
- ii) The bandwidth of the carrier is much wider than the bandwidth of the data modulation.
- iii) Reception is accomplished by cross correlation of the received wide band signal with either a synchronously generated replica of the wide band carrier, or asynchronously through the use of a matched filter. Here, the term pseudorandom describes a signal which is random in appearance but reproducible by deterministic means.

2.5.1 Direct Sequence Direct Sequence is probably the most common implementation of spread spectrum communication systems today. A typical configuration is the stored reference model. The spreading code in the transmitter spreads the data over a wide bandwidth, which is then transmitted on a given channel. A brief discussion of Direct Sequence modulation will now follow.

For simplicity, consider a pseudorandom sequence (PNS), as shown in Figure 2.5b, spreading a binary sequence, Figure 2.5a. The waveform shown in Figure 2.5c is the result of a modulo-2 addition of the baseband BPSK signal and pseudorandom sequence, and in turn directly modulates an appropriate carrier for transmission over the channel. At the receiver, carrier and bit synchronization occurs, and a modulo-2 addition of the incoming signal and the identical PNS is performed. The integration of the product of the desired signal with the stored reference produces a correspondingly large value (correlation). The integration of unwanted signals multiplied by the stored reference yields a small value (partial correlation). For example, if a jamming signal is present at the receiver (Figure 2.6), it will be spread while the desired signal is being despread as in Figure 2.7. As a result, the jamming signal may be rejected, while the desired is extracted.

The Direct Sequence Code Division Multiple Access (DS-CDMA) technique allows for multiple users on the same communication channel to transmit simultaneously. Each user simply uses a different PN sequence. The code properties are such that there is very low partial cross correlation between different sequences. At the receiver, the reception of

Figure 2.5 A simplified example of direct sequence spreading

Figure 2.6 Received signal plus jamming

Figure 2.7 Rejecting jamming by despreading

undesired codes may simply be modelled as noise. It should be noted that other baseband modulation techniques include Quadrature Phase Shift Keying (QPSK), staggered QPSK (SQPSK), and Minimum Shift Keying (MSK). With QPSK, the PNS is used to modulate the output of sine and cosine carriers of the same frequency. The two carriers are summed and the resulting waveform has equally likely phase values of 45° , 135° , 225° , and 315° . Staggered QPSK staggers the PNS in one channel by half a chip. Thus, the phase may change every half chip, but the magnitude of this change is limited to 90° [3, p.47]. MSK offers sidelobe suppression and fast sidelobe rolloff, and a null to null bandwidth of 1.5 times the chip rate [8, p. 28].

2.5.2 Frequency Hopping Frequency Hopping (FH) is another common spread spectrum technique. Unlike Direct Sequence, FH is actually a narrow band transmission. However, the carrier frequency is altered or 'hopped' several times per second. In this way, as shown in Figure 2.8, the transmission is effectively spread across a wider bandwidth. In simple terms, Frequency Hopping is simply Frequency Shift Keying (FSK) with a wider set of frequencies to select from. Ideally, each frequency, or cell, should be occupied with equal probability and furthermore, the transition probabilities from one cell to another should be identical [3, p.27]. Two methods of hopping exist: fast frequency hopping (FFH) and slow frequency hopping (SFH). In FFH, the carrier hops multiple times per symbol. Conversely, there are several symbols per carrier hop in SFH [7, p. 154]. Obviously, FFH would be more secure, but has a higher complexity.

Figure 2.8 A typical frequency hopping time-frequency map

The Low Probability of Intercept and Antijamming features are readily evident. An interceptor must monitor a wide bandwidth, but requires a narrowband filter for signal detection. Thus, unless the interceptor has a priori knowledge of the hopping pattern, the signal to noise ratio (SNR) in intercept receiver will be much too low for reliable detection. Without a priori knowledge of the hopping pattern, jamming is also difficult. The jamming signal must simultaneously occupy the identical narrow frequency band as the signal being transmitted. The data signal is hopped several times per second, further making the jammer's task more arduous. Hence, effective disruption or exploitation of communication service would require virtually one hundred percent coverage of a wide frequency band; a practical impossibility for a remote, ground based jammer. As was the case with Direct Sequence, any signal which was not spread with the replica code in the receiver will be rejected through spreading via multiplication.

Figure 2.9 shows a typical Frequency Hopping System architecture. Ideally, the synthesizer output is a single instantaneous frequency. Practically, the output is a composite spectrum of the desired frequencies, hopping sidebands, and spurious by-products [8, p. 29]. Transmission is typically non-coherent and the ideal spectrum has a uniform distribution. The probability of any frequency slot being occupied is identical. Thus, the system bandwidth is simply the number of different carriers or frequencies multiplied by the channel separation. It should be noted that although processing gain (G_p) may be determined using equation 2.1, a better measure [5] is simply the number of available frequency choices (N), i.e.:

Figure 2.9 Frequency hopping system architecture

$$G_p = N \quad (2.4)$$

The determination of hopping rate and number of frequencies used is a function of three parameters [8, p. 34]: the type and rate of transmitted data, the amount of redundancy used, and the distance to potential interference. Although data is generally sent in any non-coherent form, M-FSK is typically used. The number of frequencies or slots is dependent on an acceptable error rate for the system. Finally, the jamming power received will be dependent upon the distance to the jammer.

2.5.3 Hybrid Systems A single spread spectrum technique may not be fully adequate for a particular application. Hybrid systems combine the desirable attributes of, usually two, different spread spectrum techniques. For example [3, p.71], protection against a particular or several different types of jamming may be most economically achieved using a hybrid approach. Frequently, a mixture of methods may reduce overall cost and complexity for a given capability or specification. It may be easier to design a system which hops away from a strong, narrowband jammer as opposed to designing a Direct Sequence receiver with enough processing gain to suppress this. Finally, hybrid systems may also facilitate interoperability and the networking of radios.

As an illustration of reduced complexity, consider a system which is designed to operate over 500 MHz of bandwidth. Using Direct Sequence, a chipping rate of at least 250 Mb/sec is required. If Frequency Hopping is employed, and assuming a 10 KHz channel spacing, 50

000 different frequencies would be required. Conversely, a hybrid system would meet the requirements using a 50 Mb/sec chipping code and ten different frequencies with 50 MHz separation.

The most common configurations of hybrid systems are Frequency Hopping-Direct Sequence (FH-DS), Time and Frequency Hopping (TH-FH), and Time Hopping coupled with Direct Sequence (TH-DS). The former is more common and will be discussed in more detail. The latter two are not as prevalent. TH-FH is usually employed when a large number of users with widely variable distances or transmitter powers are operating simultaneously on a single link. TH-DS may be used in a CDMA environment when the cross correlational properties of the spreading codes are not sufficient. Time Hopping is added to the system to create extra channels via time division multiplexing [8, pp. 42-44].

A conventional FH-DS system is depicted in Figure 2.10. The data is first spread by Direct Sequence. The same PNS is then used to hop the carrier frequency periodically. A frequency domain representation is given in Figure 2.11. As expected, the DS spreading gives a sinc^2 spectrum. This spectrum is then hopped across the entire frequency band pseudorandomly. Upon reception, the signal is first synchronized, and subsequently dehopped and despread.

The overall processing gain of a hybrid system is a function of the spread bandwidth over the data bandwidth. However, when viewing a particular signal, the jamming margin as given

Figure 2.10 A frequency hopped-direct sequence hybrid system

Figure 2.11 Hybrid signal spectrum (logarithmic scale)

in equation 2.3 is not entirely accurate [8, p. 50]. This is due to the fact that in any segment of the overall input spectrum, the receiver can only reject the interference/jamming which is present in the Direct Sequence partial band. For example, suppose the system bandwidth is 1 GHz, with a DS bandwidth of 100 MHz. The RF bandwidth used when calculating the processing gain is therefore 100 MHz, not 1 GHz. Dixon [8, p. 51], gives the processing gain of a hybrid system as a sum of the processing gains due to each technique, i.e.

$$G_p = 10 \log_{10}(\text{no. of channels}) + 10 \log_{10} \left(\frac{W}{B} \right) . \quad (2.5)$$

A FH-DS configuration combats interference in two ways. First, the random hopping in frequency avoids potential interference and/or jamming part of the time. Secondly, should the signal hop into interference, the undesired signal is spread due to the Direct Sequence technique. Hence, the baseband signal may be extracted by filtering as explained previously. The performance of a hybrid system in the presence of partial band interference does not differ substantially from that of a conventional Frequency-Hopping system, due to the wideband filter at the front end of the receiver. Interference which would normally be filtered in a conventional frequency hopping system is allowed to pass in a hybrid system, as explained above. Conversely, in the presence of multiple access interference, a hybrid system will demonstrate a performance advantage relative to a Direct Sequence system where a near-far problem exists [7, p.218].

2.6 Pseudorandom Noise Sequences

Pseudorandom Noise Sequences (PNS), or PN codes, are the backbone upon which spread spectrum is designed. The length, type, and chip rate of the code used set the bounds on system performance. Ideally, one desires a purely random code to spread the data and provide communication security. Unfortunately, this is not realizable as synchronization at the receiver requires the generation of a replica code. As such, deterministic PN codes are employed.

PN codes possess the following properties [5]:

- i) They are easy to generate
- ii) They have randomness properties
- iii) They have long periods
- iv) They are difficult to reconstruct from short sequences

In addition, PN codes possess other desirable characteristics [8, pp. 58-78]. As discussed previously, PN codes provide interference rejection capability and reduce the effects of noise. This is accomplished via processing gain. Secondly, PN codes provide security, as any potential eavesdropper requires a replica of the code in order to demodulate the signal.

In practice, linear feedback shift registers (LFSR) are used for code generation, as they can physically realize properties i, ii, and iii, but not property iv. Several classes of codes exist,

e.g., Gold, Kasami, and Bent. Statistical code properties such as autocorrelation and cross correlation will be examined, and will demonstrate the suitability of spread spectrum as a multiple access scheme. The actual method of code generation will be dealt with initially.

2.6.1 PN Code Generation Probably the easiest physical code realization is through the use of a simple shift register generator (SSRG), as depicted in Figure 2.12. Each box represents a binary element (1 or 0), which is sequentially shifted to the right after each clock pulse. Some or all of the binary elements are linearly combined and fed back to the first stage. The code sequence, C_n , may therefore be mathematically represented using the following recursive formula:

$$C_n = \sum_{k=1}^r a_k C_{n-k} \pmod{2}, \quad a_r = 1. \quad (2.6)$$

The code period is dependent upon the initial state, that is the initial contents of each coefficient of the shift register (a_k 's), and on which elements are tapped or fed back. One need not go through tedious calculations to obtain a desired sequence. Tables exist [12, pp. 62-65] which give characteristic polynomials for any number of shift registers.

2.6.2 Maximal Length Sequences Spread spectrum systems utilize maximal length sequences, that is sequences of period $N = 2^m - 1$, with the all zero state being invalid. Maximal length (m) sequences possess the following properties [2], [5], [8, p.60]:

i) in one code period there exists 2^{m-1} ones and $2^{m-1}-1$ zeros

Figure 2.12 Code generation using a ssrg

- ii) the modulo-2 addition of an m sequence with a phase shifted version of itself results in a different phase of the same sequence.
- iii) in any code period, half the runs of consecutive ones or zeros are of length one, one quarter are of length two, one eighth are of length three, etc.
- iv) an m-sequence has a periodic autocorrelation function given by:

$$R_c(\tau) = \begin{cases} 1 & \tau = 0, N, 2N \dots \\ -\frac{1}{N} & \text{otherwise} \end{cases} \quad (2.7)$$

and is illustrated in Figure 2.13.

The power spectrum of the m-sequence is simply the Fourier transform of the autocorrelation function, and is easily derived [11, p.387]:

$$S_c(f) = \begin{cases} \frac{N+1}{N^2} \text{sinc}^2\left(\frac{\pi m}{N}\right) \delta(f - mf_c) & m \neq 0, f_0 = \frac{1}{NT_c} \\ \frac{1}{N^2} & m = 0 \end{cases} \quad (2.8)$$

This yields the power spectrum depicted in Figure 2.14. Chapter III gives a detailed derivation of the autocorrelation and power spectrum functions.

The sequences most often used for spread spectrum purposes include m-sequences and Gold Codes [13]. Both Kasami [14] and Bent [15] sequences are used less commonly.

Figure 2.13 Autocorrelation function of an m-sequence

Figure 2.14 Power spectrum of an m-sequence

2.6.3 Additional Useful Properties In addition to the aforementioned properties, m-sequences contain additional desirable characteristics including low partial correlation over a window smaller than the full period, low cross correlation among different codes, and both even and odd cross correlation functions [5]. The first is necessary to keep the self noise of the system as low as possible, as the integration time is much less than a code period. The latter two properties allow for multiple access schemes. Interference is reduced through the code properties, hence enabling multiple simultaneous use on the same channel.

2.7 Exploitation Of Low Probability Of Intercept Signals

Although Direct Sequence Spread Spectrum signals are transmitted well below the ambient noise floor of both the target and intercept receiver, there are methods of detecting not only the presence, but also features of such a signal. Features may be defined as any information, independent of the data, recovered from an intercepted signal. Examples include data rate, chip rate, carrier frequency, and type of modulation. Some of the more popular methods of exploiting a spread spectrum signal include wide band radiometry, carrier detection, and chip-rate detection. All of the methods of exploitation which will be discussed involve the performance of a non-linear function on the received signal followed by some kind of spectral processing. Radiometry and carrier detection will be explained briefly, followed by a more detailed description and analysis of chip rate detectors. The focus will be on BPSK modulation, with some other modulation schemes illustrated where appropriate.

2.7.1 The Wide Band Radiometer Probably the most common method of signal interception is wide band radiometry, depicted in Figure 2.15. Ideally, the device detects arbitrarily weak signals by integrating received energy over an arbitrary period of time. However, the radiometer is particularly susceptible to background noise fluctuations. These fluctuations may be attributed to such phenomena as temperature drift in the receiver, changes in external noise due to antenna pointing, or fluctuations in the intensity of interfering signals [16]. In particular, a fluctuating background may result in an erroneous threshold setting. As a result, the probability of detection, and hence reliability, decreases. This is not to say, however, that the radiometer is not a useful tool. Recent studies [17] have shown that the wide band radiometer is still the most sensitive device for detecting signal presence. It should be noted, however, that a radiometer cannot extract features from an intercepted signal, but merely indicate its presence or absence. The expected value of the radiometer output over time T , assuming band-limited white Gaussian noise with two-sided power spectral density $N_0/2$ and $n(t)$ is a zero mean process, is [7, p.229]:

$$E[V] = \int_0^T s^2(t) dt + \int_0^T E[n^2(t)] dt \quad (2.9)$$

$$E = N_0 TW$$

where W is the bandwidth of the bandpass filter.

Now, writing both $s(t)$ and $n(t)$ in terms of the in phase and quadrature phase components,

Figure 2.15 Radiometer operation

$$s(t) = s_c(t)\cos(\omega_c t) - s_s(t)\sin(\omega_c t) \quad (2.9a)$$

$$n(t) = n_c(t)\cos(\omega_c t) - n_s(t)\sin(\omega_c t) \quad (2.9b)$$

and noting that

$$s_c(t) = \sum_{i=-\infty}^{\infty} s_c\left(\frac{i}{W}\right) \text{sinc}(Wt - i) \quad (2.10)$$

the following may be used to describe the signal entering the comparator:

$$V = \frac{1}{2W} \sum_{i=1}^{\alpha} \left[s_c\left(\frac{i}{W}\right) + n_c\left(\frac{i}{W}\right) \right]^2 + \frac{1}{2W} \sum_{i=1}^{\alpha} \left[s_s\left(\frac{i}{W}\right) + n_s\left(\frac{i}{W}\right) \right]^2 \quad (2.11)$$

where α is the largest integer less than or equal to TW .

2.7.2 Carrier Detection Carrier detection also employs a squaring nonlinearity. As shown (Figure 2.16), the signal is filtered after squaring. The spectral component of interest is located at twice the carrier frequency, and depending upon the modulation scheme, other spectral components may arise.

Mathematically, carrier detection occurs as follows. Ignoring noise in the ideal case, the received signal $r(t)$ consists of the transmitted baseband signal $s(t)$ modulating a carrier at ω_c . The output of the squaring circuit may be denoted as $y(t)$, where $y(t)$ is $s(t)s(t-d)$, which may

Figure 2.16 Carrier detection operation

be rewritten as [18]:

$$y(t) = \frac{1}{2} \operatorname{Re} \left[s(t) s^*(t-d) e^{j\omega_c d} \right] \quad (2.12)$$

$$+ \frac{1}{2} \operatorname{Re} \left[s(t) s^*(t-d) e^{j\omega_c (2t-d)} \right].$$

As shown, the squared signal consists of two separate and discrete spectral components; the first at dc and the latter around twice the carrier frequency. If the carrier frequency is large compared to the clock frequency, the two spectra will not overlap. The power spectral density of the carrier component (γ_c) is denoted by

$$\gamma_c(f) = \frac{1}{16} [\gamma_b(f-2f_c) + \gamma_b(f+2f_c)] \quad (2.13)$$

γ_b is, of course, the power spectrum of the baseband data. It is widely known that certain modulation schemes (QPSK) will not give a spectral component at $2f_c$, and a fourth power nonlinearity is necessary. However, BPSK modulation gives rise to a spectral line at $2f_c$ (and $2f_c \pm 1/T_c$). These spectral lines are maximized for $d=0$ [19]. The magnitude of the spectral lines will depend upon both the type of filtering and time-bandwidth product. Laboratory experimentation has demonstrated that practical carrier detection may be achieved within about 1 dB of theory [18].

2.7.3 Chip-Rate Detection Chip-rate detectors (CRD) may be found in the literature under several different names, including the symbol rate detector [16], the delay line clock

regenerator (DLCR) [23], or the prefilter delay and multiply device (PFDM) [19]. In any event, the intent of such a device is to produce a spectral component such that the chip rate of the intercepted signal may be determined. This information may then be used to identify the origin of the signal, or perhaps as one step in the partial or full demodulation of the information carried within.

The fact that Direct-Sequence Spread Spectrum signals have a pseudocontinuous spectrum makes them difficult to detect by conventional means such as the Fast Fourier Transform (FFT). As shown with wide-band radiometry and carrier detection above, performing a nonlinear operation on the signal of interest results in the emergence of discrete spectral components. This is also true of chip-rate detection. Here, the operation consists of a delay and multiply circuit, depicted in Figure 2.17. The output $o(t)$ consists of $\text{signal} \bullet \text{signal}$, $\text{signal} \bullet \text{noise}$, and $\text{noise} \bullet \text{noise}$ terms, resulting from the following operation:

$$o(t) = [s(t) + n(t)] \cdot [s(t-d) + n(t-d)] . \quad (2.14)$$

It should be noted that the noise is modelled as complex Gaussian [21] [22], such that

$$n(t) = n_r(t) + jn_i(t) . \quad (2.15)$$

It is also assumed that the interceptor has a priori knowledge of the carrier frequency such that the baseband representation of $s(t)$ may be used:

Figure 2.17 Chip-rate detection

$$s(t) = \sum_n a_n p(t-d-nT) . \quad (2.16)$$

Note that the a_n 's may take values of ± 1 , and that the message sequence $\{a_n\}$ has zero mean and is composed of independent and identically distributed pseudorandom elements. As a result, the output may be described by [20]:

$$\begin{aligned} o(t) = & (s(t) * h(t)) (s'(t-d) * h'(t-d)) \\ & + (s(t) * h(t)) (n'(t-d) * h'(t-d)) \\ & + (n(t) * h(t)) (s'(t-d) * h'(t-d)) \\ & + (n(t) * h(t)) (n'(t-d) * h'(t-d)) \end{aligned} \quad (2.17)$$

where $h(t)$ is the impulse response of the filter preceding the delay and multiply device.

2.7.3.1 CRD Figure of Merit In spread spectrum applications, the received signal is typically weak. Since noise predominates, the cross terms may be discarded [19] [20] such that the output signal strength consists of the signal component only (the first term of 2.17) and the output noise strength consists of the noise only term (the last term of 2.17). The output signal component will contain spectral lines at the chip rate and its harmonics ($\omega = k/T_c$, $k = \pm 1, \pm 2, \dots$), and the time averaged power of one of these lines is considered to be the output signal strength [20]. i.e.:

$$S_{ss} \triangleq \left| \left\langle [s(t) * h(t)][s'(t-d) * h'(t-d)] e^{-j2\pi\omega t} \right\rangle \right|^2 . \quad (2.18)$$

In the same manner, the output noise power is defined over a small bandwidth B_n centered at α , and time averaged:

$$S_{nn}(\alpha) \triangleq \int_{\alpha - B_n/2}^{\alpha + B_n/2} S_n(f) df \quad (2.19)$$

where $s_n(t)$ is

$$s_n(t) \triangleq \left| \left\langle [n(t) * h(t)][n(t-d) * h(t-d)] e^{j2\pi\alpha t} \right\rangle \right|^2 \quad (2.20)$$

The output signal to noise ratio (SNR) hence consists of two terms: the amplitude squared of the k^{th} coefficient of the Fourier series expansion about α , and the amplitude of the power spectral density of $o(t)$ in the band centered at α . Spooner and Gardner [20] derived the output SNR to therefore be

$$SNR = \frac{\frac{\sigma_a^4}{T_c^2} \left| \int_{-\infty}^{\infty} H(f+\alpha/2) H^*(f-\alpha/2) S_{nn}(f+\alpha/2) S_{nn}^*(f-\alpha/2) e^{j2\pi f d} df \right|^2}{4B_n \int_{-\infty}^{\infty} \cos^2(2\pi f d) |H(f+\alpha/2) H^*(f-\alpha/2)|^2 S_{nn}(f+\alpha/2) S_{nn}(f-\alpha/2) df} \quad (2.21)$$

where σ_a^2 is

$$\sigma_a^2 \triangleq \lim_{K \rightarrow \infty} \frac{1}{2K+1} \sum_{k=-K}^K |a_k|^2 \quad (2.22)$$

Kuehls and Geraniotis [19] have arrived at a similar result and have demonstrated that, with no bandpass filtering, a maximum occurs when the delay is $T_c/2$, that is, one half of a chip period [18] [22].

2.7.3.2 CRD Performance Analysis There have been numerous theoretical examinations of performance of chip rate detectors [18]-[20], [23],[24]. As mentioned previously, output SNR is maximized with a half chip delay and with no bandpass filtering. In addition, the bandwidth of the receiving filter must be slightly larger than the Nyquist bandwidth for best results [18]. Spooner and Gardner [20] have demonstrated, through simulation, that the optimum parameters are a delay of $0.52 T_c$, and a filter bandwidth of $1.4 / T_c$. This was, however, with a priori knowledge of the chip rate. One would expect performance to degrade significantly in the absence of this information, as the filter bandwidth would be substantially larger.

Kuehls and Geraniotis [19] demonstrated some interesting results. For known bit rates and input SNR's much less than unity, observation time increased by a factor of four for every 3 dB decrease in input SNR. Detection performance is therefore very sensitive to input SNR. Furthermore, the optimal prefilter was one matched to the pulse shape. Their results clearly illustrate that in the case of unknown bit rates and BPSK modulation, the performance using a half chip delay could not exceed that of a squarer. In the case of unknown bit rates, using an FFT for detecting spectral lines does not significantly reduce detection, and actually offers a measure of robustness.

Reed [16] has shown that a radiometer will outperform a CRD in a stationary environment, simply due to the fact that the CRD is sensitive to bandwidth mismatch. Conversely, if

interference fluctuation is present (i.e. a non-stationary environment), the CRD will outperform the radiometer. This comes as no surprise, as radiometers are susceptible to background noise fluctuations. However, Gardner and Spooner [24] have demonstrated that cyclic feature detectors can outperform the radiometer in some instances where the background noise is stationary.

Very little practical experimentation has been published. Hill and Felstead [17] have reported that CRD's perform about 0.7dB better than theory. This, however, may have been due to a slightly erroneous value of detector sensitivity obtained from previous work in this area [24] [25]. Myers [21] has demonstrated that practical performance is achievable to within 2.2 dB of theory.

CHAPTER III

3.0 A THEORETICAL DEVELOPMENT OF CHIP-RATE HOPPING

3.1 Introduction

This chapter will describe the theory behind the design of a Direct Sequence Spread Spectrum communication system to suppress the chip rate feature. Specifically, the novel technique of chip-rate hopping (CRH) will be presented. A matched filtering scheme will be developed for demodulation.

As seen in the previous chapter, the performance of chip-rate detectors is sensitive to bandwidth mismatches. In order to suppress the chip-rate feature, one must therefore exploit this bandwidth sensitivity. The most obvious solution is to change the chip rate periodically. This may be accomplished in several ways. First, one could step through a predetermined set of frequencies during the transmission time. The drawback of this approach is that should a few transmissions be intercepted, an interceptor could easily predict the sequence of chip rates and thus modify the intercept system to exploit this. A second method could involve changing the chip rate in a random manner. This would avoid the problem of chip-rate prediction by an interceptor. Here, the problem is one of synchronization. If the change in chip rates was truly random, a communicator would have a tremendous problem

synchronizing to the transmission at the receiver. If synchronization were possible at all, the circuitry would most likely be overly complex.

Thus far, two potential problems have been described: chip-rate prediction and synchronization. One could overcome both problems by changing the chip rate in a pseudorandom fashion. In addition to the m-sequence used to spread the data, a second m-sequence could be employed to change the chip rate. This is entirely analogous to Frequency-Hopping except it is the chip rate which is changing as opposed to the carrier frequency; hence the term Chip-Rate Hopping. Unless an interceptor knows which m-sequence is being employed, reliable chip-rate prediction is difficult. Since the communicator knows the m-sequence a priori, synchronization may be achieved.

A final way to suppress the chip-rate feature would include a burst transmission scheme. As a result, an interceptor would have to be tuned to the correct frequency at the proper time in order to intercept a transmission. This could be employed in a couple of ways. A communicator could simply transmit spread data only at different times; that is, transmit spread data, wait for some arbitrary amount of time, then transmit spread data again at the next chip rate. The primary disadvantage of this scheme is that should an intercept occur, the m-sequence being used to spread the data could be easily reconstructed. A more covert method would be to use a third m-sequence which is transmitted when data is not present. This m-sequence would have to be sufficiently lengthy such that it does not repeat over the transmission time frame. This would ensure that even if intercepted, reconstruction of the

m-sequence would be difficult. Furthermore, the time, or number of chip rates, between data bursts would have to be sufficiently large as to present a low probability of intercept.

Synchronization and demodulation of a chip-rate-hopped signal may be accomplished through the use of a matched filter. The physical implementation could be one of three possible schemes. First, a bank of matched filters, one for each chip rate, could be employed. A decision would be made on the filter which gives the greatest output. Secondly, a single matched filter whose coefficients are changed dynamically such that one set of coefficients corresponds to a particular chip rate could be used. The result would be a different set of coefficient values being loaded into the filter upon a change in chip rate. A variation on this would involve dynamically changing the sampling rate, while using one set of coefficients. The primary advantage of this method would be a reduction in complexity.

In summary, a scheme has been proposed which has both a low probability of intercept (LPI) and a low probability of exploitation (LPE). It is LPI in that the chip rate changes in a pseudorandom manner, making prediction of the next chip rate difficult. In addition, employing a burst transmission scheme reduces the probability that a data burst will be intercepted. It is LPE in that even if a burst is intercepted, there is a high probability that it does not contain data. This being the case, reconstruction of the m-sequence is difficult. Finally, demodulation may be accomplished via a dynamic matched filter.

3.2 Conventional Autocorrelation and Power Spectrum

Both the autocorrelation function and the power spectrum of m-sequences were briefly examined in Section 2.6.2. This theory will now be examined in more detail and then extended to the case of chip-rate hopping. Most of the following is condensed from [11, pp. 720-727], unless otherwise noted.

Let the spreading code $c(t)$ be a maximal length sequence of period NT_c and length $N = 2^m - 1$ and let $c(t)$ be defined as:

$$c(t) = \sum_{n=0}^{N-1} a_n p(t - nT_c) \quad (3.1)$$

where

$$a_n = \pm 1 \quad (3.2a)$$

and

$$p(t) = \begin{cases} 1 & 0 \leq t < T_c \\ 0 & \text{otherwise.} \end{cases} \quad (3.2b)$$

The continuous autocorrelation function is therefore given by

$$R_c(\tau) = \frac{1}{T} \int_0^T c(t)c(t+\tau) dt. \quad (3.3)$$

Now consider a second m-sequence of the same period given by

$$d(t) = \sum_m b_m p(t - mT_c) \quad (3.4)$$

with

$$b_m = \pm 1 \quad (3.4a)$$

and $p(t)$ as described in (3.2b). The continuous cross correlation of $c(t)$ and $d(t)$ is therefore

$$R_{cd}(\tau) = \frac{1}{T} \int_0^T d(t)c(t+\tau) dt \quad (3.5)$$

Substitution of (3.1) and (3.4) into (3.5) gives

$$R_{cd}(\tau) = \sum_m \sum_n b_m a_n \int_0^T p(t - mT_c) p(t + \tau - nT_c) dt \quad (3.6)$$

This is non-zero only when $p(t - mT_c)$ and $p(t + \tau - nT_c)$ overlap. Let τ be defined as

$$\tau = kT_c + \tau_c \quad (3.7)$$

Hence, overlap occurs only when $n=k+m$ and $n=k+m+1$. Thus, (3.6) may be rewritten as

$$R_{cd}(\tau) = \frac{1}{N} \sum_{m=0}^{N-1} d_m C_{k+m} \frac{1}{T_c} \int_0^{T_c - \tau_c} p(t - mT_c) p(t - mT_c + \tau_c) dt$$

$$+ \frac{1}{N} \sum_{m=0}^{N-1} d_m C_{k+m-1} - \frac{1}{T_c} \int_{T_c - \tau_c}^T p(t - mT_c) p(t - mT_c - T_c + \tau_c) dt \quad (3.8)$$

Letting $\gamma = t - mT_c$, (3.8) is rewritten as

$$R_{cd}(\tau) = \frac{1}{N} \sum_{m=0}^{N-1} d_m C_{k+m-1} - \frac{1}{T_c} \int_0^{T_c - \tau_c} p(\gamma) p(\gamma + \tau_c) d\gamma$$

$$+ \frac{1}{N} \sum_{m=0}^{N-1} d_m C_{k+m-1} - \frac{1}{T_c} \int_{T_c - \tau_c}^T p(\gamma) p(\gamma - T_c + \tau_c) d\gamma \quad (3.9)$$

Now, the discrete cross-correlation is given by

$$\theta_{cd}(k) = \frac{1}{N} \sum_{n=0}^{N-1} a_n b_{n+k} \quad (3.10)$$

hence

$$R_{cd}(k, \tau_c) = \left(1 - \frac{\tau_c}{T_c}\right) \theta_{cd}(k) + \frac{\tau_c}{T_c} \theta_{cd}(k+1) \quad (3.11)$$

The discrete autocorrelation is therefore

$$R_i(\tau) = (1 - \frac{\tau_i}{T_i}) \theta_i(k) + \frac{\tau_i}{T_i} \theta_i(k+1) . \quad (3.12)$$

Now,

$$\theta_i(k) = \begin{cases} 1 & k = lN, l \text{ any integer} \\ 0 & \text{otherwise} . \end{cases} \quad (3.13)$$

Thus, there are 3 possibilities for the discrete autocorrelation of an m-sequence:

$$R_i(\tau) = \begin{cases} 1 - \frac{\tau}{T_i} \left(1 + \frac{1}{N} \right) & 0 \leq \tau < T_i \\ \frac{1}{N} & T_i \leq \tau < (N-1)T_i \\ \frac{\tau_i}{T_i} \left(1 + \frac{1}{N} \right) - \frac{1}{N} & (N-1)T_i \leq \tau < NT_i \end{cases} \quad (3.14)$$

where τ_i may be defined as

$$\tau_i = \tau - \frac{(N-1)T_i}{N} \quad 0 \leq \tau_i < T_i . \quad (3.15)$$

The discrete autocorrelation function may therefore be completely specified as

$$R_i(\tau) = \frac{1}{N} + \frac{N+1}{N} \sum_{i=1}^N \Lambda \left(\frac{\tau - iNT_i}{T_i} \right) \quad (3.16)$$

where

$$\Lambda(\tau) = \begin{cases} 1 - 2|\tau|, & |\tau| \leq \frac{1}{2} \\ 0 & \text{otherwise} . \end{cases} \quad (3.16a)$$

By definition, the power spectrum is

$$S_s(f) = \mathcal{F}[|R_s(\tau)|^2] . \quad (3.17)$$

Hence,

$$S_s(f) = \begin{cases} \frac{N+1}{N^2} \operatorname{sinc}^2\left(\frac{\pi m}{N}\right) \delta(f - mf_0) & m \neq 0, f_0 = \frac{1}{NT_s} \\ \frac{1}{N^2} & m = 0 . \end{cases} \quad (3.18)$$

Both the autocorrelation and power spectrum were depicted in Figures 2.16 and 2.17 respectively.

3.3 Autocorrelation and Power Spectrum of Chip-Rate Hopping

Chip-rate hopping has been defined to be the transmission of a DSSS signal whose chip rate is determined by a second different m-sequence. This second sequence may be any valid m-sequence.

Consider an m-sequence $m(t)$ of length $L = 2^l - 1$, defining the chip rate of a second m-sequence $c(t)$ of length $N = 2^m - 1$. Further, let L denote the number of chip rates available over the frequency band of interest (B_h), with B_l and B_u corresponding to the lower and upper frequencies in the hop band respectively, i.e.

$$B_h = B_u - B_l . \quad (3.19)$$

In addition, the number of chip rates is real and they are uniformly distributed over $[B_l, B_u]$. Thus, the probability density function, $f(f_i)$, of the available chip rates is given by [26, p.101]

$$f(f_i) = \begin{cases} \frac{1}{B_u - B_l} & B_l \leq f_i \leq B_u \\ 0 & \text{otherwise} \end{cases} \quad (3.20)$$

with mean

$$\mu = \frac{B_l + B_u}{2} \quad (3.21)$$

and variance

$$\sigma^2 = \frac{(B_u - B_l)^2}{12} \quad (3.22)$$

Let the chip-rate spacing, Z , be defined

$$Z = \frac{B_u - B_l}{L} \quad (3.23)$$

Modification of (3.14) by summing over all chip rates gives the autocorrelation function:

$$R_{c,m}(\tau) = \begin{cases} \sum_{l=1}^{\frac{1}{B_h}} \frac{1 - \frac{\tau}{T_c} \left(1 + \frac{1}{N}\right)}{L} & 0 \leq \tau < T_c, \Delta T_c \leq \frac{1}{Z} \\ -\frac{1}{N} & \frac{1}{B_h} \leq T_c < \frac{1}{B_l}, T_c \leq \tau < (N+1)T_c \\ \sum_{l=1}^{\frac{1}{B_l}} \frac{\frac{\tau}{T_c} \left(1 + \frac{1}{N}\right)}{L} - \frac{1}{N} & (N+1)T_c \leq \tau < NT_c, \Delta T_c \leq \frac{1}{Z} \end{cases} \quad (3.24)$$

with τ_c as defined in (3.15). Thus, the complete autocorrelation function may be completely specified by:

$$R_{c,m}(\tau) = \frac{1}{N} + \frac{N+1}{N} \sum_{l=1}^{\frac{B_l}{B_h}} \sum_{l'=1}^{\frac{B_l}{B_h}} \Lambda \left(\frac{\tau - iNT_c}{T_c} \right) \quad (3.25)$$

The complete power spectrum of chip-rate hopping is therefore given by the Fourier transform,

$$S_{c,m}(f) = \begin{cases} \sum_{l=1}^{\frac{B_l}{N}} \frac{\left(\frac{N+1}{N^2} \operatorname{sinc}^2 \left(\frac{\pi m}{N} \right) \delta(f - mf_0) \right)}{L} & m \neq 0, \Delta f_0 \leq Z \\ \frac{1}{N^2} & m = 0 \end{cases} \quad (3.26)$$

3.3.1 A Specific Example of Chip-Rate Hopping Having defined both the autocorrelation and power spectrum functions of chip-rate hopping, a specific example will now be examined for the purpose of illustration.

Consider a Binary Phase Shift Keyed baseband signal modulating an m-sequence of length $N=1023$. Further, a second m-sequence of length $L=255$ is used to map $L=255$ different chip rates between 9 and 10 MHz. Thus,

$$\begin{aligned} N &= 1023 \\ L &= 255 \\ B_h &= 1 \text{ MHz} \\ B_l &= 9 \text{ MHz} \\ B_u &= 10 \text{ MHz} \\ Z &= \frac{1 \text{ MHz}}{255} \approx 3922 \text{ Hz} \\ 1 \cdot 10^{-7} \text{ sec.} &\leq T_c < 1.11 \cdot 10^{-7} \text{ sec.} \end{aligned}$$

The net effect, as depicted in Figure 3.1, is to smooth the autocorrelation function. Shown are the results of the autocorrelation function of chip-rate hopping between 9 and 10 MHz, with the autocorrelation function of a single chip rate at 9.5 MHz depicted for comparison purposes. The transition to the value of $1/N$ is no longer sharp at $\tau = NT_c$, as is the case with a single chip rate. Figure 3.2 shows the resulting power spectrum and serves to demonstrate that the sharp nulls one would expect at multiples of the chip rate have been suppressed considerably. In this case, simulation shows a 32 dB difference as compared to the power spectrum of a single chip rate at 9.5 MHz.

3.4 Conventional Matched Filtering

Matched filtering involves the passive correlation of the transmitted data with a stored reference; a finite length of received signal $s(t)$ is convolved with the complex conjugate of

Figure 3.1 Comparison of the autocorrelation functions of chip-rate hopping (9-10 MHz) with that of a single chip rate at 9.5 MHz

Figure 3.2 Comparison of the power spectra of chip-rate hopping (9-10 MHz) with that of a single chip rate at 9.5 MHz

the spreading code $c(t)$. The result is then tested against a predetermined threshold and a decision is made with regards to presence and polarity of the received signal. In practice, a matched filter is implemented as a tapped delay line followed by a passive filter matched to a single PN chip waveform [9, pp. 67-84]. The required impulse response of this filter, matched to a finite length signal $s(t)$ of duration T_u is therefore

$$h(t) = \begin{cases} s(T_u - t) & 0 \leq t < T_u \\ 0 & \text{otherwise} \end{cases} \quad (3.27)$$

with Fourier transform

$$H(f) = S(f) e^{-j2\pi f T_u} \quad (3.28)$$

Now let $s(t)$ be an m-sequence of length N . Hence

$$T_u = NT_c \quad (3.29)$$

and

$$s(t) = \sum_{n=1}^N a_n p[t - (n-1)T_c] \quad (3.30)$$

with $p(t)$ as defined in (3.2b). The impulse response of the matched filter is therefore

$$H(f) = P^*(f) \sum_{n=1}^N a_n e^{-j2\pi f(N-n+1)T_c} \quad (3.31)$$

where $P^*(f)$ is the complex conjugate of the Fourier transform of $p(t)$.

Thus, the output of the baseband matched filter is simply

$$x(t) = s(t) * h(t) \quad (3.32)$$

$$= \int_0^{T_c} s(t-\tau)h(\tau) d\tau$$

$$= \int_0^{T_c} s(t-\tau)s^*(T_c-\tau) d\tau .$$

Generalizing to any time shift and substituting (3.31) into (3.33)

$$x(t) = \sum_{n=1}^N a_n \int_{(n-1)T_c}^{nT_c} s(t+T_c-\tau-NT_c) d\tau . \quad (3.33)$$

For a practical implementation, consider the sampling $x(t)$ at time instants $t=(N+k/S)T_c$ at a multiple S of the chip rate, i.e.

$$x_k = \sum_{n=1}^N a_n \int_{(n-1)T_c}^{nT_c} s\left(\frac{k}{S}T_c + T_c - \tau\right) d\tau$$

$$= \sum_{n=1}^N a_n \sum_{r=1}^S \int_{(n-1)S \cdot T_c + (r-1)\frac{T_c}{S}}^{(n-1)S \cdot T_c + n\frac{T_c}{S}} s\left(\frac{k}{S}T_c + T_c - \tau\right) d\tau . \quad (3.34)$$

Equation (3.34) is the expression for the discrete time correlation and is easily implemented in digital form, as shown in Figure 3.3. The processing gain (G_p) is simply given by the

Figure 3.3 Physical realization of a digital matched filter

number of delay elements [8, p. 231].

3.4.1 Digital Matched Filtering of Chip-Rate Hopping Equation (3.34) is easily generalized to the case of chip-rate hopping. Again consider an m-sequence of length $N = 2^m - 1$ whose chip rate is determined by a second m-sequence of length $L = 2^l - 1$, which determines a set of L chip rates. The matched filter response would therefore be

$$x_{k_j} = \sum_{n=1}^N a_n \sum_{l=1}^L \int_{(n-1)S + (l-1)\frac{T_c}{S}}^{(n-1)S + l\frac{T_c}{S}} s\left(\frac{k}{S}T_c + T_a - t\right) d\tau \quad 1 \leq j \leq L. \quad (3.35)$$

Thus, depending on the implementation, there would be either:

- i) L matched filters each corresponding to a particular chip rate or,
 - ii) L different sets of coefficients, each corresponding to the coefficients required for a particular chip rate. Each set of coefficients would be dynamically loaded into the matched filter as required or,
 - iii) L different sampling rates, each corresponding to an integer multiple of each chip rate.
- From a practical point of view, implementation of ii) or iii) would give a more efficient and cost effective design.

3.4.2 A Specific Example of Digital Matched Filtering of a Chip-Rate Hopped Signal

Consider a chip-rate hopped system whereby the sampling frequency is dynamically changed

to be an integer multiple, i , of the chip rate, thus giving an equal number of samples per chip. Recall the parameters given in Section 3.3.1, i.e.

$$\begin{aligned}
 N &= 1023 \\
 L &= 255 \\
 B_h &= 1 \text{ MHz} \\
 B_l &= 9 \text{ MHz} \\
 B_u &= 10 \text{ MHz} \\
 Z &= \frac{1 \text{ MHz}}{255} = 3922 \text{ Hz} \\
 1 \cdot 10^{-7} \text{ sec.} &\cdot T_c < 1.11 \cdot 10^{-7} \text{ sec.}
 \end{aligned}$$

Specifically, let the sequence under examination have a chip rate of 9.5 MHz. Further, it is desired to have three samples per chip, thus dictating that the sampling frequency is required to be 28.5 MHz. The number of taps required is simply $3N$ or 3069. Assuming ideal synchronization, 1023 chips per bit, and no noise, correlation peaks of magnitude 3069 will be visible, separated by 3069 samples.

CHAPTER IV

4.0 THE IMPLEMENTATION OF A CHIP-RATE SUPPRESSED DSSS COMMUNICATION SYSTEM

4.1 Introduction

This chapter will describe the implementation of a DSSS communication system designed to suppress the chip-rate feature. The design and implementation of both a hardware based digital transmitter and a software based digital matched filter despreader will be presented and discussed. The rationale behind certain design decisions will be explained.

It was decided to build a simple proof-of-concept communication system to accomplish the goal of suppressing the chip rate from an interceptor. Investigation demonstrated that a hardware transmitter coupled with a software despreader would provide the best solution, given the available time frame and laboratory resources. In particular, the availability of Stanford Telecom's STEL 1032 PRN coder chip and the STEL 1172B numerically-controlled oscillator provided for a relatively simple hardware solution. Binary Phase Shift Keying (BPSK) was employed as a baseband modulation scheme due to its ease of implementation.

The challenge with any spread spectrum communication system is synchronization. The incoming waveform must be synchronized to the receiver clock to allow for accurate despreading and demodulation to the baseband bit stream. The use of a matched filter provides for passive correlation and thus a reduction in receiver complexity. However, upon examination of commercially available components, nothing suitable for this unique application was evident. In particular, commercial matched filters of 64 taps are available (Stanford Telecom), but only four of these may be cascaded thus limiting the length of the m-sequence one may use to $N=127$. Although sufficient for commercial applications, such a short code is unsuitable for a military application. The use of a finite impulse response (FIR) filter was considered, but investigation showed that the maximum number of filter taps available was on the order of 1000. The Nyquist theorem dictates that an m-sequence of length $N=512$, maximum, would have to be employed. Again, this code length is not feasible. It was therefore decided to implement a matched filter in software where in theory, the number of taps available is infinite. In any event, software demodulators are becoming more common.

4.2 Transmitter Design Considerations

The availability of the Stanford Telecom's STEL 1032 PRN Coder chip was the starting point for the hardware implementation of a chip-rate hopped transmitter. This particular chip has several advantageous features, most notably three independent on-chip PRN (Pseudorandom Noise) code generators, each capable of generating maximal length

sequences of up to 32 taps. A provision for on-chip code modulation is available, and operation up to 30 MHz is possible. Having decided to hop the chip rate in a pseudorandom fashion to avoid chip-rate prediction, it became obvious that a direct digital synthesis (DDS) circuit would be required. A Stanford Telecom STEL 1272B frequency-synthesizer board, utilizing a Stanford Telecom STEL 1172B numerically controlled oscillator was available. Its primary features include 32-bit frequency control, clock speeds up to 50 MHz, and an output frequency of up to 20 MHz. These two components served as the starting point for the design and set the main design criteria: the availability of 3 PRN codes of up to 32 taps, a maximum clock speed of 20 MHz or less, and the use of a single 50-MHz crystal from which all timing must be coherently derived.

In order to make interception and code extraction difficult, it was decided to employ a burst transmission scheme. As a result, all three of the available PRN code generators could be utilized: one to determine the chip rate, one to spread data when data was being transmitted, and the last to be transmitted when data was not present. Furthermore, since the latter would be transmitted for a majority of the time, it was necessary to make this code quite lengthy; if it did not repeat over the operational time of the total transmission, reconstruction of this m-sequence by an interceptor would be difficult.

4.3 Selection Of Transmitter Design Parameters

Upon consultation with experts in the intercept field, it was decided to employ a 1 MHz

range in the chip rates. Further consultation resulted in the knowledge that interceptors typically use filters in the range of 10 Hz to 1 kHz. As a result, the chip rates must be separated by at least 1 kHz to avoid interception of more than one chip rate at a time. Furthermore, the communicator must have a sufficiently large selection of chip rates from which to choose. Conversely, this selection must be small enough as to avoid an overly complex receiver. An m-sequence of length 255 ($m=8$) was therefore employed, resulting in 255 possible chip rates over the available 1 MHz, spaced at 3.922-kHz intervals. As a result, there are 16 possible codes which may be used [5]. In addition, it was decided to use a hop band from 9 to 10 MHz to provide a sufficient processing gain, as will be described in Section 4.3.5.

It was decided that an m-sequence of length 1023 ($m=10$) would be used to spread the data. This would allow for the provision to choose 1 of 60 possible codes [5]. As described above, the non-data code should not repeat over the operational time of the transmission to prevent m-sequence reconstruction. Thus, an m-sequence of length 2^m-1 ($m=31$) was employed, resulting in a choice of one of over 69 million possible codes [5].

It was decided that the dwell time at each chip rate would be five milliseconds. This would force an interceptor to use a filter bandwidth of 200 Hz (5 msec^{-1}) or greater in order to resolve a single chip rate. It was also decided to use a burst sequence of one in eight. This ensures that every chip rate has an equal chance of containing data, as demonstrated in Table 4.1. As shown, every chip rate is used, and the same chip rate is visited every 255 data

bursts. Since the burst sequence is one in eight, the net result is that a particular chip rate will contain data once every 2040 hops ($8 \times 255 = 2040$).

Table 4.1 Demonstration Of Equal Probability Of A Chip Rate Containing Data

Sequence Number	Chip Rates Containing Data
1	1, 9, 17, 25, 33, 41, 49, 57, 65, 73, 81, 89, 97, 105, 113, 121, 129, 137, 145, 153, 161, 169, 177, 185, 193, 201, 209, 217, 225, 233, 241, 249
2	2, 10, 18, 26, 34, 42, 50, 58, 66, 74, 82, 90, 98, 106, 114, 122, 130, 138, 146, 154, 162, 170, 178, 186, 194, 202, 210, 218, 226, 234, 242, 250
3	3, 11, 19, 27, 35, 43, 51, 59, 67, 75, 83, 91, 99, 107, 115, 123, 131, 139, 147, 155, 163, 171, 179, 187, 195, 203, 211, 219, 227, 235, 243, 251
4	4, 12, 20, 28, 36, 44, 52, 60, 68, 76, 84, 92, 100, 108, 116, 124, 132, 140, 148, 156, 164, 172, 180, 188, 196, 204, 212, 220, 228, 236, 244, 252
5	5, 13, 21, 29, 37, 45, 53, 61, 69, 77, 85, 93, 101, 109, 117, 125, 133, 141, 149, 157, 165, 173, 181, 189, 197, 205, 213, 221, 229, 237, 245, 253
6	6, 14, 22, 30, 38, 46, 54, 62, 70, 78, 86, 94, 102, 110, 118, 126, 134, 142, 150, 158, 166, 174, 182, 190, 198, 206, 214, 222, 230, 238, 246, 254
7	7, 15, 23, 31, 39, 47, 55, 63, 71, 79, 87, 95, 103, 111, 119, 127, 135, 143, 151, 159, 167, 175, 183, 191, 199, 207, 215, 223, 231, 239, 247, 255
8	8, 16, 24, 32, 40, 48, 56, 64, 72, 80, 88, 96, 104, 112, 120, 128, 136, 144, 152, 160, 168, 176, 184, 192, 200, 208, 216, 224, 232, 240, 248

Several different data rates were considered. The prime considerations were that it must be coherently derived from the 50-MHz reference, and that it be close to a standard data rate. Furthermore, it was decided that although the chip rate would vary, the data rate would remain fixed. As a result, the processing gain, or the number of chips per bit, would vary from burst to burst. It was felt that a software demodulator could easily compensate for this. The number of data bits per burst remained fixed at 64.

Since only one in eight bursts contain data, the actual data rate is simply the data rate employed to modulate the spreading code divided by eight. It was therefore decided to use a data modulating rate of 12.2 kb/second ($50 \text{ MHz} / 2^{12}$) resulting in a net data rate of 1526 bits per second (1.526 kb/sec). In order to achieve a standard data rate of 1200 bits per second (b/sec), 326 of these bits could be dedicated to synchronization, that is, bits used by the receiver for synchronization or resynchronization.

Having employed a hopping band from 9 to 10 MHz and a fixed data rate of 12.2 kb/sec, the system processing gain is easily calculated from Equation 2.1:

$$G_{p_{\min}} = 10 \log_{10} \left(\frac{W_{\min}}{B} \right) = 10 \log_{10} \left(\frac{9 \times 10^6}{12.2 \times 10^3} \right) = 28.7 \text{ dB}$$

$$G_{p_{\max}} = 10 \log_{10} \left(\frac{W_{\max}}{B} \right) = 10 \log_{10} \left(\frac{9.996188 \times 10^6}{12.2 \times 10^3} \right) = 29.1 \text{ dB}$$

In linear terms, the number of chips per bit ranges from 737.7 to 819.4.

4.4 High Level Transmitter Board Implementation

Several methods of board implementation were investigated. Initially, it was desired to pursue a printed circuit board (PCB) solution. However, it was felt that such an implementation could not be fabricated and debugged in the allotted time frame. As a result, it was decided to implement the transmitter board using wire-wrap, as this would provide for quick construction and debugging. It was felt since the operating frequencies were sufficiently low, wire-wrap would not introduce any undesirable effects such as electromagnetic coupling or the introduction of excessive noise.

A quick examination of parts requirements showed that the board would use readily available and standard digital logic components. Consequently, two implementation options were available. The majority of the logic components could be integrated in a Field Programmable Gate Array (FPGA), or the separate components could be procured and wired together. In order to provide for the widest possible design experience, it was decided to pursue the latter.

Table 4.2 gives the overall operational characteristics to which the design was implemented. As mentioned, 16 different hopping codes are available for selection. The only concern was that one of these codes was less random than another. Simulation of the hopping pattern employing each of the 16 codes showed that there was no significant advantage or disadvantage of one over another. As a result, a code employing feedback taps [8, 4, 3, 2]

was arbitrarily selected. The hopping pattern employed is shown in Figure 4.1, while Figure 4.2 gives a partial time-frequency map.

Table 4.2 Operational Features of the Chip-Rate Hopping Transmitter

MODE	HOPPING		CONVENTIONAL
DATA	Present	Absent	Absent
modulating code feedback taps	[10, 8, 3, 2]	[31, 29, 21, 17]	[31, 29, 21, 17]
modulating code length	$2^{10} - 1$	$2^{31} - 1$	$2^{31} - 1$
transmission duration	5 msec	35 msec	user defined
chip-rate hops	1	7	0
chip rates available	$2^8 - 1 = 255$	$2^8 - 1 = 255$	1

A wide selection was available for choosing the two other codes; the code when data was present ($m=10$ sequence) and when data was absent ($m=31$ sequence). The selection was arbitrary, as again, there is no significant advantage or disadvantage of one over another. Thus, the $m=10$ sequence selected employs [10, 8, 3, 2] feedback taps, while the $m=31$ sequence employs [31, 29, 21, 17] feedback taps.

Finally, it was determined that a test mode employing a single, fixed chip rate would be

Figure 4.1 Chip-rate hop pattern using 8 stage code [8, 4, 3, 2] with $2^8 - 1 = 255$ different chip rates

Figure 4.2 Time-frequency map of implemented transmitter

useful. This mode could be used to compare the intercept results between conventional DS and chip-rate hopping. Furthermore, it provides for stable operation during implementation for measuring various circuit parameters such as correct code generation.

4.5 Low Level Transmitter Board Implementation

Having fully established the design criteria and functionality, the detailed chip-level design could then proceed. The board is separated into several functional blocks. High level circuit operation will be explained first. The two most important functional blocks, the interfaces to the STEL 1032 and 1272B, will be examined in detail with supporting timing diagrams. Each timing diagram was reproduced from results obtained using a Hewlett Packard HP1615A Logic Analyser. The other functional blocks and supporting timing diagrams, namely control, data, chip clock, and output may be viewed in Appendix A and are briefly discussed in Section 4.5.4. Figure 4.3 shows the implemented circuit.

4.5.1 High Level Circuit Operation Figure 4.4 serves as an overall functional block diagram of the implemented circuit. On reset, each of the three code generators in the STEL1032 is initialized using a 6.25-MHz setup clock. PNS0 governs the chip rate, PNS1 serves as the data spreading code, while PNS2 is the code transmitted when data is not present. Once each of the code generators has been initialized, two control signals, SETUP_DONE and STP_DONE are generated. This allows for the first frequency to be strobed into the STEL1272B frequency-synthesizer board. Consequently, the transmitter is in

Figure 4.3 Implemented chip-rate hopping transmitter board

Figure 4.4 Overview of transmitter board operation

conventional mode, with PNS2 being transmitted at a fixed chip rate of 9.498094 MHz.

The transmitter remains in conventional mode until a switch is activated to change to hopping mode. In hopping mode, the control circuit generates several signals including:

- i) a 192-Hz clock signal ($T = 5.2 \text{ ms}$) used to change the chip rate (CHANGE FREQUENCY)
- ii) a 192-Hz, 1/8 duty cycle clock signal used to enable the transmission of data (DATA_ENABLE)
- iii) a 12.2-kHz clock used as the data clock

The change frequency clock is also used to clock the shift register generator governing PNS0. As a result, the chip rate changes every five milliseconds. On every eighth chip rate, data is transmitted at a rate of 12.2 kb/sec, which is in turn spread by PNS1 at the present chip rate. Once all of the data has been transmitted (43 seconds), the data circuit generates a DATA DONE signal, which returns the transmitter to conventional mode. Figure 4.5 provides a high level timing diagram of the circuit.

4.5.2 STEL 1032 Setup Circuit The 1032 PRN coder chip contains three independent 32 bit code generators or registers. The feedback tap combinations are controlled by the three 32-bit MASK registers. The codes may be started at any random point simply by initializing each of the three 32-bit INIT registers with the desired starting sequence, and a specific position in any of the codes may be detected through comparison with the value stored in

Figure 4.5 Overview of transmitter board timing

each of the 32-bit EPOCH registers. When identical, an epoch pulse is generated. Three eight bit control (CTL) registers define whether or not the coders are reloaded at any point during operation. Three clock inputs are present, CHIPCLK0 to CHIPCLK2, which clock coders 0 to 2 respectively. Both the data and address busses are eight bits. Thus, each of the coders is programmed by the data contained on the data bus. The eight bit data is mapped to the appropriate 32-bit register by address bits 0 through 4, while the particular coder being programmed is selected by address bits 5 through 7.

The implemented circuit and timing may be viewed in Figures 4.6 and 4.7 respectively. An E²PROM (U11) was selected to provide data for the data bus. The software used to program the E²PROM is provided in Appendix B. The counter outputs (U2 and U3) serve as the address bus and are used to address both the E²PROM and the 1032. Address bits 5 and 6 also serve to turn the circuit off once the 1032 has been programmed. The circuit is cleared on reset. Once reset is high, clocks A (6.25 MHz), B (3.125 MHz), and D (781.25 kHz), the outputs of U1, are started. SCSN, the active low chip enable signal, mirrors clock D while A5 and A6 are not simultaneously high. SWRN, the active low write enable signal, writes the data present on the data bus to the register specified by address bits 0 to 7. The data is latched on the rising edge of SWRN. This signal is simply the chip enable signal delayed by one ClkB period (U4A) which ensures that the data is written and latched while SCSN is low, but not at the same time as SCSN goes low. The SWRN signal is further delayed by one more ClkB period (U4B), and two ClkA periods (U8) to generate a load signal, thus ensuring that data is not loaded into the 1032 while a register is unstable (SWRN low).

Figure 4.7 STEL 1032 setup circuit timing

Address bits 5 and 6 are used to determine which coder is programmed (U10, SLOADX). Each of the three coders are programmed sequentially, although Figure 4.7 only shows the programming of coder 0. Once A5 and A6 are simultaneously high, SETUP_DONE is latched high. STP_DONE, a pulse derived from SETUP_DONE is also generated. Both signals are used in the 1272B interface circuit their purpose will be described in the following section. SETUP_DONE is also used in the chip clock circuit and is used to turn off the selection of the setup clock used in the programming of the coders. Setup occurs in 81.92 microseconds, corresponding to 64 ClkD periods.

4.5.3 STEL 1272B Board Interface The Stanford Telecom 1272B utilizes the Stanford Telecom STEL 1172B 32 bit numerically controlled oscillator (NCO). Although both sine and cosine signals are generated, only the sine output was used. The board is programmed through an eight-bit data bus and a two-bit address bus. The address bus determines which byte of the 32-bit Δ -phase register is written to. The output frequency is determined by the 32-bit value stored in the Δ -phase register and is defined by:

$$f_o = \frac{(f_{clock})(\Delta \text{ phase})}{2^{32}} \quad (4.1)$$

where f_{clock} is the clock frequency employed (50 MHz).

The schematic and timing information are provided in Figures 4.8 and 4.9 respectively. It should be noted that the 1272B requires an ECL clock. This is provided by a TTL to ECL

- NOTES:
1. 0.1 μ F decoupling capacitors between Vcc and GND
 2. J1- 25 pin D female connector
 3. J2, SMA jack (female)

Figure 4.8 STEL 1272B board interface

Figure 4.9 STEL 1272B board interface timing

converter (U10). An E²PROM (U7) was used to store the Δ -phase information. The code used to program the E²PROM is provided in Appendix B. The E²PROM utilizes ten bit addressing. The upper eight bits are derived from an eight bit series to parallel converter (U1). The purpose of this converter is to generate an eight-bit word from PNS0, the hopping code, thus defining one of 255 possible frequencies. The lower two bits are used to access a particular byte, as defined by the address bus (U5), and depend upon which of the four bytes of the Δ -phase register is being changed. On reset, the circuit is cleared and is idle until the STEL 1032 has been initialized, signified by a SETUP_DONE. This signal clears the latch (U4). The STP_DONE pulse arrives 80 nanoseconds later and serves as a single clock pulse for the purposes of loading the counter (U5) with a value of eight and clocking the first bit of PNS0 into U1. It should be noted that the resulting word used to define the frequency is 00000000, the state which never occurs in m-sequence generation. It was decided to map the zero state to 9.498094 MHz, the midpoint of the hop band, to provide a convenient start-up frequency. Four write signals (FWRN) are generated through the 12.5 MHz clock which write data to the byte of the buffer register defined by the address bus (FA0, FA1). Each byte is written on the falling edge of FWRN. A load pulse is generated (FLOAD) once the count has reached 12, which loads the Δ -phase register with the contents of the buffer register on the rising edge of the 50-MHz clock after the rising edge of an FLOAD. The NCO output changes 34 clock cycles (680 nanoseconds) later due to NCO pipeline delays. FLOAD also prevents the generation of any further write pulses. Once the count reaches 15, the RCO of U5 is used to reset the latch (U4) to prevent a further frequency

research, it was decided to drive the A/D board with an harmonic of the transmitter chip clock, and to use the data enable signal on the transmitter board to trigger data collection. The A/D board hardware only permits operation in the single-channel mode when using an external clock, thus limiting available memory to 512 kilobytes or 256 kilosamples. Spectral examination showed the third harmonic of the chip clock to be relatively strong, thus resulting in the provision for three samples per chip. The following calculation shows that available memory would not be exceeded:

max. no. of samples required = (max. chip rate)(samples per chip)(collection time)

$$\left(9.996078 \cdot 10^6 \frac{\text{chips}}{\text{sec}} \right) \left(3 \frac{\text{samples}}{\text{chip}} \right) (5 \cdot 10^{-3} \text{ sec})$$

150 kilosamples

300 kilobytes .

Thus the Nyquist limit is satisfied while utilizing available memory.

A final consideration involved the transfer of data, once collected. As described, data is collected every eighth chip rate. As a result, there are 35 milliseconds available between data collections to transfer the data from board memory to another location. The following calculation shows that a disk transfer speed of roughly 8.8 MHz would be required:

$$\text{required disk transfer speed} = \frac{\text{data size}}{\text{available time}}$$

$$\frac{(300)(1024) \text{ bytes}}{35 \text{ msec}}$$

$$= 8.8 \text{ MHz .}$$

2

OF/DE

2

PM-1 3 1/2" x 4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

Figure 4.10 Flowchart of matched filter program

control circuit to disable hopping.

The chip clock circuit is used to determine the clocking of each of the three PRN coders. During setup, all three coders are clocked using the 6.25-MHz setup clock produced in the 1032 setup circuit. Once the setup is complete, the transmitter enters conventional mode and coder 2 is clocked by the 9.498094-MHz sine output of the 1272B frequency-synthesizer board. In hopping mode, the CHANGE_FREQUENCY signal generated in the control circuit is used to clock coder 0 (chip clock 0). Recall that PNS0 is the m-sequence used to define the chip rate in the 1272B interface circuit. Coder 1 and coder 2 are clocked by the sine output of the 1272B. If the current hop contains data, coder 1 is clocked (CHIPCLK1). Conversely, coder 2 is clocked in the absence of data (CHIPCLK2). A test point (TESTCLK) is also produced to aid in debugging.

The output circuit is used to select the transmission of either MODCODE1 or PNS2 (TXOUT). In conventional mode, TXOUT consists of PNS2 only. In hopping mode, TXOUT consists of both MODCODE1 and PNS2. MODCODE1 is the result of the modulo-2 addition of the spreading code (PNS1), with the data appearing at the STIM1 input. This modulo-2 addition occurs on the falling edge of STLD. Recall that STLD is a replica of the 12.2-kHz data clock and is generated in the data circuit only during a data hop. If the current hop does not contain data, PNS2 is selected. The signal TXOUT is therefore continuous in either mode. Test points are also provided to aid in debugging: CODE0, CODE1, CODE2, and TESTCLK.

4.6 Matched Filter Hardware Design Considerations and Implementation

The first and foremost consideration was choice of platform. It was realized that processor speeds, as compared to a hardware implementation, are relatively slow for this type of application. Both DOS-based and UNIX-based platforms were available. Careful evaluation of both platforms demonstrated that a DOS-based machine was inferior for several reasons, most notably computational speed. As a result, a Sun SPARC5 (85 MHz) was selected.

The second design consideration involved data collection. It was recognized that the collection would have to be performed by an Analog-to-Digital (A/D) board, and that the board would have to operate at a minimum of twice the maximum operating frequency of the transmitter board. A search of commercially available S-Bus A/D boards showed that none operated at the required speed. Thus, it was required to use a PC-Bus A/D board, as several were available at speeds of up to 500 MHz. A Signatec DA60 A/D board was selected. This dual-channel board is capable of conversion speeds of up to 60 MHz, and has several advantageous features including provisions for both an external trigger and clock, on-board memory of 512 kilobytes per channel, and 12-bit sampling. As a result, data collection occurred on a PC (75 MHz Pentium), while processing occurred on the SPARC5.

Having selected the data collection and processing means, a further consideration was which sampling frequency to employ. A single sampling frequency would result in a non-integer multiple of samples per chip. Since true synchronization was beyond the scope of this

research, it was decided to drive the A/D board with an harmonic of the transmitter chip clock, and to use the data enable signal on the transmitter board to trigger data collection. The A/D board hardware only permits operation in the single-channel mode when using an external clock, thus limiting available memory to 512 kilobytes or 256 kilosamples. Spectral examination showed the third harmonic of the chip clock to be relatively strong, thus resulting in the provision for three samples per chip. The following calculation shows that available memory would not be exceeded:

$$\begin{aligned}
 \text{max. no. of samples required} &= (\text{max. chip rate})(\text{samples per chip})(\text{collection time}) \\
 &= \left(9.996078 \cdot 10^6 \frac{\text{chips}}{\text{sec}} \right) \left(3 \frac{\text{samples}}{\text{chip}} \right) (5 \cdot 10^{-3} \text{ sec}) \\
 &= 150 \text{ kilosamples} \\
 &= 300 \text{ kilobytes} .
 \end{aligned}$$

Thus the Nyquist limit is satisfied while utilizing available memory.

A final consideration involved the transfer of data, once collected. As described, data is collected every eighth chip rate. As a result, there are 35 milliseconds available between data collections to transfer the data from board memory to another location. The following calculation shows that a disk transfer speed of roughly 8.8 MHz would be required:

$$\begin{aligned}
 \text{required disk transfer speed} &= \frac{\text{data size}}{\text{available time}} \\
 &= \frac{(300)(1024) \text{ bytes}}{35 \text{ msec}} \\
 &= 8.8 \text{ MHz} .
 \end{aligned}$$

At present, PC disk transfer speeds are on the order of 2.5 MB/sec. Therefore, another method had to be employed. A much faster data transfer may occur by defining a Random Access Memory (RAM) drive. The speed increase is simply due to direct electronic transfer, as opposed to physically moving the write head across a hard disk. A 30 MB RAM drive was therefore defined, allowing for the collection of 100 data bursts. The data was then transferred to the SPARC5 workstation via a 10 Mb/sec ETHERNET link for processing.

4.7 Matched Filter Software Design Considerations and Implementation

It was decided to implement the digital matched filter using a high level language, namely C. Specifically, a GNU C compiler was available on the SPARC5 workstation. Furthermore, numerous useful and fully debugged functions were available [27 pp 398-471]. Finally, experienced C programmers were readily available for consultation, if required.

The next design consideration was that of algorithm selection. Although the processing does not occur in real time, it was still desired to despread as rapidly as possible. The basic method of despreading involves convolution (correlation), and two implementations are possible. The first and most obvious is to simply perform a time domain correlation directly. Investigation showed that this implementation would be unnecessarily time consuming. A second implementation involves the properties of the Fourier transform. It is widely known that convolution may be achieved by performing a Fourier transform on both the complex conjugate of the stored reference and the data under study, multiplying, and performing an

inverse Fourier transform, i.e;

$$R_c(\tau) = \mathcal{F}^{-1} \left[\mathcal{F}[s(t)] \cdot \mathcal{F}^*[h(t)] \right] \quad (4.2)$$

Furthermore, a tremendous decrease in processing time may be achieved by employing the Fast Fourier Transform (FFT) since the number of multiplications and additions required is on the order of $N \log_2(N)$, as opposed to N^2 for convolution [28, p.263]. It was therefore concluded that an FFT based realization of a software digital matched filter would be the most time efficient.

A final design consideration was the size of FFT to employ. Recall that the chip rates are spaced at 3922-Hz intervals. As a result, the FFT must have a 3922-Hz resolution or better. The following calculation determines the size required:

$$N = \frac{\text{max. sampling frequency}}{\text{required resolution}}$$

$$= \frac{(9.996078 \cdot 10^6)(3)}{3922}$$

$$= 7646$$

$$\therefore N = 8192$$

An 8192-point FFT was therefore implemented.

4.7.1 Program Description The implemented C code may be examined in Appendix C, and a flowchart description of operation is given in Figure 4.10.

Figure 4.10 Flowchart of matched filter program

The initial step of the digital matched filter program involves the generation of the hopping code. Each of the 255 unique words corresponding to each of the 255 different chip rates is generated and stored in the order that they are generated on the transmitter board in the array HOPSEQ. Next, a byte swap is performed on each sample in the stored data file, since UNIX and DOS store integers in reverse formats, and the result is cast as a floating point number for use in the FFT routines. The program then enters the main or outer loop.

The outer loop is defined by the variable NUMHOPS, which is simply the number of hops which are to be despread. The number of chips per data bit is first calculated. The stored reference ($m=10$ sequence) is then generated and shifted to ensure that the first chip of the stored reference coincides with the first chip used to spread the first data bit. This was known a priori from examination of the generated on-board sequence during verification of the transmitter board. The resulting m -sequence is then sampled and zero padded from three times the number of chips per data bit to 8192, the FFT size. Despreading therefore occurs on a bit by bit basis. The program then enters a second inner loop whose purpose is to despread an entire data burst as defined by NUMBITS, the number of bits in the data burst (64).

The first data bit is retrieved and the floating point samples are scaled to between -1 and 1 to ensure correct bit polarity after despreading. The sampled spread data bit is then zero padded from the three times the number of chips per data bit to 8192. An 8192 point FFT is performed on both vectors. To ensure correct correlation, the complex conjugate of the

stored reference is calculated simply by inverting the sign of the imaginary part of the resulting FFT. A complex multiplication is performed, followed by an 8192 point inverse FFT, and the result is stored for later examination. The spreading code is then shifted and sampled for alignment with the next data bit, and to ensure that the correct chips are employed for despreading. This is necessary since the number of chips per data bit varies from about 738 to 819 (Section 4.3), somewhat less than the code length of 1023. As a result, despreading must occur on a bit-by-bit basis as opposed to a hop-by-hop basis. Thus, the matched filter is not only dynamic in that the number of taps change from hop to hop, but also the taps themselves change from bit-to-bit. The inner loop continues as described until all 64 of the data bits have been despread. Program control is then returned to the outer, main loop.

Each hop is processed as described above until all 100 of the hops have been completed. The results are then stored in a data file for later graphical analysis using GNUPLOT, a 2 dimensional graphics tool found on the SPARC5 workstation. Although not implemented, one could extract the clock and set a hard limit for threshold detection. One would therefore know approximately where to look (i.e. sample number) in the despread data, thus limiting the number of bit errors normally incurred due to fluctuating noise peaks.

CHAPTER V

5.0 CHIP-RATE HOPPING TESTING AND RESULTS

5.1 Introduction

This chapter describes the test procedures and results of both the detectability and communication effectiveness of the chip-rate hopped system described and implemented in Chapter IV. The transmitter board was constructed and first determined to operate correctly. The transmitted signal was then analyzed using an envelope-type chip-rate detector for the purpose of determining the detectability of the chip rate when chip-rate hopping was used. Both a swept-superheterodyne and an FFT spectrum analyser were employed using an input E_s / N_0 down to -10 dB (i.e. below the limit of detectability). All detection tests were performed to favour the interceptor and will be explained in more detail in Section 5.3. A detection was declared if the chip-rate detector produced one or more spectral lines at the frequency band being examined over the test period. A dynamic matched filter was then implemented and tested down to an input E_s / N_0 of -20 dB (i.e. the point at which bit errors begin to appear). A bit detection decision was made on the basis of graphical analysis of the correlation results.

5.2 Verification of Correct Operation

The transmitter board was examined using both a logic analyser (HP1615A), and a spectrum analyser (HP 8568B) to verify the high-level design. The logic analyser was used to determine that the correct PN codes were generated. 100 chips of the hopping ($m=8$) and spreading ($m=10$) codes were compared with MATLAB-generated replicas of the desired sequences and it was verified that the correct sequences were produced. Due to the length of the non-data code, it was not feasible to perform the same test procedure. Furthermore, it was more important to determine code period rather than the actual code sequence since the communicator does not require any knowledge of the non-data code for despreading. As a result this PN code was examined to ensure that it was pseudorandom and that the period was sufficiently lengthy. Verification was accomplished by using the EPOCH2 output of the 1032 PRN coder chip and clocking it continuously using the continuous 12.5 MHz clock. Simply, when 31 bits of the generated sequence match the 31 bits which were used to initialize the code generator, a single pulse of one clock period in length is produced. The logic analyser was therefore configured to trigger on this EPOCH2 pulse. The following calculation demonstrates that using the 12.5-MHz clock, the code is expected to repeat every 172 seconds:

$$\text{code period} = \frac{2^{31} - 1 \frac{\text{chips}}{\text{code}}}{12.5 \cdot 10^6 \frac{\text{chips}}{\text{sec}}}$$

171.7 seconds

2 minutes 52 seconds

Examination showed the generation of an EPOCH2 pulse at the expected time. It was therefore determined that the non-data code was produced correctly.

The board was next tested to ensure that all of the chip rates in the 1-MHz hopping band (9 to 10 MHz) were utilized. The maximum hold feature was employed on the spectrum analyser, both the resolution and video bandwidths were set to 10 kHz, and a frequency span of 1.2 MHz was examined. Figure 5.1 shows a flat response over 1.02 MHz, thus confirming that the correct hopping band is employed and that all chip rates are present. It was next necessary to confirm the chip-rate spacing of 3922 Hz. Several preliminary tests were performed employing various spectrum analyser configurations, and it was determined that the best results would be obtained again using the maximum hold feature, but utilizing a span of 39.22 kHz, exactly ten times the desired spacing. Figure 5.2 shows the expected ten chip rates, each equally spaced at 3922 Hz.

A further test involved the confirmation of frequency accuracy. As explained in Section 4.5.3, the board was designed to employ a chip rate of 9.498094 MHz on reset. Figure 5.3 gives an observed frequency of 9.497935 MHz, an error of 159 Hz or 16 parts per million

Figure 5.1 Verification of hopping band and chip rates

Figure 5.2 Verification of chip-rate spacing

Figure 5.3 Verification of frequency accuracy

(ppm). This may be attributed to the accuracy of the 50-MHz crystal and is within the 25-ppm specification cited.

A final test verified that the operational time of the transmitter was 43 seconds, as expected. That is, once hopping mode was entered, the chip rate changed every five milliseconds for a total of 43 seconds and then returned to conventional mode. It was therefore concluded that the transmitter board operated as designed.

5.3 Chip-Rate Detection Test Parameters

Having verified correct operation, detection testing could commence. Two types of chip-rate detectors were employed: the swept-superheterodyne and FFT spectrum analysers. The experimental setup is given in Figure 5.4. A standard intermediate frequency (IF) of 70 MHz was chosen to utilize available components. The first attenuator provides a means of varying the input SNR to the chip-rate detector. Here, SNR is defined as E_c / N_0 , where E_c is the energy per chip and N_0 is one-sided power spectral density of additive white Gaussian noise (AWGN). A 20-MHz band pass filter was employed, resulting in a 1-dB variation from optimal in the SNR of a detected chip rate [26]. A second attenuator was necessary to prevent saturation of the frequency doubler.

All tests were performed to favour the interceptor. Preliminary testing showed a potential interceptor would have the most success using the maximum-hold feature. This feature

simply displays the maximum spectral value calculated at a particular frequency bin over multiple examinations of that bin during a specified cumulative observation time. The interceptor also had a priori knowledge of the frequency band being utilized.

Two cumulative search times were utilized. It was judged that a 30-second search would provide for a realistic intercept test. A second search time of ten minutes was employed to test the ultimate detectability of the chip-rate hopped signal. The ten-minute search time is well in excess of the 43-second operational time of the transmitter. The DATA_DONE signal, which normally terminates hopping mode and returns the transmitter to conventional mode, was disconnected to allow for unlimited operation in hopping mode.

Both hopping and conventional modes were examined. The conventional mode was used to determine correct chip-rate detector operation and also serves as a comparison with hopping mode results.

5.4 Chip-Rate Detector Results

The resolution bandwidth of the spectrum analyser was set to emulate the narrowband detection filter of a chip-rate detector. It was decided to examine a 39.22-kHz span, exactly ten times the chip-rate spacing, for two reasons. First, the display could be configured to ensure that the chip-rate lines produced would coincide with the vertical display lines on the

spectrum analyser, thus aiding in detection decisions. Secondly, preliminary testing showed that spans much larger than this tended to blend the chip-rate lines together. Figure 5.5 shows the cumulative results of a ten-minute search using an input E_c/N_0 of 0 dB and a 300-Hz detection filter. As expected, all ten chip-rate lines are evident. A 3-dB degradation in detected SNR, as compared to a single fixed chip rate, is also observed. Using a more realistic search time of 30 seconds, Figure 5.6 demonstrates that typically only four of the ten chip-rate lines were evident, each with a varying amount of degradation in detected SNR as compared to a single fixed chip rate. In a realistic covert scenario, a communicator would be operating at a maximum E_c/N_0 of -10 dB. Figure 5.7 shows that no chip-rate lines were detected with an input E_c/N_0 of -10 dB and a cumulative ten-minute search time, even when the detection filter bandwidth was reduced to 30 Hz.

The above results, although promising, are not conclusive. The fundamental limitation when employing a swept-spectrum superheterodyne chip-rate detector is one of coincidence. Unless the spectrum analyser happens to be sweeping a particular frequency at the particular time that a particular chip rate is present, a detection will not occur. A laboratory FFT analyser was available and therefore utilized. An FFT analyser overcomes the coincidence problem by simultaneously examining all frequencies in a given band. Figures 5.8 and 5.9 present the results of a ten-minute search, again using input E_c/N_0 's of 0 and -10 dB respectively, and again using a resolution bandwidth of 30 Hz. As with the swept-spectrum analyser, no chip-rate lines are evident in the -10 dB case. A Hanning window was used and

Figure 5.5 Detected chip-rate lines with (upper) and without (lower) hopping: $E_c/N_0 = 0$ dB, cumulative sweep time = 10 minutes

Figure 5.6 Detected chip-rate lines with (upper) and without (lower) hopping: $E_c/N_0 = 0$ dB, cumulative sweep time = 30 seconds

Figure 5.7 Detected chip-rate lines with (upper) and without (lower) hopping: $E_c/N_0 = -10$ dB, cumulative sweep time = 10 minutes

Figure 5.8 Hopping mode detection results using an FFT analyser: $E_c/N_0 = 0$ dB, cumulative sweep time = 10 minutes

Figure 5.9 Hopping mode detection results using an FFT analyser: E/N_0 = -10 dB, cumulative sweep time = 10 minutes

Figure 5.10 Illustration of the 200 Hz bandwidth of the chip-rate spectra

140 data collections occurred. It may be concluded that at present, the chip rate detector cannot detect a chip-rate hopped signal when an input E_c/N_0 of -10 dB is employed. This is due to three reasons. First, when a swept-spectrum superheterodyne CRD is used, coincidence problems prevent some detections: the instantaneously tuned analysis frequency does not match the current chip rate. Secondly, when an FFT analyser is used, there is an overhead problem. The FFT analyser only collects data for roughly 30% of the search time, while the other 70% of the time is spent processing the collected data. Even in an examination of 140 spectra, a chip rate will occasionally be missed (e.g. Fig. 5.8). Finally, detection is fundamentally limited by the 200-Hz bandwidth of the chip-rate spectra, the reciprocal of the five-millisecond dwell time, as illustrated in Figure 5.10. Here, the 3 dB bandwidth was measured to be 160 Hz. The conclusion which may be inferred is that with this particular system, chip-rate detector performance will improve as the detection bandwidth is decreased to 200 Hz. Any further bandwidth reduction will have no effect on performance. In contrast, a single fixed chip rate (e.g. a non hopping system) can be resolved using a detection bandwidth of 10 Hz or less. This allows an interceptor to extract a chip rate at lower E_c/N_0 's simply because all of the signal energy will be contained in the detection filter bandwidth.

5.5 Dynamic Matched Filter Testing

The experimental setup may be viewed in Figure 5.11. Since the primary goal of this research was detection avoidance, synchronization has been assumed. As described in

Figure 5.11 Matched filter test setup

Section 4.6, the third harmonic of the chip clock was used to drive the sampling clock on the A/D board to ensure three samples per chip, regardless of the chip rate. Data collection was triggered using the DATA_ENABLE signal on the transmitter board. Carrier synchronization was assumed and all tests were performed at baseband. The software was first evaluated to ensure correct operation by using the dual channel mode of the A/D board and no added noise. One channel was used to collect the sampled transmitted data, while the other was used to collect the sampled spreading code. This was possible through the implementation of the CODE1 output which was introduced during board construction as a test point. The software was deemed to work correctly as the resulting correlation peaks had the expected magnitude, three times the number of chips per bit, and appeared at the correct sample number, again at three times the number of chips per bit.

The despreading performance was next evaluated at various values of E_b / N_0 . Figures 5.12 through 5.15 show the correlation results achieved using E_b / N_0 's of ∞ , 0, -15 and -20 dB respectively. Only the first five data bit results, the first five of the burst, are displayed for clarity. These particular data collections arise from data, here logic one's, spread using a chip rate of 9.003922 MHz. 738 chips per bit, or 2214 samples per chip, are therefore expected.

The results using an E_b / N_0 of ∞ (Figure 5.12) serve as a benchmark and demonstrate a minor problem with the despreading algorithm. Correlation peaks appear at the correct sample number, but their magnitude is diminished from the 2214 expected. This is explained as

Figure 5.12 Despreading results; chip rate = 9.003922 MHz, $E_c/N_0 = \infty$

Figure 5.13 Despreading results; chip rate = 9.003922 MHz, $E_c/N_0 = 0$ dB

Figure 5.14 Despreading results; chip rate = 9.003922 MHz, $E_b/N_0 = 15$ dB

Figure 5.15 Despreading results; chip rate = 9.003922 MHz, $E_b/N_0 = 20$ dB

It may also be concluded that effective communication is possible. A software based dynamic digital matched filter was implemented in C on a UNIX platform using an FFT-based despreading algorithm. The data was collected by digitizing the incoming baseband waveform at three samples per chip and was accomplished by using the third harmonic of the chip clock to drive a PC-bus A/D board. Data collection occurred only when data on the transmitter board was spread and transmitted. This was considered to be acceptable as the goal of the research was not one of synchronization but one of detection avoidance. The samples were then sent to a SPARC5 workstation for processing via a 10-Mbit per second ETHERNET link. The results show acceptable results at an E_b / N_0 of -15 dB, 5 dB below the demonstrated limit of the chip-rate detectors. Performance begins to degrade, in the form of bit errors, below -15 dB. At this point, false bit detections and decisions begin to occur due to the predominance of noise.

6.3 Future Work

Although satisfactory results have been achieved, there are two identifiable areas for improvement and further work. These may be categorized as design improvements and further testing. Transmitter improvements include integrating the standard logic components into an FPGA, implementation of an in-house frequency synthesizer, true data input/output, operation from a single power supply, and PCB construction. A provision could also be implemented to stop and start the hopping mode at any point during operation. This would be useful from a testing standpoint. The receiver could be improved in two ways. Primarily,

6 dB from the ideal. The limit of ideal system performance appears to be at an E_c / N_0 of -15 dB (Figure 5.14). Correlation peaks are still clearly visible and are roughly 13 dB above the RMS noise floor. Figure 5.15, with an E_c / N_0 of -20 dB, shows the beginning of the introduction of bit errors and correlation peaks about 0.5 dB above the RMS noise floor. It should be noted that as the SNR degrades, a correlation 'envelope' seems to appear. This is simply due to the fact that the entire m-sequence was not used in the correlation.

Again, the goal of this research was primarily to avoid chip-rate detection, and secondarily to prove that effective communication was possible. It is recognized that the despreading algorithm and corresponding implementation could be improved. However, even using a rudimentary despreader, it has been proven that effective communication has been achieved using an E_c / N_0 of -15 dB. Recall that it was proven that the chip rates could not be detected at an E_c / N_0 of -10 dB. As a result, it may be concluded that this laboratory proof-of-concept communication system is sufficiently covert [29].

CHAPTER VI

6.0 SUMMARY AND RECOMMENDATIONS

6.1 Introduction

The objective of this research was to design, implement, and evaluate a communication system which is difficult to detect when using a filter-and-square envelope-type chip rate detector (CRD). Chapter II introduced spread spectrum theory and techniques. In particular Direct Sequence (DS) and methods for interception were presented. An innovative technique to overcome both interception and exploitation, Chip-Rate Hopping (CRH) was developed and presented in Chapter III. The design and implementation of both a hardware CRH transmitter board employing DS-Binary-Phase-Shift-Keying (DS-BPSK) modulation, and a software digital matched filter were presented in Chapter IV. Supporting experimental evidence, found in Chapter V, demonstrated that Chip-Rate Hopping is a valid technique to counter filter-and-square envelope-type chip-rate detectors. Demodulation results employing a software implementation of a digital matched filter may also be found in Chapter V and show that acceptable communication performance has been achieved. The following section will summarize the obtained results, while Section 6.3 provides recommended areas for future work and improvements.

6.2 Summary

A chip-rate hopped spread spectrum communication system which avoids envelope-type chip-rate detection was designed and implemented. A hardware wire-wrapped transmitter board was constructed around Stanford Telecom's STEL 1032 PRN Coder and STEL 1272B Frequency Synthesizer board. Standard digital logic components were employed and the design was verified using an HP1615A Logic Analyser. The chip rate was varied in a pseudorandom manner employing an $m=8$ sequence resulting in 255 different chip rates equally spaced by 3922 Hz. The dwell time at each chip rate was five milliseconds and a hopping band from 9 to 10 MHz was employed. Data was transmitted on every eighth chip rate and was spread using an $m=10$ sequence. When data was not present, an $m=31$ sequence was transmitted. Two modes were implemented: a conventional or fixed chip-rate mode and a hopping mode. The conventional mode was a useful benchmark for comparison of the results achieved when the hopping mode was tested.

Chip-rate detection tests were performed using both swept superheterodyne and FFT spectrum analysers. All tests were performed to favour the interceptor; the maximum hold feature on the spectrum analysers was employed and the search was limited to the frequency band being used. A 70-MHz intermediate frequency was chosen to utilize available laboratory components. The transmitter was first tested using the swept superheterodyne spectrum analyser and the results show that the chip rate was not detected during a ten minute search and using an input E_c / N_0 of -10 dB and a detection bandwidth of 30 Hz.

Although promising, it could not be confidently concluded that the chip rate was undetectable due to the fundamental coincidence problem encountered when using a swept-spectrum analyser. To overcome the coincidence problem, detection tests were then performed using a laboratory FFT analyser which simultaneously examines all frequency bins in a frequency band of interest. The results were promising in that no chip rates were evident in a 10-minute search using an input E_c / N_0 of -10 dB and a detection bandwidth of 30 Hz. The limitation encountered when using the FFT analyser is one of overhead. Data is collected for approximately 30% of the time, while the remaining time is spent on processing. It is therefore possible to miss detections while processing.

Examination of the chip-rate spectra demonstrates that detection is fundamentally limited by its 200-Hz bandwidth, the reciprocal of the five-millisecond dwell time. Thus, detection performance will improve as the narrowband filter bandwidth of the chip-rate detector decreases to 200 Hz, where detection is maximized. A further detection bandwidth reduction will not result in improved detection performance.

It may be concluded that using present chip-rate detection technology, the implemented system may not be detected when using a practical E_c / N_0 of -10 dB or less (LPI). In particular, neither the swept-superheterodyne spectrum analyser nor the FFT spectrum analyser will detect the chip rate. The limitation, as demonstrated in Figures 5.8 and 5.9, is one of degrading SNR.

It may also be concluded that effective communication is possible. A software based dynamic digital matched filter was implemented in C on a UNIX platform using an FFT-based despreading algorithm. The data was collected by digitizing the incoming baseband waveform at three samples per chip and was accomplished by using the third harmonic of the chip clock to drive a PC-bus A/D board. Data collection occurred only when data on the transmitter board was spread and transmitted. This was considered to be acceptable as the goal of the research was not one of synchronization but one of detection avoidance. The samples were then sent to a SPARC5 workstation for processing via a 10-Mbit per second ETHERNET link. The results show acceptable results at an E_b / N_0 of -15 dB, 5 dB below the demonstrated limit of the chip-rate detectors. Performance begins to degrade, in the form of bit errors, below -15 dB. At this point, false bit detections and decisions begin to occur due to the predominance of noise.

6.3 Future Work

Although satisfactory results have been achieved, there are two identifiable areas for improvement and further work. These may be categorized as design improvements and further testing. Transmitter improvements include integrating the standard logic components into an FPGA, implementation of an in-house frequency synthesizer, true data input/output, operation from a single power supply, and PCB construction. A provision could also be implemented to stop and start the hopping mode at any point during operation. This would be useful from a testing standpoint. The receiver could be improved in two ways. Primarily,

true synchronization and real time despreading could be implemented. At present, technology points to a hardware solution. In future, a real-time software solution will be possible once CPU speeds on PC's improve and true multi-tasking is possible. Secondly, the despreading algorithm could be optimized to provide better performance. At present, performance begins to degrade below an E_c/N_0 of -15 dB, well above the theoretical limit. Other algorithms may also be tested and comparisons of the speed and computational efficiency may be examined.

With regard to further testing, it would be interesting to produce a bit-error rate curve and compare the performance with theory. Results could then be produced containing both the bit error rate and detection performance, hopefully showing that for a desired bit error rate at a given SNR, the chip rate could not be detected. Finally, it would be interesting to employ QPSK modulation and examine both the carrier and chip-rate suppression.

Appendix A: Circuit and Timing Diagrams

NOTES:

1. 0.1 μ F decoupling capacitors between Vcc and GND

Figure A1 Control circuit schematic

Figure A2 Control circuit timing

NOTES:
1. 0.1µF decoupling capacitors between Vcc and GND

Figure A3 Data circuit schematic

Figure A4 Data circuit timing

NOTES:

1. 0.1 μ F decoupling capacitors between Vcc and GND
2. J1- SMA jack (female)

Figure A5 Chip clock circuit schematic

SELECTING AN OSCILLATOR

SELECTION

CHANGING FREQUENCY
at 100 Hz

DATA SAMPLE
at 10 Hz

CHUNK 1

CHUNK 2

CHUNK 3

Figure A6 Chip clock circuit timing

Appendix B: Software Listings For E²PROM Data

```
#include "stdio.h"
#include "stdlib.h"

/*****
This program generates data to program an EPROM. The data
is then used to program the STEEL 1172B NCO. It is designed
so that on start up, a 9.5 MHz chip rate will be generated.
The 255 chip rates range from 9.000000MHz to 9.996188MHz, with
a spacing of 3922 Hz.
*****/
```

S Record format is S1 BC AAAA H CC where
 S1 is record type
 BC is the byte count (3+#data bytes)
 AAAA is the address
 H is the data byte
 CC is the checksum (one's complement binary addition
 of the preceding bytes in the record)

NOTE: end of record is S903AAAACC where AAAA is the address
 after the last data byte

NOTE: everything is in hex notation

W.P. McPherson Aug. 16, 1995

REVISED: Oct. 25, 1995:

- 1) changed clock to 50 MHz
- 2) changed so that end of record is generated automatically
- 3) changed starting frequency to 9 MHz (from 9MHz+3922 Hz)

Dec. 15, 1995

- 1) changed power up frequency to 9.5 MHz

```
*****
```

```
void main(void)
{
    FILE *fp,*fptr;
    int i,j, zero=0;
    int addrstart=4,bytecount=7;
    int increment=1e6/255 + 1; /*round up*/
    float bitreg=4.29497e9; /*=2^32*/
    unsigned long int clock=50e6;
    unsigned long int freqout[257],dphase[257],data_byte[257];
    int address[257],checksum[257],tempsum[257];
    int eprom_data[256];
    int temp,temp1,temp2,temp3;

    /* generate the address where it should go */
    address[1]=zero;
    for(i=2,i<=257;i++)
    {
        address[i]=addrstart + (i-2)*4;
    }

    /* generate the frequencies and corresponding delta phases
    for the delta phase register of the nco this will be
    the data byte field of the EPROM */

    fp=fopen("freq.dat","w");
    for (i=1,i<=256,i++)
```



```

        temp2=(short)temp3&255;
        temp=(temp1+temp2+tempsum[i]+3);
    }

    /*take lowest 8 bits and invert = ones compliment*/
    checksum[i]=~(short)temp&255;
}

/* now construct S records for the eprom*/

fptr=fopen("ncoeeprom.dat","w");

for (i=1; i<=256;i++)
{
    fprintf(fptr,"%s%X","S10",bytecount);
    if (address[i]<=15) /* pad address with 3 zeros*/
    {
        fprintf(fptr,"%X%X%X%X",zero,zero,zero,
            address[i]);
    }
    if (address[i]<=255 && address[i]>=16)
        /*pad with 2 zeros*/
    {
        fprintf(fptr,"%X%X%X",zero,zero,address[i]);
    }
    if(address[i]>255)/*pad with one zero*/
    {
        fprintf(fptr,"%X%X",zero,address[i]);
    }

    if(data_byte[i]<=15) /*pad with one zero*/
    {
        fprintf(fptr,"%X",zero);
    }
    fprintf(fptr,"%X",data_byte[i]);

    if(checksum[i]<=15) /*pad with one zero*/
    {
        fprintf(fptr,"%X",zero);
    }
    fprintf(fptr,"%X\n",checksum[i]);
}

/*now construct the end of file record*/
fprintf(fptr,"%s%X%X%X\n","S903",zero,
    address[257],checksum[257]);

fclose(fptr);
} /*end ncoeeprom.c*/

```

```

#include "stdio.h"
#include "stdlib.h"
#include "time.h"

/*****
This program generates data to program an EPROM. The data
is then used to program the STEL 1032 PRN Coder
S Record format is S1 BC AAAA H CC where
    S1 is record type
    BC is the byte count (3+ #data bytes)
    AAAA is the address
    H is the data byte
    CC is the checksum (one's complement binary addition
    of the preceding bytes in the record
NOTE everything is in hex notation

```

The data to initialize and configure each coder is explicitly declared. Thus, changing the output codes requires a change of the string variables below.

W P McPherson
Dec 7, 1995

```

*****/
void main(void)
{
    FILE *fptr,
    int i,j,k, zero=0,
    int addrstart=0,bytecount=7,
    int address[22],checksum[22],
    short tempsum,
    char *string1="8E000000", /*32 bits of mask, epoch
                                registers, coder 0*/
    char *string2="01000000", /*32 bits of init register,
                                coder 0*/
    char *string3="08000000", /*mux and ctl register data
                                coder 0*/
    char *string4="00000000", /*zeroed data, where needed*/
    char *string5="86020000", /*32 bits of mask,epoch,init
                                registers, coder 1*/
    char *string6="0A000000", /*mux and ctl register data
                                coder 1*/
    char *string7="0001150", /*32 bits of mask,epoch,
                                init registers,coder 2*/
    char *string8="1F000000", /*mux and ctl register data,
                                coder 2*/

    /* generate the addresses */
    address[1]=addrstart;
    for(i=2;i<=22,i++)
    {
        address[i]=addrstart + (i-1)*4,
    }

    /* now generate the checksum */
    for (i=1;i<=22,i++)
    {
        if(address[i]<8)
        {
            tempsum=(8*16)+14+address[i]+bytecount;
        }
        if(address[i]>=8)
        {
            tempsum=address[i]+bytecount+1,

```

S107010C2F67C54050
S10701102F6CE94023
S10701142F720D40F5
S10701182F773140C8
S107011C2F7C55409B
S10701202F8179406E
S10701242F869D4041
S10701242F8BC14014
S107012C2F90E540E7
S10701302F960940B9
S10701342F9B2D408C
S10701382FA051405F
S107013C2FA5754032
S10701402FAA994005
S10701442FAFBD40D8
S10701482FB4E140AB
S107014C2FBA05407D
S10701502FBB294050
S10701542FC44D4023
S10701582FC97140F6
S107015C2FCE9540C9
S10701602FD3B9409C
S10701642FD8DD406F
S10701682FDE014041
S107016C2FE3254014
S10701702FE84940E7
S10701742FED6D40BA
S10701782FF291408D
S107017C2FF7B54060
S10701802FFCD94033
S10701843001FD4005
S107018830072140D7
S107018C300C4540AA
S1070190301169407D
S107019430168D8010
S1070198301BB180E3
S107019C3020D580B6
S10701A03025F98089
S10701A4302B1D805B
S10701A8303041802E
S10701AC3035658001
S10701B0303A8980D4
S10701B4303FAD80A7
S10701B83044D1807A
S10701BC3049F5804D
S10701C0304F19801F
S10701C430543D80F2
S10701C830596180C5
S10701CC305E858098
S10701D03063A9806B
S10701D43066CD803E
S10701D8306DF18011
S10701DC30731580E3
S10701E030783980B6
S10701E4307D5D8089
S10701E8308281805C
S10701EC3087A5802F
S10701F0308CC98002
S10701F43091ED80D5
S10701F830971180A7
S10701FC309C35807A
S107020030A67D8023
S107020430ABA180F6
S107020830B0C580C9
S107020C30B5E9809C
S107021030BB0D806E
S107021430C0318041
S107021830C5558014
S107021C30CA7980E7
S107022030CF9D80BA
S107022430D4C1808D
S107022830D9F58060

S107022C30DF09R032
S107023030E42D8055
S107023430E95180D8
S107023430EE7580AB
S107023430F199807F
S107024030F88D8051
S107024430FDE18024
S107024831030580F5
S107024C31082980C8
S1070250310D40809B
S1070254311271806E
S10702583117958041
S107025C311C898014
S10702603121DDC6A7
S1070264312701C079
S1070268312C25C04C
S107026C313149C01F
S107027031366DC0F2
S1070274313B91C0C5
S10702783140B5C098
S107027C3145D9C06B
S1070280314AFDC03E
S1070284315021C010
S1070288315545C0E3
S107028C315A69C0B6
S1070290315F8DC089
S10702943164B1C05C
S10702983169D5C02F
S107029C316EF9C002
S10702A031741DC0D4
S10702A4317941C0A7
S10702A8317E65C07A
S10702AC318389C04D
S10702B03188A0C020
S10702B4318DD1C0F3
S10702B83192F5C0C6
S10702BC319819C098
S10702C0319D3DC06B
S10702C431A261C03E
S10702C831A785C011
S10702CC31ACA9C0E4
S10702D031B1C0C0B7
S10702D431B6F1C08A
S10702D831BC15C05C
S10702DC31C139C02F
S10702E031C65DC002
S10702E431CB81C0D5
S10702E831D0A5C0A8
S10702EC31D5C9C07B
S10702F031DAEDC04E
S10702F431E011C020
S10702F831E535C0F3
S10702FC31EA59C0C6
S107030031EF7DC098
S107030431F4A1C06B
S107030831F9C5C03E
S107030C31FEE9C011
S107031032040DC0E2
S1070314320931C0B5
S1070318320E55C088
S107031C321379C05B
S107032032189DC02E
S1070324321DC1C001
S10703283222E5C0D4
S107032C322809C0A6
S1070330322D7E0038
S1070334323252000B
S107033832377600DF
S107033C323C9A00B1
S107034032418E0084
S10703443246E20057
S1070348324C760029

S107034C32512A00FC
S107035032564E00CF
S1070354325B7200A2
S10703583260960075
S107035C3265BA0048
S1070360326ADE001B
S107036432700200ED
S107036832752600C0
S107036C327A4A0093
S1070370327F6E0066
S10703743284920039
S10703783289B600C
S107037C328EDA001
S10703803293FE00E2
S10703843299220084
S1070388329E460057
S107038C32A36A002A
S107039032A88E00FD
S107039432ADB200D0
S107039832B2D600A3
S107039C32B7FA0076
S10703A032BD1E0048
S10703A432C242001B
S10703A832C76600EE
S10703AC32CC8A00C1
S10703B032D1AE0094
S10703B432D6D20067
S10703B832DBF6003A
S10703BC32E11A000C
S10703C032E63E00DF
S10703C432EB6200B2
S10703C832F0860086
S10703CC32F5AA0058
S10703D032FACE002B
S10703D432FFF200FE
S10703D832051600CF
S10703DC323CA3A00A2
S10703E0320F5E0075
S10703E43214820048
S10703E83219A6001B
S10703EC321ECA00EE
S10703F0323EE00C1
S10703F4329120094
S10703F832E360066
S10703FC32335A0039
S9030400F8

```
#include "stdio.h"
#include "stdlib.h"
#include "time.h"
```

```
/*.....
```

This program generates data to program an EPROM. The data is then used to program the STEL 1032 PRN Coder.

S Record format is S1 BC AAAA H CC where

S1 is record type

BC is the byte count (3+#data bytes)

AAAA is the address

H is the data byte

CC is the checksum (one's complement binary addition of the preceding bytes in the record)

NOTE: everything is in hex notation

The data to initialize and configure each coder is explicitly declared. Thus, changing the output codes requires a change of the string variables below.

W.P. McPherson

Dec 7, 1995

```
/*...../
```

```
void main(void)
```

```
{
```

```
FILE *fptr;
```

```
int i,j,k, zero=0;
```

```
int addrstart=0,bytecount=7;
```

```
int address[22],checksum[22];
```

```
short tempsum;
```

```
char *string1="8E000000", /*32 bits of mask, epoch registers, coder 0*/
```

```
char *string2="01000000", /*32 bits of init register, coder 0*/
```

```
char *string3="08000000", /*mux and ctl register data, coder 0*/
```

```
char *string4="00000000", /*zeroed data, where needed*/
```

```
char *string5="86020000", /*32 bits of mask,epoch,init registers, coder 1*/
```

```
char *string6="0A000000", /*mux and ctl register data, coder 1*/
```

```
char *string7="00001150", /*32 bits of mask,epoch, init registers,coder 2*/
```

```
char *string8="1F000000", /*mux and ctl register data, coder 2*/
```

```
/* generate the addresses */
```

```
address[1]=addrstart;
```

```
for(i=2;i<=22,i++)
```

```
{
```

```
address[i]=addrstart + (i-1)*4;
```

```
}
```

```
/* now generate the checksum */
```

```
for (i=1;i<=22,i++)
```

```
{
```

```
if(address[i]<8)
```

```
{
```

```
tempsum=(8*16)+14+address[i]+bytecount;
```

```
}
```

```
if(address[i]>=8)
```

```
{
```

```
tempsum=address[i]+bytecount+1;
```

Appendix C: Software Listing For Dynamic Matched Filter

```

    }

    if (address[i] == 12) /*coder0 count reg*/
    {
        fprintf(ptr,"%s% X%s% X\n", "000",address[i],string4,
            checksum[i]);
    }
    if (address[i] == 16) /*coder0 mux,ctl regs*/
    {
        fprintf(ptr,"%s% X%s% X\n", "00",address[i],string3,
            checksum[i]);
    }
    if (address[i] >=17 && address[i]<32) /*coder0 dont cares*/
    {
        fprintf(ptr,"%s% X%s% X\n", "00",address[i],string4,
            checksum[i]);
    }
    if (address[i] >= 32 && address[i]<44)
        /*coder1 mask epoch mit registers*/
    {
        fprintf(ptr,"%s% X%s% X\n", "00",address[i],string5,
            checksum[i]);
    }
    if (address[i]==44) /*coder1 count*/
    {
        fprintf(ptr,"%s% X%s% X\n", "00",address[i],string4,
            checksum[i]);
    }
    if (address[i]==48) /*coder1 mux,ctl*/
    {
        fprintf(ptr,"%s% X%s% X\n", "00",address[i],string6,
            checksum[i]);
    }
    if (address[i] >49 && address[i]<64) /*coder2 dont cares*/
    {
        fprintf(ptr,"%s% X%s% X\n", "00",address[i],string4,
            checksum[i]);
    }
    if (address[i] >=64 && address[i]<76)
        /*coder2 mask epoch mit registers*/
    {
        fprintf(ptr,"%s% X%s% X\n", "00",address[i],string7,
            checksum[i]);
    }
    if (address[i]==76) /*coder2 count reg*/
    {
        fprintf(ptr,"%s% X%s% X\n", "00",address[i],string4,
            checksum[i]);
    }

    if (address[i] ==80) /*coder2 mux,ctl regs*/
    {
        fprintf(ptr,"%s% X%s% X\n", "00",address[i],string8,
            checksum[i]);
    }
}

/*construct end of file record*/
fprintf(ptr,"%s% X% X\n", "S90300",address[22],checksum[22]);

fclose(ptr);
} /*end pineprom c*/

```


OUTPUT FILE PRNEPROM.DAT

S10700008E0000006A
S107000048E00000066
S1070000801000000EF
S1070000C00000000EC'
S107001000000000E0
S107001400000000E54
S107001800000000E0
S107001C0000000019C'
S10700208602000050
S1070024860200004C'
S10700288602000048
S107002C00000000CC'
S10700300A000000BE
S10700340000000C'4
S10700380000000C'0
S107003C00000000BC'
S10700400000115057
S10700440000115053
S1070048000011504F
S107004C00000000AC'
S10700501F00000089
S9030054A8

Appendix C: Software Listing For Dynamic Matched Filter

```
/* General Definitions for spread spectrum demod */
```

```
/* gender h */
```

```
#define FSIZE 150000
```

```
#define LENGTH 1024
```

```
#define FFLENGTH 8192
```

```
#define PI (float)acos(-1.0)
```

```
#define SAMPSPERCHIP 3
```

```
#define NUMHOPS 1
```

```
#define NUMBITS 6
```

```
#define HOPSPACE 3922
```

```
void ruseq(float *x, float *y, int n);
```

```
void makehop(int *x);
```

```
void shift(float *x, unsigned int n, unsigned int m)
```

```
void zeropad(float *x, float *y, unsigned int n, unsigned int m)
```

```
void reformat(float *w, float *x, float *y, float *z,
```

```
            unsigned int n);
```

```
void getnextbit(float *w, float *x, float *y, float *z,
```

```
            unsigned int n, unsigned int m);
```

```
int bitpos(int k, int l);
```

```
void bitrev(float *x, float *y, unsigned int n);
```

```
void fff(float *x, float *y, unsigned int n)
```

```
void fswap(float *x, float *y);
```

```
void conjugate(float *x, int y);
```

```
void multiply(float *x1, float *x2, float *y1, float *y2,
```

```
            float *z1, float *z2, unsigned int n)
```

```
void rfft(float *x, float *y, unsigned int n);
```

```
void maketfreq(float *x, float *y);
```

```
void findmaxmin(float *x, float *n, float *m, unsigned int n);
```

```
void scale(float *x, float *y, float *z, unsigned int n)
```

```
void decimate(float *w, float *x, float *y, float *z,
```

```
            unsigned int n);
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "pendef.h"
```

```
/* demod.c
```

This program despreads sampled spread spectrum data using a dynamic matched filter algorithm. An FFT is performed on the sampled data and stored reference. The FFT size is given by FFTLENGTH. The complex conjugate of the stored reference is determined. The results are multiplied followed by an inverse FFT. The filter taps are then readjusted for the next data bit.

Once complete, the despread data is written to a data file, corrin.dat, for later examination.

Wayne McPherson
Feb. 9, 1996

```
*/
```

```
main()
{
```

```
FILE *fp;
char filename1[10], filename2[10];
int i, j, k, z, initphase, hopnum, numshifts,
int chipsperbit, nextphase, startpoint, endpoint;
float max, min, sum;
unsigned short int infile1[FSIZE], infile2[FSIZE], a, b;
float realkdata1[FSIZE], imdata1[FSIZE], /*raw data*/
float realkdata2[FSIZE], imdata2[FSIZE],
float rdata1[LENGTH], idata1[LENGTH],
float realprod[FFTLENGTH*2], improd[FFTLENGTH*2],
float rcode[LENGTH], imcode[LENGTH];
float rtmpcode[LENGTH], imtmpcode[LENGTH];
float srcode[FFTLENGTH], simcode[FFTLENGTH],
float bitstream[FFTLENGTH*64],
int hopseq[NUMHOPS],
```

```
/*generate the hopping sequence*/
makehop(hopseq);
```

```
/*Read in the first stored data file*/
```

```
printf("Enter the data file name #1 including extension ");
scanf("%s", filename1);
fp=fopen(filename1, "r");
if (fp==NULL)
{
    printf("unable to open %s\n", filename1);
    exit(-1);
}
fseek(fp, 256, 0); /*ignore 256 byte DA60 header */
fread(infile1, sizeof(unsigned short int), FSIZE, fp);
fclose(fp);
```

```
/* now do a byte swap to change from pc int to unix int
and cast as floating point to use in fft routines*/
```

```
(p=fopen("sampled.dat", "w"),
```

```

for(i=0,i<=(FSIZE-1),i++)
{
    a=infile1[i]<<8,
    b=infile1[i]>>8,
    infile1[i]=a/b,
    realkdata1[i]=(float)infile1[i],
    fprintf(fp,"%f\n",realkdata1[i]),
    imdata1[i]=0,
}
fclose(fp).

/* loop until all hops completed */

for(hopnum=0,hopnum<NUMHOPS,hopnum++)
{
    /*calculate chips per bit*/

    if(hopseq[hopnum]==0 || hopseq[hopnum]==254)
    {
        chipsperbit=(int)((9e6+255*HOPSPACE)/12200),
    }
    else if(hopseq[hopnum]==1)
    {
        chipsperbit=(int)((9e6+127*HOPSPACE)/12200),
    }
    else if(hopseq[hopnum]>=127 && hopseq[hopnum]!=254)
    {
        chipsperbit=(int)((9e6+(hopseq[hopnum]+1)
                                *HOPSPACE)/12200),
    }
    else
    {
        chipsperbit=(int)((9e6+(hopseq[hopnum]-1)
                                *HOPSPACE)/12200),
    }

    /*generate m-sequence*/
    mseq(rcode,rcode,LENGTH),

    /*shift, scale, zeropad*/
    initphase=265,
    shift(rcode,initphase,LENGTH),
    reformat(rcode,rcode,srcode,simcode,LENGTH),
    zeropad(srcode,simcode,chipsperbit*SAMPSPERCHIP,
                                                    FFTLENGTH),
    sum=0,

    /*repeat until all data bits in burst have
    been despread*/

    for(j=0,j<NUMBITS,j++)
    {
        getnextbit(realkdata1,imdata1,realkdata2,imdata2,
                    j*chipsperbit*SAMPSPERCHIP,
                    chipsperbit*SAMPSPERCHIP),
        max=0,min=4096,
        findmaxmin(realkdata2,&max,&min,
                    SAMPSPERCHIP*chipsperbit),
        /*scale to -1<= x <=1 */
        scale(realkdata2,&max,&min,
                    SAMPSPERCHIP*chipsperbit),
        zeropad(realkdata2,imdata2,chipsperbit*
                    SAMPSPERCHIP,FFTLENGTH),

        /* do correlation */

```

```

fft(srccode,simccode,FFTLENGTH);
conjugate(simccode,FFTLENGTH);
fft(realdata2,imdata2,FFTLENGTH);
multiply(realdata2,imdata2,srccode,simccode,
         realprod,improd,FFTLENGTH);
ifft(realprod,improd,FFTLENGTH);

/*store result*/
startpoint=j*chipsperbit*SAMPSPERCHIP,
k=0;
printf("startpoint is %i\n",startpoint);
for(i=startpoint,i<FFTLENGTH+startpoint,i++)
{
    bitstream[i]=realprod[k++];
}

/*change filter taps*/
shift(rcode,chipsperbit+1,LENGTH);
reformat(rcode,imcode,srccode,simccode,LENGTH);
zeropad(srccode,simccode,chipsperbit*
        SAMPSPERCHIP,FFTLENGTH);

printf("done bit %i\n",i);
}/*end for*/
printf("done hop %i\n",hopnum);

}/*end for*/

/*save to file*/
fp=fopen("corrln.dat","w");
for(i=0,i<FFTLENGTH*NUMBITS*NUMHOPS,i++)
{
    fprintf(fp,"%f\n",bitstream[i]);
}
printf("wrote %i samples\n",i);
fclose(fp);

} /*END MAIN*/

void makehop(int *x)
/*
    routine to generate the hop pattern
    x - array to contain the hop pattern
*/
{
    unsigned short int ssrg[8]={0,1,1,1,0,0,0,1};
    unsigned short int i,j,fb1,fb2,fb3,tmp1,tmp2,tmp3,tmp;
    unsigned short int prev=1;
    int count;
    FILE *fp;

    for(i=0,i<NUMHOPS*2+1;i++)
    {
        tmp1=ssrg[7]; /*shift bit out*/
        tmp2=prev<<1; /*left shift previous
                        word by 1*/
        tmp3=tmp2|tmp1; /*or it with bit shifted
                        out*/
        prev=tmp3 & 255; /*and with 255 to zero
                        upper 8 bits*/
    }

```

```

        x[i]=prev;
        fb1=srg[7]^srg[3]; /*xor feedback bits*/
        fb2=fb1^srg[2];
        fb3=fb2^srg[1];
        for(j=7;j>=1;j--)
        {
            srg[j]=srg[j-1];
        }
        srg[0]=fb3;
    }
} /*end makehop*/

void msecq(float *x, float *y, int a)

/* routine to generate the m=1023 sequence used for demod
feedback taps [10 8 3 2] are used
x- real part of msecq
y- imag part of msecq
*/

{
    short int srg[10]={0,1,1,0,0,0,0,1,0,1};
    short int i,j,fb1,fb2,fb3,tmp;

    for(i=0;i<a;i++)
    {
        tmp=srg[9];
        if(tmp==1) x[i]=-1;
        else x[i]=1; /*scale to 1 and -1
                     for analog mult */

        y[i]=0;
        fb1=srg[9]^srg[7];
        fb2=fb1^srg[2];
        fb3=fb2^srg[1];
        for(j=9;j>=1;j--)
        {
            srg[j]=srg[j-1];
        }
        srg[0]=fb3;
    }
} /*end msecq*/

void shift(float *x, unsigned int n, unsigned int m)
/*
routine to shift array x by n positions to the left
m- number of elements in the array
*/

{
    int i,j;
    float tmp;

    for(i=0;i<n;i++)
    {
        tmp=x[0];
        for(j=0;j<m;j++)
        {
            x[j]=x[j+1];
        }
        x[m]=tmp;
    }
} /*end shift*/

```

```

void zeropad(float *x, float *y, unsigned int n, unsigned int m)
/*
    routine to add zeros to the end of x and y
    n - point to start adding zeros
    m - stop point
*/
{
    int i;

    for(i = n; i < m; i++)
    {
        x[i] = 0;
        y[i] = 0;
    }
} /*end zeropad*/

```

```

void reformat(float *w, float *x, float *y, float *z, unsigned int n)
/*
    routine to format an array to a sampling rate
    of SAMPSPERCHIP
    w,x - the real and imaginary unsampled arrays
    y,z - the real and imaginary sampled arrays
    n - number of chips to sample
*/
{
    int i, j, k = 0;

    for(i = 0; i < n; i++)
    {
        for(j = 0; j < SAMPSPERCHIP; j++)
        {
            y[k] = w[i];
            z[k] = x[i];
            k++;
        }
    }
} /*end reformat*/

```

```

void getnextbit(float *w, float *x, float *y, float *z,
                unsigned int n, unsigned int m)
/*
    routine to fill arrays y and z with elements from arrays
    w and x
    n - starting point in arrays x and y
    m - stop point in array x and y
*/
{
    int i, j = 0;

    for(i = n; i < m; i++)
    {
        y[j] = w[i];
        z[j] = x[i];
        j++;
    }
} /*end getnextbit*/

```

```

int bitpost(int k, int l)

```

```

/* k - position to be bit reversed

```



```

    1 - number of bits to represent position
*/

{
    register int i,mask,reversed;

    mask=1;
    reversed=0;
    for(i=0,i<1,i+=mask<<=1)
        reversed=(reversed<<1)|(i&mask)^(i/2);
    return(reversed);
}

void bitrev(float *x, float *y, unsigned int n)

/*
    Bit Reversal Subroutine (Floating Point)
    This routine places the numbers stored in the arrays x
    and y in bit reversed order. The bit reversal is carried
    out in place.

    x, y - pointers to arrays of double precision numbers
    n - number of points
*/

{
    int m;
    register int i,j;

    m=(int)(log((double)n)/log(2.0)+0.5);
    /* Factor of 0.5 is needed if the log functions
    get evaluated to a number slightly less than their
    true value */

    for(i=0,i<n,i++)
    {
        j=bitpost(i,m);

        if(i>j)
        {
            fswap(x+i,x+j);
            fswap(y+i,y+j);
        }
    }
} /*end bitrev*/

void fft(float *x, float *y, unsigned int n)

/* This routine performs a complex in place Fast Fourier Transform
based on a decimation in time algorithm

    x, y - arrays of double precision numbers
    n - number of points in the FFT (must be a power of 2)
*/

{
    int rep,disp,m,j2;
    register int i,j,k;
    float pi,arg,tw1,t1,t2;

    pi=(float)(acos(-1.0));
    bitrev(x,y,n); /*Perform Bit Reversal */

    m=(int)(log((double)n)/log(2.0)+0.5)

```

```

/* Factor of 0.5 is needed if the log functions get evaluated
to a number slightly less than their true value */

for(i=0;i<n,i++)
{
    rep=(int)(pow(2.0,(double)(i+1))-0.5);
    disp=rep/2;
    arg=2*pi/rep;
    for(j=0;j<disp,j++)
    {
        twl=j*arg;
        for(k=j,k+=rep)
        {
            j2=k+disp;
            t1=(float)t*cos((double)twl*x[j2]
                        +sin((double)twl*y[j2]));
            t2=(float)t*(-1.0*sin((double)twl*x[j2]+
                        cos((double)twl*y[j2]));
            x[j2]=x[k]+t1;
            y[j2]=y[k]+t2;
            x[k]=x[k]+t1;
            y[k]=y[k]+t2;
        }
    }
}
/*end fft*/

void fswap(float *x, float *y)
{
    float temp;
    temp=*x;
    *x=*y;
    *y=temp;
} /*end fswap*/

void conjugate(float *x, int y)

/* routine to generate the conjugate of x
   x= array of floating point numbers
   y= number of elements in the array
*/

{
    int i;

    for(i=0;i<y-1,i++)
    {
        if(x[i]>=0) x[i]=-0.5*x[i];
        else x[i]=abs(x[i]);
    }
} /*end conjugate*/

void multiply(float *x1, float *x2, float *y1, float *y2,
            float *z1, float *z2, unsigned int n)

/* routine to find the product of 2 fft's
   x1,x2= fft1, real and im
   y1,y2= fft2, real and im
   z1,z2= product of fft1xfft2
*/

```

```

{
    int i;
    float real1, real2, im1, im2, tmp;

    for(i=0; i<n-1; i++)
    {
        real1=x1[i]*y1[i];
        tmp=x2[i]*y2[i];
        if(tmp > 0) real2=0+tmp;
        else real2=abs(tmp);
        z1[i]=real1+real2;
        im1=x2[i]*y1[i];
        im2=y2[i]*x1[i];
        z2[i]=im1+im2;
    }
} /*end multiply*/

void ifft(float *x, float *y, unsigned int n)

/* This routine performs a complex in place Inverse
   Fast Fourier Transform based on a decimation in time algorithm
   x and y arrays of double precision numbers
   n = number of points in the FFT (must be a power of 2)
*/

{
    int rep, disp, m, j2;
    register int i, j, k;
    float pi, arg, tw1, t1, t2;

    pi=(float)acos(-1.0);
    bitrev(x,y,n); /*Perform Bit Reversal */

    m=(int)(log((double)n/log(2.0))+0.5);
    /* Factor of 0.5 is needed if the log functions get evaluated
    to a number slightly less than their true value */

    for(i=0; i<m; i++)
    {
        rep=(int)(pow(2.0,(double)(i+1))+0.5);
        disp=rep/2;
        arg=2*pi/rep;
        for(j=0; j<disp; j++)
        {
            tw1=j*arg;
            for(k=j; k<n; k+=rep)
            {
                j2=k+disp;
                t1=(float)(cos((double)tw1)*x[j2]-
                           sin((double)tw1)*y[j2]);
                t2=(float)(sin((double)tw1)*x[j2]+
                           cos((double)tw1)*y[j2]);
                /* t1 and t2 change for inverse fft */

                x[j2]=x[k]-t1;
                y[j2]=y[k]-t2;
                x[k]=x[k]+t1;
                y[k]=y[k]+t2;
            }
        }
    }
    /* Final Results Must Be Divided By n For Inverse FFT */
    for(i=0; i<n; i++)

```

```

    {
        x[i] = x[i]/n;
        y[i] = y[i]/n;
    }
} /*end fff*/

void findmaxmin(float *x, float *n, float *m, unsigned int z)
/*
    routine to find the max value in an array
    x: the array
    n: the maximum value
    m: the minimum value
    z: the number of elements in the array
*/
{
    int i, nposit, npos;
    float maxval = 4096, minval = 4096, tmp;

    for(i = 0; i < z; i++)
    {
        if(x[i] > maxval) {maxval = x[i]; nposit = i;}
        if(x[i] < minval) {minval = x[i]; npos = i;}
    }
    *n = maxval;
    *m = minval;
} /*end findmaxmin*/

void scale(float *x, float *y, float *z, unsigned int n)
/*
    routine to scale array x between -1 and 1
    y is the max value in the array
    z is the min value in the array
    n is array length
*/
{
    int i;
    float average, a;
    FILE *fp;

    average = (*y + *z)/2;
    a = (average - *z);

    for(i = 0; i < n; i++)
    {
        x[i] = 0 - ((x[i] - average)/a);
    }
} /*end scale */

```

REFERENCES

- [1] Rustad, J.E., Skaug, R., and A. Aasen, "New Radio Networks for Tactical Communication", *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 5, pp. 713-727, 1990.
- [2] Fitts, R.E., *The Strategy of Electromagnetic Conflict*, Los Altos, CA: Peninsula Publishing, 1980.
- [3] Nicholson, D.L. *Spread Spectrum Signal Design: LPE and AJ Systems*, Rockville, Maryland: Computer Science Press, 1988.
- [4] Skaug, R., and J.F. Hjelmstad, *Spread Spectrum in Communication*, London: Peter Peregrinus Ltd., 1985.
- [5] Pickholtz, R.L., Schilling, D.L., and L.B. Milstein, "Theory of Spread Spectrum Communications - A Tutorial", *IEEE Trans. Comm*, vol. 30, no. 5, pp. 855 - 883, 1982
- [6] Tsui, T.S.D., and T.G. Clarkson, "Spread Spectrum Communication Techniques", *Electronics and Communication Engineering Journal*, pp. 3 - 12, February 1994.

- [7] Torrieri, D.J., *Principles of Secure Communication Systems*, Dedham, MA: Artech House, 1985.
- [8] Dixon, R.C., *Spread Spectrum Systems*, New York: John Wiley and Sons, 1976.
- [9] Simon, M.K., Omura, J.K., Scholtz, R.A. and B.K. Levitt, *Spread Spectrum Communications, vols. 1-3*, Rockville, Maryland: Computer Science Press, 1985.
- [10] Scholtz, R.A., "The Origins of Spread-Spectrum Communications", *IEEE Transactions on Communications*, vol. 30, no.5, pp. 822-854, 1982.
- [11] Ziemer, R.E. and R.L. Peterson, *Digital Communications and Spread Spectrum Systems*, New York: Macmillan Publishing Company, 1985.
- [12] Golomb, S.W., *Shift Register Sequences*, San Fransisco, CA: Holden Day, 1967.
- [13] Gold, R., "Maximal Recursive Length Sequences With 3-Valued Recursive Cross-Correlation Functions", *IEEE Transactions on Information Theory*, Vol. IT-14, pp. 154-156, 1968.
- [14] Sarwate, D.V., and M.B. Pursley, "Crosscorrelation Properties of Pseudorandom and Related Sequences", *Proceedings of the IEEE*, vol. 68, pp. 593-619, 1980.

- [15] Olson, J.D., Scholtz, R.A., and L.R. Welch, "Bent Function Sequences", *IEEE Transactions on Information Theory*, vol. IT-28, pp. 858-868, 1982.
- [16] Reed, D.E., "Comparison of Symbol-Rate Detector and Radiometer Intercept Receiver Performances In A Nonstationary Environment", *Proceedings of MILCOM '89*, Vol. 19, pp. 19.5.1-19.5.5, 1989.
- [17] Hill, D.A., and E.B. Felstead, "Laboratory Performance of Spread Spectrum Detectors", *IEE Proc.-commun.*, vol.142, no. 4, pp. 243 - 249, Aug 1995.
- [18] Imbeaux, J.C. "Performances of the Delay-Line Multiplier Circuit for Clock and Carrier Synchronization in Digital Satellite Communications", *IEEE Journal on Selected Areas in Communications*, vol. 1, no. 1, pp. 82-95, 1983.
- [19] Kuehls, J.F. and E. Geraniotis, "Presence Detection of Binary-Phase-Shift-Keyed and Direct-Sequence Spread-Spectrum Signals Using a Prefilter-Delay-and -Multiply Device", *IEEE Journal on Selected Areas In Communications*, vol. 8, no. 5, pp. 915- 933, 1990
- [20] Spooner, C.M., and W.A. Gardner, "Robust Feature Detection For Signal Interception", *IEEE Transactions on Communications*, vol. 42, no. 5, pp. 2165-2173, 1994.
- [21] Myers, G.A., "Performance of Envelope Detectors in Direct Sequence Spread

Spectrum Systems", *RF Design*, vol. 13, no. . pp. 68-71, 1990.

[22] Feher, K , and G.S. Takhar, "A New Symbol Timing Recovery Technique For Burst Modem Applications", *IEEE Transactions On Communications*, vol. COM. 26, pp. 100-108, 1978.

[23] D'Andrea, A.N., and U. Mengali, "Performance Analysis of the Delay Line Clock Regenerator", *IEEE Transactions on Communications*, vol. 34, no. 4, pp. 321 - 328, 1986.

[24] Gardner, W.A, and C.M. Spooner, "Signal Interception: Performance Advantages of Cyclic Feature Detectors", *IEEE Transactions on Communications*, vol. 40, no. 1, pp. 149-159, 1992.

[25] Reed, D.E. and M.A. Wickert, "Minimization of Detection of Baud Rate Spectral Lines By Delay and Multiply Receivers", *Proceedings of MILCOM '86*, pp. 10.3.1-10.3.5, 1986.

[26] Leon-Garcia, A., *Probability and Random Processes for Electrical Engineering*, New York: Addison Wesley Publishing Company, 1994.

[27] Press, W.H. et al. *Numerical Recipes in C*, New York: Cambridge University Press, 1988.

[28] Stanley, W.D., Dougherty, G.R., and R. Dougherty, *Digital Signal Processing*, Restville, Virginia: Reston Publishing Company Inc., 1984.

[29] McPherson, W.P., Hill, D.A., and J.S. Wight, "Chip Rate Hopping Reduces the Detectability of Direct Sequence Spread Spectrum Signals", *Proceedings on the 18th Biennial Queen's Symposium on Communications*, pp. 216 - 220, 1996.

END

1 8 - 0 6 - 9 7

FIN