ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF TECHNOLYGY

Frequency Hop Spread Spectrum Communication System

BY

Endale Befekadu Demena

January 2003

ADISS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF TECHNOLGY

ELECTRICAL ENGINEERING DEPARTEMENT

Frequency Hop Spread Spectrum System

Communication System

A THESIS SUBMITTED TO SCHOOL OF GRADUTES STUDIES OF ADDIS ABABA UNIVERSITY IN PARTIAL FULFILMENT FOR THE DEGREE OF MASTER OF SCIENCE IN ELECTRIACL ENGINEERING

BY

Endale Befekadu Demena

Advisor

Prof.Devrajan Gopal

Addis Ababa

University

January 2003

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

"Frequency Hop Spread Spectrum Communication System"

By

Endale Befekadu

Faculty of Technology

Approval by Broad of Examiners

_____

Chairman of the Examination Committee

_____

Adisor

_____

Examiner

_____

External Examiner

Acknowledgement

I would like to thank Prof.Devrajan for giving me the initial guide lines for framing the structures of the project. He has told me which parts to include and which part to ignore. Also has corrected the final draft of the project through e-mail. With out his advice the project could not have been successful.

Also I would like to thank the system administrators for the invaluable help they gave me during the preparation the project. I would also thank the Librarians for giving me the books I needed for the project.

I am also indebted to Dr.Mohamed Abdo the current department head for for giving me the correct information about the schedule .He has guided me how to program through my final thesis.

Finally I would like to thank my instructors for showing me the right way to succeed in Electrical Enginnering Field.The computer laboratory technicians were also very helpful for opening the computer room when there was no key.

Tables of Contents

List of Figures

List of tables

Abstract

Among the various types of the spread spectrum systems the slow frequency hop spread spectrum system is discussed.In the transmitter the data input ,which is, text is converted to binary data by the ASCII code conversion table.This binary data is modulated using the the FSK modulator.Then pseudo-random frequencies are generated from pseudo-random code generators and a frequency synthesizer. Using these generated frequencies the modulator output signal is hopped or spread to different frequencies. After the spreading process, the final signal output is then transmitted through a non Gaussian plus Gaussian noise channel. At the receiver before the demodulation process the initial estimate of the receiver phase is made using the sequential detection method. Using the intial estimate of the phase the rest of the received phase is tracked using a second order tracking loop. After tracking the phase at the receiver end the next step is to dispread the frequency hopped signal. Then demodulation is carried out using the matched filter FSK detector .After detection the data output of the demodulator is converted back to text using the ASCII code table. Finally the receiver gets back the text data which was sent at the transmitter

1. Description about the Project

1.1 GENERAL BACKGRAOUND

The modulation - demodulation techniques so widely used are designed to communicate digital information from one place to another as efficiently as possible in a stationary additive white Gaussian noise (*AWGN*) environment .The transmitted signals are selected to be relatively efficient in their use of the communication resources of *power* and *bandwidth.* The demodulations are designed to yield minimum bit error probability for the given transmitted signal power in *AWGN*.

Although many real -system communication channels are accurately modeled as stationary *AWGN* channels, there are other important channels that do not fit the model. For example, concider a military communication system that can be jammed by a continuous wave (*CW*) tone near the modem's center frequency or by the distorted retransmission of the modem's own signal .The interference can not be modeled as stationary *AWGN* in either of these cases. Another jammer may transmit *AWGN*, but the jamming signal may be pulsed and therefore not stationary. Another type of interference, which does not fit the stationary *AWGN* model, occurs when there are multiple propagation paths between the transmitter and receiver.The modem then interferes with itself via a delayed reception of its own signal .This phenomenon is called *multipath reception* and is a problem in line-of-sight microwave digital radios such as those used for long-haul telephone transmission and in urban mobile radio, among other places .Spread communication system is a modulation and demodulation technique that can be used as aid mitigating the deleterious effect of the the types of interferences described above. The transmission bandwidth employed is much greater than the minimum bandwidth required to transmit the digital information.

To be classified as spread spectrum system, the modem must have the following characteristics:

- The transmitted signal energy must occupy a bandwidth which is larger than the information bit rate (usually much larger) and which is independent of the information bit rate:
- Demodulation must be accomplished ,in part, by correlation of the received signal with replica of the signal used in the transmitter to spread the information signal.

A number of modulation techniques use a transmission bandwidth much larger than the minimum required for data transmission but are not spread spectrum modulations. *Low-rate coding*, for example, results in increased transmission and width but does not satisfy either of the the conditions above. *Wideband frequency modulation* also results a large transmission bandwidth but is not spread spectrum communication system. Spread spectrum technique can be very useful in solving a wide range of communication problems.

1.1 Low Probability of Detection

The two most common spread spectrum problems are THE PULSE NOIE JAMMING PPROBLEM AND LOW PROBABILTY OF DETECTION (LPD) problem the one that is considered here is the LPD,but the first one is discussed in Appendix D.

Situation exist where it desirable that communication link be operated without knowledge of certain parties .***Low probability of detection (LP*D)** ,communication system are designed to make their detection as difficult as possible by anyone but the intended receiver .This of ,course implies that the minimum signal power required to achieve a particular performance is used as the goal of the LPD system designer.It is a signaling scheme that results in the minimum probability of being detected within some time interval .Spread spectrum techniques can significantly aid the system designer in achieving this goal.

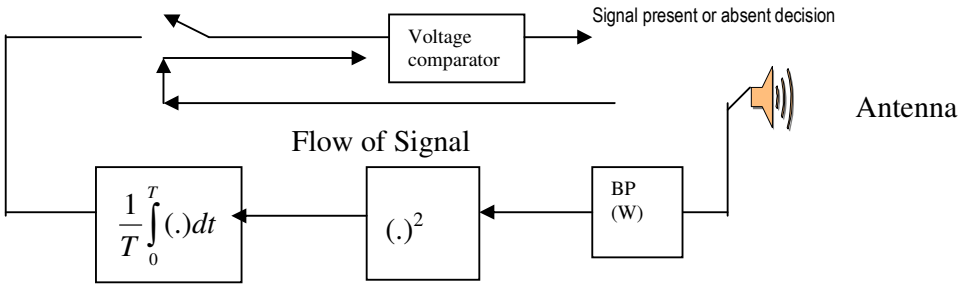


Figure.1.1.Energy detector or radiometer.

Assume that the detector is using a radiometer. A radiometer detects energy received in bandwidth W .By filtering to this bandwidth, it squares the output of this filter.Then it integrates the output of the squarer for a time interval T.Having done that it compares the intergrater output with a thresh hold.The signal is declared present,if the integrator output is higher than the thershhold; otherwise, the signal is declared absent. The performance of the radiometer in detecting the desired communication signal is known if the probability density function of the integrator output at time t is determined .This probability density function is used to calculate the probability , P_d , of detecting the signal if it is indeed pesent,and probability of falsely declaring a detection when noise alone is present , P_{fa} .Two approximations are often used for the integrator output PDF. The first one , and which is considered here is , when the *time bandwidth product* TW is large relative to the received energy to noise power spectral density ratio $\frac{E}{N_0}$,Then it is to approximate the pdf as Gaussian .In this case , P_d , P_{fa} ,TW,and E/N_0 are related by

$$P_D = \Phi\left\{\left[\frac{P}{N_0}\sqrt{\frac{T}{W}} - \Phi^{-1}\left(1 - P_{fa}\right)\right]\right\} \qquad \text{----------1-6}$$

where

$$\Phi(y) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{y} e^{\left(-\frac{1}{2}\xi^2\right)} d\xi \qquad \text{----------1-7}$$

and $\Phi^{-1}(x)$ is equal to the variate y such that Φ (y) = x. For a fixed P_{fa} the probability of detection can be made smaller by reducing $\frac{P}{N_0}$ or increasing W. Integration time, T, is controlled by the detector, but W can be increased using spread-spectrum techniques. Thus another important application for spread spectrum is the reduction of signal detectability for fixed SNR, integration time, and detector false-a1arm probability.

1.2. FREQUENCY- HOP SPREAD SPECTRUM

A method for widening the spectrum of a data-modulated carrier is to change the frequency of the carrier periodically. Typically, each carrier frequency is chosen from a set of 2^k frequencies *which are spaced approximately the width of the data modulation spectrum apart*, although neither condition is absolutely necessary .The spreading code in this case does not directly modulate the data-modulated carrier but is instead used to control the sequence of carrier frequencies. Because the transmitted signal appears as a data-modulated carrier which is hopping from one frequency to the next, this type of spread spectrum is called *frequency-hop* {FH) spread spectrum. In the receiver, the frequency hopping is removed by mixing (down-converting) with a local oscillator signal which is hopping synchronously with the received signal.

1.3. Coherent Slow-Frequency-Hop Spread Spectrum

Consider, for example, the FH system shown in **Figure 1-2**. The frequency synthesizer output is a sequence of tones of duration T_c so $h_T(t)$ can be written

$$h_T(t) = \sum_{n=-\infty}^{\infty} 2p(t - nT_c)\cos(\varpi_n t + \phi_n) \text{ ----------(1-8)}$$

where p(t) is a unit amplitude pulse of duration T_c starting at time zero, and ϖ_n and φ_n are the radian frequency and phase during the n th frequency-hop interval.The frequency ϖ_n is taken from a set of 2^k frequencies. The spreading code here is used k bits at a time. The

transmitted signal is the data-modulated carrier up-converted to a new frequency $\varpi_m + \varpi_n$ where m=1,2 on each FH chip,

$$s_t(t) = \left[s_d(t) \sum_{n=-\infty}^{\infty} 2p(t-nT_c)\cos(\varpi_n t + \varphi_n) \right]_{\substack{sumfeq. \\ components}} \text{----------}. \quad (1\text{-}9)$$

Calculation of the transmitted power spectrum is accomplished using the frequency convolution theorem of Fourier theory. Define $S_d(f)$ to be the power spectral density of the data-modulated carrier and $S_h(f)$ to be the power spectral density of the hop carrier $h_T(t)$. These two signals are independent, so that the power spectrum of the of the transmitted signal is the sum frequency term of the convolution of $S_d(f)$ with $S_h(f)$. The signal $h_T(t)$ may or may not be periodic. In most cases, if $h_T(t)$ were periodic, its period would be sufficiently long that little error would be made in considering the period infinite. This assumption is made in the following. Thus $h_T(t)$ is considered a purely random sequence of frequencies. For the coherent frequency-hop system being considered, the same phase φ_m is used each time $h_T(t)$ returns to frequency ϖ_m, that is $\varphi_n \in \{\varphi_m, m=1,2,3,\ldots,2^K\}$. is used each time $h_T(t)$. With these assumptions, $S_h(f)$ is given by

$$S_h(f) = \frac{1}{T^2_c} \sum_{n=-\infty}^{\infty} \left| \sum_{m=1}^{2^k} p_m G_m(n/T_c) \right|^2 \delta(f - \frac{n}{T_c})$$

$$+ \frac{1}{T^2_c} \sum_{m=1}^{2^k} p_m(1-p_m) \left| G_m(f) \right|^2 - \frac{2}{T} \sum_{m=1}^{2^k} \sum_{m'=1}^{2^k} p_m p_{m'} \mathrm{Re}\left[G_m(f) G_{m'}^{*}(f) \right] \quad ,m \neq m', m<m' \quad \text{----}$$

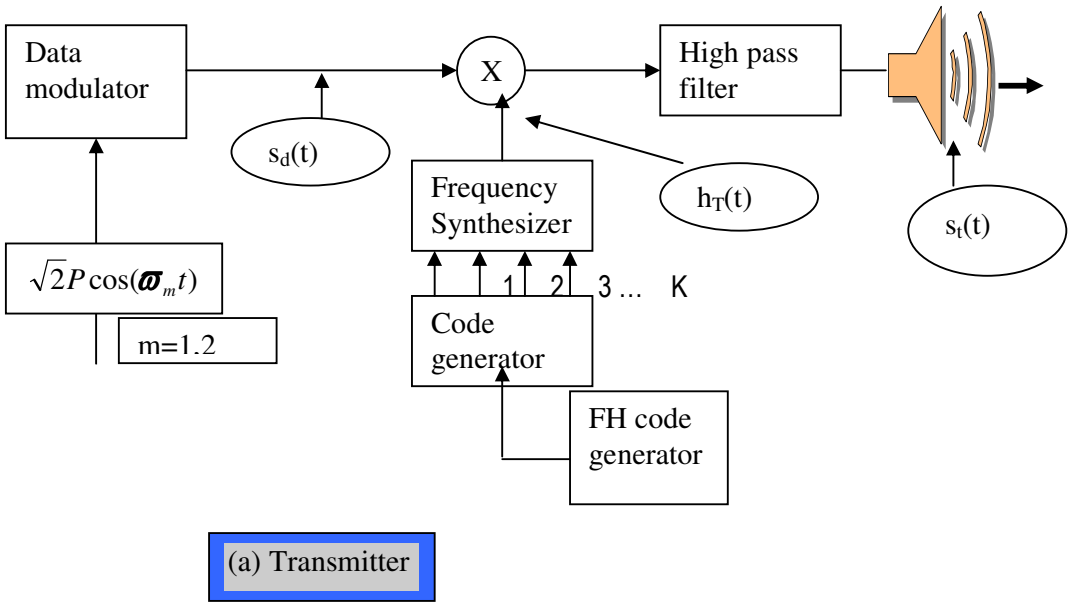
(1.10.a)

where p_m is the probability that frequency m is selected, and $G_m(f)$ is the Fourier transform of $g_m(t)$, where

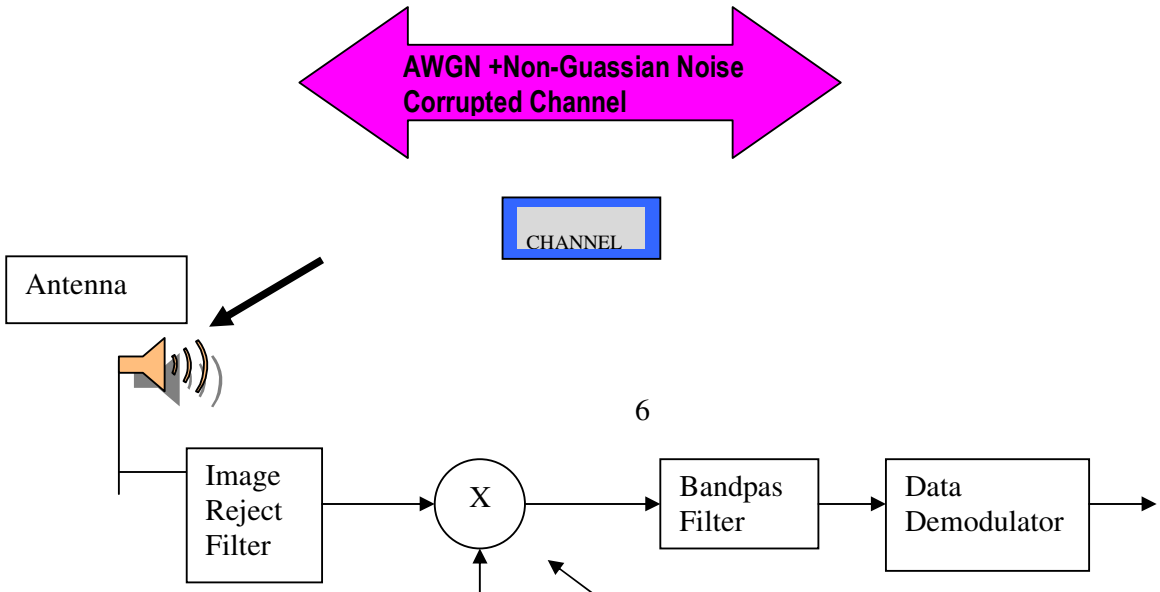
*The function $\Phi(y) = x$. is easily related to the Q-function, which can be calculated using the polynomial approximation

$$g_m(t) = \{2p(t)\cos(\varpi_m t + \varphi_m) \quad ,0 \le t \le T_c$$

$$= 0 \quad \text{elsewhere} \quad \text{----------} (1\text{-}10)$$



(a) Transmitter



Estimated data (output)

y(t)

1 2 3 ..... K

Figure.1.2.Coherent frequency hop spread-spectrum modem

Observe that this psd has discrete components due to the assumption that the same phase is used each time $h_T(t)$ returns to frequency ϖ_m. The Fourier transform $G_m(f)$ is

$$G_m(f) = T_c \exp\left(-j[\pi(f - f_m]T_c - \varphi_m\right)\sin c[(f - f_m)] + T_c \exp\left(-j[\pi(f + f_m)T_c + \varphi_m]\right)\sin c[(f + f_m)T_c]$$

----------(1-11)

Calculation of $S_h(f)$ can be simplified if the assumption is made that $G_m(f)$ and $G_{m'}(f)$ are non-overlapping for $m \neq m'$. In this case, $G_m(f)G^*_{m'}(f) = 0$ and the third term of (1-10(a)) vanishes. This assumption is very good whenever $1/T_c$ is small with respect to the minimum frequency spacing. Assuming also that all frequencies ϖ_m are equally likely leads to

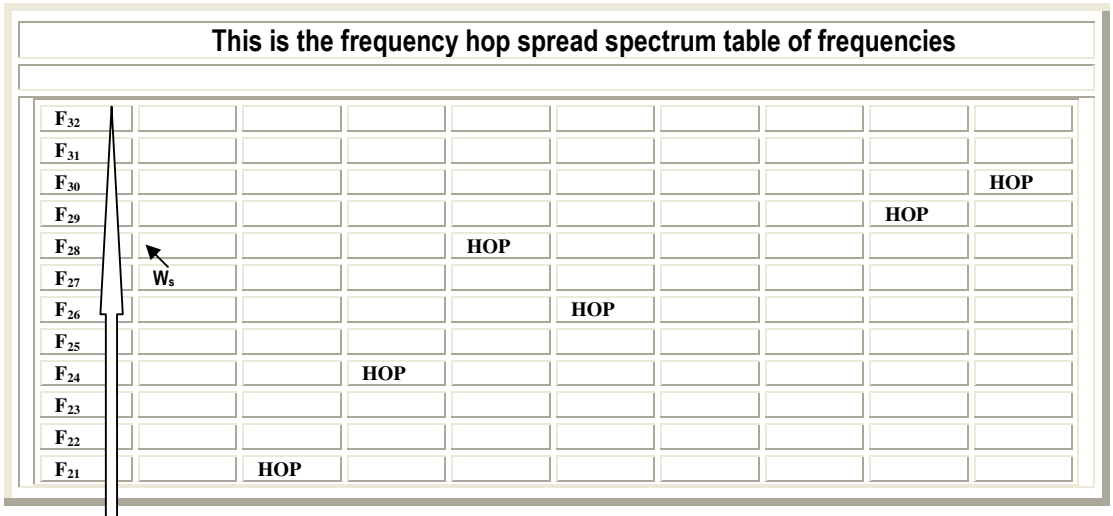
$$S_h(f) \approx \frac{1}{(T_c 2^k)^2} \sum_{m=\infty}^{\infty} \sum_{m=1}^{2^k} \left| G_m\left(\frac{n}{T_c}\right) \right|^2 \delta\left(f - \frac{n}{T_c}\right) + \frac{1}{T_c}\frac{1}{2^k}\left(1 - \frac{1}{2^k}\right) \sum_{m=1}^{2^k} |G_m(f)|^2 \quad \text{----------(1-12)}$$

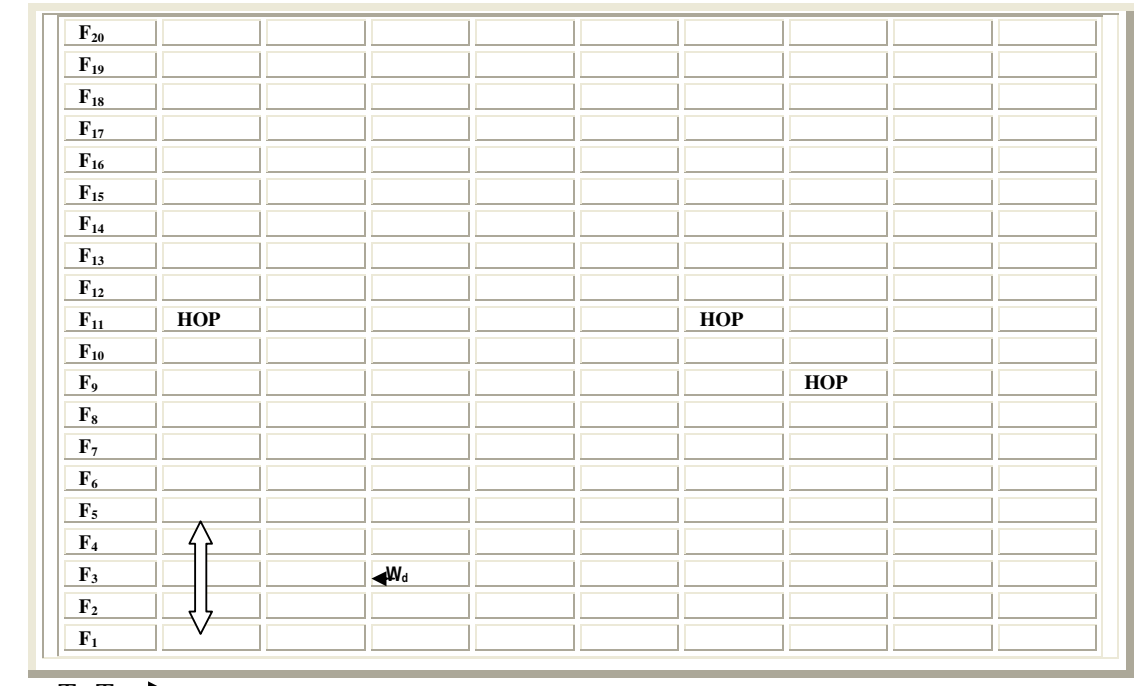
The discrete components of $S_h(f)$ are negligible only when $T_c 2^k >> 1$, which is not usually the case in systems of interest.

1.4. Noncoherent Slow Freqency Hop Spread Spectrum System

A common data modulation for FH systems is M-ary frequency shift keying .Suppose for example that the data modulator outputs one of 2^L tones each LT seconds ,where T is the duration of one information bit .Usually these tones are spaced far enough apart so that the transmitted signals are orthogonal. This implies that the data modulator frequency spacing is at least 1/LT and that the data modulator out put spectral width is approximately 2^L/LT .Each T_c seconds ,the data modulator output is translated to a new frequency by the frequency –hop-modulator. When $T_c \geq$ LT the frequency hop system is is called slo*w-frequency –hop-system.* The output of this spread-spectrum modulator is illustrated in **Figure 1-3**.In this figure the "instantaneous " transmitted spectrum is shown as a function of time for a system with L=2 and k=3.Two data bits are collected each $2T=T_s$ seconds and one of four frequencies is generated by the data modulator .This frequency is translated to one of $2^k=8$ frequency-hop bands by the FH modulator .In this example new frequency -hop band is selected after each group of 2 symbols or 4 bits is transmitted . In the receiver, the transmitted signal is down-converted using a local oscillator which outputs the sequence of frequencies $0,5W_d$, $6W_d, 2W_d, 7W_d$…and the output of the down-converted is a sequence of tones in the

first (lowest) FH band representing the data .The down-converted output is is illustrated in **Figure 1-3-b** this signal can be demodulated. Using the conventional methods of noncoherent MFSK (i.e. a bank of bandpass filters with energy detector detectors at their outputs).A very preliminary estimate of the processing gain of the FH system just described can be obtained by the noise jammer. In the absence of frequency hoping, the jammer chooses a bandwidth W_d centered on the proper carrier frequency and forces the receiver operating signal-to-noise-ratio to $E_b/N_J=E_bW_d/J$,where J is the average jammer power .When frequency hopping is added ,the jammer must place noise in all 2^k frequency –hop bands in order to cause the receiver to have the same performance as before.Thus the jammer requires a total power 2^k times as large as before and the processing gain is $2^k = W_s/W_d$.



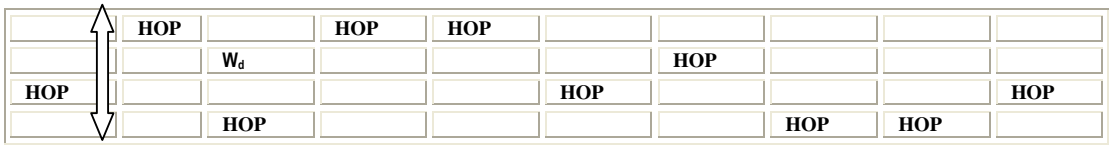


0 1 1 1 0 0 1 1 1 1 0 1 1 0 0 0 0 0 0 1

$W_d \approx 2^L/LT = 2^L/T_s$ **L=2**

$W_s = 2^k w_d$ **k=3**

(a)



(b)

Figure 1.3.Pictorial representation of (a): transmitted signal for an M-ary FSK slow frequency hop spread-spectrum system (b) receiver downconvertor output

The spectrum of the ideal frequency hopper output is as shown below.

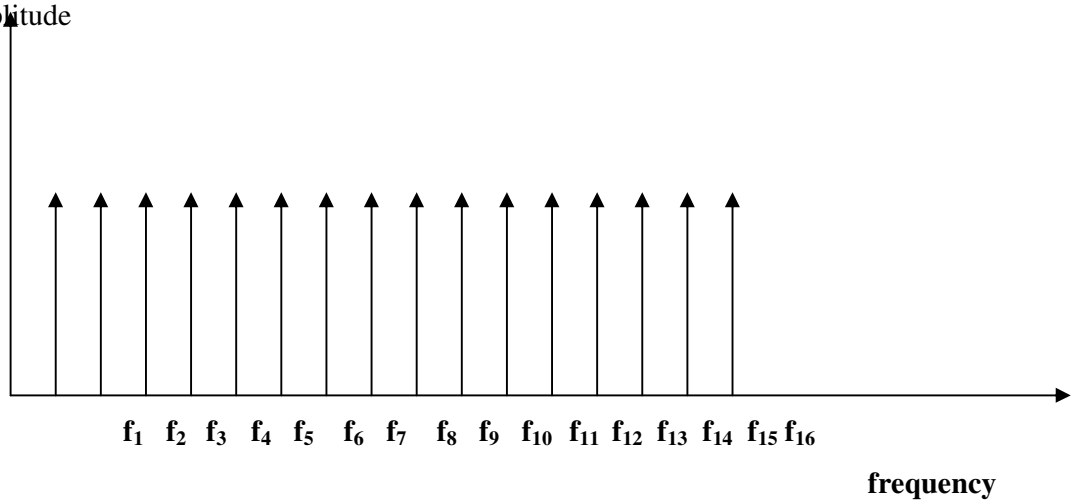


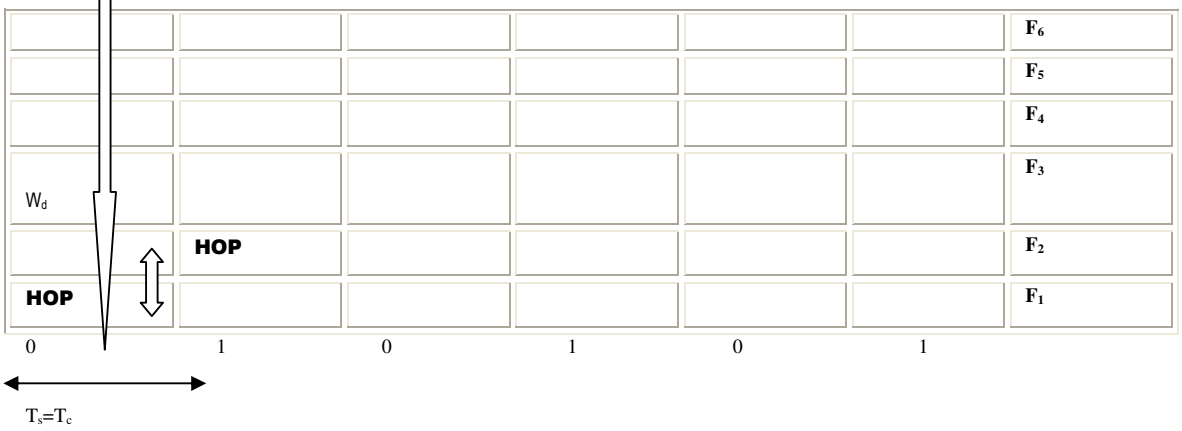
Figure 1-4 Ideal frequency hopper output spectrum

1.5 The Frequency Aspect of the Project

In this project ,W_d= 2^L/LT where L=1,T=1/freq. and LT=1/freq.Gives the value W_d=2freq.W_s=$2^k W_d$=2^3.2.W_d=16freq.

T_c=1/freq.Since the condition $T_c \geq$LT is equality in this case the system is slow frequency hop spread spectrum system.Say if freq.=500MHz then W_d=1000MHz and $_s$=16000MHz.

						F_{16}
			HOP			F_{15}
						F_{14}
W_s				HOP		F_{13}
						F_{12}
		HOP				F_{11}
					HOP	F_{10}
						F_9
						F_8
						F_7



$W_d \approx 2^L/LT = 2^L/T_s$ $L=1$ $T_s=1/freq$

$W_s = 2^k w_d$ $k=3$

(a)

	HOP		HOP		HOP
HOP		HOP		HOP	

(b)

Figure 1.5.Pictorial representation of (a): transmitted signal for FSK slow frequency hop spread-spectrum system (b) receiver downconvertor output

1.6. Project Description

Chapter 1 The general description of the project is as well as the frequency aspect is dicussed. Chapter 2 Some theoretical background about code generators is discussed .And how to design a code generator from table is described. Chapter 3 The sequential type of intial synchronization of the carrier phase is designed and implemented. Chapter 4 Once the intial phase of the carrier is determined then a second order code tracking loop is used to track the rest of the phase. Chapter 5 Results of the computer simulation are discussed .The possible areas of application the computer simulation (or the results of the project results are discussed.)

2.Coding

2.1. Definition and Mathematical Background

All of the spreading codes to be discussed are periodic sequences of ones and zeros with period N. It is convenient to represent a sequence of binary digits. .., b_{-2},b_{-1}, b_0, b_1, b_2, ...by a polynomial

$$b(D) = ... + b_{-2} D^{-2} + b_{-1}D^{-1} + b_0 + b_1D + b_2D^2 + \ldots \quad \text{-------(2.1)}$$

The delay operator D implies simply that the binary symbol which multiplies D^j occurs during the j^{th} time interval of the sequence. Because the code is periodic, $b_n = b_{N+n}$ for any n. The spreading waveform c(t) derived from this spreading code is also periodic with period $T = NT_c$ and is specified by

$$c(t) = \sum_{n=-\infty}^{\infty} a_n p(t - nT_c) \text{ ---------- (2-2)}$$

where $a_n = (-1)^{bn}$, and p(t) is a unit pulse beginning at O and ending at T_c. The waveform c(t) is deterministic, so that its autocorrelation function is defined by

$$R_c(\tau) = \frac{1}{T}\int_0^T c(t)c(t+\tau)dt \text{ ----------(2-3)}$$

Since c(t) is periodic with period T, it follows that $R_c(\tau)$ is also periodic with period T. Consider two different spreading waveforms c(t) and c'(t). The crosscorrelalion function of these two deterministic waveforms is $R_c(\tau) = \frac{1}{T}\int_0^T c'(t)c(t+\tau)dt$ ----------(2.4)

2.2. Sequence Generator Fundamental

It is known that shift registers with feedback and /or feedforward connections can be used to multiply and divide polynomials over GF(2) .Using this property it is possible to generate binary sequences having a specific period length.

Consider the logic circuits illustrated in **Figure 2-1**. In this figure the boxes represent unit delays or shift registers ,circles containing subscribed by letter coefficients represent a connection if the coefficient is 1 or no connection if the coefficient is 0, and circles containing a "+" represent modulo-2 adders or Exclusive-OR gates.

Mathematical Background

Pseudo Random Numbers or PRN's are used in a variety of applications in the digital electronics industry. A few applications include computer simulation of random events, data encryption, and spectrum spreading. As the demand for the unique properties of PRN's increases, the need for PRN generators is likewise increasing. The 8-bit-Pseudo Random

Code Generator implemented in this project is suitable for use as a code sequence generator in a communications application. Before the many uses of this integrated circuit can be appreciated, an understanding of the mathematics behind code generation must be achieved.A code generator consists of a multiple stage shift register, with taps (outputs) taken from various stages are mathematially combined and fed back into the shift register (SR). The traditional approach, as shown in **Figure 2.1**, involves simple modulo-2 addition of the taps (implemented as an exclusive-OR, XOR), and the result is wired as the input to the SR. As long as there is at least one non-zero bit in the shift-register at the start of code generation, a sequence of 1's and 0's will be generated. Whether this code repeats or not, or is maximal-length, depends on which taps are used. If the generated code causes the shift-register to take on all possible state combinations for its individual stages before repeating, then the code is considered "maximal." Each bit that is output from the code generator is called a chip. If a shift register with r stages is used, then the maximum number of chips in a non-repeating binary sequence from the generator is 2^r-1 chips, after which the code repearts.

Maximal length codes have several interesting properties:

- There is one more logic one than logic zeros in a full PRN sequence. In other words, if there are five SR stages in a given code generator, the maximal sequence will contain 31 chips, of which 16 will be ones and 15 will be zeros. This is an important property which will be revisited later.
- The length and number of consecutive ones or zeros in a PRN code is a constant for a given SR length. The relative ordering of these runs may vary with tap selection.
- In a bipolar signalling scheme (-1 is assigned to logic zero, +1 is assigned to logic one), the autocorrelation function of a maximal length code is -1 for all values of shifting except within one chip of zero. Within that narrow interval, the autocorrelation function makes a sharp peak whose height is equal to the length of the code.
- Any tap taken from an SR stage in a maximal generator will reveal the same PRN as the output, only shifted in time. Additionally, if this tap is XORed with the output, the result will once again be the original PRN shifted in time by some different time value.
- If a sample window of a length equal to the number of SR stages is taken from the output of a maximal code generator once every cycle, every possible binary combination (except the all-bits-zero case) will appear once, and only once during the period of the PRN.

The circles containing subscribed coefficients may also be viewed as modulo-2 multiplication of the input by the coefficient.

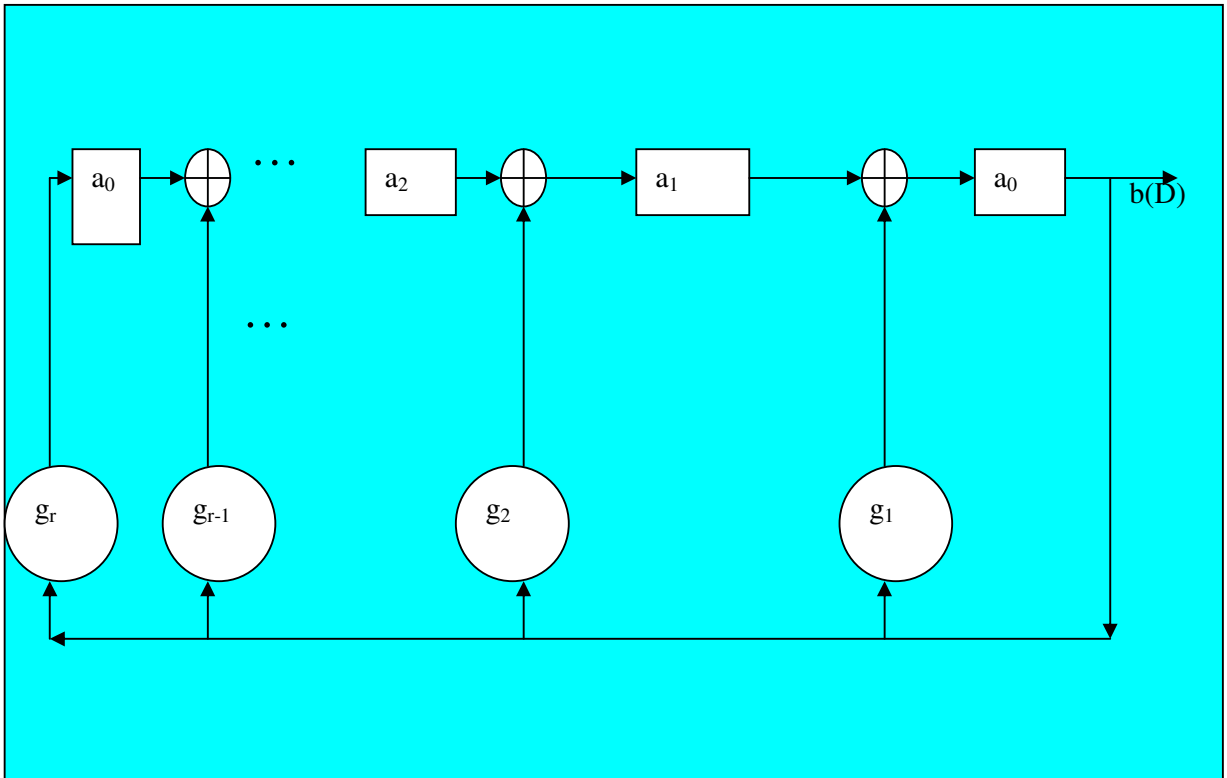


Figure 2-1.Two equivalent circuits for multiplying polynomials

An initial value is stored in the shift registers which are different from zero. This value is given by the polynomial a(D).where

$$a(D) = a_0 + a_1 D + a_2 D^2 + ... + a_{r-1} D^{r-1}$$ ----------(2-5)

The output sequence is given by b(D) where:

$$b(D) = b_0 + b_1 D + b_2 D^2 + ... + b_N D^N$$

The output if the sequence generator is space $\mathbf{b(D) = \frac{a(D)}{g(D)}}$ --------(2.6)

GF(2) Binary Galois field

2.3 Maximal Length Sequence

The maximum possible period of a shift register generator is the smallest N for which the reciprocal g(D) of the generator polynomial g(D) divides D^{N+1}. It can be demonstrated that the reciprocal of a primitive polynomial is also primitive. Thus the smallest N for which a primitive polynomial g(D) of degree r divides D^{N+1} is $N = 2^r - 1$. This means that a shift register initial condition exists which results in a cycle with period $N = 2^r - 1$. Since an r-stage shift register has a total of $2^r - 1$ nonzero states, all states are passed through in this cycle having period $N = 2^r - 1$, *and there is only one possible cycle*. Shift register sequences having the maximum possible period for an r-stage shift register are called Maximal Length sequences or m-sequences. Since the shift register passes through all possible states, each different initial condition results in a different phase of the same m-sequence.

2.4 Tables of Polynomials Yielding m-Sequences

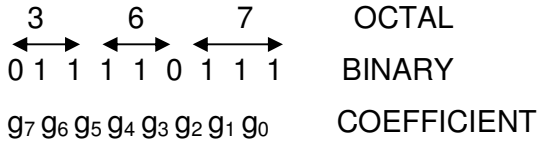
It is often necessary to design circuits that generate m-sequences having a particular number of stages. Since finding the primitive polynomials used to generate these sequences is difficult, a number of authors have generated tables of primitive polynomials for quick reference. In the table, all polynomials are specified by an octal number which defines the coefficients of g(D). The table also defines some polynomials which are not primitive and therefore will not yield a maximal-Length sequence. An example entry in the table is

DEGREE 7 1 211E 3 217E 5 235E 7 367H 9 277E
11 325G 13 203F 19 313H 21 345G.

The entry for each polynomial consists of an integer whose use will be described later, an octal number defining g(D), and a letter designating the type of polynomial. The letters E, F, A, and H designate primitive polynomials. The octal number gives the coefficients of g(D) beginning with g_o on the right and proceeding to g_r in the last nonzero position on the left.

EXAMPLE HOW TO USE THE TABLE

The table contains the entry 7 367H. The letter H means that the entry is a primitive polynomial. Expanding the octal entry 367 into binary form yields



So that $g(D)=1+D+D^2+D^4+D^5+D^6+D^7$

The shift register generator can be either the form of **Figure 2.1**.**Table 2-1** is a list of primitive polynomials of all degrees up to 21. All polynomials in **Table 2-1** are primitive so

that the letter designation has been dropped as well as the number preceding the octal generator specification.Each entry in brackets represents one primitive polynomial as a series of octal numbers, exactly as explained above. The entries followed by an asterisk correspond to circuit implementation with only two feedback connections. No reciprocal polynomials are listed in **Table 2-1**. Since the reciprocal polynomial of a primitive polynomial is also primitive, each entry of **Table 2-1** can be used to generate two distinct m-sequences. It can be demonstrated that the sequences generated by the reciprocal polynomial $g_r(D)$ is equivalent to the reverse of the sequence generated by g(D).

Table.2-1 Primitive Polynomials Having Degree r=2 to 21

DEGREE	OCTAL REPRESENTATION OF GENERATOR POLYNOMIAL (g_0 on the right to g_r on the left)
2	**[7]***
3	**[13]***
4	**[23]***
5	**[45]*,[75],[67]**
6	**[103]*,[147],[155]**
7	**[211]*,[217],[235],[367],[277],[325],[203]*,[313],[345]**
8	**[435] ,[551],[747],[453],[545],[537],[703],[543]**
9	**[1021]*,[1131] ,[1461],[1423],[1055],[1167] ,[1541],**
	[1333] ,[1605j ,[1751],[1743],[1617],[1553] ,[1157]
10	**[2011]*,[2415],[3771],[2157],[3515],[2773],[2033],**
	[2443] ,[2461],[3023],[3543],[2745],[2431],[3177]
11	**[4005]*,[4445],[4215],[4055],[6015],[7413],[4143],**

	[4563] ,[4053] ,[5023],[5623] ,[4577],[6233],[6673]
12	**[10123],[15647],[16533],[16047],[11015],[14127],**
	[17673] , [13565] ,[15341] , [15053] , [15621] , [15321] ,
	[11417],[13505]
13	**[20033],[23261] ,[24623],[23517],[30741],[21643],**
	[30171],[21277] ,[27777],[35051],[34723],[34047],
	[32535] ,[31425]
14	**[42103],[43333],[51761],[40503],[77141],[62677],**
	[44103] , [45145] ,[76303] , [64457] , [57231] , [64167] ,
	[60153],[55753]
15	**[100003]*,[102043],[110013],[102067] ,[104307] ,[100317],**
	[177775] ,[1034511,[110075],[102061] ,[114725] ,[103251],
	[100021]*,[100201]*
16	**[2100 13] , [234313] , [233303] , [307107] , [307527], [306357] ,**
	[201735] , [272201] , [242413] , [270155] ,[302157] , [210205] ,
	[305667],[236107]
17	**[400011]*,[400017],[400431],[525251],[410117],[400731],**
	[411335] ,[444257],[600013],[403555],[525327],[411077],
	[400041]*,[400101]*
18	**[1000201]*,[I000247],[1002241],[1002441],[1100045],**
	[1000407] ,[1003011],[1020121],[1101005],[1000077],
	[1001361] ,[1001567],[1001727],[1002777]
19	**[2000047],[2000641],[2001441],[2000107],[2000077],**
	[2000157] ,[2000175] ,[2000257] ,[2000677],[2000737] ,
	[2001557] ,[200 1637] , [2005775] , [2006677]
20	**[4000011]*,[4001051],[4004515],[6000031],[4442235]**

21	**[10000005]*,[10040205] ,[10020045],[10040315] ,[10000635],**
	[10103075] ,[10050335],[10002135],[17000075]

The octal representation of the codes generated in the project is:

- 255
- the digital representation is 010101101
- the polynomial is $g(D) = 1 + D^2 + D^3 + D^5 + D^7$
- a(D)=1

The other two codes generated in this project are:

- 551
- the digital representation is 101101001
- the polynomial is $g(D) = 1 + D^3 + D^5 + D^6 + D^8$
- a(D)=1
- 747
- the digital representation is 111100111
- the polynomial is $g(D) = 1 + D + D^2 + D^5 + D^6 + D^7 + D^8$
- a(D)=1

The period of the codes is 2^8-1=255.

3.Synchronization

3.1 .Sequential Detection.

Consider the system illustrated in **Figure 3-1** for detecting whether the phase of the receiver generated spreading waveform is correct. Once again, if the phase is correct, the received waveform will be dispread, and signal power will appear at the output of the bandpass filter. The purpose of the envelope detector and the sequential detection processor is to detect this signal power reliably but without generating excessive false alarms when no signal is present.

For any K = 1,2, ..., let $x_K = (x_1, x_2, x_3, x_4, \ldots x_K)$ denote a sequence of samples of the envelope detector output, and assume that the joint pdf of these samples is known. This pdf is denoted $p_s(\mathbf{x_K})$ when signal is present and $p_n(\mathbf{x_K})$ when noise alone is present. During the evaluation of a single spreading waveform phase, samples of the detector output are taken one at a time and input to the sequential detection processor. After each sample (including the first), the likelihood ratio calculator com-

putes

$$\Lambda(x_1, x_2, \ldots x_K) = p_s(\mathbf{x_K}) / p_n(\mathbf{x_K}) \qquad \text{----------(3.1)}$$

This likelihood ratio is input to the sequential detection logic. The logic function compares $\Lambda(\mathbf{x_K})$ with an upper threshold A and a lower threshold B where A > B.If $\Lambda(\mathbf{x_K})$ > A, the signal is declared present and the test ends. If $\Lambda(\mathbf{x_K})$ < B, the
signal is declared absent and the test ends. However, if B < $\Lambda(\mathbf{x_K})$ < A, no decision is made about the presence or absence of the signal and the test continues by taking another sample, calculating another likelihood ratio, and so on.The likelihood ratio indicates whether the sequence of samples is more likely to have resulted from signal and noise at the filter output, $\Lambda(\mathbf{x_K})$ > 1.0, or from noise alone, $\Lambda(\mathbf{x_K})$ < 1.0. When $\Lambda(\mathbf{x_K})$ >1.0, the sequence of samples is much more likely to have come from signal and noise and a decision that the spreading waveform phase is correct is very reliable.

Similarly, when $\Lambda(\mathbf{x_K})$ < 1.0 it is much more likely that the sequence of samples are the result of noise alone. The thresholds A and B are selected so that a specific reliability is achieved on the signal present/absent decision. **Figure 3-2** illustrates $p_s(x_1)$ and $p_n(x_l)$ for three possible first sample values. For sample S_1, $\Lambda(x_1)$ << 1.0; for sample S_2 , $\Lambda(x_1)$ = 1.0,S_3,$\Lambda(x_1)$>>1

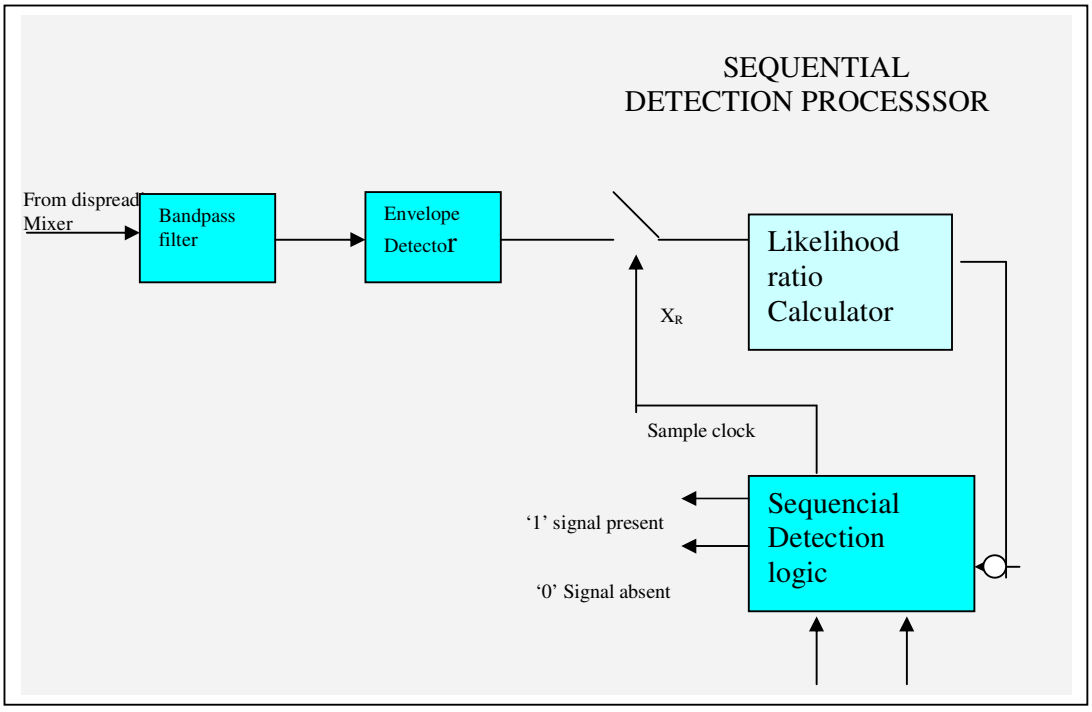


Figure3.1. Sequential detector.

The probability density function for signal plus noise shown in this figure is calculated for a signal-to-ratio of about +3dB. Assume that the samples of the envelope detector output are spaced sufficiently in time that they may be considered independent. then the joint pdf's are products of single sample pdf's,that is,

$$p_s(\mathbf{x_K}) = \prod_{k=1}^{K} p_s(x_k) \text{ ----------(3.2)}$$

$$p_n(\mathbf{x_K}) = \prod_{k=1}^{K} p_n(x_K) \text{ ----------(3.3)}$$

and

$$\mathbf{\Lambda(x_K)} = \prod_{k=1}^{k=K} \lambda(x_K) \quad \text{----------(3.4)}$$

where

$$\lambda(x_k) \triangleright \frac{p_s(x_K)}{p_n(x_K)} \text{ ----------(3.5)}$$

When signal is present at the design point SNR, envelope detector samples usually result in $\lambda(x_K)>1.0$ and the product $\mathbf{\Lambda(x_K)}$ grows with increasing K. When no signal is present, the envelope detector samples usually result in $\lambda(x_K)<1.0$ and the product $\mathbf{\Lambda(x_K)}$ approaches zero as K increases. In the limit as $K \to \infty$, $\mathbf{\Lambda(x_K)}$ will always cross the upper threshold when signal is present at the design point SNR or the lower threshold when no signal is present.

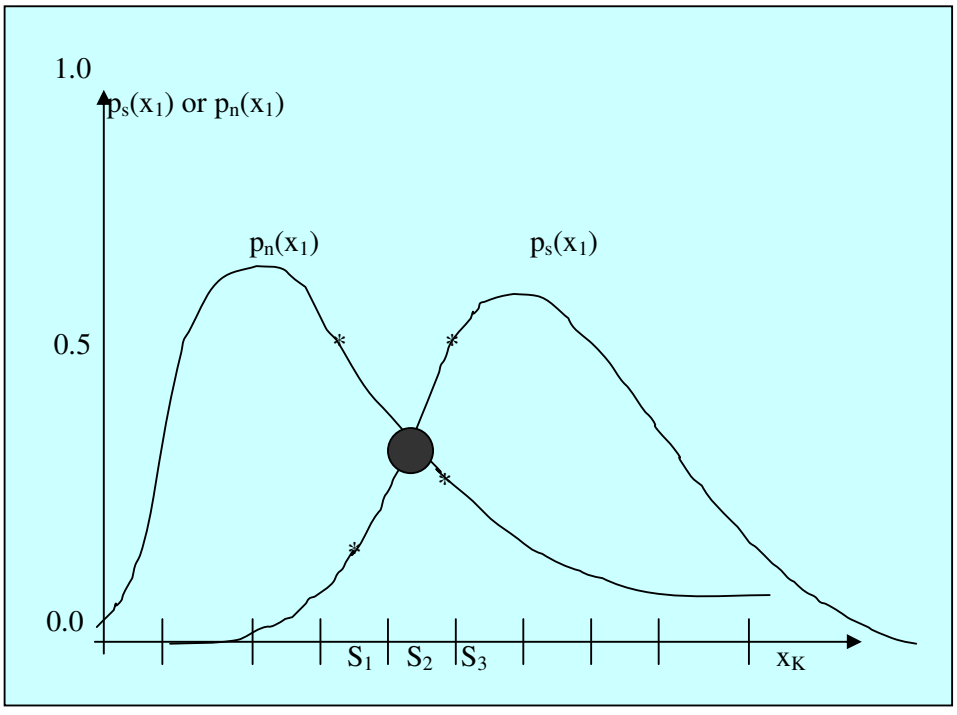


Figure 3.2. Envelope detector output probability density function

For finite K, errors can be made. A missed detection occurs whenever $\Lambda(\mathbf{x_K})$ crosses the lower threshold before crossing the upper threshold when signal is indeed present. The probability of this event is 1.0 -P_d. A false alarm occurs whenever $\Lambda(\mathbf{x_K})$ crosses the upper threshold prior to crossing the lower threshold when no signal is present. The probability of this event is P_{fa}. Typical sequences of values of $\Lambda(\mathbf{x_K})$ for signal plus noise and noise alone are illustrated in **Figure 3-3**. No errors occur in the cases shown.The sample values move toward one of the thresholds with a step size that is a random variable. The statistics of this random variable and the absolute value of the thresholds will determine the average number of samples that must be processed to cross one of the thresholds.The average sample number (ASN) is directly related to the quantities T_{da} and T_i, which must be evaluated to calculate the average synchronization time. The ASN is a function of the received signal-to-noise ratio, the detector characteristic, and the thresholds A and B.

Detectors having the general form shown in **Figure 3-1** are called sequential detectors. Sequential detection is sometimes referred to, as the *sequential probability ratio test* (SPRT). It has been proven that sequential detection is optimum in the sense that it yields the minimum average detection time for a specified P_d and P_{fa}

3.2 Relationship Between Thresholds and Error Probabilities.

To design systems that use sequential detection, it is necessary to be able to relate the thresholds and the probabilities P_d and P_{fa} for any siganal-to-noise-ratio.Consider the set of all real vectors $\mathbf{x_K}$ for all K = 1, 2, ..., and assume that the functions $p_s(\mathbf{x_K})$ and $p_n(\mathbf{x_K})$ are known for the particular SNR. Let Γ_1 denote the set of all vectors that result in a decision that signal is present at the output of the IF bandpass filter .

Thus, for every $\mathbf{x_K} \in \mathbf{\Gamma}_1$,

$$A < p_s(\mathbf{x_K})/p_n(\mathbf{x_K}) \qquad \text{----------(3.6)}$$

Note that all vectors in Γ_1 do not have the same dimensionality since different sequences $\mathbf{x_K}$ may result in a threshold crossing after different numbers of samples

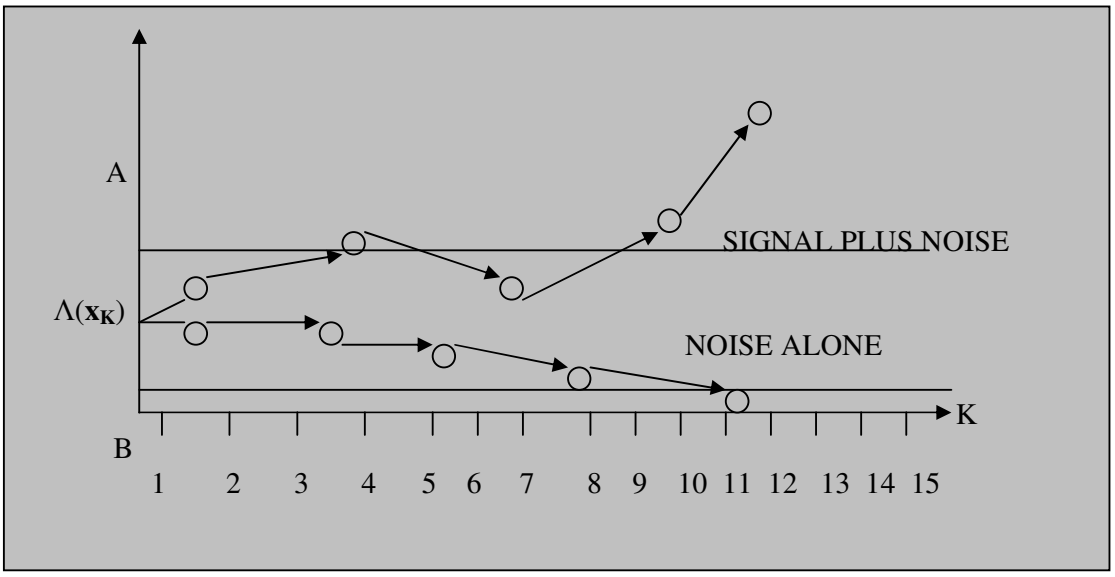


Figure 3.3 Typical sequences of values of $\Lambda(x_K)$ for sequential detection

are taken. Assume now that the bandpass filter output is noise alone. A false alarm occurs for any $\mathbf{x_K} \in \mathbf{\Gamma_1}$ and the probability of this event is

$$P_{fa} = \int_{x_K \in \Gamma_1} p_n(\mathbf{x_K}) d\mathbf{x_K} \qquad \text{---------- (3.7a)}$$

Similarly, when the filter output is signal plus noise, a correct detection occurs whenever $\mathbf{x_K} \in \mathbf{\Gamma_1}$ and the probability of this event is

$$P_d = \int_{x_K \in \Gamma_1} p_s(\mathbf{x_K}) d\mathbf{x_K} \qquad \text{----------(3.7b)}$$

T_{da} the average dwell time at incorrect spreading wave form phase.T_i integration time.

Multiplying both sides of (3-6) by $p_n(\mathbf{x_K})$ integrating over $\mathbf{\Gamma_1}$, and using the results of (3-7) yields

$P_d > AP_{fa}$ ----------(3.8)

Let $\mathbf{\Gamma_2}$ denote the set of all vectors which result in a decision that noise alone is present at the output of the filter. For every $\mathbf{x_K} \in \mathbf{\Gamma_2}$

$$\frac{p_s(\mathbf{x_K})}{p_n(\mathbf{x_K})} < B \qquad \text{----------(3.9)}$$

Another manipulation leads:

$$1 - P_d = B(1 - P_{fa}) \qquad \text{----------(3.10)}$$

For small signal–to-noise –ratio:

$$P_{fa} = \frac{1 - B}{A - B} \qquad \text{----------(3.11)}$$

$$P_d = A\frac{(1 - B)}{(A - B)} \qquad \text{----------(3.12)}$$

3.3 Using a Linear Envelope Detector and the Log - Likelihood Ratio.

Although the sequential test can be analyzed in its most basic form where the actual likelihood ratio is calculated, the multiplications of the likelihood ratios as in (3-4) are

inconvenient and not necessary. The performance of the SPRT is unchanged if, instead of using the likelihood ratio, any monotonic strictly increasing function of the likelihood ratio is used.

Of course, the thresholds are also modified by this same monotonic function. Since the likelihood ratio is positive, the logarithm is a convenient function to use since it also converts the undesirable product functions to summations. At this point it is also convenient to limit attention to the specific detector and associated probability densities of most interest for application to spread-spectrum systems. The detector is a linear envelope detector and the synchronization hardware is trying to detect the presence or absence of a sine wave at the bandpass filter output. Therefore, the pdf of a single sample of the envelope detector output is given by (3-13), which is due to Rice.

$$p_z(\alpha) = \frac{\alpha}{N}\exp\left(-\frac{\alpha^2 + A^2}{2N}\right) I_0\left(\frac{\alpha A}{N}\right)$$

for $\alpha \geq 0$ ----------(3.13)

$=0$ elsewhere

where $I_0(x)$ is Zeroth order Bessel function.

The likelihood ratio, using (3-13) and (3-5) with slightly modified notation, is

$$\lambda(x_K) = \frac{p_s(x_K)}{p_n(x_K)}$$

$$\lambda(x_K) = \frac{(\frac{x_K}{N})\exp\left(-\frac{\langle x^2{}_K + 2P\rangle}{2N}\right) I_0(x_K \frac{\sqrt{2P}}{N})}{(\frac{x_K}{N})\exp(\frac{-x^2{}_K}{2N})} \qquad \text{----------(3.14)}$$

where $p_n(\mathbf{x_k})$ is found from (3-13) by setting the signal power P = 0. The output of the bandpass filter has been assumed to be AWGN with power N or to be AWGN with power N plus a sine wave with power P and amplitude $\sqrt{2P}$. Simplifying (3-14) and taking the logarithm yields the log-likelihood ratio for a single sample

$$\ln[\lambda(x_K)] = -\frac{P}{N} + \ln\left[I_0\left(\frac{x_K\sqrt{2P}}{N}\right)\right] \qquad \text{----------(3.15)}$$

The log –likelihood ratio for a sequence of samples is the logarithm of (3-14) which is

$$\ln[\Lambda(\mathbf{x_K})] = \ln\left[\prod_{k=1}^{K}\lambda(x_K)\right]$$

$$= \sum_{k=1}^{K}\ln[\lambda(x_K)]$$

$$= \sum_{k=1}^{K}\left[\ln\left[I_0\left(\frac{x_K\sqrt{2P}}{N}\right)\right] - \frac{P}{N}\right] \qquad \text{----------(3-16)}$$

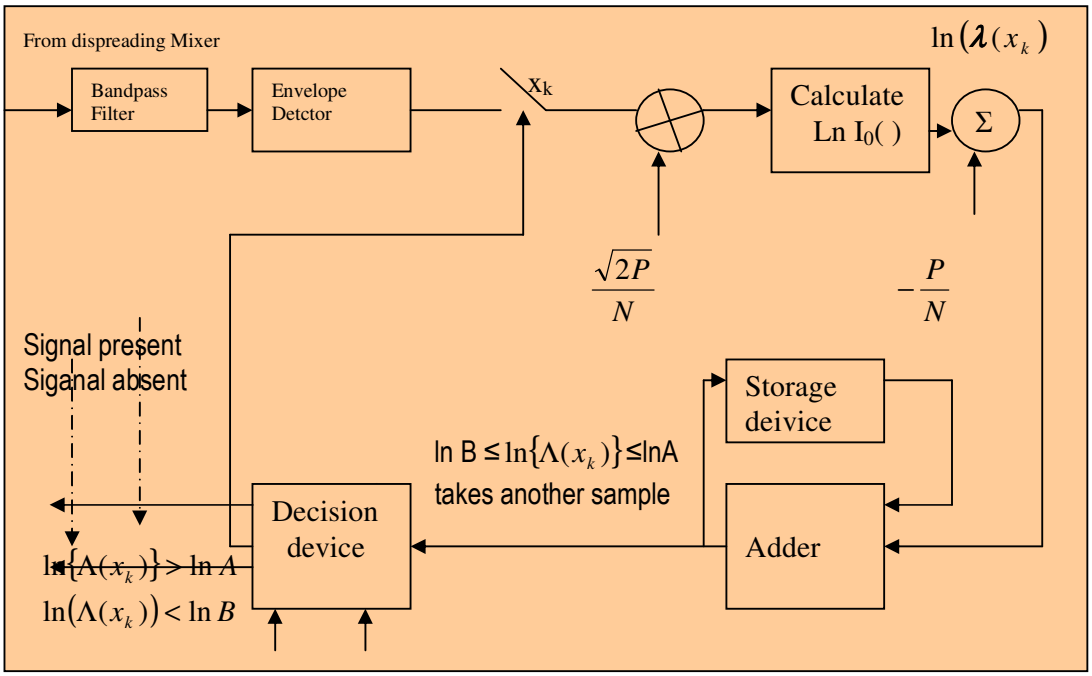


Figure-3-4 Sequential detector using linear envelope detector and LLR

In the project the value of P_d=0.99 and P_{fa}=0.01. Solving for (3-11) and (3-12) gives the respective values of A and B :

$$B = \frac{(1 - P_d)}{(1 - P_{fa})} \qquad \text{----------(3-17)}$$

$$A = \frac{P_d}{P_{fa}} \qquad \text{----------(3-16)}$$

$$B = \frac{1 - 0.99}{1 - 0.01}$$

$$= 0.0101$$

$$A = \frac{0.99}{0.01}$$

$$= 99$$

And the logarithm thresholds are :

$$\ln(A) = \ln(99) = 4.5951$$

$$\ln(B) = \ln(0.0101) = -4.5951$$

The number of samples taken in the project is 3.That is K=3;x_k=(x_1,x_2,x_3).Each 50 time units apart.

4.Code Tracking

4.1 Baseband Full Time Early Late Tracking Loop

The function of a baseband full-time loop is to track the time-varying phase of the received spreading waveform c(t -T $_d$). The function $\hat{T}_d(t)$ will denote the receiver estimate of T $_d$(t), and T $_d$ and $\hat{T}_d$ are always functions of time, whether or not this dependence is written explicitly. The received signal consists of the spreading waveform $\sqrt{P}c(t-T_d)$ with power P and additive white Gaussian noise n(t) with two-sided power spectral density $\frac{N_0}{2}$ W /Hz. That is,

$$s_r(t) = \sqrt{P}c(t-T_d) + n(t) \qquad \text{----------(4-1)}$$

Figure 3-1 is a conceptual block diagram of the tracking loop. It consists of a phase discriminator, a loop filter, a voltage-controlled oscillator, and a spreading-wave-form generator .The received signal is input to the delay-lock discriminator where, after power division, it is correlated with an early spreading waveform $c(t-\hat{T}_d+(\Delta/2)T_c)$ and a late spreading waveform, $c(t-\hat{T}_d-(\Delta/2)T_c)$. The parameter Δ is the total normalized time difference between the early and late discriminator channels.

Consider the operation of the delay-lock discriminator as a static phase-measuring device in a noiseless environment. That is, let T $_d$ and $\hat{T}_d$ be fixed and determine the output of the discriminator. This output will contain a component which is a

function of $\delta = \frac{T_d - \hat{T}_d}{T_c}$ and is suitable for driving the VCO just as the Phase-Locked-Loop multiplier output $\sin(\theta_i - \theta_0)$.In the static case ,it is convenient to write $y_1(t)$,$y_2(t)$,and and $\epsilon(t, \delta)$ as explicit functions of $T_d, \hat{T}_d$,and t.. Thus the early-correlator output is $y_1(t, T_d, \hat{T}_d) = K_1\sqrt{\frac{P}{2}}c(t - T_d)c(t - \hat{T}_d + \frac{\Delta T_c}{2})$) ----------(4-2)

and the late-correlator output is

$$y_2(t, T_d, \hat{T}_d) = K_1\sqrt{\frac{P}{2}}c(t - T_d)c(t - \hat{T}_d - \frac{\Delta T_c}{2}) \quad \text{----------(4-3)}$$

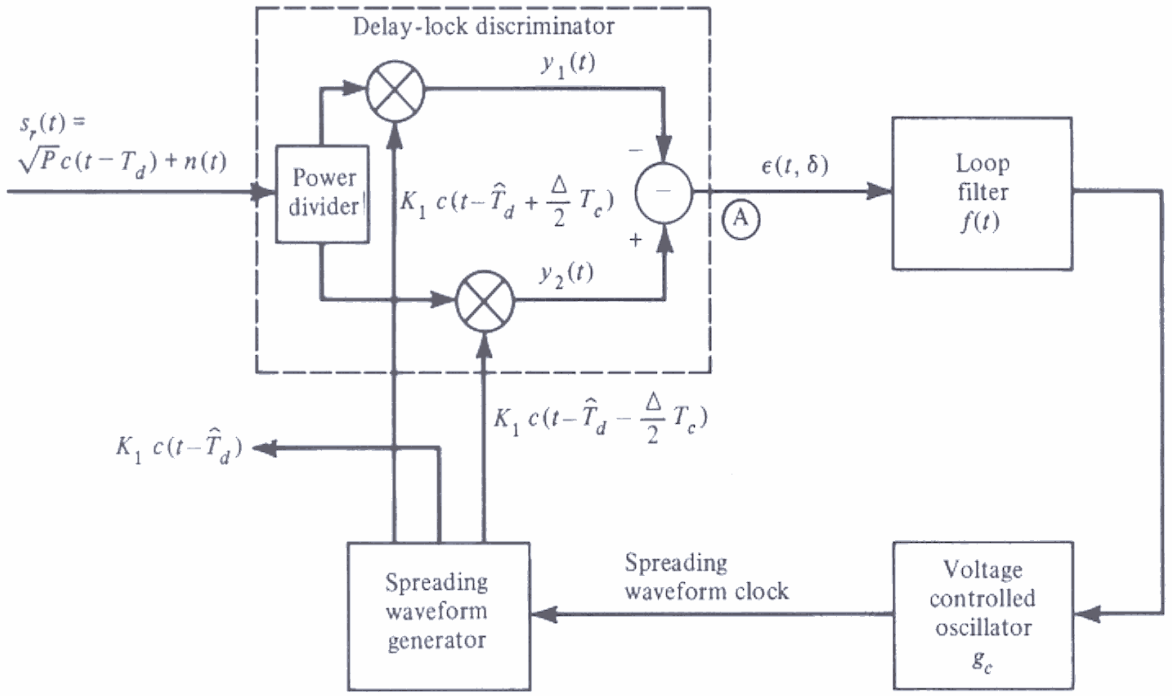


Figure.4.1. Conceptual block diagram:baseband delay-lock tracking loop

In these expressions, K_1 is the multiplier gain and is dependent on the particular multiplier hardware implementation. The input signal has been divided by $\sqrt{2}$ to account for

the power division, and noise has been ignored. The delay-lock discriminator output is the difference of $y_2(t)$ and $y_1(t)$ and is

$$\varepsilon(t,T_d,\hat{T_d}) = y_2(t,T_d,\hat{T_d}) - y_1(t,T_d,\hat{T_d})$$

$$= K_1\sqrt{\frac{P}{2}}c(t-T_d)\left[c\left(t-\hat{T_d}-\frac{\Delta T_c}{2}\right)-c\left(t-\hat{T_d}+\frac{\Delta T_c}{2}\right)\right] \qquad \text{----------(4-4)}$$

The dc component of this signal is used for code tracking. The time-varying component, which is also a function of δ, is called *code self-noise.*The dc component of ϵ(t, T_d, $\hat{T_d}$) is denoted $K_1\sqrt{\frac{P}{2}}D_\Delta(T_d,\hat{T_d})$ and is the time average of $\varepsilon(t,T_d,\hat{T_d})$.Thus

$$K_1\sqrt{\frac{P}{2}}D_\Delta(T_d,\hat{T_d}) = \frac{1}{NT_c}\int_{\frac{-NT_c}{2}}^{\frac{NTc}{2}} K_1\sqrt{\frac{P}{2}}c(t-T_d)\left[c\left(t-\hat{T_d}-\frac{\Delta T_c}{2}\right)-c\left(t-\hat{T_d}-\frac{\Delta T_c}{2}\right)\right]dt$$

----------(4.5)

where NT_c is the period of c(t). Recalling the definition of the autocorrelation function of c(t) from (3-2),

$$D_\Delta(T_d,\hat{T_d}) = R_c(T_d-\hat{T_d}-\frac{\Delta T_c}{2}) - R_c(T_d-\hat{T}_d+\frac{\Delta T_c}{2})$$

$$= R_c\left[\left(\delta-\frac{\Delta}{2}\right)T_c\right] - R_c\left[\left(\delta+\frac{\Delta}{2}\right)T_c\right]$$

$$\triangleright\!\!= D_\Delta(\delta)$$

This function is plotted in **Figure 4-2** for the value of Δ=1, where c(t) is the waveform derived from a maximal-length sequence .

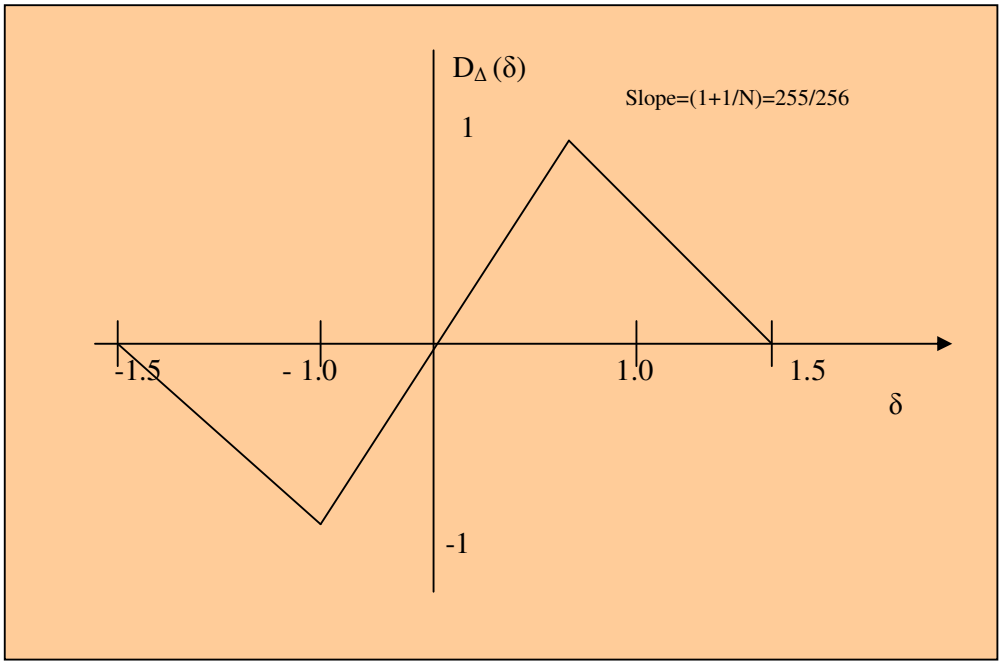


Figure 4.2 Delay-lock Discriminator dc outputs for maximal–length sequence spreading codes for the values of Δ=1

In the analyses of code tracking loops the time-varying component of ϵ(t, T_d, $\hat{T}_d$), which is denoted $K_1\sqrt{\frac{P}{2}}N_\Delta(t, T_d, \hat{T}_d)$, can be safely ignored since most of this self-noise power is at frequencies which are well outside the bandwidth of the tracking loop. An overbound on the magnitude of the power spectrum of ϵ(t, T_d, $\hat{T}_d$) is easily obtained, however, using results developed previously. This overbound is sufficient in most instances to prove that the code self-noise can be ignored. The overbound is obtained by observing that both terms of (4-4) have identical power spectra. The worst-case power spectrum of the sum is obtained by assuming that all components add phase synchronously. Thus the worst-case

power spectrum of ϵ(t, T_d, $\hat{T}_d$), is four times the magnitude of the power spectrum of

$$\varepsilon'(t,T_d,\hat{T}_d) = K_1\sqrt{\frac{P}{2}}c(t-T_d)c(t-\hat{T}_d-\frac{\Delta T_c}{2}) \quad \text{----------(4-5)}$$

The code self-noise can be ignored whenever the magnitude of the bound just calculated is sufficiently below the magnitude of the thermal noise power-spectral density calculated at the same point. A lower bound on code self-noise power spectrum is obtained by assuming that the two terms of (4-4) are uncorrelated. When this is true, the power spectrum of $K_1\sqrt{\frac{P}{2}}N_\Delta(t,T_d,\hat{T}_d)$ is twice the magnitude of the power spectrum of ϵ(t, T_d, $\hat{T}_d$). When $\Delta \geq 1.0$ and the spreading code is an m-sequence, the two terms of (4-4) are nearly uncorrelated. Consider now the operation of the phase discriminator in an AWGN environment. With noise included,

$$y_1(t,T_d,T_d) = K_1\sqrt{\frac{P}{2}}c(t-T_d)c(t-\hat{T}_d+\frac{\Delta T_c}{2})+\frac{K_1}{\sqrt{2}}c(t-\hat{T}_d+\frac{\Delta T_c}{2})n(t) \quad \text{----------(4-6)}$$

$$y_2(t,T_d,\hat{T}_d) = K_1\sqrt{\frac{P}{2}}c(t-T_d)c(t-T_d-\frac{\Delta T_c}{2})+\frac{K_1}{\sqrt{2}}c(t-\hat{T}_d-\frac{\Delta T_c}{2})n(t) \quad \text{----------(4-7)}$$

The discriminator output is

$$\varepsilon(t,T_d,\hat{T}_d) = K_1\sqrt{\frac{P}{2}}\left[D_\Delta(\delta)+N_\Delta(t,T_d,\hat{T}_d)\right]+\frac{K_1}{\sqrt{2}}n(t)\left[c(t-\hat{T}_d-\frac{\Delta T_c}{2})-c(t-\hat{T}_d+\frac{\Delta T_c}{2}\right]$$

----------(4-8)

Assuming that code self-noise can be ignored, this can be written

$$\varepsilon(t,T_d,\hat{T}_d) = K_1\sqrt{\frac{P}{2}}\left[D_\Delta(\delta)+\frac{1}{\sqrt{P}}n'(t)\right]) \quad \text{----------(4-9)}$$

where

$$n'(t) = n(t)\left[c(t-\hat{T}_d-\frac{\Delta T_c}{2})-c(t-\hat{T}_d+\frac{\Delta T_c}{2})\right] \quad \text{----------(4-10)}$$

To evaluate the noise performance of the tracking loop, the power spectrum of n'(t) must be calculated. This calculation is accomplished by first evaluating the autocorrelation

function of n'(t). Care must be exercised in defining the autocorrelation function since, with T_d and Δ fixed, the random process n'(t) defined above is not wide-sense stationary and therefore does not possess a power spectrum in the normal sense.With T_d interpreted as a random variable, stationarity is achieved and

$$R_{n'}(\boldsymbol{\tau}) = E\left\{ n(t)n(t+\boldsymbol{\tau}) \left[c(t - T_d - \frac{\Delta T_c}{2}) - c(t - T_d + \frac{\Delta T_c}{2}) \right] X \left[c(t+\boldsymbol{\tau} - \hat{T}_d - \frac{\Delta T_c}{2}) - c(t+\boldsymbol{\tau} - \frac{\Delta T_c}{2}) \right] \right\}$$

----------(4-11)

The white noise n(t) is independent of the spreading code c(t), so that the expected value can be factored to obtain

$$R_{n'}(\tau) = E[n(t)n(t+\tau)] E\left\{ \left[c(t - \hat{T}_d - \frac{\Delta T_c}{2}) - c(t - \hat{T}_d + \frac{\Delta T_c}{2}) \right] \left[c(t+\tau - \hat{T}_d - \frac{\Delta T_c}{2}) - c(t+\tau - T_d + \frac{\Delta T_c}{2}) \right] \right\}$$

----------(4-12)

The autocorrelation. function of the noise is a delta function, that is,

$$E[n(t)n(t+\tau)] = \frac{N_0}{2}\delta(\tau) \qquad \text{----------(4-13)}$$

Substituting (4-13) into (4-12) and setting $\tau = 0$, since $\delta(\tau)$ iz zero, for all $\tau \neq 0$,results in

$$R_{n'}(\tau) = \frac{N_0}{2}\delta(\tau) \left\{ E\left[c^2(t - \hat{T}_d - \frac{\Delta T_c}{2}) - 2E[c(t - \hat{T}_d - \frac{\Delta T_c}{2}) c(t - \hat{T}_d + \frac{\Delta T_c}{2}) \right] + E\left[c^2(t - \hat{T}_d + \frac{\Delta T_c}{2}) \right] \right\}$$

----------(4-14)

The spreading waveforms take on only values of±1, so that $c^2(t) = 1,0$. Assume that the spreading waveforms are derived from maximal-length sequences. Because of the shift-and-add property of m-sequences, the function $c(t-\hat{T}_d-\frac{\Delta T_c}{2})\ c(t-\hat{T}_d-\frac{\Delta T_c}{2})$ may be viewed as a signal that switches between two phases.Thus

$$R_{n'}(\tau) = N_0\delta(\tau)(1+\frac{1}{N})$$

for $\Delta \geq 1.0$

$$= N_0\delta(\tau)\Delta(1+\frac{1}{N})$$

for $\Delta < 1.0$

----------(4-15)

The two-sided power spectral density of n'(t) is the Fourier transform of $R_{n'},(\tau)$ and is

$$S_{n'}(f) = N_0(1+\frac{1}{N})$$

for $\Delta \geq,1.0$

$$= N_0\Delta(1+\frac{1}{N})$$

for $\Delta < 1.0$

----------(4-16)

At this point the delay-lock discriminator of **Figure 4-1** has been adequately characterized to enable the development of a linear model of the entire tracking loop which will be valid for small tracking errors δ. Consider the voltage-controlled oscillator. The output frequency of

this oscillator is $f_o + g_c v(t)$, where f_o is the rest or quiescent frequency and g_c is the VCO gain in Hz/V. The instantaneous output phase of this oscillator is the integral of frequency and is

$$2\pi f_0 t + 2\pi \frac{\hat{T}_d}{T_c} = 2\pi f_0 t + 2\pi \int_0^t g_c v(\lambda) d\lambda + \theta_0$$

----------(4-17)

where θ_0 is the phase at time zero. The initial phase is set equal to zero in all that follows. The left side of the equation is the oscillator output phase written directly as a function of $\hat{T}_d(t)$. Subtracting $2\pi f_o t$ from both sides of (4-17) and dividing by 2π yields

$$\frac{\hat{T}_d(t)}{Tc} = g_c \int_0^t v(\lambda) d\lambda \qquad \text{----------(4-18)}$$

and taking the Laplace transform yields

$$\frac{\hat{T}_d(s)}{T_c} = \frac{g_c V(s)}{s} \qquad \text{----------(4.19)}$$

where $\hat{T}_d(s)$ represents the Laplace transform of T(t).The tracking loop filter is assumed to be passive, linear , and time invariant, so that its input-output relationship can be described by a differential equation having the form

$$a_m \frac{d^m \varepsilon}{dt^m} + a_{m-1} \frac{d^{m-1} \varepsilon}{dt^{m-1}} + ... + a_0 \varepsilon$$
$$= b_n \frac{d^n v}{dt^n} + b_{n-1} \frac{d^{n-1} v}{dt^{n-1}} + ... + b_b v \quad \text{----------(4.20)}$$

where $m \leq n$ and ϵ represents $\epsilon(t,\delta)$. This linear differential equation can be solved using classical techniques. Using Laplace transform techniques, the ratio of the Laplace transform of the output, V(s), to the Laplace transform of the input E(s, δ) is found to be

$$\frac{V(s)}{E(s,\delta)} = \frac{a_m s^m + a_{m-1} s^{m-1} + ... + a_0}{b_n s^n + b_{n-1} s^{n-1} + ... + b_0} \qquad \text{----------(4-21)}$$

$$\triangleright F(s)$$

$$=$$

where F(s) is the transfer function of the filter. This filter can also be described using its impulse response function f(t). The filter output is the convolution of the input signal and the impulse response. That is,

$$v(t) = \int_{-\infty}^{t} \varepsilon(\lambda, \delta) f(t-\lambda) d\lambda \qquad \text{---------(4-22)}$$

The impulse response f(t) is the inverse Laplace transform of the transfer function F(s). Using (4-9) to represent the delay-lock discriminator output, and (4-18) and (4-22) to represent the VCO and loop filter characteristics, the nonlinear integral equation representing the operation of the tracking loop of Figure 4-1 is

$$\frac{\hat{T}_d}{T_c} = g_c \int_0^t v(\lambda) d\lambda \qquad \text{---------(4-23)}$$

$$= g_c \int_0^t \int_{-\infty}^{\lambda} \varepsilon(\alpha, \delta) f(\lambda - \alpha) d\alpha d\lambda$$

$$= g_c \int_0^t \int_{-\infty}^{\lambda} \left\{ K_1 \sqrt{\frac{P}{2}} D \Delta \left[\delta(\alpha) + \frac{K_1}{\sqrt{2}} n'(\alpha) \right] \right\} f(\lambda - \alpha) d\alpha d\lambda$$

where n'(t) is defined in (4-10). In this equation, the dependence of the normalized phase error $\delta(t) = \frac{T_d(t) - \hat{T}_d(t)}{T_c}$ on time has been shown explicitly. Equation (4-23) suggests the equivalent circuit illustrated in **Figure 4-2**. The equation that describes this circuit can be written down by inspection, and will be identical to (4-23). This circuit is the nonlinear

[because of the function $D_\Delta(\delta)$] equivalent circuit for the baseband ful1-time early-late tracking loop.

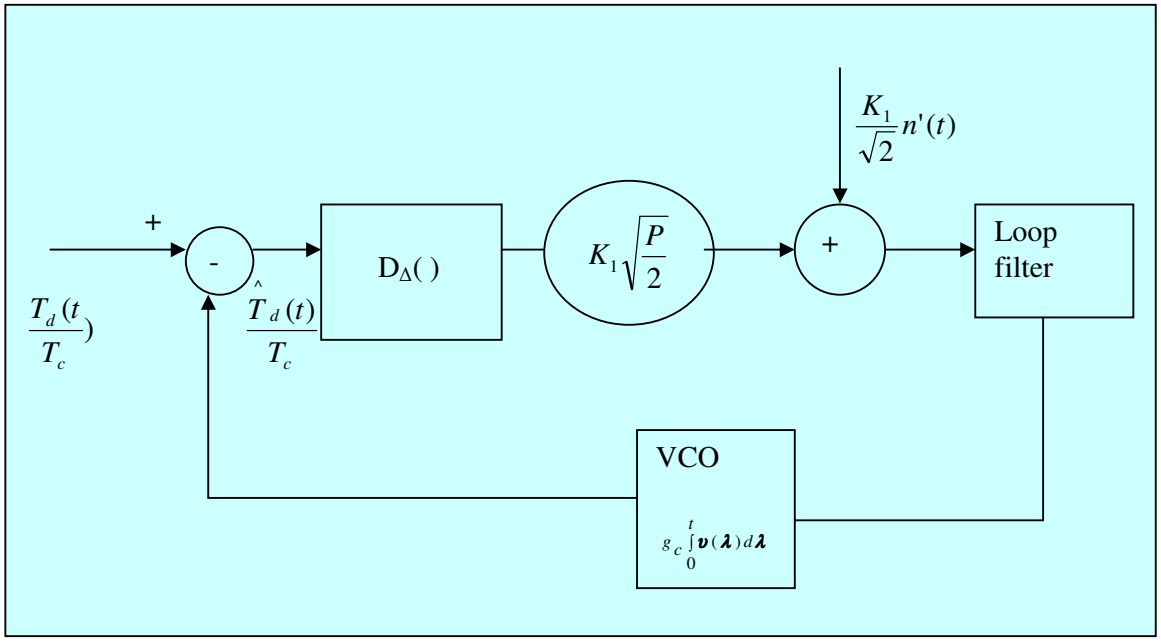


Figure 4-3 Nonlinear equivalent circuit for the baseband full-time early late tracking loop.

For small tracking error, the phase discriminator output is a linear function of the tracking error and (4-23) can be written

$$\frac{T_d(t)}{T_c} = g_c\int_0^t\int_{-\infty}^{\lambda}\left[K_1\frac{\sqrt{P}}{2}2(1+\frac{1}{N})\frac{T_d(\alpha)-\hat{T}_d(\alpha)}{T_c}+\frac{K_1}{\sqrt{2}}n'(\alpha)\right]f(\lambda-\alpha)d\alpha d\lambda$$

----------(4-24)

Consider the system illustrated in **Figure 4- 4**. The equation that describes the operation of this system can be written down by inspection, and is identical to (4-24), where K_d is defined in Figure 4-4. Therefore the system of Figure 4-4

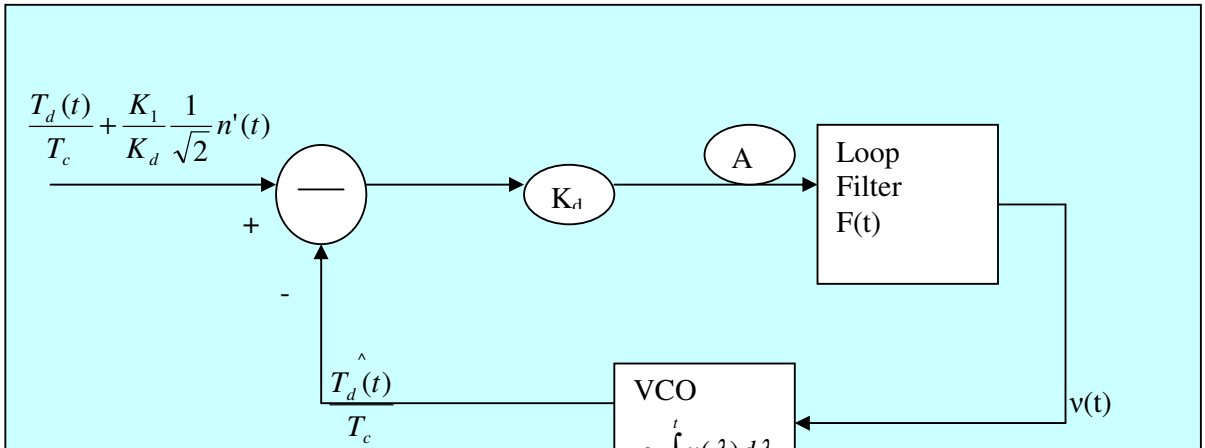


Figure 4-4 Linear equivalent circuit for the baseband full-time early –late tracking

describes the operation of the baseband full-time early late tracking loop for small tracking errors. The noise function at the input to the circuit of Figure 4- 4 will produce noise at point A, which is identical to the noise at point A of Figure 4-1. The power spectral density of the white noise function n'(t) was given in (4-16). The power spectral density of $\frac{K_1 n'(t)}{\sqrt{2}K_d}$ is $\frac{K_1}{\left(\sqrt{2}K_d\right)^2}$ times the result of (4-16).The systems of **Figures 4-1** and **4- 4** are equivalent, so that either can be analyzed to find the tracking loop's noise and dynamic tracking performance. To illustrate the equivalence of the two systems, the loop filter was characterized by its impulse response and the VCO was characterized by an integral of its control voltage. These functions can also be characterized by the Laplace transform relationships of (4-19) and (4-21). The Laplace transform is a linear operation, so that the input difference circuit can be characterized by the difference of the Laplace transforms of $T_d(t)/T_c$ and $\frac{\hat{T}_d(t)}{T_c}$. Using transform notation, the system of Figure 4-4 can be represented by

the system of **Figure 4-5.**The noise process n'(t) is not Gaussian. The pdf of n'(t) contains an impulse function at zero amplitude. Since noise of zero amplitude has no effect on tracking, and since the remaining pdf is Gaussian, negligible errors will be obtained if the Gaussian analysis is used.The tracking loop output $\hat{T}_d(s)$ can be related directly to its input $T_d(s)$ using **Figure 4-5**. This relationship is written by inspection and is

$$\frac{\hat{T}_d(s)}{T_c} = \frac{T_d(s) - \hat{T}_d(s)}{T_c}\left[K_d g_c \frac{F(s)}{s}\right] \text{ ----------(4-25)}$$

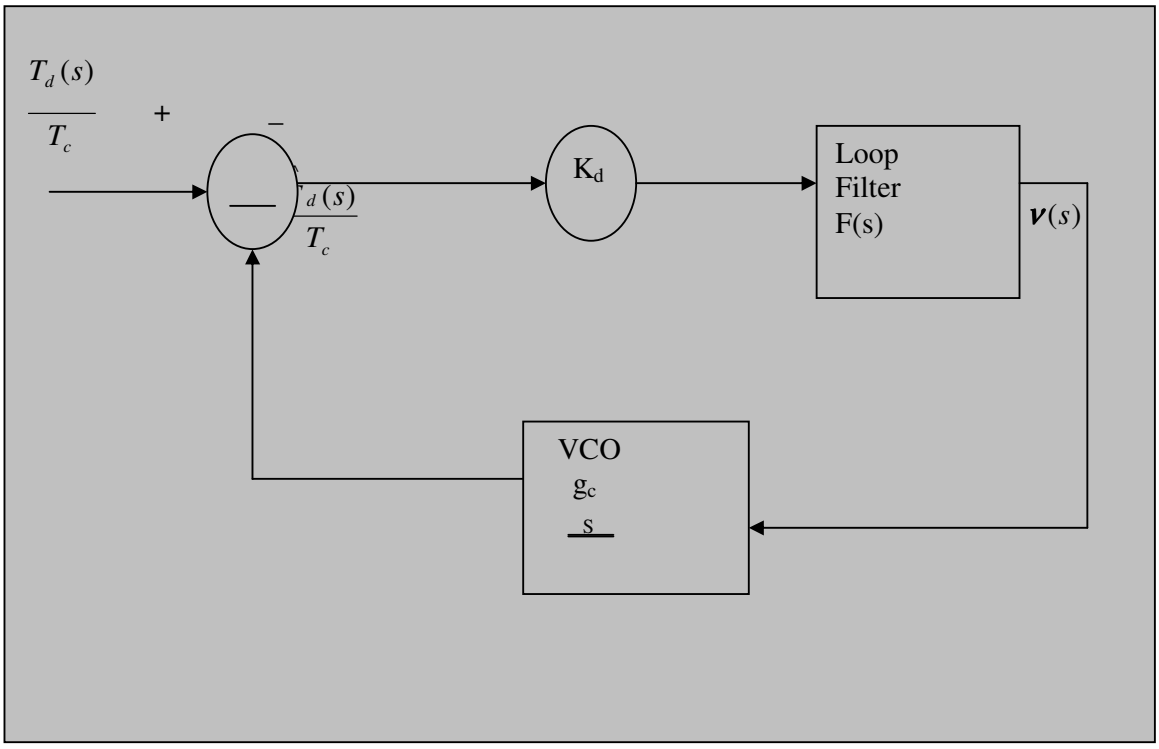


Figure 4.5. Laplace transform model of linear equivalent circuit for baseband full-time early late tracking

Solving this equation for $\frac{\hat{T}_d(s)}{T_d(s)}$ yields

$$\frac{\hat{T}_d(s)}{T_d(s)} = \frac{K_d g_c F(s)}{s + K_d g_c F(s)} \triangleq H(s) \qquad \text{----------(4-26)}$$

and solving for $\frac{\hat{T}_d(s) - T_d(s)}{T_c}$ which is the code tracking error, yields

$$\frac{T_d(s) - \hat{T}_d(s)}{T_c} = \frac{T_d(s)}{T_c}\left[\frac{s}{s + K_d g_c F(s)}\right] \qquad \text{----------(4-27)}$$

These equations are the classical closed-loop servomechanism equations and from this point on the analysis of this tracking loop is identical to the analysis of any other servo loop. In particular, the response of the tracking loop to steps, ramps, and parabolas of input phase can be determined for various loop filter configurations. In addition, the model of **Figure 4-5** is the same, except for the definition of some of the constants, as the PLL model, so that much PLL analysis also applies to the code tracking loop.

A result that is particularly important to the spread-spectrum system designer is the relationship between the mean-square tracking error or tracking jitter and the received signal-to-noise ratio in the loop bandwidth. The power spectrum of the tracking jitter is given by

$$S_{\delta}(f) = \left|H(j2\pi f)\right|^2 S_{n''}(f) \qquad \text{----------(4.28)}$$

where H(s) is the closed-loop transfer function defined by (4-26), and $S_{n''}(f)$ is the two sided power spectrum of the Gaussian noise process at the input to the loop model of **Figure 4-4**. The power spectrum of n"(t) is $\frac{K^2_1}{K^2_d}$ times the result of (4-16) and is

$$S_{n''}(f) = \frac{1}{2}\left(\frac{K_1}{K_d}\right)^2 N_0\left(1+\frac{1}{N}\right) \quad \text{----------for } \Delta \geq 1.0$$

-----(4-29)

$$= \frac{1}{2}\left(\frac{K_1}{K_d}\right)^2 N_0 \Delta\left(1+\frac{1}{N}\right) \quad \text{---------- for } \Delta < 1.0$$

Then the variance of δ which is denoted σ^2_δ is

$$\sigma_\delta^{\,2} = \int_{-\infty}^{\infty} S_{n''}(f)\left|H(j2\pi f)\right|^2 df \qquad \text{----------(4.30)}$$

Since $S_{n''}(f)$ is constant over all f [i.e., n''(t) is a white noise process],

$$\sigma^2\delta = S_{n''}(f)\int_{-\infty}^{\infty} \left|H(j2\pi f)\right|^2 df \quad \text{----------(4-31)}$$

The integral on the right side of this equation is defined as the two-sided noise bandwidth

$$W_L = \int_{-\infty}^{\infty}\left|H(j2\pi f)\right|^2 df \quad \text{----------(4.32)}$$

$$\sigma_\delta^{\,2} = \frac{1}{2}\left(\frac{K_1}{K_d}\right)N_0\left(1+\frac{1}{N}\right)W_L \qquad \Delta \geq 1.0$$

------(4 - 33)

$$= \frac{1}{2}\left(\frac{K_1}{K_d}\right)N_0\Delta\left(1+\frac{1}{N}\right)W_L \qquad \Delta \leq 1.0$$

The units of W_L are hertz. The final result for tracking jitter is then .Observe that the tracking error variance is reduced as Δ decreases for Δ ≤1.0.This can be explained by noticing that as Δ decreases, the slope of the discriminator characteristic remains constant

while some of the input noise cancels in (4-10) when the early and late codes overlap. Evaluating K_d as a function of K_1, P, and N in (4-33), and defining $\rho_L = \frac{2P}{N_0 W_L}$ yields

$$\sigma^2_{\delta} = \frac{1}{2\rho_L}\left(\frac{N}{N+1}\right) \quad \text{for } \Delta \geq 1.0$$

$$= \frac{1}{2\rho_L}\left(\frac{N}{N+1}\right)\Delta \quad \text{for } \Delta < 1.0 \qquad \text{------(4-34)}$$

where ρ_L is the signal-to-noise ratio in the loop bandwidth. A commonly used tracking loop filter is the simple lead-lag filter whose transfer function is

$$F(s) = \frac{1+\tau_2 s}{\tau_1 s}$$

With this loop filter, the closed-loop transfer function of (4-26) becomes

$$H(s) = \frac{K_d g_c\left(\frac{\tau_2}{\tau_1}\right)s + \frac{K_d g_c}{\tau_1}}{s^2 + \left[K_d g_c\left(\frac{\tau_2}{\tau_1}\right)\right]s + \frac{K_d g_c}{\tau_1}}$$

$$H(s) = \frac{2\xi\omega_n s + \omega^2_n}{s^2 + 2\xi\omega_n s + \omega^2_n} \qquad \text{----------(4-36)}$$

where the usual loop natural frequency and damping factor have been defined by

$$\omega_n = \sqrt{\frac{K_d g_c}{\tau_1}} \qquad \text{----------(4-39)}$$

$$\xi = \frac{\tau_2}{2}\omega_n \qquad \text{----------(4-40)}$$

Since K_d is a function of the input signal level, both K_d and ξ are functions of P .This implies that automatic gain control is an important issue in tracking loop design. The two-sided noise bandwidth for a second-order loop with this loop filter is given by

$$W_L = \varpi_n\left(\xi + \frac{1}{4\xi}\right)$$

This result is derived using contour integration to evaluate (4-32). The units of W_L are Hertz. This completes the analysis of the baseband full-time early-late tracking loop.

5.Discussion,Results and Conclusion

5.1 Generation ,Transmition and Reception of input data

1. In the project data is input to the system from the keyboard
2. The input data is then converted to binary digits by the use of

- American Standard Code for Information Interchange(ASCII)
- Non Return to Zero (NRZ) waveform

Input of the system

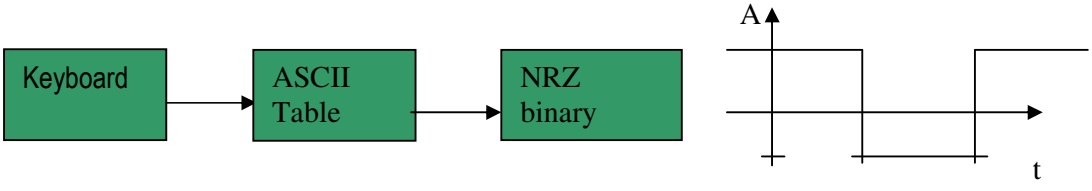


Table 5-1 The ASCII code table

		B_7		0	0	1	1	1	1
		B_6		1	1	0	0	1	1
B_1	B_2	B_3 \ B_5	B_4	0	1	0	1	0	1
0	0	0	0	SP	0	@	P	‘	p
0	0	0	1	(	8	H	X	H	x
0	0	1	0	$	4	D	T	D	t
0	0	1	1	,	<	L	/	L	∫
0	1	0	0	“	2	B	R	B	r
0	1	0	1	*	:	J	Z	J	z
0	1	1	0	&	6	F	V	F	v
0	1	1	1	.	>	N	^	N	~
1	0	0	0	!	1	A	Q	A	q
1	0	0	1	)	9	I	Y	I	y
1	0	1	0	%	5	E	U	E	u
1	0	1	1	_	=	M	[	M	}
1	1	0	0	#	3	C	S	C	s
1	1	0	1	+	;	K	]	K	{
1	1	1	0	‘	7	G	W	G	w
1	1	1	1	/	?	O	-	o	DEL

This is the title page of the project

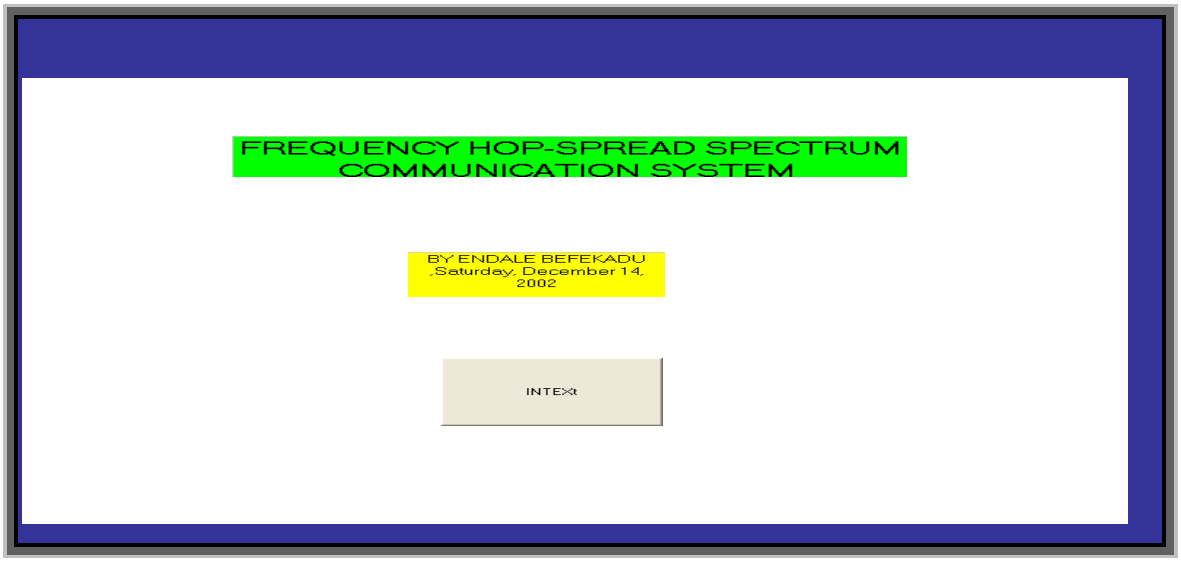


Figure 5.1. The title page of the project

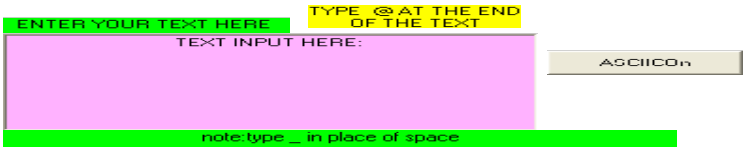


Figure 5-2 The Input window where you type the text

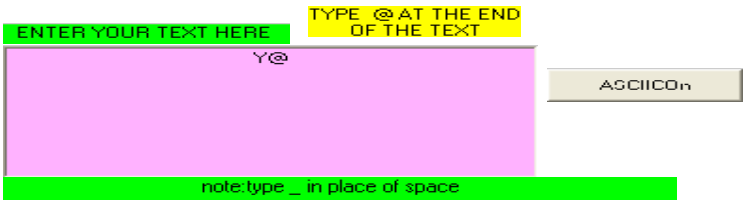


Figure 5.3. The Input Window with the Letter Y Written On to It

It can be typed any sentence you like on the input winow.Once the input data has been typed, the *ASCIICOn* button is clicked. Then the following figure, which is, the ASCII Output appears on the Figure.

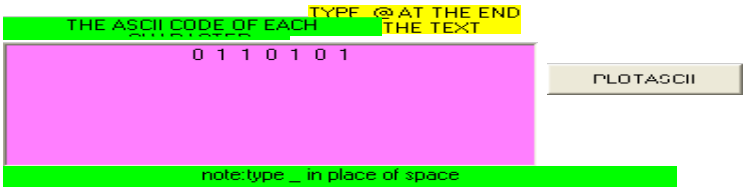


Figure 5.4. The ASCII conversion of the Letter Y

Then the ASCII converted bits are drawn in the next step. For example in the figure below the binary data input is plotted for the sentence

Frequency Hop Spread Spectrum Communication System@

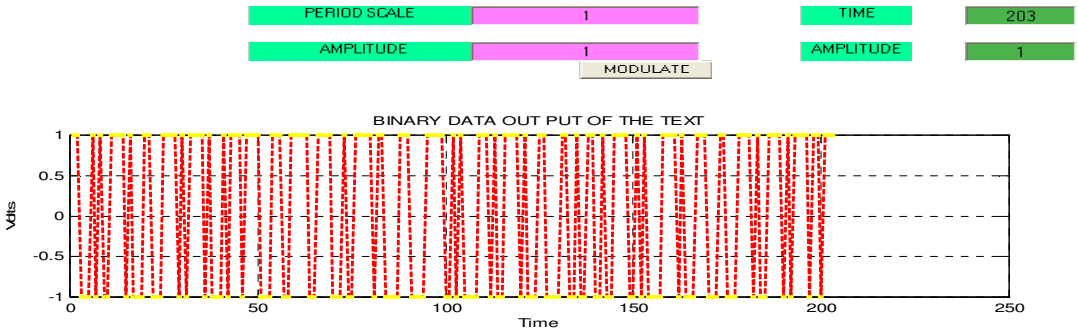
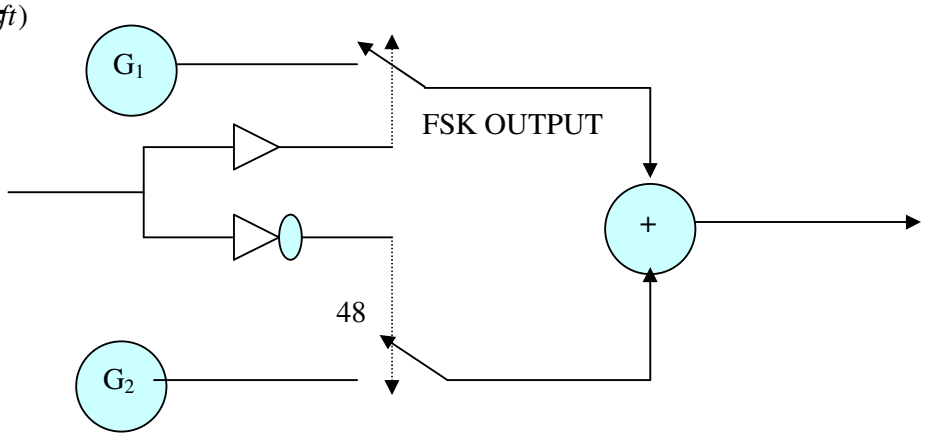


Figure 5.5 The plot of the data input

The units of the time are in micro-seconds and the units of the amplitude are in Volts. Then the binary data output is modulated using the frequency shift keying method. The frequency shift keying system is achieved as the superpositions of two ASK modulators it shown below:

IF"0"

$A\sin(2\pi ft)$



IF "1"

$A\sin(2\pi(f + delf)t)$

Figure 5.6 FSK as superposition of two ASK waveforms

The output of the FSK modulator is shown below: before '*modulating the input data*' the frequency with which to modulate the data is input by one of the two methods

- The first is the poup menu
- The second is by editing on an input dialogue box

The frequency spectrums which can be input to the modulator are from: 50MHz to 450MHz.Once the the frequency is input to the modulator the FSK button is clicked

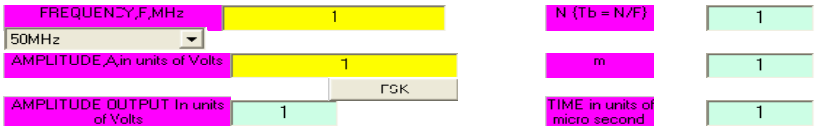


Figure 5.7.The frequency Input to the modulator

The frequency input to the system is 50MHz.Then the FSK modulator output is shown below:

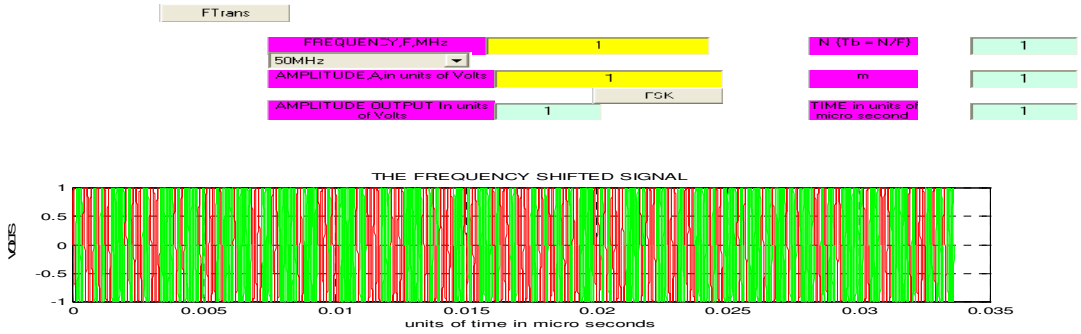


Figure 5.8. The frequency shift keying modulator output Then

the Fourier Transform of the modulated signal is then obtained.

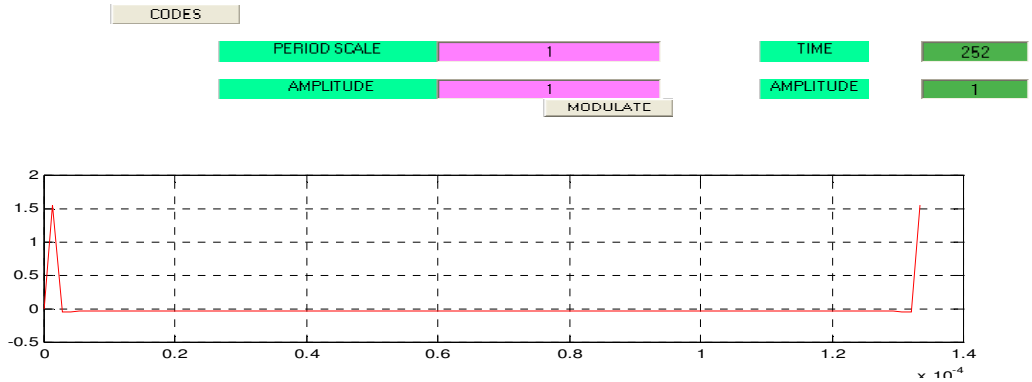


Figure 5.9. The Fourier transform of the FSK modulator output.

When 50MHz is the input frequency the two frequencies used are 50MHz and 100MHz.Therefore the Fourier transform is tow delta functions at their respective frequencies.The CODES button is clicked .Then the three codes $c_1(t), c_2(t)$ and $c_3(t)$ are generated. To plot the three generated codes the PLOTCODES1 key is pressed resulting the figure shown below.

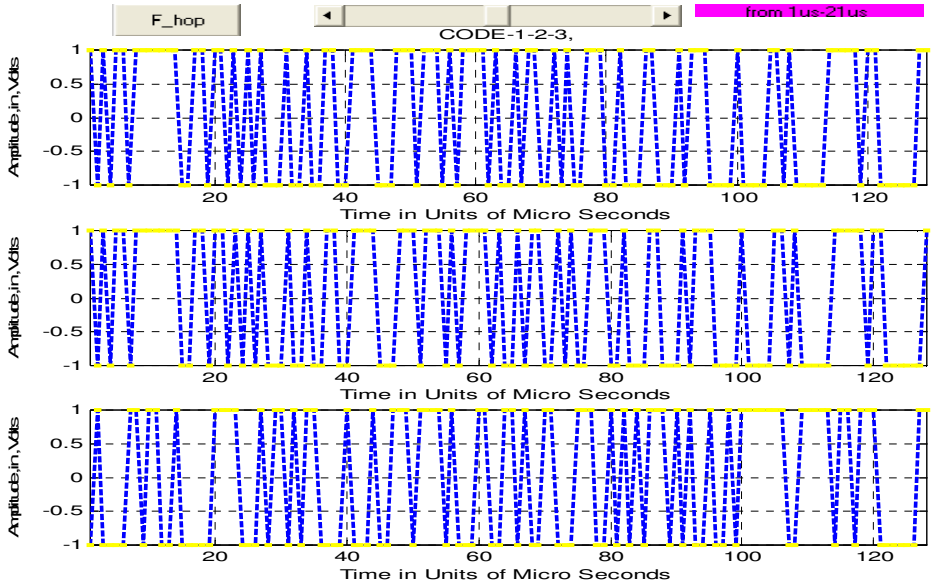


Figure 5.10 The three codes generated

The frequency hopped signal is shown in the figure below:

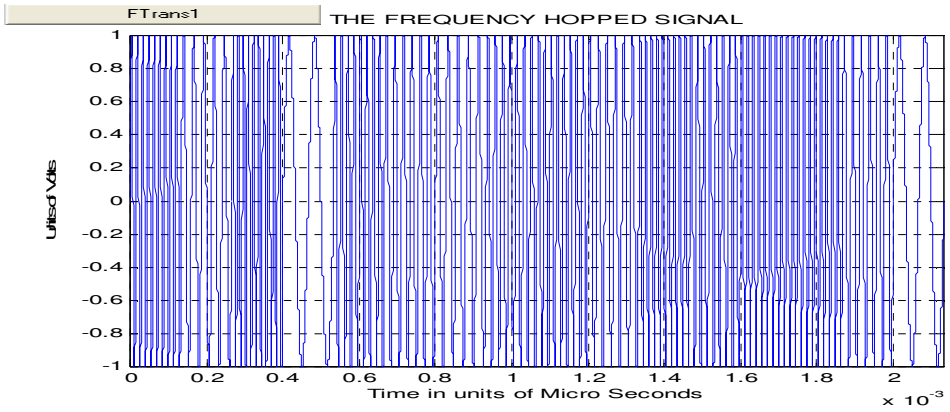


Figure 5.11 The frequency hopped signal

The frequency hopped signal is passed through a Fourier transform as is shown below:

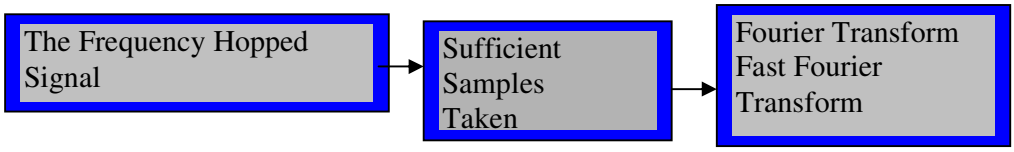


Figure 5-12 The algorithm that is employed to compute the Fourier Transform of the Frequency Hopped Signal

The Fourier transform is shown below:

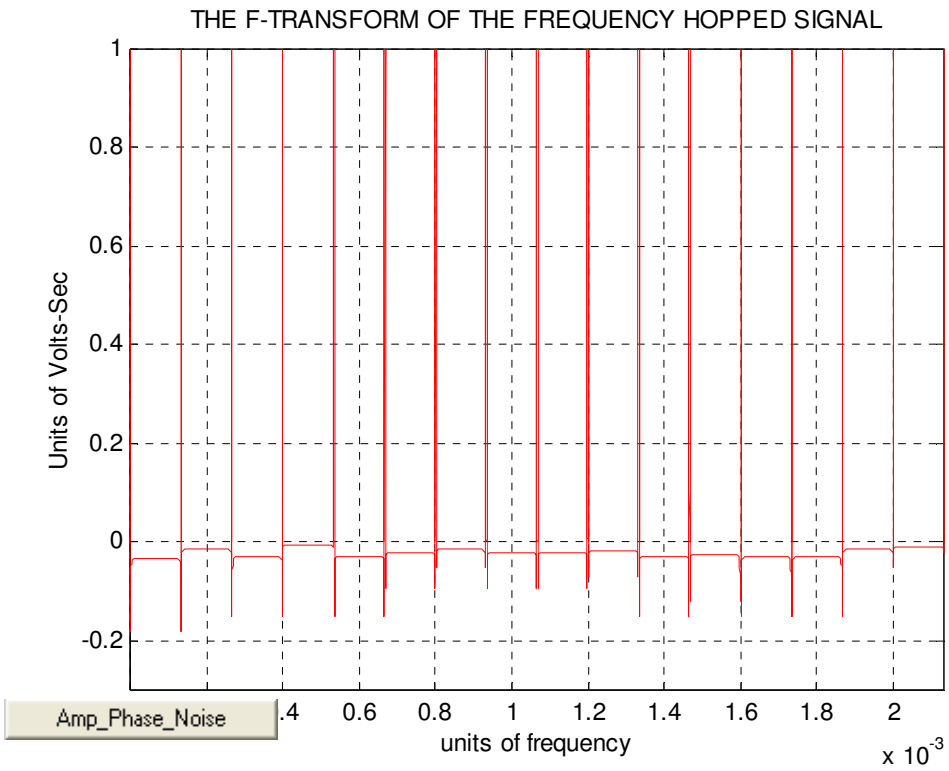


Figure 5-13 The Fourier transform of the Frequency Hopped Signal

Next the amplitude and the phase noises are generated:

- The Amplitude noise is white Gaussian noise with zero mean and with variance equal 100XA.
- The phase noise is uniformly distributed the interval 0 and 0.3.

The results of the computer generated noises are shown below:

The signal is of the form:

$s(t) = A\sin(\varpi_n t + \phi(t)) + \xi(t)$ where ϖ_n is the hoped frequency , $\phi(t)$ is the phase noise and $\xi(t)$ is the amplitude noise:

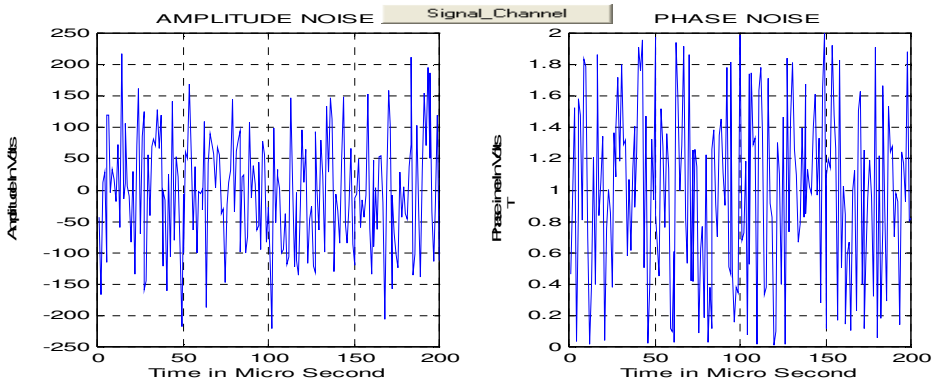


Figure 5-14 The Channel Noises

The next step in the project is to simulate the signals that are necessary for the *initial synchronization* of the phase:

In block diagram the procedure is shown below:

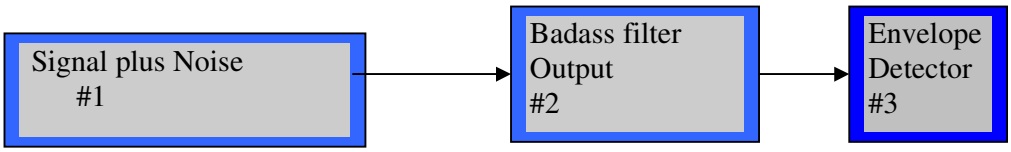


Figure 5-15 Pre-Signals for Initial Synchronization of the phase

These are the signals of the above block diagrams numbered #1, #2 and #3

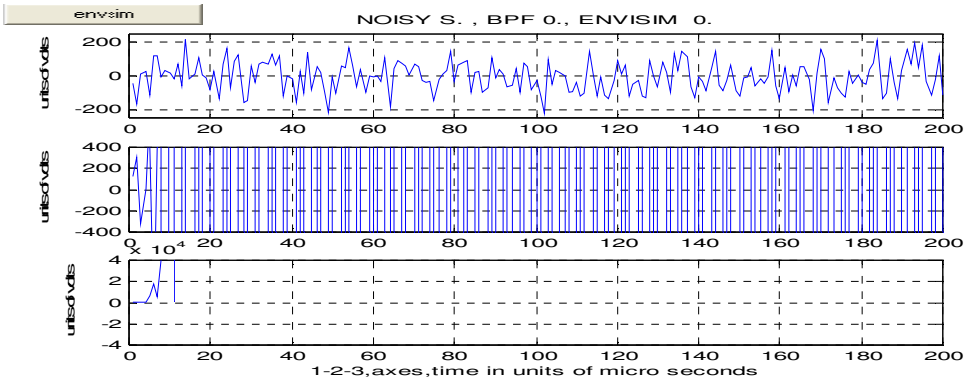


Figure 5-16 The signals required before the initial synchronization

The envelope detector used in the project is shown below:

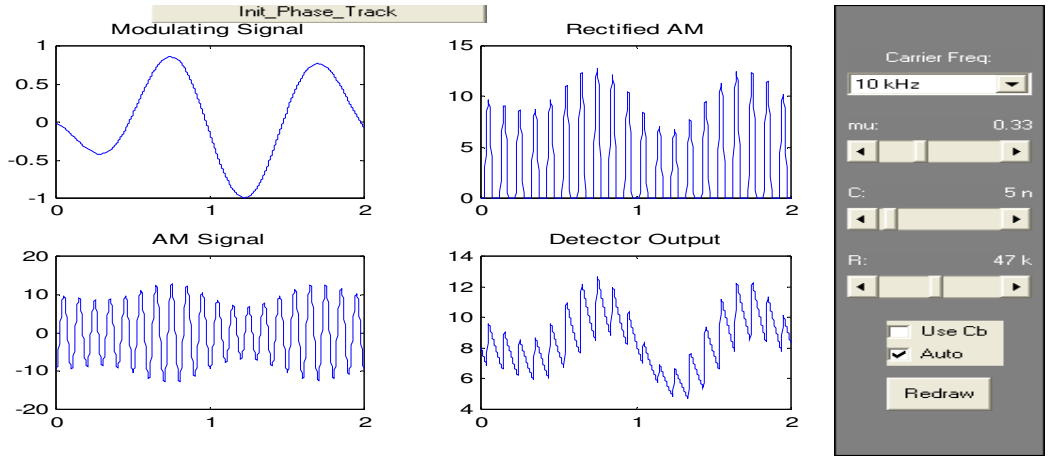


Figure 5-17 The envelope detector output

The initial synchronization output of the initial synchronizer is shown below:

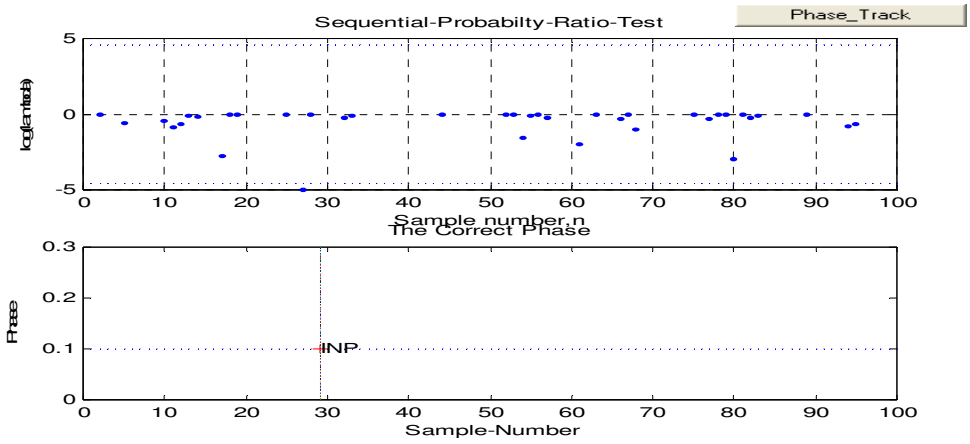


Figure 5-18 The correct initial phase is shown as INP

The outputs of the phase tracker are shown below:

1. The first figure is the phase noise
2. The second figure is the phase difference
3. The third figure is the tracked phase
4. The fourth figure is the value of υ

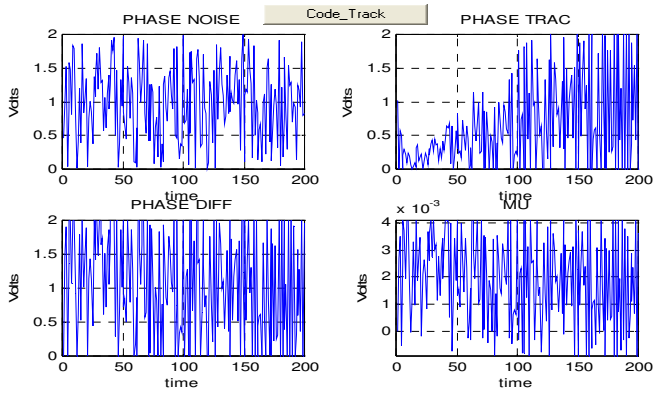


Figure 5-19 The outputs of the code tracker

Finally in the Figure below the signal to be tracked and the out put of the tracker

Unlike the phase tracking process ,in this case the actual overall sinusoidal signal is tracked.

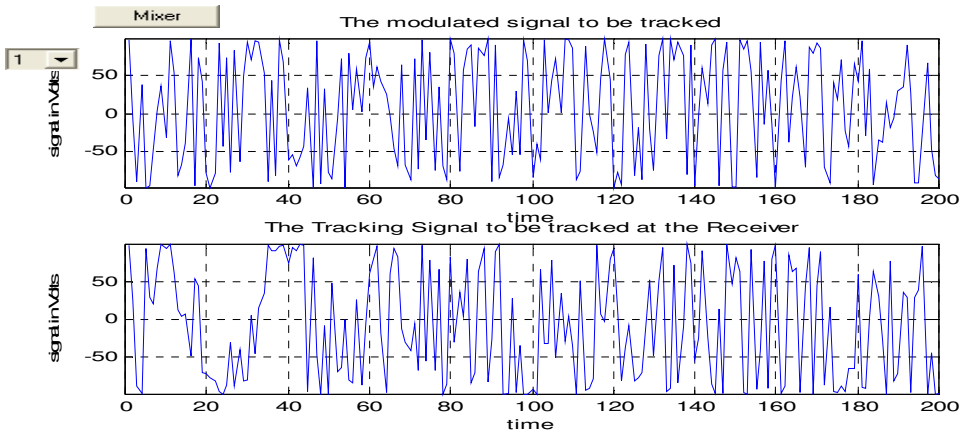


Figure 5-20 The Hopped Signal in a dual noise is tracked.

The output of the mixer is shown below:

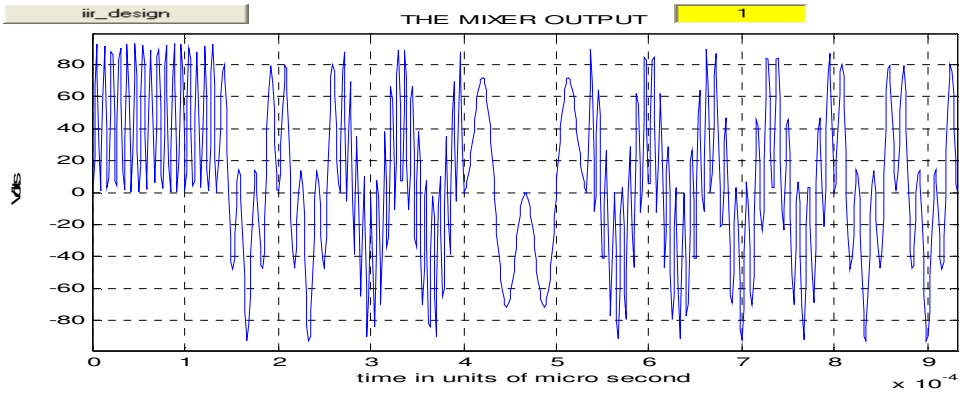


Figure 5-21.The out put of the Mixer

The iir_design prompts the sentences below:

Enter order of prototype filter =>30

Enter one of butt/bess/cheb1/cheb2/elli =>elli

Enter passband ripple in dB =>0.1

Enter stopband attinuation in dB =>100

The results of the prompts are shown below:

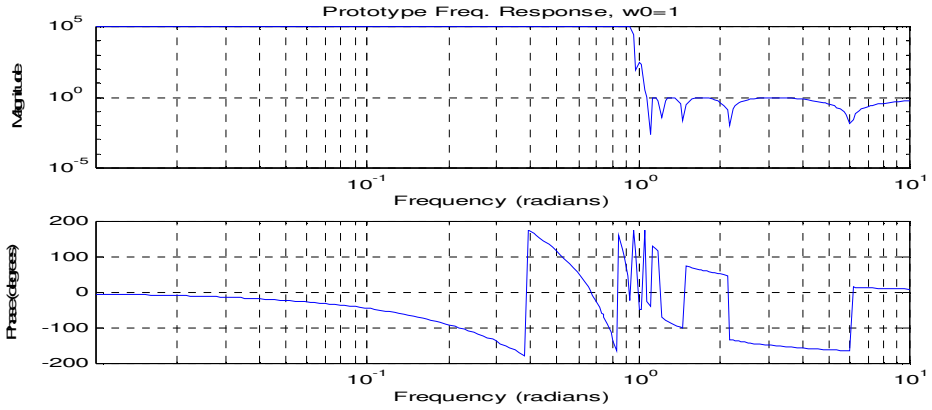


Figure 5-22 The Magnitude and the phase outputs of the particular type of filter with the specified parameters

The pole of the protype Filter is shown below:

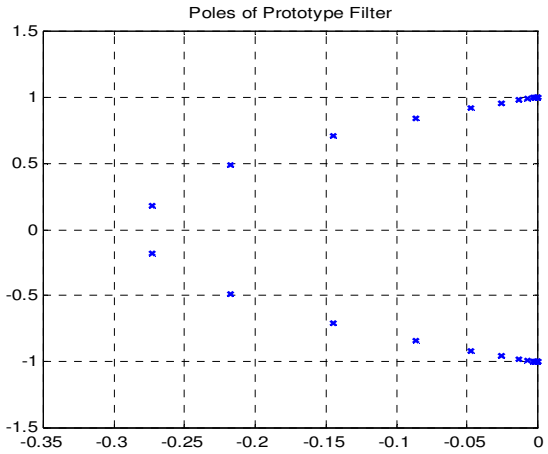


Figure 5-23 The poles of the prototype filter

Enter freq-BAND conversion to lp/hp/bp/bs/none =>bp

Enter new cutoff freq in Hz =>100

Enter Bandwidth in Hz =>200

Next the above prompts are shown, resulting the figures below:

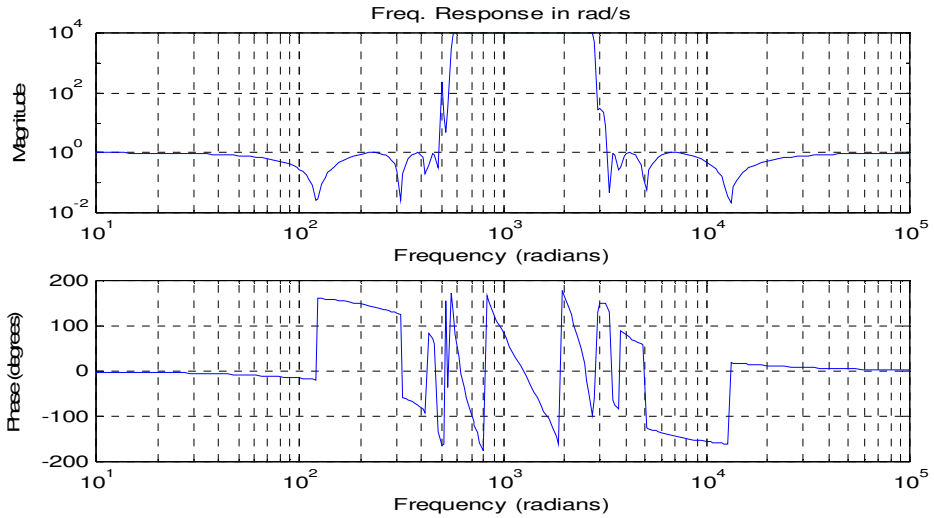


Figure 5-24 The Magnitude and the Phase frequency response

Enter bilinear/impulse invariant (bil/imp) =>bil

Enter sampling frequency in Hz =>300

After entering the above prompt the following result is obtained:

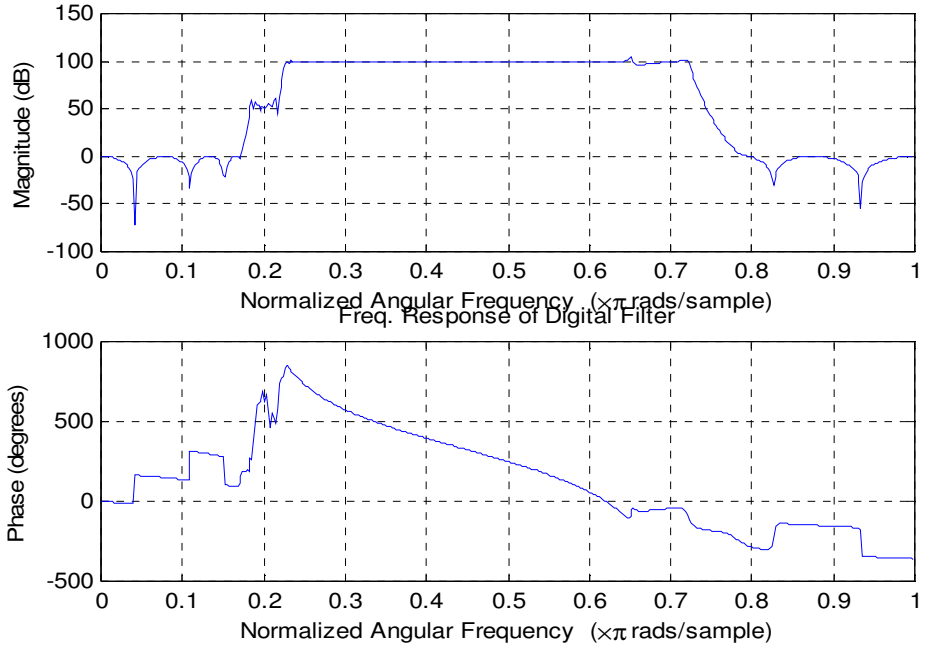


Figure 5-25 Amplitude and Phase response of the Digital Filter

The poles and zeros of the digital filter are shown below:

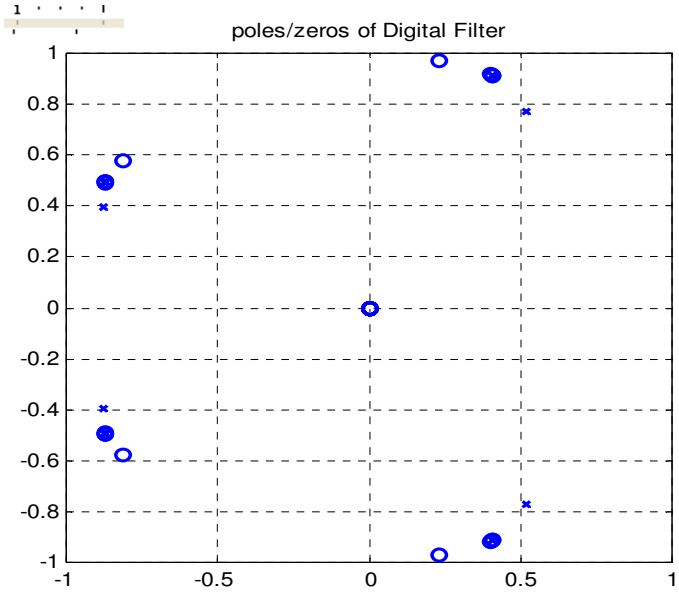


Figure 5-26 The poles and the zeros of the digital filter

The output of the bandpass filter is shown below:

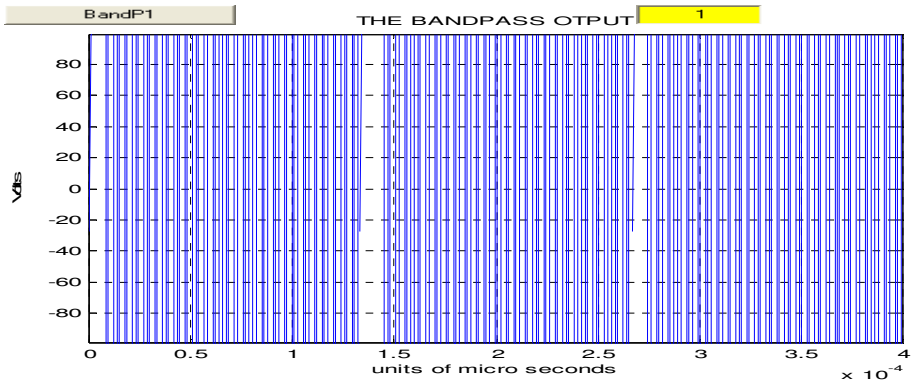


Figure 5-27 The output of the bandpass filter output

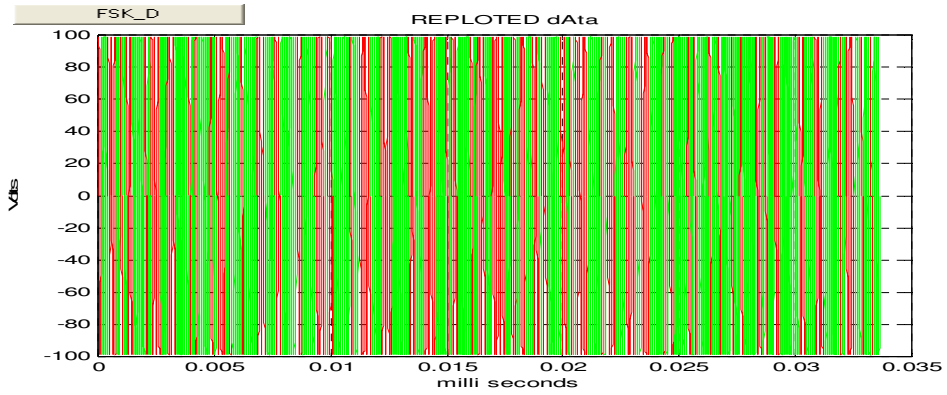


Figure 5-28 The output of bandpass filter repotted

The frequency shift keying detector used in the projects is the type of the matched filter and is shown below:

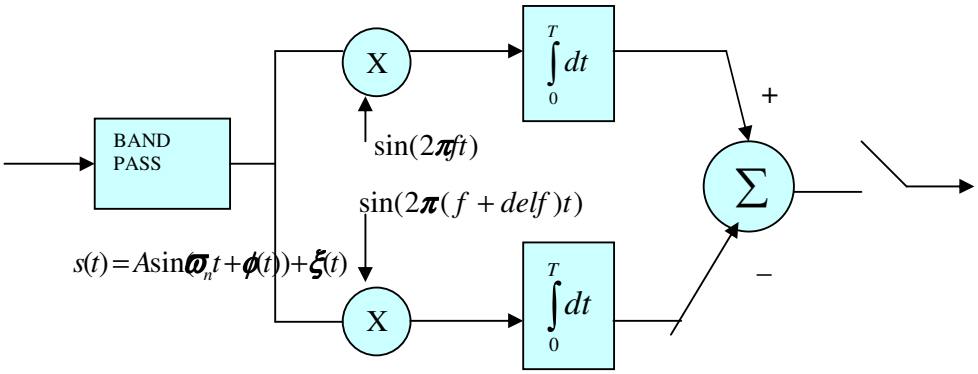


Figure 5-29 Matched Filter Detectors for FSK

The output of the FSK detector is shown below:

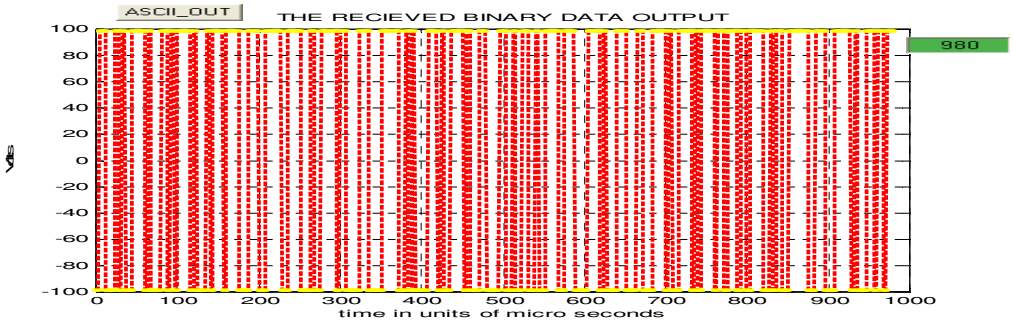


Figure 5.30 The output of the FSK DETECOR

Next the output of the FSK DETECTOR is converted to ASCII code. The result is shown below:

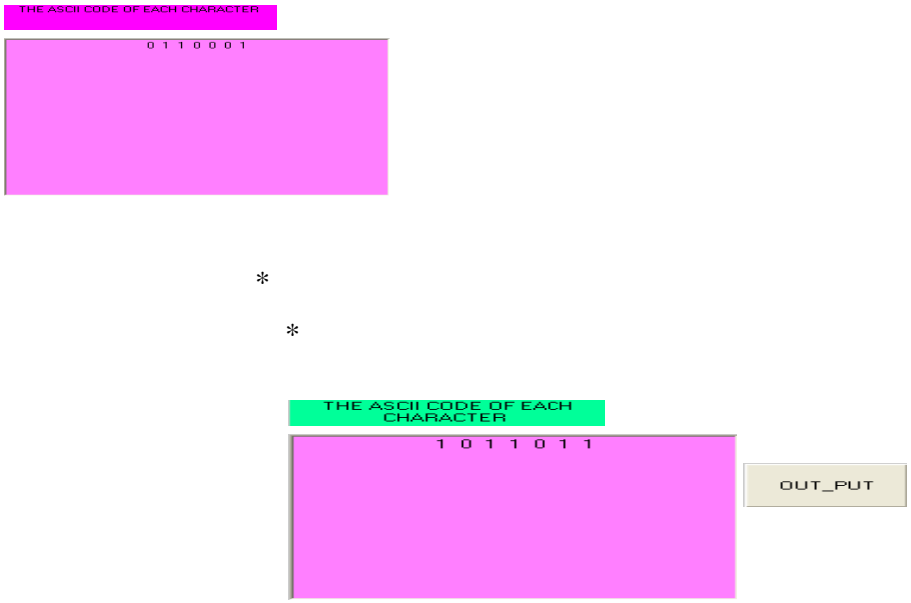


Figure 5.31. The ASCII converted data output

Next the final output of the text is shown below.

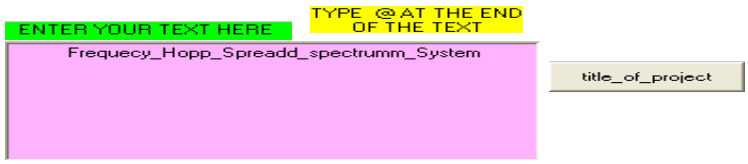


Figure 5-32 The final output text

5.2 Application and Findings

5.2.1 An efficient use of bandwidth (CDMA)

When we have a limited bandwidth resource to transmit through, the use of spread spectrum communication is essential. One central station wants to transmits to a number of sub-stations. The channel has a bandwidth capacity of W_{max}, if the signal is transmitted for each of the receivers separately .If we have N receivers the channel bandwidth required would be N W_{max}.But now we like to transmit the information through only $W_{max.}$To achieve this we use the Slow Frequency Hop Spectrum Spectrum Communication System. This is done by giving each of the N Sub-stations a different code to use with. Let these codes be $C_1(t), C_2(t)...C_N(t)$.Then we can transmit the data through only the available bandwidth.Otherwise if all the signals are transmitted through the available bandwidth using the conventional method:

(a) the signals would interfere with each other

(b) one substation may get the signal which was intended to the other substation.

But these problems rae erradicated using spread spectrum communication system.

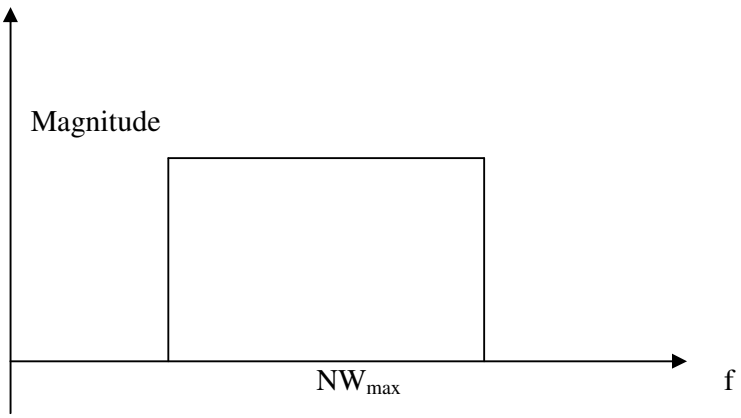


Figure 5.32 (a) the bandwidth required when the conventional Communication system is used.

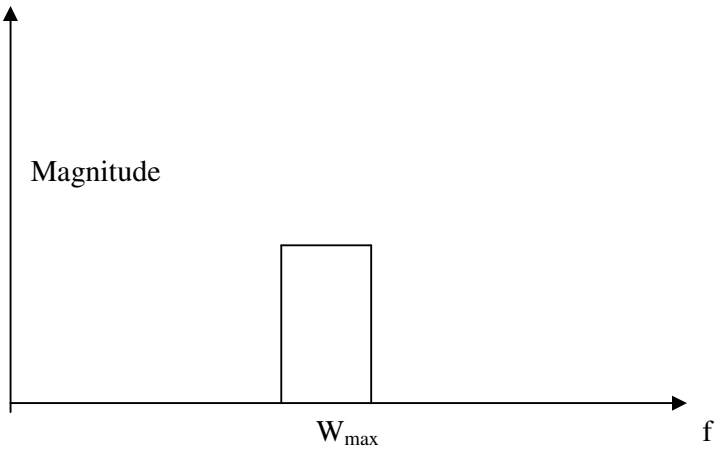


Figure 5.32 (b) the bandwidth required when the Spread Spectrum Communication system is used.

There is a bandwidth gain of $NW_{max}/W_{max}=N$

5.2.2 Security Communication

The other application of Spread Spectrum Communication System is security communication. When we want to communicate information secretly the use of Spread Spectrum Communication System is one of the intelligent options .By assigning a different code to each the parities it is possible to communicate as secretly as wanted .Any of the participants can not access the other participant's information because they don't have the code. So that they can get the only information they are assigned to.

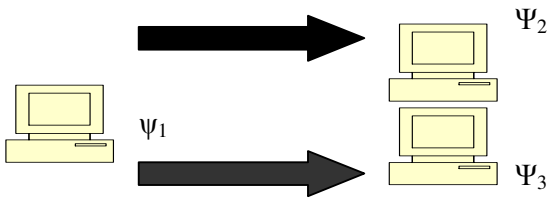


Figure 5 -33 Security Communications

In the above Communication system when ψ_1 communicates with ψ_2, ψ_3 can not get the information. Also when ψ_1 communicates with ψ_3, ψ_2 can not get the information..Becuase they have a different code.

5.2.3 Conclusion

As seen a whole the receiver part is more complex than the transmitter one,because of the inclusion of code acquisition and code tracking.The parameters that are needed for the system such as probability of detection P_d,probability of flase alarm P_{fa},the treshholds A and B are computed manually.

The computer simulation agree very much with the actual designed values.In the initial synchronization the program has to be run many times before the upper thresholds is reached .The calculated values has to accumulate many times before reaching the threshold.

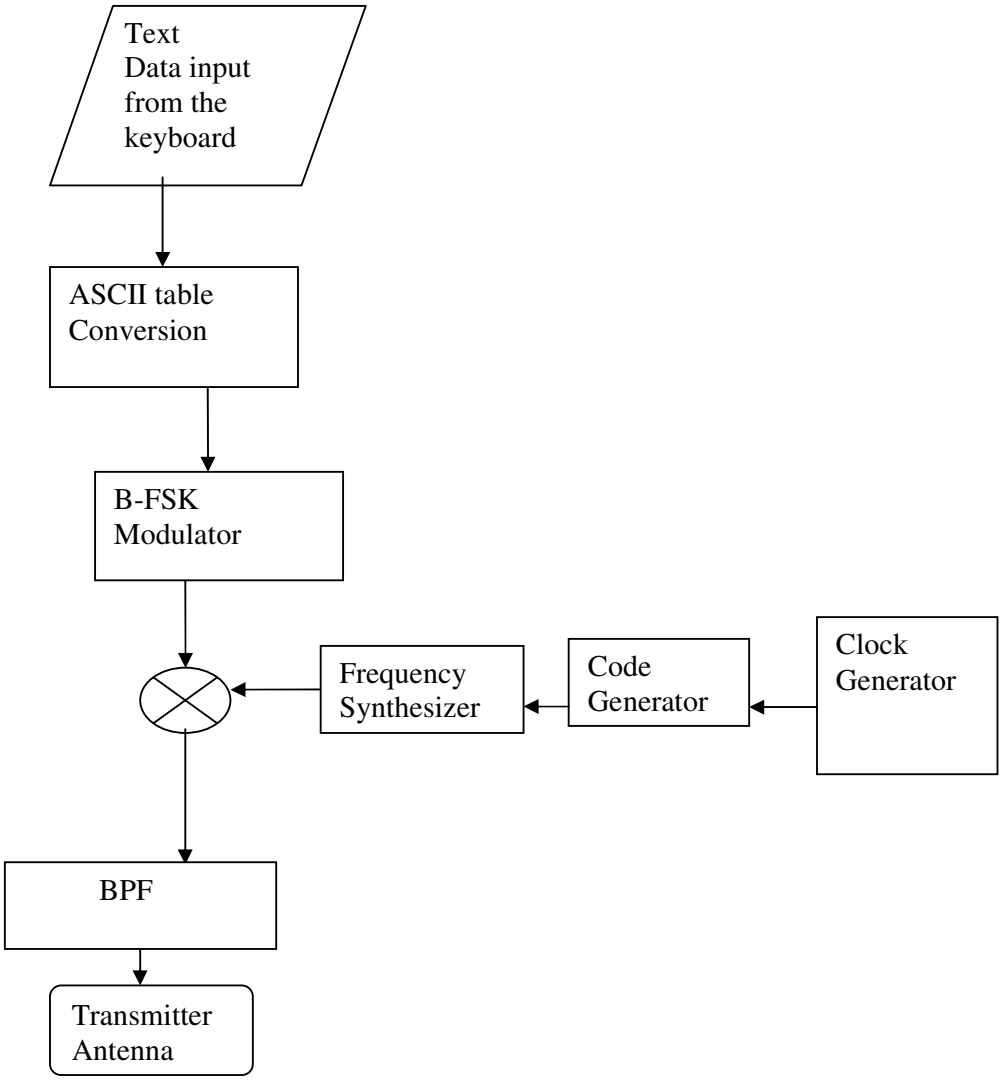
The software can be easily implemented in actual real life problems to secretly to communicate written materials such as examinations only between the concerned parties.

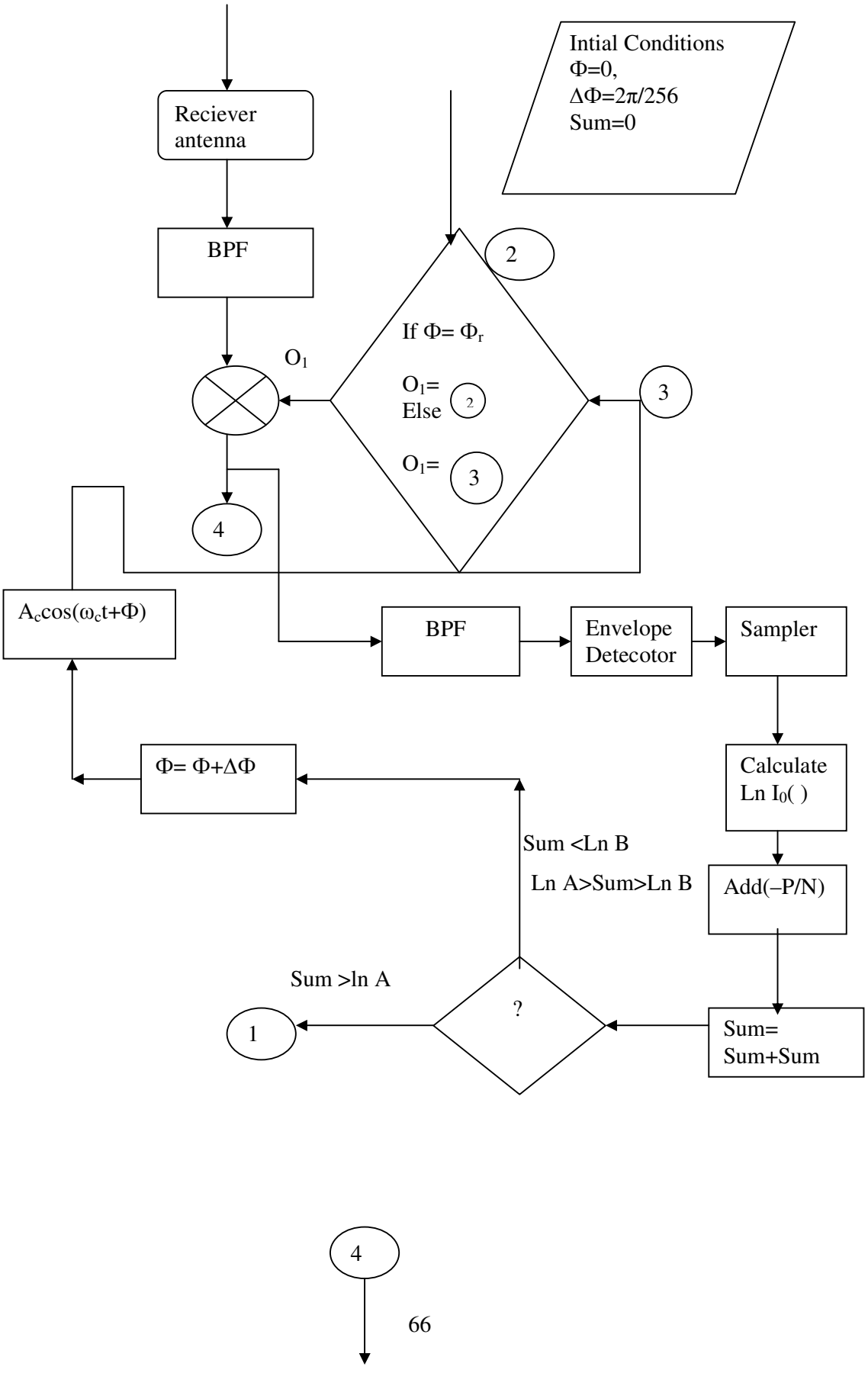
It can also be applied in areas where thre is a scarcity of bandwidth .For example ,in broadcasting ,one can receive different broadcasts on the same bandwidth by simply changing the codes.

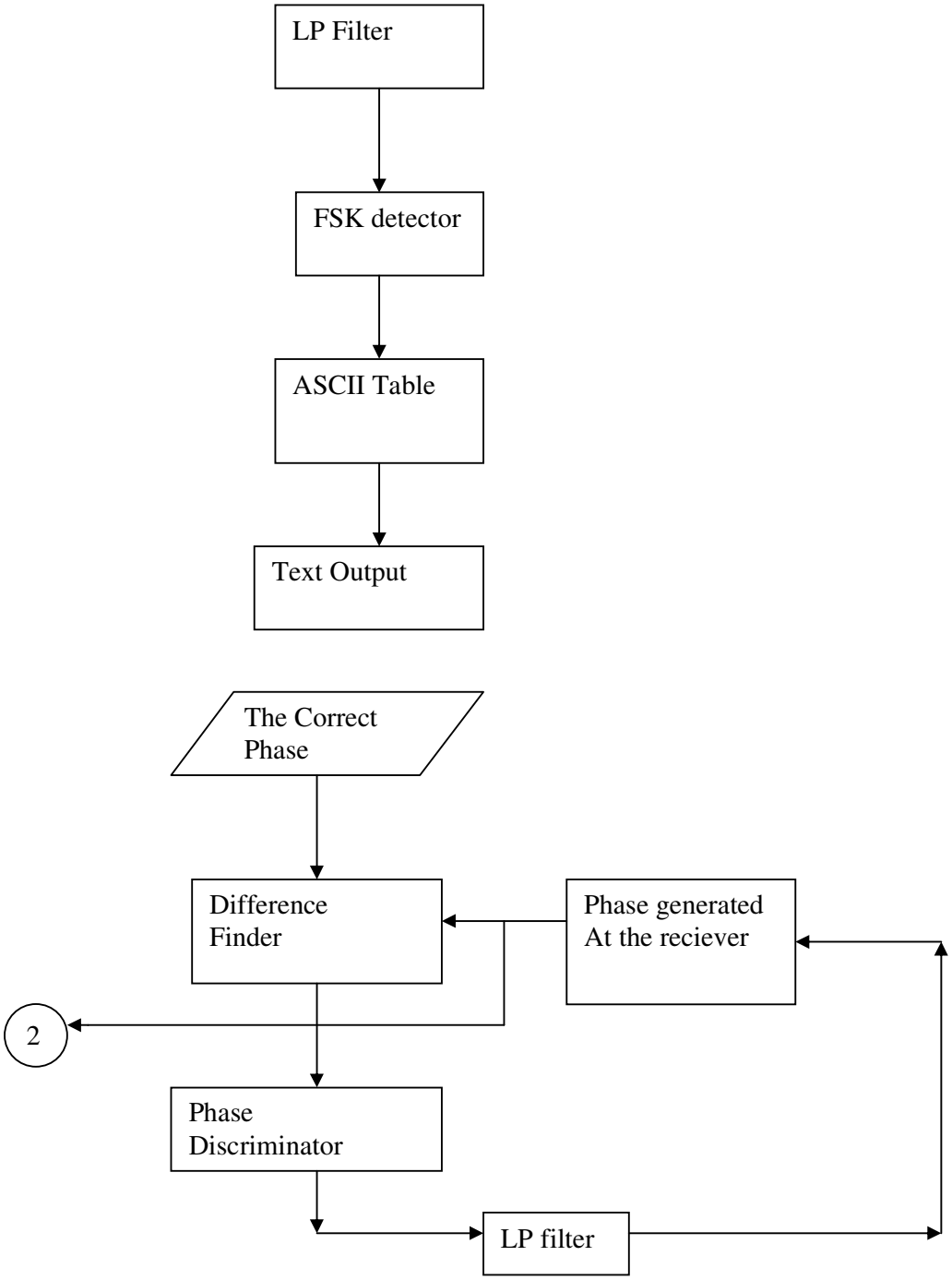
Spread spectrum communication being a recent area of scientific research ,the project lays a strong foundation for further study in this area.Those who want to go into deep study about the topic can be very much benefited from this thesis.

Appendix A

Flow Chart Of The Main Progarms.




Intial Conditions
Φ=0,
ΔΦ=2π/256
Sum=0
Reciever antenna
BPF
2
If Φ= Φr
O1
O1= Else 2
O1= 3
3
4
Ac cos(ωc t+Φ)
BPF
Envelope Detecotor
Sampler
Φ= Φ+ΔΦ
Calculate Ln I0( )
Sum <Ln B
Ln A>Sum>Ln B
Add(–P/N)
Sum >ln A
?
1
Sum= Sum+Sum
4


LP Filter
FSK detector
ASCII Table
Text Output
The Correct Phase
Difference Finder
Phase generated At the reciever
2
Phase Discriminator
LP filter

Appenidx B

These are the MatLab Program Written to generate the Spread Spectrum Communication System

File 1 (title_of_project)

```
%This is a Matlab program that creates a title on the computer screen
%*********************************************************************
f = figure ('pos', [4 35 1050 690]);%to postion the figure
%onto which the title appears
%*********************************************************************
postx1 = [.19 .78 .61 .09]; % the position for  the title
postx2 = [.35 .51 .232 .1];bkcolofed = [1 0.7 1];% the type of the color of
the input text
bkcoloftx=[0 1 0];% the color of the text message
bkcoloftx2=[1 1 0];% the color of the text message
%*********************************************************************
tx1 = uicontrol('style','text','units',...
   'normalized','position',postx1,...
   'string','FREQUENCY    HOP-SPREAD    SPECTRUM    COMMUNICATION    SYSTEM
','tag','text1','fontsize',20,'BackgroundColor',bkcoloftx);
tx2 = uicontrol('style','text','units',...
   'normalized','position',postx2,...
    'string','BY    ENDALE    BEFEKADU    ,Saturday,    December    14,
2002','tag','text2','fontsize',10,'BackgroundColor',bkcoloftx2);
poscmd1 = [.38 .22 .2 .15]; % display input text command buttoned1 =
uicontrol('style','edit','units',...


cmd1 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd1, ...
      'String','INTEXt');
set(cmd1,'callback','INTEXt');% to call the program INTEXt.m
```

File 2 (INTEXt.m)

```
%*********************************************************************
%This is a Matlab program that accepts a text from the keyboard
%You have to type @ at the end of each text
%*************************************************************
f = figure ('pos', [50 300 490 190]);%to postion the figure
%onto which the user types his text
%*************************************************************
posed1 = [.19 .19 .58 .58]; %  the  postion for input TEXT
postx1 = [.19 .78 .31 .09]; % the position for  input message for TEXT
postx2 = [.521 .81 .232 .132];%what should we type at the end of the TEXT
postx3 = [.19 .091 .732 .092];%what should we type at the end of the TEXT
bkcolofed = [1 0.7 1];% the type of the color of the input text
bkcoloftx=[0 1 0];% the color of the text message
bkcoloftx2=[1 1 0];% the color of the text message
bkcoloftx3=[0 1 0];% the color of the space rule

%*************************************************************
ed1 = uicontrol('style','edit','units',...
   'normalized','position',posed1,...
   'string','TEXT                    INPUT                    HERE:
','tag','edit1','fontsize',7,'BackgroundColor',bkcolofed);
```

```
tx1 = uicontrol('style','text','units',...
   'normalized','position',postx1,...
   'string','ENTER               YOUR               TEXT               HERE
','tag','text1','fontsize',6,'BackgroundColor',bkcoloftx);
tx2 = uicontrol('style','text','units',...
   'normalized','position',postx2,...
    'string','TYPE            @      AT      THE      END      OF      THE
TEXT','tag','text2','fontsize',4,'BackgroundColor',bkcoloftx2);
tx3 = uicontrol('style','text','units',...
   'normalized','position',postx3,...
    'string','      note:type      _      in      place      of      space
','tag','text3','fontsize',2,'BackgroundColor',bkcoloftx3);
poscmd1 = [.78 .52 .212 .15]; % display input text command buttoned1 =
uicontrol('style','edit','units',...


cmd1 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd1, ...
      'String','ASCIICOn');
set(cmd1,'callback','ASCIICOn');% to call the program ASCIICOn
%*********************************************************************
```

File 3 (ASCIIOn)

```
%This is a Matlab program that accepts the text and convert it to ASCII
code
%*********************************************************************
**
%*********************************************************************
**
TEXT=get(ed1,'String');%to convert the input text into string;
count=1;%Initializing the counter;
%*********************************************************************
**
while TEXT(count)~='@'%condtion on the WHILE loop

switch TEXT(count)%the switch loop
%case '_'
    %BIN=[0 0 0 0 0 0 0];

case 'P'
    BIN=[0 0 0 0 1 0 1];
case '0'
    BIN=[0 0 0 0 1 1 0];
case 'p'
    BIN=[0 0 0 0 1 1 1];
case 'H'
    BIN=[0 0 0 1 0 0 1];
case '('
    BIN=[0 0 0 1 0 1 0];
case 'h'
    BIN=[0 0 0 1 0 1 1];
case 'X'
    BIN=[0 0 0 1 1 0 1];
case '8'
    BIN=[0 0 0 1 1 1 0];
case 't'
```

```
    BIN=[0 0 0 1 1 1 1];
case 'D'
    BIN=[0 0 1 0 0 0 1];
case '$'
    BIN=[0 0 1 0 0 1 0];

case 'd'
    BIN=[0 0 1 0 0 1 1];

case 'T'
    BIN=[0 0 1 0 1 0 1];
case '4'
    BIN=[0 0 1 0 1 1 0];

case 'L'
    BIN=[0 0 1 1 0 0 1];
case ','
    BIN=[0 0 1 1 0 1 0];
case 'l'
    BIN=[0 0 1 1 0 1 1];
case '/'
    BIN=[0 0 1 1 1 0 1];
case '<'
    BIN=[0 0 1 1 1 1 0];

case'B'
    BIN=[0 1 0 0 0 0 1];
case'"'
    BIN=[0 1 0 0 0 1 0];
case'b'
    BIN=[0 1 0 0 0 1 1];
case'2'
    BIN=[0 1 0 0 1 1 0 ];
case'r'
    BIN=[0 1 0 0 1 1 1 ];
case'J'
    BIN=[0 1 0 1 0 0 1 ];
case'*'
    BIN=[0 1 0 1 0 1 0 ];
case'j'
    BIN=[0 1 0 1 0 1 1 ];
case'Z'
    BIN=[0 1 0 1 1 0 1 ];
case':'
    BIN=[0 1 0 1 1 1 0 ];
case'z'
    BIN=[0 1 0 1 1 1 1 ];
case'F'
    BIN=[0 1 1 0 0 0 1 ];
case'&'
    BIN=[0 1 1 0 0 1 0 ];
case'f'
    BIN=[0 1 1 0 0 1 1 ];
case'Y'
    BIN=[0 1 1 0 1 0 1 ];
case'6'
    BIN=[0 1 1 0 1 1 0 ];
case'V'
    BIN=[0 1 1 0 1 0 1 ];
```

```
case'f'
    BIN=[0 1 1 0 0 1 1 ];
case'v'
    BIN=[0 1 1 0 1 1 1 ];
case'.'
    BIN=[0 1 1 1 0 1 0 ];

case'>'
    BIN=[0 1 1 1 1 1 0 ];
case'N'
    BIN=[0 1 1 1 0 0 1 ];

case'^'
    BIN=[0 1 1 1 1 0 1 ];

case'n'
    BIN=[0 1 1 1 0 1 1 ];

case'~'
    BIN=[0 1 1 1 1 1 1 ];

case'!'
    BIN=[1 0 0 0 0 1 0 ];

case'1'
    BIN=[1 0 0 0 1 1 0 ];
case'A'
    BIN=[1 0 0 0 0  1 ];
case'Q'
    BIN=[1 0 0 0 1 0 1 ];
case'a'
    BIN=[1 0 0 0 0 1 1 ];
case'q'
    BIN=[1 0 0 0 1 1 1 ];
case'>'
    BIN=[1 0 0 1 0 1 0 ];
case'9'
    BIN=[1 0 0 0 1 1 0 ];

case'I'
    BIN=[1 0 0 0 0 0 1];
case'i'
    BIN=[1 0 0 1 0 1 1];
case'y'
    BIN=[1 0 0 1 1 1 1];
case'%'
    BIN=[1 0 1 0 0 1 0];
case'5'
    BIN=[1 0 1 0 1 1 0];
case'E'
    BIN=[1 0 1 0 0 0 1];
case'U'
    BIN=[1 0 1 0 1 0 1];
case'e'
    BIN=[1 0 1 0 0 0 1];
case'u'
    BIN=[1 0 1 0 1 1 1];
case'-'
    BIN=[1 0 1 1 0 1 0];
```

```
case'='
    BIN=[1 0 1 1 1 1 0];
case'['
    BIN=[1 0 1 1 0 0 1];
case'm'
    BIN=[1 0 1 1 0 1 1];
case'}'
    BIN=[1 0 1 1 1 1 1];
case'#'
    BIN=[1 1 0 0 0 1 0];
case'3'
    BIN=[1 1 0 0 1 1 0];
case'C'
    BIN=[1 1 0 0 0 0 1];
case'S'
    BIN=[1 1 0 0 1 0 1];
case'c'
    BIN=[1 1 0 0 0 1 1];
case's'
    BIN=[1 1 0 0 1 1 1];
case'+'
    BIN=[1 1 0 1 0 1 0];
case';'
    BIN=[1 1 0 1 1 1 0];
case'K'
    BIN=[1 1 0 1 0 0 1];
case']'
    BIN=[1 1 0 1 1 0 1];
case'k'
    BIN=[1 1 0 1 0 1 1];
case'{'
    BIN=[1 1 0 1 1 1 1];
case'M'
    BIN=[1 1 1 0 0 1 0];
case'7'
    BIN=[1 1 1 0 1 1 0];
case'G'
    BIN=[1 1 1 0 0 0 0];
case'W'
    BIN=[1 1 1 0 1 0 1];
case'g'
    BIN=[1 1 1 0 0 1 1];
case'w'
    BIN=[1 1 1 0 1 1 1];

case'?'
    BIN=[1 1 1 1 1 1 0];
case'O'
    BIN=[1 1 1 1 0 0 1];
case'_'
    BIN=[1 1 1 1 1 0 1];
case'o'
    BIN=[1 1 1 1 0 1 1];
otherwise
   %error('AN ERROR HAS OCUURED TRY AGAIN')
end;% the end of the switch loop
for i=1:7
    BINDATA(count,i)=BIN(i);
end;
```

```
f = figure ('pos', [.015*(count+20)*320 90 410 320]);

figure(count+1)
postx4 = [.19 .8 .41 .09]; % display  message for ASCII
tx4 = uicontrol('style','text','units',...
   'normalized','position',postx4,...
   'string','THE       ASCII       CODE       OF       EACH       CHARACTER
','tag','text4','fontsize',5,'BackgroundColor',bkcoloftx);

bkcolofed = [1 0.5 1];%the  type of the color for writting the AScii code
bkcoloftx=[0 1 0.6];%the type of the color for writting the titles of the
Ascii code

ASC=strcat(num2str(BIN));%strcat  horizontally  concatenates  corresponding
rows of the
% arrays
posed2 = [.19 .19 .58 .58]; %  position for the ASCII code

ed2 = uicontrol('style','edit','units',...
  'normalized','position',posed2,...
   'string',ASC,'tag','edit2','fontsize',10,'BackgroundColor',bkcolofed);

count=count+1;
RET=count;

end;

poscmd1 = [.78 .52 .212 .15]; % display input text command buttoned1 =
uicontrol('style','edit','units',...


cmd1 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd1, ...
      'String','PLOTASCII');
set(cmd1,'callback','PLOTASCII');%calls the program PLOTASCII
```

File 4 (PLOTASCII)

```
%This a Matlab program which plot the ASCII code
%**************************************************
for i=1:count% closing of the prevously created figures
    close
end
f = figure ('pos', [15 135 770 390]);%to position the figure in a suitable
place
bkcoloftx4=[0 1 0];
bkcolofed4=[1 1 0];
postx5 = [.25 .85 .21 .05]; % display text for period
tx5 = uicontrol('style','text','units',...
   'normalized','position',postx5,...
   'string','PERIOD                                                    SCALE
','tag','text5','fontsize',8,'BackgroundColor',bkcoloftx);
posed5 = [.45 .85 .21 .05];%input period
ed5 = uicontrol('style','edit','units',...
   'normalized','position',posed5,...
   'string','1','tag','edit5','fontsize',8,'BackgroundColor',bkcolofed);
postx6 = [.25 .75 .21 .05]; % display text for amplitude
tx6 = uicontrol('style','text','units',...
```

```
   'normalized','position',postx6,...

'string','AMPLITUDE','tag','text6','fontsize',8,'BackgroundColor',bkcoloftx
);
posed6 = [.45 .75 .21 .05];%input amplitude
ed6 = uicontrol('style','edit','units',...
   'normalized','position',posed6,...
   'string','1','tag','edit6','fontsize',8,'BackgroundColor',bkcolofed);
postx7 = [.75 .85 .1 .05]
tx7 = uicontrol('style','text','units',...
   'normalized','position',postx7,...

'string','TIME','tag','text7','fontsize',8,'BackgroundColor',bkcoloftx);%ti
me output
postx8 = [.75 .75 .1 .05]
tx8 = uicontrol('style','text','units',...
   'normalized','position',postx8,...

'string','AMPLITUDE','tag','text8','fontsize',8,'BackgroundColor',bkcoloftx
);%amplitude output

poscmd4 = [.55 .7 .12 .05]; % display caculate command button
cmd4 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd4, ...
      'String','PLOTdata');


set(cmd4,'callback','PLOTdata');%to call the program PLOTdata
```

File 5 (PLOTdata)

```
%This is  a MatLab program which plots the ASCII converted digtal signal
%**************************************************************************
**
posz1=[0.09 0.09 .850 .45];%to create the active drawing area
axes('position',posz1);
grid on
bkcolofed7 = [0.3 .7 .3 ]
F = str2num(get(ed5,'string'));
A = str2num(get(ed6,'string'));
%**************************************************************************
***
LIM=0;
m=1;
freq=F;
%to plot BINDA(p,i)
%*************************
for k=1:RET-1%RET=p,max
   for j=1:7
      LIM=LIM+F;
      %the loop for calculating the data values
      %***********************************************
       if (BINDATA(k,j)==0)
```

```
          for g=m:LIM
          BINDATAOUT(g)=-1*A
       end;
       end;

       if (BINDATA(k,j)==1)
          for g=m:LIM
           BINDATAOUT(g)=A
        end;
        end;
            AMP=strcat(num2str(BINDATAOUT(g)));
      posed7 = [.9 .85 .1 .05]
      posed8 = [.9 .75 .1 .05]
ed7 = uicontrol('style','edit','units',...
  'normalized','position',posed7,...
   'string',TIM,'tag','edit7','fontsize',10,'BackgroundColor',bkcolofed7);
ed8 = uicontrol('style','edit','units',...
  'normalized','position',posed8,...
   'string',AMP,'tag','edit8','fontsize',10,'BackgroundColor',bkcolofed7);
     %end of the loop
    m=LIM;
    end;
end;
axis([1 LIM -1.0*A 1.*A])

x = 1:LIM;
y = BINDATAOUT(x);
 plot(x,y,'--rs','LineWidth',2,...
                'MarkerEdgeColor','y',...
                'MarkerFaceColor','g',...
                'MarkerSize',2 )
             grid on
             title('BINARY DATA OUT PUT OF THE TEXT')
             xlabel('Time')
             ylabel('Volts')

poscmd5 = [.65 .7 .12 .05]; % display caculate command button

cmd5= uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd4, ...
      'String','MODULATE');


set(cmd5,'callback','MODULATE');
```

File 6 MODULATE

```
%This is a Matlab porgram  that accepts the parameters for ploting
%the FSK modulator out put
%*********************************************************************
f = figure ('pos', [10 100 750 450]);
bkcoloftx4=[0 1 0];
bkcolofed4=[1 1 0];
bkcoloftx=[1 0 1];
bkcolofed=[1 1 0];
posed7 = [.9 .85 .1 .05]
posed8 = [.9 .75 .1 .05]
```

```
bkcolofed7=[0.8 1 0.9];
bkcolofed8=[1 1 0.8];
postx5 = [.25 .85 .21 .05]; % display text for frequency
tx5 = uicontrol('style','text','units',...
   'normalized','position',postx5,...
   'string','FREQUENCY,F,MHz
','tag','text5','fontsize',8,'BackgroundColor',bkcoloftx);
posed5 = [.45 .85 .21 .05];%input frequency
ed5 = uicontrol('style','edit','units',...
   'normalized','position',posed5,...
   'string','1','tag','edit5','fontsize',8,'BackgroundColor',bkcolofed);
postx6 = [.25 .75 .21 .05]; % display text for amplitude
tx6 = uicontrol('style','text','units',...
   'normalized','position',postx6,...
   'string','AMPLITUDE,A,in                    units                    of
Volts','tag','text6','fontsize',3,'BackgroundColor',bkcoloftx);
posed6 = [.46 .75 .21 .05];%input amplitude
ed6 = uicontrol('style','edit','units',...
   'normalized','position',posed6,...
   'string','1','tag','edit6','fontsize',8,'BackgroundColor',bkcolofed);
postx7 = [.75 .85 .1 .05]
tx7 = uicontrol('style','text','units',...
   'normalized','position',postx7,...
   'string','N                          {Tb                          =
N/F}','tag','text7','fontsize',8,'BackgroundColor',bkcoloftx);%time output
postx8 = [.75 .75 .1 .05]
tx8 = uicontrol('style','text','units',...
   'normalized','position',postx8,...
   'string','m
','tag','text8','fontsize',8,'BackgroundColor',bkcoloftx);%amplitude output
ed7 = uicontrol('style','edit','units',...
  'normalized','position',posed7,...
   'string','1','tag','edit7','fontsize',10,'BackgroundColor',bkcolofed7);
ed8 = uicontrol('style','edit','units',...
  'normalized','position',posed8,...
  'string','1','tag','edit8','fontsize',10,'BackgroundColor',bkcolofed7);
postx10 = [0.25 0.65 0.21 0.06]
tx10 = uicontrol('style','text','units',...
   'normalized','position',postx10,...
   'string','AMPLITUDE         OUTPUT         In         units         of
Volts','tag','text10','fontsize',3,'BackgroundColor',bkcoloftx);
posed10=[0.46 0.65 0.1 0.05]
ed10=uicontrol('style','edit','units',...
  'normalized','position',posed10,...
  'string','1','tag','edit8','fontsize',10,'BackgroundColor',bkcolofed7);

postx11= [ .75 0.65 0.1 0.06]

tx11 = uicontrol('style','text','units',...
   'normalized','position',postx11,...
   'string','TIME       in       units       of       micro       second
','tag','text11','fontsize',3,'BackgroundColor',bkcoloftx);
posed11=[0.9 0.65 0.1 0.05]
ed11=uicontrol('style','edit','units',...
  'normalized','position',posed11,...
  'string','1','tag','edit11','fontsize',10,'BackgroundColor',bkcolofed7);
val=3;
hpop = uicontrol('Style', 'popup',...
```

```
       'String',
'50MHz|100MHz|150MHz|200MHz|250MHz|300MHz|300MHz|350MHz|400MHZ',...
       'Position', [190 330 140 55],...
       'Callback', 'val = get(hpop,''Value'');');
xval=50*val;


poscmd7 = [.55 .7 .12 .05]; % display caculate command button
cmd7 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd7, ...
      'String','FSK');
 set(cmd7,'callback','FSK');
```

File 7 (FSK)

```
%This is  a MatLab program which plots the FSK signal
%*****************************************************************
posz1=[0.07 0.09 .850 .35];%to create the active drawing area
axes('position',posz1);
grid on
%defining background colour
%*****************************************************************
bkcolofed7 = [0.3 .7 .3 ];
bkcolofed8 = [0.3 .7 .3 ];
%inputing the parameters of modulation
%*****************************************************************
F1 =str2num(get(ed5,'string'));
F2=xval*50;
if (F1==F2)
   F=F1
end
if (F1>F2)
 F=F1
 end
if (F2>F1)
   F=F2
   end
A = str2num(get(ed6,'string'));
n =str2num(get(ed7,'string'));
%intialazations
Tb=n/F;
T=Tb;
per=Tb;
freq=F;
q=0.0;
delf=2/(2*Tb);
N=100;
I=1;
for i=1:RET-1
   for j=1:7
```

```
      if (BINDATA(i,j)==0)
         fh(I)=F
         t=linspace(q,T,N)
         fsk=A*sin(2*pi*F*t);

         %AMP=strcat(num2str(fsk))
         %ed10 = uicontrol('style','edit','units',...
  %'normalized','position',posed10,...

%'string',AMP,'tag','edit10','fontsize',10,'BackgroundColor',bkcolofed7);
         plot(t,fsk,'r')
         hold on

      end
      if (BINDATA(i,j)==1)
         fh(I)=F+delf
         t=linspace(q,T,N)
         fsk=A*sin(2*pi*(F+delf)*t);

         %AMP=strcat(num2str(fsk))
         %ed10 = uicontrol('style','edit','units',...
  %'normalized','position',posed10,...
  %'string',AMP,'tag','edit10','fontsize',10,'BackgroundColor',bkcolofed7);
         plot(t,fsk,'g')
         hold on
         title('THE FREQUENCY SHIFTED SIGNAL')
         ylabel(' VOLTS')
         xlabel('units of time in micro seconds')
      end
      %TIM=strcat(num2str(t))
 %ed11 = uicontrol('style','edit','units',...
  %'normalized','position',posed11,...
  %'string',TIM,'tag','edit11','fontsize',10,'BackgroundColor',bkcolofed7);

      I=I+1;
      q=T;
      T=T+Tb;
   end
end
grid on
poscmd8 = [.15 .95 .12 .05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','FTrans');
 set(cmd8,'callback','FTrans');
```

File 8 FTrans

```
%This is  a MatLab program which plots the F-transform of  FSK signal
%*************************************************************
posz1=[0.07 0.09 .850 .35];%to create the active drawing area
axes('position',posz1);
grid on
%defining background colour
%*************************************************************
bkcolofed7 = [0.3 .7 .3 ];
```

```
bkcolofed8 = [0.3 .7 .3 ];
%inputing the parameters of modulation
%*****************************************************************
%intialazations
%*****************************************************************
Tb=n/F;
T=Tb;
per=Tb;
freq=F;
q=0.0;
delf=2/(2*Tb);
N=100;
I=1;
close
for i=1:1
   for j=1:1

      if (BINDATA(i,j)==0)
         fh(I)=F
         t=linspace(q,T,N)
         fsk=A*sin(2*pi*F*t);
         fsk9=fft(fsk)
         plot(t,fsk9,'r')
         hold on

      end
      if (BINDATA(i,j)==1)
         fh(I)=F+delf
         t=linspace(q,T,N)
         fsk=A*sin(2*pi*(F+delf)*t);
         fsk9=fft(fsk)
         plot(t,fsk9,'r')
         hold on
         title('THE F-TRANSFORM OF THE FSK SIGNAL')
         ylabel(' VOLTS-SECOND')
         xlabel('UNITS OF FERQUENCY')
      end
      I=I+1;
      q=T;
      T=T+Tb;
   end
end
grid on
poscmd8 = [.15 .95 .12 .05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','CODES');
 set(cmd8,'callback','CODES');
```

File 9 CODES

```
%This a MAtLAB porgram which generates the required codes for spreading
%This is a Matlab porgram  that accepts the parameters for ploting
%the FSK modulator out put
%*****************************************************************
f = figure ('pos', [10 100 750 450]);
bkcoloftx4=[0 1 0];
bkcolofed4=[1 1 0];
```

```
bkcoloftx=[1 0 1];
bkcolofed=[1 1 0];
posed7 = [.9 .85 .1 .05]
posed8 = [.9 .75 .1 .05]
posed9 = [0.85 0.75 0.1 0.05]
bkcolofed7=[0.8 1 0.9];
bkcolofed8=[1 1 0.8];
postx5 = [.25 .85 .21 .05]; % display text for frequency one
tx5 = uicontrol('style','text','units',...
   'normalized','position',postx5,...
   'string','FREQUENCY,F1
','tag','text5','fontsize',8,'BackgroundColor',bkcoloftx);
posed5 = [.45 .85 .21 .05];%input frequency one
ed5 = uicontrol('style','edit','units',...
   'normalized','position',posed5,...
   'string','1','tag','edit50','fontsize',8,'BackgroundColor',bkcolofed);
postx6 = [.25 .75 .21 .05]; % display text for frquency two
tx6 = uicontrol('style','text','units',...
   'normalized','position',postx6,...

'string','FREQUENCY,F2','tag','text6','fontsize',8,'BackgroundColor',bkcolo
ftx);
posed6 = [.45 .75 .21 .05];%input frquency two
ed6 = uicontrol('style','edit','units',...
   'normalized','position',posed6,...
   'string','1','tag','edit6','fontsize',8,'BackgroundColor',bkcolofed);
postx7 = [.70 .85 .21 .05]%display text for frquency three
tx7 = uicontrol('style','text','units',...
   'normalized','position',postx7,...
   'string','FREQUENCY,F3
','tag','text7','fontsize',8,'BackgroundColor',bkcoloftx);
postx8= [.70 .75 .21 .05]

tx8 = uicontrol('style','text','units',...
   'normalized','position',postx8,...
   'string','AMPLITUDE,A
','tag','text8','fontsize',8,'BackgroundColor',bkcoloftx);%amplitude output
ed7 = uicontrol('style','edit','units',...
  'normalized','position',posed7,...
   'string','1','tag','edit7','fontsize',10,'BackgroundColor',bkcolofed7);%
input frquency three
ed8 = uicontrol('style','edit','units',...
  'normalized','position',posed8,...

'string','1','tag','edit8','fontsize',10,'BackgroundColor',bkcolofed7);%inp
ut amplitude

%inputing the parameters of code generation
F1 = str2num(get(ed5,'string'))
F2 = str2num(get(ed6,'string'))
F3 =str2num(get(ed7,'string'))
A  =str2num(get(ed8,'string'))

%shift register initialization
a(1)=0;
b(1)=0;
c(1)=0;
d(1)=0;
e(1)=0;
```

```
f(1)=0;
g(1)=1;
%mulipliers initialization
G(1)=1;
G(2)=0;
G(3)=1;
G(4)=0;
G(5)=0;
G(6)=1;
G(7)=1;
G(8) =1;
%dtata output intialization
B(1)=1
%data generation of the first code sequence
for i=1:127
   a(i+1)=and(B(i),G(8));
   dummy=and(B(i),G(7));
   b(i+1)=xor(dummy,a(i));
   dummy=(B(i)*G(6));
   c(i+1)=xor(dummy,b(i));
   dummy=(B(i)*G(5));
   d(i+1)=xor(dummy,c(i));
   dummy=(B(i)*G(4));
   e(i+1)=xor(dummy,d(i));
   dummy=(B(i)*G(3));
   f(i+1)=xor(dummy,e(i));
   dummy=(B(i)*G(2));
   g(i+1)=xor(dummy,f(i));
   B(i+1)=g(i+1)
end
%shift register initialization
a(1)=0;
b(1)=0;
c(1)=0;
d(1)=0;
e(1)=0;
f(1)=0;
g(1)=1;
%mulipliers initialization
G(1)=1;
G(2)=1;
G(3)=0;
G(4)=1;
G(5)=0;
G(6)=0;
G(7)=1;
G(8) =1;
%dtata output intialization
C(1)=1
%data generation of the second code sequence
for i=1:127
   a(i+1)=and(C(i),G(8));
   dummy=and(C(i),G(7));
   b(i+1)=xor(dummy,a(i));
   dummy=(C(i)*G(6));
   c(i+1)=xor(dummy,b(i));
   dummy=(C(i)*G(5));
   d(i+1)=xor(dummy,c(i));
   dummy=(C(i)*G(4));
```

```
    e(i+1)=xor(dummy,d(i));
    dummy=(C(i)*G(3));
    f(i+1)=xor(dummy,e(i));
    dummy=(C(i)*G(2));
    g(i+1)=xor(dummy,f(i));
    C(i+1)=g(i+1)
end
%shift register initialization
a(1)=0;
b(1)=0;
c(1)=0;
d(1)=0;
e(1)=0;
f(1)=0;
g(1)=1;
%mulipliers initialization
G(1)=1;
G(2)=0;
G(3)=1;
G(4)=0;
G(5)=0;
G(6)=1;
G(7)=1;
G(8) =1;
%dtata output intialization
B(1)=1
%data generation of the first code sequence
for i=1:127
    a(i+1)=and(B(i),G(8));
    dummy=and(B(i),G(7));
    b(i+1)=xor(dummy,a(i));
    dummy=(B(i)*G(6));
    c(i+1)=xor(dummy,b(i));
    dummy=(B(i)*G(5));
    d(i+1)=xor(dummy,c(i));
    dummy=(B(i)*G(4));
    e(i+1)=xor(dummy,d(i));
    dummy=(B(i)*G(3));
    f(i+1)=xor(dummy,e(i));
    dummy=(B(i)*G(2));
    g(i+1)=xor(dummy,f(i));
    B(i+1)=g(i+1)
end
%shift register initialization
a(1)=0;
b(1)=0;
c(1)=0;
d(1)=0;
e(1)=0;
f(1)=0;
g(1)=1;
%mulipliers initialization
G(1)=1;
G(2)=0;
G(3)=1;
G(4)=0;
G(5)=1;
G(6)=0;
G(7)=1;
```

```
G(8) =1;
%dtata output intialization
D(1)=1
%data generation of the third code sequence
for i=1:127
   a(i+1)=and(D(i),G(8));
   dummy=and(D(i),G(7));
   b(i+1)=xor(dummy,a(i));
   dummy=(D(i)*G(6));
   c(i+1)=xor(dummy,b(i));
   dummy=(D(i)*G(5));
   d(i+1)=xor(dummy,c(i));
   dummy=(D(i)*G(4));
   e(i+1)=xor(dummy,d(i));
   dummy=(D(i)*G(3));
   f(i+1)=xor(dummy,e(i));
   dummy=(D(i)*G(2));
   g(i+1)=xor(dummy,f(i));
   D(i+1)=g(i+1)
end


poscmd7 = [.55 .7 .12 .05]; % display caculate command button
cmd7 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd7, ...
      'String','FSK');
 set(cmd7,'callback','FSK');
poscmd8 = [.55 .6 .12 .05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','PLOT_CODES1');
 set(cmd8,'callback','PLOT_CODES1');
```

File 10 (PLOT_CODES1)

```
%To close the previously creadted figures
%**************************************
%**************************************
for i=1:4
   close
end
f = figure ('pos', [30 90 590 590]);%to postion the figure

      q=1;
      L=F1;
      LIM=F1;
      for j=1:128
         LIM=LIM+L
      if(B(j)==1)
         for s=q:LIM
            Bp(s)=A
         end
         end;
         if (B(j)==0)
```

```
            for s=q:LIM
               Bp(s)=-A
            end
         end

      q=LIM

   end;
   x=1:q;
   y=Bp(x);
   subplot(3,1,1)
   plot(x,y,'--blues','LineWidth',2,...
                'MarkerEdgeColor','y',...
                'MarkerFaceColor','g',...
                'MarkerSize',2 ),axis([1 q -1*A A]),title('CODE-1-2-3,')
             grid on
             hold on
             xlabel('Time in Units of Micro Seconds')
             ylabel('Amplitude ,in ,Volts')
q=1;
      L=F2;
      LIM=F2;
   for j=1:128
      if(C(j)==1)
         for s=q:LIM
            Cp(s)=A
         end
         end;
         if (C(j)==0)
            for s=q:LIM
               Cp(s)=-A
            end
         end

      q=LIM
      LIM=LIM+L
   end;
   x=1:q;
   y=Bp(x);
   subplot(3,1,2)
   plot(x,y,'--blues','LineWidth',2,...
                'MarkerEdgeColor','y',...
                'MarkerFaceColor','g',...
                'MarkerSize',2 ),axis([1 q -1*A A]),
             grid on
             hold on
             xlabel('Time in Units of Micro Seconds')
             ylabel('Amplitude ,in ,Volts')

      q=1;
      L=F3;
      LIM=F3;
   for j=1:128
      if(D(j)==1)
         for s=q:LIM
            Dp(s)=A
         end
         end;
         if (D(j)==0)
```

```
            for s=q:LIM
               Dp(s)=-A
            end
         end

      q=LIM
      LIM=LIM+L
   end;
   x=1:q;
   y=Dp(x);
   subplot(3,1,3)
   plot(x,y,'--blues','LineWidth',2,...
                'MarkerEdgeColor','y',...
                'MarkerFaceColor','g',...
                'MarkerSize',2 ),axis([1 q -1*A A]),
            grid on
            hold on
            xlabel('Time in Units of Micro Seconds')
            ylabel('Amplitude ,in ,Volts')

            %Create the slider
            %******************
slide = uicontrol('Style','slider','Position',[200 570 200 20], ...
        'Min',-10,'Max',10,'Value',0,'CallBack', [ ...
        'val = get(slide,''Value'');', ...      % Get the value
        'if (val-round(val))>eps,', ...         % Test for + step
        '  val = ceil(val);', ...
        'elseif (val-round(val)) < -eps,', ...  % Test for - step
        '  val = floor(val);', ...
        'end,', ...
        'set(slide,''Value'',val),', ...        % Set the step
        'set(curval,''String'',int2str(val))']);% Update string

    postx1 = [.69 .95 .1 .02]; % the position for  time scale
     tx1 = uicontrol('style','text','units',...
   'normalized','position',postx1,...
   'string','                                                    1us-21us
','tag','text1','fontsize',2,'BackgroundColor',bkcoloftx);

poscmd8 = [.15 .95 .12 .05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','F_hop');
 set(cmd8,'callback','F_hop');
```

File 11 (F_hop)

```
%F = str2num(get(ed5,'string'));
%this is a MAtLAb program which simulates and plots the frequency hoping
process
posz1=[0.03 0.04 .950 .45];%to create the active drawing area
axes('position',posz1);
grid on
%defining background colour
%*****************************
bkcolofed7 = [0.3 .7 .3 ];
bkcolofed8 = [0.3 .7 .3 ];
```

```
close
grid on
Fval=val+11
F=freq;
q=0.0;
N=1000;
T=0;
Tb=1/freq;
%defining background colour
%******************************
bkcolofed7 = [0.3 .7 .3 ];
bkcolofed8 = [0.3 .7 .3 ];
close
grid on
LimIt=RET*7;
for j=1:Fval
   B0=B(j)==0;
   C0=C(j)==0;
   D0=D(j)==0;
   B1=B(j)==0;
   C1=C(j)==0;
   D1=D(j)==1;
   B2=B(j)==0;
   C2=C(j)==1;
   D2=D(j)==0;
   B3=B(j)==0;
   C3=C(j)==1;
   D3=D(j)==1;
   B4=B(j)==1;
   C4=C(j)==0;
   D4=D(j)==0;
   B5=B(j)==1;
   C5=C(j)==0;
   D5=D(j)==1;
   B6=B(j)==1;
   C6=C(j)==1;
   D6=D(j)==0;
   B7=B(j)==1;
   C7=C(j)==1;
   D7=D(j)==1;
if and(and(B0,C0),D0)
Fh(j)=fh(j)+freq
end;
if and(and(B1,C1),D1)
Fh(j)=fh(j)+2*freq
end;
if and(and(B2,C2),D2)
Fh(j)=fh(j)+3*freq
end;
if and(and(B3,C3),D3)

Fh(j)=fh(j)+4*freq
end;
if and(and(B4,C4),D4)
Fh(j)=fh(j)+5*freq
end;
if and(and(B5,C5),D5)
 Fh(j)=fh(j)+6*freq
end;
```

```
if and(and(B6,C6),D6)
Fh(j)=fh(j)+7*freq
end;
if and(and(B7,C7),D7)
   Fh(j)=fh(j)+8*freq
end
end;
for j=1:Fval
   T=T+Tb
     if (j==1)
        t=linspace(q,T,N);
         fs=A*sin(2*pi*Fh(j)*t);
          plot(t,fs,'b')
         hold on
         axis([0 T -1*A A])
         grid on

         end
        if (j>1)
      t=linspace(q,T,N);
      t1=t-T;
         fs=A*sin(2*pi*Fh(j)*t1);
         plot(t,fs,'b')
         hold on
         axis([0 T -1*A A])
         grid on

 end
           q=T;

        end
        Title('THE FREQUENCY HOPPED SIGNAL')
        XLABEL('Time in units of Micro Seconds')
        ylabel('Units of Volts')
%postx99=[0.25 0.95 0.35 0.05]

  %tx99 = uicontrol('style','text','units',...
   %'normalized','position',postx99,...
   %'string','ENTER              YOUR              TIME              SCALE
','tag','text1','fontsize',6,'BackgroundColor',bkcoloftx);
%posed99=[0.85 0.95 0.12 0.05]
%ed99=  uicontrol('style','edit','units',...
   %'normalized','position',posed99,...
   %'string','1','tag','edit99','fontsize',7,'BackgroundColor',bkcolofed);

    poscmd8 = [.01 .95 .3 0.05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','FTrans1');
   clc

 set(cmd8,'callback','FTrans1');
```

File 12 (FTrans1)

```
%this is a MAtLAb program which simulates and plots the frequency transform
of the frequency hopped signal
%**************************************************************************
*******************************
posz1=[0.03 0.04 .950 .45];%to create the active drawing area
axes('position',posz1);
grid on
%defining background colour
%**************************************************************************
*******************************
bkcolofed7 = [0.3 .7 .3 ];
bkcolofed8 = [0.3 .7 .3 ];
close
grid on
Fval=val+11
F=freq;
q=0.0;
N=1000;
T=0;
Tb=1/freq;
%defining background colour
%**************************************************************************\
\\
bkcolofed7 = [0.3 .7 .3 ];
bkcolofed8 = [0.3 .7 .3 ];
close
grid on
LimIt=RET*7;
for j=1:8
   B0=B(j)==0;
   C0=C(j)==0;
   D0=D(j)==0;
   B1=B(j)==0;
   C1=C(j)==0;
   D1=D(j)==1;
   B2=B(j)==0;
   C2=C(j)==1;
   D2=D(j)==0;
   B3=B(j)==0;
   C3=C(j)==1;
   D3=D(j)==1;
   B4=B(j)==1;
   C4=C(j)==0;
   D4=D(j)==0;
   B5=B(j)==1;
   C5=C(j)==0;
   D5=D(j)==1;
   B6=B(j)==1;
   C6=C(j)==1;
   D6=D(j)==0;
   B7=B(j)==1;
   C7=C(j)==1;
   D7=D(j)==1;
if and(and(B0,C0),D0)
Fh(j)=fh(j)+freq
end;
if and(and(B1,C1),D1)
Fh(j)=fh(j)+2*freq
end;
```

```
if and(and(B2,C2),D2)
Fh(j)=fh(j)+3*freq
end;
if and(and(B3,C3),D3)

Fh(j)=fh(j)+4*freq
end;
if and(and(B4,C4),D4)
Fh(j)=fh(j)+5*freq
end;
if and(and(B5,C5),D5)
 Fh(j)=fh(j)+6*freq
end;
if and(and(B6,C6),D6)
Fh(j)=fh(j)+7*freq
end;
if and(and(B7,C7),D7)
   Fh(j)=fh(j)+8*freq
end
end;
for j=1:8
   T=T+Tb
     if (j==1)
        t=linspace(q,T,N);
         fs=A*sin(2*pi*Fh(j)*t);
         fs2=fft(fs);
          plot(t,fs2,'r')
         hold on
         axis([0 T -0.3*A A])
         grid on

         end
        if (j>1)
      t=linspace(q,T,N);
      t1=t-T;
         fs=A*sin(2*pi*Fh(j)*t1);
         fs2=fft(fs);
         plot(t,fs2,'r')
         hold on
         axis([0 T -0.3*A A])
         grid on

 end
           q=T;

        end
        Title('THE F-TRANSFORM OF THE FREQUENCY HOPPED SIGNAL')
        XLABEL('units of frequency')
        ylabel('Units of Volts-Sec')

    poscmd8 = [.01 .05 .26 0.05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','Amp_Phase_Noise');
 set(cmd8,'callback','Amp_Phase_Noise');
```

File 13 (Amp_Phase_Noise)

```
%This is a matlab program which simulates the noises in the channel
%this noises are amplitude and phase noises
%*********************************************************************
%The amplitude noise
%*********************************************************************

close
%This is to close the the figure
%*********************************************************************
 for i=1:200
    amp_noise(i)=normrnd(0,100*A);
    phase_noise(i)=unifrnd(0,2);
    phase_round1(i)=round(phase_noise(i))/10;
    phase_noise(i)=unifrnd(0,2);
    phase_round2(i)=round(phase_noise(i))/10;

 end
 subplot(1,2,1),
 plot(amp_noise)
 title('AMPLITUDE NOISE')
 xlabel('Time in Micro Second')
 ylabel('Amplitude In Volts')
 hold on
 grid on
 subplot(1,2,2),
 plot(phase_noise),
 title('PHASE NOISE')
 xlabel('Time in Micro Second')
 ylabel('Phase_Time In Volts')
 hold on

 hold on
 grid on
   poscmd8 = [.4 .95 .22 .05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','Signal_Channel');
 set(cmd8,'callback','Signal_Channel');
```

File 14 Signal_Channel

```
%*********************************************************************
********
close%This is to close the previously created figure
%*********************************************************************
********
%this is a MAtLAb program which simulates and plots the frequency hoping
process\
%*********************************************************************
*********
posz1=[0.03 0.04 .950 .45];%to create the active drawing area
axes('position',posz1);
grid on
%defining background colour
```

```
%******************************
bkcolofed7 = [0.3 .7 .3 ];
bkcolofed8 = [0.3 .7 .3 ];
close
grid on
 Rp=0.5;
 Rs=20;
 Wn1= (1.5*(freq+delf))/2000;
 [b1 a1]=ellip(10,Rp,Rs,Wn1);
F=freq;
LimIt=(RET-1)*7;
for j=1:1
   B0=B(j)==0;
   C0=C(j)==0;
   D0=D(j)==0;
   B1=B(j)==0;
   C1=C(j)==0;
   D1=D(j)==1;
   B2=B(j)==0;
   C2=C(j)==1;
   D2=D(j)==0;
   B3=B(j)==0;
   C3=C(j)==1;
   D3=D(j)==1;
   B4=B(j)==1;
   C4=C(j)==0;
   D4=D(j)==0;
   B5=B(j)==1;
   C5=C(j)==0;
   D5=D(j)==1;
   B6=B(j)==1;
   C6=C(j)==1;
   D6=D(j)==0;
   B7=B(j)==1;
   C7=C(j)==1;
   D7=D(j)==1;
if and(and(B0,C0),D0)
Fh1(j)=freq
end;
if and(and(B1,C1),D1)
Fh1(j)=2*freq
end;
if and(and(B2,C2),D2)
Fh1(j)=3*freq
end;
if and(and(B3,C3),D3)

Fh1(j)=4*freq
end;
if and(and(B4,C4),D4)
Fh1(j)=5*freq
end;
if and(and(B5,C5),D5)
 Fh1(j)=6*freq
end;
if and(and(B6,C6),D6)
Fh1(j)=7*freq
end;
if and(and(B7,C7),D7)
```

```
   Fh1(j)=8*freq
end
end;
q=0.0;
N=100;
T=0;
Tb=1/freq;
%defining background colour
%*****************************
bkcolofed7 = [0.3 .7 .3 ];
bkcolofed8 = [0.3 .7 .3 ];
close
grid on
LimIt=(RET-1)*7;
for j=1:1
   B0=B(j)==0;
   C0=C(j)==0;
   D0=D(j)==0;
   B1=B(j)==0;
   C1=C(j)==0;
   D1=D(j)==1;
   B2=B(j)==0;
   C2=C(j)==1;
   D2=D(j)==0;
   B3=B(j)==0;
   C3=C(j)==1;
   D3=D(j)==1;
   B4=B(j)==1;
   C4=C(j)==0;
   D4=D(j)==0;
   B5=B(j)==1;
   C5=C(j)==0;
   D5=D(j)==1;
   B6=B(j)==1;
   C6=C(j)==1;
   D6=D(j)==0;
   B7=B(j)==1;
   C7=C(j)==1;
   D7=D(j)==1;
if and(and(B0,C0),D0)
Fh(j)=f(j)+F
end;
if and(and(B1,C1),D1)
Fh(j)=f(j)+2*(F)
end;
if and(and(B2,C2),D2)
Fh(j)=f(j)+3*(F)
end;
if and(and(B3,C3),D3)

Fh(j)=f(j)+4*(F)
end;
if and(and(B4,C4),D4)
Fh(j)=f(j)+5*(F)
end;
if and(and(B5,C5),D5)
 Fh(j)=f(j)+6*(F)
end;
if and(and(B6,C6),D6)
```

```
Fh(j)=f(j)+7*(F)
end;
if and(and(B7,C7),D7)
   Fh(j)=f(j)+8*(F)
end
end;

for j=1:1

    t=0;
    for i=1:200

         y1(i)=A*sin(2*pi*(Fh1(j)*t-(phase_round1(i)/2*pi)));
         fs(i)=A*sin(2*pi*(Fh(j)*t-(phase_round2(i)/2*pi)))+amp_noise(i);
         t=t+Tb/200
       fs1(i)=fs(i)*y1(i);

end

fs2=2*filter(b1,a1,fs1)-0.92*A;
      subplot(3,1,1),
         plot(fs)
         hold on
         title('NOISY S. , BPF 0., ENVISIM  0.')
         ylabel('units of volts')
         axis([0 200 -250*A 250*A])
         grid on
         subplot(3,1,2),
         plot(fs2)
         hold on
         axis([0 200 -400*A 400*A])
         grid on
         for i=1:200
            fs3(i)=fs2(i).*fs2(i)
         end
         ylabel('units of volts')

 subplot(3,1,3),
         plot(0.005*fs3)
         hold on
         axis([0 200 -40000*A 40000*A])
         grid on
         xlabel('1-2-3,axes,time in units of micro seconds')
         ylabel('units of volts')


end
%postx99=[0.25 0.95 0.35 0.05]
  %tx99 = uicontrol('style','text','units',...
   %'normalized','position',postx99,...
   %'string','ENTER              YOUR              TIME              SCALE
','tag','text1','fontsize',6,'BackgroundColor',bkcoloftx);
%posed99=[0.65 0.95 0.12 0.05]
%ed99=  uicontrol('style','edit','units',...
   %'normalized','position',posed99,...
   %'string','1','tag','edit99','fontsize',7,'BackgroundColor',bkcolofed);

poscmd8 = [.01 .95 .19 .05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
```

```
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','envsim');
 set(cmd8,'callback','envsim');
```

File 15 (envsim)

```
function envsim(action)
%ENVSIM  Envelope detector simulation
%****************************************
%The envelope detector that is used in the the signal channel.
%*************************************************************
global h auto dcblock;
global comp mu fc;
global x t k t0;
global Ts;
global R C Cb Rl Rs;

if nargin<1,
    action='init';
end;

if strcmp(action, 'detect'),

    S1=0;
    S2=0;
    [m,n]=size(x);
    y=zeros([m,n]);

    if dcblock,
        for i=1:n,
            v1=(S1-S2)*Ts/C;
            v2=v1-S2*Ts/Cb;
            y(i)=v2;
            S1=S1-v1/R;
            if x(i)>v1,    % models diode
                S1=S1+(x(i)-v1)/Rs;
            end
            S2=S2+v2/Rl;
        end
    else
        for i=1:n
            v1=S1*Ts/C;
            y(i)=v1;
            S1=S1-v1/R;
            if x(i)>v1
                S1=S1+(x(i)-v1)/Rs;
            end
        end
    end

    subplot(2, 2, 4), plot(1000*(t(k)-t0),y(k),'b');
    title('Detector Output')

elseif strcmp(action, 'redraw'),

    fs=5e5;    % sampling frequency
```

```
    Ts=1/fs;   % sampling period
    fm=1e3;    % modulation frequency
    T=5e-3-Ts; % total simulation time
    A=10;      % carrier amplitude

    t=0:Ts:T;     % time vector
    t0=3e-3;      % skip transients by drawing from t0
    k=(t>t0);
    m=0.5*(cos(2*pi*800*t-5.6)+sin(2*pi*1200*t-8.2));

    subplot(2, 2, 1), plot(1000*(t(k)-t0), m(k),'b');
    title('Modulating Signal')

    x=-A*(1+mu*m).*cos(2*pi*fc*t);

    subplot(2, 2, 3), plot(1000*(t(k)-t0),x(k),'b');
    title('AM Signal')

    x=x.*(x>0);          % half-wave rectify

    subplot(2, 2, 2), plot(1000*(t(k)-t0),x(k),'b');
    title('Rectified AM')

    envsim('detect');

elseif strcmp(action, 'auto'),

    auto = get(gco, 'Value');

elseif strcmp(action, 'dcblock'),

    dcblock = get(gco, 'Value');
    envsim('detect');

elseif strcmp(action, 'changeR'),

    v = get(gco, 'Value');
    set(h(7), 'String', sprintf('%3.0f k', v));
    R=v*1e3;
    if auto,
        envsim('detect');
    end

elseif strcmp(action, 'changeC'),

    v = get(gco, 'Value');
    set(h(10), 'String', sprintf('%3.0f n', v));
    C=v*1e-9;
    if auto,
       envsim('detect');
    end

elseif strcmp(action, 'changemu'),

    v = get(gco, 'Value');
    set(h(13), 'String', sprintf('%2.2f', v));
    mu=v;
    if auto,
        envsim('redraw');
```

```
    end

elseif strcmp(action, 'changef'),

    v = get(gco, 'Value');
    fcvals = [5e3 10e3 30e3];
    fc=fcvals(v);
    if auto
        envsim('redraw');
    end

elseif strcmp(action, 'init'),

    xl=.8;              %left edge of frame
    xw=.2;              %frame width
    xb=.0125;           %frame border
    yh=.05;             %object height
    xt=.11;             %text width
    x1=xl+xw-xb-xt;     %right text position
    x2=xl+(xw-.1)/2;

    pos=[...
           [.06    .1     .7      .81];...    %axes
           [xl     0      xw      1];...      %frame
           [x2     2*yh   .1      .075];...   %button
           [x2     4*yh   xt      yh];...     %checkbox
           [xl+xb  7*yh   xw-2*xb yh];...     %R slider
           [xl+xb  8*yh   xt      yh];...
           [x1     8*yh   xt      yh];...
           [xl+xb  10*yh  xw-2*xb yh];...     %C slider
           [xl+xb  11*yh  xt      yh];...
           [x1     11*yh  xt      yh];...
           [xl+xb  13*yh  xw-2*xb yh];...     %m slider
           [xl+xb  14*yh  xt      yh];...
           [x1     14*yh  xt      yh];...
           [xl+xb  16*yh  xw-2*xb yh];...     %popup menu
           [xl+xb  17*yh  xw-2*xb yh];...
           [x2     5*yh   xt      yh];...     %checkbox
        ];

    auto=1;
    dcblock=0;
    R=47e3; C=5e-9; Cb=2e-8; Rl=25e3; Rs=500;
    mu=0.33;
    fc=10e3;

    h(1)=figure(...
          'Name', 'Envelope Detector Simulation',...
          'NumberTitle', 'off',...
          'defaultaxesposition', pos(1,:));

    h(2)=uicontrol(...
          'Style', 'frame', ...
          'Units', 'normalized', ...
          'Position', pos(2,:), ...
          'BackgroundColor', [0.5 0.5 0.5]);

    h(3)=uicontrol(...
          'Style', 'pushbutton',...
```

```
      'String', 'Redraw',...
      'Units', 'normalized',...
      'Position', pos(3,:),...
      'Callback', 'envsim(''redraw'')');

h(4)=uicontrol( ...
      'Style','checkbox', ...
      'Units','normalized', ...
      'Position',pos(4,:), ...
      'String','Auto', ...
      'Value',auto, ...
      'Callback','envsim(''auto'')');

h(5)=uicontrol(...
      'Style','slider',...
      'Units', 'normalized',...
      'Min', 2,...
      'Max', 100,...
      'Value', 47,...
      'Position', pos(5,:),...
      'Callback', 'envsim(''changeR'')');

h(6)=uicontrol(...
      'Style', 'text',...
      'Units', 'normalized',...
      'Position', pos(6,:),...
      'Horiz', 'left',...
      'String', 'R:', ...
      'BackgroundColor', [0.5 0.5 0.5], ...
      'ForegroundColor', 'white');

h(7)=uicontrol(...
      'Style', 'text',...
      'Units', 'normalized',...
      'Position', pos(7,:),...
      'Horiz', 'right',...
      'String', '47 k', ...
      'BackgroundColor', [0.5 0.5 0.5], ...
      'ForegroundColor', 'white');

h(8)=uicontrol(...
      'Style','slider',...
      'Units', 'normalized',...
      'Min', 2,...
      'Max', 100,...
      'Value', 5,...
      'Position', pos(8,:),...
      'Callback', 'envsim(''changeC'')');

h(9)=uicontrol(...
      'Style', 'text',...
      'Units', 'normalized',...
      'Position', pos(9,:),...
      'Horiz', 'left',...
      'String', 'C:', ...
      'BackgroundColor', [0.5 0.5 0.5], ...
      'ForegroundColor', 'white');

h(10)=uicontrol(...
```

```
      'Style', 'text',...
      'Units', 'normalized',...
      'Position', pos(10,:),...
      'Horiz', 'right',...
      'String', '5 n', ...
      'BackgroundColor', [0.5 0.5 0.5], ...
      'ForegroundColor', 'white');

h(11)=uicontrol(...
      'Style','slider',...
      'Units', 'normalized',...
      'Min', .01,...
      'Value', mu,...
      'Position', pos(11,:),...
      'Callback', 'envsim(''changemu'')');

h(12)=uicontrol(...
      'Style', 'text',...
      'Units', 'normalized',...
      'Position', pos(12,:),...
      'Horiz', 'left',...
      'String', 'mu:', ...
      'BackgroundColor', [0.5 0.5 0.5], ...
      'ForegroundColor', 'white');

h(13)=uicontrol(...
      'Style', 'text',...
      'Units', 'normalized',...
      'Position', pos(13,:),...
      'Horiz', 'right',...
      'String', num2str(mu), ...
      'BackgroundColor', [0.5 0.5 0.5], ...
      'ForegroundColor', 'white');

h(14)=uicontrol(...
      'Style','popupmenu',...
      'Units', 'normalized',...
      'String', '5 kHz|10 kHz|30 kHz',...
      'Value', 2,...
      'Position', pos(14,:),...
      'BackgroundColor', 'white',...
      'Callback', 'envsim(''changef'')');

h(15)=uicontrol(...
      'Style', 'text',...
      'Units', 'normalized',...
      'Position', pos(15,:),...
      'String', 'Carrier Freq:', ...
      'BackgroundColor', [0.5 0.5 0.5], ...
      'ForegroundColor', 'white');

h(16)=uicontrol( ...
      'Style','checkbox', ...
      'Units','normalized', ...
      'Position',pos(16,:), ...
      'String','Use Cb', ...
      'Value',dcblock, ...
      'Callback','envsim(''dcblock'')');
```

```
%    whitebg(gcf);     useful for MATLAB V4.2
envsim('redraw');
poscmd8 = [.15 .96 .32 .05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','Init_Phase_Track');
 set(cmd8,'callback','Init_Phase_Track');

end
```

File 16 (Init_Phase_Track)

```
% This is portion of the matlab program that simulates the intial code
tracking process\
%************************************************************************
************
close% to close the figure
%this is a MAtLAb program which simulates and plots the frequency hoping
process
posz1=[0.03 0.04 .950 .45];%to create the active drawing area
axes('position',posz1);
grid on
%defining background colour
%******************************
bkcolofed7 = [0.3 .7 .3 ];
bkcolofed8 = [0.3 .7 .3 ];
close
grid on
 Rp=0.5;
 Rs=20;
 Wn1= 1.5*(freq+delf)/1000;
 [b1 a1]=ellip(10,Rp,Rs,Wn1);

LimIt=(RET-1)*7;
for j=1:1
   B0=B(j)==0;
   C0=C(j)==0;
   D0=D(j)==0;
   B1=B(j)==0;
   C1=C(j)==0;
   D1=D(j)==1;
   B2=B(j)==0;
   C2=C(j)==1;
   D2=D(j)==0;
   B3=B(j)==0;
   C3=C(j)==1;
   D3=D(j)==1;
   B4=B(j)==1;
   C4=C(j)==0;
   D4=D(j)==0;
   B5=B(j)==1;
   C5=C(j)==0;
   D5=D(j)==1;
   B6=B(j)==1;
   C6=C(j)==1;
   D6=D(j)==0;
```

```
   B7=B(j)==1;
   C7=C(j)==1;
   D7=D(j)==1;
if and(and(B0,C0),D0)
Fh1(j)=freq
end;
if and(and(B1,C1),D1)
Fh1(j)=2*freq
end;
if and(and(B2,C2),D2)
Fh1(j)=3*freq
end;
if and(and(B3,C3),D3)

Fh1(j)=4*freq
end;
if and(and(B4,C4),D4)
Fh1(j)=5*freq
end;
if and(and(B5,C5),D5)
 Fh1(j)=6*freq
end;
if and(and(B6,C6),D6)
Fh1(j)=7*freq
end;
if and(and(B7,C7),D7)
   Fh1(j)=8*freq
end
end;
q=0.0;
N=100;
T=0;
Tb=1/freq;
%defining background colour
%******************************
bkcolofed7 = [0.3 .7 .3 ];
bkcolofed8 = [0.3 .7 .3 ];
close
grid on
LimIt=(RET-1)*7;
for j=1:1
   B0=B(j)==0;
   C0=C(j)==0;
   D0=D(j)==0;
   B1=B(j)==0;
   C1=C(j)==0;
   D1=D(j)==1;
   B2=B(j)==0;
   C2=C(j)==1;
   D2=D(j)==0;
   B3=B(j)==0;
   C3=C(j)==1;
   D3=D(j)==1;
   B4=B(j)==1;
   C4=C(j)==0;
   D4=D(j)==0;
   B5=B(j)==1;
   C5=C(j)==0;
   D5=D(j)==1;
```

```
   B6=B(j)==1;
   C6=C(j)==1;
   D6=D(j)==0;
   B7=B(j)==1;
   C7=C(j)==1;
   D7=D(j)==1;
if and(and(B0,C0),D0)
Fh(j)=f(j)+F
end;
if and(and(B1,C1),D1)
Fh(j)=f(j)+2*(F)
end;
  if and(and(B2,C2),D2)
Fh(j)=f(j)+3*(F)
end;
if and(and(B3,C3),D3)

Fh(j)=f(j)+4*(F)
end;
if and(and(B4,C4),D4)
Fh(j)=f(j)+5*(F)
end;
if and(and(B5,C5),D5)
 Fh(j)=f(j)+6*(F)
end;
if and(and(B6,C6),D6)
Fh(j)=f(j)+7*(F)
end;
if and(and(B7,C7),D7)
   Fh(j)=f(j)+8*(F)
end
end;
pHase(1)=0.0;
pHase(2)=0.1;
pHase(3)=0.2;



Tb=1/freq;
for j=1:1
    t=0;
    for i=1:200
if (i>=3)
       pHase(i)=i-round(i/3)*3;
   end;
   y1(i)=A*sin(2*pi*(Fh1(j)*t-(pHase(i)/2*pi)));

     fs(i)=A*sin(2*pi*(Fh(j)*t-(phase_round2(i)/2*pi)))+amp_noise(i);


    fs1(i)=fs(i)*y1(i);


    t=t+Tb/200 ;
end
end
fs2=2*filter(b1,a1,fs1)-0.92*A;
for i=1:200
fs3(i)=fs2(i).*fs2(i);
```

```
end
Lamb=1;
i=1
Pd=0.99;
Pfa=1e-2;
B=(1-Pd)/(1-Pfa);
A=Pd/Pfa
k=1;
 j=round(unifrnd(1,100))

while(i<=100)

   h1=k*0.01*fs3(i);
   h2=h1*h1;
   h3=h2*h2;
   h4=h2/4-h3/64;
   h5=h4-A/200;
   h6=k*0.01*fs3(i+50);
   h7=h6*h6;
   h8=h7*h7;
   h9=h7/4-h8/64;
   h10=h9-A/200;
   h11=k*0.01*fs3(i+100);
   h12=h11*h11;
   h13=h12*h12;
   h14=h12/4-h13/64;
   h15=h14-A/200;
   Lamb=h5+h10+h15
   subplot(2,1,1),
   plot(i,1e-5*Lamb,'.'),
   hold on,
   axis([0 100 -5 5]),
   grid on,
   title('Sequential-Probabilty-Ratio-Test'),
   xlabel('Sample number,n'),
   ylabel('log(lambda)'),
   subplot(2,1,1),
   plot(i,log(A)),
   hold on
   subplot(2,1,1)
   plot(i,log(B)),
   hold on,
   axis([0 100 -5 5]),
   grid on
   subplot(2,1,2)
   plot(i,phase_round1(j)),
   hold on,
   axis([0 100 0 0.3]),

    i=i+1
end
t=linspace(0,0.3,100),
  subplot(2,1,2),
  plot(j,t),
  hold on,
  axis([0 100 0 0.3]),
  subplot(2,1,2),
  plot(j,phase_round1(j),'r+')
  hold on,
```

```
  axis([0 100 0 0.3]),
  text(j,phase_round1(j),'INP')
  title('The Correct Phase')
  xlabel(' Sample-Number'),
  ylabel('Phase'),

poscmd8 = [.75 .95 .22 .05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','Phase_Track');
 set(cmd8,'callback','Phase_Track');
```

File 17 (Phase_Track)

```
%This is a MatLab program which simulates the Phase Tracking proceedure
%*********************************************************************
close
Tc=1/freq;
Kd=0.00475;
gc=0.00008975;
N=256;
%*********************************************************************
 Rp=0.5;
 Rs=10;
 Wn1= 1.5*(freq+delf)/2000;
 [b1 a1]=ellip(10,Rp,Rs,Wn1);
%*********************************************************************

Phase_Trac(1)=phase_noise(1);
count1=0
for i=1:200
   count1=count1+1;
   Phase_Diff(i)=phase_noise(i)/Tc-Phase_Trac(i)/Tc;
   mu(i)=filter(b1,a1,Kd*Phase_Diff(i))
   sum1=0;
   q=0;
   T=i+1;
   for t=q:1/i*N:T
      h=1/i*N;
         if or((t==q),(t==T))
            sum1=sum1+(h/2)*mu(i);
         else
               sum1=sum1+h*mu(i);
            end;
         end;

            Phase_Trac(i+1)=(gc*(sum1));


         end;
         subplot(2,2,1),
         plot(phase_noise),
         hold on,
         grid on,
         axis([0 200 0 2]),
         Title('PHASE NOISE'),
         xlabel('time'),
```

```
          ylabel('Volts'),
          subplot(2,2,2),
          plot((4/1.8)*Phase_Trac),
          hold on,
          grid on,
          axis([0 200 0 2])
          Title('PHASE TRAC'),
          xlabel('time'),
          ylabel('Volts'),

          subplot(2,2,3),
          plot(0.0002*Phase_Diff),
          hold on,
          grid on,
          axis([0 200 0 2])
          Title('PHASE DIFF'),
          xlabel('time'),
          ylabel('Volts'),
          subplot(2,2,4),
          plot(0.00006*mu),
          hold on,
          grid on,
          axis([0 200 -0.0009 0.0041])
          Title('MU'),
          xlabel('time'),
          ylabel('Volts'),



poscmd8 = [.4 .95 .22 .05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','Code_Track');
 set(cmd8,'callback','Code_Track');
```

File 17 (Code_Track)

```
%this is a MAtLAb program which simulates and plots the frequency hoping
process
posz1=[0.03 0.04 .950 .45];%to create the active drawing area
axes('position',posz1);
grid on
%defining background colour
%*****************************
bkcolofed7 = [0.3 .7 .3 ];
bkcolofed8 = [0.3 .7 .3 ];
close
grid on
 Rp=0.5;
 Rs=20;
 Wn1= 1.5*(freq+delf)/1000;
 [b1 a1]=ellip(10,Rp,Rs,Wn1);

LimIt=(RET-1)*7;
```

```
for j=1:1
   B0=B(j)==0;
   C0=C(j)==0;
   D0=D(j)==0;
   B1=B(j)==0;
   C1=C(j)==0;
   D1=D(j)==1;
   B2=B(j)==0;
   C2=C(j)==1;
   D2=D(j)==0;
   B3=B(j)==0;
   C3=C(j)==1;
   D3=D(j)==1;
   B4=B(j)==1;
   C4=C(j)==0;
   D4=D(j)==0;
   B5=B(j)==1;
   C5=C(j)==0;
   D5=D(j)==1;
   B6=B(j)==1;
   C6=C(j)==1;
   D6=D(j)==0;
   B7=B(j)==1;
   C7=C(j)==1;
   D7=D(j)==1;
if and(and(B0,C0),D0)
Fh1(j)=freq
end;
if and(and(B1,C1),D1)
Fh1(j)=2*freq
end;
if and(and(B2,C2),D2)
Fh1(j)=3*freq
end;
if and(and(B3,C3),D3)

Fh1(j)=4*freq
end;
if and(and(B4,C4),D4)
Fh1(j)=5*freq
end;
if and(and(B5,C5),D5)
 Fh1(j)=6*freq
end;
if and(and(B6,C6),D6)
Fh1(j)=7*freq
end;
if and(and(B7,C7),D7)
   Fh1(j)=8*freq
end
end;
q=0.0;
N=100;
T=0;
Tb=1/freq;
%defining background colour
%*****************************
bkcolofed7 = [0.3 .7 .3 ];
bkcolofed8 = [0.3 .7 .3 ];
```

```
close
grid on
LimIt=(RET-1)*7;
for j=1:1
   B0=B(j)==0;
   C0=C(j)==0;
   D0=D(j)==0;
   B1=B(j)==0;
   C1=C(j)==0;
   D1=D(j)==1;
   B2=B(j)==0;
   C2=C(j)==1;
   D2=D(j)==0;
   B3=B(j)==0;
   C3=C(j)==1;
   D3=D(j)==1;
   B4=B(j)==1;
   C4=C(j)==0;
   D4=D(j)==0;
   B5=B(j)==1;
   C5=C(j)==0;
   D5=D(j)==1;
   B6=B(j)==1;
   C6=C(j)==1;
   D6=D(j)==0;
   B7=B(j)==1;
   C7=C(j)==1;
   D7=D(j)==1;
if and(and(B0,C0),D0)
Fh(j)=f(j)+F
end;
if and(and(B1,C1),D1)
Fh(j)=f(j)+2*(F)
end;
if and(and(B2,C2),D2)
Fh(j)=f(j)+3*(F)
end;
if and(and(B3,C3),D3)

Fh(j)=f(j)+4*(F)
end;
if and(and(B4,C4),D4)
Fh(j)=f(j)+5*(F)
end;
if and(and(B5,C5),D5)
 Fh(j)=f(j)+6*(F)
end;
if and(and(B6,C6),D6)
Fh(j)=f(j)+7*(F)
end;
if and(and(B7,C7),D7)
   Fh(j)=f(j)+8*(F)
end
end;

for j=1:1

    t=0;
    for i=1:200
```

```
       fs14(i)=A*sin(2*pi*(Fh(j)*t-(phase_noise(i)/2*pi)));%the    modulated
signal to be tracked
       fs24(i)=A*sin(2*pi*(Fh(j)*t-(Phase_Trac(i)/2*pi)));%the      tracking
signal
         t=t+Tb/200

end
end
subplot(2,1,1),
plot(fs14),
hold on,
grid on,
axis([0 200 -1*A 1*A]),
Title('The modulated signal to be tracked')
ylabel('signal in Volts'),
xlabel('time')
%************************************************************
subplot(2,1,2),
plot(fs24),
hold on,
grid on,
axis([0 200 -1.001*A 1.001*A]),
Title('The Tracking Signal to be tracked at the Receiver')
ylabel('signal in Volts'),
xlabel('time')
bkcoloftx=[1 0 1]
hpop = uicontrol('Style', 'popup',...
       'String', '1|2|3|4|5|6|7',...
       'Position', [10 330 40 50],...
       'Callback', 'val = get(hpop,''Value'');');

poscmd8 = [.1 .95 .12 .05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','Mixer');
 set(cmd8,'callback','Mixer');
```

File 18 (Mixer)

```
%F1 = str2num(get(ed99,'string'));
F1=abs(val)
%this is a MAtLAb program which simulates and plots the frequency hoping
process
posz1=[0.03 0.04 .950 .45];%to create the active drawing area
axes('position',posz1);
grid on
%defining background colour
%*****************************
bkcolofed7 = [0.3 .7 .3 ];
bkcolofed8 = [0.3 .7 .3 ];
close
grid on
%shift register initialization
a(1)=0;
b(1)=0;
c(1)=0;
```

```
d(1)=0;
e(1)=0;
f(1)=0;
g(1)=1;
%mulipliers initialization
G(1)=1;
G(2)=0;
G(3)=1;
G(4)=0;
G(5)=0;
G(6)=1;
G(7)=1;
G(8) =1;
%dtata output intialization
B(1)=1
%data generation of the first code sequence
for i=1:127
   a(i+1)=and(B(i),G(8));
   dummy=and(B(i),G(7));
   b(i+1)=xor(dummy,a(i));
   dummy=(B(i)*G(6));
   c(i+1)=xor(dummy,b(i));
   dummy=(B(i)*G(5));
   d(i+1)=xor(dummy,c(i));
   dummy=(B(i)*G(4));
   e(i+1)=xor(dummy,d(i));
   dummy=(B(i)*G(3));
   f(i+1)=xor(dummy,e(i));
   dummy=(B(i)*G(2));
   g(i+1)=xor(dummy,f(i));
   B(i+1)=g(i+1)
end
%shift register initialization
a(1)=0;
b(1)=0;
c(1)=0;
d(1)=0;
e(1)=0;
f(1)=0;
g(1)=1;
%mulipliers initialization
G(1)=1;
G(2)=1;
G(3)=0;
G(4)=1;
G(5)=0;
G(6)=0;
G(7)=1;
G(8) =1;
%dtata output intialization
C(1)=1
%data generation of the second code sequence
for i=1:127
   a(i+1)=and(C(i),G(8));
   dummy=and(C(i),G(7));
   b(i+1)=xor(dummy,a(i));
   dummy=(C(i)*G(6));
   c(i+1)=xor(dummy,b(i));
   dummy=(C(i)*G(5));
```

```
   d(i+1)=xor(dummy,c(i));
   dummy=(C(i)*G(4));
   e(i+1)=xor(dummy,d(i));
   dummy=(C(i)*G(3));
   f(i+1)=xor(dummy,e(i));
   dummy=(C(i)*G(2));
   g(i+1)=xor(dummy,f(i));
   C(i+1)=g(i+1)
end
%shift register initialization
a(1)=0;
b(1)=0;
c(1)=0;
d(1)=0;
e(1)=0;
f(1)=0;
g(1)=1;
%mulipliers initialization
G(1)=1;
G(2)=0;
G(3)=1;
G(4)=0;
G(5)=0;
G(6)=1;
G(7)=1;
G(8) =1;
%dtata output intialization
B(1)=1
%data generation of the first code sequence
for i=1:127
   a(i+1)=and(B(i),G(8));
   dummy=and(B(i),G(7));
   b(i+1)=xor(dummy,a(i));
   dummy=(B(i)*G(6));
   c(i+1)=xor(dummy,b(i));
   dummy=(B(i)*G(5));
   d(i+1)=xor(dummy,c(i));
   dummy=(B(i)*G(4));
   e(i+1)=xor(dummy,d(i));
   dummy=(B(i)*G(3));
   f(i+1)=xor(dummy,e(i));
   dummy=(B(i)*G(2));
   g(i+1)=xor(dummy,f(i));
   B(i+1)=g(i+1)
end
%shift register initialization
a(1)=0;
b(1)=0;
c(1)=0;
d(1)=0;
e(1)=0;
f(1)=0;
g(1)=1;
%mulipliers initialization
G(1)=1;
G(2)=0;
G(3)=1;
G(4)=0;
G(5)=1;
```

```
G(6)=0;
G(7)=1;
G(8) =1;
%dtata output intialization
D(1)=1
%data generation of the third code sequence
for i=1:127
   a(i+1)=and(D(i),G(8));
   dummy=and(D(i),G(7));
   b(i+1)=xor(dummy,a(i));
   dummy=(D(i)*G(6));
   c(i+1)=xor(dummy,b(i));
   dummy=(D(i)*G(5));
   d(i+1)=xor(dummy,c(i));
   dummy=(D(i)*G(4));
   e(i+1)=xor(dummy,d(i));
   dummy=(D(i)*G(3));
   f(i+1)=xor(dummy,e(i));
   dummy=(D(i)*G(2));
   g(i+1)=xor(dummy,f(i));
   D(i+1)=g(i+1)
End

for j=1:F1
   B0=B(j)==0
   C0=C(j)==0;
   D0=D(j)==0;
   B1=B(j)==0;
   C1=C(j)==0;
   D1=D(j)==1;
   B2=B(j)==0;
   C2=C(j)==1;
   D2=D(j)==0;
   B3=B(j)==0;
   C3=C(j)==1;
   D3=D(j)==1;
   B4=B(j)==1;
   C4=C(j)==0;
   D4=D(j)==0;
   B5=B(j)==1;
   C5=C(j)==0;
   D5=D(j)==1;
   B6=B(j)==1;
   C6=C(j)==1;
   D6=D(j)==0;
   B7=B(j)==1;
   C7=C(j)==1;
   D7=D(j)==1;
if and(and(B0,C0),D0)
fh1(j)=freq
end;
if and(and(B1,C1),D1)
fh1(j)=2*freq
end;
if and(and(B2,C2),D2)
fh1(j)=3*freq
end;
if and(and(B3,C3),D3)
```

```
fh1(j)=4*freq
end;
if and(and(B4,C4),D4)
fh1(j)=5*freq
end;
if and(and(B5,C5),D5)
 fh1(j)=6*freq
end;
if and(and(B6,C6),D6)
fh1(j)=7*freq
end;
if and(and(B7,C7),D7)
   fh1(j)=8*freq
end
end;
q=0.0;
N=100;
T=0;
Tb=1/freq;
%defining background colour
%******************************
for j=1:F1
   T=T+Tb
     if (j==1)
        t=linspace(q,T,N);
        m1=A*sin(2*pi*fh1(j)*t);
         fs=A*sin(2*pi*Fh(j)*t).*m1;
          plot(t,0.0095*fs,'b')
         hold on
         axis([0 T -1*A A])
         grid on
         end         if (j>1)
      t=linspace(q,T,N);
      t1=t-T;
      m1=A*sin(2*pi*fh1(j)*t1);
         fs=A*sin(2*pi*Fh(j)*t1).*m1;
         plot(t,0.0095*fs,'b')
         hold on
         axis([0 T -1*A A])


 end
           q=T;

        end
        title('THE MIXER OUTPUT')
        xlabel('time in units of micro second')
        ylabel(' Volts')


%postx99=[0.25 0.95 0.35 0.05]
  %tx99 = uicontrol('style','text','units',...
   %'normalized','position',postx99,...
   %'string','ENTER              YOUR              TIME               SCALE
','tag','text1','fontsize',6,'BackgroundColor',bkcoloftx);
posed99=[0.65 0.95 0.12 0.05]
ed99=  uicontrol('style','edit','units',...
   'normalized','position',posed99,...
   'string','1','tag','edit99','fontsize',7,'BackgroundColor',bkcolofed);
```

```
    poscmd8 = [.05 .95 .22 .05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','iir_design');
 set(cmd8,'callback','iir_design');
```

File 19 (iir_design)

```
%
% IIR Filter Design Program
%
clc
close
N=input('Enter order of prototype filter =>');
kind=input('Enter one of butt/bess/cheb1/cheb2/elli =>','s');
switch kind
case 'butt',
   [z,p,k]=buttap(N);
case 'bess',
   [z,p,k]=besselap(N);
case 'cheb1',
   rp=input('Enter passband ripple in dB =>');
   [z,p,k]=cheb1ap(N,rp);
case 'cheb2',
   rs=input('Enter stopband ripple in dB =>');
   [z,p,k]=cheb2ap(N,rs);
case 'elli',
   rp=input('Enter passband ripple in dB =>');
   rs=input('Enter stopband ripple in dB =>');
   [z,p,k]=ellipap(N,rp,rs);
end;
num=poly(z);
den=poly(p);
figure(1); freqs(num,den); title('Prototype Freq. Response, w0=1');
figure(2); plot(p,'x','linewidth',2),grid on;
axis square; title('Poles of Prototype Filter');
type=input('Enter freq-BAND conversion to lp/hp/bp/bs/none =>','s');
switch type
case 'lp',
   W0=input('Enter new cutoff freq in Hz =>');
   W0=2*pi*W0;
   [numt,dent]=lp2lp(num,den,W0);
case 'hp',
   W0=input('Enter new cutoff freq in Hz =>');
   W0=2*pi*W0;
   [numt,dent]=lp2hp(num,den,W0);
case 'bp',
   W0=input('Enter new cutoff freq in Hz =>');
   BW=input('Enter Bandwidth in Hz =>');
   W0=2*pi*W0; BW=2*pi*BW;
   [numt,dent]=lp2bp(num,den,W0,BW);
case 'bs',
   W0=input('Enter new cutoff freq in Hz =>');
   BW=input('Enter Bandwidth in Hz =>');
   W0=2*pi*W0; BW=2*pi*BW;
   [numt,dent]=lp2bs(num,den,W0,BW);
end;
figure(3); freqs(numt,dent); title('Freq. Response in rad/s');
```

```
digital=input('Enter bilinear/impulse invariant (bil/imp) =>','s');
switch digital
case 'bil',
   Fs=input('Enter sampling frequency in Hz =>');
   [numd,dend]=bilinear(numt,dent,Fs);
case 'imp',
   Fs=input('Enter sampling frequency in Hz =>');
   [numd,dend]=impinvar(numt,dent,Fs);
end;
figure(4);
freqz(numd,dend); title('Freq. Response of Digital Filter');

polesd=roots(dend);
zerosd=roots(numd);
figure(5);
plot(polesd,'x','linewidth',2); hold on,grid on;
 poscmd8 = [.15 .95 .12 .05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','BandP');
 set(cmd8,'callback','BandP');


for i=1:size(zerosd,1)
   if zeros(i)==0
      plot(0,0,'o','linewidth',2);
   end;
end;
plot(zerosd,'o','linewidth',2);
plot(exp(j*2*pi*[-0.5:0.01:0.5]),'-');
axis([-1,1,-1,1]); title('poles/zeros of Digital Filter');
axis square; hold off;
```

File 20 (BandP)

```
%F1 = str2num(get(ed99,'string'));
F1=abs(val)
%this is a MAtLAb program which simulates and plots the frequency hoping
process
posz1=[0.03 0.04 .950 .45];%to create the active drawing area
axes('position',posz1);
grid on
%defining background colour
%******************************
bkcolofed7 = [0.3 .7 .3 ];
bkcolofed8 = [0.3 .7 .3 ];
Wn1= 1.5*(freq+delf)/2000;
 [b1 a1]=ellip(10,Rp,Rs,Wn1);
 close
 close
 close
 close
 close
grid on
%shift register initialization
a(1)=0;
b(1)=0;
c(1)=0;
```

```
d(1)=0;
e(1)=0;
f(1)=0;
g(1)=1;
%muliplіers initialization
G(1)=1;
G(2)=0;
G(3)=1;
G(4)=0;
G(5)=0;
G(6)=1;
G(7)=1;
G(8) =1;
%dtata output intialization
B(1)=1
%data generation of the first code sequence
for i=1:127
   a(i+1)=and(B(i),G(8));
   dummy=and(B(i),G(7));
   b(i+1)=xor(dummy,a(i));
   dummy=(B(i)*G(6));
   c(i+1)=xor(dummy,b(i));
   dummy=(B(i)*G(5));
   d(i+1)=xor(dummy,c(i));
   dummy=(B(i)*G(4));
   e(i+1)=xor(dummy,d(i));
   dummy=(B(i)*G(3));
   f(i+1)=xor(dummy,e(i));
   dummy=(B(i)*G(2));
   g(i+1)=xor(dummy,f(i));
   B(i+1)=g(i+1)
end
%shift register initialization
a(1)=0;
b(1)=0;
c(1)=0;
d(1)=0;
e(1)=0;
f(1)=0;
g(1)=1;
%muliplіers initialization
G(1)=1;
G(2)=1;
G(3)=0;
G(4)=1;
G(5)=0;
G(6)=0;
G(7)=1;
G(8) =1;
%dtata output intialization
C(1)=1
%data generation of the second code sequence
for i=1:127
   a(i+1)=and(C(i),G(8));
   dummy=and(C(i),G(7));
   b(i+1)=xor(dummy,a(i));
   dummy=(C(i)*G(6));
   c(i+1)=xor(dummy,b(i));
   dummy=(C(i)*G(5));
```

```
   d(i+1)=xor(dummy,c(i));
   dummy=(C(i)*G(4));
   e(i+1)=xor(dummy,d(i));
   dummy=(C(i)*G(3));
   f(i+1)=xor(dummy,e(i));
   dummy=(C(i)*G(2));
   g(i+1)=xor(dummy,f(i));
   C(i+1)=g(i+1)
end
%shift register initialization
a(1)=0;
b(1)=0;
c(1)=0;
d(1)=0;
e(1)=0;
f(1)=0;
g(1)=1;
%mulipliers initialization
G(1)=1;
G(2)=0;
G(3)=1;
G(4)=0;
G(5)=0;
G(6)=1;
G(7)=1;
G(8) =1;
%dtata output intialization
B(1)=1
%data generation of the first code sequence
for i=1:127
   a(i+1)=and(B(i),G(8));
   dummy=and(B(i),G(7));
   b(i+1)=xor(dummy,a(i));
   dummy=(B(i)*G(6));
   c(i+1)=xor(dummy,b(i));
   dummy=(B(i)*G(5));
   d(i+1)=xor(dummy,c(i));
   dummy=(B(i)*G(4));
   e(i+1)=xor(dummy,d(i));
   dummy=(B(i)*G(3));
   f(i+1)=xor(dummy,e(i));
   dummy=(B(i)*G(2));
   g(i+1)=xor(dummy,f(i));
   B(i+1)=g(i+1)
end
%shift register initialization
a(1)=0;
b(1)=0;
c(1)=0;
d(1)=0;
e(1)=0;
f(1)=0;
g(1)=1;
%mulipliers initialization
G(1)=1;
G(2)=0;
G(3)=1;
G(4)=0;
G(5)=1;
```

```
G(6)=0;
G(7)=1;
G(8) =1;
%dtata output intialization
D(1)=1
%data generation of the third code sequence
for i=1:127
   a(i+1)=and(D(i),G(8));
   dummy=and(D(i),G(7));
   b(i+1)=xor(dummy,a(i));
   dummy=(D(i)*G(6));
   c(i+1)=xor(dummy,b(i));
   dummy=(D(i)*G(5));
   d(i+1)=xor(dummy,c(i));
   dummy=(D(i)*G(4));
   e(i+1)=xor(dummy,d(i));
   dummy=(D(i)*G(3));
   f(i+1)=xor(dummy,e(i));
   dummy=(D(i)*G(2));
   g(i+1)=xor(dummy,f(i));
   D(i+1)=g(i+1)
end

for j=1:F1
   B0=B(j)==0
   C0=C(j)==0;
   D0=D(j)==0;
   B1=B(j)==0;
   C1=C(j)==0;
   D1=D(j)==1;
   B2=B(j)==0;
   C2=C(j)==1;
   D2=D(j)==0;
   B3=B(j)==0;
   C3=C(j)==1;
   D3=D(j)==1;
   B4=B(j)==1;
   C4=C(j)==0;
   D4=D(j)==0;
   B5=B(j)==1;
   C5=C(j)==0;
   D5=D(j)==1;
   B6=B(j)==1;
   C6=C(j)==1;
   D6=D(j)==0;
   B7=B(j)==1;
   C7=C(j)==1;
   D7=D(j)==1;
if and(and(B0,C0),D0)
fh1(j)=freq
end;
if and(and(B1,C1),D1)
fh1(j)=2*freq
end;
if and(and(B2,C2),D2)
fh1(j)=3*freq
end;
if and(and(B3,C3),D3)
```

```
fh1(j)=4*freq
end;
if and(and(B4,C4),D4)
fh1(j)=5*freq
end;
if and(and(B5,C5),D5)
 fh1(j)=6*freq
end;
if and(and(B6,C6),D6)
fh1(j)=7*freq
end;
if and(and(B7,C7),D7)
   fh1(j)=8*freq
end
end;
q=0.0;
N=100;
T=0;
Tb=1/freq;
%defining background colour
%*******************************
for j=1:F1
   T=T+Tb
     if (j==1)
        t=linspace(q,T,N);
        y1=A*sin(2*pi*fh1(j)*t);
        fs=A*sin(2*pi*Fh(j)*t).*y1;
        fs1=2*filter(b1,a1,fs)-0.45*A
          plot(t,0.615*fs1,'b')
         hold on
         axis([0 T -1*A A])
         grid on
         end
        if (j>1)
      t=linspace(q,T,N);
      t1=t-T;
      y1=A*sin(2*pi*fh1(j)*t1);
      fs=A*sin(2*pi*Fh(j)*t1).*y1;
       fs1=2*filter(b1,a1,fs)-0.45*A

      plot(t,0.615*fs1,'b')
         hold on
         axis([0 T -1*A A])
         grid on

 end
           q=T;

        end
        title('THE BANDPASS OTPUT')
        xlabel('units of micro seconds')
        ylabel('Volts')
%postx99=[0.25 0.95 0.35 0.05]
  %tx99 = uicontrol('style','text','units',...
   %'normalized','position',postx99,...
   %'string','ENTER              YOUR              TIME              SCALE
','tag','text1','fontsize',6,'BackgroundColor',bkcoloftx);
posed99=[0.65 0.95 0.12 0.05]
ed99=  uicontrol('style','edit','units',...
```

```
   'normalized','position',posed99,...
   'string','1','tag','edit99','fontsize',7,'BackgroundColor',bkcolofed);

    poscmd8 = [.05 .95 .22 .05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','BandP1');
 set(cmd8,'callback','BandP1');
```

File 21 BandP1

```
    %This is a MAtLAb program which re-plots the data  of the band pass
filter out put
%The graph is the one which was ploted earlier in the FSK modulator output
%Here it is redrawn ,because it is the actual grapf obtianed in the ideal
case
%**************************************************************************
********************
close
Tb=1/freq;
Tb=per;
T=Tb;
q=0;
for i=1:RET-1
      for j=1:7
      if (BINDATA(i,j)==0)
         t=linspace(q,T,N);
         fsk=A*sin(2*pi*freq*t);
         plot(t,fsk,'r')
         hold on
              end
      if (BINDATA(i,j)==1)
         t=linspace(q,T,N);
         fsk=A*sin(2*pi*(freq+delf)*t);
         plot(t,fsk,'g')
         hold on
          end
      q=T;
      T=T+Tb;
   end
end
title('REPLOTED dAta')
xlabel(' milli seconds')
ylabel('Volts')
grid on
poscmd8 = [.05 .95 .22 .05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','FSK_D');
 set(cmd8,'callback','FSK_D');
```

File 23 (FSK_D)

```
%This is a MAtLAb program which simulates the FSK DEMODULATOR filter after
the mixer
%of the receiver
%*************************************************************************
******
close
posz1=[0.03 0.04 .950 .45];%to create the active drawing area
axes('position',posz1);
grid on
%defining background colour
%*******************************
bkcolofed7 = [0.3 .7 .3 ];
bkcolofed8 = [0.3 .7 .3 ];
close
grid on
%*************************************************************************
%This is the program which simulates and plots the FSK demodulator
%*************************************************************************
clc
Tb=per;
T=Tb;
q=0;
N=100;
for i=1:RET-1
   for j=1:7
      sum1=0;sum2=0;
      if (BINDATA(i,j)==0)
         for t=q:(T-q)/N:T
            h=(T-q)/N;%/2
            fsk=A*sin(2*pi*freq*t);
            e1=A*sin(2*pi*freq*t);
            e2=A*sin(2*pi*(freq+delf)*t);
            z1=fsk.*e1;
            z2=fsk.*e2;
            if or((t==q),(t==T))
               sum1=sum1+(h/2)*z1;
            else
               sum1=sum1+h*z1;
            end;
            if or((t==q),(t==T))
               sum2=sum2+(h/2)*z2;
            else
               sum2=sum2+h*z2;
            end;
           end

       end
        decide=sum1-sum2;
        if(decide>=0)
           datrec(i,j)=0;
        else
           datrec(i,j)=1;
           end

      if (BINDATA(i,j)==1)
         for t=q:(T-q)/N:T
            h=(T-q)/N;
           fsk=A*sin(2*pi*(freq+delf)*t);
           e1=A*sin(2*pi*freq*t);
```

```
            e2=A*sin(2*pi*(freq+delf)*t);
            z1=fsk.*e1;
            z2=fsk.*e2;
            if or((t==q),(t==T))
               sum1=sum1+(h/2)*z1;
            else
               sum1=sum1+h*z1;
            end;
            if or((t==q),(t==T))
               sum2=sum2+(h/2)*z2;
            else
               sum2=sum2+h*z2;
            end;

end
end

         decide=sum1-sum2;
         if(decide>=0)
            datrec(i,j)=0;
         else
            datrec(i,j)=1;
            end

      q=T;
      T=T+Tb;
   end

end
datrec
F=val;
LIM=0;
TIM=0;
m=1;
for k=1:RET-1%RET=p,max
   for j=1:7
      LIM=LIM+F;
       %the loop for calculating the data values
       if (datrec(k,j)==0)
          for p=m:LIM
          BINDATAOUT1(p)=-1*A
       end;
       end;

       if (datrec(k,j)==1)
          for p=m:LIM
           BINDATAOUT1(p)=A
        end;
        end;
             TIM=strcat(num2str(p));
      %AMP=strcat(num2str(BINDATAOUT1(p)));
      posed7 = [.9 .85 .1 .05]
      posed8 = [.9 .75 .1 .05]
ed7 = uicontrol('style','edit','units',...
  'normalized','position',posed7,...
   'string',TIM,'tag','edit7','fontsize',10,'BackgroundColor',bkcolofed7);
%ed8 = uicontrol('style','edit','units',...
  %'normalized','position',posed8,...
   %'string',AMP,'tag','edit8','fontsize',10,'BackgroundColor',bkcolofed7);
```

```
      %end of the loop
    m=LIM;
    end;
end;
axis([1 LIM -1.0*A 1.*A])

x = 1:LIM;
y = BINDATAOUT1(x);
 plot(x,y,'--rs','LineWidth',2,...
                 'MarkerEdgeColor','y',...
                 'MarkerFaceColor','g',...
                 'MarkerSize',2 )
             grid on
             title('THE RECIEVED BINARY DATA OUTPUT')
             xlabel('time in units of micro seconds')
             ylabel('Volts')



grid on
 poscmd8 = [.15 .95 .12 .05]; % display caculate command button
cmd8 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd8, ...
      'String','ASCII_OUT');
 set(cmd8,'callback','ASCII_OUT');
```

File 23 (ASCII_OUT)

```
   %This is a Matlab program that converts  the recieved digital data into
ASCII code
close
count=1;
for i=1:RET-1
   for j=1:7
      BIN_OUT(j)=datrec(i,j)
      end
      %if BIN_OUT==[0 0 0 0 0 0 0];
         %TEXT_OUT(count)='_'
         %end
   if  BIN_OUT==[0 0 0 0 1 0 1];
      TEXT_OUT(count)='P'
   end
   if BIN_OUT==[0 0 0 0 1 1 0]
      TEXT_OUT(count)='0'
   end
   if BIN_OUT==[0 0 0 0 1 1 1]
      TEXT_OUT(count)='p'
   end
   if BIN_OUT==[0 0 0 1 0 0 1 ]
      TEXT_OUT(count)='H'
   end
```

```
   if BIN_OUT==[0 0 0 1 0 1 0]
       TEXT_OUT(count)='('
    end
    if BIN_OUT==[0 0 0 1 0 1 1]
       TEXT_OUT(count)='h'
       end
if BIN_OUT==[0 0 0 1 1 0 1]
       TEXT_OUT(count)='X'
       end
if BIN_OUT==[0 0 0 1 1 1 0]
       TEXT_OUT(count)='8'
       end
if BIN_OUT==[0 0 0 1 1 1 1]
       TEXT_OUT(count)='t'
       end
if BIN_OUT==[0 0 1 0 0 0 1]
       TEXT_OUT(count)='D'
       end
if BIN_OUT==[0 0 1 0 0 1 0]
       TEXT_OUT(count)='$'
       end
if BIN_OUT==[0 0 1 0 0 1 1]
       TEXT_OUT(count)='d'
       end
if BIN_OUT==[0 0 1 0 1 0 1]
       TEXT_OUT(count)='T'
       end
if BIN_OUT==[0 0 1 0 1 1 0]
       TEXT_OUT(count)='4'
       end

if BIN_OUT==[0 0 1 1 0 0 1]
       TEXT_OUT(count)='L'
       end
if BIN_OUT==[0 0 1 1 0 0 1]
       TEXT_OUT(count)='L'
       end
if BIN_OUT==[0 0 1 1 0 1 0]
       TEXT_OUT(count)=','
       end
if BIN_OUT==[0 0 1 1 0 1 1]
       TEXT_OUT(count)='l'
       end
if BIN_OUT==[0 0 1 1 1 0 1]
       TEXT_OUT(count)='/'
       end
if BIN_OUT==[0 0 1 1 1 1 0]
       TEXT_OUT(count)='<'
       end
if BIN_OUT==[0 1 0 0 0 0 1]
       TEXT_OUT(count)='B'
       end
if BIN_OUT==[0 1 0 0 0 1 0]
       TEXT_OUT(count)='"'
       end
if BIN_OUT==[0 1 0 0 0 1 1]
       TEXT_OUT(count)='b'
       end
if BIN_OUT==[0 1 0 0 1 1 0]
```

```
        TEXT_OUT(count)='2'
        end
if BIN_OUT==[0 1 0 0 1 1 1]
        TEXT_OUT(count)='r'
        end

if BIN_OUT==[0 1 0 1 0 0 1]
        TEXT_OUT(count)='J'
        end
if BIN_OUT==[0 1 0 1 0 1 0]
        TEXT_OUT(count)='*'
        end
if BIN_OUT==[0 1 0 1 0 1 1]
        TEXT_OUT(count)='j'
        end
if BIN_OUT==[0 1 0 1 1 0 1]
        TEXT_OUT(count)='Z'
        end

if BIN_OUT==[0 1 0 1 1 0 1]
        TEXT_OUT(count)='Z'
        end
if BIN_OUT==[0 1 0 1 1 1 0]
        TEXT_OUT(count)=':'
        end
if BIN_OUT==[0 1 0 1 1 1 1]
        TEXT_OUT(count)='z'
        end
if BIN_OUT==[0 1 1 0 0 0 1]
        TEXT_OUT(count)='F'
        end
if BIN_OUT==[0 1 1 0 0 0 1]
        TEXT_OUT(count)='F'
        end
if BIN_OUT==[0 1 1 0 0 1 0]
        TEXT_OUT(count)='&'
        end

if BIN_OUT==[0 1 1 0 0 1 1]
        TEXT_OUT(count)='f'
        end
if BIN_OUT==[0 1 1 0 1 0 1]
        TEXT_OUT(count)='Y'
        end
if BIN_OUT==[0 1 1 0 1 1 0]
        TEXT_OUT(count)='6'
        end
if BIN_OUT==[0 1 1 0 1 0 1]
        TEXT_OUT(count)='V'
        end
if BIN_OUT==[0 1 1 0 0 1 1]
        TEXT_OUT(count)='f'
        end
if BIN_OUT==[0 1 1 0 1 1 1]
        TEXT_OUT(count)='v'
        end
if BIN_OUT==[0 1 1 1 0 1 0]
        TEXT_OUT(count)='.'
        end
```

```
if BIN_OUT==[0 1 1 1 1 1 0]
       TEXT_OUT(count)='>'
       end
if BIN_OUT==[0 1 1 1 0 0 1]
       TEXT_OUT(count)='N'
       end

if BIN_OUT==[0 1 1 1 1 0 1]
       TEXT_OUT(count)='^'
       end
if BIN_OUT==[0 1 1 1 0 1 1]
       TEXT_OUT(count)='n'
       end
if BIN_OUT==[0 1 1 1 1 1 1]
       TEXT_OUT(count)='~'
       end
if BIN_OUT==[1 0 0 0 0 1 0]
       TEXT_OUT(count)='!'
       end

if BIN_OUT==[1 0 0 0 1 1 0]
       TEXT_OUT(count)='1'
       end

if BIN_OUT==[1 0 0 0 0 0 1]
       TEXT_OUT(count)='A'
       end

if BIN_OUT==[1 0 0 0 1 0 1]
       TEXT_OUT(count)='Q'
       end
if BIN_OUT==[1 0 0 0 0 1 1]
       TEXT_OUT(count)='a'
       end
if BIN_OUT==[1 0 0 0 1 1 1]
       TEXT_OUT(count)='q'
       end

if BIN_OUT==[1 0 0 1 0 1 0]
       TEXT_OUT(count)='>'
       end
if BIN_OUT==[1 0 0 0 1 1 0]
       TEXT_OUT(count)='9'
       end

if BIN_OUT==[1 0 0 0 0 0 1]
       TEXT_OUT(count)='I'
       end
if BIN_OUT==[1 0 0 1 0 1 1]
       TEXT_OUT(count)='i'
       end
if BIN_OUT==[1 0 0 1 1 1 1]
       TEXT_OUT(count)='y'
       end
if BIN_OUT==[1 0 1 0 0 1 0]
       TEXT_OUT(count)='%'
       end
if BIN_OUT==[1 0 1 0 1 1 0]
       TEXT_OUT(count)='5'
```

```
        end
if BIN_OUT==[1 0 1 0 0 0 1]
        TEXT_OUT(count)='E'
        end
if BIN_OUT==[1 0 1 0 1 0 1]
        TEXT_OUT(count)='U'
        end
if BIN_OUT==[1 0 1 0 0 0 1]
        TEXT_OUT(count)='e'
        end
if BIN_OUT==[1 0 1 0 1 1 1]
        TEXT_OUT(count)='u'
        end
if BIN_OUT==[1 0 1 1 0 1 0]
        TEXT_OUT(count)='-'
        end
if BIN_OUT==[1 0 1 1 0 1 0]
        TEXT_OUT(count)='-'
        end

if BIN_OUT==[1 0 1 1 1 1 0]
        TEXT_OUT(count)='='
        end
if BIN_OUT==[1 0 1 1 0 0 1]
        TEXT_OUT(count)='['
        end
if BIN_OUT==[1 0 1 1 0 1 1]
        TEXT_OUT(count)='m'
        end
if BIN_OUT==[1 0 1 1 1 1 1]
        TEXT_OUT(count)=')'
        end
if BIN_OUT==[1 0 1 1 1 1 1]
        TEXT_OUT(count)='}'
        end
if BIN_OUT==[1 1 0 0 0 1 0]
        TEXT_OUT(count)='#'
        end
if BIN_OUT==[1 1 0 0 1 1 0]
        TEXT_OUT(count)='3'
        end
if BIN_OUT==[1 1 0 0 0 0 1]
        TEXT_OUT(count)='C'
        end
if BIN_OUT==[1 1 0 0 1 0 1]
        TEXT_OUT(count)='S'
        end
if BIN_OUT==[1 1 0 0 0 1 1]
        TEXT_OUT(count)='c'
        end
if BIN_OUT==[1 1 0 0 1 1 1]
        TEXT_OUT(count)='s'
        end
if BIN_OUT==[1 1 0 1 0 1 0]
        TEXT_OUT(count)='+'
        end
if BIN_OUT==[1 1 0 1 1 1 0]
        TEXT_OUT(count)=';'
        end
```

```
if BIN_OUT==[1 1 0 1 0 0 1]
       TEXT_OUT(count)='K'
       end
if BIN_OUT==[1 1 0 1 1 0 1]
       TEXT_OUT(count)=']'
       end
if BIN_OUT==[1 1 0 1 0 1 1]
       TEXT_OUT(count)='k'
       end
if BIN_OUT==[1 1 0 1 1 1 1]
       TEXT_OUT(count)='{'
       end
if BIN_OUT==[1 1 1 0 0 1 0]
       TEXT_OUT(count)='M'
       end
if BIN_OUT==[1 1 1 0 1 1 0]
       TEXT_OUT(count)='7'
       end
if BIN_OUT==[1 1 1 0 0 0 0]
       TEXT_OUT(count)='G'
       end
if BIN_OUT==[1 1 1 0 1 0 1]
       TEXT_OUT(count)='W'
       end
if BIN_OUT==[1 1 1 0 0 1 1]
       TEXT_OUT(count)='g'
       end
if BIN_OUT==[1 1 1 0 1 1 1]
       TEXT_OUT(count)='w'
       end
if BIN_OUT==[1 1 1 1 1 1 0]
       TEXT_OUT(count)='?'
       end
if BIN_OUT==[1 1 1 1 0 0 1]
       TEXT_OUT(count)='O'
       end
if BIN_OUT==[1 1 1 1 1 0 1]
       TEXT_OUT(count)='_'
       end
if BIN_OUT==[1 1 1 1 0 1 1]
       TEXT_OUT(count)='o'
       end

f = figure ('pos', [.015*(count+10)*320 90 410 320]);

figure(count+1)
postx4 = [.19 .8 .41 .09]; % display  message for ASCII
tx4 = uicontrol('style','text','units',...
   'normalized','position',postx4,...
   'string','THE       ASCII       CODE       OF       EACH       CHARACTER
','tag','text4','fontsize',5,'BackgroundColor',bkcoloftx);

bkcolofed = [1 0.5 1];%the  type of the color for writting the AScii code
bkcoloftx=[0 1 0.6];%the type of the color for writting the titles of the
Ascii code

ASC=strcat(num2str(BIN_OUT));%strcat        horizontally        concatenates
corresponding rows of the
% arrays
```

```
posed2 = [.19 .19 .58 .58]; %  position for the ASCII code

ed2 = uicontrol('style','edit','units',...
  'normalized','position',posed2,...
   'string',ASC,'tag','edit2','fontsize',10,'BackgroundColor',bkcolofed);

count=count+1;
%RET=count;
end;

poscmd1 = [.78 .52 .212 .15]; % display input text command buttoned1 =
uicontrol('style','edit','units',...


cmd1 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
      'Position',poscmd1, ...
      'String','OUT_PUT');
set(cmd1,'callback','OUT_PUT');%calls the program PLOTASCII
```

File 25 (OUT_PUT)

```
%This is a Matlab program that accepts a text from the keyboard
%You have to type @ at the end of each text
%*****************************************************************
for i=1:count% closing of the prevously created figures
    close
end
%*****************************************************************
f = figure ('pos', [50 300 490 190]);%to postion the figure
%onto which the user types his text
%*****************************************************************
posed1 = [.19 .19 .58 .58]; %  the  postion for input TEXT
postx1 = [.19 .78 .31 .09]; % the position for  input message for TEXT
postx2 = [.521 .81 .232 .132];%what should we type at the end of the TEXT
bkcolofed = [1 0.7 1];% the type of the color of the input text
bkcoloftx=[0 1 0];% the color of the text message
bkcoloftx2=[1 1 0];% the color of the text message
%*****************************************************************
ed1 = uicontrol('style','edit','units',...
   'normalized','position',posed1,...
   'string',TEXT_OUT
,'tag','edit1','fontsize',7,'BackgroundColor',bkcolofed);
tx1 = uicontrol('style','text','units',...
   'normalized','position',postx1,...
   'string','ENTER                YOUR              TEXT              HERE
','tag','text1','fontsize',6,'BackgroundColor',bkcoloftx);
tx2 = uicontrol('style','text','units',...
   'normalized','position',postx2,...
    'string','TYPE            @      AT      THE      END      OF      THE
TEXT','tag','text2','fontsize',4,'BackgroundColor',bkcoloftx2);
poscmd1 = [.78 .52 .212 .15]; % display input text command buttoned1 =
uicontrol('style','edit','units',...


cmd1 = uicontrol('Style','pushbutton',...
      'Units','normalized', ...
```

```
      'Position',poscmd1, ...
      'String','title_of_project');
set(cmd1,'callback','title_of_project');% to call the program ASCIICOn
%*************************************************************************
```

Appendix C

Zeroth order Bessel Function

$$I_o(z)\sum_{m-0}^{\infty}\frac{z^{2m}}{2^{2m}(m!)^2}$$

$$=1+\frac{z^2}{4}+\frac{z^4}{64}+\frac{z^6}{2304}+....$$

For small ,$I_0(z)$ can be approximated by the first three terms of the series .

Logarithmic approximation

$$\ln(1+\alpha) = \alpha - \frac{\alpha^2}{2} + \frac{\alpha^3}{3} - \ldots$$

Let $\alpha \cong \frac{z^2}{4} + \frac{z^4}{64}$ combine the above equation

$$\ln(I_0(z)) \approx \left(\frac{z^2}{4} + \frac{z^4}{64}\right) - \frac{1}{2}\left(\frac{z^2}{2} + \frac{z^4}{64}\right) + \frac{1}{3}\left(\frac{z^2}{4} + \frac{z^4}{64}\right) - \ldots$$

$$\approx \frac{z^2}{4} - \frac{z^4}{64}$$

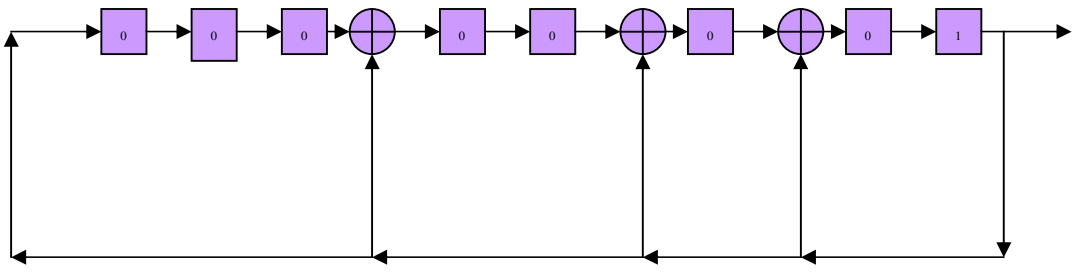
where all powers of Z graeter than 4 have been ignored .Substituing $x_K \frac{\sqrt{2P}}{N}$ yields

$$\ln\left[I_0\left(x_k \frac{\sqrt{2P}}{N}\right)\right] \approx \left(\frac{P}{2N^2}\right) x^2{}_k - \left(\frac{P^2}{16N^4}\right) x^4{}_k$$

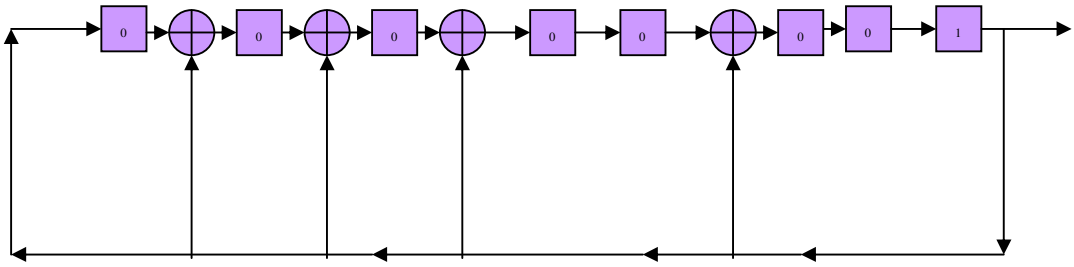
These values aree substituted in the actal program when simulating the **Sequential Detector**

Appendix D

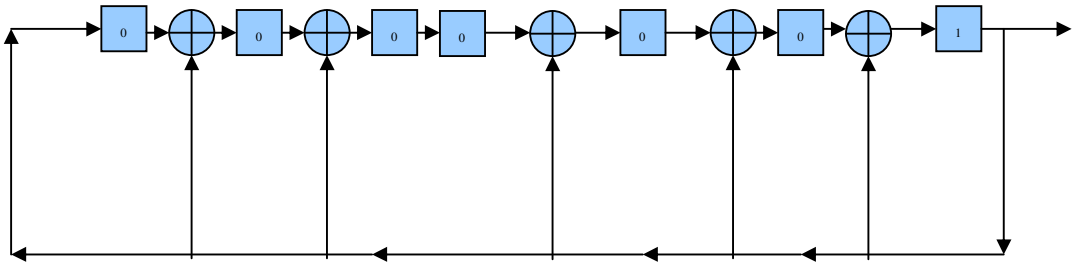
255 code generator,b(D) =1011...



551 code generator, b(D) =10101...



747 code generator,b(D) =1111...



Appendix E

Concider a coherent binary phase shift keyed (BPSK) communication system which is being used in the precence of a pulse noise jammer.A pulse noise jammer transmits pulses of bandlimited white Gaussian noise having total average power J refered to the receiver front end.The jammer may tune to the receivers center frequency and bandwidth .In addition the jammer chooses it's pulse duty factor ρ to cause maximum degradation to communication link while maitaining constant transmitted power.

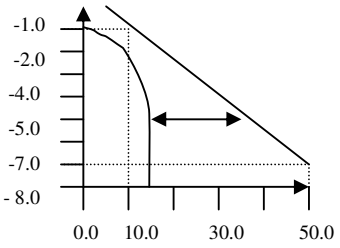
The bit error probability of coherent BPSK is given by

$$P_E = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \quad \text{----------(1)}$$

Under the above circumstances the above equation reduces to

$$\bar{P}_{E,\max} = \frac{1}{\sqrt{2\pi e}} \cdot \frac{1}{\frac{2E_b}{N_J}} \text{ ----------(2)}$$

when equations 1 and 2 are plotted:



The severe degradation in system performance caused by a combination of spread spectrum techniques and forward error correction coding with appropriate interleaving .The effect of the spectrum spreading will be to change the abscissa from $\frac{E_b}{N_J}$ to $\frac{KE_b}{N_J}$,where K is a constant about equal to $\frac{W}{R}$,where R is the data rate of the spread spectrum system.

BIBLIOGRAPHY

1. Jack K.Holmes(Holmes Associate), Coherent Spread Spectrum Systems,A Wiley-Interscience publication,1982
2. R.C.DIXON, Spread Spectrum Systems ,John Wiley&Sons,Inc.,1976
3. J.M.Wozencraft and I.M.Jacobs,Principle of communication Enigineering,(New York;Wiley,1965).
4. A.Papoulis,Probability ,Random Variables,and Stochastic Processes(New York:McGraw-Hill,1965)
5. R.A. Sholtz,"The spread Spectrum Concept,"IEE Trans.Commn.,August,1977
6. J.J.Spliker and D.T.Magill."The Delay –Lock Discriminator-An opitimum Tracking Device,"Poc.IRE,Semptember, 1961.
7. R.B.Ward ,"Acquistion of Pseudonoise Siganals by Sequencial Estimation,"IEEE,Trans.Commun,Technol.,December 1965.
8. Rodger E.Ziemer and Rodger L.Peterson,Digital Communications and Spectrum Systems, Macmillan Publishing Company,1985

9. Martin S.Roden,Digital And Data Communication Systems,Prentice-Hall,Engle wood Cliffs,1982

DECALARATION

I,the undersigned ,here by declare that this thesis is my original work,has not been presented for a degree in iany other university that all sources used for the thesis have been duly acknowledged

Name: Endale Befekadu

Signature: _____

Place : Addis Ababa

Date of submission : _____

This thesis has been submitted for examination with my approval as a university advisor.

_____ _____

Advisor Name Signature