Direct sequence spread spectrum communication simulation system

1987

Mark D. Patz
University of Central Florida

Find similar works at: https://stars.library.ucf.edu/rtd

University of Central Florida Libraries http://library.ucf.edu

Part of the Engineering Commons

STARS Citation


This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
DIRECT SEQUENCE SPREAD SPECTRUM COMMUNICATION SIMULATION SYSTEM

BY

MARK D. PATZ
B.S.E., University of Central Florida, 1983

THESIS
 Submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering in the Graduate Studies Program of the College of Engineering University of Central Florida Orlando, Florida

Summer Term
1987
ABSTRACT

The purpose of this work is to develop a fast, dynamic, and menu driven direct sequence spread spectrum communication simulation system for the Digital Equipment Corporation's VAX VMS line of computers. This software breaks the system down into two sections, the transmitter and the receiver. Each of the blocks that comprise the system is user-definable in order to program the desired conditions. The input data and spreading data located within the transmitter are generated by a maximal length sequence generator, up to the tenth order. Encoding of the data is achieved by using an (n,k) cyclic code. A level shifter and pulse shaper may be implemented prior to modulation to improve the system signal-to-noise ratio. The filter can be used after modulation to emulate channel bandwidth limitations. The input to the receiver is the sum of the transmitted wave, with White Gaussian noise, and hostile jamming capability. A coherent oscillator is available at the receiver to obtain the demodulated signal; this signal is then compared against a preset threshold level. The output from the comparator is then unspread by the correlator using an acquisition sequence to lock-in a duplicate of the spreading sequence generator. Once unspread, the signal is then decoded with a syndrome calculator routine that detects and/or corrects single bit errors.
TABLE OF CONTENTS

LIST OF TABLES ........................................ iv
LIST OF FIGURES ........................................ v

Chapter

I. INTRODUCTION ........................................ 1
   Spread Spectrum Fundamentals .................... 1
   Direct Sequence System ............................ 3
   Coding Fundamentals .............................. 7

II. SYSTEM DESCRIPTION ................................. 12
   Hardware ........................................... 12
   Software .......................................... 12

III. WAVEFORM ANALYSIS ............................... 33

IV. CONCLUSION ......................................... 34

APPENDIX ................................................ 35

REFERENCES ............................................. 108
LIST OF TABLES

1. Second Order PN Output ........................................ 4
2. (7,4) Cyclic Code Encoder Output ................................. 10
3. (7,4) Cyclic Code Decoder Output ................................ 11
4. System Main Command Menu ........................................ 13
5. Transmitter Command Menu ......................................... 16
6. Primitive Polynomial Menu ........................................ 17
7. Encoder's Typical Menu ........................................... 20
8. Receiver Command Menu ............................................ 21
9. Signal Specification Menu .......................................... 24
LIST OF FIGURES

1. Block Diagrams of Spread Spectrum Transmitters 2
2. Block Diagram of the Transmitter 14
3. Block Diagram of the Receiver 15
4. Sixth Order PN 18
5. Second Order PN 18
6. Encoder Schematic 19
7. Correlator Schematic 22
8. Decoder Schematic 23
9. Plot of Signal 1 25
10. Plot of Signal 2 25
11. Plot of Signal 3 26
12. Plot of Signal 4 26
13. Plot of Signal 5 27
14. Plot of Signal 6 27
15. Plot of Signal 7 28
16. Plot of Signal 8 28
17. Plot of Signal 9 29
18. Plot of Signal 10 29
19. Plot of Signal 11 30
20. Plot of Signal 12 30
21. Plot of Signal 13 31
22. Plot of Signal 14 ......................... 31
23. Plot of Signal 15 ......................... 32
CHAPTER I

INTRODUCTION

Spread Spectrum Fundamentals

Spread spectrum is the spreading of information energy over time or in frequency beyond the required information bandwidth. The criteria required to accomplish this are that the transmitted signal must have a higher bandwidth than the information signal and the transmitted bandwidth must be determined independently from the message. [1] This spreading makes the transmitted RF signal somewhat transparent in the channel, which is desirable in a hostile or crowded environment. The types of spread spectrum systems include: direct sequence, frequency hopping, time hopping, chirp, and combinations of these modulation methods. A direct sequence system, commonly referred to as a pseudonoise system, is an averaging system since interference is averaged over a large interval of time to reduce its ill effects. Hopping and chirp systems are avoidance systems since the signal is made to avoid the interference most of the time.

In a direct sequence spread spectrum system the message is mixed with a higher rate pseudorandomly generated binary valued periodic serial sequence. [2] This resultant spread information waveform is then modulated and transmitted; refer to Figure 1a for the block diagram.
interval. This scheme could spread the bits over time if the bit time is subdivided and pseudorandomly distributed in time. Chirp spread spectrum systems utilize linear and nonlinear frequency modulation of the carrier to spread the bandwidth. These systems are like frequency hopping, except the time variation of frequency is not pseudorandom, it is deterministic. Also, hybrid systems may be developed by combining any of the above modulation methods. This thesis deals with the direct sequence spread spectrum technique.

Many benefits can be achieved by the use of spread spectrum communication. A spread spectrum system provides excellent narrow-band antijamming capabilities, good interference rejection, multiple user carriers, and a low probability of interception. [3]

**Direct Sequence System**

A binary message source, \( m(t) \), is represented by Equation 1,

\[
  m(t) = \begin{cases} 
  0, & \text{for low level} \\ 
  1, & \text{for high level} 
  \end{cases}
\]

for \( iT_b < t < (i+1)T_b \), \( i = \text{integer} \), \( T_b \) is the bit period

where \( 1/T_b \) is the message bit rate. Spreading of the message signal in a direct sequence system is done with a high bit rate serial sequence. This sequence is generated by a pseudorandom sequence generator that is autonomous with regards to the message sequence. A maximal length pseudorandom sequence is one that repeats every \( (2^n-1) \) bits, \( n \) is the order. It is mathematically represented with primitive polynomials and is implemented with shift registers and Exclusive-OR gates.

Algebraically, a second order pseudorandom sequence generator, \( PN \), is
The ratio of the transmitted signal's bandwidth to the message signal's bandwidth is equal to the ratio of the spreading sequence's bit rate to the message sequence's bit rate, and is called the processing gain.

\[ \frac{T_B}{f_c} \]

Figure 1. Block Diagrams of Spread Spectrum Transmitters
(a) Direct Sequence
(b) Frequency Hopping

A block diagram of a frequency hopping spread spectrum system is shown in Figure 1b. In this type of system the message bit stream is modulated by a frequency synthesizer to spread the bandwidth. The frequencies are pseudorandomly generated. Time hopping spread spectrum systems are characterized by dividing the time axis into intervals, frames, which are each divided into time slots. The message bits are transmitted within one of the slots, determined pseudorandomly, for each...
depicted by Equation 2
\[ p(x) = x^2 + x + 1 \] (2)
and logically as
\[ c(x) = x^2 \oplus x \] (3)
and its output, \( c_j \), with initial register contents equal to \((1,1)\), is in Table 1. The output repeats every three bits as expected, and can be expressed by Equation 4,
\[ c_j(t) = (\text{Reg } 1)_j \oplus (\text{Reg } 2)_j \] (4)
where \((\text{Reg } 1)_j = c_{j-1}\), and \((\text{Reg } 2)_j = (\text{Reg } 1)_{j-1}\)
for \( jT_c < t < (j+1)T_c \), \( j = \text{integer} \)

\begin{table}[h]
\centering
\begin{tabular}{cccc}
\hline
TIME & REGISTER 1 & REGISTER 2 & OUTPUT \\
& \( x \) & \( x^2 \) & \( c_j \) \\
\hline
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 \\
2 & 1 & 0 & 1 \\
3 & 1 & 1 & 0 \\
4 & 0 & 1 & 1 \\
5 & 1 & 0 & 1 \\
6 & 1 & 1 & 0 \\
7 & 0 & 1 & 1 \\
8 & 1 & 0 & 1 \\
\hline
\end{tabular}
\caption{SECOND ORDER PN OUTPUT}
\end{table}

where \( 1/T_c \) is the chip rate, commonly called the spreading bit rate. Processing gain is the ratio of the chip rate to the message bit rate and is a multiple of \((2^n-1)\), \( n \) is the spreading PN order. The chip rate, shown on the following page,
\[ 1/T_c = k \left(2^n - 1\right)/T_b \] (5)

is dependent on the PN’s order \(n\), a chosen integer constant \(k\), and the message bit rate. This yields an integral amount of spreading sequences per message bit. After the message signal is mixed with the spreading signal it is modulated and transmitted. The transmitted signal is given by Equation 6.

\[
s(t) = m(t) \, c(t) \, A \cos(\omega_0 t)
\]

for \(A = \) amplitude of the carrier

This can then be expressed as the product of two functions

\[
s(t) = d(t) \, c(t)
\]

if we let,

\[
d(t) = A \, m(t) \, \cos(\omega_0 t)
\]

The power spectral density function of Equation 7 is computed to be the convolution of the power spectral density function of each of the two functions.

\[
S_s(f) = S_d(f) \ast S_c(f)
\]

Also written

\[
S_s(f) = \int_{-\infty}^{+\infty} S_d(x) \, S_c(f-x) \, dx
\]

where

\[
S_d(f) = \frac{1}{2} A^2 T_b \left\{ \text{sinc}^2[(f-f_o)T_b] + \text{sinc}^2[(f+f_o)T_b] \right\}
\]

and

\[
S_c(f) = \mathcal{F}\{R_c(\tau)\}
\]

Equation 11 represents the Fourier transform of the autocorrelation
function of the spreading signal. If the spreading signal was an infinite random sequence with equal probability of occurrence, then the autocorrelation function is a triangle wave

\[ R_c(\tau) = A(\tau/T_c) \]  

with corresponding Fourier transform

\[ S_c(f) = T_c \text{sinc}^2(fT_c) \]  

After substituting equations 11 and 14 into Equation 10 and simplifying we get

\[ S_s(f) = \frac{1}{2} A^2 T_b T_c \int_{-\infty}^{+\infty} \text{sinc}^2(x-f_0)T_b \text{sinc}^2[(f-x)T_c] \, dx \]  

\[ + \frac{1}{2} A^2 T_b T_c \int_{-\infty}^{+\infty} \text{sinc}^2(x+f_0)T_b \text{sinc}^2[(f+x)T_c] \, dx \]  

If \( T_b \) is very large in comparison to \( T_c \), very large processing gain, then

\[ \text{sinc}^2[(x+f_0)T_b] = 1/T_b \delta(x+f_0) \]  

and Equation 15 can be reduced to Equation 17.\[4\]

\[ S_s(f) = \frac{1}{2} A^2 T_c \{\text{sinc}^2[(f-f_0)T_c] + \text{sinc}^2[(f+f_0)T_c]\} \]  

When the modulated direct sequence spread spectrum information signal is received it is demodulated and unspread to provide the message signal. A correlator is used to unspread the information. It accomplishes this by using an exact replica of the spreading pseudorandom sequence. To do this the receiver must have knowledge of the PN used in the transmitter.\[5\] An acquisition sequence is a string of bits that can be produced by the spreading PN and is used to acquire
lock-in of the correlator's PN. Its length is one more than the order of the PN, since the PN sequence has unique series of length \( n+1 \). Thus a correlator based on a second order spreading PN would have a three bit acquisition sequence. One can choose any three consecutive output bits listed in Table 1 to be the correlator acquisition sequence. From this sequence one algebraically determines the set-up conditions of the correlator’s PN registers. When three consecutive bits of the information sequence match the acquisition sequence, the correlator's PN is started. It will be locked-in-phase with the spreading PN in an error-free environment. As the correlator’s output is mixed with the information the spreading is removed and this results in the message signal as the output.

**Coding Fundamentals**

Throughput error rate, \( P_e(r_t) \), is a product of the transmission rate, \( r_t \), and the probability of error. For the binary transmission case the probability of error, without encoding is shown in Equation 18.

\[
P_e = P(0) P_e(0) + P(1) P_e(1)
\]

(18)

where

\[
P(0) = \text{probability of transmitting a '0'}
\]

\[
P_e(0) = \text{probability of error when transmitting a '0'}
\]

\[
P(1) = \text{probability of transmitting a '1'}
\]

\[
P_e(1) = \text{probability of error when transmitting a '1'}
\]

\[
P(0) + P(1) = 1
\]

In a symmetric channel

\[
P(0) = P(1)
\]

(19)

thus
\[ P(0) = \frac{1}{2} \quad \text{and} \quad P(1) = \frac{1}{2} \]  

When transmitting data at a rate, \( r_d \), equal to 8000 bits/sec within a channel that is characterized by the values in Equation 21,

\[ P_e(0) = P_e(1) = 5 \times 10^{-8} \]

we obtain an overall system error rate of

\[ P_e(8000) = 8000 \left[ \frac{1}{2} \left( 5 \times 10^{-8} \right) + \frac{1}{2} \left( 5 \times 10^{-8} \right) \right] \]

\[ = 4 \times 10^{-4} \text{ bits/sec} \]

If the data rate is increased to 14000 bits/sec then

\[ P_e(14000) = (4 \times 10^{-4}) \frac{14000}{8000} \]

\[ = 7 \times 10^{-4} \text{ bits/sec} \]

Overall system error rate can be improved by encoding the binary data to produce the message signal.

Linear block codes, algebraic codes, can be represented with matrix algebra.\(^6\) A subclass of linear block codes called cyclic codes are designed using polynomial algebra and constructed with shift registers. These codes are easily implemented. Encoding using an \((n,k)\) cyclic code is very efficient for error detection and can be used for error correction. They are defined by "n," the total number of bits per message word, and "k," the number of data bits per message word. The message word is composed of the \( k \) data bits and \((n-k)\) code bits that are algebraically developed from the data bits. An \((n,k)\) cyclic code is generated by a polynomial that is a factor of \((x^n + 1)\). In Equation 24 we see that there are three possible polynomials that can be used.

\[ (x^7 + 1) = (x^3 + x^2 + 1) (x^3 + x + 1) (x + 1) \]  

\[ (24) \]

By choosing the first function to be the polynomial generator,

\[ g(x) = (x^3 + x^2 + 1) \]  

\[ (25) \]

the linear block code is defined and the number of data bit per word is
equal to four. It can be represented with matrix algebra as shown in

$$G = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1
\end{bmatrix}$$

(26)

Equation 25. The encoder generator matrix, $G$, is multiplied by the data word matrix, $D$, to obtain the message word matrix, $M$, which is the encoder output.

$$M = D G$$

(27)

The message bit rate is related to the data rate as shown below.

$$r_m = r_d \left( \frac{n}{k} \right)$$

(28)

$$= 8000 \ (7/4)$$

$$= 14000 \ \text{bits/sec}$$

If some, $c$, of the message bits in error can be corrected then the probability of error with encoding is equal to

$$P_e = 1 - (1 - P_e)^n - \sum_{i=1}^{c} \binom{n}{i} (P_e)^i (1 - P_e)^{n-i}$$

(29)

where $$\binom{n}{i} = \frac{n!}{i! \ (n-i)!}$$

System probability of error for our example used in equations 17 thru 23 with encoding using a (7,4) cyclic code and correcting for single bit errors is

$$P_e = 1 - (1 - 5 \times 10^{-8})^7 - \binom{7}{1} (5 \times 10^{-8}) (1 - 5 \times 10^{-8})^6$$

(30)

$$= 1 - 0.999999999999 - 3.4999999 \times 10^{-7}$$

$$= 2.0 \times 10^{-12}$$

for a seven bit word and the error rate is

$$P_e <14000> = 14000 \ (2.0 \times 10^{-12}) / 7$$

(31)

$$= 4 \times 10^{-9} \ \text{bits/sec}$$
When compared to the value in Equation 22 the error rate improved by a factor of one hundred thousand. Table 2 contains the message words generated by the transmitter encoder for each data word based on the polynomial in Equation 25. The decoder generator matrix, $H^T$, also called the parity check matrix, is shown in Equation 32.\[7\]

$$H^T = \begin{bmatrix}
1 & 0 & 1 \\
1 & 1 & 1 \\
1 & 1 & 0 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix} \tag{32}$$

$$S = R H^T \tag{33}$$

It is used to create the error detection matrix, $S$, that is used to check the received message matrix, $R$, for errors. If $S = [0 \ 0 \ 0]$, then no errors have occurred and the first four message bits are valid data. To correct for single bit errors we complement the received word's bit that corresponds to the row that matches $S$ in $H^T$. For example if $S = [0 \ 1 \ 1]$, the fourth row of $H^T$, we would compliment the fourth bit of the received word. All possible received words and their corrected decoder data bits are contained in Table 3 on the following page.

**TABLE 2**

(7,4) CYCLIC CODE ENCODER OUTPUT

<table>
<thead>
<tr>
<th>DATA</th>
<th>MESSAGE</th>
<th>DATA</th>
<th>MESSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0 0 0 0</td>
<td>1 0 0 0</td>
<td>1 0 0 0 1 0 1</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>0 0 0 1 0 1 1</td>
<td>1 0 0 1</td>
<td>1 0 0 1 1 1 0</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>0 0 1 1 0 1 0</td>
<td>1 0 1 0</td>
<td>1 0 1 0 0 1 1</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>0 0 1 1 1 0 1</td>
<td>1 0 1 1</td>
<td>1 0 1 1 0 0 0</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>0 1 0 0 1 1 1</td>
<td>1 1 0 0</td>
<td>1 1 0 0 0 1 0</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>0 1 0 1 1 0 0</td>
<td>1 1 0 1</td>
<td>1 1 0 1 0 0 1</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>0 1 1 0 0 0 1</td>
<td>1 1 1 0</td>
<td>1 1 1 0 1 0 0</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>0 1 1 1 0 1 0</td>
<td>1 1 1 1</td>
<td>1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>
### TABLE 3
(7,4) CYCLIC CODE DECODER OUTPUT

<table>
<thead>
<tr>
<th>RECEIVE DATA</th>
<th>RECEIVE DATA</th>
<th>RECEIVE DATA</th>
<th>RECEIVE DATA</th>
<th>RECEIVE DATA</th>
<th>RECEIVE DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000</td>
<td>0000</td>
<td>0100000</td>
<td>0000000</td>
<td>0000000</td>
<td>1100000</td>
</tr>
<tr>
<td>0000001</td>
<td>0000c</td>
<td>0100001</td>
<td>0110d</td>
<td>1000001</td>
<td>1000d</td>
</tr>
<tr>
<td>0000010</td>
<td>0000c</td>
<td>0100010</td>
<td>1100d</td>
<td>1000010</td>
<td>1100d</td>
</tr>
<tr>
<td>0000011</td>
<td>0001d</td>
<td>0100011</td>
<td>0100c</td>
<td>1000011</td>
<td>1010d</td>
</tr>
<tr>
<td>0000100</td>
<td>0000c</td>
<td>0100100</td>
<td>0101d</td>
<td>1000100</td>
<td>1000c</td>
</tr>
<tr>
<td>0000101</td>
<td>1000d</td>
<td>0100101</td>
<td>0100c</td>
<td>1000101</td>
<td>1000</td>
</tr>
<tr>
<td>0000110</td>
<td>0010d</td>
<td>0100110</td>
<td>0100c</td>
<td>1000110</td>
<td>1010d</td>
</tr>
<tr>
<td>0000111</td>
<td>0100d</td>
<td>0100111</td>
<td>0100</td>
<td>1000111</td>
<td>1000c</td>
</tr>
<tr>
<td>0001000</td>
<td>0000d</td>
<td>0101000</td>
<td>0101c</td>
<td>1001000</td>
<td>1110d</td>
</tr>
<tr>
<td>0001001</td>
<td>0001c</td>
<td>0101001</td>
<td>1101d</td>
<td>1001001</td>
<td>1101d</td>
</tr>
<tr>
<td>0001010</td>
<td>0001c</td>
<td>0101010</td>
<td>0111d</td>
<td>1001010</td>
<td>1101c</td>
</tr>
<tr>
<td>0001011</td>
<td>0001</td>
<td>0101011</td>
<td>0011d</td>
<td>1001011</td>
<td>1011d</td>
</tr>
<tr>
<td>0001100</td>
<td>0101d</td>
<td>0101100</td>
<td>0101</td>
<td>1001100</td>
<td>1001c</td>
</tr>
<tr>
<td>0001101</td>
<td>0011d</td>
<td>0101101</td>
<td>0101c</td>
<td>1001101</td>
<td>1000d</td>
</tr>
<tr>
<td>0001110</td>
<td>1001d</td>
<td>0101110</td>
<td>0101c</td>
<td>1001110</td>
<td>1001</td>
</tr>
<tr>
<td>0001111</td>
<td>0001c</td>
<td>0101111</td>
<td>0100d</td>
<td>1001111</td>
<td>1111d</td>
</tr>
<tr>
<td>0010000</td>
<td>0000d</td>
<td>0110000</td>
<td>0110c</td>
<td>1010000</td>
<td>1111d</td>
</tr>
<tr>
<td>0010001</td>
<td>0011d</td>
<td>0110001</td>
<td>0110</td>
<td>1010001</td>
<td>1111d</td>
</tr>
<tr>
<td>0010010</td>
<td>0010c</td>
<td>0110010</td>
<td>0111d</td>
<td>1010010</td>
<td>1101c</td>
</tr>
<tr>
<td>0010011</td>
<td>0110d</td>
<td>0110011</td>
<td>0110</td>
<td>1010011</td>
<td>1101d</td>
</tr>
<tr>
<td>0010100</td>
<td>0010c</td>
<td>0110100</td>
<td>1110d</td>
<td>1010100</td>
<td>1110</td>
</tr>
<tr>
<td>0010101</td>
<td>0011d</td>
<td>0110101</td>
<td>0110c</td>
<td>1010101</td>
<td>1000d</td>
</tr>
<tr>
<td>0010110</td>
<td>0010</td>
<td>0110110</td>
<td>0010d</td>
<td>1010110</td>
<td>0010d</td>
</tr>
<tr>
<td>0010111</td>
<td>0010c</td>
<td>0110111</td>
<td>0100d</td>
<td>1010111</td>
<td>1001c</td>
</tr>
<tr>
<td>0011000</td>
<td>1011d</td>
<td>0110000</td>
<td>0111c</td>
<td>1010000</td>
<td>1111d</td>
</tr>
<tr>
<td>0011001</td>
<td>0011c</td>
<td>0110001</td>
<td>0110d</td>
<td>1010001</td>
<td>1111d</td>
</tr>
<tr>
<td>0011010</td>
<td>0111d</td>
<td>0110100</td>
<td>0111</td>
<td>1010100</td>
<td>1111d</td>
</tr>
<tr>
<td>0011011</td>
<td>0001d</td>
<td>0110110</td>
<td>0111c</td>
<td>1010110</td>
<td>1101d</td>
</tr>
<tr>
<td>0011100</td>
<td>0011c</td>
<td>0111000</td>
<td>0110d</td>
<td>1011000</td>
<td>1111c</td>
</tr>
<tr>
<td>0011101</td>
<td>0011</td>
<td>0111001</td>
<td>0110d</td>
<td>1011001</td>
<td>1111d</td>
</tr>
<tr>
<td>0011110</td>
<td>0010d</td>
<td>0111110</td>
<td>0111c</td>
<td>1011110</td>
<td>1111d</td>
</tr>
<tr>
<td>0011111</td>
<td>0011c</td>
<td>0111111</td>
<td>1111d</td>
<td>1011111</td>
<td>1111</td>
</tr>
</tbody>
</table>

**c** = single bit error detected and code bit corrected
**d** = single bit error detected and data bit corrected
CHAPTER II

SYSTEM DESCRIPTION

Hardware

The program that has been developed requires a VAX computer to run the software and a Tektronix 4100 series terminal to display the block diagrams, schematics, and the signal plots. A Tektronix Color Printer can be used to obtain hardcopies of plots, diagrams, and schematics from the terminal's screen. The entire system is modeled mathematically and coded in VAX FORTRAN. The simulation assumes that a coherent carrier as well as a clock frequency are available at the receiver so that the transmitter and receiver are synchronized.

Software

The software is capable of reducing the time required to analyze direct sequence spread spectrum communication system designs. All parameters are user-definable and the system is menu driven. When the program is executed (by entering RUN COMMUNIC) it performs a set-up of the system variables using the initial values contained in the file START.ICS, it also defines all text and graphics default values. At this point, the program is ready for interactive commands and the user is prompted, by the Main Command Menu, to determine the next step.
TABLE 4
SYSTEM MAIN COMMAND MENU

Main Command Menu

1 - Show system block diagrams
2 - Transmitter Section
3 - Receiver Section
4 - Read/write a set-up file
5 - Plot a signal
6 - Exit the program

If the user requests command number 1, Show system block diagrams, then both the transmitter block diagram and the receiver block diagram will be shown on the screen. Refer to Figure 2 and Figure 3 on the following pages during their description. The transmitter is based on a data waveform that is generated with a PN and is defined by its order, initial conditions, and the transmission rate of the data bits. An encoder is included to allow for correction of errors that may be caused by the channel. The spreading wave is required to be a pseudorandom sequence and is defined by a PN's order, initial conditions, and a processing gain. Processing gain is the number of spreading sequences to be repeated per encoded data bit and is used to determine the spreading bit rate. After the encoded wave and the spreading wave are mixed the signal level can be changed by multiplying the amplitude by a defined value and adding a defined displacement. Then the square pulse many be shaped by picking an available shape. When the carrier amplitude and frequency are selected they are used for modulation of the shaped waveform. Prior to transmission the modulated wave is filtered using an available setting. At the receiver, levels for the channel noise [8] and hostile jamming [9] are used to emulate
Figure 2. Block Diagram of the Transmitter
Figure 3. Block Diagram of the Receiver
real world conditions. The total received waveform, given by Equation 34, is then demodulated using the same carrier frequency and a defined

\[ r(t) = T(t) + n(t) + j(t) \]  

(34)

where \( r \) is the received signal, \( T \) is the transmitted signal, \( n \) is the noise, and \( j \) is the jamming amplitude. This signal is then compared to a defined level to produce the input to the correlator. The correlator uses an acquisition sequence to lock-in a PN that matches the spreading waveform generator. Its function is to generate the unspreading sequence. After the spreading has been removed, the signal is then decoded and single bit errors are detected and/or corrected.

Commands 2 and 3 of the main command menu allow the user to access a submenu to define either the transmitter section or the receiver section. Within the transmitter there are many possible selections, shown in Table 5, and the user is capable of changing all, some or none of these variables. After selecting the transmitter parameters the user must return to the main menu in order to exercise the simulation.

<table>
<thead>
<tr>
<th>TABLE 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSMITTER COMMAND MENU</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Transmitter Menu</td>
</tr>
<tr>
<td>1 - Data waveform specification</td>
</tr>
<tr>
<td>2 - Encoder choice</td>
</tr>
<tr>
<td>3 - Spreading waveform specification</td>
</tr>
<tr>
<td>4 - Level shifter change</td>
</tr>
<tr>
<td>5 - Pulse shaper choice</td>
</tr>
<tr>
<td>6 - Carrier parameter change</td>
</tr>
<tr>
<td>7 - Transmitter filter choice</td>
</tr>
<tr>
<td>8 - Return to Main Menu</td>
</tr>
</tbody>
</table>
On the preceding page is the transmitter menu which lists all possible selections. Choosing function 1 of the transmitter menu allows the user to change the data specifications which are used to create a PN that generates a maximal length sequence. The PN is defined by a primitive polynomial shown in Table 6 and is implemented with shift registers and Exclusive OR gates. Examples of a sixth order PN with 101101 as the initial conditions and a second-order PN with 11 as the initial conditions are shown in figures 4 and 5. Figures 9 and 11 show the output plots from these pseudorandom sequence generators.

<table>
<thead>
<tr>
<th>TABLE 6</th>
<th>PRIMATIVE POLYNOMIAL MENU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitive Polynomial Selection Menu</td>
<td></td>
</tr>
<tr>
<td>2 - x^2 + x + 1</td>
<td></td>
</tr>
<tr>
<td>3 - x^3 + x + 1</td>
<td></td>
</tr>
<tr>
<td>4 - x^4 + x + 1</td>
<td></td>
</tr>
<tr>
<td>5 - x^5 + x^2 + 1</td>
<td></td>
</tr>
<tr>
<td>6 - x^6 + x + 1</td>
<td></td>
</tr>
<tr>
<td>7 - x^7 + x^3 + 1</td>
<td></td>
</tr>
<tr>
<td>8 - x^8 + x^4 + x^3 + x^2 + 1</td>
<td></td>
</tr>
<tr>
<td>9 - x^9 + x^4 + 1</td>
<td></td>
</tr>
<tr>
<td>10 - x^10 + x^3 + 1</td>
<td></td>
</tr>
</tbody>
</table>

Encoding is done by taking an amount of data bits and using a cyclic code polynomial to determine the code bits that are transmitted after each group of data bits. The total number of bits that make up the transmitted word is the sum of the data and the code bits. Function 2 of the transmitter menu is used to program the number of total bits and number of code bits that specify an (n,k) cyclic code. Table 7 shows a list of allowable encoder parameters, a (7,4) cyclic code encoder’s
Figure 4. Sixth Order PN

Figure 5. Second Order PN
Figure 6. Encoder Schematic
schematic is shown in Figure 6.

**TABLE 7**  
**ENCODER REQUEST MENU**

<table>
<thead>
<tr>
<th>Encoder Choice</th>
<th>Valid pairing of an (n,k) cyclic encoder values and their associated polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>(7,4)</td>
<td>$g(x) = x^3 + x^2 + 1$</td>
</tr>
<tr>
<td>(7,6)</td>
<td>$g(x) = x + 1$</td>
</tr>
</tbody>
</table>

Level shifting is based on the following equation and is used to overcome errors caused by channel noise. Amplitude and displacement are changed by executing function 4 of the transmitter menu. Function 5 of the transmitter menu gives the operator the chance to alter the shape of the pulse to a predefined shape. This function modifies the transmitted signal's bandwidth and hence modifies the signal-to-noise ratio. It is useful to reduce the transmitted signal's bandwidth, which will increase the signal-to-noise ratio. Carrier parameters can be changed by means of function 6 of the transmitter menu to enable the user to define the carrier frequency and the amplitude for modulation. The final transmitter variable is defined using function 7 of the transmitter menu. The operator may choose a predefined filter shape to emulate channel conditions.

\[ \text{new value} = (\text{old value} \times \text{amplitude}) + \text{displacement} \quad (35) \]
Table 8 lists the possible selections for the receiver section. The noise is representative of White Gaussian noise and is used to simulate typical channel conditions. Jamming is also useful to depict a crowded channel or hostile transmissions actively seeking to disrupt the transmission. These two channel characteristics are set using function 1 of the receiver menu. A coherent carrier that is synchronized with the transmitted carrier demodulates the signal; the receiver’s demodulation amplitude is set by the user with function 2 of the receiver menu. Within the receiver is a comparator that compares the demodulated waveform to a defined value to determine the signal logic level. The comparator’s threshold level is defined by means of function 3 of the receiver menu. Its output will match the output of the spreading mixer in the transmitter except for errors caused by noise and jamming in the channel. Function 4 of the receiver menu allows an acquisition sequence to be entered; this sequence is necessary to acquire lock-in of the correlator. Finally, the decoder transforms the data and code bits that may have errors to data bits with no errors. Schematics for the correlator and decoder are shown in figures 7 and 8.

<table>
<thead>
<tr>
<th>Receiver Menu</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1</strong> - Noise and Jamming specification</td>
</tr>
<tr>
<td><strong>2</strong> - Coherent carrier choice</td>
</tr>
<tr>
<td><strong>3</strong> - Comparator level change</td>
</tr>
<tr>
<td><strong>4</strong> - Correlator specification</td>
</tr>
<tr>
<td><strong>5</strong> - Decoder choice</td>
</tr>
<tr>
<td><strong>6</strong> - Return to Main Menu</td>
</tr>
</tbody>
</table>
Figure 7. Correlator Schematic
Figure 8. Decoder Schematic
The decoder is defined when the encoder in the transmitter is defined since it must decode what has been encoded. Function 5 of the receiver menu displays the required values.

The values of the user-defined variables can be read from or written to a data file using command 4 of the main command menu. The program will inquire the name of the data file to be used. Command 5 of the main command menu is used to plot any waveform along the transmitter and receiver signal path. Contained in Table 9 is a list of all signals that may be plotted. Figures 8 thru 22 show the time plots for each of the signals. After all computations are completed the user may exit the program with command 6 of the main command menu.

<table>
<thead>
<tr>
<th>TABLE 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGNAL SPECIFICATION MENU</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Plot Signal Menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - Data waveform</td>
</tr>
<tr>
<td>2 - Output of encoder</td>
</tr>
<tr>
<td>3 - Spreading waveform</td>
</tr>
<tr>
<td>4 - XOR sum of signals 2 and 3</td>
</tr>
<tr>
<td>5 - Output of shifter</td>
</tr>
<tr>
<td>6 - Output of pulse shaper</td>
</tr>
<tr>
<td>7 - Carrier modulation of signal 6</td>
</tr>
<tr>
<td>8 - Transmitted signal after filter</td>
</tr>
<tr>
<td>9 - Noise and/or Jamming signal</td>
</tr>
<tr>
<td>10 - Transmitted signal w/ channel interference</td>
</tr>
<tr>
<td>11 - Demodulated signal</td>
</tr>
<tr>
<td>12 - Compared to threshold</td>
</tr>
<tr>
<td>13 - Correlator output</td>
</tr>
<tr>
<td>14 - Link data output</td>
</tr>
<tr>
<td>15 - Data out</td>
</tr>
</tbody>
</table>
Figure 9. Plot of Signal 1

Figure 10. Plot of Signal 2
Figure 11. Plot of Signal 3

Figure 12. Plot of Signal 4
Figure 13. Plot of Signal 5

Figure 14. Plot of Signal 6
Figure 15. Plot of Signal 7

Figure 16. Plot of Signal 8
Figure 19. Plot of Signal 11

Figure 20. Plot of Signal 12
Figure 19. Plot of Signal 11

Figure 20. Plot of Signal 12
Figure 21. Plot of Signal 13

Figure 22. Plot of Signal 14
Figure 23. Plot of Signal 15
Visual analysis is allowed by displaying the desired signal in the time domain. A Fast Fourier Transform can be performed to generate a plot of the signal in the frequency domain if desired. The program must be modified to allow one to evaluate the signal-to-noise ratio and/or the power spectral density function for the system.
CHAPTER IV

CONCLUSION

Fundamentals of spread spectrum communications and cyclic code encoding techniques have been reviewed. A need to analyze various system designs without spending undo time on computations exists. The direct sequence spread spectrum communication system that has been simulated on a VAX 11/750 computer with a Tektronix 4107 terminal allows one to evaluate many different conditions. Once the system is defined, the user can view system block diagrams or subsystem schematics and display any waveform along the transmitter or receiver signal path. Work is expected to continue and may include the following capabilities to enhance system evaluation with the software: convolutional encoding, noise analysis, and probability of error.
APPENDIX

PROGRAM LISTING

START

Set-up Text
Print Introduction
Set-up Graphics
Initialize Variables

Print Main Menu
Inquire Command

Plot Block
Diagrams
Transmitter
Submenu
Receiver
Submenu
Load/Save
Data File
Plot
Signal
Exit

Figure 24. Program Flow Diagram

The basic flow diagram for the program is shown in Figure 24. On
the following pages is the program listing.
VARIABLE_LIST COMMON.VAR

Used in subroutines: COMMUNIC, COMP_SIG, IODATA, TXTREC, TXTXMT, XMIT

REAL RN, DATA_RATE,
2 PULSE_VAL(10), CARR_Amp, CARR_FREQ,
3 FLTR_VAL(10), NOISE_Amp, JAM_Amp,
4 OSC_Amp, OSC_FREQ,
5 COMPR_LEV

INTEGER IN, J, K1, K2, K3, K4, K5, N,
2 VALID_SIG,
3 DATA_POLY_ORD, DATA_IC(10),
4 DATA_PSN_FDBK(10),
5 ENCODE_TYPE,
6 ENCODE_TOTAL, ENCODE_DATA,
7 SPRD_POLY_ORD, SPRD_IC(10),
8 SPRD_BLOCK, SPRD_GAIN,
9 SPRD_PSN_FDBK(10),
1 LEVEL_Amp, LEVEL_DISP,
2 PULSE_ID,
3 CARR_CYCLE,
4 FLTR_ID,
5 CORR_AQVIS(11),
6 CORR_IC(10),
7 DECODE_TOTAL, DECODE_DATA

BYTE IFILE(30)

CHARACTER*8 TIM

EQUIVALENCE (CARR_FREQ, OSC_FREQ)
EQUIVALENCE (ENCODE_TOTAL, DECODE_TOTAL)
EQUIVALENCE (ENCODE_DATA, DECODE_DATA)
COMMON /VARIABLES/

DATA_IC, DATA_POLY_ORD, DATA_RATE,
ENCODE_TYPE, ENCODE_TOTAL, ENCODE_DATA,
SPRD_IC, SPRD_POLY_ORD, SPRD_BLOCK,
SPRD_GAIN, LEVEL_Amp, LEVEL_DISP,
PULSE_ID, PULSE_VAL,
PULSE_ID, PULSE_VAL,
CARR_Amp, CARR_CYCLE, CARR_FREQ,
FLTR_ID, FLTR_VAL,
NOISE_Amp, JAM_Amp, OSC_Amp,
COMPR_LEV, CORR_ACQIS, CORR_IC,
VALID_SIG
START.ICS

6
1 0 1 1 0 0 1 1 1 1
1000.
7 4
2
1 1 1 1 1 1 1 1 1 1
3
2 -1
0
0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
5.
12
0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
0.0010 0.0010
5.
0.0
0 1 1 0 0 0 0 1 1 0 1
! Data file
! DATA_POLY_ORD
! DATA_IC
! DATA_RATE
! ENCODE_TOTAL,!
! ENCODE_DATA
! SPRD_POLY_ORD
! SPRD_IC
! SPRD_BLOCK
! LEVEL_amp,LEVEL_DISP
! PULSE_ID
! PULSE_VAL
! CARR_amp
! CARR_CYCLE
! FLTR_ID
! FLTR VAL
! NOISE amp,JAM amp
! OSC amp
! COMPR LEV
! CORR ACQIS
PROGRAM COMMUNIC.FOR

Subroutines called: SETTXT, SETGRA
BLOCKS
XMIT
PLOT

[BELKERDID.PATZ.FFT] FFT/LIB
[BELKERDID.PATZ.PLOT] PLOT/LIB

IMPLICIT NONE
INCLUDE 'COMMON.VAR'

Begin main program

VALID SIG=0
CALL SETTXT
! Initialization of text dialog
CALL IODATA(O,'START.ICS')
! Read default parameters
WRITE (6,10)
FORMAT(' ',/,21X,44('*'),/,21X,'*',42X,'*',/
2 ,21X,'*', /,21X,'*',('/',21X,'*',42X,'*','*)/
3 ,21X,'*', /,21X,'*',('/',21X,'*',42X,'*','*)/
4 ,21X,'*', /,21X,'*',('/',21X,'*',42X,'*','*)/
5 ,21X,'*', /,21X,'*',('/',21X,'*',42X,'*','*)/
6 ,21X,'*', /,21X,'*',('/',21X,'*',42X,'*','*)/
7 ,21X,'*', /,21X,'*',('/',21X,'*',42X,'*','*)/
CALL SETGRA
! Initialization of graphics

Main Menu

20 CALL TIME(TIM)
! Read current time
WRITE (6,21) 27,76,90
! Clear dialog scroll
21 FORMAT('+',9A1)
WRITE (6,22) TIM
22 FORMAT(' ('$',10X,'MAIN COMMAND MENU',20X,'Time ',1A8,/ 2 ,'/,' ('$',6X,') - Show system block diagrams',
3 ,'/,' ('$',6X,') - Transmitter section',
4 ,'/,' ('$',6X,') - Receiver section',
5 ,'/,' ('$',6X,') - Read/write a data file',
6 ,'/,' ('$',6X,') - Plot a signal',
7 ,'/,' ('$',6X,') - Exit',/
8 ,'/,' ('$',6X,') - Enter the digit corresponding to the',
9 ,'/,' ('$',6X,') - desired command :')
READ (5,23,ERR=20) N
23 FORMAT(1I2)
IF ((N.LT.0).OR.(N.GT.6)) GOTO 20
WRITE (6,21) 27,76,90
! Clear dialog scroll
IF (N.EQ.6) GOTO 999

System Block Diagrams
IF (N.EQ.1) THEN
   CALL BLOCKS(LEVEL_AMP,LEVEL_DISP,CARR_AMP,CARR_FREQ)
ENDIF

Transmitter Section

IF (N.EQ.2) THEN
   CALL XMIT
ENDIF

Receiver Section

IF (N.EQ.3) THEN
   CALL REC
ENDIF

Read/write a data file

IF (N.EQ.4) THEN
   40 WRITE (6,21) 27,76,90
    ! Clear dialog scroll
    WRITE (6,41)
   41 FORMAT(/,'$','6X','If you wish to read a data file enter 0,'
         '2','$/','6X','if you wish to write to a data file enter 1. : ')
    READ (5,23,ERR=40) J
    WRITE (6,45)
   45 FORMAT(/,/$',6X,'Enter the file name : ')
    READ (5,46,ERR=40) IN,IFILE
    IFILE(IN+1)=0
    CALL IODATA(J,IFILE)
ENDIF

Plot Signal Section

IF (N.EQ.5) THEN
   CALL PLOT
ENDIF

GOTO 20

999 WRITE (6,21) 27,37,33,49
   ! Select code ANSI mode
END
SUBROUTINE BLOCKC(ORD,ACQ,IC,RATE)

C This subroutine plots the schematic for the correlator
C
IMPLICIT NONE
INTEGER ORD,ACQ(11),IC(10), acquisition seq, PSN IC's
2 I,J1,J2,J3,K1,K2,K3,K4,K5
REAL RATE

WRITE (6,10) 27,76,86,48 invisible
10 FORMAT('+',17A1)

WRITE (6,10) 27,83,79,62 ! Begin segment
14 J1 = ORD*200 + 1200
J2 = ORD*220 + 350
CALL COOR(O,J1,J2,K1,K2,K3,K4,K5)
WRITE (6,10) 27,82,87,K1,K2,K3,K4,K5
(J1,J2)
WRITE (6,10) 27,77,76,62 ! Return
WINDOW (0,0) -
to #14, black
WRITE (6,10) 27,77,84,60 ! Set line color
COLOR TO #12, purple
J1 = J1/2 - 143
J2 = J2 - 40
CALL COOR(1,J1,J2,K1,K2,K3,K4,K5)
CALL TEXT(24,'SCHEMATIC FOR CORRELATOR')
WRITE (6,10) 27,77,84,50 ! Set graphtext
color to #2, red
WRITE (6,10) 27,77,84,62 ! Set graphtext
color to #14, black
CALL COOR(1,20,194,K1,K2,K3,K4,K5)
CALL TEXT(5,'INPUT')
CALL COOR(1,190,194,K1,K2,K3,K4,K5)
CALL COOR(2,106,210,K1,K2,K3,K4,K5)
CALL COOR(2,90,226,K1,K2,K3,K4,K5)
CALL COOR(1,116,210,K1,K2,K3,K4,K5)
CALL COOR(2,150,210,K1,K2,K3,K4,K5)
DO 1 I=1,ORD
J1 = I*200 - 50
CALL COOR(O,J1,40,K1,K2,K3,K4,K5)
WRITE (6,10) 27,83,88,54,K1,K2,K3,K4,K5
position (J1,J2)
 Write (6,10) 27,76,75,54 ! Include copy of segment #6
CALL COOR(1,J1,210,K1,K2,K3,K4,K5)
J2 = (ORD-I)*220 + 430
CALL COOR(2,J1,J2,K1,K2,K3,K4,K5)
J3 = ORD*200 + 250
CALL COOR(2,J3,J2,K1,K2,K3,K4,K5)
CALL COOR(0,J3,J2-150,K1,K2,K3,K4,K5)
WRITE (6,10) 27,83,88,53,K1,K2,K3,K4,K5
position (J1,J2)
WRITE (6,10) 27,76,75,53

segment #5
CALL COOR(0, J3+190, J2-60, K1, K2, K3, K4, K5)
WRITE (6,10) 27, 83, 88, 52, K1, K2, K3, K4, K5
position (J1, J2)
WRITE (6,10) 27, 76, 75, 52

segment #4
CALL COOR(1, J3-20, J2-116, K1, K2, K3, K4, K5)
WRITE (6,20) 27, 76, 84, 49, ACQ(ORD-I+2)
FORMAT('+', 4Al, 11l)
CALL COOR(1, J1+20, 40, K1, K2, K3, K4, K5)
CALL COOR(2, J1+20, 30, K1, K2, K3, K4, K5)

ENDDO
CALL COOR(2, 106, 30, K1, K2, K3, K4, K5)
CALL COOR(2, 90, 46, K1, K2, K3, K4, K5)
CALL COOR(2, 90, 14, K1, K2, K3, K4, K5)
CALL COOR(1, 20, 14, K1, K2, K3, K4, K5)
CALL TEXT(5, 'CLOCK')
J1 = ORD*200 + 150
CALL COOR(1, J1, 210, K1, K2, K3, K4, K5)
CALL COOR(2, J1+100, 210, K1, K2, K3, K4, K5)
CALL COOR(1, J1+80, 94, K1, K2, K3, K4, K5)
WRITE (6,10) 27, 83, 88, 53, K1, K2, K3, K4, K5
position (J1, J2)
WRITE (6,10) 27, 76, 75, 53

segment #5
CALL COOR(0, J1+290, 150, K1, K2, K3, K4, K5)
WRITE (6,10) 27, 83, 88, 52, K1, K2, K3, K4, K5
position (J1, J2)
WRITE (6,10) 27, 76, 75, 52

segment #4
DO I=1, ORD+1
   J3 = J1 + 310
   J2 = I+220 - 60
   CALL COOR(1, J3, J2, K1, K2, K3, K4, K5)
   J3 = J3 + (I-1)*10
   CALL COOR(2, J3, J2, K1, K2, K3, K4, K5)
   J2 = 60 + I*15
   CALL COOR(2, J3, J2, K1, K2, K3, K4, K5)
   CALL COOR(2, J1+400, J2, K1, K2, K3, K4, K5)
ENDDO
CALL COOR(0, J1+400, 60, K1, K2, K3, K4, K5)
WRITE (6,10) 27, 76, 80, K1, K2, K3, K4, K5, 49
BOUNDARY
CALL COOR(2, J1+400, 260, K1, K2, K3, K4, K5)
CALL COOR(2, J1+450, 260, K1, K2, K3, K4, K5)
DO I=2, 177, 5
   J3 = J1+450 + 150*SIN(3.1415926*I/180)
   J2 = 160 + 100*COS(3.1415926*I/180)
   CALL COOR(2, J3, J2, K1, K2, K3, K4, K5)
ENDDO
CALL COOR(2, J1+450, 60, K1, K2, K3, K4, K5)
CALL COOR(2,J1+400,60,K1,K2,K3,K4,K5)  
WRITE (6,10) 27,75,69  
BOUNDARY  
CALL COOR(1,J1+600,160,K1,K2,K3,K4,K5)  
CALL COOR(2,J1+700,160,K1,K2,K3,K4,K5)  
CALL COOR(1,J1+627,168,K1,K2,K3,K4,K5)  
CALL TEXT(4,'INIT')  
CALL COOR(0,J1+700,60,K1,K2,K3,K4,K5)  
WRITE (6,10) 27,83,88,49,K1,K2,K3,K4,K5  
position (J1,J2)  
WRITE (6,10) 27,76,75,49  
segment #1  
WRITE (6,10) 27,77,84,60  
color to #12, purple  
J1 = J1 + 708  
CALL COOR(1,J1,202,K1,K2,K3,K4,K5)  
CALL TEXT(13,'UNSPREAD WAVE')  
CALL COOR(1,J1,164,K1,K2,K3,K4,K5)  
CALL TEXT(9,'PSN ERR =')  
CALL COOR(1,J1,116,K1,K2,K3,K4,K5)  
CALL TEXT(6,'RATE =')  
CALL COOR(1,J1,68,K1,K2,K3,K4,K5)  
CALL TEXT(6,'IC s =')  
WRITE (6,10) 27,77,84,50  
color to #2, red  
c  
WRITE (6,10) 27,77,84,62  
color to #14, black  
CALL COOR(1,J1+108,164,K1,K2,K3,K4,K5)  
WRITE (6,21) 27,76,84,50,ORD  
21 FORMAT('+',4A1,1I2)  
CALL COOR(1,J1+72,116,K1,K2,K3,K4,K5)  
WRITE (6,22) 27,76,84,59,RATE  
22 FORMAT('+',4A1,1PE10.3)  
DO I=1,ORD  
   J1 = I*12 + ORD*200 + 968  
   CALL COOR(1,J1,68,K1,K2,K3,K4,K5)  
   WRITE (6,20) 27,76,84,49,IC(I)  
ENDDO  
J1 = ORD*200 + 1050  
CALL COOR(1,J1,160,K1,K2,K3,K4,K5)  
CALL COOR(2,J1+50,160,K1,K2,K3,K4,K5)  
CALL COOR(2,J1+34,144,K1,K2,K3,K4,K5)  
CALL COOR(1,J1+50,160,K1,K2,K3,K4,K5)  
CALL COOR(2,J1+34,178,K1,K2,K3,K4,K5)  
CALL COOR(1,J1+60,144,K1,K2,K3,K4,K5)  
CALL TEXT(6,'OUTPUT')  
WRITE (6,10) 27,83,67  
CALL LIB$WAIT(5.0)  
WRITE (6,10) 27,83,75,62  
C  
WRITE (6,10) 27,12  
WRITE (6,10) 27,76,86,49  
RETURN
END
SUBROUTINE BLOCKD(N,K)

This subroutine plots the schematic for the decoder

IMPLICIT NONE
INTEGER N,K,
and Code bits
2 I,J1,J2,K1,K2,K3,K4,K5
WRITE (6,10) 27,76,86,48
invisible
FORMAT ('+',17A1)
WRITE (6,10) 27,83,79,62
#14
J1 = N*200 + 850
J2 = 700
CALL COOR(0,J1,J2,K1,K2,K3,K4,K5)
WRITE (6,10) 27,82,87,K1,K2,K3,K4,K5
(J1,J2)
WRITE (6,10) 27,77,76,62
to #14, black
WRITE (6,10) 27,77,84,60
color to #12, purple
J1 = J1/2 - 125
CALL COOR(1,J1,560,K1,K2,K3,K4,K5)
CALL TEXT(21,'SCHEMATIC FOR DECODER')
WRITE (6,10) 27,77,84,50
color to #2, red
WRITE (6,10) 27,77,84,62
color to #14, black
CALL COOR(1,20,484,K1,K2,K3,K4,K5)
CALL TEXT(5,'INPUT')
CALL COOR(1,90,484,K1,K2,K3,K4,K5)
CALL COOR(2,106,500,K1,K2,K3,K4,K5)
CALL COOR(2,90,516,K1,K2,K3,K4,K5)
CALL COOR(1,106,500,K1,K2,K3,K4,K5)
CALL COOR(2,150,500,K1,K2,K3,K4,K5)
CALL COOR(0,150,400,K1,K2,K3,K4,K5)
WRITE (6,10) 27,83,88,49,K1,K2,K3,K4,K5
position (J1,J2)
WRITE (6,10) 27,76,75,49
segment #1
WRITE (6,10) 27,77,84,60
color to #12, purple
CALL COOR(1,158,552,K1,K2,K3,K4,K5)
CALL TEXT(6,'BUFFER')
CALL COOR(1,158,504,K1,K2,K3,K4,K5)
CALL TEXT(6,'BITS =')
WRITE (6,10) 27,77,84,50
color to #2, red
WRITE (6,10) 27,77,84,62
color to #14, black
CALL COOR(1,230,504,K1,K2,K3,K4,K5)
WRITE (6,20) 27,76,84,50,N

! Number of Total Dummies
! Set dialog
! Begin segment
! Return
! Window (0,0) -
! Set line color
! Set graphtext
! Move
! GRAPHIC TEXT
! Move
! Move
! Move
! Move
! Return
! Set segment #1
! Include copy of
! Set graphtext
! Move
! GRAPHIC TEXT
! Move
! GRAPHIC TEXT
! Set graphtext
! Move
! GRAPHIC TEXT
20 FORMAT('+',4A1,1I2)
    CALL COOR(1,350,500,K1,K2,K3,K4,K5) ! Move
    CALL COOR(2,400,500,K1,K2,K3,K4,K5) ! Draw
    CALL COOR(1,375,500,K1,K2,K3,K4,K5) ! Move
    CALL COOR(2,375,272,K1,K2,K3,K4,K5) ! Draw
    CALL COOR(2,400,272,K1,K2,K3,K4,K5) ! Return
    CALL COOR(0,400,100,K1,K2,K3,K4,K5) ! Set segment #1
WRITE (6,10) 27,83,88,49,K1,K2,K3,K4,K5
position (J1,J2)
WRITE (6,10) 27,76,75,49
segment #1
WRITE (6,10) 27,77,84,60
! Include copy of
color to #12, purple
CALL COOR(1,408,252,K1,K2,K3,K4,K5) ! Move
CALL TEXT(2, 'IN') ! GRAPHIC TEXT
CALL COOR(1,408,156,K1,K2,K3,K4,K5) ! Move
CALL TEXT(7, 'ENCODER') ! GRAPHIC TEXT
WRITE (6,10) 27,77,84,50
! Set graphtext
c WRITE (6,10) 27,77,84,62
color to #14, black
DO I=1,N-K
    J1 = 564
    J2 = 300 - 48*I
    CALL COOR(1,J1,J2,K1,K2,K3,K4,K5) ! Move
    CALL TEXT(1, 'S') ! GRAPHIC TEXT
    J1 = J1 + 12
    CALL COOR(1,J1,J2,K1,K2,K3,K4,K5) ! Move
    WRITE (6,21) 27,76,84,49,I
    FORMAT('+',4A1,1I1)
    IF (((N.EQ.7).AND.(K.EQ.4).AND.(I.EQ.2)).OR.
        ((N.EQ.7).AND.(K.EQ.4).AND.(I.EQ.3))) THEN
        WRITE (6,10) 27,77,76,50
        to #2, red
        WRITE (6,10) 27,77,76,62
c WRITE (6,10) 27,77,76,62
to #14, black
    J2 = J2 + 36
    CALL COOR(1,564,J2,K1,K2,K3,K4,K5) ! Move
    CALL COOR(2,586,J2,K1,K2,K3,K4,K5) ! Draw
    J2 = J2 - 36
    WRITE (6,10) 27,77,76,62
to #14, black
    ENDIF
    J2 = J2 + 20
    CALL COOR(1,600,J2,K1,K2,K3,K4,K5) ! Move
    CALL COOR(2,650,J2,K1,K2,K3,K4,K5) ! Draw
ENDDO
DO I=N-K+1,4
    J2 = 320 - 48*I
    CALL COOR(1,625,J2,K1,K2,K3,K4,K5) ! Move
    CALL COOR(2,650,J2,K1,K2,K3,K4,K5) ! Draw
ENDDO
CALL COOR(0,650,100,K1,K2,K3,K4,K5) ! Return
WRITE (6,10) 27,76,80,K1,K2,K3,K4,K5,49 ! BEGIN PANEL
BOUNDARY
CALL COOR(2,650,300,K1,K2,K3,K4,K5)
CALL COOR(2,675,300,K1,K2,K3,K4,K5)
DO 1=2,177,5
   J1 = 675 + 100*SIN(3.1415926*I/180)
   J2 = 200 + 100*COS(3.1415926*I/180)
   CALL COOR(2,J1,J2,K1,K2,K3,K4,K5)
ENDO
CALL COOR(2,675,100,K1,K2,K3,K4,K5)
CALL COOR(2,650,100,K1,K2,K3,K4,K5)
WRITE (6,10) 27,76,69
BOUNDARY
CALL COOR(1,775,200,K1,K2,K3,K4,K5)
J1 = N*200 + 400
CALL COOR(2,J1,200,K1,K2,K3,K4,K5)
CALL COOR(2,J1,400,K1,K2,K3,K4,K5)
DO I=1,N
   J1 = I*200 + 200
   CALL COOR(0,J1,330,K1,K2,K3,K4,K5)
   WRITE (6,10) 27,83,88,54,K1,K2,K3,K4,K5
   position (J1,J2)
   WRITE (6,10) 27,76,75,54
   segment #6
   CALL COOR(1,J1+20,330,K1,K2,K3,K4,K5)
   CALL COOR(2,J1+20,320,K1,K2,K3,K4,K5)
ENDO
CALL COOR(2,105,320,K1,K2,K3,K4,K5)
CALL COOR(2,90,304,K1,K2,K3,K4,K5)
CALL COOR(1,108,320,K1,K2,K3,K4,K5)
CALL COOR(2,90,336,K1,K2,K3,K4,K5)
CALL COOR(1,20,304,K1,K2,K3,K4,K5)
CALL TEXT(5,'CLOCK')
CALL COOR(0,J1+200,350,K1,K2,K3,K4,K5)
WRITE (6,10) 27,83,88,53,K1,K2,K3,K4,K5
position (J1,J2)
WRITE (6,10) 27,76,75,53
segment #5
CALL COOR(1,J1+400,450,K1,K2,K3,K4,K5)
CALL COOR(2,J1+450,450,K1,K2,K3,K4,K5)
CALL COOR(2,J1+500,475,K1,K2,K3,K4,K5)
CALL COOR(1,J1+500,413,K1,K2,K3,K4,K5)
CALL TEXT(1,'C')
CALL COOR(1,J1+500,450,K1,K2,K3,K4,K5)
CALL COOR(2,J1+550,450,K1,K2,K3,K4,K5)
CALL COOR(2,J1+534,434,K1,K2,K3,K4,K5)
CALL COOR(1,J1+550,450,K1,K2,K3,K4,K5)
CALL COOR(2,J1+534,466,K1,K2,K3,K4,K5)
CALL COOR(1,J1+560,434,K1,K2,K3,K4,K5)
CALL TEXT(6,'OUTPUT')
WRITE (6,10) 27,83,67
CALL LIB$WAIT(5.0)
WRITE (6,10) 27,83,75,62
WRITE (6,10) 27,12
WRITE (6,10) 27,76,86,49
visible
SUBROUTINE BLOCKE(T,N,K)

This subroutine plots the schematic for the encoder

IMPLICIT NONE
INTEGER T,N,K,

of total and code bits
2 FDBK(10),
3 I,J1,J2,K1,K2,K3,K4,K5

Set feedback path for encoder

DO I=1,10
  FDBK(I)=0
ENDDO
IF ((N.EQ.7).AND.(K.EQ.4)) THEN
  FDBK(1)=1
ENDIF
IF ((N.EQ.7).AND.(K.EQ.6)) THEN
  WRITE (6,10) 27,76,86,48
  ! Set dialog
  WRITE (6,10) 27,83,79,62
  #14
  J1=(N-K+1)*200+320
  J2=760
  CALL COOR(0,J1,J2,K1,K2,K3,K4,K5)
  WRITE (6,10) 27,82,87,K1,K2,K3,K4,K5
  (J1,J2)
  WRITE (6,10) 27,77,76,62
  to #14, black
  WRITE (6,10) 27,77,84,60
  color to #12, purple
  J1=J1/2 - 125
  CALL COOR(1,J1,710,K1,K2,K3,K4,K5)
  CALL TEXT(21,'SCHEMATIC FOR ENCODER')
  WRITE (6,10) 27,77,84,50
  color to #2, red
  WRITE (6,10) 27,77,84,62
  color to #14, black
  CALL COOR(1,20,154,K1,K2,K3,K4,K5)
  CALL TEXT(5,'INPUT')
  CALL COOR(1,90,154,K1,K2,K3,K4,K5)
  CALL COOR(2,106,170,K1,K2,K3,K4,K5)
  CALL COOR(2,90,186,K1,K2,K3,K4,K5)
  CALL COOR(1,106,170,K1,K2,K3,K4,K5)
  CALL COOR(2,200,170,K1,K2,K3,K4,K5)
  CALL COOR(1,200,175,K1,K2,K3,K4,K5)
  CALL COOR(2,250,220,K1,K2,K3,K4,K5)
  CALL COOR(2,300,220,K1,K2,K3,K4,K5)
  CALL COOR(1,225,205,K1,K2,K3,K4,K5)
  CALL COOR(2,225,235,K1,K2,K3,K4,K5)
CALL COOR(2, 215, 225, K1, K2, K3, K4, K5) ! Draw
CALL COOR(1, 225, 235, K1, K2, K3, K4, K5) ! Move
CALL COOR(2, 235, 225, K1, K2, K3, K4, K5) ! Draw
CALL COOR(1, 195, 132, K1, K2, K3, K4, K5) ! Move
CALL TEXT(1, 'A') ! GRAPHIC TEXT
CALL COOR(1, 195, 276, K1, K2, K3, K4, K5) ! Move
CALL TEXT(1, 'B') ! GRAPHIC TEXT
CALL COOR(0, 300, 170, K1, K2, K3, K4, K5) ! Return
WRITE (6, 10) 27, 83, 88, 53, K1, K2, K3, K4, K5 ! Set segment #5
position (J1, J2)
WRITE (6, 10) 27, 76, 75, 53 ! Include copy of

DU I=1, N-K
J1=I*200+300
J2=100
CALL COOR(0, J1, J2, K1, K2, K3, K4, K5) ! Return
WRITE (6, 10) 27, 83, 88, 54, K1, K2, K3, K4, K5 ! Set segment #6
position (J1, J2)
WRITE (6, 10) 27, 76, 75, 54 ! Include copy of

J1=J1+188
J2=J2+130
CALL COOR(1, J1, J2, K1, K2, K3, K4, K5) ! Move
CALL TEXT(1, 'S') ! GRAPHIC TEXT
J1=J1+12
CALL COOR(1, J1, J2, K1, K2, K3, K4, K5) ! Move
WRITE (5, 20) 27, 76, 84, 49, I ! GRAPHIC TEXT
ENDDO
CALL COOR(1, 20, 74, K1, K2, K3, K4, K5) ! Move
CALL TEXT(5, 'CLOCK') ! GRAPHIC TEXT
CALL COOR(1, 90, 74, K1, K2, K3, K4, K5) ! Move
CALL COOR(2, 106, 90, K1, K2, K3, K4, K5) ! Draw
CALL COOR(2, 90, 106, K1, K2, K3, K4, K5) ! Draw
CALL COOR(1, 106, 90, K1, K2, K3, K4, K5) ! Move
DU I=N-K, 1, -1
J1=I*200+320
J2=90
CALL COOR(2, J1, J2, K1, K2, K3, K4, K5) ! Draw
J2=J2+10
CALL COOR(2, J1, J2, K1, K2, K3, K4, K5) ! Draw
J1=J1-200
CALL COOR(1, J1, J2, K1, K2, K3, K4, K5) ! Move
ENDDO
CALL COOR(1, 20, 34, K1, K2, K3, K4, K5) ! Move
CALL TEXT(4, 'INIT') ! GRAPHIC TEXT
CALL COOR(1, 90, 34, K1, K2, K3, K4, K5) ! Move
CALL COOR(2, 106, 50, K1, K2, K3, K4, K5) ! Draw
CALL COOR(2, 90, 66, K1, K2, K3, K4, K5) ! Draw
CALL COOR(1, 106, 50, K1, K2, K3, K4, K5) ! Move
DU I=N-K, 1, -1
J1=I*200+380
J2=50
CALL COOR(2, J1, J2, K1, K2, K3, K4, K5) ! Draw
J2=J2+50
CALL COOR(2, J1, J2, K1, K2, K3, K4, K5) ! Draw
IF (I.EQ.1) J1=I*200+180
CALL COOR(1, J1, J2, K1, K2, K3, K4, K5) ! Move
ENDDO
J1=(N-K+1)*200+300
J2=270
CALL COOR(1, J1, J2, K1, K2, K3, K4, K5) ! Move
J2=J2+370
CALL COOR(2, J1, J2, K1, K2, K3, K4, K5) ! Draw
J1=J1-200
DO I=1, N-K-1
IF (FDBK(I).EQ.O) THEN
  J1=J1-200
ELSE
  CALL COOR(2, J1, J2, K1, K2, K3, K4, K5) ! Draw
  J2=J2+50
  CALL COOR(0, J1, J2, K1, K2, K3, K4, K5) ! Return
  WRITE (6,10) 27,83,88,56, K1, K2, K3, K4, K5 ! Set segment #8
  position (J1, J2)
WRITE (6,10) 27,76,75,56 ! Include copy of segment #8
  J2=J2-150
  CALL COOR(1, J1, J2, K1, K2, K3, K4, K5) ! Move
  CALL COOR(2, J1, 270, K1, K2, K3, K4, K5) ! Draw
  J1=J1-200
  J2=J2+50
  CALL COOR(1, J1, J2, K1, K2, K3, K4, K5) ! Move
ENDIF
ENDDO
CALL COOR(2, 300, J2, K1, K2, K3, K4, K5) ! Draw
CALL COOR(2, 300, 320, K1, K2, K3, K4, K5) ! Draw
CALL COOR(1, 300, J2, K1, K2, K3, K4, K5) ! Move
CALL COOR(2, 150, J2, K1, K2, K3, K4, K5) ! Draw
CALL COOR(2, 150, 270, K1, K2, K3, K4, K5) ! Draw
CALL COOR(2, 200, 270, K1, K2, K3, K4, K5) ! Draw
CALL COOR(1, 300, 220, K1, K2, K3, K4, K5) ! Move
CALL COOR(2, 300, 130, K1, K2, K3, K4, K5) ! Draw
CALL COOR(2, 400, 130, K1, K2, K3, K4, K5) ! Draw
CALL COOR(2, 384, 146, K1, K2, K3, K4, K5) ! Move
CALL COOR(1, 400, 130, K1, K2, K3, K4, K5) ! Draw
CALL COOR(2, 384, 114, K1, K2, K3, K4, K5) ! Move
CALL COOR(1, 410, 114, K1, K2, K3, K4, K5) ! Move
CALL TEXT(6, 'OUTPUT') ! GRAPHIC TEXT
WRITE (6,10) 27,83,67 ! End segment
CALL LIB$WAIT(5.0) ! Delete seg #14
WRITE (6,10) 27,83,75,62 ! Page
WRITE (6,10) 27,12 ! Set dialog
WRITE (6,10) 27,76,86,49
RETURN
END
SUBROUTINE BLOCKL(LCH,IC,FDBK)
C
C Plots the Latch Circuit Diagram for PSN Generation
C
IMPLICIT NONE
INTEGER I,J1,J2,K1,K2,K3,K4,K5,
LCH,IC(10),FDBK(10) ! Number of latches, IC’s, Feedback path
WRITE (6,10) 27,76,86,48 ! Set dialog invisible
FORMAT(‘+’,17A1)
WRITE (6,10) 27,83,79,62 ! Begin segment #14
#14
J1=(LCH+1)*200+20
J2=760
CALL COOR(0,J1,J2,K1,K2,K3,K4,K5) ! Return
WRITE (6,10) 27,82,88,54,K1,K2,K3,K4,K5 ! Window (0,0) -
(J1,J2)
DO I=1,LCH
J1=I*200
J2=100
CALL COOR(0,J1,J2,K1,K2,K3,K4,K5)
WRITE (6,10) 27,83,88,54,K1,K2,K3,K4,K5 ! Set segment #6
position (J1,J2)
WRITE (6,10) 27,76,75,62 ! Include copy of segment #6
ENDDO
WRITE (6,10) 27,77,76,62 ! Set line color to #14, black
WRITE (6,10) 27,77,84,50 ! Set graphtext color to #2, red
WRITE (6,10) 27,77,84,62 ! Set graphtext color to #14, black
CALL COOR(1,20,74,K1,K2,K3,K4,K5) ! Move
CALL TEXT(5,'CLOCK') ! GRAPHIC TEXT
CALL COOR(1,90,74,K1,K2,K3,K4,K5) ! Move
CALL COOR(2,106,90,K1,K2,K3,K4,K5) ! Move
CALL COOR(2,90,106,K1,K2,K3,K4,K5) ! Move
CALL COOR(1,106,90,K1,K2,K3,K4,K5) ! Move
DO I=LCH,1,-1
J1=I*200+20
J2=90
CALL COOR(2,J1,J2,K1,K2,K3,K4,K5) ! Draw
J2=J2+10
CALL COOR(2,J1,J2,K1,K2,K3,K4,K5) ! Draw
J1=J1-200
CALL COOR(1,J1,J2,K1,K2,K3,K4,K5) ! Move
ENDDO
CALL COOR(1,20,34,K1,K2,K3,K4,K5) ! Move
CALL TEXT(4,’INIT’) ! GRAPHIC TEXT
CALL COOR(1,90,34,K1,K2,K3,K4,K5) ! Move
CALL COOR(2,106,50,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,90,66,K1,K2,K3,K4,K5) ! Draw
CALL COOR(1, 106, 50, K1, K2, K3, K4, K5) ! Move
DO I=LCH, 1, -1
   J1=I*200+80+60*IC(I)
   J2=50
   CALL COOR(2, J1, J2, K1, K2, K3, K4, K5) ! Draw
   J2=J2+50
   CALL COOR(2, J1, J2, K1, K2, K3, K4, K5)
   IF (I.NE.1) THEN
      J1=I*200-120+60*IC(I-1)
      CALL COOR(1, J1, J2, K1, K2, K3, K4, K5) ! Move
   ENDIF
   CALL COOR(1, J1, J2, K1, K2, K3, K4, K5)
ENDDO
J1=(LCH+1)*200
J2=270
CALL COOR(1, J1, J2, K1, K2, K3, K4, K5) ! Move
J2=J2+370
CALL COOR(2, J1, J2, K1, K2, K3, K4, K5) ! Draw
J1=J1-200
DO I=1, LCH-1
   IF (FDBK(I).EQ.0) THEN
      J1=J1-200
   ELSE
      CALL COOR(2, J1, J2, K1, K2, K3, K4, K5) ! Draw
      J2=J2+50
      CALL COOR(0, J1, J2, K1, K2, K3, K4, K5)
      WRITE (6, 10) 27, 83, 88, 56, K1, K2, K3, K4, K5
      position (J1, J2)
      WRITE (6, 10) 27, 76, 75, 56 ! Include copy of segment #8
      J2=J2-150
      CALL COOR(1, J1, J2, K1, K2, K3, K4, K5)
      CALL COOR(2, J1, 270, K1, K2, K3, K4, K5)
      J1=J1-200
      J2=J2+50
      CALL COOR(1, J1, J2, K1, K2, K3, K4, K5) ! Move
   ENDIF
ENDDO
CALL COOR(2, J1, J2, K1, K2, K3, K4, K5) ! Draw
CALL COOR(1, 200, 270, K1, K2, K3, K4, K5) ! Move
CALL COOR(2, 200, 730, K1, K2, K3, K4, K5) ! Draw
CALL COOR(2, 400, 730, K1, K2, K3, K4, K5) ! Draw
CALL COOR(1, 384, 714, K1, K2, K3, K4, K5) ! Move
CALL COOR(2, 400, 730, K1, K2, K3, K4, K5) ! Draw
CALL COOR(2, 384, 746, K1, K2, K3, K4, K5) ! Draw
CALL COOR(1, 410, 714, K1, K2, K3, K4, K5) ! Move
CALL TEXT(7, 'PSN OUT') ! GRAPHIC TEXT
WRITE (6, 10) 27, 83, 67 ! End segment
CALL LIB$WAIT(5.0) ! Delete seg #14
WRITE (6, 10) 27, 83, 75, 62 ! Page
WRITE (6, 10) 27, 12 ! Set dialog
WRITE (6, 10) 27, 76, 86, 49
visible
RETURN
END
SUBROUTINE BLOCKS

Plots the System Block Diagrams

IMPLICIT NONE

WRITE (6,10) 27,76,86,48  ! SET
DIALOG INVIS
10 FORMAT ('$',15A1)
WRITE (6,10) 27,77,84,50  ! Set
GRAPHTEXT color index #2, red
WRITE (6,10) 27,77,84,62  ! Set
GRAPHTEXT color index #14, black

C Transmitter Block Diagram

WRITE (6,10) 27,82,87,39,96,122,39,90,47,98,116,54,67  ! Set
window (1000,1000) to (2830,2000)
WRITE (6,10) 27,83,88,58,39,96,122,39,90  ! Set seg
#10 position (1000,1000)
WRITE (6,10) 27,83,88,58,49  ! Set seg
#10 visible
CALL TXTXMT
WRITE (6,10) 27,83,86,58,48  ! Set seg
#10 invisible
WRITE (6,10) 27,12  ! PAGE,
Erase Graphics

C Receiver Block Diagram

WRITE (6,10) 27,82,87,39,96,122,39,90,47,98,116,54,67  ! Set
window (1000,1000) to (2830,2000)
WRITE (6,10) 27,83,88,60,39,96,122,39,90  ! Set seg
#12 position (1000,1000)
WRITE (6,10) 27,83,88,60,49  ! Set seg
#12 visible
CALL TXTREC
WRITE (6,10) 27,83,86,60,48  ! Set seg
#12 invisible
WRITE (6,10) 27,12  ! PAGE,
Erase Graphics

WRITE (6,10) 27,76,86,49  ! Set
dialog visible

RETURN
END
SUBROUTINE COMP_SIG(SIGNAL)

C Determines the function values of all signals required to plot
the desired signal

C Subroutines called : PSN GEN
                        ENCODE_FUNCTION

IMPLICIT NONE
INTEGER SIGNAL, CODE, IC(10),
  2 RESOLUT, RES_SIN, MAX_PTS
REAL DX, Y1(1023), RN2
INCLUDE 'COMMON.VAR'

RESOLUT = 10                ! Number
of points / pulse
RES_SIN = 2                  ! Number
of points for sin wave
MAX_PTS = 5000               ! Maximum
points in a data file

IF (SIGNAL.GT.VALID_SIG) THEN
  GOTO (100, 200, 300, 400, 500, 600, 700,
        2 800, 900, 1000, 1100, 1200, 1300,
        3 1400, 1500, 2000), (VALID_SIG+1) ENDIF
GOTO 2000

Data waveform

OPEN (UNIT=1, FILE='SIG01.DAT', STATUS='NEW')
DX = 1.0 / DATA_RATE
WRITE (1,110) DX

110 FORMAT(X,1PE13.6,14X,' Delta X')
CALL PSN_GEN(DATA_POLY_ORD,DATA_IC,Y1)
K2 = 2**DATA_POLY_ORD - 1
WRITE (1,111) K2

111 FORMAT(X,18)
DO K1=1,K2
  WRITE (1,112) Y1(K1)
112  FORMAT(1PE13.6)
ENDDO
DO K1=1,15
  WRITE (1,112) Y1(K1)
ENDDO
CLOSE(UNIT=1)
VALID_SIG=1
GOTO 1

Encoded waveform

OPEN (UNIT=1, FILE='SIG02.DAT', STATUS='NEW')
OPEN (UNIT=2, FILE='SIG01.DAT', STATUS='OLD')
READ (2,210) DX

210 FORMAT(F16.0)
DX = (1.0 / DATA_RATE) * ENCODE_DATA / ENCODE_TOTAL
WRITE (1,110) DX
CODE = ENCODE_TOTAL - ENCODE_DATA
RN = (2**DATA_POLY_ORD-1) / ENCODE_DATA
J = INT(RN+1) * ENCODE_TOTAL
WRITE (1,111) J
READ (2,111) N
DO K1=0,INT(RN)
  DO K2=1,ENCODE_DATA
    READ (2,210) Y1(K2)
    IC(K2) = INT(Y1(K2))
  ENDDO
  IF (CODE.EQ.0) GOTO 250
  CALL ENCODE_FUNCTION(ENCODE_DATA,CODE,IC)
  DO K2=1,CODE
    Y1(ENCODE_DATA+K2) = IC(K2)
  ENDDO
  DO K2=1,ENCODE_TOTAL
    WRITE (1,112) Y1(K2)
  ENDDO
250 ENDDO
CLOSE(UNIT=2)
CLOSE(UNIT=1)
VALID_SIG=2
GOTO 1

C Spreading waveform
C
300 OPEN (UNIT=1,FILE='SIG03.DAT',STATUS='NEW')
DX = (1.0 / DATA_RATE) * ENCODE_DATA / ENCODE_TOTAL * 2
   (1.0 / (SPRD_GAIN * (2**SPRD_POLY_ORD-1)))
WRITE (1,110) DX
CALL PSN_GEN(SPRD_POLY_ORD,SPRD_IC,Y1)
K2 = 2**SPRD_POLY_ORD - 1
WRITE (1,111) K2
DO K1=1,K2
  WRITE (1,112) Y1(K1)
ENDDO
DO K1=1,15
  WRITE (1,112) Y1(K1)
ENDDO
CLOSE(UNIT=1)
VALID_SIG=3
GOTO 1

C Spread data waveform
C
400 OPEN (UNIT=1,FILE='SIG04.DAT',STATUS='NEW')
OPEN (UNIT=2,FILE='SIG03.DAT',STATUS='OLD')
OPEN (UNIT=3,FILE='SIG02.DAT',STATUS='OLD')
READ (2,210) DX
READ (3,210) DX
DX = (1.0 / DATA_RATE) * ENCODE_DATA / ENCODE_TOTAL * 2
   (1.0 / (SPRD_GAIN * (2**SPRD_POLY_ORD-1)))
WRITE (1,110) DX
READ (2,111) N
READ (3,111) J
WRITE (1,111) N*J*SPRD_GAIN
CLOSE(UNIT=2)
DO K1=1,J
  READ (3,210) RN
  OPEN (UNIT=2,FILE='SIG03.DAT',TYPE='OLD')
  READ (2,210) DX
  READ (2,111) N
  DO K2=1,N
    READ (2,210) Y1(K2)
  ENDDO
  CLOSE(UNIT=2)
  DO K2=1,SPRD_GAIN
    DO K3=1,N
      RN2 = MOD(INT(RN+Y1(K3)),2)
      WRITE (1,112) RN2
    ENDDO
  ENDDO
ENDDO
CLOSE(UNIT=3)
CLOSE(UNIT=1)
VALID SIG=4
GOTO 1

C Shifted waveform
C
OPEN (UNIT=1,FILE='SIG05.DAT',STATUS='NEW')
OPEN (UNIT=2,FILE='SIG04.DAT',STATUS='OLD')
READ (2,210) DX
DX = (1.0 / DATA RATE) * ENCODE_DATA / ENCODE_TOTAL * 2 (1.0 / (SPRD_GAIN * (2**SPRD_POLY_ORD-1)))
WRITE (1,110) DX
READ (2,111) J
WRITE (1,111) J
DO K1=1,J
  READ (2,210) RN
  RN = RN * LEVEL AMP + LEVEL_DISP
  WRITE (1,112) RN
ENDDO
CLOSE(UNIT=2)
CLOSE(UNIT=1)
VALID SIG=5
GOTO 1

C Pulse shaped waveform
C
OPEN (UNIT=1,FILE='SIG06.DAT',STATUS='NEW')
OPEN (UNIT=2,FILE='SIG05.DAT',STATUS='OLD')
READ (2,210) DX
DX = (1.0 / DATA RATE) * ENCODE_DATA / ENCODE_TOTAL * 2 (1.0 / (SPRD_GAIN * (2**SPRD_POLY_ORD-1))) / RESOLUT
WRITE (1,110) DX
READ (2,111) J
J = MIN(J,MAX_PTS/RESOLUT)
WRITE (1,111) RESOLUT*J
DO K1=1,J
   READ (2,210) RN
   DO K2=1,RESOLUT
      Y1(1) = RN * PULSE VAL(K2)
      WRITE (1,112) Y1(1)
   ENDDO
ENDDO
CLOSE(UNIT=2)
CLOSE(UNIT=1)
VALID SIG=6
GOTO 1

C Modulated waveform

700 OPEN (UNIT=1,FILE='SIG07.DAT',STATUS='NEW')
OPEN (UNIT=2,FILE='SIG06.DAT',STATUS='OLD')
READ (2,210) DX
DX = (1.0 / DATA_RATE) * ENCODE DATA / ENCODE TOTAL /
2 SPRD_GAIN / (2**SPRD_POLY_ORD-1) / RESOLUT / RES_SIN
WRITE (1,110) DX
READ (2,111) J
J = MIN(J,MAX PTS/RES_SIN)
WRITE (1,111) -RES_SIN;J
DO K1=1,J
   READ (2,210) RN
   DO K2=1,RES_SIN
      RN = RN * CARR AMP * SIN(6.2831852*CARR_FREQ*Y1(1)*DX)
   END.DO
ENDDO
ENDDO
CLOSE(UNIT=2)
CLOSE(UNIT=1)
VALID SIG=7
GOTO 1

C Filtered waveform

800 OPEN (UNIT=1,FILE='SIG08.DAT',STATUS='NEW')
OPEN (UNIT=2,FILE='SIG07.DAT',STATUS='OLD')
READ (2,210) DX
DX = (1.0 / DATA_RATE) * ENCODE DATA / ENCODE TOTAL /
2 SPRD_GAIN / (2**SPRD_POLY_ORD-1) / RESOLUT / RES_SIN
WRITE (1,110) DX
READ (2,111) J
J = MIN(J,MAX PTS)
WRITE (1,111) -J
DO K1=1,J/RESOLUT
   READ (2,210) RN
   RN = RN * FLTR VAL(K2)
   WRITE (1,112) RN
END.DO
ENDDO
CLOSE(UNIT=2)
CLOSE(UNIT=1)
VALID_SIG=8
GOTO 1

C Noise and jamming waveform
C
C 900 OPEN (UNIT=1,FILE='SIG09.DAT',STATUS='NEW')
OPEN (UNIT=2,FILE='SIG08.DAT',STATUS='OLD')
READ (2,210) DX
DX = (1.0 / DATA_RATE) * ENCODE_DATA / ENCODE_TOTAL / 2 SPRD_GAIN / (2**SPRD_POLY_ORD-1) / RESOLUT / RES_SIN
WRITE (1,110) DX
READ (2,111) J
J = MIN(J,MAX_PTS)
CLOSE(UNIT=2)
WRITE (1,111) J
DO K1=1,J
   RN = NOISE_AMP + JAM_AMP
   WRITE (1,112) RN
ENDDO
CLOSE(UNIT=1)
VALID_SIG=9
GOTO 1

C Received waveform
C
C 1000 OPEN (UNIT=1,FILE='SIG10.DAT',STATUS='NEW')
OPEN (UNIT=2,FILE='SIG09.DAT',STATUS='OLD')
OPEN (UNIT=3,FILE='SIG08.DAT',STATUS='OLD')
READ (2,210) DX
READ (3,210) DX
DX = (1.0 / DATA_RATE) * ENCODE_DATA / ENCODE_TOTAL / 2 SPRD_GAIN / (2**SPRD_POLY_ORD-1) / RESOLUT / RES_SIN
WRITE (1,110) DX
READ (2,111) J
READ (3,111) J
J = MIN(J,MAX_PTS)
WRITE (1,111) J
DO K1=1,J
   READ (2,210) RN
   READ (3,210) Y1(1)
   RN = RN + Y1(1)
   WRITE (1,112) RN
ENDDO
CLOSE(UNIT=3)
CLOSE(UNIT=2)
CLOSE(UNIT=1)
VALID_SIG=10
GOTO 1

C Demodulated waveform
C
C 1100 OPEN (UNIT=1,FILE='SIG11.DAT',STATUS='NEW')
OPEN (UNIT=2,FILE='SIG10.DAT',STATUS='OLD')
READ (2,210) DX
DX = (1.0 / DATA_RATE) * ENCODE_DATA / ENCODE_TOTAL /
2 SPRD_GAIN / (2**SPRD_POLY_ORD-1) / RESOLUT / RES_SIN
WRITE (1,110) DX
READ (2,111) J
J = MIN(J,MAX_PTS)
WRITE (1,111) J
DO K1=1,J
   READ (2,210) RN
   Y1(1) = 1.0 * K1
   IF (ABS(SIN(6.2831852*OSC_FREQ*Y1(1)*DX)).GT.(0.01)) THEN
      RN = RN * OSC_AMP / SIN(6.2831852*OSC_FREQ*Y1(1)*DX)
   ENDIF
   WRITE (1,112) RN
ENDDO
CLOSE(UNIT=2)
CLOSE(UNIT=1)
VALID_SIG=11
GOTO 1

C Compared waveform
C
1200 OPEN (UNIT=1,FILE='SIG12.DAT',STATUS='NEW')
OPEN (UNIT=2,FILE='SIG11.DAT',STATUS='OLD')
READ (2,210) DX
DX = (1.0 / DATA_RATE) * ENCODE_DATA / ENCODE_TOTAL * 
2 (1.0 / (SPRD_GAIN * (2**SPRD_POLY_ORD-1)))
WRITE (1,110) DX
READ (2,111) J
J = MIN(J/RESOLUT/RES_SIN,MAX_PTS)
WRITE (1,111) J
DO K1=1,J
   RN = 0.0
   DO K2=1,RESOLUT*RES_SIN
      READ (2,210) Y1(I)
      RN = RN + Y1(I)
   ENDDO
   RN = RN / RESOLUT / RES_SIN
   IF (RN.LT.COMPR_LEV) THEN
      RN = 0.0
   ELSE
      RN = 1.0
   ENDIF
   WRITE (1,112) RN
ENDDO
CLOSE(UNIT=2)
CLOSE(UNIT=1)
VALID_SIG=12
GOTO 1

C Synchronized correlator waveform
C
1300 OPEN (UNIT=1,FILE='SIG13.DAT',STATUS='NEW')
OPEN (UNIT=2,FILE='SIG12.DAT',STATUS='OLD')
READ (2,210) DX
DX = (1.0 / DATA_RATE) * ENCODE_DATA / ENCODE_TOTAL *
2 (1.0 / (SPRD_GAIN * (2**SPRD_POLY_ORD-1)))
WRITE (1,110) DX
READ (2,111) J
J = MIN(J,MAX_PTS)
WRITE (1,111) J
DO K1=1,SPRD_POLY_ORD
  READ (2,210) Y1(K1)
ENDDO
RN = -1.0
DO K1=1,J-SPRD_POLY_ORD-1
  READ (2,210) Y1(SPRD_POLY_ORD+1) IN=O
  DO K2=1,SPRD_POLY_ORD+1
    IF (Y1(K2) .NE. CORR_ACQIS(K2)) IN=1
  ENDDO
  IF (IN.EQ.0) GOTO 1310 ! Acquired lock-
in
  WRITE (1,112) RN
  DO K2=1,SPRD_POLY_ORD
    Y1(K2) = Y1(K2+1)
  ENDDO
ENDDO
1310 CALL PSN GEN(SPRD_POLY_ORD,CORR_IC,Y1)
IN = K1 - 1
1320 DO K1=1,2**SPRD_POLY_ORD-1
  IN = IN + 1
  WRITE (1,112) Y1(K1)
ENDDO
IF (IN.LT.J) GOTO 1320
CLOSE(UNIT=2)
CLOSE(UNIT=1)
VALID_SIG=13
GOTO 1
C
C Link data out waveform
C
1400 OPEN (UNIT=1,FILE='SIG14.DAT',STATUS='NEW')
OPEN (UNIT=2,FILE='SIG13.DAT',STATUS='OLD')
OPEN (UNIT=3,FILE='SIG12.DAT',STATUS='OLD')
READ (2,210) DX
READ (3,210) DX
DX = (1.0 / DATA_RATE) * ENCODE_DATA / ENCODE_TOTAL
WRITE (1,110) DX
READ (2,111) J
READ (3,111) J
IN = SPRD_GAIN * (2**SPRD_POLY_ORD-1)
J = J / IN
J = MIN(J,MAX_PTS)
WRITE (1,111) J
DO K1=1,J
  RN = 0.0
  DO K2=1,IN
    READ (2,210) Y1(1)
    READ (3,210) Y1(2)
IF (Y1(l).EQ.-1.0) THEN
RN = RN + 0.5
ELSE
RN = RN + 1.0 * MOD(INT(Y1(l)+Y1(2)),2)
ENDIF
ENDDO
IF (RN/IN .LT. 0.5) THEN
RN = 0.0
ELSE
RN = 1.0
ENDIF
WRITE (1,112) RN
ENDDO
CLOSE(UNIT=3)
CLOSE(UNIT=2)
CLOSE(UNIT=1)
VALID_SIG=14
GOTO i
Decoded waveform
C
C 1500 OPEN (UNIT=1,FILE='SIG15.DAT',STATUS='NEW')
OPEN (UNIT=2,FILE='SIG14.DAT',STATUS='OLD')
READ (2,210) DX
DX = 1.0 / DATA RATE
WRITE (1,110) DX
READ (2,111) J
J = J / DECODE_TOTAL
WRITE (1,111) DECODE_DATA+J
DO Kl=1,J
DO K2=1,DECODE_DATA
READ (2,210) RN
WRITE (1,112) RN
ENDDO
DO K2=DECODE_DATA+1,DECODE_TOTAL
READ (2,210) RN
ENDDO
ENDDO
CLOSE(UNIT=2)
CLOSE(UNIT=1)
VALID_SIG=15
GOTO i
C
2000 RETURN
END

SUBROUTINE PSN_GEN(NUM,IC,SIG)

This subroutine computes the PSN waveform by the output of the shift registers
C
IMPLICIT NONE
INTEGER K1,K2,NUM,IC(10),IC2(10)
REAL SIG(1039)
DO K1=1,NUM
  IC2(K1)=IC(K1)
ENDDO

DO K1=1,2**NUM-1
  IF (NUM.EQ.2) SIG(K1) = IC2(1)+IC2(2)+0.0
  IF (NUM.EQ.3) SIG(K1) = IC2(2)+IC2(3)+0.0
  IF (NUM.EQ.4) SIG(K1) = IC2(3)+IC2(4)+0.0
  IF (NUM.EQ.5) SIG(K1) = IC2(3)+IC2(5)+0.0
  IF (NUM.EQ.6) SIG(K1) = IC2(5)+IC2(6)+0.0
  IF (NUM.EQ.7) SIG(K1) = IC2(4)+IC2(7)+0.0
  IF (NUM.EQ.8) SIG(K1) = IC2(4)+IC2(5)+IC2(6)+IC2(8)+0.0
  IF (NUM.EQ.9) SIG(K1) = IC2(5)+IC2(9)+0.0
  SIG(K1) = MOD(INT(SIG(K1)),2)+0.0
  DO K2=NUM,2,-1
    IC2(K2)=IC2(K2-1)
  ENDDO
  IC2(1)=INT(SIG(K1))
ENDDO

C Extend SIG array to hold extra points

IF (NUM.EQ.2) THEN
  SIG(4)=SIG(1)
  SIG(5)=SIG(2)
  SIG(6)=SIG(3)
  SIG(7)=SIG(1)
  SIG(8)=SIG(2)
  SIG(9)=SIG(3)
  SIG(10)=SIG(1)
  SIG(11)=SIG(2)
  SIG(12)=SIG(3)
  SIG(13)=SIG(1)
  SIG(14)=SIG(2)
  SIG(15)=SIG(3)
ENDIF

IF (NUM.EQ.3) THEN
  SIG(8)=SIG(1)
  SIG(9)=SIG(2)
  SIG(10)=SIG(3)
  SIG(11)=SIG(4)
  SIG(12)=SIG(5)
  SIG(13)=SIG(6)
  SIG(14)=SIG(7)
  SIG(15)=SIG(1)
ENDIF

RETURN
END

SUBROUTINE ENCODE_FUNCTION (N,K,IC)
This subroutine computes the code bit values for an encoder
Encoder is limited to 10 data and 10 code bits

IMPLICIT NONE
INTEGER N,K,IC(10),I,J,NEW,IC2(10)

DO I=1,K
   IC2(I)=0
ENDDO

DO I=1,N ! Set-up encoder
   IF ((N+K.EQ.7).AND.(N.EQ.4)) NEW = IC(I) + IC2(2) + IC2(3)
   IF ((N+K.EQ.7).AND.(N.EQ.6)) NEW = IC(I) + IC2(2)
   DO J=K,2,-1
      IC2(J)=IC2(J-1)
   ENDDO
   IC2(1)=MOD(NEW,2)
ENDDO

DO I=1,K ! Get code bits
   IF ((N+K.EQ.7).AND.(N.EQ.4)) NEW = IC2(2) + IC2(3)
   IF ((N+K.EQ.7).AND.(N.EQ.6)) NEW = IC2(2)
   DO J=K,2,-1
      IC2(J)=IC2(J-1)
   ENDDO
   IC2(1)=0
   IC(I)=MOD(NEW,2)
ENDDO

RETURN
END
SUBROUTINE COOR(FCT,X,Y,I1,I2,I3,I4,I5)

FCT=0  Returns the TEK xy-coordinates for actual xy-position
    =1  Moves to (X,Y)
    =2  Draws from current position to (X,Y)

IMPLICIT NONE
INTEGER FCT, X, Y,
Function desired
2    position
3    I1,I2,I3,I4,I5
coordinates

I1=32+IBITS(Y,7,5)
I2=96+4*IBITS(Y,0,2)+IBITS(X,0,2)
I3=96+IBITS(Y,2,5)
I4=32+IBITS(X,7,5)
I5=64+IBITS(X,2,5)

IF (FCT.EQ.1) WRITE (6,10) 27,76,70,I1,I2,I3,I4,I5 ! Move to (X,Y)
IF (FCT.EQ.2) WRITE (6,10) 27,76,71,I1,I2,I3,I4,I5 ! Draw to (X,Y)
10 FORMAT(’+’,8A1)

RETURN
END
SUBROUTINE ENCODE(T,N,K)

This subroutine inquires and checks validity of the encoder parameters

Subroutines called: BLOCKE

IMPLICIT NONE
INTEGER I,T,N,K,FDBK(10)

WRITE (6,10) 27,76,90
10 FORMAT('$',3A1)
WRITE (6,20)
20 FORMAT(/,'$',10X,'ENCODER CHOICE',/
   1 ,/,'$',6X,'1 - None',/
   2 ,/,'$',6X,'2 - Cyclic code with error detection',/
   3 ,/,'$',6X,'3 - Cyclic code with error correction',/
   4 ,/,'$',6X,'4 - Convolutional encoding',/
   5 ,/,'$',X,'Enter the digit corresponding to the',/
   6 ,/,'$',X,'desired choice : ') READ (5,25,ERR=5) I
25 FORMAT(1I3)
26 IF ((I.GT.4).OR.(I.LT.1)) GOTO 5
T = I

IF (T.EQ.1) THEN
   N = 1
   K = 1
ENDIF
IF ((T.EQ.2).OR.(T.EQ.3)) THEN
   WRITE (6,30)
30 FORMAT(/,' Valid pairing for (n,k) cyclic codes and their encoder',/
   1 ,/,' parameters :',/,'/',
   2 ,/,' (7,6) g(X) = X + 1')
   IF (T.EQ.3) THEN
      WRITE (6,31)
31 FORMAT(/,' (7,4) g(X) = X^3 + X^2 + 1')
   ENDIF
40 WRITE (6,41) N,K
41 FORMAT(/,' The current (n,k) encoder values are
   1 ,'/','I2,','I2,')
WRITE (6,50) N
50 FORMAT('$ Enter the new number of encoder output bits',/
   1 ,/,' if desired <',',I2,' : ')
   READ (5,25,ERR=40) I
   IF (I.GT.1) N=I
60 WRITE (6,65) K
65 FORMAT('$ Enter the new number of encoder data bits',/
   1 ,/,' if desired <',',I2,' : ')
   READ (5,25,ERR=60) I
   IF ((I.GT.0).AND.(I.LE.N)) K=I
   IF (K.GT.N) THEN
      WRITE (6,70) N
70 FORMAT(' Illegal value of k for n = ',I2)
GOTO 40
ENDIF
IF (((N.EQ.7).AND.(K.EQ.4)).OR.
2 ((N.EQ.7).AND.(K.EQ.6))) THEN
   CALL BLOCKE(T,N,K,FDBK)
ELSE
   WRITE (6,10) 27,76,90
   WRITE (6,80) N,K
   FORMAT('$( ',I2,' ,',I2,') is not a valid pairing. ',/)
   GOTO 40
ENDIF
ENDIF
IF (T.EQ.4) THEN
   WRITE (6,90)
   FORMAT(/,' Valid convolutional encoding parameters :',/,,
2 ' CHANGE TYPE')
   READ (5,25,ERR=5) I
   GOTO 26
ENDIF
C
RETURN
END
SUBROUTINE IODATA(J,IFILE)

IMPLICIT NONE
INCLUDE 'COMMON.VAR'

C Type of I/O : J = 0 then READ
C = 1 then WRITE

IF (J.EQ.0) THEN
    OPEN(UNIT=1,FILE=IFILE,STATUS='OLD',READONLY)
    READ (1,10) DATA_POLY_ORD
    FORMAT(10I3)
    READ (1,10) (DATA_IC(N),N=1,DATA_POLY_ORD)
    READ (1,20) DATA_RATE
    READ (1,10) ENCODE_TOTAL,ENCODE_DATA
    READ (1,10) SPRD_POLY_ORD
    READ (1,10) (SPRD_IC(N),N=1,SPRD_POLY_ORD)
    FORMAT(2F12.0)
    READ (1,10) SPRD_BLOCK
    SPRD_GAIN = SPRD_BLOCK * (2**SPRD_POLY_ORD-1)
    READ (1,10) LEVEL AMP,LEVEL_DISP
    READ (1,10) PULSE_ID
    READ (1,30) (PULSE_VAL(N),N=1,10)
    FORMAT(10F4.0)
    READ (1,20) CARR_VOL
    READ (1,10) CARR_CYCLE
    CARR_FREQ = DATA_RATE * ENCODE_TOTAL / ENCODE_DATA / 4 * 
2 * SPRD_GAIN * (2**SPRD_POLY_ORD-1) * CARR_CYCLE
    READ (1,10) FLTR_ID
    READ (1,30) (FLTR_VAL(N),N=1,10)
    READ (1,20) NOISE_VOL,JAM_VOL
    READ (1,20) OSC_VOL
    READ (1,20) COMP_LVL
    READ (1,10) (CORR_ACQIS(N),N=1,SPRD_POLY_ORD+1)
    DO N=1,SPRD_POLY_ORD
        CORR_IC(N) = CORR_ACQIS(N)
    ENDDO
    CLOSE(UNIT=1)
    VALID_SIG=0
ENDIF

IF (J.EQ.1) THEN
    OPEN(UNIT=1,FILE=IFILE,STATUS='NEW')
    WRITE (1,10) DATA_POLY_ORD
    WRITE (1,10) (DATA_IC(N),N=1,DATA_POLY_ORD)
    WRITE (1,20) DATA_RATE
    WRITE (1,10) ENCODE_TOTAL,ENCODE_DATA
    WRITE (1,10) SPRD_POLY_ORD
    WRITE (1,10) (SPRD_IC(N),N=1,SPRD_POLY_ORD)
    WRITE (1,10) SPRD_BLOCK
    WRITE (1,10) LEVEL AMP,LEVEL_DISP
    WRITE (1,10) PULSE_ID
    WRITE (1,30) (PULSE_VAL(N),N=1,10)
    WRITE (1,20) CARR_VOL
    WRITE (1,10) CARR_CYCLE
    WRITE (1,10) FLTR_ID
WRITE (1,30) (FLTR VAL(N),N=1,10)
WRITE (1,20) NOISE AMP, JAM AMP
WRITE (1,20) OSC AMP
WRITE (1,20) COMPR LEV
WRITE (1,10) (CORR ACQIS(N),N=1,SPRD_POLY_ORD+1)
CLOSE(UNIT=1)
ENDIF

C
RETURN
END
SUBROUTINE PLOT

Subroutine called : COMP SIG
DUAL: [BELKEDID.PATZ.PLOT] PLOT/LIB

IMPLICIT NONE
BYTE IOUT(30)
INTEGER I, J, NUM, SIGNAL
REAL SX, DX, Y(600),
2 XMIN, XMAX, YMIN, YMAX
CHARACTER*9 NAME(15)
NAME(1) = 'SIG01.DAT'
NAME(2) = 'SIG02.DAT'
NAME(3) = 'SIG03.DAT'
NAME(4) = 'SIG04.DAT'
NAME(5) = 'SIG05.DAT'
NAME(6) = 'SIG06.DAT'
NAME(7) = 'SIG07.DAT'
NAME(8) = 'SIG08.DAT'
NAME(9) = 'SIG09.DAT'
NAME(10) = 'SIG10.DAT'
NAME(11) = 'SIG11.DAT'
NAME(12) = 'SIG12.DAT'
NAME(13) = 'SIG13.DAT'
NAME(14) = 'SIG14.DAT'
NAME(15) = 'SIG15.DAT'

WRITE (6,10) 27, 76, 90 CLEAR DIALOG SCROLL
10 FORMAT(' ', 10A1)
WRITE (6,15)
15 FORMAT(2, 'S', 10X, 'SIGNAL PLOT/READ SELECTION MENU', /
3 /, 'S', 5X, '1 - Data waveform', /
4 /, 'S', 5X, '2 - Output of encoder', /
5 /, 'S', 5X, '3 - Spreading waveform', /
6 /, 'S', 5X, '4 - XOR sum of signals 2 and 3', /
7 /, 'S', 5X, '5 - Output of shifter', /
8 /, 'S', 5X, '6 - Output of pulse shaper', /
9 /, 'S', 5X, '7 - Signal 6 at carrier frequency', /
10 /, 'S', 5X, '8 - Transmitted signal after filter', /
11 /, 'S', 5X, '9 - Noise and/or jamming signal', /
12 /, 'S', 5X, '10 - Received signal plus interference', /
13 /, 'S', 5X, '11 - Demodulated signal', /
14 /, 'S', 5X, '12 - Compared to level', /
15 /, 'S', 5X, '13 - Correlator output', /
16 /, 'S', 5X, '14 - Link data output', /
17 /, 'S', 5X, '15 - Data out', /
18 /, 'S', X, '19 - Enter the digit corresponding', /
19 /, 'S', X, '20 - to the desired signal number : ')
READ (5, 20, ERR=5) SIGNAL
IF ((SIGNAL .GT. 15) .OR. (SIGNAL .LT. 1)) GOTO 5
WRITE (6,10) 27, 76, 90 CLEAR DIALOG SCROLL
C PLOT SIGNAL
CALL COMP_SIG(SIGNAL)
OPEN(UNIT=1, FILE=NAME(SIGNAL), STATUS='OLD')
READ (1,30) DX
30 FORMAT(F16.0)
SX=0
YMIN= 999.9
YMAX=-999.9
READ (1,31) NUM
31 FORMAT(I10)
NUM=MIN(NUM,200)
DO I=1,NUM
   READ (1,30) Y(I)
   YMIN=MIN(YMIN,Y(I))
   YMAX=MAX(YMAX,Y(I))
ENDDO
CLOSE(UNIT=1)
IF ((SIGNAL.LE.5).OR.(SIGNAL.GE.12)) THEN ! Make
digital signals square
   DX=DX/10
   NUM=MIN(10*NUM,600)
   DO I=NUM,10,-10
      Y(I)=Y(I/10)
      Y(I-1)=Y(I/10)
      Y(I-2)=Y(I/10)
      Y(I-3)=Y(I/10)
      Y(I-4)=Y(I/10)
      Y(I-5)=Y(I/10)
      Y(I-6)=Y(I/10)
      Y(I-7)=Y(I/10)
      Y(I-8)=Y(I/10)
      Y(I-9)=Y(I/10)
   ENDDO
ENDIF
YMIN=YMIN-.001
YMAX=YMAX+.001
WRITE (6,10) 27,76,86,48 ! Set dia
log invisible
WRITE (6,10) 27,82,87,63,111,127,63,95 ! SET WIN
DOW (0,0) to (4095,4095)
CALL PLTSUB(NUM,1,SX,DX,Y,0,1,0,1,1,2,1,1,XMIN,XMAX,YMIN,YMAX)
CALL TXTPLT(SIGNAL,XMIN,XMAX,YMIN,YMAX)
CALL LIB$WAIT(10.0) ! Delay 1
0 seconds
WRITE (6,10) 27,12 ! PAGE, E
rase Graphics
WRITE (6,10) 27,76,86,49 ! Set dia
log visible
RETURN
END
SUBROUTINE REC

C Subroutines called: BLOCKC
C BLOCKD
C
IMPLICIT NONE
INTEGER FEED(10)
INCLUDE 'COMMON.VAR'
C Write receiver menu
C
11 WRITE (6,10) 27,76,90
! CLEAR DIALOG SCROLL
10 FORMAT('$',3A1)
WRITE (6,12)
12 FORMAT(/,'$',1OX,'RECEIVER MENU',/)
2  ,/,'$',6X,'1 - Noise and Jamming specification'
3  ,/,'$',6X,'2 - Coherent carrier choice'
4  ,/,'$',6X,'3 - Comparator level change'
5  ,/,'$',6X,'4 - Correlator specification'
6  ,/,'$',6X,'5 - Decoder choice'
7  ,/,'$',6X,'6 - Return to Main Menu'/
8  ,/,'$',X,'Enter the digit corresponding'
9  ,/,'$',X,'to the desired function : ')
READ (5,15,ERR=11) IN
15 FORMAT(1I3)
IF ((IN.GT.6).OR.(IN.LT.1)) GOTO 11
IF (IN.EQ.1) THEN
20 WRITE (6,10) 27,76,90 CLEAR DIALOG SCROLL
WRITE (6,21) NOISE AMP
21 FORMAT(/,'$',1OX,'NOISE AND JAMMING SPECIFICATION',/)
2  ,/,'$',X,'Enter the new noise amplitude'
3  ,/,'$',X,'if desired <',1PE10.3,'> : ')
READ (5,22,ERR=2O) RN
22 FORMAT(F12.0)
IF (RN.GE.0) NOISE AMP=RN
WRITE (6,23) JAM AMP
23 FORMAT(/,'$',X,'Enter the new jamming amplitude'
3  ,/,'$',X,'if desired <',1PE10.3,'> : ')
READ (5,22,ERR=2O) RN
IF (RN.GE.0) JAM AMP=RN
ENDIF
IF (IN.EQ.2) THEN
25 WRITE (6,10) 27,76,90 CLEAR DIALOG SCROLL
WRITE (6,26) OSC FREQ,OSC AMP
26 FORMAT(/,'$',1OX,'COHERENT CARRIER CHOICE',/)
2  ,/,'$',X,'The carrier frequency is the same as in the '
3  ,/,'transmitter, freq = ',1PE10.3,/n
4  ,/,'$',X,'Enter the new carrier amplitude if desired <'
5  ,/,'1PE10.3,'> : ')
READ (5,22,ERR=3O) RN
IF (RN.NE.0) OSC AMP=RN
ENDIF
IF (IN.EQ.3) THEN
    WRITE (6,10) 27,76,90 ! CLEAR D
    IALOG SCROLL
    WRITE (6,31) COMPR LEV
    FORMAT(/,'$',10X,'COMPARITOR LEVEL CHANGE',/,$
    2    '$',X,'Enter the new comparitor level',
    3    '$', '<',1PE10.3,' >' : ')
    READ (5,22,ERR=30) COMPR LEV
ENDIF

IF (IN.EQ.4) THEN
    WRITE (6,10) 27,76,90 ! CLEAR D
    IALOG SCROLL
    WRITE (6,41) (CORR_ACQIS(J), J=1,(SPRD_POLY_ORD+1))
    FORMAT(/,'$',10X,'CORRELATOR SPECIFICATION',/,$
    2    '$',X,'Enter the new correlator acquisition',
    3    'sequence if desired <',<SPRD_POLY_ORD+l>Il,' >' : ')
    READ (5,42,ERR=40) IN
    FORMAT(I20)
    IF (IN.NE.0) THEN
        DO J=1,SPRD_POLY_ORD+1
            CORR_ACQIS(J)=IN/(10**(SPRD_POLY_ORD+1-J))
            CORR_ACQIS(J)=CORR_ACQIS(J)-10*INT(CORR_ACQIS(J)/10)
        ENDDO
    ENDIF
    CALL IC_FIND(SPRD_POLY_ORD,CORR_ACQIS,CORR_IC)
    RN= DATA RATE * ENCODE TOTAL / ENCODE_DATA *
    2 **(SPRD_POLY_ORD) * SPRD_GAIN
    CALL BLOCKC(SPRD_POLY_ORD,CORR_ACQIS,CORR_IC,RN)
    CALL FEEDBACK(SPRD_POLY_ORD,FEED)
    CALL BLOCKL(SPRD_POLY_ORD,CORR_IC,FEED)
ENDIF

IF (IN.EQ.5) THEN
    WRITE (6,10) 27,76,90 ! CLEAR D
    IALOG SCROLL
    WRITE (6,50) DECODE TOTAL,DECODE DATA
    FORMAT(/,'$',10X,'DECODER CHOICE',/,$
    2    '$',X,'The decoder values used are the same',
    3    'as those used for the encoder',/,$
    4    '$',X,'Pairing (n,k) = (',I2,',',I2,')')
    CALL LIB$WAIT(6.0)
    CALL BLOCKD(DECODE TOTAL,DECODE DATA)
ENDIF

IF (IN.NE.6) THEN
    VALID_SIG= MIN(IN+8,VALID_SIG)
    GOTO 11
ENDIF
RETURN
END

SUBROUTINE IC_FIND(ORD,SEQ,IC)

Finds the IC's for a FSN of order ORD to give SEQ as the first outputs
IMPLICIT NONE
INTEGER ORD,SEQ(11),IC(10),I

IF (ORD.EQ.2) THEN
  IC(1) = MOD((SEQ(1)+SEQ(2)),2)
  IC(2) = SEQ(2)
ENDIF

IF (ORD.EQ.6) THEN
  DO I=1,ORD
    IC(I) = SEQ(ORD-I+1)
  ENDDO
ENDIF

IF (ORD.EQ.2) I=IC(1)+IC(2)+1
IF (ORD.EQ.3) I=IC(2)+IC(3)
IF (ORD.EQ.4) I=IC(3)+IC(4)
IF (ORD.EQ.5) I=IC(3)+IC(5)
IF (ORD.EQ.6) I=IC(5)+IC(6)
IF (ORD.EQ.7) I=IC(4)+IC(7)
IF (ORD.EQ.8) I=IC(4)+IC(5)+IC(6)+IC(8)
IF (ORD.EQ.9) I=IC(5)+IC(9)
IF (ORD.EQ.10) I=IC(7)+IC(10)
SEQ(ORD+1) = MOD(I,2)

RETURN
END
SUBROUTINE SETGRA

IMPLICIT NONE
INTEGER I, J1, J2, K1, K2, K3, K4, K5

Set-up graphics area text defaults

FORMAT('+', 2OA1)

WRITE (6,10) 27,82,87,32,96,96,32,64,63,111,127,63,95
   (0,0) - (4095,4095)  ! Window
WRITE (6,10) 27,82,86,32,96,96,32,64,55,111,127,63,95
   t (0,0) - (4095,3071)  ! Viewpor
WRITE (6,10) 27,77,86,48  ! Set sol
   id line style
WRITE (6,10) 27,84,77,51,49,49  ! Color m
   ode: HLS, Opaque, and Normal Modes
WRITE (6,10) 27,84,66,48,70,52,48  ! Set bac
   kground to white (0,100,0)
WRITE (6,10) 27,84,71,49,52,62,48,48  ! Set ind
   x #14 color to black (0,0,0)
WRITE (6,10) 27,77,76,62  ! Set lin
   e color to #14, black
WRITE (6,10) 27,77,80,39  ! Select
   solid fill color #7, yellow
WRITE (6,10) 27,77,81,50  ! Set gra
   phtext stroke precision
WRITE (6,10) 27,77,70,48  ! Set gra
   phtext font USA ASCII
WRITE (6,10) 27,77,82,48,48  ! Set gra
   phtext 0 degrees rotation
WRITE (6,10) 27,77,65,48,48  ! Set gra
   phtext 0 degrees slant
WRITE (6,10) 27,77,78,48  ! Set cha
   racter path to the right
WRITE (6,10) 27,77,84,60  ! Set gra
   phtext color to #12, purple
WRITE (6,10) 27,77,67,66,55,67,59,60  ! Set gra
   phtext size: 39 wide, 59 high, 12 spaces (std)
WRITE (6,10) 27,77,67,58,66,48,50  ! Set gra
   phtext size: 10 wide, 32 high, 2 space

Set-up Graphics Segments

WRITE (6,10) 27,83,75,33  ! DELETE
   ALL SEGMENTS
WRITE (6,10) 27,83,80,32,96,96,32,64  ! SET PIV
   OT POINT (0,0)

#1, Square
WRITE (6,10) 27,83,79,49  ! BEGIN S
   EGMENT
CALL COOR(0,0,0,K1,K2,K3,K4,K5) ! Return
WRITE (6,10) 27,76,80,K1,K2,K3,K4,K5,49 ! BEGIN P
ANEL BOUNDARY
CALL COOR(2,200,0,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,200,200,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,0,200,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,0,0,K1,K2,K3,K4,K5) ! Draw
WRITE (6,10) 27,76,69 ! END PAN
EL BOUNDARY
WRITE (6,10) 27,83,67 ! END SEG

C

#2, Triangle
WRITE (6,10) 27,83,79,50 ! BEGIN S
EGMENT
CALL COOR(0,0,0,K1,K2,K3,K4,K5) ! Return
WRITE (6,10) 27,76,80,K1,K2,K3,K4,K5,49 ! BEGIN P
ANEL BOUNDARY
CALL COOR(2,200,100,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,0,200,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,0,0,K1,K2,K3,K4,K5) ! Draw
WRITE (6,10) 27,76,69 ! END PAN
EL BOUNDARY
WRITE (6,10) 27,83,67 ! END SEG

C

#3, Circle
WRITE (6,10) 27,83,79,51 ! BEGIN S
EGMENT
CALL COOR(0,0,50,K1,K2,K3,K4,K5) ! Return
WRITE (6,10) 27,76,80,K1,K2,K3,K4,K5,49 ! BEGIN P
ANEL BOUNDARY
DO I=1,359
   J1=-50*COS(I*3.14159/180)+50
   J2=50*SIN(I*3.14159/180)+50
   CALL COOR(2,J1,J2,K1,K2,K3,K4,K5) ! Draw
ENDDO
WRITE (6,10) 27,76,69 ! END PAN
EL BOUNDARY
WRITE (6,10) 27,83,67 ! END SEG

C

#4, Small Circle
WRITE (6,10) 27,83,79,52 ! BEGIN S
EGMENT
CALL COOR(1,0,10,K1,K2,K3,K4,K5) ! Move
WRITE (6,10) 27,76,80,K1,K2,K3,K4,K5,49 ! BEGIN P
ANEL BOUNDARY
DO I=1,356,5
   J1=-10*COS(I*3.14159/180)+10
   J2=10*SIN(I*3.14159/180)+10
   CALL COOR(2,J1,J2,K1,K2,K3,K4,K5) ! Draw
ENDDO
WRITE (6,10) 27,76,69 ! END PAN
EL BOUNDARY
WRITE (6,10) 27, 83, 67

MENT

#5, XOR
WRITE (6,10) 27, 83, 79, 53

EGMENT
CALL COOR(1, 10, 0, K1, K2, K3, K4, K5)
DO I=-33, 33
   J1=(545*COS(I*3.14159/180) - 425)/3
   J2=545*SIN(I*3.14159/180)/3+100
   CALL COOR(2, J1, J2, K1, K2, K3, K4, K5)
ENDDO
CALL COOR(0, 40, 0, K1, K2, K3, K4, K5)
WRITE (6, 10) 27, 76, 80, K1, K2, K3, K4, K5, 49
ANEL BOUNDARY
DO I=-33, 33
   J1=(545*COS(I*3.14159/180) - 335)/3
   J2=545*SIN(I*3.14159/180)/3+100
   CALL COOR(2, J1, J2, K1, K2, K3, K4, K5)
ENDDO
DO I=100, 0, -1
   J1=-0.06*(I-50)**2+190
   J2=2*I
   CALL COOR(2, J1, J2, K1, K2, K3, K4, K5)
ENDDO
WRITE (6,10) 27, 76, 69

EL BOUNDARY
CALL COOR(1, 0, 50, K1, K2, K3, K4, K5)
CALL COOR(2, 32, 50, K1, K2, K3, K4, K5)
CALL COOR(1, 0, 150, K1, K2, K3, K4, K5)
CALL COOR(2, 32, 150, K1, K2, K3, K4, K5)
CALL COOR(1, 190, 100, K1, K2, K3, K4, K5)
CALL COOR(2, 200, 100, K1, K2, K3, K4, K5)
WRITE (6,10) 27, 83, 67
ENDESEG

#6, D-Latch
WRITE (6,10) 27, 83, 79, 54

EGMENT
CALL COOR(0, 40, 40, K1, K2, K3, K4, K5)
WRITE (6,10) 27, 76, 80, K1, K2, K3, K4, K5, 49
ANEL BOUNDARY
CALL COOR(2, 40, 200, K1, K2, K3, K4, K5)
CALL COOR(2, 180, 200, K1, K2, K3, K4, K5)
CALL COOR(2, 180, 40, K1, K2, K3, K4, K5)
CALL COOR(2, 40, 40, K1, K2, K3, K4, K5)
WRITE (6,10) 27, 76, 69
ENDESEG

EL BOUNDARY
CALL COOR(1, 20, 0, K1, K2, K3, K4, K5)
CALL COOR(2, 20, 120, K1, K2, K3, K4, K5)
CALL COOR(2, 40, 120, K1, K2, K3, K4, K5)
CALL COOR(1, 40, 100, K1, K2, K3, K4, K5)
CALL COOR(2, 60, 120, K1, K2, K3, K4, K5)
CALL COOR(2, 40, 140, K1, K2, K3, K4, K5)
CALL COOR(1, 0, 170, K1, K2, K3, K4, K5)
CALL COOR(2,40,170,K1,K2,K3,K4,K5) \hspace{1cm} ! Draw
CALL COOR(1,180,170,K1,K2,K3,K4,K5) \hspace{1cm} ! Move
CALL COOR(2,200,170,K1,K2,K3,K4,K5) \hspace{1cm} ! Draw
CALL COOR(0,70,20,K1,K2,K3,K4,K5) \hspace{1cm} ! Return
WRITE (6,10) 27,83,88,52,K1,K2,K3,K4,K5 \hspace{1cm} ! SET SEG MENT #4
WRITE (6,10) 27,76,75,52 \hspace{1cm} ! INCLUDE
CALL COOR(1,80,20,K1,K2,K3,K4,K5) \hspace{1cm} ! Move
CALL COOR(2,80,0,K1,K2,K3,K4,K5) \hspace{1cm} ! Draw
CALL COOR(1,140,40,K1,K2,K3,K4,K5) \hspace{1cm} ! Move
CALL COOR(2,140,0,K1,K2,K3,K4,K5) \hspace{1cm} ! Draw
CALL COOR(1,48,154,K1,K2,K3,K4,K5) \hspace{1cm} ! Move
CALL TEXT(1,'D') \hspace{1cm} ! GRAPHIC
CALL COOR(1,160,154,K1,K2,K3,K4,K5) \hspace{1cm} ! Graphic
CALL TEXT(1,'Q') \hspace{1cm} ! Move
CALL COOR(1,74,48,K1,K2,K3,K4,K5) \hspace{1cm} ! Draw
CALL TEXT(1,'R') \hspace{1cm} ! Graphic
CALL COOR(1,134,48,K1,K2,K3,K4,K5) \hspace{1cm} ! Move
CALL TEXT(1,'S') \hspace{1cm} ! Graphic
WRITE (6,10) 27,83,67 \hspace{1cm} ! END SEG MENT

#7, Summation Symbol
WRITE (6,10) 27,83,79,55 \hspace{1cm} ! BEGIN SEG MENT
WRITE (6,10) 27,77,76,60 \hspace{1cm} ! Set col or #12, purple
CALL COOR(1,32,6,K1,K2,K3,K4,K5) \hspace{1cm} ! Move
CALL COOR(2,32,0,K1,K2,K3,K4,K5) \hspace{1cm} ! Draw
CALL COOR(2,0,0,K1,K2,K3,K4,K5) \hspace{1cm} ! Draw
CALL COOR(2,16,26,K1,K2,K3,K4,K5) \hspace{1cm} ! Draw
CALL COOR(2,0,52,K1,K2,K3,K4,K5) \hspace{1cm} ! Draw
CALL COOR(2,32,52,K1,K2,K3,K4,K5) \hspace{1cm} ! Draw
CALL COOR(2,32,46,K1,K2,K3,K4,K5) \hspace{1cm} ! Draw
WRITE (6,10) 27,77,76,62 \hspace{1cm} ! Set col or #14, black
WRITE (6,10) 27,83,67 \hspace{1cm} ! END SEG MENT

#8, Mirror XOR
WRITE (6,10) 27,83,79,56 \hspace{1cm} ! BEGIN SEG MENT
CALL COOR(0,0,0,K1,K2,K3,K4,K5) \hspace{1cm} ! Return
WRITE (6,10) 27,83,88,53,K1,K2,K3,K4,K5 \hspace{1cm} ! SET SEG MENT #5
WRITE (6,10) 27,76,75,53 \hspace{1cm} ! INCLUDE
WRITE (6,10) 27,83,67 \hspace{1cm} ! ENDSEG MENT
WRITE (6,10) 27,83,73,56,49,48,49,48,75,52,48,32,96,
#8 IMAGE XFORM x1,y1,180 deg at (100,100)

C10, Transmitter Block

! BEGIN SEG

WRITE (6,10) 27,83,79,58

SEGMENT

CALL COOR(0,0,800,K1,K2,K3,K4,K5)

WRITE (6,10) 27,83,88,49,K1,K2,K3,K4,K5

MENT #1

WRITE (6,10) 27,76,75,49

COPY OF #1

CALL COOR(1,100,800,K1,K2,K3,K4,K5)

CALL COOR(2,100,700,K1,K2,K3,K4,K5)

CALL COOR(0,0,500,K1,K2,K3,K4,K5)

WRITE (6,10) 27,83,88,49,K1,K2,K3,K4,K5

MENT #1

WRITE (6,10) 27,76,75,49

COPY OF #1

CALL COOR(1,100,500,K1,K2,K3,K4,K5)

CALL COOR(2,100,400,K1,K2,K3,K4,K5)

CALL COOR(1,200,300,K1,K2,K3,K4,K5)

CALL COOR(2,100,300,K1,K2,K3,K4,K5)

CALL COOR(2,100,200,K1,K2,K3,K4,K5)

CALL COOR(0,0,0,K1,K2,K3,K4,K5)

WRITE (6,10) 27,83,88,49,K1,K2,K3,K4,K5

MENT #1

WRITE (6,10) 27,76,75,49

COPY OF #1

CALL COOR(0,200,250,K1,K2,K3,K4,K5)

WRITE (6,10) 27,83,88,53,K1,K2,K3,K4,K5

MENT #5

WRITE (6,10) 27,76,75,53

COPY OF #5

CALL COOR(1,400,350,K1,K2,K3,K4,K5)

CALL COOR(2,500,350,K1,K2,K3,K4,K5)

CALL COOR(2,1800,350,K1,K2,K3,K4,K5)

CALL COOR(2,1800,430,K1,K2,K3,K4,K5)

CALL COOR(2,1820,450,K1,K2,K3,K4,K5)

CALL COOR(2,1780,450,K1,K2,K3,K4,K5)

CALL COOR(2,1800,430,K1,K2,K3,K4,K5)

CALL COOR(0,500,250,K1,K2,K3,K4,K5)

WRITE (6,10) 27,83,88,50,K1,K2,K3,K4,K5

MENT #2

WRITE (6,10) 27,76,75,50

COPY OF #2

CALL COOR(0,800,300,K1,K2,K3,K4,K5)

WRITE (6,10) 27,83,88,51,K1,K2,K3,K4,K5

MENT #3

WRITE (6,10) 27,76,75,51

COPY OF #3

CALL COOR(0,834,324,K1,K2,K3,K4,K5)

WRITE (6,10) 27,83,88,55,K1,K2,K3,K4,K5

MENT #7

WRITE (6,10) 27,76,75,55
COPY OF #7
CALL COOR(1, 850, 300, K1, K2, K3, K4, K5)  ! Move
CALL COOR(2, 850, 200, K1, K2, K3, K4, K5)  ! Draw
CALL COOR(0, 750, 0, K1, K2, K3, K4, K5)  ! Return
WRITE (6, 10) 27, 83, 88, 49, K1, K2, K3, K4, K5  ! SET SEG
MENT #1
WRITE (6, 10) 27, 76, 75, 49  
COPY OF #1
CALL COOR(0, 1000, 250, K1, K2, K3, K4, K5)  ! Return
WRITE (6, 10) 27, 83, 88, 49, K1, K2, K3, K4, K5  ! SET SEG
MENT #1
WRITE (6, 10) 27, 76, 75, 49  
COPY OF #1
CALL COOR(0, 1300, 300, K1, K2, K3, K4, K5)  ! Return
WRITE (6, 10) 27, 83, 88, 51, K1, K2, K3, K4, K5  ! SET SEG
MENT #3
WRITE (6, 10) 27, 76, 75, 51  
COPY OF #3
CALL COOR(1, 1350, 300, K1, K2, K3, K4, K5)  ! Move
CALL COOR(2, 1350, 200, K1, K2, K3, K4, K5)  ! Draw
CALL COOR(0, 1250, 0, K1, K2, K3, K4, K5)  ! Return
WRITE (6, 10) 27, 83, 88, 49, K1, K2, K3, K4, K5  ! SET SEG
MENT #1
WRITE (6, 10) 27, 76, 75, 49  
COPY OF #1
CALL COOR(0, 1500, 250, K1, K2, K3, K4, K5)  ! Return
WRITE (6, 10) 27, 83, 88, 49, K1, K2, K3, K4, K5  ! SET SEG
MENT #1
WRITE (6, 10) 27, 76, 75, 49  
COPY OF #1
WRITE (6, 10) 27, 77, 76, 60  
or #12, purple
CALL COOR(1, 1328, 372, K1, K2, K3, K4, K5)  ! Move
CALL COOR(2, 1372, 328, K1, K2, K3, K4, K5)  ! Draw
CALL COOR(1, 1328, 328, K1, K2, K3, K4, K5)  ! Move
CALL COOR(2, 1372, 372, K1, K2, K3, K4, K5)  ! Draw
WRITE (6, 10) 27, 77, 76, 62  
or #14, black
CALL COOR(1, 1790, 460, K1, K2, K3, K4, K5)  ! Move
CALL COOR(2, 1810, 480, K1, K2, K3, K4, K5)  ! Draw
CALL COOR(2, 1810, 470, K1, K2, K3, K4, K5)  ! Draw
CALL COOR(2, 1830, 490, K1, K2, K3, K4, K5)  ! Draw
CALL COOR(2, 1822, 490, K1, K2, K3, K4, K5)  ! Draw
CALL COOR(2, 1830, 490, K1, K2, K3, K4, K5)  ! Draw
CALL COOR(2, 1830, 482, K1, K2, K3, K4, K5)  ! Draw
CALL COOR(1, 822, 952, K1, K2, K3, K4, K5)  ! Move
CALL TEXT(16, 'BLOCK DIAGRAM OF')  
GRAPHIC
TEXT
CALL COOR(1, 804, 904, K1, K2, K3, K4, K5)  ! Move
CALL TEXT(19, 'TRANSMITTER SECTION')  
GRAPHIC
TEXT
CALL COOR(1, 8, 952, K1, K2, K3, K4, K5)  ! Move
CALL TEXT(13, 'DATA WAVEFORM')  
GRAPHIC
TEXT
CALL COOR(1, 8, 904, K1, K2, K3, K4, K5)  ! Move
CALL TEXT(9, 'PSN ORD =')
CALL COOR(1, 8, 856, K1, K2, K3, K4, K5)
CALL TEXT(6, 'RATE =')
CALL COOR(1, 8, 808, K1, K2, K3, K4, K5)
CALL TEXT(6, 'IC s =')
CALL COOR(1, 108, 718, K1, K2, K3, K4, K5)
CALL TEXT(5, 'ENCODER')
CALL COOR(1, 8, 652, K1, K2, K3, K4, K5)
CALL TEXT(7, 'Sig 1')
CALL COOR(1, 8, 604, K1, K2, K3, K4, K5)
CALL TEXT(3, 'n =')
CALL COOR(1, 8, 556, K1, K2, K3, K4, K5)
CALL TEXT(3, 'k =')
CALL COOR(1, 108, 404, K1, K2, K3, K4, K5)
CALL TEXT(5, 'Sig 2')
CALL COOR(1, 108, 304, K1, K2, K3, K4, K5)
CALL TEXT(5, 'Sig 3')
CALL COOR(1, 152, K1, K2, K3, K4, K5)
CALL TEXT(14, 'SPREADING WAVE')
CALL COOR(1, 8, 104, K1, K2, K3, K4, K5)
CALL TEXT(9, 'PSN ORD =')
CALL COOR(1, 8, 56, K1, K2, K3, K4, K5)
CALL TEXT(11, 'PROC GAIN =')
CALL COOR(1, 8, 8, K1, K2, K3, K4, K5)
CALL TEXT(6, 'IC s =')
CALL COOR(1, 408, 354, K1, K2, K3, K4, K5)
CALL TEXT(5, 'Sig 4')
CALL COOR(1, 508, 354, K1, K2, K3, K4, K5)
CALL TEXT(9, 'AMPLITUDE')
CALL COOR(1, 758, 152, K1, K2, K3, K4, K5)
CALL TEXT(12, 'DISPLACEMENT')
CALL COOR(1, 908, 354, K1, K2, K3, K4, K5)
CALL TEXT(5, 'Sig 5')
CALL COOR(1, 1008, 402, K1, K2, K3, K4, K5)
CALL TEXT(12, 'PULSE SHAPER')
CALL COOR(1, 1008, 354, K1, K2, K3, K4, K5)
CALL TEXT(13, 'RAISED COSINE')
CALL COOR(1,1208,354,K1,K2,K3,K4,K5) ! Move
CALL TEXT(5,'Sig 6') ! GRAPHIC
CALL COOR(1,1408,354,K1,K2,K3,K4,K5) ! Move
CALL TEXT(5,'Sig 7') ! GRAPHIC
CALL COOR(1,1258,152,K1,K2,K3,K4,K5) ! Move
CALL TEXT(7,'CARRIER') ! GRAPHIC
CALL COOR(1,1258,104,K1,K2,K3,K4,K5) ! Move
CALL TEXT(5,'AMP =') ! GRAPHIC
CALL COOR(1,1258,56,K1,K2,K3,K4,K5) ! Move
CALL TEXT(6,'FREQ =') ! GRAPHIC
CALL COOR(1,1708,354,K1,K2,K3,K4,K5) ! Move
CALL TEXT(5,'Sig 8') ! GRAPHIC
CALL COOR(1,1508,402,K1,K2,K3,K4,K5) ! Move
CALL TEXT(8,'LOW PASS') ! GRAPHIC
CALL COOR(1,1508,354,K1,K2,K3,K4,K5) ! Move
CALL TEXT(6,'FILTER') ! GRAPHIC
WRITE (6,10) 27,83,67 END SEG

C #12, Receiver Block
WRITE (6,10) 27,83,79,60 BEGIN SEG
CALL COOR(1,0,470,K1,K2,K3,K4,K5) ! Move
CALL COOR(2,20,450,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,20,460,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,40,440,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,32,440,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,40,440,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,40,448,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,40,440,K1,K2,K3,K4,K5) ! Draw
CALL COOR(1,30,400,K1,K2,K3,K4,K5) ! Move
CALL COOR(2,10,420,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,50,420,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,30,400,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,30,350,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,130,350,K1,K2,K3,K4,K5) ! Draw
CALL COOR(0,130,300,K1,K2,K3,K4,K5) ! Draw
WRITE (6,10) 27,83,88,51,K1,K2,K3,K4,K5 ! RETURN
WRITE (6,10) 27,83,88,51,K1,K2,K3,K4,K5 ! SET SEG
MENT #3
WRITE (6,10) 27,76,75,51 ! INCLUDE
COPY OF #3
CALL COOR(0,164,324,K1,K2,K3,K4,K5) ! Return
WRITE (6,10) 27,83,88,55,K1,K2,K3,K4,K5 ! SET SEG
MENT #7
WRITE (6,10) 27,76,75,55 ! INCLUDE
COPY OF #7
CALL COOR(1,180,300,K1,K2,K3,K4,K5) ! Move
CALL COOR(2,180,200,K1,K2,K3,K4,K5) ! Draw
CALL COOR(0,80,0,K1,K2,K3,K4,K5) ! Return
WRITE (6,10) 27,83,88,49,K1,K2,K3,K4,K5 ! SET SEG
MENT #1
WRITE (6,10) 27,76,75,49 ! INCLUDE
COPY OF #1
CALL COOR(1,230,350,K1,K2,K3,K4,K5) ! Move
CALL COOR(2,380,350,K1,K2,K3,K4,K5) ! Draw
CALL COOR(0,380,300,K1,K2,K3,K4,K5) ! Return
WRITE (6,10) 27,83,88,51,K1,K2,K3,K4,K5 ! SET SEG
MENT #3
WRITE (6,10) 27,76,75,51 ! INCLUDE
COPY OF #3
CALL COOR(1,430,300,K1,K2,K3,K4,K5) ! Move
CALL COOR(2,430,200,K1,K2,K3,K4,K5) ! Draw
CALL COOR(0,330,0,K1,K2,K3,K4,K5) ! Return
WRITE (6,10) 27,83,88,49,K1,K2,K3,K4,K5 ! SET SEG
MENT #1
WRITE (6,10) 27,76,75,49 ! INCLUDE
COPY OF #1
CALL COOR(1,480,350,K1,K2,K3,K4,K5) ! Move
CALL COOR(2,630,350,K1,K2,K3,K4,K5) ! Draw
CALL COOR(0,630,200,K1,K2,K3,K4,K5) ! Return
WRITE (6,10) 27,83,88,50,K1,K2,K3,K4,K5 ! SET SEG
MENT #2
WRITE (6,10) 27,76,75,50 ! INCLUDE
COPY OF #2
CALL COOR(1,830,300,K1,K2,K3,K4,K5) ! Move
CALL COOR(2,930,300,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,880,300,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,880,150,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,1180,150,K1,K2,K3,K4,K5) ! Draw
CALL COOR(2,1230,200,K1,K2,K3,K4,K5) ! Draw
CALL COOR(0,930,200,K1,K2,K3,K4,K5) ! Return
WRITE (6,10) 27,83,88,49,K1,K2,K3,K4,K5 ! SET SEG
MENT #1
WRITE (6,10) 27,76,75,49 ! INCLUDE
COPY OF #1
CALL COOR(1,1130,300,K1,K2,K3,K4,K5) ! Move
CALL COOR(2,1230,300,K1,K2,K3,K4,K5) ! Draw
CALL COOR(0,1230,150,K1,K2,K3,K4,K5) ! Return
WRITE (6,10) 27,83,88,53,K1,K2,K3,K4,K5 ! SET SEG
MENT #5
WRITE (6,10) 27,76,75,53 ! INCLUDE
COPY OF #5
CALL COOR(1,1143,250,K1,K2,K3,K4,K5) ! Move
CALL COOR(2,1530,250,K1,K2,K3,K4,K5) ! Draw
CALL COOR(0,1530,150,K1,K2,K3,K4,K5) ! Return
WRITE (6,10) 27,83,88,49,K1,K2,K3,K4,K5 ! SET SEG
MENT #1
WRITE (6,10) 27,76,75,49 ! INCLUDE
COPY OF #1
CALL COOR(1,1730,250,K1,K2,K3,K4,K5) ! Move
CALL COOR(2,1830,250,K1,K2,K3,K4,K5) ! Draw
WRITE (6,10) 27,77,76,60
or #12, purple
CALL COOR(1,408,372,K1,K2,K3,K4,K5)
CALL COOR(2,452,328,K1,K2,K3,K4,K5)
CALL COOR(1,408,328,K1,K2,K3,K4,K5)
CALL COOR(2,452,372,K1,K2,K3,K4,K5)
WRITE (6,10) 27,77,75,62
or #14, black
CALL COOR(1,708,430,K1,K2,K3,K4,K5)
CALL TEXT(33,'BLOCK DIAGRAM OF RECEIVER SECTION')
CALL COOR(1,88,152,K1,K2,K3,K4,K5)
CALL TEXT(9,'NOISE AND')
CALL COOR(1,88,104,K1,K2,K3,K4,K5)
CALL TEXT(7,'JAMMING')
CALL COOR(1,88,56,K1,K2,K3,K4,K5)
CALL TEXT(3,'N =')
CALL COOR(1,88,8,K1,K2,K3,K4,K5)
CALL TEXT(3,'J =')
CALL COOR(1,188,214,K1,K2,K3,K4,K5)
CALL TEXT(5,'Sig 9')
CALL COOR(1,238,354,K1,K2,K3,K4,K5)
CALL TEXT(6,'Sig 10')
CALL COOR(1,488,354,K1,K2,K3,K4,K5)
CALL TEXT(6,'Sig 11')
CALL COOR(1,338,152,K1,K2,K3,K4,K5)
CALL TEXT(5,'AMP =')
CALL COOR(1,338,8,K1,K2,K3,K4,K5)
CALL TEXT(6,'FREQ =')
CALL COOR(1,638,314,K1,K2,K3,K4,K5)
CALL TEXT(7,'COMPARE')
CALL COOR(1,838,304,K1,K2,K3,K4,K5)
CALL TEXT(6,'Sig 12')
CALL COOR(1,938,352,K1,K2,K3,K4,K5)
CALL TEXT(10,'CORRELATOR')
CALL COOR(1,938,306,K1,K2,K3,K4,K5)
CALL TEXT(11,'ACQUISITION')
CALL COOR(1,938,258,K1,K2,K3,K4,K5) ! Move
CALL TEXT(5,'SEQ =') ! GRAPHIC
TEXT
CALL COOR(1,1138,304,K1,K2,K3,K4,K5) ! Move
CALL TEXT(6,'Sig 13') ! GRAPHIC
TEXT
CALL COOR(1,1438,254,K1,K2,K3,K4,K5) ! Move
CALL TEXT(6,'Sig 14') ! GRAPHIC
TEXT
CALL COOR(1,1538,302,K1,K2,K3,K4,K5) ! Move
CALL TEXT(7,'DEC0DER') ! GRAPHIC
TEXT
CALL COOR(1,1538,254,K1,K2,K3,K4,K5) ! Move
CALL TEXT(3,'n =') ! GRAPHIC
TEXT
CALL COOR(1,1538,206,K1,K2,K3,K4,K5) ! Move
CALL TEXT(3,'k =') ! GRAPHIC
TEXT
CALL COOR(1,1738,254,K1,K2,K3,K4,K5) ! Move
CALL TEXT(6,'Sig 15') ! GRAPHIC
TEXT

WRITE (6,10) 27,83,67 ! END SEG
MENT
C
C SET SEGMENT VISIBILITY
WRITE (6,10) 27,83,86,33,48 ! Make al
l segments invisible
C
RETURN
END
SUBROUTINE SETTXT

Set-up dialog area text defaults

WRITE (6,10) 27,37,33,48      ! SELECT CODE mode=TEK
FORMAT(’+’,5A1)
WRITE (6,10) 27,76,90      ! CLEAR DIALOG SCROLL

SET DIALOG AREA COLOR MAP
  Index#0=black  Index#1=white  Index#2=red  Index#3=green
  Index#4=blue  Index#5=cyan  Index#6=magenta  Index#7=yellow
WRITE (6,20) 27
FORMAT(’+’,1A1,’TF430F40’)      ! Index#3=white

SET DIALOG AREA LINES
WRITE (6,10) 27,76,76,66,48       !32 lines visible

SET DIALOG AREA INDEX
WRITE (6,10) 27,76,73,49,52,52      ! Char=Index#1,
   Cell=Index#4,  Back=Index#4

SET DIALOG AREA WRITING MODE
WRITE (6,10) 27,76,77,48      ! Replace blank and underscore

PAGE
WRITE (6,10) 27,12      ! Erase the screen

RETURN
END
SUBROUTINE TEXT(NUM,WORDS)
C
C This subroutine writes the ASCII string WORDS on the TEK4100 screen at the current position
C
IMPLICIT NONE
INTEGER NUM, ! Number of characters
2 I,K1,K2,K3
CHARACTERWORDS(100)! Limited to 100 characters
C
IF (NUM.GT.1023) THEN
K1=32+16+IBITS(NUM,0,4)
K2=64+IBITS(NUM,4,6)
K3=64+IBITS(NUM,10,5)
WRITE (6,10) 27,76,84,K3,K2,K1,(WORDS(I),I=1,NUM)
10 FORMAT(’+’,6A1,<NUM>A1)
ENDIF
C
IF ((NUM.GT.15).AND.(NUM.LE.1023)) THEN
K1=32+16+IBITS(NUM,0,4)
K2=64+IBITS(NUM,4,6)
WRITE (6,20) 27,76,84,K2,K1,(WORDS(I),I=1,NUM)
20 FORMAT(’+’,5A1,<NUM>A1)
ENDIF
C
IF ((NUM.GT.0).AND.(NUM.LE.15)) THEN
K1=32+16+IBITS(NUM,0,4)
WRITE (6,30) 27,76,84,K1,(WORDS(I),I=1,NUM)
30 FORMAT(’+’,4A1,<NUM>A1)
ENDIF
C
RETURN
END
SUBROUTINE TXTPLT(SIGNAL,XMIN,XMAX,YMIN,YMAX)
C
C Write the xy-dimensions of the screen plot
C
C IMPLICIT NONE
C
INTEGER SIGNAL,K1,K2,K3,K4,K5
REAL XMIN,XMAX,YMIN,YMAX

WRITE (6,10) 27,77,67,66,55,67,59,60 ! Set graphtext
size 39 wide, 59 high, 12 spaces
10 FORMAT(’+’,8A1)
WRITE (6,10) 27,77,84,60 ! Set graphtext
color #12, purple
CALL COOR(1,200,25,K1,K2,K3,K4,K5) ! Move
WRITE (6,20) 27,76,84,58,XMIN ! Move
20 FORMAT(’+’,4A1,1PE10.3)
CALL COOR(1,3600,25,K1,K2,K3,K4,K5) ! Move
WRITE (6,20) 27,76,84,58,XMAX ! Move
CALL COOR(1,25,100,K1,K2,K3,K4,K5) ! Move
WRITE (6,20) 27,76,84,58,YMIN ! Move
CALL COOR(1,25,3050,K1,K2,K3,K4,K5) ! Move
WRITE (6,20) 27,76,84,58,YMAX ! Move
CALL TEXT(8,’Signal #’) ! Move
WRITE (6,30) 27,76,84,58,SIGNAL ! SET GRAPHTEXT
30 FORMAT(’+’,4A1,1I2)
WRITE (6,10) 27,77,67,58,66,48,50 ! SET GRAPHTEXT
SIZE 10 wide, 32 high, 2 spaces
C
RETURN
END
SUBROUTINE TXTREC

C Write all of the variable values to the receiver block diagram

IMPLICIT NONE
INCLUDE 'COMMON.VAR'

WRITE (6,10) 27,83,79,61  ! Begin segment #13
10 FORMAT ('+',8A1)    ! Move
CALL COOR(1,1124,1056,K1,K2,K3,K4,K5)  ! Move
WRITE (6,20) 27,76,84,58,NOISE_AMP ! GRAPHIC
WRITE (6,20) 27,76,84,58,JAM_AMP ! GRAPHIC
WRITE (6,20) 27,76,84,58,OSC_AMP ! GRAPHIC
WRITE (6,20) 27,76,84,58,OSC_FREQ ! GRAPHIC
WRITE (6,20) 27,76,84,58,COMPR_LEV ! GRAPHIC
WRITE (6,30) 27,76,84,(48+DATA_POLY_ORD+1),2CORR_ACQIS(J),J=1,DATA_POLY_ORD+1) ! GRAPHIC
30 FORMAT ('+',4A1,<DATA_POLY_ORD+1>Il) ! Move
CALL COOR(1,1638,1266,K1,K2,K3,K4,K5) ! Move
WRITE (6,40) 27,76,84,50,DECODE_TOTAL ! GRAPHIC
40 FORMAT ('+',4A1,Il2) ! Move
CALL COOR(1,1998,1258,K1,K2,K3,K4,K5) ! Move
WRITE (6,40) 27,76,84,50,DECODE_DATA ! GRAPHIC
CALL LIB$WAIT(5.0) ! Delete
WRITE (6,10) 27,83,75,61  ! Delete segment #13

RETURN
END
SUBROUTINE TXTXMT

C
C Write all of the variable values to the transmitter block diagram
C
IMPLICIT NONE
INCLUDE 'COMMON.VAR'

C
WRITE (6,10) 27,83,79,59 ! Begin segment
#11
10 FORMAT ('+',8A1)
   CALL COOR(1,1116,1904,K1,K2,K3,K4,K5) ! Move
   WRITE (6,20) 27,76,84,50,DATA_POLY_ORD ! GRAPHIC TEXT
20 FORMAT('+',4A1,112)
   CALL COOR(1,1080,1856,K1,K2,K3,K4,K5) ! Move
   WRITE (6,30) 27,76,84,58,DATA_RATE ! GRAPHIC TEXT
30 FORMAT('+',4A1,1PE10.3)
   CALL COOR(1,1080,1808,K1,K2,K3,K4,K5) ! Move
   WRITE (6,40) 27,76,84,(48+DATA_POLY_ORD),
               2               (DATA_IC(J),J=1,DATA_POLY_ORD) ! GRAPHIC TEXT
40 FORMAT('+',4A1,<DATA_POLY_ORD>I1)
   CALL COOR(1,1056,1604,K1,K2,K3,K4,K5) ! Move
   WRITE (6,45) 27,76,84,58,ENCODE_TOTAL ! GRAPHIC TEXT
45 FORMAT('+',4A1,I2)
   CALL COOR(1,1056,1556,K1,K2,K3,K4,K5) ! Move
   WRITE (6,45) 27,76,84,58,ENCODE_DATA ! GRAPHIC TEXT
   CALL COOR(1,1116,1104,K1,K2,K3,K4,K5) ! Move
   WRITE (6,20) 27,76,84,50,SPRD_POLY_ORD ! GRAPHIC TEXT
   CALL COOR(1,1140,1056,K1,K2,K3,K4,K5) ! Move
   WRITE (6,20) 27,76,84,50,SPRD_GAIN ! GRAPHIC TEXT
   CALL COOR(1,1080,1008,K1,K2,K3,K4,K5) ! Move
   WRITE (6,60) 27,76,84,(48+SPRD_POLY_ORD),
               2               (SPRD_IC(J),J=1,SPRD_POLY_ORD) ! GRAPHIC TEXT
60 FORMAT('+',4A1,<SPRD_POLY_ORD>I1)
   CALL COOR(1,1508,1306,K1,K2,K3,K4,K5) ! Move
   WRITE (6,70) 27,76,84,52,LEVEL_AMP ! GRAPHIC TEXT
70 FORMAT('+',4A1,I4)
   CALL COOR(1,1758,1104,K1,K2,K3,K4,K5) ! Move
   WRITE (6,70) 27,76,84,52,LEVEL_DISP ! GRAPHIC TEXT
   CALL COOR(1,2008,1306,K1,K2,K3,K4,K5) ! Move
   WRITE (6,20) 27,76,84,58,PULSE_ID ! GRAPHIC TEXT
   CALL COOR(1,2318,1104,K1,K2,K3,K4,K5) ! Move
   WRITE (6,30) 27,76,84,58,CARR_AMP ! GRAPHIC TEXT
   CALL COOR(1,2230,1056,K1,K2,K3,K4,K5) ! Move
   WRITE (6,30) 27,76,84,58,CARR_FREQ ! GRAPHIC TEXT
   CALL COOR(1,2508,1306,K1,K2,K3,K4,K5) ! Move
   WRITE (6,20) 27,76,84,58,FLTR_ID ! GRAPHIC TEXT
   WRITE (6,10) 27,83,67 ! End segment
   CALL LIB$WAIT(5.0)
   WRITE (6,10) 27,83,75,59 ! Delete segment
#11
RETURN
END
SUBROUTINE XMIT  

Subroutines called:
  BLOCKL
  FEEDBACK
  PSN
  ENCODE

IMPLICIT NONE
REAL RATE, TIME

! Used to compute pulse shape
INCLUDE 'COMMON.VAR'

Write transmitter menu

11 WRITE (6,10) 27,76,90
! CLEAR DIALOG SCROLL
10 FORMAT('$$',3A1)
WRITE (6,12)
12 FORMAT(/,'$$',10X,'TRANSMITTER MENU',/)
  ',/'$$',6X,'1 - Data waveform specification'
  ',/'$$',6X,'2 - Encoder choice'
  ',/'$$',6X,'3 - Spreading waveform specification'
  ',/'$$',6X,'4 - Level shifter change'
  ',/'$$',6X,'5 - Pulse shaper choice'
  ',/'$$',6X,'6 - Carrier parameter change'
  ',/'$$',6X,'7 - Transmitter filter choice'
  ',/'$$',6X,'8 - Return to Main Menu'/
  ',/'$$',X,'Enter the digit corresponding'
  ',/'$$',X,'to the desired function : ')
READ (5,20,ERR=11) IN
20 FORMAT(I3)
IF ((IN.GT.8).OR.(IN.LT.1)) GOTO 11
IF (IN.EQ.1) THEN
  21 WRITE (6,10) 27,76,90
  ! CLEAR DIALOG SCROLL
  25 WRITE (6,25) DATA RATE
  25 FORMAT(/,'$$',10X,'DATA WAVEFORM SPECIFICATION',/)
  27 ',/'$$',X,'Enter the new data rate if desired <',1PE10.3
  28 ',/'$$',X,'
  29 READ (5,26,ERR=21) RN
  26 FORMAT(F12.0)
  IF (RN.NE.0) DATA RATE=RN
  CALL PSN(DATA_POLY_ORD,DATA_IC) ! Get data pseudo random information
  CALL FEEDBACK(DATA_POLY_ORD,DATA_PSN_FDBK)
  CALL BLOCKL(DATA_POLY_ORD,DATA_IC,DATA_PSN_FDBK)
ENDIF
IF (IN.EQ.2) THEN
  CALL ENCODE(ENCODE_TYPE,ENCODE_TOTAL,ENCODE_DATA)
ENDIF
IF (IN.EQ.3) THEN
  30 WRITE (6,10) 27,76,90
  ! CLEAR DIALOG SCROLL
  35 WRITE (6,36) SPRD BLOCK
  36 FORMAT(/,'$$',10X,'SPREADING WAVEFORM SPECIFICATION',/)

Enter the new number of spreading blocks if desired <',I6,'> : ')

READ (5,37,ERR=30) J

FORMAT(I4)
IF (J.GT.10) GOTO 35
IF (J.NE.0) SPRD_GAIN=J
CALL PSN(SPRD_POLY_ORD,SPRD_IC)
! Get data pseudo random information
CALL FEEDBACK(SPRD_POLY_ORD,SPRD_PSN_FDBK)
CALL BLOCKL(SPRD_POLY_ORD,SPRD_IC,SPRD_PSN_FDBK)
SPRD_GAIN = SPRD_BLOCK * (2 ** SPRD_POLY_ORD-1)
ENDIF

IF (IN.EQ.4) THEN
40 WRITE (6,10) 27,76,90
! CLEAR DIALOG SCROLL
WRITE (6,41) LEVEL
41 FORMAT(/,'$',IOX,'LEVEL SHIFTER',/,/
2 ' ', '$',X,'Enter the new level multiplier'
3 ' ', factor if desired <',I14,'> : ')
READ (5,37,ERR=40) J
IF (J.NE.0) LEVEL=J
WRITE (6,42) LEVEL_DISP
42 FORMAT(/,'$',X,'Enter
2 ' ', the new level displacement'
3 ' ', if desired <',I14,'> : ')
READ (5,37,ERR=40) J
IF (J.NE.0) LEVEL_DISP=J
ENDIF

IF (IN.EQ.5) THEN
50 WRITE (6,10) 27,76,90
! CLEAR DIALOG SCROLL
WRITE (6,51) CARR_AMP
51 FORMAT(/,'$',IOX,'CARRIER PARAMETERS',/,/
2 ' ', '$',X,'Enter the new carrier amplitude'
3 ' ', if desired <',IPE10.3,'> : ')
READ (5,62,ERR=60) RN
IF (RN.NE.0) CARR_amp=RN
WRITE (6,63) CARR_CYCLE
 FORMAT(/,'$',X,'Enter the new number of carrier 1/4'
 2 , 'cycles if desired <',113,',' : ')
 READ (5,20,ERR=60) IN
 IF (IN.NE.0) CARR CYCLE=IN
 CARR FREQ = DATA RATE * ENCODE TOTAL / ENCODE DATA / 4 *
 2 SPRD_GAIN * (2**SPRD_POLY_ORD-1) * CARR CYCLE
ENDIF
IF (IN.EQ.7) THEN
 70 WRITE (6,10) 27,76,90
   ! CLEAR DIALOG SCROLL
 WRITE (6,71)
71 FORMAT(/,'$',10X,'TRANSMITTER FILTER SELECTION MENU',/ 2 ,'/,'$',6X,'0 - None (Allpass)',
 3 '/,'$',6X,'1 - Lowpass, frequency = ',
 4 '/,'$',X,'Enter the new filter ID : ')
 READ (5,20,ERR=70) FLTR_ID
 IF ((FLTR_ID.LT.O).OR.(FLTR_ID.GT.O)) GOTO 70
 IF (FLTR_ID.EQ.O) THEN
 11 DO J=1,10
    FLTR VAL(J)=1.0
  ENDDO
ENDIF
ENDIF
IF (IN.NE.8) THEN
  VALID SIG= MIN(IN-1,VALID SIG)
  GOTO 11
ENDIF
RETURN
END

SUBROUTINE PSN(POLY_ORD,IC)

Queries for Pseudo Random Characteristics

Subroutine called : FEEDBACK

IMPLICIT NONE
REAL RN
INTEGER IN,N,POLY_ORD,IC(10),FEED(10)

11 WRITE (6,10) 27,76,90
   ! CLEAR DIALOG SCROLL
10 FORMAT('$',3A1)
 WRITE (6,15) POLY ORD
15 FORMAT(/,'$',10X,'PRIMITIVE POLYNOMIAL SELECTION MENU',// 2 '/,'$',6X,'2 - x^2 + x + 1' , / 3 '/,'$',6X,'3 - x^3 + x + 1' , / 4 '/,'$',6X,'4 - x^4 + x + 1' , / 5 '/,'$',6X,'5 - x^5 + x^2 + 1' , / 6 '/,'$',6X,'6 - x^6 + x + 1', / 7 '/,'$',6X,'7 - x^7 + x^3 + 1', / 8 '/,'$',6X,'8 - x^8 + x^4 + x^3 + x^2 + 1', / 9 '/,'$',6X,'9 - x^9 + x^4 + 1', / 1 '/,'$',6X,'10 - x^10 + x^3 + 1', ///
2 ,'$',X, 'Enter the new order if desired <',I2,'> : ') 
READ (5,20,ERR=11) IN
20 FORMAT (I12)
IF ((IN.LT.0).OR.(IN.GT.10)) GOTO 11
IF (IN.NE.0) POLY_ORD=IN
C
WRITE (6,30) (IC(N), N=1,POLY_ORD)
30 FORMAT(/,'$',X,'Enter the new ICs if desired <' 
2 ,<POLY_ORD>II,'> : ') 
READ (5,35,ERR=11) IN
35 FORMAT (I110)
IF (IN.NE.0) THEN
   DO N=1,POLY_ORD
      IC(N)=IN/(10**-(POLY_ORD-N))
      IC(N)=IC(N)-10*INT(IC(N)/10)
   ENDDO
ENDIF
IN=O
DO N=1,POLY ORD
   IN=IN+IC(N)
ENDDO
IF (IN.EQ.0) IC(POLY_ORD)=1
C
RETURN
END

SUBROUTINE FEEDBACK(POLY_ORD,FEED)
C This subroutine determines the feedback path for a PSN
C
IMPLICIT NONE
INTEGER I,POLY_ORD,FEED(10)
C
DO I=1,10
   FEED(I)=0
ENDDO
C
IF (POLY_ORD.EQ.2) THEN ! Degree 2
   Primitive Polynomial 
   FEED(1)=1
ENDIF
IF (POLY_ORD.EQ.3) THEN ! Degree 3
   Primitive Polynomial 
   FEED(1)=1
ENDIF
IF (POLY_ORD.EQ.4) THEN ! Degree 4
   Primitive Polynomial 
   FEED(1)=1
ENDIF
IF (POLY_ORD.EQ.5) THEN ! Degree 5
   Primitive Polynomial 
   FEED(2)=1
ENDIF
IF (POLY_ORD.EQ.6) THEN ! Degree 6
   Primitive Polynomial
FEED(1)=1
ENDIF
IF (POLY_ORD.EQ.7) THEN ! Degree 7
  Primitive Polynomial
  FEED(3)=1
ENDIF
IF (POLY_ORD.EQ.8) THEN ! Degree 8
  Primitive Polynomial
  FEED(2)=1
  FEED(3)=1
  FEED(4)=1
ENDIF
IF (POLY_ORD.EQ.9) THEN ! Degree 9
  Primitive Polynomial
  FEED(4)=1
ENDIF
IF (POLY_ORD.EQ.10) THEN ! Degree 10
  Primitive Polynomial
  FEED(3)=1
ENDIF
FEED(POLY_ORD)=1
RETURN
END
SUBROUTINE PLTSUB(NPTS, IDTYPE, SX, DX, DATA, IPOS, ILN, IZL, IAXIS, IMNMX, ISCL, IXST, IYST, XMN, XMAX, YMIN, YMAX)

C THIS SUBROUTINE PLOTS A DATA FILE.
C USES SUBROUTINES SCALE, AXIS, ENTRY, TERM, PLOT, QIOB
C REQUIRES
C NPTS=NUMBER OF DATA POINTS
C IDTYPE=DATA TYPE
C 1=Y ONLY DATA
C 2=ALTERNATING X & Y DATA
C SX=STARTING X VALUE (SHOULD BE 0.0 IF IDTYPE=2)
C DX=X INCREMENT (SHOULD BE 0.0 IF IDTYPE=2)
C DATA=THE ARRAY CONTAINING THE Y OR ALTERNATING X & Y VALUES
C IPOS=NUMBER INDICATING THE DESIRED PLOT POSITION
C 0=WHOLE SCREEN
C 1=UPPER HALF
C 2=LOWER HALF
C ILN=1 FOR LINE PLOT, ILN=-1 FOR POINT PLOT
C IZL=0 FOR NO ZERO LINE, IZL=1 FOR ZERO LINE
C IAXIS=0 FOR NO AXES OR LABELS, IAXIS=1 FOR AXES AND LABELS PLOTTED
C IMNMX=0 FOR NO MIN AND MAX, IMNMX=1 FOR MIN AND MAX VALUES PLOTTED
C ISCL=SCALING TYPE
C 1=AUTOSCALE BOTH X & Y
C 2=AUTOSCALE X & SEMI-AUTOSCALE Y, REQUIRES YMIN & YMAX
C 3=AUTOSCALE Y & SEMI-AUTOSCALE X, REQUIRES XMIN & XMAX
C 4=SEMI-AUTOSCALE BOTH X & Y, REQUIRES XMIN, XMAX, YMIN, YMAX
C 5=NO AUTOSCALING, REQUIRES XMIN, XMAX, YMIN, YMAX
C IXST=X SCALE TYPE (1=LINEAR, 2=LOG)
C IYST=Y SCALE EXTREMES (MUST BE PASSED AS VARIABLES)
C XMN & XMAX=X SCALE EXTREMES (MUST BE PASSED AS VARIABLES)
C YMIN & YMAX=Y SCALE EXTREMES (MUST BE PASSED AS VARIABLES)
C L. D. LUKER 7/24/80 MOD. 10/14/83
C M. D. PATZ 11/25/86 MODIFIED FOR TEK 4100 SERIES TERMINAL

DIMENSION DATA(1)
BYTE NAME(30), ILPA(30), ITYP(2)
DATA NAME/30*' ']/
XMN=XMN
XMX=XMX
YMN=YMIN
YMX=YMAX

D TYPE *, 'PLTSUB', NPTS, IDTYPE, SX, DX, DATA(1), DATA(2), IPOS, ILN, IZL
D TYPE *, ISCL, IXST, IYST, XMN, XMX, YMIN, YMAX
TXMIN=85.
TXMAX=980.
TYMIN=65.
TYMAX=720.
IF (IPOS.EQ.1) TYMIN=445
IF (IPOS.EQ.2) TYMAX=340
ITXMIN=TXMIN
ITXMAX=TXMAX
ITYMIN=TYMIN
ITYMAX=TYMAX
DXMIN = SX
DXMAX = SX + (NPTS - 1) * DX
IF (IDTYPE.EQ.1) GOTO 110
DXMIN = DATA(1)
DXMAX = DATA(1)
DO 110 I = 1, 2*NPTS, 2
DXMIN = MIN(DXMIN, DATA(I))
DXMAX = MAX(DXMAX, DATA(I))
110 CONTINUE
IF (ISCL.EQ.1 .OR. ISCL.EQ.2) GOTO 120
DXMIN = AMAX1(XMN, DXMIN)
DXMAX = AMIN1(XMX, DXMAX)
120 IF (ISCL.GT.2) GOTO 130
XMN = DXMIN
XMX = DXMAX
IF (IXST.EQ.2 .AND. XMN.EQ.0.0) XMN = MAX(DX, .01*XMX)
130 IF (ISCL.NE.5) CALL SCALE(IYST, XMN, XMX, XMIN, XMAX, NXDIV)
JSTT = 2
JSTP = NPTS+2
IF (IDTYPE.EQ.2) GOTO 135
JSTT = NPTS*(DXMIN-SX)/((NPTS-1)*DX)+1
JSTP = NPTS*(DXMAX-SX)/((NPTS-1)*DX)
135 CONTINUE
D TYPE *, 'DXMIN=', DXMIN, 'DXMAX=', DXMAX, 'JSTT=', JSTT, 'JSTP=', JSTP
DYMIN = 1.0E+36
DYMAX = -1.0E+36
DO 150 IX = JSTT, JSTP, IDTYPE
IF (DATA(IX).GE.DYMIN) GOTO 140
PYMIN = SX + (IX-1)*DX
IF (IDTYPE.EQ.2) PYMIN = DATA(IX-1)
140 IF (DATA(IX).LE.DYMAX) GOTO 150
PYMAX = SX + (IX-1)*DX
IF (IDTYPE.EQ.2) PYMAX = DATA(IX-1)
150 CONTINUE
IF (ISCL.NE.1 .AND. ISCL.NE.3) GOTO 160
YMN = DYMIN
YMX = DYMAX
160 IF (ISCL.NE.5) CALL SCALE(IYST, YMN, YMX, YMIN, YMAX, NYDIV)
170 IF (IAXIS.EQ.1) CALL AXIS(XMIN, XMAX, NXDIV, TXMIN, TXMAX, YMIN, YMAX, 1
YMX, YMX, YMN, YMN, NXDIV, TYMIN, TYMAX, IZL, IXST, IYST)
XRANG = XMAX - XMIN
TXRANG = TXMAX - TXMIN
YRANG = YMAX - YMIN
TYRANG = TYMAX - TYMIN
XP = SX + (JSTT-1)*DX
IF (IDTYPE.EQ.2) XP = DATA(JSTT-1)
IF (IXST.EQ.2 .AND. XP.EQ.0.0) XP = XM0 * 0.0001
IF (IXST.EQ.2) XP = ALOG10(XP)
IX = ((XP-XMIN)*TXRANG/(XRANG)) + TXMIN
YP = DATA(JSTT)
IF (IXST.EQ.2) YP = ALOG10(YP)
IY = ((YP-YMIN)*TYRANG/(YRANG)) + TYMIN
IF (IX.GE.ITXMIN .AND. IX.LE.ITXMAX) GOTO 180
IPLFL=1
GOTO 200
180 IF (IY.GE.ITYMIN .AND. IY.LE.ITYMAX) GOTO 190
IPLFL=1
GOTO 200
190 CALL TPLOT(IX,IY,0)
IPLFL=0
200 DO 230 I1=JSTT,JSTP,IDTYPE
XP=DXMIN+(I1-JSTT)*DX
IF (IDTYPE.EQ.2) XP=DATA(I1-1)
IF (IXST.EQ.2 .AND. XP.EQ.0) XP=XMX*0.0001
IF (IXST.EQ.2) XP=ALOG10(XP)
IX=((XP-XMIN)*TXRANG/(XRANG))+TXMIN
YP=DATA(I1)
IF (IYST.EQ.2) YP=ALOG10(YP)
IY=((YP-YMIN)*TYRANG/(YRANG))+TYMIN
IF (IX.LT.ITXMIN .OR. IX.GT.ITXMAX) GOTO 205
IF (IY.GE.ITYMIN .AND. IY.LE.ITYMAX) GOTO 210
205 IPLFL=1
GOTO 230
210 IF (IPLFL.EQ.0) GOTO 220
IPLFL=0
CALL TPLOT(IX,IY,0)
GOTO 230
220 CALL TPLOT(IX,IY,ILN)
230 CONTINUE
CALL TPLOT(ITXMIN-50,ITYMIN-57,0)
CALL TERM(2,0)
IF (IMNMX.EQ.1)
WRITE(6,240)27,76,84,67,53,PYMIN,DYMIN,PYMAX,DYMAX
CALL TPLOT(ITXMIN+50,ITYMAX+17,0)
CALL TERM(2,0)
RETURN
C FORMAT STATEMENTS
240 FORMAT(+'\+',5A1,\'MIN(\',1PE10.3,\',\',1PE10.3,\')',
1\' MAX(\',1PE10.3,\',\',1PE10.3,\')')
END

SUBROUTINE SCALE(ITYP,DMN,DMX,SMN,SMX,NDIV)
C "SCALE" DETERMINES AN APPROPRIATE SCALE WITH NEAT LABELS
C AT MULTIPLES OF 1, 2, OR 5.
C REQUIRES:
C ITYP=SCALE TYPE (!=LINEAR, 2=LOG)
C DMN=MIN DATA VALUE
C DMX=MAX DATA VALUE
C RETURNS:
C SMN=MIN SCALE VALUE
C SMX=MAX SCALE VALUE
C NDIV=NUMBER OF SCALE DIVISIONS
C L. D. LUKER 8/15/80 MOD. 10/14/83
C REVISED BY TINA RUGGIERO
D TYPE *,\'SCALE \', ITYP=\',ITYP, DMN=\',DMN, DMX=\',DMX
IF(IHTP .EQ. 1)GO TO 110
IF (DMN.EQ.0) DMN=DMX*.0.01
SMIN=INT(ALOG10(DMN))
SMAX=INT(ALOG10(DMX)+.999999)
NDIV=INT(SMAX-SMIN)
GO TO 190

110 TEMP1=(DMX-DMN)/6.
TEMP2=10.**(ZINT(LOG10(TEMP1)))
TEMP3=TEMP1/TEMP2
IF (TEMP3.GT.2.) GOTO 120
IF (TEMP3.EQ.1.) GOTO 140
TEMP2=2.*TEMP2
GOTO 140

120 IF (TEMP3.GT.5.) GOTO 130
TEMP2=5.*TEMP2
GOTO 140

130 TEMP2=10.*TEMP2

140 SMIN=ZINT(DMN/TEMP2)
SMIN=TEMP2*(SMIN+.0.2.)

150 IF (((SMIN-DMN)/(DMX-DMN)).LE.1E-5) GOTO 160
SMIN=SMIN-TEMP2
GOTO 150

160 SMAX=ZINT(DMX/TEMP2)
SMAX=TEMP2*(SMAX-.2.)

170 IF (((DMX-SMAX)/(DMX-DMN)).LE.1E-5) GOTO 180
SMAX=SMAX+TEMP2
GOTO 170

180 NDIV=NINT((SMAX-SMIN)/TEMP2)

190 CONTINUE
SMN=SNGL(SMIN)
SMX=SNGL(SMAX)

D TYPE *, 'SCALE ', ', SMN=', SMN, ', SMX=', SMX, ', NDIV=', NDIV
RETURN
END

SUBROUTINE AXIS(XMN, XMX, NXDIV, TXMN, TXMX, YMN, YMX, NYDIV,
1TYMN, TMYX, TIZL, IXST, IYST)

C DRAWS AND LABELS THE X & Y AXES
C 
C REQUIRES:
C XMN=MIN X VAL
C XMX=MAX X VAL
C NXDIV=NUMBER OF X SCALE DIVISIONS
C TXMN=TERMINAL X VAL CORRESPONDING TO MIN X VAL
C TXMX=TERMINAL X VAL CORRESPONDING TO MAX X VAL
C ALSO ALL THE ABOVE FOR Y
C TIZL=0 DON'T DRAW ZERO LINE , TIZL=1 DRAW ZERO LINE
C IXST=X SCALE TYPE (1=LINEAR, 2=LOG)
C IYST=Y " "
C L. D. LUKER 8/15/80 MOD. 1/10/83
C 
C REVISED BY TINA RUGGIERO

DOUBLE PRECISION DP
TINT(Z)=ANINT(Z-.499999)
SCLFCT(VMN, VMX)=10.**(3*TINT((TINT(ALOG10(
1MAX(ABS(VMN), ABS(VMX))+ABS((VMX-VMN)/1E4))))/3.))
NFMT(VAL,FMX) = MIN1(FMX - INT(1. + AL0G10(ABS(VAL))), FMX)
D TYPE *, 'AXIS=', XMIN, XMX, NXDIV, TXMN, TXMX, YMIN, YMX, NYDIV, TYMN, TYMX
D TYPE *, IZL, IXST, IYST
FMX = 3.
C DO C, X AXIS

T1 = SCLFCT(XMN, XMX)
IF (IXST.EQ.2) T1 = 10. ** (3. * AINT(XMX/3.))
X1 = XMN/T1
XDIV = (XMX - XMN)/(2*NXDIV)
XDX = (TXMX - TXMN)/(2*NXDIV)
IX = TXMN
IY = TYMN
CALL TPL0T(IX-38, IY-30, 0)
N=3
IF (ABS(X1).GT.00001) N = NFMT(X1, FMX)
CALL TERM(2, 0)
IF (IXST .EQ. 1) GO TO 230

200
ANS = (10. ** XMN)/T1
IF (ABS(ANS).GT.00001) N = NFMT(ANS, FMX)
WRITE(6,330)27,76,84,53,ANS
DO 220 H = INT(XMN), INT(XMX-1)
DO 210 DP = 10. ** H, 1.0001 * 10. ** (H+1), 10. ** H
CALL TPL0T(IX, IY, 0)
IX = TXMN + (DL0G10(DP) - XMN)/(XMX - XMN) * (TXMX - TXMN)
CALL TPL0T(IX, IY, 1)
CALL TPL0T(IX, IY+5, 1)
210 CONTINUE
CALL TPL0T(IX, IY+10, 1)
CALL TPL0T(IX-38, IY-30, 0)
CALL TERM(2, 0)
ANS = 10. ** (H+1)/T1
IF (ABS(ANS).GT.00001) N = NFMT(ANS, FMX)
WRITE(6,330)27,76,84,53,ANS
220 CONTINUE
GO TO 250

230
WRITE(6,330)27,76,84,53,X1
DO 240 I = 1, NXDIV+2
X1 = (XMN+I*XDIV)/T1
IF (ABS(X1).LT..1) X1 = 0.
IX = TXMN + (I-1) * XDX
CALL TPL0T(IX, IY, 0)
IX = TXMN + I * XDX
CALL TPL0T(IX, IY, 1)
CALL TPL0T(IX, IY+5, 1)
IF (I/2 .NE. AINT(I/2.)) GOTO 240
CALL TPL0T(IX, IY+10, 1)
CALL TPL0T(IX-38, IY-30, 0)
N = 3
IF (ABS(X1).GT.00001) N = NFMT(X1, FMX)
CALL TERM(2, 0)
WRITE(6,330)27,76,84,53,X1
240 CONTINUE
250 CALL TPL0T(IX-62, IY-57, 0)
CALL TERM(2, 0)
IT1 = ANINT(AL0G10(T1))
WRITE(6,340)27,76,84,55,IT1

C DO Y AXIS
T1=SCFCT(YMN, YMX)
IF (IYST.EQ.2) T1=10.**(3.*AINT(YMX/3.))
Y1=YMNT1
YDIV=(YMXYMN)/(2*NYDIV)
YDX=(TYMX-TYMN)/(2*NYDIV)
IY=TYMN
IX=TXMN
CALL TPLOT(IX-76, IY-5, 0)
N=3
IF (ABS(Y1).GT.0.00001) N=NFMT(Y1, FMX)
CALL TERM(2,0)
IF (IYST.EQ.1) GO TO 290
260 ANS=(10.**YMNT1)
IF (ABS(ANS).GT.0.00001) N=NFMT(ANS, FMX)
WRITE(6,330)27,76,84,53, ANS
DO 280 H=INT(YMN),INT(YMX-1)
DO 270 DP=10.**H,1.0001*10.**(H+1),10.**H
CALL TPLOT(IX, IY, 0)
IY=TYMN+(DLOG10(DP)-YMNTYMXTMYN)/(TYMX-TYMN)
CALL TPLOT(IX, IY, 1)
CALL TPLOT(IX+5, IY, 1)
270 CONTINUE
CALL TPLOT(IX+10, IY, 1)
CALL TPLOT(IX-76, IY+11, 0)
CALL TERM(2,0)
ANS=10.**(H+1)/T1
IF (ABS(ANS).GT.0.00001) N=NFMT(ANS, FMX)
WRITE(6,330)27,76,84,53, ANS
280 CONTINUE
GO TO 310
290 WRITE(6,330)27,76,84,53, Y1
DO 300 I=1, NYDIV+2
Y1=(YMNIYDIV)/T1
IF (ABS(Y1).LT.1) Y1=0.
IY=TYMN+(I-1)*YDX
CALL TPLOT(IX, IY, 0)
IY=TYMN+I*YDX
CALL TPLOT(IX, IY, 1)
CALL TPLOT(IX, IY, 1)
IF (I/2.NE.AINT(I/2.)) GO TO 300
CALL TPLOT(IX+10, IY, 1)
CALL TPLOT(IX-76, IY-11, 0)
N=3
IF (ABS(Y1).GT.0.00001) N=NFMT(Y1, FMX)
CALL TERM(2,0)
WRITE(6,330)27,76,84,53, Y1
300 CONTINUE
310 CALL TPLOT(IX-83, IY+19, 0)
CALL TERM(2,0)
IT1=ANINT(ALOG10(T1))
WRITE(6,340)27,76,84,55,IT1
C FINISH UP
ITXMN=TXMN
ITXMX=TXMX
ITYMN=TYMN
ITYMX=TYMX

CALL TPL() (ITXMN,ITYMX,O)
CALL TPL() (ITXMX,ITYMX,1)
CALL TPL() (ITXMX,ITYMN,1)

IF (IZL.EQ.0) GOTO 320
IF (YMN*YMX.GT.O) GOTO 320
IYZERO=(-YMN)*(TYMX-TYMN)/(YMX-YMN)+TYMN
CALL TPL() (ITXMN,IYZERO,0)
CALL TPL() (ITXMX,IYZERO,1)

320 RETURN

C FORMAT STATEMENTS
330 FORMAT(’+’,4A1,F5.<N>)
340 FORMAT(’+ ’,4A1,’(*E’,13,’)’) END

SUBROUTINE ENTRY
C THIS SUBROUTINE PRINTS A HEADING (CALL ENTRY), AND ALLOWS
C ENTRY OF EITHER INTEGER (CALL IENT(IVAL)) OR REAL (CALL RENT(RVAL))
C VALUES WHERE THE DEFAULT IS NO CHANGE OF CURRENT VALUES.
C THE CALLS SHOULD BE PRECEDED BY WRITE STATEMENTS LABELING THE ITEM.
C L. D. LUKER 3/5/79 MOD 6/10/82
CALL ERRSET(64,,.FALSE.,,.FALSE.,,) ! STOPS STD. ERROR MESS.
WRITE(6,110)
110 FORMAT(/T10, ’ITEM’,T45,’CURRENT VALUE’,T62,’NEW VALUE’/)
RETURN

C ENTRY IENT(IVAL)
120 WRITE(6,130)IVAL
130 FORMAT(’+’,T47,I7,T61,’ ’,$)
READ(5,140,ERR=150,END=230)LEN,ITEMP
140 FORMAT(Q,112)
IF (LEN.GT.O) IVAL=ITEMP
GOTO 170
150 WRITE(6,160)
160 FORMAT(T61,’ENTRY ERROR!’/)
GOTO 180
170 RETURN

C ENTRY RENT(RVAL)
180 WRITE(6,190)RVAL
190 FORMAT(’+’,T45,1PG13.6,T61,’ ’,$)
READ(5,200,ERR=210,END=230)LEN,RTEMP
200 FORMAT(Q,F15.O)
IF (LEN.GT.0) RVAL=RTEMP
GOTO 220
210 WRITE(6,160)
220 RETURN

C 230 WRITE(6,240) ! CTRL-Z EXIT
240 FORMAT(’/’ EXITING PROGRAM!’/) CALL EXIT
SUBROUTINE TERM (K,L)
C
C THIS SUBROUTINE WILL MANIPULATE THE TERMINAL--
C K=0, L=0 ERASE SCREEN
C K=1, L=0 COPY SCREEN
C K=2, L=0 RETURN TO ALPHA MODE
C K=3, L=0 PURGE THE QIO BUFFER
C K=B, L=C IMPLEMENT MULTIPLEXER
   WHERE B IS BOARD SELECT NUMBER 0-3
   WHERE C IS CONTROL NUMBER TERMINAL(1), A(2), B(3), C(4)
   (COMBINATIONS OF TERMINALS ARE ALLOWED)
C
C THIS PARTICULAR VERSION IS FOR USE WITH BUFFERED
C PLOTTING, AND EVERY CALL TO TERM WILL PURGE THE BUFFER.
C
BYTE IOUT(3)
C
I=2
IOUT(1)="33" !ESCAPE
C
IF (L.NE.0) GOTO 10
   KX=K+1
   GOTO (2,3,5,40),KX
1
   IOUT(2)="14" !CLEAR THE SCREEN
   GOTO 30
2
   IOUT(2)="27" !COPY THE SCREEN
   GOTO 30
3
   IOUT(1)="37" !RETURN TO ALPHA MODE
   I=1
   GOTO 30
10
   IOUT(2)=K+"60"
   IOUT(3)=2**(L-1)+"60"
   I=3
C
C PREPARE ASCII CHARACTERS FOR MUX BOARD AND CONTROL NUMBERS
10
   CALL QIOB (IOUT,I,ISW)
   AND PURGE THE BUFFER
   CALL QIOP (IDAT,O,ISW)
C
C PAUSE A MOMENT IF SCREEN IS BEING CLEARED
   IF ((K+L).NE.O) RETURN
   CALL WAIT (600,1,M)
RETURN
C
END

SUBROUTINE TPLOT (IX, IY, M)
C
C SUBROUTINE TO PLOT ON THE TEKTRONIX 4010 AND 613 DISPLAY
TERMINALS (AS CHOSEN IN "TERM" SUBROUTINE).

VALUES TO PLOT: IX,IY
MODES TO PLOT: M>0 (BRIGHT), M=0 (DARK), M<0 (POINT)
(THE SUBROUTINE USES QIOB.)
REMEMBER TO PURGE THE BUFFER WHEN DONE BY CALLING TERM.

BYTE IOUT(6)
INTEGER*2 JX,JY
JX=IX
JY=IY

TYPE *,IX,IY,M,JX,JY
I=0
IOUT(1)=000
IF (M.GT.0) GOTO 11

INITIAL PLOT, DARK PLOT, POINT PLOT--
10 I=I+1
IOUT(I)="35

ALL MODES--SEPARATE COORDINATES INTO HIGH- AND LOW-ORDER BYTES
11 I=I+1
IOUT(I)=JY/32+32
I=I+1
IOUT(I)=96+JY-32*(JY/32)
I=I+1
IOUT(I)=JX/32+32
I=I+1
IOUT(I)=64+JX-32*(JX/32)

IF (M.GE.0) GOTO 20
REINFORCE FOR POINT PLOT
12 I=I+1
IOUT(I)=IOUT(I-1)

EXECUTE QIO AND RETURN
20 CALL QIOB (IOUT,I,ISW)

RETURN
END

SUBROUTINE QIOB (IDAT,INUM,ISW)

SUBROUTINE TO EXECUTE BUFFERED QIO [QIOB] FOR PLOTTING.

THIS SUBROUTINE WILL BUFFER QIO REQUESTS UNTIL 128
CHARACTERS ARE ASSEMBLED, AND OUTPUT THEM AS A GROUP.
PLOTTING IS NORMALLY EFFECTED WITH SIX-CHARACTER QIO’S,
SO THIS METHOD COULD SAVE AS MUCH AS 95% IN OVERHEAD.
IDAT----CHARACTER ARRAY NAME
INUM----CHARACTER COUNT
ISW----DIRECTIVE STATUS WORD
[SAME AS QIO DSW’S EXCEPT ISW=2 MEANS BUFFER NOT PURGED]
C IMPLICIT INTEGER*4 (A-Z)
BYTE BUFFER(128), IDAT(INUM)
INTEGER*2 STATUS BLOCK(4)
CHARACTER TERMINAL*2 /'TT'/
DATA CHAN ASN/O/
EXTERNAL IO$_WRITEVBLK, IO$M_NOFORMAT
C
C NOTE THAT WHEN INITIALLY CALLED BY A PROGRAM, J IS 0.
C IN SUBSEQUENT CALLS, J IS THE BUFFER POINTER.
C IE: THIS IS IN NO WAY REENTRANT!
C FIRST, TRANSFER QIOPED CHARACTERS INTO BUFFER:
K=1
10 DO 100 I=K, INUM
   J=J+1
   BUFFER(J)=IDAT(I)
100 IF (J.EQ.128) GOTO 500 !PURGE BUFFER
   ISW=2
200 RETURN
C
C TO PURGE BUFFER:
ENTRY QIOP
I=INUM
C
ASSIGN AN I/O CHANNEL
500 CONTINUE
IF (CHAN ASN .EQ. 0) THEN
   STATUS = SYS$ASSIGN (TERMINAL, TERM_CHAN,,)
   IF (.NOT.STATUS) CALL LIB$STOP (%VAL(STATUS))
   CHAN ASN = 1
END IF
C
C QUEUE THE I/O
   FUNC_CODE = %LOC(IO$_WRITEVBLK).OR.%LOC(IO$M_NOFORMAT)
   STATUS = SYS$QIOW (%VAL(1), %VAL(TERM_CHAN), %VAL(FUNC_CODE),
                  1  STATUS_BLOCK,, STATUS_BLOCK,, %REF(BUFFER),
                  1  %VAL(J), %VAL(0),,)
   IF (.NOT.STATUS) CALL LIB$STOP (%VAL(STATUS))
C
   J=0
   ISW=STATUS
C
CHECK TO SEE THAT ALL INUM CHARACTERS WERE OUTPUT:
IF (I.EQ.INUM) GOTO 200
K=I+1
GOTO 10
C
END
REFERENCES


1976.

& Sons, 1982.


