AN INVESTIGATION OF ERROR CORRECTING TECHNIQUES FOR OMV AND AXAF

NASA GRANT: NAG8-104
QUARTERLY REPORT
NOVEMBER 16, 1991

SUBMITTED TO:

Mr. Reggie Inmon
Mrs. Lee Ann Thomas
EB33
NASA/MSFC
MSFC, AL 35812
205+544-3677

SUBMITTED BY:

Frank Ingels, Ph.D.
Principal Investigator

John Fryer
Research Assistant

Mississippi State University
Mississippi State, MS 39762
601+325-3912

AN INVESTIGATION OF ERROR CORRECTING TECHNIQUES FOR OMV AND AXAF

NASA GRANT: NAG8-104
QUARTERLY REPORT
NOVEMBER 16, 1991

SUBMITTED TO:

Mr. Reggie Inmon
Mrs. Lee Ann Thomas
EB33
NASA/MSFC
MSFC, AL 35812
205 + 544-3677

SUBMITTED BY:

Frank Ingels, Ph.D.
Principal Investigator

John Fryer
Research Assistant

Mississippi State University
Mississippi State, MS 39762
601 + 325-3912

QUARTERLY WORK STATEMENT: August 16, 1991 - November 16, 1991

During this period, a final copy of the report concerning "AN INVESTIGATION OF ERROR CORRECTING TECHNIQUES FOR OMV AND AXAF [TESTING OF ANA/NASA REED/SOLOMON ECC CHIPS]" has been mailed to Mrs. L. A. Thomas and Mr. Warner Miller. A copy of this report is attached as Appendix I.

A revised final report concerning "A PRELIMINARY REPORT ON PREDICTED FIELDS IN SHUTTLE CARGO BAY DUE TO WISP ANTENNA" is attached to this report as Appendix II.

Attempts to use the NEC code to solve some well known classical problems using patches has not been fruitful to date. The problem lies in successfully predicting the size of the plates required to achieve symmetrical results in the calculated values.

APPENDIX I

AN INVESTIGATION OF ERROR CORRECTING TECHNIQUES FOR OMV AND AXAF [TESTING OF AHA/NASA REED-SOLOMON ECC CHIPS]

NASA GRANT: NAG8-104
SEPTEMBER 28, 1991

SUBMITTED TO:
Mrs. Lee Ann Thomas
EB33
NASA/MSFC
MSFC,AL 35812
205+544-3662

SUBMITTED BY:
John N. Fryer and Ken Lawrence
Associate Investigators
Frank Ingels, Ph.D
Principal Investigator
Mississippi State University
Mississippi State, MS 39762
601+325-3912

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

AHA - Advanced Hardware Architectures, Inc.

BlkBegN - Block Begin Not, decoder control input (CMOS)

CLK - Clock

CORRECT - Decoder output, indicates correct data is available

DATA - 8 parallel TTL data lines

DataVldN - Data Valid Not, decoder input control

DUAL - Dual field selector, identifies ROM contents

ErrPattN - Error Pattern Not, decoder control input

FIFO - First In First Out data buffer for decoder

INC - Input Control for encoder

OUTC - Output Control from encoder

R/S - Reed Solomon code

SELECT - Selects encoder interleave depth

Status Byte - Count of the number of errors in a decoded block

SysRstN - System Reset Not, decoder input control

CHAPTER 1

INTRODUCTION

The original objectives of this project were to build a test system for the NASA 255/223 Reed/Solomon encoding/decoding chip set and circuit board. This test system was then to be interfaced with a convolutional system at Marshall to examine the performance of the concantinated codes. After considerable work, it was discovered that the convolutional system could not function as needed. This report documents the design, construction, and testing of the test apparatus for the R/S chip set. The approach taken was to verify the error correcting behavior of the chip set by injecting known error patterns onto data and observing the results. Error sequences were generated using pseudo-random number generator programs, with Poisson time distribution between errors, and Gaussian burst lengths. Sample means, variances, and number of un-correctable errors were calculated for each data set before testing.

The NASA 255/223 Reed Solomon encoder/decoder is a set of five Application Specific Integrated Circuits (ASICs) designed by Advanced Hardware Architectures and NASA's Microelectronics Research Center at the University of Idaho. An evaluation circuit board was provided to mount the chips for testing. The encoder is a single chip, with three control inputs and the eight data inputs. At the output there is a single control line and eight data lines. The decoder consists of the remaining 4 chips, using four control inputs and eight data lines. There are two

condition flags at the decoder output, along with the data lines. The evaluation board has two switches, which provide additional control signals. This report is organized into five sections. The first section describes the procedure required to operate this chip set. Implementing these steps and verifying the result is the objective of the test system.

The second section details many of the operating characteristics of the chip set and circuit board. This information was extracted from NASA's documentation for the Encoder Chip, Decoder Chip Set, and Decoder Board. Also helpful were 2 updated versions of the AHA system documentation and conversations with Dr. Maki and Don Wiseman of the University of Idaho.

The third section covers the peripheral hardware used to support the AHA circuit board. To provide flexibility a PC-AT was chosen as data source and system controller. Two parallel ports are providing the interface between computer and test apparatus.

The fourth section deals with software, describing the programs for generating data, analysis, and control. The procedure to modify this code is also described.

The final section contains sample test data (with plots), results for the R/S system, and program listings. These programs are available on disk, both compiled (*.exe) and in ASCII.

CHAPTER 2

OPERATING PROCEDURE

The following is the sequence of events necessary to use the chip set on the evaluation board.

2.A Encoder

The test program ENCODE.EXE exercises the encoder.

1 Power on, input off

2 Clock high

3 Select Interleaving depth, wait 1.5 microseconds.

4 Reset Encoder - Bring INC low, clock low/high 32 times, leave clock high and INC high.

5 Input first block of data (between 196 and 223 bytes). The length of this block sets the length of all subsequent blocks until a reset.

6 Stop data, bring INC low, and cycle clock low/high 32 times, leave clock high and INC high.

7 Goto 4

Data appears at output after a delay of 2 clock cycles. OUTC is simply

INC passed through along with the data. It is high when data is output and low when the 32 check bytes are output. It will be necessary to generate a Block Begin Not signal to tell the decoder where data blocks begin.

2.B Decoder

1 Power on, input off

2 Clock high

3 Reset Decoder - Bring SystRstN low for 8 clocks, high for at least 7 clocks.

4 Bring BlkBegN low for one clock cycle coincident with the first data byte input.

5 Clock in the rest of the code block.

6 Goto 4, repeat until buffers are empty.

Output is valid when DataVldN is low. If ErrPattN is high, the corrected data along with a status byte are output. The status byte is in place of the first check byte, the remaining 31 check bytes are set to zeros.

If ErrPattN is low, the detected error pattern is supplied at the data output lines.

CHAPTER 3

SYSTEM CHARACTERISTICS

This section details many of the operating characteristics of the chip set and circuit board. This information was extracted from NASA's documentation for the Encoder Chip, Decoder Chip Set, and Decoder Board. Also helpful were 2 updated versions of the AHA system documentation and conversations with Dr. Maki and Don Wiseman of AHA.

NOTE: - Always turn on power before applying an input signal.

3.A Encoder

3.A.1 Input Description

- All inputs are TTL compatible

DATA - TTL input Data byte.

Clock - Rising edge triggered, up to 10Mhz, data must be valid for 33ns before the clock rises and held for 10ns afterwards.

INC - Goes high with first data byte, stays high during data input.

- The number of clock cycles that this line is high determines block length. The decoder allows the block length to vary between 255 and 228 bytes, including the 32 check bytes.
- To change block length, you must reset the encoder and decoder.

- Don't forget to clock out the data in the decoder pipeline.

SELECT - Sets interleave depth, currently set to 1. Lines must be stable 1.5 us after "power up sequence" (reset) or change in interleave depth.

Timing - Clock can stop & restart, as long as the data flow remains synchronized with the clock.

- Clear encoder on start up: bring INC low, clock 32 times.
- There is a 2 clock delay for data through encoder.
- At end of each data block, stop data flow and hold INC low for 32 clocks to output check bytes.
- The INC signal can be used to trigger an oscilloscope on the first data byte or the first check byte.

3.A.2 Output Description

- All outputs are TTL compatible

DATA - Data and check bytes

OUTC - INC passed through with the data, goes high with first data byte out, low during check byte output. Logic is inverted from BlkBgnN, and does not provide the needed one clock strobe.

3.B Decoder

3.B.1 Input Description

- ErrPattN and SysRstN are CMOS only!
- BlkBegN - Goes low for one clock cycle at the beginning of each code block. The number of clock cycles between the first and second strobes of BlkBegN after reset determine the length of the code blocks.
- BlkBegN is a mixed CMOS/TTL signal

Clock - Separate from encoder clock. Same description as for encoder.

DATA - Data and check bytes

DUAL - Select dual or normal field (PROM's contain the field).
It is selected by the jumper labeled S1.

ErrPattN - Error Pattern Not **CMOS** Selects whether the corrected data or detected error pattern will be output.

- Low means error pattern and check bytes are output.
High means data (correct or not) and status byte are output.

SysRstN - System Reset Not **CMOS**

Timing - Reset to clear pipeline at startup or when changing block length. Bring SysRstN low for at least 8 clocks, then high for 7 clocks. Any data in the FIFO will be lost.

- BlkBgnN must go low for one clock cycle in conjunction with the first data byte of every block.

- There is a pipeline delay of about 3.5 code blocks between data entering the decoder and the corrected results coming out. The delay for a 255 byte block is 908 clocks. There does not appear to be any good way to calculate the delay.

3.B.2 Output Description

- All outputs are TTL compatible.

Correct - High indicates that the block of data being output was correctable.

- Low indicates that the data was not correctable, no corrections were made. Whatever was received will be passed straight through to the output without change.

- This Correct is valid when DataVldN is low.

DATA - Data and check bytes

DataVldN - Data Valid Not is low when data (correct or not) is being output. High for status byte and 31 padded blanks.

Status Byte - First byte output after data block, corresponds to first check byte in timing. Bit 7 = 0 means data valid, bit 7 = 1 means uncorrectable errors occurred. Bits 0-4 indicate the binary count of corrected errors, up to 16, with bit 4 the most significant. The remaining check bytes are all zero. If no errors occur, the status byte and 31 check bytes will all be zero.

Timing - The DataVldN signal can be used to trigger an oscilloscope on

the first data byte or the first check byte. Due to time variations of the program loops, the scope trace tends to jump around a lot. For some test runs all maskable interrupts were disabled, so the only way to stop the program is to reset the computer.

3.C Buffers

The buffers isolate the ASIC chips from the input and output sockets. Switch 2 jumpers the enable line on the 74LS244s to +5 volts or ground. Ground enables the buffers, +5 disables them. The 74LS244s have three-state output latches, but the control lines are not implemented. The buffers add two 18ns delays through the circuit. This does not change the required setup and hold times, but if the board is inserted into another synchronous circuit, this may slow the maximum clock speed of that circuit.

Notes: - The 74LS244 decoder output buffer is wired backwards on the circuit board. *It must not be connected.*

- Text on the block diagram in the circuit board documentation reads upside down. Pin one on the buffers and PROMs is to the top of the board.

3.D FIFO

The FIFO is 1K x 9, id number "IDT 7202SA35j" in a PLCC package. It is available from:

Integrated Device Technology, Inc.

4930 Corporate Drive, Suite 1

Huntsville, AL 35805 205-721-0211

Sales Engineer Steve Gary provided two of them at no charge. These are CMOS chips.

CHAPTER 4

PERIPHERAL SUPPORT FOR R/S BOARD

Figure 1 shows a block diagram of the experimental equipment including the computer, error injection, and R/S board. Function of the various subassemblies is discussed in this section.

4.A Computer

A PC-AT type computer was selected to be the controller and data source for this project. The machine used operated at 10 MHz, and had 640K bytes of RAM, a 20M byte hard disk, EGA monitor, an 80287 math coprocessor, and two parallel ports. The search for fast I/O ports turned up several boards, however none were designed for the speeds needed for this project. Examination showed that direct port control using assembly language and/or C was essential to obtaining maximum speed when downloading data. None of the available I/O boards indicated that they could operate faster than the built-in parallel ports operated via assembly language. The computer ports provide all timing and control signals. The process used to generate computer code to exercise the board via parallel ports is given in the computer section. A 2 Megabyte RAM disk was added during testing.

4.B Power

Power supply should be between 4.75 and 5.5 volts, and must not exceed -0.3 to 7.0 volts. Power must be supplied to the ASIC chips before signals are applied. If the power is cycled off/on, allow all capacitors to discharge and remove all inputs before turning back on. Failure to do so may lead to latchup and chip failure! Current to the board should not exceed 1.2 amperes.

On this test set, a 7805 voltage regulator steps down the PC's 12 volt 8 amp. (yellow wire) line to 5 volts for the AHA-NASA circuit. It also limits the current to no more than 1.5 amps. The purpose of the 7805 is to prevent the accidental connection of the 12 volt line (or any other voltage) from the unkeyed power socket to the board. If the 7805 is pushed beyond its limits (in excess of 1.5 amps until it reaches thermal overload), it can oscillate. Limiting the current to 1 amp or less using a fuse and placing a 0.1uF capacitor across the output prevents the conditions leading to oscillation. A switch can cut off the 12 supply to the 7805, removing the power to the R/S board. This is intended for use with the buffered input configuration only. DO NOT APPLY POWER TO THE BOARD WITH SIGNALS ALREADY PRESENT AT THE UNBUFFERED INPUTS. THIS MAY BE FATAL TO A CHIP. A 5.1 volt 1 watt zeiner diode is present to trap overvoltage surges.

The computers 5 volt 20 amp. (red wire) line supplies all of the other circuits used in the test bed. A 2 amp fast blow fuse and a cutout switch protect those circuits. A voltage monitor circuit has been designed to shut down the system if the supply voltage on the board should exceed the specified safe range. The circuit was

not used with the testbed, since the computer supply appeared to have good regulation.

4.C Parallel Ports

The parallel ports are generally known as Centronics ports. This is misleading, they are actually somebody's implementation of Epson's modification to Centronics modification to the Intel 8255A Programmable Peripheral Interface. Not all of the modes defined for the Interface are implemented. By experimenting with defined modes and writing to individual bits, the location and means of accessing the pins were obtained.

TABLE 4.1

Epson/Centronics Printer Port Pinout

pin#	Standard Definition	Direction
1	Strobe	Out
2	Data	"
3	"	"
4	"	"
5	"	"
6	"	"
7	"	"
8	"	"
9	"	"
10	AcknowledgeNot	In
11	Busy	In
12	Paper Empty	In
13	+Select	not defined
14	Auto Line Feed	Out
15	Error Not	In
16	Initialize Not	Out
17	-Select Not	Out
18-25	Ground	---

The port is logically divided into four 8 bit ports as follows

TABLE 4.2

Parallel Port Addresses

Port	PC/AT Address (Hex)		
	Video Card (lpt1)	Printer Card (lpt2)	Direction
Data	0X0278	0X0378	out/in
Status	0X0279	0X0379	in
Print control	0X027A	0X037A	out
Port control	0X027B	0X037B	---

The port modes are supposed to control direction and functional relations between the logical ports. In particular, in modes 1 or 2, data appearing on the output pins should automatically be accompanied by a strobe on pin one. The speed of the port was determined by writing a short assembly routine to dump alternating ones and zeros to the data lines, and measuring the resulting output signal with an oscilloscope. Without any handshaking overhead, the maximum rate achieved to date is about 600,000 bytes per second. This is close to the maximum rate that this particular I/O card can physically support, the signals are getting rounded and are not reaching their full amplitude. Nothing in the documentation appears to indicate that the port should not be capable of running at the computers bus speed (usually

8 Mhz), but your standard cheap I/O card may not be that good. The built-in strobe may not be reliable for a high speed clock. The strobe is by far the most rounded of all of the output signals. It seems this is intended to delay the strobe until the data setup requirements have been met. Simply putting a long RC timeconstant on the strobe provides the delay.

4.D Cables

The connector on the printer port is a 25 pin D shell. The current cable is a 36 wire ribbon. Data and ground line alternate, with all of the grounds joined at both ends. At higher frequencies it may be desirable to leave one end of all but one of the grounds disconnected. The port pins are currently assigned as follows:

TABLE 4.3

Ribbon Cable Connections

	LPT 2		LPT 1	
pin#	Function	Direction	Function	Direction
1	Not used	Out	Not used	Out
2-9	Data	Out	Error	Out
10	Not used	In	Not used	Out
11	"	"	"	"
12	"	"	"	"
13	"	"	"	"
14	Clock Out	Out	Clock	Out
15	Not used	Not defined	Not used	Not defined
16	Not used	Out	Not used	Out
17	INC	Out	INC	Out

4.E Error Injection System

The error signal is generated by the second parallel port on the PC. Data is Exclusive ORed with the error signal at the output of the encoder. Note that the data has a two clock latency passing through the encoder, so errors are injected onto data which entered the system two clock cycles before the error.

4.F Block Begin Not Generator and Clock Inversion

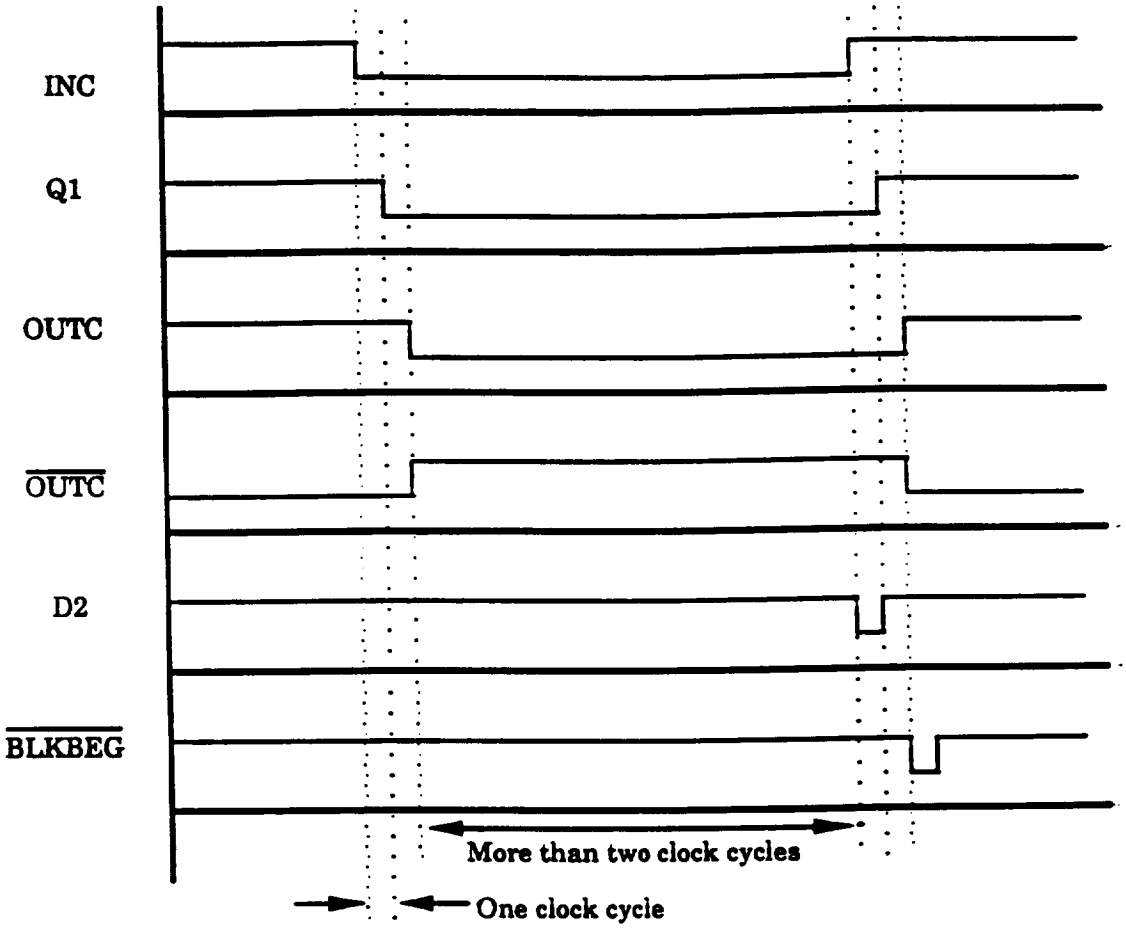
The control signal BlkBgnN is generated using the NAND of INC (delayed one clock) and OUTC. The signal is delayed one more clock, then applied to the decoder input. The timing and logic diagrams for generating BlkBgnN are illustrated in Figure 4.1. The clock for the Decoder is also inverted at this stage. This is to provide a delay (for setup time) between the application of the clock to the encoder and the decoder. This is known as a split phase clock.

4.G Miscellaneous Notes

All chips mount with the notch to the top of the board. The Block diagram is misleading. All of the ground pins are on the outside edge of the connectors. Don't try to use both the buffered and unbuffered inputs at once.

The encoder and decoder clocks are not common on the board.

CORRECT and bit 7 of the status word should indicate the same error condition. CORRECT is available during the entire data block, but if it indicates an uncorrectable error it will be low during the data output, high during the check bytes. To latch the check byte, pass the data through a bank of D flipflops, then



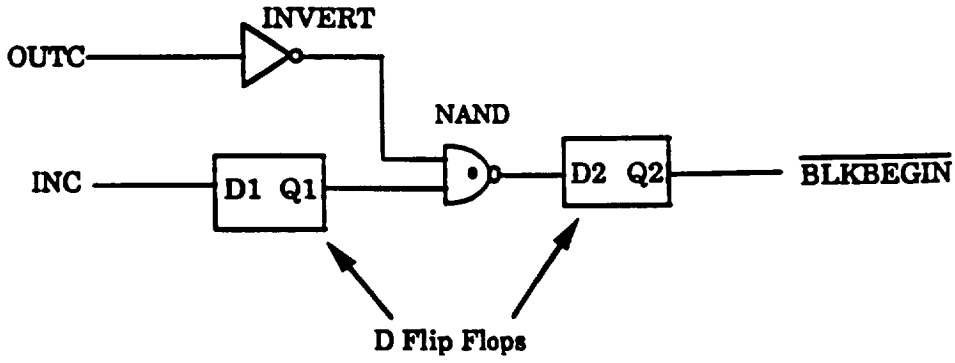


Figure 4.1 Block Begin Not Generator and Timing Diagram

AND the clock with DataVldN. One clock after the check byte is output DataVldN goes high, and the status byte will be on the flipflop output lines. The output of the AND signals that the check byte is on the D flipflop output terminals. The status word may be stored along with the data for a record of how well the system was performing. The count of corrected errors might be used as a measure of the performance improvement produced by other factors in the system such as addition of soft decisions or a convolutional code. The documentation does not indicate when ErrPattN can change.

Maximum rated clock speed is 10 Mhz, yielding an equivalent input bit rate of $(N-32)/(N) * 10^6 * 8$ where N is the number of bytes in a code block, $228 <= N <= 255$. A interesting problem occurred due to the fast response of these chips. The counter used to monitor uncorrectable errors was highly erratic. Adjusting the sensitivity reduced the magnitude of the miscount, but could not stabilize the counter. Connecting two counters in parallel did fix the problem. It seems that the rise time on the decoder output is so rapid that the counters had difficulty keeping up with the signal. Placing two counters in parallel supplied enough load that the signal was damped slightly. The two counters subsequently produced accurate and consistent results.

The signals injected into and produced by the chip set were monitored using a dual timebase oscilloscope. Using the external trigger on the block begin not or data valid not signals allows examination of any part of the waveform. High impedance

tuned oscilloscope leads were used to minimize loading effects on the circuit during testing.

The documentation indicates that the encoder can implement some form of interleaving. It does not provide details of how the interleaving is arranged or how it is to be undone. Apparently, only the check bytes are affected. The three interleaving select lines are tied low through 4700 ohm resistors, selecting an interleave depth of one. Further information has been requested several times, but has not been forthcoming.

Figure 4.2 shows the arrangement of the test apparatus for loopback testing of the Reed/Solomon board alone.

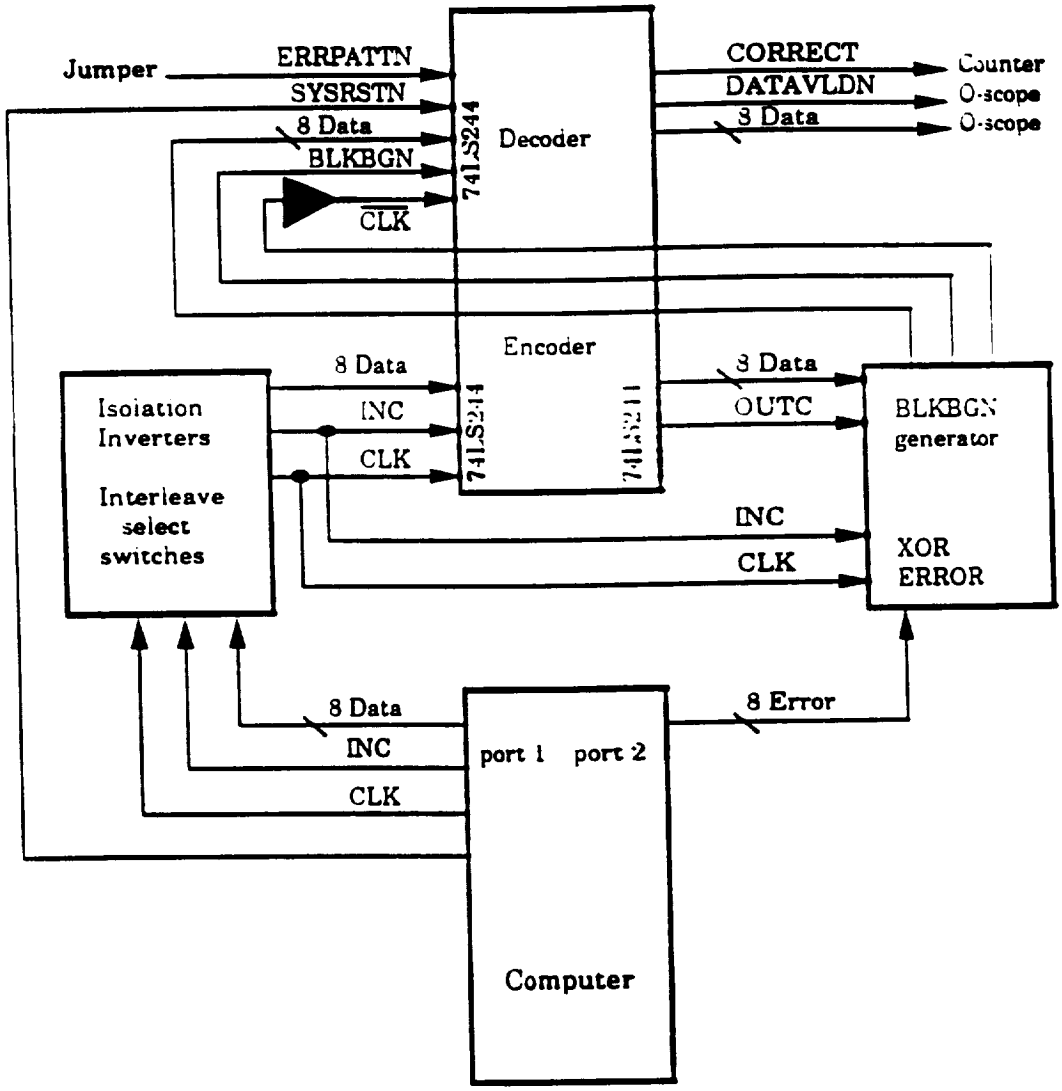


Figure 4.2

Test Configuration

CHAPTER 5

SOFTWARE

There are several distinct tests to be performed to satisfy the stated objectives of this work. For each of these tests, the statistical distribution, burst length, spacing, and other characteristics of the noise and resulting error patterns need to be specified. This chapter describes test procedures for the 255/223 Reed/Solomon encoder/decoder made by AHA for NASA. Each 255 byte code block (word) is currently arranged into 223 consecutive data bytes (8 bits/byte), followed by 32 consecutive check bytes. Programs were written in the Microsoft Quick C editor, compiled to an assembly language listing with Microsoft's C Compiler 5.1, optimized in assembly using a program editor, compiled to a .obj format using Microsoft's macro assembler MASM, and compiled to .exe format using the C compiler again. This involved process is only necessary for maximum speed on the output routines. The compiling steps have been largely automated using batch files. Ordinary C or Fortran code may be used to generate the data and place it into arrays or files. These assembly I/O routines can be inserted into C programs using the latest version of Microsoft QuickC. Some of the advantages of this method are:

-The QC (Quick C) editor checks syntax for you.

-The C5.1 Compiler supports very large data arrays and takes care of operating system bookkeeping.

-Some simple steps in assembly can vastly increase speed.

-Assembly language gives direct access to the bits of the ports.

Error patterns were applied to the all zeros data block and/or alternating ones/zeros byte data blocks. Testing was performed in two major sections, corresponding to the method used in generating the data and error vectors. Initial work was performed using "immediate" data, meaning that the data were part of the assembly language programs. This is the simplest and fastest implementation. Output was monitored using an oscilloscope triggering on the DatavalidN line. Both the data and the Correct lines were monitored. The Status byte was also examined and in all cases identified the number of correctable errors. This mode was principally used to debug the system. The second mode of testing was to generate large data files on disk, then dump the data to the output port.

The clock rate is typically about 225,000 Hz for the parallel immediate mode data (1.8 MHz if converted to serial data, this exceeds the data rate of the OMV system, but it is not difficult to slow down). Exact speeds are code dependent, but all were close to this rate. This corresponds to over 810 million bytes or 6.48 billion bits per hour.

5.A Immediate Mode Data Testing

The first testing was primarily aimed at debugging the computer interface with the R/S board. These same error vectors were applied to the board once the interface

was functioning properly. The data is part of the assembly language program, loaded into the CPU registers directly.

5.A.1 Single Bit Errors

Initial testing was performed by inserting a single bit error on bit zero of the third from last data byte of each code block (byte 221). The program ran in a continuous loop clocking out this error on the all zeros data block. Subsequently this error was moved to each bit location of that byte. This error pattern was applied for about 40 hours over a period of two weeks during initial testing. Approximately 6 hours of this time the system was closely monitored, the balance was to allow burn-in of the test system. No indication of incorrect decoding was ever observed. (Incorrect decoding is failure to correct errors, failure to identify uncorrectable errors, or insertion of additional errors.)

5.A.2 Byte Errors

Most of the testing has been performed using the case of every bit of every error vector being all ones. (ie. all error bytes = 11111111). First, the same byte as above was corrupted, then data bytes 219 and 221 were both corrupted. Then this pattern (bad byte, good byte, bad byte), was moved to the first data location. This pattern was used to examine the alignment of the DatavalidN and BlockbeginN timing signals. This pattern was applied for about 20 hours over several days. Again no faulty decoding was observed.

5.A.3 Block Errors

Next, blocks of data were corrupted, and the error location in the data block moved through the entire data block. This was implemented by using a set of nested loops and adjusting the loop counters. The error block length ranged from a single byte up to 17 consecutive bytes. (All error bytes for burst errors were 11111111, 17 consecutive error bytes is 17 x 8 = 136 consecutive ones). These processes ran for 25 hours. All cases of 16 or fewer errors were corrected. The case of 17 errors was properly identified as uncorrectable. Lastly, the extreme case of an entire data block was corrupted, clocking out all ones errors on all of the data and check bytes. The system correctly indicated an uncorrectable error.

5.B Data File Testing

A more sophisticated method is to read the necessary data from a hard disk file into RAM memory and then output the data from RAM to the encoder. Tests can be defined using various error distributions such as Gaussian, Poission, etc.

The next step in testing was to generate large error data files and run them through the system. Two methods are being used to generate these data files: First a Fortran program was used to generate Gaussian distributed random numbers. To generate about 14 errors per 255 byte code block, find the area corresponding to 14/255 = 0.0549 under the standard normal curve. This corresponds to the X axis value -1.6, so each Gaussian number is interpreted as an all ones error byte if the random number is less than or equal to -1.6, or an all zeros error byte if the number is greater than -1.6. The decimal equivalents 255 or 0 were stored in a file to be read

and output to the encoder. A file containing 25,500 of these error bytes was created and is currently running on the system. The number of errors per code block can be changed by finding the X axis point corresponding to the area under the standard normal curve, then re-evaluating the decimal random numbers.

The statistical analysis software SAS was to have been used to generate data files. Unfortunately, data format information for the storage of eight bit unsigned integer data was not readily available. A Fortran program generating Gaussian random data has been implemented following the appendix on random numbers in Orfanidus' Signal Processing book. The data is random burst errors (all ones) with a Gaussian burst length (ie. number of bytes/burst) and Poisson time distribution between bursts.

The data is analyzed before the tests so that the number and length of correctable, uncorrectable, and undetectable errors is known for each data file. Counting the results out of the decoder provides collaboration. These same error files could be injected into the serial data stream for the concantinated codes.

Speed problems arise when using the disk file data files. First, the clock out is stopped every time the disk is accessed for more data. The clock rate while data is being transferred to the encoder is essentially unchanged, but when memory is empty, more data must be read in from disk. The rapid data output rate leads to constant disk access. Implementation of a 2 megabyte RAM disk to supplement the hard disk improved data access and reduce wear on the hard drive. Data compression has also been introduced by storing only the length of error bursts and the number of clear bytes between error bursts. For the best case, this represents

each 255 byte block in two eight bit numbers, for a compression of over one hundred to one. The data are stored in two files, along with a discriptor file containing the number of terms to read from each data file per memory update.

CHAPTER 6

RESULTS AND RECOMMENDATIONS

Histograms, sample means, and variances of each data set were generated to document the characteristics of test data. The following three tests are representative samples of the general types of data used and results obtained. The results show that the system is working exactly as predicted. For example, in test 12, there were 42,519 code blocks (86,738,760 bits) known to be uncorrectable out of 200,892 code blocks (409,819,680 bits) in the test run. For three successive runs the experimental count showed exactly 42,519 uncorrectable code blocks. Observation of the correctable errors into and out of the decoder (on an oscilloscope) showed that these errors were in fact corrected. This means that the errors contaminating 157,481 code blocks were in fact identified and corrected.

In the experimental data some code blocks have multiple error bursts, while others have no errors. This causes a difference in the number of error bursts and the number of code blocks in the tests.

The histogram data files for the Gaussian Burst lengths are included in the Appendices. Histogram plots of both Gaussian Burst length and Poisson spacing data are also included. Data files for Poisson contain several thousand entries, so they were not included.

6.A Sample Tests

The following three tests are representative samples of the type of experiments performed and corresponding data.

Test 4

Gaussian Burst: Mean = 1, Variance = 7 Poisson Spacing: Lambda = 128

Number of:

error bursts	100,000	
code blocks	99,996	(25,498,980 bytes, or 203,991,840 bits)
correctable code blocks	99,281	(25,316,655 bytes, or 202,533,240 bits)
uncorrectable code blocks	715	(182,325 bytes, or 1,458,600 bits)

This is 0.71% uncorrectable errors. The test was repeated 3 times. The number of measured uncorrectable blocks was 715 for each run, matching exactly the known number of uncorrectable errors.

Test 9

Gaussian Burst: Mean = 8, Variance = 4 Poisson Spacing: Lambda = 128

Number of:

error bursts	200,000	
code blocks	104,284	(26,592,420 bytes, or 212,739,360 bits)
correctable code blocks	65,740	(16,763,700 bytes, or 134,109,600 bits)
uncorrectable code blocks	38,554	(308,432 bytes, or 2,467,456 bits)

This is 36.9% uncorrectable errors. The test was repeated 3 times. The number of measured uncorrectable blocks was 38,554 for each run, matching exactly the known number of uncorrectable errors.

Test 12

Gaussian Burst: Mean = 10, Variance = 8 Poisson Spacing: Lambda = 255

Number of:

error bursts	200,000	
code blocks	200,892	(51,227,460 bytes, or 409,819,680 bits)
correctable code blocks	158,373	(40,385,115 bytes, or 323,080,920 bits)
uncorrectable code blocks	42,519	(10,842,345 bytes, or 86,738,760 bits)

This is 21.1% uncorrectable errors. The test was repeated 3 times. The number of measured uncorrectable blocks was 42,519 for each run, matching exactly the known number of uncorrectable errors.

6.B Problems & Suggestions

The following recommendations break down into two parts: modifications in the chip set itself, and information for the user. The 'problems' described are things which make the encoder/decoder set more difficult to use. While a given user can adapt his/her system to deal with these characteristics, the chip set could be easier to use.

1 Problem: Verifying the correct function of the encoder/decoder independent of the communication system in which it is embedded is difficult or impossible. When building a communications system, it is a good idea to test each subsystem independently. It may also be useful to be able to request such a self-test of an orbiting system.

Suggestion: Facilitate the performance of a loop-back self-test in the chip set.

One way to implement a self-test is to feedback the output of the encoder into the input of the decoder. This requires generation of various control signals. The following two features would make it easier to implement such a self-test.

* Include a Block Begin Not generator on the encoder chip. A simple circuit to generate this signal is shown in Figure 4.1.
* Supply a delayed clock signal at the output of the encoder. This is to allow setup time for data and control input to the decoder.

Another way tests of the encoder and decoder might be performed is to provide an input sequence to each from ROM, then compare the encoded/decoded data with known good results also stored in ROM.

2 Problem: The system in which the chip set is embedded must operate the encoder and decoder by providing timing and control signals.

2.A The user must reset both encoder and decoder manually. The present system also gives no indication of when a successful reset has occurred.

2.B The system using the chip set must provide the encoder and decoder control signals INC and Block Begin Not. Generating control signals to tell the chip set when every data block begins and ends demands additional work from the communication system.

2.C The encoder input data stream is not continuous, being interrupted every code block for the time required to clock out the check bytes. Halting the data flow into the encoder while the check bytes are clocked out requires that the data source count out each data block individually.

Suggestion: Make the code/decoder chip set as independent as possible of the surrounding system. This might be accomplished by modeling the interface logic on a standard communications protocol such as RS-232C. The data source asserts Ready-To-Send (RTS), the receiver acknowledges with Clear-To-Send (CTS). When the receiver needs to stop accepting data, it negates CTS. The internal function of the encoder/decoder would be essentially transparent to the user.

2.A A simple sequential circuit could execute a system reset in response to a request. If each chip provided a reset signal, a logic circuit could generate CTS.

2.B INC could be replaced with two variable length counters, one for the number of data bytes, and the second for the number of check bytes. These count lengths could be selected using jumpers for standard code lengths. After a system reset, the first clock would correspond the first data byte.

The decoder block length should also be set using counters as described above for the encoder. Once the data stream is initialized, the encoder would know where each new block begins without having to be told.

2.C Making the input data flow continuous requires that the data be clocked out of the encoder faster than it is clocked in. If the input data is buffered in a FIFO, the input could continue uninterrupted while the check bytes are output. The necessary clock rates are related by the factor Code-Block-Length-Out/Data-Block-Length-In, 255/223 (=1.143497..., a continued fraction) for the nominal code. Multiplying by 255 and then dividing by 223 produces very high intermediate frequencies. At a 10 MHz output rate, 2230 MHz intermediate clock rate would be necessary. The input data rate would be 223/255 * 10 MHz, or about 8.745 MHz. The integer 223 is prime, so the ratio can not be reduced to smaller multipliers. The ratio might be adjusted by adding bytes between code blocks (sync. patterns perhaps), by shortening the data blocks, or some combination of the two. Perhaps a single low frequency reference could be multiplied by 255 and 223 separately to produce the needed clocks.

At 10 MHz, even a small percent error would quickly lead to FIFO overflow or underflow. Allowing one of the clocks to vary its period slightly (within bounds determined by the FIFO size) could get around this problem.

If the above modifications were implemented (except 2.C), the following input and output signals would be needed:

Encoder

Input Signals	Output Signals
Data	Data
Clock	Delayed Clock
RTS (Request-To-Send)	OutC
RR (Request Reset)	Block Begin Not
	CTS (Clear-To-Send)
	RV (Reset Valid)

Decoder

Data	Data
Clock	CTS (Clear-To-Send)
RTS (Request-To-Send)	RV (Reset Valid)
RR (Request Reset)	Data Valid Not
	Correct

3 Provide up-to-date documentation to the end user. A lot of time could have been saved had the latest version of the documentation accompanied the hardware. Much time was spent deciphering inconsistencies and errors that had already been fixed in later versions of the documentation. Also, it was known that there was a flaw in the circuit board output buffer. It should be discussed in an errata appendix to the circuit board documentation.

4 Provide applications notes (tutorial) on how the system works and interfacing requirements such as timing, reset sequences, and TTL to CMOS voltage interfacing. It is likely that many of the people wanting to use these chips will be communications oriented, not digital circuit designers. Some documentation should

be directed to general system design and construction requirements affecting incorporation of these chips into other systems.

5 Provide documentation on interleaving and deinterleaving, perhaps including advantages/disadvantages, implementation techniques and a sample circuit to work with this system.

CHAPTER 7

APPENDIX

APPENDIX 7.A

Experimental Data

Test data were designed to subject the system to a variety of conditions. In general the data were designed to provide a large number of code blocks with correctable errors, either multiple short bursts scattered in each code block, or a few long bursts more widely spaced. The following table shows distributions of some of the tests run on the system.

TABLE 7.1 Test Data Statistical Distributions

Burst Length		Burst Spacing
Mean (Gaussian)	Variance (Gaussian)	Lambda (Poisson)
1	15	255
2	15	255
4	12	255
6	12	255
8	8	255
10	8	255
1	7	128
2	7	128
4	6	128
6	6	128
8	4	128
10	4	128

The following data show details of tests 4, 9, and 12 as discussed in the results section above. Histograms document the distribution of the random errors applied to the test apparatus. Note that the graphs are connected point-to-point, not as stairsteps.

TABLE 7.2 Test 4, Gaussian Burst Length Data

Burst Length	Number of Occurrences
0	867
1	2043
2	4349
3	8017
4	13143
5	19133
6	24506
7	27689
8	27676
9	24487
10	19131
11	13205
12	8024
13	4347
14	2041
15	867
16	318
17	113
18	35
19	6
20	3
21	0
22	0
23	0
24	0
25	0
26	0
27	0
28	0
29	0
30	0
31	0
32	0
33	0
34	0
35	0
36	0
37	0
38	0
39	0
40	0

TABLE 7.3 Test 9, Gaussian Burst Length Data

Burst Length	Number of Occurrences
0	39
1	223
2	979
3	3317
4	8799
5	18387
6	29989
7	38287
8	38309
9	29994
10	18339
11	8808
12	3283
13	972
14	232
15	37
16	3
17	3
18	0
19	0
20	0
21	0
22	0
23	0
24	0
25	0
26	0
27	0
28	0
29	0
30	0
31	0
32	0
33	0
34	0
35	0
36	0
37	0
38	0
39	0
40	0

TABLE 7.4 Test 12, Gaussian Burst Length Data

Burst Length	Number of Occurrences
0	45537
1	45443
2	39543
3	29736
4	19590
5	11046
6	5492
7	2352
8	877
9	293
10	66
11	20
12	5
13	0
14	0
15	0
16	0
17	0
18	0
19	0
20	0
21	0
22	0
23	0
24	0
25	0
26	0
27	0
28	0
29	0
30	0
31	0
32	0
33	0
34	0
35	0
36	0
37	0
38	0
39	0
40	0

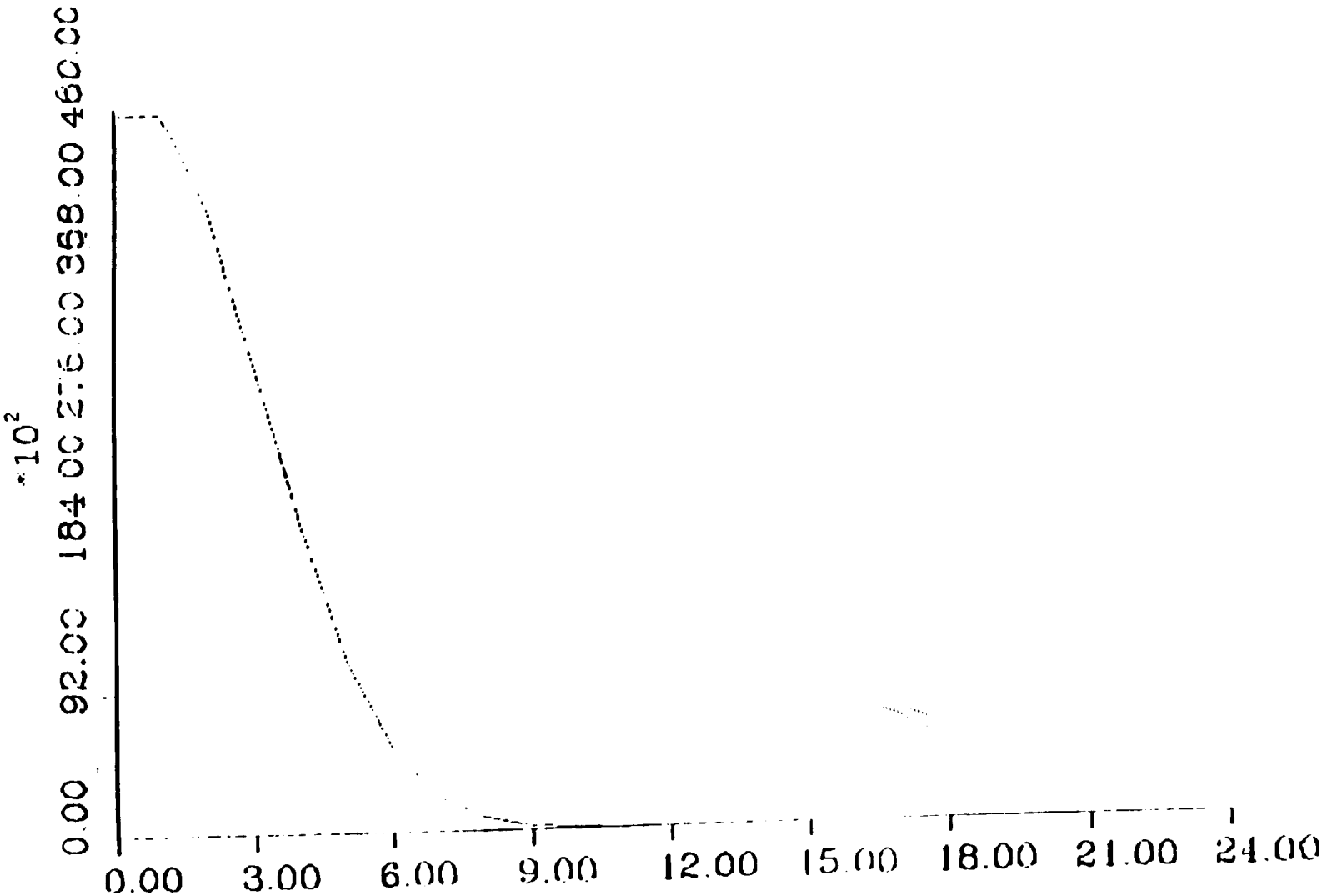


Figure 7.1 Test 4 Gaussian Distribution, Mean = 1, Variance = 7

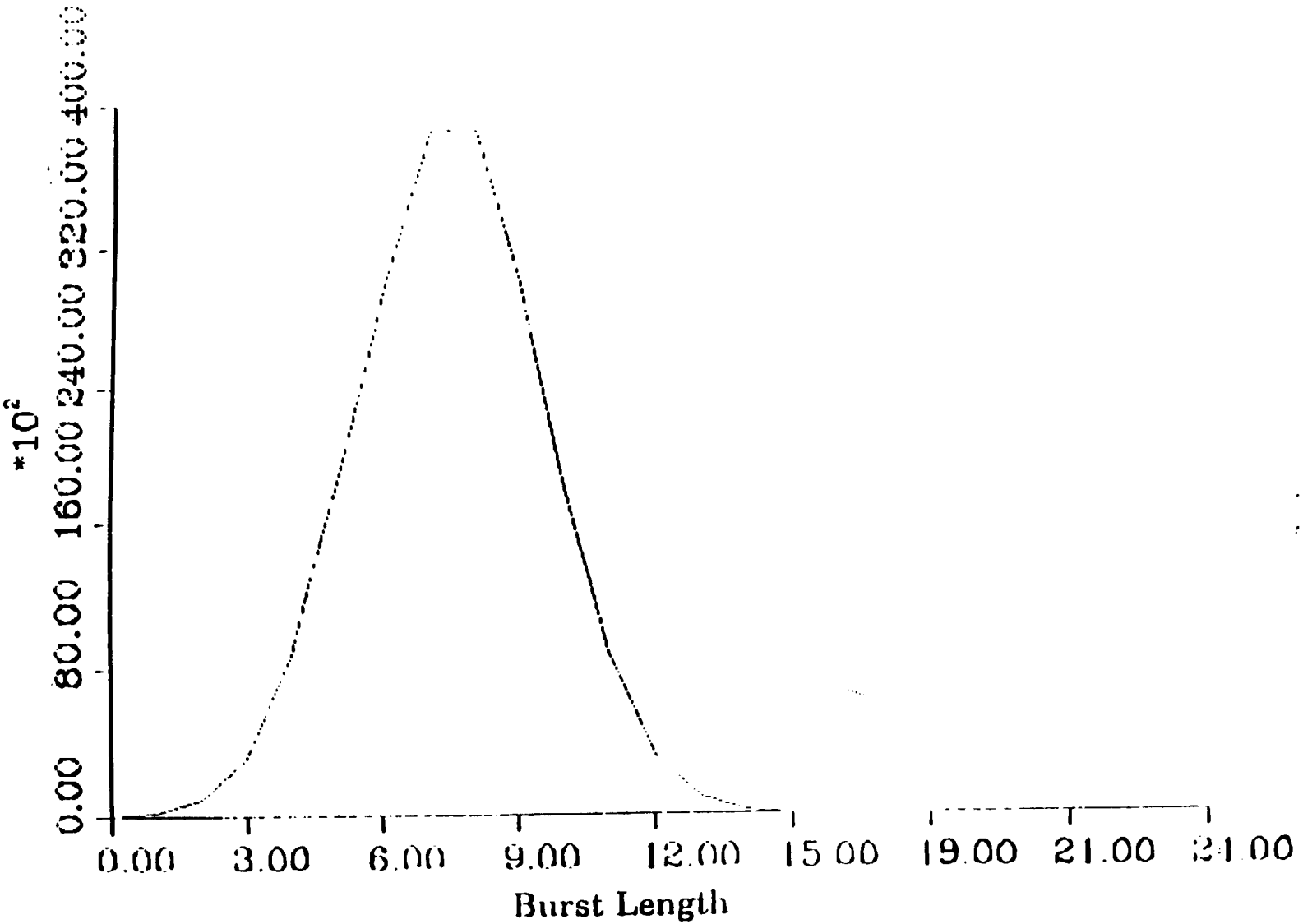


Figure 7.2 Test 9 Gaussian Distribution, Mean = 8, Variance = 4

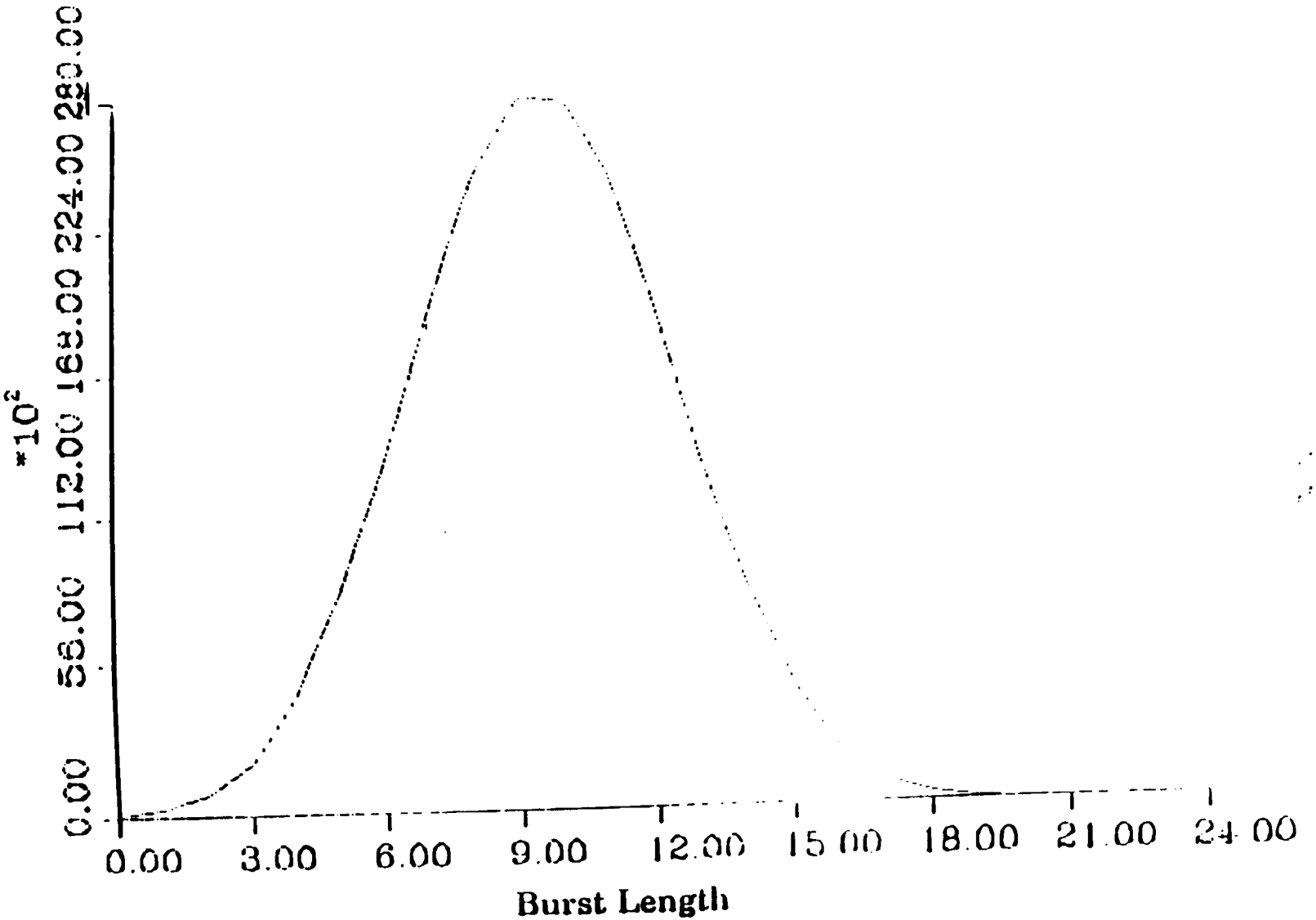


Figure 7.3 Test 12 Gaussian Distribution, Mean = 10, Variance = 8

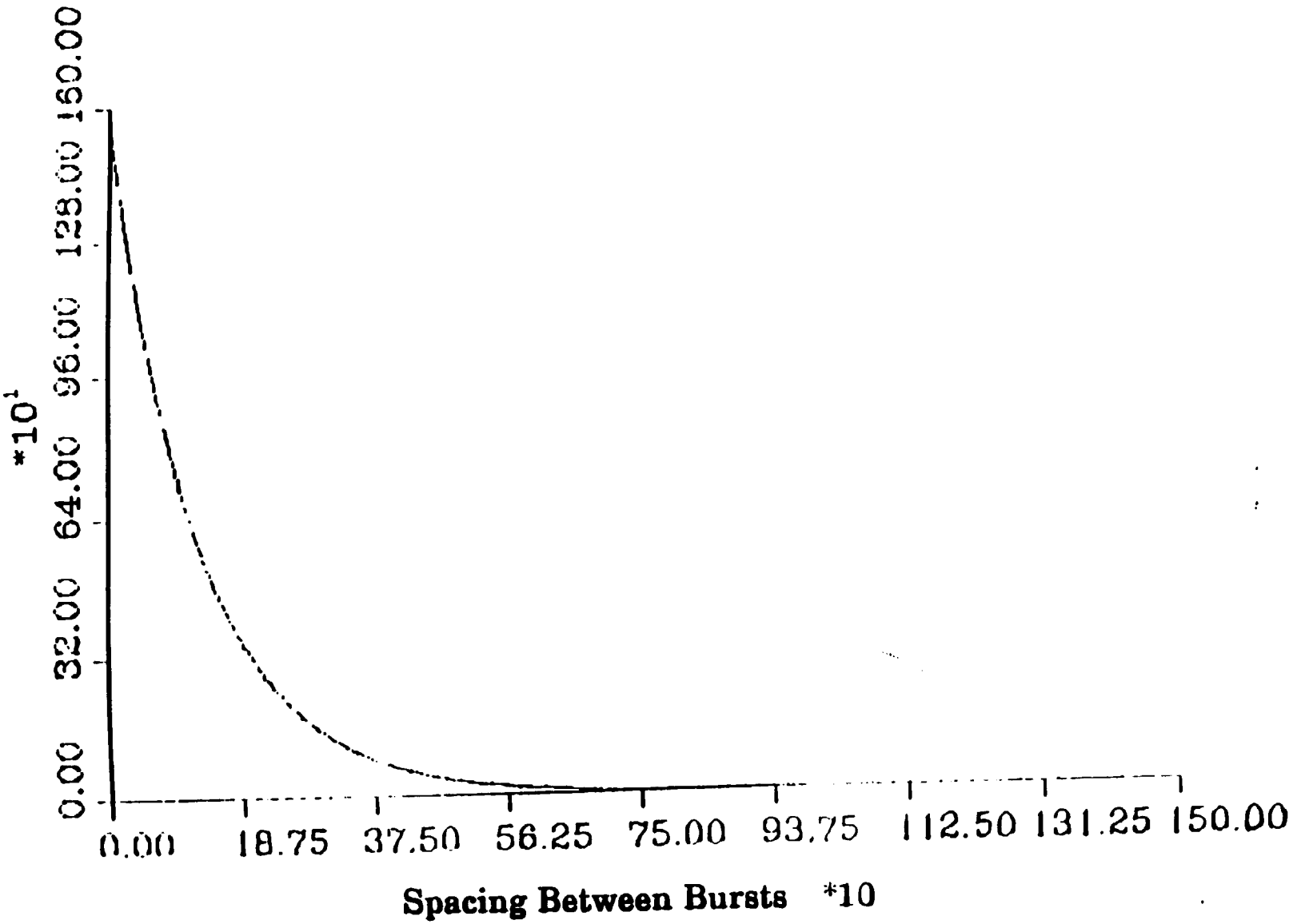


Figure 7.4 Poisson Distribution, Lambda = 128

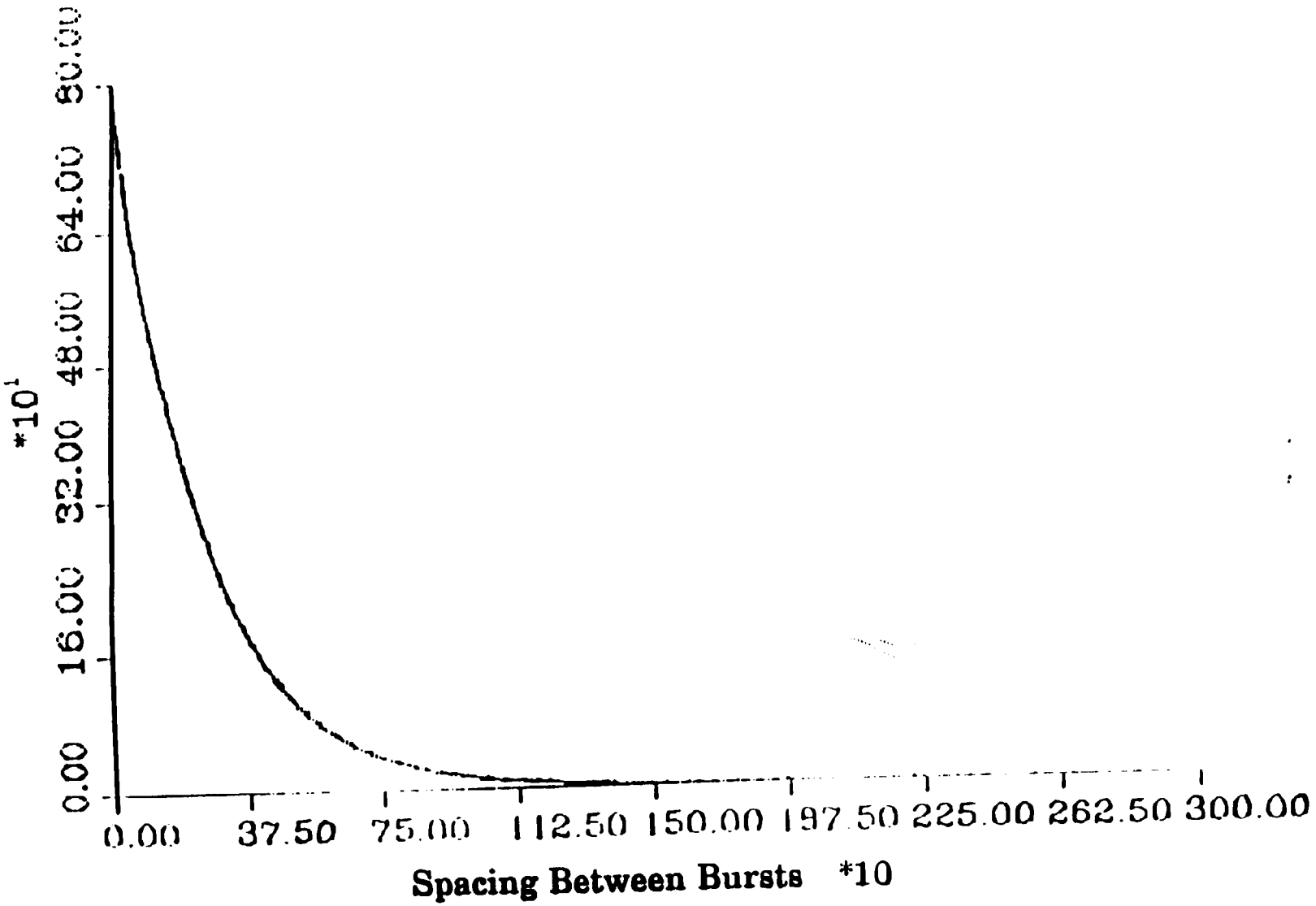


Figure 7.5 Poisson Distribution, Lambda = 255

APPENDIX 7.B
Program Listings

This appendix contains printouts of the programs developed to control and supply data to the encoder/decoder. There were many variations of the immediate mode programs, applying various control sequences and single bit or byte errors to different parts of the data blocks. For example, the reset sequence in OUT19F.ASM was put into a loop to debug a problem with initilizing the system. Progressively moving a single bit error to new locations on the data block by adjusting loop counters permits identification of timing marks on the oscilloscope.

OUT19F.ASM This is an Immediate mode program used in initial testing. It performs the following functions:

- clears data and error ports
- resets encoder and decoder
- then begins infinite loop to output
  - -- alternating zeros and ones data, inserting an error on the 218th data byte
  - -- the check bytes.

BURST.FOR Creates the Gaussian burst data of a specified mean and variance, storing the data in file BURST.DAT.

STAT.FOR Calculates the sample mean and variance for the data in BURST.DAT. Note that the data is constrained, there can be no bursts of length less than zero.

The upper magnitude limit is 2^{16}, since the data is represented as a two byte integer. This means that although the user specifies a desired mean and variance in BURST.FOR, the experimental values may be slightly skewed. STAT.FOR calculates the actual experimental statistics.

POISSON.FOR Creates the Poisson spacing data given the input $\lambda = 1.0/T$, where T is the spacing between bursts. The data is stored in the file POISSON.DAT

COUNT2.FOR Creates the loop counter file COUNT.DAT. These are the number of bytes of corrupted and uncorrupted data needed to fill out the output array.

COUNT3.FOR Creates the loop counter files LOOP1.DAT, and LOOP2.DAT. These counters are associated with the arrays BURST[], and POISSON[] that are continually filled and emptied of the data being read from the RAM disk.

CPROG.EXE This program will read through the BURST.DAT, and POISSON .DAT data files once to calculate the number of code blocks containing 17 or more errors per 255 bytes.

OUT22.ASM Uses all of the data created above to drive the R/S circuit board with Poisson distributed error bursts having Gaussian distributed lengths.

7.B.1 Program OUT19F.ASM

```
; port on I/O card          ;
;    data_port = 0X0378 = 888;
;  status_port = 0X0379 = 889;
; control_port = 0X037A = 890;
; port_control = 0x037B = 891;
; port on video card    ;
;    data_port = 0x03BC = 956;
;  status_port = 0x03BD = 957;
; control_port = 0x03BE = 958;
; port_control = 0x03BF = 959;

;    Static Name Aliases
     TITLE   out3.c
     NAME    out3

     .286p
     .287
_TEXT SEGMENT  WORD PUBLIC 'CODE'
_TEXT ENDS
_DATA SEGMENT  WORD PUBLIC 'DATA'
_DATA ENDS
CONST SEGMENT  WORD PUBLIC 'CONST'
CONST ENDS
_BSS  SEGMENT  WORD PUBLIC 'BSS'
_BSS  ENDS
DGROUP      GROUP CONST, _BSS, _DATA
     ASSUME  CS: _TEXT, DS: DGROUP, SS: DGROUP
;EXTRN      __acrtused:ABS
;EXTRN      __chkstk:NEAR
_TEXT      SEGMENT
     ASSUME      CS: _TEXT
; Line 7
     PUBLIC      _main
_main PROC NEAR
; CLEAR AX REGISTER
     xor   ax,ax

; DISABLE INTERUPTS, CLOCK
     cli

;CLEAR DATA PORT AND ERROR PORT
     mov   dx,888
     mov   ax,0
     out   dx,al                ;clear data port
     mov   dx,956
     out   dx,al
     mov   dx,890               ;clear error port
     mov   ax,15
     out   dx,al          ;clear control port,sysrstn high
```

```
;CLEAR ENCODER BY CLOCKING 32 TIMES WITH INC LOW, LEAVE INC LOW
;AND CLOCK HIGH  **REMEMBER  PORT 890 HAS INVERTED LOGIC**

    mov  cx,32
    mov  dx,890
clk0:
    mov  ax,15
    out  dx,al          ;CLOCK LOW & INC LOW  1111
    mov  ax,13
    mov  ax,13
    out  dx,al          ;CLOCK HIGH & INC LOW 1101
    loop clk0

;RESET DECODER BY CLOCKING 8 TIMES WITH SysRstN LOW, THEN 7
;TIMES WITH SysRstN HIGH

    mov  ax.9               ;BRING SysrstN LOW  1101
    out  dx.al

    mov  cx,9
clk1:                   ;CLOCK 8 TIMES
    mov  ax,11
    out  dx,al          ;CLOCK LOW  1111
    mov  ax,9
    mov  ax,9
    mov  ax,9
    out  dx,al          ;CLOCK HIGH  1101
    loop clk1

    mov  ax,13
    out  dx,al          ;BRING SysrstN HIGH  1001

    mov  cx,9
clk2:               ;CLOCK 7 TIMES
    mov  ax,15
    mov  ax,15
    out  dx,al          ;CLOCK LOW  1011
    mov  ax,13
    mov  ax,13
    mov  ax,13
    out  dx,al          ;CLOCK HIGH 1001
    loop clk2

 ;CLEAR DATA PORT AND ERROR PORT
    mov  dx,888
    mov  ax,0
    out  dx,al          ;clear data port
    mov  dx,956
    mov  ax,0
    out  dx,al          ;clear error port
```

```
    mov  dx,890
    mov  ax,5
    out  dx,al          ;inc high & clock high 0001

start:                  ;START MAIN LOOP
; THE FOLLOWING LINES ARE COMMENTED OUT
;       mov dx,888
;       mov ax,255
;      out dx,al        ;put all ones out first;
;
;      mov dx,890
;       mov ax,7
;       out dx,al       ;inc high & clock low  0011
;       mov ax,5
;       out dx,al       ;inc high & clock high 0001
;       mov ax,7
;       out dx,al       ;inc high & clock low  0011
;       mov ax,5
;       out dx,al       ;inc high & clock high 0001

;OUTPUT 220 DATA BYTES ALTERNATING ZEROS 00000000 AND ONES 11111111
;WITH INC HIGH

    mov cx,110          ;SET LOOP COUNTER
dump:
    mov  dx,888
    mov  ax,0
    out  dx,al
    mov  dx,890
    mov  ax,7
    out  dx,al      ;INC HIGH & CLOCK LOW  0011
;   mov  ax,5       ;SOMETIMES USED TO PROVIDE DELAY
    mov  ax,5
    out  dx,al      ;INC HIGH & CLOCK HIGH  0001
    mov  dx,888
    mov  ax,255     ;LOAD ALL ONES INTO DATA REGISTER
    out  dx,al
    mov  dx,890
    mov  ax,7
    out  dx,al
    mov  ax,5
;    mov  ax,5
    out  dx,al
    loop dump

;INSERT ERROR ON THE LAST DATA BYTE
    mov  dx,956
    mov  ax,255
    out  dx,al      ;OUTPUT THE ERROR 11111111

    mov  dx,890
```

```
    mov  ax,7
    out  dx,al          ;INC HIGH & CLOCK LOW  0011
    mov  ax,5
    out  dx,al          ;INC HIGH & CLOCK HIGH 0001
    mov  dx,956
    mov  ax,255
    out  dx,al          ;output the error 00000000
    mov  dx,890
    mov  ax,7
    out  dx,al            ;inc high & clock low  0011
    mov  ax,5
    out  dx,al            ;inc high & clock high 0001
    mov  dx,956
    mov  ax,255
    out  dx,al
    mov  dx,890
    mov  ax,7
    out  dx,al            ;inc high & clock low  0011
    mov  ax,5
    out  dx,al            ;inc high & clock high 0001

    mov  dx,956             ;clear errors
    mov  ax,0
    out  dx,al

;OUTPUT CHECK BYTES BY CLOCKING 32 TIMES WITH INC LOW

    mov  cx,32                ;SET LOOP COUNTER
    mov  dx,890
    mov  ax,13
    out  dx,al          ;bring inc low
clk3:
    mov  ax,15
    out  dx,al          ;INC LOW & CLOCK LOW  1011
    mov  ax,13
;    mov  ax,13
    out  dx,al          ;INC LOW & CLOCK HIGH  1001
    loop clk3
    mov  ax,5
    out  dx,al          ;bring inc high

;    mov dx,956
;    mov ax,0
;    out dx,al
    loop start

_main ENDP
_TEXT ENDS
END
```

7.B.2 Program BURST.FOR

```
PROGRAM BURST.FOR
C     THIS PROGRAM GENERATES AN N SAMPLE GAUSSIAN DISTRIBUTED
C     SEQUENCE WITH GIVEN MEAN AND VARIANCE
C
C     INPUT PARAMETERS
C
C     N      NUMBER OF REAL WHITE NOISE SAMPLES DESIRED
C     VAR    DESIRED VARIANCE
C     MEAN   DESIRED MEAN
C     ISEED  SEED NUMBER FOR RANDOM NUMBER GENERATOR
C
C     OUTPUT PARAMETERS
C
C     W      OUTPUT ARRAY
C
C
      REAL      W(100), MEAN, VAR
      INTEGER   N, ISEED
      OPEN (UNIT=7,FILE='D:\BURST.DAT')
       PRINT*, 'ENTER THE NUMBER OF SAMPLES TO GENERATE'
       READ*, N
       PRINT*, ' '
       PRINT*, 'ENTER THE MEAN'
       READ*, MEAN
       PRINT*, ' '
       PRINT*, 'ENTER THE VARIANCE OF THE NOISE SAMPLES'
       READ*, VAR
       PRINT*, ' '
       PRINT*, 'ENTER THE SEED NUMBER OF RANDOM GENERATION'
       READ*, ISEED
       CALL WGN(N,VAR,ISEED,W,MEAN)
       STOP
       END
C      ********** WGN SUBROUTINE **********
       SUBROUTINE  WGN(N,VAR,ISEED,W,MEAN)
       REAL        MEAN, W(1), TWOPI, ROOTVA, U1, U2, TEMP, RAN
       INTEGER     I, K, L, M
       INTEGER     J
       K=0
       TWOPI=8*ATAN(1.0)
       M=100
       ROOTVA = SQRT(VAR)
       IF (MOD(N,2) .NE. 0) M=N+1
5        CALL RANDOM(M,ISEED,W)
         L = 50
         DO 10 I=1,L
           U1=W(2*I-1)
           U2=W(2*I)
           IF (U1 .LE. 0.0)  U1=0.5
```

```
          TEMP=SQRT(-2.0*ALOG(U1))
          W(2*I-1)=TEMP*COS(TWOPI*U2)*ROOTVA
          W(2*I)=TEMP*SIN(TWOPI*U2)*ROOTVA
10      CONTINUE
        DO 30 I=1,100
           W(I)=W(I)+MEAN
           IF ((W(I) .LT. 256) .AND. (W(I) .GE.0)) THEN
             J=INT(W(I))
             K=K+1
             WRITE (7,45) J
             IF (K .EQ. N) GO TO 40
           ENDIF
30      CONTINUE
        IF (K.LT.N) GOTO 5
        CLOSE(UNIT=7)
45      FORMAT(I2)
40      PRINT*, 'NUMBER OF SAMPLES GENERATED',K
        RETURN
        END
C       ********** RANDOM SUBROUTINE **********
        SUBROUTINE RANDOM(M,ISEED,W)
        REAL  W(1), RAN
        DO 20 I=1,M
             W(I)=RAN(ISEED)
20      CONTINUE
        RETURN
        END
C       ********** RAN FUNCTION **********
        FUNCTION RAN(ISEED)
        INTEGER A,C
        A=25173
        C=13849
        M=65536
        ISEED=MOD(A*ISEED+C,M)
        RAN=FLOAT(ISEED)/FLOAT(M)
        RETURN
        END
```

7.B.3 Program POISSON.FOR

```
PROGRAM POISSON.FOR
C     THIS PROGRAM GENERATES AN N SAMPLE NORMALLY DISTRIBUTED
C     NOISE SEQUENCE AND AN N SAMPLE SEQUENCE OF POISSON
C     SPACED NUMBERS
C
C     INPUT PARAMETERS
C
C     N      NUMBER OF REAL NOISE SAMPLES DESIRED
C     ISEED   SEED NUMBER FOR RANDOM NUMBER GENERATOR
C     LAMBDA  AVERAGE POISSON SPACING
C
C     OUTPUT PARAMETERS
C
C     W      REAL ARRAY OF DIMENSION Nx1 OF NOISE SAMPLES
C     P      REAL ARRAY OF POISSON SPACED NUMBERS
C
C
      REAL        LAMBDA, P, RANDOM, W
      INTEGER     N, I, A, C, M, K, ISEED
      INTEGER     J
      PRINT*, 'ENTER THE NUMBER OF SAMPLES TO GENERATE'
      READ*, N
      PRINT*, ' '
      PRINT*, 'ENTER THE AVERAGE POISSON SPACING'
      READ*, LAMBDA
      PRINT*, ' '
      PRINT*, 'ENTER THE SEED NUMBER OF RANDOM GENERATION'
      READ*, ISEED

      LAMBDA = 1.0/LAMBDA
      A=25173
      C=13849
      M=65536
      K=0
     OPEN (UNIT=7,FILE='D:POISSON.DAT')

10          ISEED=MOD(A*ISEED+C,M)
            W=FLOAT(ISEED)/FLOAT(M)
            P=(-1.0*ALOG(1.0-W))/LAMBDA
            IF (P .GE. 0.0) THEN
              J = INT(P)
              K = K + 1
              WRITE (7,20) J
              ENDIF
      IF (K.LT.N) GOTO 10
      PRINT*, 'NUMBER OF SAMPLES GENERATED',K
 20    FORMAT(I4)
      STOP
      END
```

7.B.4 Program COUNT1.FOR

```
PROGRAM COUNT1.FOR
C     THIS PROGRAM CREATES THE COUNT.DAT WHICH CONTAINS
C     THEN NUMBER OF SAMPLES TO READ FROM BURST AND POISSON
C     TO FILL THE OUTPUT ARRAY OF APPROXIMATE SIZE GIVEN
C
C     ARRAY=255*32=8160
C
C     OUTPUT FILE IS COUNT.DAT
C     ALSO THE PROGRAM WILL OUTPUT J, THE NUMBER
C     OF ELEMENTS IN COUNT.DAT
C
      INTEGER ARRAY,Y,COUNT(20000)
      INTEGER   B,P
      PRINT*, 'ENTER THE NUMBER OF SAMPLES TO COUNT'
      READ*, N
      PRINT*, 'ENTER THE ARRAY SIZE (OUTPUT) MULTIPLE OF 255 (8160)'
      READ*, ARRAY
     OPEN (UNIT=5,FILE='D:\BURST.DAT')
     OPEN (UNIT=7,FILE='D:\POISSON.DAT')
      Y=0
      K=0
      J=0
      L=0
      DO 20 I=1,N
            READ (5,*) B
            READ (7,*) P
            Y=Y+B+P
            K=K+1
            IF (Y .GE. ARRAY) THEN
                  J=J+1
                  COUNT(J)=K
                  Y=0
                  K=0
            ENDIF
 20    CONTINUE
      IF (K .NE. 0) THEN
            J=J+1
            COUNT(J)=K
      ENDIF
      CLOSE (UNIT=5)
      CLOSE (UNIT=7)
      OPEN (UNIT=5,FILE='C:\PROG\MINE\COUNT.DAT')
      DO 50 I=1,J
            WRITE (5,40) COUNT(I)
 50    CONTINUE
 40    FORMAT (I5)
      PRINT*, 'NUMBER OF ELEMENTS IN COUNT.DAT',J
      STOP
      END
```

7.B.5 Program COUNT2.FOR

```
PROGRAM COUNT2.FOR
C     THIS PROG READS FILE COUNT.DAT AND CREATES
C     THE FILES LOOP1.DAT AND LOOP2.DAT
C
C     ARRAY=3400
      INTEGER B(20000),Y,Z,LOOP1(1000),LOOP2(1000),ARRAY
      PRINT*, 'ENTER THE NUMBER OF ELEMENTS IN COUNT.DAT'
      READ*, N
      PRINT*, 'ENTER THE ARRAY SIZE (BURST & POISSON)'
      READ*, ARRAY
      OPEN (UNIT=5,FILE='C:\PROG\MINE\COUNT.DAT')
      DO 10 I=1,N
            READ (5,*) B(I)
 10    CONTINUE
      CLOSE (UNIT=5)
      Y=0
      K=0
      Z=0
      J=0
      DO 20 I=1,N
            Y=Y+B(I)
            Z=Z+B(I)
            K=K+1
            IF (Y .GE. ARRAY) THEN
                  J=J+1
                  LOOP1(J)=K
                  LOOP2(J)=Y
                  Y=0
                  K=0
            ENDIF
 20    CONTINUE
      IF (K .NE. 0) THEN
            J=J+1
            LOOP1(J)=K
            LOOP2(J)=Y
      ENDIF
      OPEN (UNIT=7,FILE='C:\PROG\MINE\LOOP1.DAT')
      DO 30 I=1,J
            WRITE (7,100) LOOP1(I)
 30    CONTINUE
      CLOSE (UNIT=7)
      OPEN (UNIT=7,FILE='C:\PROG\MINE\LOOP2.DAT')
      DO 40 I=1,J
            WRITE (7,100) LOOP2(I)
 40    CONTINUE
      PRINT*, 'NUMBER OF ELEMENTS IN LOOP1 AND LOOP2',J
 100   FORMAT (I6)
      STOP
      END
```

7.B.6 Program CPROG.FOR

```
C     THIS PROG CALCULATES THE NUMBER OF UNCORRECTABLE
C     CODE WORDS IN THE DATA GENERATED
C
      INTEGER POISSO(3500),BURST(3500),COUNT(6500)
      INTEGER LOOP1(100),LOOP2(100).OUTPUT(8200),TEMP(256)
      INTEGER R,S,T,W,X,M,N,Y,Z,I,K,STOP,START
      PRINT*, 'ENTER THE NUMBER OF ELEMENTS IN COUNT.DAT'
      READ*, NCOUNT
      PRINT*, 'ENTER THE NUMBER OF ELEMENTS IN LOOP1.DAT'
      READ*, NLOOP
      OPEN (UNIT=5,FILE='C:\PROG\COUNT.DAT')
      DO 10 I=1,NCOUNT
            READ (5,*) COUNT(I)
 10    CONTINUE
      CLOSE (UNIT=5)
      OPEN (UNIT=5.FILE='C:\PROG\LOOP1.DAT')
      DO 20 I=1,NLOOP
            READ (5,*) LOOP1(I)
 20    CONTINUE
      CLOSE (UNIT=5)
      OPEN (UNIT=5,FILE='C:\PROG\LOOP2.DAT')
      DO 30 I=1,NLOOP
            READ (5,*) LOOP2(I)
 30    CONTINUE
      CLOSE (UNIT=5)
      R=0
      T=0
      S=0
      K=1
      NN=31
      OPEN (UNIT=5,FILE='D:\BURST.DAT')
      OPEN (UNIT=7,FILE='D:\POISSON.DAT')
      OPEN (UNIT=6,FILE='ERROR.DAT')
      DO 40 Z=1,NLOOP
C           PRINT*, Z
            W=1
            DO 50 N=1,LOOP2(Z)
                  READ (5,*) BURST(N)
 50           CONTINUE
            DO 60 N=1,LOOP2(Z)
                  READ (7,*) POISSO(N)
 60           CONTINUE
            DO 70 N=1,LOOP1(Z)
                  Y=1
                  DO 80 X=1,COUNT(K)
                        I=1
 90                      IF (I .LE. BURST(W)) THEN
                        OUTPUT(Y)=255
C                       PRINT*, Y,OUTPUT(Y)
```

```
                    Y=Y+1
                    IF (Y .GT. 8160) I=BURST(W)
                    I=I+1
                    GO TO 90
                    ENDIF
                    I=1
110                  IF (I .LE. POISSO(W)) THEN
                    OUTPUT(Y)=0
C                    PRINT*, Y,OUTPUT(Y)
                    Y=Y+1
                    IF (Y .GT. 8160) I=POISSO(W)
                    I=I+1
                    GO TO 110
                    ENDIF
C                    PRINT*, BURST(W),POISSO(W)
                    W=W+1
80                CONTINUE
                  IF (N .EQ. 1 .AND. Z .EQ. 1) GO TO 135
                  S=0
                  R=R+1
                  TEMP(254)=OUTPUT(1)
                  TEMP(255)=OUTPUT(2)
                  DO 130 X=1,255
                     IF (TEMP(X) .EQ. 255) S=S+1
130               CONTINUE
                  IF (S .GE. 17) T=T+1
135               DO 140 X=1,253
                        TEMP(X)=OUTPUT(7907+X)
140               CONTINUE
                  DO 150 X=1,7905
                        OUTPUT(X)=OUTPUT(X+2)
150               CONTINUE
              IF (Z .EQ. NLOOP .AND. N .EQ. LOOP1(Z)) NN=28
                  DO 160 I=1,NN
                        R=R+1
                        S=0
                        START=((I-1)*255)+1
                        STOP=I*255
                        DO 170 M=START,STOP
                         IF (OUTPUT(M) .EQ. 255) S=S+1
170                       CONTINUE
                        IF (S .GE. 17) T=T+1
160               CONTINUE
                 K=K+1
C                 PRINT*, N
 70          CONTINUE
            WRITE (6,*) Z,R,T
 40   CONTINUE
      STOP
      END
```

7.B.7 Program OUT22.C

```
; port on I/O card          ;
;    data_port = 0X0378 = 888;
;  status_port = 0X0379 = 889;
; control_port = 0X037A = 890;
; port_control = 0x037B = 891;
; port on video card    ;
;    data_port = 0x03BC = 956;
;  status_port = 0x03BD = 957;
; control_port = 0x03BE = 958;
; port_control = 0x03BF = 959;

;       Static Name Aliases
;
        TITLE   c:\prog\disk13.c
        NAME    disk13

        .286p
        .287
_TEXT SEGMENT  WORD PUBLIC 'CODE'
_TEXT ENDS
_DATA         SEGMENT  WORD PUBLIC 'DATA'
_DATA         ENDS
CONST         SEGMENT  WORD PUBLIC 'CONST'
CONST         ENDS
_BSS   SEGMENT  WORD PUBLIC 'BSS'
_BSS   ENDS
DGROUP        GROUP         CONST, _BSS, _DATA
        ASSUME  CS: _TEXT, DS: DGROUP, SS: DGROUP
EXTRN         __acrtused:ABS
EXTRN         _fclose:NEAR
EXTRN         _exit:NEAR
EXTRN         _fopen:NEAR
EXTRN         __chkstk:NEAR
EXTRN         _fscanf:NEAR
EXTRN         _gets:NEAR
EXTRN         _printf:NEAR
_DATA     SEGMENT
$SG223          DB      'r', 00H
$SG224          DB      'count.dat', 00H
$SG228          DB      '%u ', 00H
$SG229          DB      'r', 00H
$SG230          DB      'loop1.dat', 00H
$SG234          DB      '%u ', 00H
$SG235          DB      'r', 00H
$SG236          DB      'loop2.dat', 00H
$SG240          DB      '%u ', 00H
$SG241          DB      'r', 00H
$SG242          DB      'd:burst.dat', 00H
$SG243          DB      'r', 00H
```

```
$SG244          DB      'd:poisson.dat', 00H
$SG248          DB      '%u ', 00H
$SG252          DB      '%u ', 00H
$SG256          DB      '%u ', 00H
_DATA       ENDS
_TEXT       SEGMENT
        ASSUME      CS: _TEXT
; Line 10
        PUBLIC      _main
_main  PROC NEAR

    xor  ax,ax

; DISABLE INTERUPTS, CLOCK
    cli

;CLEAR DATA PORT AND ERROR PORT
    mov  dx,888
    mov  ax,0
    out  dx,al              ;clear data port
    mov  dx,956
    out  dx,al
    mov  dx,890             ;clear error port
    mov  ax,15
    out  dx,al          ;clear control port,sysrstn high

;CLEAR ENCODER BY CLOCKING 32 TIMES WITH INC LOW, LEAVE INC LOW
;AND CLOCK HIGH  **REMEMBER  PORT 890 HAS INVERTED LOGIC**

    mov  cx,32
    mov  dx,890
clk0:
    mov  ax,15
    out  dx,al          ;CLOCK LOW & INC LOW  1111
    mov  ax,13
    out  dx,al          ;CLOCK HIGH & INC LOW 1101
    loop clk0

;RESET DECODER BY CLOCKING 8 TIMES WITH SysRstN LOW, THEN 7
;TIMES WITH SysRstN HIGH

    mov  ax,9               ;BRING SysrstN LOW  1101
    out  dx,al

    mov  cx,9
clk1:                   ;CLOCK 8 TIMES
    mov  ax,11
    out  dx,al          ;CLOCK LOW  1111
    mov  ax,9
    out  dx,al          ;CLOCK HIGH  1101
    loop clk1
```

```
    mov  ax,13
    out  dx,al          ;BRING SysrstN HIGH  1001

    mov  cx,9
clk2:                   ;CLOCK 7 TIMES
    mov  ax,15
    out  dx,al          ;CLOCK LOW  1011
    mov  ax,13
    out  dx,al          ;CLOCK HIGH 1001
    loop clk2

;CLEAR DATA PORT AND ERROR PORT
    mov  dx,888
    mov  ax,0
    out  dx,al          ;clear data port
    mov  dx,956
    mov  ax,0
    out  dx,al          ;clear error port
    mov  dx,890
    mov  ax,5
    out  dx,al          ;inc high & clock high 0001


;ARRAY AND VARIABLE INTERNAL SETUP

        push   bp
        mov    bp,sp
        mov    ax,25658
        call   __chkstk
        push   si
;       poisson = -25418
;       loop1 = -18418
;       loop2 = -25618
;       w = -8204
;       x = -11708
;       m = -25620
;       n = -25634
;       y = -18214
;       z = -18216
;       i = -18212
;       k = -18218
;       stop = -8202
;       start = -25632
;       count = -18210
;       zurst = -11706
;       output = -8200
;       PAUSE = -25630
;       infile1 = -8206
;       infile2 = -11710
; Line 19
```

```
        push    OFFSET DGROUP:$SG223
        push    OFFSET DGROUP:$SG224
        call    _fopen
        add     sp,4
        mov     WORD PTR [bp-8206],ax       ;infile1
; Line 20
        mov     WORD PTR [bp-25634],1       ;n
$L20001:
; Line 21
; Line 22
        mov     si,WORD PTR [bp-25634]      ;n
        lea     ax,[bp-18210][si]
        push    ax
        push    OFFSET DGROUP:$SG228
        push    WORD PTR [bp-8206] ;infile1
        call    _fscanf
        add     sp,6
; Line 23
        inc     WORD PTR [bp-25634]         ;n
        cmp     WORD PTR [bp-25634],6277    ;n
        jbe     $L20001
; Line 24
        push    WORD PTR [bp-8206] ;infile1
        call    _fclose
        add     sp,2
; Line 25
        push    OFFSET DGROUP:$SG229
        push    OFFSET DGROUP:$SG230
        call    _fopen
        add     sp,4
        mov     WORD PTR [bp-8206],ax       ;infile1
; Line 26
        mov     WORD PTR [bp-25634],1       ;n
$L20002:
; Line 27
; Line 28
        mov     si,WORD PTR [bp-25634]      ;n
        shl     si,1
        lea     ax,[bp-18418][si]
        push    ax
        push    OFFSET DGROUP:$SG234
        push    WORD PTR [bp-8206] ;infile1
        call    _fscanf
        add     sp,6
; Line 29
        inc     WORD PTR [bp-25634]         ;n
        cmp     WORD PTR [bp-25634],59      ;n
        jbe     $L20002
; Line 30
        push    WORD PTR [bp-8206] ;infile1
        call    _fclose
```

```
        add     sp,2
; Line 31
        push    OFFSET DGROUP:$SG235
        push    OFFSET DGROUP:$SG236
        call    _fopen
        add     sp,4
        mov     WORD PTR [bp-8206],ax       ;infile1
; Line 32
        mov     WORD PTR [bp-25634],1       ;n
$L20003:
; Line 33
; Line 34
        mov     si,WORD PTR [bp-25634]      ;n
        shl     si,1
        lea     ax,[bp-25618][si]
        push    ax
        push    OFFSET DGROUP:$SG240
        push    WORD PTR [bp-8206] ;infile1
        call    _fscanf
        add     sp,6
; Line 35
        inc     WORD PTR [bp-25634]         ;n
        cmp     WORD PTR [bp-25634],59      ;n
        jbe     $L20003
; Line 36
        push    WORD PTR [bp-8206] ;infile1
        call    _fclose
        add     sp,2
; Line 38
        lea     ax,WORD PTR [bp-25630]      ;PAUSE
        push    ax
        call    _gets
        add     sp,2
; Line 39
        mov     WORD PTR [bp-18218],1       ;k
; Line 40
        push    OFFSET DGROUP:$SG241
        push    OFFSET DGROUP:$SG242
        call    _fopen
        add     sp,4
        mov     WORD PTR [bp-8206],ax       ;infile1
; Line 41
        push    OFFSET DGROUP:$SG243
        push    OFFSET DGROUP:$SG244
        call    _fopen
        add     sp,4
        mov     WORD PTR [bp-11710],ax      ;infile2
; Line 42
        mov     WORD PTR [bp-18216],1       ;z
        jmp     $F245
$L20004:
```

```
; Line 47
; Line 48
        mov     si,WORD PTR [bp-25634]      ;n
        lea     ax,[bp-11706][si]
        push    ax
        push    OFFSET DGROUP:$SG252
        push    WORD PTR [bp-8206] ;infile1
        call    _fscanf
        add     sp,6
; Line 49
        inc     WORD PTR [bp-25634]         ;n
$L20005:
        mov     si,WORD PTR [bp-18216]      ;z
        shl     si,1
        mov     ax,WORD PTR [bp-25634]      ;n
        cmp     [bp-25618][si],ax
        jae     $L20004
; Line 50
        mov     WORD PTR [bp-25634],1       ;n
        jmp     SHORT $L20007
$L20006:
; Line 51
; Line 52
        mov     si,WORD PTR [bp-25634]      ;n
        shl     si,1
        lea     ax,[bp-25418][si]
        push    ax
        push    OFFSET DGROUP:$SG256
        push    WORD PTR [bp-11710]         ;infile2
        call    _fscanf
        add     sp,6
; Line 53
        inc     WORD PTR [bp-25634]         ;n
$L20007:
        mov     si,WORD PTR [bp-18216]      ;z
        shl     si,1
        mov     ax,WORD PTR [bp-25634]      ;n
        cmp     [bp-25618][si],ax
        jae     $L20006
; Line 54
        mov     WORD PTR [bp-25634],1       ;n
        mov     si,WORD PTR [bp-18216]      ;z
        shl     si,1
        cmp     WORD PTR [bp-18418][si],1
        jae     $JCC379
        jmp     $FC246
$JCC379:
        mov     si,WORD PTR [bp-18216]      ;z
        shl     si,1
        mov     ax,[bp-18418][si]
        mov     WORD PTR [bp-25654],ax
```

```
        mov     si,WORD PTR [bp-8204]       ;w
        shl     si,1
        lea     ax,[bp-25418][si]
        mov     WORD PTR [bp-25656],ax
        mov     ax,WORD PTR [bp-25654]
        mov     WORD PTR [bp-25658],ax
        add     WORD PTR [bp-25634],ax      ;n
$L20016:
; Line 55
; Line 56
        mov     WORD PTR [bp-18214],1       ;y
; Line 57
        mov     WORD PTR [bp-11708],1       ;x
        mov     si,WORD PTR [bp-18218]      ;k
        mov     al,[bp-18210][si]
        sub     ah,ah
        cmp     ax,1
        jae     $JCC449
        jmp     $FB262
$JCC449:
        mov     WORD PTR [bp-25644],ax
        mov     ax,WORD PTR [bp-25656]
        mov     WORD PTR [bp-25646],ax
        mov     ax,WORD PTR [bp-25644]
        mov     WORD PTR [bp-25648],ax
        add     WORD PTR [bp-11708],ax      ;x
$L20010:
; Line 58
; Line 59
        mov     WORD PTR [bp-18212],1       ;i
; Line 60
        mov     si,WORD PTR [bp-8204]       ;w
        mov     al,[bp-11706][si]
        sub     ah,ah
        cmp     ax,1
        jb      $FB265
        mov     WORD PTR [bp-25636],ax
        mov     WORD PTR [bp-25638],ax
$L20008:
; Line 61
; Line 62
        mov     si,WORD PTR [bp-18214]      ;y
        mov     BYTE PTR [bp-8200][si],255
; Line 63
        inc     WORD PTR [bp-18214]         ;y
; Line 64
        cmp     WORD PTR [bp-18214],8160    ;y
        jbe     $I266
; Line 65
; Line 66
        mov     ax,WORD PTR [bp-25638]
```

```
        mov     WORD PTR [bp-18212],ax      ;i
; Line 67
; Line 68
$I266:
        inc     WORD PTR [bp-18212]         ;i
; Line 69
        mov     ax,WORD PTR [bp-18212]      ;i
        cmp     WORD PTR [bp-25636],ax
        jae     $L20008
$FB265:
; Line 70
        mov     WORD PTR [bp-18212],1       ;i
; Line 71
        mov     bx,WORD PTR [bp-25646]
        cmp     WORD PTR [bx],1
        jb      $FB269
        mov     ax,WORD PTR [bx]
        mov     WORD PTR [bp-25640],ax
        mov     WORD PTR [bp-25642],ax
$L20009:
; Line 72
; Line 73
        mov     si,WORD PTR [bp-18214]      ;y
        mov     BYTE PTR [bp-8200][si],0
; Line 74
        inc     WORD PTR [bp-18214]         ;y
; Line 75
        cmp     WORD PTR [bp-18214],8160    ;y
        jbe     $I270
; Line 76
; Line 77
        mov     ax,WORD PTR [bp-25642]
        mov     WORD PTR [bp-18212],ax      ;i
; Line 78
; Line 79
$I270:
        inc     WORD PTR [bp-18212]         ;i
; Line 80
        mov     ax,WORD PTR [bp-18212]      ;i
        cmp     WORD PTR [bp-25640],ax
        jae     $L20009
$FB269:
; Line 81
        add     WORD PTR [bp-25646],2
        add     WORD PTR [bp-25656],2
        inc     WORD PTR [bp-8204] ;w
; Line 82
        dec     WORD PTR [bp-25648]
        je      $JCC636
        jmp     $L20010
$JCC636:
```

```
$FB262:
; Line 83
        mov     WORD PTR [bp-18212],33      ;i
        mov     WORD PTR [bp-25650],255
        mov     WORD PTR [bp-25652],255
$L20015:
; Line 84
; Line 85
        mov     ax,WORD PTR [bp-25650]
        sub     ax,254
        mov     WORD PTR [bp-25632],ax      ;start
; Line 86
        mov     ax,WORD PTR [bp-25650]
        sub     ax,32
        mov     WORD PTR [bp-8202],ax       ;stop
; Line 87
        mov     ax,WORD PTR [bp-25632]      ;start
        mov     WORD PTR [bp-25620],ax      ;m
        jmp     SHORT $L20012
        nop
$L20011:
; Line 88
; Line 89
        mov     si,WORD PTR [bp-25620]      ;m
        mov     al,[bp-8200][si]
        sub     ah,ah
        mov     dx,956
        out     dx, al

        mov     dx,890
        mov     ax,7
        out     dx,al
        mov     ax,5
        out     dx,al

; Line 90
        inc     WORD PTR [bp-25620]         ;m
$L20012:
        mov     ax,WORD PTR [bp-8202]       ;stop
        cmp     WORD PTR [bp-25620],ax      ;m
        jbe     $L20011
; Line 91
        mov     ax,WORD PTR [bp-25652]
        sub     ax,31
        mov     WORD PTR [bp-25632],ax      ;start
; Line 92
        mov     ax,WORD PTR [bp-25652]
        mov     WORD PTR [bp-8202],ax       ;stop
; Line 93
        mov     ax,WORD PTR [bp-25632]      ;start
        mov     WORD PTR [bp-25620],ax      ;m
```

```
        jmp     SHORT $L20014
        nop
$L20013:
; Line 94
; Line 95
        mov     si,WORD PTR [bp-25620]      ;m
        mov     al,[bp-8200][si]
        sub     ah,ah
        mov     dx,956
        out     dx, al

        mov     dx,890
        mov     ax,15
        out     dx,al
        mov     ax,13
        out     dx,al

; Line 96
        inc     WORD PTR [bp-25620]         ;m
$L20014:
        mov     ax,WORD PTR [bp-8202]       ;stop
        cmp     WORD PTR [bp-25620],ax      ;m
        jbe     $L20013
; Line 97
        add     WORD PTR [bp-25650],255
        add     WORD PTR [bp-25652],255
        cmp     WORD PTR [bp-25652],8160
        ja      $JCC796
        jmp     $L20015
$JCC796:
; Line 98
        inc     WORD PTR [bp-18218]         ;k
; Line 99
        dec     WORD PTR [bp-25658]
        je      $JCC809
        jmp     $L20016
$JCC809:
$FC246:
        inc     WORD PTR [bp-18216]         ;z
$F245:
        cmp     WORD PTR [bp-18216],59      ;z
        ja      $FB247
; Line 43
; Line 44
        push    WORD PTR [bp-18216]         ;z
        push    OFFSET DGROUP:$SG248
        call    _printf
        add     sp,4
; Line 45
        mov     WORD PTR [bp-8204],1        ;w
; Line 46
```

```
        mov     WORD PTR [bp-25634],1       ;n
        jmp     $L20005
        nop
$FB247:
; Line 101
        push    WORD PTR [bp-8206] ;infile1
        call    _fclose
        add     sp,2
; Line 102
        push    WORD PTR [bp-11710]         ;infile2
        call    _fclose
        add     sp,2
; Line 103
        push    0
        call    _exit
        add     sp,2
; Line 104
        pop     si
        leave
        ret
        nop

_main   ENDP
_TEXT   ENDS
END
```

APPENDIX 7.C

Bibliography

Sophocles J. Orfanidis, Optimum Signal Processing: An Introduction, sec. ed. 1988 McGraw-Hill.

Shu Lin and Daniel J. Costello Jr., Error Control Coding: Fundamentals and Applications, 1983 Prentice-Hall Inc., Englewood Cliffs, New Jersey

J. Terry Godfrey, Applied C: the IBM microcomputers, 1990 Prentice-Hall Inc., Englewood Cliffs, New Jersey

APPENDIX II

PRELIMINARY REPORT ON PREDICTED FIELDS IN SHUTTLE CARGO BAY DUE TO WISP ANTENNA

Mr. Dave Harris
Mr. Reggie Inmon
Mr. Todd Freestone
Mr. Herb Sims
NASA/MSFC
Fax: 205 + 544-9582
Phone: 205 + 544-3676

Frank Ingels
September 30, 1991
Phone: 601 + 325-3912
Fax: 601 + 325-2298

TABLE OF CONTENTS

CREDITS

Mr. Bobby Cox. Ph.D. student at Mississippi State University. was kind enough to provide 5 days of his time to run the NEC code results. His efforts are highly appreciated.

1.0 ANALYTICAL EXPRESSION FOR THE NEAR FIELD

The near field for a dipole in free space with no surrounding structure is expressed as (1):

$$E_\rho(\bar{r}) = \frac{j\omega\mu I_z(o)}{4\pi k\rho}\left(\frac{z-h}{R_{1h}}\,e^{-jkR_{1h}} + \frac{z+h}{R_{2h}}\,e^{-jkR_{2h}} - \frac{2z}{r}(coskh)\,e^{-jkr}\right) \quad (1)$$

and

$$E_z(\bar{r}) = \frac{-j\omega\mu I_z(o)}{4\pi k}\left(\frac{e^{-jkR_{1h}}}{R_{1h}} + \frac{e^{-jkR_{2h}}}{R_{2h}} - \frac{2}{r}\,(coskh)\,e^{-jkr}\right) \quad (2)$$

for arbitrary antenna length ℓ (tip–to–tip) = 2h and k is the complex propagation constant. $\beta + j\alpha$. The term $\beta = \frac{\omega}{c}$ is the phase constant and the term α is the the attenuation constant. The terms R_{1h}, R_{2h} and r are derived from the geometry relationship between the antenna and the observation point at which the field is calculated. Figure 1 illustrates the coordinate system and the antenna. Cylindrical coordinates are used. The term ρ (rho) is the radial distance. in the x–y plane. from the z axis. The term r is the radial distance from the origin to a point. $p = (\theta, \phi, z) = p(\phi, z) = p(x,y,z)$. Due to antenna pattern symmetry. θ is not important.

The electromagnetic field calculation code. NEC. will be used to evaluate the possible E and H field intensities using a crude model of the Space Shuttle Cargo Bay. To provide a check of the NEC code results. an analytical calculation of the E field intensities at one or two points is feasible for the dipole in free space. The analytical calculation of the E and H fields for a dipole with the surrounding cargo bay support structure is not feasible. However. the estimation of the maximum E field possible using a simple "corner reflector" is feasible and will be presented. This will serve to guide interpretation of the NEC code.

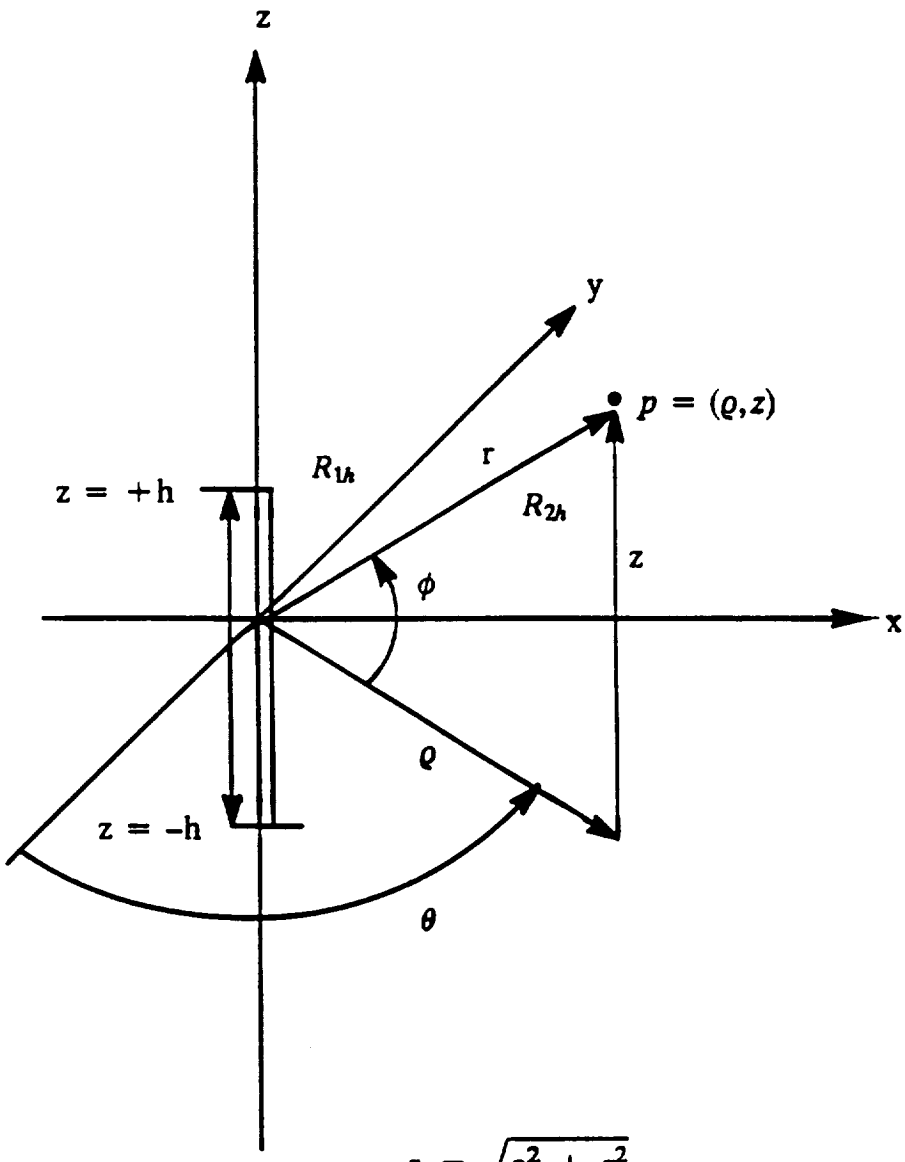
As a checkpoint for the NEC code. the E_ρ and E_z field components will be calculated for a free space half wave dipole using the analytical expressions. presented as Eq. 1 and Eq. 2. for two cases:

Case I Antenna length $\ell = 2h = \lambda/3$

Case II Antenna length $\ell = 2h = \lambda/2$.

For each case. the fields will be calculated at an arbitrary point x. y. z of (.6466. 0. –.6466) corresponding to the coordinate system illustrated in Figure 1.

Unfortunately. the text by King and Harrison uses cylindrical coordinates and the NEC code uses rectangular coordinates. Fortunately. the conversion between the two is easily accomplished.



$$r = \sqrt{\varrho^2 + z^2}$$

$$R_{1h} \triangleq \sqrt{(h-z)^2 + \varrho^2}$$

$$R_{2h} \triangleq \sqrt{(h+z)^2 + \varrho^2}$$

FIGURE 1 - GEOMETRY

1.1 Case I Antenna Length, $\ell = 2h = \lambda/3$

For this case, it is assumed that the antenna is in free space and hence $\alpha = 0$. The calculations can be accomplished for the dissipative medium of the F layer by specifying the attenuation constant α for the F layer.

The following parameters are assumed:

Antenna length tip-to-tip. $\ell = 50$ meters $= 2h$

Frequency, f = 2 MHz: Wavelength, $\lambda = \frac{c}{f} = 150$ *meters*

$$\mu = \mu_o = 4\pi \cdot 10^{-7}\ ;\quad c = 3 \cdot 10^{8}\ ;\quad k = \beta = \frac{\omega}{c} = 4.188 \cdot 10^{-2}\ ,$$

using mks units.

For the point of interest, we have the following values for the parameters indicated:

$$\varrho = \sqrt{x^2 + y^2} = .6466$$

$$r = \sqrt{\varrho^2 + z^2} = \sqrt{.6466^2 + (-.6466)^2} = .91443$$

$$R_{1h} = \sqrt{[25 - (-.6466)]^2 + .6466^2} = 25.655$$

$$R_{2h} = \sqrt{[25 + (-.6466)]^2 + .6466^2} = 24.362$$

$$\beta = \frac{\omega}{c} = \frac{2\pi}{\lambda} = 4.189 \cdot 10^{-2} = .04189$$

$$\alpha = 0.$$

The one remaining parameter is $I_z(o)$, the antenna driving point current. The NEC code is run with 1 volt applied to the antenna driving point. The resulting current is then calculated by the code after calculating the driving point impedance for the specified frequency.

For the analytical calculation, the driving point current is expressed in King and Harrison but is very messy to calculate. Thus, the antenna driving point impedance will be estimated using the ARRL Antenna Book curves for a dipole (2) [see Figures 50 and 51 of the next page].

For the parameters assumed we have as the electrical height in degrees, h = 25 meters $= \frac{\lambda}{6} =$ 60 degrees. The resulting estimate for the driving point impedance of a dipole of length $\ell = \lambda/3$ is

$$Z_{\substack{est \\ \frac{\lambda}{3}}} = 24 - j400 = 400.7\angle -86.6^\circ$$

The NEC code has calculated the driving point impedance for the $\lambda/3$ dipole as (3):

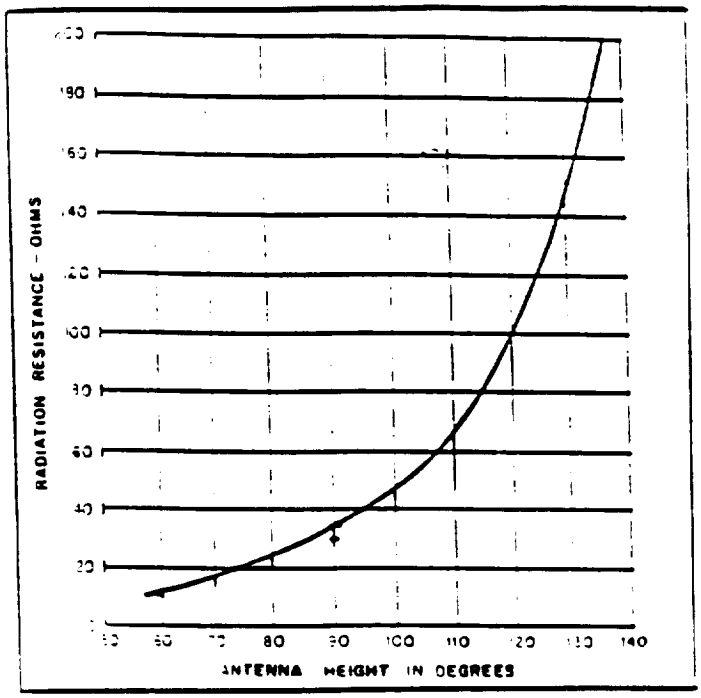


Fig. 49 — Radiation resistance vs. free-space antenna height in electrical degrees for a vertical antenna over perfectly conducting ground, or over a highly conducting ground plane. This curve also may be used for cylindrical antennas in free space by multiplying the radiation resistance by two; the height in this case is half the actual antenna length.

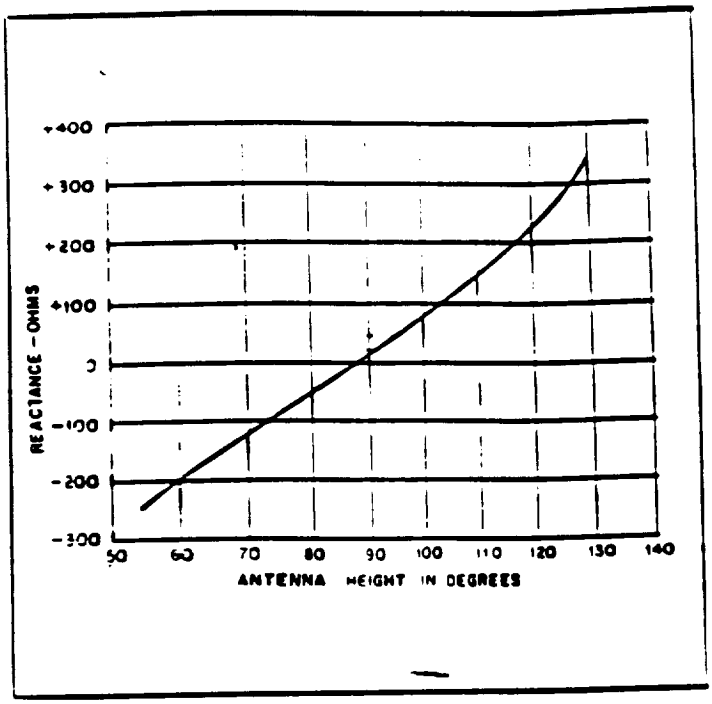


Fig. 51 — Approximate reactance of a vertical antenna over perfectly conducting ground and having a length/diameter ratio of about 1000. Actual values will vary considerably with length/diameter ratio. The remarks under Fig. 49 also apply to this curve.

terminals and can be tuned to resonance by means of a capacitance of the proper value. This is shown in Fig. 47A. As the length is increased progressively from 1/4 to 1/2 wavelength the current loop moves up the antenna, always being at a point 1/4 wavelength from the top. When the height is 1/2 wavelength the current distribution is as shown at B in Fig. 47. There is a voltage loop (current node) at the base, and power can be applied to the antenna through a parallel-tuned circuit, resonant at the same frequency as the antenna, as shown in the figure.

Up to a little more than 1/2 wavelength, increasing the height compresses the directive pattern in the vertical plane; this results in an increase in field strength for a given power input at the very low radiation angles. The theoretical improvement is about 1.7 dB for a half-wave antenna when compared with a quarter-wave antenna, as shown in Fig. 48A.

When the height of a vertical antenna is increased beyond a half wavelength, secondary lobes appear in the pattern, Fig. 48A and B. These become major lobes at relatively high angles when the length approaches 3/4 wavelength. At one wavelength the low-angle lobe disappears and a single lobe remains at approximately 35°.

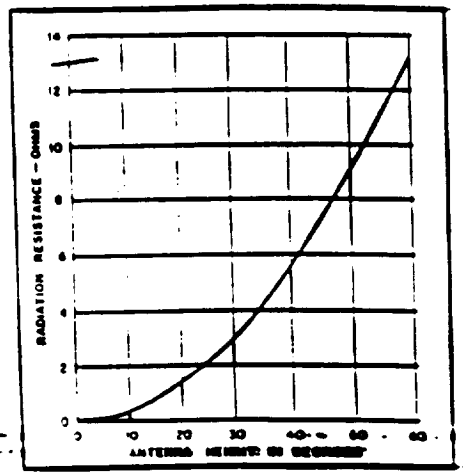


Fig. 50 — Same as Fig. 49, for heights below 60 degrees.

Radiation Resistance

The radiation resistance of a grounded vertical antenna, as measured between the base of the antenna and ground, varies as shown in Fig. 49 as a function of the antenna height. The word "height" as used in this connection has the same meaning as "length" as applied to a horizontal antenna. This curve is for an antenna based on (but not directly connected to) ground of perfect conductivity. The height is given in electrical degrees, the 60 to 135 degree range shown corresponding to heights from 1/6 to 3/8 wavelength. The antenna is approximately self-resonant at a height of 90 degrees (1/4 wavelength). The actual resonant height will be somewhat less because of the length/diameter ratio mentioned earlier.

In the range of heights covered by Fig. 49 the radiation resistance is practically independent of the length/diameter ratio. At greater heights the length/diameter ratio is important in determining the actual value of radiation resistance. At a height of 1/2 wavelength the radiation resistance may be as high as several thousand ohms.

The variation in radiation resistance with heights below 60 degrees is shown in Fig. 50. The values in this range are very low.

A very approximate curve of reactance vs. height is given in Fig. 51. The actual reactance will depend on the length/diameter ratio, so this curve should be used only as a rough guide. It is based on a ratio of about 1000 to 1. Thicker antennas can be expected to show lower reactance at a given height, and thinner antennas should show more. At heights below and above the range covered by the curve, large reactance values will be encountered, except for heights in the vicinity of 1/2 wavelength. In this region the reactance decreases, becoming zero when the antenna is resonant.

Efficiency

The *efficiency* of the antenna is the ratio of the radiation resistance to the total resistance of the system. The total resistance includes radiation resistance, resistance in conductors and dielectrics, including the resistance of loading coils if used, and the resistance of the grounding system, usually referred to as "ground resistance."

$$Z_{NEC} = 26.2 - j427.7 = 428.5\angle -86.5^\circ$$

The two impedances are in fairly good agreement (within approximately 5%), and the NEC code impedance should be the more correct as reading the curves from Reference 2 [Figures 49 and 51 of the previous page] leaves one room for error (the curves are not detailed in fine increments).

To maintain an independence from NEC code values in the analytical expressions, the estimated value for Z will be used.

The calculated $I_z(o)$ and the NEC code $I_z(o)$ values are:

$$I_{z_{est}}(o) = \frac{1}{Z_{est}} = 2.5 \cdot 10^{-3}\angle 86.6^\circ = 1.48 \cdot 10^{-4} + j2.495 \cdot 10^{-3}$$

and

$$I_{z_{NEC}}(o) = 2.33 \cdot 10^{-3}\angle 86.49^\circ = 1.43 \cdot 10^{-4} + j2.33 \cdot 10^{-3}$$

Again the results are in good agreement (approximately 6.8%).

Proceeding to the equation for $E_\rho(\vec{r})$, Eq. 1, we have:

$$E_\rho(\vec{r}) = \frac{j2\pi 2\cdot 10^6\ 4\pi\cdot 10^{-7}\ 2.5\cdot 10^{-3}\angle 86.6^\circ}{4\pi 4.188\cdot 10^{-2}(.6466)}\left[\frac{-25.6466}{25.655}\ e^{-j1.07443} + \frac{24.35}{24.36}\ e^{-j1.02} + 1.414e^{-j3.83\cdot 10^{-2}}\ e\cos(1.097)\right]$$

or

$$E_\rho(\vec{r}) = j1.16\cdot 10^{-1}\angle 86.6^\circ\left[-.476 + j.879 + .5231 - j.8517 + .7065 - j.02707\right]$$

$$= 1.16\cdot 10^{-1}\angle 176.6^\circ\left[0.7536 - j2.01\cdot 10^{-4}\right] = .087417\ \angle 176.6^\circ = -.08726 + j.00518$$

The $E_\rho(\vec{r})$ vector in cylindrical components is equivalent to the following composite sum of rectangular coordinates:

$$E_\rho(\vec{r}) = E_x\cos\phi + E_y\sin\phi$$

where $\phi = \tan^{-1} y/x$.

For our assumed x, y, z coordinates then we have $\phi = \tan^{-1}(o) = 0$ and

$$E_{\varrho}(\vec{r}) = E_{x} .$$

Thus, we may compare the calculated $E_{\varrho}(\vec{r})$ value at the coordinates (.6466, 0, -.6466) directly with the E_x value at that point as calculated by the NEC code.

$$E_{\underset{\frac{\lambda}{3}}{\varrho}}(\vec{r}) = -.08726 + j.00518 = .087417\angle 176.6^{\circ}$$

$$E_{\underset{\alpha = .034\,m}{\underset{\frac{\lambda}{3}}{x_{NEC}}}}(\vec{r}) = -.117 + j.0021 = 0.117\angle 179.7^{\circ}$$

The result is within a factor of approximately 25% in magnitude. Part of the contributing factor is the use of the actual antenna wire radius in the NEC code, whereas the analytical equation assumes an infinitesimally small wire. The NEC code also runs an analysis using the actual optimum antenna length versus the theoretical antenna length relative to the wavelength.

Now let's calculate the other field component $E_z(\vec{r})$. From Eq. 2 we have,

$$E_{\underset{\frac{\lambda}{3}}{z}}(\vec{r}) = \frac{-j\omega\mu I_z(o)}{4\pi k}\left[\frac{e^{-jkR_{1h}}}{R_{1h}} + \frac{e^{-jkR_{2h}}}{R_{2h}} - \frac{2}{r}\,(coskh)\,e^{-jkr}\right]$$

$$= \frac{-jw4\pi\cdot 10^{-7}\ 2.5\cdot 10^{-3}\angle 86.6^{\circ}}{4\pi\ \omega/c}\left[\frac{e^{-j1.07443}}{25.655} + \frac{e^{-j1.02}}{24.362} - \frac{2}{.91443}\cos(1.047)\,e^{-j.0383}\right]$$

$$= -j(.075)\angle 86.6^{\circ}\ [.018563 - j.0343 + .0215 - j.035 - 1.093 + j.0419]$$

$$= -j(.075)\ 86.6^{\circ}\ [-1.0529 - j.0274] = -j(.075)\angle 86.6^{\circ}\ [1.053256\angle -178.5^{\circ}]$$

$$E_{\underset{\frac{\lambda}{3}}{z}}(\vec{r}) = .079\angle -181.2^{\circ} = -.0788 + j.00165$$

$$E_{\underset{\alpha = .034\,m}{\underset{\frac{\lambda}{3}}{z_{NEC}}}}(\vec{r}) = .098\angle -179.29^{\circ} = -.098 - j.0012 .$$

Again, we see an agreement to within 19% in magnitude. [The value of α is the antenna wire diameter.]

The phase associated with the terms is related to the driving point voltage. Reversing the sign of the driving point voltage reverses the phase of the NEC voltages.

1.2 Case II Antenna Length, $\ell = 2h = \lambda/2$

As a second check, the two field components will be calculated for the half-wave dipole in free space at a specific point, x, y, z, (.6466, 0, -.6466).

For this case, we have the following parameters: f = 3 MHz, λ = 100 meters, α = 0; $\beta = \frac{\omega}{c} = \frac{2\pi}{\lambda} = \frac{\pi}{50}$, h = 25 meters = ℓ/z.

The same point for x, y, z will be used and, hence, the values for ϱ, r, R_{1h}, R_{2h} will remain the same. However, k = β is changed. Hence, $kR_{1h} = 6.28 \cdot 10^{-2}(25.655) = 1.612$, $kR_{2h} = 1.53$ and kr $= 5.74 \cdot 10^{-2}$.

From the ARRL antenna curves of Figures 49 and 50, we have for this case the estimated driving point impedance as

$$Z_{\substack{est \\ \lambda/2}} = 2(34.7 - j20) = 70 + j40 = 80.6 \angle 29.7^\circ .$$

The NEC code calculates

$$Z_{\substack{NEC \\ \frac{\lambda}{2}}} = 80 + j42.8 = 90.7 \angle 28.1^\circ .$$

The agreement is within 11.2% and is considered acceptable. The driving current as calculated and as seen by NEC are

$$I_{\substack{z \\ est \\ \frac{\lambda}{2}}} = 1.24 \cdot 10^{-2} \angle -29.7^\circ = 1.078 \cdot 10^{-2} - j6.15 \cdot 10^{-3}$$

and

$$I_{\substack{z \\ NEC \\ \frac{\lambda}{2}}} = 1.1 \cdot 10^{-2} \angle -28.1^\circ = 9.71 \cdot 10^{-3} - j5.19 \cdot 10^{-3}$$

and agreement is within 12.7% and is acceptable. A 1-volt driving point signal was assumed.

Using Eq. 1 for $E_\varrho(\bar{r})$, we have

$$E_{\rho}(\bar{r}) = \frac{jc\omega I_z(o)}{4\pi\rho}\left(-.9997e^{-j1.612} + .9996e^{-j1.53} + 1.414\cos(1.57)e^{-j5.74\cdot10^{-2}}\right)$$

$$= j46.4I_z(o)\left(+4.118\cdot10^{-2} + j.9988 + 4.076\cdot10^{-2} - j.99876 + 1.124\cdot10^{-3} - j6.459\cdot10^{-5}\right)$$

$$= .575\,\underline{|\,60.3^\circ}\left(8.3\cdot10^{-2} - j2.46\cdot10^{-5}\right) = .575\,\underline{|\,60.3^\circ}\left(8.3\cdot10^{-2}\,\underline{|\,-1.7\cdot10^{-2^\circ}}\right)$$

$$E_{\rho}(\bar{r}) = 4.77\cdot10^{-2}\,\underline{|\,60.3^\circ} = 2.36\cdot10^{-2} + j414\cdot10^{-2}$$

and

$$E_{\substack{NEC \\ \frac{\lambda}{2} \\ a=.034\,m}}(\bar{r}) = 9.88\cdot10^{-2}\,\underline{|\,160^\circ} = -9.28\cdot10^{-2} + j3.38\cdot10^{-2}$$

where a = wire diameter.

For E_z we have

$$E_z(\bar{r}) = -j30I_z(o)\left[\frac{e^{-j1.612}}{25.655} + \frac{e^{-j1.53}}{24.36} - \frac{2(8\cdot10^{-4})}{.91443}\,e^{-j5.74\cdot10^{-2}}\right]$$

$$= 3.72\cdot10^{-1}\,\underline{|\,-119.7^\circ}\left[-1.6\cdot10^{-3} + j1.56\cdot10^{-2} + 1.67\cdot10^{-3} - j4.1\cdot10^{-2} - 1.75\cdot10^{-3} + j1\cdot10^{-4}\right]$$

$$= .372\,\underline{|\,-120^\circ}\left[-1.68\cdot10^{-3} - j2.53\cdot10^{-2}\right] = .372\,\underline{|\,-120^\circ}\left(2.53\cdot10^{-2}\,\underline{|\,-93.8^\circ}\right)$$

$$E_z(\bar{r}) = 9.41\cdot10^{-3}\,\underline{|\,-213.8^\circ} = -7.8\cdot10^{-3} + j5.23\cdot10^{-3}$$

and

$$E_{\substack{z \\ NEC \\ \frac{\lambda}{2} \\ a=.034\,m}} = 1.15\cdot10^{-1}\,\underline{|\,175.8^\circ} = -114.7\cdot10^{-3} + j8.4\cdot10^{-3}$$

1.3 Comments on Results of Analysis Versus NEC Code

It should be noted that the differences between

E_{ρ} and $E_{\substack{z \\ NEC \\ \frac{\lambda}{2} \\ a=.034}}$ and again for E_z and $E_{\substack{z \\ NEC \\ \frac{\lambda}{2} \\ a=.034}}$

may be attributed to the .034 m wire diameter used in the NEC code versus the assumed zero diameter wire used in the analytical equations. To demonstrate the wire diameter effect, the NEC code was run for various wire diameters. Table 1 compares the results, and the asymptotic behavior of the NEC code is illustrated, as is the clear trend towards the mathematical analysis results for zero diameter wire.

To verify the NEC model, the analysis was calculated for $\lambda/3$ and $\lambda/2$ lengths as noted previously. The NEC code estimates are more realistic and accurate, utilize numerical solution of integral formulations that have no closed form analytic solutions. To check the scaling of results run for 1-volt driving point sources, the $\lambda/3$ free space dipole case was run for 10 volts as well. Table 2 illustrates the results. As expected, the E field scales in direct proportion to differences in $E_{\text{driving point}}$ and in proportion to the square root of the difference in radiated power.

This section has presented analytical results and NEC code calculations for comparison. At this point, one should have confidence that the NEC code results are will be close to those results predicted by analysis. The crowning feature of the NEC code is that it may be used to calculate fields for structures for which no analytical results are possible, such as the Space Shuttle Cargo Bay.

2.0 NEC CODE RESULTS FOR DIPOLE AND SHUTTLE CARGO

The field strengths that could be induced in the Shuttle Cargo Bay by the WISP dipole antenna are the items of interest. It is not feasible to derive an analytical solution for this structure. However, NEC can be used to calculate approximate field intensities in the cargo bay. The term approximate is used since it is not feasible to exactly model the Shuttle Cargo Bay in the NEC code.

An approximate model of the cargo bay may be comprised using a series of "patch plates" and an outside closed surface of large rectangular plates as is illustrated in Figure 2a. In Figure 2a, the WISP dipole is illustrated as a horizontal dipole lying along the x axis. The Shuttle Cargo Bay is depicted as lying along the y axis. It should be noted that the orientation of the antenna is in a different direction than the orientation used for the calculations of Section 1.0. This is for convenience in modeling the cargo bay with the NEC code.

From Figure 2a, one may observe the patch structure used to model the cargo bay. The coordinate points labeled in Figure 2a are always depicted x, y, z coordinates. It may be noted that the origin of the coordinate system is located 21" under the center of the WISP dipole as illustrated in the lower right-hand corner of Figure 2a. Figure 2b illustrates the interior of the cargo bay model in greater detail. For the NEC code calculation, a series of points in various x,z planes were calculated. These points were arbitrarily chosen. The lower figure in Figure 2b depicts the ten calculation points as located in the x,z plane for $y = 0$. The tail of the Shuttle is in the negative y direction, and the nose of the Shuttle is in the positive y direction. The ten calculated points in the x,z plane were also

TABLE 1

ILLUSTRATION OF WIRE DIAMETER EFFECT FOR NEC CODE ESTIMATE VERSUS IDEAL ANALYTIC SOLUTION

Antenna is $\lambda/2$ tip–to–tip, f = 3 MHz, λ = 100 meters, $\ell = 2h$ = 50 meters, $E_{\text{driving point}}$ = 1 volt, P_R = 4.63 mw
E fields observed at coordinates x, y, z = (.6466, 0, –.6466). Antenna as Figure 1.

a = wire diameter	E_x	E_z
NEC: a = .034 m	$.0988 \angle 160.6^\circ = -9.28\cdot10^{-2} + j3.38\cdot10^{-2}$	$.1150 \angle +175.8^\circ = -114.7\cdot10^{-3} + j8.4\cdot10^{-3}$
a = .00034 m	$.0508 \angle 132.0^\circ = -3.4\cdot10^{-2} + j3.8\cdot10^{-2}$	$.0732 \angle 172.5^\circ = -7.25\cdot10^{-2} + j9.55\cdot10^{-3}$
a = .0000034 m	$.0424 \angle 111^\circ = -1.5\cdot10^{-2} + j3.95\cdot10^{-2}$	$.0587 \angle 170^\circ = -5.78\cdot10^{-2} + j1.02\cdot10^{-2}$
a = .00000000 1 m	$.0428 \angle 96.3^\circ = -4.7\cdot10^{-3} + j4.25\cdot10^{-2}$	$.0491 \angle 168.25^\circ = -4.8\cdot10^{-2} + j9.99\cdot10^{-3}$
Analytic: a = 0 m	$.0477 \angle 60.3^\circ = 2.36\cdot10^{-2} + j4.14\cdot10^{-2}$	$.0941 \angle -213.8^\circ = -7.8\cdot10^{-3} + j5.23\cdot10^{-3}$

As the wire diameter approaches 0, the NEC code incurs numerical limitations. However, the trend towards the analytical results is illustrated.

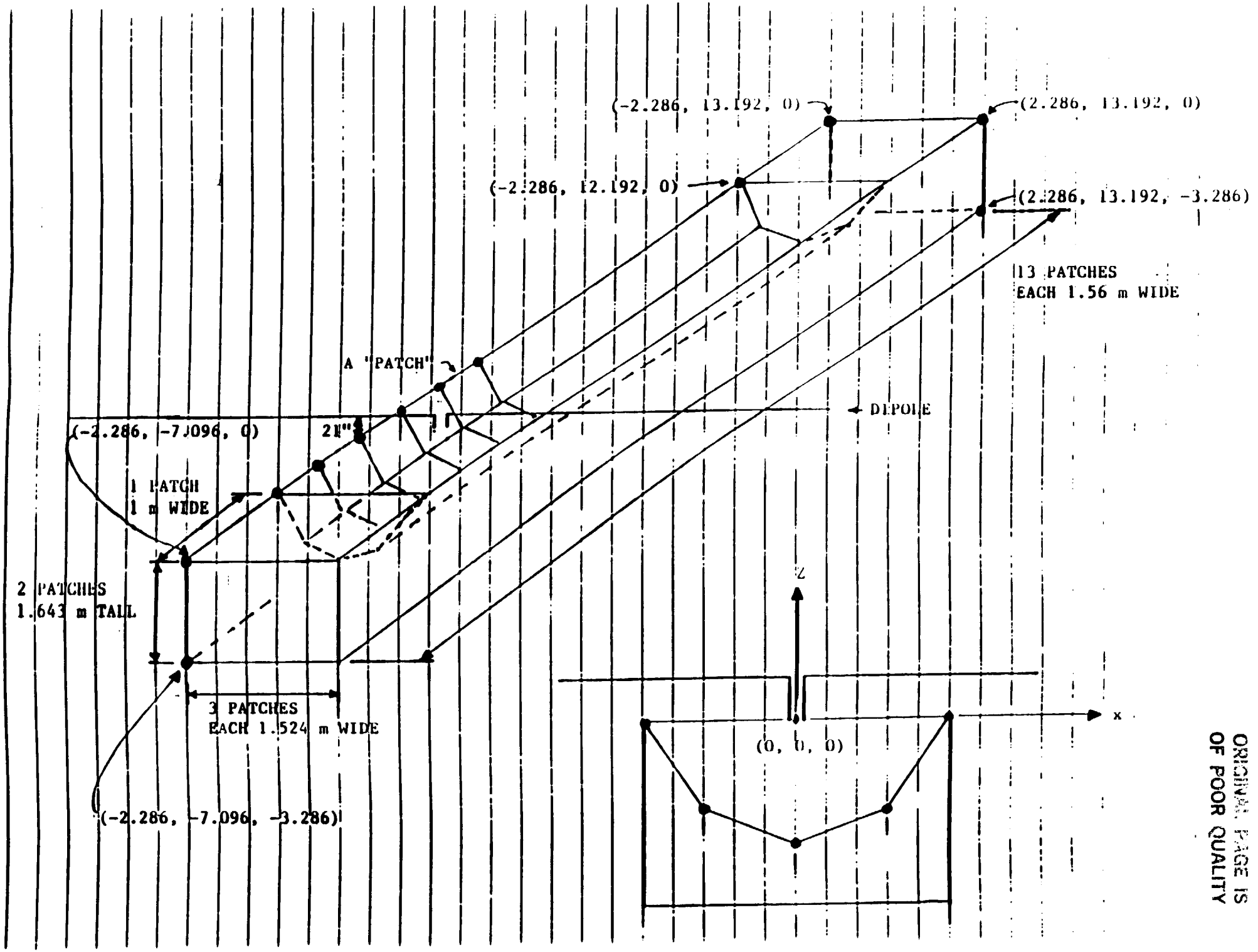
TABLE 2

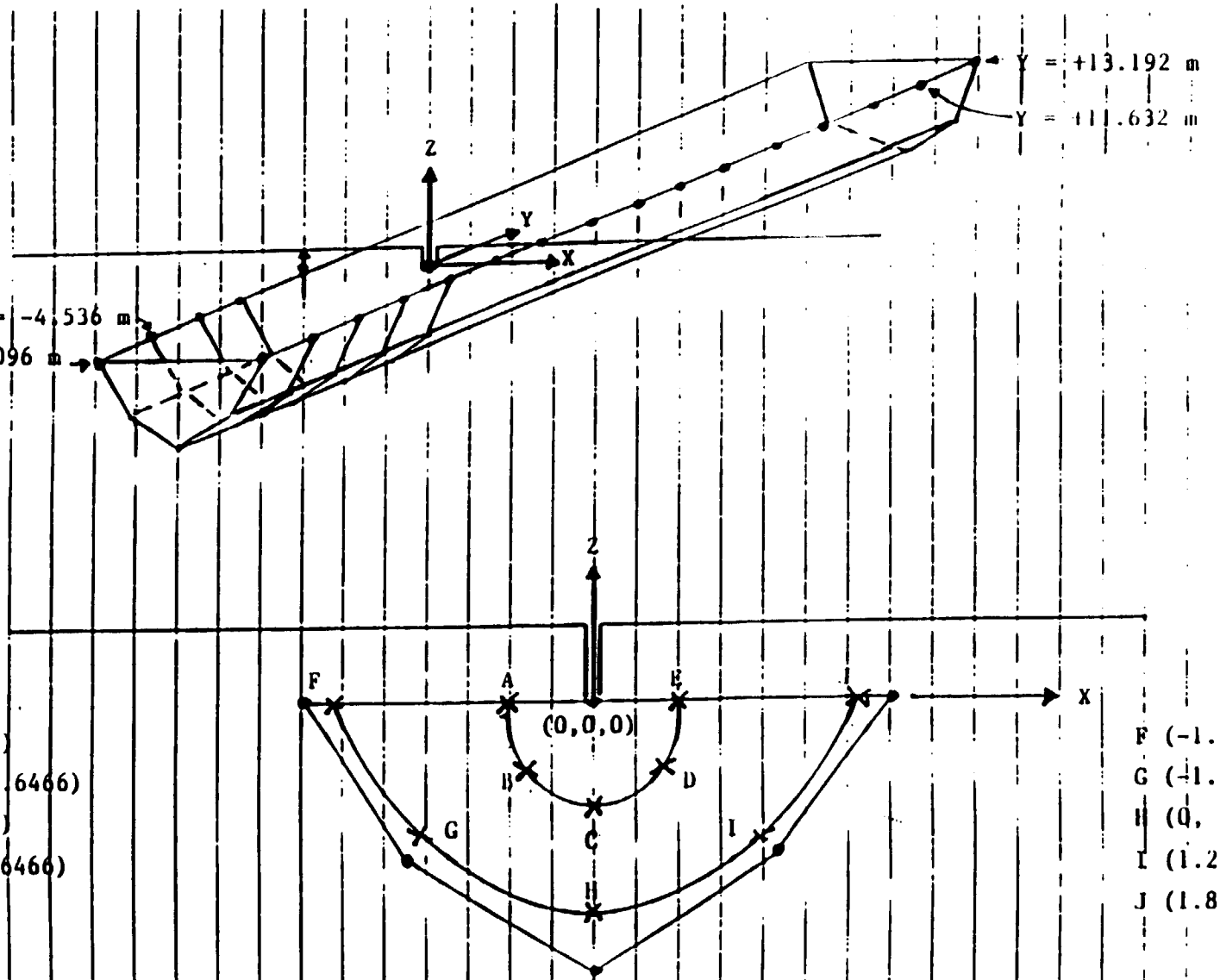
A CHECK ON SCALING 1 VOLT TO 10 VOLT SOURCES

Antenna is λ/3 tip–to–tip, f = 2 MHz, λ = 150 meters, ℓ = 2h = 50 meters, a = .034 m
E fields observed at coordinates x, y, z = (.6466, 0, –.6466). Antenna as Figure 1.
NEC estimated driving point impedance for all cases: z = 26 – j416.

$E_{driving\ point}$ (Volts)	E_x (Volts/meter)	E_z (Volts/meter)	P_r (Watts)
1	0.125 ⌊ 179.7°	0.103 ⌊ – 179.3°	0.000075
10	1.25 ⌊ 179.7°	1.03 ⌊ – 179.3°	0.0075
1000	125 ⌊ 179.7°	103 ⌊ – 179.3°	75
2000	250 ⌊ 179.7°	206 ⌊ – 179.3°	300

As expected, the fields scale in direct proportion to $E_{driving\ point}$ and in proportion to the square root of the power difference.





A (-.9144, 0, 0)
B (-.6466, 0, -.6466)
C (0, 0, -.9144)
D (.6466, 0, -.6466)
E (.9144, 0, 0)

F (-1.828, 0, 0)
G (-1.293, 0, -1.293)
H (0, 0, -1.828)
I (1.293, 0, -1.293)
J (1.828, 0, 0)

X = DATA CALCULATION POINT
COORDINATES ARE FOR THE Y=0 (X,Z PLANE) CUT

FIGURE 2b - SHUTTLE CARGO BAY MODEL

calculated in planes parallel to the x,z plane but for various values of y separated at 1.56 meter increments, such as y = -6.096 m, y = -4.536 m, etc., as illustrated in Figure 2b.

The NEC code results for the Shuttle Loading case are not reliable nor accurate. That this is the case is illustrated by the three sets of data in Table 3. Several potential reasons for the nonreliable results might be:

1. Patch size outside window of allowable parameters relative to the wavelength.
2. Patch coordinates incorrectly specified.
3. NEC not suitable for such close-in geometry.

The NEC code problems will be investigated in the following time period of October 1991. Hopefully, consistent results can be obtained.

3.0 CONCLUSIONS

From Table 2 one may observe that for the HALFWAVE DIPOLE in free space at a point within .646 meters of the antenna fairly large field strengths are to be expected. For 2000 volt driving point potential, 300 watts radiated power, a field strength of 250 volts/m could easily be expected. In the presence of the Shuttle, the structure could create an increase of 2 or more. Thus, field strengths of 500 v/m or more could exist in the Shuttle Cargo Bay near the antenna.

TABLE 3

NEC CODE RESULTS FOR SHUTTLE LOADING E-FIELD VALUES CALCULATED AT THE POINTS A, B, C, D, E OF FIGURE 2b FOR THREE PATCH SIZES

LACK OF SYMMETRY, |E| AT B NOT EQUAL TO |E| AT D, AND VARYING RESULTS AT THE SAME POINTS FOR VARYING PATCH SIZE INDICATE NON VALID RESULTS.

$E_{driving\ point}$ = 10 volts, $\ell = \lambda/2$.

	PATCH SIZE		
POINT	1.36 m SQUARE	0.73 m SQUARE	0.4 m SQUARE
A	20 V/m	18.4 V/m	12.5 V/m
B	18 V/m	13.6 V/m	10.8 V/m
C	15 V/m	8.6 V/m	11.20 V/m
D	15 V/m	5.9 V/m	20.96 V/m
E	9 V/m	6.1 V/m	30.25 V/m

4.0 REFERENCES

1. Antennas and Waves. King and Harrison. 1969. MIT Press. SBN 262-11033-4. page 257.
2. ARRL Antenna Handbook. 1984. ISBN 0-87259-414-9. pages 2-24. Figures 49 and 50.
3. NEC Code Calculations. Personal Memo. Bobby Cox. Mississippi State University. August 10. 1991.
4. The Corner-Reflector Antenna. J. D. Kraus. IRE. November 1940. pages 513-519.