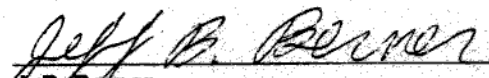


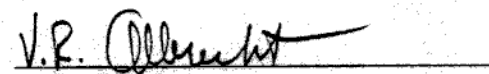
JPL D-4233

NASA Deep Space Command Detector Unit Development Final Engineering Report

Prepared by:


J. B. Berner
CDU Development Engineer

Approved by:


V. R. Albrecht
CDU Cognizant Engineer

9 JUNE 1987

National Aeronautics and
Space Administration

JPL

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

CONTENTS

PART I - INTRODUCTION	
1.	INTRODUCTION 1-1
1.1	ORGANIZATION OF THE REPORT 1-1
1.2	METHODS USED 1-2
1.3	ACKNOWLEDGEMENTS 1-2
PART II - ANALYSIS	
2.	OVERVIEW 2-1
2.1	THE CDU IN GENERAL 2-1
2.2	BASIC SYSTEM DESCRIPTION 2-2
2.2.1	Analog Circuitry 2-2
2.2.2	Subcarrier Tracking 2-2
2.2.3	Automatic Gain Control 2-3
2.2.4	Bit Synchronizer 2-3
2.2.5	Lock Detector 2-4
2.2.6	Multiple Data Rates 2-4
2.3	DIFFERENCES BETWEEN THE STANDARD CDU AND THE DEEP SPACE CDU 2-4
2.4	NXT/CDU INTERFACE 2-6
2.5	REFERENCES 2-7
3.	ANALOG SECTIONS 3-1
3.1	THE BANDPASS FILTER 3-1
3.1.1	Transfer Function 3-2
3.1.2	Noise Bandwidth 3-5
3.1.3	3 dB Bandwidth 3-10
3.1.4	Filter Results 3-12

CONTENTS (cont'd)

3.2	THE AGC AMPLIFIER	3-12
3.2.1	The R-2R Ladder Network	3-12
3.2.2	The DAC	3-16
3.2.3	The Dual Gain Amplifier	3-17
3.2.3.1	Out-of-Lock	3-18
3.2.3.2	In-Lock	3-19
3.3	REFERENCES	3-20
4.	THE SUBCARRIER TRACKING LOOP	4-1
4.1	SUBCARRIER LOOP MODEL	4-1
4.2	ANALYSIS	4-2
4.2.1	Coherent Sampling Demodulation	4-2
4.2.2	Derivation of the Closed Loop Transfer Function	4-5
4.2.2.1	Assumptions	4-5
4.2.2.2	Z-Transform	4-6
4.2.3	Derivation of B_L	4-10
4.2.4	Derivation of Loop Jitter	4-11
4.2.5	Derivation of the Probability of Cycle Slip	4-11
4.3	DERIVATION OF THE COEFFICIENTS	4-12
4.3.1	ESCL(r)	4-12
4.3.2	ECF1	4-13
4.3.3	ECF0	4-14
4.4	RESULTS	4-16
4.4.1	Data Rate Dependent Results	4-16
4.4.2	Phase Jitter	4-17

CONTENTS (cont'd)

4.4.3	Probability of Cycle Slip	4-17
4.4.4	Doppler Rate Induced Phase Error	4-18
4.5	REFERENCES	4-19
4A	Z-TRANSFORM OF AN ACCUMULATE-AND-DUMP	4-21
4B	Z-TRANSFORM OF THE DIGITAL FILTER	4-25
4C	Z-TRANSFORM OF THE DIGITALLY CONTROLLED OSCILLATOR	4-25
4D	CALCULATION OF I_2	4-29
5.	THE AUTOMATIC GAIN CONTROL LOOP	5-1
5.1	AGC LOOP MODEL	5-1
5.1.1	AGC Loop Components	5-1
5.1.2	Assumptions	5-3
5.1.3	Simplified AGC Model	5-3
5.2	ANALYSIS	5-5
5.2.1	Large Scale Analysis	5-5
5.2.2	Small Signal Analysis	5-6
5.2.3	Z-Transform AGC Analysis	5-10
5.3	DERIVATION OF THE COEFFICIENTS	5-13
5.3.1	Derivation of α	5-13
5.3.2	Derivation of ASCL	5-15
5.3.2.1	Limit Due to the ADC Saturation	5-15
5.3.2.2	Limit Due to Accumulator Overflow	5-19
5.3.2.3	Calculating <DACC>	5-21
5.3.2.4	Calculating ASCL	5-23

CONTENTS (cont'd)

5.4	RESULTS	5-24
5.4.1	Derivation Results	5-24
5.4.2	Larger Signal Analysis Results	5-24
5.5	REFERENCES	5-25
5A.	IMPLEMENTATION OF e^x IN SOFTWARE	5-26
6.	BIT TIMING SYNCHRONIZATION LOOP	6-1
6.1	BIT SYNC TRACKING LOOP MODEL	6-1
6.1.1	Bit Sync Loop Block Diagram	6-1
6.1.2	Assumptions	6-3
6.1.3	Equivalent Loop Model	6-4
6.1.4	Loop Transfer Function	6-6
6.1.5	Loop Noise Bandwidth	6-7
6.1.6	Mean-Square Timing Jitter	6-8
6.2	ANALYSIS OF THE BIT SYNC LOOP	6-9
6.2.1	Bit Sync Loop Values	6-9
6.2.2	Derivation of BSCL	6-10
6.3	RESULTS	6-13
6.4	REFERENCES	6-14
7.	LOCK DETECTOR	7-1
7.1	LOCK DETECTOR MODEL	7-1
7.1.1	In-Lock Model	7-2
7.1.2	Out-Of-Lock Model	7-2

CONTENTS (cont'd)

7.2	ANALYSIS	7-3
7.2.1	Derivation	7-3
7.2.1.1	Density of the ADC	7-3
7.2.1.2	Density of Z_n	7-9
7.2.2	Implementation	7-13
7.3	THRESHOLD NUMBER CALCULATIONS	7-13
7.3.1	Unlock-to-Lock Probabilities	7-14
7.3.2	Lock-to-Unlock Probabilities	7-16
7.4	RESULTS	7-18
7.4.1	Threshold Numbers	7-18
7.4.2	Threshold Figures	7-19
7.5	REFERENCES	7-34
7A	PROBABILITY THAT AN EVENT OCCURS AT LEAST TWO SUCCESSIVE TIMES DURING N TRIALS, GIVEN THE PROBABILITY OF ITS OCCURRENCE IN A SINGLE TRIAL	7-35
8.	SINGLE EVENT UPSETS AND SNR TELEMETRY WORD	8-1
8.1	SINGLE EVENT UPSETS	8-1
8.2	SNR TELEMETRY WORD	8-2
8.3	REFERENCES	8-6
PART III - IMPLEMENTATION		
9.	HARDWARE	9-1
9.1	ANALOG SECTION	9-1
9.1.1	Automatic Gain Control Amplifier	9-2
9.1.2	Sample-and-Hold/ADC	9-3

CONTENTS (cont'd)

9.2	DIGITAL SECTION	9-4
9.2.1	System Timing	9-4
9.2.2	Single Event Upsets	9-5
9.2.2.1	Single Event Upset Protection Circuit . .	9-6
9.2.2.2	Single Event Upset Timing Considerations	9-6
9.3	REFERENCES	9-8
10.	CDU SOFTWARE	10-1
10.1	CHOICE OF LANGUAGE	10-1
10.2	SAMPLING SCHEME	10-1
10.3	SOFTWARE DESCRIPTION	10-3
10.3.1	INIT	10-3
10.3.2	NORM	10-5
10.3.3	MB	10-5
10.3.4	EOB	10-5
10.4	SOFTWARE OUTPUT	10-6
10.4.1	CDU Status Word	10-6
10.4.2	CDU SNR Word	10-6
10.5	SEU HANDLING	10-6
10.6	REFERENCES	10-7
10A	SOFTWARE FLOW CHARTS	10-9
10B	CDU CODE	10-21

CONTENTS (cont'd)

PART IV - TEST RESULTS

11.	TEST RESULTS	11-1
11.1	BER TESTS	11-1
11.2	PROBABILITY OF FALSE ACQUISITION	11-1
11.3	PROBABILITY OF DE-ACQUISITION	11-11
11.4	PROBABILITY OF OUT-OF-LOCK	11-12
11.5	PROBABILITY OF ACQUISITION	11-12
11.6	SUBCARRIER JITTER	11-14
11.7	BIT SYNCHRONIZATION JITTER	11-18
11.8	PROBABILITY OF CYCLE SLIP	11-19
11.9	CONCLUSION	11-22
11.10	REFERENCES	11-22

CONTENTS (cont'd)

Figures

2-1.	CDU Block Diagram	2-3
3-1.	Bandpass Filter Schematic	3-1
3-2.	First Stage Schematic	3-1
3-3.	BPF Magnitude	3-13
3-4.	BPF Phase	3-13
3-5.	Resistor Ladder	3-14
3-6.	Amplifier	3-14
4-1.	Subcarrier Loop Block Diagram	4-1
4-2.	Simplified Block Diagram	4-6
4A-1.	Frequency Response - Accumulate-and-Dump	4-24
4A-2.	Frequency Response - Integrate-and-Dump	4-24
4B-1.	Digital Filter Block Diagram	4-25
4C-1.	Time Periods Between Samples	4-26
5-1.	AGC Block Diagram	5-2
5-2.	Simplified AGC Model	5-3
5-3.	AGC Recovery From Large Step Input	5-25
6-1.	Bit Sync Tracking Loop Block Diagram	6-1
6-2.	Equivalent Data Transition Tracking Loop	6-4
7-1.	Lock Detector Block Diagram	7-1
7-2.	Quantized Density of $x(t)$	7-5
7-3.	Threshold Value, $r = 3$, Rate = 500 bps	7-20
7-4.	Threshold Value, $r = 4$, Rate = 250 bps	7-21
7-5.	Threshold Value, $r = 5$, Rate = 125 bps	7-22
7-6.	Threshold Value, $r = 6$, Rate = 62.5 bps	7-23

CONTENTS (cont'd)

Figures

7-7.	Threshold Value, $r = 7$, Rate = 31.25 bps	7-24
7-8.	Threshold Value, $r = 8$, Rate = 15.625 bps	7-25
7-9.	Threshold Value, $r = 9$, Rate = 7.8125 bps	7-26
7-10.	Threshold Value, $r = 3$, Rate = 500 bps	7-27
7-11.	Threshold Value, $r = 4$, Rate = 250 bps	7-28
7-12.	Threshold Value, $r = 5$, Rate = 125 bps	7-29
7-13.	Threshold Value, $r = 6$, Rate = 62.5 bps	7-30
7-14.	Threshold Value, $r = 7$, Rate = 31.25 bps	7-31
7-15.	Threshold Value, $r = 8$, Rate = 15.625 bps	7-32
7-16.	Threshold Value, $r = 9$, Rate = 7.8125 bps	7-33
8-1.	Input Network	8-2
9-1.	NASA Deep Space CDU Breadboard Block Diagram	9-1
9-2.	AGC Variable Gain Amplifier	9-2
9-3.	Track/Hold Amplifier and ADC	9-3
9-4.	ADC Conversion Timing	9-4
9-5.	Digital Block Diagram	9-5
9-6.	SEU Reset Circuit	9-6
9-7.	Protection Circuit Timing Diagram	9-7
9-8.	Timing for First Stage Recovery Circuit	9-8
10-1.	Sample Timing	10-2
10-2.	CDU Processor Timing	10-4
10A-1.	Data Detection Algorithm	10-10
10A-2.	Lock Detection Algorithm	10-11
10A-3.	Subcarrier Tracking Algorithm	10-12

CONTENTS (cont'd)

Figures

10A-4.	Bit Tracking Algorithm	10-14
10A-5.	Automatic Gain Control Algorithm	10-18
11.1-1.	BER Test Data Summary - Bit Rate = 500 bps	11-4
11.1-2.	BER Test Data Summary - Bit Rate = 250 bps	11-5
11.1-3.	BER Test Data Summary - Bit Rate = 125 bps	11-6
11.1-4.	BER Test Data Summary - Bit Rate = 62.5 bps	11-7
11.1-5.	BER Test Data Summary - Bit Rate = 31.25 bps	11-8
11.1-6.	BER Test Data Summary - Bit Rate = 15.625 bps	11-9
11.1-7.	BER Test Data Summary - Bit Rate = 7.8125 bps	11-10
11.5-1.	Probability of Acquisition - Bit Rate = 500 bps	11-15
11.5-2.	Probability of Acquisition - Bit Rate = 31.25 bps	11-16
11.5-3.	Probability of Acquisition - Bit Rate = 7.8125 bps	11-17
11.8-1.	Probability of Cycle Slip - Bit Rate = 500 bps	11-21

Tables

4-1.	Values for ESCL, B_L , and ω_n	4-16
5.3-1.	Calculations for <DACC>	5-23
5.4-1.	Results of Equations 5.2-40 and 5.3-36	5-24
6-1.	Data Rate Dependent Bounds on BSCL	6-12
6-2.	In-Lock Values	6-13
6-3.	Out-of-Lock Values	6-14
7.4-1.	Threshold Values	7-19
7A-1.	Values of $A_{i,N}$	7-36

CONTENTS (cont'd)

Tables

7A-2.	Required Single Event Probabilities	7-37
11.0-1.	CDU Breadboard Test Matrix	11-2
11.2-1.	False Acquisition Probability Confidence	11-11
11.3-1.	De-Acquisition Probability Confidence	11-12
11.4-1.	Out-of-Lock Probability Confidence	11-13
11.5-1.	Probability of Acquisition for 10 dB	11-13
11.5-2.	Probability of Acquisition for Lower ST/N_0	11-14
11.6-1.	Subcarrier Jitter, ST/N_0 Equal to 10 dB	11-18
11.7-1.	Bit Sync Jitter, ST/N_0 Equal to 10 dB	11-19
11.8-1.	Probability of Cycle Slip	11-20

SECTION 1

INTRODUCTION

The purpose of this report is to provide a single source of information about the design and workings of the NASA Deep Space Command Detector Unit (CDU) and to document the results of the testing of the breadboard model hardware. The goal of this report is to supply sufficient information so that anyone can read it and understand the design configuration, signal processing algorithms, and the logic behind certain decisions that were made.

The CDU development's directive was to design and develop a command detector based on the NASA Standard CDU (Standard) design, but using the Jet Propulsion Laboratory (JPL) flight qualified 80C86 microprocessor family chip set, and the same signal processing algorithms, implemented in 80C86 assembly language. This was done to minimize cost of analysis and to maximize the carry over from a unit that has been operating successfully in near-earth applications for several years.

1.1 ORGANIZATION OF THE REPORT

The report contains eleven sections, which are broken down into four parts: Introduction, Analysis, Implementation, and Test Results. The sections are listed below:

Part I - Introduction

Section 1	Introduction
-----------	--------------

Part II - Analysis

Section 2	Overview
Section 3	Analog Sections
Section 4	Subcarrier Tracking Loop
Section 5	AGC Loop
Section 6	Bit Sync Tracking Loop
Section 7	Lock Detector Loop
Section 8	Single Event Upsets/SNR Word

Part III - Implementation

Section 9	Hardware
Section 10	Software

Part IV - Test Results

Section 11	Test Results
------------	--------------

1.2 METHODS USED

The main body of the CDU development draws upon two documents:

- (1) NASA Standard Command Detector Unit Engineering Report, Motorola, February 1977. This document contains the analysis used for the Standard; thus, it was heavily referred to during the CDU development.
- (2) Design Requirement, NASA Deep Space Command Detector Unit, Document number DM514438, Rev. A, V. R. Albrecht, August 1986. This document provides the design requirements for the CDU implementation.

Since the CDU is primarily a digital system, most of the analysis is done in the discrete time domain, using z-transform theory.

1.3 ACKNOWLEDGEMENTS

The CDU development is the result of the hard work of a lot of people. In particular, the following people were directly involved with the design, implementation, and testing of the CDU: Vic Albrecht (design and task manager), Jeff Berner (analysis and software), Gene Hightower (software), Frank Kollar (hardware and software), Jeff Packard (test equipment), Kermit Peterson (testing), and Greg Stark (hardware and software).

Also deserving mention and thanks are M. A. Koerner and Dr. M. K. Simon, who answered many questions and provided guidance during the design and analysis phase.

SECTION 2

OVERVIEW

This section begins with a general description of the Deep Space CDU, moves on to describe the design in a little more detail, discusses the differences between the Standard CDU and the Deep Space CDU, and then describes the command channel interface characteristics.

2.1 THE CDU IN GENERAL

The NASA Deep Space CDU is a coherent demodulator for non-return to zero (NRZ) data, binary phase-shift-keyed (PSK) modulated onto a 16 kHz sinusoidal subcarrier. The unit receives the subcarrier from the NASA X-band Transponder (NXT).

Functionally, the CDU consists of a coherent automatic gain control (AGC) system, a sample-and-hold (S/H) circuit, an analog-to-digital converter (ADC), a second order data-aided subcarrier tracking loop, a data-transition bit synchronization loop, and a lock detector. Structurally, the CDU consists of the signal-conditioning assemblies (AGC amplifier, S/H, ADC), and a fully buffered microprocessor with an associated read-only memory (ROM) and a random-access memory (RAM). Appropriate output buffers are provided to interface with the spacecraft command and data handling subsystem for command data transfer, engineering telemetry, and data rate control.

An important part of the CDU, which is resident in the NXT, is the bandpass filter (BPF). The BPF, which has a noise bandwidth of about 4 kHz, filters the signal that is input to the CDU. This filtering reduces the noise that enters the CDU and bandlimits the signal to one-half the sampling rate, which allows sampling without aliasing.

The coherent AGC system, which keeps the signal power constant, utilizes a logarithmic-linear digitally controlled amplifier with a dynamic range in excess of 40 dB. The AGC loop performance (i.e., noise jitter, settling time, etc.) is determined by the AGC loop algorithm and loop coefficients stored in the ROM.

Upon receipt of the appropriate command from the digital processing assembly, the sample-and-hold circuit freezes the analog input to the eight-bit successive-approximation ADC and keeps it constant over the ADC conversion time.

In contrast to the conventional data-aided loop implementation, which requires mixers, filters, and ADCs in both the inphase and quadrature channels, the subcarrier tracking loop takes advantage of the digital implementation by creating both the inphase and quadrature synchronization channels with appropriate dedicated sample accumulators within the digital processing assembly. The requirement for data-rate dependent digital filtering in each of the two channels is eliminated by maintaining a constant sampling rate over all data rates. A second order subcarrier loop is implemented in the tracking

algorithm, with ROM-resident loop coefficients selected to minimize steady-state phase jitter, while meeting all specifications on acquisition time. The subcarrier loop resolution is 1/64 of a cycle with a maximum allowable single phase correction of 45 degrees.

The data-transition bit synchronization loop is driven by the contents of another dedicated sample accumulator within the digital processing assembly. The CDU design takes advantage of coherence between the subcarrier and the data so that the bit sync loop resolution need only be accurate to within a subcarrier cycle to provide effective bit sync resolution of 1/64 of a subcarrier cycle. The maximum allowable phase correction is one-quarter of a bit interval, with corrections made every eight data transitions.

The lock detection algorithm is also ROM-resident, with coefficients chosen to exceed the specifications for probability of acquisition and deacquisition. Again, all coefficients, and the algorithm itself, are completely ROM-resident.

2.1 BASIC SYSTEM DESCRIPTION

The CDU uses a coherent sampling approach to demodulate the subcarrier. This removes the subcarrier and translates the input sequences to baseband. By sampling in quadrature, one channel represents the inphase, or data, channel and the other represents the quadrature, or error, channel. The data channel is used to perform bit synchronization, data detection, lock detection, and AGC'ing, while the error channel is used to perform subcarrier tracking.

A block diagram of the CDU is presented in Figure 2-1 and a description of each of the individual components follows.

2.2.1 Analog Circuitry

The analog input circuitry consists of the AGC amplifier, the sample-and-hold, and the ADC. The AGC amplifier, discussed in detail in Section 3, is a digitally controlled amplifier, whose gain is set by the AGC control word received from the processing assembly. The sample-and-hold amplifier, operating in the track-and-hold mode, takes an error sample and a data sample every 125 microseconds (every other subcarrier cycle). This sample is input to the ADC for quantizing and coding. The CDU uses a high-speed 12-bit ADC with a ± 5 volt input range, which produces a bipolar complementary offset binary coded output, which is later converted to a two's complement output. The sample-and-hold and ADC are discussed further in Section 7.

2.2.2 Subcarrier Tracking

To obtain the best performance, a data-aided subcarrier tracking loop is used. The loop is a perfect second-order loop, so it can handle expected Doppler effects associated with deep space missions. Subcarrier

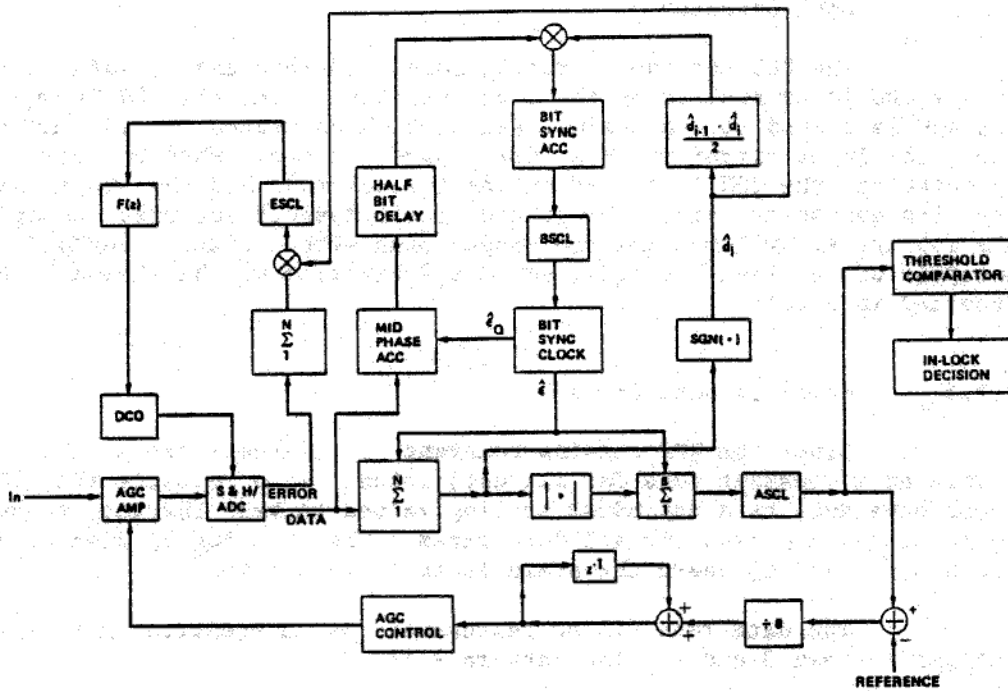


Figure 2-1. CDU Block Diagram

tracking corrections are made at the end of each bit, which limits the amount of Doppler that can be tracked. However, for the expected offsets, no appreciable bit error rate degradation is expected. The subcarrier tracking loop design is discussed in detail in Section 4.

2.2.3 Automatic Gain Control

A coherent AGC loop is used in order to control the input signal power over the anticipated range of signal levels from the NXT. This loop configuration allows the loop dynamics to be held relatively constant for all input levels and signal-to-noise ratios. Its model is developed and analyzed in detail in Section 5.

2.2.4 Bit Synchronizer

The data transition bit synchronizer loop implementation creates a measure of the normalized synchronization error by integrating the data channel over a bit time for two cases: from the estimate of the beginning of the bit, and from the estimate of the middle of the bit. A tracking correction is made (if needed) every eight data transitions. The loop is analyzed and further discussed in Section 6.

2.2.5 Lock Detector

The CDU has two operating modes, in-lock and out-of-lock; the mode of the CDU is controlled by the lock detector. When the CDU is out-of-lock, the AGC is forced to its maximum gain, which saturates the AGC amplifier; thus, the input circuitry acts like a hard limiter. When an uplink subcarrier is detected, the CDU is placed in the in-lock mode and the AGC is allowed to seek its quiescent gain. The lock detector detects the presence or absence of the subcarrier by comparing the signal mean with a fixed threshold number. An analysis of the lock detector, and the derivation of the threshold numbers, is provided in Section 7.

2.2.6 Multiple Data Rates

Since the BPF remains constant for all data rates, the CDU must sample at a constant rate for all data rates. This requires that the various loops have data rate dependent scaling values to keep the loop bandwidths and performances constant for all data rates. This scaling is discussed in Sections 4, 5, and 6, where the scale factors are derived.

The data rate can be represented as an equation in r , where r is an integer between 3 and 9. The data rate is

$$DR(r) = 4000/2^r \quad (2.2-1)$$

Thus, the available data rates are: 500, 250, 125, 62.5, 31.25, 15.625, and 7.8125 bps.

2.3 DIFFERENCES BETWEEN THE STANDARD CDU AND THE DEEP SPACE CDU

In implementing the CDU with the 80C86 microprocessor family of parts, several differences between the CDU and the Standard developed. Since this design began with the Standard as a baseline, a listing of the differences that the change in hardware forced is provided. These differences are:

- (1) The CDU hardware implementation uses hardware that is JPL flight approved class S parts, with the design centered around the 80C86 microprocessor family of parts. The Standard used Level-B custom LSI devices.
- (2) The CDU takes a set of samples (one error, one data) every other subcarrier cycle (8 kHz sampling rate). The Standard takes a set of samples every subcarrier cycle. The reason for doing this is that the 80C86 is not fast enough to do the processing needed between sample sets in one subcarrier cycle. The Standard's microprocessor was custom designed for this task, so it could handle the processing in one subcarrier cycle. The only effect of this change that the user

sees is that the maximum possible data rate is reduced [see item (3)]. At the maximum data rate, 500 bps, the CDU is operating at $8 \text{ k}/(2*500) = 8$ times the Nyquist rate, causing no degradation due to sampling.

- (3) The maximum data rate of the CDU is 500 bps; the standard also had bit rates of 2000 and 1000 bps. As was explained in item (2), the 80C86 cannot process at the speed required to handle 1000 and 2000 bps.
- (4) The pre-detection filter of the CDU is narrower than the filter (noise bandwidth = 12 kHz) of the Standard. This was done to bring the noise bandwidth of the filter down to 4 kHz, one-half of the sampling rate. This could be accomplished because the maximum data rate of the CDU is one-fourth of the Standard's maximum rate; thus, the 3 dB bandwidth could have been cut up to a factor of 4. The bandpass filter, which is a part of the NXT, is a 4-pole bandpass (2-pole low pass equivalent) Butterworth filter with a measured noise bandwidth of 3.907 kHz.
- (5) The data rate dependent scaling factors in the CDU signal processing algorithms are different than the Standard's. This is due to the changes in the sampling rate and the noise bandwidth of the predetection filter. The CDU signal processing algorithms are the same; only the scaling factors have changed. These changes are transparent to the user.
- (6) The CDU has hardware and software that provide single event upset (SEU) reset capability; the Standard has no SEU reset capability. This was added because the 80C86 is susceptible to SEUs. An SEU reset causes the CDU to drop lock and to reinitialize the software to prevent the CDU from going into an infinite loop or a halt state.
- (7) The various signal processing algorithms occur at different times during a data bit. Thus, the software is broken up into divisions that reflect the part of the bit in which the processing is done (i.e., EOB, MB, etc.). The division of the CDU's software into these blocks of processing time is now two sample period increments, instead of the Standard's one sample period increment. Again, this was due to the relatively slow speed of the 80C86.
- (8) The AGC amplifier's DAC is implemented with discrete parts and uses only the nine most significant bits (the tenth bit is always on). The Standard uses an integrated circuit (AD7520) for its DAC, but this part is not JPL flight qualified and could not be used in the CDU design. The fact that the least significant bit is always on does not affect the AGC performance in the range that the CDU is operated in.

- (9) The parameter that controls the bit sync tracking loop bandwidth, BSCL, has different values for the in-lock condition and the out-of-lock condition for the lower bit rates (the Standard had one value for both conditions, due to limitations in ROM size, which the CDU does not have). This was done to speed up the locking up of the bit sync loop without degrading the tracking [i.e., the bit error rate (BER)] performance. The Standard had to trade between lock up time and BER performance; the CDU can optimize both.

2.4 NXT/CDU INTERFACE

The CDU interface with the NXT is designed to accommodate the following interface characteristics:

NXT

Output impedance	≤10 ohms maximum (in series with a 0.01μF output capacitor)
Output signal level (maximum range from all sources including uplink mod-index of 0.5-1.3 rad.)	50-300 mV rms
Output variations in signal level (temperature, life, radiation, etc.)	±2.5 dB
Output noise level (receiver locked, data rate = 7.8125 bps)	2.0 V rms max
Output noise level (receiver out-of-lock)	15 V peak max
Output SNR ($S/N_0 = 10.5$ dB command threshold, data rate = 7.8125 bps, noise bandwidth = 4 kHz)	-16.5 dB
Prediction bandpass filter (part of NXT):	
• Type	4-pole Butterworth
• center frequency	16 kHz
• ±3 dB response	3.3 kHz minimum
• noise bandwidth	4 kHz maximum

CDU

Input impedance	10 kohms ±5% (in series with a 0.1 μF input capacitor)
-----------------	---

2.5 REFERENCES

- 2-1 NASA Standard Command Detector Unit Engineering Report, Motorola, February 1977.
- 2-2 CDU PDR Action Item 5 Response, IOM 3392-86-146, Jeff Berner to L.W. Randolph, October 8, 1986 (a JPL internal document).
- 2-3 Albrecht, V. R., Design Requirements NASA Deep Space Command Detector Unit, DM514438, Rev. A, August 1986.

SECTION 3

ANALOG SECTIONS

To support subsequent sections of this report, there are two analog circuits that require analysis, the bandpass filter and the AGC amplifier. The AGC amplifier gain and the predetection bandpass filter (BPF) noise bandwidth are needed in Section 7.

3.1 THE BANDPASS FILTER

A schematic of the BPF used for the breadboard test is shown in Figure 3-1. It is a two stage filter, with each stage being the same general form; this was used to simulate the command channel predetection filter of the transponder. Thus, to find the transfer function and the noise bandwidth, we can look at one stage and then cascade the transfer functions.

To find the transfer function of the BPF, we will work with just one stage. The schematic of this stage, and the node current definitions is shown in Figure 3-2.

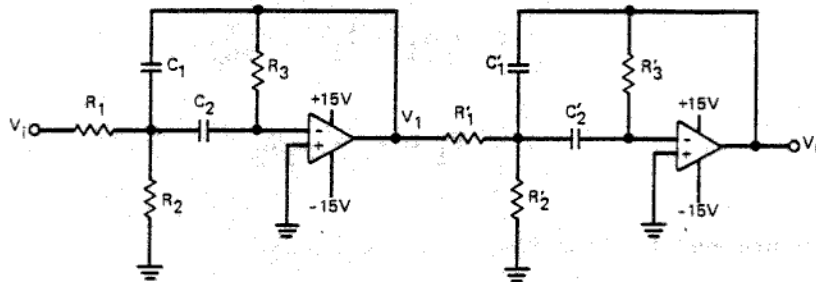


Figure 3-1. Bandpass Filter Schematic

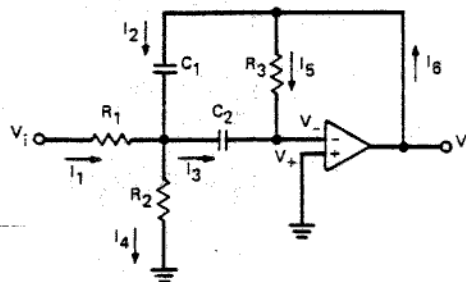


Figure 3-2. First Stage Schematic

3.1.1 Transfer Function

Using Figure 3-2, we get the following equations:

$$I_1 + I_2 - I_3 - I_4 = 0 \quad (3.1-1)$$

$$V_+ = V_- \quad (3.1-2)$$

$$V_+ = 0 \quad (3.1-3)$$

$$I_3 + I_5 = 0 \quad (3.1-4)$$

$$V_i = I_1 R_1 + I_4 R_2 \quad (3.1-5)$$

$$V_1 = \frac{I_2}{sC_1} + I_4 R_2 \quad (3.1-6)$$

$$V_- = V_i - I_1 R_1 - \frac{I_3}{sC_2} \quad (3.1-7)$$

$$V_1 = V_- + I_5 R_3 \quad (3.1-8)$$

We will now proceed to find the ratio of V_o/V_i .

First, we use equations 3.1-4, 3.1-3, and 3.1-8:

$$I_3 = -I_5 \quad (3.1-9)$$

$$V_1 = V_- + I_5 R_3$$

$$I_5 = V_1/R_3 \quad (3.1-10)$$

$$I_3 = -V_1/R_3 \quad (3.1-11)$$

Next, we combine equations 3.1-7, 3.1-11, and 3.1-3.

$$V_- = V_i - I_1 R_1 - \frac{I_3}{sC_2}$$

$$0 = V_i - I_1 R_1 - \frac{1}{sC_2} \left(\frac{-V_1}{R_3} \right)$$

$$I_1 R_1 = V_i + \frac{V_1}{sC_2 R_3}$$

$$I_1 = \frac{V_i}{R_1} + \frac{V_1}{sC_2 R_1 R_3} \quad (3.1-12)$$

Making use of equations 3.1-12 and 3.1-5:

$$V_i = I_1 R_1 + I_4 R_2$$

$$I_4 R_2 = V_i - R_1 \left(\frac{V_i}{R_1} + \frac{V_1}{sC_2 R_1 R_3} \right)$$

$$= V_i - V_i - \frac{V_1}{sC_2 R_3}$$

$$I_4 = \frac{-V_1}{sC_2 R_2 R_3} \quad (3.1-13)$$

Using equations 3.1-6 and 3.1-13 we get:

$$V_1 = \frac{I_2}{sC_1} + I_4 R_2$$

$$\frac{I_2}{sC_1} = V_1 - R_2 \left(\frac{-V_1}{sC_2 R_2 R_3} \right)$$

$$I_2 = V_1 sC_1 + \frac{V_1 C_1}{C_2 R_3} \quad (3.1-14)$$

Combining equations 3.1-1 and 3.1-11 through 3.1-14, we get:

$$0 = I_1 + I_2 - I_3 - I_4$$

$$= \frac{V_i}{R_1} + \frac{V_1}{sC_2 R_1 R_3} + V_1 sC_1 + \frac{V_1 C_1}{C_2 R_3} + \frac{V_1}{R_3} + \frac{V_1}{sC_2 R_2 R_3}$$

$$\frac{-V_i}{R_1} = V_1 \left(\frac{1}{sC_2 R_1 R_3} + sC_1 + \frac{C_1}{C_2 R_3} + \frac{1}{R_3} + \frac{1}{sC_2 R_2 R_3} \right)$$

$$= \frac{V_1}{sC_2 R_1 R_2 R_3} (R_2 + s^2 C_1 C_2 R_1 R_2 R_3 + sC_1 R_1 R_2 + sC_2 R_1 R_2 + R_1)$$

$$\frac{V_1}{V_i} = \frac{-sC_2 R_2 R_3}{s^2 (C_1 C_2 R_1 R_2 R_3) + s(C_1 R_1 R_2 + C_2 R_1 R_2) + (R_1 + R_2)}$$

$$= \frac{-sC_2 R_2 R_3}{C_1 C_2 R_1 R_2 R_3 \left[s^2 + s \left(\frac{C_1 + C_2}{C_1 C_2 R_3} \right) + \frac{R_1 + R_2}{C_1 C_2 R_1 R_2 R_3} \right]}$$

$$\frac{V_1}{V_i} = \frac{-s}{C_1 R_1} \frac{1}{s^2 + s \left(\frac{C_1 + C_2}{C_1 C_2 R_3} \right) + \frac{1}{R_3 C_1 C_2} \left(\frac{1}{R_2} + \frac{1}{R_2} \right)} \quad (3.1-15)$$

The transfer function of the second stage, V_o/V_1 , is of the same form as V_1/V_i , just with different values for the components. So, we get the following transfer function:

$$H(s) = \frac{V_1}{V_i} \frac{V_o}{V_1} = \frac{V_o}{V_i}$$

$$H(s) = \frac{s^2 / C_1 C_1' R_1 R_1'}{\left[s^2 + s \left(\frac{C_1 + C_2}{C_1 C_2 R_3} \right) + \frac{1}{R_3 C_1 C_2} \left(\frac{1}{R_1} + \frac{1}{R_2} \right) \right] \left[s^2 + s \left(\frac{C_1' + C_2'}{C_1' C_2' R_3'} \right) + \frac{1}{R_3' C_1' C_2'} \left(\frac{1}{R_1'} + \frac{1}{R_2'} \right) \right]}$$

(3.1-16)

3.1.2 Noise Bandwidth

The values of the components used in the BPF are given below. Notice that the two stages are identical:

$$C_1 = 1200 \text{ pF} \quad (3.1-17)$$

$$C_1' = 1200 \text{ pF} \quad (3.1-18)$$

$$C_2 = 1200 \text{ pF} \quad (3.1-19)$$

$$C_2' = 1200 \text{ pF} \quad (3.1-20)$$

$$R_1 = 13.3 \text{ k} \quad (3.1-21)$$

$$R_1' = 13.3 \text{ k} \quad (3.1-22)$$

$$R_2 = 1.47 \text{ k} \quad (3.1-23)$$

$$R_2' = 1.47 \text{ k} \quad (3.1-24)$$

$$R_3 = 52.3 \text{ k} \quad (3.1-25)$$

$$R_3' = 52.3 \text{ k} \quad (3.1-26)$$

Thus, let us make the following definition:

$$G = \frac{1}{C_1 R_1} = \frac{1}{C_1' R_1'} \quad (3.1-27)$$

$$B_1 = \frac{C_1 + C_2}{R_3 C_1 C_2} = \frac{C_1' + C_2'}{R_3' C_1' C_2'} \quad (3.1-28)$$

$$W_1 = \frac{1}{R_3 C_1 C_2} \left(\frac{1}{R_1} + \frac{1}{R_2} \right) = \frac{1}{R_3' C_1' C_2'} \left(\frac{1}{R_1'} + \frac{1}{R_2'} \right) \quad (3.1-29)$$

This allows us to state the following:

$$H(s) = \frac{G^2 s^2}{(s^2 + B_1 s + W_1)^2} \quad (3.1-30)$$

$$= \frac{G^2 s^2}{s^4 + s^3 (2B_1) + s^2 (W_1 + W_1 + B_1^2) + s(2B_1 W_1) + W_1^2}$$

$$H(s) = \frac{G^2 s^2}{s^4 + 2B_1 s^3 + (2W_1 + B_1^2) s^2 + 2B_1 W_1 s + W_1^2} \quad (3.1-31)$$

The one-sided noise bandwidth, B_N , of a filter is defined as:

$$B_N = \frac{1}{|H(\omega_0)|^2} \frac{1}{j2\pi} \int_0^{j\infty} |H(s)|^2 ds$$

$$B_N = \frac{1}{2 |H(\omega_0)|^2} \frac{1}{j2\pi} \int_{-j\infty}^{j\infty} |H(s)|^2 ds \quad (3.1-32)$$

$$B_N = \frac{1}{2 |H(\omega_0)|^2} I_4 \quad (3.1-33)$$

where

$$I_4 = \frac{1}{j2\pi} \int_{-j\infty}^{j\infty} |H(s)|^2 ds \quad (3.1-34)$$

From [3-1], we know I_4 , if $H(s)$ is of the form:

$$H(s) = \frac{e_3 s^3 + e_2 s^2 + e_1 s + e_0}{d_4 s^4 + d_3 s^3 + d_2 s^2 + d_1 s + d_0} \quad (3.1-35)$$

So, for our case we get

$$e_3 = 0 \quad (3.1-36)$$

$$e_2 = G^2 \quad (3.1-37)$$

$$e_1 = 0 \quad (3.1-38)$$

$$e_0 = 0 \quad (3.1-39)$$

$$d_4 = 1 \quad (3.1-40)$$

$$d_3 = 2B_1 \quad (3.1-41)$$

$$d_2 = 2W_1 + B_1^2 \quad (3.1-42)$$

$$d_1 = 2B_1 W_1 \quad (3.1-43)$$

$$d_0 = W_1^2 \quad (3.1-44)$$

And from [3-1], I_4 reduces down to:

$$I_4 = \frac{e_2^2 d_o d_1 d_4}{2d_o d_4 (-d_o d_3^2 - d_1^2 d_4 + d_1 d_2 d_3)} \quad (3.1-45)$$

$$I_4 = \frac{e_2^2 d_1}{2 (-d_o d_3^2 - d_1^2 d_4 + d_1 d_2 d_3)} \quad (3.1-46)$$

Thus, using equations 3.1-36 to 3.1-44 with equation 3.1-46, we get:

$$I_4 = \frac{G^4 (2B_1 W_1)}{2 (-W_1^2 (2B_1)^2 - (2B_1 W_1)^2 + 2B_1 W_1 (2W_1 + B_1^2)(2B_1))} \quad (3.1-47)$$

$$= \frac{B_1 W_1 G^4}{[-4B_1^2 W_1^2 - 4B_1^2 W_1^2 + 4B_1^2 W_1 (2W_1 + B_1^2)]}$$

$$= \frac{B_1 W_1 G^4}{4B_1^4 W_1}$$

$$I_4 = \frac{G^4}{4B_1^3} \quad (3.1-48)$$

Now, the BPF is designed so that the center frequency is ω_o . Thus, we have the relation:

$$W_1 = \omega_o^2 \quad (3.1-49)$$

So,

$$|H(\omega_o)|^2 = H(j\omega_o) H(-j\omega_o)$$

$$= \frac{(-G^2 \omega_o^2) (-G^2 \omega_o^2)}{(\omega_o^4 - j\omega_o^3 (2B_1) - \omega_o^2 (2\omega_o^2 + B_1^2) + j\omega_o (2B_1 \omega_o^2) + \omega_o^4)}$$

$$\times \frac{(\omega_o^4 + j\omega_o^3 (2B_1) - \omega_o^2 (2\omega_o^2 + B_1^2) - j\omega_o (2B_1 \omega_o^2) + \omega_o^4)}{}$$

$$= \frac{G^4 \omega_o^4}{\left[(\omega_o^4 - 2\omega_o^4 - \omega_o^2 B_1^2 + \omega_o^4) + j (-2B_1 \omega_o^3 + 2B_1 \omega_o^3) \right]}$$

$$\times \frac{1}{\left[(\omega_o^4 - 2\omega_o^4 - B_1^2 \omega_o^2 + \omega_o^4) + j (2B_1 \omega_o^3 - 2B_1 \omega_o^3) \right]}$$

$$= \frac{G^4 \omega_o^4}{(-\omega_o^2 B_1^2) (-\omega_o^2 B_1^2)}$$

$$|H(\omega_o)|^2 = \frac{G^4}{B_1^4}$$

(3.1-50)

Thus, we then get B_N :

$$\begin{aligned}
 B_N &= \frac{1}{2} \frac{I_4}{|H(\omega_0)|^2} \\
 &= \frac{1}{2} \frac{G^4}{4B_1^3} \frac{B_1^4}{G^4} \\
 B_N &= \frac{B_1}{8} \tag{3.1-51}
 \end{aligned}$$

Using equation 3.1-28 and the component values, we get the value of B_N :

$$\begin{aligned}
 B_N &= \frac{1}{8} \frac{C_1 + C_2}{R_3 C_1 C_2} \\
 &= \frac{1}{8} \frac{(1200 \times 10^{-12} + 1200 \times 10^{-12})}{(52.3 \times 10^3) (1200 \times 10^{-12})^2} \\
 B_N &= 3.983 \text{ KHz} \tag{3.1-52}
 \end{aligned}$$

The actual measured B_N of the filter used for CDU breadboard testing is 3.907 KHz.

3.1.3 3 dB Bandwidth

The other useful characteristic of the BPF is its 3 dB bandwidth. From equations 3.1-30 and 3.1-49 we know that

$$H(s) = \frac{G^2 s^2}{(s^2 + B_1 s + \omega_0^2)^2} \tag{3.1-53}$$

and

$$|H(\omega)|^2 = \frac{G^4 \omega^4}{\left[(\omega_o^2 - \omega^2)^2 + B_1 \omega^2 \right]^2} \quad (3.1-54)$$

The 3 dB frequencies are those such that

$$\frac{|H(\omega_3)|^2}{|H(\omega_o)|^2} = \frac{1}{2} \quad (3.1-55)$$

From equation 3.1-50, $|H(\omega_o)|^2$ is known; this gives us the following:

$$\frac{1}{2} = \frac{B_1^4 \omega_3^4}{\left[(\omega_o^2 - \omega_3^2)^2 + B_1^2 \omega_3^2 \right]^2} \quad (3.1-56)$$

We now solve for ω_3 :

$$\left[(\omega_o^2 - \omega_3^2)^2 + B_1^2 \omega_3^2 \right]^2 = 2 B_1^4 \omega_3^4$$

$$(\omega_o^2 - \omega_3^2)^2 + B_1^2 \omega_3^2 = \sqrt{2} B_1^2 \omega_3^2$$

$$(\omega_o^2 - \omega_3^2)^2 = (\sqrt{2} - 1) B_1^2 \omega_3^2$$

$$(\omega_o^2 - \omega_3^2) = \pm \sqrt{(\sqrt{2} - 1)} B_1 \omega_3$$

$$0 = \omega_3^2 \pm (\sqrt{2} - 1) B_1 \omega_3 - \omega_o^2 \quad (3.1-57)$$

We can now use the quadratic formula to find the 3 dB frequencies:

$$\omega_3 = \frac{\mp \sqrt{(\sqrt{2} - 1) B_1} \pm \sqrt{(\sqrt{2} - 1) B_1^2 + 4\omega_0^2}}{2} \quad (3.1-58)$$

Substituting in the values of B_1 and ω_0 we get:

$$\omega_3 = 111306.53 \text{ and } 90796.836 \text{ radians}$$

$$f_3 = \frac{\omega_3}{2\pi} = 17.715 \text{ KHz and } 14.451 \text{ KHz} \quad (3.1-59)$$

This gives us a 3.264 KHz 3 dB bandwidth. The actual measured 3 dB bandwidth of the filter used for CDU breadboard testing is 3.20 KHz.

3.1.4 Filter Results

A plot of the magnitude squared of the BPF is given in Figure 3-3; a plot of the phase response is given in Figure 3-4.

3.2 THE AGC AMPLIFIER

The AGC amplifier consists of an R-2R ladder network, connected to a current to voltage converter amplifier (I/V amp) to form a digital-to-analog converter (DAC), and a dual gain amplifier stage to provide the needed gain as a function of the lock status. The simplified circuit is shown in Figures 3-5 and 3-6.

3.2.1 The R-2R Ladder Network

The AGC loop outputs a digital control word, which controls the switches in the resistance ladder. These switches control the amount of current that goes to the I/V amp, thus controlling the gain of the first stage.

Since the input to the I/V amp is at virtual ground, no matter which position the switches are in, the input impedance seen by the input signal to the CDU is constant. Using simple circuit theory we see that Z_{in} , the CDU input impedance, is:

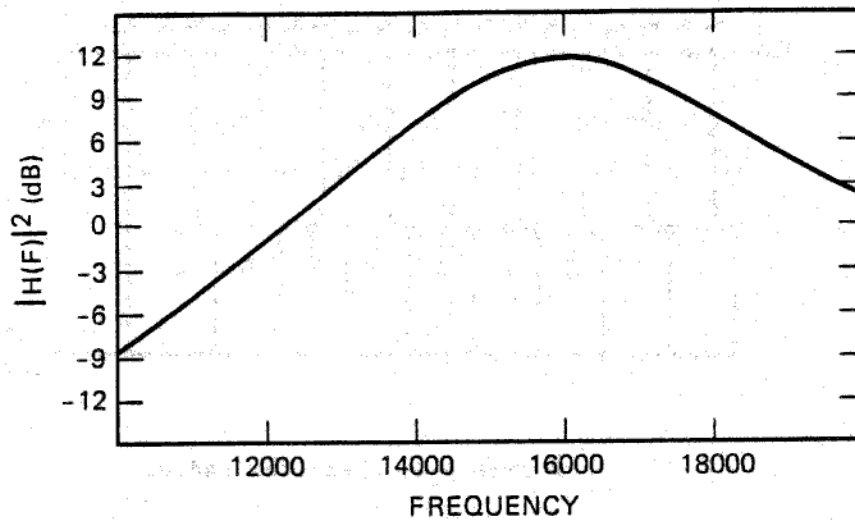


Figure 3-3. BPF Magnitude

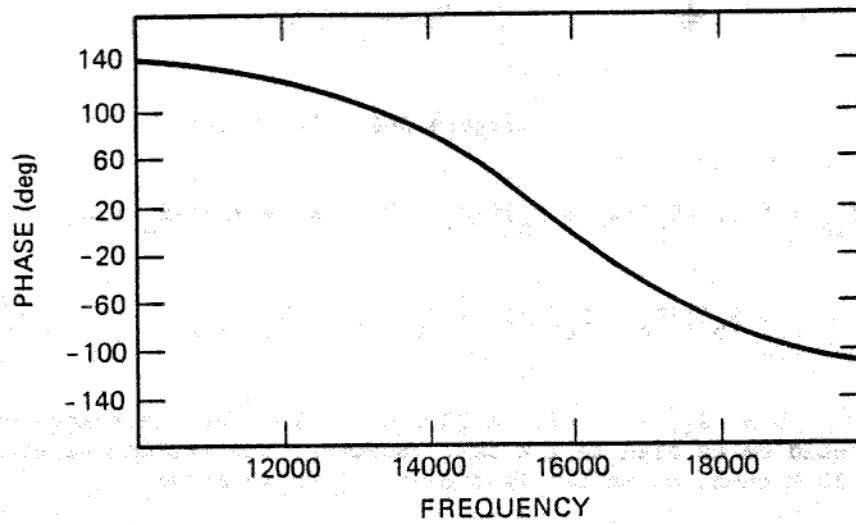


Figure 3-4. BPF Phase

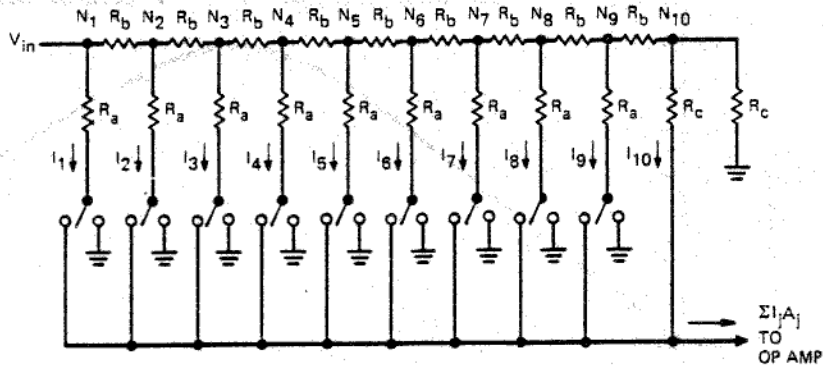


Figure 3-5. Resistor Ladder

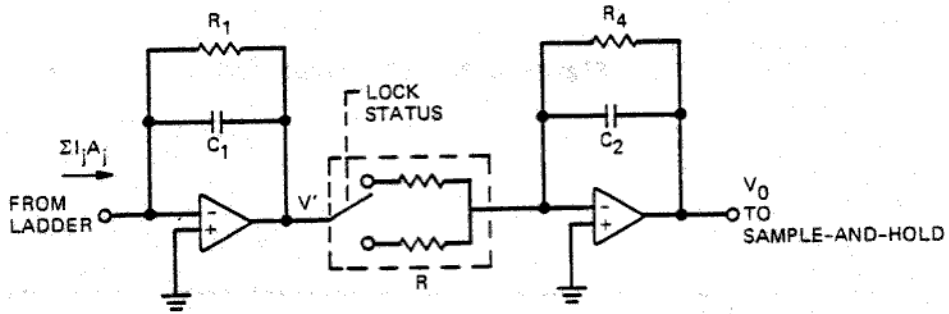


Figure 3-6. Amplifier

$$Z_{in} = (\dots (R_c || R_c + R_b) || R_a + R_b) || R_a + R_b || R_a + R_b || R_a + R_b || R_a + R_b || R_a + R_b || R_a + R_b || R_a + R_b || R_a \quad (3.2-1)$$

Now, R_a , is equal to 20 k ohms plus the resistance of the switch, which could be as high as 1 k ohm. However, the switch resistance is just 5% of the 20 k ohms, so we can ignore it. Thus we have:

$$R_a = 20 \text{ k} \quad (3.2-2)$$

$$R_b = 10 \text{ k} \quad (3.2-3)$$

$$R_c = 20 \text{ k} \quad (3.2-4)$$

Substituting in these values, we quickly see that:

$$Z_{in} = 10 \text{ k} \quad (3.2-5)$$

Also notice that at any node along the ladder where there is a switch on the resistor that the impedance looking into the op amp is 10 k, which is equal to R_b .

Moving on, we wish to calculate the currents I_1 through I_{10} . Since the impedance at any node, N_1 through N_{10} , is R_b , we see that the voltage at node j , V_j , is just:

$$V_j = \frac{R_b}{R_b + R_b} V_{j-1}$$
$$V_j = \frac{V_{j-1}}{2} \quad (3.2-6)$$

This is for j equal to 2 through 10. For j equal to 1 we have:

$$V_1 = V_{in} \quad (3.2-7)$$

Thus, we can see that:

$$V_j = \frac{V_{in}}{2^{j-1}}, \quad j = 1 \text{ through } 10 \quad (3.2-8)$$

Then, the current through any branch is

$$I_j = \frac{V_{in}}{2^{j-1} R_a} \cdot \frac{1}{R_a}$$

$$= \frac{10^{-4}}{2^j} V_{in}$$

$$I_j = \frac{10^{-4}}{1024} 2^{10-j} V_{in} \quad (3.2-9)$$

3.2.2 The DAC

We now connect the output of the ladder to the op amp. Given the fact that the $j = 10$ branch is always connected (no switch), we see that:

$$0 = \frac{V'}{Z_f} + V_{in} \left(\frac{10^{-4}}{1024} + \sum_{j=1}^9 \frac{10^{-4}}{1024} A_j 2^{10-j} \right)$$

$$\frac{V'}{Z_f} = -\frac{10^{-4}}{1024} V_{in} \left(1 + \sum_{j=1}^9 A_j 2^{10-j} \right)$$

$$\frac{V'}{V_{in}} = -Z_f \frac{10^{-4}}{1024} \left(1 + \sum_{j=1}^9 A_j 2^{10-j} \right) \quad (3.2-10)$$

where Z_f is the feedback impedance of the op amp, V' is the output voltage of the op amp, and A_j is 1 if switch j is on and 0 if switch j is off.

We see that Z_f is:

$$Z_f = \frac{\frac{1}{C_1}}{s + \frac{1}{C_1 R_1}} \quad (3.2-11)$$

Given the following:

$$R_1 = 196 \text{ k} \quad (3.2-12)$$

$$C_1 = 100 \text{ pF} \quad (3.2-13)$$

Thus:

$$Z_f = \frac{10^{10}}{s + (1.95 \times 10^{-5})^{-1}} \quad (3.2-14)$$

We are interested in frequencies around 16 kHz. So:

$$|Z_f| = \frac{10^{10}}{\left((2 \times 3.14159 \times 16 \times 10^3)^2 + (1.96 \times 10^{-5})^{-2} \right)^{1/2}}$$

$$|Z_f| = 88.7 \text{ k} \quad (3.2-15)$$

Thus:

$$\frac{V'}{V_{in}} = - \frac{8.87}{1024} \left(1 + \sum_{j=1}^9 A_j 2^{10-j} \right) \quad (3.2-16)$$

3.2.3 The Dual Gain Amplifier

The input resistor to the second stage amplifier is controlled by the lock status of the CDU. If the CDU is out-of-lock, the value is 3.09 k; if the status is in-lock, the value is 12.1 k. Since we are dealing with smaller resistors, we must include the resistance of the switch; thus, the values are 4.09 k and 13.1 k, respectively.

Designating the input resistance as R, we see that the gain of the amplifier is:

$$\frac{V_o}{V'} = \frac{\frac{1}{C_2}}{s + \frac{1}{C_2 R_4}} \frac{1}{R} \quad (3.2-17)$$

The values for R_4 and C_2 are:

$$R_4 = 69.1 \text{ k} \quad (3.2-18)$$

$$C_2 = 100 \text{ pF} \quad (3.2-19)$$

For the 16 kHz range of frequencies, we get the following:

$$\left| \frac{V_o}{V'} \right| = \frac{56.75 \text{ k}}{R} \quad (3.2-20)$$

Thus:

$$\left| \frac{V_o}{V_{in}} \right| = \frac{496.4}{R} \left(1 + \sum_{j=1}^9 A_j 2^{10-j} \right) \quad (3.2-21)$$

We will now examine the two cases.

3.2.3.1 Out-of-Lock

When the CDU is out-of-lock, all switches are on and R equals 4.09 k. Thus:

$$\left| \frac{V_o}{V_{in}} \right| = 124.17 \quad (3.2-22)$$

This gain is not realized over the entire signal range due to amplifier and ADC saturation. Measured values give the following results:

For 20 mV_{rms} input, gain is 129

For 50 mV_{rms} input, gain is 106.4

For 300 mV_{rms} input, gain is 19.2

3.2.3.2 In-Lock

When the CDU is in-lock, the value of R is 13.1 k. So:

$$\left| \frac{V_o}{V_{in}} \right| = 3.79 \times 10^{-2} \left(1 + \sum_{j=1}^9 A_j 2^{10-j} \right) \quad (3.2-23)$$

The gain then ranges from 3.79×10^{-2} to 38.76, a 30.1 dB voltage dynamic range.

3.3 REFERENCES

- 3-1 Newton, Gould, and Kaiser, Analytical Design of Linear Feedback Controls, Appendix E, Wiley, 1957.
- 3-2 Albrecht, V. R., Design Requirement NASA Deep Space Command Detector Unit, DM514438, Rev. A, August 1986.

SECTION 4

THE SUBCARRIER TRACKING LOOP

The CDU tracks the subcarrier by using coherent sampling to demodulate the subcarrier and a second order suppressed-carrier data-aided loop for tracking. The loop uses a data rate dependent scaling to maintain a constant phase jitter for all data rates.

This section provides the details of the design, analysis, and implementation of the subcarrier tracking.

4.1 SUBCARRIER LOOP MODEL

As will be described in subsection 4.2.1, the input signal to the subcarrier loop may be considered at baseband. This baseband signal is just the phase of the subcarrier at the instant of the error sample. Thus, the subcarrier loop block diagram is just a phase diagram and is shown in Figure 4-1.

The various blocks of Figure 4-1 are:

- (1) **Sample-and-Hold/ADC** - The sample-and-hold is triggered by the output of the digitally controlled oscillator (DCO). The subcarrier loop is trying to sample at the zero crossing of the input sinusoid. If the error in detecting the zero crossing is ϕ and the peak amplitude of the input is $A(r)$ (which is determined by the AGC loop), the data modulation is

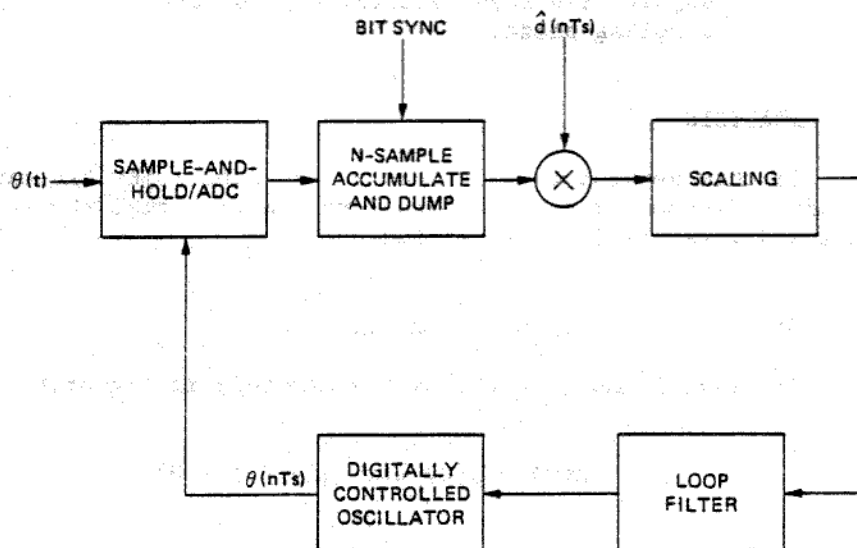


Figure 4-1. Subcarrier Loop Block Diagram

then the output of the sample-and-hold which is $A(r)d(t)\sin(\phi)$. The ADC (analog-to-digital converter) converts this output into an eight bit (seven bits magnitude, one bit sign) two's complement byte. The saturation voltage of the ADC is ± 5 volts; thus, the output of the ADC is:

$$127/5 A(r) d(t) \sin(\phi)$$

- (2) N sample Accumulate-and-Dump - This is also known as the error accumulator (EACC). N is the data rate dependent number of samples which is equal to the number of samples per bit; thus, the output of the EACC is the sum over one bit period.
- (3) $d(nT_s)$ - This is the data estimate for the bit time that was just accumulated in DACC. The data-aided loop uses the data estimate to remove the data modulation and to improve the tracking performance (hence the name "data-aided").
- (4) Scaling - To keep the loop dynamics the same for all data rates, the output of the error accumulator is scaled by a data rate dependent constant.
- (5) Loop filter - This consists of a two branch digital filter, which is described in Appendix 4B.
- (6) Digitally Controlled Oscillator (DCO) - This issues the zero crossing sample time command after counting a certain number of clock cycles based on the subcarrier frequency. When a correction count is received from the loop filter, the DCO adjusts its count accordingly to advance or retard the sampling phase.

4.2 ANALYSIS

In this section we will look at the subcarrier tracking loop in detail. After discussing coherent sampling, we will find the z-transform of the loop and derive the loop properties.

4.2.1 Coherent Sampling Demodulation

The signal input to the sample-and-hold is (ignoring the noise):

$$x(t) = A(r) d(t) \sin(\omega t + \theta) \quad (4.2-1)$$

The sampling period is equal to the nominal sampling period (the reciprocal of the sampling rate) minus the DCO adjustment. Thus:

$$T_k = T_s - a_{k-1} \quad (4.2-2)$$

where

T_k = kth sampling period

T_s = nominal sampling period

a_{k-1} = (k-1)th DCO adjustment

The sampling instant, t_k , is just the sum of the sampling periods, or:

$$t_k = \sum_{i=1}^k T_i \quad (4.2-3)$$

$$= \sum_{i=1}^k (T_s - a_{i-1})$$

$$t_k = kT_s - \sum_{i=0}^{k-1} a_i \quad (4.2-4)$$

So, at t_k ,

$$x(t_k) = A(r) d(t_k) \sin(\omega t_k + \theta_k)$$

$$x(t_k) = A(r) d(t_k) \sin\left(\omega k T_s + \theta_k - \sum_{i=0}^{k-1} \omega a_i\right) \quad (4.2-5)$$

Now, if we choose T_s such that

$$\omega T_s = 2n\pi \quad (4.2-6)$$

we then have:

$$x(t_k) = A(r) d(t_k) \sin\left(2nk\pi + \theta_k - \sum_{i=0}^{k-1} \omega a_i\right)$$

$$x(t_k) = A(r) d(t_k) \sin\left(\theta_k - \sum_{i=0}^{k-1} \omega a_i\right) \quad (4.2-7)$$

Thus, we just have the sine of the phase error at time t_k [4-4].

In other words:

$$\phi_k = \theta_k - \sum_{i=0}^{k-1} \omega a_i \quad (4.2-8)$$

This ϕ_k is just the zero crossing error we discussed in Section 4.1.

For the CDU, the subcarrier frequency is 16 kHz and the sampling rate is 8 kHz. Thus:

$$T_s = \frac{1}{8 \times 10^3} \text{ sec.} \quad (4.2-9)$$

$$\omega T_s = \frac{2\pi \times 16 \times 10^3}{8 \times 10^3}$$

$$\omega T_s = 4\pi \quad (4.2-10)$$

Since we are sampling in 4π increments, the subcarrier phase component drops out; thus our sampling scheme demodulates the subcarrier and provides a baseband signal.

Before we leave the demodulation section, one final question must be asked: "Is the sampling bandwidth wide enough?"

To avoid aliasing, the bandwidth of the data modulation should be less than or equal to one-half the sampling rate, or 4 kHz. The worst case condition occurs at the highest data rate, which is 500 bps. From [4-7] we know that the normalized spectrum of the baseband NRZ signal is

$$S(f) = \frac{\sin^2(\pi f T_b)}{(\pi f T_b)^2} \quad (4.2-11)$$

Thus, in the range of 4 kHz (say 3.9 kHz) and at 500 bps, we have

$$S(f) = \frac{\sin^2(\pi (3900)(1/500))}{(\pi (3900)(1/500))^2}$$

$$S(f) = 5.75 \times 10^{-4} \quad (4.2-12)$$

$$S(f) = -32.4 \text{ dB}$$

The signal is 32.4 dB below its peak, so, we can safely sample at the 8 kHz rate, since the signal is well contained in the 4 kHz bandwidth.

4.2.2 Derivation of the Closed Loop Transfer Function

To find the closed loop transfer function we must first generate the z-transform of our system. To do this, we must make two assumptions to facilitate the analysis.

4.2.2.1 Assumptions

The first assumption is that the error in finding the zero crossing, ϕ , is small enough that we may say:

$$\sin(\phi) \approx \phi \quad (4.2-13)$$

The second assumption is that the N sample accumulate-and-dump can be replaced by a gain of N , and a reduction in the system's clock rate of N (i.e., the system's clock rate was N times per bit; it is now once per bit). This assumption is discussed further in Appendix 4A.

4.2.2.2 Z-Transform

Making use of these assumptions, we get the following tracking loop block diagram.

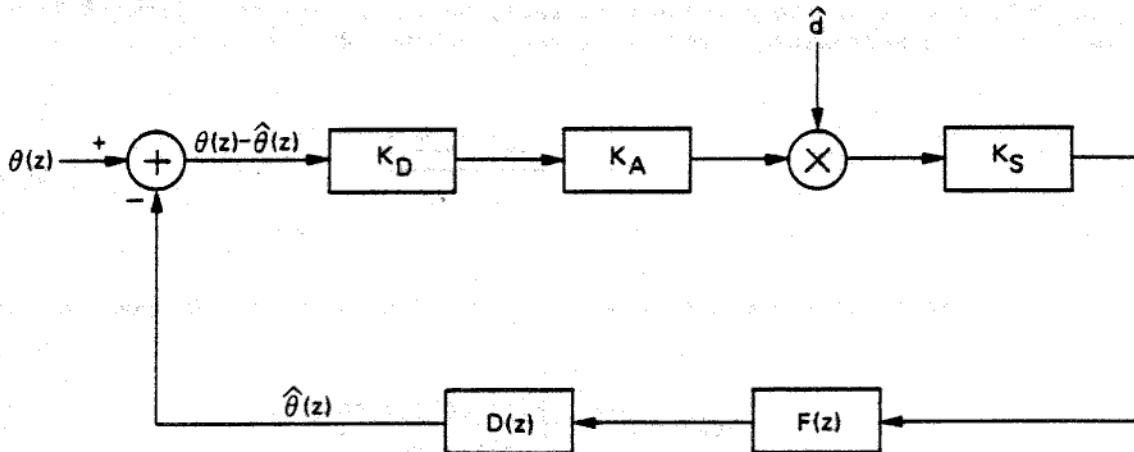


Figure 4-2. Simplified Block Diagram

where:

$$\begin{aligned} K_D &= \text{Detector gain (sample-and-hold/ADC)} \\ &= 127/5 A(r) d(t) \end{aligned} \quad (4.2-14)$$

$$\begin{aligned} K_A &= \text{number of samples per bit} \\ &= N(r) \end{aligned} \quad (4.2-15)$$

$$\begin{aligned} K_S &= \text{Loop scaling factor} \\ &= ESCL(r) \end{aligned} \quad (4.2-16)$$

$F(z)$ = Loop filter transfer function (derived in Appendix 4B)

$D(z)$ = DCO transfer function (derived in Appendix 4C)

$$D(z) = \frac{2\pi}{64} \frac{1}{z-1} \quad (4.2-17)$$

If we open the loop, we can find the open loop transfer function

$G(z)$:

$$\begin{aligned} G(z) &= \left. \frac{\hat{\theta}(z)}{\theta(z)} \right|_{\text{open}} \\ &= K_D K_A K_S \hat{d} F(z) D(z) \\ &= K_D K_A K_S \frac{2\pi}{64} \frac{F(z)}{z-1} \\ G(z) &= K \frac{F(z)}{z-1} \quad (4.2-18) \end{aligned}$$

where:

$$\begin{aligned} K &= K_D K_A K_S \hat{d} \frac{2\pi}{64} \\ &= \frac{127}{5} A(r) d(t) \hat{d} N(r) ESCL(r) \frac{2\pi}{64} \\ K &= 0.79375 \pi A(r) N(r) ESCL(r) d(t) \hat{d} \quad (4.2-19) \end{aligned}$$

The product $d(t)\hat{d}$ may be replaced by its statistical average [4-3].

Thus:

$$\begin{aligned} E\{d(t)\hat{d}\} &= (+1) P_r (d(t) = \hat{d}) + (-1) P_r (d(t) \neq \hat{d}) \\ &= (1 - P_e) - P_e \quad (4.2-20) \end{aligned}$$

$$E\{\hat{d}\} = 1 - 2 P_e \quad (4.2-21)$$

where

$$P_e = 0.5 \operatorname{erfc}(\sqrt{R_s}) \quad (4.2-22)$$

$$R_s = S T_b / N_o \quad (4.2-23)$$

and

$$\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} \exp(-t^2) dt \quad (4.2-24)$$

$$\operatorname{erf}(x) = 1 - \operatorname{erfc}(x) \quad (4.2-25)$$

Thus:

$$E\{\hat{d}\} = 1 - \operatorname{erfc}(\sqrt{R_s})$$

$$E\{\hat{d}\} = \operatorname{erf}(\sqrt{R_s}) \quad (4.2-26)$$

This gives us the following for K:

$$K = 0.79375 \pi A(r) N(r) \operatorname{ESCL}(r) \operatorname{erf}(\sqrt{R_s}) \quad (4.2-27)$$

The closed loop transfer function, $H(z)$, is:

$$H(z) = \frac{\hat{\theta}(z)}{\theta(z)}$$

$$= \frac{G(z)}{1 + G(z)}$$

$$= \frac{\frac{K F(z)}{(z-1)}}{1 + \frac{K F(z)}{(z-1)}}$$

$$H(z) = \frac{K F(z)}{(z-1) + K F(z)} \quad (4.2-28)$$

In Appendix 4B, we find

$$F(z) = \left(ECFO + \frac{ECF1}{z-1} \right) \quad (4B-2)$$

Substituting equation 4B-2 into 4.2-28, we get:

$$H(z) = \frac{K \left(\frac{ECFO (z-1) + ECF1}{z-1} \right)}{(z-1) + K \left(\frac{ECFO (z-1) + ECF1}{z-1} \right)}$$

$$= \frac{K (ECFO z + (ECF1 - ECFO))}{(z-1)^2 + K (ECFO z + (ECF1 - ECFO))}$$

$$H(z) = \frac{K (ECFO)z + K (ECF1 - ECFO)}{z^2 + z (K (ECFO) - 2) + (K (ECF1 - ECFO) + 1)} \quad (4.2-29)$$

From the closed loop transfer function we can derive the loop bandwidth.

4.2.3 Derivation of B_L

For a loop whose update time is T_b (the bit time), the one-sided loop bandwidth is:

$$B_L = \frac{W_L}{2}$$

$$B_L = \frac{1}{2} \frac{1}{T_b} \frac{1}{|H(1)|^2} \frac{1}{j2\pi} \oint_{|z|=1} H(z) H(z^{-1}) \frac{dz}{z} \quad (4.2-30)$$

Let:

$$I_2 = \frac{1}{j2\pi} \oint_{|z|=1} H(z) H(z^{-1}) \frac{dz}{z} \quad (4.2-31)$$

and with the fact:

$$|H(1)| = 1 \quad (4.2-32)$$

we get:

$$B_L = \frac{1}{2T_b} I_2 \quad (4.2-33)$$

I_2 is calculated in Appendix 4D. Using that result, we get:

$$B_L = \frac{1}{2T_b} \frac{2 \text{ECF1} + K (\text{ECF1})^2 - 3K (\text{ECFO})(\text{ECF1}) + 2K (\text{ECFO})^2}{(\text{ECF1} - \text{ECFO}) (2K (\text{ECFO}) - K (\text{ECF1}) - 4)} \quad (4.2-34)$$

4.2.4 Derivation of Loop Jitter

The loop signal-to-noise ratio may be defined as [4-3]:

$$\rho = \frac{R_s \operatorname{erf}^2(\sqrt{R_s})}{B_L T_b} \quad (4.2-35)$$

We also know [4-3] that for large ρ that the phase jitter, σ_ϕ , is:

$$\sigma_\phi^2 = \frac{1}{\rho} \quad (4.2-36)$$

Thus:

$$\sigma_\phi = \sqrt{\frac{B_L T_b}{R_s}} \frac{1}{\operatorname{erf}(\sqrt{R_s})} \quad (4.2-37)$$

where σ_ϕ is in radians.

4.2.5 Derivation of the Probability of Cycle Slip

The analysis in this section will make heavy use of the results and observations of [4-3] and [4-8].

For large ρ , [4-3] states that the average rate of cycle slips, \bar{S} , is:

$$\bar{S} = 2 B_L \sqrt{\frac{\rho\pi}{2}} \exp \left[- \left(2 - \frac{\pi^2}{8} \right) \rho \right] \quad (4.2-38)$$

Although this equation is for a first order data-aided loop, [4-3] gives experimental data that the actual rate is within a factor of ten of \bar{S} .

Thus, for this analysis, we will assume

$$\bar{S} < 20 B_L \sqrt{\frac{\rho\pi}{2}} \exp \left[- \left(2 - \frac{\pi^2}{8} \right) \rho \right] \quad (4.2-39)$$

From [4-8] we know that cycle slips are a Poisson process, with rate \bar{S} . From [4-9], we know that the probability of k occurrences in time t of a Poisson process with rate λ is:

$$\text{Prob}(k) = e^{-\lambda t} \frac{(\lambda t)^k}{k!} \quad (4.2-40)$$

If we let N_b be the number of bits in the period of time we are checking, the probability of a cycle slip is:

$$\rho(\text{cycle slip}) = 1 - \rho(\text{no cycle slip})$$

$$\rho(\text{cycle slip}) = 1 - \exp \left[- \bar{S} N_b T_b \right] \quad (4.2-41)$$

4.3 DERIVATION OF THE COEFFICIENTS

Now that all the analysis has been done, we are left with three coefficients to solve for: ESCL(r), ECF1, and ECFO. Their solutions follow.

4.3.1 ESCL(r)

For the subcarrier loop dynamics to remain constant, the product $B_L T_b$ must be independent of the data rate so:

$$B_L T_b = \frac{1}{2} \frac{2 ECF1 + K (ECF1)^2 - 3K (ECFO)(ECF1) + 2K (ECFO)^2}{(ECF1 - ECFO) (2K (ECFO) - K (ECF1) - 4)} \quad (4.3-1)$$

Since ECFO and ECF1 are constant for all data rates, for $B_L T_b$ to be constant, K must be constant for all data rates.

Recall equation 4.2-27:

$$K = 0.79375 \pi A(r) N(r) \text{ESCL}(r) \text{erf}(\sqrt{R_s}) \quad (4.2-27)$$

In Section 5, we will see that

$$A(r) N(r) = \frac{5}{127} A_d(r) N(r)$$

$$A(r) N(r) = \langle \text{DACC} \rangle \frac{5}{127} \quad (4.3-2)$$

where $\langle \text{DACC} \rangle$ is the expected value of the data accumulator.

Thus:

$$K = \frac{\pi}{32} \langle \text{DACC} \rangle \text{ESCL}(r) \quad (4.3-3)$$

(We assume $\text{erf}(\sqrt{R_g}) \approx 1$, which is true at threshold.)

K is a constant that determines $\text{ESCL}(r)$ and its choice is left to us, so let us say:

$$K = \frac{\pi}{2} \quad (4.3-4)$$

Then we get:

$$\text{ESCL}(r) = \frac{16}{\langle \text{DACC} \rangle} \quad (4.3-5)$$

Since $\langle \text{DACC} \rangle$ is calculated in Section 5, we now know $\text{ESCL}(r)$.

4.3.2 ECF1

As is discussed in Appendix B, the second order branch of the digital filter has a limit placed on it due to the requirements of [4-10]. This limit is that the second order branch may not contribute more than a one-sixteenth of a subcarrier bump per DCO bump.

The DCO issues the sample command in units of sixty-fourths of a subcarrier period. The limit of $1/16 T_{\text{sub}}$ (T_{sub} is the subcarrier period) means that the maximum correction provided by the second order branch is 4, since $1/16$ equals $4/64$. The worst case is when we are the limiting point and are making the maximum correction. Thus:

$$4 = 2^7 \text{ECF1} \quad (4.3-6)$$

$$ECF1 = 2^{-5} \quad (4.3-7)$$

4.3.3 ECFO

Since all other loop parameters have been solved for, ECFO will determine the final characteristics of subcarrier loop.

The variation of the subcarrier compared to the bit time is small. Thus we can say:

$$sT_b \ll 1 \quad (4.3-8)$$

The mapping between the discrete z-plane and the analog s-plane is

$$z = e^{sT_b} \quad (4.3-9)$$

Since equation 4.3-8 is true, we can state:

$$z \approx 1 + sT_b \quad (4.3-10)$$

Substituting equation 4.3-10 into the denominator of equation 4.2-29, we get:

$$\begin{aligned} H_D(s) &= (1 + sT_b)^2 + (1 + sT_b)(K(ECFO) - 2) + [K(ECF1 - ECFO) + 1] \\ &= 1 + 2T_b s + s^2 T_b^2 + K(ECFO) - 2 + sT_b K(ECFO) \\ &\quad - 2T_b s + K(ECF1) - K(ECFO) + 1 \\ &= s^2 T_b^2 + s(2T_b - 2T_b + T_b K(ECFO)) \\ &\quad + (1 + K(ECFO) - 2 + K(ECF1) - K(ECFO) + 1) \\ &= s^2 T_b^2 + sT_b K(ECFO) + K(ECF1) \\ H_D(s) &= T_b^2 \left[s^2 + \frac{K(ECFO)}{T_b} s + \frac{K(ECF1)}{T_b^2} \right] \quad (4.3-11) \end{aligned}$$

The standard definition of a denominator of the form of equation 4.3-11 is:

$$H_D(s) = s^2 + 2 \omega_n \zeta s + \omega_n^2 \quad (4.3-12)$$

Thus:

$$\omega_n^2 = \frac{K(ECF1)}{T_b^2} \quad (4.3-13)$$

$$2 \omega_n \zeta = \frac{K(ECFO)}{T_b} \quad (4.3-14)$$

$$\zeta = \frac{ECFO}{2} \sqrt{\frac{K}{ECF1}} \quad (4.3-15)$$

From [4-12] we have a target value for ζ of 0.76. Since we already know ECF1 and K, we can find ECFO:

$$\begin{aligned} ECFO &= 2 (0.76) \sqrt{\frac{1}{\frac{32}{\pi}} \frac{\pi}{2}} \\ &= \frac{1.52}{\sqrt{16\pi}} \end{aligned}$$

$$ECFO = 0.214 \quad (4.3-16)$$

Since we want to implement ECFO with bit shifts, we choose the closest power of 2 to this value. Thus:

$$ECFO = 2^{-2} \quad (4.3-17)$$

$$\zeta = 0.886 \quad (4.3-18)$$

$$\omega_n = \frac{0.222}{T_b}$$

(4.3-19)

Thus, we have all the parameters of the loop solved for.

4.4 RESULTS

The numeric results of our previous analyses follows.

4.4.1 Data Rate Dependent Results

With an assist from Section 5, the values for ESCL(r), B_L , and ω_n are given in Table 4-1.

Table 4-1. Values for ESCL, B_L , and ω_n

Data Rate	r	<DACC>	ESCL	B_L	ω_n
500	3	2^{10}	2^{-6}	78.3	111
250	4	2^{10}	2^{-6}	39.2	55.8
125	5	2^{11}	2^{-7}	19.6	27.8
62.5	6	2^{12}	2^{-8}	9.8	13.9
31.25	7	2^{12}	2^{-8}	4.9	6.95
15.625	8	2^{13}	2^{-9}	2.45	3.48
7.8125	9	2^{14}	2^{-9}	1.23	1.74

4.4.2 Phase Jitter

At threshold, R_s is 10.5 dB, or 11.22. Recalling equation 4.2-37, we get:

$$\sigma_\phi = \sqrt{\frac{B_L T_b}{R_s}} \frac{1}{\text{erf}(\sqrt{R_s})} \quad (4.2-37)$$

$$= \sqrt{\frac{0.157}{11.22}} \frac{1}{\text{erf}(3.35)}$$

$$= \frac{0.118}{0.99999} \text{ rad rms}$$

$$\sigma_\phi = 6.78 \text{ degrees rms} \quad (4.4-1)$$

4.4.3 Probability of Cycle Slip

For the threshold R_s of 10.5 dB, using equation 4.2-35, we get:

$$\rho = 71.8 \quad (4.4-2)$$

Given ρ , we now calculate the probability of a cycle slip in 128,000 bits at threshold from equations 4.2-39 and 4.2-41:

$$\text{Prob} = 1 - \exp \left[-20 \left(\frac{0.157}{T_b} \right) \sqrt{\frac{71.8\pi}{2}} \exp \left[- \left(2 - \frac{\pi^2}{8} \right) (71.8) \right] (128000) T_b \right]$$

$$\text{Prob} = 1 - 1 = 0$$

The probability of a cycle slip occurring at threshold is well below the required 1.0×10^{-5} [4-10].

At R_s equals 7 dB, we find:

$$\rho = 31.8 \quad (4.4-3)$$

$$\text{Prob} = 1 - \exp \left(-20 \left(\frac{0.157}{T_b} \right) \sqrt{\frac{31.8\pi}{2}} \exp \left[- \left(2 - \frac{\pi^2}{8} \right) (31.8) \right] (128000) T_b \right)$$

$$= 7.42 \times 10^{-5}$$

Thus, at 7 dB, the probability of a cycle slip is 7.42×10^{-5} .

Note that these results are based on an assumed formula for \bar{S} which is only known to be true for the first order DAL.

4.4.4 Doppler Rate Induced Phase Error

The digital filter that we are using is a perfect integrator. Thus, the input to the loop is

$$\theta(t) = \frac{1}{2} R t^2 + \omega_D t + \theta_0 \quad (4.4-4)$$

where R is the Doppler rate and ω_D is the Doppler offset. The steady state phase error is [4-13]:

$$\theta_{ss} = \frac{R}{\omega_n^2} \quad (4.4-5)$$

where ω_n^2 is defined by equation 4.3-19.

From [4-10] we can state that

$$R = (9.765 \times 10^{-3}) \frac{2 \pi \times 16 \times 10^3}{2^{2(r-3)}} \quad (4.4-6)$$

Using the definition of T_b :

$$T_b = \frac{2^{r-1}}{2000} \quad (4.4-7)$$

we get:

$$R = (9.765 \times 10^{-3}) \frac{\pi \times 32 \times 10^3}{(2000 T_b)^2 2^{-4}}$$

$$R = (1.2499 \times 10^{-3}) \frac{\pi}{T_b^2} \quad (4.4-8)$$

Recalling equation 4.3-19:

$$\omega_n = \frac{0.222}{T_b} \quad (4.3-19)$$

we can now get:

$$\theta_{ss} = \frac{(1.2499 \times 10^{-3}) \frac{\pi}{T_b^2}}{\left(\frac{0.222}{T_b}\right)^2}$$

$$\theta_{ss} = 0.0797 \text{ rad} \quad (4.4-9)$$

$$\theta_{ss} = 4.57 \text{ degrees} \quad (4.4-10)$$

Thus, in the presence of the maximum Doppler rate, the steady state phase error is only 4.57 degrees.

4.5 REFERENCES

- 4-1 Gill, Gurnam S., and Gupta, Someshwar C., "On Higher Order Discrete Phase-Locked Loops," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-8, No. 5, September 1972.

- 4-2 Weinberg, Aaron, and Liu, Bede, "Discrete Time Analysis of Nonuniform Sampling First- and Second-Order Digital Phase Locked Loops," IEEE Transactions On Communications, Vol. COM-22, No. 2, February 1974.
- 4-3 Simon, M., and Springett, J.C., The Theory, Design, and Operation of the Suppressed Carrier Data-aided Tracking Receiver, JPL Technical Report 32-1583, June 15, 1973.
- 4-4 Chie, Chak-Ming, Analysis of Digital Phased-Locked Loops, PhD Dissertation, University of Southern California, January 1977.
- 4-5 Jury, E.I., Theory and Application of the Z-Transform Method, John Wiley and Sons, Inc., New York, 1964.
- 4-6 Lindsey, W.C., and Simon, M.K., Telecommunications Systems Engineering, Prentice-Hall, Englewood Cliffs, New Jersey, 1973.
- 4-7 Yuen, Joseph H., Deep Space Telecommunications Systems Engineering, Chapter 2, NASA, July 1982.
- 4-8 Lindsey, W.C., Synchronization Systems in Communications and Control, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972.
- 4-9 Feller, William, An Introduction to Probability Theory and Its Applications, Volume I, Third Edition, John Wiley and Sons, New York, 1950.
- 4-10 Albrecht, V.R., Design Requirement NASA Deep Space Command Detector Unit, DM514438, Rev. A, August 1986.
- 4-11 NASA Standard Command Detector Unit Engineering Report, Motorola, February 1977.
- 4-12 Tausworthe, Robert C., Theory and Practical Design of Phase-Locked Receivers, Volume I, JPL Technical Report 32-819, February 15, 1966.
- 4-13 Viterbi, Andrew J., Principles of Coherent Communications, McGraw-Hill, New York, 1966.
- 4-14 Pierce, John R., and Posner, Edward C., Introduction to Communication Science and Systems, Plenum Press, New York, 1980.

4A

Z-TRANSFORM OF AN ACCUMULATE-AND-DUMP

The accumulate-and-dump can be represented by:

$$y(k) = \sum_{i=0}^{N-1} x(k-i) \quad (4A-1)$$

The z-transform is:

$$Y(z) = \sum_{i=0}^{N-1} X(z) z^{-i} \quad (4A-2)$$

$$= X(z) \sum_{i=0}^{N-1} (z^{-1})^i$$

$$Y(z) = X(z) \frac{z^{-N} - 1}{z^{-1} - 1} \quad (4A-3)$$

Thus, the transfer function, $A(z)$, is:

$$A(z) = \frac{z^{-N} - 1}{z^{-1} - 1} \quad (4A-4)$$

$A(z)$ is "observed" once every N samples.

Let's look at the frequency domain. Remembering that the sampling rate is T_b/N , where N is the number of samples per bit, we get the following substitution:

$$z = e^{j\omega \frac{T_b}{N}} \quad (4A-5)$$

we then find:

$$\begin{aligned}
 |A(\omega)| &= \left| \frac{e^{-j\omega N \frac{T_b}{N}} - 1}{e^{-j\omega \frac{T_b}{N}} - 1} \right| \\
 &= \left| \frac{e^{-j\omega \frac{T_b}{2}}}{e^{-j\omega \frac{T_b}{2N}} - e^{j\omega \frac{T_b}{2}}} \right| \left| \frac{e^{-j\omega \frac{T_b}{2}} - e^{j\omega \frac{T_b}{2}}}{e^{-j\omega \frac{T_b}{2N}} - e^{j\omega \frac{T_b}{2N}}} \right| \\
 |A(\omega)| &= \frac{\left| \sin \left(\omega \frac{T_b}{2} \right) \right|}{\left| \sin \left(\omega \frac{T_b}{2N} \right) \right|} \quad (4A-6)
 \end{aligned}$$

Now, let us find the transfer function of an integrate-and-dump filter. Notice:

$$\int_0^{T_b} = \int_{-\infty}^{T_b} - \int_{-\infty}^0 \quad (4A-7)$$

So, since we know the transfer function of an integrator integrating from negative infinity to a time t , [4-14], we have:

$$\begin{aligned}
 H(\omega) &= \frac{1}{T_b} \left[\frac{e^{-j\omega T_b}}{j\omega} - \frac{1}{j\omega} \right] \quad (4A-8) \\
 &= \frac{1}{T_b} e^{-j\frac{\omega}{2} T_b} \left(\frac{e^{-j\frac{\omega}{2} T_b} - e^{j\frac{\omega}{2} T_b}}{j\omega} \right)
 \end{aligned}$$

$$H(\omega) = e^{-j \frac{\omega}{2} T_b} \frac{\sin \frac{\omega T_b}{2}}{\frac{\omega T_b}{2}} \quad (4A-9)$$

The factor of $1/T_b$ is to normalize the output. Thus, the magnitude is:

$$|H(\omega)| = \left| \frac{\sin \frac{\omega T_b}{2}}{\frac{\omega T_b}{2}} \right| \quad (4A-10)$$

If we scale $|A(\omega)|$ by $1/N$ (to give it unity value at its maximum) and compare the frequency responses, we can see that the two transfer functions are almost identical for the range of values that we are interested in (see Figures 4A-1 and 4A-2). Thus, we can say that the accumulate-and-dump behaves like an integrate-and-dump, with a gain of N . And since $A(z)$ is only "observed" once every N samples, it reduces the system clock rate by a factor of $1/N$. Thus we can state:

The accumulate-and-dump can be replaced by a DC gain of N and sampling rate is reduced by a factor of N .

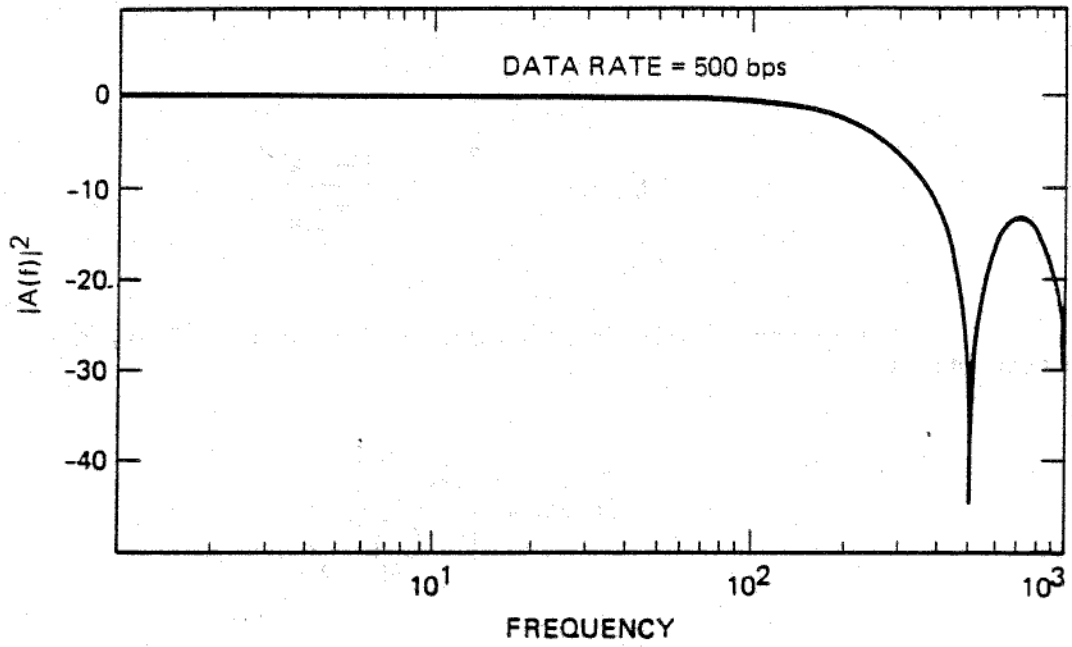


Figure 4A-1. Frequency Response - Accumulate-and-Dump

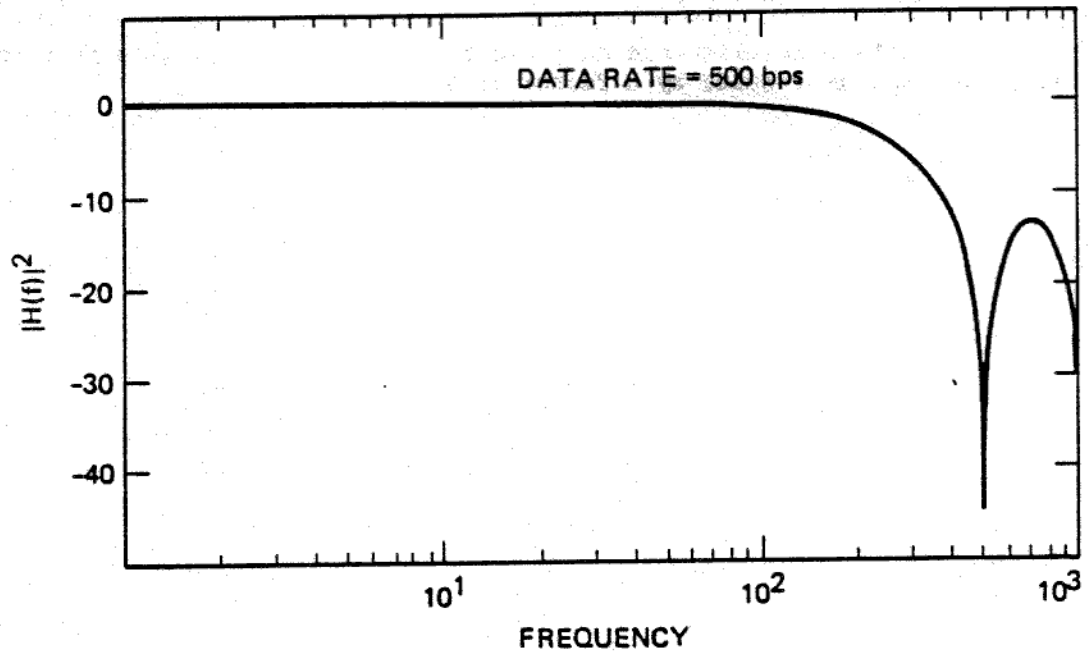


Figure 4A-2. Frequency Response - Integrate-and-Dump

Z-TRANSFORM OF THE DIGITAL FILTER

The block diagram of the digital filter is shown in Figure 4B-1.

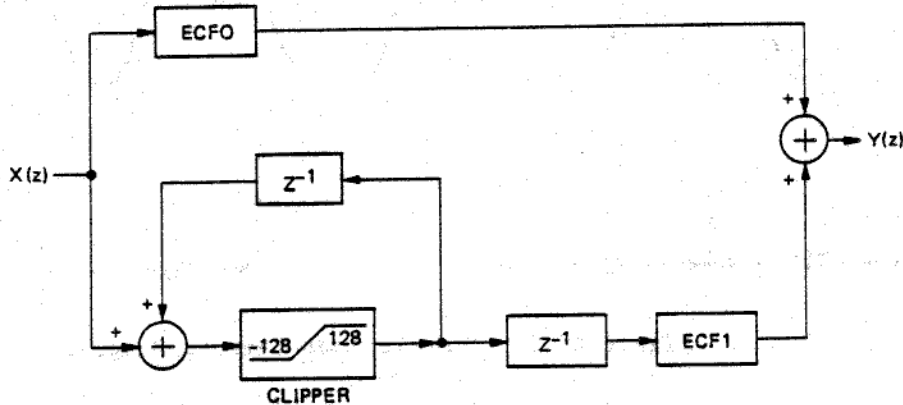


Figure 4B-1. Digital Filter Block Diagram

The largest bump that the DCO may make is one-eighth of a subcarrier period. Of this, a maximum of one-sixteenth may be caused by the second order branch. Thus one-half of the maximum correction may come from the second order branch. This translates into limiting the range of the integrator to the range -128 to 128. That is why the clipper is placed in the second order branch.

The z-transform of the filter is:

$$Y(z) = ECFO X(z) + z^{-1} ECF1 \frac{X(z)}{1 - z^{-1}} \quad (4B-1)$$

$$F(z) = \frac{Y(z)}{X(z)} \quad (4B-2)$$

$$F(z) = \left(ECFO + \frac{ECF1}{z - 1} \right)$$

4C. Z-TRANSFORM OF THE DIGITALLY CONTROLLED OSCILLATOR

To obtain the z-transform of the DCO, we must first discuss some Digital Phase Locked Loop (DPLL) theory, specifically nonuniform sampled DPLL's. Most of the following comes from references [4-1] and [4-2].

The DCO's job is to convert the output of the digital filter into a phase bump that is an integer number of 64ths of a subcarrier cycle. The DCO does this by adjusting the sampling time to sample at the zero crossing of the subcarrier. This adjustment leads to nonuniform time periods between samples. Observe Figure 4C-1:

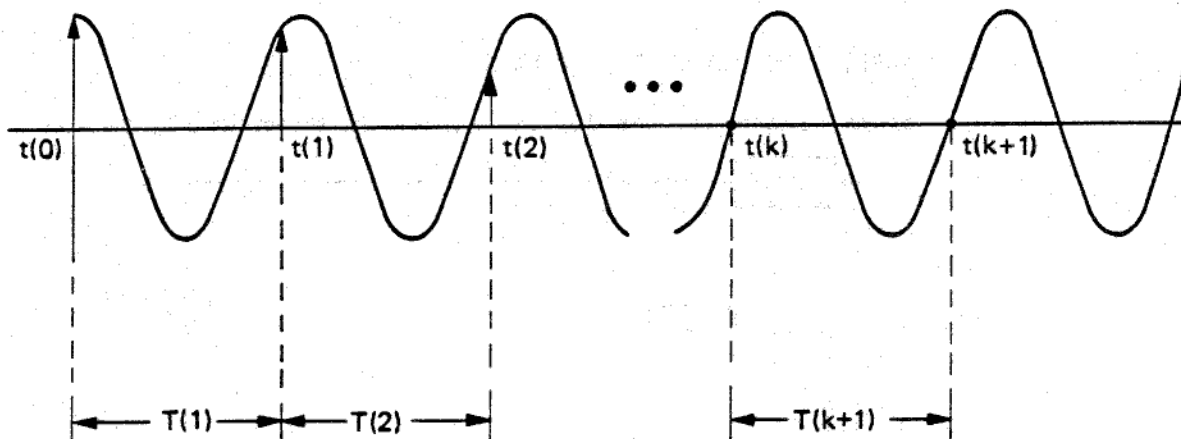


Figure 4C-1. Time Periods Between Samples

The $t(i)$ are the sampling times; $T(i)$ are the time periods between successive samples. Obviously, from Figure 4C-1:

$$T(j) = t(j) - t(j - 1) \quad (4C-1)$$

$$t(j) = \sum_{i=1}^j T(i) \quad (4C-2)$$

The DCO issues the sample command at time $t(j)$; thus, the phase of the output from the sampler is

$$\beta(j) = \omega_0 t(j) + \theta_0(t(j))$$

$$\beta(j) = \omega_0 t(j) + \theta_0(j) \quad (4C-3)$$

where ω_0 is the subcarrier frequency and $\theta_0(j)$ is the estimate of the modulated data.

The DCO's output pulse (sampling command) is a measure of its phase; the DCO issues the pulse once per cycle, or, once per 2π radians. The DCO's phase is also, by definition, the output phase of the sampler. So, after j pulses, we have:

$$\beta(j) = 2\pi j \quad (4C-4)$$

By definition, the natural period of the DCO, T , is:

$$T = \frac{2\pi}{\omega_0} \quad (4C-5)$$

The sampling period, $T(j)$, is equal to the natural period, T , minus correction term. This correction term is the previous output of the loop filter, $f(j-1)$. So:

$$T(j) = T - f(j-1) \quad (4C-6)$$

Rewriting equation 4C-3:

$$\theta_0(j) = \beta(j) - \omega_0 t(j) \quad (4C-7)$$

Combining equations 4C-3 through 4C-6:

$$\begin{aligned} \theta_0(j) &= 2\pi j - \omega_0 \sum_{i=0}^j T(i) \\ &= 2\pi j - \omega_0 \sum_{i=1}^j (T - f(i-1)) \\ &= 2\pi j - \omega_0 Tj + \omega_0 \sum_{i=0}^j f(i-1) \\ &= 2\pi j - 2\pi j + \omega_0 \sum_{i=1}^j f(i-1) \\ \theta_0(j) &= \omega_0 \sum_{i=1}^j f(i-1) \end{aligned} \quad (4C-8)$$

Then:

$$\begin{aligned}\theta_o(j+1) &= \omega_o \sum_{i=1}^{j+1} f(i-1) \\ &= \omega_o f(j) + \omega_o \sum_{i=1}^j f(i-1)\end{aligned}\tag{4C-9}$$

So:

$$\theta_o(j+1) - \theta_o(j) = \omega_o f(j)\tag{4C-10}$$

Taking the z-transform:

$$\begin{aligned}z \theta_o(z) - \theta_o(z) &= \omega_o F(z) \\ \theta_o(z) (z-1) &= \omega_o F(z) \\ \frac{\theta_o(z)}{F(z)} &= \frac{\omega_o}{z-1}\end{aligned}\tag{4C-11}$$

One step remains. The output of the DCO must be scaled so that it is in steps of one sixty-fourth of a cycle. So, we must scale $f(j)$ so it is in these units as well. This means we want

$$f'(j) = \frac{2\pi}{64} f(j)\tag{4C-12}$$

From equation 4C-5, we know

$$2\pi = \omega_o T$$

So:

$$f'(j) = \frac{\omega_o T}{64} f(j)\tag{4C-13}$$

Thus:

$$\theta_o(j+1) - \theta_o(j) = \frac{\omega_o T}{64} f(j) \quad (4C-14)$$

$$\theta_o(z) (z-1) = \frac{\omega_o T}{64} F(z) \quad (4C-15)$$

$$D(z) = \frac{\theta_o(z)}{F(z)} = \frac{\omega_o T}{64} \frac{1}{(z-1)} \quad (4C-16)$$

$$D(z) = \frac{2\pi}{64} \frac{1}{z-1} \quad (4C-17)$$

4.D CALCULATION OF I_2

From [4-5] we know that if we have

$$H(z) = \frac{b_o z^2 + b_1 z + b_2}{a_o z^2 + a_1 z + a_2} \quad (4D-1)$$

we can then find I_2 :

$$I_2 = \frac{1}{j2\pi} \oint_{|z|=1} H(z) H(z^{-1}) \frac{dz}{z} \quad (4D-2)$$

$$I_2 = \frac{B_o a_o e_1 - B_1 a_o a_1 + B_2 (a_1^2 - a_2 e_1)}{a_o [(a_o^2 - a_2^2) e_1 - a_1^2 (a_o - a_2)]} \quad (4D-3)$$

where

$$B_o = b_o^2 + b_1^2 + b_2^2 \quad (4D-4)$$

$$B_1 = 2 (b_0 b_1 + b_1 b_2) \quad (4D-5)$$

$$B_2 = 2 b_0 b_2 \quad (4D-6)$$

$$e_1 = a_0 + a_2 \quad (4D-7)$$

Equation 4.2-29 gives us $H(z)$ (for ease of notation, we replace ECF1 by E_1 , and ECFO by E_0):

$$H(z) = \frac{K E_0 z + K (E_1 - E_0)}{z^2 + z (K E_0 - 2) + [K (E_1 - E_0) + 1]} \quad (4D-8)$$

This gives us the following:

$$a_0 = 1 \quad (4D-9)$$

$$a_1 = K E_0 - 2 \quad (4D-10)$$

$$a_2 = K (E_1 - E_0) + 1 \quad (4D-11)$$

$$b_0 = 0 \quad (4D-12)$$

$$b_1 = K E_0 \quad (4D-13)$$

$$b_2 = K (E_1 - E_0) \quad (4D-14)$$

$$B_0 = K^2 E_0^2 + K^2 (E_1 - E_0)^2 \quad (4D-15)$$

$$B_1 = 2 K^2 E_0 (E_1 - E_0) \quad (4D-16)$$

$$B_2 = 0 \quad (4D-17)$$

$$e_1 = 2 + K (E_1 - E_0) \quad (4D-18)$$

Let's define N and D such that

$$I_2 = \frac{N}{D} \quad (4D-19)$$

Then:

$$N = B_0 a_0 e_1 - B_1 a_0 a_1 + B_2 (a_1^2 - a_2 e_1) \quad (4D-20)$$

$$D = a_0 [(a_0^2 - a_2^2) e_1 - a_1^2 (a_0 - a_2)] \quad (4D-21)$$

First, let's solve for N:

$$\begin{aligned} N &= [K^2 E_0^2 + K^2 (E_1 - E_0)^2] (1) [2 + K (E_1 - E_0)] \\ &\quad - 2 K^2 E_0 (E_1 - E_0) (K E_0 - 2) + 0 \\ &= K^2 [(2 E_0^2 - 2 E_0 E_1 + E_1^2) (2 + K (E_1 - E_0)) \\ &\quad - (E_1 - E_0) (2 K E_0^2 - 4 E_0)] \\ &= K^2 [4 E_0^2 - 4 E_0 E_1 + 2 E_1^2 + (E_1 - E_0) \\ &\quad \times (2 K E_0^2 - 2 K E_0 E_1 + K E_1^2 - 2 K E_0^2 + 4 E_0)] \end{aligned}$$

$$\begin{aligned}
N &= K^2 [4 E_0^2 - 4 E_0 E_1 + 2 E_1^2 + (E_1 - E_0)] \\
&\quad \times (-2 K E_0 E_1 + K E_1^2 + 4 E_0) \\
&= K^2 [4 E_0^2 - 4 E_0 E_1 + 2 E_1^2 - 2 K E_0 E_1^2 + K E_1^3 \\
&\quad + 4 E_0 E_1 + 2 K E_0^2 E_1 - K E_0 E_1^2 - 4 E_0^2] \\
&= K^2 [2 E_1^2 - 3 K E_0 E_1^2 + K E_1^3 + 2 K E_0^2 E_1] \\
N &= K^2 E_1 [2 E_1 - 3 K E_0 E_1 + K E_1^2 + 2 K E_0^2] \quad (4D-22)
\end{aligned}$$

Now, we solve for D:

$$\begin{aligned}
D &= (1) [1 - (K (E_1 - E_0) + 1)^2] (2 + K (E_1 - E_0)) \\
&\quad - (K E_0 - 2)^2 (1 - (K (E_1 - E_0) + 1)) \\
&= (2 + K (E_1 - E_0)) (1 - 1 - 2 K (E_1 - E_0) - K^2 (E_1 - E_0)^2) \\
&\quad - (K E_0 - 2)^2 (-K (E_1 - E_0)) \\
&= K (E_1 - E_0) [(2 + K (E_1 - E_0)) (-2 - K (E_1 - E_0)) \\
&\quad + (K E_0 - 2)^2] \\
&= K (E_1 - E_0) [-((K E_0 - 2) - K E_1)^2 + (K E_0 - 2)^2]
\end{aligned}$$

$$D = K (E_1 - E_0) [- (K E_0 - 2)^2 + 2 K E_1 (K E_0 - 2)$$

$$- K^2 E_1^2 + (K E_0 - 2)^2]$$

$$= K (E_1 - E_0) [2 K E_1 (K E_0 - 2) - K^2 E_1^2]$$

$$D = K^2 E_1 (E_1 - E_0) (2 K E_0 - K E_1 - 4) \quad (4D-23)$$

So, we get:

$$I_2 = \frac{K^2 E_1 [2 E_1 - 3 K E_0 E_1 + K E_1^2 + 2 K E_0^2]}{K^2 E_1 (E_1 - E_0) (2 K E_0 - K E_1 - 4)}$$

$$I_2 = \frac{2 E_1 - 3 K E_0 E_1 + K E_1^2 + 2 K E_0^2}{(E_1 - E_0) (2 K E_0 - K E_1 - 4)} \quad (4D-24)$$

or

$$I_2 = \frac{2 ECF1 - 3 K (ECFO)(ECF1) + K (ECF1)^2 + 2 K (ECFO)^2}{(ECF1 - ECFO) (2 K (ECFO) - K (ECF1) - 4)} \quad (4D-25)$$

SECTION 5

THE AUTOMATIC GAIN CONTROL LOOP

The automatic gain control (AGC) loop provides a constant signal level to the ADC. Without the AGC, variations in communication link parameters, such as signal strength (due mainly to modulation index changes and range changes) and receiver components (due mainly to aging and thermal variations), could cause the signal amplitude to vary greatly. The removal of these variations clearly requires a coherent (constant signal power) AGC with a dynamic range in excess of 40 dB.

This section describes the design, analysis, and implementation of the Deep Space CDU's coherent AGC loop. The loop dynamics are constant for all bit rates, due to data rate dependent scaling. Thus, performance is independent of the data rate.

5.1 AGC LOOP MODEL

The block diagram of the AGC is shown in Figure 5-1 and the simplified z-transform version block diagram is shown in Figure 5-2. A description of the loop components follows.

5.1.1 AGC Loop Components

The various blocks of Figure 5-1 are:

- (1) AGC Amplifier - This is the analog component discussed in Section 3.2. The amplifier's gain is determined by the output of the control function.
- (2) Sample-and-Hold/ADC - The AGC loop requires data samples to be taken one quarter of a subcarrier cycle after the error sample used by the subcarrier loop. Since the subcarrier tracks the zero crossing, the data sample is on the peak of the subcarrier. So, if the zero crossing error is ϕ and the peak amplitude of the subcarrier is A, then the output of the sample-and-hold is:

$$A \sin (\phi + 90^\circ) = A \cos \phi$$

The ADC (analog-to-digital converter) outputs an eight bit, two's complement digital byte (i.e., one sign bit, seven data bits). The saturation voltage of the ADC is ± 5 volts; the maximum digital output is 127. Thus, the output of the ADC is an integer equal to:

$$(127/5)(A \cos (\phi))$$

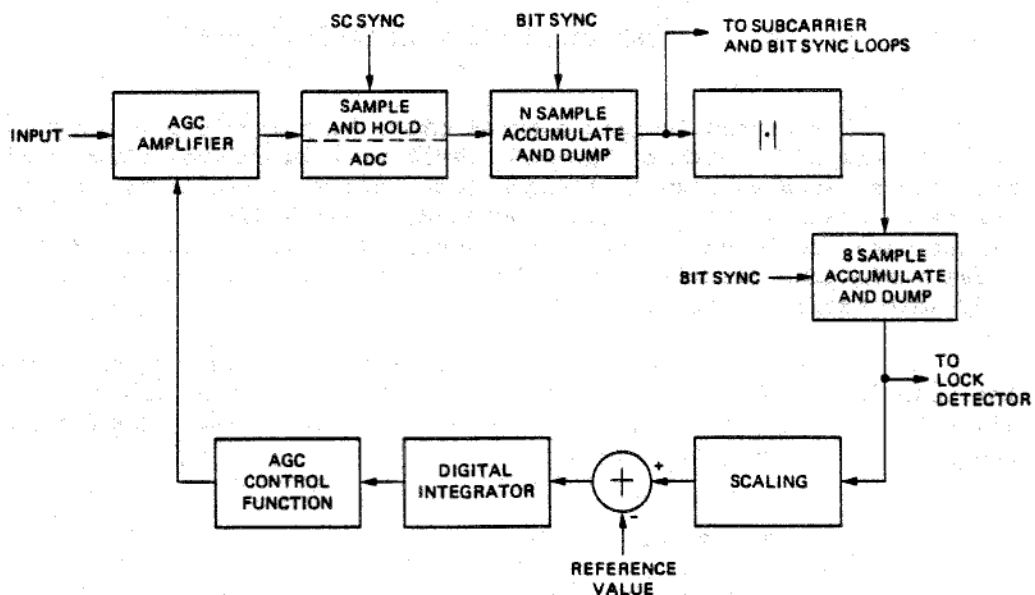


Figure 5-1. AGC Block Diagram

- (3) N sample accumulate-and-dump - This is also known as the data accumulator (DACC). N is equal to the number of samples per bit. Thus, the output is the sum over one-bit period.
- (4) Absolute value operator - This removes the modulation from the data accumulator.
- (5) Eight-sample accumulate-and-dump - This does an accumulation over eight bits to help "smooth" the AGC response.
- (6) Scaling - To maintain constant loop dynamics over the range of data rates, a data rate dependent scaling takes place. Once this scaling occurs, the loop is independent of the data rate.
- (7) Reference Value - The output of the scaling is compared to a reference value to create an error signal. The loop's function is to drive this error signal to zero.
- (8) Digital Integrator - This is the loop's filter. It provides a perfect integration of the error signal. A perfect integrator was chosen since it minimizes the mean-squared step error response.

- (9) AGC control function - This function feeds back the integrated error signal to the AGC amplifier. The exponential function was chosen because the loop gain of an exponential loop is constant, causing the small signal response to be independent of the input signal level [5-1].

5.1.2 Assumptions

Before we can move on to the simplified model, we must discuss two assumptions that are made to facilitate the analysis. The first assumption is used throughout this report, namely that an N-sample accumulate-and-dump operator is equivalent to a reduction in the clock rate by a factor of N and a signal gain of N. This result was discussed in Appendix 4A.

The second assumption concerns the limiting effects of the AGC control feedback leg. The limiting effects occur when the AGC amplifier is driven to saturation, an event that occurs when the CDU is out-of-lock or when the AGC is requiring maximum gain. The AGC is designed to not require the latter condition, so we can disregard it. There will be some limiting during acquisition, but the AGC will quickly move away from the maximum gain condition, so the effects of the limiting are minor. Thus, we shall neglect the saturation effects.

5.1.3 Simplified AGC Model

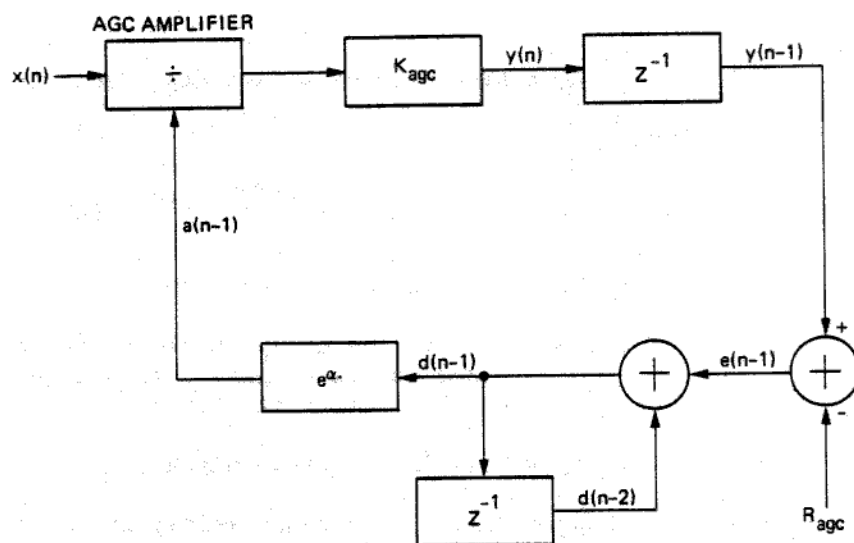


Figure 5-2. Simplified AGC Model

The various components are:

- (1) $x(n)$ - The sampled signal plus noise. For analysis sake, we assume the sampling has taken place outside the AGC loop, but not the analog to digital conversion.
- (2) AGC amplifier - This is modelled as an attenuator or divider.
- (3) K_{agc} - This is all the gains experienced in the loop. K_{agc} is equal to:

$$K_{agc} = K_{adc} \times K_N \times K_8 \times K_S$$

where

$$K_{adc} = \text{ADC gain}$$

$$= \frac{127}{5}$$

$$K_N = 2^{r+1}$$

$$K_8 = 8 \text{ sample accumulate-and-dump gain}$$

$$= 8$$

$$K_S = \text{ASCL, the data rate dependent scale factor}$$

Thus

$$K_{agc} = \left(\frac{127}{5}\right)(2^{r+4})(\text{ASCL})$$

- (4) $y(n)$ - This is the scaled output of the AGC accumulators.
- (5) z^{-1} - This is the delay caused by the accumulators. Since we are summing over 8 bits, the delay is equal to eight bit times.
- (6) $e(n)$ - This is the error signal.
- (7) R_{agc} - This is reference value; it is equal to 128.
- (8) $a(n)$ - This is the control word that goes to the AGC amplifier to control the gain.

5.2 ANALYSIS

Now that we have the model of the AGC, we can begin the analysis. We will begin with a large signal analysis, move on to a small signal analysis and finish up with a z-transform analysis (the final analysis depends on the small signal results).

5.2.1 Large Scale Analysis

We start by referring to Figure 5-2 to get the relationships between $x(n)$, $y(n)$, $e(n)$, $d(n)$, and $a(n)$. From Figure 5-2 we see that:

$$y(n) = K_{agc} \frac{x(n)}{a(n-1)} \quad (5.2-1)$$

$$e(n) = y(n) - R_{agc} \quad (5.2-2)$$

$$d(n) = d(n-1) + e(n) \quad (5.2-3)$$

$$a(n) = \exp [\alpha \cdot d(n)] \quad (5.2-4)$$

Our goal in this section is to find $y(n)$ in terms of $x(n)$, R_{agc} , and K_{agc} [5-1].

First, a few standard manipulations

$$d(n) = \frac{\ln[a(n)]}{\alpha} \quad (5.2-5)$$

$$d(n-1) = \frac{\ln[a(n-1)]}{\alpha} \quad (5.2-6)$$

$$a(n-1) = K_{agc} \frac{x(n)}{y(n)} \quad (5.2-7)$$

$$a(n) = K_{agc} \frac{x(n+1)}{y(n+1)} \quad (5.2-8)$$

Substituting 5.2-2 into 5.2-3:

$$d(n) = d(n-1) + y(n) - R_{agc} \quad (5.2-8A)$$

Combining 5.2-8 with 5.2-5, and 5.2-6 with 5.2-7, we get:

$$d(n) = \left(\frac{\ln \left[K_{agc} \frac{x(n+1)}{y(n+1)} \right]}{\alpha} \right) \quad (5.2-9)$$

$$d(n-1) = \left(\frac{\ln \left[K_{agc} \frac{x(n)}{y(n)} \right]}{\alpha} \right) \quad (5.2-10)$$

Combining the above two equations with 5.2-8A, we get:

$$\frac{1}{\alpha} \ln \left[K_{agc} \frac{x(n+1)}{y(n+1)} \right] = \frac{1}{\alpha} \ln \left[K_{agc} \frac{x(n)}{y(n)} \right] + y(n) - R_{agc}$$

$$-\ln [y(n+1)] = \ln \left[\frac{x(n)}{y(n)} \right] - \ln [x(n+1)] + \alpha(y(n) - R_{agc})$$

$$y(n+1) = \exp \left[\ln \left(\frac{x(n+1)}{x(n)} \right) + \ln(y(n)) - \alpha(y(n) - R_{agc}) \right] \quad (5.2-11)$$

After we derive α in Section 5.3.2, we will return to use equation 5.2-11 (in Section 5.4.2).

5.2.2 Small Signal Analysis

The purpose of this section is to develop a set of equations that describe the loop in the presence of a small perturbation [5-1, 5-2, 5-3]. These equations will be used in section 5.2.3 to derive the loop transfer function.

We begin by assuming that the loop is in the steady state:

$$x(n) = \bar{x} \quad (5.2-12)$$

$$y(n) = \bar{y} = R_{agc} \quad (5.2-13)$$

$$a(n) = \bar{a} \quad (5.2-14)$$

$$e(n) = \bar{y} - R_{agc} = 0 \quad (5.2-15)$$

$$d(n) = \bar{d} \quad (5.2-16)$$

We now perturb the system by $\tilde{x}(n)$ (and assume that the perturbations are small; we also only consider first order terms) and get:

$$x(n) = \bar{x} + \tilde{x}(n) \quad (5.2-17)$$

$$y(n) = \bar{y} + \tilde{y}(n) \quad (5.2-18)$$

$$a(n) = \bar{a} + \tilde{a}(n) \quad (5.2-19)$$

Recalling equation 5.2-1 and plugging in the above equations and keeping only the first order terms, we get:

$$y(n) = K_{agc} \frac{x(n)}{a(n-1)}$$

$$R_{agc} + \tilde{y}(n) = K_{agc} \frac{x(n)}{a(n-1)}$$

$$= K_{agc} \frac{\bar{x} + \tilde{x}(n)}{\bar{a} + \tilde{a}(n-1)}$$

$$= K_{agc} \frac{\bar{x}}{\bar{a}} \frac{1 + \frac{\tilde{x}(n)}{\bar{x}}}{1 + \frac{\tilde{a}(n-1)}{\bar{a}}}$$

$$\tilde{y}(n) = -R_{agc} + K_{agc} \left(\frac{\bar{x}}{\bar{a}} \right) \left[1 + \frac{\tilde{x}(n)}{\bar{x}} - \frac{\tilde{a}(n-1)}{\bar{a}} \right]$$

We know that

$$\bar{y} = K_{agc} \frac{\bar{x}}{a} = R_{agc}$$

which allows us to say:

$$\tilde{y}(n) = -\frac{\bar{x}}{a} K_{agc} + K_{agc} \left(\frac{\bar{x}}{a} \right) + K_{agc} \frac{\tilde{x}(n)}{a} - K_{agc} \frac{\bar{x}}{a^2} \tilde{a}(n-1)$$

$$= K_{agc} \frac{\tilde{x}(n)}{a} - K_{agc} \frac{\bar{x}}{a^2} \tilde{a}(n-1)$$

$$= K_{agc} \frac{\tilde{x}(n)}{\left(K_{agc} \frac{\bar{x}}{R_{agc}} \right)} - K_{agc} \tilde{a}(n-1) \frac{\bar{x}}{\left(K_{agc} \frac{\bar{x}}{R_{agc}} \right)^2}$$

$$\tilde{y}(n) = R_{agc} \frac{\tilde{x}(n)}{\bar{x}} - \frac{R_{agc}^2}{K_{agc}} \frac{\tilde{a}(n-1)}{\bar{x}} \quad (5.2-20)$$

For the error signal, using equation 5.2-15, we get:

$$e(n) = \bar{e} + \tilde{e}(n) = \bar{y} + \tilde{y}(n) - R_{agc}$$

$$\bar{e} + \tilde{e}(n) = \tilde{y}(n) + \bar{y} - R_{agc}$$

$$\tilde{e}(n) = \tilde{y}(n) + (\bar{y} - R_{agc} - \bar{e})$$

$$\tilde{e}(n) = \tilde{y}(n) \quad (5.2-21)$$

For the integrator output, using equations 5.2-15 and 5.2-3, we get:

$$d(n) = d(n-1) + e(n)$$

$$= \bar{d} + \tilde{d}(n)$$

$$\bar{d} + \tilde{d}(n) = \bar{d} + \tilde{d}(n-1) + \tilde{e}(n) + \bar{e}$$

$$\tilde{d}(n) = \tilde{d}(n-1) + \tilde{e}(n) \quad (5.2-22)$$

Finally, for the AGC control word, using equation 5.2-4, we get:

$$\bar{a} + \tilde{a}(n) = a(n)$$

$$= \exp(\alpha d(n))$$

$$= \exp[\alpha(\bar{d} + \tilde{d}(n))]$$

$$\tilde{a}(n) = a(n) - \bar{a}$$

$$= \exp[\alpha \bar{d} + \alpha \tilde{d}(n)] - \exp(\alpha \bar{d})$$

$$= \exp(\alpha \bar{d}) [-1 + \exp(\alpha \tilde{d}(n))]$$

Since $\tilde{d}(n)$ is small, we can use

$$\exp(a) \approx 1 + \frac{a}{1!} = 1 + a$$

which gives us:

$$\tilde{a}(n) \approx \exp(\alpha \bar{d}) (-1 + 1 + \alpha \tilde{d}(n))$$

$$\tilde{a}(n) = \alpha \tilde{d}(n) \exp(\alpha \bar{d}) \quad (5.2-23)$$

So, to sum up the results of this section:

$$\tilde{y}(n) = R_{\text{agc}} \frac{\tilde{x}(n)}{x} - \frac{R_{\text{agc}}^2}{K_{\text{agc}}} \frac{\tilde{a}(n-1)}{x} \quad (5.2-24)$$

$$\tilde{e}(n) = \tilde{y}(n) \quad (5.2-25)$$

$$\tilde{d}(n) = \tilde{d}(n-1) + \tilde{e}(n) \quad (5.2-26)$$

$$\tilde{a}(n) = \alpha \tilde{d}(n) \exp(\alpha \bar{d}) \quad (5.2-27)$$

5.2.3 Z-Transform AGC Analysis

Our goal in this section is to find the small signal z-transform transfer function and obtain the AGC loop's time constant from it.

We start by taking the z-transform of equations 5.2-24 through 5.2-27 and substituting among them to get a single equation in terms of Y(z) and X(z). So:

$$\tilde{Y}(z) = R_{\text{agc}} \frac{\tilde{X}(z)}{x} - \frac{R_{\text{agc}}^2}{K_{\text{agc}}} \frac{\tilde{A}(z)}{x} z^{-1} \quad (5.2-28)$$

$$\tilde{E}(z) = \tilde{Y}(z) \quad (5.2-29)$$

$$\tilde{D}(z) = z^{-1} \tilde{D}(z) + \tilde{E}(z) \quad (5.2-30)$$

$$\tilde{A}(z) = \alpha \tilde{D}(z) \exp(\alpha \bar{d}) \quad (5.2-31)$$

$$\tilde{D}(z) (1 - z^{-1}) = \tilde{E}(z)$$

$$= \tilde{Y}(z)$$

$$\tilde{D}(z) = \frac{\tilde{Y}(z)}{(1 - z^{-1})}$$

$$\tilde{A}(z) = \alpha \exp(\alpha \bar{d}) \left(\frac{\tilde{Y}(z)}{1 - z^{-1}} \right)$$

$$\exp(\alpha \bar{d}) = \bar{a}$$

$$= \frac{K_{agc}}{R_{agc}} \bar{x} \quad (\text{from 5.2-13 and 5.2-1})$$

$$\tilde{A}(z) = \alpha \left(\frac{K_{agc}}{R_{agc}} \right) \bar{x} \left(\frac{\tilde{Y}(z)}{1 - z^{-1}} \right)$$

$$\tilde{Y}(z) = R_{agc} \frac{\tilde{X}(z)}{\bar{x}} - \frac{R_{agc}^2}{K_{agc}} \frac{1}{\bar{x}} \alpha \left(\frac{K_{agc}}{R_{agc}} \right) \bar{x} z^{-1} \frac{\tilde{Y}(z)}{1 - z^{-1}}$$

$$= R_{agc} \frac{\tilde{X}(z)}{\bar{x}} - R_{agc} \alpha \frac{\tilde{Y}(z)}{1 - z^{-1}} z^{-1}$$

$$\tilde{Y}(z) \left(1 + \frac{\alpha R_{agc} z^{-1}}{1 - z^{-1}} \right) = R_{agc} \frac{\tilde{X}(z)}{\bar{x}}$$

$$\tilde{Y}(z) \left(\frac{1 - z^{-1} + \alpha R_{agc} z^{-1}}{1 - z^{-1}} \right) = R_{agc} \frac{\tilde{X}(z)}{\bar{x}}$$

$$\tilde{Y}(z) \left(\frac{z - (1 - \alpha R_{agc})}{z - 1} \right) = R_{agc} \frac{\tilde{X}(z)}{\bar{x}}$$

$$\frac{\tilde{Y}(z)}{\tilde{X}(z)} = \frac{R_{agc}}{\bar{x}} \frac{z - 1}{z - (1 - \alpha R_{agc})}$$

(5.2-32)

Since we are interested in the fractional change in the output due to the fractional change in the input, let's define $\tilde{H}(z)$ as:

$$\begin{aligned}\tilde{H}(z) &= \frac{\left(\frac{\tilde{Y}(z)}{\bar{y}}\right)}{\left(\frac{\tilde{X}(z)}{\bar{x}}\right)} \\ &= \left(\frac{\tilde{Y}(z)}{\tilde{X}(z)}\right) \left(\frac{\bar{x}}{\bar{y}}\right) \\ \tilde{H}(z) &= \left(\frac{\tilde{Y}(z)}{\tilde{X}(z)}\right) \left(\frac{\bar{x}}{R_{agc}}\right)\end{aligned}\tag{5.2-33}$$

So, combining equations 5.2-32 and 5.2-33, we finally get:

$$\tilde{H}(z) = \frac{z - 1}{z - (1 - \alpha R_{agc})}\tag{5.2-34}$$

Defining:

$$\hat{x}(n) = \frac{\tilde{x}(n)}{\bar{x}}\tag{5.2-35}$$

$$\hat{y}(n) = \frac{\tilde{y}(n)}{\bar{y}}\tag{5.2-36}$$

We can now find the time domain representation:

$$\begin{aligned}\tilde{H}(z) &= \frac{\hat{Y}(z)}{\hat{X}(z)} \\ \hat{Y}(z) (z - (1 - \alpha R_{agc})) &= \hat{X}(z) (z - 1) \\ \hat{Y}(z) (1 - z^{-1}(1 - \alpha R_{agc})) &= \hat{X}(z) (1 - z^{-1}) \\ \hat{y}(n) - \hat{y}(n-1) (1 - \alpha R_{agc}) &= \hat{x}(n) - \hat{x}(n-1) \\ \hat{y}(n) &= \hat{x}(n) - \hat{x}(n-1) + \hat{y}(n-1) (1 - \alpha R_{agc})\end{aligned}\tag{5.2-37}$$

Moving on, we now approach the end of the journey; finding a small-signal time constant [5-1]. Apply a step input of $\hat{x}(n) = b$, for $n \geq 0$. Using equation 5.2-37 and assuming that $\hat{x}(n)$ and $\hat{y}(n)$ are zero for $n < 0$, we get:

$$\hat{y}(n) = b(1 - \alpha R_{agc})^n \quad (5.2-38)$$

By definition, the time constant is the time interval that it takes the output to decay by a factor of e from the output value at the beginning of the interval. So:

$$\frac{b(1 - \alpha R_{agc})^n}{b(1 - \alpha R_{agc})^0} = e^{-1}$$

$$n = \frac{-1}{\ln [1 - \alpha R_{agc}]} \quad (5.2-39)$$

Equation 5.2-39 is the number of samples; each sample of $8T_b$ ($T_b = \text{bit time}$) wide. We also multiply by 2π to make the units right. Finally:

$$\tau_{agc} = \frac{-2\pi \cdot 8 T_b}{\ln [1 - \alpha R_{agc}]} \quad (5.2-40)$$

5.3 DERIVATION OF THE COEFFICIENTS

In this section we derive the coefficients used by the CDU to implement the AGC loop, ASCL and α . One driving point in these derivations is to have all scaling factors be powers of 2 so that multiplications can be done by left and right accumulator shifts.

5.3.1 Derivation of α

The choice of α determines the AGC loop bandwidth, since R_{agc} is already chosen to be 128. We want to set the loop bandwidth as low as possible, while still being able to track out the amplitude variation (due to signal variation and hardware variations) with the smallest time constant. We assume that the time constant is conservatively bounded by [5-3]:

$$\tau_{agc} \approx 30 \text{ seconds} \quad (5.3-1)$$

The worst case occurs at the lowest data rate (due to the fact that there is more variation in the time period), which is 7.8125 bps. So, using equation 5.2-40, we get

$$30 = -2\pi \frac{8}{7.8125} \frac{1}{\ln(1 - \alpha 128)}$$

$$\ln(1 - \alpha 128) = \frac{-16\pi}{(30)(7.8125)}$$

$$1 - 128 \alpha = 0.8069722$$

$$\alpha = \frac{0.193}{128}$$

$$\alpha = 0.001508 \quad (5.3-2)$$

From appendix 5A, we know that we must select an α such that equation 5A-7 is satisfied and α' is a power of 2. So:

$$\alpha' = \frac{64}{\ln(2)} \alpha \quad (5A-7)$$

$$= 0.13924$$

$$= 2^{-2.84}$$

$$\alpha' \approx 2^{-3} = \frac{1}{8} \quad (5.3-3)$$

Now, going backwards, we get:

$$\alpha = \frac{\ln(2)}{64} \alpha'$$

$$\alpha = 0.0013538 \quad (5.3-4)$$

Using equation 5.2-40, we get:

$$\tau_{agc} = 33.8 \text{ seconds} \quad (5.3-5)$$

Thus, the minimum time constant at the lowest data rate is 33.8 seconds. A table of the time constants and the associated cutoff frequency, f_{agc} , which is equal to $1/\tau_{agc}$ is located in Section 5.4.1.

5.3.2 Derivation of ASCL

There are two steps involved in generating the scaling number ASCL. First, we must generate the limit required on the amplitude of the signal. There are two limits to consider: saturating the ADC and overflowing the data accumulator (DACC). With these limits in hand, we then can calculate the scaling factor ASCL.

5.3.2.1 Limit Due to the ADC Saturation

We define the input signal as:

$$X_i = A(r) d(t) \cos(\omega t + \phi) + n(t) \quad (5.3-6)$$

where

- X_i = the signal out of the AGC amplifier, sampled at time i
- $A(r)$ = the data rate dependent peak amplitude
- r = data rate index; ranges from 3 to 9
- $d(t)$ = the data modulation (± 1)
- ω = subcarrier frequency
- ϕ = phase error
- $n(t)$ = band limited, white gaussian noise

We set the probability of saturating the ADC to be less than 0.1.

Thus:

$$\text{Prob} \left(|X_i| > V_S \right) < 0.1 \quad (5.3-7)$$

where V_s is the ADC saturation voltage, which is ± 5 volts. Since we have white gaussian noise, the probability density function for X (note we are dropping the subscript i from X_i) is:

$$f_x(x) = \frac{1}{\sqrt{2\pi \sigma^2}} e^{-(x-\mu)^2/2\sigma^2} \quad (5.3-8)$$

where

$$\mu = \pm A(r)$$

$$\sigma^2 = N_0 B_N$$

N_0 = noise spectral density

B_N = noise bandwidth of the bandpass filter (one-sided)

We are given the requirement to design the AGC for threshold operation, which is defined to be at ST_b/N_0 equal to 10.5 dB [5-4]. Thus we have:

$$N_0 = \frac{ST_b}{10^{1.05}} \quad (5.3-9)$$

We have the definitions:

$$S = \frac{A^2(r)}{2} \quad (5.3-10)$$

$$T_b = \frac{2^{r-1}}{2000} \quad (5.3-11)$$

Returning to equation 5.3-7, we find:

$$\text{Prob } (\langle X \rangle > V_s) = \text{Prob } (X > V_s) + \text{Prob } (X < -V_s)$$

$$\text{Prob } (\langle X \rangle > V_s) = 1 - \text{Prob } (-V_s \leq X \leq V_s) \quad (5.3-12)$$

Or:

$$\text{Prob} (|X| > V_s) = 1 - \left[\int_{-\infty}^{V_s} f_x(x) dx - \int_{-\infty}^{-V_s} f_x(x) dx \right] \quad (5.3-13)$$

Making a substitution:

$$t = \frac{x - \mu}{\sigma} \quad (5.3-14)$$

$$dt = \frac{dx}{\sigma} \quad (5.3-15)$$

$$f_x = \frac{e^{-(x-\mu)^2/2\sigma^2}}{\sqrt{2\pi}\sigma} = \frac{e^{-t^2/2}}{\sigma\sqrt{2\pi}} \quad (5.3-16)$$

We then get:

$$\text{Prob} = 1 - \left[\int_{-\infty}^{\frac{V_s - \mu}{\sigma}} \frac{e^{-t^2/2}}{\sqrt{2\pi}} dt - \int_{-\infty}^{\frac{-V_s - \mu}{\sigma}} \frac{e^{-t^2/2}}{\sqrt{2\pi}} dt \right] \quad (5.3-17)$$

Recalling the definition of $Q(x)$, the normal random variable distribution function:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-t^2/2} dt \quad (5.3-18)$$

$$Q(-x) = 1 - Q(x) \quad (5.3-19)$$

We get:

$$\text{Prob} = 1 - \left[\left(1 - Q \left(\frac{V_s - \mu}{\sigma} \right) \right) - \left(1 - Q \left(\frac{-V_s - \mu}{\sigma} \right) \right) \right]$$

$$= 1 - \left[Q \left(\frac{\mu - V_s}{\sigma} \right) - Q \left(\frac{V_s + \mu}{\sigma} \right) \right]$$

$$\text{Prob} (|X| > V_s) = Q \left(\frac{V_s - \mu}{\sigma} \right) + Q \left(\frac{V_s + \mu}{\sigma} \right) \quad (5.3-20)$$

Since we are requiring that equation 5.3-20 be less than 0.1, we are in the tail of the distribution, so the $Q \left(\frac{V_s - \mu}{\sigma} \right)$ term will dominate; this gives us:

$$0.1 > Q \left(\frac{V_s - \mu}{\sigma} \right) \quad (5.3-21)$$

$$Q^{-1} (0.1) < \frac{V_s - \mu}{\sigma}$$

$$1.281 < \frac{V_s - \mu}{\sigma}$$

$$1.281\sigma - V_s < -\mu$$

$$\mu < V_s - 1.281\sigma \quad (5.3-22)$$

Since we have equiprobable data, we can let μ equal $A(r)$. From Section 3.1, we know that B_N equals 3907 Hz. Solving for σ^2 , we find:

$$\begin{aligned}\sigma^2 &= B_N N_0 \\ &= 3907 \frac{1}{10^{1.05}} \frac{A^2(r)}{2} \frac{2^{r-1}}{2000} \\ &= 0.0435 A^2(r) 2^r \\ \sigma &= 0.2086 2^{r/2} A(r)\end{aligned}\tag{5.3-23}$$

Plugging into equation 5.3-22:

$$\begin{aligned}A(r) &< 5.0 - 1.281 (0.2086 2^{r/2} A(r)) \\ A(r) (1 + 0.2673 2^{r/2}) &< 5.0 \\ A(r) &< \frac{5.0}{1 + 0.2673 2^{r/2}}\end{aligned}\tag{5.3-24}$$

We must now calculate the other limit.

5.3.2.2 Limit Due to Accumulator Overflow

Since we have a 16-bit accumulator, we must calculate the probability of overflowing the data accumulator over the course of summing over a data bit period. We require that the probability of an overflow to be less than 0.001. So:

$$\text{Prob} (|Y| > 2^{15} - 1) < 0.001\tag{5.3-25}$$

where

$$Y = \sum_{i=1}^{NSB} X_i$$

NSB = Number of samples per bit, 2^{r+1}

We assume that the samples X_i and X_j , $i \neq j$, are independent. This gives us the sum of independent gaussians. So:

$$\sigma_y^2 = NSB \sigma^2 \quad (5.3-26)$$

$$\mu_y = NSB \mu \quad (5.3-27)$$

In fact, the distribution for the DAC overflow is the same as the ADC saturation's distribution, with the following changes:

$$X \rightarrow Y$$

$$\sigma^2 \rightarrow 2^{r+1} \sigma^2$$

$$\mu \rightarrow 2^{r+1} \mu$$

$$V_s \rightarrow 2^{15} - 1 = 32767$$

Again, realizing that we are in the tail of the distribution, we get

$$0.001 > Q\left(\frac{32767 - \mu}{\sigma}\right) \quad (5.3.28)$$

$$Q^{-1}(0.001) < \frac{32767 - \mu}{\sigma}$$

$$3.09 < \frac{32767 - \mu}{\sigma}$$

$$3.09 \sigma - 32767 < -\mu$$

$$\mu < 32767 - 3.09 \sigma \quad (5.3-29)$$

Substituting our new definitions (from above) into equation 5.3-29, we get:

$$2^{r+1} A(r) < 32767 - 3.09 (2^{r+1})^{1/2} (0.2086) 2^{r/2} A(r)$$

$$A(r) (2^{r+1} + 0.911565 2^r) < 32767$$

$$A(r) < \frac{32767}{2.911565} 2^{-r}$$

$$A(r) < 11254 2^{-r} \quad (5.3-30)$$

From the above, it is obvious that equation 5.3-24 is the dominant constraint, and, therefore, equation 5.3-24 will determine the value of A(r).

5.3.2.3 Calculating <DACC>

The final step we must take before arriving at the scale factor is to calculate the expected value of DACC, <DACC>.

First, we must convert A(r) to a digital number. Since the maximum voltage is +5 volts and the minimum is -5 volts, we get an output digital number between -128 and 127. So if we define V_c to be the center of the quantization level and N to be the digital output value, we have:

$$V_c = \frac{10}{254} N$$

$$V_c = \frac{5}{127} N \quad (5.3-31)$$

The quantization level switch points, i.e., the point where N goes up or down by 1, are the mid-points between adjacent quantization level centers. Thus, we get for the switch point value, S_p :

$$S_p = \frac{5}{256} (2N + 1) \quad (5.3-32)$$

Thus, if a voltage can be represented by N, it satisfies:

$$256 (2(N - 1) + 1) < \text{Voltage} < 256 (2N + 1)$$

$$256 (2N - 1) < \text{Voltage} < 256 (2N + 1) \quad (5.3-33)$$

Finally, we use equations 5.3-24 and 5.3-33 to get a digital representation for the expected value of A(r). We chose the closest power of 2 that is below this value as the digital value of A(r), and call it A_d(r). We are allowed to do this since 5.3-24 is just an upper bound and we do it to allow easy bit shift manipulations.

<DACC> is easily calculated from A_d(r), since it is just the accumulation over a bit of A_d(r). So:

$$\langle \text{DACC} \rangle = \text{NSB } A_d(r)$$

$$\langle \text{DACC} \rangle = 2^{r+1} A_d(r) \quad (5.3-34)$$

Table 5.3-1 shows the results of our calculations:

$$\mu_1(\text{max}) = \text{bound of equation 5.3-24}$$

$$\overline{\mu_1} = \text{digital value of } \mu_1(\text{max}), \text{ from 5.3-33}$$

$$\mu_2 = \text{bound of equation 5.3-30}$$

NSB = number of samples per bit

A_d(r) = the selected digital value of A(r)

A(r) = the voltage value represented by A_d(r)

<DACC> = expected value of DACC

Table 5.3-1. Calculations for <DACC>

Data Rate	r	$\mu_1(\text{max})$	$\overline{\mu_1}$	$\mu_2(\text{max})$	NSB	$A_d(r)$	A(r)	<DACC>
500	3	2.85	73	1406.8	2^4	2^6	2.52	2^{10}
250	4	2.42	62	703.4	2^5	2^5	1.26	2^{10}
125	5	1.99	51	351.7	2^6	2^5	1.26	2^{11}
62.5	6	1.59	41	175.8	2^7	2^5	1.26	2^{12}
31.25	7	1.24	31	87.9	2^8	2^4	0.630	2^{12}
15.625	8	0.95	24	44.0	2^9	2^4	0.630	2^{13}
7.8125	9	0.71	18	22.0	2^{10}	2^3	0.315	2^{13}

Please note that for 7.8125 bps, $A_d(r)$ was chosen to be 2^3 and not 2^4 . This was due to the fact that the assumption that we can ignore one of the terms in equation 5.3-20 is not quite valid for this rate only.

5.3.2.4 Calculating ASCL

The last step in calculating ASCL is very simple. We want to scale the output of the 8-bit accumulator to be equal to R_{agc} when the input signal level is equal to $A_d(r)$. Since the 8-bit accumulator accumulates 8 outputs of the DACC, we get

$$R_{agc} = 2^3 \langle \text{DACC} \rangle \text{ASCL} \quad (5.3-35)$$

$$\text{ASCL} = \frac{128}{8 \langle \text{DACC} \rangle}$$

$$\text{ASCL} = \frac{2^4}{\langle \text{DACC} \rangle} \quad (5.3-36)$$

5.4 RESULTS

The results of the AGC analysis are summarized in the following two sections. First we look at the scaling value, ASCL, and then we look at the larger signal results.

5.4.1 Derivation Results

The results of equations 5.2-40 and 5.3-36 are shown in Table 5.4-1.

Table 5.4-1. Results of Equations 5.2-40 and 5.3-36

Data Rate	r	τ_{agc}	f_{agc}	ASCL
500	3	0.53	1.89	2 ⁻⁶
250	4	1.06	0.946	2 ⁻⁶
125	5	2.11	0.473	2 ⁻⁷
62.5	6	4.23	0.237	2 ⁻⁸
31.25	7	8.45	0.118	2 ⁻⁸
15.625	8	16.9	0.059	2 ⁻⁹
7.8125	9	33.8	0.030	2 ⁻⁹

5.4.2 Larger Signal Analysis Results

To demonstrate the speed that the AGC recovers from a large step input, the following simulation was run: we assume a noise-free case, with the AGC loop at equilibrium. The maximum that the input signal can vary is from 50 mV_{rms} to 300 mV_{rms} [5-4], or a factor of 6. As can be seen in Figure 5-3, the AGC quality tracks this large change; within 23 update times, it is within 1% of the reference value.

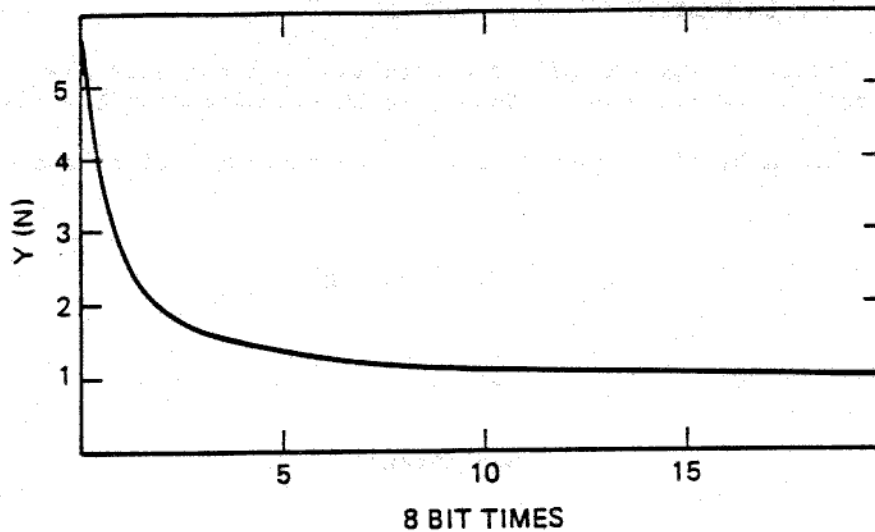


Figure 5-3. AGC Recovery From Large Step Input

5.5 REFERENCES

- 5-1 Morgan, Dennis R., "On Discrete Time AGC Amplifiers," IEEE Transaction on Circuits and Systems, Vol. CAS-22, pp. 135-146, February 1975.
- 5-2 Victor, W.K., and Brockman, M.H., "The Application of Linear Servo Theory to the Design of AGC Loops," Proceedings of the IRE, pp. 234-238, February 1960.
- 5-3 NASA Standard Command Detector Unit Engineering Report, Motorola, February 1977.
- 5-4 Albrecht, V.R., Design Requirements NASA Deep Space Command Detector Unit, DM514438, Rev. A, August 1986.

5A. IMPLEMENTATION OF e^X IN SOFTWARE

Implementing the AGC loop requires that the software can calculate e^X . The method described below does just that, using only bit shifts [5-3].

Let X be the argument of the exponential function and define A and B so that

$$X = 64A + B \quad (5A-1)$$

Or, in other words:

$$A = \text{Integer} \left(\frac{X}{64} \right) \quad (5A-2)$$

$$B = X \pmod{64} \quad (5A-3)$$

Notice that B takes on 64 values, from 0 to 63. It is easy to show (by a simple computer program), that:

$$\left(1 + \frac{B}{64} \right) - 2^{\frac{B}{64}} \leq 0.086069 \quad (5A-4)$$

for B in the range 0 to 63. This allows us to state the following:

$$2^{\left(\frac{X}{64} \right)} = 2^{\left(\frac{(64A+B)}{64} \right)}$$

$$2^{\left(\frac{X}{64} \right)} \approx 2^A \left(1 + \frac{B}{64} \right) \quad (5A-5)$$

and, by definition of exponentials and logarithms:

$$2^{\left(\frac{X}{64} \right)} = e^{(\ln 2) \left(\frac{X}{64} \right)}$$

$$2^{\left(\frac{X}{64} \right)} = e^{\left(\frac{\ln 2}{64} \right) X} \quad (5A-6)$$

So, to implement $e^{\alpha x}$, we can define α' such that

$$e^{\alpha x} = e^{\left(\frac{\ln 2}{64}\right) \alpha' x} \quad (5A-7)$$

$$\alpha' = \left(\frac{64}{\ln 2}\right) \alpha \quad (5A-8)$$

Then, to take $e^{\alpha x}$ we do the following. First, multiply x by α' (this means that α' must be a power of 2). Then shift the result to the right 6 times (dividing by 2^6 or 64). The remaining number is A , the part that was shifted away is B . Due to the fact that we have 16 bit registers, left shifting by A (multiplying by 2^A) is the same as right shifting by $16-A$. Create a 32 bit register by combining two 16 bit registers; put 0001 hex into the upper register and put the shifted out portion of X into the lower register. The lower register now contains $B/64$, the upper contains 1, together we have $1 + B/64$. Right shift the registers by $16-A$ and the result in the lower register is $e^{\alpha x}$. This implementation is the most efficient in terms of processor clock cycles used for this calculation.

Substituting equations 6.2-1, 6.2-3, and 6.2-4 into equation 6.1-19, we get:

$$B'_L = \frac{1}{4} \frac{127}{5} 8 \text{ BSCL} \frac{\pi}{2T_b} K_g A$$

$$B'_L = 25.4 \pi \frac{\text{BSCL} K_g A}{T_b} \quad (6.2-5)$$

So all that now remains is to derive BSCL.

6.2.2 Derivation of BSCL

To derive BSCL, we make use of the following constraint

$$W_L T_b = 2B'_L T_b \ll 1 \quad (6.2-6)$$

This gives us an upper bound on BSCL. We then run bit error tests (BER tests) at the command threshold condition, with and without Doppler offsets, with BSCL values that are below this bound and choose the one, for each data rate, that gives the best bit error performance.

To make use of equation 6.2-6, we must first find K_g . Recalling equations 6.1-7 and 6.1-8 we have:

$$K_g = \text{erf}(\sqrt{R_s}) - \frac{e^{-R_s}}{2} \sqrt{\frac{R_s}{\pi}} \quad (6.1-7)$$

$$R_s = \frac{A^2 T_b}{N_0} \quad (6.1-8)$$

Now, the threshold ST_b/N_0 is defined [6-8] to be 10.5 dB, and we know that

$$\frac{ST_b}{N_0} = \frac{A^2 T_b}{2 N_0} \quad (6.2-7)$$

So:

$$\begin{aligned} R_s &= \frac{A^2 T_b}{N_0} \\ &= \frac{2ST_b}{N_0} \end{aligned} \quad (6.2-8)$$

$$R_s = 22.44$$

Then, substituting equation 6.2-8 into equation 6.1-7, we get:

$$K_g = 1 \quad (6.2-9)$$

Using the above and equation 6.2-5 we get:

$$\begin{aligned} 2B_L' T_b &= (2) (25.4 \pi) \frac{(BSCL)(A)}{T_b} T_b \\ 2B_L' T_b &= 50.8 \pi (A) (BSCL) \end{aligned} \quad (6.2-10)$$

Combining equations 6.2-10 and 6.2-6 gives us the following:

$$50.8 \pi (A) (BSCL) < 1$$

$$BSCL < \frac{1}{50.8 \pi A} \quad (6.2-11)$$

Defining this bound on BSCL as $BSCL_B$ and using the results of Section 5, we can calculate the data rate dependent bounds on BSCL, which are presented in the following table.

Table 6-1. Data Rate Dependent Bounds on BSCL

Data Rate	r	A	BSCL _B
500	3	2.52	2 ^{-8.65}
250	4	1.26	2 ^{-7.65}
125	5	1.26	2 ^{-7.65}
62.5	6	1.26	2 ^{-7.65}
31.25	7	0.630	2 ^{-6.65}
15.625	8	0.630	2 ^{-6.65}
7.8125	9	0.315	2 ^{-5.65}

We can now demonstrate that our assumption that $R_s/B_L'T_b$ is large is true. From equation 6.2-6 we get:

$$2B_L'T_b < 1$$

Or

$$\frac{1}{B_L'T_b} > 2 \quad (6.2-12)$$

Combining the above with equation 6.2-8, we see that:

$$\frac{R_s}{B_L'T_b} > (2) \quad (22.44)$$

$$\frac{R_s}{B_L'T_b} > 44.88 \quad (6.2-13)$$

Thus our assumption holds.

Combining equations 6.1-23, 6.1-21, and 6.2-5, we get the following bit sync jitter:

$$\sigma_{\theta}^2 = (2\pi)^2 \left(\frac{1}{2}\right) \frac{1}{R_s} (25.4 \pi) \frac{(\text{BSCL})(A)}{T_b} T_b \text{ rad}^2$$

$$= \left(\frac{180}{\pi}\right)^2 \frac{(50.8 \pi^3)}{22.44} (\text{BSCL}) (A) \text{ deg}^2$$

$$\sigma_{\theta}^2 = (73347.594) \pi (A) (\text{BSCL}) \text{ deg}^2 \quad (6.2-14)$$

Thus, all that remains is to pick the BSCL numbers and substitute them into equations 6.2-14 and 6.2-5.

6.3 RESULTS

The final results are shown in Table 6-2. The values chosen for BSCL are the values that provided the best BER performance at the threshold condition. Given the optimum BSCL, B'_L and σ_{θ} are found.

Table 6-2. In-Lock Values

Data Rate	r	A	BSCL	B'_L (Hz)	σ_{θ} (deg)
500	3	2.52	2 ⁻¹¹	49.1	16.8
250	4	1.26	2 ⁻¹¹	12.3	11.9
125	5	1.26	2 ⁻¹¹	6.14	11.9
62.5	6	1.26	2 ⁻¹²	1.53	8.4
31.25	7	0.630	2 ⁻¹²	0.384	5.95
15.625	8	0.630	2 ⁻¹⁰	0.767	11.9
7.8125	9	0.315	2 ⁻⁹	0.384	11.9

As can be seen in Table 6-2, the loop bandwidths for the lower data rates are quite small. This caused problems during acquisition; the bandwidth was too small to achieve bit sync in the required time to achieve lock. To correct this, the four lowest data rates have different BSCL values when the CDU is out-of-lock. When lock is achieved, the BSCL values are changed back to the values in Table 6-2, tightening up the loop. The out-of-lock values are given in Table 6-3.

Table 6-3. Out-of-Lock Values

Data Rate	r	BSCL (OOL)	B'_L (OOL)
62.5	6	2 ⁻⁹	12.27
31.25	7	2 ⁻⁹	3.07
15.625	8	2 ⁻⁷	6.14
7.8125	9	2 ⁻⁶	3.07

6.4 REFERENCES

- 6-1 Simon, M. K., "An Analysis of the Steady-State Phase Noise Performance of a Digital Data-Transition Tracking Loop," JPL Space Programs Summary 37-55, Vol. III, February 28, 1969, pp. 54-62.
- 6-2 Simon, M. K., An Analysis of the Steady State Phase Noise Performance of a Digital Data-Transition Tracking Loop, JPL Document 900-222, November 21, 1968.
- 6-3 Simon, M. K., Cycle Slipping of a Data-Transition Bit Synchronizer, JPL Document 900-228, November 13, 1968.
- 6-4 Lindsey, W. C., and Simon, M. K., Telecommunications Systems Engineering, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973, pp. 442-457.
- 6-5 Holmes, Jack K., and Tegnalia, Carl R., "A Second-Order All-Digital Phase-Locked Loop," IEEE Transactions On Communications, Vol. COM-22, January 1974, pp 62-68.
- 6-6 Simon, M. K., and Springett, J. C., The Theory, Design, and Operation of the Suppressed Carrier Data - Aided Tracking Receiver, JPL Technical Report 32-1583, June 15, 1973.

- 6-7 NASA Standard Command Detector Unit Engineering Report, Motorola,
February 1977.
- 6-8 Albrecht, V. R., Design Requirement NASA Deep Space Command Detec-
tor Unit, DM514438, Rev. A, August 1986.
- 6-9 Simon, M. K., Private Communication.

SECTION 6

BIT TIMING SYNCHRONIZATION LOOP

For the proper accumulation of the data samples, an accurate estimation of the end-of-bit epoch is required. The bit sync loop generates this estimate for the CDU.

The bit sync loop is a digital data-transition tracking loop, and is made up of two branches. The first branch (the in-phase branch) monitors the data bit estimates and checks for transitions. The second branch (the mid-phase branch) is used as a measure of the lack of sync. This loop has been heavily analyzed at the Jet Propulsion Laboratory [6-1 to 6-4]. To avoid reinventing the wheel, the results of this previous analysis will be used throughout this section, which describes the design and implementation of the bit sync tracking loop.

6.1 BIT SYNC TRACKING LOOP MODEL

The block diagram of the bit sync loop is shown in Figure 6-1; the equivalent transition tracking loop is shown in Figure 6-2. A description of each follows.

6.1.1 Bit Sync Loop Block Diagram

The various blocks of Figure 6-1 are:

- (1) **Sample and Hold/ADC** - The bit sync loop requires the data samples to measure the bit sync. These are the same samples used by the AGC loop as well. The sample is taken one

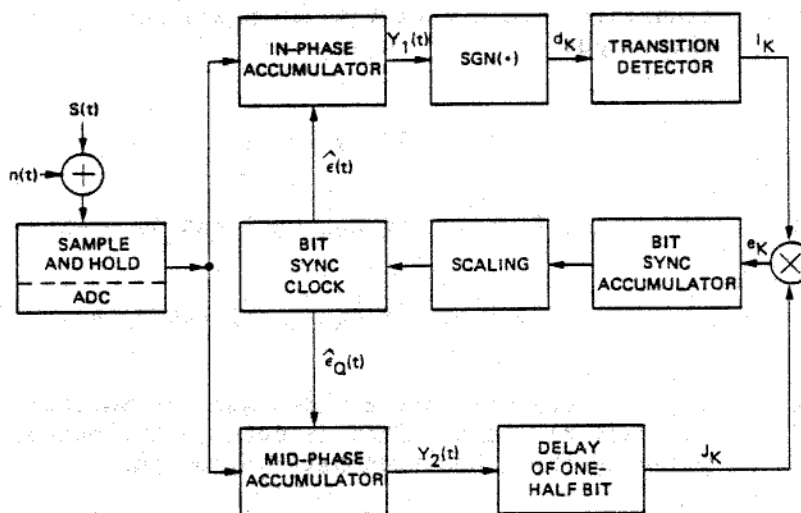


Figure 6-1. Bit Sync Tracking Loop Block Diagram

quarter of a cycle after the error sample. Thus, if the zero crossing error of the subcarrier tracking loop is ϕ and the peak amplitude is A, then the output of the sample and hold is:

$$A \sin (\phi + 90^\circ) = A \cos (\phi)$$

The analog-to-digital converter (ADC) outputs a 7 data bit plus 1 sign bit two's complement digital byte. The saturation voltage of the ADC is ± 5 volts; the maximum digital output is 127. Thus, the output of the ADC is an integer equal to:

$$\frac{127}{5} A \cos (\phi)$$

- (2) In-Phase accumulator - This is the data accumulator used in the AGC loop. The samples are accumulated over a bit time, with the starting point being the estimate of the beginning of the bit period.
- (3) Mid-phase accumulator - These samples are accumulated over one bit time; the only difference is that the starting point is the estimate of the mid bit time.
- (4) Sgn (*) - The received data bit is estimated to be the sign bit of the in-phase accumulator. The output is either -1 or 1.
- (5) Transition Detector - This determines if there was a data transition and scales the mid-phase accumulator accordingly. The output is:

$$I_k = 0 \quad \text{if } d_k = d_{k-1}$$

$$I_k = 1 \quad \text{if } d_{k-1} = 1 \text{ and } d_k = -1$$

$$I_k = -1 \quad \text{if } d_{k-1} = -1 \text{ and } d_k = 1$$

- (6) Delay - This delays the mid-phase accumulator output by one-half bit time. This is done so the output can be multiplied by the Transition Detector output.

- (7) Bit Sync Accumulator - This accumulates the result of multiplying the Transition Detector's output with the delayed mid-phase accumulator, which is the error signal. In the absence of noise, if there was no transition, no accumulation is done for that sample.
- (8) Scaling - The bit sync accumulator is scaled by a data rate dependent value to limit the size of the bit sync correction.
- (9) Bit Sync Clock - The bit sync clock is advanced or retarded as a function of the scaled bit sync accumulator output. The goal is to move the estimated end-of-bit to the nearest sub-carrier cycle. This, along with the subcarrier tracking loop, provides adequate overall bit sync resolution.

While the analysis of this loop is quite involved, the operation is relatively simple. The loop is made up of two branches, the in-phase branch and the mid-phase branch (see Figure 6-1). The mid-phase branch accumulates samples for one bit period, starting at the estimate of the middle of the bit. In the noise free case, if the loop is in perfect sync, and there was a transition, this accumulation would equal zero; if there were no accumulation, this accumulation would be large and void of any sync information. When the loop is not in sync and there is a transition, the mid-phase accumulator has a nonzero value; its sign depends on whether the transition was low to high or high to low and whether the sync estimate is ahead of or behind the actual bit epoch. This is where the in-phase branch comes in. This branch takes the data estimate and checks for a transition; if there was no transition, it outputs a zero; if there was a low to high, it outputs a -1; and if there was a high to low, it outputs a 1. This result is multiplied with the mid-phase accumulator; the result is a zero value if no transition occurred, a positive value if the estimate is behind the epoch, and a negative value if the estimate is ahead. These results are summed over eight transitions and then scaled and used for the bit sync correction.

The bit sync tracking loop block diagram can be expressed as an equivalent phase-locked loop. This will be done in Section 6.1.3, after we discuss the assumptions needed to create this equivalent model.

6.1.2 Assumptions

To ease the analysis, we make the following two assumptions:

- (1) The bit sync tracking error is slowly varying relative to the data rate.
- (2) The accumulate-and-dumps contribute gains equal to the number of samples accumulated and the clock rate is slowed by the number of samples.

The second assumption further constrains the first assumption, since the tracking error must vary slowly compared to the new clock rate. The first

assumption is equivalent to saying that the bit sync two-sided loop bandwidth, W_L , must be much smaller than the data rate:

$$W_L \ll 1/T_b \quad (6.1-1)$$

$$W_L T_b \ll 1 \quad (6.1-2)$$

6.1.3 Equivalent Loop Model

Figure 6-2 is a diagram for the loop in terms of the normalized input epoch, where:

$\epsilon(t)$ = The actual input epoch

$\hat{\epsilon}(t)$ = The estimated input epoch

T_b = The bit period

λ = Normalized bit sync timing error

$$= \frac{\epsilon(t)}{T_b} - \frac{\hat{\epsilon}(t)}{T_b}$$

$g(\lambda)$ = Equivalent loop nonlinearity

$n_\lambda(t)$ = Equivalent timing noise process, which has spectrum $S(\omega, \lambda)$

$F(s)$ = Loop filter

K_{VCO} = Gain of digital VCO

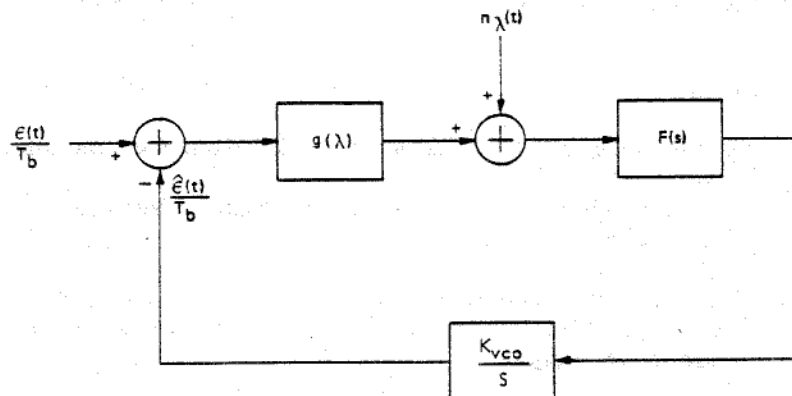


Figure 6-2. Equivalent Data Transition Tracking Loop

In the CDU implementation, the loop filter is just the bit sync accumulator, followed by the loop scaling; thus:

$$F(s) = K_p \quad (6.1-3)$$

Now, $g(\lambda)$ is defined as the conditional expectation on λ , with respect to the noise and symbol sequence, of e_k (see Figure 6-1), the error signal random variable [6-1]. We can define the normalized, nonlinearity, $g_n(\lambda)$, to be:

$$g_n(\lambda) = \frac{g(\lambda)}{K_2 A T_b} \quad (6.1-4)$$

where

A = Input signal amplitude

T_b = Bit time

K_2 = Gain of the midphase accumulator

For our area of interest, $g_n(\lambda)$ can be approximated by:

$$g_n(\lambda) \approx K_g \lambda \quad (6.1-5)$$

where

$$K_g = \left. \frac{\partial g_n(\lambda)}{\partial \lambda} \right|_{\lambda=0} \quad (6.1-6)$$

$$K_g = \operatorname{erf}(\sqrt{R_s}) - \frac{1}{2} \sqrt{\frac{R_s}{\pi}} e^{-R_s} \quad (6.1-7)$$

and

$$R_s = \frac{A^2 T_b}{N_0} \quad (6.1-8)$$

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \quad (6.1-9)$$

Thus, we get the following for $g(\lambda)$:

$$g(\lambda) = K_2 A T_b K_g \lambda \quad (6.1-10)$$

Now, the stochastic differential equation that describes the loop in Figure 6-2 is [6-9]:

$$\frac{d\lambda}{dt} = -K_{vco} K_p [g(\lambda) + n_\lambda(t)] \quad (6.1-11)$$

Using a generalization of the Fokker-Planck technique, we find the probability density function of λ , namely:

$$P(\lambda) = C_1 \exp \left[-\frac{2R_s}{B_L T_b} \frac{K_g}{h(0)} \int_0^\lambda g_n(y) dy \right] \quad |\lambda| \leq \frac{1}{2} \quad (6.1-12)$$

where

B_L = one-sided noise loop bandwidth

$$h(0) = 1 + \frac{R_s}{2} - \frac{1}{2} \left[\frac{1}{\sqrt{\pi}} e^{-R_s} + \sqrt{R_s} \operatorname{erf}(\sqrt{R_s}) \right]^2 \quad (6.1-13)$$

C_1 = Normalizing constant

6.1.4 Loop Transfer Function

To find the loop noise bandwidth, we first must find $H(s)$, the closed loop transfer function. By inspection of Figure 6-2, we see that:

$$\begin{aligned} \frac{\hat{\epsilon}}{T_b} &= K_2 A T_b K_g \lambda K_p \frac{K_{vco}}{s} \quad (6.1-14) \\ &= K_2 A T_b K_g \left[\frac{\epsilon - \hat{\epsilon}}{T_b} \right] K_p \frac{K_{vco}}{s} \end{aligned}$$

$$\hat{\epsilon} \left(\frac{1}{T_b} + \frac{K_2 K_g K_P K_{vco} A T_b}{s T_b} \right) = \frac{K_2 K_g K_P K_{vco} A T_b}{s T_b} \epsilon$$

$$\hat{\epsilon} (s + K_2 K_g K_P K_{vco} A T_b) = K_2 K_g K_P K_{vco} A T_b \epsilon$$

$$H(S) = \frac{\hat{\epsilon}}{\epsilon} = \frac{K_2 K_g K_P K_{vco} A T_b}{s + K_2 K_g K_P K_{vco} A T_b} \quad (6.1-15)$$

6.1.5 Loop Noise Bandwidth

The loop noise bandwidth is defined as

$$B_L = \frac{1}{2} \frac{1}{|H|_{\max}^2} \frac{1}{j2\pi} \int_{-j\infty}^{j\infty} |H(s)|^2 ds \quad (6.1-16)$$

From observing equation 6.1-15, we see that the maximum value of $H(s)$ occurs at $s = 0$ and:

$$|H|_{\max}^2 = 1 \quad (6.1-17)$$

The integral in equation 6.1-16 is easily done using residue theory, which gives the result:

$$B_L = \frac{1}{4} K_2 K_g K_P K_{vco} A T_b \quad (6.1-18)$$

Note that in equation 6.1-18, B_L is in units of bit times. To get B_L in units of Hz, we divide by T_b :

$$B'_L = \frac{K_2 K_g K_P K_{vco} A}{4} \text{ Hz} \quad (6.1-19)$$

6.1.6 Mean-Square Timing Jitter

The mean-square timing jitter, also known as the bit sync jitter, is defined as the variance of λ :

$$\sigma_{\lambda}^2 = \int_{-1/2}^{1/2} \lambda^2 P(\lambda) d\lambda \quad (6.1-20)$$

If we assume that $R_s/B'_L T_b$ is large (which will be demonstrated later), then $P(\lambda)$ approaches a gaussian distribution, and if R_s is large (which is our case) then the following is true for σ_{λ}^2 :

$$\sigma_{\lambda}^2 = \frac{1}{2} \frac{B'_L T_b}{R_s} \quad (6.1-21)$$

Now, σ_{λ}^2 is the variance of the normalized error, i.e., it is unitless. We are interested in the mean-square jitter in units of radians². For every bit, we advance in phase one cycle or 2π radians. Thus, the error, in units of radians, is:

$$\begin{aligned} \theta_e &= 2\pi \left(\frac{\epsilon(t)}{T_b} - \frac{\hat{\epsilon}(t)}{T_b} \right) \\ &= 2\pi\lambda \end{aligned} \quad (6.1-22)$$

Thus:

$$\begin{aligned} \sigma_{\theta}^2 &= \text{Var} (2\pi\lambda) \\ &= (2\pi)^2 \text{Var} (\lambda) \\ \sigma_{\theta}^2 &= (2\pi)^2 \sigma_{\lambda}^2 \end{aligned} \quad (6.1-23)$$

6.2 ANALYSIS OF THE BIT SYNC LOOP

The loop analysis is done in two stages. First, we derive the loop parameters in terms of the actual loop values and then we derive the variable scaling factor BSCL.

6.2.1 Bit Sync Loop Values

First we begin with K_p :

K_p is equal to the number of accumulations done by the bit sync accumulator, which is 8, times the loop scaling factor, BSCL. Thus:

$$K_p = 8 \text{ BSCL} \quad (6.2-1)$$

Now we find K_{vco} :

Every 8-bit transitions, the loop makes a correction. So a correction is made every $8 T_b$ seconds. The output of the scaled bit sync accumulator is Δ , where $\Delta = 0, \pm 1, \pm 2, \dots, \pm (\text{number of samples/bit})/4$. When we make a bit sync bump, we bump in units of sample periods. The number of these units we bump is equal to Δ . Each sampling period is equal to two subcarrier cycles, or $2(2\pi) = 4\pi$ radians. Every $8 T_b$ seconds, we make a bump of $4\pi\Delta$ radians. So, in the model of Figure 6-2, the constant K_{vco} is equal to [6-5]:

$$K_{vco} = \frac{4\pi}{8T_b} \quad (6.2-2)$$

$$K_{vco} = \frac{\pi}{2T_b} \quad (6.2-3)$$

Finally, we find K_2 :

From Section 6.1.1 we know that the ADC converts volts to digital numbers. Thus, the gain is:

$$K_2 = \frac{127}{5} \quad (6.2-4)$$

SECTION 7

LOCK DETECTOR

The CDU lock detector performs two functions. When the CDU is out-of-lock it detects when a signal is present. When the CDU is in-lock, it detects when the signal is absent. The lock detector accomplishes this by comparing the received signal with either a lock threshold number or an unlock threshold number (depending on the CDU's current state). The purpose of this section is to derive these threshold numbers such that the lock detector algorithm achieves the specified lock and unlock probabilities.

7.1 LOCK DETECTOR MODEL

The lock detector relies heavily on the AGC algorithm discussed in Section 5. The block diagram of the lock detector is shown in Figure 7-1:

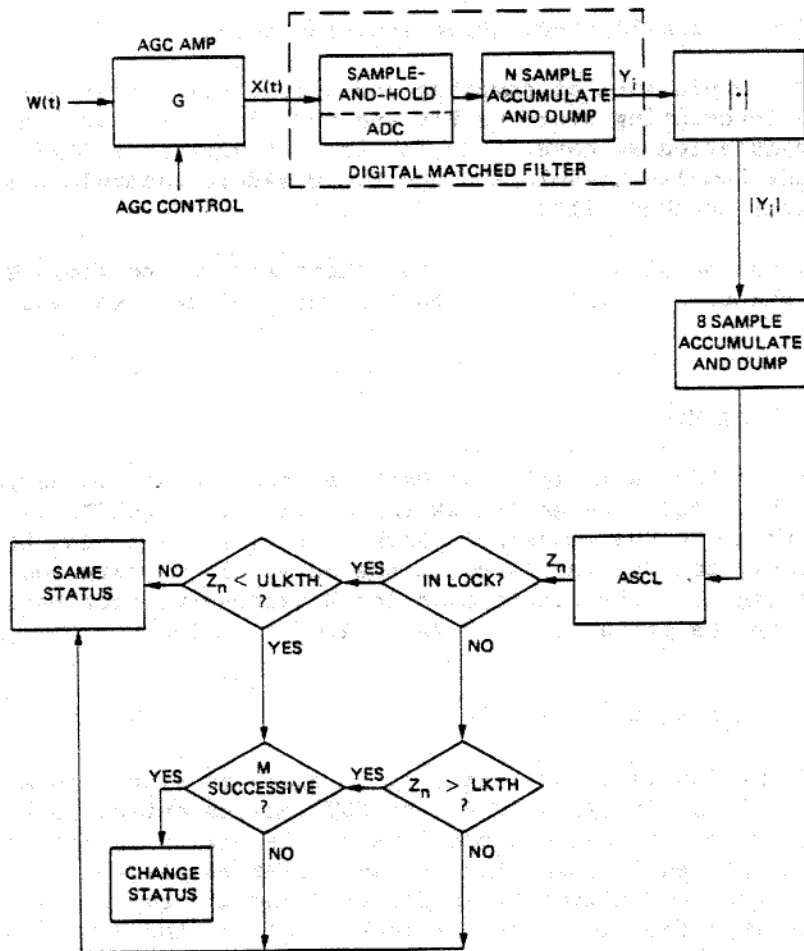


Figure 7-1. Lock Detector Block Diagram

As can be seen in Figure 7-1, up through the definition of signal z_n , the lock detector block diagram is identical to the block diagram of the AGC, which was discussed in Section 5.1. For the upcoming analysis, a couple of new definitions are made. First we define $w(t)$, the input signal to the AGC amplifier ($w(t)$ is the output of the NKT band pass filter), as:

$$w(t) = W d(t) \sin (\omega t + \phi) + n(t) \quad (7.1-1)$$

where:

W = the peak amplitude of the signal

$d(t)$ = the data modulation (± 1)

ω = the subcarrier frequency, 16 kHz

ϕ = the phase error

$n(t)$ = band limited, white gaussian noise.

The combination of sampling and accumulating over one bit time is equivalent to integrating over one bit and dumping in the analog world. For the type of modulation we have, this is just matched filtering; thus, the combination of the Sample-and-Hold/ADC and the N-sample accumulate and dump is called a digital matched filter..

As can be seen in Figure 7-1, there are two possible paths to take, depending on whether the CDU is in-lock or out-of-lock. We will now look at each case.

7.1.1 In-Lock Model

If the CDU is in-lock, it compares the scaled, accumulated over eight bits number (z_n) with an unlock threshold number (ULKTH). If z_n is above ULKTH, then the CDU remains in-lock. If z_n is less than ULKTH for M successive times, the CDU is placed in the out-of-lock state and attempts to reacquire the signal. The requirement for M successive test failures is required to keep the probability of false drop lock low.

7.1.2 Out-Of-Lock Model

If the CDU is out-of-lock, z_n is compared with a lock threshold number (LKTH). If z_n is below LKTH, the CDU remains out-of-lock. If z_n is above LKTH for M successive times, the CDU is placed in the in-lock state. When the CDU is placed in-lock, the AGC amplifier is allowed to seek its quiescent gain (the AGC algorithm is disabled and the amplifier's gain is set to the maximum when the CDU is out-of-lock). Again, the M successive test successes are required to keep the probability of false acquisition low.

7.2 ANALYSIS

The analysis of the lock detector is broken down into two sections: first, we do the actual analysis and then we describe the implementation of this analysis.

7.2.1 Derivation

The end result of our analysis is to provide four probabilities: the probability of not acquiring, the probability of false acquisition, the probability of not dropping lock, and the probability of false drop lock. Thus, it is not surprising that we will be dealing with probability densities and distributions exclusively in this section. The derivation of the probability density of the Sample-and-Hold/ADC is done first. Then the calculation of the probability that z_n is less than a certain value is done. With that probability, we can obtain the needed in-lock and out-of-lock threshold numbers.

7.2.1.1 Density of the ADC

The net result of sampling and then quantizing is to take the continuous probability density function (PDF) of $w(t)$ and convert it into a discrete PDF.

We sample the incoming signal at a rate of 8 kHz, which is one-half the 16 kHz subcarrier frequency. The sample occurs at one-quarter of a subcarrier cycle from the zero crossing of the subcarrier. Thus the sampling instants are:

$$t_s = 2nT_s + \frac{T_s}{4} \quad (7.2-1)$$

where T_s is the subcarrier period.

From Figure 7-1 we see that the signal going into the Sample-and-Hold, $x(t)$, is:

$$x(t) = Gw(t) \quad (7.2-2)$$

Substituting equation 7.2-1 into equation 7.2-2, and assuming perfect phase reference, we get:

$$\begin{aligned} x(t_s) &= G Wd(t_s) \sin \left(\omega \left(2nT_s + \frac{T_s}{4} \right) \right) + G n(t_s) \\ &= G [Wd(t_s) \sin (4\pi n + \frac{\pi}{2}) + n(t_s)] \end{aligned}$$

$$x(t_s) = G[Wd(t_s) + n(t_s)] \quad (7.2-3)$$

Since $n(t)$ is zero mean, band limited white gaussian noise, with variance of σ^2 , $x(t_s)$ are just samples of a white gaussian random variable with mean $GWd(t_s)$ and variance $G^2 \sigma^2$. Also, since the band pass filter's noise bandwidth, B_N , is approximately one-half the sampling rate, the samples are independent.

Making the following definitions:

$$\frac{ST_b}{N_0} = R \quad (7.2-4)$$

$$T_b = \frac{2^{r-1}}{2000}, \quad r = 3 \text{ to } 9 \quad (7.2-5)$$

$$S = \left(\frac{W}{\sqrt{2}}\right)^2 = \frac{W^2}{2} \quad (7.2-6)$$

$$\sigma^2 = N_0 B_N \quad (7.2-7)$$

$$\sigma_x^2 = G^2 \sigma^2 \quad (7.2-8)$$

We then get the following:

$$\begin{aligned} \sigma_x^2 &= G^2 B_N N_0 \\ &= G^2 B_N \frac{T_b S}{R} \\ &= \frac{G^2 B_N W^2 2^{r-1}}{R 2 2000} \\ \sigma_x^2 &= \frac{G^2 W^2 B_N 2^r}{8000 R} \end{aligned} \quad (7.2-9)$$

And since we have equiprobable data, we can assume that the mean is:

$$\mu = GW \quad (7.2-10)$$

Now that we have the samples, we must quantize them in the ADC. This is where the continuous PDF is converted to a discrete PDF.

The ADC outputs an integer between -128 and 127, corresponding to a voltage range of -5 to +5 volts. Thus, the continuous PDF is quantized as is shown in Figure 7-2:

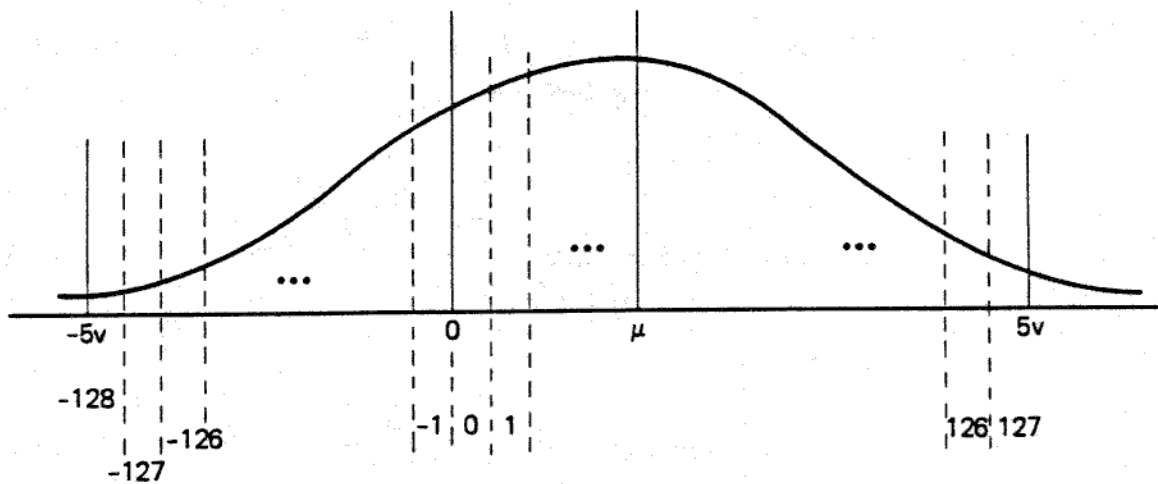


Figure 7-2. Quantized Density of $x(t)$

Except for the two end bins (-128 and 127), all bins are the same width. The bin edges are equidistant between the two adjacent bins (the end bins have only one edge, their other edge is at infinity). If we call the width of the bin q , we get:

$$q = \frac{5}{128} \quad (7.2-11)$$

Thus, the center point of the bin whose digital number is N is:

$$\begin{aligned} \text{center} &= \frac{5}{128} \left(N + \frac{1}{2}\right) \\ &= \frac{q}{2} (2N + 1) \end{aligned} \quad (7.2-12)$$

and the bin edges, are

$$\begin{aligned} \text{edges} &= \left(\frac{5}{128}\right) \left(N + \frac{1}{2} \pm \frac{1}{2}\right) \\ &= \left(\frac{5}{2}\right) \left((2N + 1) \pm 1\right) \end{aligned} \quad (7.2-13)$$

Recall the following definitions for a gaussian random variable

$$\text{Prob } (X > a) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_a^{\infty} e^{-\frac{(t - \mu)^2}{2\sigma^2}} dt \quad (7.2-14)$$

$$\text{Prob } (X < a) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^a e^{-\frac{(t - \mu)^2}{2\sigma^2}} dt \quad (7.2-15)$$

$$\text{erf } (x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \quad (7.2-16)$$

We can now calculate the discrete PDF. There are three cases:

Case I ($k = 127$):

$$P(127) = \frac{1}{\sqrt{2\pi\sigma_x^2}} \int_{127q}^{\infty} e^{-\frac{(x-\mu)^2}{2\sigma_x^2}} dx \quad (7.2-17)$$

$$= \frac{1}{\sqrt{2\pi\sigma_x^2}} \int_{\frac{127q-\mu}{\sqrt{2}\sigma_x}}^{\infty} e^{-t^2} dt \sqrt{2}\sigma_x$$

$$= \frac{1}{2} \frac{2}{\sqrt{\pi}} \int_{\frac{127q-\mu}{\sqrt{2}\sigma_x}}^{\infty} e^{-t^2} dt$$

$$P(127) = \frac{1}{2} \left[1 - \operatorname{erf} \left(\frac{127q - \mu}{\sqrt{2}\sigma_x} \right) \right] \quad (7.2-18)$$

Case II ($k = -128$):

$$P(-128) = \frac{1}{\sqrt{2\pi\sigma_x^2}} \int_{-\infty}^{-127q} e^{-\frac{(x-\mu)^2}{2\sigma_x^2}} dx \quad (7.2-19)$$

$$= \frac{1}{\sqrt{2\pi\sigma_x^2}} \int_{-\infty}^{\frac{-127q - \mu}{\sqrt{2}\sigma_x}} e^{-t^2} \sqrt{2}\sigma_x dt$$

$$= \frac{1}{2} \frac{2}{\sqrt{\pi}} \int_{-\infty}^{\frac{-127q - \mu}{\sqrt{2}\sigma_x}} e^{-t^2} dt$$

$$P(-128) = \frac{1}{2} \left[1 - \operatorname{erf} \left(\frac{127q + \mu}{\sqrt{2}\sigma_x} \right) \right] \quad (7.2-20)$$

Case III ($-127 \leq k \leq 126$):

$$P(k) = \frac{1}{\sqrt{2\pi\sigma_x^2}} \int_{kq}^{(k+1)q} e^{-\frac{(x-\mu)^2}{2\sigma_x^2}} dx \quad (7.2-21)$$

$$= \frac{1}{\sqrt{2\pi\sigma_x^2}} \int_{\frac{kq - \mu}{\sqrt{2}\sigma_x}}^{\frac{(k+1)q - \mu}{\sqrt{2}\sigma_x}} e^{-t^2} \sqrt{2}\sigma_x dt$$

$$= \frac{1}{2} \frac{2}{\sqrt{\pi}} \left[\int_0^{\frac{(k+1)q - \mu}{\sqrt{2} \sigma_x}} e^{-t^2} dt - \int_0^{\frac{kq - \mu}{\sqrt{2} \sigma_x}} e^{-t^2} dt \right]$$

$$P(k) = \frac{1}{2} \left[\operatorname{erf} \left(\frac{(k+1)q - \mu}{\sqrt{2} \sigma_x} \right) - \operatorname{erf} \left(\frac{kq - \mu}{\sqrt{2} \sigma_x} \right) \right] \quad (7.2-22)$$

To sum the cases, and using the notation $P_{1,x}(k)$ to represent the probability of having the value k in one sample of x , we get

$$P_{1,x}(k) = \begin{cases} .5 \left[1 - \operatorname{erf} \left(\frac{127q - \mu}{\sqrt{2} \sigma_x} \right) \right] & k = 127 \\ .5 \left[\operatorname{erf} \left(\frac{(k+1)q - \mu}{\sqrt{2} \sigma_x} \right) - \operatorname{erf} \left(\frac{kq - \mu}{\sqrt{2} \sigma_x} \right) \right] & -127 \leq k \leq 126 \\ .5 \left[1 - \operatorname{erf} \left(\frac{127q + \mu}{\sqrt{2} \sigma_x} \right) \right] & k = -128 \end{cases} \quad (7.2-23)$$

where q is defined in equation 7.2-11.

Now that we have the density of the Sample-and-Hold/ADC, we can find the PDF of the system.

7.2.1.2 Density of z_n

To obtain the density of z_n , we must first sum x over a bit, take the absolute value of the sum, sum the result over eight bits, and finally scale it by ASCL.

We first sum over the number of samples per bit, NSB, which is equal to 2^{r+1} . Adding two samples is equivalent to convolving the two

densities. If we define $P_{j,x}(k)$ as the probability that j samples of $x(t)$ sum to k and define y as the sum of x over one bit, we then get:

$$\begin{aligned}
 P_y(k) &= P_{2^{r+1},x}(k) && (7.2-24) \\
 &= P_{1,x} * P_{1,x} * \dots * P_{1,x}(k) \quad (2^{r+1} \text{ terms})
 \end{aligned}$$

where

$$A * B(k) = \sum_{j=-\infty}^{\infty} A(j) B(k-j) \quad (7.2-25)$$

Now,

$$\begin{aligned}
 P_y(k) &= (P_{1,x} * P_{1,x}) * (P_{1,x} * P_{1,x}) * \dots * (P_{1,x} * P_{1,x}) \quad (2^{r+1} \text{ terms}) \\
 &= (P_{2,x} * P_{2,x}) * \dots * (P_{2,x} * P_{2,x}) \quad (2^r \text{ terms}) \\
 &= (P_{4,x} * P_{4,x}) * \dots * (P_{4,x} * P_{4,x}) \quad (2^{r-1} \text{ terms}) \\
 &\quad \vdots \\
 P_y(k) &= \left(P_{2^r,x} * P_{2^r,x} \right)(k) && (7.2-26)
 \end{aligned}$$

Or, working backwards:

$$\begin{aligned}
 P_y(k) &= P_{2^r, x} * P_{2^r, x} \\
 &= \left(P_{2^{r-1}, x} * P_{2^{r-1}, x} * P_{2^r, x} \right) \\
 &= \left(\left(P_{2^{r-2}, x} * P_{2^{r-2}, x} \right) * P_{2^{r-1}, x} \right) * P_{2^r, x} \\
 &= \left(\left(\left(P_{2^{r-3}, x} * P_{2^{r-3}, x} \right) * P_{2^{r-2}, x} \right) * P_{2^{r-1}, x} \right) * P_{2^r, x} \\
 &\quad \vdots \\
 P_y(k) &= \left(\dots (P_{1, x} * P_{1, x}) * P_{2, x} \right) * P_{4, x} * \dots * P_{2^r, x} (k) \quad (7.2-27)
 \end{aligned}$$

Equation 7.2-27 has just $r+1$ terms, so to find the density of y , convolve the density of x with itself, then convolve the result with itself, then convolve that result with itself, etc. When $r+1$ convolutions have been done, the result is the discrete PDF for y .

Since our accumulators are fifteen bits magnitude, there is a possibility of accumulator overflow. This can occur when the maximum sample value times the number of samples exceeds the maximum accumulator value. Thus:

$$2^{r+1} (2^7 - 1) \geq 2^{15} - 1$$

$$2^{r+1} \geq 258.007$$

$$r+1 \geq 8.01$$

$$r \geq 7.01 \quad (7.2-28)$$

Since r is an integer, equation 7.2-28 tells us that there is a possibility of accumulator overflow for $r = 8$ or 9 . The probability values for sums greater than ± 32767 are folded into the value for ± 32767 . Thus, we can state generally:

$$P_y(k) = \begin{cases} \sum_{j=32767}^{\infty} P_{2^{r+1},x}(j) & k = 32767 \\ P_{2^{r+1},x}(k) & |k| < 32767 \\ \sum_{j=-\infty}^{-32767} P_{2^{r+1},x}(j) & k = -32767 \end{cases} \quad (7.2-29)$$

We now take the absolute value of y , which gives us:

$$P_{|y|}(k) = \begin{cases} P_y(k) + P_y(-k) & k > 0 \\ P_y(0) & k = 0 \\ 0 & k < 0 \end{cases} \quad (7.2-30)$$

Then we sum over eight bits. Using equation 7.2-27, we have:

$$P_{8,|y|}(k) = \left[\left\{ (P_{1,|y|} * P_{1,|y|}) * P_{2,|y|} \right\} * P_{4,|y|} \right] \quad (7.2-31)$$

All that remains is to scale the above result by ASCL. Scaling the eight bit sum has the effect of compressing the total number of discrete values which are allowed. So if we define:

$$L = \frac{1}{ASCL} \quad (7.3-32)$$

$$M = \text{Min} (2^{r+1} (127), 32767) \quad (7.2-33)$$

Then

$$P_z(j) = \sum_{\substack{k=(j-0.5)L \\ k > 0}}^{(j+0.5)L-1} P_{8,|y|}(k), \quad j = 0, 1, \dots, (\text{ASCL}) \cdot (M) \quad (7.2-34)$$

Finally, we can calculate the probability distribution of z_n simply as:

$$F_z(j) = \sum_{i=0}^j P_z(i) \quad (7.2-35)$$

Now all we have to do is calculate these numbers.

7.2.2 Implementation

While the theory presented in Section 7.2.1 is nice, it is useless if we cannot actually compute $F_z(j)$. For example, the lowest data rate, 7.8125, requires ten convolutions to calculate $P_y(k)$, the last two consisting of convolving two arrays of 32767 points.

To ease the strain of computation an approximation was introduced. After the first three convolutions of the summing over a bit, we scale the result by a factor of 2 after each succeeding convolution. This reduces the amount we have to scale the result of the 8 bit sum by a factor of 2. While this does introduce some roundoff error, it is not significant and it allows us to generate the results used in Sections 7.3 and 7.4 in minutes as opposed to hours or days of computer time.

7.3 THRESHOLD NUMBER CALCULATIONS

We now wish to calculate the probabilities used to compute the thresholds. For each threshold, we need to calculate two probabilities, the probability that z_n is greater than k , when there is only noise present, and the probability that z_n is less than k , when there is a signal plus noise present.

7.3.1 Unlock-to-Lock Probabilities

When the CDU is out-of-lock, the AGC amplifier's gain is set to its maximum. This saturates the input to the ADC. Also, depending on the input signal level to the AGC amplifier, it too saturates. Thus the maximum gain of the AGC amplifier depends on the input signal level.

The input signal voltage range is 50 mV_{rms} to 300 mV_{rms} [5-1]. The measured gain is 106.4 for 50 mV_{rms} and 19.2 for 300 mV_{rms}. We also have the following values:

$$B_N = 3907 \text{ Hz} \quad (7.3-1)$$

$$R = 10.5 \text{ dB} = 11.22 \quad (7.3-2)$$

For the signal plus noise case, the worst case condition is the minimum signal level. Thus:

$$W = 50 \sqrt{2} \times 10^{-3} \text{ volts peak} \quad (7.3-3)$$

Substituting into equations 7.2-9, and 7.2-10, we have:

$$\sigma_x^2 = \frac{(106.4)^2 (0.05 \sqrt{2})^2 (3907)^2 2^{\frac{1}{2}}}{(8000) (11.22)}$$

$$\sigma_x = (1.57) 2^{\frac{1}{2}} \quad (7.3-4)$$

$$\mu = (106.4) (0.05 \sqrt{2})$$

$$\mu = 7.52 \quad (7.3-5)$$

As can be seen, if there is a signal when the CDU is out-of-lock, it will saturate the ADC.

The probability that z_n is less than k is just equation 7.2-35. Thus, to get this probability, we just substitute in equations 7.3-4 and 7.3-5.

For the noise only case, we want the maximum level of noise going into the CDU. Since we are dealing at the threshold ST_b/N_0 , the maximum noise input, for a constant ST_b/N_0 , occurs for the maximum signal level. Thus:

$$W = 300 \sqrt{2} \times 10^{-3} \text{ volts peak} \quad (7.3-6)$$

We then get:

$$\sigma_x^2 = \frac{(19.2)^2 (0.3\sqrt{2}) (3907) 2^F}{(8000) (11.22)}$$

$$\sigma_x = (1.70) 2^{\frac{F}{2}} \quad (7.3-7)$$

$$\mu = 0 \quad (7.3-8)$$

The probability that z_n is greater than k is just 1 minus the probability that z_n is less than or equal to k , or:

$$P(z_n > k) = 1 - F_z(k) \quad (7.3-9)$$

The two probabilities, $P(z_n > \text{LKTH})$ for noise only and $P(z_n < \text{LKTH})$ for signal plus noise are shown, for each bit rate, in Figures 7-3 to 7-9, which are in Section 7.4.2.

To obtain LKTH, we must find a threshold that satisfies both the probability of false lock requirement (noise only case) and the probability of not acquiring requirement (signal plus noise case). The two requirements are:

$$P_1 = P(\text{false lock}) \leq 1.0 \times 10^{-4} \quad (7.3-10)$$

$$P_2 = P(\text{not acquiring}) \leq 1.0 \times 10^{-4} \quad (7.3-11)$$

A false lock occurs when for two successive lock intervals z_n is above LKTH. Since each interval is independent of all other intervals,

P (false lock) is just the square of the probability of z_n being greater than LKTH for one interval. Thus:

$$P_f = \sqrt{P_1} \leq 1.0 \times 10^{-2} \quad (7.3-12)$$

The acquisition when there is a signal must take place in 176 bits or approximately 22 eight-bit intervals. To acquire, we require two consecutive intervals with n greater than LKTH. Although we have 22 intervals, we assume that approximately two-thirds of this time is required for achieving sync. So, we assume that we have seven intervals to detect lock, in the worst case.

The requirement of not acquiring is equal to one minus the probability of acquiring. Thus, using the notations of Appendix 7A, we have

$$P(2,5) \geq 1 - 1.0 \times 10^{-4} \quad (7.3-13)$$

$$P(2,5) \geq 0.9999$$

From Table 7A-2, we know that the required probability of z_n being greater than some k is 0.957. So we want the probability that z_n is less than our threshold to be:

$$P_A \leq 1 - 0.957 = 0.043 \quad (7.3-14)$$

From observation of Figures 7-3 to 7-9, we see that we have plenty of margin, so we reduce our requirement to 0.001 for both P_f and P_A , to allow for the difficulty of calculating the probabilities.

7.3.2 Lock-to-Unlock Probabilities

When the CDU is in-lock, the AGC amplifier's gain is set to keep the signal level going into the sample and hold equal to the value calculated

in Section 5 (presented in Table 5.3-1). Thus, for any signal level, we can assume that the gain is unity and the signal level is $A(r)$. So, for the noise only case:

$$\sigma_x^2 = \frac{(1)^2 (A(r))^2 (3907) 2^r}{(8000) (11.22)}$$

$$\sigma_x = 0.0435 (A(r)) 2^{\frac{r}{2}} \quad (7.3-15)$$

$$\mu = 0 \quad (7.3-16)$$

and, for the signal plus noise case:

$$\sigma_x = 0.0435 (A(r)) 2^{\frac{r}{2}} \quad (7.3-17)$$

$$\mu = (1) (A(r)) = A(r) \quad (7.3-18)$$

To obtain ULKTH, we must find a threshold that satisfies both the probability of dropping lock (noise only case) and the probability of false lock drop (signal plus noise case). The two requirements are:

$$P_3 = P(\text{dropping lock}) \geq 0.98 \quad (7.3-19)$$

$$P_4 = P(\text{false lock drop}) \leq 2.5 \times 10^{-9} \quad (7.3-20)$$

For the CDU to drop lock, there must be two consecutive eight-bit intervals where z_n is less than the threshold. Since each interval is independent, we get

$$[P(z_n < \text{ULKTH})]^2 \leq 2.5 \times 10^{-9}$$

$$P(z_n < \text{ULKTH}) \leq 5 \times 10^{-5} \quad (7.3-21)$$

For the CDU to drop lock after removal of the signal, there must be two successive intervals with z_n less than the threshold. Thus:

$$[P(z_n < \text{ULKTH})]^2 \geq 0.98$$

$$[1 - P(z_n \geq \text{ULKTH})]^2 \geq 0.98$$

$$1 - P(z_n \geq \text{ULKTH}) \geq 0.99$$

$$P(z_n \geq \text{ULKTH}) \leq 0.01 \quad (7.3-22)$$

Now, $P(z_n \geq \text{ULKTH})$ for noise only and $P(z_n < \text{ULKTH})$ for signal plus noise are plotted together for the various bit rates in Figures 7-10 to 7-16.

7.4 RESULTS

Using the results of Section 7.3 and the figures presented in Section 7.4.2, we choose the bit rate dependent thresholds LKTH and ULKTH. These numbers are then verified by testing with the CDU breadboard unit. The thresholds are presented in the following section.

7.4.1 Threshold Numbers

The threshold numbers are given in Table 7.4.1.

Notice that the ULKTH numbers are all close to the same value. This is due to the fact that when the CDU is in-lock, the signals are scaled so that the loops are data rate independent.

Table 7.4-1. Threshold Values

r	LKTH	ULKTH
3	130	68
4	213	70
5	166	70
6	123	68
7	180	72
8	132	70
9	190	76

7.4.2 Threshold Figures

A list of the figures in this section follows:

- Figure 7-3: 500 bps, Lock Threshold
- Figure 7-4: 250 bps, Lock Threshold
- Figure 7-5: 125 bps, Lock Threshold
- Figure 7-6: 62.5 bps, Lock Threshold
- Figure 7-7: 31.25 bps, Lock Threshold
- Figure 7-8: 15.625 bps, Lock Threshold
- Figure 7-9: 7.8125 bps, Lock Threshold
- Figure 7-10: 500 bps, Unlock Threshold
- Figure 7-11: 250 bps, Unlock Threshold
- Figure 7-12: 125 bps, Unlock Threshold
- Figure 7-13: 62.5 bps, Unlock Threshold
- Figure 7-14: 31.25 bps, Unlock Threshold
- Figure 7-15: 15.625 bps, Unlock Threshold
- Figure 7-16: 7.8125 bps, Unlock Threshold

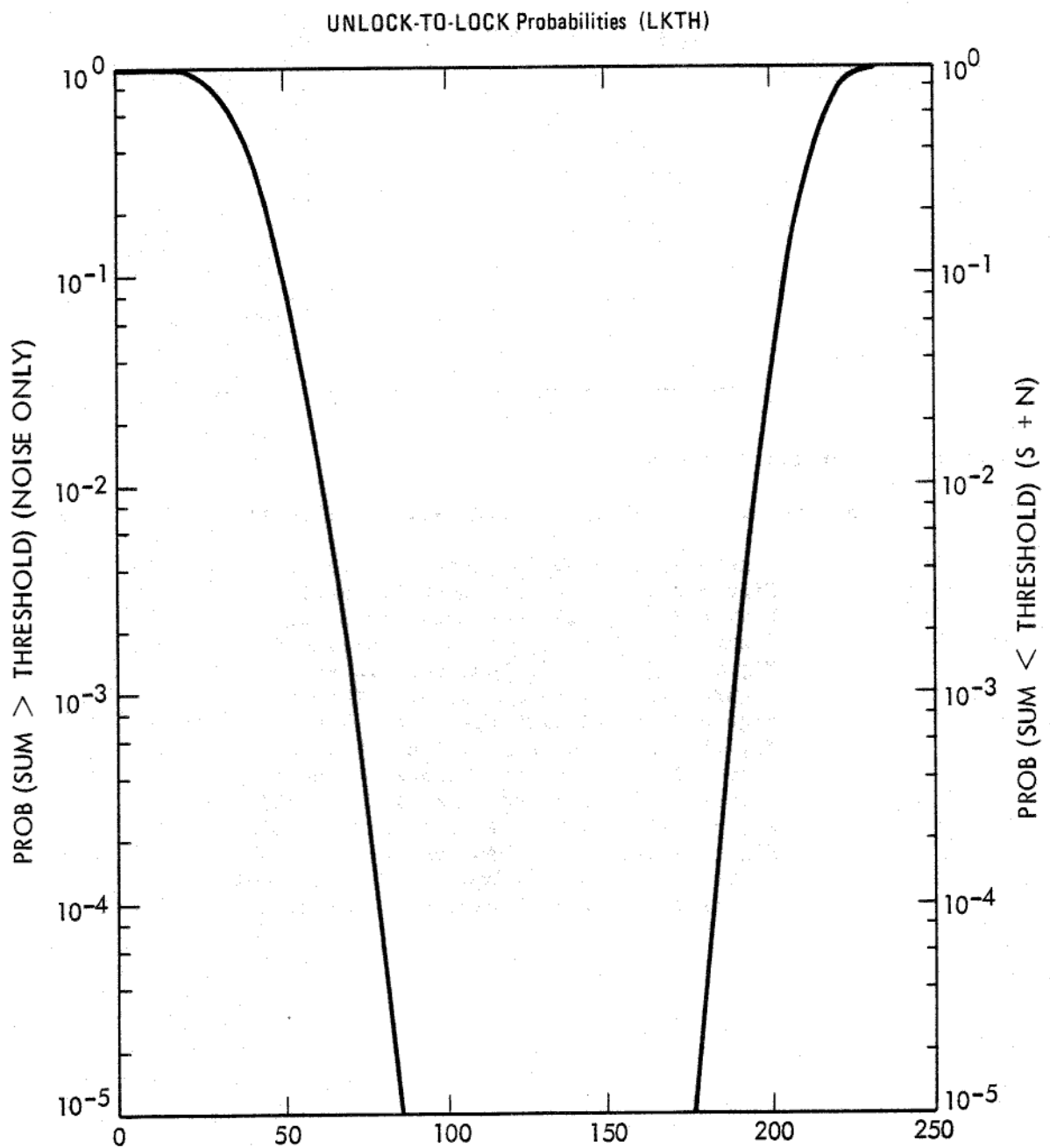


Figure 7-3. Threshold Value, $r = 3$, Rate = 500 bps

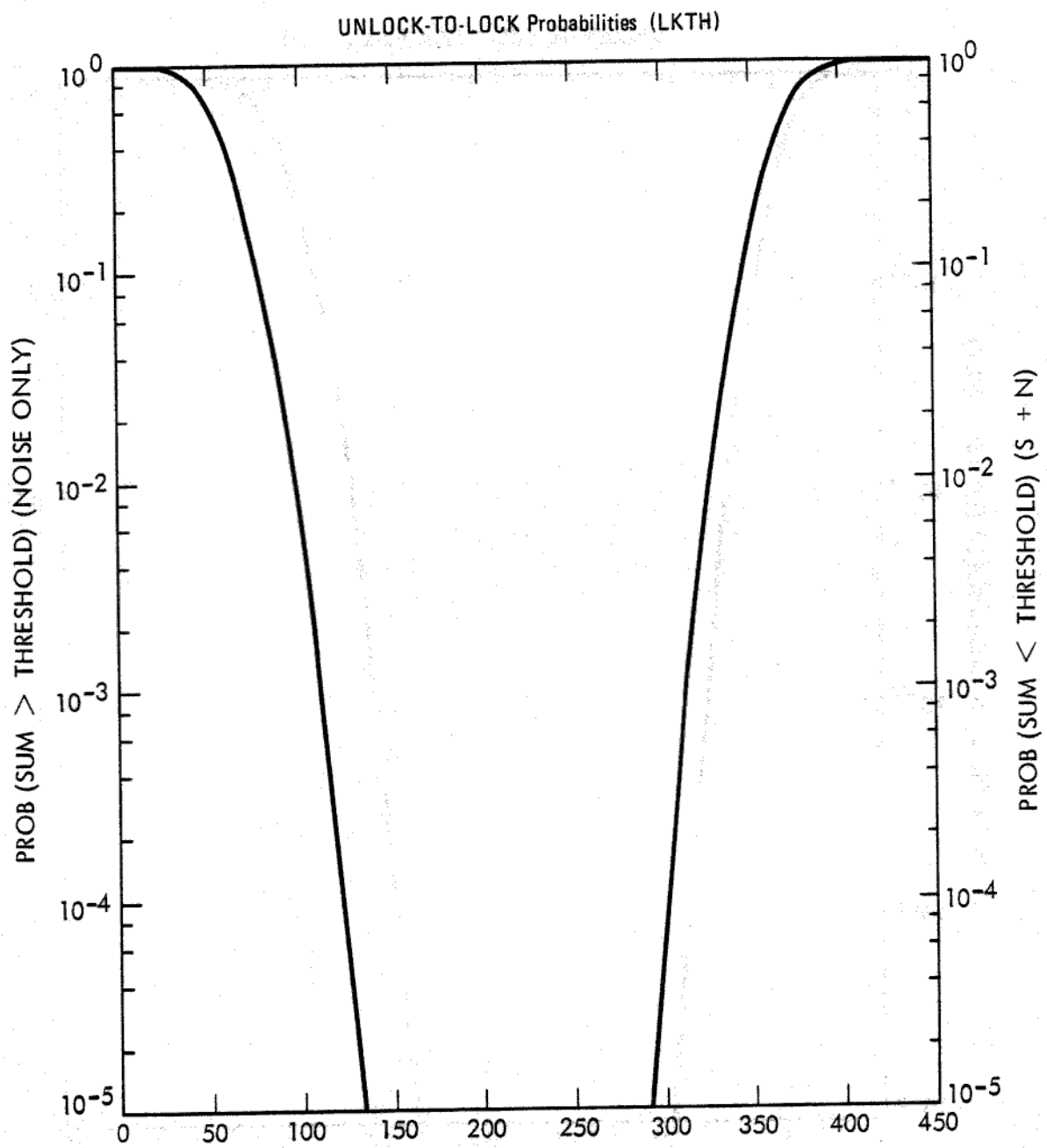


Figure 7-4. Threshold Value, $r = 4$, Rate = 250 bps

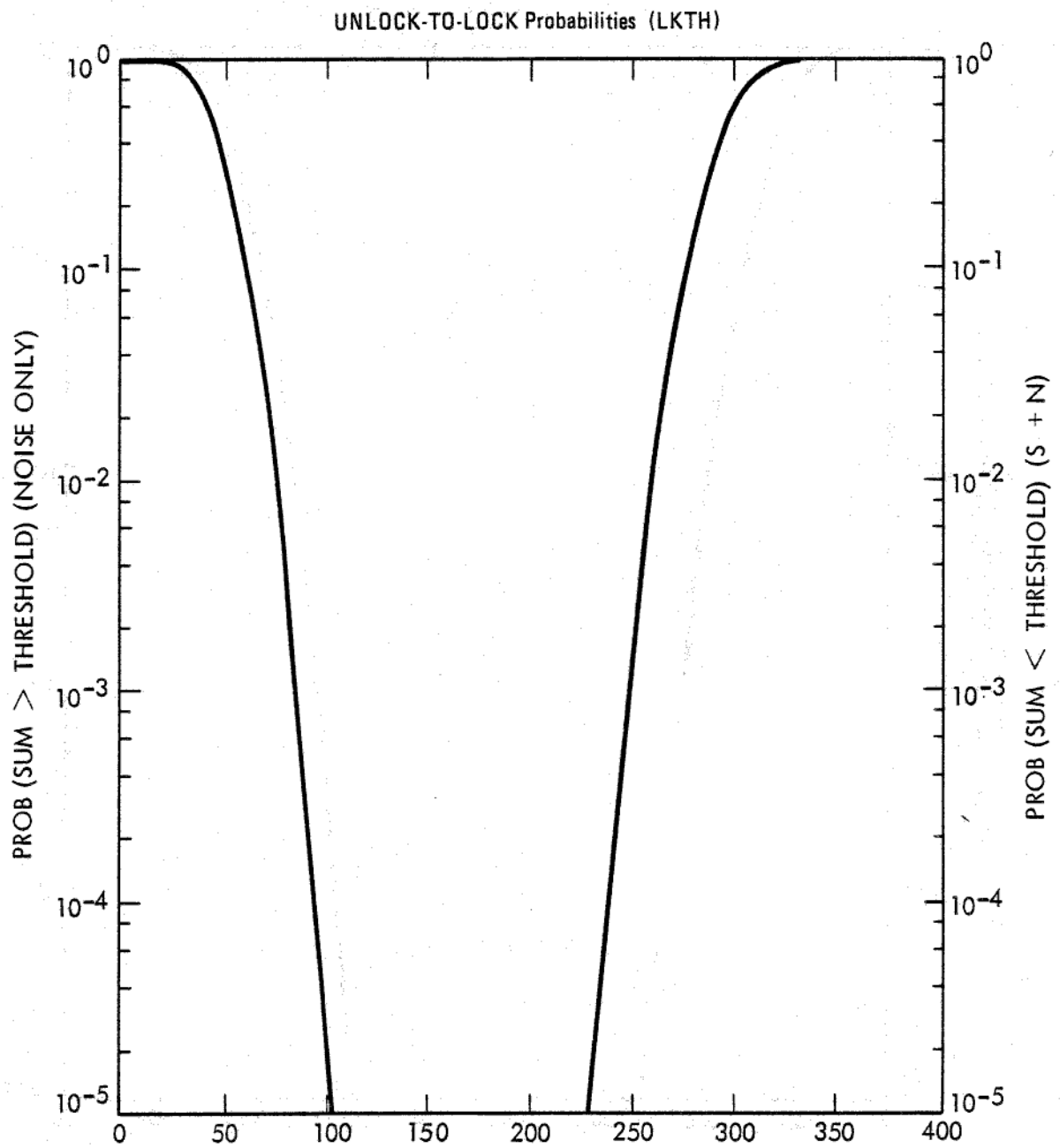


Figure 7-5. Threshold Value, $r = 5$, Rate = 125 bps

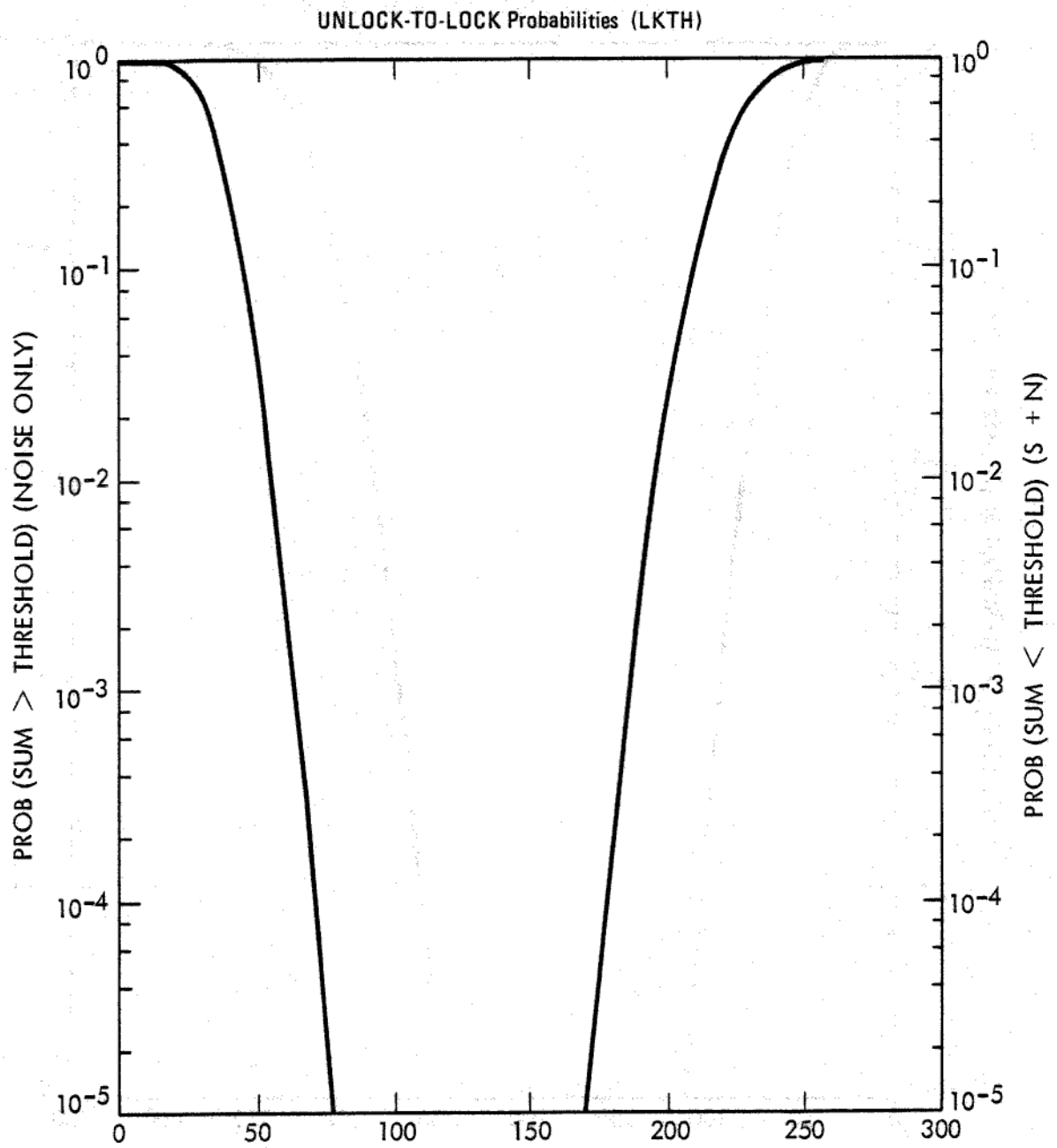


Figure 7-6. Threshold Value, $r = 6$, Rate = 62.5 bps

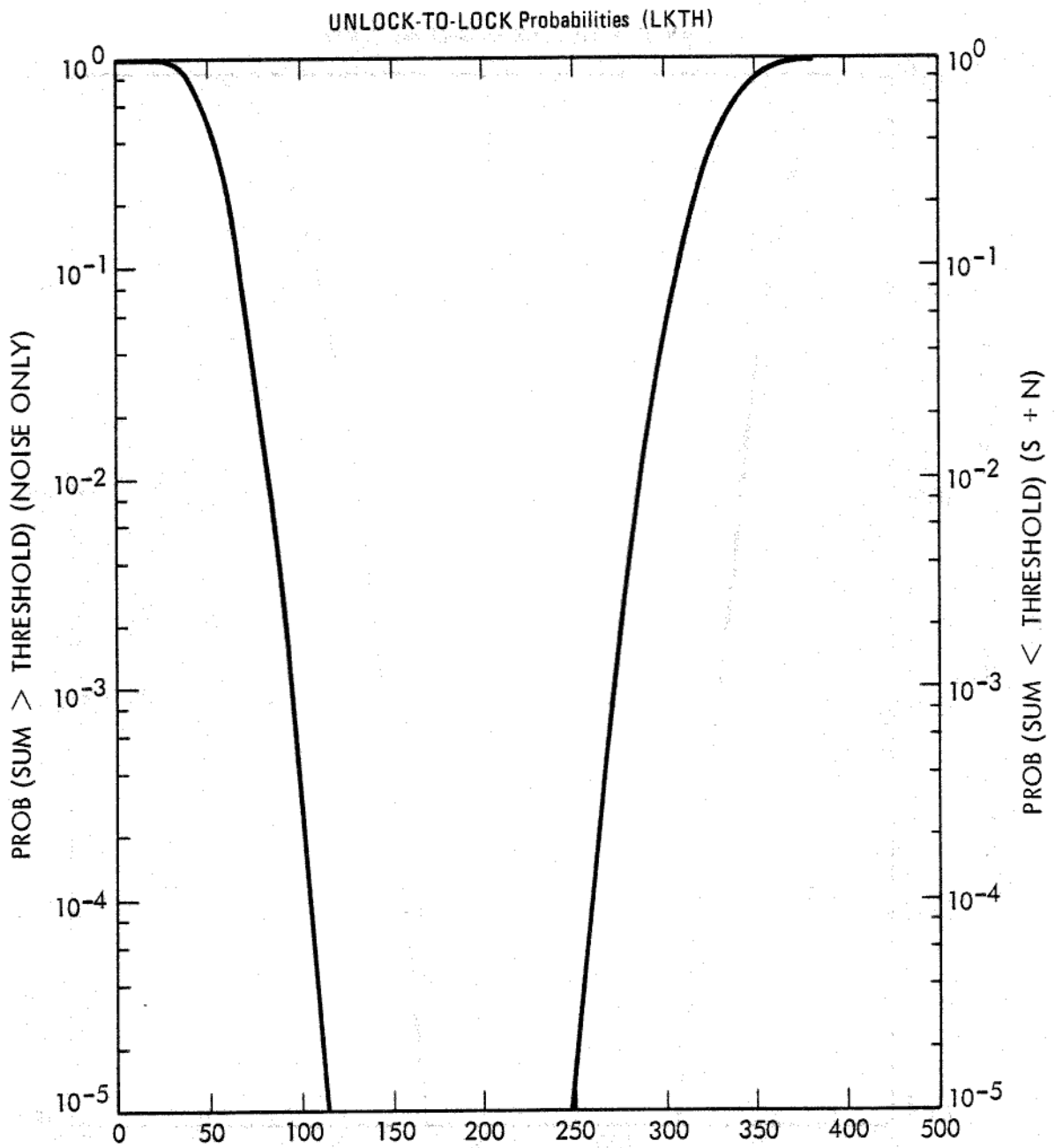


Figure 7-7. Threshold Value, $r = 7$, Rate = 31.25 bps

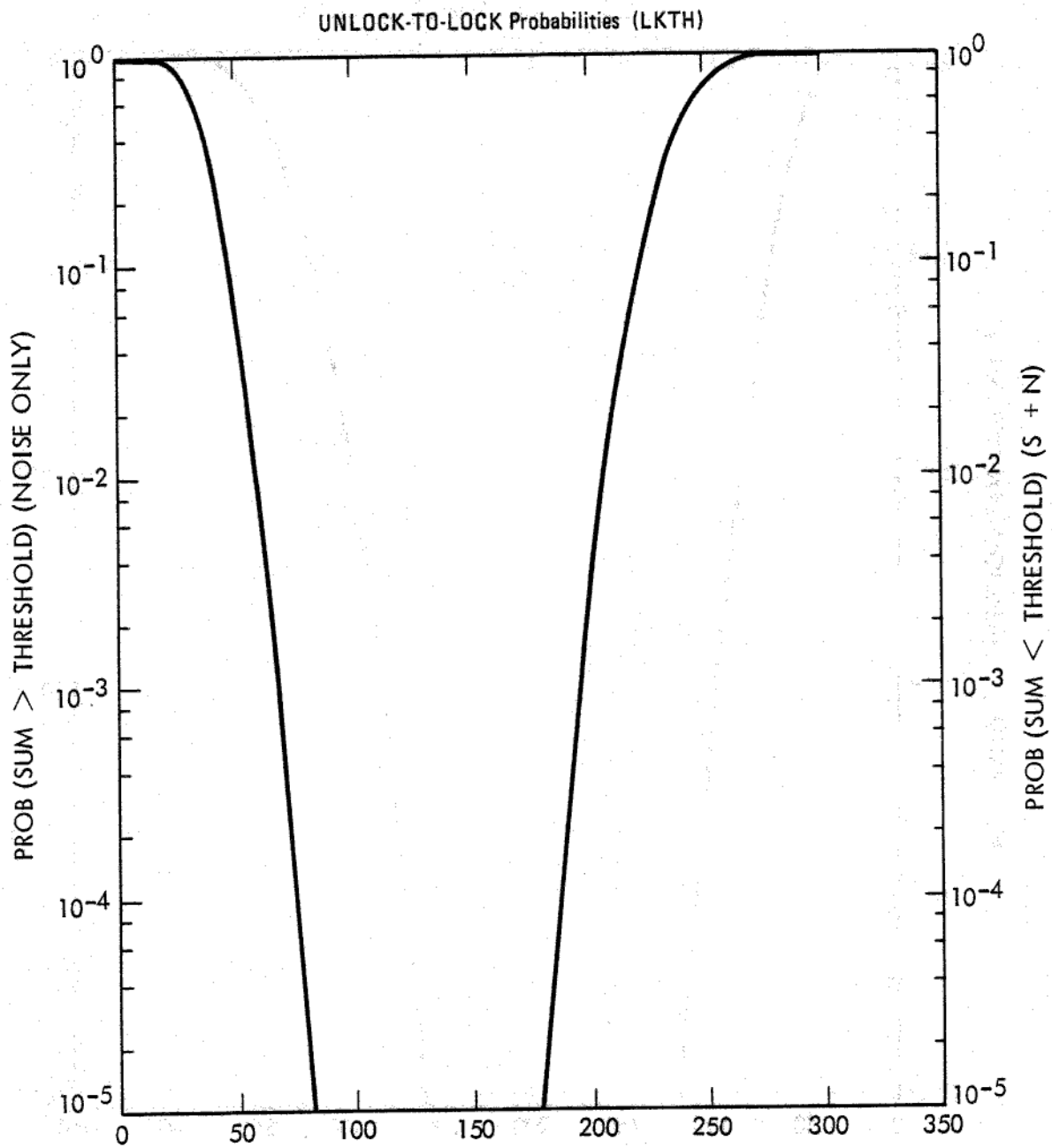


Figure 7-8. Threshold Value, $r = 8$, Rate = 15.625 bps

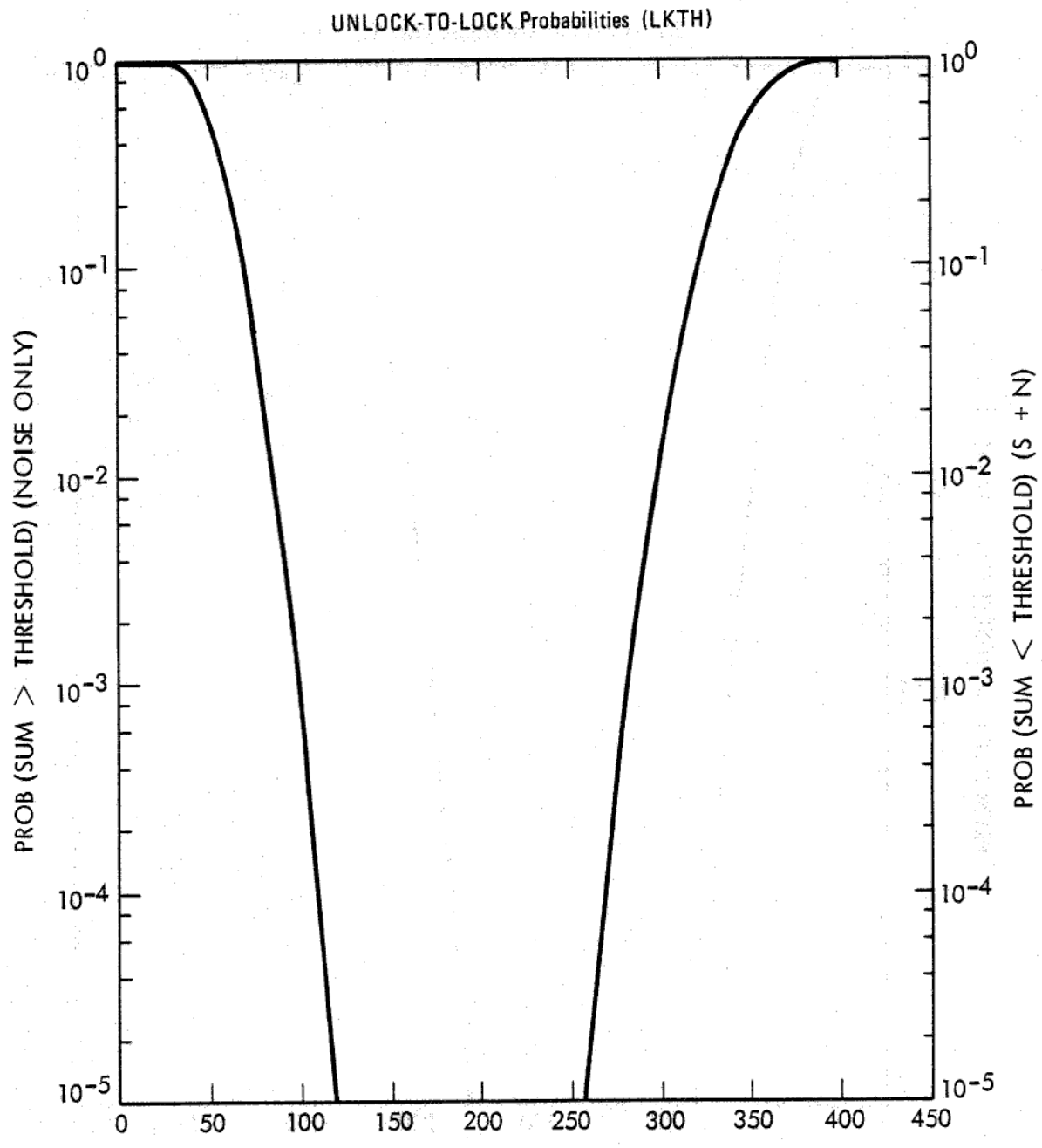


Figure 7-9. Threshold Value, $r = 9$, Rate = 7.8125 bps

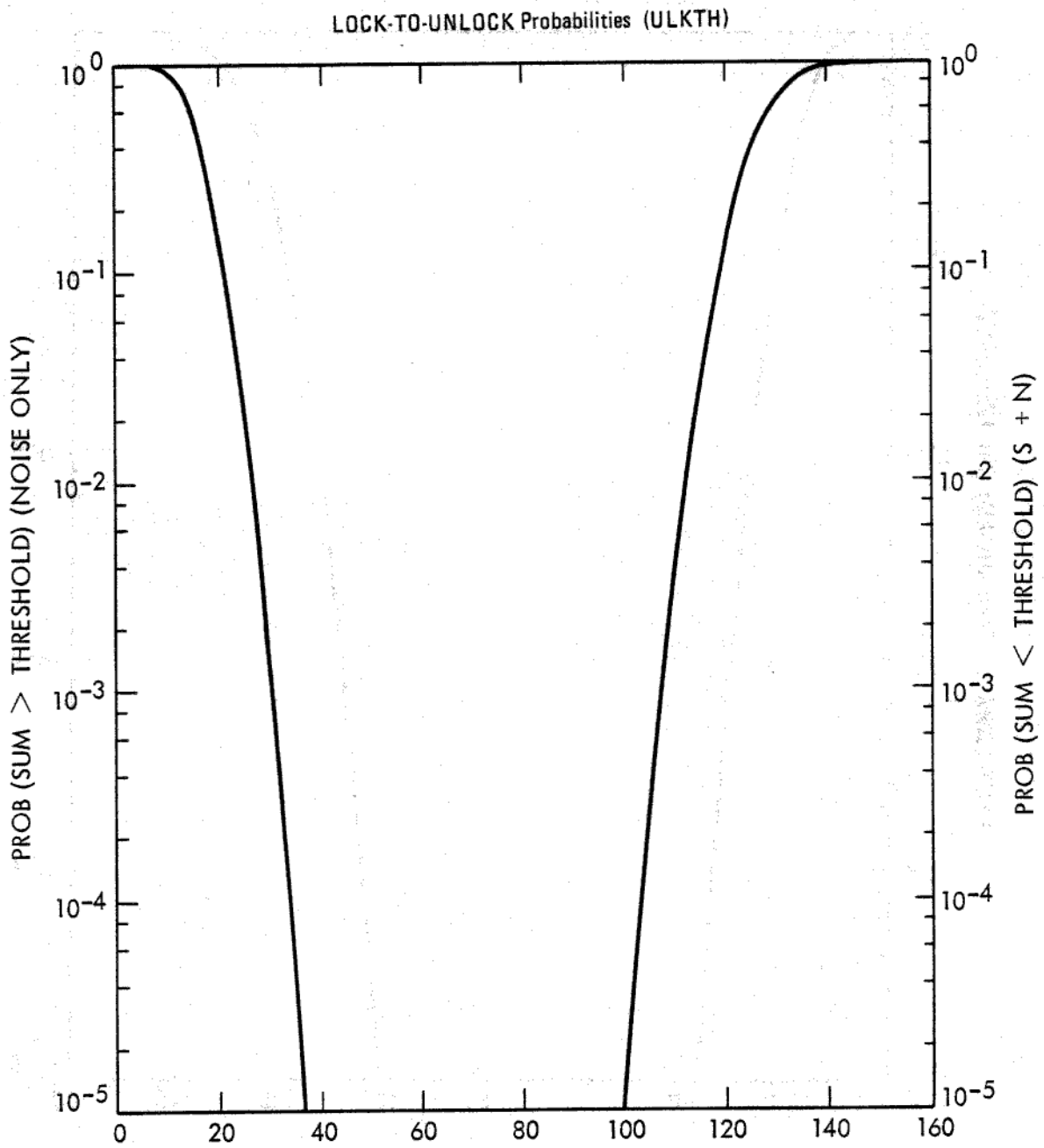


Figure 7-10. Threshold Value, $r = 3$, Rate = 500 bps

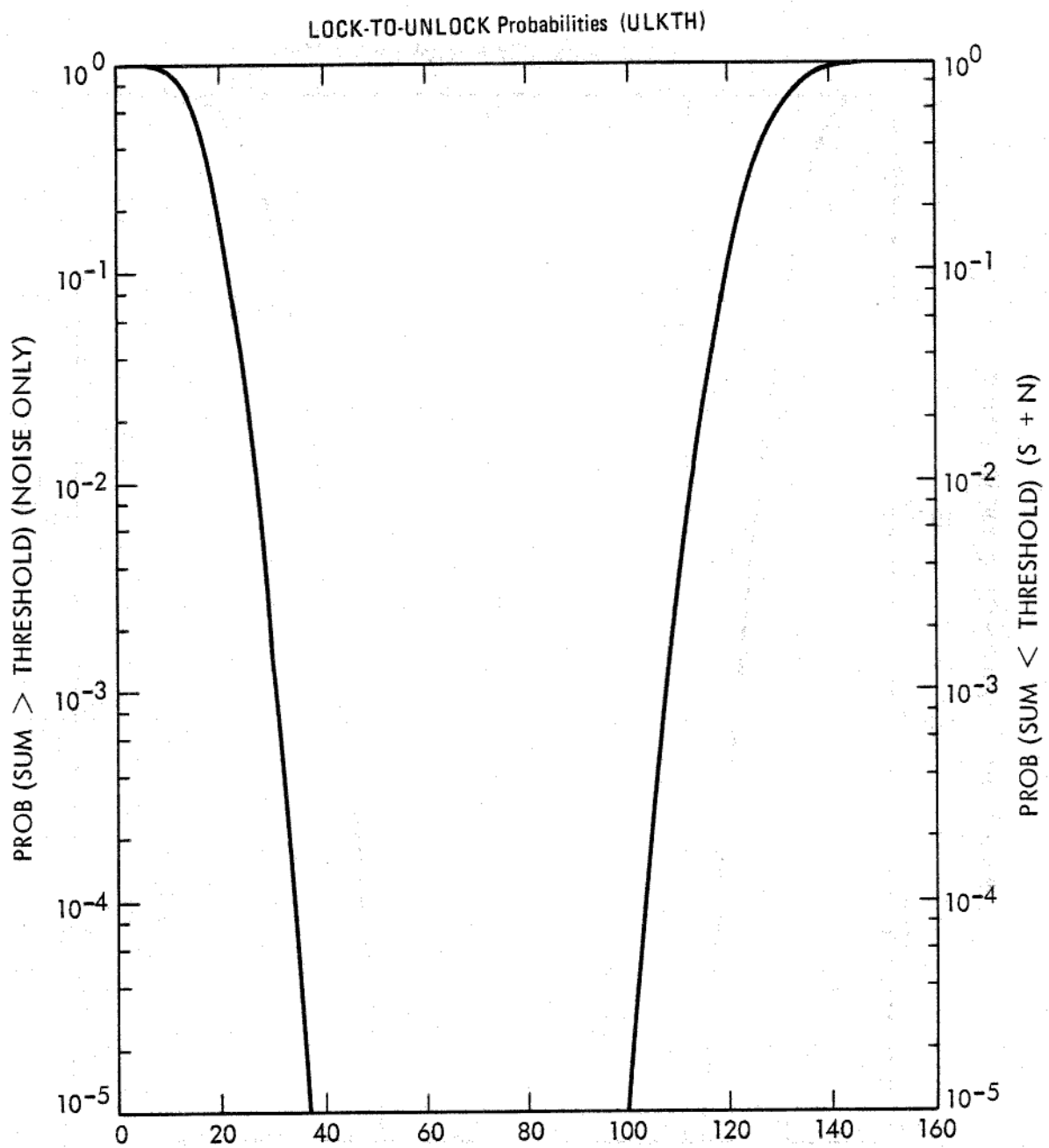


Figure 7-11. Threshold Value, $r = 4$, Rate = 250 bps

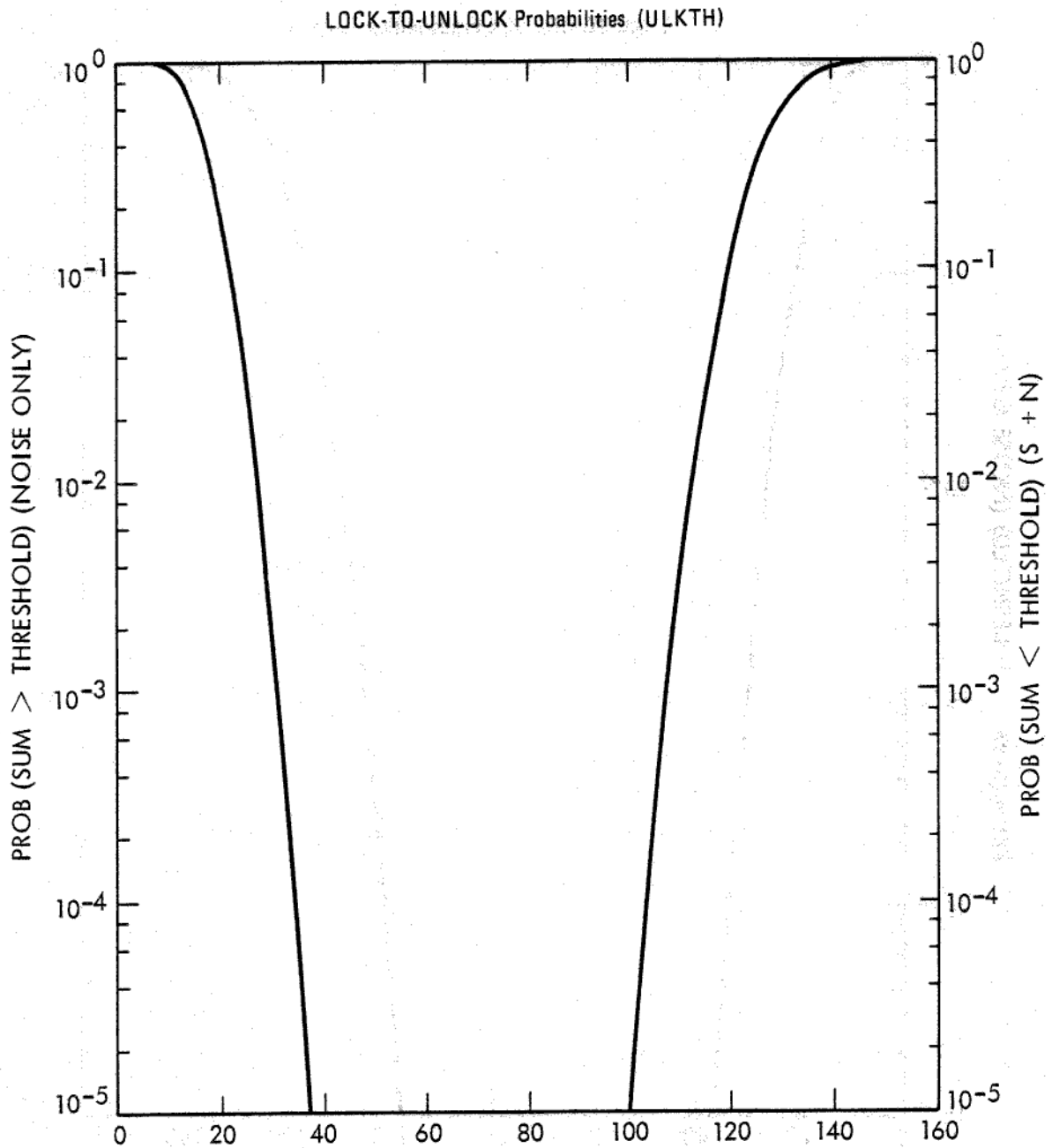


Figure 7-12. Threshold Value, $r = 5$, Rate = 125 bps

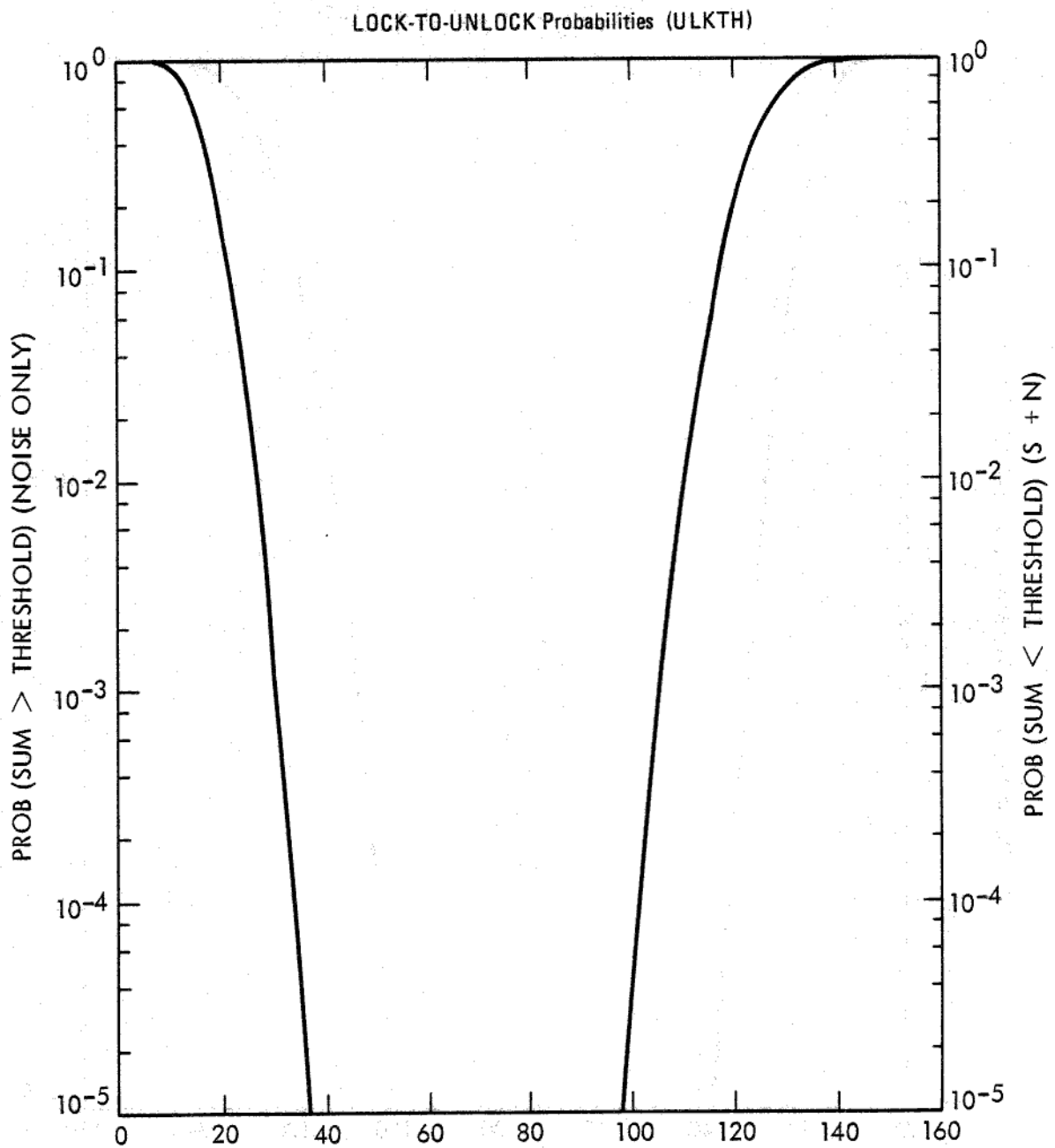


Figure 7-13. Threshold Value, $r = 6$, Rate = 62.5 bps

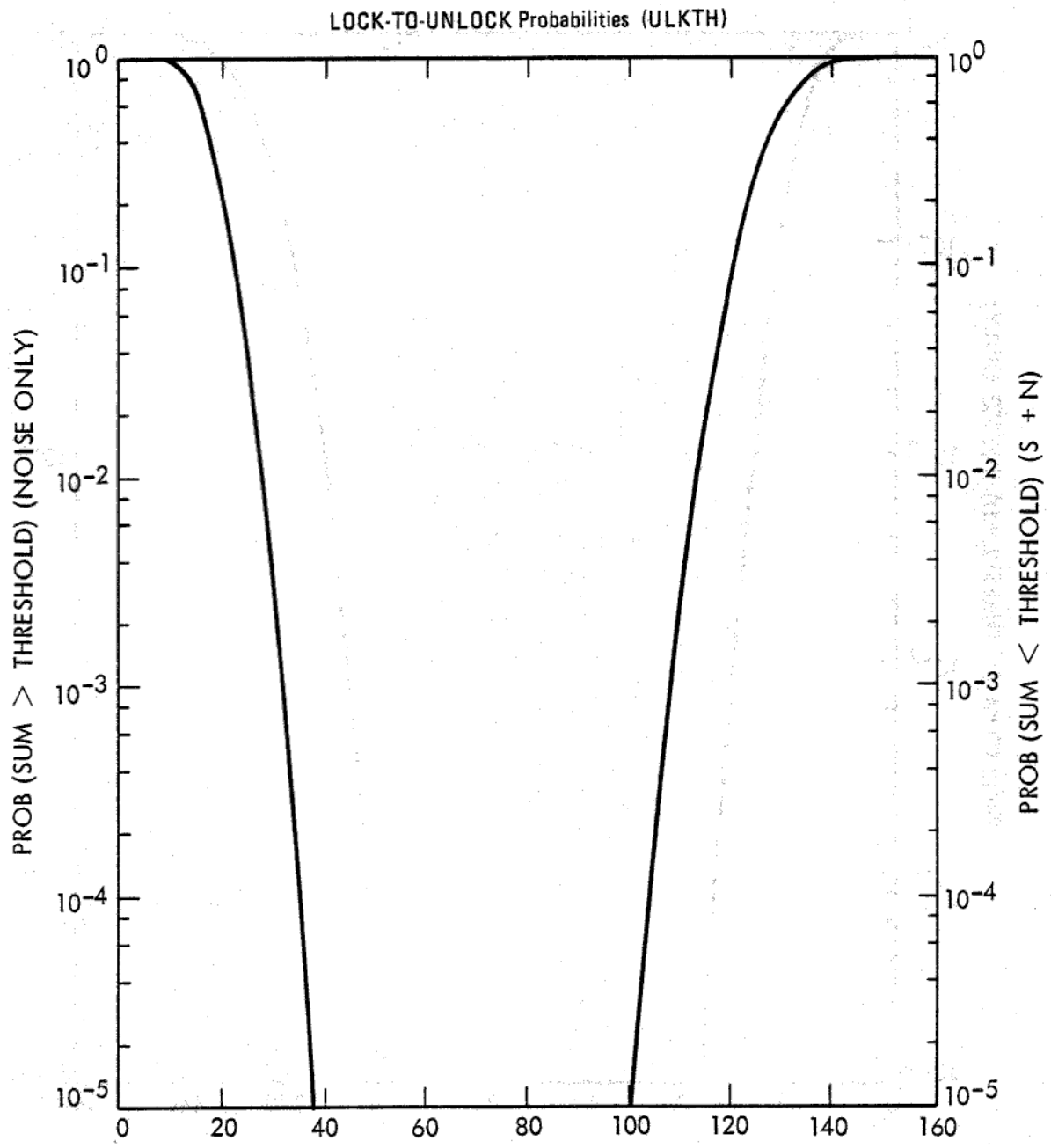


Figure 7-14. Threshold Value, $r = 7$, Rate = 31.25 bps

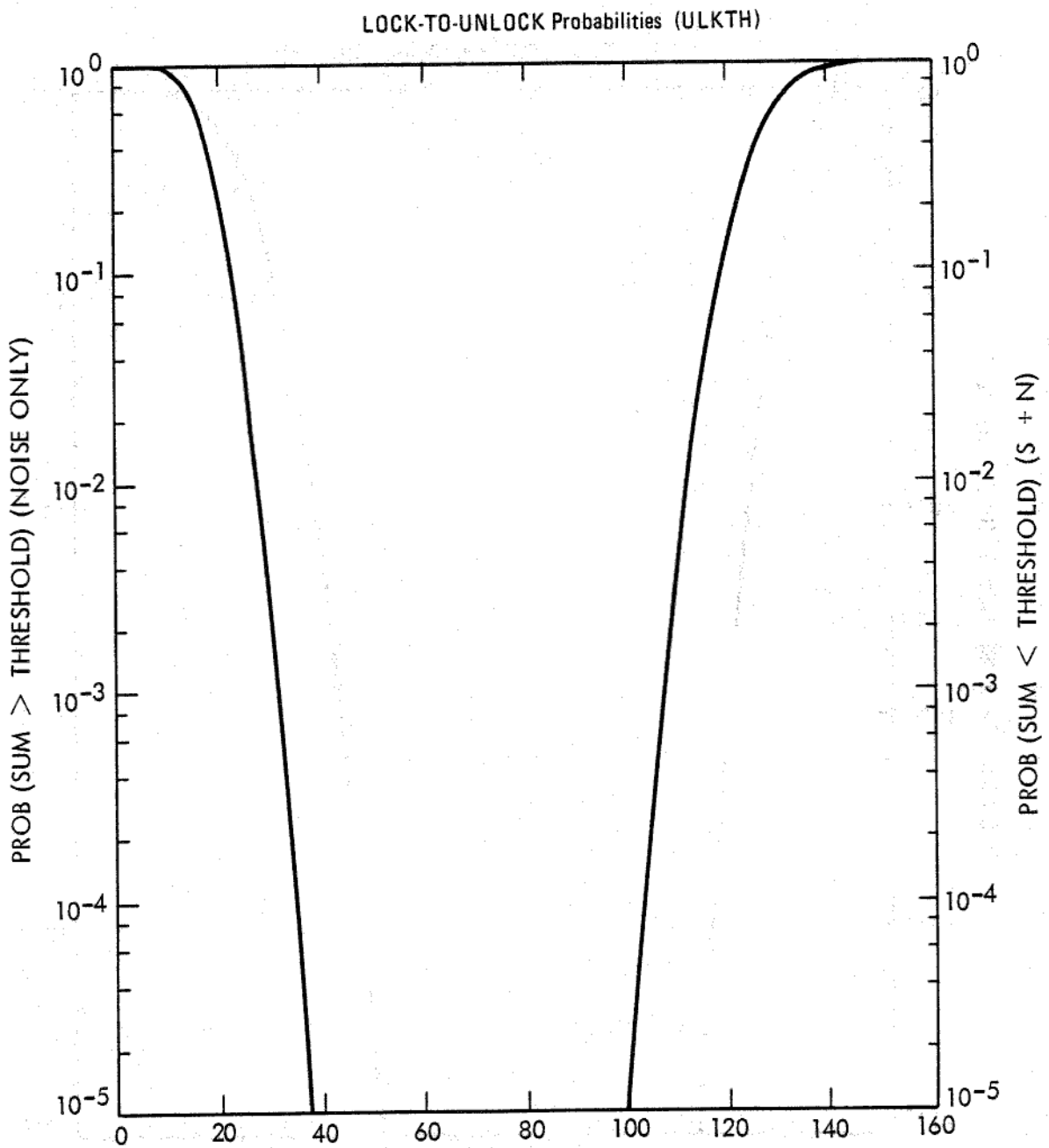


Figure 7-15. Threshold Value, $r = 8$, Rate = 15.625 bps

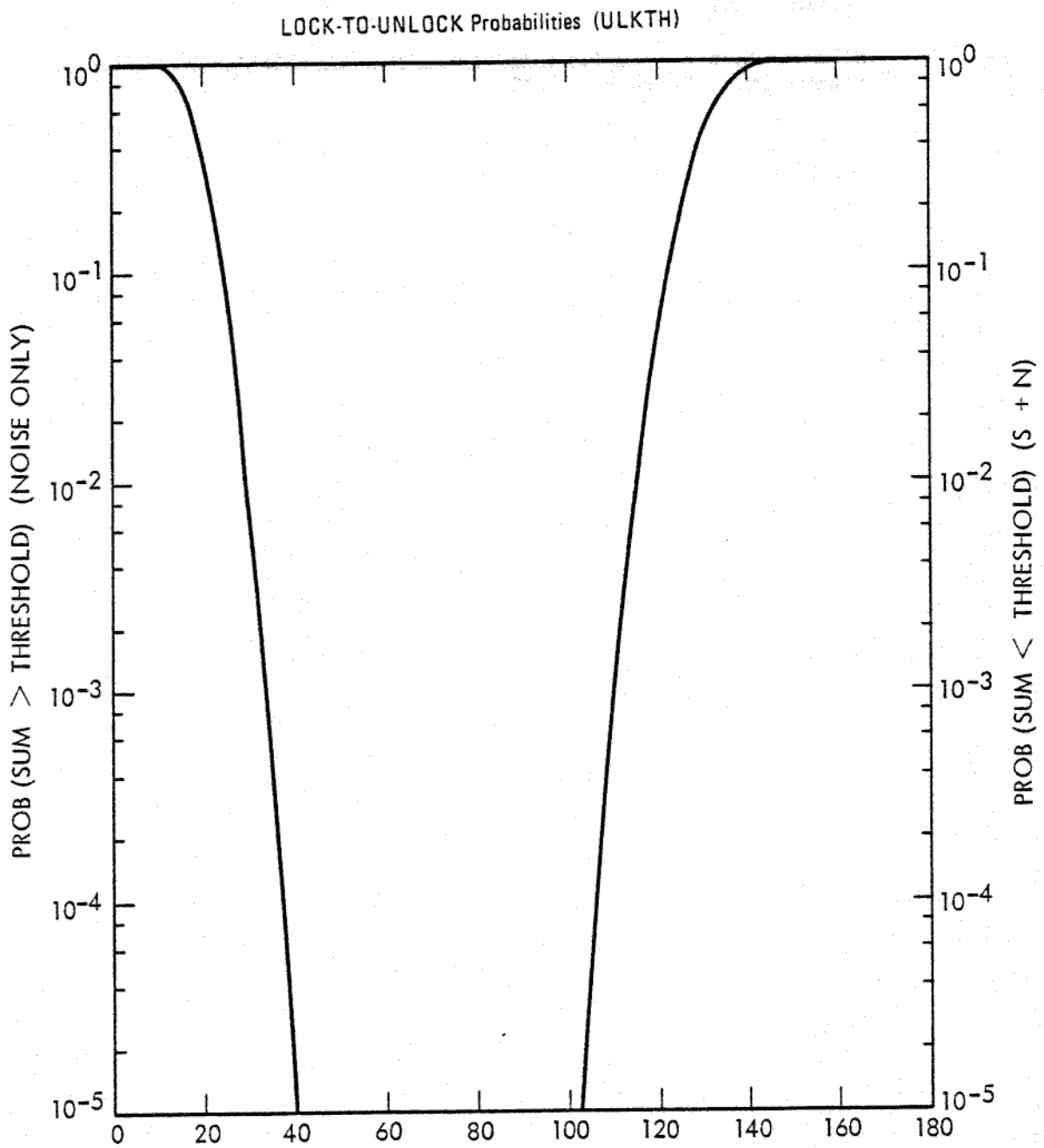


Figure 7-16. Threshold Value, $r = 9$, Rate = 7.8125 bps

7.5 REFERENCES

- 7-1 Albrecht, V.R., Design Requirements NASA Deep Space Command Detector Unit, DM514438, Rev. A, August 1986.
- 7-2 NASA Standard Command Detector Unit Engineering Report, Motorola, February 1977.
- 7-3 Miller, Alan R., Fortran Programs For Scientists and Engineers, Sybex, Inc., Berkeley, 1982.

7A PROBABILITY THAT AN EVENT OCCURS AT LEAST TWO SUCCESSIVE TIMES
 DURING N TRIALS, GIVEN THE PROBABILITY OF ITS OCCURRENCE IN A
 SINGLE TRIAL

We wish to find the probability of an event occurring at least twice in succession in N trials. We define the following:

$$P = \text{Probability of event occurrence in a single trial} \quad (7A-1)$$

$$(1-P) = \text{Probability of no event occurrence in a single trial} \quad (7A-2)$$

$$P(2,N) = \text{Probability of at least two successive occurrences during N independent trials} \quad (7A-3)$$

For example, for N equal to 4 we have:

$$\begin{aligned} P(2,4) &= \text{Prob (2 successive in 4 trials)} \\ &\quad + \text{Prob (3 successive in 4 trials)} \\ &\quad + \text{Prob (4 successive in 4 trials)} \\ &= 3p^2(1-p)^2 + 2p^3(1-p)p + 2p^3(1-p) + p^4 \end{aligned} \quad (7A-4)$$

$$= 3p^2 - 6p^3 + 3p^4 + 4p^3 - 4p^4 + p^4$$

$$P(2,4) = 3p^2 - 2p^3 \quad (7A-5)$$

If we define $A_{i,N}$ as the coefficient of the i th term for N trials we can state $P(2,N)$ as:

$$P(2,N) = \sum_{i=2}^N A_{i,N} p^i \quad (7A-6)$$

Thus, calculating the $A_{i,N}$ becomes just an exercise in combinatorics to be done on a computer. The results are shown in Table 7A-1.

Table 7A-1. Values of $A_{i,N}$

i	2	3	4	5	6	7	8	9	10	11	12
N											
2	1	0									
3	2	-1	0								
4	3	-2	0	0							
5	4	-3	-1	1	0						
6	5	-4	-3	4	-1	0					
7	6	-5	-6	9	-3	0	0				
8	7	-6	-10	16	-5	-2	1	0			
9	8	-7	-15	25	-6	-9	6	-1	0		
10	9	-8	-21	36	-5	-24	18	-4	0	0	
11	10	-9	-28	49	-1	-50	39	-7	-3	1	0
12	11	-10	-36	64	7	-90	70	-4	-18	8	-1

We now wish to find the probability of the single event, P , given $P(1,N)$. Since we have the equation for $P(2,N)$ (equation 7A-6 and Table 7A-1), we just need to find the root of the equation for the given value of $P(1,N)$.

Using a root-finding algorithm based on Newton's method [7-3], a computer was programmed to find the roots for $P(2,N)$ equal to 0.99 and 0.9999, with N in the range of 2 to 12. The results are in Table 7A-2. Note that P is just the probability of z_n being greater than some k ; i.e., equation 7.2-33.

Table 7A-2. Required Single Event Probabilities

P		
N	$P(2,N) = 0.99$	$P(2,N) = 0.9999$
2	0.995	0.999
3	0.990	0.999
4	0.941	0.994
5	0.910	0.990
6	0.861	0.971
7	0.823	0.957
8	0.785	0.933
9	0.751	0.913
10	0.721	0.890
11	0.693	0.869
12	0.668	0.848

SECTION 8

SINGLE EVENT UPSETS AND SNR TELEMETRY WORD

This section provides data on two subjects that have been mentioned in other sections: the CDU's ability to handle Single Event Upsets (SEU's) and the SNR telemetry word that the CDU outputs for inclusion into the spacecraft engineering telemetry data.

8.1 SINGLE EVENT UPSETS

An SEU occurs when a cosmic ray hits the 80C86 or any SEU sensitive device with enough energy to change the state of a bit. This bit flip is only temporary and does not affect the microprocessor's operation after the fact. The susceptibility of a microprocessor to SEU's depends on how the chip is manufactured and how it is packaged. As of this time, the CDU project does not have sufficient information to do a detailed analysis of the effects of SEU's on the CDU. The following results are based on an assumption that the CDU will "see" 2.0 SEU's per day.

There are three things that can occur when a bit flip occurs:

- (1) A bit in the program counter is flipped, causing the microprocessor to step through the code incorrectly or to jump to a nonexistent memory location.
- (2) A bit in one of the registers or one of the memory locations is flipped, causing one of the loop parameters to be altered.
- (3) The output data bit is flipped.

Each problem has a solution that is presented below:

- (1) A "proper operation" pulse is sent to an SEU recovery circuit, consisting of a retriggerable timer, every 125 microseconds. If this pulse is not received and the timer is allowed to timeout, the CDU undergoes an SEU reset, which resets the CPU. This causes the CDU to lose lock.
- (2) Depending on which bit in the loop parameter is flipped (the most significant, the least significant, or one in between), the error may cause the output data bit to flip, the CDU to lose lock, or nothing. As will be shown below, if the output data bit is affected, the error can be corrected and a loss of lock can be recovered from.
- (3) Assuming that we have a bit error rate of 1.0×10^{-5} out of the CDU and an uplink at a rate of 31.25 bits per second for 2 hours per day, we would average 2.25 errors per day (without SEU problems). Assuming that all SEU's occur in this 2 hour period and that each SEU causes an output bit flip, we

would now average 4.25 errors per day, or a bit error rate out of the CDU of 1.9×10^{-5} . Also assuming that the spacecraft will use the CSDS recommended uplink code (as Galileo and Magellan do), [8-1] shows that the bit error rate after error correction will be around 10^{-10} , which will prevent the SEU induced error from affecting the data stream error rate.

So in the worst case, the CDU will lose lock an additional 2 times per day as a result of SEUs.

8.2 SNR TELEMETRY WORD

The SNR telemetry word allows the analysts on Earth to generate an estimate of the ST/N_0 that the CDU is receiving. This section will describe how to use the SNR telemetry word to generate an estimate of the ST/N_0 .

The network that the input signal plus noise must travel is shown in Figure 8-1. Notice that this is just the value that is used by the AGC Loop to generate an error signal and by the Lock Detector to compare against the threshold numbers.

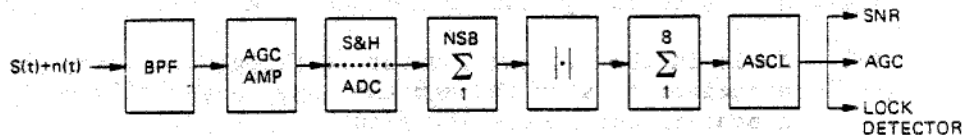


Figure 8-1. Input Network

The input modulated signal is $s(t)$ and the additive, white gaussian noise is $n(t)$. The signal has peak amplitude A , thus its power, S , is equal to:

$$S = \frac{A^2}{2} \quad (8.2-1)$$

This signal plus noise is bandlimited by the predetection bandpass filter, whose noise bandwidth is B_N . Since we have white gaussian noise, we get:

$$\sigma^2 = N_0 B_N \quad (8.2-2)$$

or

$$N_0 = \frac{\sigma^2}{B_N} \quad (8.2-3)$$

The ADC samples at the peak of the subcarrier sinewave. This means that the ADC has an effective voltage gain of $\sqrt{2}$; thus, the noise power is increased by a factor of 2. Since the noise is zero mean, this means that the output of the ADC is a gaussian random variable with mean A and variance $2\sigma^2$. We will represent this with the notation $N(A, 2\sigma^2)$. We are assuming here that we have sufficiently high ST/N_0 to avoid saturating the ADC, which would modify the gaussian distribution.

We then sum over the number of samples per bit, 2^{r+1} . Since we are summing independent gaussian random variables, the output of this operation is $N(2^{r+1}A, 2^{r+2}\sigma^2)$.

Next, we take the absolute value of the sum. If we are at sufficiently high ST/N_0 (say 8 dB and higher), the tail of the distribution does not contribute much to the mean and variance. So we will assume that the random variable output of the absolute value is the same as the input.

The absolute value is followed by a sum over eight bits. Again we are summing independent gaussians, so the result is $N(2^{r+4}A, 2^{r+5}\sigma^2)$.

Finally, to get the telemetry word, we scale the output of the sum over eight bits by the data rate dependent scaler ASCL, which gives us the distribution of the SNR telemetry word: $N((ASCL)2^{r+4}A, (ASCL)^2 2^{r+5}\sigma^2)$.

To generate the estimate of ST/N_0 , we observe the statistics of the SNR telemetry word. Working with the sample mean, μ' , and the sample variance, σ'^2 , we get the following:

$$\mu' = (ASCL) 2^{r+4} A \quad (8.2-4)$$

$$\sigma'^2 = (ASCL)^2 2^{r+5} \sigma^2 \quad (8.2-5)$$

Which gives us:

$$A^2 = (ASCL)^{-2} 2^{-2r-8} \mu'^2 \quad (8.2-6)$$

$$\sigma^2 = (ASCL)^{-2} 2^{-r-5} \sigma'^2 \quad (8.2-7)$$

Also recall that:

$$T = \frac{2^{r-1}}{2000} \quad (8.2-8)$$

Using equations 8.2-1 and 8.2-3, along with equation 8.2-8, we get the following for ST/N_0 :

$$\begin{aligned} ST/N_0 &= (A^2/2)T/N_0 \\ &= \frac{A^2 2^{r-2} B_N}{2000} \end{aligned} \quad (8.2-9)$$

Utilizing our results in equations 8.2-6 and 8.2-7, we get:

$$\begin{aligned} ST/N_0 &= \frac{(ASCL)^{-2} 2^{-2r-8} \mu'^2 2^{r-2} B_N}{(2000(ASCL)^{-2} 2^{-r-5} \sigma'^2)} \\ &= \left(\frac{B_N}{2000} \right) 2^{-5} \left(\frac{\mu'^2}{\sigma'^2} \right) \end{aligned} \quad (8.2-10)$$

$$ST/N_0 = \frac{B_N}{64000} \left(\frac{\mu'}{\sigma'} \right)^2$$

There is just one thing that remains to be done. Since we are dealing with a statistical measurement, we want to provide a confidence interval for this measurement. Since the AGC keeps the observed mean very close to the expected mean, we are only interested in a confidence interval for the variance of our measurement. From [8-2], we know that for a gaussian random variable, with n samples, the estimator of the variance, σ'^2 , has a chi-square distribution with $n-1$ degrees of freedom. For the estimate of ST/N_0 , n is quite large. From [8-3], we know that for large n , the probability that

a chi-square variable is less than C is approximately equal to the probability that a N(0,1) (gaussian, mean zero, variance one) variable is less than X, where X is:

$$X = \frac{(C - n)}{\sqrt{2n}} \quad (8.2-11)$$

We want a 95% confidence interval for the true variance, VAR. This is equivalent to saying:

$$\text{Prob}(n/b \sigma'^2 < \text{VAR} < n/a \sigma'^2) = 0.95 \quad (8.2-12)$$

where a and b are defined as follows:

$$\text{Prob}(C < a) = 0.025 \quad (8.2-13)$$

$$\text{Prob}(C > b) = 0.025 \quad (8.2-14)$$

Using the relationship between the chi-square random variable, C, and the N(0,1) random variable, X, (equation 8.2-11), we can say:

$$\text{Prob}(C < a) = \text{Prob}(X < q) = 0.025 \quad (8.2-15)$$

$$\text{Prob}(C > b) = \text{Prob}(X > r) = 0.025 \quad (8.2-16)$$

Or, since X is a normal random variable:

$$q = -1.960 \quad (8.2-17)$$

$$r = 1.960 \quad (8.2-18)$$

Using equation 8.2-11, we get:

$$C = n + \sqrt{2n} X \quad (8.2-19)$$

So:

$$a = n + \sqrt{2n} (-1.96) \quad (8.2-20)$$

$$b = n + \sqrt{2n} (1.96) \quad (8.2-21)$$

So, using equations 8.2-20 and 8.2-21 in equation 8.2-12 provides the confidence interval limits on the variance. To get the confidence interval on ST/N_0 , just substitute these limits into equation 8.2-10 in place of σ^2 .

8.3 REFERENCES

- 8-1 J. Berner, R. McEliece, E. Posner, "Error and Erasure Probabilities for Galileo Uplink Code," TDA Progress Report 42-83, July-September, 1985.
- 8-2 Hogg, Robert V., and Craig, Allen T., Introduction to Mathematical Statistics, Fourth Edition, Macmillan Publishing Co., Inc., New York, 1978.
- 8-3 Abramowitz, Milton, and Stegun, Irene A., Handbook of Mathematical Functions, Dover Publications, Inc., New York, 1964.

SECTION 9

HARDWARE

The hardware of the CDU is made up of two distinct sections, the analog section and the digital section. The 80C86 microprocessor is the heart of the system; it controls both the analog and digital sections. All interfaces between the two sections are buffered and, if needed, latched. Figure 9-1 provides a block diagram of the hardware implementation.

9.1 ANALOG SECTION

The analog section consists of two parts, the automatic gain control amplifier (AGC amp) and the sample-and-hold/ADC. These sections receive from the digital section the AGC control word, the in-lock/out-of-lock status signal, the ADC start convert pulse, and the ADC clock; the analog section sends to the digital section the digitized samples of the input signal, which is received from the NXT. A description of the two parts of the analog section follow.

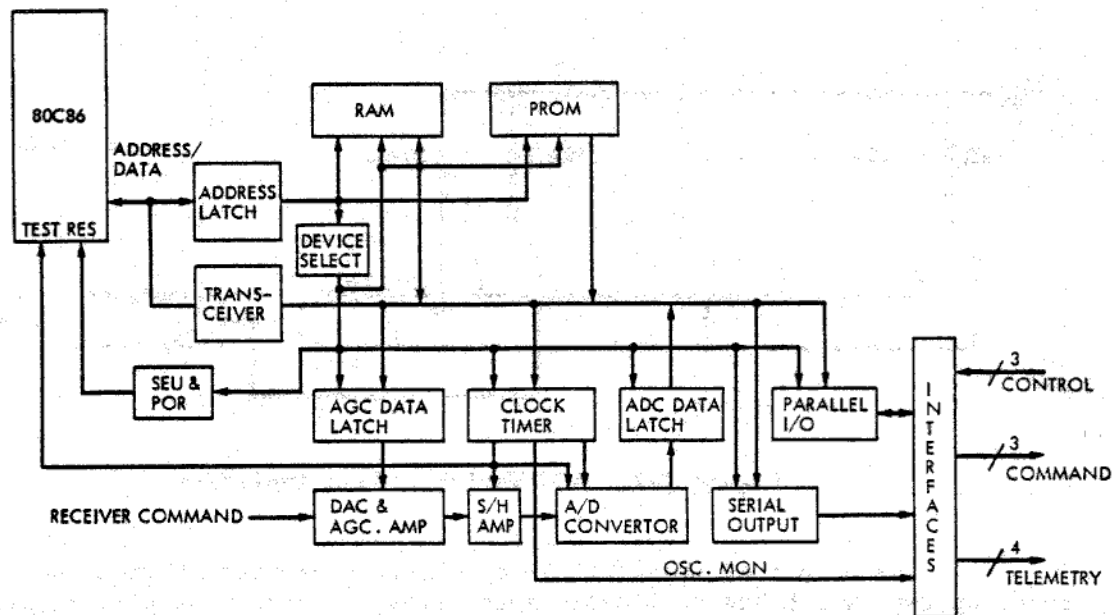


Figure 9-1. NASA Deep Space CDU Breadboard Block Diagram

9.1.1 Automatic Gain Control Amplifier

The AGC amp is a variable gain amplifier that is used to provide a constant amplitude signal to the sample-and-hold circuit. The AGC circuit consists of two sections, a 9-bit DAC and a dual gain amplifier. Figure 9-2 provides a diagram of the AGC amplifier circuit.

The AGC amplifier gain is controlled by a 10 bit word (the AGC control word) that is generated by the AGC software algorithm. The nine MSB's are used to control nine switches in an R-2R resistance ladder (in our case, R is 10 K). This ladder is used, along with an op amp which converts the current through the ladder into a voltage, as a digital-to-analog converter (DAC); this discrete component configuration is used because there is not a flight approved rad hard DAC available. The signal received from the NXT is applied to the input of the ladder network, which splits the resulting current between Output 1 and Output 2. The bits of the AGC control word control whether the current goes to Output 1 or Output 2. At each branch of the network, the current that flows to the next branch is one-half of the current that entered the branch. The current from DAC Output 1 is the input to the current to voltage converter, and the current from Output 2 is shunted to ground. The gain of this circuit is discussed in Section 3.2 and the algorithm that controls it is described in Section 5.

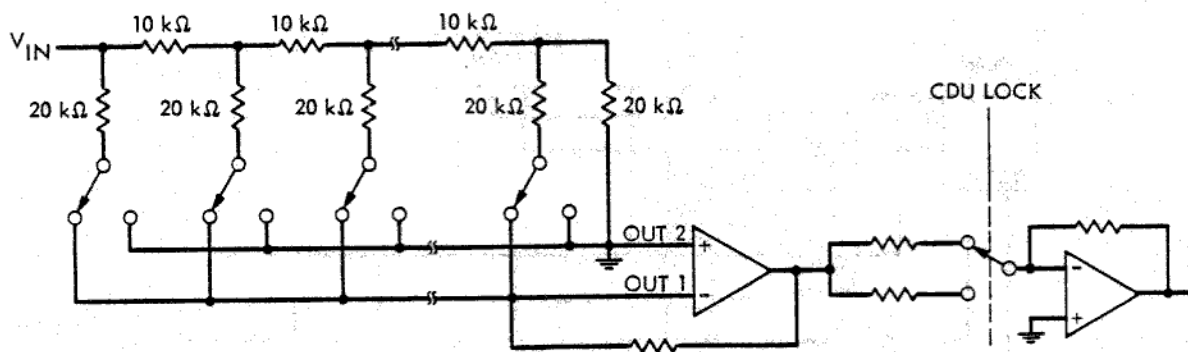


Figure 9-2. AGC Variable Gain Amplifier

The output of the current-to-voltage converter is then fed into a second op amp, which has dual selectable gains. The gain of this amplifier is selected by the status of the in-lock/out-of-lock signal. When the CDU is out-of-lock, it requires the maximum possible gain to insure acquisition of the uplink signal. When it is in-lock, the maximum gain is not desired, since it saturates the ADC and causes accumulator overflow (discussed in Section 5). Refer to Section 7 for the discussion of the in-lock/out-of-lock gain numbers.

9.1.2 Sample-and-Hold/ADC

The sample-and-hold/ADC circuit, consists of a track and hold (T/H) amplifier, followed by the ADC, as shown in Figure 9-3.

This circuitry is controlled by the ADC clock and the start convert pulse, which are provided by the digital section. The start convert pulse and the 0.9984 MHz ADC clock are asynchronous; thus the pulse width of the start convert pulse must be wider than the period of the ADC clock to assure that the pulse will be active during a low to high transition of the ADC clock. The start convert pulse is generated by counting down three cycles of the peripheral clock, a 2.946 MHz clock, which results in a pulse that is 1.2 microseconds wide, providing us a 20% timing margin. The start convert pulse is also used to place the track and hold amplifier in the hold state through a set/reset flip-flop.

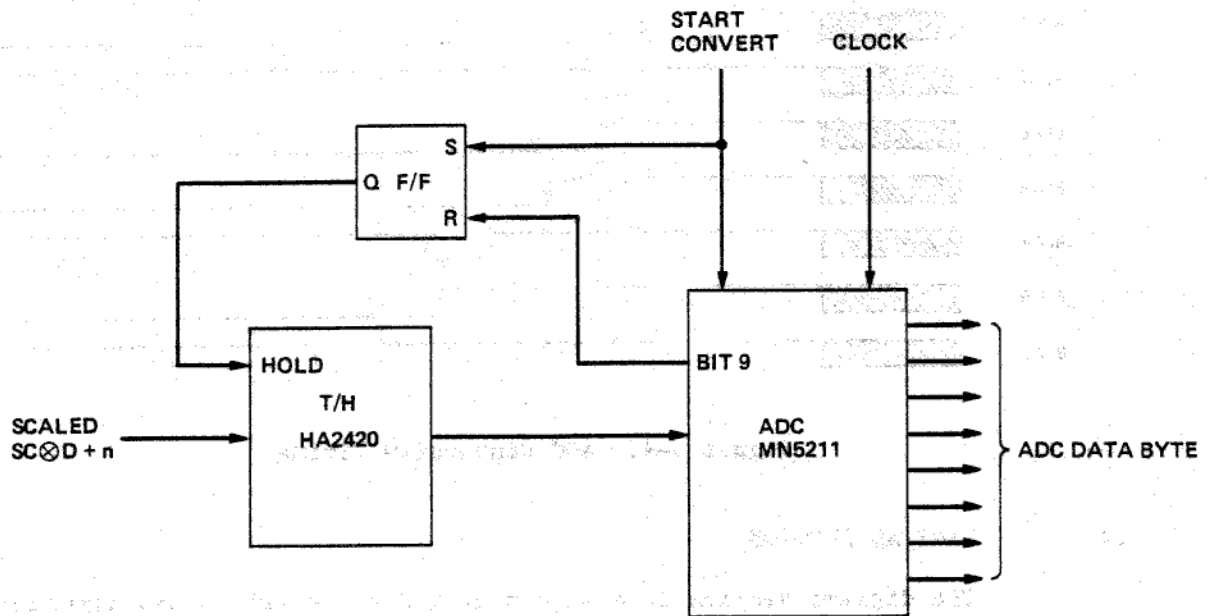


Figure 9-3. Track/Hold Amplifier and ADC

When the ADC starts converting the ninth bit, the bit 9 output has a high to low transition. Thus, when the ADC has converted the eighth bit, this high to low transition is used to reset the flip-flop, which allows the T/H amplifier to track the input again. The eight bit converted sample is then sent to the digital section for processing by the CDU software. A timing diagram for the ADC is provided in Figure 9-4.

Considering all of the delays in the convert command circuitry, the maximum convert time is 9.2 microseconds (1.2 microsecond pulse plus eight cycles of the ADC clock); the T/H amplifier has an acquisition time of 5 microseconds to track the signal once it is released from the hold state.

Thus we require 14.2 microseconds, worst case, for each sample; the shortest possible time between the convert pulses is 15.625 microseconds, which gives this process margin of 1.425 microseconds.

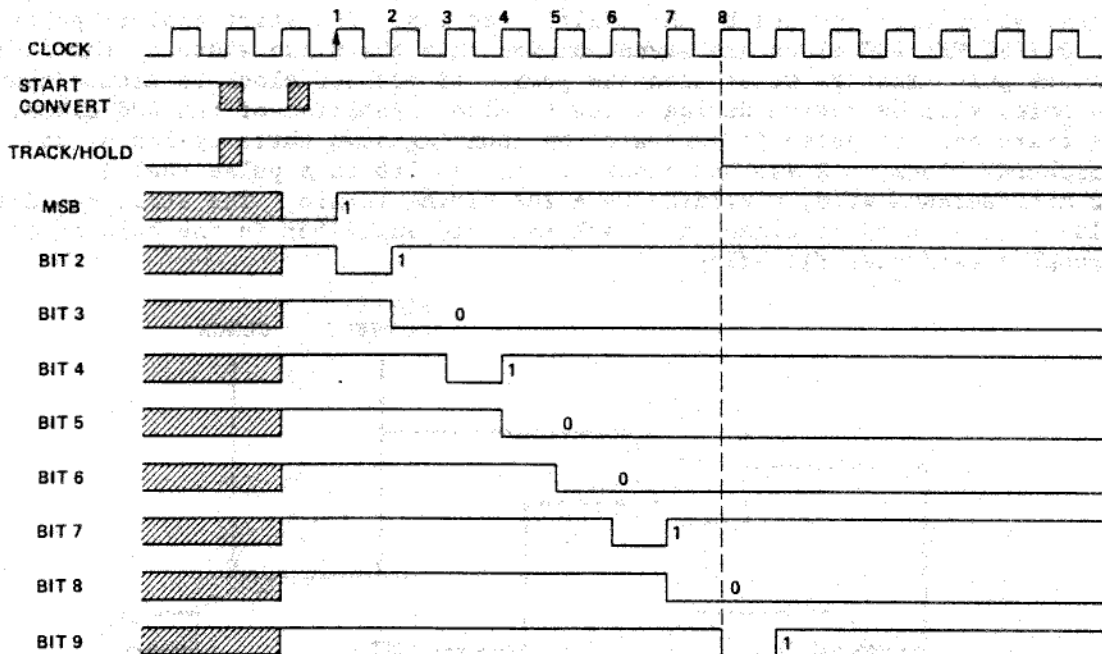


Figure 9-4. ADC Conversion Timing

9.2 DIGITAL SECTION

The digital section is designed around the Harris 80C86 sixteen bit microprocessor and associated family of peripheral chips. It has 2 K bytes of RAM for temporary storage of program variables and 4 K bytes of ROM for storage of the operating system software. A programmable timer is used along with a crystal controlled clock to generate all of the needed clock signals. SEU protection circuitry is provided as discussed in Section 9.2.2. All of the microprocessor I/O is performed through memory mapping, and is latched, using the device select. A block diagram of the digital section is provided in Figure 9-5.

9.2.1 System Timing

The CDU basic clock is generated by a crystal controlled 82C85 clock controller/generator. A crystal frequency of 14.976 MHz was selected to provide a frequency (4.992 MHz) that is as close to the 80C86's maximum clock

rate of 5 MHz as possible while providing a clock period that is an integer multiple of the subcarrier period so that it can be used for generating exact timing between the error and data samples.

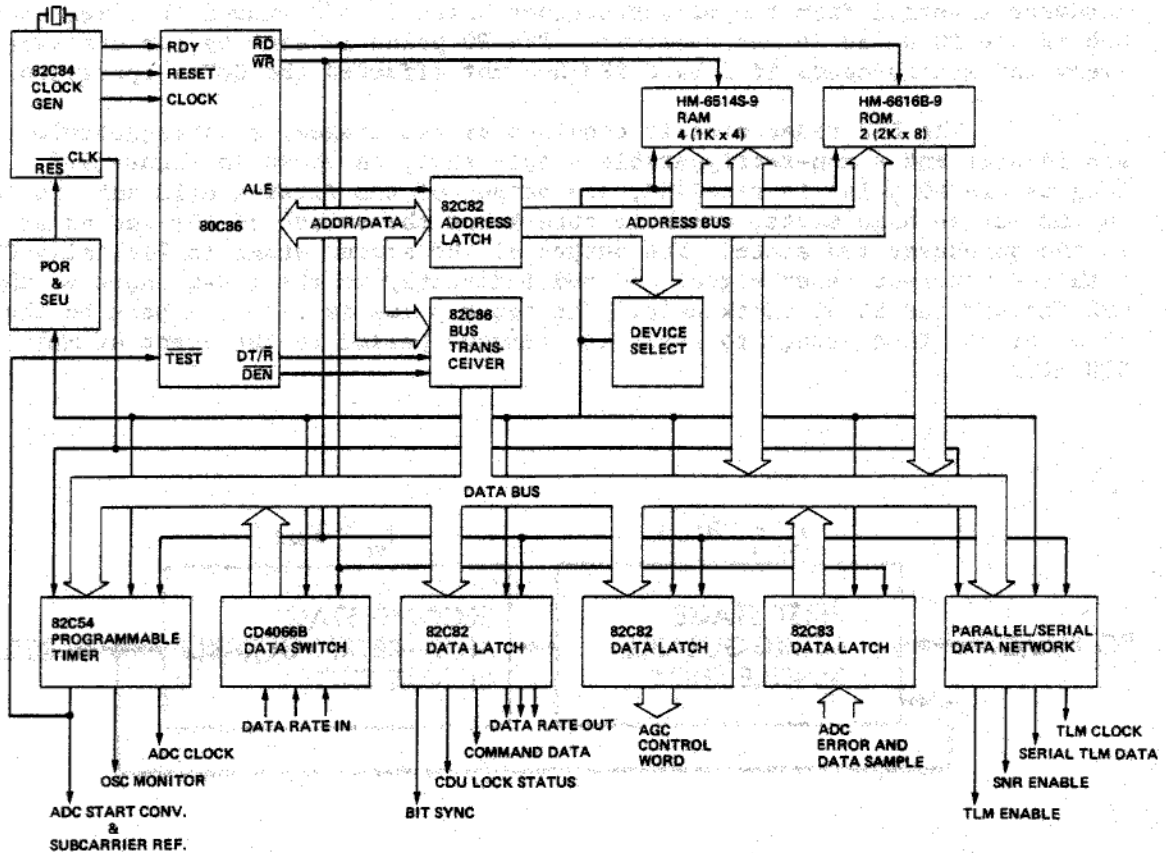


Figure 9-5. Digital Block Diagram

9.2.2 Single Event Upsets

Although the 80C86 is radiation hardened to 100 K rads, it is still susceptible to single event upsets. The CDU design provides the ability to detect and recover from single event upsets (SEUs).

There are two types of SEUs, soft and hard. Soft SEUs are non-detectable occurrences, such as flag bits being changed, data values being changed, instructions being changed from one valid instruction to another, and address bits being changed from one valid address to another. Hard SEUs are detectable and could appear as an instruction being changed to a nonexistent instruction, or an address being changed to an illegal or nonexistent address; either of these could cause an unprotected system to fail. Soft SEUs might cause a loss of lock, which is recoverable; however, a hard SEU could result in an SEU induced processor halt or an endless loop state, which would require external intervention to correct. To eliminate the need for this external intervention, the CDU design provides an SEU protection circuit.

9.2.2.1 Single Event Upset Protection Circuit

The SEU protection scheme is made up of two parts: software generated proper operation (PO) pulse when there is no detected SEU and hardware external from the microprocessor which is SEU immune that resets the CPU if the PO pulse is not received. The PO pulse is sent by the software every 125 microseconds if a hard SEU has not affected the CDU's operation.

The SEU reset circuit consists of two stages, a retriggerable single shot and a non-retriggerable single shot, as shown in Figure 9-6. As long as the PO pulse is received, the output of the first single shot is kept in the active high state, which in turn keeps the output of the second stage in the quiescent low state. The output of the second stage is logically OR'ed with the power-on reset signal and fed indirectly to the reset input of the CPU through the 82C85 clock chip. The reset pulse is also fed back to the input of the first stage to reset its timeout period in the event of multiple SEU hits.

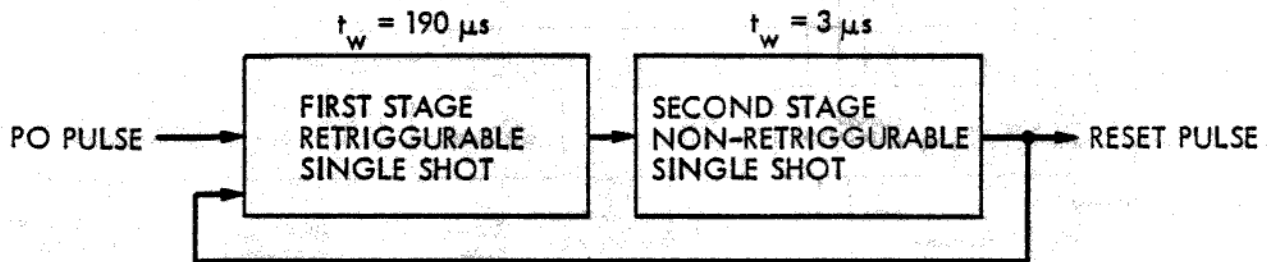


Figure 9-6. SEU Reset Circuit

Figure 9-7 provides a timing diagram of the protection circuitry in the event of an SEU. As can be seen, if the next PO pulse is not received 190 microseconds after the last PO pulse, the output of the first stage will fall to the low state, which triggers the second stage's 3 microsecond reset pulse. This reset pulse resets the CPU and the first stage. The reset causes the CPU to reinitialize itself, restart the PO pulses, and begin to reacquire the uplink signal. If an additional SEU were to occur, the procedure would repeat itself.

9.2.2.2 Single Event Upset Timing Considerations

In order for the 80C86 microprocessor to recognize an external reset, its reset pin must be held active for at least 4 clock cycles, which is equal to 800 nanoseconds for a 5 MHz clock. The reset pulse of the recovery circuit is 3 microseconds wide, or 3.75 times as large as needed, providing

ample timing margin. Worst-case circuit analysis (WCCA) [9-1] shows that the timing for the first stage of the recovery circuit may vary by $\pm 27.5\%$. This is shown in Figure 9-8.

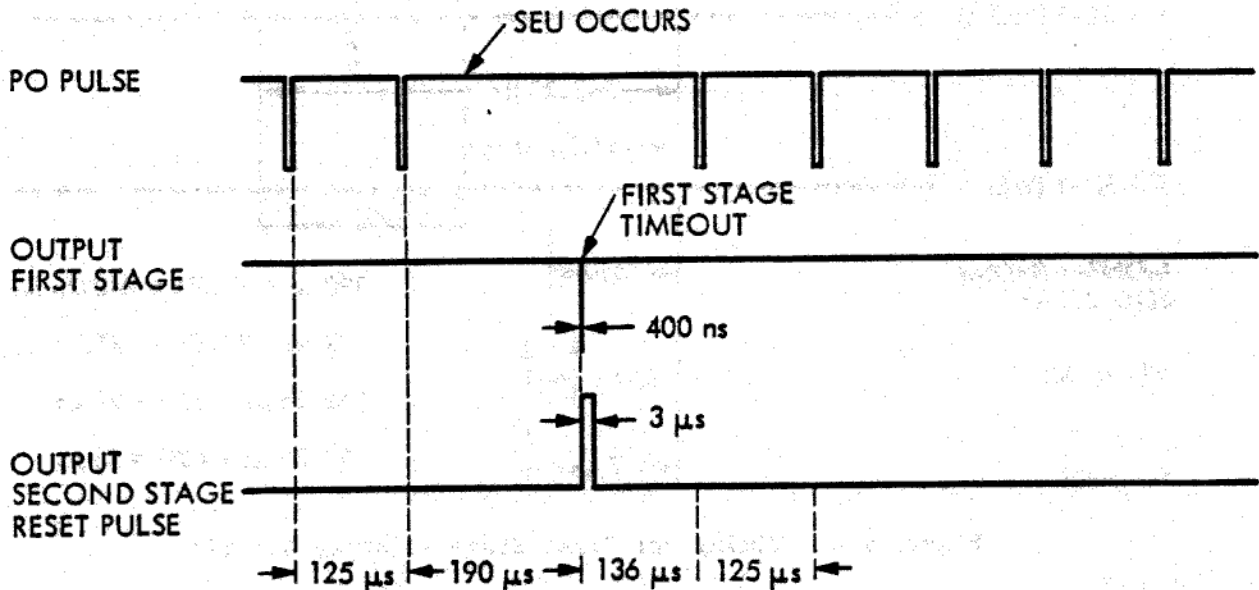


Figure 9-7. Protection Circuit Timing Diagram

As can be seen, the 190 microsecond pulse width can vary from 137.7 microseconds to 242.25 microseconds. Comparing these WCCA times with the 125 microsecond PO pulse, we see that we have a drift margin of 12.75 microseconds for the 137.75 microsecond pulse, and a drift margin of 7.75 microseconds for the 242.25 microsecond pulse. Also, there is a further timing requirement that the PO pulses be separated by at least 40% of the first stage's pulse width (242.25 microseconds in the worst case), or 97 microseconds. Thus, we get the following constraint on the period of the PO pulse: $97 \text{ microseconds} < PO < 137.75 \text{ microseconds}$. The PO pulse itself may also vary, once per bit time, due to the timing bump applied by the Subcarrier Tracking Loop (see Section 4). In the case of a maximum bump, the PO pulse period would be 133 microseconds; for a minimum bump, the period would be 118 microseconds. Thus, for all values of subcarrier bumps and for all possible timing variations of the first stage, the PO pulse will prevent the SEU protection circuitry from falsely resetting the CDU. This means that, if the PO pulse does not retrigger the first stage and the CDU is reset, an SEU has indeed occurred.

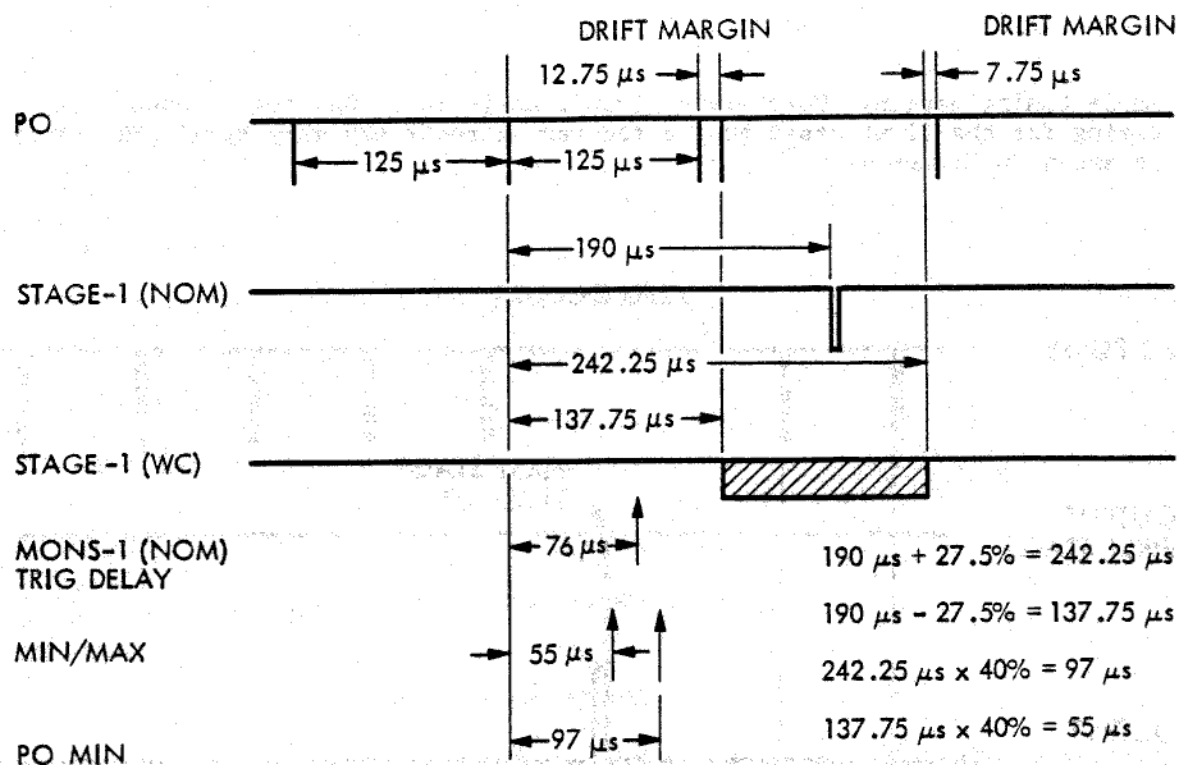


Figure 9-8. Timing for First Stage Recovery Circuit

9.3 REFERENCES

- 9-1 Kleiner, Charles T., NASA Deep Space Command Detector Unit Worst-Case Circuit Analysis, December 1986.
- 9-2 The TTL Data Book for Design Engineers, Texas Instruments, Inc., 1973.
- 9-3 1984 Harris CMOS Digital Data Book, Harris Corporation, 1984.
- 9-4 Rad-Hard/Hi-Rel CIGD Data Book, Harris Corporation, 1987.
- 9-5 1986 Analog Data Book, Harris Corporation, 1986.
- 9-6 RCA CMOS Integrated Circuits, RCA Corporation, 1983.
- 9-7 Albrecht, V.R., Design Requirement NASA Deep Space Command Detector Unit, DM514438, Rev. A, August 21, 1986.
- 9-8 Data Conversion Products 1984, Micro Networks/Unitrode, 1984.
- 9-9 Schematic Diagram, NASA Deep Space Command Detector Unit, 10122813, Rev. A.

SECTION 10

CDU SOFTWARE

It is the software that implements all of the algorithms discussed in Part II of this report. The software is written in 80C86 assembly language and is burned into a PROM. The CDU cannot be reprogrammed in flight.

This section will discuss five aspects of the software design and operation: choice of language, sampling scheme, software sectionalization, software output, and SEU handling. There are also two appendices: Appendix A presents the software flowcharts, and Appendix B provides the assembled code.

10.1 CHOICE OF LANGUAGE

In general, it is preferred that software be written in a higher level language, such as the C programming language, to create a program that is easy to maintain and to transfer from machine to machine. However, there were four reasons why the CDU could not use a higher level language and was forced to use the native 80C86 assembly language. These reasons are:

- (1) Speed of resultant code - Assembly language code is faster by at least a factor of two. This speed is mandatory to complete the necessary computations within the allotted time between samples.
- (2) Size of resultant code - Assembly language programming produces code that requires less storage than equivalent C code.
- (3) Access to hardware devices - There is no provision in the C programming language for access to memory mapped device registers or I/O ports. This access is required for the CDU.
- (4) Synchronization - The C programming language makes no provisions for task or program synchronization with external events. The CDU is required to be able to track the sub-carrier's zero crossings.

Thus, for the above reasons, the CDU was programmed in 80C86 assembly language. This choice provides sufficient CPU cycles in each section of the program to perform the needed computations when executing the longest path.

10.2 SAMPLING SCHEME

To perform correctly, the software must be synchronized with the uplink subcarrier zero crossings. Since it is impossible to exactly calculate the time for the next sample in software (due to branch jumps), the

synchronization is obtained by using the 80C86's WAIT instruction and a programmable interval timer chip.

As is described in Section 4, the Subcarrier Tracking Loop tracks the zero crossings of the sinewave subcarrier. The software achieves this synchronization by adjusting timing intervals between the samples. As can be seen in Figure 10-1, if the software takes the sample at the zero crossing, it must take a sample 15.625 microseconds later and then sample the zero crossing again 109.375 microseconds after that.

The software controls the hardware to do this. The 80C86 has an instruction, the WAIT instruction, that causes operations to be suspended until a signal is asserted on the TEST pin of the 80C86. This signal is generated by the 82C54 programmable interval timer. The timer is programmed to operate as a divide-by-N counter. At the end of each N count, the counter reloads the value of N and starts counting again. Thus, the timer can have the next count loaded into its buffer before it finishes counting, allowing the software to program different sampling periods before executing the WAIT instruction. In other words, when the timer causes the operation suspension to end so that the error sample can be taken, the time to count until the data sample is already in the timer's buffer. So the new time is loaded in and the timer starts counting down.

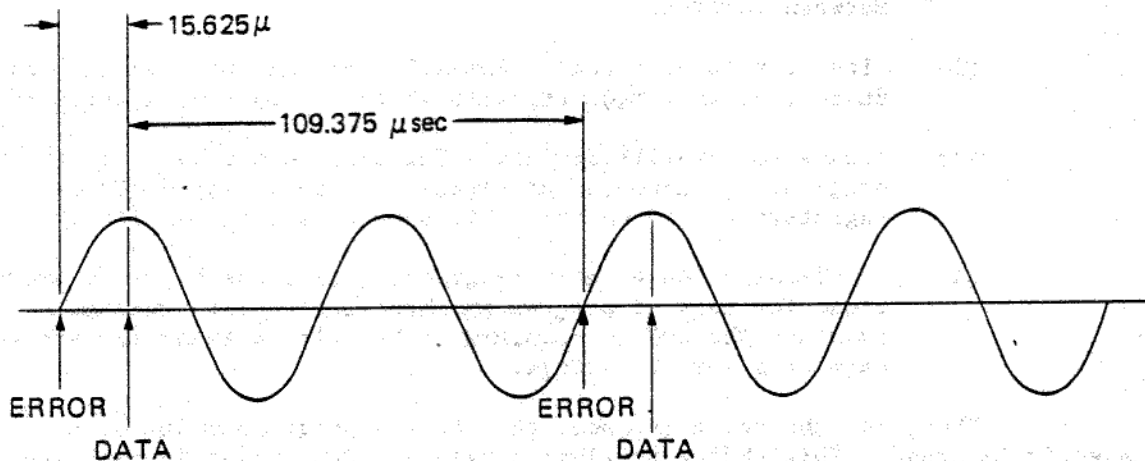


Figure 10-1. Sample Timing

The sequence of operation is as follows: The counter counts down a programmed number of microprocessor clock cycles, issues a pulse, reloads the count down start with the value in its buffer and starts counting down again. When the signal is issued by the counter, the ADC starts converting and the microprocessor "wakes up". Upon "waking up", the program places in the counter's buffer the next value to be counted down from and then does its computations. When it is finished with its computations, the program executes the WAIT instruction again and waits for the next sample command.

Every other subcarrier cycle an error sample and a data sample are taken, with the data sample being taken one-quarter of a subcarrier cycle after the error (subcarrier zero crossing) sample. The data sample is taken 15.625 microseconds after the error sample, and if the CDU is in perfect sync, the next error sample is taken 109.375 microseconds after the data sample. If the CDU is not in perfect sync with the subcarrier, once per bit the time between the data sample and the error sample is adjusted to correct the offset (this is known as the subcarrier bump). This correction, adjusting the number of clock cycles between the samples, converts the phase error of the subcarrier tracking loop into the appropriate number of clock cycles. A lookup table, stored in ROM, provides the conversion values.

10.3 SOFTWARE DESCRIPTION

The CDU's software can be broken down into four distinct sections: initialize (INIT), normal (NORM), midbit (MB), and end-of-bit (EOB). Each of the CDU's algorithms are implemented in one or more of these sections.

Figure 10-2 provides a graphical representation of the algorithmic breakdown. While the flowcharts of the individual algorithms are given in Appendix A, a brief description of the four software sections follows:

10.3.1 INIT

When the CDU is reset, either by a power-on reset, an SEU reset, or a data rate change, the software reinitializes itself by entering the INIT routine. Once inside INIT, the following functions are performed:

First, the hardware counters are initialized. Then the AGC amplifier is set to the maximum gain and the CDU is placed in the out-of-lock state. The error, data, and midphase accumulators are then cleared. The data rate is then input and used to look up the in-lock BSCL, the out-of-lock BSCL, ESCL/ASCL, LKTH, NSB2, and ULKTH from tables stored in ROM and indexed by the data rate. The roundoff factors for the two BSCL values and ESCL/ASCL are calculated next. The number of samples in a quarter bit, NSB4, is calculated from the number of samples in a half bit, NSB2. Finally, any remaining internal variables, such as loop counters, subcarrier loop accumulators, etc., are reset. Control is then passed to the routine NORM.

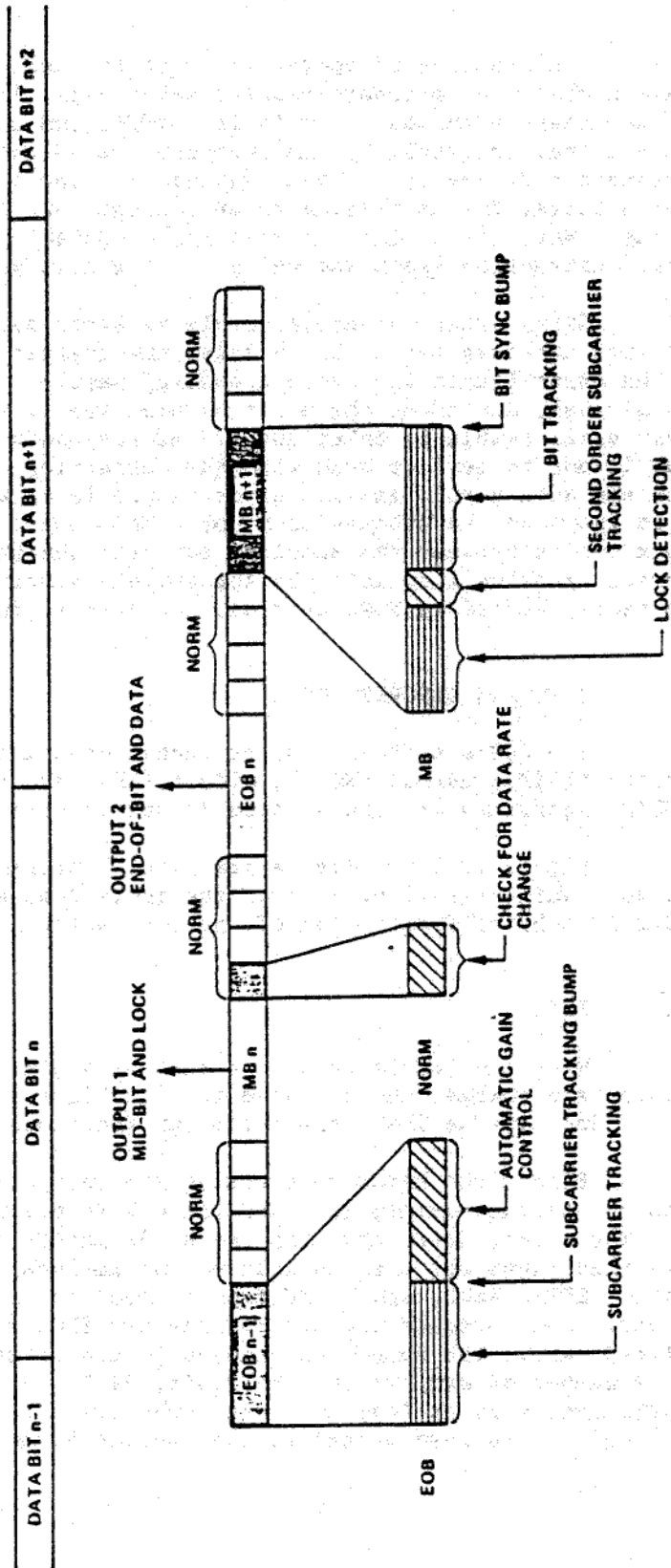


Figure 10-2. CDU Processor Timing

10.3.2 NORM

The NORM routine has the job of accumulating the samples in between the operation of the MB or EOB routines. NORM also outputs the CDU Telemetry status and SNR data words at the appropriate times, and it also checks for a data rate change. A brief description of the flow follows.

The error and data samples are taken and accumulated. This procedure includes the setting of the programmable timer to countdown the time to the next sample. Then, NORM checks to see if it is time to output the SNR data; if it is, the SNR word is sent. The next thing that NORM does is check to see if it is time to output the Telemetry status word; if it is, the word is sent. The data rate input is then checked; if the data rate has changed, control jumps to the INIT routine. If the data rate has not changed, the sub-carrier cycle counter is incremented by one. When this counter is equal to NSB2, the counter is reset and a decision is made to either jump to MB or jump to EOB. Otherwise, NORM loops back to the beginning of NORM.

10.3.3 MB

The MB routine takes place once each data bit period, at the predicted midbit position. It is four sample periods (eight subcarrier periods) in length; thus it accumulates four sets of error and data samples. A brief description of what takes place follows:

First, the Lock Detection algorithm is performed. Then the second order portion (the integration) of the Subcarrier Tracking Loop algorithm is done. Next, the Bit Sync Tracking algorithm is completed. In this loop, MB checks to see if there have been 64 consecutive bits without a transition. If this is the case, the bit sync bump is set to NSB4. If there has been a transition in the last 64 bits, the bit sync bump is set to the value obtained from the Bit Sync Loop. Finally, the bit sync bump is applied. This is done by varying the number of samples between MB and EOB (by adjusting the subcarrier cycle counter), which is the same as varying the number of times that NORM is run between MB and EOB. Control then jumps to the NORM routine.

10.3.4 EOB

The EOB routine takes place at the predicted end of each bit. It is five sample periods (ten subcarrier periods) in length; thus it contains five sets of error and data sample and accumulates. Briefly, the routine proceeds as follows.

First, the Subcarrier Tracking Loop calculations, which were started in MB are completed. Then the subcarrier bump is applied. The subcarrier bump is accomplished by using the result of the Subcarrier Tracking Loop as an index for a lookup table of times to program the programmable sampling timer. These times allow the CDU to adjust the time between the data sample and the error sample to try to get the error sample to occur at the subcarrier's zero crossing. After the subcarrier bump is applied, EOB does the AGC Loop calculations. Finally, control jumps to the NORM routine.

10.4 SOFTWARE OUTPUT

In addition to command data, bit timing, and lock indicator signals, the CDU software also outputs two bytes of information, the CDU Status Word and the CDU SNR Word.

10.4.1 CDU Status Word

The CDU outputs an eight bit byte of status information with the following format:

<u>Bit Number</u>	<u>Bit Description</u>
1	Data rate LSB
2	Data rate bit 2
3	Data rate MSB
4	CDU lock status
5	SEU reset indicator
6	Not used
7	Not used
8	Not used

Where bit 1 is the least significant bit and bit 8 is the most significant bit.

10.4.2 CDU SNR Word

The SNR word is the sum over eight bit times of the absolute value of the data accumulator, scaled by ASCL. This is used to provide an estimate of the uplink SNR (see Section 8).

10.5 SEU HANDLING

Of the several things that an SEU can do to the CDU, only two conditions exist that could potentially cause the CDU to go into a state from which it could not recover by itself: an endless loop caused by a mis-jump into nonexistent memory or a CPU halt caused by temporary malfunction of the internal CPU sequencing system.

In order to allow the CDU to recover from either of these two conditions, the program provides a "proper operation" signal to the hardware SEU recovery system every 125 microseconds (every other subcarrier cycle). If this signal is not received by the recovery system in time, the system will order an SEU reset, which is the equivalent to a power-on reset (but the power is not cycled off/on). The first CDU status word sent after the reset will indicate a SEU reset has occurred.

10.6 REFERENCES

- 10-1 Albrecht, V.R., Design Requirement NASA Deep Space Command Detector Unit, DM514438, Rev. A, August 1986.
- 10-2 Harris Corporation, CMOS Digital Design Data Book, Vol. 5, 1986.

This page left intentionally blank.

APPENDIX A

SOFTWARE FLOW CHARTS

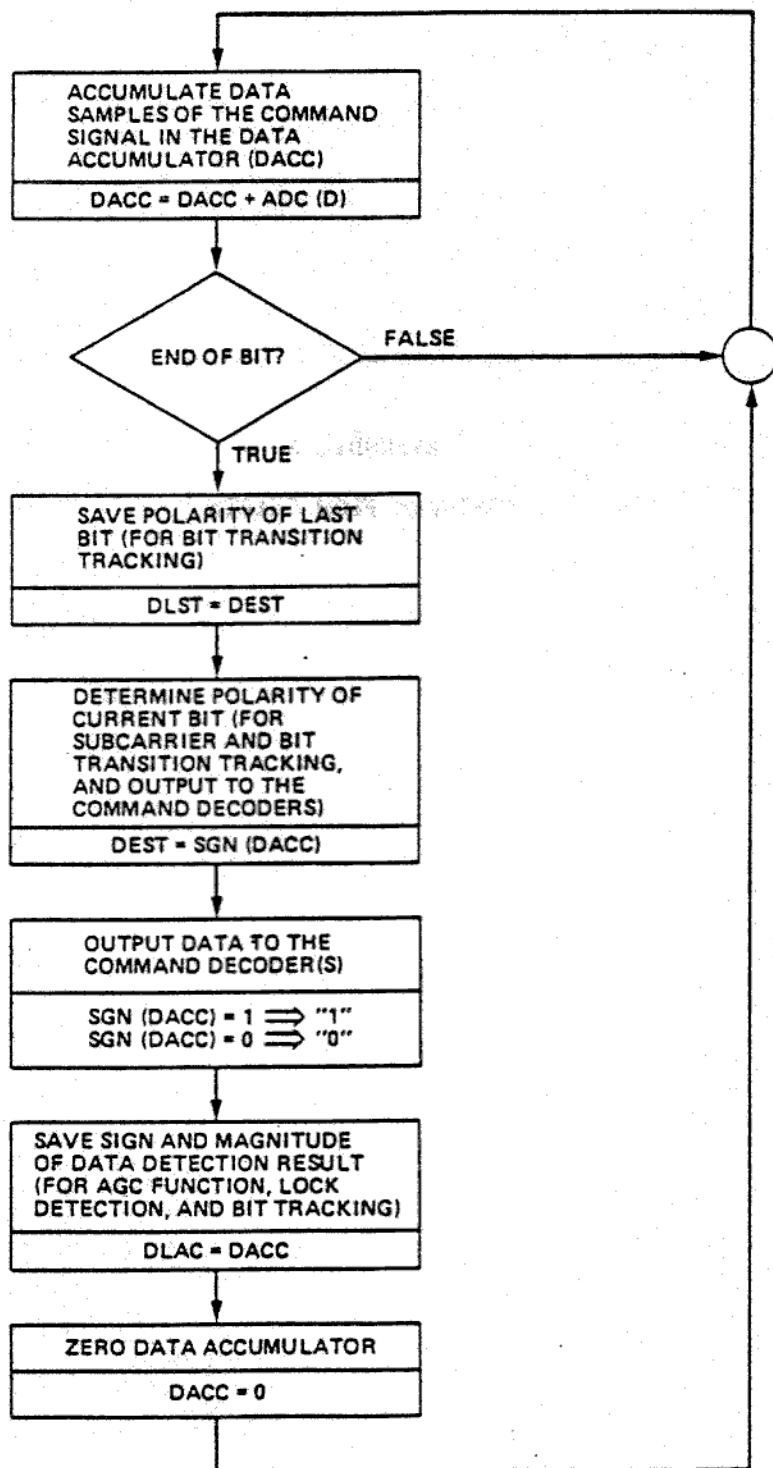


Figure A-1. Data Detection Algorithm

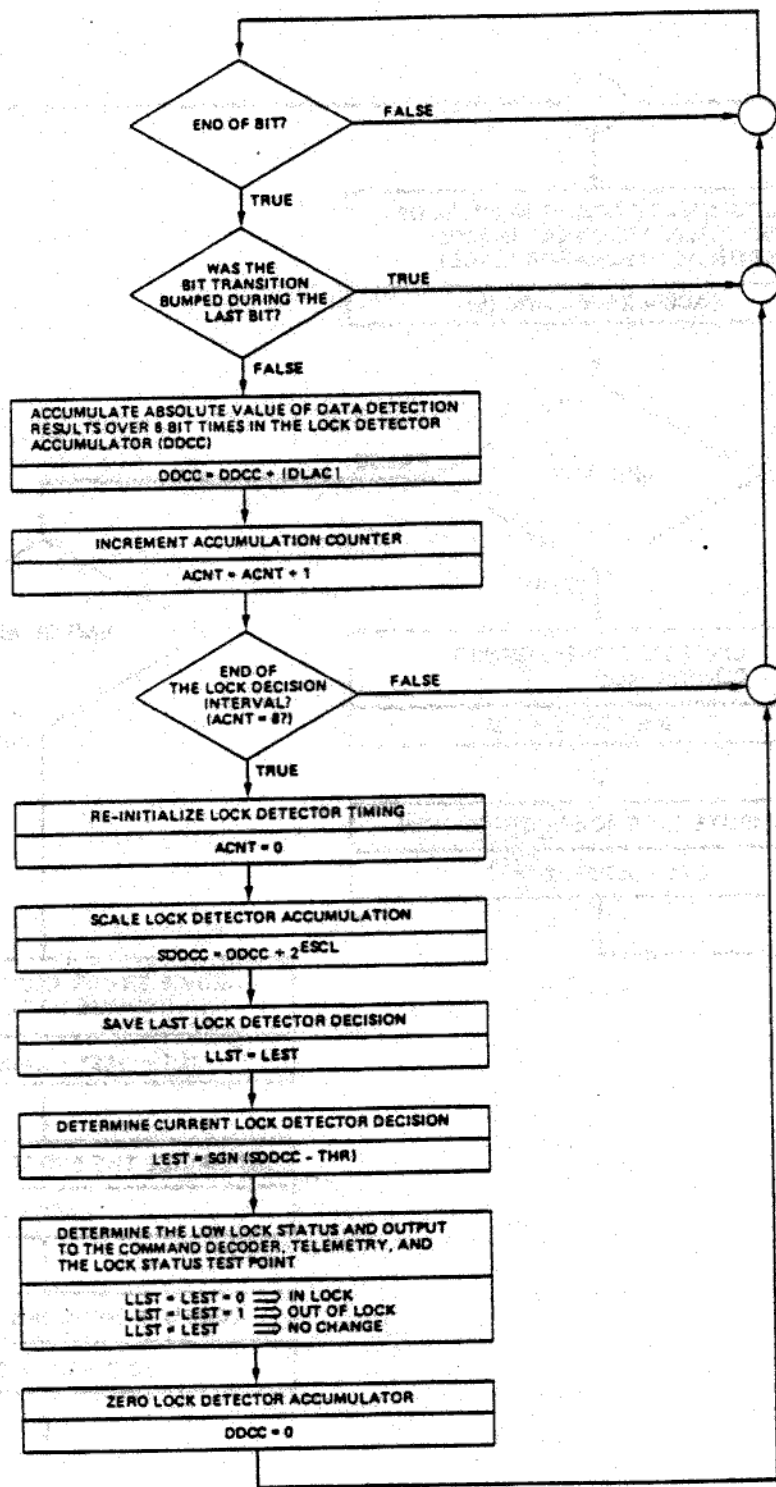


Figure A-2. Lock Detection Algorithm

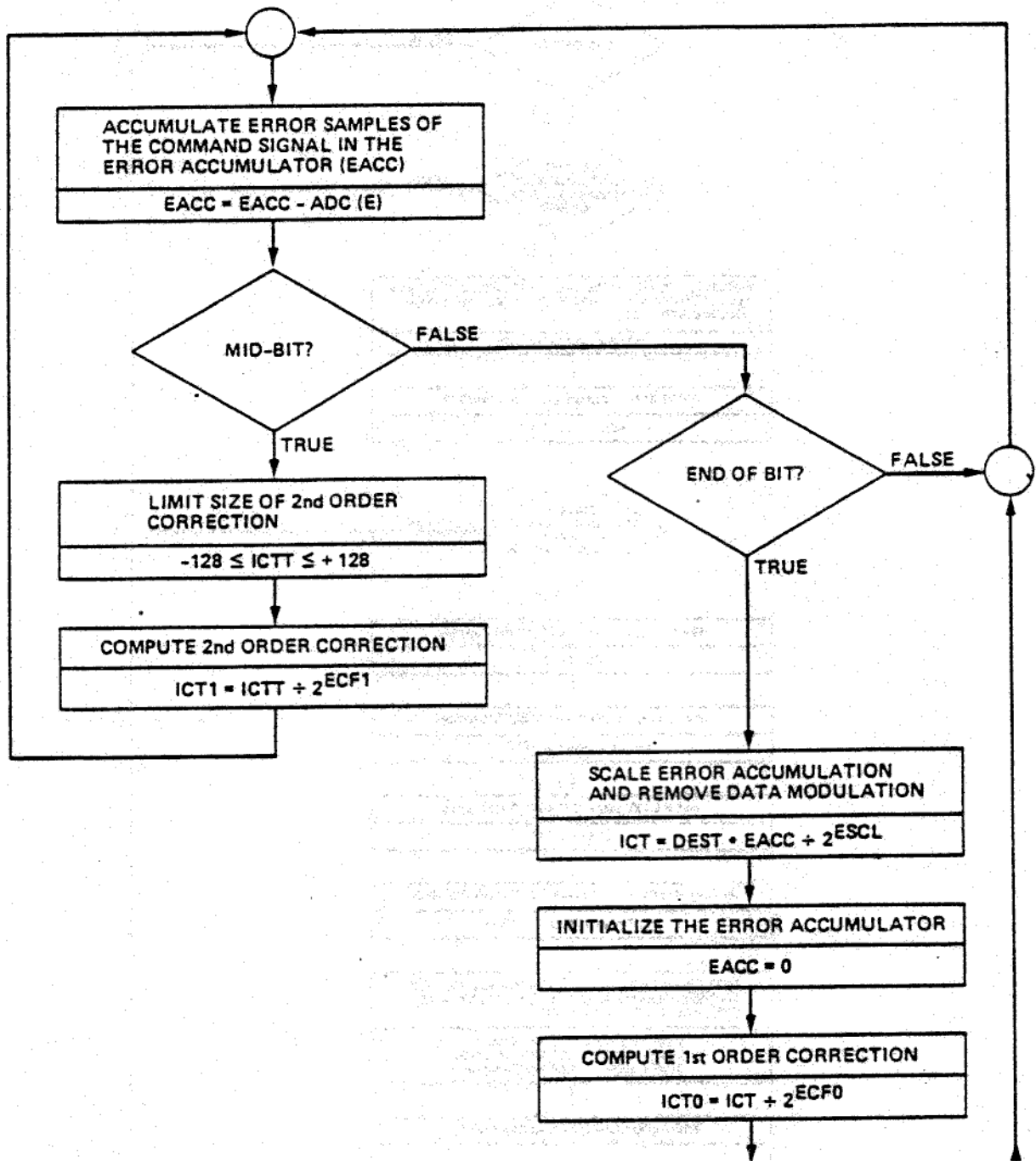


Figure A-3. Subcarrier Tracking Algorithm

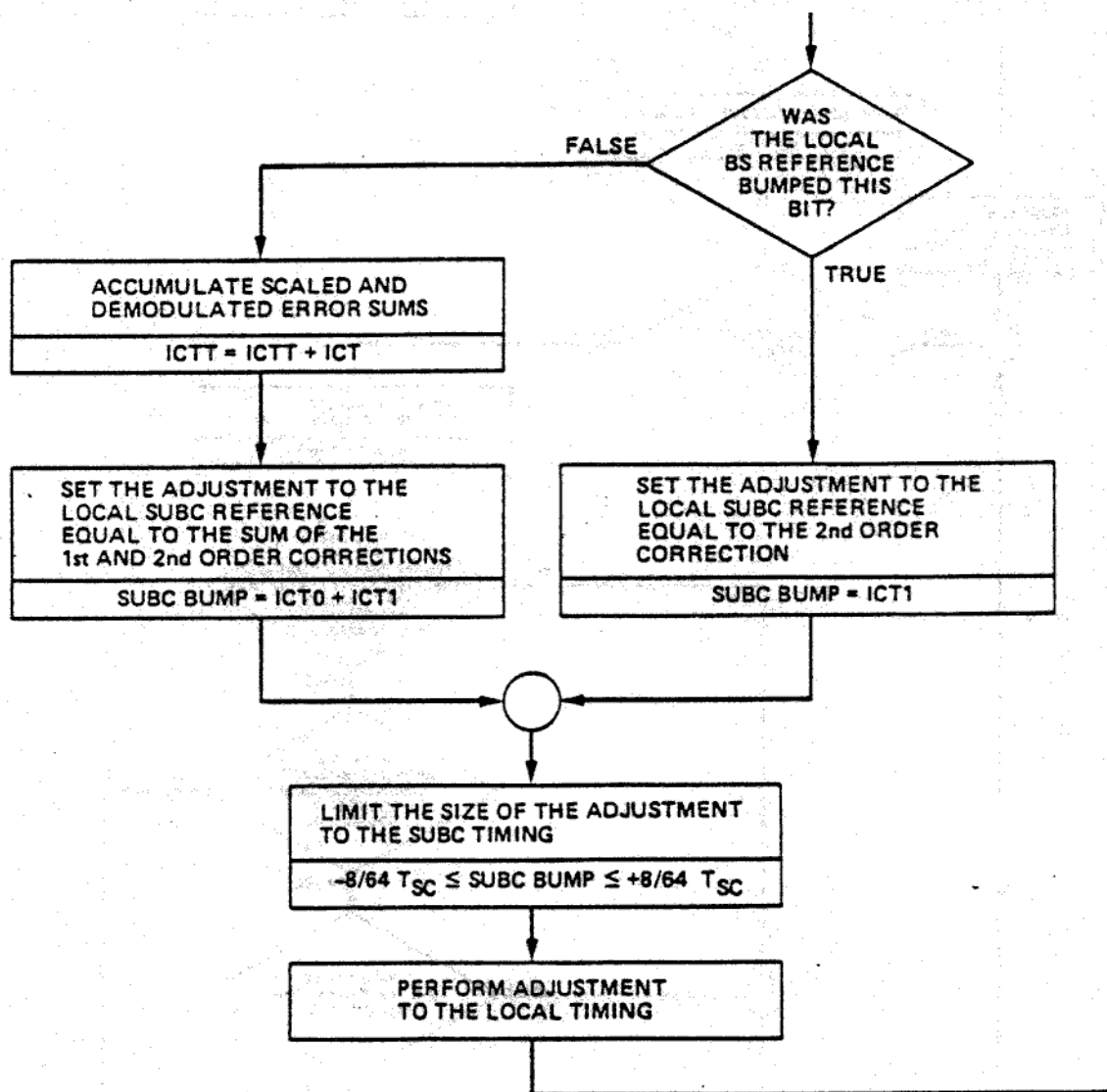


Figure A-3. Subcarrier Tracking Algorithm (Continued)

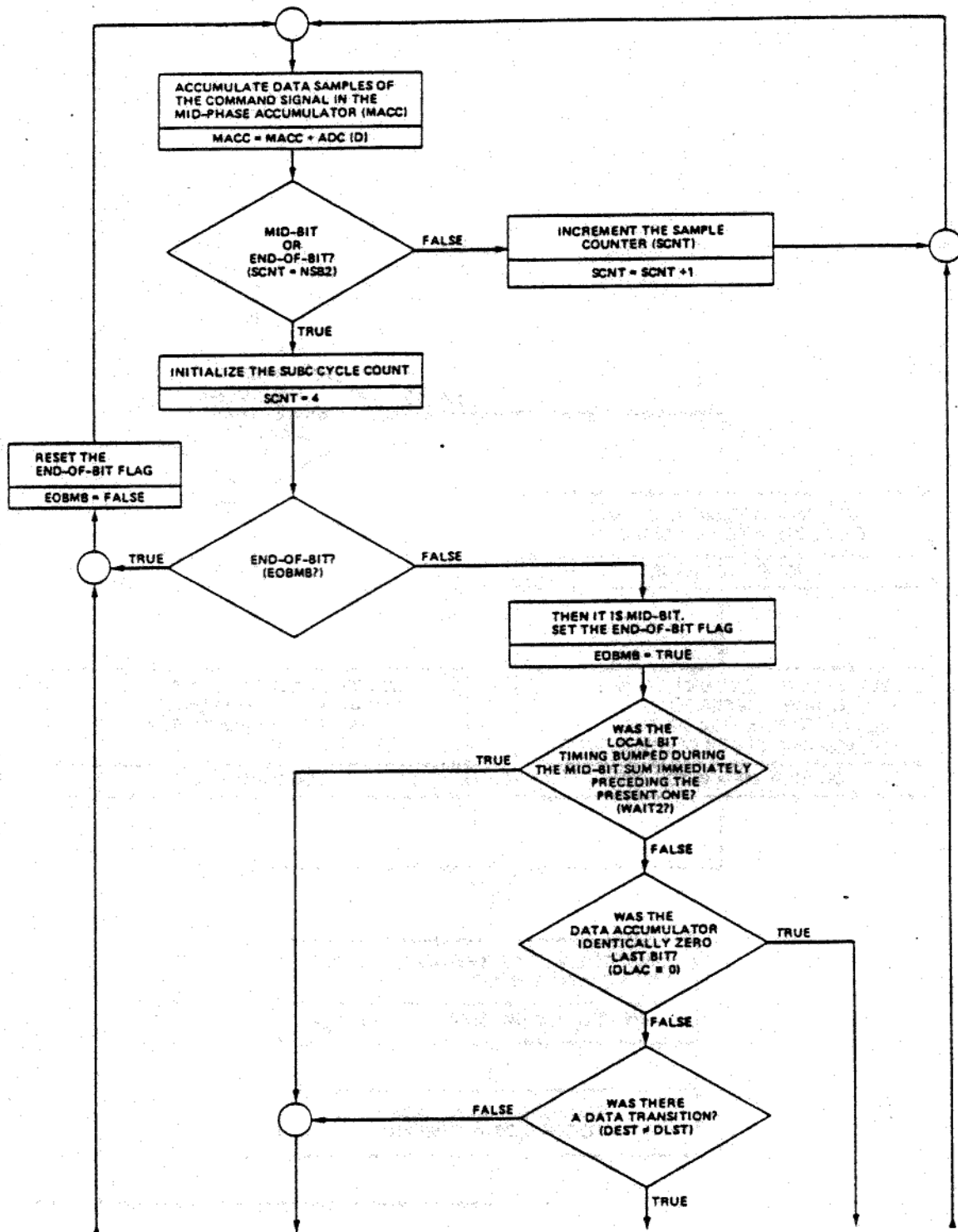


Figure A-4. Bit Tracking Algorithm

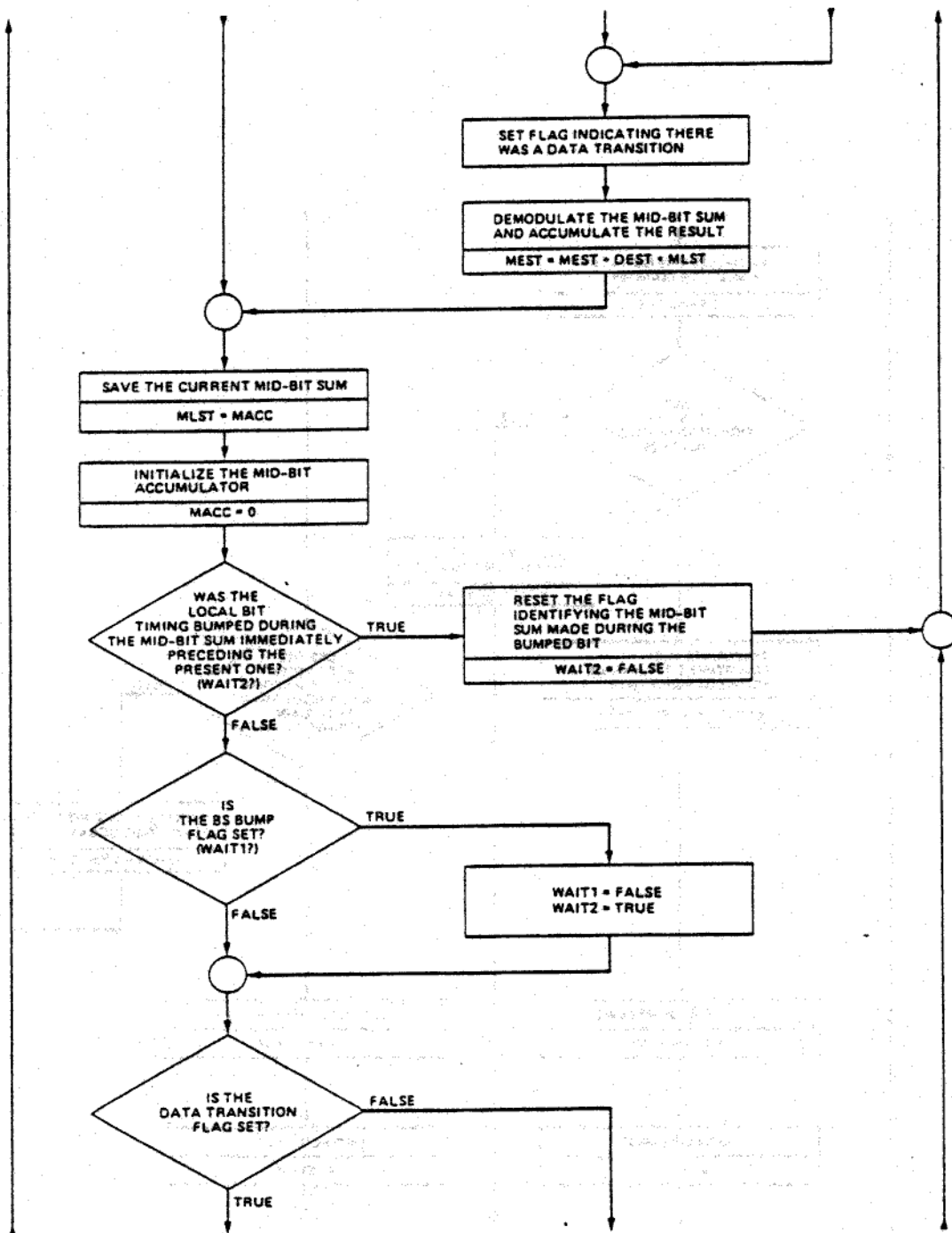


Figure A-4. Bit Tracking Algorithm (Continued)

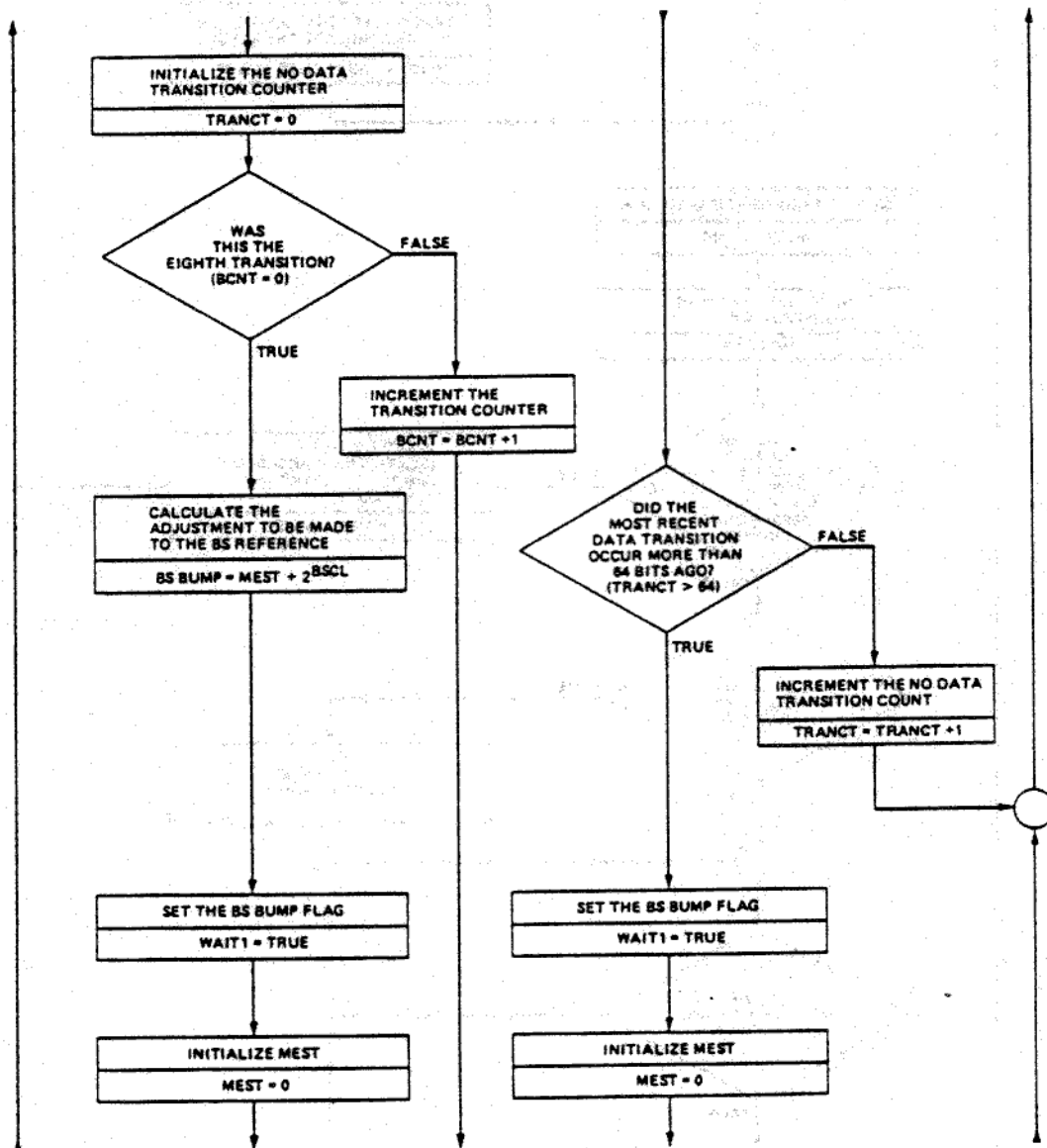


Figure A-4. Bit Tracking Algorithm (Continued)

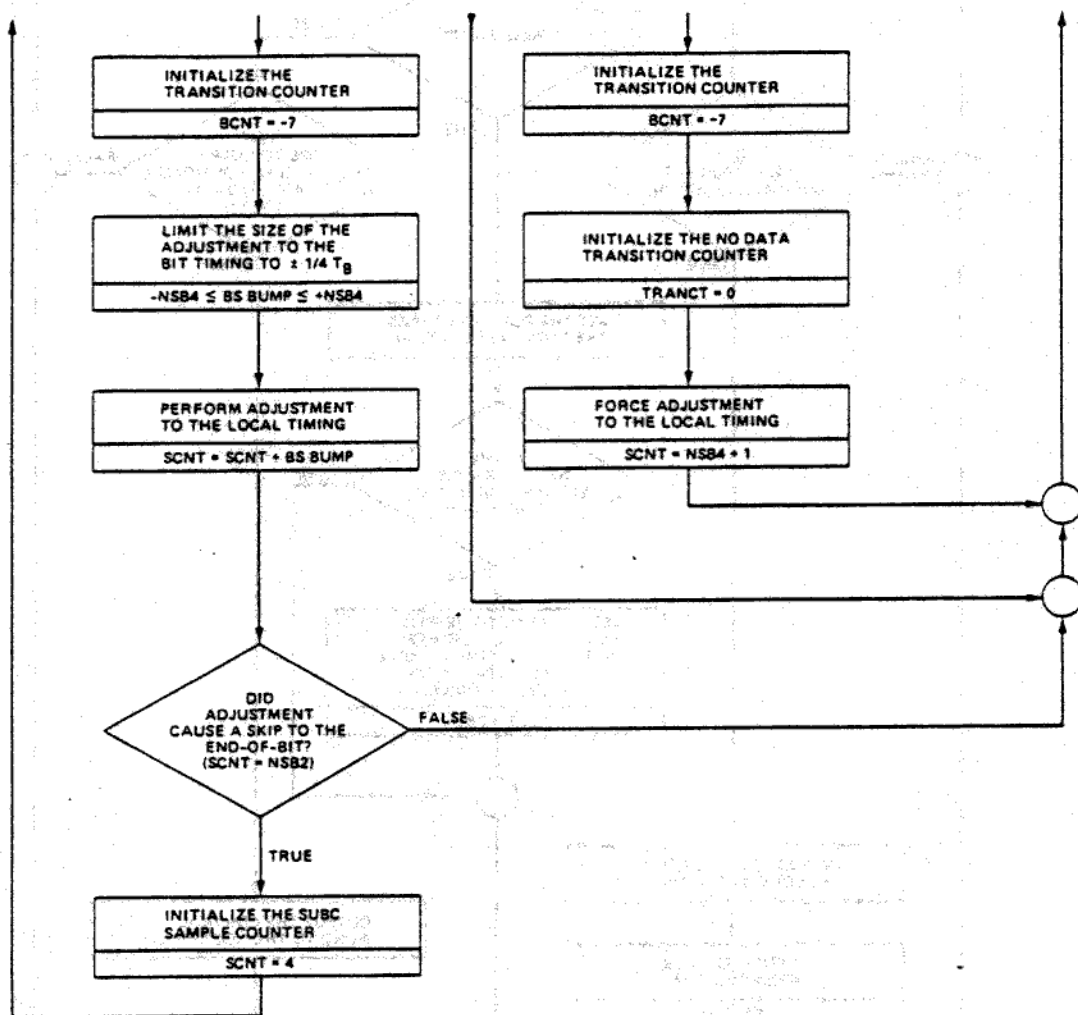


Figure A-4. Bit Tracking Algorithm (Continued)

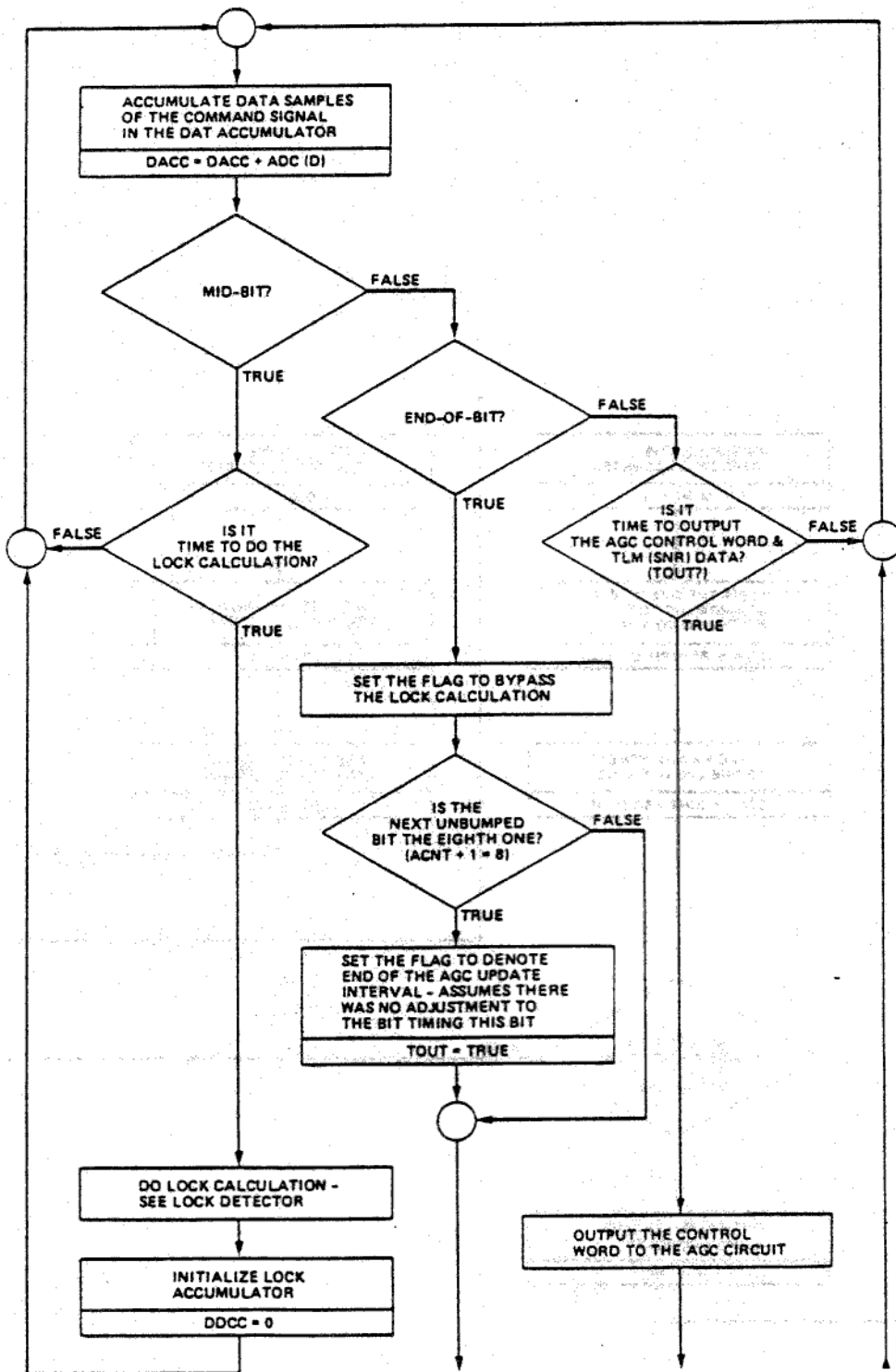


Figure A-5. Automatic Gain Control Algorithm

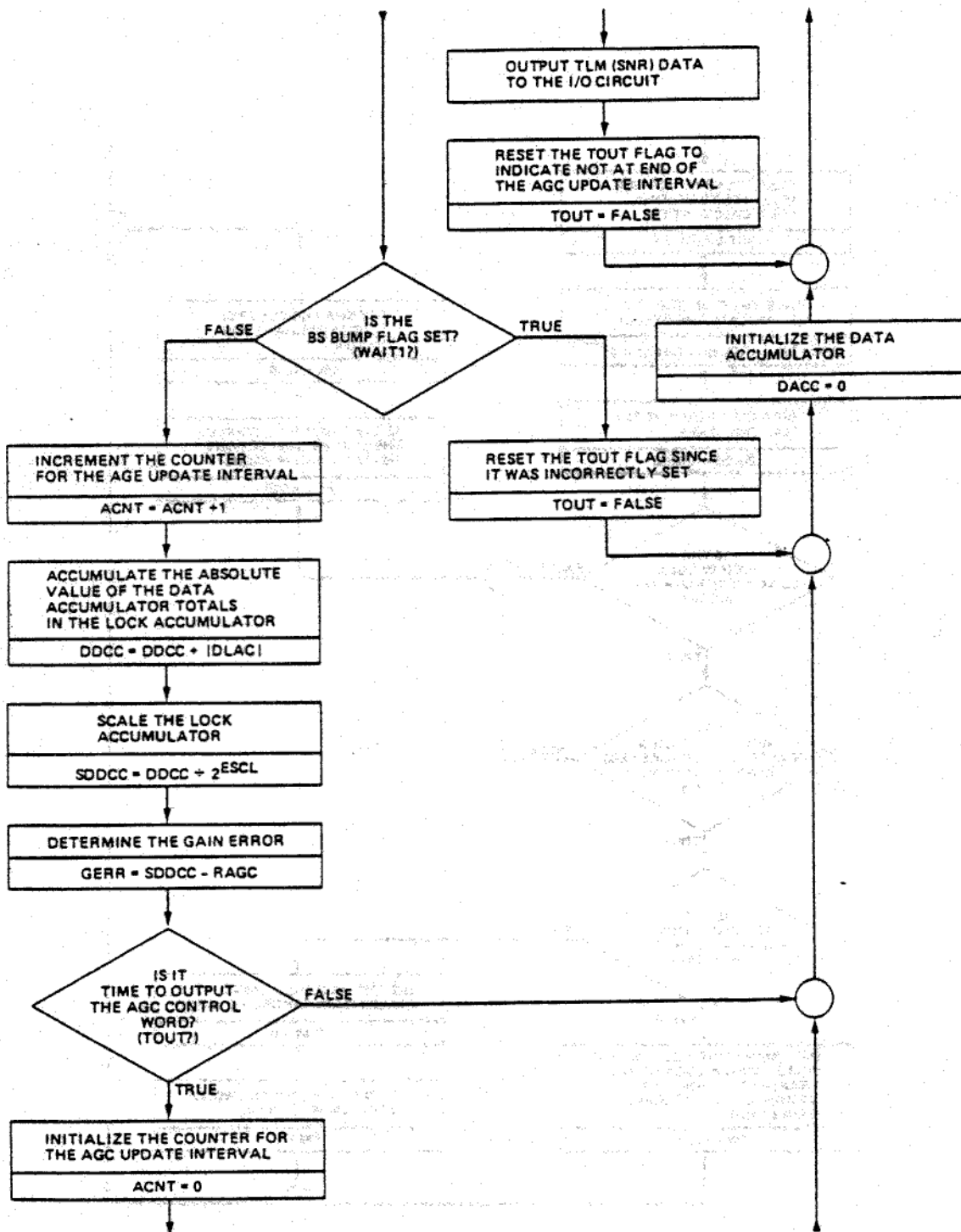


Figure A-5. Automatic Gain Control Algorithm (Continued)

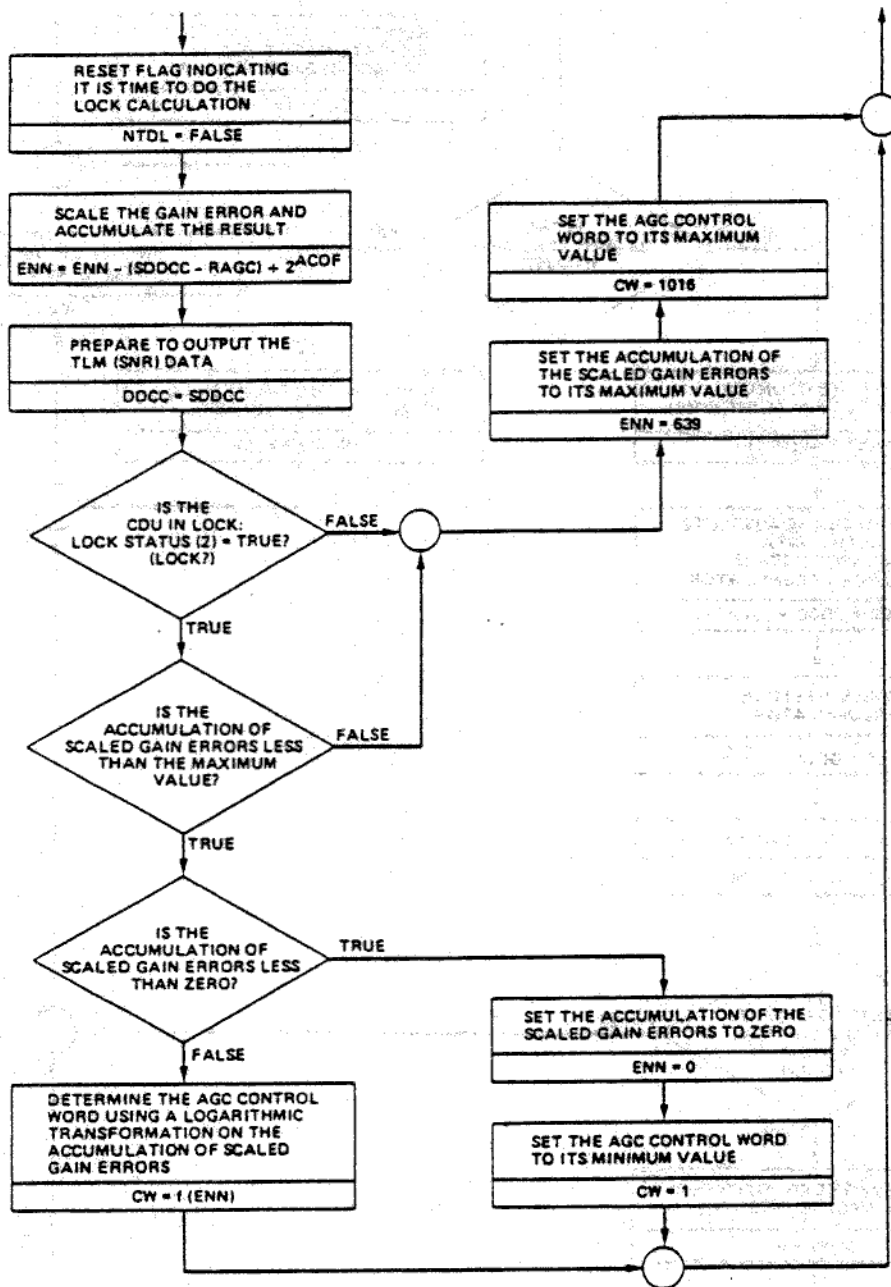


Figure A-5. Automatic Gain Control Algorithm (Continued)

APPENDIX B

CDU CODE

DOS 8086/87/88/186 MACRO ASSEMBLER V2.0 ASSEMBLY OF MODULE CDU86
 OBJECT MODULE PLACED IN CDUFLT.OBJ
 ASSEMBLER INVOKED BY: C:\INTEL\ASM86.EXE CDUFLT.ASM

LOC OBJ LINE SOURCE

1 +1 \$SYMBOLS
 2
 3 ++
 4
 5
 6
 7
 8
 9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50

File: c d u , a s m
 Date: 31-Mar-86
 Abstract: Program code for CDU-86.
 Rev: 861201.0 SOFTWARE FROZEN ...
 Flight Version 870202.0

C MACRO
 #
 #
 #
 #
 #
 #

```

name cdU86
CodeMacro AbsVal destRw ; Absolute value of word register.
Segfix dest
DB 85h ; test rw, rw
ModRM dest, dest ; jns end
DB 0279h ; neg rw
DB 0F7h ; end:
ModRM 3, dest
EndM
    
```

C MACRO
 #
 #
 #
 #

```

CodeMacro Clr destRw ; Clear word register.
Segfix dest
DB 33h ; xor rw, rw
ModRM dest, dest
EndM
    
```

C MACRO
 #
 #
 #
 #

```

CodeMacro Clr destRb ; Clear byte register.
Segfix dest
DB 32h ; xor rb, rb
ModRM dest, dest
EndM
    
```

```

; The old AGC coefficient.
; Address of analog to digital converter.
; The largest AGC amp control register.
; Address of 82C54 (U 29) control reg.
; Address of 82C54 (U 30) control reg.
; Clock zero data reg.
; Clock zero reg.
; Clock one data reg.
; Clock one reg.
; Clock two data reg.
; Clock two reg.
    
```

CDU86

8086/87/88/186 MACRO ASSEMBLER

```

LOC  OBJ          LINE          SOURCE
51      +22      ; Cycle count from data to error sample.
52      +22      ; The ms byte of the above.
53      +22      ; Cycle count from data to error sample
54      +22      ; with room for the subcarrier bump.
55      +22      ; From error sample to data sample.
56      +22      ; MS byte
57      +22      ; Output bit position for data bit.
58      +22      ; If equal to one, assemble all debug code.
59      +22      ; Shift count used with lctt.
60      +22      ; Shift count used with lctt.
61      +22      ; Largest allowable ann number.
62      +22      ; Boolean false.
63      +22      ; One more than NASA standard cdu code.
64      +22      ; Output bit position for lock1 indication.
65      +22      ; Output bit position for lock2 indication.
66      +22      ; Number of bits before bit sync bump.
67      +22      ; Address of the input / output port.
68      +22      ; Automatic gain control reference number.
69      +22      ; Base address of random access memory.
70      +22      ;
71      +22      ; Base address of the read only memory.
72      +22      ; Number of samples per module.
73      +22      ; The middle entry in the sc bump table.
74      +22      ; The number of entries in the sc bump table.
75      +22      ; Address used by SEU recovery system.
76      +22      ; Serial i/o port address.
77      +22      ; Output position of lock2 in serial word.
78      +22      ; Address of the tim enable gate.
79      +22      ; Output position for reset in serial word.
80      +22      ; Output bit position for bit timing signal.
81      +22      ; Boolean true.
82      +22      ; Address of the tim enable gate.
83      +22      ;
84      +22      ;
85      +22      ;
86      +22      ;
87      +22      ;
88      +22      ;
89      +22      ;
90      +22      ;
91      +22      ;
92      +22      ;
93      +22      ;
94      +22      ;
95      +22      ;
96      +22      ;
97      +22      ;
98      +22      ;
99      +22      ;
100     +22      ; Subcarrier sample macro. Takes both error and data samples.
101     +22      ;
102     +22      ;
103     +22      ;
104     +22      ;
105     +22      ;

assume csiabs_seg, dstabs_seg
abs_seg segment at 0
D000  +1 org 0D000H ; All vars should be in RAM.

```

CDU86

8086/87/88/186 MACRO ASSEMBLER

LOC	OBJ	LINE	SOURCE
D000	????	106	
D002	????	107	; Words.
D004	????	108	2nd_decision
D006	????	109	acnt
D008	????	110	bcnt
D00A	????	111	bit_m
D00C	????	112	bsci_in
D00E	????	113	bsci_in_rnd
D010	????	114	bsci_out
D012	????	115	bsci_out_rnd
D014	????	116	bsci_rnd
D016	????	117	daccave
D018	????	118	ddcc1
D01A	????	119	ddcc2
D01C	????	120	dest
D01E	????	121	dist
D020	????	122	enn
D022	????	123	enbmb
D024	????	124	escl
D026	????	125	ictl
D028	????	126	lcth
D02A	????	127	lock1
D02C	????	128	lock2
D02E	????	129	mest
D030	????	130	mlst
D032	????	131	notran
D034	????	132	nsb2
D036	????	133	nsb4
D038	????	134	ntd1
D03A	????	135	escl_rnd
D03C	????	136	sio_not_done
D03E	????	137	snr_check
D040	????	138	tranc
D042	????	139	ulcth
D044	????	140	wait1
D046	????	141	wait2
D048	????	142	
D04A	????	143	
		144	
		145	
		146	
		147	+1
		148	
		149	
		150	; Bytes.
D04C	??	151	il
D04D	??	152	olddr
D04E	??	153	output
D04F	??	154	soutput
		155	+1
		156	
		157	
		158	; Device registers bound to hardware dependent addresses.
		159	+1
		160	+2
A000			org 0A000H

Boolean.
 Boolean, bit sync scale factor.
 Current factor used when in lock.
 Round-off when in lock, out of lock.
 Scale factor used when out of lock.
 Round-off when out of lock.
 Current round off constant.
 High 16-bits of 32-bit ddec.
 Low 16-bits of 32-bit ddec.
 Boolean.
 Also used as ascl.
 Unlock to lock threshold.
 Boolean.
 Bump used when no transitions happen.
 Number of samples per bit divided by two.
 Number of samples per bit divided by four.
 Not time to Do Lock. Boolean.
 Boolean.
 Lock to unlock threshold.
 Boolean.
 Boolean.
 Basic output word when in-lock.
 Old data rate.
 Output word buffer.
 Serial output buffer.

CDU86

8086/87/88/186 MACRO ASSEMBLER

LOC	OBJ	LINE	SOURCE	?
A000	??	161 +1	atod	db
B000	????	162 +1		org 0B000H
B000	????	163 +2	agc	dw
9005	??	164 +2		?
9005	??	165 +1		
9005	??	166 +1		
9005	??	167 +1	ccw_1	org 9006H
9005	??	168 +2		db
9005	??	169 +2		?
9005	??	170 +1		
9005	??	171 +1	ccw_2	org 8806H
9005	??	172 +2		db
9005	??	173 +2		?
9000	??	174 +1		
9000	??	175 +1		
9000	??	176 +2	cc0_1	org 9000H
9000	??	177 +2		db
9000	??	178 +2		?
9000	??	179 +1		
9000	??	180 +2	cc0_2	org 8800H
9000	??	181 +2		db
9000	??	182 +1		?
9002	??	183 +1		
9002	??	184 +2	cc1_1	org 9002H
9002	??	185 +2		db
9002	??	186 +1		?
9002	??	187 +1		
9002	??	188 +2	cc1_2	org 8802H
9002	??	189 +2		db
9002	??	190 +1		?
9004	??	191 +2		
9004	??	192 +2	cc2_1	org 9004H
9004	??	193 +1		db
9004	??	194 +1		?
9004	??	195 +1		
9004	??	196 +2	cc2_2	org 8804H
9004	??	197 +2		db
9004	??	198 +1		?
9004	??	199 +1		
9004	??	200 +2	pio	org 0C000H
9004	??	201 +2		db
9004	??	202 +1		?
9004	??	203 +1		
9004	??	204 +2	seu	org 0E000H
9004	??	205 +2		db
9004	??	206 +1		?
9004	??	207 +1		
9004	??	208 +2	sio	org 8008H
9004	??	209 +2		db
9004	??	210 +1		?
9004	??	211 +1		
9004	??	212 +2	snr	org 8020H
9004	??	213 +2		db
9004	??	214 +1		?
9004	??	215 +1		

8086/B7/88/186 MACRO ASSEMBLER CDU86

LOC	OBJ	LINE	SOURCE
8010	??	216 +2	org 8010h
8010	??	217 +2	db ?
		219 +1	
0000		220	org 0
0000		221	start:
0000	A200E0	222 +1	mov seu, al
0003	A04FD0	223 +1	al, _soutput
0006	0410	224 +1	add al, 1 shl 04h
0008	A24FD0	225 +1	mov _soutput, al
000B		226	
000B	A200E0	227 +1	mov seu, al
000E	C606069072	228 +1	ccw 1, 072h
0013	C606029003	229 +1	cc1 1, 3
0018	C606029000	230	cc1 1, 0
001D	C606069034	231 +1	ccw 1, 034h
0022	C60600904E	232 +1	cc0 1, 4EH
0027	C606009000	233	cc0 1, 00h
002C	C606049005	234 +1	ccw 1, 096h
0031	C606068816	235	cc2 1, 005h
0036	C606008827	236	ccw 2, 016h
0040	C606068852	237 +1	ccw 2, 052h
0045	C606028808	238	cc1 2, 8
004A	C606068892	239	cc2 2, 8
004F	C606048808	240	agc, 03FBh
0054	C70600B0F803	241 +1	sacc = 0
005A	33F6	242	dacc = 0
005C	33E4	243	macc = 0
005E	33ED	244	
0060	A000C0	245	Read data rate.
0063	2407	246 +1	Mask off junk bits.
0065	A24DD0	247 +1	Save in RAM.
0068	A24ED0	248	Initialize output word with out-of-lock word.
006B	A700C0	249 +1	Set the output to the "out of lock" word.
006E	8A20	250	Set up basic in lock word.
0070	80CC70	251 +1	Set lock bit. (in lock)
0073	80CC10	252 +1	Set sync bit. (eob)
0076	88264CD0	253	Save this as the "in lock" word.
007A	3C07	254	Check for invalid data rate.
007C	7502	255	
007E	32C0	256	
0080	8A1E4FD0	257	
0084	80E3F0	258	
		259	
		260	
		261	
		262	
		263	
		264	
		265	
		266	
		267	
		268	
		269	
		270	

8086/87/88/186 MACRO ASSEMBLER CDU86

LOC	OBJ	LINE	SOURCE
0087	0AD8	271	or
0089	8B1E4FD0	272	mov
008D	32E4	273	cir
008F	D1E0	274	sar
		275	+1
0091	BB4CF790	276	mov
0095	03D8	277	add
0097	8B0F	278	mov
0099	890E0AD0	279	mov
		280	+1
009D	BA0100	281	mov
00A0	49	282	dec
00A1	D3E2	283	shl
00A3	89160CD0	284	mov
		285	+1
00A7	BB5AF790	286	mov
00AB	03D8	287	add
00AD	8B0F	288	mov
00AF	890E0ED0	289	mov
		290	+1
00B3	BA0100	291	mov
00B6	49	292	dec
00B7	D3E2	293	shl
00B9	891610D0	294	mov
		295	+1
00BD	BB78F790	296	mov
00C1	03D8	297	add
		298	+1
00C3	A200E0	299	mov
		300	+1
00C6	8B0F	301	mov
00C8	890E22D0	302	mov
		303	+1
00CC	BA0100	304	mov
00CF	49	305	dec
00D0	D3E2	306	shl
00D2	89163AD0	307	mov
00D6	891618D0	308	mov
		309	+1
00DA	BB66F790	310	mov
00DE	03D8	311	add
00E0	8B0F	312	mov
00E2	890E28D0	313	mov
		314	+1
		315	+1
00E6	BB94F790	316	mov
00EA	03D8	317	add
00EC	8B0F	318	mov
00EE	890E34D0	319	mov
		320	+1
00F2	D1F9	321	sar
00F4	890E36D0	322	mov
00F8	41	323	inc
00F9	890E32D0	324	mov
		325	+1

```

bi, al
soutput, bl
ah
ax, 1
; Paste the data rate on it.
; Store back in RAM for later output.
; Clear high byte for 16-bit math.
; Make it a word pointer.
bx, offset bsc1_in_table + 0F000H
cx, [bx]
mov [bx], cx
; Macro left bsc1_in in cx,
; now cx is bsc1_in - 1.
; bsc1_in_rnd is 2 ^ (bsc1_in - 1)
; Used as round-off const in lock.
dx, 1
cx, cl
_bsc1_in_rnd, dx
; Macro left bsc1_out_table + 0F000H
; Used as round-off const in lock.
bx, offset bsc1_out_table + 0F000H
cx, [bx]
mov [bx], cx
; Macro left bsc1_out in cx,
; now cx is bsc1_out - 1.
; bsc1_out_rnd is 2 ^ (bsc1_out - 1)
; Used as round-off const out of lock.
dx, 1
cx, cl
_bsc1_out_rnd, dx
; Macro left escl_table + 0F000H
; Used as round-off const out of lock.
bx, offset escl_table + 0F000H
cx, [bx]
mov [bx], cx
; Let the seu reset system know what's what.
seu, al
; Load with a bit.
; cx is now escl - 1.
; Shift the bit left.
; escl_rnd = 2 ^ (escl - 1)
; Round-off constant for ddcc.
dx, 1
cx, [bx]
mov [bx], cx
; Round-off constant for ddcc.
ddcc7, dx
; Macro left lkth_table + 0F000H
; Used as round-off constant for ddcc.
bx, offset lkth_table + 0F000H
cx, [bx]
mov [bx], cx
; Macro left nsb2_table + 0F000H
; Used as round-off constant for nsb2.
dx, 1
cx, [bx]
mov [bx], cx
; Macro left nsb2 in cx, now cx is nsb4.
; Store it in ram also.
; Calculate notran as
; nsb4 + 1.
notran, cx

```

```

LOC OBJ          LINE SOURCE
00FD BBA2F790    326 +2
0101 03DB       327 +1
0103 8B0F       328 +1
0105 890E46D0   329 +2
0109 33C0       330 +1
010B A302D0     331
010E A304D0     332
0111 8B1E0ED0   333
0115 891E08D0   334
0119 8B1E10D0   335
011D 891E12D0   336
0121 A314D0     337
0124 A316D0     338
0127 A318D0     339
012A A31CD0     340
012D A31ED0     341
0130 A320D0     342
0133 A326D0     343
0136 A300D0     344
0139 A32AD0     345
013C A32CD0     346
013F A32ED0     347
0142 A330D0     348
0145 C79638D00100 349
014B C7063CD00400 350 +1
0151 A304D0     351 +1
0157 A30AD0     352
015A A34AD0     353
015D A34BD0     354
0160 A34AD0     355
0163 EB0190     356
                                357
                                358
                                359
                                360
0166             361 +1
0166 9B         362 +1
0167 A200E0     363 +1
016A C606009022 364 +1
016F C606009002 365 +2
0174 8BD6       366 +2
0176 A000A0     367 +1
0179 98         368 +1
017A 2BD0       369 +1
017C 9B         370 +1
017D C60600904E 371 +1
0182 C606009000 372 +2
0187 7002       373 +2
0189 8BF2       374 +2
018B             375 +2
                                376 +2
                                377 +1
                                378 +1
                                379 +1
0163 EB0190     380 +2
                                LX00:

bx, offset ulkth_table + 0F000H
mov add
mov cx, [bx]
mov _ulkth, cx

clr ax
mov acnt, ax
mov bcnt, ax
mov bx, bsci_out
mov bsci, bx
mov bsci_out_rnd, bx
mov daccsavr, ax
mov dest, ax
mov dist, ax
mov enn, ax
mov iptt, ax
mov inddecision, ax
mov lock1, ax
mov lock2, ax
mov mest, ax
mov mlt, ax
mov ntdl, 01H
mov scnt, 04H
mov sig_not_done, 00H
mov bit_m, ax
mov trant, ax
mov wait1, ax
mov wait2, ax
norm
jmp

;-----
norm:

; Sync for error sample.
; Reset SEU

; "Load" eacc.
; Get the error sample.
; Convert atod input into two's comp word.
; eacc - error_sample

; Sync for data sample.
; If sub overflowed then don't store.
; "Store" eacc.

```

CDU86

8086/87/88/186 MACRO ASSEMBLER

LOC	OBJ	LINE	SOURCE	ASSEMBLY	COMMENT
018B	A000A0	381		mov al, atod	; Get the data sample.
018E	78	382		cbw	
018F	88D0	383		mov dx, ax	; Save copy for later.
0191	03C4	384		add ax, sp	; Add dacc.
0193	7002	385		jo LY01	; Don't store if overflow.
0195	8BE0	386		mov sp, ax	; "Store" dacc.
0197	03D5	387		add dx, bp	; Add macc to copy saved.
0199	7002	389		jo LZ02	; Don't store if overflow.
019B	8BEA	392		mov bp, dx	; "Store" macc.
019D		394			
019D		395			
019D		396			
019D	A106D0	397		mov ax, _bit_m	; See if it is time for snr output.
01A0	23063ED0	398		and ax, _sio_not_done	; If not Mth bit or sio hasse been done then
01A4	D1D8	399		ax, I	jump over serial i/o code altogether.
01A6	7938	400		jnc n300	
01A8	A120D0	401		mov ax, _eobmb	
01AB	D1D8	402		ax, I	
01AD	7212	403		rcr n100	
01AF	A140D0	405		mov ax, _snr_check	; The snr word is in snr_check.
01B2	A20880	407		sio, al	; Load snr word into output reg.
01B5	A22080	408		snr, al	; Gate the output.
01B8	C7063ED00000	409		sio_not_done, 00H	
01BE	EB2090	410		jmp n300	; Skip other serial i/o for this bit.
01C1	A04FD0	411	n100:	mov al, _soutput	; Fetch serial tim word.
01C4	8B1E2CD0	412		bx, I	; Test lock2 status.
01C8	D1D8	414		rcr n200	; If clear then do output.
01CA	7302	415		jnc	; If set then
01CC	0C08	417		or al, 1 shl 03H	; set lock2 bit in serial output word.
01CE	A20880	419		mov sio, al	; Load tim word into output reg.
01D1	A21080	421		tim, al	; Gate the output.
01D4	C7063ED00000	422		mov sio_not_done, 00H	
01DA	C70606D00000	423		mov _bit_m, 00H	; We are now done with this Mth bit.
01E0	A000C0	424	n300:	mov al, pio	; See if the data rate has changed.
01E0	A000C0	425		and al, 7	; Mask out junk bits.
01E3	2407	426		cmp al, _olddr	; If new_dr != old_dr then
01E5	3A064DD0	429		je	
01E9	7403	430		jmp init	; reinitialize constants and start from top.
01EB	E91DFE	431			
01EE	A13CD0	432	n400:	mov ax, _scent	; Fetch the cycle count
01EE	A13CD0	433			
01EE	A13CD0	434			
01EE	A13CD0	435			

LOC	OBJ	LINE	SOURCE
01F1	40	436	ax
01F2	A33CD0	437	inc
01F3	3B0634D0	438	mov
01F4	7403	439	cmp
01F5	E968FF	440	je
01F6	C7063CD00400	441	jmp
01F7	C7063ED00100	442	norm
01F8	A120D0	443	scnt, 04H
01F9	D0D8	444	_sio_not_done, 01H
0200	7303	445	ax, _eobmb
0201	E9A302	446	si, 1
0202		447	mb
0203		448	eob
0211		450	jmp
0214		451	;
0215	9B	452	mbt
0216	A200E0	453	wait
0217	C606009022	454	mov
0218	C606009002	455	mov
0219	8BD6	456	mov
0220	A000A0	457	mov
0221	98	458	mov
0222	2BD0	459	cbw
0223	9B	460	sub
0224	C60600904E	461	wait
0225	C606009000	462	mov
0226	7002	463	mov
0227	BBF2	464	jo
0228	A000A0	465	mov
0229	8BD0	466	mov
0230	98	467	mov
0231	8BD0	468	cbw
0232	03C4	469	add
0233	7002	470	jo
0234	BBE0	471	mov
0235	03D5	472	add
0236	7002	473	jo
0237	8BEA	474	mov
0238		475	
0239		476	
0240		477	
0241		478	
0242		479	
0243		480	
0244		481	
0245		482	
0246		483	
0247		484	
0248		485	
0249		486	
0250		487	
0251		488	
0252		489	
0253		490	

6086/87/88/186 MACRO ASSEMBLER CDU86

LOC	OBJ	LINE	SOURCE
0251	A138D0	491	ax, _ntdl ; If not time to do lock then
0252	D1D8	492	ax, _I
0253	7303	493	mov m100
0254		494	jnc
025B	E9A000	495	jmp m800 ; skip over lock detection code.
025C		496	
025D	A1ZCD0	497	ax, _lock2 ; If in lock, check if out of lock.
025E	D1D8	498	ax, _I ; Otherwise jump to in lock test.
025F	7345	499	mov m400
0260		500	jnc
0262	A118D0	501	ax, _ddcc2 ; Check ddcc against threshold.
0263	380646D0	502	ax, _ulkth ; If in lock then skip block 1.
0264	792D	503	cmp m300
0265		504	jns
026B	C7062AD00000	505	mov lock1, 00H
0271	A100D0	506	ax, _2nd_decision
0274	D0D8	507	mov al, _I
0276	7209	508	rcr m200 ; If jagc is 00H then
0277		509	jc
0278	C70600D00100	510	mov _2nd_decision, 01H ; first pass, don't set lock2.
027E	EB6F90	511	jmp m700
0281	33C0	512	
0283	A32CD0	513	
0286	A300D0	514	clr lock2, ax ; Get a fast 00H into ax.
0289	A10ED0	515	mov _2nd_decision, ax
028C	A308D0	516	ax, _bscl_out ; Now we are out-of-lock so
028F	A110D0	517	mov bscI, ax ; bscI gets bscI_out.
0292	A312D0	518	ax, _bscl_out_rnd
0295	EB5890	519	mov bscI_rnd, ax ; Same with round-off const.
0298	C70600D00000	520	mov m700
029E	C7062AD00100	521	_lock1, 01H
02A4	EB4990	522	jmp m300 ; 2nd decision, 00H
02A7	A118D0	523	ax, _ddcc2
02AA	8B1E28D0	524	bx, _lkth
02AE	3BD8	525	cmp bx, ax ; If ddcc > lock threshold then
02B0	7931	526	jns m600 ; CDU is in lock.
02B2	C7062AD00100	527	mov lock1, 01H
02B8	A100D0	528	ax, _I
02BB	D0D8	529	rcr m500 ; If jagc is 00H then
02BD	7209	530	jc
02BF	C70600D00100	531	mov _2nd_decision, 01H ; first pass, don't set lock2.
02C5	EB2890	532	jmp m700
02C8	C7062CD00100	533	mov lock2, 01H
02CE	C70600D00000	534	mov _2nd_decision, 00H
02D0		535	
02D1		536	
02D2		537	
02D3		538	
02D4		539	
02D5		540	
02D6		541	
02D7		542	
02D8		543	
02D9		544	
02DA		545	

8086/87/88/186 MACRO ASSEMBLER CDU86

```

LOC  OBJ          LINE  SOURCE
02D4  A10AD0      546
02D7  A308D0      547
02DA  A10CD0      548
02DD  A312D0      549
02E0  EB0D90      550
02E3  C70600D00000 551
02E3  C70600D00000 552
02E9  C7052AD00000 553
02EF  C70616D00000 554
02F5  A13AD0       555
02F8  A318D0       556
02FB                                     557
02FB  9B           558
02FC  A200E0       559
02FF  C606009022  560
0304  C606009002  561
0309  8BD6        562
030B  A000A0      563
030E  9B         564
030F  2BD0        565
0311  9B         566
0312  C60600904E  567
0317  C606009000  568
031C  7002        569
031E  8BF2        570
0320  A000A0      571
0323  9B         572
0324  8BD0        573
0326  03C4        574
0328  7002        575
032A  8BE0        576
032C  03D5        577
032E  7002        578
0330  8BEA        579
0332                                     580
0332  A126D0      581
0335  85C07902F7D8 582
033B  BB8000      583
033E  3BD8        584
033E                                     585
033E                                     586
033E                                     587
033E                                     588
033E                                     589
033E                                     590
033E                                     591
033E                                     592
033E                                     593
033E                                     594
033E                                     595
033E                                     596
033E                                     597
033E                                     598
033E                                     599
033E                                     600

```

```

    mov ax, bsc1_in      ; Now we are out-of-lock so
    mov bsc1_ax, ax      ; bsc1 gets bsc1_in.
    mov ax, bsc1_in_rnd
    mov bsc1_rnd, ax     ; Same with round-off const.
    jmp m700

    mov _2nd_decision, 00H
    mov _lock1, 00H

    mov _ddcc1, 0       ; Load MSW with zero.
    mov ax, _escl_rnd   ; Load LSM with round-off const.
    mov _ddcc2, ax

    wait
    mov seu, al          ; Sync for error sample.
    mov cc0_1, 22H      ; Reset SEU
    mov cc0_1, 02H
    mov dx, si          ; "Load" eacc.
    mov al, atod        ; Get the error sample.
    mov dx, ax          ; Convert atod input into two's comp word.
    sub dx, ax          ; eacc - error_sample

    wait
    mov cc0_1, 4EH      ; Sync for data sample.
    mov cc0_1, 00H
    mov LX06
    mov si, dx          ; If sub overflowed then don't store.
    mov si, dx          ; "Store" eacc.

    mov al, atod        ; Get the data sample.
    mov dx, ax          ; Save copy for later.
    mov ax, sp          ; Add dacc.
    mov LY07            ; Don't store if overflow.
    mov sp, ax          ; "Store" dacc.

    add dx, bp          ; Add macc to copy saved.
    mov LZ08            ; Don't store if overflow.
    mov bp, dx          ; "Store" macc.

    mov ax, _icct       ; Second order sc tracking stuff.
    mov absval, ax     ;
    mov dx, B0H        ;
    mov bx, ax         ; If abs( icct ) >= RAGC then
    cmp

```

CDU86

8086/87/88/186 MACRO ASSEMBLER

LOC	OBJ	LINE	SOURCE
0340	A126D0	601	mov ax, ictt
0343	7908	602	jns m1000
0345	85C0	603	test ax, ax
0347	7902	604	jns m900
0349	F7DB	605	neg bx
034B	8BC3	606	mov ax, bx
034D	A326D0	607	mov ictt, ax
0350	051000	608	add ax, 1 shl (05H - 1) ; Add a round-off constant to ictt.
0353	B105	609	mov cl, 05H
0355	D3F8	610	sar ax, cl
0357	A324D0	611	mov ictt, ax
035A	892E42D0	612	mov _tmp, bp
035E	9B	613	; ** Sample.
035F	A200E0	614	wait
0362	C606009032	615	mov seu, al
0367	C606009002	616	mov cc0_1, 22H
036C	8BD6	617	mov cc0_1, 02H
036E	A000A0	618	mov dx, si
0371	98	619	mov al, atod
0372	2BD0	620	cbw dx, ax
0374	9B	621	sub dx, ax
0375	C60600904E	622	wait
037A	C606009000	623	mov mov cc0_1, 4EH
037F	7002	624	mov mov cc0_1, 00H
0381	8BF2	625	mov mov m1100
0383	A000A0	626	mov si, dx
0386	9B	627	mov al, atod
0387	8BE8	628	cbw bp, ax
0389	03C4	629	mov ax, sp
038B	7002	630	add m1200, ax
038D	8BE0	631	mov sp, ax
038F		632	; "Store" eacc.
0392	A04ED0	633	; Get the data sample.
0394	24EF	634	; First sample for mid-phase acc.
039A	A200C0	635	; Add dacc.
		636	; Don't store if overflow.
		637	; "Store" dacc.
		638	
		639	
		640	
		641	
		642	
		643	
		644	
		645	
		646	
		647	
		648	
		649	
		650	
		651	
		652	
		653	
		654	
		655	


```

LOC  OBJ          LINE  SOURCE
0397  8B1E1CD0    656
039B  8B3E1AD0    657
039F  A114D0      658
03A7  85C0         659
03A4  7304         660
03A6  98DB        661
03A8  33FF        662
03AA  33FB        663
03AC  A1A2D0      664
03AF  D1DB        665
03B1  721B        666
03B3  85FF        667
03B5  741F        668
03B7  8B1E30D0   669
03BB  A1A2D0      670
03BE  85C0         671
03C0  7402        672
03C2  F7DB        673
03C4  A12ED0      674
03C7  03C3        675
03C9  7003        676
03CB  A32ED0      677
03CE  A142D0      678
03D1  A330D0      679
03D4  9B          680
03D5  A200E0      681
03D8  C606009022 682
03DD  C606009002 683
03E2  8BD6        684
03E4  A000A0      685
03E7  98          686
03E8  2BD0        687
03EA  9B          688
03EB  C60600904E 689
03FD  C606009000 690
03F5  7002        691
03F7  8BF2        692
03F9

bx, _dlst
di, _dest
mov ax, _daccsave
test ax, ax
jnz m1300

bx
di
clear registers to force di to zero.

; If dlst == dest then di will be zero.
; If dlst == dest then di will be zero.

; If data is negative then
; m1st = - m1st

; m1st = m1st + m1st * sign-of-data.
; If addition does not cause overflow then
; store new m1st in RAM.

; M1st gets macc ( stored in tmp ).

; Sync for error sample.
; Reset SEU

; "Load" eacc.
; Get the error sample.
; Convert atod input into two's comp word.
; eacc = error_sample
; Sync for data sample.

; If sub overflowed then don't store.
; "Store" eacc.

```

8086/87/88/186 MACRO ASSEMBLER CD086

LOC	OBJ	LINE	SOURCE
03F9	A000A0	711 +1	mov al, atod
03FC	98	712 +1	cbw dx, ax
03FD	8BD0	713 +1	mov ax, sp
03FF	03C4	714 +2	add LY0A
0401	7002	715 +1	jo sp, ax
0403	8BE0	716 +1	mov dx, bp
0405	03D5	717 +1	add LZ0B
0407	7002	718 +2	jo bp, dx
0409	8BEA	719 +1	mov ax, I wait2
040B		720 +2	rcr m1600
040B	A14AD0	721 +1	mov wait2, 00H
040E	D1D8	722 +1	norm
0410	7309	723 +1	mov ax, I wait1
0412	C7064AD00000	724 +2	mov wait1, 00H
0418	E94BFD	725 +1	jmp norm
041B	A148D0	726 +2	mov ax, I wait1
041E	D1D8	727 +1	rcr m1700
0420	730C	728 +1	mov wait1, 00H
0422	C70648D00000	729 +1	mov wait2, 01H
0428	C7064AD00100	730 +1	mov norm
042E	85FF	731 +1	test di, di
0430	752B	732 +1	jnz m1900
0432	A14AD0	733 +1	mov ax, I tranct
0435	8B400	734 +1	mov bx, 64
0438	8BD8	735 +1	cmp dx, ax
043A	7807	736 +1	js m1600
043C	40	737 +1	inc ax
043D	A34AD0	738 +1	mov tranct, ax
0440	E923FD	739 +1	jmp norm
0443		740 +1	;
0443	A132D0	741 +1	;
0446	A33CD0	742 +1	;
0449	33C0	743 +1	;
044B	A32ED0	744 +1	;
044E	A304D0	745 +1	;
0451	C70648D000100	746 +1	;
0457	A344D0	747 +1	;
045A	E909FD	748 +1	;

CDU86

8086/87/88/186 MACRO ASSEMBLER

LOC	OBJ	LINE	SOURCE
045D	C70644D00000	766	
045E	A104D0	767	
0466	40	768	
0467	A304D0	769	
046A	3D0800	770	
046D	7409	771	
046F	E9F4FC	772	
		773	
		774	
		775	
		776	
		777	
0472	8B0E08D0	778	
0478	A12ED0	779	
0479	030612D0	780	
047D	D3F8	781	
047F	C7062ED00000	782	
0485	C70604D00000	783	
048B	C70648D00100	784	
0491	8B0E38D0	785	
0495	3BC1	786	
0497	7802	787	
		788	
0499	8BC1	789	
		790	
049B	8BD8	791	
049D	03D9	792	
049F	7904	793	
		794	
		795	
		796	
04A1	8BC1	797	
04A3	F7D8	798	
		799	
04A5	8B1E3CD0	800	
04A9	03C3	801	
04AB	D1E1	802	
04AD	3BC1	803	
04AF	7406	804	
		805	
04B1	A33CD0	806	
		807	
04B4	E9AFFC	808	
04B7		809	
		810	
		811	
		812	
		813	
		814	
04B7		815	
		816	
04B7	9B	817	
04B8	A200E0	818	
04BB	C606009022	819	
04C0	C606009022	820	

```

mov    tranct, 0
ax, _bcnt
inc    ax
mov    _bcnt, ax
ax, 08H
m2000
norm
jmp
; Bit transition detected.
; bcnt = bcnt + 1
; If bcnt is less then zero then

mov    cx, _bscl
ax, _mest
add    ax, _bscl_rnd
sar    ax, cl
mov    _mest, 0
mov    _bcnt, 0
mov    _wait1, 01H
mov    cx, _nsb4
mov    ax, _cx
cmp    ax, m2100
js
mov    ax, cx
; Scale mest by shifting right by bscl.
; Add roundoff const.
; Do the shift.
; Reset RAM version of mest.
; Reset bcnt to count 8 more bits.
; Compare scaled mest to nsb4.
; If mest > nsb4 then
;   mest gets nsb4.
; Move a copy of the scaled mest to bx.
; Add this copy to nsb4.
; If mest is less then minus nsb4 then
;   mest gets minus nsb4.
; Add mest ( scaled and limited ) to scnt.
; CX is now nsb2.
; If this new scnt is not equal to nsb2 then
;   Store new scnt in RAM and eat up cycles
;   in norm.

mov    bx, _scnt
add    ax, bx
sal    cx, 1
cmp    ax, cx
je    m2300
mov    _scnt, ax
jmp    norm
; Sync for error sample.
; Reset SEU
wait
mov    cc0_1, 22H
mov    cc0_1, 02H

```

8086/87/88/186 MACRO ASSEMBLER CDU86

LOC	OBJ	LINE	SOURCE
04C5	8BD6	821	dx, si
04C7	A000A0	822	al, atod
04CA	98	823	dx, ax
04CB	2BD0	824	dx, ax
04CD	9B	825	wait
04CE	C60600904E	826	cc0_1, 4EH
04D3	C606009000	827	cc0_1, 00H
04DB	7002	828	LX0C
04DA	8BF2	829	si, dx
04DC	A000A0	830	al, atod
04DF	98	831	dx, ax
04E0	8BD0	832	ax, sp
04E2	03C4	833	LY0D
04E4	7002	834	sp, ax
04E6	8BE0	835	dx, bp
04E8	03D5	836	LZ0E
04EA	7002	837	bp, dx
04EC	8BEA	838	eax, 00H
04EE		839	ax, dest
04EF	C70620D00000	840	ax, I
04F4	A1A00	841	wait
04F7	A31CD0	842	el00
04FA	A18D0	843	ax, _acnt
04FD	D1D8	844	ax, _acnt
04FF	7218	845	ax, 08H
0501	A102D0	846	ax, _acnt
0504	40	847	inc ax
0505	A302D0	848	mov ax, _acnt
0508	3D0800	849	cmp ax, 08H
050B	750C	850	jnc e100
050D	C70606D00100	851	bitm, 01H
0513	C70602D00000	852	_acnt, 0
0519		853	e100:
0519	9B	854	; ** Sample.
051A	A200E0	855	wait
051D	C6060090D4	856	mov ax, al
0522	C606009001	857	cc0_1, 0D4H
		858	cc0_1, 01H

8086/87/88/186 MACRO ASSEMBLER CDUB6

LOC	OBJ	LINE	SOURCE
0527	8BD6	876	
0529	A000A0	877	
052C	98	878	
052D	2BD0	879	
052F	98	880	
0530	C60600904E	881	
0535	C606009000	882	
053A	7002	883	
053C	8BF2	884	
053E	A000A0	885	
0541	98	886	
0542	8BD0	887	
0544	03C4	888	
0546	7002	889	
0548	8BE0	890	
054A	03D5	891	
054C	7002	892	
054E	8BEA	893	
0550		894	
0550		895	
0550		896	
0550		897	
0554	03D5	898	
0556	85E4	899	
0558	7902	900	
055A	F7DE	901	
055C	8B0E22D0	902	
0560	A148D0	903	
0563	D1D8	904	
0565	720D	905	
0567	013626D0	906	
056B	83C602	907	
056E	B102	908	
0570	D3FE	909	
0572	03D6	910	
0574	8B0800	911	
0577	2BC2	912	
0579	7902	913	
		914	
		915	
		916	
		917	
		918	
		919	
		920	
		921	
		922	
		923	
		924	
		925	
		926	
		927	
		928	
		929	
		930	

```

; "Load" eacc.
; Get the error sample.
; eacc - error_sample
; Sync for data sample.
; Useless load of clock/timer chip.
; Required to give atod time to work.
; "Store" eacc.
; Get the data sample.
; Save copy for later.
; Add dacc.
; Don't store if overflow.
; "Store" dacc.
; Add macc to copy saved.
; Don't store if overflow.
; "Store" macc.
; eacc >> escl
; Test dacc.
; If the sign of dacc is neg then
; negate the scaled error acc.
; If wait1 then use only ict1.
; ictt, si (02H - 1) ; Add the round-off const.
; Shift it right by ecf0. Si is now ict0.
; Add ict0 to ict1 to form sc_bump.
; Offset sc_bump to the range [0..scb_num]
; If sc_bump < 0 then

```

CDU86

8086/87/88/186 MACRO ASSEMBLER

```

LOC  OBJ          LINE          SOURCE
057B  33C0          931          ; make it zero.
057D          932          ax
057D  3D0F00        933          ; If sc_bump > scb_num then
0580  7803          934          e800
0582  B80F00        935          ;
0582          936          ax, 0FH ; make it scb_num.
0585          937          +1
0585  BB68F790      938          ;
0589  D7          939          +1
0589          940          bx, offset bump_table + 0F00H
0589          941          ; Fetch new clock count. Vic's instruction.
058A  A20090        942          ; Load clock timer chip with count from table.
058D  33C0          943          cc0_1, al
058D          944          ax gets sign of dacc.
058F  A20090        945          cc0_1, al
058F          946          Save dacc in RAM.
0592  892614D0      947          sp, i
0592          948          Shift the sign bit out of dacc.
0598  D1E4          949          rcl, i
0598          950          Rotate that bit into ax. This is now dest.
059A  A31AD0        951          mov dest, ax
059D  8BD8          952          mov ax, dest
059F  C70638D00100  953          ; It's not time to do lock.
05A5  9B          954          wait
05A6  C60600904E   955          cc0_1, 4EH
05AB  C606009000   956          cc0_1, 00H
05B0  9B          957          ; ** Sample.
05B0          958          wait
05B1  A200E0        959          mov seu, al
05B4  C606009022   960          cc0_1, 22H
05B9  C606009002   961          cc0_1, 02H
05BE  8B363AD0      962          si, escl_rnd
05C1  A000A0        963          al, atod
05C5  9B          964          ; Initialize eacc.
05C5          965          ; Get the error sample.
05C6  2BF0          966          sub si, ax
05C6          967          ; eacc - error_sample
05C8  9B          968          ; Sync for data sample.
05C9  C60600904E   969          cc0_1, 4EH
05CE  C606009000   970          cc0_1, 00H
05D3  A000A0        971          al, atod
05D6  9B          972          ; Get the data sample.
05D7  8BE0          973          cbw
05D9  03C5          974          mov sp, ax
05DB  7002          975          add ax, bp
05DB          976          ; Initialize dacc.
05DD  8BE8          977          ; Acc for mid-phase.
05DF          978          ; Store only on overflow.
05DF          979          bp, ax
05DF          980          ;
05DF          981          e900:
05DF          982          ; **
05DF          983          ; Output data at this point.
05DF          984          ;
05DF          985          ;

```

CDU86

8086/87/88/186 MACRO ASSEMBLER

LOC	OBJ	LINE	SOURCE
05DF	A1ZCD0	986	mov ax, lck2
05E2	D1D8	987	rcr ax, l
05E4	7320	988	jnc e1100
		989	
		990	
		991	+1
		992	+3
		993	+2
05E6	D0E3	994	shl bl, 1
		995	+4
		996	+3
05E8	D0E3	997	shl bl, 1
		998	+4
		999	+3
05EA	D0E3	1000	shl bl, 1
		1001	+4
		1002	+3
		1003	+1
05EC	A04CD0	1004	; Shift data bit into correct position. word into al.
05EF	24B7	1005	mov al, il
05F1	0AC3	1006	and al, not ((1 shl 06H) or (1 shl 06H))
05F3	8B0E2AD0	1007	or al, bl
05F7	D0D9	1008	mov cx, lck1
05F9	7302	1009	rcr cx, l
		1010	jnc e1000
		1011	or al, 1 shl 06H ; Otherwise set lck_1 bit in output.
		1012	
05FD	A24ED0	1013	mov output, al
0600	A200C0	1014	pio, al ; Save output word in RAM.
0603	EB1490	1015	jmp e1300 ; Write the output word to the pio port.
		1016	
		1017	
0606	A04DD0	1018	mov al, _olddr
0609	8B0E2AD0	1019	cx, lck1 ; Set the output word to the "out of lock"
060D	D0D9	1020	rcr cx, l ; Grab the status of lock one.
060F	7302	1021	jnc e1200 ; If the bit is clear,
		1022	skip over.
		1023	
0611	0C40	1024	or al, 1 shl 06H ; Otherwise set lck_1 bit in output.
		1025	
0613	A24ED0	1026	mov output, al
0616	A200C0	1027	pio, al ; word, i.e. just the dr.
		1028	pio, al ; Send this word out.
		1029	
0619	A14BD0	1030	mov ax, _wait1
061C	D1D8	1031	rcr ax, l
061E	7303	1032	jnc e1400 ; Check to see if we are waiting.
		1033	
		1034	
0620	EB2690	1035	jmp e1600 ; If we are waiting then don't do AGC.
		1036	
0623	8B1E18D0	1037	mov bx, _ddcc2 ; LSM of ddcc.
0627	A114D0	1038	mov ax, _daccsave ; Add dacc.
062A	85C07902F7D8	1039	absval ax
		1040	

CDU86

8086/87/88/186 MACRO ASSEMBLER

LOC	OBJ	LINE	SOURCE	ASSEMBLY	COMMENT
0630	8B1616D0	1041		dx, _ddcc1	; MSW of ddcc.
0634	03D8	1042		bx, ax	; If carry then
0636	7301	1043		ei1500	
0638	42	1044		dx	; add to MSW of ddcc.
0639	891E18D0	1045			
063D	891616D0	1046	e1500:	ddcc2, bx	; Store LSM in RAM.
0641	A106D0	1047		_ddcc1, dx	Same for MSW.
0644	D1D8	1048		ax, bit_m	; If not time to set the AGC then
0646	723A	1049		ax, 1	
0648		1050		ei1700	
0648		1051			
0648		1052			
0648		1053			
0648		1054			
0648		1055			
0648		1056			
0648		1057			
0648		1058			
0648		1059			
0648		1060			
0648		1061			
0648		1062			
0648		1063			
0648		1064			
0648		1065			
0648		1066			
0648		1067			
0648		1068			
0648		1069			
0648		1070			
0648		1071			
0648		1072			
0648		1073			
0648		1074			
0648		1075			
0648		1076			
0648		1077			
0648		1078			
0648		1079			
0648		1080			
0648		1081			
0648		1082			
0648		1083			
0648		1084			
0648		1085			
0648		1086			
0648		1087			
0648		1088			
0648		1089			
0648		1090			
0648		1091			
0648		1092			
0648		1093			
0648		1094			
0648		1095			
0649	A700E0	1096		wait	; Sync for error sample.
064C	C606009022	1097		mov ax, 22H	; Reset SEU
0651	C606009022	1098		mov cc0_1, 02H	
0656	8BD6	1099		mov dx, si	"Load" eacc.
0658	A000A0	1100		mov al, atod	Get the error sample.
065B	98	1101		cbw ax, atod	Convert atod input into two's comp word.
065C	2BD0	1102		sub dx, ax	eacc - error_sample
065E	98	1103		wait	; Sync for data sample.
065F	C60600904E	1104		mov cc0_1, 4EH	
0664	C606009000	1105		mov cc0_1, 00H	
0669	7002	1106		jo LX0F	; If sub overflowed then don't store.
066B	8BF2	1107		mov si, dx	; "Store" eacc.
066D	A000A0	1108	LX0F:	al, atod	; Get the data sample.
0670	98	1109		cbw dx, ax	; Save copy for later.
0671	8BD0	1110		mov ax, sp	Add dacc.
0673	03C4	1111		add LY10	Don't store if overflow.
0675	7002	1112		jo	
0677	8BE0	1113		mov sp, ax	; "Store" dacc.
0679	03D5	1114			
067B	7002	1115		add dx, bp	; Add macc to copy saved.
067D	8BEA	1116		jo LZ11	Don't store if overflow.
067F	E9E4FA	1117		mov bp, dx	; "Store" macc.
067F	E9E4FA	1118	LZ11:	norm	; take one more sample and go to norm.
0682	C70638D00000	1119		jmp	
0688	8B0E22D0	1120		mov ntd1, 00H	; It is time to do the lock detection.
068C	8BF9	1121		mov cx, _esc1	
		1122		mov di, _cx	
		1123			
		1124			
		1125			
		1126			
		1127			
		1128			
		1129			
		1130			
		1131			
		1132			
		1133			
		1134			
		1135			
		1136			
		1137			
		1138			
		1139			
		1140			
		1141			
		1142			
		1143			
		1144			
		1145			
		1146			
		1147			
		1148			
		1149			
		1150			
		1151			
		1152			
		1153			
		1154			
		1155			
		1156			
		1157			
		1158			
		1159			
		1160			
		1161			
		1162			
		1163			
		1164			
		1165			
		1166			
		1167			
		1168			
		1169			
		1170			
		1171			
		1172			
		1173			
		1174			
		1175			
		1176			
		1177			
		1178			
		1179			
		1180			
		1181			
		1182			
		1183			
		1184			
		1185			
		1186			
		1187			
		1188			
		1189			
		1190			
		1191			
		1192			
		1193			
		1194			
		1195			

CDU86

8086/87/88/186 MACRO ASSEMBLER

LOC	OBJ	LINE	SOURCE
068E	D3EB	1096	
0690	B91000	1097	
0693	2BCF	1098	
0695	D3E2	1099	
0697	0BDA	1100	
0699	891E18D0	1101	
069D	891E40D0	1102	
06A1	83EB7C	1103	
		1104	+1
		1105	+1
		1106	+1
		1107	+1
06A4	9B	1108	
06A5	A200E0	1109	
06A8	C606009022	1110	
06AD	C606009002	1111	
06B2	8BD6	1112	
06B4	A000A0	1113	
06B7	98	1114	
06B8	2BD0	1115	
		1116	+1
		1117	+2
		1118	+2
		1119	+2
		1120	+1
		1121	+1
		1122	+1
		1123	+2
		1124	+1
06C9	A000A0	1125	
06CC	98	1126	
06CD	8BD0	1127	
06CF	03C4	1128	
06D1	7002	1129	
		1130	+1
06D3	8BE0	1131	
		1132	+2
06D5	03D5	1133	
06D7	7002	1134	
		1135	+2
		1136	+1
06D9	8BEA	1137	
		1138	+2
06DB		1139	
		1140	+1
06DB	A1ZCD0	1141	
06DE	D1DB	1142	
06E0	7324	1143	
		1144	+1
06E2	B103	1145	
06E4	D3FB	1146	
06E6	A11ED0	1147	
06E9	2BC3	1148	
06EB	A31ED0	1149	
06EE	790F	1150	

```

bx, cl
cx, 16
dx, di
dx, cl
bx, dx
ddcc2, bx
snr_check, bx
bx, 780H - 04H

;
; bx is ddcc2, 16-bit version of ddcc.
; Gets scaled ddcc for use in lock detection.
; This is the SNR word.

;
; Sync for error sample.
; Reset SEU
;
; "Load" eacc.
; Get the error sample.
; Convert atod input into two's comp word.
; eacc - error_sample
;
; Sync for data sample.
;
; If sub overflowed then don't store.
;
; "Store" eacc.
;
; Get the data sample.
;
; Save copy for later.
; Add dacc.
; Don't store if overflow.
;
; "Store" dacc.
;
; Add macc to copy saved.
; Don't store if overflow.
;
; "Store" macc.
;
; If not in lock then gain up.
;
; Now bx is en.
; enn = enn - en
;
; If enn is less than zero then

```

CDU86

8086/87/88/186 MACRO ASSEMBLER

LOC	OBJ	LINE	SOURCE	ASSEMBLY	COMMENT
06F0	C7061ED00000	1151		mov ax, 0	;; enn gets zero.
06F6	C70600B00100	1152		mov agc, 1	;; Agc word gets 1.
06FC	E567FA	1153		jmp norm	
06FF	B97F02	1154	e1800: +1	mov cx, 027FH	;; Compare enn to ennmax.
0702	3BC8	1155		cmp cx, ax	;; If enn is larger than ennmax
0704	790F	1156		jns e2000	
0706	C7061ED07F02	1157	e1900: +1	mov ax, 027FH	;; enn gets ennmax and
070C	C70600B0F803	1158		mov agc, 03F8H	;; set for max gain.
0712	E951FA	1159		jmp norm	;; goto norm.
0715	33D2	1160	e2000: dx	clr dx	;; Now ax is enn and in the range [0..ennmax]
0717	D1E8	1161		shr ax, 1	;; Shift right into carry.
0719	D1DA	1162		rcr dx, 1	;; Rotate right from carry.
071B	D1E8	1163		shr ax, 1	;; Shift right into carry.
071D	D1DA	1164		rcr dx, 1	;; Rotate right from carry.
071F	D1E8	1165		shr ax, 1	;; Shift right into carry.
0721	D1DA	1166		rcr dx, 1	;; Rotate right from carry.
0723	D1E8	1167		shr ax, 1	;; Shift right into carry.
0725	D1DA	1168		rcr dx, 1	;; Rotate right from carry.
0727	D1E8	1169		shr ax, 1	;; Shift right into carry.
0729	D1DA	1170		rcr dx, 1	;; Rotate right from carry.
072B	D1E8	1171		shr ax, 1	;; Shift right into carry.
072D	D1DA	1172		rcr dx, 1	;; Rotate right from carry.
072F	B91000	1173		mov cx, 16	;; 16 - enn is now the shift count.
0732	2BC8	1174		sub ax, ax	;; A one goes in ax.
0734	B80100	1175		mov di, cx	;; Shift (ax, dx) >> cl. Save shift count.
0737	8BF9	1176		shr dx, cl	;; Shift LS word by count.
0739	D3EA	1177		shr cx, 16	;; 16 - shift count goes into cx.
073B	B91000	1178		mov cx, di	;; Shift the MS word the OTHER DIRECTION!
073E	2BC8	1179		sub ax, ax	;; Put it together result is in dx.
0740	D3E0	1180		shr dx, cl	;; Now dx is the (10-Bit) agc control word.
0742	0BD0	1181		or agc, dx	
0744	B91600B0	1182		mov norm	
0748	E91BFA	1183		jmp norm	

LOC	OBJ	LINE	SOURCE
074B	90	1206	;
074C	0900	1207	;
074E	0A00	1208	;
0750	0C00	1209	;
0752	0C00	1210	;
0754	0B00	1211	;
0756	0B00	1212	;
0758	0B00	1213	;
075A	0600	1214	;
075C	0700	1215	;
075E	0900	1216	;
0760	0900	1217	;
0762	0B00	1218	;
0764	0B00	1219	;
0766	0B00	1220	;
0768	76	1221	;
0769	71	1222	;
076A	6C	1223	;
076B	67	1224	;
076C	62	1225	;
076D	5D	1226	;
076E	58	1227	;
076F	53	1228	;
0770	4E	1229	;
0771	49	1230	;
0772	44	1231	;
0773	3F	1232	;
0774	3A	1233	;
0775	35	1234	;
0776	30	1235	;
0777	2B	1236	;
0778	0900	1237	;
077A	0900	1238	;
077C	0800	1239	;
077E	0800	1240	;
0780	0700	1241	;
0782	0600	1242	;
0784	0600	1243	;
0786	BE00	1244	;
0788	B400	1245	;
078A	B400	1246	;
078C	7B00	1247	;
078E	A600	1248	;
		1249	;
		1250	;
		1251	;
		1252	;
		1253	;
		1254	;
		1255	;
		1256	;
		1257	;
		1258	;
		1259	;
		1260	;

Look-up tables for data rate dependent numbers.

```

even
bscl_in_table
    9 10
    7 12
    9 12
    11 11
    11 11
    11 11
    7 8125 bps. Selected 12-11-86 to
    9 15.625 bps. accommodate doppler
    7 31.25 bps. tracking....
    9 67.5 bps.
    11 125 bps.
    11 250 bps.
    11 500 bps.

```

```

bscl_out_table
    6 7
    7 7
    9 9
    11 11
    11 11
    11 11
    7 8125 bps. Selected 12-11-86 to
    9 15.625 bps. accommodate doppler
    7 31.25 bps. tracking....
    9 67.5 bps.
    11 125 bps.
    11 250 bps.
    11 500 bps.

```

```

bump_table
    118 db
    108 db
    103 db
    98 db
    93 db
    88 db
    83 db
    78 db
    73 db
    68 db
    63 db
    58 db
    53 db
    48 db
    43 db

```

Table center is 78 clock cycles.

```

even
escl_table
    9 9
    8 8
    8 8
    7 7
    6 6
    7 8125 bps.

```

```

1kth_table
    190 dw
    132 dw
    180 dw
    123 dw
    166 dw
    7 8125 bps.
    8 15.625 bps.
    8 31.25 bps.
    7 67.5 bps.
    6 125 bps.
    190 10/29/86
    132 10/29/86
    180 10/29/86
    123 10/29/86
    166 10/29/86

```

CDU86

8086/87/88/186-MACRO ASSEMBLER

LOC	OBJ	LINE	SOURCE			
0790	D500	1261		dw	213	
0792	B200	1262		dw	130	250 bps. 300 bps.
0794	0002	1263				10/29/86 10/29/86
0796	0001	1264	nsb2_table	dw	512	
0798	8000	1265		dw	256	
079A	4000	1266		dw	128	
079C	2000	1267		dw	64	
079E	1000	1268		dw	32	
07A0	0800	1269		dw	16	
07A2	4C00	1271		dw	8	
07A4	4800	1272				500 bps.
07A6	4800	1273	ulkth_table	dw	76	7.8125 bps. 15.625 bps. 31.25 bps. 62.5 bps. 125 bps. 250 bps. 500 bps.
07A8	4400	1274		dw	70	10/29/86 10/29/86 10/29/86 10/29/86 10/29/86 10/29/86 10/29/86
07AA	4600	1275		dw	72	
07AC	4600	1276		dw	68	
07AE	4400	1277		dw	70	
07AF	4400	1278		dw	68	
07B0		1279				
07B1		1280	org 0FF0h			
07B2		1281	jmp			
07B3		1282		start		
07B4		1283				
07B5		1284	abs_seg ends			
07B6		1285	end			
07B7		1286				
07B8		1287				
07B9		1288				
07BA		1289				
07BB		1290				
07BC		1291				
07BD		1292				
07BE		1293				
07BF		1294				
07C0		1295				
07C1		1296				
07C2		1297				
07C3		1298				
07C4		1299				
07C5		1300				
07C6		1301				
07C7		1302				
07C8		1303				
07C9		1304				
07CA		1305				
07CB		1306				
07CC		1307				
07CD		1308				
07CE		1309				
07CF		1310				
07D0		1311				
07D1		1312				
07D2		1313				
07D3		1314				
07D4		1315				
07D5		1316				
07D6		1317				
07D7		1318				
07D8		1319				
07D9		1320				
07DA		1321				
07DB		1322				
07DC		1323				
07DD		1324				
07DE		1325				
07DF		1326				
07E0		1327				
07E1		1328				
07E2		1329				
07E3		1330				
07E4		1331				
07E5		1332				
07E6		1333				
07E7		1334				
07E8		1335				
07E9		1336				
07EA		1337				
07EB		1338				
07EC		1339				
07ED		1340				
07EE		1341				
07EF		1342				
07F0		1343				
07F1		1344				
07F2		1345				
07F3		1346				
07F4		1347				
07F5		1348				
07F6		1349				
07F7		1350				
07F8		1351				
07F9		1352				
07FA		1353				
07FB		1354				
07FC		1355				
07FD		1356				
07FE		1357				
07FF		1358				
E90DF0						

SYMBOL TABLE LISTING

NAME	TYPE	VALUE	ATTRIBUTES
2ND_DECISION	V WORD	D000H	ABS_SEG
ACNT	V WORD	D002H	ABS_SEG
BCNT	V WORD	D004H	ABS_SEG
BIT_M	V WORD	D006H	ABS_SEG
BSCL	V WORD	D008H	ABS_SEG
BSCL_IN	V WORD	D00AH	ABS_SEG
BSCL_IN_RND	V WORD	D00CH	ABS_SEG
BSCL_OUT	V WORD	D00EH	ABS_SEG
BSCL_OUT_RND	V WORD	D010H	ABS_SEG
BSCL_RND	V WORD	D012H	ABS_SEG
DACCSAVE	V WORD	D014H	ABS_SEG
DDCC1	V WORD	D016H	ABS_SEG
DDCC2	V WORD	D018H	ABS_SEG
DEST	V WORD	D01AH	ABS_SEG
DLST	V WORD	D01CH	ABS_SEG
ENN	V WORD	D01EH	ABS_SEG
EOBMB	V WORD	D020H	ABS_SEG
ESCL	V WORD	D022H	ABS_SEG
ESCL_RND	V WORD	D024H	ABS_SEG
ICT1	V WORD	D026H	ABS_SEG
IL	V WORD	D028H	ABS_SEG
LKTH	V WORD	D02AH	ABS_SEG
LOCK1	V WORD	D02CH	ABS_SEG
LOCK2	V WORD	D02EH	ABS_SEG
MEST	V WORD	D030H	ABS_SEG
MLST	V WORD	D032H	ABS_SEG
NOTRAN	V WORD	D034H	ABS_SEG
NSB2	V WORD	D036H	ABS_SEG
NSB4	V WORD	D038H	ABS_SEG
NTDL	V WORD	D040H	ABS_SEG
OLDDR	V WORD	D042H	ABS_SEG
OUTPUT	V WORD	D044H	ABS_SEG
SCNT	V WORD	D046H	ABS_SEG
SIO_NOT_DONE	V WORD	D048H	ABS_SEG
SNR_CHECK	V WORD	D04AH	ABS_SEG
SOUTPUT	V WORD	D04CH	ABS_SEG
TMP	V WORD	D04EH	ABS_SEG
TRANCT	V WORD	D050H	ABS_SEG
ULKTH	V WORD	D052H	ABS_SEG
WAIT1	V WORD	D054H	ABS_SEG
WAIT2	V WORD	D056H	ABS_SEG
775SEG	SEGMENT	D058H	ABS_SEG
ABS_SEG	SEGMENT	D05AH	ABS_SEG
ABSYAL	MACRO	D05CH	ABS_SEG
AGC	V WORD	D05EH	ABS_SEG
ATOD	V WORD	D060H	ABS_SEG
BSCL_IN_TABLE	V WORD	D062H	ABS_SEG
BSCL_OUT_TABLE	V WORD	D064H	ABS_SEG
BUMP_TABLE	V WORD	D066H	ABS_SEG
CCO_I	V WORD	D068H	ABS_SEG
		D06AH	ABS_SEG
		D06CH	ABS_SEG
		D06EH	ABS_SEG
		D070H	ABS_SEG
		D072H	ABS_SEG
		D074H	ABS_SEG
		D076H	ABS_SEG
		D078H	ABS_SEG
		D07AH	ABS_SEG
		D07CH	ABS_SEG
		D07EH	ABS_SEG
		D080H	ABS_SEG
		D082H	ABS_SEG
		D084H	ABS_SEG
		D086H	ABS_SEG
		D088H	ABS_SEG
		D08AH	ABS_SEG
		D08CH	ABS_SEG
		D08EH	ABS_SEG
		D090H	ABS_SEG
		D092H	ABS_SEG
		D094H	ABS_SEG
		D096H	ABS_SEG
		D098H	ABS_SEG
		D09AH	ABS_SEG
		D09CH	ABS_SEG
		D09EH	ABS_SEG
		D0A0H	ABS_SEG
		D0A2H	ABS_SEG
		D0A4H	ABS_SEG
		D0A6H	ABS_SEG
		D0A8H	ABS_SEG
		D0AAH	ABS_SEG
		D0ACH	ABS_SEG
		D0AEH	ABS_SEG
		D0A0H	ABS_SEG
		D0A2H	ABS_SEG
		D0A4H	ABS_SEG
		D0A6H	ABS_SEG
		D0A8H	ABS_SEG
		D0AAH	ABS_SEG
		D0ACH	ABS_SEG
		D0AEH	ABS_SEG
		D0B0H	ABS_SEG
		D0B2H	ABS_SEG
		D0B4H	ABS_SEG
		D0B6H	ABS_SEG
		D0B8H	ABS_SEG
		D0BAH	ABS_SEG
		D0BCH	ABS_SEG
		D0BEH	ABS_SEG
		D0B0H	ABS_SEG
		D0B2H	ABS_SEG
		D0B4H	ABS_SEG
		D0B6H	ABS_SEG
		D0B8H	ABS_SEG
		D0BAH	ABS_SEG
		D0BCH	ABS_SEG
		D0BEH	ABS_SEG
		D0C0H	ABS_SEG
		D0C2H	ABS_SEG
		D0C4H	ABS_SEG
		D0C6H	ABS_SEG
		D0C8H	ABS_SEG
		D0CAH	ABS_SEG
		D0CBH	ABS_SEG
		D0CEH	ABS_SEG
		D0D0H	ABS_SEG
		D0D2H	ABS_SEG
		D0D4H	ABS_SEG
		D0D6H	ABS_SEG
		D0D8H	ABS_SEG
		D0DAH	ABS_SEG
		D0DBH	ABS_SEG
		D0DEH	ABS_SEG
		D0E0H	ABS_SEG
		D0E2H	ABS_SEG
		D0E4H	ABS_SEG
		D0E6H	ABS_SEG
		D0E8H	ABS_SEG
		D0EAH	ABS_SEG
		D0EBH	ABS_SEG
		D0EDH	ABS_SEG
		D0E0H	ABS_SEG
		D0E2H	ABS_SEG
		D0E4H	ABS_SEG
		D0E6H	ABS_SEG
		D0E8H	ABS_SEG
		D0EAH	ABS_SEG
		D0EBH	ABS_SEG
		D0EDH	ABS_SEG
		D0F0H	ABS_SEG
		D0F2H	ABS_SEG
		D0F4H	ABS_SEG
		D0F6H	ABS_SEG
		D0F8H	ABS_SEG
		D0FAH	ABS_SEG
		D0FBH	ABS_SEG
		D0FDH	ABS_SEG
		D0F0H	ABS_SEG
		D0F2H	ABS_SEG
		D0F4H	ABS_SEG
		D0F6H	ABS_SEG
		D0F8H	ABS_SEG
		D0FAH	ABS_SEG
		D0FBH	ABS_SEG
		D0FDH	ABS_SEG
		D000H	ABS_SEG
		A000H	ABS_SEG
		074CH	ABS_SEG
		075AH	ABS_SEG
		0768H	ABS_SEG
		9000H	ABS_SEG

NAME	TYPE	VALUE	ATTRIBUTES
CC0_2	V BYTE	8800H	ABS SEG
CC1_1	V BYTE	9002H	ABS SEG
CC1_2	V BYTE	8802H	ABS SEG
CC2_1	V BYTE	9004H	ABS SEG
CC2_2	V BYTE	8804H	ABS SEG
CCW_1	V BYTE	9006H	ABS SEG
CCW_2	V BYTE	8806H	ABS SEG
CLR_*	C MACRO	#DEFS=2	
E100_*	L NEAR	0519H	ABS SEG
E1000_*	L NEAR	05FDH	ABS SEG
E1100_*	L NEAR	0606H	ABS SEG
E1200_*	L NEAR	0613H	ABS SEG
E1300_*	L NEAR	0619H	ABS SEG
E1400_*	L NEAR	0623H	ABS SEG
E1500_*	L NEAR	0639H	ABS SEG
E1600_*	L NEAR	0648H	ABS SEG
E1700_*	L NEAR	0682H	ABS SEG
E1800_*	L NEAR	06FFH	ABS SEG
E1900_*	L NEAR	0706H	ABS SEG
E200_*	L NEAR	053EH	ABS SEG
E2000_*	L NEAR	0715H	ABS SEG
E300_*	L NEAR	054AH	ABS SEG
E400_*	L NEAR	0550H	ABS SEG
E500_*	L NEAR	055CH	ABS SEG
E600_*	L NEAR	0574H	ABS SEG
E700_*	L NEAR	057DH	ABS SEG
E800_*	L NEAR	05E5H	ABS SEG
E900_*	L NEAR	05DFH	ABS SEG
EOB_*	L NEAR	04B7H	ABS SEG
FSC_L_TABLE_*	V WORD	0778H	ABS SEG
I300_*	L NEAR	0080H	ABS SEG
I3000_*	L NEAR	008FH	ABS SEG
INT_*	L NEAR	008FH	ABS SEG
LRTN_TABLE_*	V WORD	0786H	ABS SEG
LX00_*	L NEAR	018BH	ABS SEG
LX03_*	L NEAR	0239H	ABS SEG
LX06_*	L NEAR	0320H	ABS SEG
LX09_*	L NEAR	03F9H	ABS SEG
LX0C_*	L NEAR	04DCH	ABS SEG
LX0F_*	L NEAR	066DH	ABS SEG
LX17_*	L NEAR	06C2H	ABS SEG
LY01_*	L NEAR	0177H	ABS SEG
LY04_*	L NEAR	0245H	ABS SEG
LY07_*	L NEAR	032CH	ABS SEG
LY0A_*	L NEAR	0405H	ABS SEG
LY0D_*	L NEAR	04E8H	ABS SEG
LY10_*	L NEAR	0679H	ABS SEG
LY13_*	L NEAR	06D5H	ABS SEG
LZ02_*	L NEAR	019DH	ABS SEG
LZ05_*	L NEAR	024BH	ABS SEG
LZ08_*	L NEAR	0332H	ABS SEG
LZ0B_*	L NEAR	040BH	ABS SEG
LZ0E_*	L NEAR	04EEH	ABS SEG
LZ11_*	L NEAR	067FH	ABS SEG
LZ14_*	L NEAR	06DBH	ABS SEG
M100_*	L NEAR	025BH	ABS SEG

NAME	TYPE	VALUE	ATTRIBUTES
M1000	L NEAR	034DH	ABS_SEG
M1100	L NEAR	038FH	ABS_SEG
M1200	L NEAR	038FH	ABS_SEG
M1300	L NEAR	03AAH	ABS_SEG
M1400	L NEAR	03C4H	ABS_SEG
M1500	L NEAR	03CEH	ABS_SEG
M1600	L NEAR	0412H	ABS_SEG
M1700	L NEAR	0442H	ABS_SEG
M1800	L NEAR	0443H	ABS_SEG
M1900	L NEAR	045DH	ABS_SEG
M2000	L NEAR	0281H	ABS_SEG
M2100	L NEAR	0472H	ABS_SEG
M2200	L NEAR	049EH	ABS_SEG
M2300	L NEAR	04A5H	ABS_SEG
M3000	L NEAR	04B7H	ABS_SEG
M4000	L NEAR	0298H	ABS_SEG
M5000	L NEAR	02A7H	ABS_SEG
M6000	L NEAR	02C8H	ABS_SEG
M7000	L NEAR	02E3H	ABS_SEG
M8000	L NEAR	02FFH	ABS_SEG
M9000	L NEAR	02F8H	ABS_SEG
MB	L NEAR	034BH	ABS_SEG
M100	L NEAR	0214H	ABS_SEG
M200	L NEAR	01C1H	ABS_SEG
M300	L NEAR	01CEH	ABS_SEG
M400	L NEAR	01EEH	ABS_SEG
M500	L NEAR	01FEH	ABS_SEG
NOBM	L NEAR	0164H	ABS_SEG
RSB2_TABLE	V WORD	0774H	ABS_SEG
P10	V BYTE	C000H	ABS_SEG
SEU	V BYTE	E000H	ABS_SEG
S10	V BYTE	8008H	ABS_SEG
SNR	V BYTE	8020H	ABS_SEG
START	L NEAR	0000H	ABS_SEG
TLM	L NEAR	8010H	ABS_SEG
ULKTH_TABLE	V WORD	07A2H	ABS_SEG

END OF SYMBOL TABLE LISTING

ASSEMBLY COMPLETE, NO ERRORS FOUND

SECTION 11

TEST RESULTS

To verify the design of the CDU, a large battery of tests were conducted. The matrix of tests run is given in Table 11.0-1. The tests were used to show that the CDU meets or exceeds the performance bounds specified in [11-1]. The tests conducted were designed to determine the following quantities: Bit Error Rate (BER), Probability of False Acquisition, Probability of De-Acquisition, Probability of Out-of-Lock, Probability of Acquisition, Sub-carrier Jitter, Bit Sync Jitter, and Probability of Cycle Slip.

11.1 BER TESTS

The BER tests made up the majority of the testing. The results for each data rate are shown in Figures 11.1-1 to 11.1-7. From the figures, the following observations may be made:

- (1) The degradation from ideal increases as ST/N_0 decreases. This is due to the fact that the AGC scaling was designed to prevent the noise from saturating the ADC at an ST/N_0 of 10.5 dB; at lower ST/N_0 's, the noise can saturate the ADC easier and cause performance degradation.
- (2) Even with the maximum doppler rate, the CDU operates inside the 1.1 dB envelope the design requirements allow. In general, the CDU is around 0.5 dB off of theory.

11.2 PROBABILITY OF FALSE ACQUISITION

The probability of false acquisition is the probability that the CDU will experience two consecutive eight-bit periods in which the lock detector value will be above the lock threshold. Since each eight-bit period is independent of all the others, the requirement is also valid for a limit on the single eight-bit period. The requirement that the CDU must meet is that the probability for a single period must be less than 1.0×10^{-2} .

The tests were run, with an input noise voltage of $2.5 V_{rms}$ for each data rate; no false acquisitions were observed. To provide a one-sided 90% confidence interval for these measurements, we use the following formula [11-2]:

$$p = \frac{-1}{n} \times \ln(1 - c) \quad (11.2-1)$$

Table 11.0-1. CDU Breadboard Test Matrix

Test	Doppler	Data Rate	Data Rate							V_s mV _{rms}
			500	250	125	62.5	31.25	15.625	7.8125	
BER/P(OOL)										
10 dB	0	PN	X	X	X	X	X	X	X	50
9 dB	0	PN	X	X	X	X	X	X	X	100
8 dB	0	PN	X	X	X	X	X	X	X	200
7 dB	0	PN	X	X	X	X	X	X	X	300
10 dB	RATE	PN	X	X	X	X	X	X	X	50
9 dB	RATE	PN	X	X	X	X	X	X	X	100
8 dB	RATE	PN	X	X	X	X	X	X	X	200
7 dB	RATE	PN	X	X	X	X	X	X	X	300
10 dB	+	PN	X	X	X	X	X	X	X	50
9 dB	+	PN	X	X	X	X	X	X	X	100
8 dB	+	PN	X	X	X	X	X	X	X	200
7 dB	+	PN	X	X	X	X	X	X	X	300
10 dB	-	PN	X	X	X	X	X	X	X	50
9 dB	-	PN	X	X	X	X	X	X	X	100
8 dB	-	PN	X	X	X	X	X	X	X	200
7 dB	-	PN	X	X	X	X	X	X	X	300
PROB ACQ.										
10 dB	+	ALT	X	X		X	X		X	50
10 dB	RATE	ALT	X	X	X	X	X		X	50
10 dB	-	ALT	X		X		X	X	X	300
9 dB	+	ALT	X				X		X	100
8 dB	-	ALT	X				X		X	200
7 dB	+	ALT	X			X	X		X	300
P(FALSE LOCK)	0	DNA	X	X	X	X	X	X	X	$V_N=2.5V$
SC JITTER										
10 dB	0	PN	X				X			50
10 dB	0	PN	X				X			300
BS JITTER										
10 dB	0	PN	X				X			50
10 dB	0	PN	X				X			300

Table 11.0-1. CDU Breadboard Test Matrix (Continued)

Test	Doppler	Data Rate	Data Rate							V_s mV _{rms}
			500	250	125	62.5	31.25	15.625	7.8125	
DROP LOCK TIME										
10.5 dB	0	PN	X	X	X	X	X	X	X	50
10.5 dB	0	PN	X	X	X	X	X	X	X	300
P(CYCLE SLIP)										
6 dB	0	PN	X							200
7 dB	0	PN	X							200
8 dB	0	PN	X							200
9 dB	0	PN	X							200
6 dB	+	PN	X							200
7 dB	+	PN	X							200
8 dB	+	PN	X							200
6 dB	-	PN	X							200
7 dB	-	PN	X							200
8 dB	-	PN	X							200
6 dB	RATE	PN	X							200
7 dB	RATE	PN	X							200
8 dB	RATE	PN	X							200

Where

RATE => Doppler rate

+ => + Offset

- => - Offset

0 => No Doppler

PN => Pseudo-noise sequence

ALT => Alternating Ones-Zeros

DNA => Does not apply

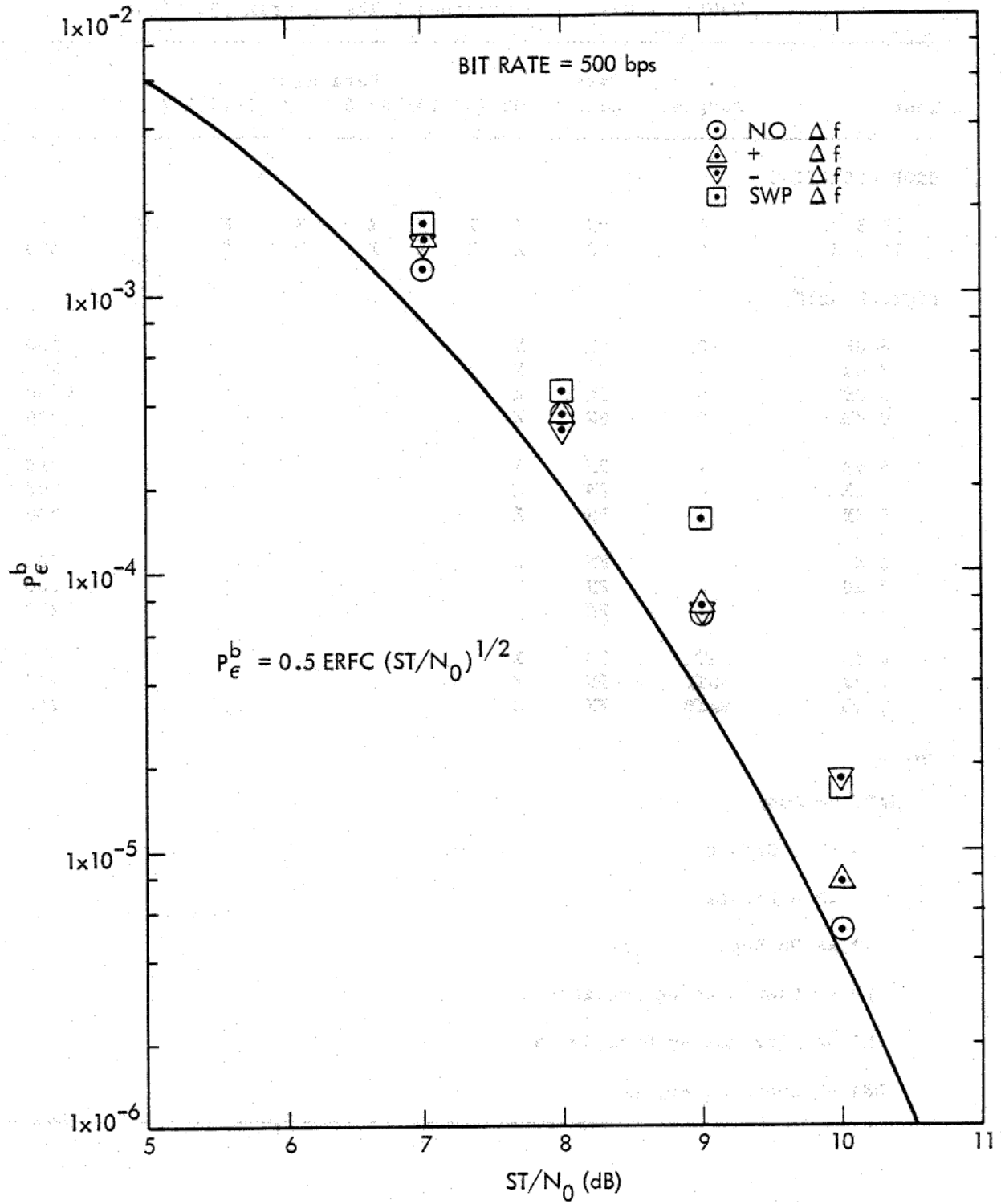


Figure 11.1-1. BER Test Data Summary - Bit Rate = 500 bps

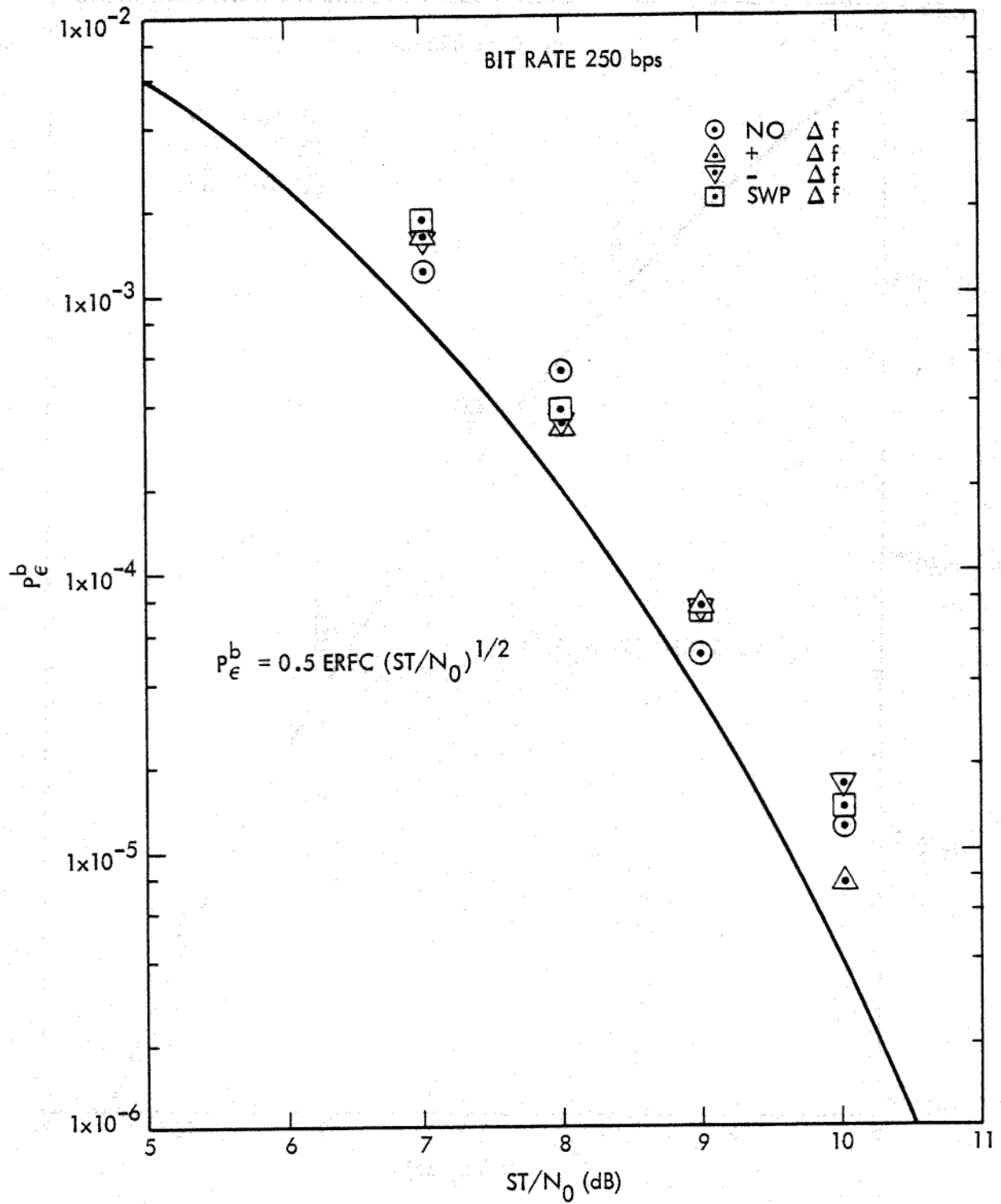


Figure 11.1-2. BER Test Data Summary - Bit Rate = 250 bps

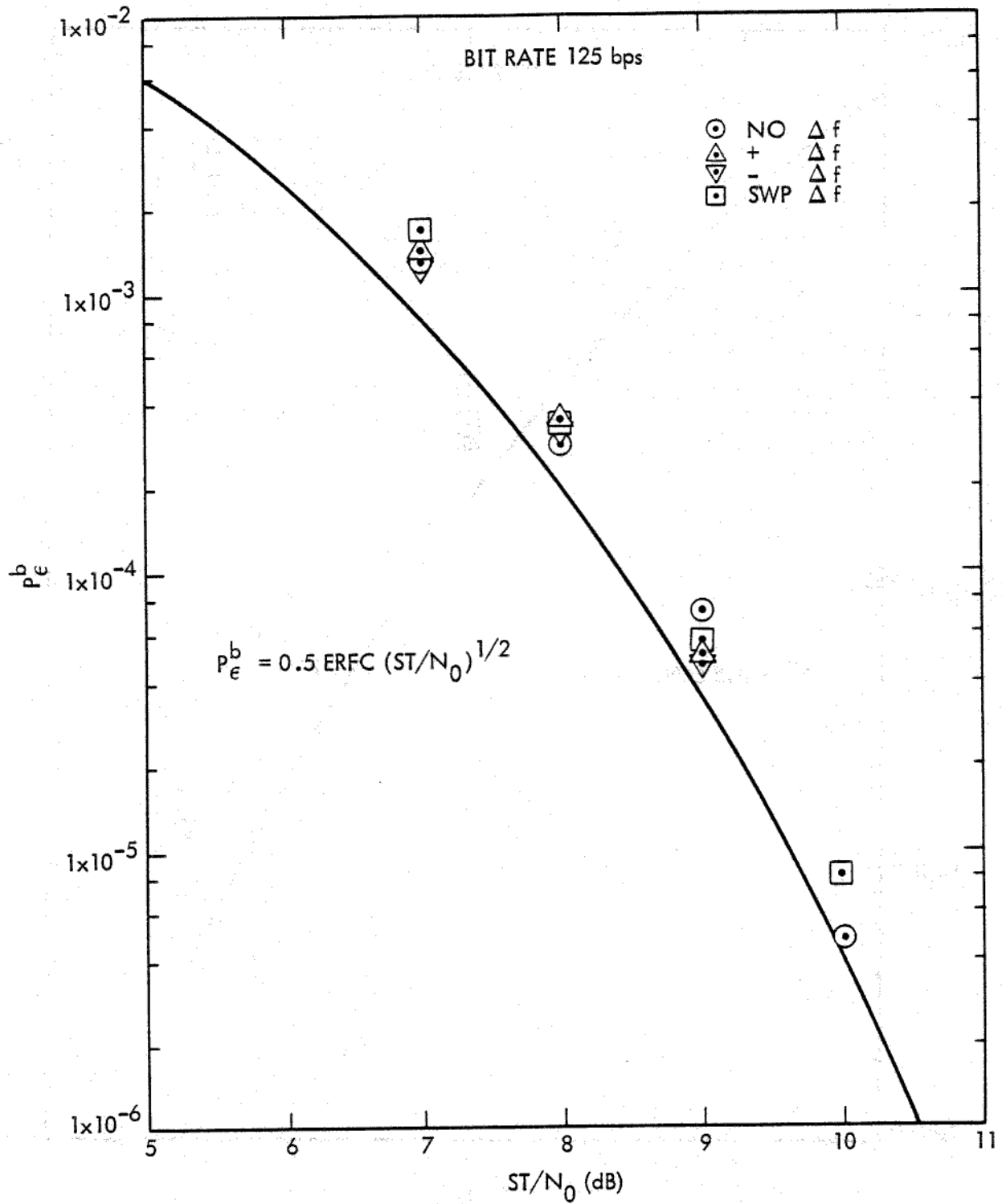


Figure 11.1-3. BER Test Data Summary - Bit Rate = 125 bps

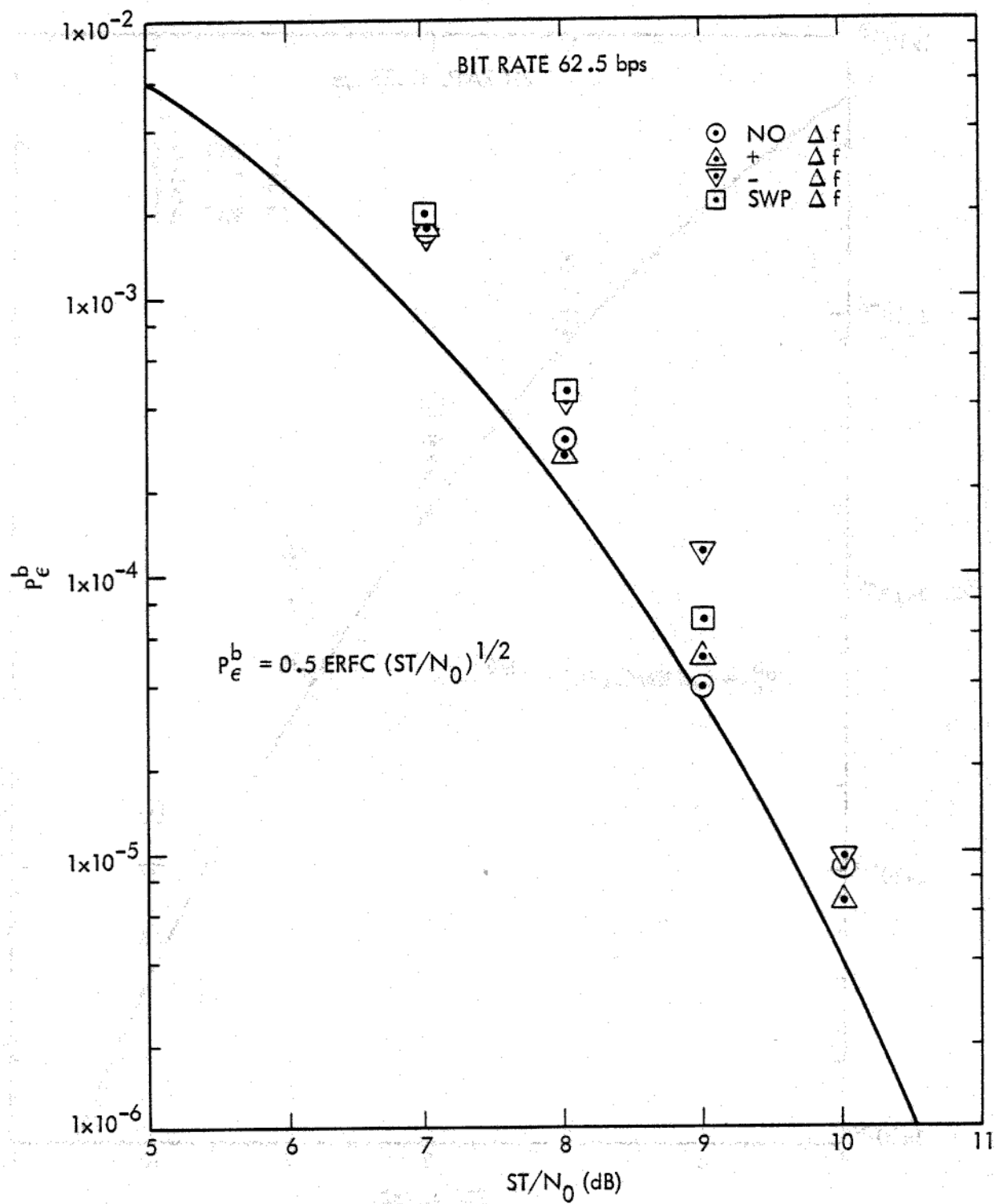


Figure 11.1-4. BER Test Data Summary - Bit Rate = 62.5 bps

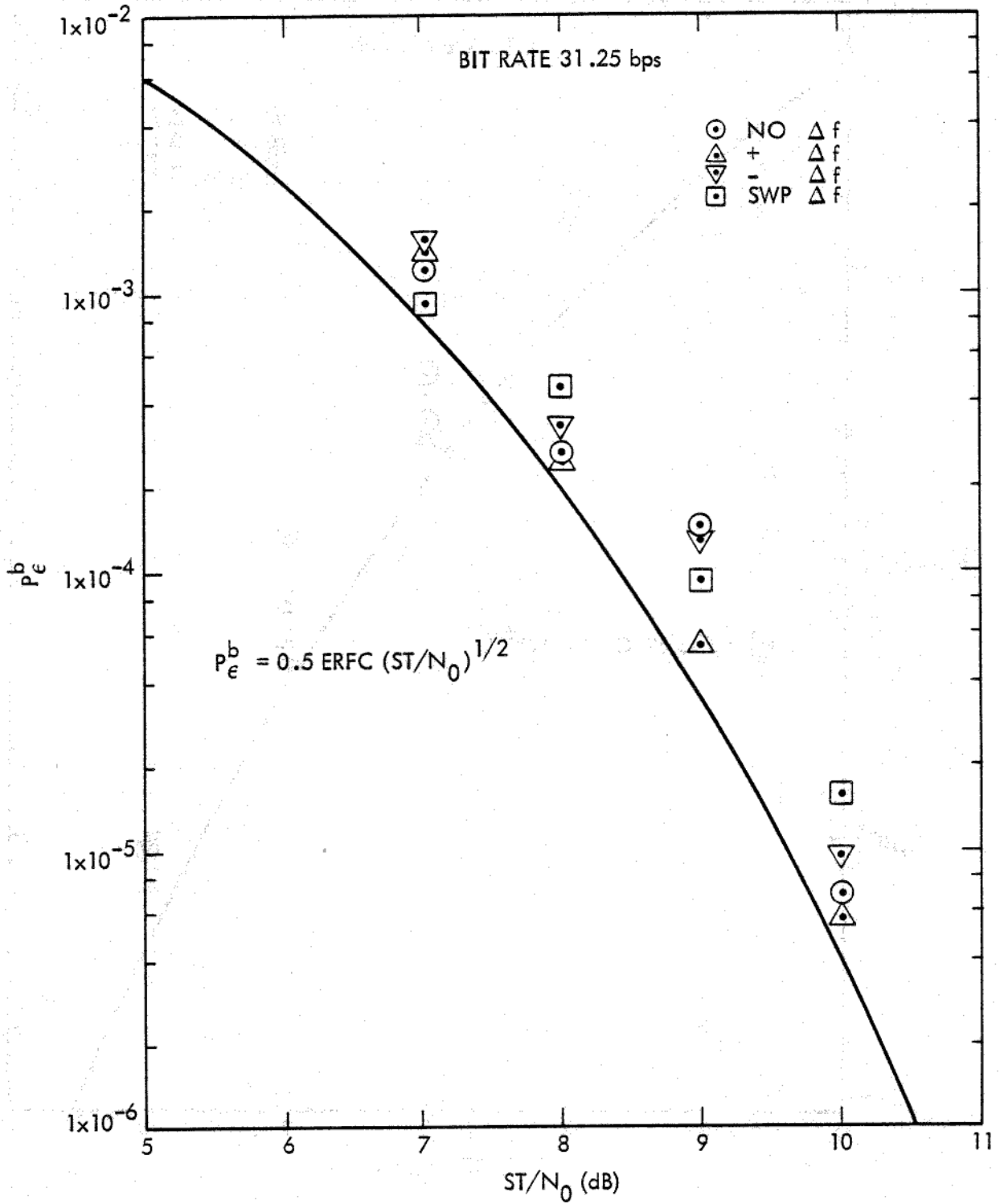


Figure 11.1-5. BER Test Data Summary - Bit Rate = 31.25 bps

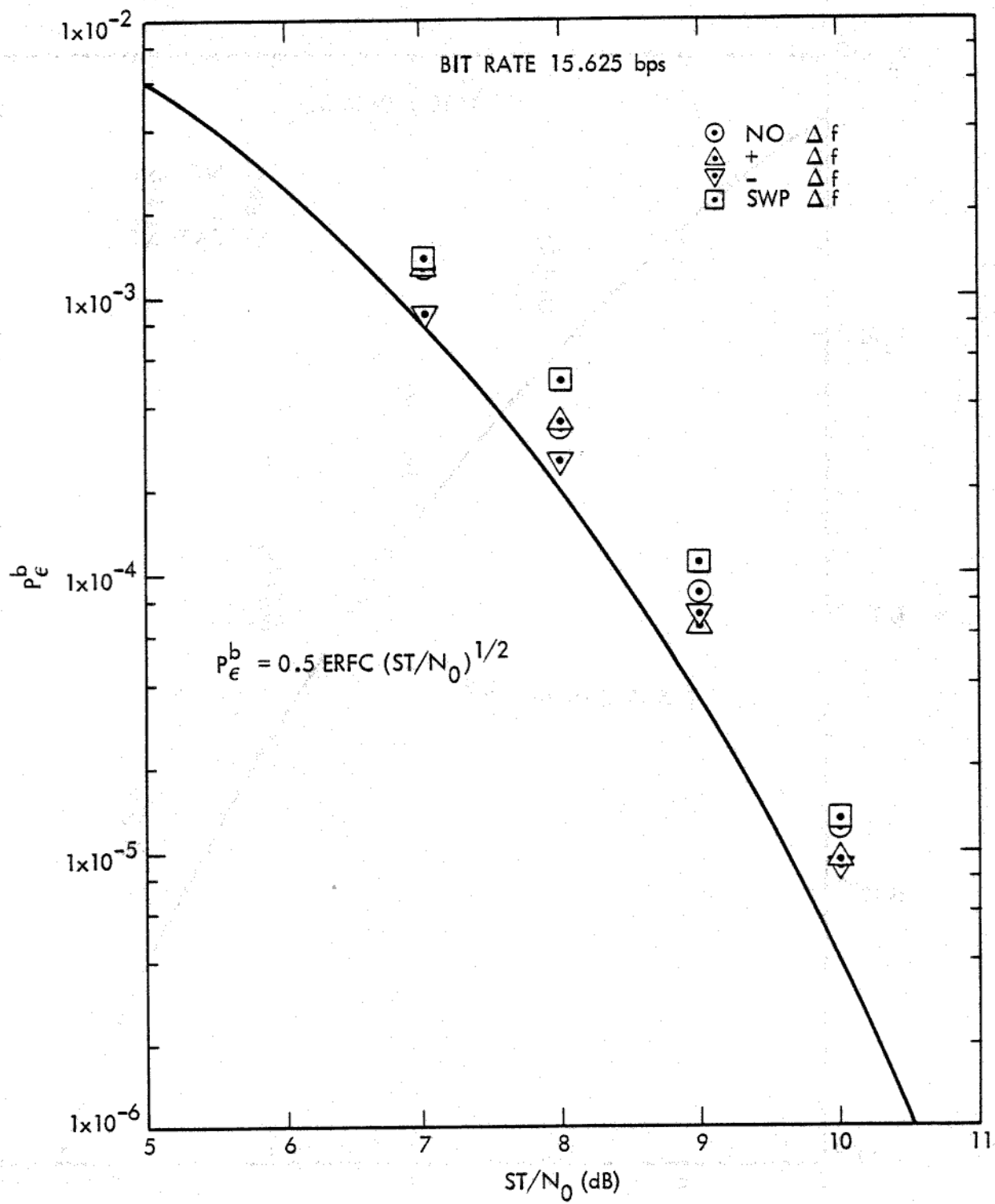


Figure 11.1-6. BER Test Data Summary - Bit Rate = 15.625 bps

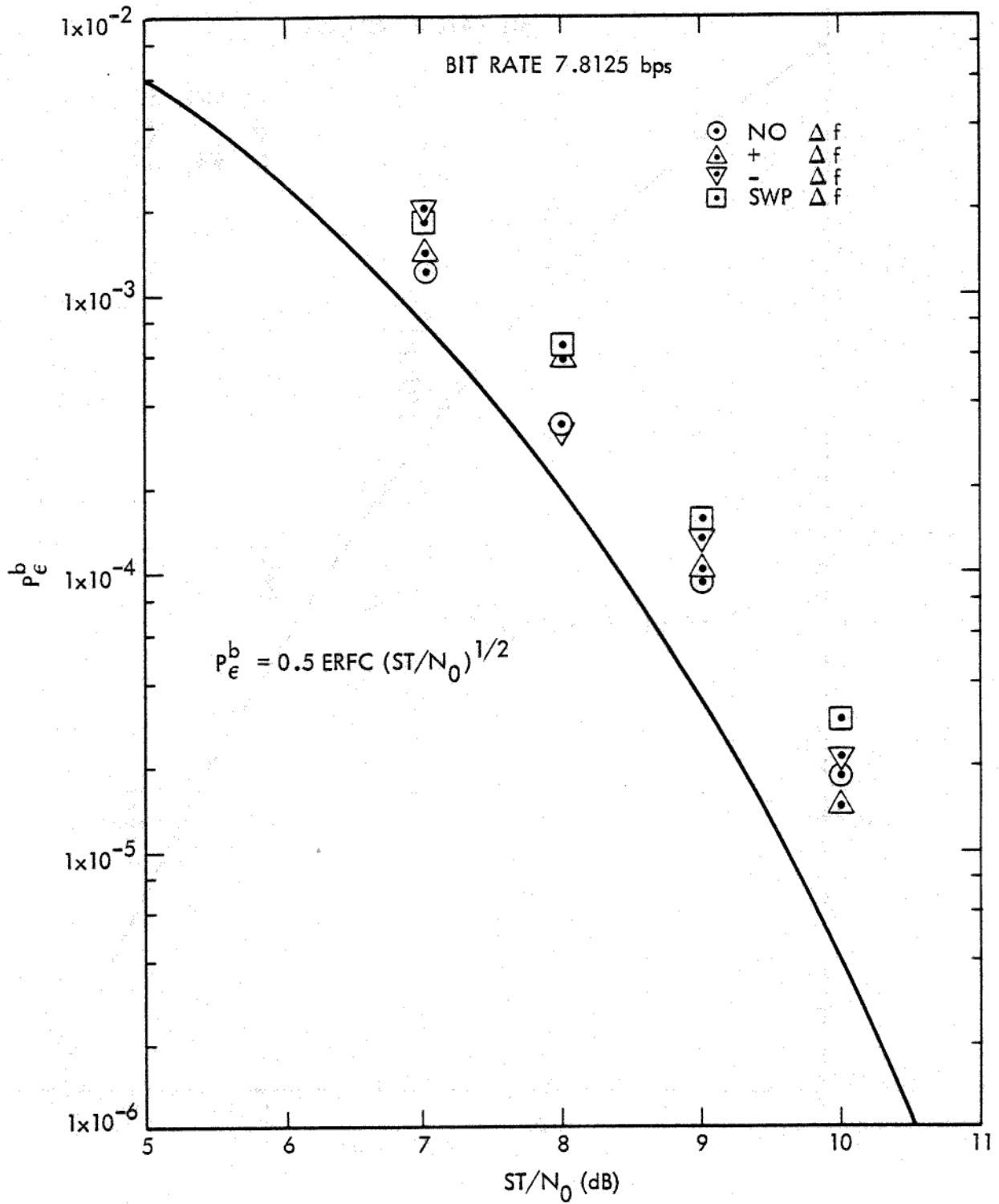


Figure 11.1-7. BER Test Data Summary - Bit Rate = 7.8125 bps

where

p = probability

n = number of tests or bits

c = percent confidence/100

In other words, from a test resulting in no errors in n bits (or bit periods), we are 100% confident that p is the probability.

In our case, we divide the number of bits by eight to get the number of eight-bit periods tested and use this in equation 11.2-1. Table 11.2-1 presents the data and our 90% confidence probability for each of the data rates. As can be seen, all rates easily meet the 1.0×10^{-2} requirement.

Table 11.2-1. False Acquisition Probability Confidence

Rate	False Locks	Bits	Bit Periods	P
500	0	1.4×10^6	1.75×10^5	1.32×10^{-6}
250	0	1.0×10^6	1.25×10^5	1.84×10^{-6}
125	0	1.0×10^6	1.25×10^5	1.84×10^{-6}
62.5	0	1.0×10^6	1.25×10^5	1.84×10^{-6}
31.25	0	1.0×10^6	1.25×10^5	1.84×10^{-6}
15.625	0	1.0×10^6	1.25×10^5	1.84×10^{-6}
7.8125	0	1.0×10^6	1.25×10^5	1.84×10^{-6}

11.3 PROBABILITY OF DE-ACQUISITION

The probability of de-acquisition is the probability that the CDU will drop lock within 27 bit times after the removal of the signal. The requirement on this probability is that the probability of not dropping lock (one minus the probability of dropping lock) be less than 1.0×10^{-2} .

The tests were run and again no failures were observed. Equation 11.2-1 is used again to provide a 90% confidence probability. The results are given in Table 11.3-1. As can be seen, the CDU meets this requirement.

Table 11.3-1. De-Acquisition Probability Confidence

Rate	Failures	Number of Tests	P
500	0	255	9.03×10^{-3}
250	0	250	9.21×10^{-3}
125	0	250	9.21×10^{-3}
62.5	0	250	9.21×10^{-3}
31.25	0	250	9.21×10^{-3}
15.625	0	250	9.21×10^{-3}
7.8125	0	250	9.21×10^{-3}

11.4 PROBABILITY OF OUT-OF-LOCK

The probability of out-of-lock is the probability that for two consecutive eight-bit periods the CDU will be below the unlock threshold (and declare an out-of-lock) when there is a signal. Since the eight-bit intervals are independent, we are concerned about the probability of one eight-bit period being below the threshold. The requirement that we must meet is for this probability to be less than 5.0×10^{-5} .

The test were run and for the threshold conditions, no out-of-locks were noted. Once again, equation 11.2-1 is used to arrive at a 90% confidence probability for the data. Table 11.4-1 provides the test data and the resulting probabilities. As can be seen, the CDU is below this requirement.

11.5 PROBABILITY OF ACQUISITION

The probability of acquisition is the probability that the CDU fails to indicate lock condition when the 176 bit long alternating ones-zeroes pattern is transmitted. This probability is required to be less than 1.0×10^{-4} . Tests were run at each bit rate with differing doppler conditions; i.e., + doppler offset, - doppler offset, and doppler rate. Also, for 500, 31.25, and 7.8125 bps, data was taken at lower ST/N_0 settings.

Once again, the data for 10 dB ST/N_0 shows no failures. However, to reach a 90% confidence level less than 1.0×10^{-4} , we would have to do over 23000 tests. Given the current test setup, this is impossible. However, it is obvious from the lack of any failures at 10 dB that the threshold requirement is met. Table 11.5-1 provides the test data for 10 dB and the corresponding 90% confidence probability. Table 11.5-2 gives the test data for the three data rates that were tested for lower ST/N_0 's. Figures 11.5-1 to 11.5-3 plot the data in Table 11.5-2.

Table 11.4-1. Out-of-Lock Probability Confidence

Rate	Out-of-Locks	Bits	Bit Periods	P
500	0	1.0×10^6	1.25×10^5	1.84×10^{-5}
250	0	5.0×10^5	6.25×10^4	3.68×10^{-5}
125	0	1.1×10^6	1.34×10^5	1.72×10^{-5}
62.5	0	8.0×10^5	1.00×10^5	2.30×10^{-5}
31.25	0	2.8×10^6	3.46×10^5	6.65×10^{-6}
15.625	0	1.4×10^6	1.74×10^5	1.32×10^{-5}
7.8125	0	1.8×10^6	2.27×10^5	1.02×10^{-5}

Table 11.5-1. Probability of Acquisition for 10 dB

Rate	Failures	Tests	90% P	P_{calc}
500	0	906	2.54×10^{-3}	0
250	0	436	5.28×10^{-3}	0
125	0	434	5.31×10^{-3}	0
62.5	0	337	6.83×10^{-3}	0
31.25	0	350	6.58×10^{-3}	0
15.625	0	170	1.35×10^{-2}	0
7.8125	0	450	5.12×10^{-3}	0

Table 11.5-2. Probability of Acquisition for Lower ST/N₀

Rate	ST/N ₀ (dB)	Failures	Tests	P _{calc}
500	7	0	268	0
500	8	0	243	0
500	9	0	231	0
500	10	0	906	0
31.25	7	54	202	2.67 x 10 ⁻¹
31.25	8	1	160	6.25 x 10 ⁻³
31.25	10	0	170	0
7.8125	7	80	211	3.79 x 10 ⁻¹
7.8125	8	13	200	6.50 x 10 ⁻²
7.8125	9	0	160	0
7.8125	10	0	450	0

11.6 SUBCARRIER JITTER

The subcarrier jitter test is a measure of the ability of the Subcarrier Tracking Loop to track the subcarrier in the presence of noise. In Section 4.4.2, we calculated the jitter at an ST/N₀ of 10.5 dB to be 6.77 degrees rms. From [11-1], the jitter is required to be less than 12.5 degrees rms.

To measure the jitter, the size of the subcarrier bump was statistically measured. This was done by writing the subcarrier bump to a special debug memory location and using a logic analyzer to read the value. The jitter was then calculated from the standard deviation of this measurement using equation 11.6-2:

$$\begin{aligned} \text{Jitter} &= \text{Std. Dev. (clock cycles rms)} \\ &\quad \times \text{Clock Cycle Period (sec/clock cycle)} \\ &\quad \times \text{Subcarrier Frequency (cycle/sec)} \\ &\quad \times 360 \text{ (degrees/cycle)} \end{aligned} \tag{11.6-1}$$

$$\text{Jitter} = 1.1538 \times (\text{Std. Dev.}) \tag{11.6-2}$$

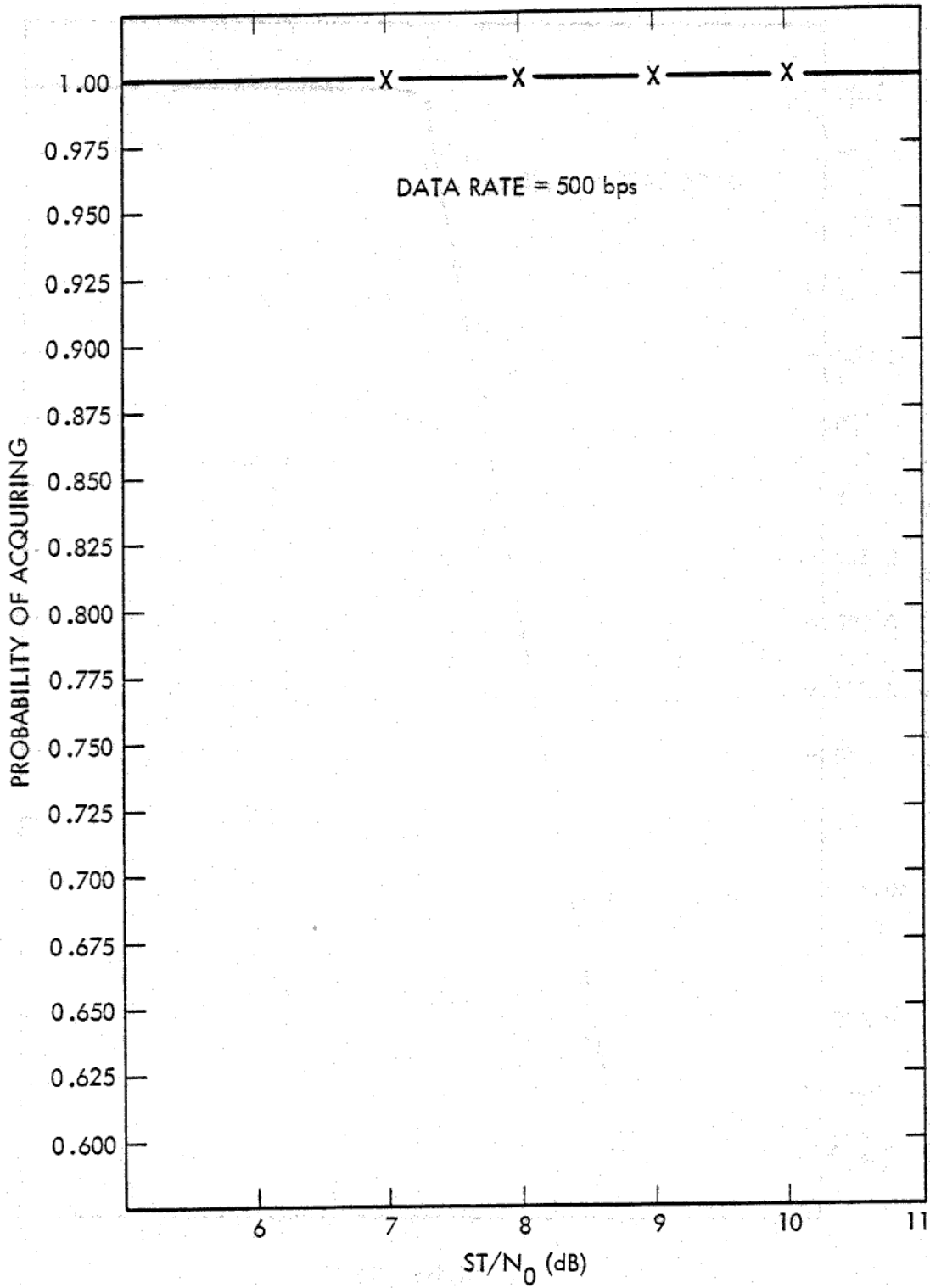


Figure 11.5-1. Probability of Acquisition - Bit Rate = 500 bps

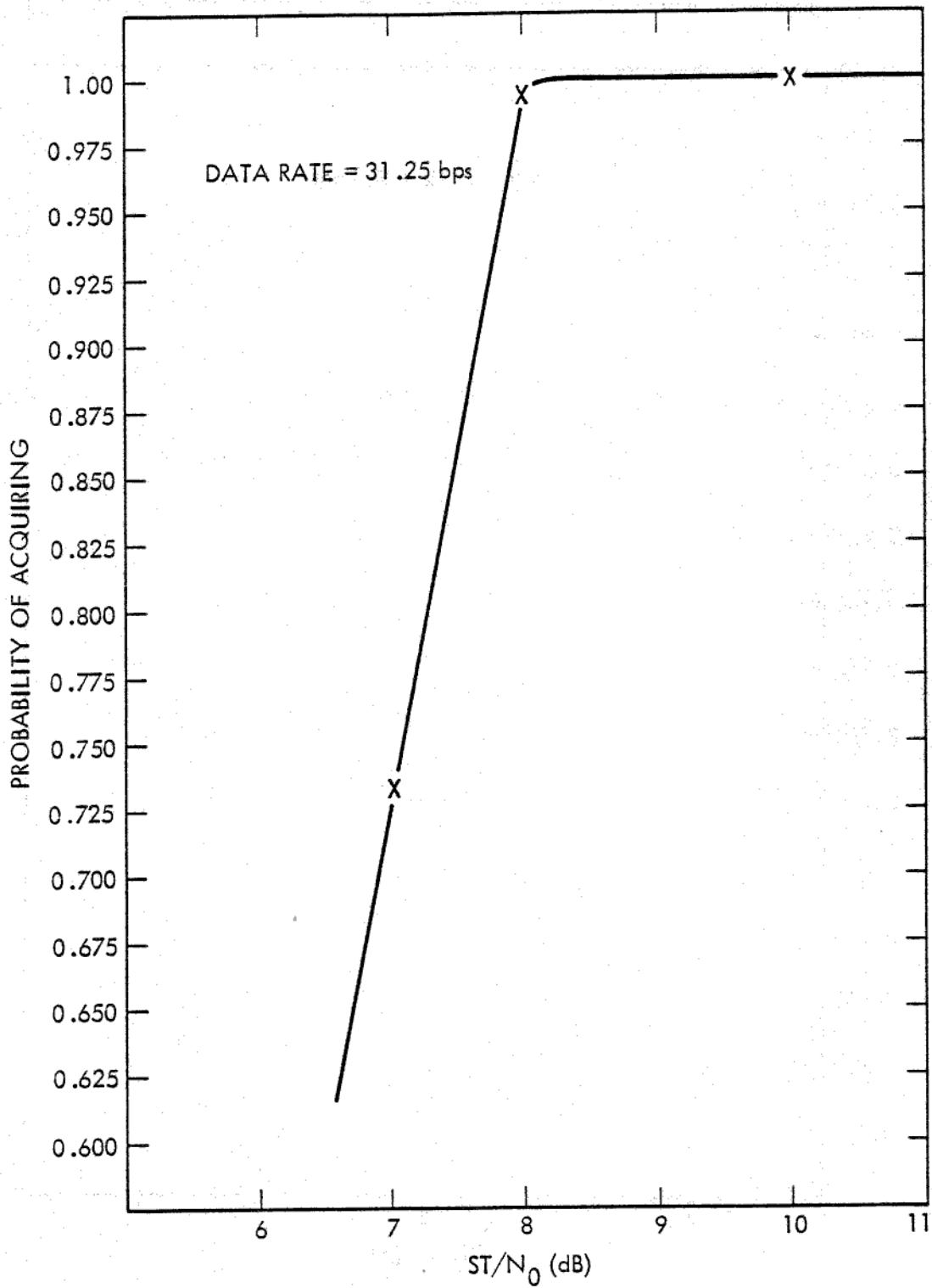


Figure 11.5-2. Probability of Acquisition - Bit Rate = 31.25 bps

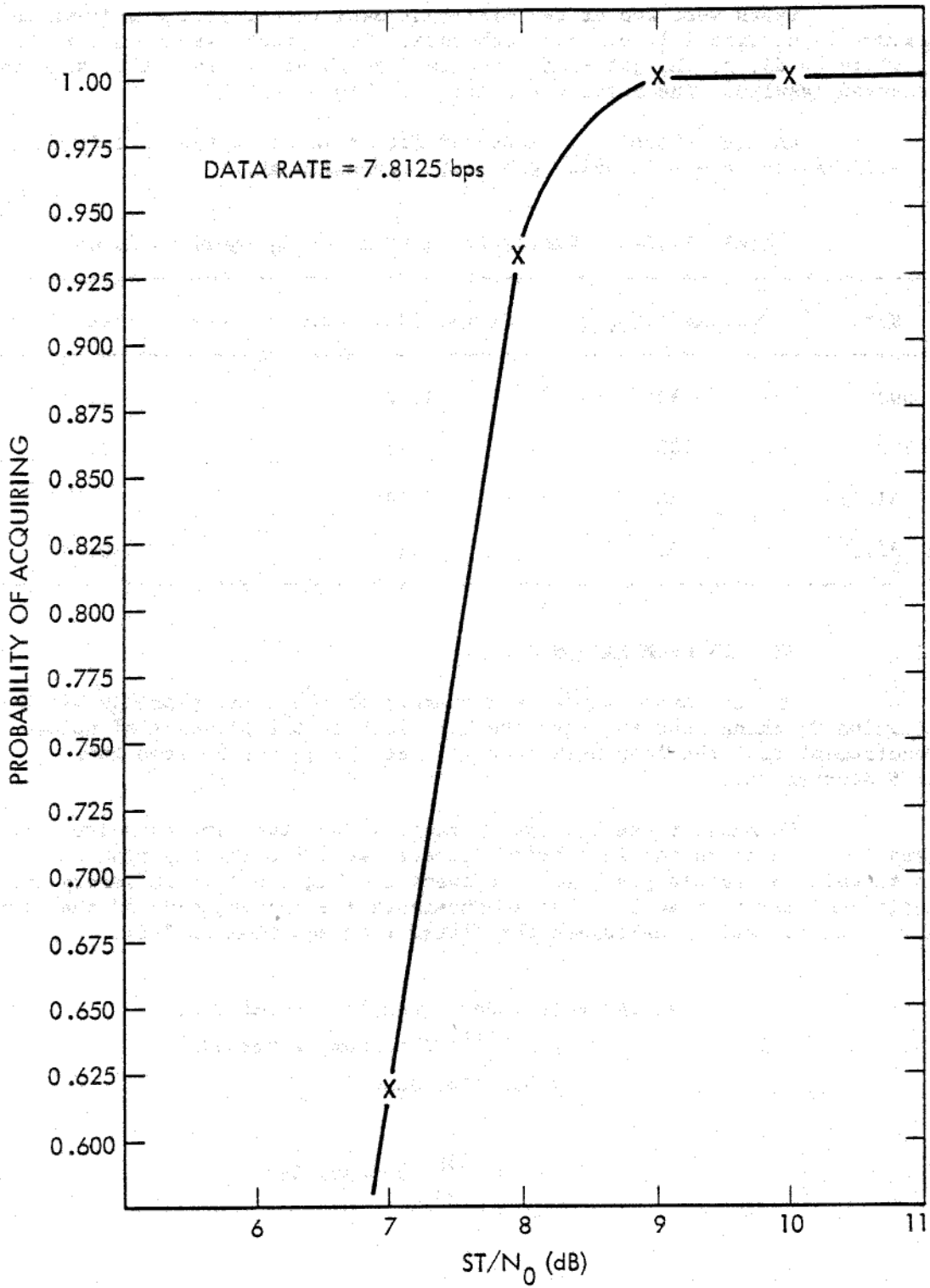


Figure 11.5-3. Probability of Acquisition - Bit Rate = 7.8125 bps

Tests were run at two different data rates, at the minimum and maximum input signal levels for each rate. These tests were run for ST/N₀ equal to 10 dB, so the jitter for the 10.5 dB threshold is better than the measured results. The results are shown in Table 11.6-1.

As can be seen, the measured jitter is below the calculated jitter of 6.77 degrees and well below the jitter requirement.

Table 11.6-1. Subcarrier Jitter, ST/N₀ Equal to 10 dB

Rate	Voltage (mV _{rms})	Jitter (deg. rms)	Calc. Jitter (deg. rms)
500	50	4.92	6.77
500	300	6.00	6.77
31.25	50	6.19	6.77
31.25	300	5.88	6.77

11.7 BIT SYNCHRONIZATION JITTER

The bit sync jitter is a measure of the ability of the Bit Synchronization Tracking Loop to track the bit epoch in the presence of noise. The requirement that the loop must meet is that the jitter be less than 22.5 degrees rms.

To measure the bit sync jitter, we use the same technique that we used in calculating the subcarrier jitter: we write the bit sync bump (which is in units of sample periods) to a special debug location in memory and use a logic analyzer to read it. The measurements are accumulated and the standard deviation is used to calculate the jitter from equation 11.7-2:

$$\begin{aligned} \text{Jitter} &= \text{Std. Dev. (sample periods rms)} \\ &\quad \times 2^{-(r+1)} \text{ (bit/sample periods)} \\ &\quad \times 360 \text{ (degrees/bit)} \end{aligned} \tag{11.7-1}$$

$$\text{Jitter} = \frac{180}{2^r} \times (\text{Std. Dev.}) \tag{11.7-2}$$

The bit sync jitter test was run for two data rates, 500 bps and 31.25 bps, and for two signal levels, 50 mV_{rms} and 300 mV_{rms}, for each, at an ST/N₀ of 10 dB. The results are shown in Table 11.7-1, along with the jitter calculated in Section 6.3.

As can be seen, we are well below the required 22.5 degrees.

Table 11.7-1. Bit Sync Jitter, ST/N₀ Equal to 10 dB

Rate	Voltage (mV _{rms})	Jitter (deg. rms)	Calc. Jitter (deg. rms)
500	50	5.72	16.8
500	300	10.55	16.8
31.25	50	1.31	5.95
31.25	300	1.01	5.95

11.8 PROBABILITY OF CYCLE SLIP

The probability of cycle slip is the probability that the Subcarrier Tracking Loop will slip its phase reference by 180 degrees. This is known as a cycle slip and the result is that the data is inverted. The requirement specifies that the probability of having a cycle slip in any 128,000 bit sequence, at threshold ST/N₀, is less than 1.0 x 10⁻⁵.

From [11-3], we know that the probability of a cycle slip in a time period is:

$$P_{cs} = 1 - \exp[-\bar{S}(t-t_0)] \quad (11.8-1)$$

$$P_{cs} = \bar{S}(t-t_0) \quad (11.8-2)$$

where

\bar{S} = average rate of cycle slips (measured in slips/bit)

$t-t_0$ = time period (measured in bits)

Since cycle slips occur with very tiny probability at higher ST/N_0 's (meaning that a test at higher ST/N_0 's would take days), data was taken for lower ST/N_0 's and the cycle slip rate for 10.5 dB was extrapolated from the plot of this data. Table 11.8-1 gives the data that was measured and Figure 11.8-1 plots this data. All measurements were done for 500 bps. Notice that no cycle slips were observed for ST/N_0 of 9 dB. As can be seen from Figure 11.8-1, if the results for 6 to 8 dB can be linearly extrapolated, the cycle slip rate for 10.5 dB, no doppler, is 3.10×10^{-9} . Using equation 11.8-2, we would get a probability of cycle slip of 3.97×10^{-4} . However the problem with extrapolating is that the CDU is undergoing ADC saturation at the lower ST/N_0 's, so the extrapolated numbers will be higher than the real numbers. The results at 9 dB (no slips) support this, suggesting that the cycle slip rate is well below 3.10×10^{-9} . Thus it is safe to say that the CDU meets the probability of cycle slip requirement.

Table 11.8-1. Probability of Cycle Slip

ST/N_0 (dB)	Doppler	Number of Slips	Number of Bits	\bar{S}
6	None	10	3.91×10^5	2.56×10^{-5}
7	None	10	3.00×10^6	3.33×10^{-6}
8	None	5	1.02×10^7	4.90×10^{-7}
9	None	0	3.10×10^6	0
6	+ Offset	17	2.38×10^5	7.14×10^{-5}
7	+ Offset	6	1.50×10^6	4.00×10^{-6}
8	+ Offset	2	3.00×10^6	6.67×10^{-7}
9	+ Offset	0	5.40×10^6	0
6	- Offset	12	4.65×10^5	2.58×10^{-5}
7	- Offset	7	1.86×10^6	3.77×10^{-6}
8	- Offset	0	4.70×10^6	0
6	Rate	28	2.38×10^5	1.18×10^{-4}
7	Rate	21	1.00×10^6	2.10×10^{-5}
8	Rate	8	4.50×10^6	1.11×10^{-6}

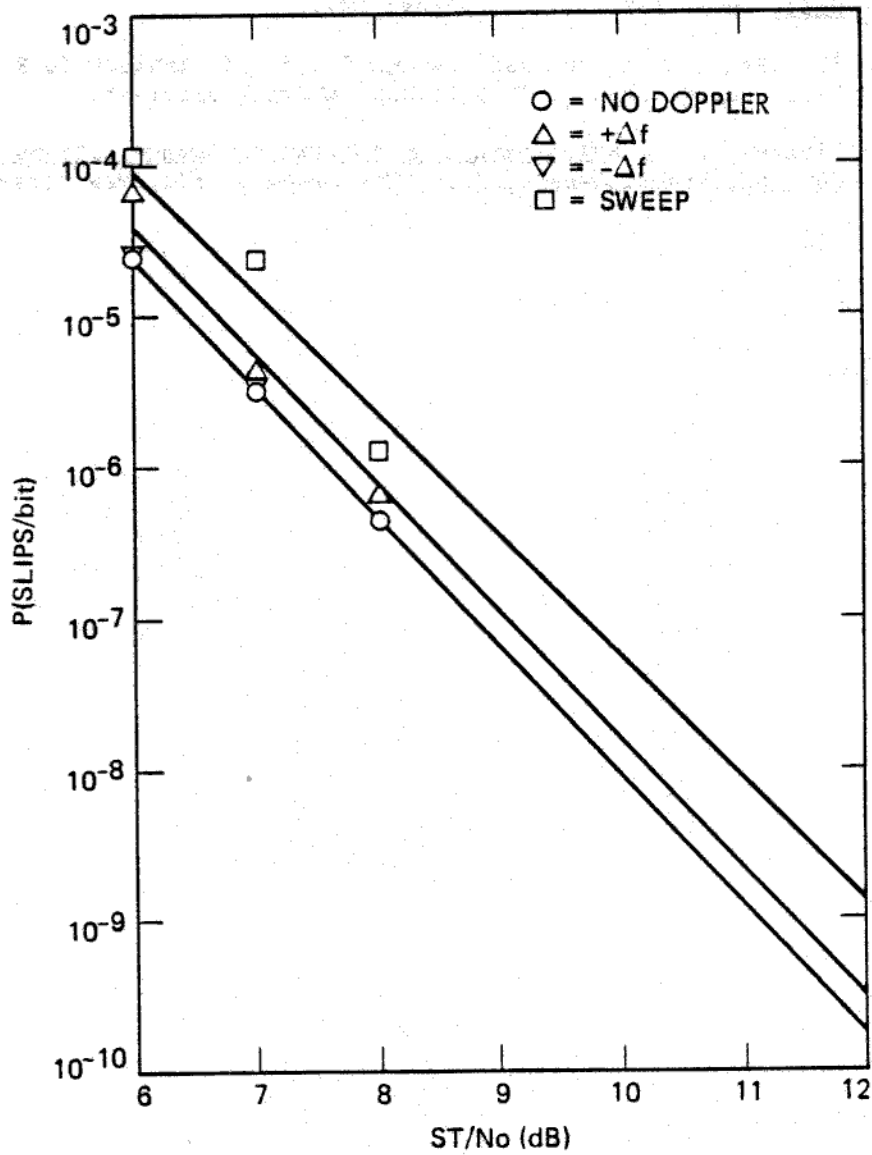


Figure 11.8-1. Probability of Cycle Slip - Data Rate = 500 bps

11.9 CONCLUSION

As the previous sections have shown, the breadboard model testing results indicate that the GDU design meets all of the requirements of [11-1].

11.10 REFERENCES

- 11-1 Albrecht, V.R., Design Requirement NASA Deep Space Command Detector Unit, DM 514438, Rev. A, August 1986.
- 11-2 IOM 3344-167, "Error Rate Estimation," J.C. Ashlock To R.G. Piereson, October 6, 1965 (a JPL Internal Document).
- 11-3 Lindsey, W.C., Synchronization Systems in Communications and Control, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972.