Air Force Institute of Technology

AFIT Scholar

12-1997

Pseudorandom Code Generation for Communication and Navigation System Applications

John F. Brendle Jr.

Follow this and additional works at: https://scholar.afit.edu/etd

Part of the Computer Sciences Commons, and the Navigation, Guidance, Control and Dynamics Commons

AFIT/GE/ENG/97D-16

Pseudorandom Code Generation for
Communication and Navigation System Applications

THESIS

John F. Brendle Jr.
Captain, USAF   DTIC QUALITY INSPECTED 2

AFIT/GE/ENG/97D-16

19980130 146

AFIT/GE/ENG/97D-16

Pseudorandom Code Generation for
Communication and Navigation System Applications

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

John F. Brendle Jr., B.S.E.E.

Captain, USAF

December, 1997
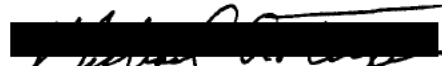
Pseudorandom Code Generation for
Communication and Navigation System Applications

John F. Brendle Jr., B.S.E.E.

Captain, USAF
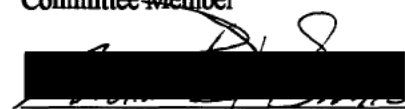
Approved:

_____   24 Nov 97
Maj. Michael A. Temple, Ph. D.   Date
Chairman

_____   24 Nov 97
Lt. Col. (s) Donald S. Gelosh, Ph. D.   Date
Committee Member

_____   24 Nov 97
Dr. Martin P. DeSimio   Date
Committee Member

Acknowledgements

Table of Contents

List of Figures

List of Tables

Abstract

This research project investigated the design, construction and evaluation of a pseudorandom code generator for communication and navigation system applications. These types of codes include spreading codes, Gold codes, Jet Propulsion Laboratory (JPL) ranging codes, syncopated codes, and non-linear codes. Such waveforms are typically used in communication and navigation systems applications. The code generator uses the Stanford Telecom STEL-1032 Pseudorandom Number (PRN) coder. A coder interface was designed and constructed for manual data entry to the registers of the PRN coder. The code generator is capable of independently clocking and generating all possible codes with lengths up to 2^{32}-1 (4,294,967,295). The codes can be started with any random phase. The code generator is capable of detecting a specific position in the code and the coders can be truncated and restarted at that point. The three independent coder outputs are combinable, expanding the lengths and versatility of the codes. The generation of a non-linear code is possible using an internally programmable look-up table. Several tests were conducted on the code generator to ensure its capability of generating the spreading codes, gold codes, JPL ranging codes, syncopated codes, and non-linear codes. The required documentation is being submitted for a U.S. patent.

1. Introduction

1.1 Background

Spread Spectrum (SS) communications grew out of research efforts during World War II to provide secure means of communications in hostile environments [4]. During the early years of spread spectrum investigation, one technique considered for operating a transmitter and receiver synchronously with a truly random spreading signal was the Transmitted Reference (TR) system. In a TR system, the transmitter sends two versions of an unpredictable wideband carrier, one modulated by data and the other unmodulated, which are transmitted on separate channels. At the receiver, the unmodulated carrier is used as a reference signal for despreading the data-modulated carrier. The principal advantage of a TR system is there are no significant synchronization problems at the receiver, since the spread data-modulated signal and the despreading waveform are transmitted simultaneously. The principle disadvantage of TR systems is the spreading code is sent in the clear and thus is available to any listener. As such, the system is easily spoofed by a jammer capable of sending a pair of waveforms acceptable to the receiver. Other disadvantages include 1) performance degradation at low signal levels due to noise being present on both transmitted signals and 2) twice the bandwidth and transmitted power are required because of the need to transmit the reference [6].

Modern spread spectrum systems use a technique called Stored Reference (SR) whereby the spreading and despreading waveforms are independently generated at the transmitter and receiver, respectively. The main advantage of an SR system is that a well-

designed code signal cannot be predicted by an unintended receiver monitoring the transmission [6]. The noiselike code signals used in SR systems cannot be "truly random" as in the case of a TR system. Rather, signals which possess noiselike properties called pseudonoise (PN) or pseudorandom signals, are employed as the spreading waveforms. While a random signal cannot be predicted, a pseudorandom signal is not random at all; it is a deterministic, periodic signal that is known to both the transmitter and receiver. Even though the signal is deterministic, it appears to have the statistical properties of sampled white noise and appears to the unauthorized listener as a truly random signal.

A linear feedback shift register is often used to generate the pseudorandom spreading code. The shift register operation is controlled by a sequence of clock pulses. At each clock pulse, the contents of each stage in the register are shifted one stage to the right and fed back through of series of interconnected taps. The shift register sequence is usually defined as the output of the last stage. The shift register sequence is dependent on the number of stages, the feedback tap connections, and the initial conditions (starting phase). The output sequence is classified as either maximal length or nonmaximal length. A maximal length sequence has the property that for an n-stage linear feedback shift register, the sequence repetition period (in clock pulses p) is $p = 2^n - 1$. If the sequence length is less than $(2^n - 1)$, the sequence is classified as a nonmaximal length sequence.

1.2 Problem Statement

The purpose of this study is to design a pseudorandom code generator for communication and navigation system applications. This thesis addresses the need for a flexible, highly efficient, means of providing pseudorandom signals for test, evaluation and development of multiple systems. In addition, the ability to explore and validate new pseudorandom codes is addressed.

1.3 Scope

The design, construction, and evaluation of the system has enormous possibilities. Due to time constraints, it was necessary to put limitations on the project. The system was designed for a computer interface but has yet to be implemented; a manual data entry and addressing technique was developed for system testing. Several test code and scenarios are analyzed and presented in Chapter 4.

1.4 Materials and Equipment

Appendix B contains the complete materials list and design schematics used in this project. A great deal of the materials and equipment used for construction and evaluations were provided by the Avionics Directorate of Wright Laboratories. Other items were procured by out of pocket expenses.

1.5 Thesis Organization

Chapter 2 presents a background on spread spectrum signal codes and the PRN coder capabilities. This chapter also includes a description of different types of spreading

codes that can be generated with this project and examples of their application. In Chapter 3, the capabilities, performance, and operation of the STEL-1032 are fully described and the interface design is presented and analyzed. Chapter 4 begins with an explanation on operating the system and concludes with the analysis of several test outputs. The conclusion and recommendations are included in Chapter 5. Appendix A includes a table with feedback connections for generation of maximal length sequences. Appendix B shows the complete circuit diagram for the system including a parts listing.

2. Pseudorandom Sequences

2.1 Pseudonoise/Pseudorandom sequences

In the Transmitted Reference (TR) system, a truly random code can be utilized for spreading and despreading since the code signal and data-modulated code signal are simultaneously transmitted over different regions of the spectrum. The Stored Reference (SR) approach cannot use a truly random code signal because a copy of the code needs to be stored or generated at the receiver. For a SR system, a pseudonoise or pseudorandom code is typically used. A truly random signal is unpredictable and future variations can only be described in a statistical sense. However, a pseudorandom signal is not really random -- it is a deterministic periodic signal that is known to both the transmitter and receiver. Even though the signal is deterministic, it possess statistical properties consistent with sampled white noise and appears to be truly random to an unauthorized listener. There are three basic properties that can be applied to any periodic binary sequence as a test for the appearance of randomness. These properties are called balance, run and correlation [6]. If all three properties are satisfied, the sequence is classified as a pseudorandom sequence.

2.1.1 Balance Property

Good balance requires that within each period of the sequence, the number of binary ones differ from the number of binary zeros by at most one digit. As an example,

the sequence (0 0 0 1 0 0 1 1 0 1 0 1 1 1 1) has seven zeros and eight ones, satisfying the balance property.

2.1.2 Run Property

A run is defined as a sequence of a single type of binary digit(s). The appearance of the alternate digit in a sequence starts a new run. The length of the run is defined as the number of digits in the run. Among the runs of ones and zeros in each sequence, it is desirable that about one-half the runs of each type are of length 1, about one-fourth are of length 2, one-eighth are of length 3, and so on. Considering the runs of zeros in the previous example, there are two (one-half) of length 1, one (one-quarter) of length 2, and one of length 3. Considering the runs of ones, there are two (one-half) of length 1, one (one-quarter) of length 2, and one of length 4. The sequence satisfies the run condition.

2.1.3 Correlation Property

If a single period of the sequence is compared term-by-term with any cyclic shift of itself, it is desirable that the number of agreements differs from the number of disagreements by not more than one count. Below is a bit-by-bit comparison of the example sequence with a single end-around shift of itself.

```
0 0 0 1 0 0 1 1 0 1 0 1 1 1 1
1 0 0 0 1 0 0 1 1 0 1 0 1 1 1
d a a d d a d a d d d d  a a a
```

Digits that agree are labeled *a* and those that disagree are labeled *d*. As shown, the number of agreements (7) differ from the number of disagreements (8) by 1. It is easily shown that the example sequence satisfies the correlation property for any cyclic shift. Given the example sequence of (0 0 0 1 0 0 1 1 0 1 0 1 1 1 1) satisfies the balance, run, and correlation properties, it is classified as a pseudorandom sequence.

2.2 Shift Registers

Many system applications require pseudorandom sequences. A convenient and simple method for generating sequences which appear random and posses the characteristics described in Section 2.1 is required. It is also desirable that the same apparatus, when operated in an identical manner, produces the same sequence and that the generator be as simple as possible. Such a sequence is easily generated using a feedback shift register configuration.

A shift register is the arrangement of *n* stages, called delay elements, in a row. Each stage contains either an "on" (1) or "off" (0); the contents of each stage are shifted to the next stage, in time with a clock pulse. If no new signals are introduced into the first stage during the shifting process, by the end of *n* shifts all of the stages will be "off" and will remain that way. One way to keep the shift register "active" is to feed back the state(s) of one or more of the *n* stages back into the first stage. The output(s) from certain stage(s) feed modulo 2 adder(s), when the next shift of the register occurs the modulo 2 sum is transferred to the first stage [2].

2.2.1 Feedback Taps

A linear code sequence generator can be made up of any combinations of stages and feedback taps. Figure 2.1 illustrates the general form of a simple linear feedback shift register. Outputs from the last stage (D_4) and from an intermediate stage (D_3) are combined in a modulo-2 adder and fed back to the input of the first stage (D_1). Table 2.1 shows the contents of each stage and the output for the first 16 clock pulses. The initial condition, or fill, was $D_1=1$, $D_2=0$, $D_3=0$ and $D_4=0$.
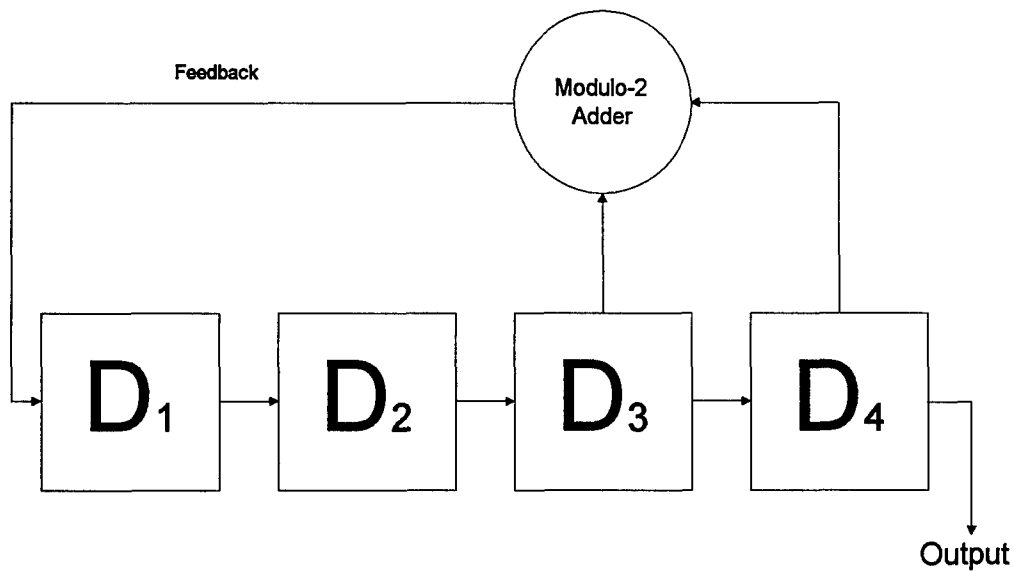
Figure 2-1. Linear Feedback Shift Register.

Table 2-1. Linear Feedback Shift Register Sequence.

State	D_1	D_2	D_3	D_4	Output
0	1	0	0	0	-
1	0	1	0	0	0
2	0	0	1	0	0
3	1	0	0	1	0
4	1	1	0	0	1
5	0	1	1	0	0
6	1	0	1	1	0
7	0	1	0	1	1
8	1	0	1	0	1
9	1	1	0	1	0
10	1	1	1	0	1
11	1	1	1	1	0
12	0	1	1	1	1
13	0	0	1	1	1
14	0	0	0	1	1
15	1	0	0	0	1

As seen in Table 2-1, all of the stages are again filled with the initial values at state 15; this linear feedback shift register output sequence repeats every 15 clock pulses. The succession of states in the shift register is periodic; with a period given by $p \leq 2^n - 1$. For an n-stage shift register, each stage containing either a '1' or '0', there are a total of 2^n possible states. Therefore, if repetition occurs it must occur somewhere in the first $2^n + 1$ states with periodicity given by $p \leq 2^n$. However, if the "all 0's" state ever occurs, all subsequent states will consist of "all 0's" and the periodicity is $p = 1$. Thus, a long period can not include this state and $p \leq 2^n - 1$ [2]. The output sequences are classified as either maximal or nonmaximal length. Maximal length sequences have a repetition period $p = 2^n - 1$. For $p < 2^n - 1$, the sequence is classified nonmaximal length.

2.2.1.1 Maximal Length Sequences

Not every combination of feedback taps results in a maximal length sequence.

Using all possible linear combinations of feedback taps for an n-stage register, there are

$[\phi\ (2^n - 1)]\ /\ n$ maximal sequences that can be generated [1]. The expression

$[\phi\ (2^n - 1)]$ is an Euler number which is the number of positive integers (including 1) that

are relatively prime to and less than $(2^n - 1)$. Table 2-2 lists the number of possible

feedback combination for a given number of stages n that produce maximal length

sequences and the length of code (period p).

Table 2-2. Number of Maximal Sequences and Code Lengths Available from Register Lengths 2 through 32.

n	Number of Codes	Code Length	n	Number of Codes	Code Length
2	1	3	18	8064	262143
3	2	7	19	27594	524287
4	2	15	20	24000	1048575
5	6	31	21	84672	2097151
6	6	63	22	120032	4194303
7	18	127	23	356960	8388607
8	16	255	24	276480	16777215
9	48	511	25	1296000	33554431
10	60	1023	26	1719900	67108863
11	176	2047	27	4202496	134217727
12	144	4095	28	4741632	268435455
13	630	8191	29	18407808	536870911
14	756	16383	30	11880000	1073741823
15	1800	32767	31	69273666	2147483647
16	2048	65535	32	67108864	4294967295
17	7710	131071			

It is difficult to find the feedback connections that produce the desired maximal

length code and to check the code once the linear feedback shift register has been

constructed. It is possible to find a set of feedback connections experimentally, but this requires not only constructing a shift register generator but also taking the time to check the length of codes generated until a maximal length sequence is achieved. Tables of feedback connections and tables of irreducible polynomials have been generated which makes the job easier. Appendix A includes a table of feedback connections for generation of maximal length sequences [5].

2.2.1.1.1 Properties of Maximal Length Sequences

Maximal Length sequences have a number of properties which are useful in their application to spread-spectrum systems [4].

1. A maximal-length sequence contains one more "1" than "0". The number of ones in the sequence is $\frac{1}{2}(n + 1)$

2. The modulo-2 sum of an *m*-sequence and any phase shift of the same sequence is another phase of the same *m*-sequence.

3. If a window of width *r* is slid along the sequence for N shifts, each *r*-tuple except the all zero *r*-tuple will appear exactly once.

4. The periodic autocorrelation function is two-valued and is given by $\theta_b(k) = 1.0$ if k equals $l*p$ or $\theta_b(k) = -1/N$ if k is not equal to $l*p$, where *l* is any integer and *p* is the sequence period.

5. Defining a run as a subsequence of identical symbols within the *m*-sequence. The length of this subsequence is the length of the run. Then, for any *m*-sequence, there is

   1. one run of "1's" of length n.

   2. one run of "0's" of length $n-1$.

   3. one run of "1's" and one run of "0's" of length $n-2$.

   4. two runs of "1's" and two runs of "0's" of length $n-2$.

   5. four runs of "1's" and four runs of "0's" of length $n-2$.
      $$\vdots$$

   n. 2^{n-3} runs of "1's" and 2^{n-3} runs of "0's" of length 1.

2.3 Composite Codes

Composite code sequences are generated by combining linear maximal length sequences. Codes constructed this way have special properties that are most advantageous under proper circumstances; for instance, the Jet Propulsion Laboratory (JPL) ranging codes and the Gold codes, though constructed from maximal sequences, are not maximal. The JPL ranging codes have special properties that permit rapid synchronization, whereas the Gold codes allow construction of families of $2^n - 1$ codes from pairs of *n*-stage shift registers which all codes have well-defined correlation characteristics [1].

2-8

2.3.1 Gold Code Sequence Generator

One of the applications of the pseudorandom sequence for spread-spectrum is to provide a means other than Frequency-Division Multiple Access (FDMA) or Time-Division Multiple Access (TDMA) of sharing limited bandwidth. When channel resources are shared using spread-spectrum techniques, all users are permitted to transmit simultaneously within the same band of frequencies. Users are each assigned different spreading codes to allow separation within the receiver via the despreading process. A goal of the spread-spectrum system designer for multiple access is to find a set of spreading codes such that as many users as possible can use a band of frequencies with as little mutual interference as possible. The specific amount of interference from a user employing a different spreading code is related to the cross-correlation between the two spreading codes. Gold codes were invented for use in multiple-access applications of spread-spectrum. Relatively large sets of Gold codes exist which have well controlled cross-correlation [4].

Gold codes are generated by modulo-2 addition of a pair of maximal length linear sequences. The code sequences are added chip-by-chip using synchronous clocking as shown Figure 2.2. The set of Gold codes generated are the same length but are nonmaximal. A multiple-register Gold code generator can generate $(2^n - 1)^r$ nonmaximal sequences of length $2^n - 1$ in addition to r maximal sequences where r is the number of registers and n is the number of stages in each register [1]. These codes are important and they have been selected by NASA for use on the Tracking and Data Relay Satellite System

(TDRSS) [4]. One of the most familiar uses of Gold codes is the Global Positioning

System (GPS).
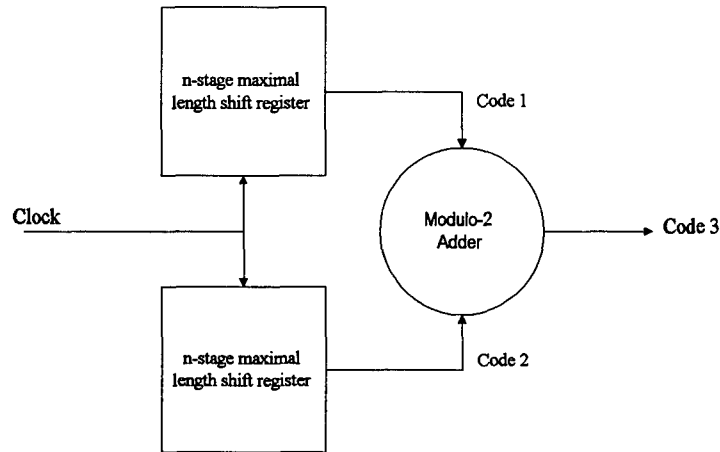
Figure 2-2. Gold Code Sequence Generator Configuration.

2.3.1.1 GPS Gold Code Generation

The Standard Positioning Service pseudorandom noise ranging code is known as

the Course / Acquisition (C/A) code. Appropriate Code-Division-Multiplexing (CDM)

techniques allow differentiating between the satellites even though they all transmit on the

same L-band frequency. The C/A code consists of 1.023 Mbps $G_i(t)$ patterns with

Modulo-2 addition of the navigation data bit train, D(t), which is clocked at 50 bps. Each

$G_i(t)$ sequence is a 1023-bit Gold-code which is the Modulo-2 sum of two 1023-bit linear

patterns, G1 and $G2_i$. The latter sequence is selectively delayed by an integer number of

chips ranging from 5 to 950 to produce 36 unique G(t) patterns. This allows the

generation of 36 unique C/A (t) code phases using the same basic code generator.

The G1 and G2 sequences are generated by 10-stage shift registers having the

following polynomials as referred to in the shift register input:

G1: $X^{10} + X^3 + 1$

G2: $X^{10} + X^9 + X^8 + X^6 + X^3 + X^2 + 1$

The initialization vector for the G1 and G2 sequence is (1111111111). The G1 and G2

registers are clocked at a 1.023 MHz rate. The effective delay of the G2 sequence to form

the G2i sequence is accomplished by combining the output of the two stages of the G2

shift register by Modulo-2 addition. The C/A-code generation including the G1 and G2

shift registers is shown in Figure 2.3 [8].
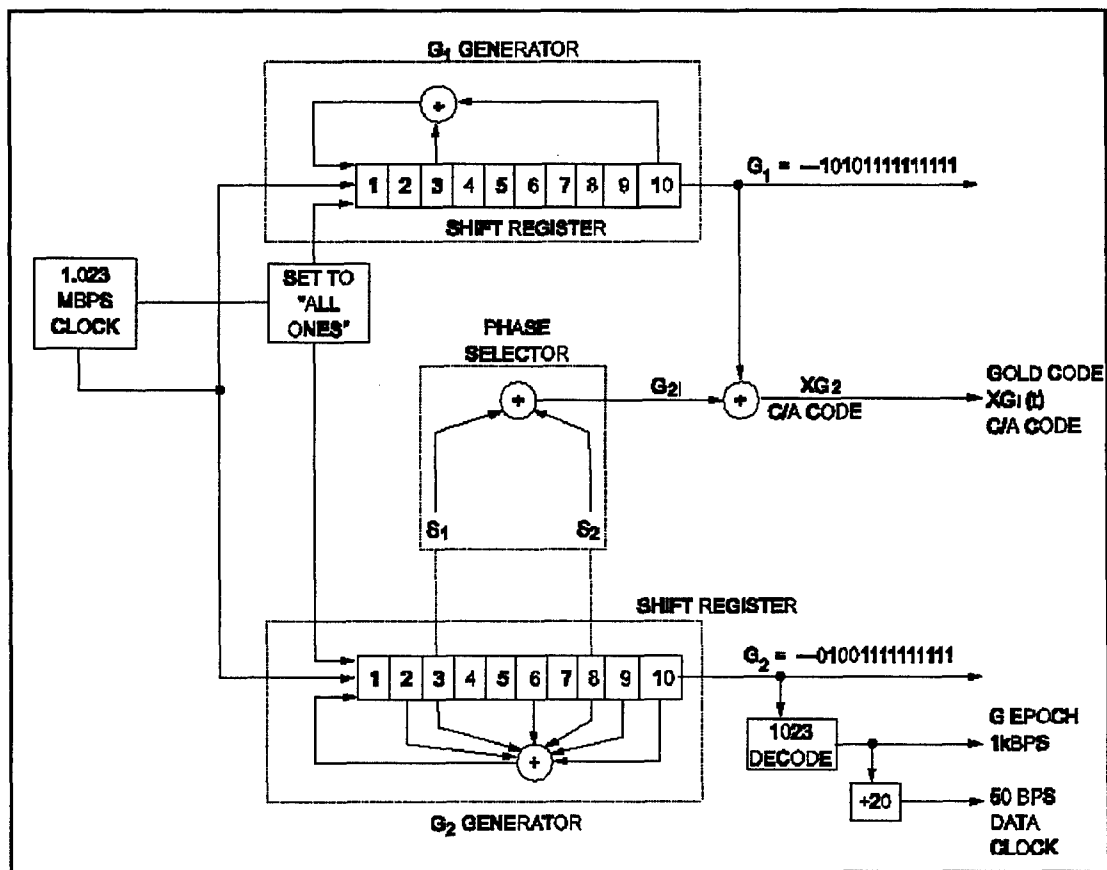
Figure 2-3. C/A Code Generation.

2.3.2 JPL Ranging Code Generator

The Jet Propulsion Laboratory (JPL) ranging codes are constructed by modulo-2 addition of two or more maximal length linear sequences which are relatively prime to one another. There are several advantages to such a technique including, 1) very long codes useful for unambiguous ranging over long ranges are available, 2) the long codes are generated by a relatively small number of shift register stages, 3) synchronization of a receiver can be accomplished by separate operations on the component code. This can greatly reduce the time required for synchronization [1].

Synchronization, via the JPL component codes is accomplished by first cross-correlating one of the component codes with the composite code. Once this component code reaches the point of synchronization with its mate, which is embedded in the composite code, a partial correlation occurs. The partial correlation is then the signal for the second component code cross-correlation to be initiated which causes the partial correlation level to be increased. This process continues until all the component codes making up the overall composite code are individually synchronized with their counterparts in the received signal. When all are individually synchronized, the correlation is the same as if the process had simply synchronized the composite code. As an example, when the component codes are 200, 500 and 1000 chips in length, a separate search process over these individual lengths (a total of 1700-chips search) can be accomplished much more rapidly than a search of the composite 10^8 chips [1].

Figure 2.4 shows a typical JPL code generator configuration, which has three basic maximal length shift register generators, each with a different number of stages. This is identical to the Gold code generator except for the difference in the individual code lengths.
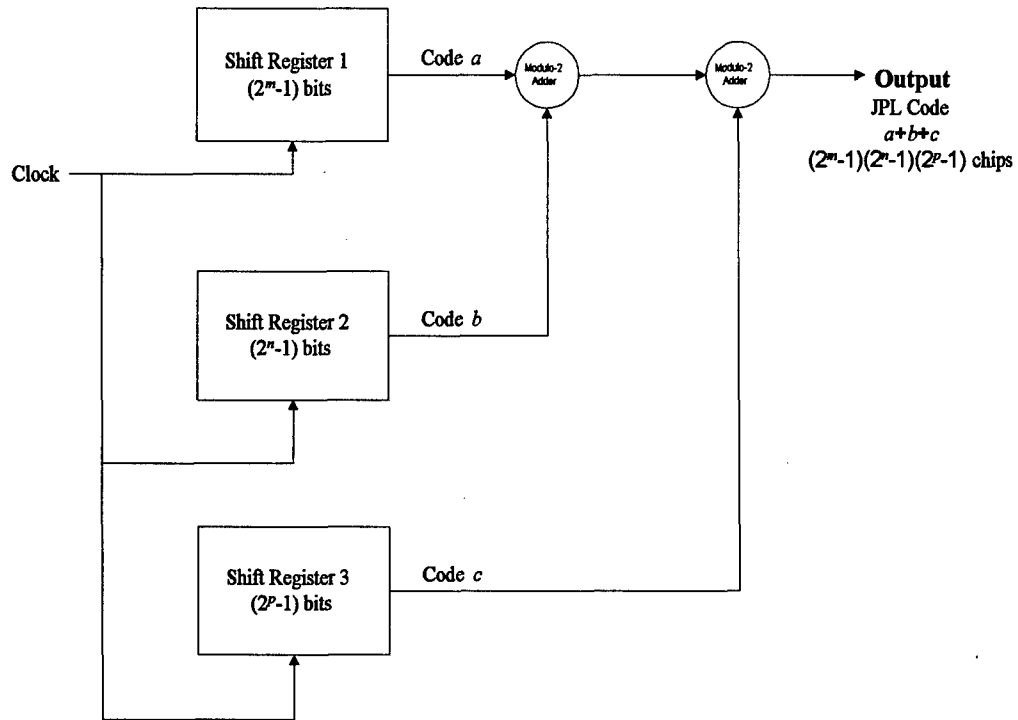
Figure 2-4. Typical JPL Code Sequence Generator Configuration.

2.3.3 Syncopated-Register Generator

A syncopated-register generator is a technique that multiplexes two or more slower generators to generate a high-rate sequence. The method is similar to that used in Gold code sequence generators in that two separate sequences are modulo-2 added to produce a composite output. The difference lies in that separate clocks, phase shifted by $360/P$ degrees, each at a rate R/P, where R is the desired output code chip rate and P is

the number of registers used. Figure 2.5 shows a two-register generator of the syncopated type. Timing for this code generator is also shown in Figure 2.6. Note the chip rate of code 1 modulo-2 added to code 2 is twice that of either code 1 or code 2 before their combination. The overall length of any number of syncopated codes is the product of the lengths of all the composite codes [1].
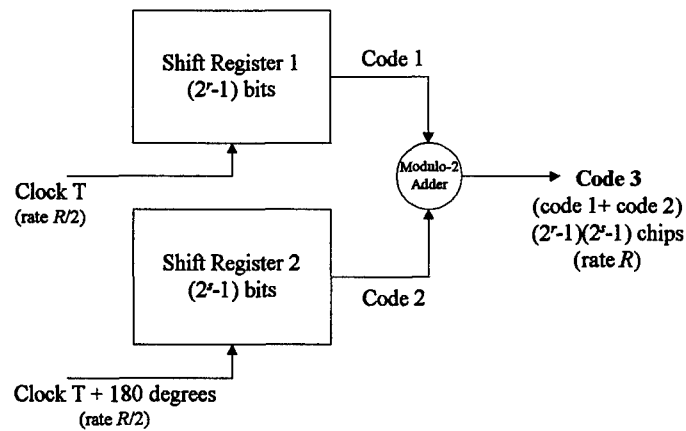
Figure 2-5. Syncopated Code Generator.
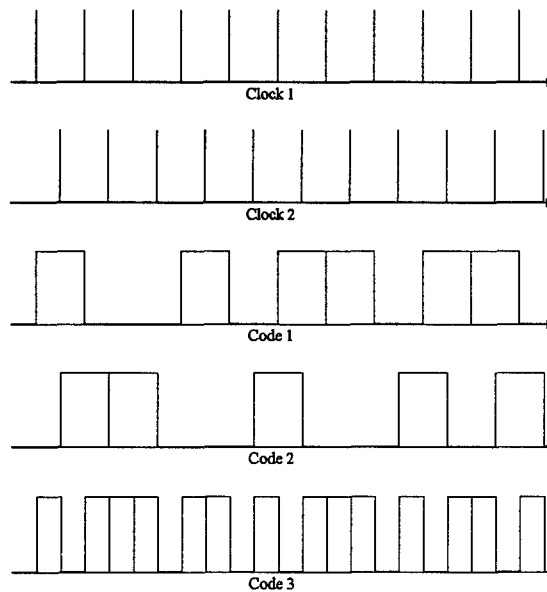
Figure 2-6. Syncopated Code Example.

2.4 Non-Linear Codes

In applications where security is important, it is necessary that the unintended listener not be able to obtain complete knowledge of the spreading code or encryption stream being employed. If this code is known, the eavesdropper can decipher or demodulate the transmitted signal just as the desired receiver. Therefore, the task of determining the code generator configuration from knowledge of the transmitted signal should be as difficult as possible. The feedback connections for an n-stage maximal length sequence can easily be determined from knowledge of $2n$ successive code symbols [1]. For this reason, m-sequences are never used when a high degree of security is required. The use of non-linear sequences is one way to increase security. These codes can not be described by the simple linear recurrence. One way to achieve a non-linear sequence is to use the output from three linear feedback systems as an address to a look-up table. This method combines the three output bits to produce one bit based on the look-up table. Other methods of increasing complexity exist such as using modulo-2 multiplication in addition to modulo-2 addition or using nonlinear feedforward.

3. *System Design*

3.1 System Overview

The purpose of this study is to design a pseudorandom code generator for communication and navigation system applications. The system designed utilizes the Stanford Telecom STEL-1032, pseudorandom number (PRN) coder [9]. The STEL-1032 PRN coder generates codes using three independent 32-bit register code generators (Coder0, Coder1, and Coder2). Feedback tap connections for each code generator are controlled by the MASK Registers content -- any combination of tap connections may be selected. In this way, each of the three code generators are capable of independently generating all possible codes with lengths up to 2^{32} - 1 (4,294,967,295) bits. The PN codes can be started with arbitrary phase by loading the starting phase code into the INIT Registers. A specific sequence in the code generator registers can be detected and a pulse generated via a 32-bit magnitude comparator and the values stored in the EPOCH Registers. The 32-bit COUNT Register is used to set the counter stop point; once triggered, the counter will run for a number of clock cycles equal to the number stored in the COUNT Register and then generate a COUNT pulse. The code generators and counters can be reset on the EPOCH or COUNT pulse. The codes can be made to restart at this point if desired. In addition to the three independent codes that can be generated, the outputs of the code generators can be EXORed together, Coder0 and Coder1 can be EXORed or all three code generator outputs can be EXORed. The output of code

combiner is also available both late and early by one-half of a clock cycle relative to the punctual code (on time). Non-linear codes are generated by means of an internally programmable look-up table. A block diagram for the STEL-1032 is shown in Figure 3.1.
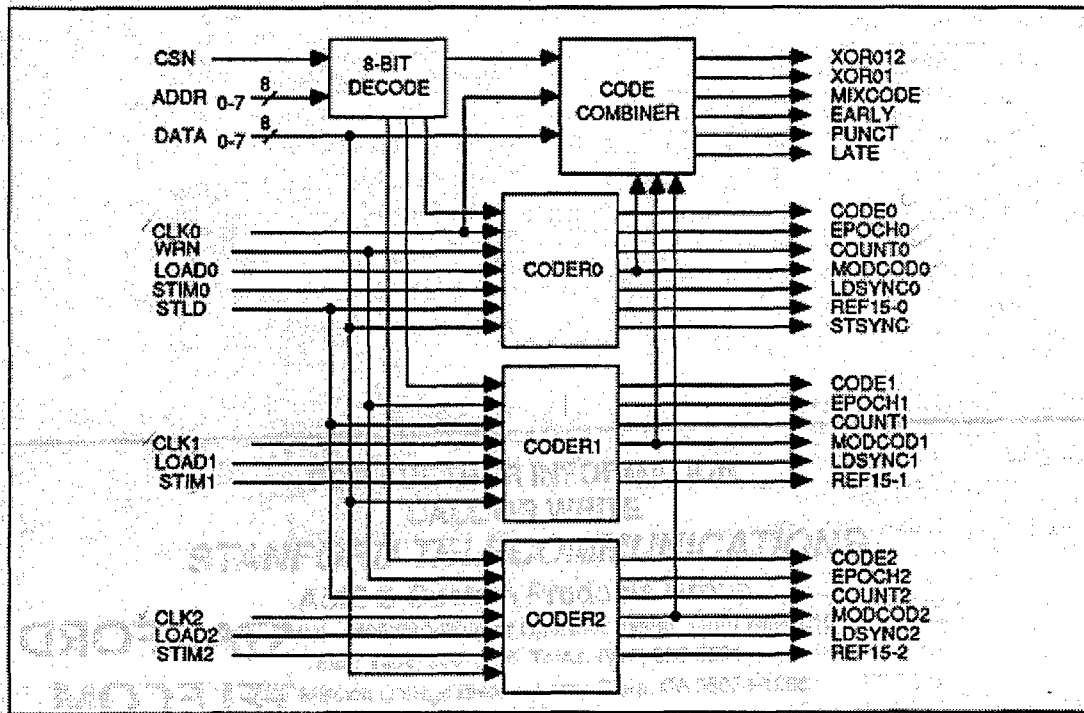
Figure 3-1. STEL-1032 Block Diagram.

Each of the three separate coders contain a number of functional blocks, shown in Figure 3-2. The three coders are completely independent, except for the COUNT and EPOCH which are interconnected through the coders' control logic. This allows each coder to be controlled by either of the two other coders.
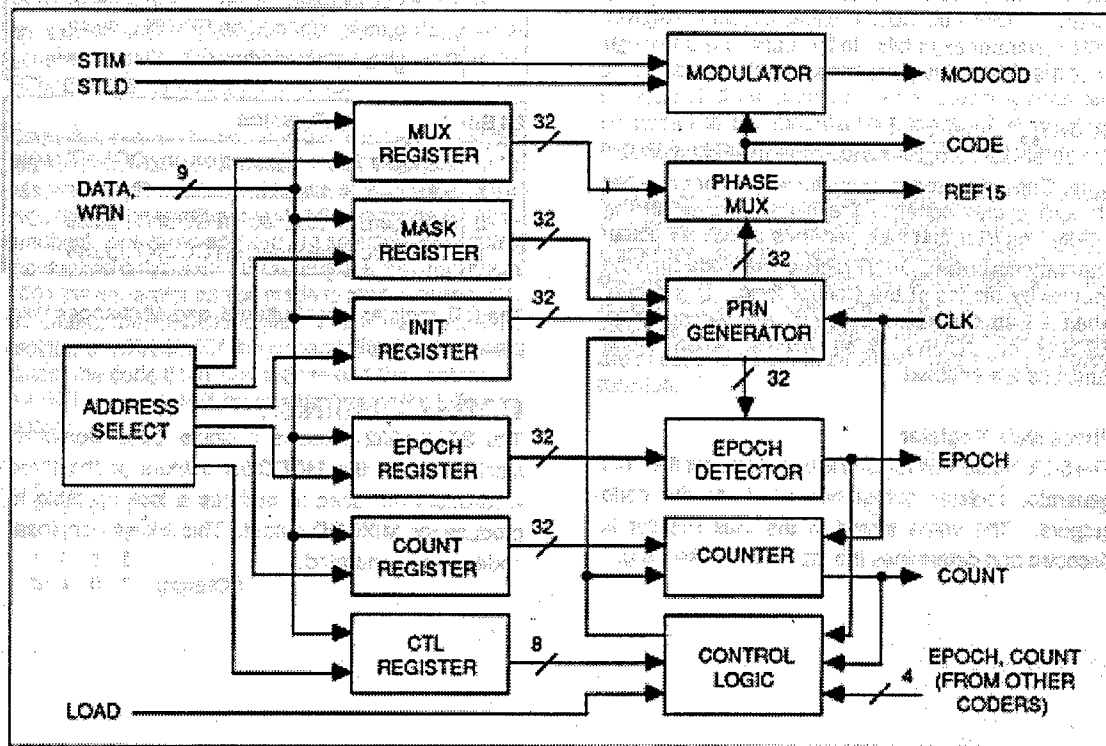
Figure 3-2. Individual Coder Block Diagram.

The STEL-1032 is programmed by means of the data stored in the registers. Each of the three independent code generators consists of a complete set of internal registers as shown in Figure 3-3. The 8-bit data bus is mapped into the 32-bit registers by means of address inputs $ADDR_0$ through $ADDR_7$.
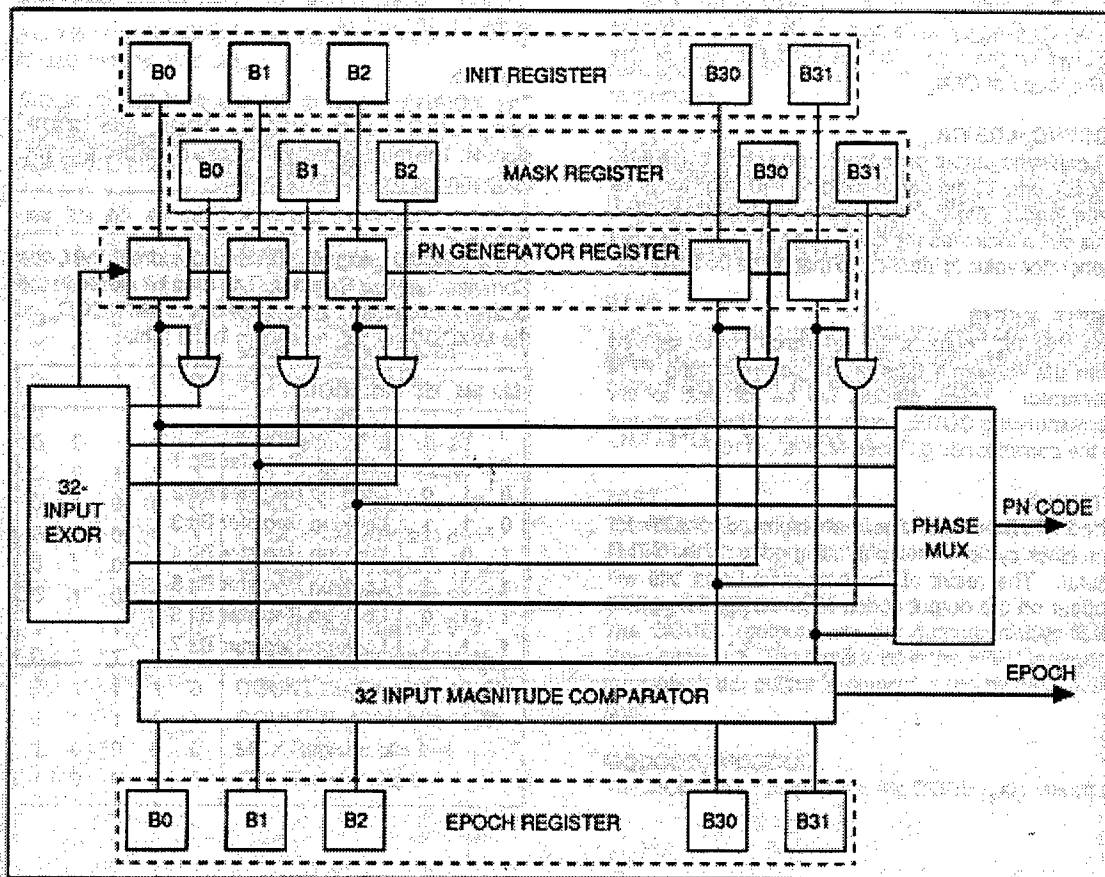
Figure 3-3. Registers Connections of Individual Coder.

3.1.1 STEL-1032 Inputs

As shown in Figure 3.1, the STEL-1032 has the following inputs: an eight bit address bus (ADDR), an eight bit data bus (DATA), three coder clocks (CLK0, CLK1, CLK2), three load pulses (LOAD0, LOAD1, LOAD2), three data lines and a clock for modulation (STIM0, STIM1, STIM2, and STLD), a chip select (CSN), and a register write control (WRN). Not shown in Figure 3.1 is the reset (RESET) line.

3.1.1.1 ADDR$_0$ - ADDR$_7$

The PRN Coder has an 8-bit address bus. The addressing scheme is shown in the tables below where A7 represents the Most Significant Bit (MSB) and A0 represents the Least Significant Bit (LSB)):

Table 3-1. Address bits ADDR$_7$ - ADDR$_5$.

Coder Selected	A7	A6	A5
Coder 0	0	0	0
Coder 1	0	0	1
Coder 2	0	1	0
All Coders	0	1	1
Code Combiner	1	0	0

Each independent code generator has the following internally programmable registers described below.

Table 3-2. Address bits ADDR$_4$ - ADDR$_0$.

Register Selected	A4	A3	A2	A1	A0
MASK	0	0	0	x_1	x_0
INIT	0	0	1	x_1	x_0
EPOCH	0	1	0	x_1	x_0
COUNT	0	1	1	x_1	x_0
MUX	1	0	0	0	0
CTL	1	0	0	0	1

Variables x_1 and x_0 are determined according to the table below such that the selected 32-bit register is sequentially loaded 8 bits at a time.

Table 3-3. Address bit ADDR₁ - ADDR₀ for 32 bit DATA.

Register bits loaded	x_1	x_0
bits 0-7	0	0
bits 8-15	0	1
bits 16-23	1	0
bits 24-31	1	1

3.1.1.1.1 Mask Register

The 32-bit PRN code generator generates codes with programmable lengths and polynomials. The polynomials are set by programming the desired taps in the MASK Register. Each bit in the MASK Register which is set to a logic "1" *enables* the corresponding tap in the PRN generator polynomial and a logic "0" *disables* the tap.

$$G = 1 + D_1(x) + D_2(x^2) + D_3(x^3)... + D_{31}(x^{31}) + D_{32}(x^{32})$$

Where D_1 = bit 0, D_2 = bit 1, D_3 = bit 2, etc.

The last tap set determines the effective length of the PRN generator register.

3.1.1.1.2 INIT Register

The 32-bit INIT Register is used to define the start value of the code generated in the PRN generator. The contents of the INIT Register are loaded into the PRN generator when a load command is issued or a control pulse occurs. A control pulse occurs when the function is enabled and a COUNT or EPOCH pulse occurs.

3.1.1.1.3 EPOCH Register

During every clock cycle, active bits of the PRN generator output are compared with the value stored in the EPOCH Register. The contents of the 32-bit EPOCH define the distinct code value to be detected. All bits in the EPOCH Register beyond the most

significant feedback tap in the PRN generator must always be set to "0", otherwise a match will never be detected.

3.1.1.1.4 COUNT Register

The 32-bit COUNT Register sets the counter stop point. Once the PRN generator starts, the counter runs for a number of clock pulses equal to the number stored in the COUNT Register and then generates a COUNT pulse. Contents of the COUNT Register are loaded into the PRN generator when a load command is issued or a control pulse occurs. A control pulse occurs when the control (CTL) function is enabled and a COUNT or EPOCH pulse occurs.

3.1.1.1.5 Phase MUX Register

The 5-bit Phase MUX Register selects the PRN generator register output bits used for the code output. The value stored in the MUX register is decoded and determines the tap number used. If the MUX register value is 00101, the code output will come from the 6th PRN generator register. For MUX Register value 00000, the 1st is used and a MUX Register value of 11111 corresponds to the 32nd tap being used.

3.1.1.1.6 CTL Register

The 8-bit CTL register determines the reloading of the coder and counter. The functions performed by the bits in the CTL register are shown in the following tables.

Table 3-4. Control (CTL) Register Bit Functions: (a) B_7-B_6, (b) B_5-B_4, (c) B_3-B_2, (d) B_1-B_0.

B_7	B_6	Function
0	0	Counter is not reloaded on any EPOCH pulse
0	1	Counter is reloaded on EPOCH$_0$ pulse
1	0	Counter is reloaded on EPOCH$_1$ pulse
1	1	Counter is reloaded on EPOCH$_2$ pulse

(a)

B_5	B_4	Function
0	0	Counter is not reloaded on any COUNT pulse
0	1	Counter is reloaded on COUNT$_0$ pulse
1	0	Counter is reloaded on COUNT$_1$ pulse
1	1	Counter is reloaded on COUNT$_2$ pulse

(b)

B_3	B_2	Function
0	0	PRN generator is not reloaded on any EPOCH pulse
0	1	PRN generator is reloaded on EPOCH$_0$ pulse
1	0	PRN generator is reloaded on EPOCH$_1$ pulse
1	1	PRN generator is reloaded on EPOCH$_2$ pulse

(c)

B_1	B_0	Function
0	0	PRN generator is not reloaded on any COUNT pulse
0	1	PRN generator is reloaded on COUNT$_0$ pulse
1	0	PRN generator is reloaded on COUNT$_1$ pulse
1	1	PRN generator is reloaded on COUNT$_2$ pulse

(d)

3.1.1.1.7 Code Combiner Lookup Register

The Code Combiner block shown in Figure 3-1 uses the MODCOD outputs of the three coders to address a lookup table, shown as Table 3-5, to produce the MIXCOD output. This allows the production of non-linear codes.

Table 3-5. Code Combiner Lookup Register Table.

MODCOD2	MODCOD1	MODCOD0	MIXCOD
0	0	0	Look-up Register Bit 0
0	0	1	Look-up Register Bit 1
0	1	0	Look-up Register Bit 2
0	1	1	Look-up Register Bit 3
1	0	0	Look-up Register Bit 4
1	0	1	Look-up Register Bit 5
1	1	0	Look-up Register Bit 6
1	1	1	Look-up Register Bit 7

3.1.1.2 $DATA_0 - DATA_7$

The 8-bit DATA bus writes data into the registers. $DATA_7$ is the most significant bit (MSB) and $DATA_0$ is the least significant bit (LSB). The 8-bit address directs the data to the proper register.

3.1.1.3 Reset

When the reset input is set low, all of the registers inside the PRN generator are reset to zero. Normal operation will not commence until an initialization value has been loaded into the coders from their corresponding INIT registers.

3.1.1.4 $CLK_0 - CLK_2$

These are the clocks for coder 0 through coder 2 respectively. All operations occur on the rising edges of the clocks, with the exception of the early and late outputs, which change on the falling edges of CLK_0. The clocks should nominally be square waves, with a maximum frequency of 30 MHz.

3.1.1.5 WRN and CSN

The register write control (WRN) is normally high. When this line goes low, data is written into the register(s) selected by the address lines and latched on the rising edge of WRN. The Chip Select (CSN) must also be low to enable the data loading.

3.1.1.6 LOAD$_0$ - LOAD$_2$

On the rising edge of the clock following the falling edge of a LOAD input, a load command is issued. This will cause the corresponding coder register and counter to be loaded with the contents of the corresponding INIT and COUNT Register, respectively.

3.1.1.7 STIM$_0$ - STIM$_2$ and STLD

Data applied to the STIM$_0$ - STIM$_2$ inputs is modulo-2 added with the outputs of the corresponding coder (Coder$_0$ - Coder$_2$). The data is latched in on the falling edge of the STLD input.

3.1.2 STEL-1032 Outputs

As shown in Figure 3-1, the STEL-1032 has a total of 25 outputs. The code combiner outputs XOR01, XOR012, MIXCOD, EARLY, LATE, and PUNCT. Each coder outputs a CODE, EPOCH, COUNT, MODCOD, LDSYNC, and REF$_{15}$. An STSYNC signal comes from Coder0.

3.1.2.1 $CODE_0$ - $CODE_2$

The $CODE_0$ - $CODE_2$ outputs are the outputs of the $Coder_0$ - $Coder_2$. The register bit in the coder from which the output is derived is set by the Phase MUX.

3.1.2.2 $MODCOD_0$ - $MODCOD_2$

The $MODCOD_0$ - $MODCOD_2$ outputs are the $CODE_0$ - $CODE_2$ signals after modulation by the $STIM_0$ - $STIM_2$ inputs. The register bit in the coder from which the output is derived is set by the Phase MUX.

3.1.2.3 $LDSYNC_0$ - $LDSYNC_2$

The LoaDSNYC output goes low for one clock cycle after the contents of the corresponding INIT register been loaded into the corresponding coder. This pulse indicates the clock cycle in which the coder value is identical to that of the INIT register.

3.1.2.4 $REF15_0$ - $REF15_2$

The $REF15_0$ - $REF15_2$ outputs are the reference codes derived from the taps number 15 of the corresponding coder. These outputs will be identical to the corresponding $CODE_0$- $CODE_2$ outputs when the data stored in the corresponding Phase MUX is 01111 (tap number 15).

3.1.2.5 $EPOCH_0$ - $EPOCH_2$

The $EPOCH_0$ - $EPOCH_2$ outputs are normally high and go low whenever the corresponding coder code is equal to the code stored in the EPOCH Register for that coder. This condition will not be detected and the EPOCH output will not go low if this

condition occurs within 2 clock cycles of the rising edge on the corresponding EPOCH input.

3.1.2.6 COUNT$_0$ - COUNT$_2$

The COUNT$_0$ - COUNT$_2$ outputs are delayed replicas of the LDSYNC$_0$ - LDSYCN$_2$ outputs. The length of the delay is equal (in clock cycles) to the value stored in the corresponding COUNT Register.

3.1.2.7 XOR$_{01}$ and XOR$_{012}$

The XOR$_{01}$ output is the result of the modulo-2 addition (XOR) of CODE$_0$ and CODE$_1$ signals. The XOR$_{012}$ output is the result of modulo-2 addition of the CODE$_0$, CODE$_1$, and CODE$_2$ signals. The results are delayed by one clock cycle before appearing on the XOR$_{01}$ and XOR$_{012}$ outputs.

3.1.2.8 MIXCOD

The MODCOD$_0$ - MODCOD$_2$ signals are used to address the Code Combiner Lookup Register, shown above in Table 3-5. The data bit stored in the location addressed by the three bits of the MODCOD$_0$ - MODCOD$_2$ is the MIXCOD output.

3.1.2.9 PUNCT, EARLY, and LATE

The PUNCT output is an exact replica of the MIXCOD output delayed by one clock cycle. The EARLY output is an exact replica of the PUNCT output advanced by a half clock cycle. The LATE output is an exact replica of the PUNCT output delayed by half a clock cycle. This is achieved by clocking the signals into the output register on the

falling edges of CLK_0. In order to make the advance and delay exactly half a clock cycle, the duty cycle of CLK_0 must be exactly 50%.

3.1.2.10 STSYNC

The STSYNC output is normally high and goes low for one clock cycle following a falling edge on the STLD input signal. The result of the new modulation bits will appear on the output codes $MODCOD_0$ - $MODCOD_2$ during this clock cycle.

3.2 Code Generator Addressing Design

An 8-bit address is utilized by the STEL-1032 to ensure that data is directed into the proper register. The addressing scheme is shown in Tables 3-1 through 3-3. Several methods were considered when designing the address interface. By examining the addressing tables, it can be seen that the address can be broken into three separate blocks, coder selection, register selection, and 32/8 bit data selection. The 32-bit registers are loaded in 8-bit groupings with the two least significant bits of the address ($ADDR_0$ and $ADDR_1$) designating which 8-bit group is being loaded. When 32-bit data is being loaded, a 2-bit binary counter is employed to toggle through the $ADDR_0$ and $ADDR_1$ combinations. When 8-bit data is being loaded, $ADDR_0$ and $ADDR_1$ are fixed. After the 8-bit address is formed, it is sent to two sets of octal buffers (part number SN74LS244) as shown in Figure 3-4, where in each set only one buffer is enabled at a time. If the code combiner lookup table was selected, octal buffer number 2 is enabled and its 8-bit address

is passed on; otherwise octal buffer number 1 is enabled and the formed address is passed

on to octal buffer number 3. When the code generator is operating in manual entry mode

the address is passed on to the STEL-1032 via octal buffer number 3. If external

programming is being performed, octal buffer number 4 passes the address. The design

of the coder selector, register selector and 32/8 bit data selection is described below.
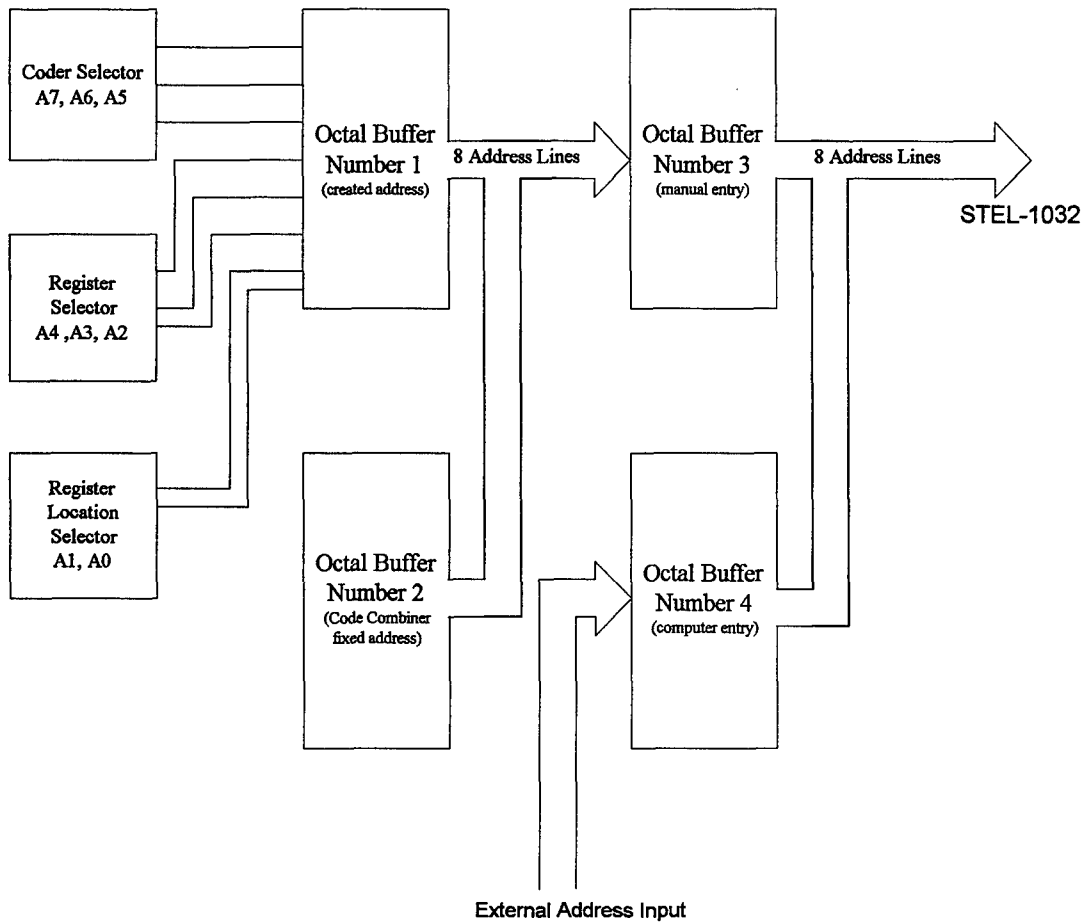
Figure 3-4. Address Block Diagram.

3.2.1 Coder Selector Design

As shown in Table 3-1, the coder selector has five options and occupies the three most significant bits of the address $(ADDR_7 - ADDR_5)$. The coder selector takes one of five choices and outputs the three bits corresponding to the address of the coder selected. A shown in Table 3-6, an 8-line to 3-line priority encoder (part no. SN54148) produces a 3-bit output based on the location of the first low level logic encounter. Figure 3-5 shows how this TTL device was used in the coder selector design.

Table 3-6. 8-Line to 3-Line Priority Encoder Function Table.

INPUTS								OUTPUTS		
0	1	2	3	4	5	6	7	A2	A1	A0
H	H	H	H	H	H	H	H	H	H	H
x	x	x	x	x	x	x	L	L	L	L
x	x	x	x	x	x	L	H	L	L	H
x	x	x	x	x	L	H	H	L	H	L
x	x	x	x	L	H	H	H	L	H	H
x	x	x	L	H	H	H	H	H	L	L
x	x	L	H	H	H	H	H	H	L	H
x	L	H	H	H	H	H	H	H	H	L
L	H	H	H	H	H	H	H	H	H	H
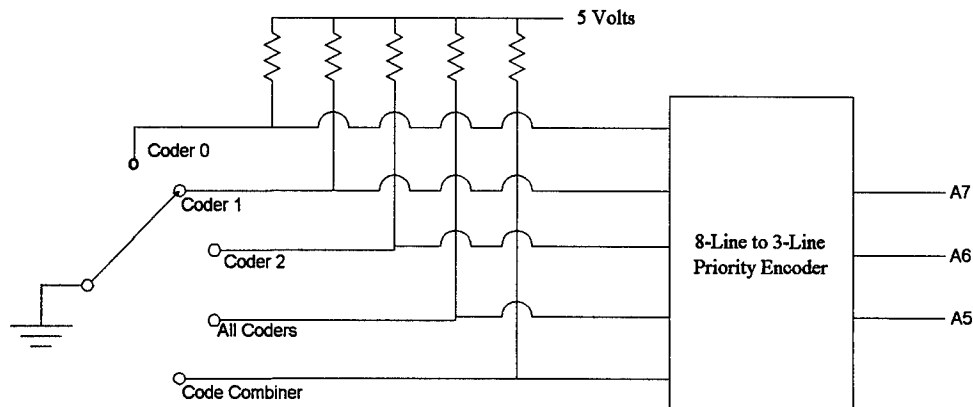
Figure 3-5. Coder Selector Design.

Using a 2.2KΩ resistor as a pull-up resistor, the input lines remain a logic high until selected. The coder selector switch is a push button switch tied to ground which allows one selection at a time. By selecting a coder, the corresponding line is grounded and a logic low appears on the input line and $ADDR_7$ - $ADDR_5$ are output from the encoder in accordance with Table 3-6.

3.2.2 Register Selector Design

As shown in Figure 3-6, the register selector is almost identical to the coder selector. The difference is there are six registers to select and only five combinations of $ADDR_4$ - $ADDR_2$ are output. The MUX and CTL, 8-bit data registers, share $ADDR_4$ - $ADDR_2$ but have different $ADDR_1$ and $ADDR_0$ bits. The MUX and CTL lines are brought through an AND gate; if either register is selected the output of the AND gate is low and input to the encoder otherwise the line is high.

Figure 3-6. Register Selector Design.

3.2.3 32 or 8 bit Data Address

Since the STEL-1032 has an 8-bit address and data bus, the 32-bit registers must be loaded in four 8-bit blocks, each with a different address. As shown in Table 3-3, the two Least Significant Bits (LSBs) of the address,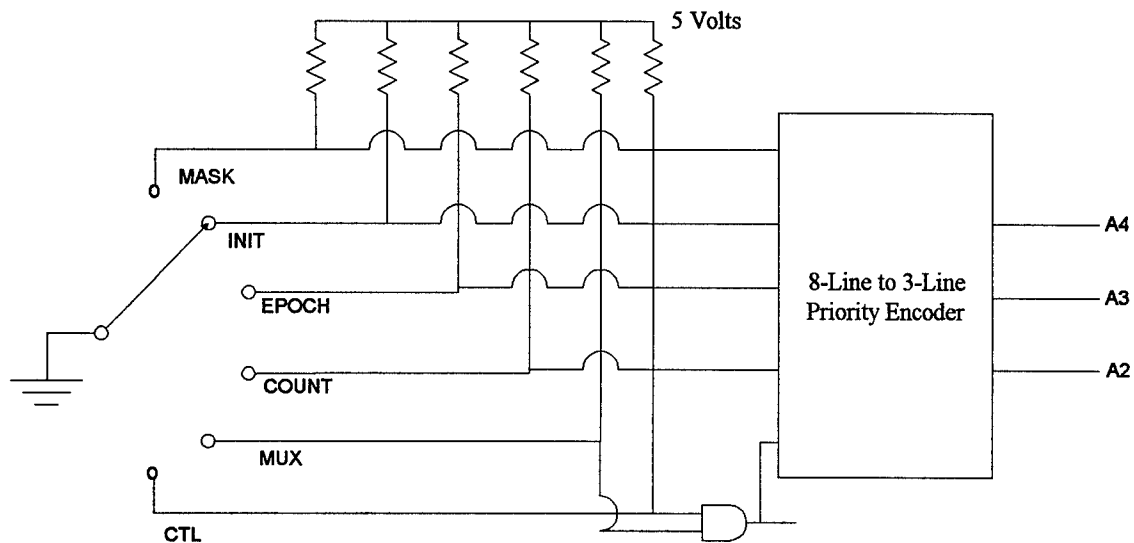 $ADDR_1$ and $ADDR_0$, direct the 32-bit data into the proper location in the register. In the 8-bit registers, $ADDR_1$ and $ADDR_0$ are fixed. Figure 3-7 shows the design for generating the $ADDR_1$ and $ADDR_0$ bit required for 32-bit data entry. With the load switch depressed, the reset on the clock (part number 555) goes high and the clock begins to run. Resistor and capacitor values are chosen to produce a high-to-low square wave clock with a frequency of about 1 Hz. This allows the address and data to stabilize before being loaded in to the PRN coder. Waveform timing is presented in Section 3.4. The clock output increments the binary counter and triggers the one-shot multivibrators used to load the registers. The clock is controlled by the reset input; it continues to run as long as the reset line is held high. Because it was desirable to have the clock run for 4 cycles and toggle through the various $ADDR_1$ and $ADDR_0$ combinations, several logic gates are employed to control clock operation.   The following conditions must simultaneously occur to force the reset line and stop the clock: a) both A_1 and A_0 from the binary counter go high, b) the clock transitions to a low state, and c) the load switch is released.
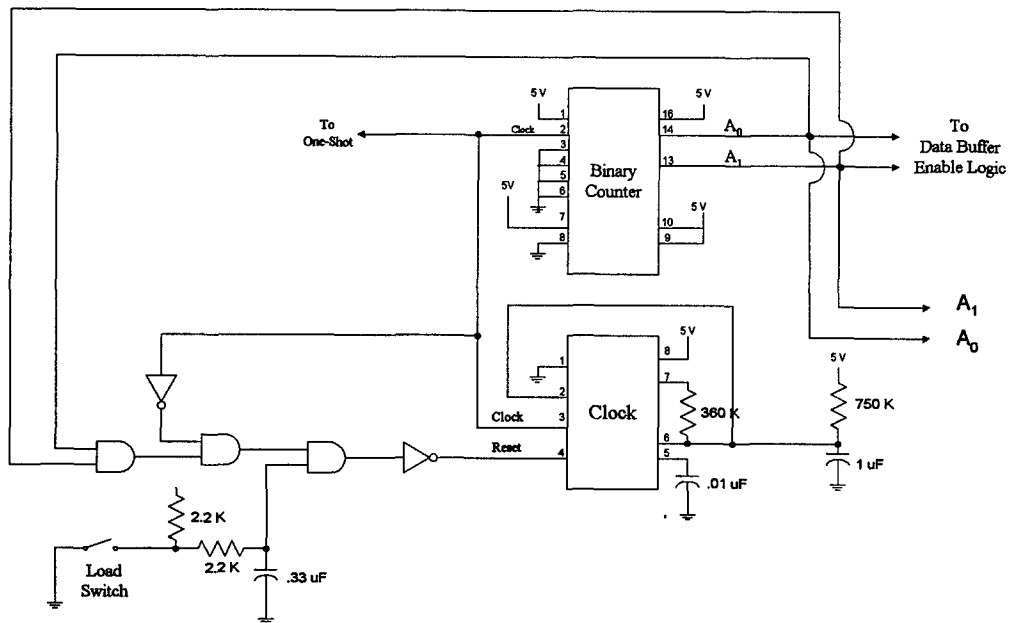
Figure 3-7. Clock and Counter Design.

As shown in Figure 3-8, when 32-bit data is being loaded, address bits $ADDR_1$ and $ADDR_0$ equal the counter output; when the MUX or CTL Registers are being loaded, $ADDR_1$ and $ADDR_0$ are fixed.
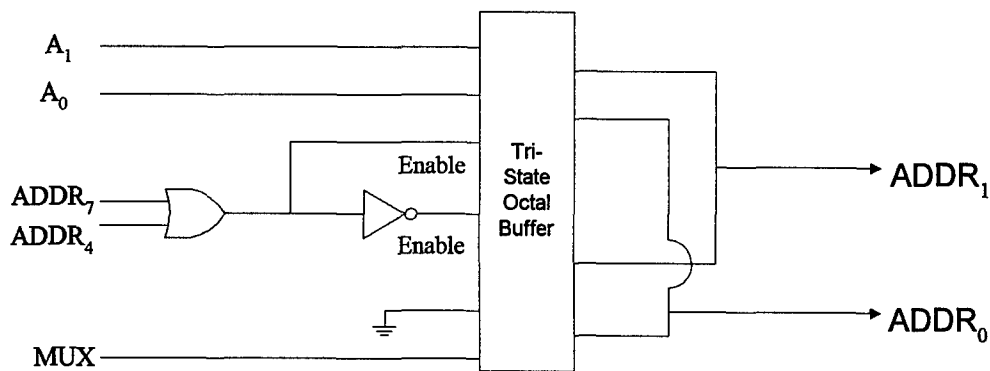
Figure 3-8. ADDR1 and ADDR0 Generation.

3.3 Code Generator Data Entry Design

Since the STEL-1032 has eight data line inputs, several options were considered for data entry. As discussed in the previous sections, the entire 32 bits of data may be entered on the thumbwheel switches but through TTL logic gates, only eight bits are loaded into the registers at a time. Figure 3-9 shows an overview of the data entry design. Data is entered via 11 octal thumbwheel switches. The 33 data lines are routed to three different banks of octal buffers. The first set of four octal buffers are connected to the 32 Least Significant Bits (LSB) of the thumbwheel switches for entering data into the following 32-bit registers: INT, EPOCH, and COUNT. The second set of buffers contains a single octal buffer connected to the LSB on each of the eight right most thumbwheel switches. This allows the 8-bit data to be entered in binary format. The third set contains four octal buffers connected to the 32 Most Significant Bits (MSB) of the 11 thumbwheel switches. This allows data to be entered into the MASK Register.

Figure 3-9. Data Entry Design Block Diagram.

3.3.1 Data Buffer Enable Design

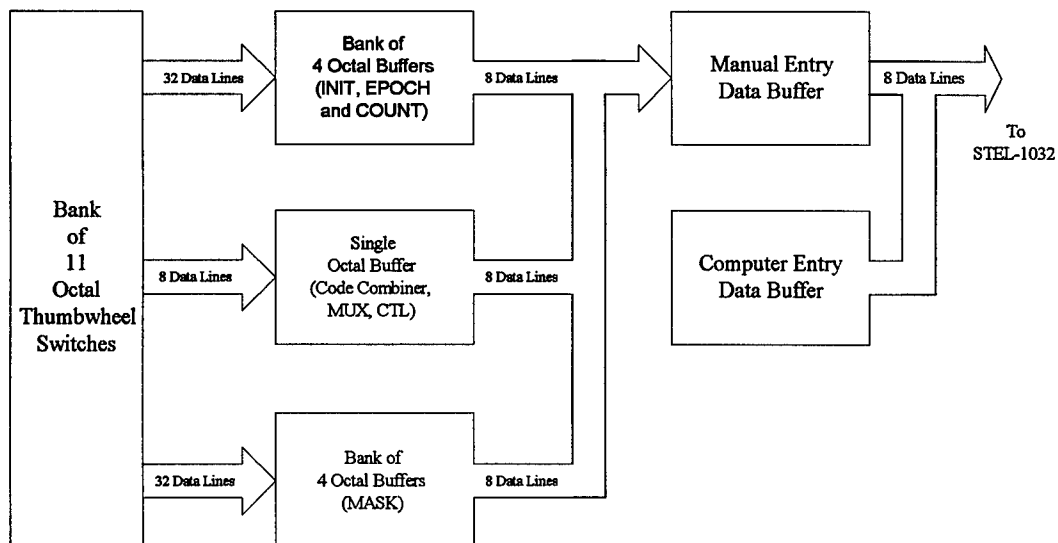The octal buffers are tri-state devices with output characteristics as shown in Table 3-7. To ensure data is properly loaded into the desired register two things must occur, the correct buffer must be enabled and the proper address must be present.

Table 3-7. Tri-State Octal Buffer Outputs.

Enable	Input	Output
L	L	L
L	H	H
H	X	HI-Z

As shown in Figure 3-10, TTL logic and the binary counter outputs used to set ADDR1 and ADDR0 lines ensure the octal buffers are enabled in the proper order. For the data to be correctly entered, only one buffer enable can be low at any time; logic gates prevent two or more buffers from being enabled at one time. The Code Combiner and MUX / CTL lines are normally low but if one is chosen the line transitions high and passes through the OR gate. The high level inverts and the low output level enters the AND gates keeping them low and disabling the 32-bit buffers. When 32-bit data is loaded, through the use of inverters and AND gates the binary counter output enables one buffer for each count. The OR gates in the 32-bit buffer enables are controlled by the MASK selection. If the MASK is selected, the line to the MASK buffer OR gates is low and its buffers are enabled by the counter otherwise it remains in a high state and the OR gate output remains in a high state disabling the buffers.
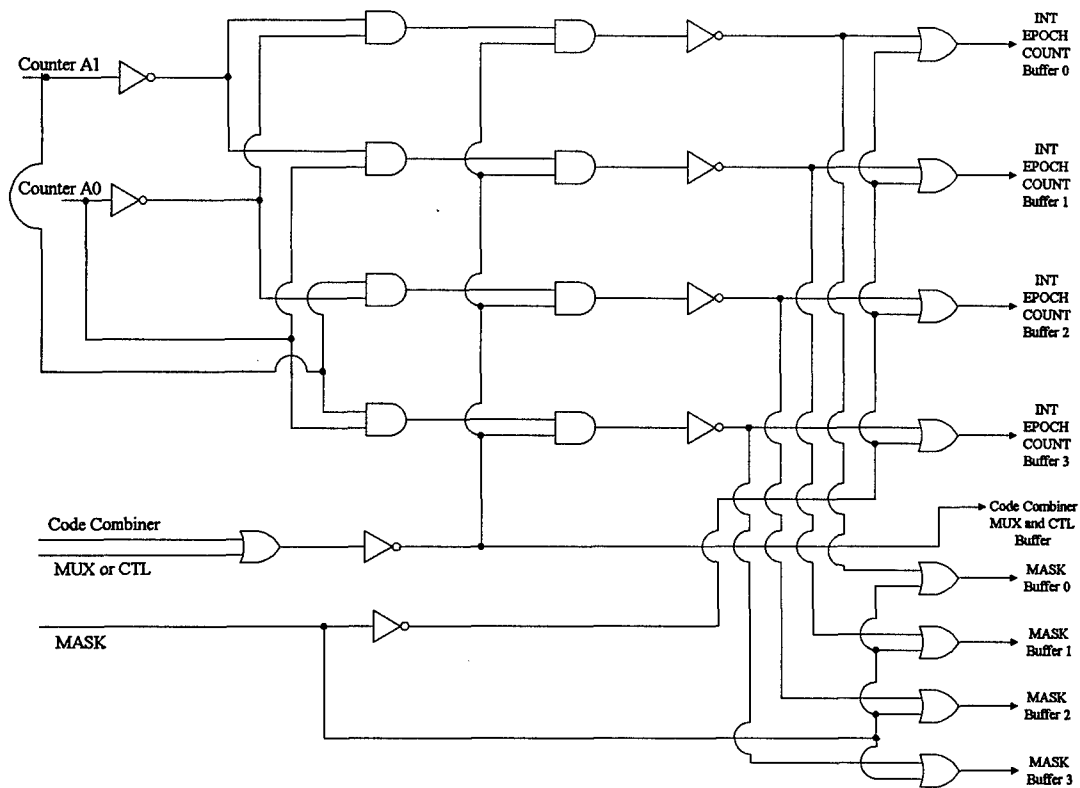
Figure 3-10. Data Buffer Enabling Design.

3.3.2 Data Entry Format

Data enters via 11 octal thumbwheel switches with the LSB occupying the rightmost position. Since data is loaded into registers with the LSB first, data is set on the rightmost thumbwheel and work to the left. The 8-bit data (Code Combiner, MUX, and CTL) is entered in binary format on the eight rightmost thumbwheels with the LSB on the right. The 32-bit data is entered in octal format. The binary-to-octal conversion is accomplished by grouping the binary bits into groups of three starting with the LSB and working toward the MSB. A group of three bits are converted to their octal equivalent and entered on the thumbwheel switches starting on the right with the least significant

3-21

octal number. For example, if the initial fill of a 17 stage register was to be 1 0 1 1 0 0 1 1

0 0 0 1 0 1 1 0 1 with the last bit to be entered into the B_0 register, the thumbwheels

would be set at 0 0 0 0 2 6 3 0 5 5. Octal thumbwheels were chosen because tables are

readily available with the octal representation of primitive polynomials that produce

maximal length sequences. Appendix A includes a partial list these octal representations.

3.4 Code Generator Data Loading

To load data into the STEL-1032, sequential Chip Select (CSN) and Write Enable

(WRN) pulses must occur following data and address stabilization. A low level CSN

input enables the loading of data via the data lines and when the WRN line is low data is

written into the register(s) selected by the address lines; the data is latched on the rising

edge of the WRN pulse. Figure 3-11 shows the how these pulse are created. Dual non-

retriggerable one-shot multivibrators (part number 74LS221) are used to generate the

necessary pulses.
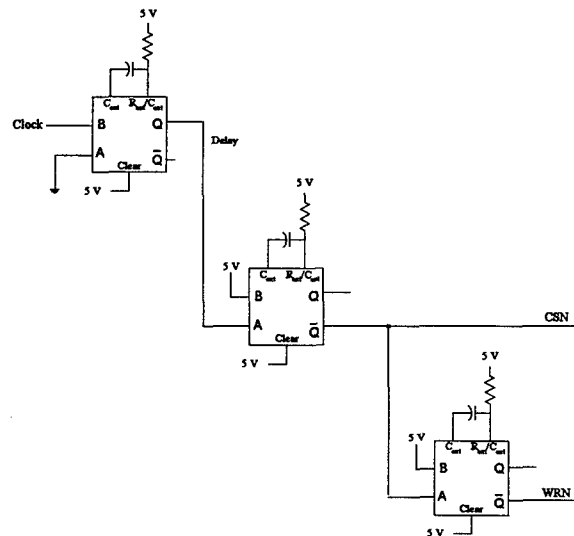
Figure 3-11. Load Pulse Design.

As discussed in Section 3.3, after the LOAD button is depressed the clock is triggered incrementing the counter that enables the buffers and sets ADDR1 and ADDR0. The clock pulse also triggers the first multivibrator which serves as a delay to allow the address and data to stabilize. On the falling edge of the delay pulse, the second multivibrator is triggered producing the CSN pulse. The falling edge of the CSN pulse triggers a third multivibrator, producing the WRN pulse. The total delay between the CSN and WRN pulses is the internal delay from the multivibrator. The pulse width are controlled by the proper selection of resistors and capacitors. Figure 3-12 shows the relationship of these pulses.

Figure 3-12. Data Loading Pulse Comparison.

After data is loaded into the registers, a LOAD pulse is necessary to fill the PRN Generator with values stored in the INIT register. On the rising edge of the clock pulse

3-23

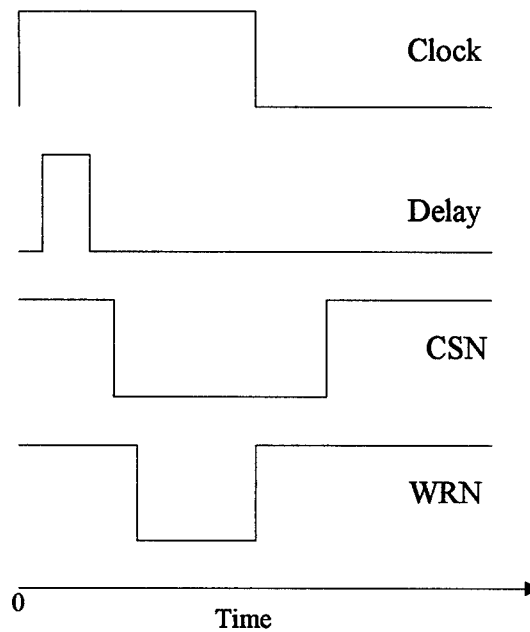following the falling edge of a LOAD pulse, the PRN Generator register and counter register are filled with the contents of the INIT and COUNT registers, respectively. A momentary push button switch is connected to a multivibrator to generate the LOAD pulse.

3.5 Other Code Generator Inputs

There are eight other inputs available on the code generator; a reset, three clocks, three external data lines and a data clock. The code generator is reset via a momentary push button switch which sets all register contents inside the PRN coder to zero. The three PRN coders are clocked independently (clock inputs are provided via a BNC connector on the front of the code generator). Data applied to the STIM inputs via BNC connectors are modulo-2 added with the outputs of the corresponding PRN coder. Data is latched on the falling edge of the signal on the BNC connector to the STLD input.

3.6 Code Generator Outputs

The code generator has a total of 25 outputs, the 15 waveforms and 10 pulses previously described in section 3.1.2. Each output is connected as shown in Figure 3-13. A 50Ω quad 2-input NOR gate line driver (SN74128) was chosen to interface between the code generator and front panel. The output of the STEL-1032 is inverted before entering the line driver inputs since it also inverts the output. The result is a waveform or pulse of proper orientation.

Output
From
STEL-1032

To BNC
Connector
on Front Panel

50 ohm
line driver
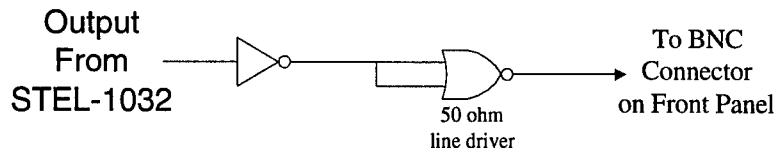
Figure 3-13. Output Design.

3.7 Code Generator Construction

The code generator was constructed using a 15" x 17" x 4" box. The layout is

shown in Figure 3-14. The 120VAC power is passed through a line filter and fuse before

going to the front panel power switch. From the switch, the 120VAC enters the system

5VDC power supply. The chassis ground and the 0Vdc are tied together because the

front panel BNC connector are grounded to the chassis. The front panel, shown in Figure

3-15, contains 12 octal thumbwheel switches, 32 BNC connectors, two 6-button selectors,

three momentary push button switches, one LED, two toggle switches, and a power

switch with an indicator light. The wire-wrap board shown in Figure 3-16 contains 45

chips along with various resistors and capacitors.
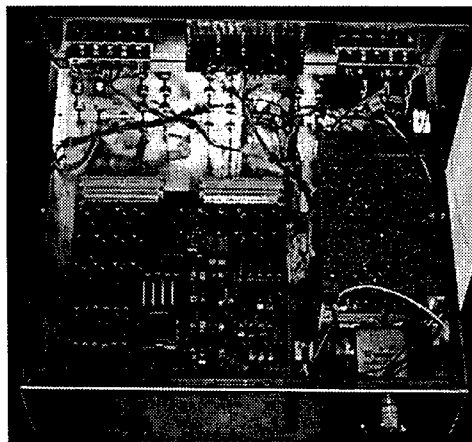
Figure 3-14. Code Generator Case Layout.
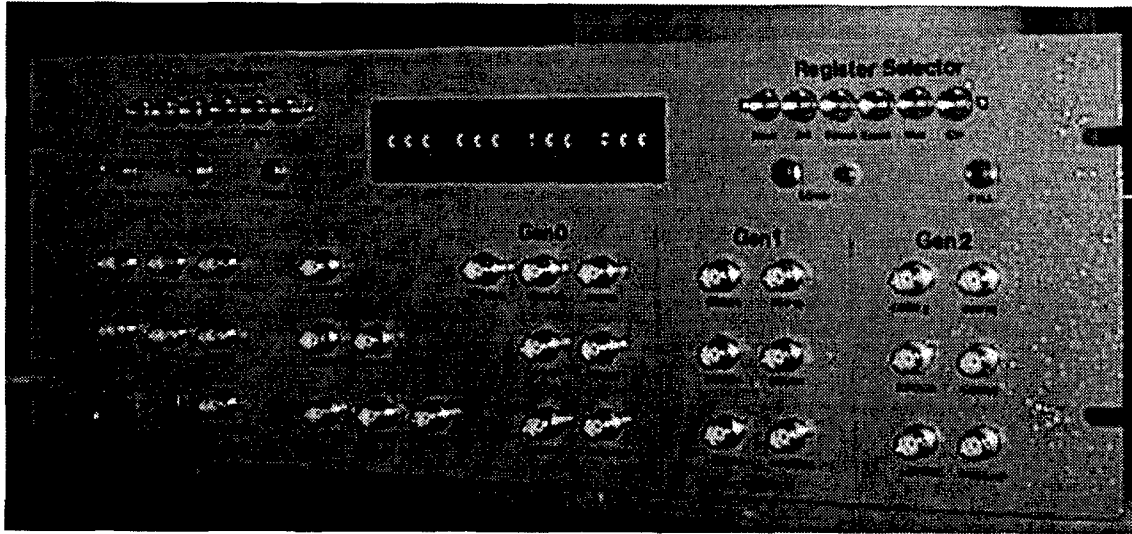
Figure 3-15. Code Generator Front Panel.
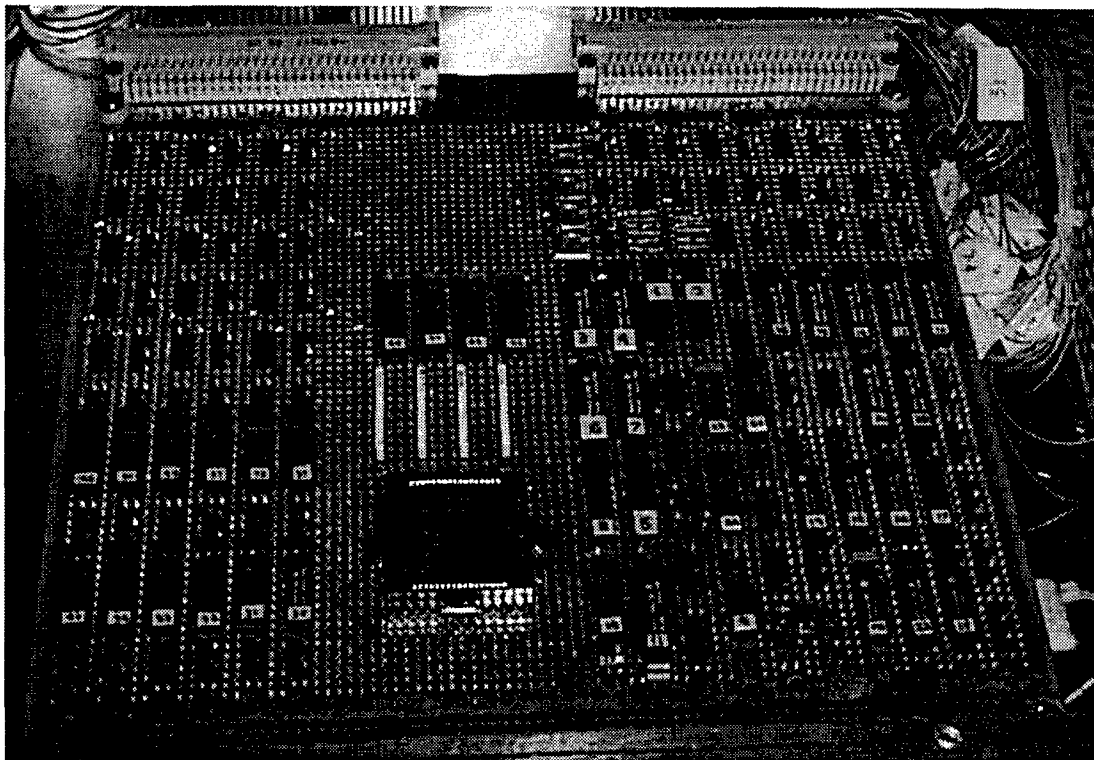
Figure 3-16. Code Generator Circuit Board.

3.8 Code Generator System Design Summary

The interface design consists of several TTL logic gates. Programming the code generator is accomplished via the front panel. Depressing the LOAD button causes the data entered on the thumbwheel switches to be loaded into the STEL-1032 in accordance the coder(s) and register(s) selected. With construction of the code generator complete, the outputs presented in Section 3.1.2 are authenticated in Chapter 4.

4. Code Generator Evaluation

4.1 Code Generator Loading Pulses

Before code generation begins, initialization data is loaded into the STEL-1032 Pseudorandom Number (PRN) coder registers. Data is loaded into the coder registers using sequential Chip Select (CSN) and Write Enable (WRN) pulses which occur following data and address stabilization. As discussed in Section 3.2.3, a clock circuit increments a counter to enable the data buffer and establish the address. A set of one-shot multivibrators are sequentially triggered to generate the CSN and WRN pulses. The following equation is used to determine the proper resistor and capacitor values to obtain the desired high-to-low square wave clock period. Period $= ln(2) * (R_A + 2R_B) * C$. For R_A equal to 750 KΩ, R_B equal to 360 KΩ, and C equal to 1 μF, the period is 1.019 seconds or .98 Hz. The actual data loading clock circuit output is shown in Figure 4-1.
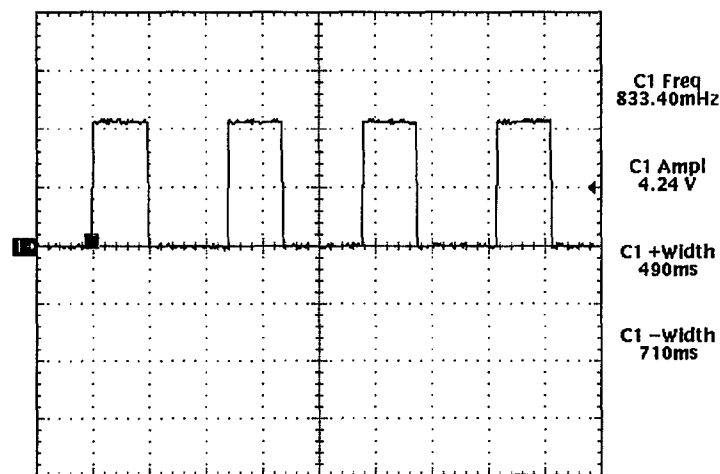

Figure 4-1. Code Generator Data Loading Clock.

As shown, the period of the actual clock is 1.2 seconds. The rising edge of the data

loading clock triggers a one-shot multivibrator, the resultant output pulse serves as a delay

which triggers another one-shot multivibrator, generating a CSN pulse. The CSN pulse

then triggers another one-shot multivibrator, producing the WRN pulse. Figure 4-2 shows

the actual pulses for loading the data. The data loading clock is displayed as Reference

number 1 (R1), the CSN pulse is on Channel 1 (1), and the WRN pulse is on Channel 2

(2). The delay between the rising edge of the clock and the CSN pulse allows the

STEL-1032 data and address inputs to stabilize prior to loading. The delay time, CSN

and WRN pulse widths are determined by the following equation. Width = $ln(2) * R *C$.

For implementation, 150 KΩ resistors are used. A delay capacitance of 1 μF, CSN

capacitance of 4.7 μF, and WRN capacitance of 2.2 μF are used to provide a delay of 104

milliseconds, a CSN pulse width of 487 milliseconds and a WRN pulse width of 229

milliseconds, respectively.

C1 –Width
484.0ms
Low signal
amplitude

C2 –Width
240.0ms

Ref1→C1 Dly
112.0ms
Low signal
amplitude
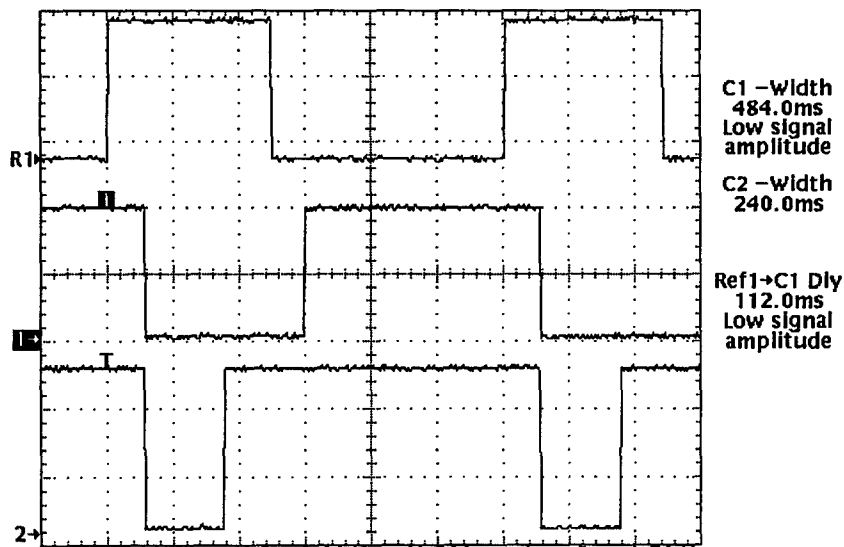
Figure 4-2. Data Loading Clock, CSN, and WRN Pulses.

4.2 Coder Outputs

The STEL-1032 code generator consists of four major components, three independent coders (Coder0, Coder1, and Coder2) and a code combiner (See Figure 3-1). The results presented in this section are derived from tests performed on Coder0 and are representative of outputs for all three coders. The following outputs are authenticated: pseudorandom code, LDSYNC, COUNT, EPOCH, REF_{15}, and MODCOD. Each test is performed using the 3-stage linear feedback shift register configuration shown in Figure 4-3.
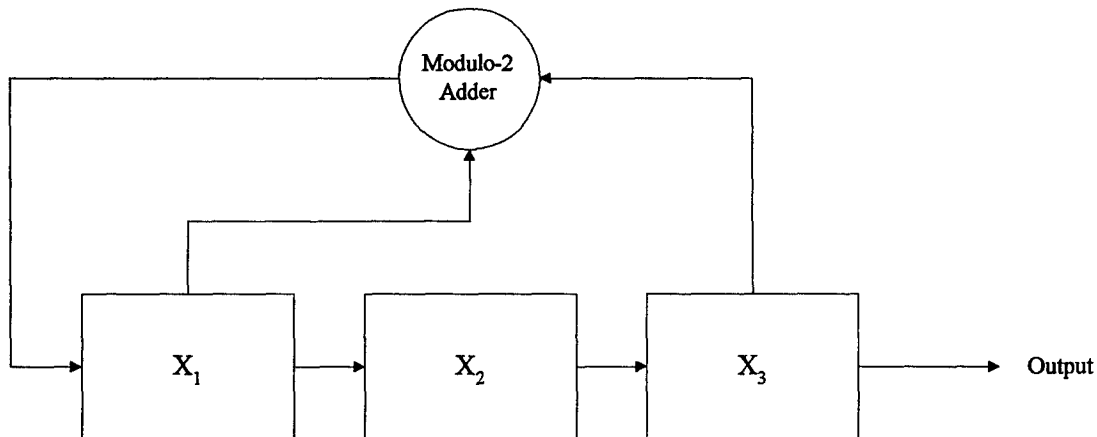
Figure 4-3. Test Linear Feedback Shift Register.

The linear feedback shift register of Figure 4-3 may be represented by a polynomial of the form $X^3 + X + 1$; the binary representation is (1 0 1 1) and octal representation which is set on the thumbwheel switches is [1 3]. The register contents following each clock cycle are shown in Table 4-1 where all of the registers are initially filled with a '1'.

Table 4-1. Linear Feedback Shift Register State Values.

| Register Contents			
State	X_1	X_2	X_3
0	1	1	1
1	0	1	1
2	1	0	1
3	0	1	0
4	0	0	1
5	1	0	0
6	1	1	0
7	1	1	1

At the seventh state (after seven clock cycles), the register contents are the same as the initial conditions and the output begins to repeat. The linear feedback shift register generates a code of length 7, a maximal length sequence for this three stage shift register configuration since $2^3 - 1 = 7$ (See Section 2.2.1). This sequence may seem trivial but is easy to analyze; therefore it is used throughout the testing of the code generator. The testing techniques and results presented here are extendible to any length code produced by the code generator.

4.2.1 Pseudorandom Output Code Demonstration

By loading the MUX register with the 5-bit data [0 0 0 1 0], a pseudorandom code is output from the third stage of Coder0. From Table 4-1, the expected output is shown as (1 1 1 0 1 0 0 . . .). The MASK, INIT and MUX Registers are loaded as previously described and an external 1 KHz clock is applied to the Coder0 clock input. The Coder0

pseudorandom output, CODE0, is shown on Channel 1 along with the coder input clock
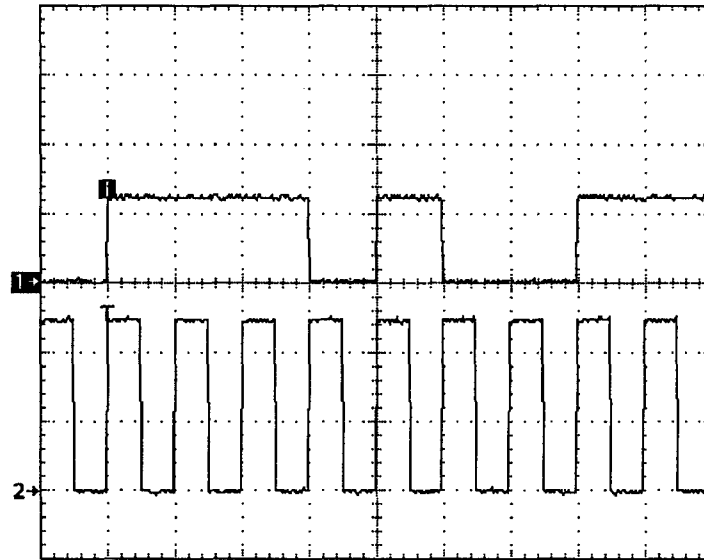
on Channel 2 in Figure 4-4.

Figure 4-4. CODE0 and Input Clock.

The code appears as expected but it is difficult to determine the period of the code

from Figure 4-4. By changing the time scale as shown in Figure 4-5, the length of the

code is easily determined as 7, i.e., 7 input clock cycles per code period.
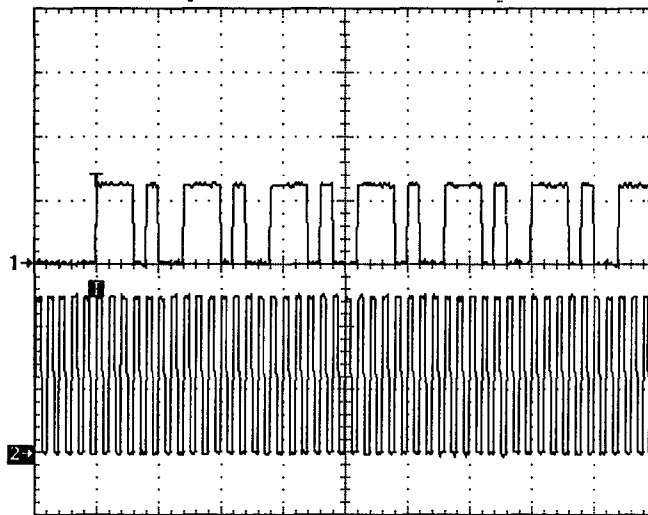
Figure 4-5. Multiple Periods of CODE0 and Clock Pulse.

4.2.2 LDSYNC Pulse Output Demonstration

After the contents of the INIT register are first loaded into the coder, the "Load Synchronization" LDSYNC output is expected to go low for one clock cycle. Shown in Figure 4-6, CODE0 is displayed on Channel 1 and the expected LDSYNC pulse is shown on Channel 2.
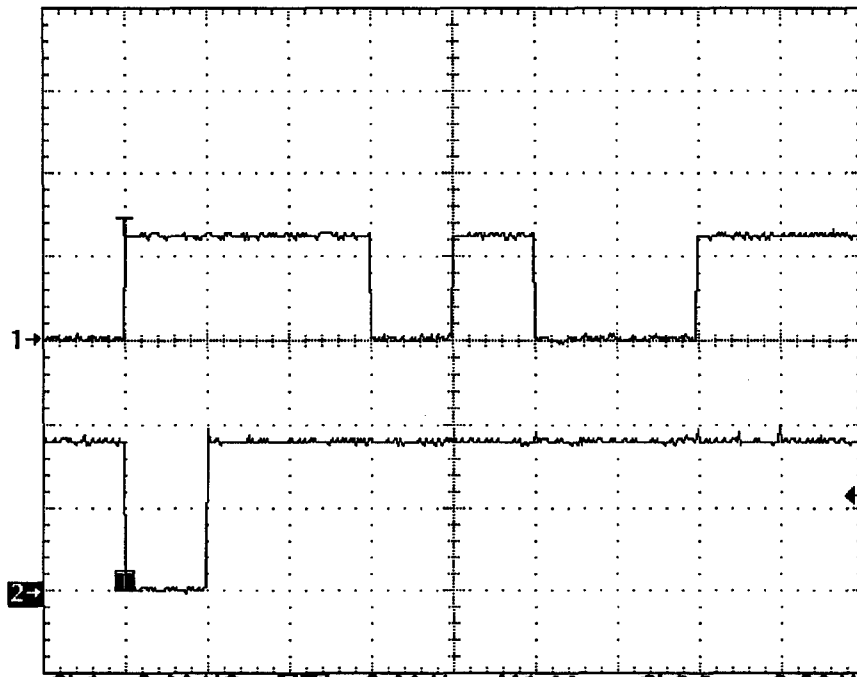
Figure 4-6. CODE0 and LDSYNC Pulse.

4.2.3 COUNT Pulse Output Demonstration

The COUNT pulse is a delayed replica of the LDSYNC. The amount of delay is determined by the value loaded into the count register. By loading the count register with [8], a low LDSYNC pulse is expected to occur on the eighth clock cycle, the first bit of the second period in CODE0. As shown in Figure 4-7, the COUNT pulse output appears as expected on the lower trace of the display.
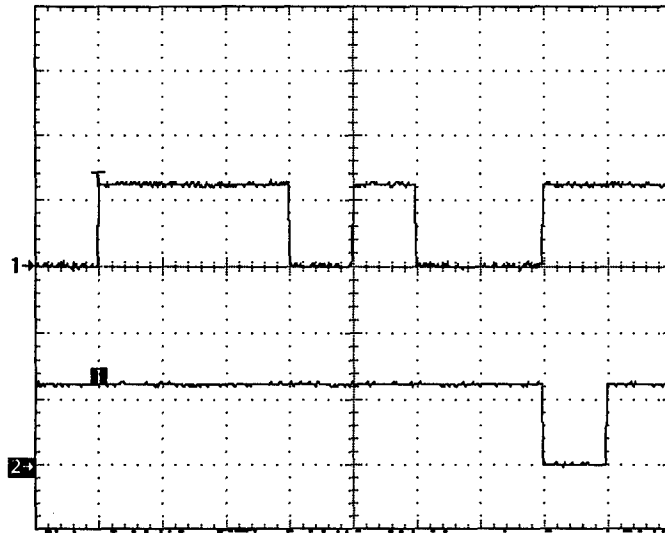
Figure 4-7. CODE0 and COUNT Pulse.

4.2.4 EPOCH Pulse Output Demonstration

The EPOCH output is normally high and goes low whenever the coder register contents are equal to the value set in the EPOCH register. Loading [4] in the EPOCH register represents stage $X_3 = 1$, $X_2 = 0$, and $X_1 = 0$. From Table 4-1, this condition occurs at state 4, the fifth output bit. Figure 4-8 shows the resultant EPOCH pulse output relative to CODE0 on traces 2 and 1, respectively.
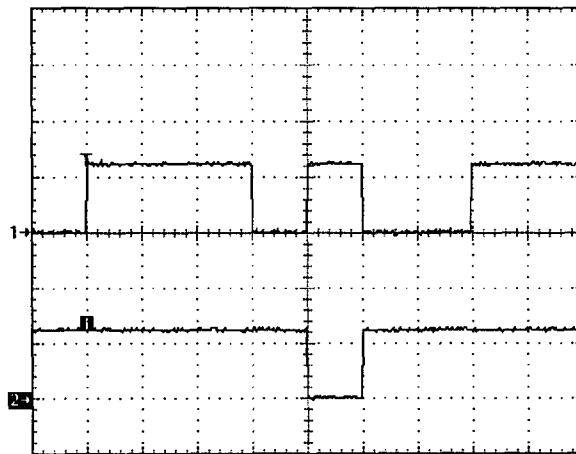
Figure 4-8. CODE0 and EPOCH Pulse.

4-7

4.2.5 Pseudorandom Code at REF15 Demonstration

The REF15 output is the pseudorandom code taken from tap number 15 (the 16th

tap, taps number from 0 to 31 per Figure 3.3). The output is identical to the CODE

output if the MUX is loaded with (0 1 1 1 1) which corresponds to tap number 15.

Figure 4-9 shows the CODE0 (third tap) on Channel 1 and the REF15 output on

Channel 2. The REF15 output remains in a low state for thirteen clock cycles which

means the REF15 output is thirteen stages from the CODE0 (tap number 2) output. This

confirms that loading the MUX with a binary fifteen (0 1 1 1 1), CODE0 output taken

from tap number 15, CODE0 and REF15 outputs are identical. This is contrary to

information provided by the vendor. [9]
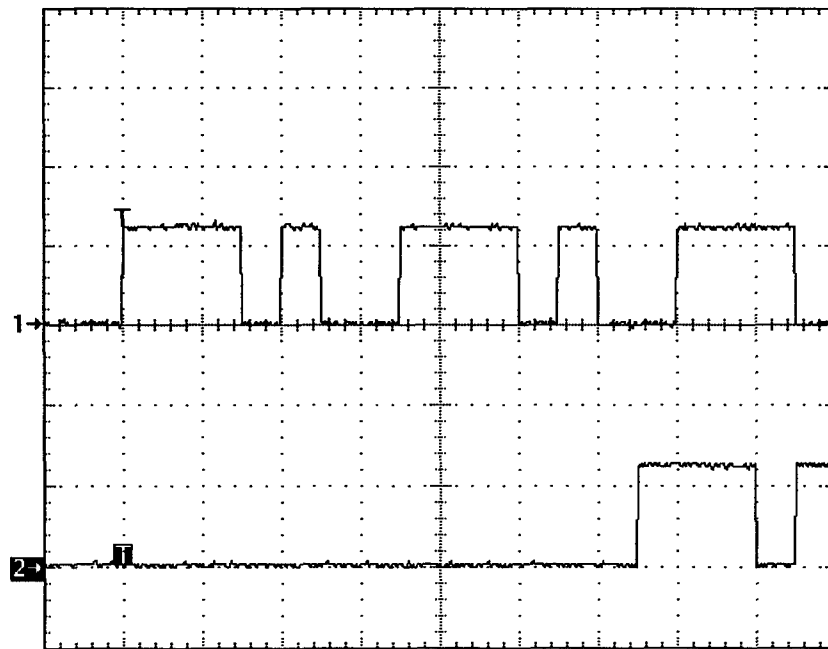
Figure 4-9. CODE0 at Tap Number 2 and REF15.

4.2.6 MODCOD Output Waveform Demonstration

The MODCOD is the CODE signal modulated by the STIM inputs and delayed by one clock cycle before appearing on the output. Figure 4-10 shows CODE0 and the MODCOD when the STIM input is '0'.
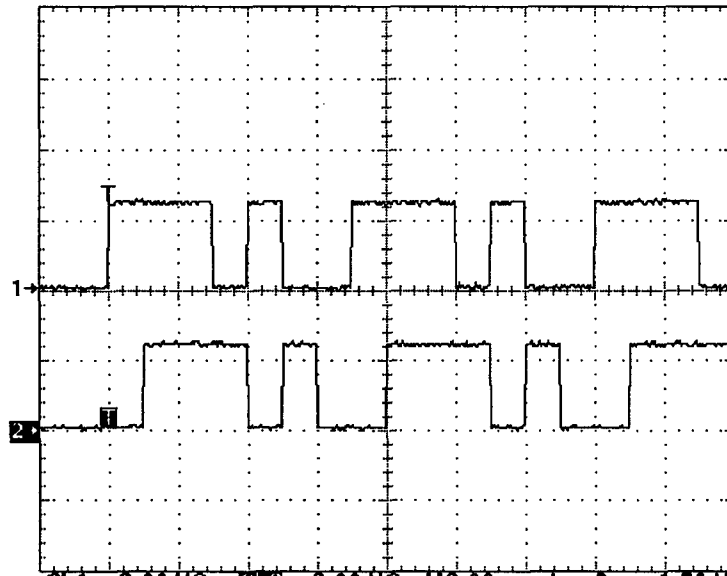
Figure 4-10. CODE0 and MODCOD.

4.3 Code Combiner Outputs

The code combiner is capable of producing the following output codes: XOR01, XOR012, MIXCODE, EARLY, PUNCT, and LATE. The following tests are performed using the linear feedback shift register configuration of Figure 4-3 for Coder0 and the configuration of Figure 4-11 for Coder1 and Coder 2. It is easily verified that the 3-stage configuration of Figure 4-11 provides a maximal length sequence.
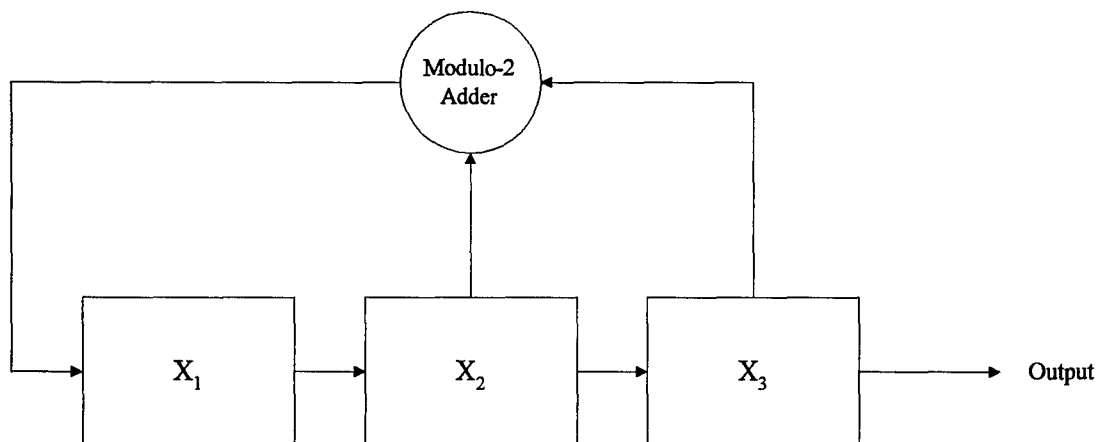
Figure 4-11. Test Linear Feedback Shift Register (Coder1 and Coder2).

The linear feedback shift register of Figure 4-11 may be represented by the polynomial

$X^3 + X^2 + 1$; its binary representation is (1 1 0 1) with an octal representation loaded into

the MASK register of [1 5]. The register contents are shown in Table 4-2 where all

registers are initially filled with a '1'. This linear feedback shift register along with the shift

register of section 4.2 are used in this section for testing combined code outputs.

Table 4-2. Linear Feedback Shift Register State Values.

	Register Contents		
State	X_1	X_2	X_3
0	1	1	1
1	0	1	1
2	0	0	1
3	1	0	0
4	0	1	0
5	1	0	1
6	1	1	0
7	1	1	1

4.3.1 XOR01 Output Waveform Demonstration

The XOR01 output is a waveform resulting from the modulo-2 addition of

CODE0 and CODE1 output signals. The resultant waveform is delayed by one clock

cycle before appearing on the XOR01 output. By loading the MUX register with the 5-bit

data [0 0 0 1 0], the pseudorandom code (CODE1) is taken from the third stage. From

Table 4-2, the expected output is shown as (1 1 1 0 0 1 0 . . .) and from section 4.2,

CODE0 is (1 1 1 0 1 0 0). The modulo-2 addition of these codes is:

CODE0: 1 1 1 0 1 0 0
CODE1: 1 1 1 0 0 1 0
XOR01: 0 0 0 0 1 1 0

Figure 4-12 shows CODE0 as Reference 1 (R1), CODE1 as Reference 2 (R2), and

XOR01 on Channel 1 (1). Since the XOR01 output is delayed one clock cycle before

appearing on the output, both CODE0 and CODE1 are shown delayed by one clock cycle.
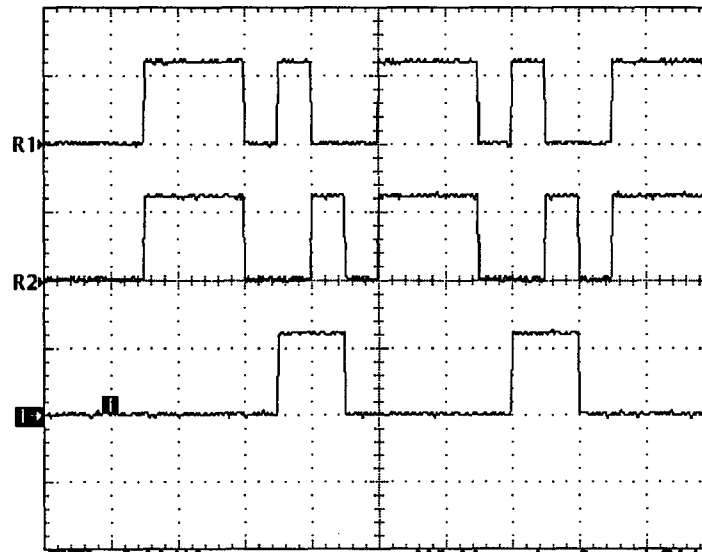
Figure 4-12. CODE0, CODE1, and XOR01.

4-11

Since CODE0 and CODE1 are maximal length sequences, the modulo-2 addition of them is a Gold code of the same length but nonmaximal. The seven bit sequence could also be generated using a single linear feedback shift register with a generator polynomial of $X^6 + X^5 + X^4 + X^3 + X^2 + X + 1$, the modulo-2 multiplication of the two generator polynomials. Since the 6 stage linear feedback shift register is nonmaximal, the initial conditions produce outputs of varied length. Using the first six bits of the Gold code generated (0 0 0 0 1 1) as the initial conditions in the 6 stage register, the pseudorandom code is identical to the Gold code. Figure 4-13 shows the Gold code, and the nonmaximal pseudorandom code generated using Coder2.
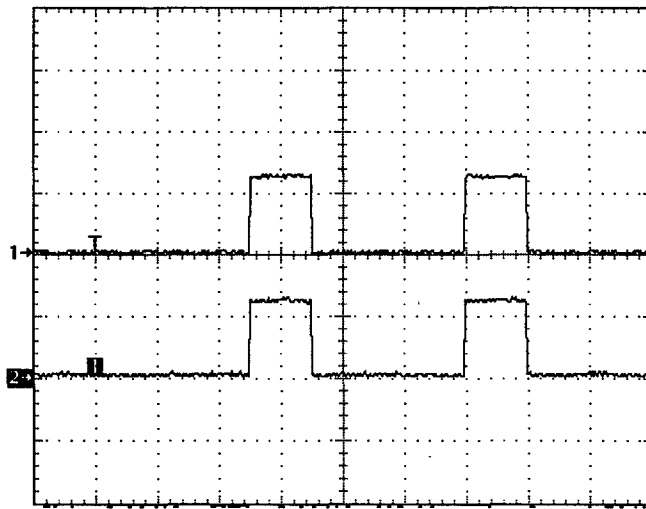
Figure 4-13. Gold Code and Nonmaximal Pseudorandom Code.

By changing the initial conditions in Coder1 (a phase shift), a different Gold code is produced. With an initial condition of $X_1 = 1$, $X_2 = 0$, and $X_3 = 0$, the expected Gold code is

CODE0: 1 1 1 0 1 0 0
CODE1: 0 0 1 0 1 1 1
Gold code: 1 1 0 0 0 1 1

4-12

The new Gold code along with CODE0 and CODE1 are shown in Figure 4-14.
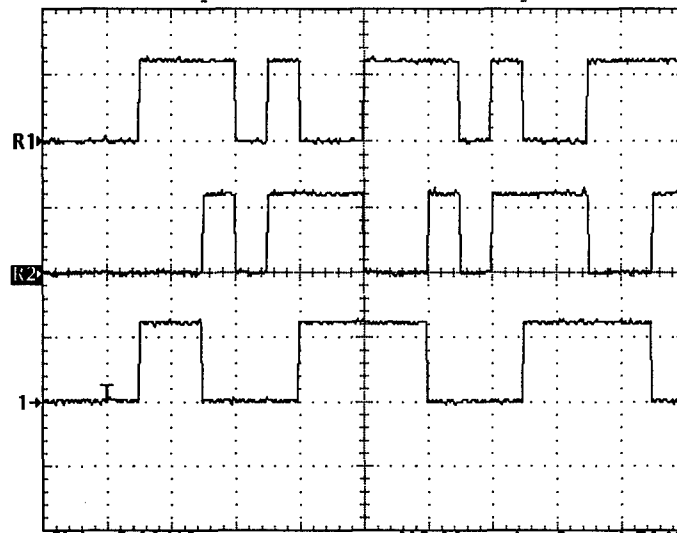
Figure 4-14. CODE0, CODE1, and Gold Code.

4.3.2 XOR012 Output Waveform Demonstration

The XOR012 output is the result of the modulo-2 addition of CODE0, CODE1,

and CODE2. With Coder0 and Coder1 unchanged from the previous demonstration,

Coder2 is configured the same as Coder1 except the initial conditions are (1 1 1). The

expected code is given as:

        CODE0: 1 1 1 0 1 0 0
        CODE1: 0 0 1 0 1 1 1
        CODE2: 1 1 1 0 0 1 0
        XOR012: 0 0 1 0 0 0 1

The generator outputs are shown in Figure 4-15 with CODE0 through CODE2 as

Reference 1 through 3, respectively, and XOR012 on Channel 1.
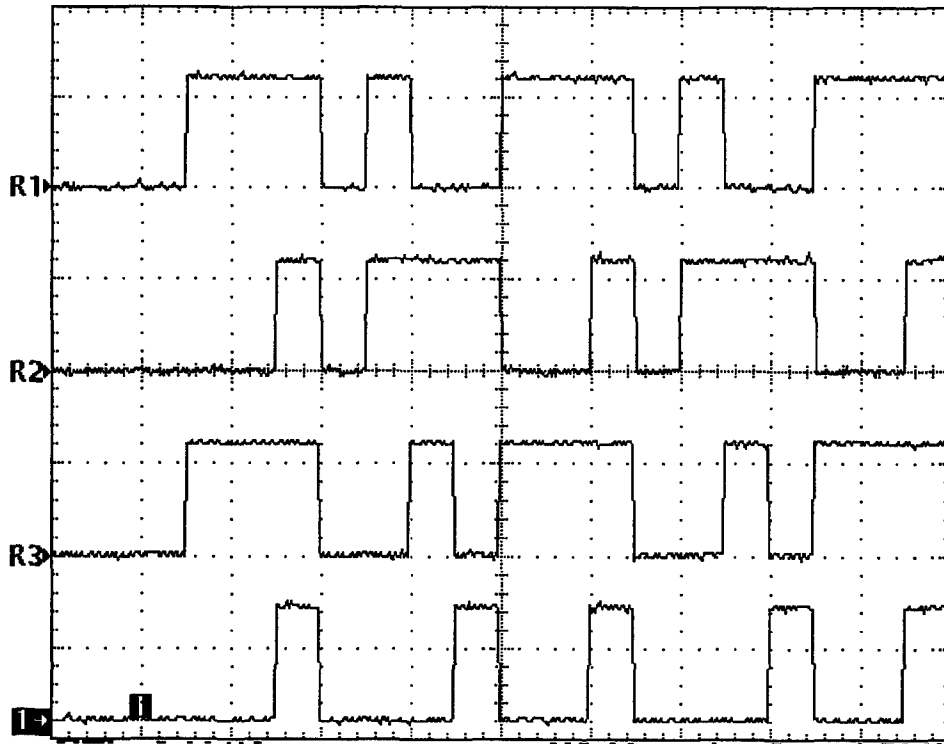
Figure 4-15. CODE0, CODE1, CODE2, and XOR012.

4.3.3 MIXCOD Output Waveform Demonstration

The $MODCOD_0$ - $MODCOD_2$ signals address the 8-bit code programmed into the

code combiner lookup register. With the $STIM_0$ - $STIM_2$ inputs held low, the $MODCOD_0$

- $MODCOD_2$ signals are a one cycle delay of CODE0 - CODE2 respectively. The

MODCOD signals are:

$$MODCOD_0: 1\ 1\ 1\ 0\ 1\ 0\ 0$$
$$MODCOD_1: 0\ 0\ 1\ 0\ 1\ 1\ 1$$
$$\underline{MODCOD_2: 1\ 1\ 1\ 0\ 0\ 1\ 0}$$

Look-up Register Address Bit: 5 5 7 0 3 6 2

The code combiner lookup register is loaded as shown in Table 4-3.

Table 4-3. Example Look-up Table.

Address	Output Bit
0	0
1	1
2	0
3	1
4	1
5	0
6	1
7	1

Using $MODCOD_0$ - $MODCOD_2$ to address Table 4-3, the predicted MIXCOD is

(0 0 1 0 1 1 0). Figure 4-16 shows the MIXCOD output waveform in addition to

$MODCOD_0$ - $MODCOD_2$ as Reference waveforms.
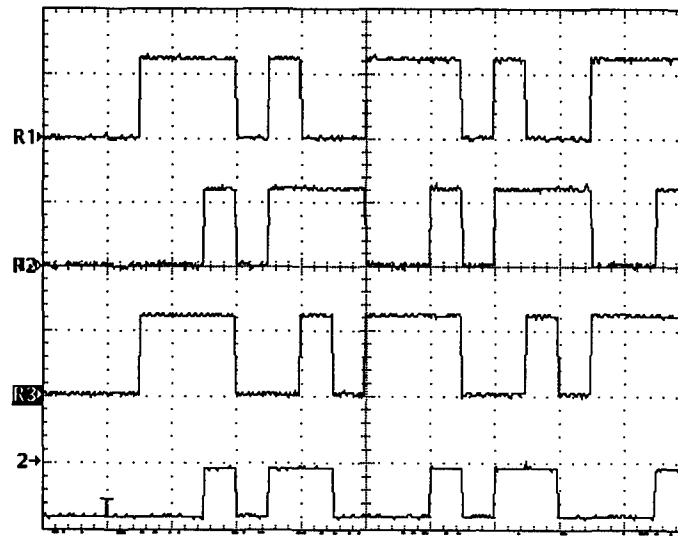
Figure 4-16. $MODCOD_0$ - $MODCOD_2$ and MIXCOD.

4.3.4 PUNCT, EARLY, and LATE Output Waveform Demonstration

The PUNCT output is an exact replica of the MIXCOD delayed by one clock

cycle; as shown in Figure 4-17. The EARLY output is an exact replica of the PUNCT

output advanced by half a clock cycle as shown Figure 4-18. The LATE output is an

exact replica of the PUNCT output delayed by half a clock cycle and is shown in

Figure 4-19.
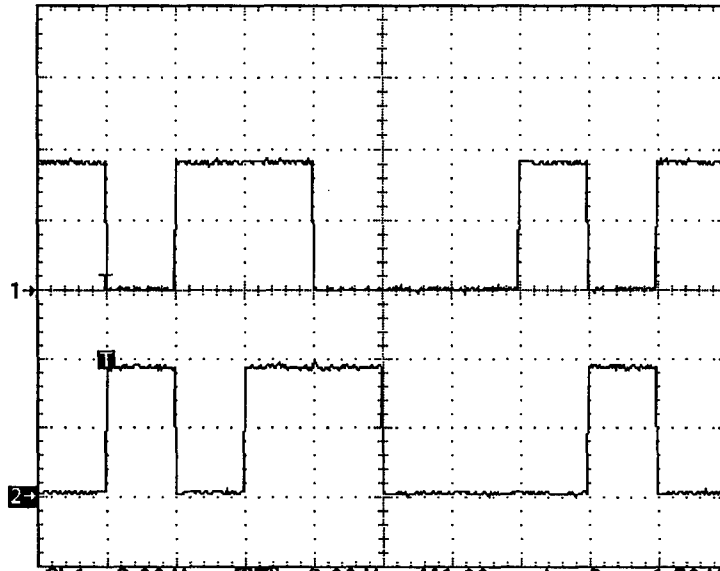
Figure 4-17. MIXCOD and PUNCT Outputs.
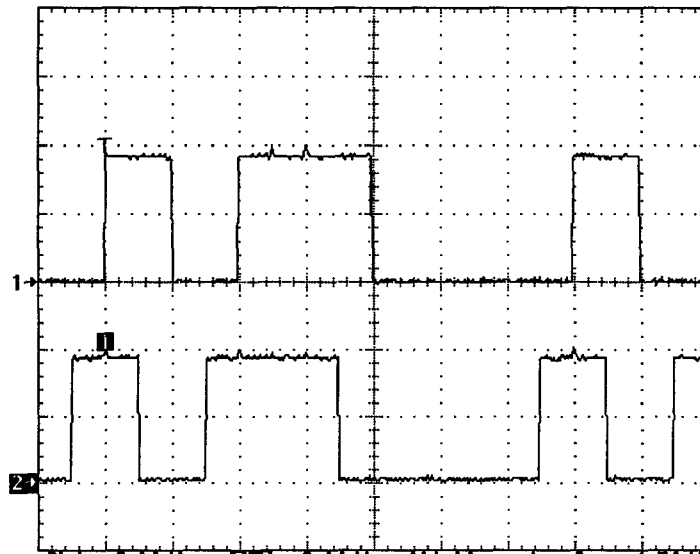
Figure 4-18. PUNCT and EARLY Outputs.
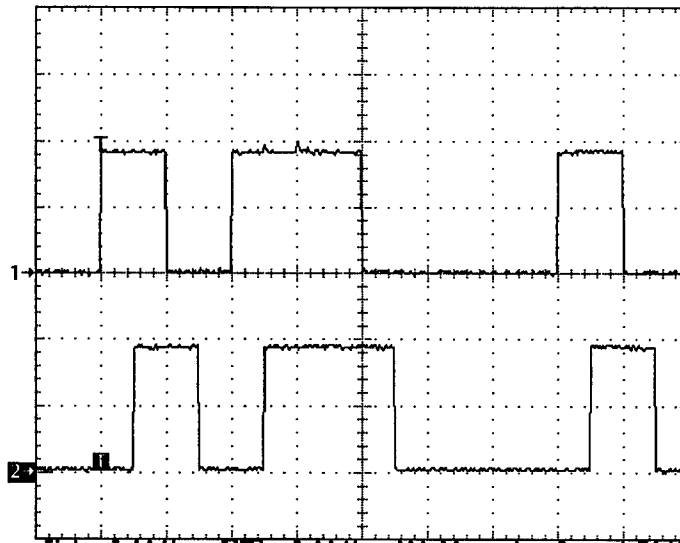
Figure 4-19. PUNCT and LATE Outputs.

4.4 Special Codes

The code generator can produce truncated pseudorandom codes. Truncating the pseudorandom code is useful when combining two pseudorandom codes where one is not an integer multiple of the other. As an example, if a 10 stage maximal length sequence generator (1023 bits) is to be combined with a seven stage maximal length sequence generator (127 bits), the 127-bit sequence can be truncated to 93 bits and produce a pseudorandom code of length 1023 bits. The 8-bit CTL Register is used to define the reloading of the coder and counter; Table 3-4 describes the bit functions. The coder can reload the initial values when an EPOCH or COUNT pulse is encountered. With the coder set as described in Section 4.2 and the EPOCH set at [4], the CTL Register is loaded with [0 0 0 0 0 1 0 0]. The result is shown in Figure 4-20; Coder0 is truncated at five bits when the EPOCH pulse occurs.

4-17
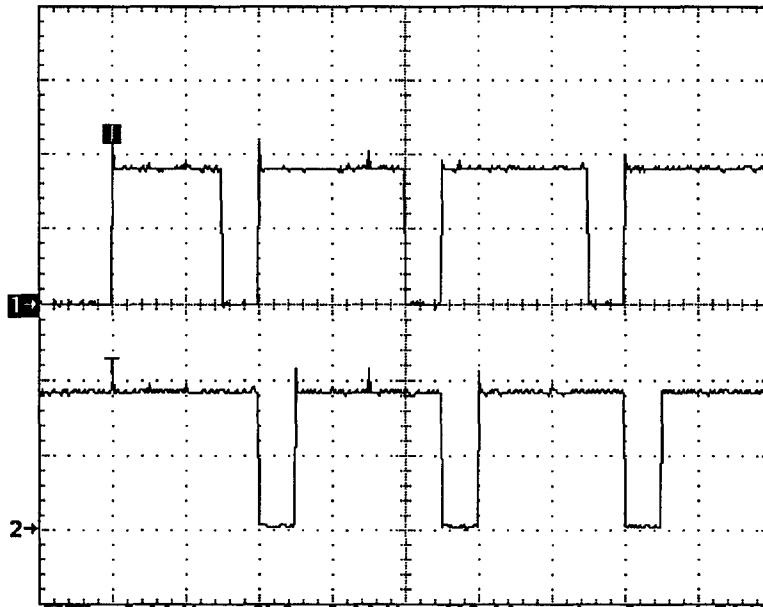
Figure 4-20. Truncated CODE0 and EPOCH Pulse.

Taking this a step further, an individual coder can reset another coder.

Figure 4-21 shows Coder0 reset by an EPOCH pulse from Coder1.
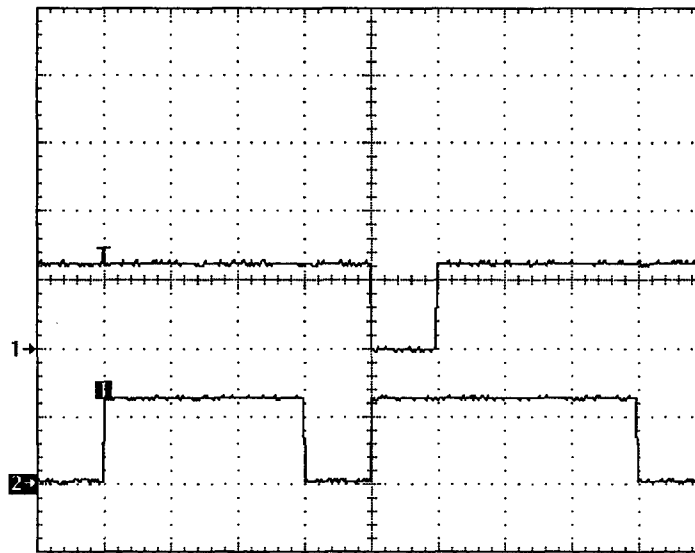
Figure 4-21. Truncated CODE0 and Coder1 EPOCH Pulse.

Another type of special pseudorandom code involves providing a pseudorandom clock to the coder. By connecting the output of a coder to the clock of another coder, the output would be another pseudorandom code. With Coder1 set as described in Section 4.3, the pseudorandom code output is (1 1 1 0 0 1 0). The coder shifts the register bits to the next stage on a low-to-high transition of the clock. With the output of Coder1 connected to the clock of Coder0, the length of each output bit is varied. The following shows a comparison:

Coder1: <u>1 1 1 0 0</u> <u>1 0</u> <u>1 1 1 0 0</u> <u>1 0</u> <u>1 1 1 0 0</u> <u>1 0</u> <u>1 1 1 0 0</u> <u>1 0</u> <u>1 1 1 0 0</u> ...
Coder0: 1   1 1   0 1   0 0   1 1

Figure 4-22 shows CODE1 and CODE0 outputs where CODE1 is used as the Coder0 clock input.
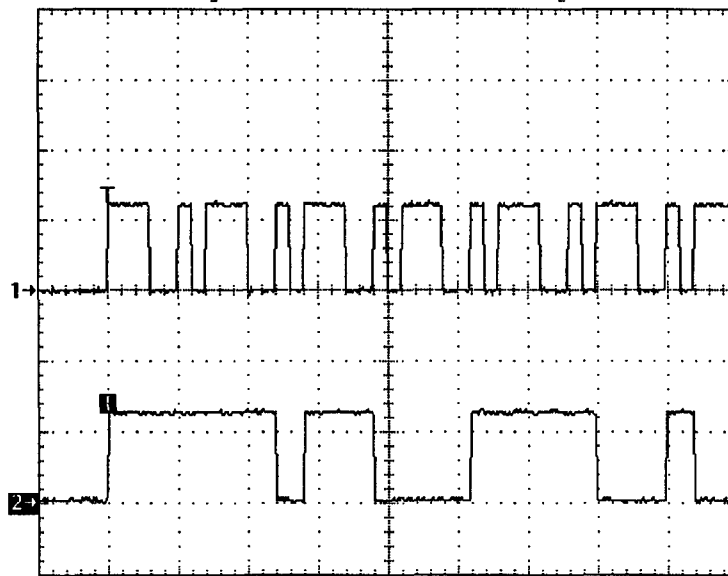
Figure 4-22. CODE1 and CODE0 with CODE1 Clock Input.

4.5 Code Generator Evaluation Summary

The test results presented in the previous section clearly demonstrates the flexibility and capabilities of the code generator design. The individual coders are capable of producing independent pseudorandom codes of various length. The code combiner provides the added capability to produce Gold codes and non-linear table look-up codes. The code generator is capable of efficiently creating codes using pseudorandom input clocks.

5. Conclusions and Recommendations

5.1 Conclusions

This research project investigated the design, construction and evaluation of a pseudorandom code generator for communication and navigation system applications. A United States Patent application is being submitted for the code generator and interface design. The heart of the system is the Standford Telecom, STEL-1032, pseudorandom number (PRN) coder. The PRN coder is comprised of three independent 32-bit linear feedback shift registers and a code combiner. The feedback tap connections for each shift register are programmably controlled by the user. Any combination of feedback tap connections is possible, giving each shift register the capability to produce all possible codes with length up to $2^{32} - 1$. Appendix A includes a table of generator polynomials used for determining feedback tap connections to produce maximal length sequences. The starting phases of the PN codes are user selectable via loading of initial contents into the shift registers. The code generator is capable of detecting a specific sequence in the PN code or a distinct length of the PN code, both of which are user defined.

The code combiner provides the added capability of producing Gold codes and Jet Propulsion Laboratory (JPL) ranging codes by EXORing the outputs of the first two linear feedback shift registers or all three outputs. A user programmable look-up table is employed by the code combiner to provide a non-linear code generation capability. This mixed code is also available in an early, punctual, and late format. The early code is

advanced by one-half a clock cycle compared to the punctual code and the late code is delayed by one-half of a clock cycle.

The code generator is able to produce a pseudorandom code modulated by operator defined data inputs. Since independent external clocks are used for each linear feedback shift register, the output code sequence of one coder may be used as the clock input to another coder. This allows codes with pseudorandom bit intervals to be produced.

5.2 Recommendations for Future Study

Manual programming of the code generator consists of making an address selection and setting input data via front panel switches. The system interface design allows for the future development of a computer interface to the code generator. The design of the interface to the STEL-1032 consists primarily of TTL logic gates making the interface design a prime candidate for Application Specific Integrated Circuit (ASIC) technology.

Given the ability to easily generate pseudorandom codes for use in communication and navigation system applications, the code generator's flexibility provides an excellent opportunity for code development and exploration. Another potential area of study is the use of the code generator for data encryption applications. The Avionics Directorate of Wright Laboratory has expressed great interest in using the code generator to assist in the development of future spread spectrum systems.

Appendix A : Octal Representation of Primitive Generator Polynomials.

The following table lists the octal representation of primitive polynomials that

produce maximal length sequences. The table is nowhere near the complete listing of

generator polynomials that produce maximal length sequences; Table 2-2 includes the

number of polynomials that generates maximal length sequences and the sequence length

for a given number of stages, *n*. The following example demonstrates the proper use of

the table. Figure A-1, represents a three stage linear feedback shift register that produces

a maximal length sequence.
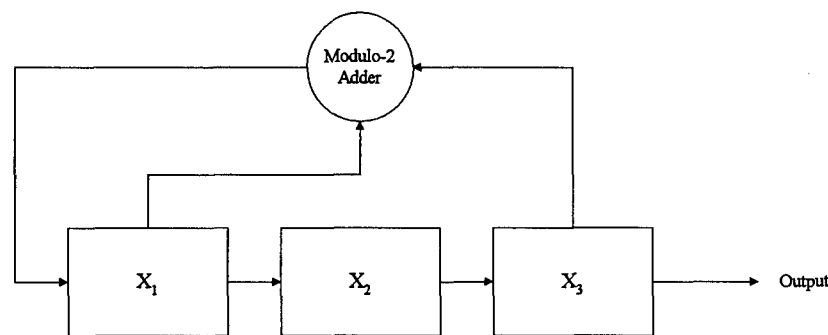
Figure A-1: 3 Stage Linear Feedback Shift Register.

The polynomial that represents the linear feedback shift register shown in Figure A-1 is

given by $X^3 + 0X^2 + X + 1$. The binary representation of the polynomial is 1011 where

the Most Significant Bit (MSB) is on the left and the Least Significant Bit (LSB) is on the

right. By combining the binary digits in groups of three starting at the LSB, the

polynomial is given by [1, 011] and converting this to octal number the polynomial

representation is [1 3]. This octal representation is include in Table A-1 for degree 3.[5]

Table A-1: Octal Representation of Primitive Generator Polynomials.

Degree	Octal Representation (d_0 on right to d_n on left)
2	[7]
3	[13], [15]
4	[23], [37]
5	[45], [75], [67]
6	[103], [147], [155]
7	[211], [217], [235], [367], [277], [325], [203], [313], [345]
8	[435], [551], [747], [453], [545], [537], [703], [543]
9	[1021], [1131], [1461], [1423], [1055], [1167], [1541], [1333], [1605], [1743], [1617], [1553]
10	[2011], [2415], [3771], [2157], [3515], [2773], [2033], [2443], [2461], [3023], [3543], [2745]
11	[4005], [4445], [4215], [4055], [6015], [7413], [4143], [4563], [4053], [5023], [5623], [4577]
12	[10123], [15647], [16533], [16047], [11015], [14127], [17673], [13565], [15341], [15053]
13	[20033], [23261], [24623], [23517], [30741], [21643], [30171], [21277], [27777], [35051]
14	[42103], [43333], [51761], [40503], [77141], [62677], [44103], [45145], [76303], [64457]
15	[100003], [102043], [110013], [102067], [104307], [100317], [177775], [103451], [110075], [102061]
16	[210013], [234313], [233303], [307107], [307527], [306357], [201735], [272201], [242413], [270155]

Degree	Octal Representation (d_0 on right to d_n on left)
17	[400011], [400017], [400431], [525251], [410117], [400731], [411335], [444257], [600013], [403555]
18	[1000201], [1000247], [1002241], [1002441], [1100045], [1000407], [1003011], [1020121]
19	[2000047], [2000641], [2001441], [2000107], [2000077], [2000157], [2000175], [2000257]
20	[4000011], [4001051], [4004515], [6000031], [4442235]
21	[10000005], [10040205], [10020045], [10040315], [10000635], [10103075], [10050335], [10002135]
22	[20000003], [20001043], [22222223], [25200127], [20401207], [20430607], [20070217]
23	[40000041], [40404041], [40000063], [40010061], [50000241], [40220151], [40006341], [40405463]
24	[100000207], [125245661], [113763063]
25	[200000011], [200000017], [204000051], [200010031], [200402017], [252001251]
26	[400000107], [430216473], [402365755], [426225667], [510664323], [473167545]
27	[1000000047], [1001007071], [1020024171], [1102210617], [1250025757], [1257242631]
28	[2000000011], [2104210431], [2000025051], [2020006031], [2002502115], [2001601071]
29	[4000000005], [4004004005], [4000010205], [4010000045], [4400000045], [4002200115]
30	[10040000007], [10104264207], [10115131333], [11362212703], [10343244533]

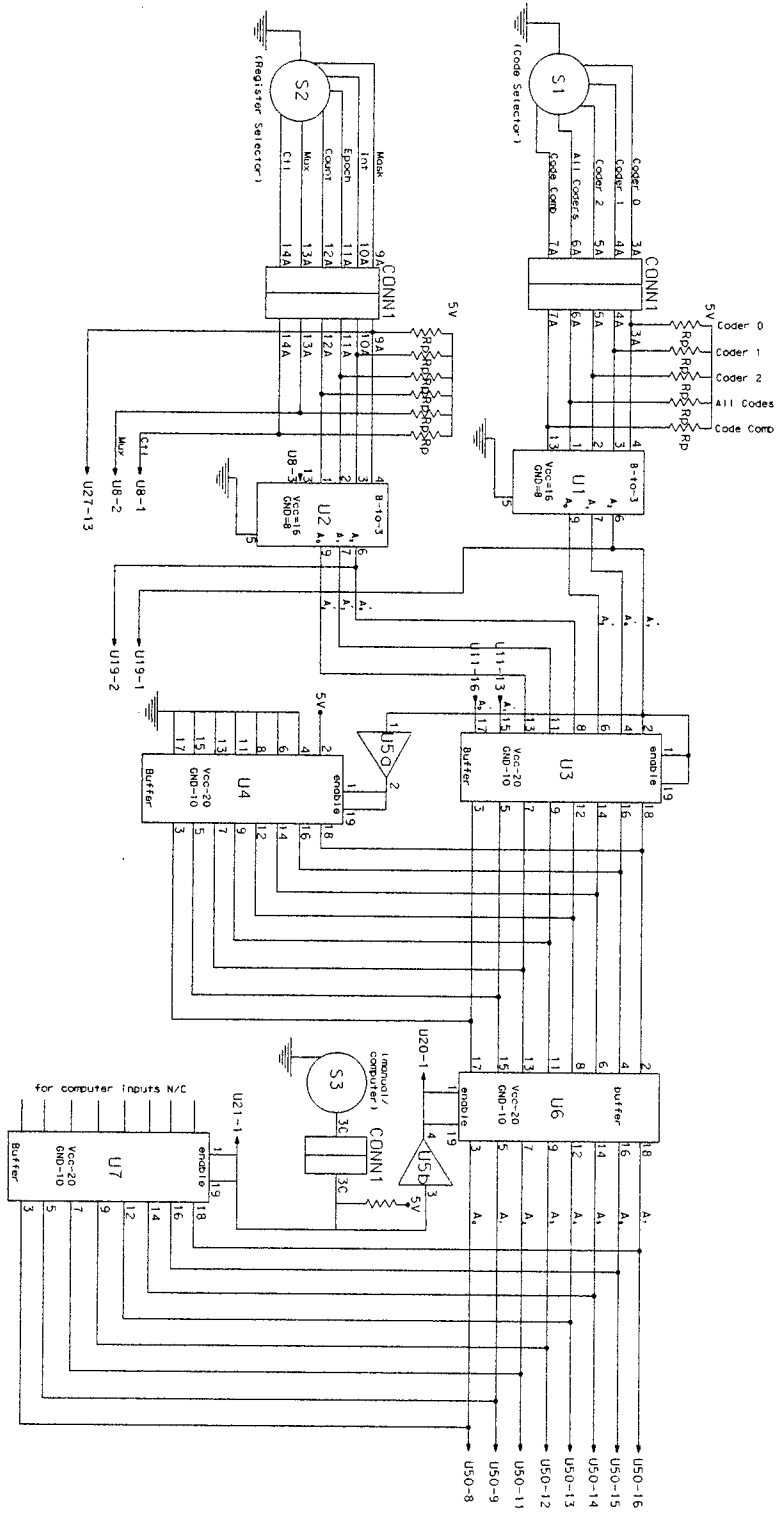| Degree	Octal Representation (d_0 on right to d_n on left)
31	[20000000011], [20000000017], [20000020411], [21042104211], [20010010017], [20005000251]
32	[40020000007], [40460216667], [40035532523], [42003247143], [41760427607]

Appendix B : Parts Listing and Circuit Diagram

Table B-1 is a complete listing of parts used in the construction of the STEL-1032

interface. The interface is constructed on a MUPAC wire wrap board with a power plane

on top and a ground plane on the bottom. A capacitor is placed in parallel between the

power and ground at each integrated circuit to filter the 5 VDC.

Table B-1: Parts Listing.

Nomenclature	Description	Part Number
S1	5 position switch	
S2	6 position switch	
S3, S6	2 position toggle switch	
S4, S5, S7	momentary push button switch	
C1, C8, C9, C10	capacitor	.33 μF
C2, C4, C7	capacitor	1 μF
C3	capacitor	.01 μF
C5	capacitor	4.2 μF
C6	capacitor	2.2 μF
Rp	pull-up resistor	2.2 KΩ
R1, R8	resistor	2.2 KΩ
R2	resistor	360 KΩ
R3	resistor	750 KΩ
R4 - R7	resistor	150 KΩ
R9	resistor	1.0 KΩ
U1,U2	8-to-3 encoder	74148
U3-4, U6-7, U11-16, U20-21, U42-45	octal tri-state buffer	74LS244
U5, U17, U27, U30-34	hex inverter	74LS04
U8, U18, U22, U26	quad AND gate	74LS08
U9	timer	555
U10	binary counter	74LS163
U19, U25, U46	quad OR gate	74LS32
U23-24	dual one-shot	74LS221
U35-41	quad NOR 50 Ω line driver	74128
U50	PRN Coder	STEL-1032

Circuit Diagram

Circuit Diagram (cont.)
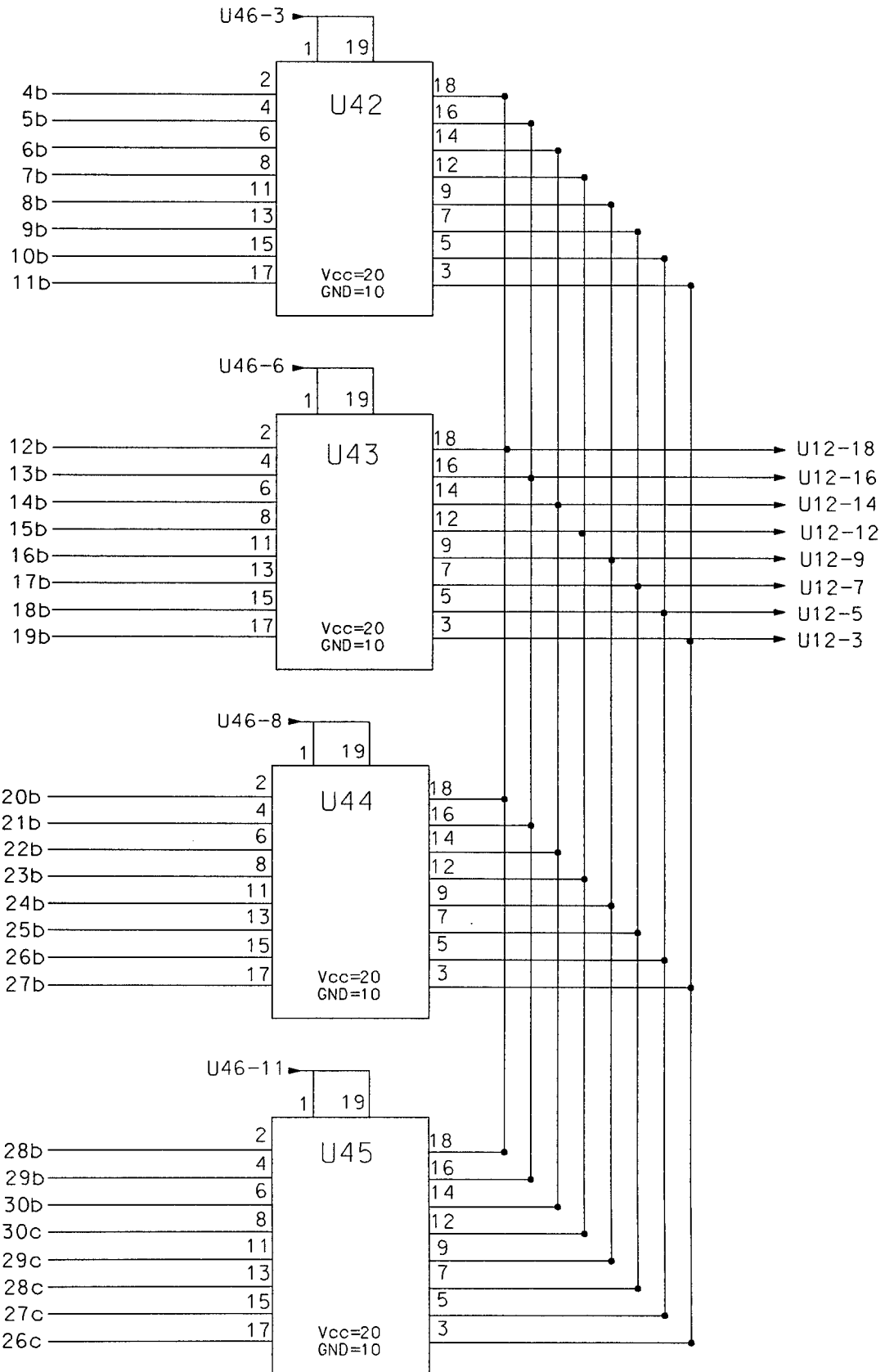
Circuit Diagram (cont.)

Circuit Diagram (cont.)

U46-3

	1	19	
4b	2	U42	18
5b	4		16
6b	6		14
7b	8		12
8b	11		9
9b	13		7
10b	15		5
11b	17	Vcc=20 GND=10	3

U46-6

	1	19		
12b	2	U43	18	U12-18
13b	4		16	U12-16
14b	6		14	U12-14
15b	8		12	U12-12
16b	11		9	U12-9
17b	13		7	U12-7
18b	15		5	U12-5
19b	17	Vcc=20 GND=10	3	U12-3

U46-8

	1	19	
20b	2	U44	18
21b	4		16
22b	6		14
23b	8		12
24b	11		9
25b	13		7
26b	15		5
27b	17	Vcc=20 GND=10	3

U46-11

	1	19	
28b	2	U45	18
29b	4		16
30b	6		14
30c	8		12
29c	11		9
28c	13		7
27c	15		5
26c	17	Vcc=20 GND=10	3

B-5

Circuit Diagram (cont.)
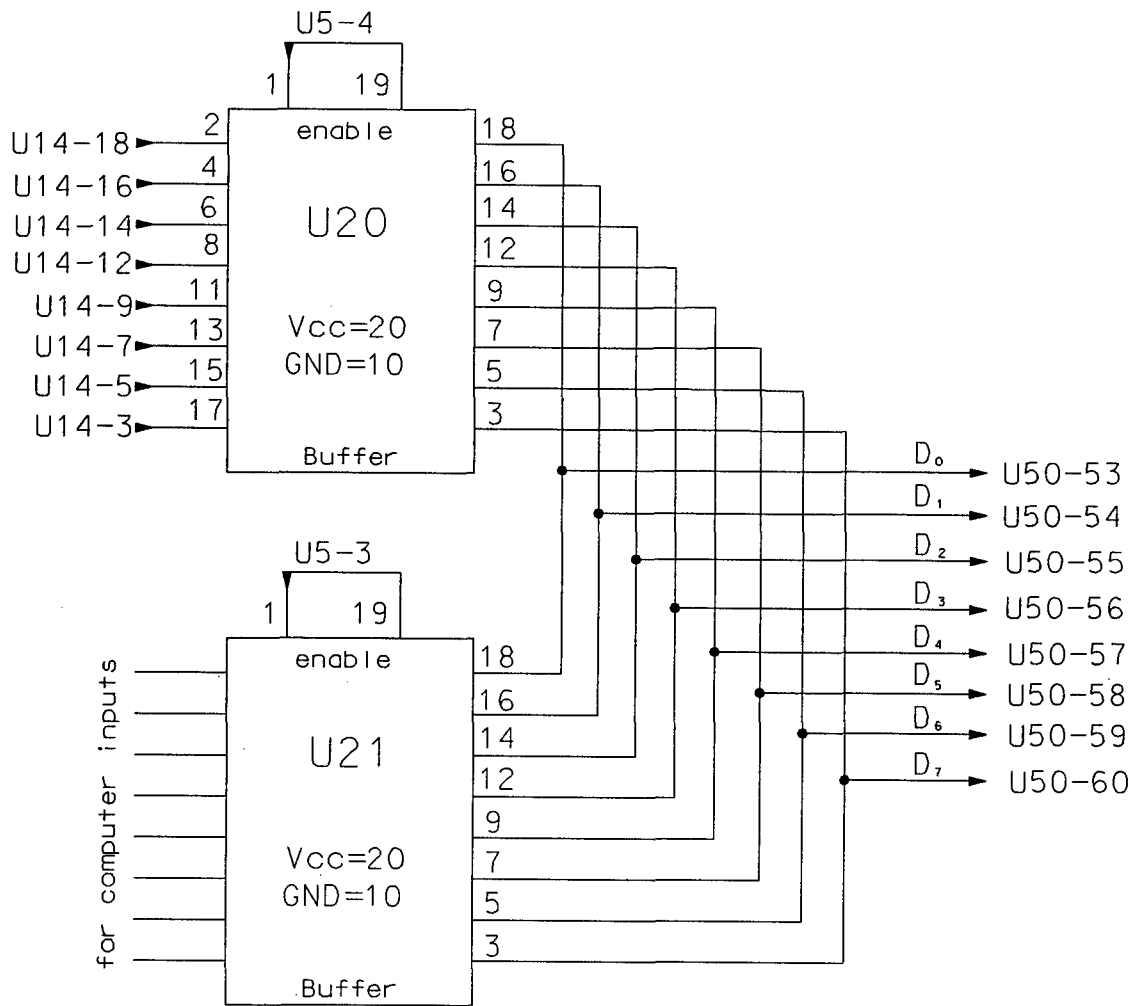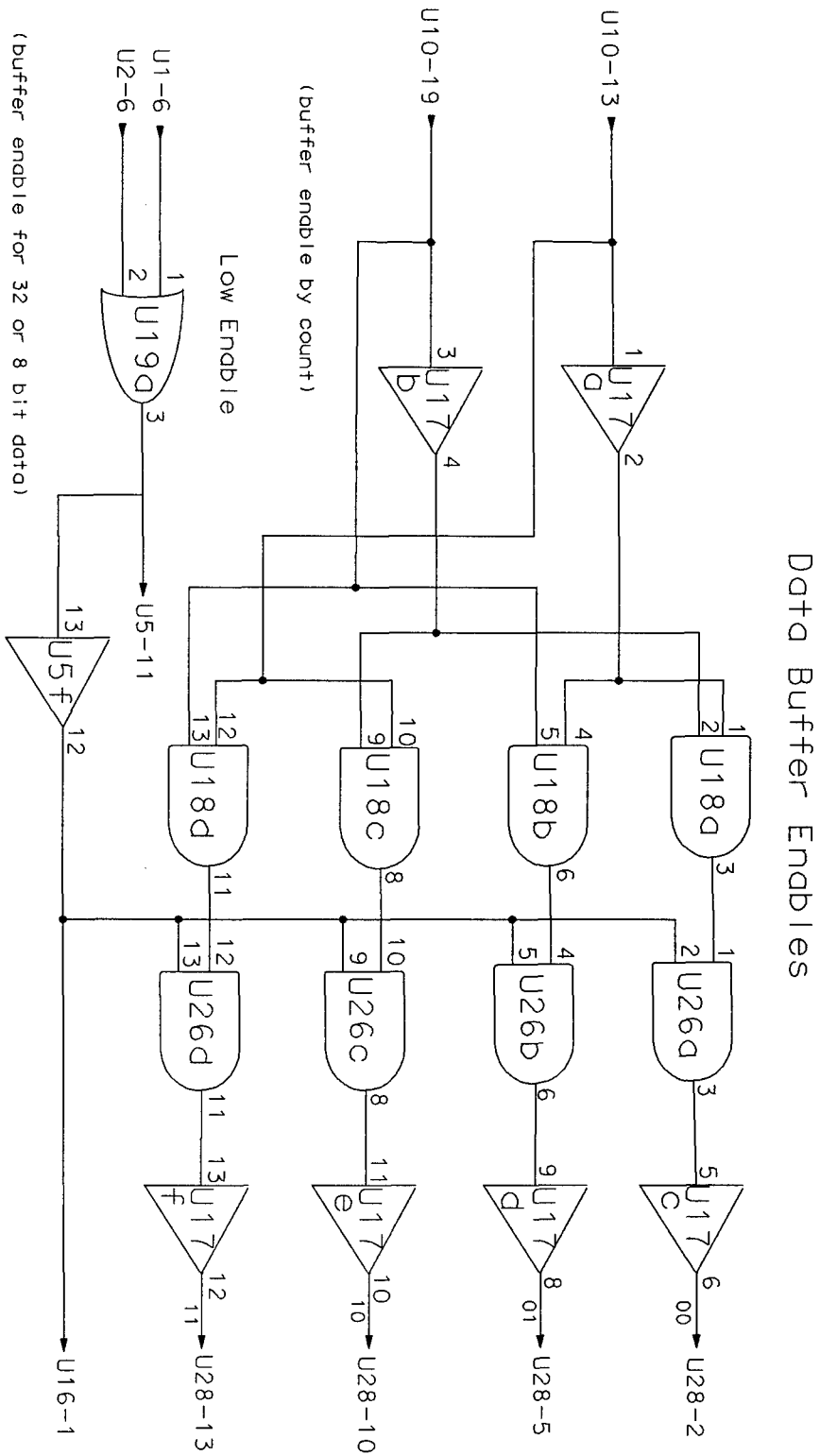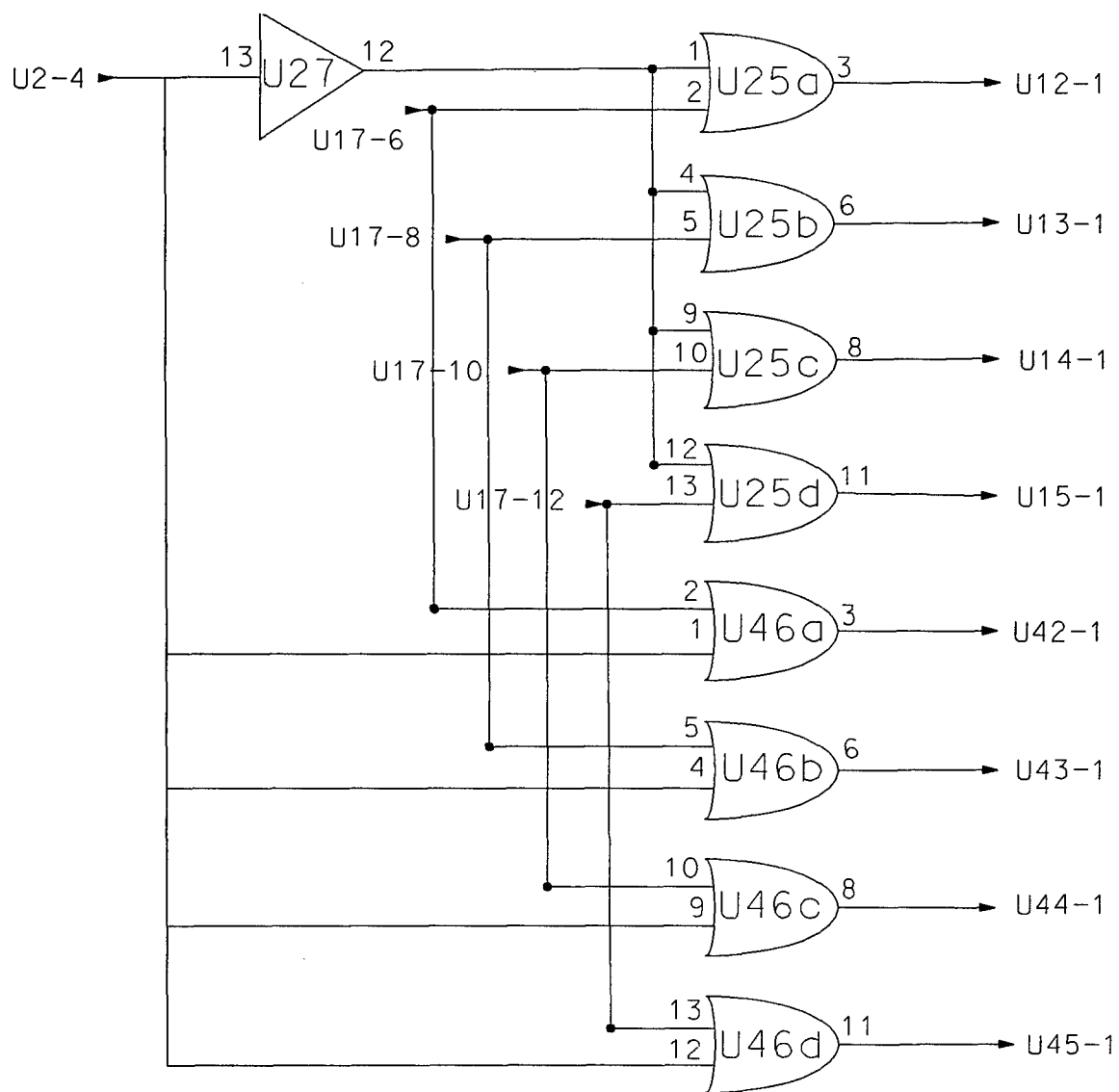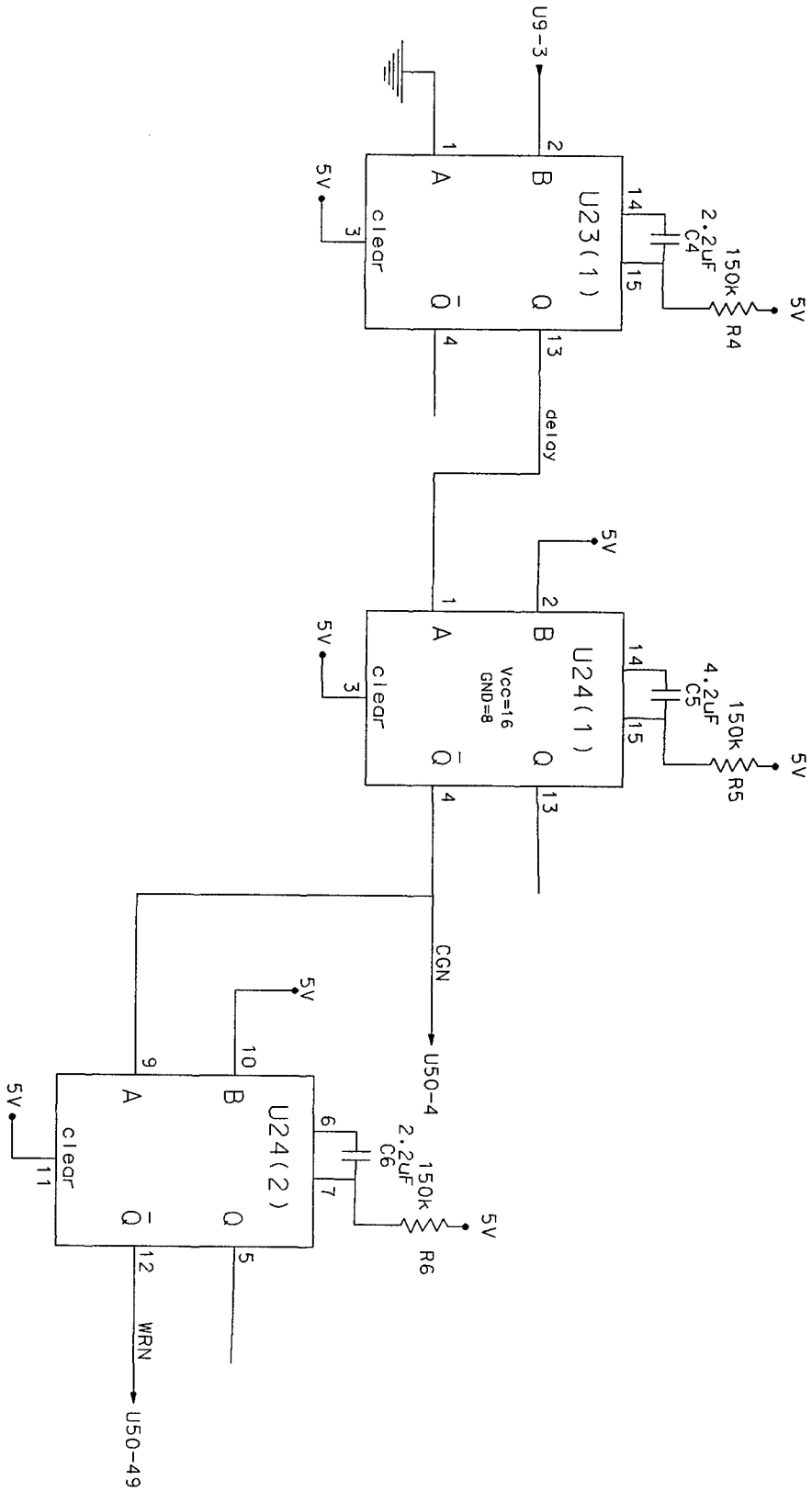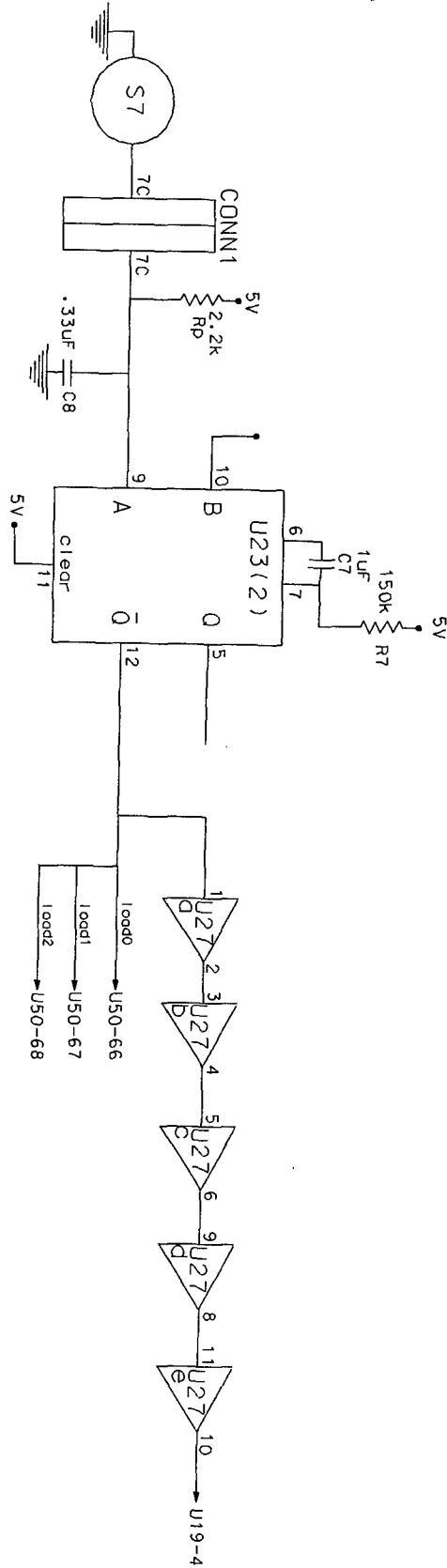

Data Buffer Enables

Circuit Diagram (cont.)

Circuit Diagram (cont.)

Circuit Diagram (cont.)

Circuit Diagram (cont.)

U50-17	XORO12	1 U30 a	2	2 3 U35a	1	1a	CONN2	1a	BNC XORO12
U50-20	XORO1	3 U30 b	4	5 6 U35b	4	3a		3a	BNC XORO1
U50-21	MIXCOD	5 U30 c	6	8 9 U35c	10	5a		5a	BNC MIXCOD
U50-22	LATE	9 U30 d	8	11 12 U35d	13	7a		7a	BNC LATE
U50-23	EARLY	11 U30 e	10	2 3 U36a	1	13a		13a	BNC EARLY
U50-24	PUNCT	13 U30 f	12	5 6 U36b	4	21a		21a	BNC PUNCT
U50-31	CODEO	1 U31 a	2	8 9 U36c	10	23a		23a	BNC CODEO
U50-28	EPOCHO	3 U31 b	4	11 12 U36d	13	25a		25a	BNC EPOCHO

CONN2

U50-27	COUNT0	5 U31 c 6	2 3 U37a 1	27a	27a	BNC COUNT0
U50-30	MODCOD0	9 U31 d 8	5 6 U37b 4	29a	29a	BNC MODCOD0
U50-46	LOSYNC0	11 U31 e 10	8 9 U37c 10	2b	2b	BNC LDSYNC0
U50-29	REF15-0	13 U31 f 12	11 12 U37d 13	4b	4b	BNC REF15-0
U50-7	STSYNC	1 U32 a 2	2 3 U38a 1	6b	6b	BNC STSYNC
U50-38	CODE1	3 U32 b 4	5 6 U38b 4	8b	8b	BNC CODE1
U50-33	EPOCH1	5 U32 c 6	8 9 U38c 10	10b	10b	BNC EPOCH1
U50-32	COUNT1	9 U32 d 8	11 12 U38d 13	12b	12b	BNC COUNT1
U50-37	MODCOD1	11 U32 e 10	2 3 U39a 1	14b	14b	BNC MODCOD1

| U50-47 | LDSYNC1 | 13 U32 f 12 | 5 6 U39b 4 | 16b | CONN2 | 16b | BNC LDSYNC1 |

Bibliography

1. Dixon, Robert C. *Spread Spectrum Systems.* New York : John Wiley & Sons, 1984.

2. Golomb, Solomon, W. *Shift Register Sequences.* Laguna Hills, California: Aegean Park Press, 1982.

3. Michelson, Arnold M. and Allen H. Levesque. *Error-Control Techniques for Digital Communication.* New York: John Wiley & Sons, 1985.

4. Peterson, Roger L., Rodger E. Ziemer and David E. Borth. *Introduction to Spread Spectrum Communications.* New Jersey: Prentice Hall, 1995.

5. Peterson, William W. and E. J. Weldon Jr. *Error-Correcting Codes.* Massachusetts: The MIT Press, 1972.

6. Sklar, Bernard. *Digital Communications: Fundamentals and Applications.* New Jersey: Prentice Hall, 1988.

7. Soclof, Sidney. *Design and Applications of Analog Integrated Circuits.* New Jersey: Prentice Hall, 1991.

8. Spliker, James J. *Global Positioning System: Theory and Applications.* Washington, D.C.: American Institute of Aeronautics and Astronautics, 1988

9. Zavrel, Robert J. *The Spread Spectrum Handbook.* Santa Clara, California: Stanford Telecommunications, Inc., 1990

Vita

Captain John F. Brendle Jr. was born ███████████████████████.

After graduating from St. Francis DeSales High School in Columbus, Ohio, he attended

The Ohio State University. He graduated with a Bachelor of Science in Electrical

Engineering and was commissioned a 2nd Lieutenant in the United States Air Force on

March 16, 1990. His first assignment was Undergraduate Pilot Training (UPT) in

December of 1990. In November 1991, he was assigned to the Aerospace Guidance and

Metrology Center at Newark Air Force Base. In December of 1995, he was selected to

attend the Air Force Institute of Technology and pursue a Master of Science in Electrical

Engineering starting in July 1996. His follow on assignment will be with the National

Reconnaissance Office in Chantilly, Virginia.

Permanent Address: ████████████
████████████

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis	

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Pseudorandom Code Generation for Communication and Navigation System Applications	

6. AUTHOR(S)

John F. Brendle Jr., Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institue of Technology	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GE/ENG/97D-16

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Wright Laboratory Avionics Directorate Wright-Patterson AFB, OH 45433-7333	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release; distribution unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This research project investigated the design, construction and evaluation of a pseudorandom code generator for communication and navigation system applications. These types of codes include spreading codes, Gold codes, Jet Propulsion Laboratory (JPL) ranging codes, syncopated codes, and non-linear codes. Such waveforms are typically used in communication and navigation system applications. The code generator uses the Stanford Telecom STEL-1032 Pseudorandom Number (PRN) coder. A coder interface was designed and constructed for manual data entry to the registers of the PRN coder. The code generator is capable of independently clocking and generating all possible codes with lengths up to 4,294,967,295 bits. The codes can be started with any random phase. The code generator is capable of detecting a specific position in the code and the coders can be truncated and restarted at that point. The three independent coder outputs are combinable, expanding the lengths and versatility of the codes. The generation of a non-linear code is possible using an internally programmable look-up table. Several test were conducted on the code generator to ensure its capability of generating Gold codes, JPL ranging codes, syncopated codes, and non-linear codes. The required documentation is being submitted for a U.S. patent.

14. SUBJECT TERMS			15. NUMBER OF PAGES 100
Pseudorandom Code, Spread Spectrum, Code Generator			16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL