1985

Design of a Gold Code Generator for Use in Code Division Multiple Access Communication System

Mark W. Young
University of Central Florida

DESIGN OF A GOLD CODE GENERATOR FOR USE IN A CODE DIVISION MULTIPLE ACCESS COMMUNICATION SYSTEM

BY

MARK WILLIAM YOUNG
B.S.E., University of Central Florida, 1982

THESIS

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

Fall Term
1985

ABSTRACT

A Gold code sequence generator suitable for use in a code division multiple access spread spectrum communication application is designed. A dual, single return shift register configuration is used to generate Gold code sequences. The code sequences are generated by the mod-2 addition of two linear maximal length pseudo-random noise codes, each of which corresponds to a sixth-order primitive polynomial. A computer model of the design is used to generate all 65 possible members of the Gold code sequence family. A tabulation of all sequences and their initial condition "keys" is provided, along with a designation as to which code sequences are balanced.

The mathematical basis of maximal length sequence generation is developed, using first the matrix characterization of a shift register generator, and then switching to the alternate treatment of a shift register generator as a polynomial division engine. The link between the matrix representation and the polynomial representation via the characteristic equation, the use of the generating function, and the three mathematical properties required of polynomials which are capable of generating maximal length sequences are described. Gold's algorithm for selecting preferred polynomial pairs is

presented, as is his technique for determining the characteristic phase of a maximal length sequence.

The actual Gold code generator is then designed and modeled in software. All Gold code sequences output from the generator are tabulated. The family of sequences is evaluated in terms of its randomness properties. Finally, the results of computer analysis of the auto and cross-correlation characteristics of the family is summarized.

ACKNOWLEDGEMENTS

I would like to thank two people whose help was instrumental in the successful completion of this thesis.

The first is Dr. Madjid Belkerdid, my academic advisor during my graduate and undergraduate years at the University and chairman of my thesis advisory committee. His wise counsel and always-ready willingness to help with the tough problems has helped to make my years of study both extremely productive and, in retrospect, thoroughly enjoyable.

The second person to whom I owe much thanks is my wife, Donna. In addition to doing all the typing of this manuscript, she has provided unfailing emotional support throughout my long years of study. To these two people, I say a heartfelt "thank you."

M.W.Y.

1985

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

A spread spectrum system is one in which a message signal to be transmitted is spread over a frequency band much, much wider than the minimum bandwidth that would otherwise be required to transmit it. In a general sense, spectrum spreading is accomplished by twice modulating a carrier, once with the message signal, and then again with a wideband encoding signal. The wideband encoding signal, for most all spread spectrum systems, is usually some pseudo-randomlike waveform generated from either a linear or non-linear binary code sequence. The type of carrier modulation employed, and the method by which the wideband encoding signal is used to spread the resulting spectrum, defines the type of spread spectrum system.

Usually, the message signal is first converted to a binary waveform via some sampling process. It is then mod-2 added to the pseudo-random noise waveform. The composite wideband encoding-message signal is next used in one of several ways.

In frequency hopping spread spectrum systems, some type of FSK (frequency shift keying) modulation technique is usually employed. The pseudo-random encoding waveform, representing a sequence of zeros and ones of length n, is decoded into one of its 2^n possible states. Each state is made to correspond to one frequency output from a signal generator or frequency synthesizer which can generate any of the 2^n possible frequencies. Each time the encoded sequence changes, the frequency output from the generator "hops" in unison. In essence, a frequency hopping system employs many different carrier frequencies. Which frequency is output varies pseudo-randomly as does the encoding waveform. The signal generator employed distributes the frequencies across the desired spread spectrum band. Therefore the rate at which the code varies determines the "hop" rate of the system, but does not directly set the system's bandwidth.

Time hopping spread spectrum systems usually use some type of pulse coded modulation technique. The pseudo-random encoding signal is used in effect to switch the transmitter "on" and "off."

Unlike frequency hopping systems which use many different carrier frequencies, direct sequence spread spectrum systems usually only employ one or two carrier frequencies. In systems employing BPSK (binary phase shift keying) or QPSK (quadrature phase shift keying) modulation, the pseudo-random waveform is used to modulate the phase of a single carrier. In systems employing MSK (minimum shift keying) modulation, it is used to modulate both carrier phase and carrier frequency, usually two different phases of two different tone frequencies. Also unlike frequency hopping systems, the bandwidth of the spread spectrum signal is directly dependent on the code rate. Faster code rates imply greater bandwidth.

A fourth type of spread spectrum system is the pulsed FM, or chirp system. Unlike other spread spectrum techniques, this one usually does not employ a pseudo-random noise code to generate wideband modulated signals, but more often depends on common linear frequency sweep techniques. This type of spread spectrum scheme is usually used in radar and ranging applications.

In addition, there are various hybrid configurations which combine two or more of the above techniques to some advantage. No matter which technique is used, however, a fundamental tradeoff is always involved, that of exchanging valuable RF spectral bandwidth for some combination of performance enhancements and features. These enhancements and features come about either as a direct result of the great increase of bandwidth, or because of the coded modulating signal format, or both. These features include: improved interference rejection (against natural and receiver noise, and intentional jamming), the ability to hide low power signals in the ever-present power spectral density noise floor and recover them reliably, provision for message screening and encryption, the capability to selectively address remote receivers, and the opportunity to enable multiple access of many different signals to a single channel by the use of code division multiplexing.

Code division multiplexing techniques are especially suited to direct sequence spread spectrum systems in that a transmitter generating a single carrier frequency can be used to transmit signals from many different sources simply by changing the pseudo-random code sequence used to toggle the phase of the output signal. If the code sequences are properly chosen to ensure low cross-correlation, multiple coded signals can occupy the same frequency band at the same time and still be successfully differentiated at their respective receivers.

There are families of pseudo-random noise code sequences, the so-called Gold codes, which do possess very attractive cross-correlation characteristics. This and several other properties make Gold codes nearly ideal for use in code division multiple access spread spectrum communications applications. In this paper, these desirable properties are investigated in some detail, and a Gold code pseudo-random pulse generator is designed and computer modeled to generate one entire family of Gold code sequences.

PROPERTIES OF PSEUDO-RANDOM NOISE CODES

Gold codes are code sequences generated by the mod-2 addition of a pair of linear maximal length pseudo-random noise (PN) code sequences. Selection of a proper pair of maximal length sequences will produce a set of codes which possess characteristics ideally suited for code divison multiple access applications. Since maximal length PN code generators are integral components of a Gold code generator, the analysis of Gold codes is prefaced with PN code theory.

A distinction between sequences and waveforms must first be stated. A waveform is a time domain representation of an analog signal which possesses as characteristics amplitude, phase, and frequency content, among others. A sequence is a digitized, quantized series of ones and zeros. A zero or a one is a symbol representing one of two possible logic states which a waveform may be said to be in within some interval of time. In the parlance of spread spectrum communications, this interval of time is referred to as the chip time, and it is common practice to refer to individual ones and zeros in a code sequence as chips. The chip rate is then the rate at which a code generator outputs sequences of ones and zeros. An alternate and slightly different interpretation of the term "chip" is implied when used in the context of frequency hopping signal generators, where a chip is defined to be one of the output frequencies of the generator. Chip time is then the amount of time occupied by that particular frequency, and chip rate is the frequency hopping rate.

This paper deals exclusively with sequences, series of ones and zeros and the properties they possess, with the implicit understanding that in actual applications waveforms will ultimately be generated from corresponding sequences.

Shift Register Sequences

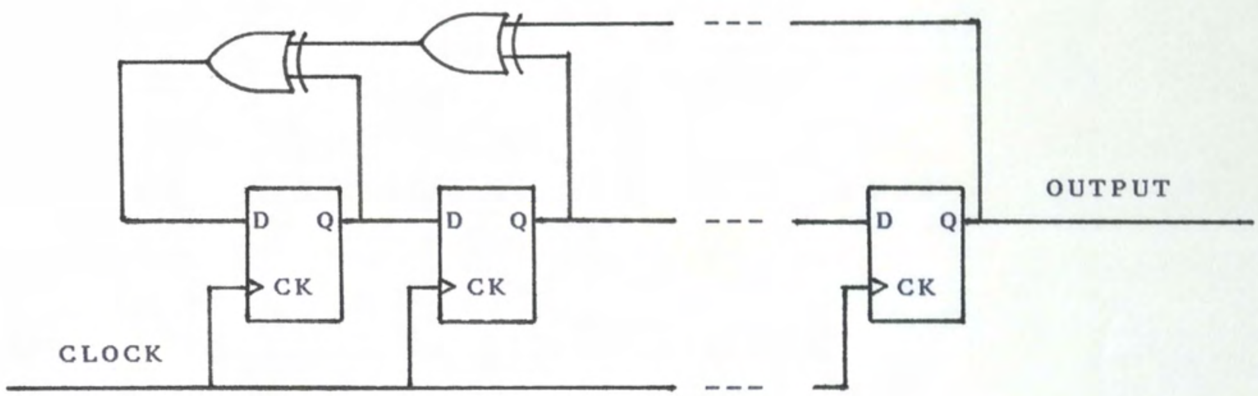
The code sequences to be studied are all generated by means of one or more shift registers with associated feedback logic. This feedback logic is designed to feed back some logical combination of the state of two or more of the shift register's stages to its input stage. For this reason, the sequences so generated are also referred to as shift register sequences. Shift register sequences are cyclic in nature, that is, the sequence repeats itself continuously as long as the shift register stages are clocked.

The sequence length is defined to be the number of chips (i.e., the number of ones and zeros) in each full, repeating cycle of the sequence. The chip rate is then the clock rate of the shift register, and the chip time is the time interval between successive active edges of the shift register clock pulses.

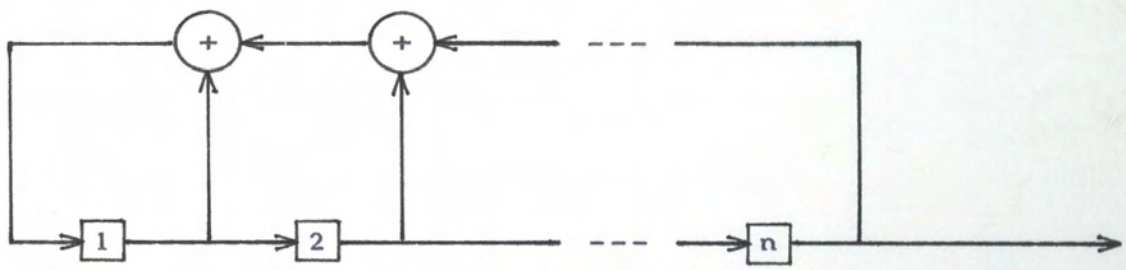
A shift register sequence may possess transients that are made up of a non-cyclic sequence of ones and zeros which are generated before the generator begins its cyclic sequence. Shift register generators which feed back multiple logical combinations of output stages to more than one shift register stage are known as multiple return shift register generators. Those which return only one logical combination to the first shift register stage are termed single return shift register generators. It has been shown that every multiple return shift register generator has an equivalent single return shift register configuration, if the sequence it outputs contains no transients [Holmes 1982].

The sequence output from a shift register may be either linear or non-linear. It has been stated [Holmes 1982] that a shift register generates a linear sequence if the feedback function can be expressed as a mod-2 sum.

All sequences considered in this paper are linear sequences, and all shift register configurations used are of the single return variety. Figure 1, part a, illustrates a typical single return shift register implementation with D flip-flops and exclusive-or gates.



a.



b.

Figure 1. Single Return Shift Register Sequence Generator. a) Typical Circuit. b) Equivalent Block Diagram.

Figure 1, part b, is an equivalent, although more abstract block diagram of the same generator.

The Randomness Postulates

In spread spectrum applications, regardless of the class of system, and irrespective of the type of modulation employed, the spectrum-spreading waveform has to be deterministic so that the modulated signal can be despread and the information unambiguously recovered in the receiver. Yet for reasons cited earlier, this waveform must possess random, noiselike temporal and spectral characteristics. For the spectrum-spreading waveform to possess these pseudo-random noiselike properties, the code sequence from which the waveform is generated must satisfy, to the greatest degree possible, three so-called ramdomness postulates [Holmes 1982].

The first randomness postulate is that the pseudo-random noise code sequence must possess the balance property. Given a code sequence of length L, the total number of ones in the sequence must not differ from the total number of zeros in the sequence by more than one, for every phase shift of the sequence. The significance of a code sequence containing an almost equal number of ones and zeros is that a waveform generated with such a sequence (assuming non-return-to-zero pulse shapes), and the resulting code modulated signal, will contain almost no DC component. No DC component is of considerable practical importance when the modulation scheme employed requires suppression of the carrier component, for example in a direct sequence spread spectrum system employing a balanced modulator. The maximum amount of carrier suppression is directly dependent upon the one and zero balance of the coding waveform. This same balance requirement is one of the reasons that the longest possible code sequences are usually used in direct sequence spread spectrum systems.

The second randomness postulate is that the pseudo-random noise code sequence must possess a characteristic run-length distribution of ones and zeros. A run is defined to be a consecutive series of ones or zeros within a sequence, and the run length is the number of ones or zeros in the series. In every sequence period, the required distribution is that half of the runs (of ones and zeros) have a length of one, one-fourth have a length of two, one-eighth have a length of three, and so on. Implied by this distribution is the fact that the number of runs of ones of each length will be equal to the number of runs of zeros of the same length.

It has been shown by Tausworthe [1965] that if a code sequence displays the above run-length distribution, then the distribution of chips within the sequence is statistically independent. This statistical independence between subset components within the sequence further enhances its random-like characteristic.

The third randomness postulate is that the autocorrelation function of a pseudo-random code sequence must be two-valued.

First, the correlation function in the context of sequences of ones and zeros is defined. For the case of computing correlation functions between discretized, quantized series of ones and zeros, the usual mathematical operation of computing the integral of the product of a function and a (time) shifted version of itself or some other function over an interval simplifies to a simple comparison test between corresponding chips of the two sequences in question. Each corresponding chip pair is said to agree if both members of the pair have a value of one or both have a value of zero, and the pair is said to disagree if both are not one or not zero. The number of agreements and disagreements are tabulated as each chip pair of the two sequences is tested one by one. The resultant correlation value is then just the number of agreements minus the number of disagreements.

Correlation = Total Agreements - Total Disagreements.

The autocorrelation function of a code sequence is then the number of agreements minus the number of disagreements as the code sequence is tested against a phase shifted replica of itself, chip by chip. The cross-correlation function is likewise the number of agreements minus the number of disagreements as one code sequence is tested against a phase shifted version of a second sequence.

To satisfy this third randomness postulate, the autocorrelation function of a code sequence must be two-valued. Furthermore, the autocorrelation value corresponding to a zero shift, representing perfect correlation, should ideally be much larger than the other correlation value, which ideally should be as close to zero as possible.

To illustrate why this is a desirable property of a PN code, a brief reference to stochastic process theory is helpful. Non-bandlimited white Gaussian noise exhibits a flat power spectral density curve and therefore its autocorrelation function is represented as a single delta function at the origin [Stremler 1982]. Hence the autocorrelation function of a code sequence, and thus any waveform generated with it, which contains a single, large "spike" at zero shift and a small, constant value for all other shifts is seen to quite closely approximate the random white Gaussian noise autocorrelation function.

This two-valued autocorrelation property, with the peak value present at the zero shift point only, is an extremely important property from another point of view, that of receiver synchronization. To despread, demodulate, and decode an incoming message successfully, the receiver's internal circuitry must quickly and reliably synchronize itself to the incoming signal. A received signal with the autocorrelation function described above is much easier to synchronize to than one with multiple correlation peaks. This reduces receiver complexity as well as the probability of false synchronization.

Maximal Length Sequences

It is possible to further classify shift register sequences as those which are maximal and those which are not. A maximal code, or a maximal length sequence, is by definition the longest code sequence that can be generated by a shift register generator employing a given number of stages or delay elements [Ziemer 1985]. If a shift register has n stages, then the longest sequence of ones and zeros possible, the maximal length sequence, has a length of $2^n - 1$ chips. For an n-stage shift register generator, the feedback connections alone determine whether the sequence output is maximal or non-maximal.

A shift register generator designed to output a maximal length sequence will generate one sequence only. Initial conditions loaded into the shift register prior to the start of a shift cycle only determine which phase shifted replica of the maximal length sequence's characteristic phase is output. A non-maximal length shift register generator on the other hand can output many different sequences. Initial conditions loaded into its shift register stages determine which sequence is generated.

The subject of characteristic phases and initial conditions is of considerable importance for the Gold code generator design to follow, and both topics are treated in some detail in the following sections.

The near ideal characteristics of maximal length code sequences has caused them to find widespread use in modern communications and ranging systems. Other types of linear codes can do no better than equal their performance in systems employing single coded signals, and maximal length code characteristics are standards against which other codes are measured.

Most importantly, maximal length sequences satisfy the three randomness postulates almost to the letter. For this reason, the terms maximal length sequence and pseudo-random noise code (PN code) are often used synonymously in the literature.

Every maximal length code exhibits the balance property. Specifically, given an output sequence from an n-stage maximal length shift register generator, the total number of ones will be $2^n/2$ and the total number of zeros will be $2^n/2-1$ [Dixon 1984]. Thus the total number of ones exceeds the total number of zeros by one, and the first randomness postulate is satisfied.

Every maximal length sequence exhibits a near ideal run-length distribution. Work by Freymodsson [Dixon 1984] has demonstrated that there are exactly $2^{n-(p+2)}$ runs of length p for both ones and zeros in every maximal length code sequence, with the exception that there is only one run containing n ones and one run containing n-1 zeros, which is to say that there are no zero runs of length n or runs of ones of length n-1. Hence the second randomness postulate is satisfied.

Finally, every maximal length sequence exhibits a binary-valued autocorrelation function [Dixon 1984]. Assuming that the correlation computation is performed over the entire sequence length, the autocorrelation function will have a value of $2^n - 1$ for a zero phase shift, and a value of -1 for every other phase shift.

Other properties of interest possessed by maximal length sequences, or the shift register generators which produce them, are stated below [Holmes 1982].

Maximal length sequences can only be generated by single return shift register generators which use an even number of feedback taps. No generator with an odd number of taps can generate a maximal length sequence.

It is possible to reverse the order of the sequence output from a shift register. If an n-stage single return shift register generator has feedback taps on stages n, k, m, ... and outputs some sequence, then the same sequence will be output, but in reverse order, from a generator with taps on stages n, n-k, n-m, ... assuming n>m>k.

A final feature of maximal length sequences, and one of particular interest from the point of view of Gold code generator design, is that they possess three special linear addition properties.

The first addition property is that if a maximal length sequence is mod-2 added to any phase shifted replica of itself, the resulting sequence is another replica of the original, but with a different phase shift. This property is important for a number of practical applications, and demonstrates the cyclic nature of maximal length sequences.

The second addition property is that if two maximal length sequences of different length, say n and r, are mod-2 added, the resulting composite sequence will have a length $(2^n-1)(2^r-1)$. Although not maximal, the composite sequence will be a segment of a longer maximal length sequence if the two shorter, original sequences are properly chosen.

The third and most important addition property, from the point of view of Gold code generation, is that if two maximal length sequences of the same length, L, are mod-2 added, the resulting sequence will also have a sequence length, L. The resulting composite sequence, although not maximal, will possess very desirable auto and cross-correlation properties if the two shorter, original maximal length sequences are properly chosen.

Advantages of Gold Codes

Gold code generators make use of this third linear addition property directly. Two maximal length sequences are selected, using Gold's selection algorithm described in the next section, and two corresponding shift register generators are designed to generate each sequence. The output of each generator is then mod-2 added, chip by chip, resulting in a composite, although non-maximal sequence which nevertheless possesses useful characteristics. A single generator configuration is capable of generating a whole set of Gold code sequences. Each member of the set has well-behaved autocorrelation functions. Of even more importance, the cross-correlation among all members of the set is always bounded by some maximum value, which can be made arbitrarily small by selection of longer and longer sequence lengths.

There are two primary reasons why the properties possessed by Gold code sequences, and the generators which generate them, are attractive from the point of view of code division multiple access systems.

The first advantage is that a single Gold code generator is capable of generating multiple, unique Gold code sequences. This is not true of a maximal length generator, which outputs one unique sequence only. The distinction between unique code sequences and sequences which are merely phase shifted replicas of one another should be noted. It will be shown that which Gold code is generated depends only upon the initial conditions loaded into the shift registers. Thus each channel in a multiple access system need not have its own unique code generator. All that is required is to load a single code generator with the proper inital condition "keys" associated with a particular channel. A different channel only requires a different set of keys.

The second major advantage, already stated and without doubt the most important, is that the cross-correlation between members of a Gold code sequence is guaranteed to be bounded below some arbitrarily small value. For systems where large numbers of coded, high speed signals must share the same frequency band, low cross-correlation values are mandatory if the probability of false synchronization and inter-signal interference is to be reduced to an acceptable level or eliminated altogether. True maximal length sequences, which possess ideal autocorrelation properties, do not possess ideal cross-correlation properties when cross-correlated against other maximal length sequences. To reduce unwanted cross-correlation, very long maximal length sequences must be used, and even then reducing cross-correlation below some maximum acceptable limit cannot be guaranteed.

A third possible advantage, related to the second, is that Gold code sequences can in general be much shorter than maximal length sequences and still achieve the same or lower acceptable cross-correlation value. Shorter codes implies shorter correlation time in a receiver and thus faster synch acquisition.

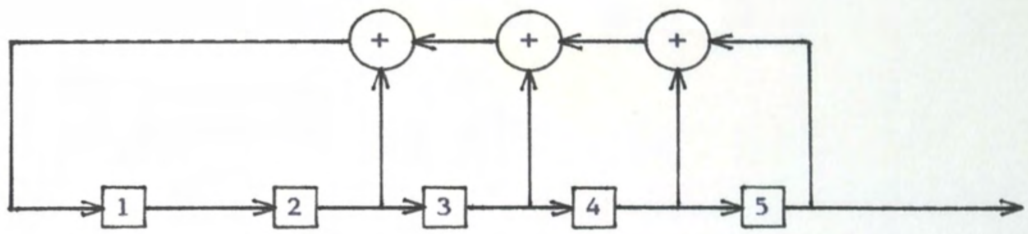
Finally, Gold code generators usually employ fairly simple shift register configurations with relatively few feedback taps and mod-2 addition elements (exclusive-or gates). Propagation delays through these elements limit the maximum chip rate obtainable for a given generator. This can be an especially troublesome problem in systems which require long code sequences in combination with very high data rates. The use of Gold codes in such a situation may then improve system performance by allowing shorter length codes and simpler shift register generators which can operate at the fastest speeds possible.

GENERATION OF GOLD CODES

So far properties and characteristics of maximal length sequences and Gold codes have only been qualitatively stated and examined. But to understand Gold's preferred pair selection algorithm, design a Gold code generator, and model it in software, a thorough mathematical foundation is required. To begin, single return shift register generators with mod-2 additive feedback are first characterized in terms of vectors and matrices.

Mathematical Description Of A Shift Register Generator

The inputs to each stage of an n-stage shift register at any point in time can be described by means of an n x n matrix, denoted hereafter as the $\bar{A}$ matrix. Since mod-2 mathematical operations are always used for shift register analysis, $\bar{A}$ matrix elements will always have values of either zero or one. Each row of the $\bar{A}$ matrix represents the input to the corresponding shift register stage. Elements in a row represent the output of every stage, and those which have a value of one indicate that the output corresponding to that stage is either fed back to the input through the mod-2 adder network or fed forward to the stage immediately following. Figure 2, part a,



a.

$$\overline{A} = \begin{array}{c} \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{array}{l} \longleftarrow \text{ Inputs to Stage 1} \\ \longleftarrow \text{ Inputs to Stage 2} \\ \quad\cdot \qquad\qquad\quad \cdot \\ \quad\cdot \qquad\qquad\quad \cdot \\ \longleftarrow \text{ Inputs to Stage 5} \end{array}$$

b.

Figure 2. Derivation of the $\overline{A}$ Matrix a) Sample Generator b) Equivalent $\overline{A}$ Matrix

is a block diagram of a five-stage generator, and Figure 2, part b, is its equivalent $\overline{A}$ matrix. In Figure 2, part b, the element values of the first row indicate that output of stages two, three, four, and five are fed back to the input of stage one. Values in the second row indicate that the only input to stage two is the fed forward output of stage one, and so on.

If the contents of the shift register stages after the k^{th} shift are denoted by the n-dimensional column vector $\overline{P}(k)$, then the shift register contents after the k+1 shift, represented by $\overline{P}(k+1)$, is given by:

$$\overline{P}(k+1) = \overline{A} \cdot \overline{P}(k)$$

The output of the generator is always the n^{th} element of the $\overline{P}(k)$ vector. Initial conditions, contents of each shift register stage prior to the start of a shift cycle, can be "loaded" into the shift register by assigning values to the contents of the $\overline{P}(k)$ vector prior to the k=0 shift. This simple equation can be used to evaluate the state of the shift register after any series of shift cycles and so mathematically represents the shift register contents and its output sequence as a function of time.

The Generating Function And Characteristic Equation

Up to this point the single return shift register generator was characterized only in terms of its ability to generate code sequences of one type or another. A more fundamental, indeed a much more powerful approach, is to think of the shift register generators described above as mod-2 polynomial division engines. This is to say, a shift register generator is a hardware-equivalent implementation of the mathematical operation of dividing one polynomial by another using mod-2 arithmetic.

If we define the quotient as G(x), then a single return shift register generator computes

$$G(x) = \frac{g(x)}{f(x)} \mod 2$$

where g(x) and f(x) are any two polynomials with one and zero coefficients and where the degree of g(x) is at most one less than the degree of f(x). This quotient, itself a polynomial, is commonly referred to as the mathematical generating function of the shift register generator [Holmes 1982] and its one and zero coefficients, taken in order of increasing powers of x, exactly corresponds to the output sequence of ones and zeros from the generator with which it corresponds.

The derivation of a completely general relationship between any linear single return shift register generator and its mathematical generating function is a straightforward task, but is does consist of a lengthly series of steps and is not presented here. A very thorough treatment of the mathematical characterization of linear shift register generations is given by Peterson [1961]. There are two pertinent conclusions that can be drawn from the results, however. The first is that the numerator polynomial of the generating function expression, g(x), represents the initial condition or state of the shift register stages prior to the start of a shift cycle. The second is that the denominator polynomial, f(x), is the characteristic equation of the $\bar{A}$ matrix, which was defined above as representing the inputs to each shift register stage at any point in time.

One may recall from linear algebra theory that the characteristic equation of an n x n matrix is derived by forming a new matrix of the form $|\bar{A} - x\bar{I}|$, computing its determinant, and setting it equal to zero.

As an example, the characteristic equation computed from the $\bar{A}$ matrix of Figure 2 is given by:

$$f(x) = x^5 + x^4 + x^3 + x^2 + 1$$

The polynomial f(x) above completely and uniquely characterizes, in a mathematical sense, the shift register generator shown in that figure.

An important observation, particularly from a design point of view and true in the general case, is that the relationship of the f(x) polynomial and this generator is demonstrated by the fact that each feedback tap of the generator corresponds to a non-zero coefficient of the polynomial. Thus it is a simple matter, given a polynomial, to design an equivalent sequence generator.

It should be obvious that the use of the generating function model and the characteristic equation enables one to take advantage of the power and elegance of matrix theory and polynomial algebra for the purpose of describing and analyzing linear code sequences, selecting those with desirable properties, and designing equipment to generate them.

Properties Of Polynomials Which Generate Maximal Length Sequences

The first theorem from polynomial algebra that is applied to the mathematical description of maximal length sequences is that if a sequence has a period p, then the characteristic polynomial corresponding to the generator which generates it, f(x), will evenly divide the polynomial $1 + x^p$, that is:

$$\frac{1 + x^p}{f(x)} = 0 \bmod 2$$

Using this theorem, one can then define the sequence period p to be the smallest possible integer such that f(x) divides $1 + x^p$ evenly [Holmes 1982]. This is an important result from the point of view of selecting polynomials to generate maximal length sequences. Given an n-stage shift register, the maximum possible sequence length is $2^n - 1$. Then the only polynomials which will generate sequences of this length are those which will evenly divide $1 + x^{(2^n-1)}$. This is a necessary requirement for a polynomial to be capable of generating a maximal length sequence, but is not by itself sufficient to guarantee that it will do so.

The second property that a polynomial must possess in order to be able to generate a maximal length sequence concerns its relationship to the numerator of the generating function. Recall that the generating function G(x) was defined to be:

$$G(x) = \frac{g(x)}{f(x)}$$

The numerator and denominator polynomials, g(x) and f(x), must not have any factors in common if a maximal length sequence is to be generated. If they do, the factors will cancel, leaving two polynomials of lesser degree, which will result in shorter, non-maximal sequence lengths. To guarantee that g(x) and f(x) have no factors in common, f(x) must be irreducible. Irreducible in the mathematical sense implies that a polynomial of degree n cannot be evenly divided, mod-2, by another polynomial of degree less than n [Peterson 1961].

To eliminate any confusion, it is noted that labeling a polynomial as irreducible does not imply that it cannot be factored. Indeed, a fundamental theorem from polynomial algebra states that any polynomial of degree n can always be reduced to the product of n factors [Hansen 1965]. The irreducible label, in the mod-2 sense, only implies there are no factors which generate a polynomial with real-valued one and zero coefficients.

If f(x) is irreducible, then the period of an output sequence is completely independent of the value of the g(x) polynomial, with the exception of the trivial case when g(x) = 0, resulting in a sequence length of one. An alternate interpretation of this is that only the feedback taps of a maximal length shift register generator represented by f(x) determine which maximal length sequence is generated, a result stated earlier. Initial register conditions, represented by g(x), influence neither the sequence length, nor which sequence is generated. The g(x) polynomial then only determines the starting point, or phase shift, of the output sequence.

As with the first requirement, the irreducible requirement is necessary to generate maximal length sequences, but it is not by itself sufficient to guarantee that a maximal length sequence will always be generated. Not every irreducible polynomial which divides $1 + x^p$ will produce a maximal length sequence. There are two cases to consider.

If $2^n - 1$ is a prime number, every irreducible polynomial of degree n will generate a maximal length sequence. If $2^n - 1$ is not a prime number, then the irreducible polynomial must be primitive if it is to generate a maximal length sequence. From algebraic field theory, an irreducible polynomial of degree n is primitive if and only if it divides $x^m - 1$ for no m less than $2^n - 1$ [Peterson 1961].

In summary, polynomials selected to generate maximal length code sequences must possess three mathematical properties. Given a polynomial of degree n, it must be irreducible, it must be primitive, and it must evenly divide $1 + x^p$ where $p = 2^n - 1$.

Tables of irreducible polynomials for each value of n over the field of two elements, zero and one (called the Galois field by mathematicians), are presented by several authors. Notable among these is Peterson's Table of Irreducible Polynomials [Peterson 1961]. These tables list all irreducible polynomials for degrees n=1 to n=16. Those irreducible polynomials which are also primitive are indicated as such. The first entry for each order is the primitive polynomial with the minimum number of non-zero coefficients. Also tabulated with each polynomial are relationships between the roots of the polynomials relative to the roots of the first primitive polynomial. Specifically, if the first polynomial has some root, β^1, the other polynomials of the same degree which have roots of β raised to some power are indicated. This information is required for Gold's preferred polynomial pair algorithm presented next.

The technique of generating a maximal length code sequence reduces then to selecting the desired code length L, computing n from $L = 2^n - 1$, and choosing a primitive, irreducible polynomial from a table such as Peterson's. The feedback taps of the required generator are connected to correspond to the non-zero coefficients of the polynomial, and the design is complete.

Gold's Preferred Polynomial Pair Algorithm

To generate Gold code sequences of length $2^n - 1$, two primitive irreducible polynomials from Peterson's table are first selected, and an equivalent maximal length code sequence generator corresponding to each polynomial is designed. Then the third linear addition property of maximal length sequences is applied by taking the output sequence of each generator and mod-2 adding them. The resulting sequences will be Gold code sequences if the two polynomials chosen conform to Gold's preferred pair requirements.

Working in the late 1960s and early 1970s, Gold was able to define a relationship between pairs of primitive, irreducible polynomials which guaranteed that the family of composite code sequences generated with the pair would always display bounded cross-correlation values when correlated among all members of family [Gold 1967]. Further work by him proved that both auto and cross-correlation functions of members of the family are always three-valued [Gold 1968]. Further, the maximum normalized cross-correlation bounding value can be made arbitrarily small by using longer code sequence lengths.

Gold's relationship defining such a preferred pair of polynomials can be stated as follows. Let $p_1(x)$ be a primitive polynomial of degree n such that n is not divisible by 4. Let ß be a root of $p_1(x)$, that is, $p_1(ß)=0$. Let $p_2(x)$ be a second primitive polynomial, of the same degree, n, with a root $ß^t$ such that:

$$t = 2^{(n-1)/2} + 1 \qquad \text{for n odd}$$

and

$$t = 2^{(n-2)/2} + 1 \qquad \text{for n even, not = 0mod4}$$

Then $p_1(x)$ and $p_2(x)$ form a preferred pair of polynomials. The n even, not = 0mod4 requirement is to ensure that t is prime relative to $2^n - 1$, which in turn guarantees that $p_1(x)$ and $p_2(x)$ are prime relative to one another.

Given a preferred pair of polynomials $p_1(x)$ and $p_2(x)$ whose corresponding shift register generators generate maximal length sequences of period $2^n - 1$, then a shift register corresponding to the product polynomial $p_1(x)p_2(x)$ will generate $2^n + 1$ different sequences each of length $2^n - 1$ and such that the cross-correlation R(k) between any sequence pair satisfies the inequality:

$$R(k) \leq 2^{(n+1)/2} + 1 \quad \text{for n odd}$$

and

$$R(k) \leq 2^{(n+2)/2} + 1 \quad \text{for n even, not} = 0\bmod 4$$

The $2^n + 1$ distinct codes so generated are Gold code sequences. Gold has shown [Holmes 1982] that if two polynomials $p_1(x)$ and $p_2(x)$ are relatively prime, then any sequence that can be generated by the shift register corresponding to the product polynomial $p_1(x)p_2(x)$ is exactly equivalent to the sum of sequences generated by shift registers corresponding to $p_1(x)$ and $p_2(x)$.

Designing a Gold code generator using two shorter polynomials $p_1(x)$ and $p_2(x)$ of order n instead of a single 2n-ordered polynomial corresponding to $p_1(x)p_2(x)$ is advantageous for a number of reasons. One reason is that implementation of lower order polynomials requires fewer feedback taps and mod-2 addition elements, allowing faster code rates. Another reason is that it is much easier with this configuration to compute and define initial shift register conditions for each Gold code sequence which will generate the greatest number of balanced Gold codes, a topic to be dealt with next.

The Characteristic Phase Of A Maximal Length Sequence

So far the mathematical analysis of maximal length sequences has been almost exclusively concerned with the denominator polynomial of the generating function, $f(x)$. To generate sets of Gold codes, one more piece of theoretical information which concerns the numerator polynomial, $g(x)$, is needed. From earlier remarks, it was noted that specifying $g(x)$ is equivalent to loading initial conditions into the shift register stages prior to the start of a shift cycle, and that although $g(x)$ influences neither the length nor the composition of the output sequence, it does determine which phase shifted replica of the sequence is output. Given a dual shift register Gold code generator, the set of initial conditions loaded into both shift registers determines which of the $2^n + 1$ possible Gold code sequences is output.

As noted previously, a maximal length sequence generator will output only one unique sequence which is

$2^n - 1$ chips in length. But dependant upon the initial load conditions, any one of $2^n - 1$ phase shifted replicas of this sequence may be generated. There is one special phase shifted version of every maximal length sequence, however, referred to as the characteristic phase sequence. The characteristic phase sequence has the remarkable property that if every other chip in the sequence is sampled, beginning with the first chip, then the resulting sequence is identical to the sampled sequence [Holmes 1982].

Gold has shown [Gold 1966] that the formula for the numerator polynomial g(x) which results in the characteristic phase for a maximal length sequence is given by:

$$g(x) = \frac{d[xf(x)]}{dx} \qquad \text{for } n \text{ odd}$$

and

$$g(x) = f(x) + \frac{d[xf(x)]}{dx} \qquad \text{for } n \text{ even}$$

In the equations above, addition is mod-2, and the expression

$$\frac{d[xf(x)]}{dx}$$

denotes the derivative of the one and zero sequence represented by f(x) [Gold 1966]. Evaluation of this expression proceeds by first multipling every term of f(x) by x. Next, the derivative of each term of the resulting polynomial is evaluated in the normal manner. Finally, the coefficients of each term which are even-valued are set equal to zero, and those which are odd-valued are set equal to one. As an example, let:

$$f(x) = x^6 + x^5 + x^2 + x + 1$$

$$xf(x) = x^7 + x^6 + x^3 + x^2 + x$$

$$\frac{d[xf(x)]}{dx} = 1x^6 + 0x^5 + 1x^2 + 0x + 1$$

$$= x^6 + x^2 + 1$$

These formulas for g(x) will be used in the next section in a procedure to define initial condition "keys" for each Gold code sequence output from the generator.

DESIGN OF A GOLD CODE GENERATOR

The Gold code sequence generator designed in this paper uses two sixth-order irreducible, primitive polynomials. Sixth-order polynomials were chosen for two reasons.

The first reason is that even-ordered polynomials will generate sets of Gold code sequences which, when cross-correlated against other members of the set, yield the lowest correlation value more often than do sets generated from odd-ordered polynomials. From Gold's 3-valued cross-correlation analysis, codes generated from even-ordered polynomials, when cross-correlated with shifted versions of other members of the set, will yield the smallest possible normalized correlation value (-1/sequence length) about 75% of the time. Codes generated from odd-ordered polynomials will exhibit this lowest attainable cross-correlation value about 50% of the time [Holmes 1982]. For this reason, Gold codes generated from even-ordered polynomials are the better choice from a systems point of view, since their use further reduces the already low probability of false synchronization and interference between coded signals sharing the same frequency band.

The second reason that sixth-order polynomials were chosen is that the next lowest even order, n=4, and the next largest even order, n=8, are both divisible by 4. Polynomial orders divisible by 4 are not allowed in order to ensure that the polynomials chosen are relatively prime, as required by Gold's preferred pair algorithm. The next useable even order is n=10. The computational difficulties encountered when working with maximal length sequences that are $2^{10}-1 = 1023$ chips long ruled out this and larger values of n for the analysis which will follow.

Selection of the Preferred Pair

For degree n=6, Peterson's Table of Irreducible Polynomials lists the following three primitive polynomials:

$$f_1(x) = x^6 + x + 1$$
$$f_2(x) = x^6 + x^5 + x^2 + x + 1$$
$$f_3(x) = x^6 + x^5 + x^3 + x^2 + 1$$

In addition, the reciprocals of the above polynomials are also primitive and irreducible. The reciprocal of an n^{th} order polynomial is defined to be [Ziemer 1985]:

$$f^{-1}(x) = x^n f(1/x)$$

Using this definition, the three reciprocal polynomials are:

$$f_4(x) = x^6 + x^5 + 1$$

$$f_5(x) = x^6 + x^5 + x^4 + x + 1$$

$$f_6(x) = x^6 + x^4 + x^3 + x + 1$$

It is noted that computing the inverse of a polynomial is equivalent to reversing the order of the output sequence of its generator. The same procedure described in an earlier section to reverse the output sequence of a shift register sequence can be applied to quickly compute the inverse of a polynomial. The original sequence, and the same sequence but in reverse order, are nevertheless two unique code sequences, so the inverse polynomials are valid candidates for the preferred pair algorithm.

For convenience and clarity, Gold's theorem for selecting preferred pairs of polynomials is repeated. Given one primitive, irreducible polynomial of order n which has a root, ß, the second polynomial of the pair will have a root, $ß^t$, where:

$$t = 2^{(n-1)/2} + 1 \qquad \text{for n odd}$$

and

$$t = 2^{(n-2)/2} + 1 \qquad \text{for n even, not} = 0 \bmod 4$$

Thus, for n=6, $t = 2^{(6-2)/2} + 1 = 5$. That is, the second polynomial must have a root, β^5. From Peterson's tables it is found that such a preferred pair, chosen from the set of six polynomials above, is:

$$p_1(x) = x^6 + x + 1$$

and

$$p_2(x) = x^6 + x^5 + x^2 + x + 1$$

Implementation - Design of the Generator

Each member of the preferred pair, being a primitive, irreducible polynomial, can generate a maximal length PN code sequence when implemented with a single return shift register employing mod-2 additive feedback. The Gold code generator employs two of these shift register configurations, one to implement $p_1(x)$, the other to implement $p_2(x)$. Since each polynomial is sixth-order, each shift register consists of six stages. Feedback taps are connected to the output of those stages which correspond to non-zero coefficents in the respective polynomials. The feedback tap values from each stage are added mod-2 and routed to the input of the first stage of the respective generator. The output of each generator, (i.e., the output of stage 6 from each generator) is also connected to a single mod-2 adder. Thus, as a maximal length code sequence chip is generated and shifted out of one generator, it is added, mod-2, to a corresponding chip from the other maximal length sequence. In other words, the maximal length sequence generated by the $p_1(x)$ polynomial is added to the sequence generated by the $p_2(x)$ polynomial as each is shifted out of its respective generator. The resultant sequence is a Gold code sequence, and the combination of the dual maximal length sequence generators with the mod-2 adder at the output comprises the Gold code generator. The block diagram of the Gold code generator using the $p_1(x)$ and $p_2(x)$ polynomials is shown in Figure 3.

Representation of the Generator with a Mathematical Model

For computational convenience, it is desirable to derive a mathematical model of the Gold code generator using code vectors and matrices. The primitive polynomials used to generate the Gold code sequences, $p_1(x)$ and $p_2(x)$, are in fact characteristic equations of two unique matrices, $\overline{A}p_1$ and $\overline{A}p_2$, which can be said to describe the state of the inputs to each stage of each shift register at any point in time. If the contents of the shift registers after some arbitrary shift, k, are designated by the column vectors $\overline{P}_1(k)$ and $\overline{P}_2(k)$, the contents of the

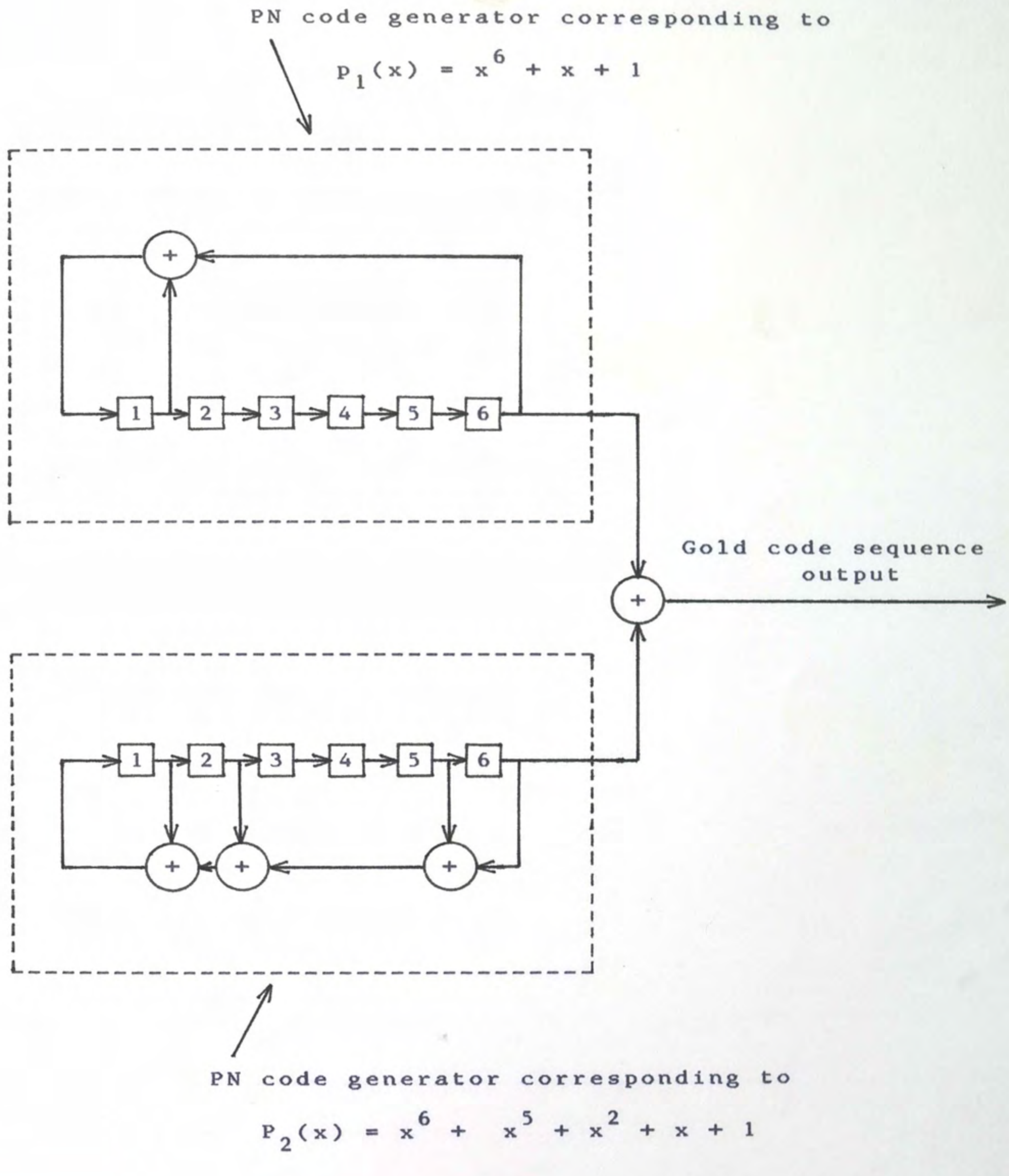


Figure 3. Gold Code Generator

shift registers after the k + 1 shift can be expressed by $\bar{P}_1(k+1)$ and $\bar{P}_2(k+1)$ as follows:

$$\bar{P}_1(k+1) = \bar{A}p_1 \cdot \bar{P}_1(k)$$

and

$$\bar{P}_2(k+1) = \bar{A}p_2 \cdot \bar{P}_2(k)$$

The matrix corresponding to the characteristic equation $p_1(x) = x^6 + x + 1$ is given by:

$$\bar{A}p_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The matrix corresponding to the characteristic equation $p_2(x) = x^6 + x^5 + x^2 + x + 1$ is given by:

$$\bar{A}p_2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The contents of the $\bar{P}_1(x)$ shift register, after the k + 1 shift, is then:

$$\begin{bmatrix} P_{11}(k+1) \\ P_{12}(k+1) \\ P_{13}(k+1) \\ P_{14}(k+1) \\ P_{15}(k+1) \\ P_{16}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_{11}(k) \\ P_{12}(k) \\ P_{13}(k) \\ P_{14}(k) \\ P_{15}(k) \\ P_{16}(k) \end{bmatrix} \quad \underline{\text{Eq. MM1}}$$

or

$$
\begin{aligned}
P_{11}(k+1) &= P_{11}(k) + P_{16}(k) \\
P_{12}(k+1) &= P_{11}(k) \\
P_{13}(k+1) &= P_{12}(k) \\
P_{14}(k+1) &= P_{13}(k) \\
P_{15}(k+1) &= P_{14}(k) \\
P_{16}(k+1) &= P_{15}(k),
\end{aligned}
$$

where " + " implies mod-2 addition.

The contents of the $\overline{P}_2(x)$ shift register, after the k + 1 shift, is given by:

$$
\begin{bmatrix} P_{21}(k+1) \\ P_{22}(k+1) \\ P_{23}(k+1) \\ P_{24}(k+1) \\ P_{25}(k+1) \\ P_{26}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_{21}(k) \\ P_{22}(k) \\ P_{23}(k) \\ P_{24}(k) \\ P_{25}(k) \\ P_{26}(k) \end{bmatrix} \qquad \underline{\text{Eq. MM2}}
$$

or

$$
\begin{aligned}
P_{21}(k+1) &= P_{21}(k) + P_{22}(k) + P_{25}(k) + P_{26}(k) \\
P_{22}(k+1) &= P_{21}(k) \\
P_{23}(k+1) &= P_{22}(k) \\
P_{24}(k+1) &= P_{23}(k) \\
P_{25}(k+1) &= P_{24}(k) \\
P_{26}(k+1) &= P_{25}(k)
\end{aligned}
$$

Again, " + " implies mod-2 addition.

The Gold code sequence vector is then simply expressed as:

$$
\begin{aligned}
\overline{GC} &= [GC_0, GC_2, \ldots GC_{L-1}] \\
&= [P_{16}(0) + p_{26}(0), p_{16}(1) + p_{26}(1), \ldots p_{16}(L-1) + \\
&\quad p_{26}(L-1)],
\end{aligned}
$$

<u>Eq. MM3</u>

where L is the sequence length, $2^6-1 = 63$, and where " + " implies mod-2 addition, as always.

In summary, equations MM1, MM2, and MM3 are the elements which comprise the mathematical model of the Gold code generator.

Designation of Initial Conditions Keys

A Gold code generator implemented with polynomials of order n will generate a set of $2^n + 1$ Gold code sequences. This set of sequences includes the maximal length sequence of each generating polynomial, in its characteristic phase, plus the $2^n - 1$ sequences, each of length $2^n - 1$, which result as one maximal length sequence is phase shifted from its characteristic phase and added to the non-shifted characteristic phase of the other, chip by chip. Hence (2^n-1) codes plus $(1+1)$ codes yields a (2^n+1) code set. The Gold code generator designed above with sixth-order polynomials can thus generate $2^6 + 1 = 65$ Gold code sequences.

Which Gold code sequence is generated from the set of 65 depends upon which phase shifted maximal length sequence is generated by one of the generators, given that the other generator is generating its non-shifted maximal length characteristic phase sequence. Furthermore, the initial conditions loaded into the shift register stages prior to the start of a shift cycle (shift k=0) determine which phase shifted sequence is generated. In other words, selection of the initial shift register conditions determines which Gold code sequence is generated. Hence, for each Gold code in the set, a set of keys must be designated, that is, the initial conditions which must by loaded into each shift register to generate that particular Gold code sequence.

Calculation of Characteristic Phase Sequences

Clearly, to be able to assign sets of initial condition keys and to begin generating Gold codes, one must first determine the characteristic phases of the maximal length sequences generated by each polynomial, $p_1(x)$ and $p_2(x)$. This is achieved by using the output sequence generating function.

Recall that the generating function for a maximal length sequence, $G(x)$, can be expressed as the ratio of two polynomials, as follows:

$$G(x) = \frac{g(x)}{f(x)}$$

Here, $f(x)$ is a primitive, irreducible polynomial. Recall also that Gold has shown that the characteristic phase of a maximal length sequence is generated if the numerator of the generating function, $g(x)$, is of the form

$$g(x) = \frac{d[xf(x)]}{dx} \qquad \text{for } n \text{ odd}$$

and

$$g(x) = f(x) + \frac{d[xf(x)]}{dx} \qquad \text{for } n \text{ even}$$

The initial register conditions required to generate the characteristic phase sequence of the $p_1(x)$ polynomial generator are calculated as follows:

$$g_1(x) = p_1(x) + \frac{d[xp_1(x)]}{dx}$$

$$= x^6 + x + 1 + \frac{d[x(x^6 + x + 1)]}{dx}$$

$$= x^6 + x + 1 + x^6 + 1$$

$$= x$$

In other words, $G_1(x)$ is the characteristic phase sequence from the $p_1(x)$ polynomial generator if $g_1(x) = x$. Proceeding:

$$G_1(x) = \frac{g_1(x)}{p_1(x)} = \frac{x}{x^6 + x + 1}$$

$$= x + x^2 + x^3 + x^4 + x^5 + \ldots$$

To generate the characteristic phase sequence with the $p_1(x)$ polynomial generator, then, the following initial conditions must be loaded into the shift register prior to the start of a shift cycle:

Stage 6 = 0
Stage 5 = 1
Stage 4 = 1
Stage 3 = 1
Stage 2 = 1
Stage 1 = 1

In a like manner, the initial conditions required to generate the characteristic phase with the $p_2(x)$ generator are computed:

$$g_2(x) = p_2(x) + \frac{d[xp_2(x)]}{dx}$$

$$= x^6 + x^5 + x^2 + x + 1 + \frac{d[x(x^6 + x^5 + x^2 + x + 1)]}{dx}$$

$$= x^6 + x^5 + x^2 + x + 1 + x^6 + x^2 + 1$$

$$= x^5 + x.$$

$$G_2(x) = \frac{g_2(x)}{p_2(x)} = \frac{x^5 + x}{x^6 + x^5 + x^2 + x + 1}$$

$$= x + x^2 + x^4 + x^8 + x^{13} + \ldots$$

From this result, it is concluded that to generate the characteristic phase sequence with the $p_2(x)$ polynomial generator, the following initial conditions must be loaded into the shift register prior to the start of a shift cycle:

Stage 6 = 0
Stage 5 = 1
Stage 4 = 1
Stage 3 = 0
Stage 2 = 1
Stage 1 = 0

Table 1 briefly summarizes these results.

Generation of the Gold Code Family

Having solved for the two characteristic phase sequences, the first two Gold code sequences in the set have then also been determined. To generate each of the remaining 63 ($2^n - 1$) members, one of the polynomial generators is forced to output only its characteristic phase sequence, and the other generator is loaded with a set of initial conditions which causes it to output one of its 63 ($2^n - 1$) possible phase shifted sequences (relative to its characteristic phase). The chip by chip sum of the non-shifted and shifted phase sequences results in the output of the corresponding Gold code sequences.

The choice of which generator to output its characteristic phase and which generator to output a phase shifted version of its characteristic phase is not altogether arbitrary. To generate a Gold code family with the greatest number of balanced sequences, the $p_2(x)$ generator must be the one which repeatedly generates its

TABLE 1

CHARACTERISTIC PHASE SEQUENCE
OF EACH SHIFT REGISTER GENERATOR

GENERATOR (POLYNOMIAL)	INITIAL LOAD CONDITION (S6.....S1)	MAXIMAL LENGTH CHARACTERISTIC PHASE SEQUENCE (IN HEX)
$p_1(x)$	0 1 1 1 1 1	3F566ED27179461O
$p_2(x)$	0 1 1 0 1 0	3442CA93C1B98EBF

non-shifted characteristic phase while the $p_1(x)$ generator is keyed to generate phase shifted replicas of its characteristic phase sequence. Table 2 lists all Gold code sequences output from the Gold code generator designed and modeled above. Included with each sequence is its unique set of keys, that is, the initial conditions which must be loaded into the shift registers to generate that particular sequence.

Data for this table was generated by two Basic language programs which implement the previously derived mathematical model for the Gold code generator described above. The programs run on a Radio Shack Model 4 Personal Computer with two floppy disk drives and the TRS-DOS Version 6.0 operating system.

The first program, "POLYSEQ," or "polynomial sequence generator," is designed to generate the characteristic phase sequence, and all its $2^n - 2$ phase shifted replicas, for any primitive, irreducible polynomial, given as input the polynomial order, the polynomial coefficients, and initial shift register contents. Pages 73 through 76 of the Appendix contain the source code listing of the "POLYSEQ" program. Pages 77 and 78 list the maximal length sequence, and all its phase shifted replicas, which is output from the program when given input for the $p_1(x)$ polynomial. Pages 79 and 80 list the output for the $p_2(x)$ polynomial. The first sequence in each list is the characteristic phase sequence. The "POLYSEQ" program can also store each set of maximal length sequences onto floppy disk for use by the second program, "GOLDGEN."

The "GOLDGEN," or "Gold code generation" program, takes the characteristic phase sequence from one of the two sets of sequences stored by the "POLYSEQ" program and mod-2 adds it, chip by chip, to each phase-shifted sequence of the other sequence set stored by "POLYSEQ," therefore generating each Gold code sequence. Pages 81 through 85 of the Appendix contain the source code listing of the "GOLDGEN" program. Pages 86 through 99 contain the output from this program, which used the $p_1(x)$ sequence set as the shifting sequence and the characteristic phase sequence from the $p_2(x)$ sequence set as the non-shifted sequence. This output listing shows both generating maximal length polynomial sequences and the resulting Gold code sequence for each Gold code in the family.

TABLE 2

SET OF GOLD CODE SEQUENCES
AND THEIR INITIAL CONDITION KEYS

GOLD CODE	$P_1(x)$ KEY (S6...S1)	$P_2(x)$ KEY (S6...S1)	GOLD CODE SEQUENCE (HEX)	BALANCED UNBALANCED
1	011111	000000	3F566ED271794610	B
2	000000	011010	3442CA93C1B98EBF	B
3	011111	011010	0B14A441B0C0C8AF	U
4	111111	011010	4AEE1737234B029F	B
5	111110	011010	491B71DA045C96FE	B
6	111101	011010	4EF1BC004A73BE3C	B
7	111010	011010	412417B4D62DEFB8	B
8	110101	011010	5E8F10DDEE914CB0	B
9	101010	011010	61D97E0F9FE80AA0	B
10	010101	011010	1F75A3AB7D1A8680	B
11	101011	011010	622C18E2B8FF9EC1	B
12	010110	011010	189F6E713335AE42	B
13	101100	011010	6DF9835624A1CF45	B
14	011001	011010	073459180B890D4A	U
15	110011	011010	52AFED8455D88955	B
16	100110	011010	799884BCE97B816A	B
17	001101	011010	2FF655CD903D9114	B
18	011011	011010	032BF22F62B1B1E9	B
19	110111	011010	5A90BBEA87A9F013	B
20	101110	011010	69E628614D9973E6	B
21	011101	011010	0F0B0F76D9F8740C	B
22	111011	011010	42D14159F13A7BD9	B
23	110110	011010	5965DD07A0BE6472	B
24	101101	011010	6E0CE5BB03B65B24	B
25	011010	011010	00DE94C245A62588	U
26	110100	011010	5D7A7630C986D8D1	B
27	101001	011010	6633B3D5D1C72262	B
28	010010	011010	10A0381FE144D704	U
29	100100	011010	7D872F8B80433DC9	B
30	001001	011010	27C900A3424CE852	U
31	010011	011010	13555EF2C6534365	B
32	100111	011010	7A6DE251CE6C150B	B
33	001110	011010	281C9B17DE12B9D6	B

TABLE 2 CONTINUED

GOLD CODE	$P_1(x)$ KEY (S6...S1)	$P_2(x)$ KEY (S6...S1)	GOLD CODE SEQUENCE (HEX)	BALANCED UNBALANCED
34	011100	011010	0CFE699BFEEFE06D	U
35	111000	011010	453B8C83BF15531B	B
36	110001	011010	56B046B33CE035F6	B
37	100010	011010	71A7D2D23B0AF82C	B
38	000101	011010	3F88FA1034DF6398	B
39	001011	011010	23D6AB942B7454F1	B
40	010111	011010	1B6A089C14223A23	U
41	101111	011010	6A134E8C6A8EE787	B
42	011110	011010	08E1C2AC97D75CCE	B
43	111100	011010	4D04DAED6D642A5D	B
44	111001	011010	46CEEA6E9802C77A	B
45	110010	011010	515A8B6972CF1D34	B
46	100101	011010	7E724966A754A9A8	B
47	001010	011010	2023CD790C63C090	U
48	010100	011010	1C80C5465A0D12E1	U
49	101000	011010	65C6D538F6D0B603	B
50	010001	011010	174AF5C5AF6BFFC6	U
51	100011	011010	7252B43F1C1D6C4D	B
52	000110	011010	386237CA7AF04B5A	B
53	001100	011010	2C033020B72A0575	U
54	011000	011010	04C13FF52C9E992B	B
55	110000	011010	5545205E1BF7A197	B
56	100001	011010	764D1F087525D0EE	B
57	000010	011010	305D61A4A881321C	U
58	000100	011010	3C7D9CFD13C8F7F9	U
59	001000	011010	243C664E655B7C33	B
60	010000	011010	14BF9328887C6BA7	B
61	100000	011010	75B879E55232448F	B
62	000001	011010	37B7AC7EE6AE1ADE	U
63	000011	011010	33A807498F96A67D	B
64	000111	011010	3B9751275DE7DF3B	U
65	001111	011010	2BE9FDFAF9052DB7	U

PERFORMANCE EVALUATION AND ANALYSIS

Given the set of Gold code sequences output from the computer model of the generator designed in the previous section, properties that they do or do not possess are now examined.

One must recall that Gold code sequences are not maximal length sequences. The relationship stating the equivalence of single and dual single return shift registers was expressed earlier, and is repeated here for convenience. Given two polynomials $p_1(x)$ and $p_2(x)$ that are prime relative to one another, then any sequence generated by the mod-2 sum of $p_1(x)$ and $p_2(x)$ will exactly equal the sequence generated by the product of $p_1(x)$ and $p_2(x)$. The preferred pair of polynomials selected using Gold's criteria are by definition relatively prime. Hence the set of Gold codes derived from the dual shift register implementation of two sixth-order polynomials can also be generated by a single shift register generator representing a twelfth-order polynomial which is the product of the two sixth-order equations.

Given $p_1(x)$ and $p_2(x)$ from the previous section, the twelfth-order product polynomial is:

$$\begin{aligned} r(x) &= p_1(x) \cdot p_2(x) \\ &= (x^6 + x + 1) \cdot (x^6 + x^5 + x^2 + x + 1) \\ &= x^{12} + x^{11} + x^8 + x^6 + x^5 + x^3 + 1 \end{aligned}$$

The equivalent generator is shown in Figure 4. The point to be noted here is that the Gold code sequences generated by the dual register generator represent the output of a single generator implementation of a twelfth-order polynomial. For the output sequences of a twelfth-order generator to be maximal length, they must have a repetition period of $2^{12} - 1 = 4095$ chips. Clearly, the Gold code sequences, with repetition periods of 63 chips, are not maximal length.

The next obvious question is that if Gold codes are not maximal length sequences, do they still satisfy the randomness postulates and qualify as pseudo-random noise codes? It turns out that all members of a Gold code family do not strictly satisfy the three randomness postulates to the same degree that maximal length sequences do, but there is always a subset of the family whose members come close enough to doing so to be useful nevertheless.

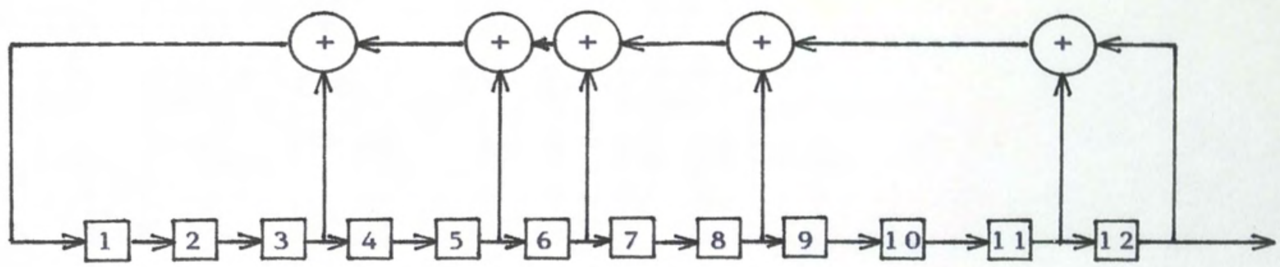


Figure 4. Equivalent single shift register implementation of the Gold code generator,
$r(x) = x^{12} + x^{11} + x^{8} + x^{6} + x^{5} + x^{3} + 1.$

Run-Length Distribution Analysis

Consider first the run-length distribution of ones and zeros. Table 3 lists run-length distribution data for the first ten Gold code sequences output by the computer model of the generator. The first two entries do indeed verify Freymodsson's run-length distribution result for maximal length sequences because these two codes are in fact true maximal length sequences. Although several of the remaining eight codes have run-length distributions that closely approximate the ideal distribution exhibited by Codes 1 and 2, most do not. Code 9, for instance, has a run of eight consecutive ones embedded within it, and Code 6 has a run of eleven consecutive zeros. Run lengths of this size are undesirable, particularly for shorter length sequences, since correlation and false synchronization errors are more likely to occur. In practice, this undesirable characteristic can be made less of a problem by using codes with longer sequence lengths.

Identification of Balanced Members of the Family

In addition to listing all Gold code sequences output from the generator along with their initial condition keys, Table 2 also indicates which sequences are balanced and which are not. Forty-nine of the 65 sequences exhibit the balanced property and 16 do not. Because of the

TABLE 3

RUN-LENGTH DISTRIBUTIONS OF THE FIRST TEN GOLD CODES

GOLD CODE NUMBER	NUMBER OF RUNS OF ONES OF LENGTH								NUMBER OF RUNS OF ZEROS OF LENGTH										
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	9	10	11
1*	8	4	2	1	0	1	0	0	8	4	2	1	1	0	0	0	0	0	0
2*	8	4	2	1	0	1	0	0	8	4	2	1	1	0	0	0	0	0	0
3	10	5	0	1	0	0	0	0	6	3	4	1	1	1	0	0	0	0	0
4	8	3	4	0	0	1	0	0	8	5	1	1	0	1	0	0	0	0	0
5	8	4	3	0	0	0	1	0	8	4	3	0	0	1	0	0	0	0	0
6	4	1	3	3	1	0	0	0	4	5	2	0	0	0	0	0	0	0	1
7	8	4	1	2	1	0	0	0	8	4	2	1	1	0	0	0	0	0	0
8	8	3	2	3	0	0	0	0	8	3	3	2	0	0	0	0	0	0	0
9	6	2	1	0	1	1	0	1	6	2	0	1	2	0	1	0	0	0	0
10	8	4	2	0	2	0	0	0	12	0	2	1	0	0	0	0	1	0	0

* Codes 1 and 2 are true maximal length sequences and exhibit the ideal run-length distribution for a sequence length of 63 chips.

importance of using balanced code sequences for the reasons cited earlier, these sixteen unbalanced codes would not be used in a practical application.

Close examination of the data in Table 2 reveals that unbalanced codes are only generated in cases where both initial condition keys have the same value (either both zero or both one in the general case, but both zero in this particular case) in the position corresponding to stage 6. This is equivalent to stating that unbalanced codes are only generated when the two generating maximal length sequences both begin with zero or both begin with one. The converse is not true, however. Balanced codes may be generated by any combination of the initial values of the generating sequence. However, if the initial values are different, balanced codes will always result.

Gold has shown that this is true in the general case [Gold 1976]. Balanced members of a Gold code family will be guaranteed if the initial condition keys are selected as follows. The generator corresponding to the preferred pair polynomial with the β^t root, the $p_2(x)$ polynomial for the generator designed in this paper, is "keyed" to generate its characteristic phase sequence. If the characteristic phase sequence of the β^t sequence begins with a zero, then any phase shifted version of the characteristic phase of the other polynomial, the β sequence, which begins with a one will always generate a balanced Gold code. Likewise, if the β^t sequence begins with a one, the β sequence must begin with a zero to ensure balance. Initial condition keys for the generator designed in this paper were chosen according to this rule, and the results tabulated in Table 2 confirm Gold's result.

Being able to predict whether a Gold code will be balanced or not simply by examination of the initial condition keys is very useful when selecting codes from a very large family of very long sequences, which will be the case in most practical applications. It is also important to realize that not all code sequences available from a Gold code generator are always suitable for use. If balanced codes are required, then out of $2^n + 1$ code sequences available, only about half are guaranteed to be balanced (a few more may be), a factor which must be considered before the code length that a system will employ is decided upon.

Correlation Analysis

The work by Gold has shown that the cross-correlation function evaluated between any two members of a Gold code family is always three-valued. Moreover, the autocorrelation function of every family will always evaluate to one of the same three values, with the addition of one occurrence of one other value, that for a shift of k=0 representing perfect correlation.

Gold's theory predicts each value, which is dependent solely on the register length, n, of the maximal length sequence generators. Table 4 summarizes the predicted unnormalized autocorrelation and normalized cross-correlation values for any member of a Gold code sequence family.

From Gold's predicted values, the autocorrelation value representing perfect correlation will be $2^6 - 1 = 63$, for n=6, while the remaining three values will always be either -1, $-(2^{(6+2)/2} + 1) = -17$, or $(2^{(6+2)/2} - 1) = 15$.

Using the same values, the normalized cross-correlation function will always evaluate to one of the three values, -1/63, -17/63, or 15/63. Hence, for a sixth-order Gold code generator, the cross-correlation between any two code sequences output from it will always have a value no greater than 17/63 = 0.2698, or about 27% of the maximum.

At this point it is worth demonstrating the performance gain which may be realized by increasing the Gold code sequence length and the tradeoffs that must be considered when doing so. Table 5 summarizes sequence lengths and maximum normalized cross-correlation values for every possible Gold code family from degree 5 to degree 21. As expected, increasing sequence length decreases maximum cross-correlation bounding value.

TABLE 4

PREDICTED CORRELATION VALUES FOR GOLD CODE FAMILY MEMBERS WITH SEQUENCE LENGTH $L = 2^n - 1$

REGISTER LENGTH n	UNNORMALIZED AUTOCORRELATION	NORMALIZED CROSS-CORRELATION
n ODD	L -1 $-(2^{(n+1)/2} + 1)$ $(2^{(n+1)/2} - 1)$	$-1/L$ $-(2^{(n+1)/2} + 1)/L$ $(2^{(n+1)/2} - 1)/L$
n EVEN AND NOT = 0MOD4	L -1 $-(2^{(n+2)/2} + 1)$ $(2^{(n+2)/2} - 1)$	$-1/L$ $-(2^{(n+2)/2} + 1)/L$ $(2^{(n+2)/2} - 1)/L$

TABLE 5

MAXIMUM NORMALIZED CROSS-CORRELATION VALUES FOR GOLD CODE SEQUENCE DEGREES FROM 5 TO 21

DEGREE N	SEQUENCE LENGTH	MAXIMUM NORMALIZED CROSS-CORRELATION
5	31	9/31 = .2903
6	63	17/63 = .2698
7	127	17/127 = .1339
9	511	33/511 = .0646
10	1023	65/1023 = .0635
11	2047	65/2047 = .0318
13	8191	129/8191 = .0157
14	16383	257/16383 = .0157
15	32767	257/32767 = .0078
17	131071	513/131071 = .0039
18	262143	1025/262143 = .0039
19	524287	1025/524287 = .0020
21	2097151	2049/2097151 = .0010

A maximum value below 10% is guaranteed for Gold code sequences with degree greater than 7, and values less than 1% are guaranteed for sequences with degree greater than 14.

The data in Table 5 further reveals, that if maximum correlation values are the sole selection criteria when choosing a Gold code sequence length, then there is no advantage in choosing a sequence length of 1023 over 511, as an example, since both sequences will have about the same maximum correlation value. Indeed, using one sequence length twice as long as another when both achieve the same cross-correlation values will only needlessly double the synch acquisition time in the receiver. On the other hand, using a sequence length of 1023 instead of 511 will at least double the number of useable Gold code sequences available. The point to note here is that code length selection has a great impact on many system parameters and is no exception to the rule that as in any engineering application, tradeoffs must be made based upon individual system requirements.

Correlation Analysis Program Description

Three Basic language programs were developed for the purpose of calculating and studying the auto and cross-correlation properties of both maximal length and Gold code sequences. Although these three programs were specifically designed to analyze data from the Gold code generator designed in this paper, they are general purpose and can be used to analyze the correlation properties of any set of code sequences. The only practical constraint is on the code sequence length. Longer sequences require more internal memory in the computer. Hence, the maximum code sequence length which can be accommodated by the three programs is limited only by the available memory in the computer on which they are used.

The first program, "AUTOCOR," or "autocorrelation calculation" program, was designed to compute autocorrelation values for all the maximal length and Gold code sequences generated. The program operates on data in disk files stored by programs such as "POLYSEQ" and "GOLDGEN" and outputs, in list form, the autocorrelation values for every possible phase shift of the input sequence. Pages 100 and 101 of the Appendix contain the source code listing of the "AUTOCOR" program. Sample output from this program is also included in the Appendix. Pages 102 and 103 show the autocorrelation calculation results for the maximal length sequence output from the $P_1(x)$ sequence generator. Page 104 is a plot of this data. It is seen that the autocorrelation function for this sequence is two valued, as predicted by theory, and demonstrates the ideal autocorrelation property of a maximal length PN code sequence. Pages 105 and 106 show the autocorrelation calculation results for the Gold Code 43 sequence. Page 107 is a plot of this data. Again, the results confirm those predicted by theory.

The second program, "CROSSCOR," or "cross-correlation calculation" program, was designed to compute cross-correlation values for all maximal length and Gold code sequences generated. Like the "AUTOCOR" program, this one also operates on data in disk files stored by programs such as "POLYSEQ" and "GOLDGEN" and outputs, in list form, the cross-correlation values for every possible phase of one code sequence relative to another. Pages 108 through 110 of the Appendix contain the source code listing of the "CROSSCOR" program. Sample output from the program is also included in the Appendix. Pages 111 and 112 show the cross-correlation calculation results when Gold Code 57 is correlated against Gold Code 13. Page 113 is a plot of these results. This data confirms the expected results that the cross-correlation function between members of a Gold code family is always three-valued. Further, the three values obtained are those predicted by theory.

The third and final Program, "CORRSUM," or "correlation analysis summary" program, is designed to compute and summarize auto and cross-correlation calculation results computed across some subset of a set of code sequences generated by programs such as "POLYSEQ" or "GOLDGEN." This program also operates on data stored in disk files, and outputs a tabulation of correlation values and the frequency of occurrence of each. Pages 114 through 117 of the Appendix list the source code listing of the "CORRSUM" program. Pages 118 through 126 contain a sample of the output from this program. This sample output tabulates the results of a complete auto and cross-correlation analysis between a subset of the Gold code family bounded between Gold codes 17 and 21. In addition to confirming earlier results, the tabulated data shows that the minimum unnormalized cross-correlation value, -1, does occur about 75% of the time, a result predicted by Gold to occur if maximal length sequence generators designed from even-ordered polynomials are used to generate the Gold code sequence family.

SUMMARY

This paper on the design of a Gold code generator and the study of Gold code properties is concluded by briefly summarizing the important points noted in the previous pages.

Of the many advantages obtained using spread spectrum techniques, one of the most important is that they are very much suited to applications where multiple message signals must share the same frequency band at the same time, or where it is necessary to selectively address remote receivers or transmitters quickly and easily. Since code sequences are used to directly control the modulation processes, code division multiplexing techniques are especially well suited for use with spread spectrum systems, especially those of the direct sequence type.

To enable multiple signals to share the same frequency band at the same time without interfering with one another, the pseudo-random code sequences used must possess certain characteristics, chief among which is that the signals generated with them must not cross-correlate to any appreciable degree. A family of linear, pseudo-random noise codes which possess near ideal properties for code division multiple access applications are the so-called Gold code sequences.

Working in the late 1960s and early 1970s at the Magnavox Research Labs, Robert Gold defined an algorthm for selecting two maximal length pseudo-random noise code sequences which, when mod-2 added, yield a family of composite sequences with some very interesting properties. The property of most importance is that the cross-correlation value between any two members of the family is always bounded by some maximum value and is dependent solely on the sequence length. Further, this maximum bounding value can be made arbitrarily small by simply increasing the sequence length. Hence this composite set of sequences, the Gold code family, is well suited for use in code division multiple access systems.

Another advantage afforded by the use of Gold codes is that a single generator designed to generate Gold code sequences can actually generate the entire family of unique codes simply by loading different initial condition "keying" sequences into the shift registers of the generator prior to the start of a shift cycle. When used in conjunction with a direct sequence spread system configuration with a single carrier frequency, for example, no retuning is required to switch channels. All that is required is to key in a different set of shift register load values to generate the code sequence designated for the new channel. In such a configuration, it is possible for a transmitter to selectively address a remote receiver which has been keyed to receive data in one channel only. Likewise different initial condition codes can be keyed into a receiver to pick out signals from different transmitters, again without retuning.

Other desirable characteristics are that Gold codes can in general be much shorter codes, allowing faster synch acquisition times. Also, Gold code generators are in general simple and can so operate at very high speed.

The procedure used to design the sixth-order Gold code generator in this paper is perfectly general, and can be used to design any Gold code generator of any valid sequence length. In addition, the procedures outlined to select polynomials with the required mathematical properties, to configure the feedback taps of single return shift register generators, and compute the characteristic phase of maximal length sequences can be used to construct maximal length pseudo-random noise code generators for systems which do not require the unique advantages afforded by Gold codes but which do require high performance pseudo-random code generators. The software programs developed for this paper are general purpose design and analytic tools which are suited to a wide range of system analysis and design tasks. Finally, since code type, code selection, and code length affect so many communication system performance parameters, it is hoped that the results presented in this paper can be used to aid the systems designer faced as always with the inevitable performance tradeoffs which must be made.

Gold codes are employed in a number of modern spread spectrum communications systems. One example is the Tracking and Data Relay Satellite System (TDRSS). In this direct sequence spread spectrum system, a long maximal length code sequence is used in conjunction with 1023 chip Gold code sequences which are used to provide initial synch acquisition. Another system is the Global Positioning Satellite System (GPSS). In this system, 1023 chip Gold code sequences are again used for synch acquisition in conjunction with a very long non-linear pseudo-random noise code. In one typical receiver configuration, signals from up to four different satellites are synchronized at the same time. Each signal can be differentiated because each is coded with a different Gold code sequence. This system takes full advantage of all the Gold code properties mentioned earlier; low cross-correlation between signals in the same band, selective addressing capability, fast synch acquisition made possible by short sequence lengths, and very high code rates.

Future systems to which Gold codes will be well suited are the new Direct Broadcast Satellite System (DBSS), as well as various new mobile communication and cellular radio applications. In each case, the properties of low cross-correlation, selective addressing, fast synch acquisition and high-speed performance afforded by the use of Gold codes should enable system designers to successfully implement systems with the utmost performance and reliability.

APPENDIX

This Appendix contains computer program material which was developed for use in the design, modeling, and analysis of the Gold code generator and its component parts. Included here are Basic language program source code listings and sample output from the following five programs:

"POLYSEQ" (Polynomial Sequence Generator Program),

"GOLDGEN" (Gold Code Generator Program),

"AUTOCOR" (Autocorrelation Calculation Program),

"CROSSCOR" (Cross-Correlation Calculation Program),

"CORRSUM" (Correlation Analysis Summary Program).

```
10 REM
20 REM                          PROGRAM "POLYSEQ"
30 REM
40 REM          MAXIMAL LENGTH POLYNOMIAL SEQUENCE GENERATOR
50 REM
60 REM      THIS PROGRAM IS DESIGNED TO COMPUTE THE CHARACTERISTIC SEQUENCE, AND
70 REM      ALL ITS PHASE-SHIFTED REPLICAS, FOR A PRIMITIVE, IRREDUCIBLE
80 REM      POLYNOMIAL, GIVEN AS INPUT THE POLYNOMIAL ORDER, THE COEFFICIENTS
90 REM      OF THE POLYNOMIAL (1'S OR 0'S), AND THE INITIAL SHIFT REGISTER
100 REM     CONTENTS.
110 REM
120 REM     BASED UPON THESE INPUTS, THE PROGRAM GENERATES THE MATRIX OF THE
130 REM     SINGLE RETURN SHIFT REGISTER MODEL CORRESPONDING TO THE INPUT
140 REM     POLYNOMIAL COEFFICIENTS AND USES IT TO COMPUTE ALL POSSIBLE
150 REM     PHASE-SHIFTED SEQUENCES, RELATIVE TO THE SEQUENCE CORRESPONDING
160 REM     TO THE INITIAL SHIFT REGISTER CONTENTS SPECIFED.
170 REM
180 REM
190 CLS
200 PRINT
210 PRINT"                 MAXIMAL LENGTH SEQUENCE GENERATION PROGRAM"
220 PRINT
230 REM
240 REM INPUT POLYNOMIAL PARAMETERS FROM OPERATOR.
250 REM
260 INPUT"ENTER ORDER OF POLYNOMIAL ---";PORDER
270 SLENGTH = (2^PORDER) - 1
280 REM
290 REM DIMENSION THE ARRAYS.
300 REM
310 DIM POLYOUT(SLENGTH),POLYCO(PORDER),MATRIX(PORDER,PORDER)
320 DIM STAGE(SLENGTH),STAGE1(SLENGTH),PBUFFER$(SLENGTH)
330 REM
```

```
340 INPUT"ENTER POLYNOMIAL COEFFICIENTS (C1, C2, ... CN)";COSTRING$
350 FOR I = 1 TO PORDER
360    IF MID$(COSTRING$,I,1) = "1" THEN POLYCO(I) = 1:GOTO 380
370    POLYCO(I) = 0
380 NEXT I
390 REM
400 REM GENERATE THE MATRIX BASED UPON THE INPUT COEFFICIENTS.
410 REM
420 FOR I = 1 TO PORDER
430    MATRIX(1,I) = POLYCO(I)
440 NEXT I
450 REM
460 FOR I = 2 TO PORDER
470       MATRIX(I,I-1) = 1
480 NEXT I
490 REM
500 REM ASK THE OPERATOR FOR THE INITIAL SHFT REGISTER CONTENTS AND THE NUMBER
510 REM OF SEQUENCES TO COMPUTE.
520 REM
530 INPUT"ENTER INITIAL REGISTER CONDITIONS (S1, S2, ...SN)";INSTAGE$
540 FOR I = 1 TO PORDER
550     IF MID$(INSTAGE$,I,1) = "1" THEN STAGE(I) = 1:GOTO 570
560     STAGE(I) = 0
570 NEXT I
580 INPUT"ENTER THE NUMBER OF SEQUENCES REQUIRED...";MSEQ
590 REM
600 REM BEGIN BY COMPUTING THE FIRST SEQUENCE, USING THE GENERATING MATRIX.
610 REM
620 FOR I = 1 TO SLENGTH
630    POLYOUT(I) = STAGE(PORDER)
640      FOR J = 1 TO PORDER
650         STAGE1(J) = 0
660            FOR K = 1 TO PORDER
670               STAGE1(J) = STAGE1(J) + STAGE(K)*MATRIX(J,K)
680            NEXT K
690      NEXT J
700 REM
```

```
710      FOR L = 1 TO PORDER
720        IF (((STAGE1(L)/2) - INT(STAGE1(L)/2))) > 0 THEN STAGE(L)=1:GOTO 740
730        STAGE(L) = 0
740     NEXT L
750 NEXT I
760 REM
770 REM THE REMAINING SEQUENCES ARE GENREATED BY SIMPLY SHIFTING THE MOST
780 REM RECENT SEQUENCE. THE SUBROUTINE CALLED DOES THIS AND RETURNS THE NEW
790 REM OUTPUT STRING IN "PSTRING". THE SHIFTED STRINGS ARE STORED IN THE
800 REM "PBUFFER$" ARRAY AS THEY ARE GENERATED.
810 REM
820 FOR M = 1 TO MSEQ
830    GOSUB 1080
840    PRINT"SHIFT";M-1;"=";PSTRING$
850    PBUFFER$(M) = PSTRING$
860    POLYSAVE = POLYOUT(1)
870       FOR I = 1 TO (SLENGTH-1)
880         POLYOUT(I) = POLYOUT(I+1)
890       NEXT I
900    POLYOUT(SLENGTH) = POLYSAVE
910 NEXT M
920 REM
930 REM NOW PROMPT THE OPERATOR TO DETERMINE WHETHER TO OUTPUT THE GENERATED
940 REM SEQUENCES TO THE PRINTER OR ONTO FLOPPY DISK.
950 REM
960 PRINT
970 PRINT"SELECT OUTPUT DESTINATION  OF THE SEQUENCES:"
980 PRINT
990 PRINT"          1 = OUTPUT TO PRINTER"
1000 PRINT"         2 = OUTPUT TO DISK"
1010 PRINT"         3 = QUIT"
1020 PRINT
1030 INPUT CHOICE
1040 IF CHOICE = 1 THEN GOSUB 1180:GOTO 970
1050 IF CHOICE = 2 THEN GOSUB 1280:GOTO 970
1060 STOP
```

```
1070 REM
1080 REM  SUBROUTINE TO CONVERT THE "POLYOUT" ARRAY FROM INTEGER ONES AND
1090 REM  ZEROS TO AN ASCII STRING.
1100 REM
1110 PSTRING$ = STRING$(SLENGTH,"0")
1120    FOR K = 1 TO SLENGTH
1130       IF POLYOUT(K) = 1 THEN MID$(PSTRING$,K,1) = "1":GOTO 1150
1140       MID$(PSTRING$,K,1) = "0"
1150    NEXT K
1160 RETURN
1170 REM
1180 REM
1190 REM THE FOLLOWING SEGMENT PRINTS THE CONTENTS OF PBUFFER$ TO THE PRINTER.
1200 REM
1210 PRINT"INPUT PAGE HEADER":INPUT HEADER$
1220 LPRINT:LPRINT HEADER$:LPRINT
1230 FOR M = 1 TO MSEQ
1240    LPRINT"       SHIFT";M-1;"= ";PBUFFER$(M)
1250 NEXT M
1260 RETURN
1270 REM
1280 REM THE FOLLOWING SEGMENT PROMPTS THE OPERATOR FOR A FILE NAME AND THEN
1290 REM WRITES THE CONTENTS OF THE PBUFFER$ ARRAY ONTO DISK.
1300 REM
1310 PRINT
1320 INPUT"       ENTER A FILE NAME";CODEFILE$
1330 REM
1340 OPEN "D",1,CODEFILE$,SLENGTH
1350 REM
1360 FIELD 1, SLENGTH AS CODEBUFFER$
1370 REM
1380 FOR I = 1 TO MSEQ
1390    LSET CODEBUFFER$ = PBUFFER$(I)
1400    PUT 1
1410 NEXT I
1420 CLOSE
1430 RETURN
```

MAXIMAL LENGTH SEQUENCE AND ALL PHASE SHIFTS FOR THE P1(X) POLYNOMIAL

```
SHIFT 0 = 011111101010110011011101101001001110001011110010100011000010000
SHIFT 1 = 111111010101100110111011010010011100010111100101000110000100000
SHIFT 2 = 111110101011001101110110100100111000101111001010001100001000001
SHIFT 3 = 111101010110011011101101001001110001011110010100011000010000011
SHIFT 4 = 111010101100110111011010010011100010111100101000110000100000111
SHIFT 5 = 110101011001101110110100100111000101111001010001100001000001111
SHIFT 6 = 101010110011011101101001001110001011110010100011000010000011111
SHIFT 7 = 010101100110111011010010011100010111100101000110000100000111111
SHIFT 8 = 101011001101110110100100111000101111001010001100001000001111110
SHIFT 9 = 010110011011101101001001110001011110010100011000010000011111101
SHIFT 10 = 101100110111011010010011100010111100101000110000100000111111010
SHIFT 11 = 011001101110110100100111000101111001010001100001000001111110101
SHIFT 12 = 110011011101101001001110001011110010100011000010000011111101010
SHIFT 13 = 100110111011010010011100010111100101000110000100000111111010101
SHIFT 14 = 001101110110100100111000101111001010001100001000001111110101011
SHIFT 15 = 011011101101001001110001011110010100011000010000011111101010110
SHIFT 16 = 110111011010010011100010111100101000110000100000111111010101100
SHIFT 17 = 101110110100100111000101111001010001100001000001111110101011001
SHIFT 18 = 011101101001001110001011110010100011000010000011111101010110011
SHIFT 19 = 111011010010011100010111100101000110000100000111111010101100110
SHIFT 20 = 110110100100111000101111001010001100001000001111110101011001101
SHIFT 21 = 101101001001110001011110010100011000010000011111101010110011011
SHIFT 22 = 011010010011100010111100101000110000100000111111010101100110111
SHIFT 23 = 110100100111000101111001010001100001000001111110101011001101110
SHIFT 24 = 101001001110001011110010100011000010000011111101010110011011101
SHIFT 25 = 010010011100010111100101000110000100000111111010101100110111011
SHIFT 26 = 100100111000101111001010001100001000001111110101011001101110110
SHIFT 27 = 001001110001011110010100011000010000011111101010110011011101101
SHIFT 28 = 010011100010111100101000110000100000111111010101100110111011010
SHIFT 29 = 100111000101111001010001100001000001111110101011001101110110100
SHIFT 30 = 001110001011110010100011000010000011111101010110011011101101001
SHIFT 31 = 011100010111100101000110000100000111111010101100110111011010010
SHIFT 32 = 111000101111001010001100001000001111110101011001101110110100100
```

```
SHIFT 33 = 110001011110010100011000010000011111101010110011011101101001001
SHIFT 34 = 100010111100101000110000100000111111010101100110111011010010011
SHIFT 35 = 000101111001010001100001000001111110101011001101110110100100111
SHIFT 36 = 001011110010100011000010000011111101010110011011101101001001110
SHIFT 37 = 010111100101000110000100000111111010101100110111011010010011100
SHIFT 38 = 101111001010001100001000001111110101011001101110110100100111000
SHIFT 39 = 011110010100011000010000011111101010110011011101101001001110001
SHIFT 40 = 111100101000110000100000111111010101100110111011010010011100010
SHIFT 41 = 111001010001100001000001111110101011001101110110100100111000101
SHIFT 42 = 110010100011000010000011111101010110011011101101001001110001011
SHIFT 43 = 100101000110000100000111111010101100110111011010010011100010111
SHIFT 44 = 001010001100001000001111110101011001101110110100100111000101111
SHIFT 45 = 010100011000010000011111101010110011011101101001001110001011110
SHIFT 46 = 101000110000100000111111010101100110111011010010011100010111100
SHIFT 47 = 010001100001000001111110101011001101110110100100111000101111001
SHIFT 48 = 100011000010000011111101010110011011101101001001110001011110010
SHIFT 49 = 000110000100000111111010101100110111011010010011100010111100101
SHIFT 50 = 001100001000001111110101011001101110110100100111000101111001010
SHIFT 51 = 011000010000011111101010110011011101101001001110001011110010100
SHIFT 52 = 110000100000111111010101100110111011010010011100010111100101000
SHIFT 53 = 100001000001111110101011001101110110100100111000101111001010001
SHIFT 54 = 000010000011111101010110011011101101001001110001011110010100011
SHIFT 55 = 000100000111111010101100110111011010010011100010111100101000110
SHIFT 56 = 001000001111110101011001101110110100100111000101111001010001100
SHIFT 57 = 010000011111101010110011011101101001001110001011110010100011000
SHIFT 58 = 100000111111010101100110111011010010011100010111100101000110000
SHIFT 59 = 000001111110101011001101110110100100111000101111001010001100001
SHIFT 60 = 000011111101010110011011101101001001110001011110010100011000010
SHIFT 61 = 000111111010101100110111011010010011100010111100101000110000100
SHIFT 62 = 001111110101011001101110110100100111000101111001010001100001000
```

MAXIMAL LENGTH SEQUENCE AND ALL PHASE SHIFTS FOR THE P2(X) POLYNOMIAL

```
SHIFT 0 = 011010001000010110010101001001111000001101110011000111010111111
SHIFT 1 = 110100010000101100101010010011110000011011100110001110101111110
SHIFT 2 = 101000100001011001010100100111100000110111001100011101011111101
SHIFT 3 = 010001000010110010101001001111000001101110011000111010111111011
SHIFT 4 = 100010000101100101010010011110000011011100110001110101111110110
SHIFT 5 = 000100001011001010100100111100000110111001100011101011111101101
SHIFT 6 = 001000010110010101001001111000001101110011000111010111111011010
SHIFT 7 = 010000101100101010010011110000011011100110001110101111110110100
SHIFT 8 = 100001011001010100100111100000110111001100011101011111101101000
SHIFT 9 = 000010110010101001001111000001101110011000111010111111011010001
SHIFT 10 = 000101100101010010011110000011011100110001110101111110110100010
SHIFT 11 = 001011001010100100111100000110111001100011101011111101101000100
SHIFT 12 = 010110010101001001111000001101110011000111010111111011010001000
SHIFT 13 = 101100101010010011110000011011100110001110101111110110100010000
SHIFT 14 = 011001010100100111100000110111001100011101011111101101000100001
SHIFT 15 = 110010101001001111000001101110011000111010111111011010001000010
SHIFT 16 = 100101010010011110000011011100110001110101111110110100010000101
SHIFT 17 = 001010100100111100000110111001100011101011111101101000100001011
SHIFT 18 = 010101001001111000001101110011000111010111111011010001000010110
SHIFT 19 = 101010010011110000011011100110001110101111110110100010000101100
SHIFT 20 = 010100100111100000110111001100011101011111101101000100001011001
SHIFT 21 = 101001001111000001101110011000111010111111011010001000010110010
SHIFT 22 = 010010011110000011011100110001110101111110110100010000101100101
SHIFT 23 = 100100111100000110111001100011101011111101101000100001011001010
SHIFT 24 = 001001111000001101110011000111010111111011010001000010110010101
SHIFT 25 = 010011110000011011100110001110101111110110100010000101100101010
SHIFT 26 = 100111100000110111001100011101011111101101000100001011001010100
SHIFT 27 = 001111000001101110011000111010111111011010001000010110010101001
SHIFT 28 = 011110000011011100110001110101111110110100010000101100101010010
SHIFT 29 = 111100000110111001100011101011111101101000100001011001010100100
SHIFT 30 = 111000001101110011000111010111111011010001000010110010101001001
SHIFT 31 = 110000011011100110001110101111110110100010000101100101010010011
SHIFT 32 = 100000110111001100011101011111101101000100001011001010100100111
```

```
SHIFT 33 = 0000011011100110001110101111110110100010000101100101010010011111
SHIFT 34 = 0000110111001100011101011111101101000100001011001010100100111110
SHIFT 35 = 0001101110011000111010111111011010001000010110010101001001111100
SHIFT 36 = 0011011100110001110101111110110100010000101100101010010011111000
SHIFT 37 = 0110111001100011101011111101101000100001011001010100100111110000
SHIFT 38 = 1101110011000111010111111011010001000010110010101001001111100000
SHIFT 39 = 1011100110001110101111110110100010000101100101010010011111000001
SHIFT 40 = 0111001100011101011111101101000100001011001010100100111110000011
SHIFT 41 = 1110011000111010111111011010001000010110010101001001111100000110
SHIFT 42 = 1100110001110101111110110100010000101100101010010011111000001101
SHIFT 43 = 1001100011101011111101101000100001011001010100100111110000011011
SHIFT 44 = 0011000111010111111011010001000010110010101001001111100000110111
SHIFT 45 = 0110001110101111110110100010000101100101010010011111000001101110
SHIFT 46 = 1100011101011111101101000100001011001010100100111110000011011100
SHIFT 47 = 1000111010111111011010001000010110010101001001111100000110111001
SHIFT 48 = 0001110101111110110100010000101100101010010011111000001101110011
SHIFT 49 = 0011101011111101101000100001011001010100100111110000011011100110
SHIFT 50 = 0111010111111011010001000010110010101001001111100000110111001100
SHIFT 51 = 1110101111110110100010000101100101010010011111000001101110011000
SHIFT 52 = 1101011111101101000100001011001010100100111110000011011100110001
SHIFT 53 = 1010111111011010001000010110010101001001111100000110111001100011
SHIFT 54 = 0101111110110100010000101100101010010011111000001101110011000111
SHIFT 55 = 1011111101101000100001011001010100100111110000011011100110001110
SHIFT 56 = 0111111011010001000010110010101001001111100000110111001100011101
SHIFT 57 = 1111110110100010000101100101010010011111000001101110011000111010
SHIFT 58 = 1111101101000100001011001010100100111110000011011100110001110101
SHIFT 59 = 1111011010001000010110010101001001111100000110111001100011101011
SHIFT 60 = 1110110100010000101100101010010011111000001101110011000111010111
SHIFT 61 = 1101101000100001011001010100100111110000011011100110001110101111
SHIFT 62 = 1011010001000010110010101001001111100000110111001100011101011111
```

```
10 REM
20 CLS:PRINT
30 REM                              PROGRAM "GOLDGEN"
40 PRINT"                      GOLD CODE GENERATOR PROGRAM"
50 PRINT"             MAXIMAL LENGTH SEQUENCE MOD-2 ADDITION PROGRAM"
60 PRINT
70 REM     THIS PROGRAM IS DESIGNED TO EMULATE A DUAL SHIFT REGISTER
80 REM   CONFIGURATION OF A GOLD CODE SEQUENCE GENERATOR. TWO MAXIMAL LENGTH
90 REM   SEQUENCES, GENERATED BY THE "POLYSEQ" PROGRAM AND STORED ON DISK,
100 REM  ARE MOD-2 ADDED CHIP BY CHIP. THE RESULTING SEQUENCE IS A GOLD CODE
110 REM  SEQUENCE.
120 REM
130 REM     THE PROGRAM WILL PROMPT THE OPERATOR FIRST FOR THE NON-SHIFT
140 REM  SEQUENCE, AND THEN THE SHIFT SEQUENCE. THE SEQUENCE LENGTH AND THE
150 REM  NUMBER OF GOLD CODE SEQUENCES TO GENERATE IS ALSO INPUT BY THE
160 REM  OPERATOR. IT IS ASSUMED THAT THE FIRST SEQUENCE STORED IN EACH
170 REM  MAXIMAL LENGTH SEQUENCE FILE IS THE CHARACTERISTIC PHASE SEQUENCE.
180 REM
190 REM
200 REM     BEGIN BY PROMPTING THE OPERATOR FOR THE INPUT FILE NAMES.
210 REM
220 INPUT"ENTER THE FILE NAME OF THE NON-SHIFT SEQUENCE";NSHSEQ$
230 PRINT
240 INPUT"ENTER THE FILE NAME OF THE SHIFTING SEQUENCE";SHSEQ$
250 PRINT
260 INPUT"ENTER THE SEQUENCE LENGTH";SLENGTH
270 PRINT
280 INPUT"ENTER THE NUMBER OF SEQUENCES TO ADD";MSEQ
290 REM
300 DIM PBUFFER$(SLENGTH)
310 DIM GOLDOUT(SLENGTH),GOLDBUFF$(MSEQ)
```

```
320 REM
330 REM  INPUT THE CHARACTERISTIC PHASE OF THE NON-SHIFT SEQUENCE.
340 REM
350 OPEN "D",1,NSHSEQ$,SLENGTH
360 REM
370 FIELD 1,SLENGTH AS NOSHIFTSEQ$
380 GET 1
390 NOSHIFT$ = NOSHIFTSEQ$
400 CLOSE
410 REM
420 REM INPUT ALL PHASE SHIFT SEQUENCES FOR THE SHIFT SEQUENCE.
430 REM
440 OPEN "D",1,SHSEQ$,SLENGTH
450 REM
460 FIELD 1,SLENGTH AS P2SEQUENCE$
470 FOR I = 1 TO MSEQ
480    GET 1,I
490    PBUFFER$(I) = P2SEQUENCE$
500 NEXT I
510 CLOSE
520 REM
530 REM  NOW OUTPUT THE FIRST TWO GOLD CODE SEQUENCES
540 REM
550 PRINT"GOLD CODE      = ";1
560 PRINT"SHIFT SEQ      = ";PBUFFER$(1)
570 PRINT"NO-SHIFT SEQ   = ";STRING$(SLENGTH,"0")
580 PRINT
590 PRINT"GOLD SEQUENCE = ";PBUFFER$(1)
600 PRINT
610 PRINT"GOLD CODE      = ";2
620 PRINT"SHIFT SEQ      = ";STRING$(SLENGTH,"0")
630 PRINT"NO-SHIFT SEQ   = ";NOSHIFT$
640 PRINT
650 PRINT"GOLD SEQUENCE = ";NOSHIFT$
660 PRINT
670 REM
```

```
680 REM
690 REM NOW BEGIN THE ADDITION PROCESS
700 REM
710 GOLDCODE$ = STRING$(SLENGTH,"0")
720 FOR M = 1 TO MSEQ
730     FOR I = 1 TO SLENGTH
740   GOLDOUT(I) = ASC(MID$(NOSHIFT$,I,1)) + ASC(MID$(PBUFFER$(M),I,1))-96
750         IF GOLDOUT(I) = 2 THEN GOLDOUT(I) = 0
760     NEXT I
770 REM
780     FOR I = 1 TO SLENGTH
790         IF GOLDOUT(I) = 1 THEN MID$(GOLDCODE$,I,1) = "1":GOTO 810
800         MID$(GOLDCODE$,I,1) = "0"
810     NEXT I
820 REM
830     GOLDBUFF$(M) = GOLDCODE$
840         PRINT
850         PRINT"GOLD CODE      = ";M+2
860         PRINT"SHIFT SEQ      = ";PBUFFER$(M)
870         PRINT"NO-SHIFT SEQ   = ";NOSHIFT$
880         PRINT
890         PRINT"GOLD SEQUENCE = ";GOLDBUFF$(M)
900 NEXT M
910 REM
920 PRINT
930 PRINT"SELECT OUTPUT DESTINATION OF THE SEQUENCES:"
940 PRINT
950 PRINT"            1 = OUTPUT TO THE PRINTER"
960 PRINT"            2 = OUTPUT TO DISK"
970 PRINT"            3 = QUIT"
980 PRINT
990 INPUT CHOICE
1000 IF CHOICE = 1 THEN GOSUB 1040:GOTO 920
1010 IF CHOICE = 2 THEN GOSUB 1440:GOTO 920
1020 STOP
1030 REM
```

```
1040 REM
1050 REM THE FOLLOWING SEGMENT PRINTS THE CONTENTS OF GOLDBUFF$ TO THE PRINTER
1060 REM
1070 PRINT"ENTER PAGE HEADER":INPUT HEADER$
1080 LPRINT
1090 LPRINT HEADER$
1100 LPRINT
1110 LPRINT
1120 LPRINT
1130 M=1:ZSTRING$ = STRING$(63,"0")
1140   LPRINT"GOLD CODE      = ";M
1150   LPRINT
1160   LPRINT"SHIFT SEQ      = ";PBUFFER$(1)
1170   LPRINT"NO-SHIFT SEQ   = ";ZSTRING$
1180   LPRINT
1190   LPRINT"GOLD SEQUENCE = ";PBUFFER$(1)
1200   LPRINT
1210    LPRINT
1220 REM
1230   LPRINT"GOLD CODE      = ";M+1
1240    LPRINT
1250   LPRINT"SHIFT SEQ      = ";ZSTRING$
1260   LPRINT"NO-SHIFT SEQ   = ";NOSHIFT$
1270   LPRINT
1280   LPRINT"GOLD SEQUENCE = ";NOSHIFT$
1290   LPRINT
1300    LPRINT
1310 FOR M = 1 TO MSEQ
1320    LPRINT"GOLD CODE      = ";M+2
1330    LPRINT
1340    LPRINT"SHIFT SEQ      = ";PBUFFER$(M)
1350    LPRINT"NO-SHIFT SEQ   = ";NOSHIFT$
1360    LPRINT
1370    LPRINT"GOLD SEQUENCE = ";GOLDBUFF$(M)
1380    LPRINT
1390    LPRINT
```

```
1400 REM
1410 NEXT M
1420 RETURN
1430 REM
1440 REM
1450 REM THE FOLLOWING SEGMENT PROMPTS THE OPERATOR FOR A FILE NAME AND THEN
1460 REM  WRITE THE CONTENTS OF THE GOLDBUFF$ ARRAY ONTO DISK.
1470 REM
1480 PRINT
1490 INPUT"        ENTER A FILE NAME";GOLDFILE$
1500 REM
1510 OPEN "D",1,GOLDFILE$,SLENGTH
1520 FIELD 1,SLENGTH AS CODEBUFFER$
1530 REM
1540 REM     FIRST INSERT THE TWO MAXIMAL LENGTH SEQUENCES INTO THE FILE AS
1550 REM     GOLD CODES 1 AND 2.
1560 REM
1570    LSET CODEBUFFER$ = PBUFFER$(1)
1580    PUT 1
1590    LSET CODEBUFFER$ = NOSHIFT$
1600    PUT 1
1610 REM
1620 REM     NOW WRITE THE REMAINING GOLD CODES INTO THE FILE
1630 REM
1640 REM
1650 FOR M = 1 TO MSEQ
1660    LSET CODEBUFFER$ = GOLDBUFF$(M)
1670    PUT 1
1680 NEXT M
1690 CLOSE
1700 RETURN
```

GOLD CODE GENERATOR OUTPUT SEQUENCES

```
GOLD CODE      =  1

SHIFT SEQ      = 011111101010110011011101101001001110001011110010100011000010000
NO-SHIFT SEQ   = 000000000000000000000000000000000000000000000000000000000000000

GOLD SEQUENCE  = 011111101010110011011101101001001110001011110010100011000010000

GOLD CODE      =  2

SHIFT SEQ      = 000000000000000000000000000000000000000000000000000000000000000
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 011010001000010110010101001001111000001101110011000111010111111

GOLD CODE      =  3

SHIFT SEQ      = 011111101010110011011101101001001110001011110010100011000010000
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 000101100010100101001000100000110110000110000001100100010101111

GOLD CODE      =  4

SHIFT SEQ      = 111111010101100110111011010010011100010111100101000110000100000
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 100101011101110000101110011011100100011010010110000001010011111
```

```
GOLD CODE     =  5

SHIFT SEQ     = 111110101011001101110110100100111000101111001010001100001000001
NO-SHIFT SEQ  = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 100100100011011011100011101101000000010001011100100101101111110

GOLD CODE     =  6

SHIFT SEQ     = 111101010110011011101101001001110001011110010100011000010000011
NO-SHIFT SEQ  = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 100111011110001101111000000000001001010011100111011111000111100

GOLD CODE     =  7

SHIFT SEQ     = 111010101100110111011010010011100010111100101000110000100000111
NO-SHIFT SEQ  = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 100000100100100001001111011010011010110001011011110111110111000

GOLD CODE     =  8

SHIFT SEQ     = 110101011001101110110100100111000101111001010001100001000001111
NO-SHIFT SEQ  = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 101111010001111000100001101110111101110100100010100110010110000

GOLD CODE     =  9

SHIFT SEQ     = 101010110011011101101001001110001011110010100011000010000011111
NO-SHIFT SEQ  = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 110000111011001011111100000111110011111111010000000101010100000
```

GOLD CODE = 10

SHIFT SEQ = 010101100110111011010010011100010111100101000110000100000111111
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 001111101110101101000111010101101111101000110101000011010000000

GOLD CODE = 11

SHIFT SEQ = 101011001101110110100100111000101111001010001100001000001111110
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 110001000101100000110001110001010111000111111110011110110000001

GOLD CODE = 12

SHIFT SEQ = 010110011011101101001001110001011110010100011000010000011111101
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 001100010011111011011100111000100110011001101011010111001000010

GOLD CODE = 13

SHIFT SEQ = 101100110111011010010011100010111100101000110000100000111111010
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 110110111111001100000110101011000100100101000011100111101000101

GOLD CODE = 14

SHIFT SEQ = 011001101110110100100111000101111001010001100001000001111110101
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 000011100110100010110010001100000001011100010010000110101001010

```
GOLD CODE     =  15

SHIFT SEQ     = 110011011101101001001110001011110010100011000010000011111101010
NO-SHIFT SEQ  = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 101001010101111111011011000010001010101110110001000100101010101

GOLD CODE     =  16

SHIFT SEQ     = 100110111011010010011100010111100101000110000100000111111010101
NO-SHIFT SEQ  = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 111100110011000100001001011110011101001011110111000000101101010

GOLD CODE     =  17

SHIFT SEQ     = 001101110110100100111000101111001010001100001000001111110101011
NO-SHIFT SEQ  = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 010111111110110010101101100110110010000001111011001000100010100

GOLD CODE     =  18

SHIFT SEQ     = 011011101101001001110001011110010100011000010000011111101010110
NO-SHIFT SEQ  = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 000001100101011111100100010111101100010101100011011000111101001

GOLD CODE     =  19

SHIFT SEQ     = 110111011010010011100010111100101000110000100000111111010101100
NO-SHIFT SEQ  = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 101101010010000101110111110101010000111101010011111000000010011
```

```
GOLD CODE     =  20

SHIFT SEQ     = 101110110100100111000101111001010001100001000001111110101011001
NO-SHIFT SEQ  = 0110100010000010110010101001001111000000110111001100011101011111

GOLD SEQUENCE = 110100111100110001010000110000101001101100110010111001111100110

GOLD CODE     =  21

SHIFT SEQ     = 011101101001001110001011110010100011000010000011111101010110011
NO-SHIFT SEQ  = 0110100010000010110010101001001111000000110111001100011101011111

GOLD SEQUENCE = 0001111000001011000001111011101101101100111111000001110100000001100

GOLD CODE     =  22

SHIFT SEQ     = 111011010010011100010111100101000110000100000111111010101100110
NO-SHIFT SEQ  = 0110100010000010110010101001001111000000110111001100011101011111

GOLD SEQUENCE = 100001011010001010000010101100111110001001110100111101111011001

GOLD CODE     =  23

SHIFT SEQ     = 110110100100111000101111001010001100001000001111110101011001101
NO-SHIFT SEQ  = 0110100010000010110010101001001111000000110111001100011101011111

GOLD SEQUENCE = 101100101100101110111010000011110100000101111100110010001110010

GOLD CODE     =  24

SHIFT SEQ     = 101101001001110001011110010100011000010000011111101010110011011
NO-SHIFT SEQ  = 0110100010000010110010101001001111000000110111001100011101011111

GOLD SEQUENCE = 110111000001100111001011011101100000011101101100101101100100100
```

```
GOLD CODE      =  25

SHIFT SEQ      = 011010010011100010111100101000110000100000111111010101100110111
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 000000011011110100101001100001001000101101001100010010110001000

GOLD CODE      =  26

SHIFT SEQ      = 110100100111000101111001010001100001000001111110101011001101110
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 101110101111010011101100011000011001001100001101101100011010001

GOLD CODE      =  27

SHIFT SEQ      = 101001001110001011110010100011000010000011111101010110011011101
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 110011000110011101100111101010111010001110001110010001001100010

GOLD CODE      =  28

SHIFT SEQ      = 010010011100010111100101000110000100000111111010101100110111011
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 001000010100000001110000001111111000010100010011010111000000100

GOLD CODE      =  29

SHIFT SEQ      = 100100111000101111001010001100001000001111110101011001101110110
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 111110110000111001011111000101110000000100001100111101110010010
```

```
GOLD CODE      =  30

SHIFT SEQ      = 001001110001011110010100011000010000011111101010110011011101101
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 010011111001001000000001010001101000010010011001110100001010010

GOLD CODE      =  31

SHIFT SEQ      = 010011100010111100101000110000100000111111010101100110111011010
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 001001101010101010111101111001011000110010100110100001101100101

GOLD CODE      =  32

SHIFT SEQ      = 100111000101111001010001100001000001111110101011001101110110100
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 111101001101101111000100101000111001110011011000001010100001011

GOLD CODE      =  33

SHIFT SEQ      = 001110001011110010100011000010000011111101010110011011101101001
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 010100000011100100110110001011110111100001001010111001110101110

GOLD CODE      =  34

SHIFT SEQ      = 011100010111100101000110000100000111111010101100110111011010010
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 000110011111110011010011001101111111110111011111110000001101101
```

GOLD CODE = 35

SHIFT SEQ = 111000101111001010001100001000001111110101011001101110110100100
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 100010100111011100011001000001110111111000101010101001100011011

GOLD CODE = 36

SHIFT SEQ = 110001011110010100011000010000011111101010110011011101101001001
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 101011010110000010001101011001100111100111000000011010111110110

GOLD CODE = 37

SHIFT SEQ = 100010111100101000110000100000111111010101100110111011010010011
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 111000110100111110100101101001000111011000010101111100000101100

GOLD CODE = 38

SHIFT SEQ = 000101111001010001100001000001111110101011001101110110100100111
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 011111110001000111110100001000000110100110111110110001110011000

GOLD CODE = 39

SHIFT SEQ = 001011110010100011000010000011111101010110011011101101001001110
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 010001111010110101010111001010000101011011101000101010011110001

GOLD CODE = 40

SHIFT SEQ = 010111100101000110000100000111111010101100110111011010010011100
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 001101101101010000010001001110000010100001000100011101000100011

GOLD CODE = 41

SHIFT SEQ = 101111001010001100001000001111110101011001101110110100100111000
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 110101000010011010011101000110001101010100011101110011110000111

GOLD CODE = 42

SHIFT SEQ = 011110010100011000010000011111101010110011011101101001001110001
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 000100011100001110000101010110010010111110101110101110011001110

GOLD CODE = 43

SHIFT SEQ = 111100101000110000100000111111010101100110111011010010011100010
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 100110100000100110110101110110101101101011001000010101001011101

GOLD CODE = 44

SHIFT SEQ = 111001010001100001000001111110101011001101110110100100111000101
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 100011011001110111010100110111010011000000001011000011101111010

```
GOLD CODE      =  45

SHIFT SEQ      = 110010100011000010000011111101010110011011101101001001110001011
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 101000101011010100010110110100101110010110011110001110100110100

GOLD CODE      =  46

SHIFT SEQ      = 100101000110000100000111111010101100110111011010010011100010111
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 111111001110010010010010110011010100111010101001010100110101000

GOLD CODE      =  47

SHIFT SEQ      = 001010001100001000001111110101011001101110110100100111000101111
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 010000000100011110011010111100100001100011000111100000010010000

GOLD CODE      =  48

SHIFT SEQ      = 010100011000010000011111101010110011011101101001001110001011110
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 001110010000000110001010100011001011010000011010001001011100001

GOLD CODE      =  49

SHIFT SEQ      = 101000110000100000111111010101100110111011010010011100010111100
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 110010111000110110101010011100011110110110100001011011000000011
```

```
GOLD CODE      =  50

SHIFT SEQ      = 010001100000100000111111010101100110111011010010011100010111 1001
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 001011101001010111101011100010110101111011010111111111111000110

GOLD CODE      =  51

SHIFT SEQ      = 100011000001000001111110101011001101110110100100111000101111 0010
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 111001001010010101101000011111100011100000111010110110001001101

GOLD CODE      =  52

SHIFT SEQ      = 000110000010000011111101010110011011101101001001110001011110 0101
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 011100001100010001101111100101001111010111100000100101101011010

GOLD CODE      =  53

SHIFT SEQ      = 001100000100000111111010101100110111011010010011100010111 1001010
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 010110000000011001100000010000010110111001010100000010101110101

GOLD CODE      =  54

SHIFT SEQ      = 011000001000001111110101011001101110110100100111000101111 0010100
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 000010011000001001111111111010100101100100111101001100100101011
```

GOLD CODE = 55

SHIFT SEQ = 110000100000111111010101100110111011010010011100010111100101000
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 101010101000101001000000101111000011011111101111010000110010111

GOLD CODE = 56

SHIFT SEQ = 100001000001111110101011001101110110100100111000101111001010001
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 111011001001101000111110000100001110101001001011101000011101110

GOLD CODE = 57

SHIFT SEQ = 000010000011111101010110011011101101001001110001011110010100011
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 011000001011101011000011010010010101000100000010011001000011100

GOLD CODE = 58

SHIFT SEQ = 000100000111111010101100110111011010010011100010111100101000110
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 011110001111011001110011111101000100111100100011110111111111001

GOLD CODE = 59

SHIFT SEQ = 001000001111110101011001101110110100100111000101111001010001100
NO-SHIFT SEQ = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 010010000111100011001100100111001100101010110110111110000110011

```
GOLD CODE      =  60

SHIFT SEQ      = 010000011111101010110011011101101001001110001011110010100011000
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 001010010111111100100110010100010001000011111000110101110100111

GOLD CODE      =  61

SHIFT SEQ      = 100000111111010101100110111011010010011100010111100101000110000
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 111010110111000011110011110010101010010001100100100010010001111

GOLD CODE      =  62

SHIFT SEQ      = 000001111110101011001101110110100100111000101111001010001100001
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 011011110110111101011000111111011100110101011100001101011011110

GOLD CODE      =  63

SHIFT SEQ      = 000011111101010110011011101101001001110001011110010100011000010
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 011001110101000000001110100100110001111100101101010011001111101

GOLD CODE      =  64

SHIFT SEQ      = 000111111010101100110111011010010011100010111100101000110000100
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE = 011101110010111010100010010011101011101111001111101111100111011
```

```
GOLD CODE      =  65

SHIFT SEQ      = 001111110101011001101110110100100111000101111001010001100001000
NO-SHIFT SEQ   = 011010001000010110010101001001111000001101110011000111010111111

GOLD SEQUENCE  = 010101111101001111111011111101011111001000001010010110110110111
```

```
10 REM
20 CLS:PRTFLAG=0
30 REM                              PROGRAM "AUTOCOR"
40 PRINT
50 PRINT"          CODE SEQUENCE AUTOCORRELATION CALCULATION PROGRAM"
60 REM
70 REM     THIS PROGRAM IS DESIGNED TO COMPUTE THE AUTOCORRELATION VALUES
80 REM   FOR ANY TYPE OF CODE SEQUENCE. IT IS ASSUMED THAT THE CODE SEQUENCES
90 REM   TO BE CORRELATED ARE STORED IN A DISK FILE. THE OPERATOR IS PROMPTED
100 REM  FOR THE FILE NAME, THE LENGTH OF THE SEQUENCE, THE NUMBER OF SEQUENCES
110 REM  TO READ FROM THE FILE, AND THE NUMBER OF AUTOCORRELATION VALUES
120 REM  TO COMPUTE. OUTPUT IS ROUTED TO EITHER THE TERMINAL OR TO A PRINTER.
130 REM
140 PRINT
150 INPUT"ENTER THE FILE NAME OF THE SEQUENCE TO USE";SHSEQ$
160 PRINT
170 INPUT"ENTER THE SEQUENCE LENGTH";SLENGTH
180 PRINT
190 INPUT"ENTER THE NUMBER OF SEQUENCES TO CORRELATE";MSEQ
200 PRINT
210 INPUT"STARTING WITH SEQUENCE ";STSEQ
220 PRINT
230 INPUT"ENTER THE NUMBER OF SHIFTS PER SEQUENCE TO CALCULATE";NSHIFT
240 REM
250 PRINT
260 INPUT"OUTPUT TO PRINTER  (Y/N)";PCHOICE$
270 IF PCHOICE$ = "Y" THEN PRTFLAG=1
280 REM
290 DIM PBUFFER$(MSEQ+STSEQ)
300 REM
310 REM     FIRST READ IN THE SEQUENCE AND ITS PHASE SHIFTED REPLICAS.
320 REM
330 OPEN "D",1,SHSEQ$,SLENGTH
340 REM
```

```
350 FIELD 1,SLENGTH AS P2SEQUENCE$
360    FOR I = 1 TO MSEQ+STSEQ
370       GET 1,I
380       PBUFFER$(I) = P2SEQUENCE$
390    NEXT I
400 CLOSE
410 REM
420 IF PRTFLAG=1 THEN INPUT"ENTER PRINT HEADER"; HEADER$:LPRINT HEADER$:LPRINT
430 REM
440 REM NOW BEGIN THE CORRELATION PROCESS
450 FOR M = STSEQ TO MSEQ+STSEQ-1
460 IF PRTFLAG=1 THEN LPRINT:LPRINT"SEQUENCE ";M;"=",PBUFFER$(M):LPRINT:GOTO 480
470    PRINT:PRINT"SEQUENCE ";M;"=",PBUFFER$(M):PRINT
480    AUTOCORR$ = PBUFFER$(M)
490 REM
500     FOR K = 1 TO NSHIFT
510      AGREE=0:DISAGREE=0
520       FOR I = 1 TO SLENGTH
530    IF MID$(PBUFFER$(M),I,1)=MID$(AUTOCORR$,I,1) THEN AGREE=AGREE+1:GOTO 550
540           DISAGREE = DISAGREE + 1
550       NEXT I
560 REM
570       IF PRTFLAG=1 THEN GOTO 600
580       PRINT"           SHIFT =";K-1,"AUTOCORRELATION = ";AGREE-DISAGREE
590       GOTO 620
600 REM
610       LPRINT"                  SHIFT =";K-1,"AUTOCORRELATION = ";AGREE-DISAGREE
620 REM  NOW SHIFT THE AUTOCORRELATION SEQUENCE
630       AUTOSAVE$ = MID$(AUTOCORR$,1,1)
640       FOR I = 1 TO SLENGTH-1
650         MID$(AUTOCORR$,I,1) = MID$(AUTOCORR$,I+1,1)
660      NEXT I
670       MID$(AUTOCORR$,SLENGTH,1) = AUTOSAVE$
680     NEXT K
690 NEXT M
700 STOP
```

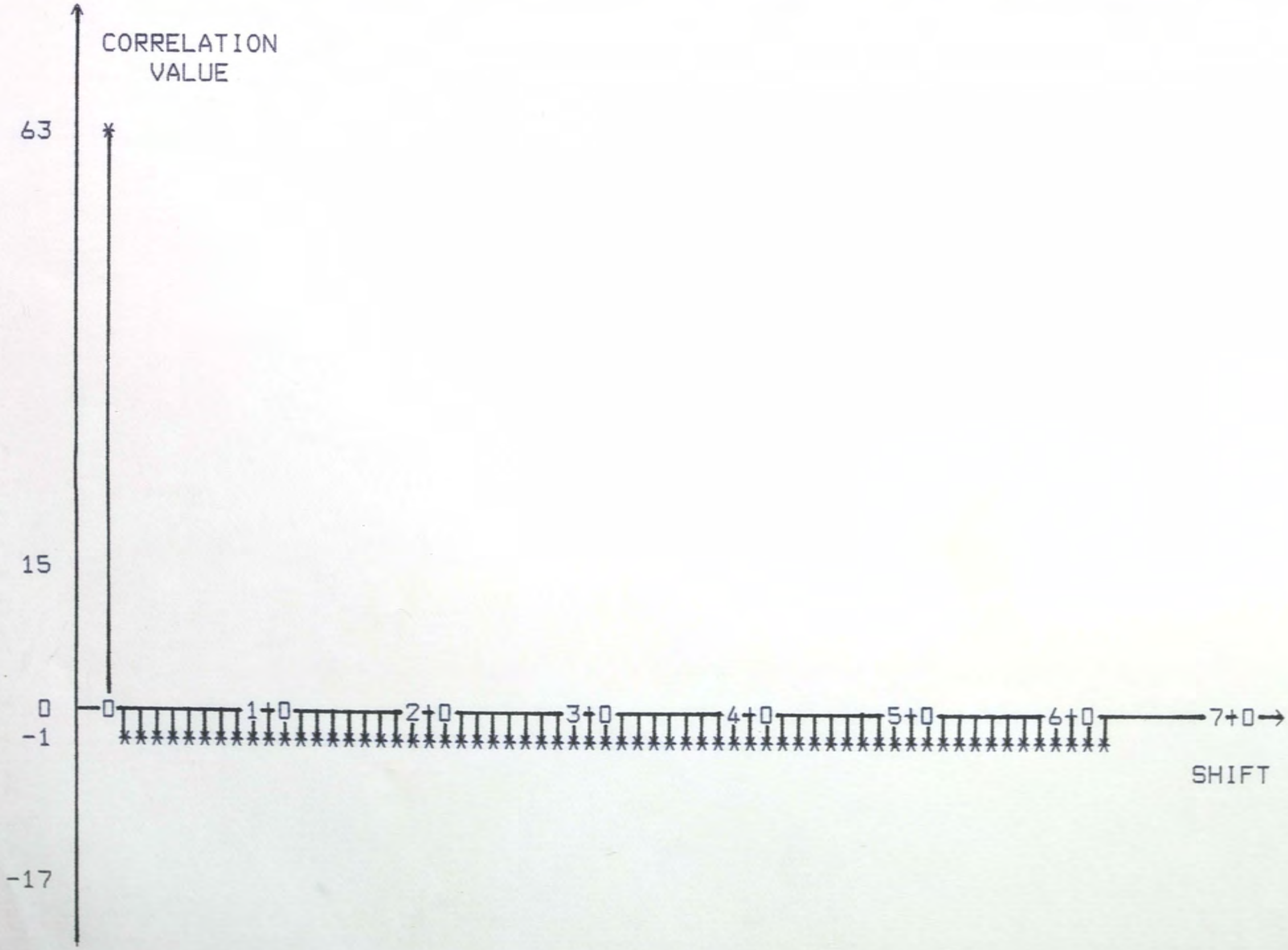
AUTOCORRELATION CALCULATIONS FOR THE P1(X) MAXIMAL LENGTH SEQUENCE

SEQUENCE 1 = 011111101010110011011101101001001110001011110010100011000010000

SHIFT = 0	AUTOCORRELATION = 63
SHIFT = 1	AUTOCORRELATION = -1
SHIFT = 2	AUTOCORRELATION = -1
SHIFT = 3	AUTOCORRELATION = -1
SHIFT = 4	AUTOCORRELATION = -1
SHIFT = 5	AUTOCORRELATION = -1
SHIFT = 6	AUTOCORRELATION = -1
SHIFT = 7	AUTOCORRELATION = -1
SHIFT = 8	AUTOCORRELATION = -1
SHIFT = 9	AUTOCORRELATION = -1
SHIFT = 10	AUTOCORRELATION = -1
SHIFT = 11	AUTOCORRELATION = -1
SHIFT = 12	AUTOCORRELATION = -1
SHIFT = 13	AUTOCORRELATION = -1
SHIFT = 14	AUTOCORRELATION = -1
SHIFT = 15	AUTOCORRELATION = -1
SHIFT = 16	AUTOCORRELATION = -1
SHIFT = 17	AUTOCORRELATION = -1
SHIFT = 18	AUTOCORRELATION = -1
SHIFT = 19	AUTOCORRELATION = -1
SHIFT = 20	AUTOCORRELATION = -1
SHIFT = 21	AUTOCORRELATION = -1
SHIFT = 22	AUTOCORRELATION = -1
SHIFT = 23	AUTOCORRELATION = -1
SHIFT = 24	AUTOCORRELATION = -1
SHIFT = 25	AUTOCORRELATION = -1
SHIFT = 26	AUTOCORRELATION = -1
SHIFT = 27	AUTOCORRELATION = -1
SHIFT = 28	AUTOCORRELATION = -1
SHIFT = 29	AUTOCORRELATION = -1
SHIFT = 30	AUTOCORRELATION = -1

```
SHIFT = 31      AUTOCORRELATION = -1
SHIFT = 32      AUTOCORRELATION = -1
SHIFT = 33      AUTOCORRELATION = -1
SHIFT = 34      AUTOCORRELATION = -1
SHIFT = 35      AUTOCORRELATION = -1
SHIFT = 36      AUTOCORRELATION = -1
SHIFT = 37      AUTOCORRELATION = -1
SHIFT = 38      AUTOCORRELATION = -1
SHIFT = 39      AUTOCORRELATION = -1
SHIFT = 40      AUTOCORRELATION = -1
SHIFT = 41      AUTOCORRELATION = -1
SHIFT = 42      AUTOCORRELATION = -1
SHIFT = 43      AUTOCORRELATION = -1
SHIFT = 44      AUTOCORRELATION = -1
SHIFT = 45      AUTOCORRELATION = -1
SHIFT = 46      AUTOCORRELATION = -1
SHIFT = 47      AUTOCORRELATION = -1
SHIFT = 48      AUTOCORRELATION = -1
SHIFT = 49      AUTOCORRELATION = -1
SHIFT = 50      AUTOCORRELATION = -1
SHIFT = 51      AUTOCORRELATION = -1
SHIFT = 52      AUTOCORRELATION = -1
SHIFT = 53      AUTOCORRELATION = -1
SHIFT = 54      AUTOCORRELATION = -1
SHIFT = 55      AUTOCORRELATION = -1
SHIFT = 56      AUTOCORRELATION = -1
SHIFT = 57      AUTOCORRELATION = -1
SHIFT = 58      AUTOCORRELATION = -1
SHIFT = 59      AUTOCORRELATION = -1
SHIFT = 60      AUTOCORRELATION = -1
SHIFT = 61      AUTOCORRELATION = -1
SHIFT = 62      AUTOCORRELATION = -1
```

AUTOCORRELATION PLOT - MAXIMAL LENGTH SEQUENCE FROM THE P1(X) GENERATOR



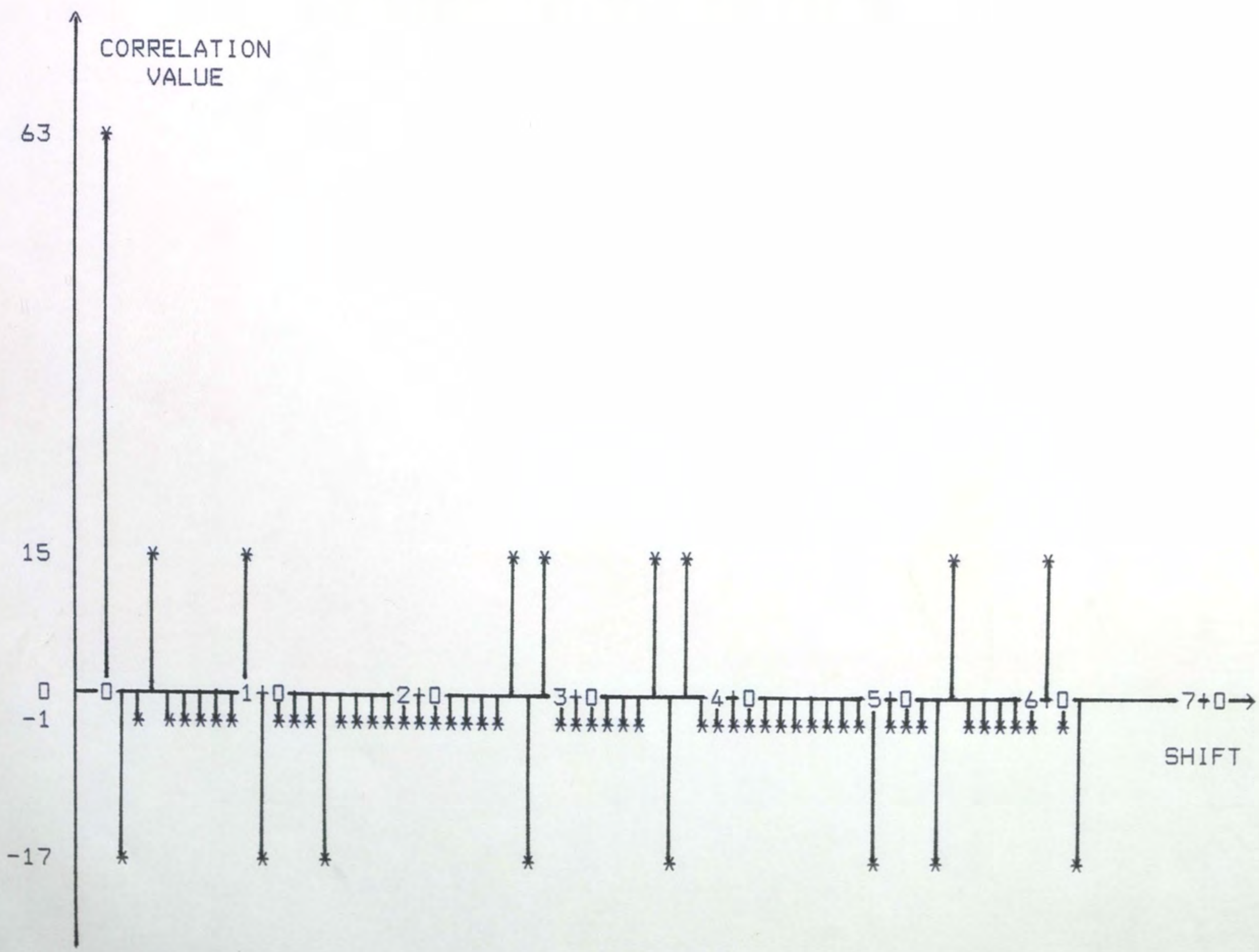
AUTOCORRELATION CALCULATIONS FOR GOLD CODE 43

SEQUENCE 43 = 100110100000100110110101110110101101101011001000010101001011101

```
SHIFT = 0       AUTOCORRELATION =  63
SHIFT = 1       AUTOCORRELATION = -17
SHIFT = 2       AUTOCORRELATION = -1
SHIFT = 3       AUTOCORRELATION =  15
SHIFT = 4       AUTOCORRELATION = -1
SHIFT = 5       AUTOCORRELATION = -1
SHIFT = 6       AUTOCORRELATION = -1
SHIFT = 7       AUTOCORRELATION = -1
SHIFT = 8       AUTOCORRELATION = -1
SHIFT = 9       AUTOCORRELATION =  15
SHIFT = 10      AUTOCORRELATION = -17
SHIFT = 11      AUTOCORRELATION = -1
SHIFT = 12      AUTOCORRELATION = -1
SHIFT = 13      AUTOCORRELATION = -1
SHIFT = 14      AUTOCORRELATION = -17
SHIFT = 15      AUTOCORRELATION = -1
SHIFT = 16      AUTOCORRELATION = -1
SHIFT = 17      AUTOCORRELATION = -1
SHIFT = 18      AUTOCORRELATION = -1
SHIFT = 19      AUTOCORRELATION = -1
SHIFT = 20      AUTOCORRELATION = -1
SHIFT = 21      AUTOCORRELATION = -1
SHIFT = 22      AUTOCORRELATION = -1
SHIFT = 23      AUTOCORRELATION = -1
SHIFT = 24      AUTOCORRELATION = -1
SHIFT = 25      AUTOCORRELATION = -1
SHIFT = 26      AUTOCORRELATION =  15
SHIFT = 27      AUTOCORRELATION = -17
SHIFT = 28      AUTOCORRELATION =  15
SHIFT = 29      AUTOCORRELATION = -1
SHIFT = 30      AUTOCORRELATION = -1
```

```
SHIFT = 31      AUTOCORRELATION = -1
SHIFT = 32      AUTOCORRELATION = -1
SHIFT = 33      AUTOCORRELATION = -1
SHIFT = 34      AUTOCORRELATION = -1
SHIFT = 35      AUTOCORRELATION =  15
SHIFT = 36      AUTOCORRELATION = -17
SHIFT = 37      AUTOCORRELATION =  15
SHIFT = 38      AUTOCORRELATION = -1
SHIFT = 39      AUTOCORRELATION = -1
SHIFT = 40      AUTOCORRELATION = -1
SHIFT = 41      AUTOCORRELATION = -1
SHIFT = 42      AUTOCORRELATION = -1
SHIFT = 43      AUTOCORRELATION = -1
SHIFT = 44      AUTOCORRELATION = -1
SHIFT = 45      AUTOCORRELATION = -1
SHIFT = 46      AUTOCORRELATION = -1
SHIFT = 47      AUTOCORRELATION = -1
SHIFT = 48      AUTOCORRELATION = -1
SHIFT = 49      AUTOCORRELATION = -17
SHIFT = 50      AUTOCORRELATION = -1
SHIFT = 51      AUTOCORRELATION = -1
SHIFT = 52      AUTOCORRELATION = -1
SHIFT = 53      AUTOCORRELATION = -17
SHIFT = 54      AUTOCORRELATION =  15
SHIFT = 55      AUTOCORRELATION = -1
SHIFT = 56      AUTOCORRELATION = -1
SHIFT = 57      AUTOCORRELATION = -1
SHIFT = 58      AUTOCORRELATION = -1
SHIFT = 59      AUTOCORRELATION = -1
SHIFT = 60      AUTOCORRELATION =  15
SHIFT = 61      AUTOCORRELATION = -1
SHIFT = 62      AUTOCORRELATION = -17
```

AUTOCORRELATION PLOT - GOLD CODE 43



```
10 REM
20 CLS:PRTFLAG=0
30 REM                          PROGRAM "CROSSCOR"
40 PRINT
50 PRINT"           CODE SEQUENCE CROSS CORRELATION CALCULATION PROGRAM"
60 REM
70 REM    THIS PROGRAM IS DESIGNED TO COMPUTE THE CROSS-CORRELATION VALUES
80 REM    FOR ANY TYPE OF CODE SEQUENCES. IT IS ASSUMED THAT THE CODE SEQUENCES
90 REM    TO BE CORRELATED ARE STORED IN A DISK FILE. THE OPERATOR IS PROMPTED
100 REM   FOR THE FILE NAME, THE LENGTH OF THE SEQUENCE, THE ID NUMBER OF THE
110 REM   SEQUENCE TO TEST, THE ID NUMBER OF THE FIRST SEQUENCE TO
120 REM   CROSS-CORRELATE WITH THE TEST SEQUENCE, THE NUMBER OF SEQUENCES
130 REM   FOLLOWING THIS FIRST SEQUENCE TO CROSS-CORRELATE WITH THE TEST
140 REM   SEQUENCE, AND THE NUMBER OF CROSS-CORRELATION VALUES TO COMPUTE
150 REM   FOR EACH TEST.
160 REM
170 PRINT
180 INPUT"ENTER THE FILE NAME OF THE SEQUENCES TO USE";SHSEQ$
190 PRINT
200 INPUT"ENTER THE SEQUENCE LENGTH";SLENGTH
210 PRINT
220 INPUT"ENTER THE TEST SEQUENCE";TSEQ
230 PRINT
240 INPUT"ENTER THE FIRST SEQUENCE TO CORRELATE THE TEST SEQUENCE AGAINST";STSEQ
250 INPUT"ENTER THE NUMBER OF FOLLOWING SEQUENCES TO CORRELATE AGAINST";MSEQ
260 PRINT
270 INPUT"ENTER THE NUMBER OF SHIFTS PER SEQUENCE TO CALCULATE";NSHIFT
280 REM
290 PRINT
300 INPUT"OUTPUT TO PRINTER  (Y/N)";PCHOICE$
310 IF PCHOICE$ = "Y" THEN PRTFLAG=1
320 REM
330 IF TSEQ > STSEQ THEN BUFDIM = TSEQ+MSEQ:GOTO 350
340 BUFDIM = STSEQ+MSEQ
350 REM
360 DIM PBUFFER$(BUFDIM)
```

```
370 REM
380 REM      FIRST READ IN ALL THE REQUIRED SEQUENCES FROM DISK AND PLACE THEM
390 REM      INTO THE "PBUFFER" ARRAY.
400 REM
410 OPEN "D",1,SHSEQ$,SLENGTH
420 REM
430 FIELD 1,SLENGTH AS P2SEQUENCE$
440 FOR I = 1 TO BUFDIM
450    GET 1,I
460    PBUFFER$(I) = P2SEQUENCE$
470 NEXT I
480 CLOSE
490 REM
500 PRINT
510 IF PRTFLAG=1 THEN INPUT"ENTER PRINT HEADER";H$:LPRINT:LPRINT H$:LPRINT
520 REM
530 REM NOW BEGIN THE CORRELATION PROCESS
540 REM
550 FOR M = STSEQ TO STSEQ+MSEQ-1
560 IF PRTFLAG = 0 THEN GOTO 610
570 LPRINT
580 LPRINT"SEQUENCE ";TSEQ;"=",PBUFFER$(TSEQ)
590 LPRINT"SEQUENCE ";M;"=",PBUFFER$(M)
600 LPRINT:GOTO 670
610 REM
620 PRINT
630 PRINT"SEQUENCE ";TSEQ;"=",PBUFFER$(TSEQ)
640 PRINT"SEQUENCE ";M;"=",PBUFFER$(M)
650 PRINT
660 REM
670 CROSSCORR$ = PBUFFER$(M)
680 REM
```

```
690 FOR K = 1 TO NSHIFT
700  AGREE=0:DISAGREE=0
710       FOR I = 1 TO SLENGTH
720 IF MID$(PBUFFER$(TSEQ),I,1) = MID$(CROSSCORR$,I,1) THEN AGREE=AGREE+1:GOTO 740
730 DISAGREE = DISAGREE + 1
740 NEXT I
750 REM
760 IF PRTFLAG=1 THEN GOTO 790
770 PRINT"SHIFT";K-1;"=","CROSS CORRELATION =",AGREE-DISAGREE
780 GOTO 820
790 REM
800 LPRINT"                SHIFT =";K-1,"CROSS CORRELATION = ";AGREE-DISAGREE
810 REM
820 REM  NOW SHIFT THE CROSS-CORRELATING SEQUENCE
830 REM
840 CROSSSAVE$ = MID$(CROSSCORR$,1,1)
850 FOR I = 1 TO SLENGTH-1
860   MID$(CROSSCORR$,I,1) = MID$(CROSSCORR$,I+1,1)
870 NEXT I
880 MID$(CROSSCORR$,SLENGTH,1) = CROSSSAVE$
890 NEXT K
900 NEXT M
910 STOP
```

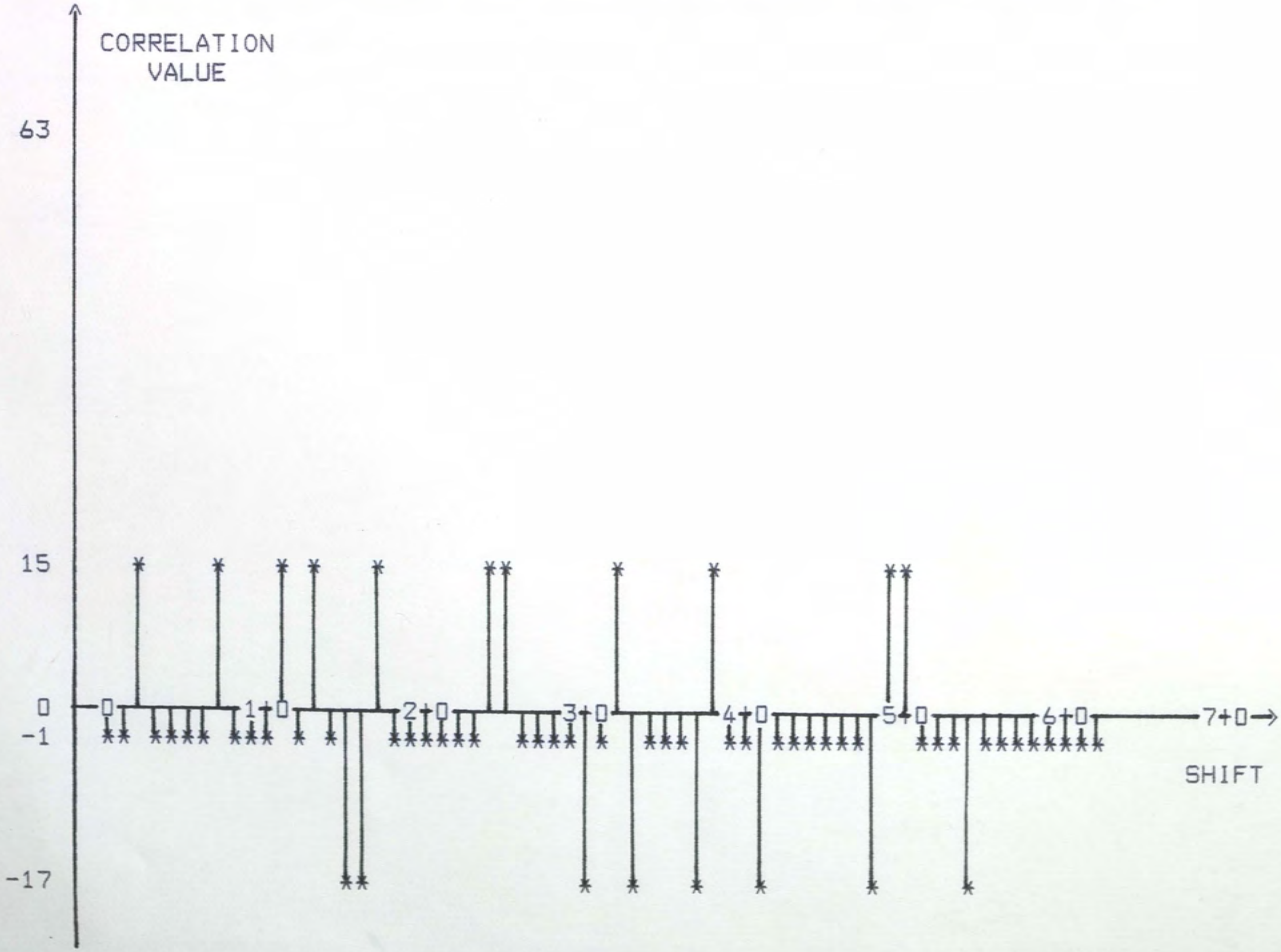
CROSS-CORRELATION CALCULATION BETWEEN GOLD CODES 13 AND 57

```
SEQUENCE  13 =  110110111111001100000110101011000100100101000011100111101000101
SEQUENCE  57 =  011000001011101011000011010010010101000100000010011001000011100

          SHIFT = 0        CROSS CORRELATION = -1
          SHIFT = 1        CROSS CORRELATION = -1
          SHIFT = 2        CROSS CORRELATION =  15
          SHIFT = 3        CROSS CORRELATION = -1
          SHIFT = 4        CROSS CORRELATION = -1
          SHIFT = 5        CROSS CORRELATION = -1
          SHIFT = 6        CROSS CORRELATION = -1
          SHIFT = 7        CROSS CORRELATION =  15
          SHIFT = 8        CROSS CORRELATION = -1
          SHIFT = 9        CROSS CORRELATION = -1
          SHIFT = 10       CROSS CORRELATION = -1
          SHIFT = 11       CROSS CORRELATION =  15
          SHIFT = 12       CROSS CORRELATION = -1
          SHIFT = 13       CROSS CORRELATION =  15
          SHIFT = 14       CROSS CORRELATION = -1
          SHIFT = 15       CROSS CORRELATION = -17
          SHIFT = 16       CROSS CORRELATION = -17
          SHIFT = 17       CROSS CORRELATION =  15
          SHIFT = 18       CROSS CORRELATION = -1
          SHIFT = 19       CROSS CORRELATION = -1
          SHIFT = 20       CROSS CORRELATION = -1
          SHIFT = 21       CROSS CORRELATION = -1
          SHIFT = 22       CROSS CORRELATION = -1
          SHIFT = 23       CROSS CORRELATION = -1
          SHIFT = 24       CROSS CORRELATION =  15
          SHIFT = 25       CROSS CORRELATION =  15
          SHIFT = 26       CROSS CORRELATION = -1
          SHIFT = 27       CROSS CORRELATION = -1
          SHIFT = 28       CROSS CORRELATION = -1
```

```
SHIFT = 29      CROSS CORRELATION = -1
SHIFT = 30      CROSS CORRELATION = -17
SHIFT = 31      CROSS CORRELATION = -1
SHIFT = 32      CROSS CORRELATION =  15
SHIFT = 33      CROSS CORRELATION = -17
SHIFT = 34      CROSS CORRELATION = -1
SHIFT = 35      CROSS CORRELATION = -1
SHIFT = 36      CROSS CORRELATION = -1
SHIFT = 37      CROSS CORRELATION = -17
SHIFT = 38      CROSS CORRELATION =  15
SHIFT = 39      CROSS CORRELATION = -1
SHIFT = 40      CROSS CORRELATION = -1
SHIFT = 41      CROSS CORRELATION = -17
SHIFT = 42      CROSS CORRELATION = -1
SHIFT = 43      CROSS CORRELATION = -1
SHIFT = 44      CROSS CORRELATION = -1
SHIFT = 45      CROSS CORRELATION = -1
SHIFT = 46      CROSS CORRELATION = -1
SHIFT = 47      CROSS CORRELATION = -1
SHIFT = 48      CROSS CORRELATION = -17
SHIFT = 49      CROSS CORRELATION =  15
SHIFT = 50      CROSS CORRELATION =  15
SHIFT = 51      CROSS CORRELATION = -1
SHIFT = 52      CROSS CORRELATION = -1
SHIFT = 53      CROSS CORRELATION = -1
SHIFT = 54      CROSS CORRELATION = -17
SHIFT = 55      CROSS CORRELATION = -1
SHIFT = 56      CROSS CORRELATION = -1
SHIFT = 57      CROSS CORRELATION = -1
SHIFT = 58      CROSS CORRELATION = -1
SHIFT = 59      CROSS CORRELATION = -1
SHIFT = 60      CROSS CORRELATION = -1
SHIFT = 61      CROSS CORRELATION = -1
SHIFT = 62      CROSS CORRELATION = -1
```

CROSS-CORRELATION PLOT - GOLD CODE 13 AGAINST GOLD CODE 57



```
10 REM
20 CLS:PRTFLAG=0
30 REM                              PROGRAM "CORRSUM"
40 PRINT
50 PRINT"          GOLD CODE CORRELATION CALCULATION SUMMARY PROGRAM"
60 REM
70 REM    THIS PROGRAM IS DESIGNED TO COMPUTE CROSS AND AUTOCORRELATION
80 REM    VALUES FOR A CONTIGUOUS SUBSET OF THE CODE SEQUENCES GENERATED
90 REM    AND STORED BY EITHER THE "POLYSEQ" OR "GOLDGEN" PROGRAMS.
100 REM
110 REM   THE TOTAL NUMBER OF OCCURENCES OF EACH CORRELATION VALUE IS
120 REM   TABULATED AND DISPLAYED FOR EACH CROSS OR AUTOCORRELATION
130 REM   CALCULATION SEQUENCE. THE OPERATOR IS PROMPTED FOR A FILE NAME,
140 REM   THE SEQUENCE LENGTH, THE STARTING SEQUENCE, AND THE NUMBER OF
150 REM   FOLLOWING SEQUENCES. OUTPUT CAN BE ROUTED TO EITHER THE TERMINAL
160 REM   OR TO A PRINTER.
170 REM
180 PRINT
190 INPUT"ENTER THE FILE NAME OF THE SEQUENCE TO USE";SHSEQ$
200 PRINT
210 INPUT"ENTER THE SEQUENCE LENGTH";SLENGTH
220 PRINT
230 INPUT"ENTER THE STARTING SEQUENCE NUMBER";STSEQ
240 PRINT
250 INPUT"ENTER THE NUMBER OF FOLLOWING SEQUENCES TO CORRELATE";MSEQ
260 REM
270 PRINT
280 INPUT"OUTPUT TO PRINTER  (Y/N)";PCHOICE$
290 IF PCHOICE$ = "Y" THEN PRTFLAG=1
300 REM
310 DIM PBUFFER$(MSEQ+STSEQ),VALARRAY(SLENGTH),ARVAL(SLENGTH)
320 REM
```

```
330 REM     BEGIN BY READING IN THE REQUIRED SEQUENCES FROM THE DISK.
340 REM
350 OPEN "D",1,SHSEQ$,SLENGTH
360 REM
370 FIELD 1,SLENGTH AS P2SEQUENCE$
380 FOR I = 1 TO MSEQ+STSEQ
390    GET 1,I
400    PBUFFER$(I) = P2SEQUENCE$
410 NEXT I
420 CLOSE
430 REM
440 REM NOW BEGIN THE CORRELATION PROCESS
450 REM
460 IF PRTFLAG=1 THEN INPUT"ENTER PRINT HEADER";H$:LPRINT:LPRINT H$:LPRINT:LPRINT
470 FOR L = STSEQ TO STSEQ+MSEQ
480 REM
490 FOR M = STSEQ TO STSEQ+MSEQ
500 IF PRTFLAG = 0 THEN GOTO 550
510 LPRINT
520 LPRINT"GOLDCODE ";L;"=",PBUFFER$(L)
530 LPRINT"GOLDCODE ";M;"=",PBUFFER$(M)
540 LPRINT:GOTO 610
550 REM
560 PRINT
570 PRINT"GOLD CODE ";L;"=",PBUFFER$(L)
580 PRINT"GOLD CODE ";M;"=",PBUFFER$(M)
590 PRINT
600 REM
610 AUTOCORR$ = PBUFFER$(M)
620 REM
630 VALNUM = 1
640 FOR I = 1 TO SLENGTH
650 VALARRAY(I) = 0:ARVAL(I)=0
660 NEXT I
```

```
670 REM
680 FOR K = 1 TO SLENGTH
690  AGREE=0:DISAGREE=0
700       FOR I = 1 TO SLENGTH
710 IF MID$(PBUFFER$(L),I,1) = MID$(AUTOCORR$,I,1) THEN AGREE=AGREE+1:GOTO 730
720 DISAGREE = DISAGREE + 1
730 NEXT I
740 REM
750 VALUE = AGREE-DISAGREE
760 REM
770 REM    CHECK FOR A NEW CORRELATION VALUE TO ADD TO THE TABLE
780 REM
790 FOR I = 1 TO VALNUM
800    IF VALARRAY(I) = VALUE THEN ARVAL(I) = ARVAL(I) + 1:GOTO 980
810 NEXT I
820 REM
830 REM IF HERE, WE HAVE A NEW VALUE, SO ADD IT TO THE TABLE
840 REM
850 VALNUM = VALNUM + 1
860 VALARRAY(VALNUM) = VALUE
870 ARVAL(VALNUM) = 1
880 GOTO 980
890 REM
900 IF PRTFLAG=1 THEN GOTO 930
910 PRINT"SHIFT";K-1;"=","CROSS CORRELATION =",AGREE-DISAGREE
920 GOTO 960
930 REM
940 LPRINT"                SHIFT =";K-1,"CROSS CORRELATION = ";AGREE-DISAGREE
950 REM
960 REM  NOW SHIFT THE  CORRELATION SEQUENCE FOR THE NEXT PASS
970 REM
980 AUTOSAVE$ = MID$(AUTOCORR$,1,1)
```

```
990 FOR I = 1 TO SLENGTH-1
1000   MID$(AUTOCORR$,I,1) = MID$(AUTOCORR$,I+1,1)
1010 NEXT I
1020 MID$(AUTOCORR$,SLENGTH,1) = AUTOSAVE$
1030 REM
1040 NEXT K
1050 REM
1060 REM    PRINT THE CORRELATION SUMMARY FOR THIS CODE PAIR
1070 REM
1080 IF PRTFLAG = 1 THEN GOTO 1190
1090 IF M <> L THEN GOTO 1120
1100 PRINT"           AUTO CORRELATION                    FREQUENCY"
1110 GOTO 1130
1120 PRINT"          CROSS CORRELATION                    FREQUENCY"
1130 PRINT"                VALUE                        OF OCCURENCE"
1140 PRINT
1150 FOR I = 2 TO VALNUM
1160 PRINT"      ",VALARRAY(I),"      ",ARVAL(I)
1170 NEXT I
1180 GOTO 1300
1190 REM
1200 IF M <> L THEN GOTO 1230
1210 LPRINT"          AUTO CORRELATION                    FREQUENCY"
1220 GOTO 1240
1230 LPRINT"         CROSS CORRELATION                    FREQUENCY"
1240 LPRINT"               VALUE                        OF OCCURENCE"
1250 LPRINT
1260 FOR I = 2 TO VALNUM
1270 LPRINT"      ",VALARRAY(I),"      ",ARVAL(I)
1280 NEXT I
1290 LPRINT
1300 NEXT M
1310 NEXT L
1320 STOP
```

GOLD CODE CORRELATION ANALYSIS SUMMARY - CODES 17 THROUGH 21

```
GOLDCODE  17 =  010111111110110010101101100110110010000001111011001000100010100
GOLDCODE  17 =  010111111110110010101101100110110010000001111011001000100010100
```

AUTO CORRELATION VALUE	FREQUENCY OF OCCURENCE
63	1
-1	46
15	8
-17	8

```
GOLDCODE  17 =  010111111110110010101101100110110010000001111011001000100010100
GOLDCODE  18 =  000001100101011111100100010111101100010101100011011000111101001
```

CROSS CORRELATION VALUE	FREQUENCY OF OCCURENCE
-1	47
15	10
-17	6

```
GOLDCODE  17 =  010111111110110010101101100110110010000001111011001000100010100
GOLDCODE  19 =  101101010010000101110111110101010000111101010011111000000010011
```

CROSS CORRELATION VALUE	FREQUENCY OF OCCURENCE
-1	43
-17	8
15	12

GOLDCODE 17 = 010111111110110010101101100110110010000001111011001000100010100
GOLDCODE 20 = 110100111100110001010000110000101001101100110010111001111100110

CROSS CORRELATION VALUE	FREQUENCY OF OCCURENCE
-1	43
-17	8
15	12

GOLDCODE 17 = 010111111110110010101101100110110010000001111011001000100010100
GOLDCODE 21 = 000111100001011000011110111011011011001111110000111010000001100

CROSS CORRELATION VALUE	FREQUENCY OF OCCURENCE
-1	51
15	8
-17	4

GOLDCODE 18 = 000001100101011111110010001011101100010101100011011000111101001
GOLDCODE 17 = 010111111110110010101101100110110010000001111011001000100010100

CROSS CORRELATION VALUE	FREQUENCY OF OCCURENCE
-1	47
-17	6
15	10

GOLDCODE 18 = 000001100101011111100100010111101100010101100011011000111101001
GOLDCODE 18 = 000001100101011111100100010111101100010101100011011000111101001

AUTO CORRELATION VALUE	FREQUENCY OF OCCURENCE
63	1
-1	54
-17	4
15	4

GOLDCODE 18 = 000001100101011111100100010111101100010101100011011000111101001
GOLDCODE 19 = 101101010010000101110111110101010000111101010011111000000010011

CROSS CORRELATION VALUE	FREQUENCY OF OCCURENCE
-1	39
-17	10
15	14

GOLDCODE 18 = 000001100101011111100100010111101100010101100011011000111101001
GOLDCODE 20 = 110100111100110001010000110000101001101100110010111001111100110

CROSS CORRELATION VALUE	FREQUENCY OF OCCURENCE
-1	51
-17	4
15	8

```
GOLDCODE  18 =  000001100101011111100100010111101100010101100011011000111101001
GOLDCODE  21 =  000111100001011000011110111011011011001111110000011101000001100

        CROSS CORRELATION                    FREQUENCY
              VALUE                        OF OCCURENCE

               -1                                51
                15                               8
               -17                               4


GOLDCODE  19 =  101101010010000101110111110101010000111101010011111000000010011
GOLDCODE  17 =  010111111110110010101101100110110010000001111011001000100010100

        CROSS CORRELATION                    FREQUENCY
              VALUE                        OF OCCURENCE

               -1                                43
                15                               12
               -17                               8


GOLDCODE  19 =  101101010010000101110111110101010000111101010011111000000010011
GOLDCODE  18 =  000001100101011111100100010111101100010101100011011000111101001

        CROSS CORRELATION                    FREQUENCY
              VALUE                        OF OCCURENCE

               -1                                39
               -17                               10
                15                               14
```

```
GOLDCODE  19 =  101101010010000101110111110101010000111101010011111000000010011
GOLDCODE  19 =  101101010010000101110111110101010000111101010011111000000010011

          AUTO CORRELATION                    FREQUENCY
                VALUE                       OF OCCURENCE

                 63                              1
                 -1                              38
                 15                              12
                -17                              12


GOLDCODE  19 =  101101010010000101110111110101010000111101010011111000000010011
GOLDCODE  20 =  110100111100110001010000110000101001101100110010111001111100110

         CROSS CORRELATION                    FREQUENCY
                VALUE                       OF OCCURENCE

                 -1                              51
                -17                              4
                 15                              8


GOLDCODE  19 =  101101010010000101110111110101010000111101010011111000000010011
GOLDCODE  21 =  000111100000101100001111011101101101100111111000011101000001100

         CROSS CORRELATION                    FREQUENCY
                VALUE                       OF OCCURENCE

                 -1                              51
                 15                              8
                -17                              4
```

```
GOLDCODE  20 =  110100111100110001010000110000101001101100110010111001111100110
GOLDCODE  17 =  010111111110110010101101100110110010000001111011001000100010100

        CROSS CORRELATION                   FREQUENCY
              VALUE                        OF OCCURENCE

               -1                               43
              -17                               8
               15                               12


GOLDCODE  20 =  110100111100110001010000110000101001101100110010111001111100110
GOLDCODE  18 =  000001100101011111100100010111101100010101100011011000111101001

        CROSS CORRELATION                   FREQUENCY
              VALUE                        OF OCCURENCE

               -1                               51
               15                               8
              -17                               4


GOLDCODE  20 =  110100111100110001010000110000101001101100110010111001111100110
GOLDCODE  19 =  101101010010000101110111110101010000111101010011111000000010011

        CROSS CORRELATION                   FREQUENCY
              VALUE                        OF OCCURENCE

               -1                               51
              -17                               4
               15                               8
```

```
GOLDCODE  20 =  110100111100110001010000110000101001101100110010111001111100110
GOLDCODE  20 =  110100111100110001010000110000101001101100110010111001111100110

          AUTO CORRELATION                        FREQUENCY
               VALUE                            OF OCCURENCE

                 63                                 1
                -1                                  46
                -17                                 8
                 15                                 8


GOLDCODE  20 =  110100111100110001010000110000101001101100110010111001111100110
GOLDCODE  21 =  000111100001011000011110111011011011001111110000111010000001100

         CROSS CORRELATION                        FREQUENCY
               VALUE                            OF OCCURENCE

                -1                                  47
                -17                                 6
                 15                                 10


GOLDCODE  21 =  000111100001011000011110111011011011001111110000111010000001100
GOLDCODE  17 =  010111111110110010101101100110110010000001111011001000100010100

         CROSS CORRELATION                        FREQUENCY
               VALUE                            OF OCCURENCE

                -1                                  51
                 15                                 8
                -17                                 4
```

```
GOLDCODE  21 =  0001111000001011000011110111011011011001111110000111010000001100
GOLDCODE  18 =  000001100101011111100100010111101100010101100011011000111101001

        CROSS CORRELATION                    FREQUENCY
              VALUE                         OF OCCURENCE

               -1                                51
                15                               8
               -17                               4


GOLDCODE  21 =  0001111000001011000011110111011011011001111110000111010000001100
GOLDCODE  19 =  1011010100100001011101111110101010000111101010011111100000010011

        CROSS CORRELATION                    FREQUENCY
              VALUE                         OF OCCURENCE

               -1                                51
                15                               8
               -17                               4


GOLDCODE  21 =  0001111000001011000011110111011011011001111110000111010000001100
GOLDCODE  20 =  110100111100110001010000110000101001101100110010111001111100110

        CROSS CORRELATION                    FREQUENCY
              VALUE                         OF OCCURENCE

               -1                                47
               -17                               6
                15                               10
```

```
GOLDCODE  21 =  000111100001011000011110111011011011001111110000111010000001100
GOLDCODE  21 =  000111100001011000011110111011011011001111110000111010000001100
```

AUTO CORRELATION VALUE	FREQUENCY OF OCCURENCE
63	1
15	16
-1	30
-17	16

LIST OF REFERENCES

Dixon, Robert C. 1984. Spread spectrum systems. New York: John Wiley & Sons, Inc.

Gold, Robert. 1966. Characteristic linear sequences and their coset functions. SIAM Journal on Applied Mathematics, 14:980-985.

Gold, Robert. 1967. Optimal binary sequences for spread spectrum multiplexing. IEEE Transcripts on Information Theory, IT-13:619-621.

Gold, Robert. 1968. Maximal recursive sequences with 3-valued cross-correlation functions. IEEE Transcripts on Information Theory, IT-14:154-156.

Gold, Robert., Dixon, Robert C., and Kaiser, Fred. 1976. TDRSS Telecommunications Systems PN Code Analysis. NASA Contract NAS 5-22546 Final Report.

Hansen, John C. 1965. Modern algebra for coding. Electro-Technology, 76:57-66.

Holmes, Jack K. 1982. Coherent spread spectrum systems. New York: John Wiley & Sons, Inc.

Peterson, Wesley W. 1961. Error-correcting codes. Cambridge: The M.I.T. Press.

Shanmugam, K. Sam. 1979. Digital and analog communication systems. New York: John Wiley & Sons, Inc.

Stremler, G. Ferrel. 1982. Introduction to communications systems. Reading: Addison-Wesley Publishing Company.

Tausworthe, Robert C. 1965. Random numbers generated by linear recurrence modulo two. Mathematical Computation, 19:201-209.

Ziemer, R.E. and Tranter, W.H. 1976. Principles of communication. Boston: Houghton Mifflin Company.

Ziemer, R.E. and Peterson, R.L. 1985. Digital communications and spread spectrum systems. New York: MacMillan Publishing Company.