APPENDIX M

PROPERTIES OF THE DIFFERENTIAL ENCODER SPECIFIED IN IRIG STANDARD 106 FOR OQPSK MODULATIONS

LIST OF FIGURES

LIST OF TABLES

ANNEX 1 TO APPENDIX M
SYSTEM LEVEL SOFTWARE REFERENCE IMPLEMENTATION OF DIFFERENTIAL ENCODER DEFINED IN IRIG STANDARD 106 FOR FQPSK AND SOQPSK MODULATIONS

REFERENCES

This page intentionally left blank.

APPENDIX M

PROPERTIES OF THE DIFFERENTIAL ENCODER SPECIFIED IN IRIG STANDARD 106 FOR OQPSK MODULATIONS

1.0   Introduction

This appendix summarizes a study of the differential encoder originally adopted by the U.S. Department of Defense (DoD) Advanced Range Telemetry (ARTM) project and the Range Commanders Council (RCC) and incorporated into the Interrange Instrumentation Group (IRIG) Standard 106 (IRIG-106) (reference [M-1]) for Feher's Quadrature Phase Shift Keying (FQPSK-B)[1] modulation. The study, performed by Mr. Robert Jefferis of the TYBRIN Corporation, was prompted by inquiries from industry representatives who were concerned that this particular differential code was not associated with commercial telecommunication standards and the fact that manufacturers had experienced confusion over correct implementation. The study results shown in this appendix prove the code to be robust, reliable, and applicable to Shaped Offset QPSK (SOQPSK-TG)[2] as well as FQPSK-B and FQPSK-JR.[3]

This appendix is organized along the following structure. Paragraph 2 describes the need for differential encoding. Paragraph 3 explains the IRIG-106 differential code for OQPSKs. Paragraph 4 demonstrates differential code's invariance with respect to constellation rotation. Paragraph 5 shows the differential decoder to be self-synchronizing. Paragraph 6 reviews the differential decoder's error propagation characteristics. Paragraph 7 analyzes a recursive implementation of the differential code and Paragraph 8 describes use of this code with frequency modulator based SOQPSK transmitters. A description of the implementation of the entire coding and decoding process can be seen at Annex 1 to this appendix.

2.0   The Need For Differential Encoding

Practical carrier recovery techniques like Costas loops and squaring loops exhibit a troublesome M-fold carrier phase ambiguity. A description of ambiguity problems and how to overcome them are shown in the following paragraphs of this appendix.

Shown below at Figure M-1 is a simplified quadriphase transmission system that is one of the methods recommended for transparent point-to-point transport of a serial binary data stream. Transparent means that only revenue bearing data is transmitted. There is no in-line channel coding nor is special bit pattern insertion allowed. The assumption is made for a non-return-to-zero-level (NRZ-L) data stream containing the bit sequence $b(nTb)$ transmitted at rate $r_b = 1/T_b$ bits per second. For QPSK and OQPSK modulations, the bit stream is divided into subsets *"e"* containing even numbered bits and *"o"* containing odd numbered bits. The transmission rate associated with the split symbol streams is $r_s = r_b/2$ symbols per second.

---

[1] FQPSK-B is a proprietary variation of "Offset" QPSK (OQPSK), Digcom Inc., El Macero, California.

[2] See Chapter 2 and Appendix-A for details on SOQPSK-TG (formerly SOQPSK-A*).

[3] FQPSK-JR is an FQPSK variant developed by Mr. Robert Jefferis, TYBRIN Corporation,and Mr. Rich Formeister, RF Networks, Inc.

Symbol values are converted to code symbols by the differential encoder described in section 3.0 below. A baseband waveform generator converts the digital symbol time series into continuous time signals suitable for driving the vector modulator as prescribed for the particular modulation in use. Thus, each subset modulates one of two orthogonal subcarriers, the "in-phase" (*I*) channel, and the "quadrature" (*Q*) channel. The modulator combines these subcarriers, creating a phase modulated RF signal *S*(*t*). On the receive side, demodulation separates the subcarriers, translates them back to baseband, and constructs replicas of the code symbol series $E'(nT_s)$ and $O'(nT_s)$. Decoding reverses the encoding process and a multiplexer (MUX) recreates a replica of the bit stream $b'(nT_b)$.
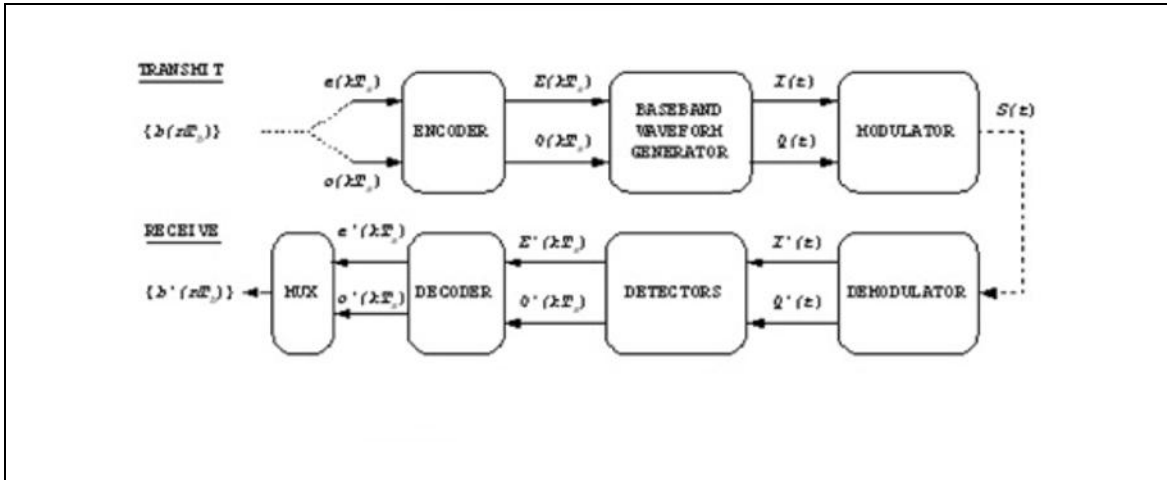

Figure M-1. Transmission System.

Most QPSK and OQPSK systems employ coherent demodulation. Figure M-2 is a simplified diagram of commonly used modulation and demodulation structures. Note the optional single bit delay shown in the odd symbol path. This creates the significant difference between QPSK and OQPSK, the delay being inserted to create OQPSK.[4] Practical carrier recovery techniques like Costas loops and squaring loops exhibit a troublesome M-fold phase ambiguity (M=4 for QPSK and OQPSK) (see reference [M-2]). Each time the demodulator carrier synchronizer phase locks to the modulator local oscillator(LO) its absolute phase relationship to the LO contains the offset term β which can take on values of 0, $\pm$ π/2, or π radians.[5]

---

[4] The delay can be inserted into either channel. The IRIG-106 convention and most published literature regarding FQPSK and SOQPSK indicate the delay in the odd (or Q) channel.

[5] The initial offset angle ϕ is generally unknown and uncontrolled; it is tracked by the carrier recovery circuitry and the symbol timing circuits automatically ignore.

Figure M-2.   OFFSET QPSK 106 Symbol to Phase Mapping Convention.

The symbol detectors have insufficient information to determine which phase offset exists. They always interpret demodulator output with the assumption that $\beta=0$. The resulting constellation axis rotations and their impact on demodulator output are shown at Figure M-3 and Table M-1. The 180-degree rotation is symmetric. The Axis (subcarrier) assignment is unchanged but the sense (polarity) of both axes gets reversed. The 90-degree and 270-degree rotations are asymmetric. Axis assignment is swapped and one axis polarity is reversed in each case.

Figure M-3.   Detection ambiguity.

TABLE M-1. CONSTELLATION AXIS ROTATIONS		
Rotation	**$+I'$**	**$+Q'$**
0	I	Q
$\pi/2$	$-Q$	I
π	$-I$	$-Q$
$3\pi/2$	Q	$-I$

3.0   A Simple Solution To The Carrier Phase Ambiguity Problem

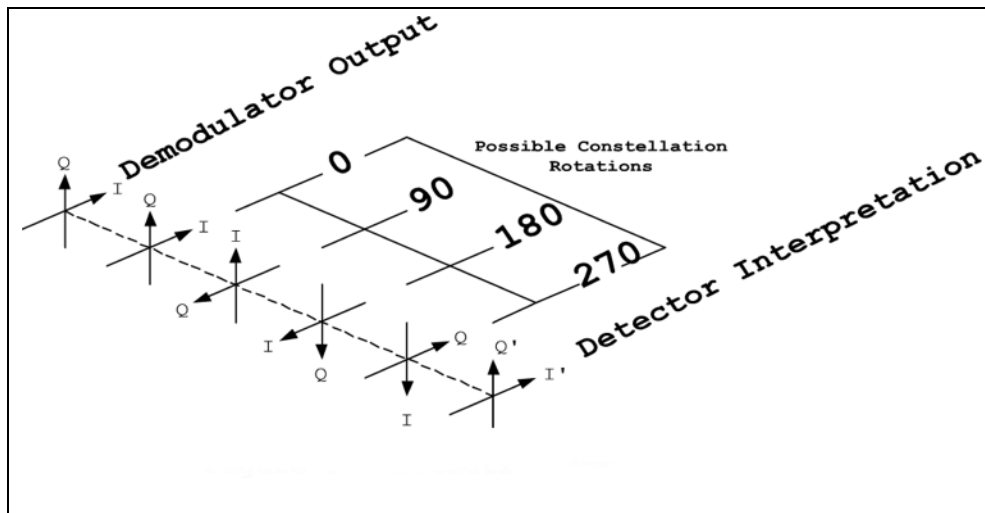Differential encoding has been used to work around the carrier ambiguity for many years. For phase modulations, source data is coded such that phase *differences* rather than absolute phase coordinates become the information-bearing attribute of the signal. The QPSK and OQPSK modulations use *I* and *Q* independently, with each channel transporting one symbol stream. Starting with the first binary digit, bit 0, even numbered bits form the sequence $\{e_k\}$ and odd numbered bits form the sequence $\{o_{k+1}\}$ where the counting index is changed from the bit index *n* to the symbol pair index

$$k = 2n \qquad\qquad\qquad k \in \{0,2,4,6,...\} \qquad\qquad (\text{M-1})$$

Figure M-4 illustrates how QPSK modulators process bits in pairs (dibits), mapping and asserting time coincident symbol phase coordinates (I_k,Q_k)[6]. Phase state changes commence and end on *symbol* interval timing boundaries, each state taking on one of four possible values at detector decision instants. However, the case of interest is shown in Figure M-5.
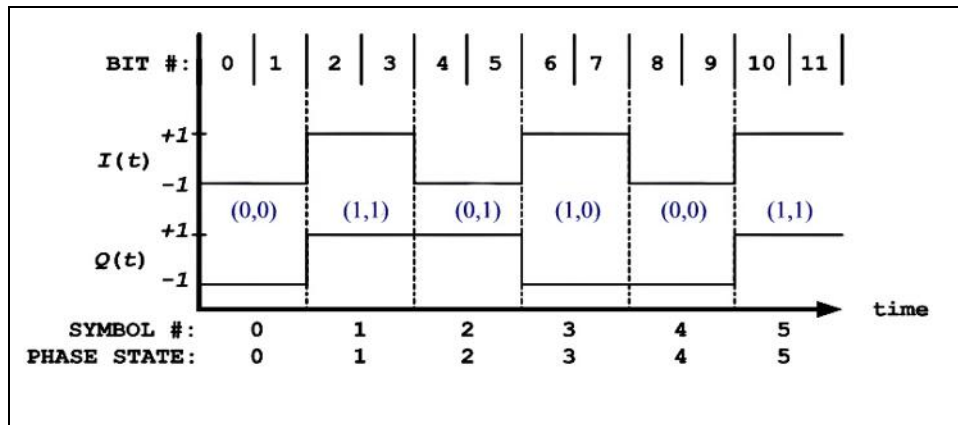

Figure M-4.   QPSK State Timing

---

[6] Rectangular *I* and *Q* baseband waveforms are used only for illustration.
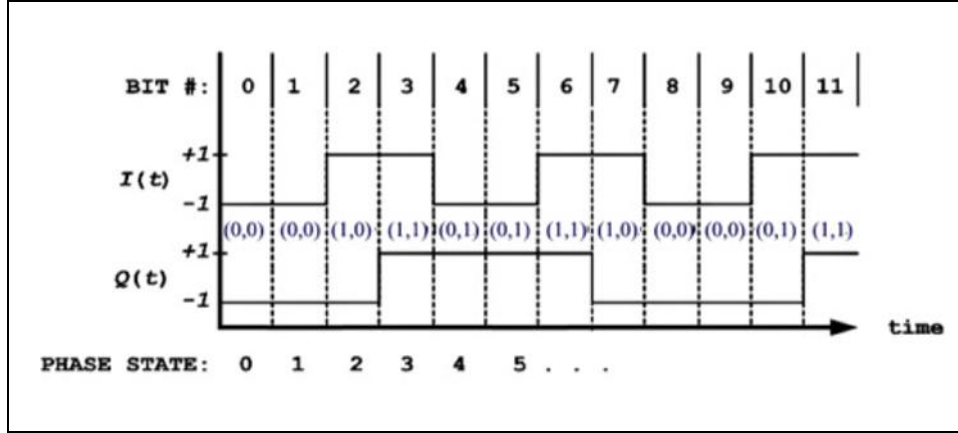
Figure M-5.   OFFSET QPSK State Timing.

The Q channel half-symbol delay causes OQPSK phase trajectories to evolve on a half-symbol (bit) rate basis. For the particular cases of FQPSK and SOQPSK-TG, carrier phase either remains unchanged or changes by $\pm\pi/4$ or $\pm\pi/2$ radians over the pending bit interval.

The OQPSK inter-channel delay might at first seem a difficult complication because it creates additional ambiguity; in other words, the receiver must resolve relative inter-channel delay. However, as shown below, this is not a problem.

The differential encoding rule adopted in IRIG-106 for OQPSK appears in reference [M-4] and is therein attributed to Clewer in reference [M-5] and Weber in reference [M-6]. Bit by bit, the code symbol sets $\{E_k\}$ and $\{O_{k+1})$ are formed with the Boolean expressions:

$$E_k \equiv e_k \oplus \overline{O}_{k-1} \qquad \text{(M-2a)}$$

$$\left. \begin{array}{c} \\ \\ \end{array} \right\} \qquad \text{(M-2)}$$

$$O_{(k+1)} \equiv o_{k+1} \oplus E_k \qquad \text{(M-2b)}$$

Two bits are coded for each value of k in a two-step process. First, the even symbol E_k is coded with current bit e_k. Then the next bit, o_{k+1} becomes current and the odd symbol O_{k+1} is computed. In each code set the exclusive-or operator is applied to the state defining variables just like BPSK differential encoding. Unlike BPSK however, the current source bit and the most recent code symbol from the other channel determine adjacent phase transitions. Also note the asymmetry of these equations introduced by the inverted code symbol in equation (M-2a). Its significance will become evident in the next section.

The code symbol sets $\{E\}$ and $\{O\}$ are applied to the I and Q channels of the OQPSK modulator. The initial assignment of $\{E\}$ to either I or Q can be made arbitrarily. However, with this code definition, once the choice is made at the modulator, decoding will fail if channel assignment conventions change anywhere during the transmission or decoding processes. Thus, the assignment convention must extend to the physical modulator and demodulator. IRIG -06 assigns I to the physical I subcarrier (also known as the "real" or "cosine" subcarrier) and Q is

applied to the physical Q subcarrier (also known as the "imaginary" or "sine" subcarrier). In order to stress this assignment convention, IRIG-106 expresses equation (M-2) explicitly in terms of the I and Q channel variables:

$$I_k \equiv e_k \oplus \overline{Q}_{(k-1)} \qquad \text{(M-3a)}$$

$$Q_{(k+1)} \equiv o_{(k+1)} \oplus I_k \qquad \text{(M-3b)}$$

$$k \in \{0,2,4,6...\} \qquad \text{(M-3)}$$

Decoding is straightforward. When $\beta=0$, $I'=I$, and $Q'=Q$, inspection of the following truth tables reveals simple decoding instructions:

Equation (M-3a)

I_k	$\overline{Q}_{(k-1)}$	e_k
0	0	0
0	1	1
1	0	1
1	1	0

Equation (M-3b)

$Q_{(k+1)}$	I_k	$o_{(k+1)}$
0	0	0
1	0	1
0	1	1
1	1	0

$\Rightarrow$ decoding equation :

$$e'_k = I'_k \oplus \overline{Q}'_{k-1} \qquad \text{(M-4a)}$$

$$o'_{k+1} = Q'_{k+1} \oplus I'_k \qquad \text{(M-4b)}$$

$$k \in \{0,2,4,6,...\} \qquad \text{(M-4)}$$

The equations at (M-3) may not convey an intuitive sense of the shift from absolute phase states to phase differences. Extending (M-3a) backwards in time by substituting (M-3b) into (M-3a) results in:

$$I_k = e_k \oplus \overline{\left(o_{k-1} \oplus I_{k-2}\right)} = I_{k-2} \oplus \overline{\left(e_k \oplus o_{k-1}\right)} \qquad \text{(M-5)}$$

Similarly, for the next bit interval the results are:

$$Q_{k+1} = o_{k+1} \oplus \left(e_k \oplus \overline{Q}_{k-1}\right) = Q_{k-1} \oplus \overline{\left(o_{k+1} \oplus e_k\right)} \qquad \text{(M-6)}$$

This recursive form clearly shows that on a bit by bit basis, the current and most recent bits control phase trajectory *motion*, not absolute phase. Note that (M-5) and (M-6) do not define the sign of a phase change. Predictable decoder output requires that two additional conventions be established and maintained. Boolean logic polarity conventions used throughout the system must be consistent. IRIG-106 *assumes* positive true logic. Finally, sign conventions and channel

assignment used within the transmitter (baseband signal generator and modulator) and the receiver (demodulator) must be constrained to produce a consistent code symbol to phase mapping convention. The IRIG-106 convention is shown in Figure M-2. For example, if {b} were to consist entirely of logic one values, i.e., a run of 1s, the differential encoding process and mapping convention will produce the phase trajectory shown in Table M-2.

TABLE M-2.					RESPONSE TO RUN OF 1S	
n	**b(n)**	**k**	I_k	Q_{k-1}	Q_{k+1}	**Phase (deg)**	**Phase Δ**
0	1	0	0	0*		225*	
1	1				1	135	-π/2
2	1	1	1	1		45	-π/2
3	1				0	315	-π/2
4	1	2	0	0		225	-π/2
5	1				1	135	-π/2
* denotes assumed initial conditions							

The trajectory spins clockwise, and the phase is retarded by 90 degrees during each bit interval.[7] Obviously, any single (unbalanced) sign change and any change to the mapping convention will alter the trajectory.

4.0   Immunity to Carrier Phase Rotation

The equations at (M-3) and (M-4) are invariant with respect to cardinal constellation rotation as shown in the following:

Proof:

The β=0 case is decoded correctly by definition according to equations (M-5) and (M-6). At Table M- 1, when β = π there is no axis swap but the decoder is presented with

$$I'_k = \bar{I}_k$$
$$Q'_{k+1} = \overline{Q}_{k+1}$$

---

[7] FQPSK-B, FQPSK -JR and SOQPSK-TG modulations respond to a run of 1s with an S(t) that is ideally, a pure tone at frequency f_c-r_b/4 Hz. This is referred as "lower sideband" mode. Similarly, a run of zeroes will produce a constant anti-clockwise trajectory spin and a tone at f_c+r_b/4 Hz ("upper sideband" mode).

Decoding will progress as follows:

<u>Step 1</u>. Even channel; apply equation (M-4a);

$$e'_k = I'_k \oplus \overline{Q}'_{k-1} = \bar{I}_k \oplus Q_{k-1} = I_k \oplus \overline{Q}_{k-1} = e_k$$

<u>Step 2</u>. Odd channel; apply equation (M-4b);

$$o'_{k+1} = Q'_{k+1} \oplus I'_k = \overline{Q}_{k+1} \oplus \bar{I}_k = Q_{k+1} \oplus I_k = o_{k+1}$$

Thus, symmetric rotation is transparent to the code. When $\beta = \pi/2$ the decoder sees

$$I'_k = \overline{Q}_{k-1}$$
$$Q'_{k+1} = I_k$$

Decoding takes place in the same sequence:

<u>Step 1</u>, even channel, apply equation (M-4a);

$$e'_k = I'_k \oplus \overline{Q}'_{k-1} = \overline{Q}_{k-1} \oplus \bar{I}_k = I_k \oplus Q_{k-1} = o_{k-1}$$

<u>Step 2</u>, odd channel, apply equation (M-4b);

$$o'_{k+1} = Q'_{k+1} \oplus I'_k = I_k \oplus \overline{Q}_{k-1} = e_k$$

In this case the bit sequence is recovered correctly and the code definition coupled with consistent sign conventions automatically compensates for the asymmetric rotation by reversing the application order of (4a) and (4b). It is noted that the output indexes are shifted back in time one bit period. Asymmetric rotation causes a one-bit delay in the decoding process. Finally, the same result is seen when $\beta = 3\pi/2$:

$$I'_k = Q_{k-1}$$
$$Q'_{k+1} = \bar{I}_k$$

<u>Step 1</u>. Even channel; apply equation (M-4a);

$$e'_k = I'_k \oplus \overline{Q}'_{k-1} = Q_{k-1} \oplus I_k = I_k \oplus Q_{k-1} = o_{k-1}$$

<u>Step 2</u>. Odd channel; apply equation (M-4b);

$$o'_{k+1} = Q'_{k+1} \oplus I'_k = \bar{I}_k \oplus Q_{k-1} = I_k \oplus \overline{Q}_{k-1} = e_k$$

In all cases the decoder correctly reproduces the original bit sequence. Decoding is instantaneous for symmetric rotations but it is delayed by one bit in 2 out of 4 possible asymmetric rotation startup scenarios.

The need for consistent function assignment now becomes clear. Application of (4b) to a code symbol formed with (3a) produces the complement of the original bit. Likewise, application of (4a) to a symbol coded with (3b) inverts the result.

At this point, the OQPSK inter-channel delay ambiguity mentioned in section 2 has not been resolved. The roles of *I'* and *Q'* reverse with asymmetric rotations and there is no way to determine when this occurs. However, as long as the code symbol time sequence is preserved at the decoder and the roles of *I'* and *Q'* do not get reversed in terms of the application of (6a) and (6b), inter-channel delay is transparent to the code with respect to reconstruction of the original data sequence.[8]

5.0   Initial Values

Equations at (M-3) and (M-4) do not impose any implementation constraints on initial values when encoding or decoding starts. To confirm this it is assumed that hardware power-up (or initial data presentation) may cause encoding to commence with either channel. It is further assumed that no provisions for specific initial values in encoder and decoder state memories have been made. If coding starts with *I* (see equation M-3a), the first code symbol will be computed:

$$\|I_0\| = e_0 \oplus \left\langle \overline{Q_{-1}} \right\rangle$$

where $\langle . \rangle$ denotes an unknown initial value and double vertical bars denote computed values influenced by initial values. Encoding equations M-3a and M-3b will progress as follows:

$$\|Q_1\| = o_1 \oplus \|I_0\|$$
$$\|I_2\| = e_2 \oplus \|\overline{Q_1}\|$$

As can be seen, the initial values do establish the absolute sense of code symbols for the duration of transmission. But, on both ends of the process, two of three terms in every equation are affected consistently by the initial value, which by symmetry has no effect on the outcome of exclusive-or operations. Obviously, identical results occur if the encoder starts with *Q*. Independent of starting channel and initial value then, the first and all subsequent adjacent code symbol pairs contain valid state *change* information.

---

[8] If for some reason the system application requires that one can determine whether a specific symbol was originally transmitted via *I* or *Q*, then this code is not appropriate.

Initial decoder values can produce errors. Again starting with *I*, and using equations (M-4a) and (4b), decoding will progress as follows:

$$\left\| e'_0 \right\| = I'_0 \oplus \left\langle \overline{Q'_{-1}} \right\rangle$$

$$o'_1 = Q'_1 \oplus I'_0$$

It is seen that on the second cycle the initial value of the decoder has been flushed out. At most, one bit will be decoded in error. Similarly, if decoding starts with *Q*, output will progress:

$$\left\| o'_1 \right\| = Q'_1 \oplus \left\langle I'_0 \right\rangle$$

$$e'_2 = I'_2 \oplus \overline{Q}'_1$$

Again, only the first decoded bit may be incorrect. The conclusion, then, is that initial values can produce at most, one decoded bit error. However, there is another source of startup errors that is seen as an initial value problem. Section 4.0 showed that odd phase rotations ($\pi/2$ and $3\pi/2$) cause a single bit delay in the decoder. Examining this further, the first symbol index value will be $k = 0$. If the decoder starts with equation (M-4a), the first decoded bit will be:

$$e'_0 = I'_0 \oplus \left\langle \overline{Q'_{-1}} \right\rangle = I_0 \oplus \left\langle Q_{-1} \right\rangle = \left\langle o_{-1} \right\rangle$$

If the decoder starts with equation (M-4b) the first result will be:

$$o'_1 = Q'_1 \oplus I'_0 = I_0 \oplus \left\langle \overline{Q_{-1}} \right\rangle = \left\| e_0 \right\|$$

The first case produces the aforementioned delay. The decoder emits an extra bit. The second bit emitted is actually the first bit of the sequence reconstruction and is still subject to the single initial value error probability of startup processing. The latter case does not produce a delay; it only presents the possibility of a first bit decoding error.

6.0   Error Propagation

Differential encoding incurs a bit error penalty because received code symbols influence more than one decoded bit. First consider a single symbol detection error in *current* symbol *E'* which is labeled ε_k. The following sequence of decoding steps shows how the error propagates. Since the *E* channel was chosen as current, decoding starts with equation (M-4a). The single detection error creates two sequential decoding errors. By symmetry we can state that the same result occurs if a single error occurs in *O'*.

$$b'_k = \varepsilon_k \oplus \overline{Q}_{k-1} = \overline{b}_k \Rightarrow \text{error}$$

$$b'_{k+1} = Q_{k+1} \oplus \varepsilon_k = \overline{b}_{k+1} \Rightarrow \text{error}$$

$$b'_{k+2} = E'_{k+2} \oplus Q'_{k+1} = b_{k+2} \Rightarrow \text{correct}$$

Next is the case of two symbol detection errors occurring consecutively on *E'* and *O'*, i.e., detectors emit error symbols $E'_k=\varepsilon_k$ and $O'_{k+1}=\varepsilon_{k+1}$. Starting again with equation (M-4a) yields:

$$b'_k = \varepsilon_k \oplus \overline{Q}_{(k-1)} = \overline{b}_k \Rightarrow \text{error}$$

$$b'_{(k+1)} = \varepsilon_{(k+1)} \oplus \varepsilon_k = O'_{(k+1)} \oplus E_k = b_{(k+1)} \Rightarrow \text{correct}$$

$$b'_{(k+2)} = E'_{(k+2)} \oplus \varepsilon_{(k+1)} = b_{(k+2)} \Rightarrow \text{error}$$

$$b'_{(k+3)} = O'_{(k+3)} \oplus E'_{(k+2)} = b_{(k+3)} \Rightarrow \text{correct}$$

Two consecutive symbol errors produce two decoding errors but the errors are not adjacent. The conclusion from this is that symbol detection errors influence no more than two decoding cycles, i.e., the maximum error multiplication factor is 2.

7.0　Recursive Processing and Code Memory

Most systems reconstruct the original bit rate clock and {*b*} by merging {*e'*} and {*o'*}. For a variety of reasons, designers might be tempted to multiplex {*I'*} and {*Q'*} into a bit rate code symbol sequence {B_n} prior to decoding. However, the same considerations that foster desire for post-multiplex decoding are likely to be accompanied by loss of transmitted code symbol order, i.e., loss of knowledge whether a given code symbol came from *I* or *Q*. The question arises as to whether {B_n} alone contains enough information for unique decoding. The answer is "no", and the proof is shown below.

Proof:

An alluring decoding function can be derived by inspection of equations (M-5) and (M-6). Equation (M-5) can be rearranged as follows:

$$I_k = e_k \oplus o_{k-1} \oplus \overline{I}_{k-2} \qquad \text{(M - 7)}$$

Similarly, from equation (M-6) we can write

$$Q_{k+1} = o_{k+1} \oplus e_k \oplus \overline{Q}_{k-1} \qquad \text{(M - 8)}$$

Here are two instances of a seemingly identical recursive relationship, i.e., the current code symbol is the difference between the current bit, the previous bit, and the inverse of the most recent code symbol from the current channel. We can consolidate these equations by converting to post-multiplex bit rate indexing, i.e.,

$$B_n = b_n \oplus b_{(n-1)} \oplus \overline{B}_{(n-2)} \qquad \text{(M-9)}$$

from which we can immediately write the decoding function

$$b'_n = b'_{(n-1)} \oplus B'_n \oplus \overline{B}'_{(n-2)} \qquad \text{(M-10)}$$

On the surface it seems that equation (M-10) will work.[9] However, these relations involve two differences, rather than one, and therefore introduce superfluous initial condition dependence. For brevity, only the pitfalls of (M-10) are examined herein, assuming that a non-recursive encoder is used. From startup, decoding will progress as follows:

$$\left\|b'_0\right\| = \left\langle b'_{-1}\right\rangle \oplus B'_0 \oplus \left\langle \overline{B}'_{-2}\right\rangle$$

$$\left\|b'_1\right\| = \left\|b'_0\right\| \oplus B'_1 \oplus \left\langle \overline{B}'_{-1}\right\rangle$$

$$\left\|b'_2\right\| = \left\|b'_1\right\| \oplus B'_2 \oplus \overline{B}'_0$$

$$\left\|b'_3\right\| = \left\|b'_2\right\| \oplus B'_3 \oplus \overline{B}'_1$$

$$\vdots$$

As seen, absolute polarity of the first and all subsequent decoded bits is determined by three (3) initial values. Absent appropriate a priori side information for selecting initial values, the post-multiplex decoder offers a 50-50 chance of decoding with correct polarity. The code sequence defined by equations at (M-3) has a two-symbol memory. Additional symbols do not provide new information regarding the trajectory history. Another way to view this problem is to note that this recursive decoder does not guarantee preservation of symbol order, which is a prerequisite to reliable decoding.

8.0   Frequency Impulse Sequence Mapping for SOQPSK

The SOQPSKs first described by Hill and Geohegan in references [M-7] and [M-8] are defined as special cases of continuous phase modulation (CPM). Since 1998, at least two manufacturers have exploited the fact that modern digital waveform synthesis techniques enable direct implementation of the CPM equations with virtually ideal frequency modulators and filter impulse responses. A generic model of these implementations is at Figure M-6. The I and Q channels, per se, do not exist in this transmitter. At the beginning of each bit interval, impulses from the bit to impulse alphabet mapper direct the impulse filter/frequency modulator to advance the carrier phase by 90°, retard it by or 90°, or leave the phase unchanged. This is accomplished with a ternary alphabet of frequency *impulses* having normalized amplitudes of {-1,0,1}.[10] Obviously, this structure cannot be mapped directly into the constellation convention of a quadriphase implementation because there is no way to control absolute phase. The equations at (M-3) can be applied to this non-quadrature architecture via pre-coding. A general treatment SOQPSK pre-coding is contained in reference [M-9]. It is easily shown that the pre-coding truth table given in Table M-3 applied to the model in Figure M-7 will yield a phase trajectory history identical to one generated by the quadriphase counterpart of Figure M-2 using the equations at (M-3). However, one more constraint is necessary to establish compatibility with the IRIG-106 quadriphase convention. Table M-3 assumes the stipulation that positive sign impulse values will cause the modulator to increase carrier frequency.

---

[9] The interested reader is left to confirm that equation (10) is indeed rotation invariant.

[10] The so-called ternary alphabet is actually 2 binary alphabets {-1,0} and {0,1}, the appropriate one chosen on a bit-by-bit basis according to certain state transition rules.
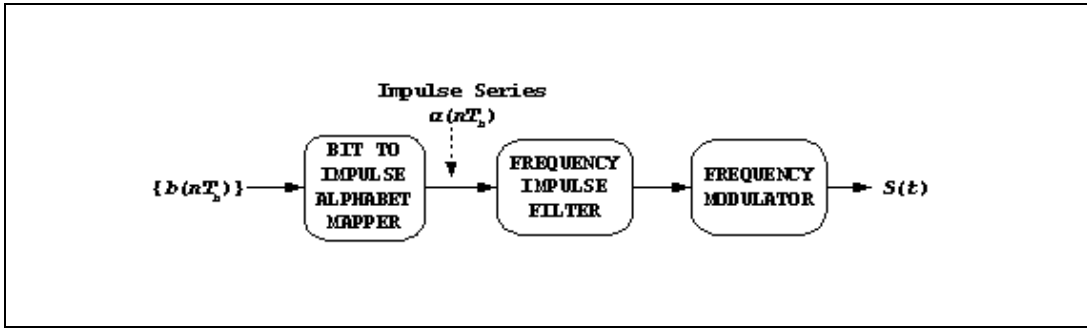
Figure M-6.   Basic SOQPSK Transmitter.

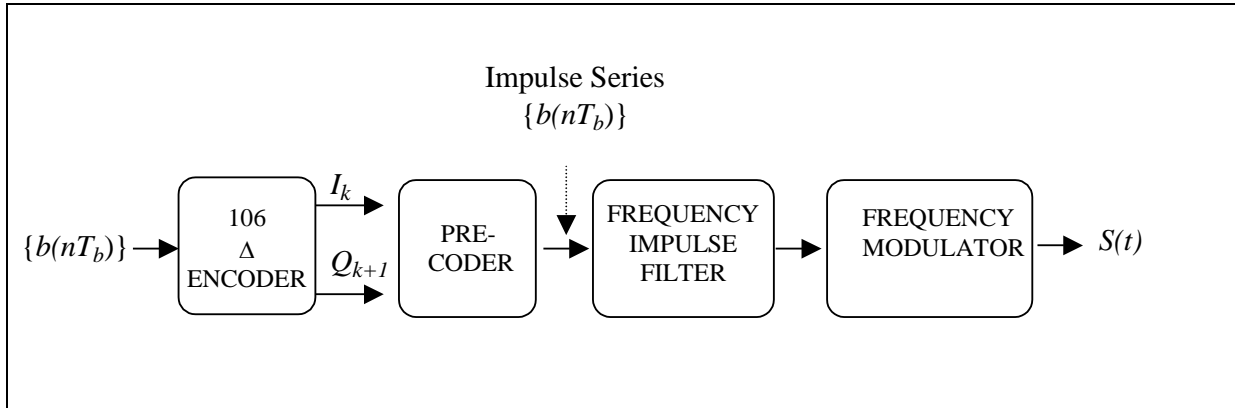| TABLE M-3.		SOQPSK PRE-CODING TABLE FOR IRIG-106 COMPATIBILITY							
MAP α_K FROM I_K					**MAP α_{K+1} FROM Q_{K+1}**				
I_k	Q_{k-1}	I_{k-2}	$\Delta\Phi$	α_k	Q_{k+1}	I_k	Q_{k-1}	$\Delta\Phi$	α_{k+1}
-1	X*	-1	0	0	-1	X*	-1	0	0
+1	X*	+1	0	0	+1	X*	+1	0	0
-1	-1	+1	$-\pi/2$	-1	-1	-1	+1	$+\pi/2$	+1
-1	+1	+1	$+\pi/2$	+1	-1	+1	+1	$-\pi/2$	-1
+1	-1	-1	$+\pi/2$	+1	+1	-1	-1	$-\pi/2$	-1
+1	+1	-1	$-\pi/2$	-1	+1	+1	-1	$+\pi/2$	+1
* Note: Does not matter if "X" is a +1 or a -1									

Figure M-7.   OQPSK Transmitter (with precoder).

9.0   Summary[11]

This investigation confirmed that the differential encoder defined in the equations at (M-3) is entirely satisfactory for SOQPSK, FQPSK-JR and FQPSK-B systems where conventional coherent demodulation and single symbol detection is used. In addition, a method of extending this code to SOQPSK is presented without proof.

Specifically, the following has been shown:

a.   When accompanied by consistent sign conventions, a consistent symbol to phase mapping rule, and preservation of symbol order, the OQPSK differential code defined in (M-3) and the decoding rule defined in (M-4) is rotation invariant and unambiguously reconstructs the original data bit sequence.

b.   Decoding is instantaneous.

c.   Equations (M-3) and (M-4) do not require attention to initial values.

d.   At most, two consecutive output bits will be in error after carrier and symbol synchronization is acquired.

e.   The recursive relations in (M-9) and (M-10) are ambiguous and therefore unreliable.

f.   The code exhibits a detection error multiplication factor of at most two.

---

[11] There is no doubt in the author's mind that well trodden ground has been traveled in this investigation. These characteristics were probably validated in reference [5] and by RF Networks Inc. before it incorporated the encoder in its model 5450F FQPSK demodulator product. Unfortunately, none of this work is in the public domain.

ANNEX 1 TO APPENDIX M

**SYSTEM LEVEL SOFTWARE REFERENCE IMPLEMENTATION OF
DIFFERENTIAL ENCODER DEFINED IN IRIG STANDARD 106 FOR FQPSK AND
SOQPSK MODULATIONS**

1.0   Introduction

The Matlab®™ program listings below provide a Matlab function "Desysdemo" and an execution control script "runDEdemo". In the context of differential encoding, the function provides a complete system simulation including a differential encoder, an ideal vector modulator, channel phase rotation, demodulation, the functional equivalent of an ideal single symbol sample and hold detector, and a decoder. The user can create sample data vectors or use the example data provided. In addition, by manipulating the initial value vectors, all possible initial value and demodulator phase rotation combinations of the quadriphase implementation model can be explored.

By setting the variable "style" to zero, the function will also emulate the pre-coded frequency modulator architecture required for SOQPSKs. However, the initial value of transmitter carrier phase is hard coded at 45 degrees. This was done to avoid proliferation of initial value options and is thought to be an insignificant omission because it does not affect generality of the phase rotation options.

It is assumed that the user is familiar with Matlab workspace operation. The program relies only on basic Matlab license libraries. There are no special toolboxes or blocksets are required.

2.0   Matlab Workspace Operation

The user should place the script (shown below in paragraph 3.0 of this Annex) in the directory of choice and make that directory current in the workspace. In order to execute the "canned" example, the user needs to create the variable "example" in the workspace and set its value to 1.

Executing the script "runDEdemo" should produce the following output:

results =

Model: Quadriphase Vector Modulator
Demodulator Phase Rotation = 0 degrees

Initial States:	Encoder Memory	Encoder Channel	Decoder Memory	Decoder Channel
	(0,0)	0	(0,0)	0

Input Bit	TX Phase	RX Phase	Output Bit	Decoding Error
1	225	225	1	0
1	135	135	1	0
1	45	45	1	0
0	45	45	0	0
0	135	135	0	0
1	135	135	1	0
0	135	135	0	0
1	135	135	1	0
1	45	45	1	0
1	315	315	1	0
0	315	315	0	0
0	45	45	0	0
1	45	45	1	0
0	45	45	0	0

The first column of the results shown above is a replica of the input data vector. The second column shows the initial value dependent evolution of transmitted phase. The third column shows the effect of any non-zero phase rotation chosen. The fourth column shows the decoded output bit stream and the fifth column flags decoding errors with values of 1. Certain combinations of phase rotation and initial values will produce values of 9 in the fourth and fifth columns; results of this nature are associated with cases that delay the output decoding process by one bit.

Variable definitions and implied instructions for manipulating the runtime options can be obtained by using the normal Matlab help command for these specific programs.

3.0   Script For Modules

Electronic copies of these programs have been provided to the DoD Range Commanders Council, Telemetry Group. The script for the modules discussed above is shown on the following pages.

```
% Control Script 'runDEdemo', for running system demonstration
% of differential encoder and phase mapping convention
% defined in RCC standard IRIG-106 for FQPSK-B modulation.
% This version extends demonstration options to the pre-coder
% required for implementing SOQPSK with frequency modulators.
%
% Each example run requires input variables in the Matlab workspace:
%
%       "example" - a flag to run with user supplied data vector or run
%               the example data set that consists of two repetitions of a
%               a 7-bit pseudo random sequence(0=user, 1=example)
%       "data" - optional user supplied binary bit sequence (arbitrary length)
%       "rotation_choice" - pointer to demodulator phase rotation options:
%               1=0, 2=pi/2, 3= pi, 4=3*pi/2
%       "initTX" - vector of binary encoder startup values:
%               initTX(1)= 1st of two encoder code symbol memory values(binary, arbitrary)
%               initTX(2)= 2nd encoder code symbol memory value(binary, arbitrary)
%               initTX(3)= starting channel for encoder(binary, 0=I, 1=Q)
%       "initRX" - vector of binary decoding startup values
%               initRX(1)= 1st of two decoder state memory values(binary, arbitrary)
%               initRX(2)= 2nd decoder state memory value(binary, arbitrary)
%               initRX(3)= starting channel for decoder(binary, 0=I, 1=Q)
%       "style" - 1=quadriphase transmitter architecture (FQPSK)
%                       0=frequency modulator transmitter architecture (SOQPSK)
%       The example values are:
%               data=[1 1 1 0 0 1 0 1 1 1 0 0 1 0]
%               rotation_choice=1
%               initTX=[0 0 0]
%               initRX=[0 0 0]
%               style=1

% R.P.Jefferis, TYBRIN Corp., JULY, 2002
% SOQPSK model added 14JUL03
% This version has been tested with Matlab versions:5.2,6.1

% *** Sample Input Setup ***
if example
  data=[1 1 1 0 0 1 0 1 1 1 0 0 1 0];
  rotation_choice=1;
  initTX=[0 0 0];
  initRX=[0 0 0];
  style=1;
end

% *** Run the Reference Implementation ***
```

```
[test,delay]=DEsysdemo(data,rotation_choice,initTX,initRX,style);

% *** Prepare Screen Output ***

ROTATION=[0 90 180 270];
if style
  results=sprintf('Model: Quadriphase Vector Modulator\n')
else
  results=sprintf('Model: Frequency modulator (SOQPSK) model\n')
end
results=[results sprintf('Demodulator Phase Rotation = %3.0f
degrees\n',ROTATION(rotation_choice))];
results=[results sprintf('Initial States: Encoder Encoder Decoder Decoder\n')];
results=[results sprintf('          Memory  Channel Memory  Channel\n')];
results=[results sprintf('-------------------------------------------------\n')];
results=[results sprintf('          (%d,%d)    %d   (%d,%d)    %d\n\n',...
    initTX(1:2),initTX(3),initRX(1:2),initRX(3))];
results=[results sprintf(' Input TX    RX   Output Decoding\n')];
results=[results sprintf(' Bit  Phase  Phase  Bit   Error\n')];
results=[results sprintf('-----------------------------------\n')];
for n=1:length(data)
  results=[results sprintf(' %d    %3.0f   %3.0f    %d     %d\n',...
      test(n,:))];
end
results

% _____END OF CONTROL SCRIPT_____

function [result,delay]= DEsysdemo(inbits,rotation_choice,initTX,initRX,style)
% Reference simulation for Range Commanders Council standard IRIG 106-2000
%       FQPSK-B differential encoding and phase mapping convention.
%
% Input arguments: see "help" for "runDEdemo" script
% Output arguments:
%       "result" - Mx5 matrix,M=number of input bits,columns contain:
%                (:,1)input bit,(:,2)TX phase,(:,3)RX phase,(:,4)output bit,(:,5)status
%       "delay"        - overall encode/decode process delay in bits

% "TX" prefixes refer to transmitter/encoder variables, "RX" prefixes
% refer to receiver/decoder variables
% Robert P. Jefferis, TYBRIN Corp., July,2002.
% SOQPSK model added 14JUL03
% This version has been tested with Matlab versions: 5.2,6.1
numbits=length(inbits)

% *******************
```

```
% *   Transmitter   *
% *******************

% *** differential encoder (also SOQPSK pre-coder)****

% encoder memory initial values:
%       [(last I ch. code symbol) (last Q ch. code symbol)]
TXlastSYM=initTX(1:2);
% point encoder to either I or Q starting channel(0=I)
TXpoint=initTX(3);
for n=1:numbits
  switch TXpoint
  case 0
    %TXlastSYM
    % compute "current" I channel code symbol
    TXnewISYM=xor(inbits(n),~TXlastSYM(2));
    TXcodeSYM(n,:)=[TXnewISYM TXlastSYM(2)]; % new phase coordinates(I,Q)
    TXlastSYM(1)=TXnewISYM; % update encoder memory state
    TXpoint = ~TXpoint; % point to Q channel eq. for next bit
  case 1
    % compute "current" Q channel code symbol
    TXnewQSYM=xor(inbits(n),TXlastSYM(1));
    TXcodeSYM(n,:)=[TXlastSYM(1) TXnewQSYM]; % new phase coordinates(I,Q)
    TXlastSYM(2)=TXnewQSYM;% update encoder memory state
    TXpoint= ~TXpoint;       % point to I channel eq. for next bit
  otherwise
    disp('Invalid Specification of Encoder starting channel');
  end
end

% *** modulate ***

switch style
case 1  % ** Quadriphase vector modulator **

  % RCC IRIG 106 FQPSK-B phase mapping convention: (I,Q)
  for n=1:numbits
    index=floor(2*TXcodeSYM(n,1)+TXcodeSYM(n,2));
    switch  index
    case  3     % [1 1]
      TXphase(n)=45; % TX phase angle, degrees
    case  1     % [0 1]
      TXphase(n)=135;
    case  0     % [0 0]
      TXphase(n)=225;
    case  2     % [1 0]
```

```
      TXphase(n)=315;
    otherwise, disp('map error')
    end
  end
case 0  % ** Frequency modulator w/pre-coder **

  % * pre-coder *
  % map code symbol sequence to frequency impulse series, alpha(n)
  alpha=zeros(1,numbits);
  TXpoint=initTX(3); % in this mode, points to start index
  for n=3:numbits
    if TXpoint % Q(k+1) map
      if TXcodeSYM(n,2)==TXcodeSYM(n-2,2)
      elseif xor(TXcodeSYM(n,2),TXcodeSYM(n-1,1))
        alpha(n)=-1;
      else
        alpha(n)=1;
      end
    else        % I(k) map
      if TXcodeSYM(n,1)==TXcodeSYM(n-2,1)
      elseif xor(TXcodeSYM(n,1),TXcodeSYM(n-1,2))
        alpha(n)=1;
      else
        alpha(n)=-1;
      end
    end
    TXpoint=~TXpoint; % switch to complement function for next bit
  end

  % convert alpha to phase trajectory
  lastTXphase=45; % initial phase of S(t)
  for n=1:numbits
    TXphase(n)=mod(lastTXphase+alpha(n)*90,360);
    lastTXphase=TXphase(n);
  end
otherwise
end

%      ************
%  * Receiver *
%      ************

% *** Demodulator Phase Rotation ***
ROTATE=[0 pi/2 pi 3*pi/2];
rotate=ROTATE(rotation_choice);
for n=1:numbits
```

```
    switch rotate
    case 0
      RXphase(n)=TXphase(n);
    case pi/2
      RXphase(n)=mod(TXphase(n)+90,360);
    case pi
      RXphase(n)=mod(TXphase(n)+180,360);
    case 3*pi/2
      RXphase(n)=mod(TXphase(n)+270,360);
    otherwise
    end
end

% *** detector ***
for n=1:numbits
  switch RXphase(n)
  case 45
    RXcodeSYM(n,:)=[1 1];
  case 135
    RXcodeSYM(n,:)=[0 1];
  case 225
    RXcodeSYM(n,:)=[0 0];
  case 315
    RXcodeSYM(n,:)=[1 0];
  otherwise
  end
end

% *** decode and reconstruct data bit sequence ***

% decoder memory initial values:
%       [(last decoded I channel bit) (last decoded Q channel bit)]
RXlastSYM=initRX(1:2);
% point decoder channel to either I or Q starting channel (0=I)
RXpoint=initRX(3);
for n=1:numbits
  switch RXpoint
  case 0
    % compute "current" decoded I channel bit
    RXbits(n)=xor(RXcodeSYM(n,1),~RXlastSYM(2));
    RXlastSYM=RXcodeSYM(n,:);  % update decoder state
    RXpoint = ~RXpoint; % point to Q channel eq. for next bit
  case 1
    % compute "current" decoded Q channel bit
    RXbits(n)=xor(RXcodeSYM(n,2),RXlastSYM(1));
    RXlastSYM=RXcodeSYM(n,:);  % update decoder state
```

```
     RXpoint= ~RXpoint; % point to I channel eq. for next bit

  otherwise
  end
end

% _____ END OF TX and RX Processing _____

% *******************
% * Assemble Output *
% ******************

% identify delay incurred in overall process
offset=xcorr(inbits,RXbits);
offset(1:numbits-1)=[];
[offset,delay]=max(offset(1:min(length(offset),10)));
delay=delay-1;

% adjust RX output bit vector to compensate for delay,
% inserting values of 9 at beginning of vector to represent
% artifact bits associated with asymmetric rotation cases
checkbits=inbits;
if delay
  newfront=ones(1,delay)*9;
  checkbits=[newfront inbits];
  checkbits(end-delay+1:end)=[];
  RXbits(1:delay)=9;
end
% identify decoding errors in reconstructed bit stream
xmsn_error=checkbits~=RXbits;
xmsn_error(1:delay)=9;
% assemble output matrix
result(:,1)=inbits';
result(:,2)=TXphase';
result(:,3)=RXphase';
result(:,4)=RXbits';
result(:,5)=xmsn_error';

% _____END OF FUNCTION DEsysdemo_____
```

REFERENCES

[1] Telemetry Standards, IRIG STANDARD 106, Secretariat, Range Commanders Council, White Sands Missile Range, Mew Mexico.

[2] Proakis, J.G., *Digital Communications*, 5th Edition, McGraw-Hill Inc., New York, 1989.

[3] Cacciamani, E.R and Wolejsza Jr., C.J, "Phase-Ambiguity Resolution in a Four-Phase PSK Communications System", IEEE Transactions on Communications, Vol. COM-19, No.6, December 1971.

[4] Feher, K., *Digital Communications: Satellite/Earth Station Engineering*, Prentice-Hall Inc., New Jersey, 1983,pp. 168-170.

[5] Cited from bibliography of reference 4 for completeness only. Not available: Clewer, R., "Report on the Status of Development of the High Speed Digital Satellite modem", RML-009-79-24, Spar Aerospace Limited, St. Anne de Bellevue, P.Q., Canada, November 1979.

[6] Weber, W.J. III, "Differential Encoding for Multiple Amplitude and Phase Shift Keying Systems", IEEE Transactions on Communications, Vol. COM-26, No. 3, March 1978.

[7] Hill T., "An Enhanced, Constant Envelope, Interoperable Shaped Offset QPSK(SOQPSK) Waveform for Improved Spectral Efficiency", Proceedings of the International Telemetering Conference, San Diego, California, October 2000.

[8] Geoghegan, M., "Implementation and Performance Results for Trellis Detection of SOQPSK", Proceedings of the International Telemetering Conference, Las Vegas, Nevada, October 2001.

[9] Simon, M.K., "Multiple-Bit Differential Detection of Offset Quadriphase Modulations", IPN Progress Report 42-151, November 15, 2002, Jet Propulsion Laboratory, Pasadena, California.

**** END OF APPENDIX M ****