

Theory and Implementation of a Splitband Modem Using the TMS32010

APPLICATION REPORT: SPRA013

George Troullinos, Peter Ehlig, Raj Chirayll,
Jon Bradley, Domingo Garcia
Digital Signal Processor Products
Semiconductor Group
Texas Instruments

Digital Signal Processing Solutions



IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Theory and Implementation of a Splitband Modem Using the TMS32010

Abstract

This report discusses the theory and implementation of a 1200-bps splitband modem, the Bell 212A/V.22. Throughout the report, the terms Bell 212A and V.22 both refer to implemented modem.

The report covers the following topics:

- ❑ Modem Functional Blocks
 - Modem Transmitter
 - Modem Receiver
- ❑ Modem Hardware Description
 - Analog Interface
 - Filters
 - Data Converters
- ❑ Functions Implemented in the TMS32010
 - Transmit Filters
 - Receive Filters
 - AGC Implementation
 - Carrier Recovery Implementation
 - Baud Clock Alignment Implementation

- ☐ Functions Implemented in the TMS7742
 - Asynchronous-to-Synchronous and Synchronous-to-Asynchronous Conversions
 - Scrambler/Descrambler
- ☐ Performance
- ☐ Other Implementation Considerations
- ☐ Conclusions and References

The report also includes the following appendixes:

- ☐ Appendix A Derivation of DM Structure Equations
- ☐ Appendix B Effects of Nonideal Hilbert Transformers
- ☐ Appendix C AGC Table Generator Code
- ☐ Appendix D TMS32010 Source Code
- ☐ Appendix E TMS7742 Source Code



Product Support

World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. New users must register with TI&ME before they can access the data sheet archive. TI&ME allows users to build custom information pages and receive new product updates automatically via email.

Email

For technical issues or clarification on switching products, please send a detailed email to dsph@ti.com. Questions receive prompt attention and are usually answered within one business day.

Introduction

With the predominant usage of computers and especially PCs, data communications are of increasing importance. Communication between the various computer systems and terminals is frequently accomplished by means of the Public Switched Telephone Network (PSTN). The essential element for this data communication is the modem, which interfaces computer systems and terminals with the telephone network.

In the past, modems have been traditionally implemented in the analog domain using discrete components. Recently, modem manufacturers have realized the flexibility and high performance offered by digital approaches. With the drastic reduction in the cost of digital signal processors, the power of Digital Signal Processing (DSP) becomes available for the implementation of medium-speed and high-speed modems.

This application report discusses the digital implementation of a modem using the TMS32010 Digital Signal Processor. Attention is focused on splitband modems, a class of modems that splits the bandwidth of the communications channel (telephone network) so that full-duplex operation can occur. The splitband technique is mainly used for implementing modems with data rates up to 2400 bps (bits per second). This report describes the theory and implementation of the Bell 212A/V.22 Recommendation, a 1200-bps splitband modem. Note that in the remainder of this report, the designations Bell 212A and V.22 are used synonymously to refer to the modem implemented. This report is not intended to provide a commercial product, but to introduce the implementation considerations and merits of digital signal processing-based approaches. Some of the protocol requirements for the Bell 212A/V.22 Recommendation are not implemented: the answer mode, the 300-bps Frequency Shift Keying (FSK) modem, and the notch filter required to reject the guard tone from the received signal.

Modems are sophisticated devices consisting of many functional blocks that must be correctly implemented. The interface of the functional blocks must also be appropriately adjusted for the overall structure to function properly. The different functional blocks can be implemented in many ways. For example, the receiver input bandpass filters can be recursive or nonrecursive, and different algorithms can be used for the carrier recovery and clock recovery. In addition to the possibility of implementing different algorithms, new algorithms may need to be added to the already existing structure, such as an adaptive equalizer or a second loop within the carrier recovery for the suppression of carrier-phase jitter. These considerations indicate the advantage of the microprocessor-based over the analog-based approach. Using the microprocessor approach, the designer can test different algorithms by simply modifying the software. Additional functional blocks can be included by simply adding new code. Therefore, high-performance modems can be implemented in a very short period of time.

The computational burden for digital modem implementation is very heavy. This implies the need of special features for the microprocessor to be used. The TMS32010, with its 200-ns cycle time, on-chip multiplier, and specialized instruction set is uniquely architected for digital signal processing. Because of this, the TMS32010 can implement the modem functional blocks using only a portion of its available processing power. Another major advantage of the microprocessor approach is the possibility of implementing variable-rate modems using the same hardware. Specifically, the same hardware used for the implementation of the Bell 212A/V.22 Recommendation can be used to implement 2400-bps splitband modems (CCITT V.22 bis Recommendation) by merely changing the software. Besides implementing various modems using the same hardware, additional functions can be included, such as speech store-and-forward and the Data Encryption Standard (DES)¹ for secure data communications.

This report is organized as follows: The first section, Modem Functional Blocks, is a description of the functional blocks required for implementation of the Bell 212A/V.22 Recommendation. The second section, Modem Hardware Description, is a brief discussion of the hardware used for the modem implementation. The functions implemented within the TMS32010 are described in detail in the third section, while the fourth section contains an overview of the functions implemented in the modem controller (the Texas Instruments TMS7742 microcomputer). The performance of the TMS32010-based modem is presented in the fifth section. Finally, the last section suggests alternative hardware configurations that can result in reduced system cost.

The prerequisites for understanding and gaining maximum benefit from this report are the level of a Bachelor's degree in Electrical Engineering and a basic understanding of Digital Signal Processing and Data Communications. Background material can be found in *Digital Signal Processing* (Chapters 1 through 7) by A.V. Oppenheim and R.W. Schaffer; "Implementation of FIR/IIR Filters with the TMS32010/TMS32020," an application report in the book, *Digital Signal Processing Applications with the TMS320 Family*, offered by Texas Instruments; and in *Understanding Communications Systems* and *Understanding Data Communications*, books published by Texas Instruments.

Modem Functional Blocks

A modem (MOdulator-DEModulator) is a device that modulates the baseband information at the transmitter, and demodulates the received signal to retrieve the baseband information at the receiver. The Bell 212A is a full-duplex modem with the receiver and transmitter sharing the available bandwidth of the communications channel. This type of modem is said to operate in either the originate or answer mode (see Figure 1). In the originate mode, it initiates the communication process, transmits with a carrier frequency of 1200 Hz, and receives at the frequency of 2400 Hz. At the other end of the communications channel is a modem that operates in the answer mode, i.e., receives at 1200 Hz and transmits at 2400 Hz.

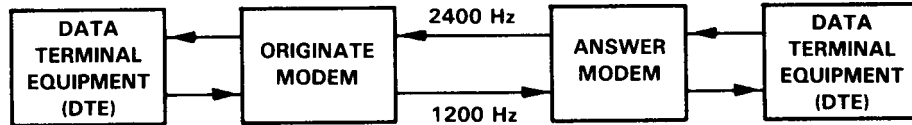


Figure 1. Originate/Answer Configuration

Table 1 shows the different functional blocks that comprize the modem transmitter and receiver.

Table 1. Modem Functional Blocks

Modem Transmitter	Implemented
Guard Tone Generator	No
Scrambler	Yes
Encoder	Yes
Digital Lowpass Filters	Yes
Originate Mode Modulator	Yes
Answer Mode Modulator	No
Modem Receiver	Implemented
Notch Filter	No
Originate Mode Bandpass Filters	Yes
Answer Mode Bandpass Filters	No
Automatic Gain Control	Yes
Demodulator	Yes
Decision Block	Yes
Decoder	Yes
Descrambler	Yes
Clock Recovery	Yes
Carrier Recovery	Yes

In the following two subsections, the operation of the modem transmitter and receiver are described. The transmitter accepts data (bits) from the Data Terminal Equipment (DTE). The DTE may be a dumb terminal, a PC, or a mainframe computer. The modem transmitter then performs the necessary processing to place this data in the proper form for transmission through the Public Switched Telephone Network. This processing basically consists of the modulation of the baseband information (logical ones and zeros (bits) sent by the DTE) into the passband of the communications channel for transmission. The receiver collects the information from the telephone network and transforms it back to its original form, i.e., the bits sent by the DTE.

Modem Transmitter

The Bell 212A/V.22 is a 1200-bps modem that uses the Differential Phase Shift Keying (DPSK) modulation technique to transmit data through the communications channel. In the first part of this subsection, the equation that describes the operation of Differential Phase Shift Keying modulation systems is derived from an intuitive approach. A rigorous derivation is given in Appendix A. The rest of this subsection discusses the functional blocks required to correctly implement this equation.

In Differential Phase Shift Keying, the information is encoded as the phase change of the transmitter carrier. With $\phi(n)$ denoting the phase that contains the information to be transmitted, the transmitted signal $s(n)$ is represented mathematically by

$$s(n) = A \cos(\omega n + \phi(n)) \quad (1)$$

where ω is the carrier frequency. The parameter A determines the amplitude of the transmitted signal. Use of the trigonometric identity

$$\cos(X + Y) = \cos(X) \cos(Y) - \sin(X) \sin(Y)$$

gives

$$s(n) = A \{ \cos(\omega n) \cos[\phi(n)] - \sin(\omega n) \sin[\phi(n)] \} \quad (2)$$

The substitution of

$$I(n) = A \cos[\phi(n)]$$

$$Q(n) = -A \sin[\phi(n)]$$

into (2) results in (3) used to describe DPSK modulation systems.

$$s(n) = I(n) \cos(\omega n) + Q(n) \sin(\omega n) \quad (3)$$

From (3) it can be seen that the transmission of the baseband sequence $\{I(n), Q(n)\}$ is accomplished by using two separate modulation carriers, a sine wave and a cosine wave. These waves are orthogonal; i.e., the information in the direction of the one wave (cosine) is independent of the information in the direction of the other wave (sine), and therefore this information is recoverable. Each value of the $\{I(n), Q(n)\}$ sequence corresponds to one signaling element (symbol) transmitted. The number of signaling elements transmitted per second is commonly referred to as the baud rate, which for the Bell 212A/V.22 is set by the protocol to 600.

Some widely used terminology becomes apparent from (3). The baseband sequence that modulates the cosine wave is called the In-phase sequence. The baseband sequence that modulates the sine wave is called the Quadrature-phase sequence since the sine-wave carrier is 90 degrees (one Quadrant) out-of-phase from the cosine-wave carrier. The part of the transmitter/receiver that processes the In-phase sequence is commonly referred to as the I-channel, while the part of the transmitter/receiver that processes the Quadrature-phase sequence is referred to as the Q-channel.

The derivation of (3) indicates that the incoming sequence $d_s(n)$ is encoded into the sequence $\{I(n), Q(n)\}$, and the latter is transmitted. The mapping rule used is unique for each system; i.e., the mapping rule used for the Bell 212A/V.22 is different from the mapping rules used for other modems (V.22 bis, V.27, V.29, etc.). For example, for the Bell 212A/V.22, the sequence $\{I(n), Q(n)\}$ contains phase information, while for the V.22 bis, it contains phase and amplitude information. The set of possible values of the sequence $\{I(n), Q(n)\}$ determines the signal constellation, which is given in a two-dimensional representation.² The signal constellation, commonly referred to as the constellation diagram, is a geometric picture that emphasizes the fact that the two channels are 90 degrees (Quadrature) out-of-phase.

The Bell 212A/V.22, with a 600-baud rate, accomplishes the transmission of 1200 bps by encoding two incoming bits (dibit) in a single baud. Since there are four possible values for every dibit, the constellation diagram for the Bell 212A/V.22 contains four points. Each constellation point, i.e., each value of the $\{I(n), Q(n)\}$ sequence, corresponds to a total phase value to be transmitted. The calculation of the total phase from the incoming dibits will be discussed later. Figure 2 shows the constellation diagram for the Bell 212A/V.22. The four constellation points, notated A, B, C, and D, lie on a circle. Since there is no amplitude information transmitted, the radius of this circle is normalized to unity. The total phase information represented by each constellation point is enclosed in parentheses.

The encoding of the incoming sequence $d_s(n)$ into the values of the sequence $\{I(n), Q(n)\}$ is implemented by the encoder. The encoder is a one-input, two-output functional block, whose function is to map every two incoming bits (dibit) of the incoming sequence $d_s(n)$ to a total phase. The total phase is then represented by the values of the sequence $\{I(n), Q(n)\}$, and the latter is transmitted. The mapping rule used to encode the total phase into the values of the coder outputs $\{I(n), Q(n)\}$ is shown in Table 2. Each $\{I, Q\}$ entry in this table corresponds to one point in the constellation diagram of Figure 2. This is indicated in the third column of Table 2.

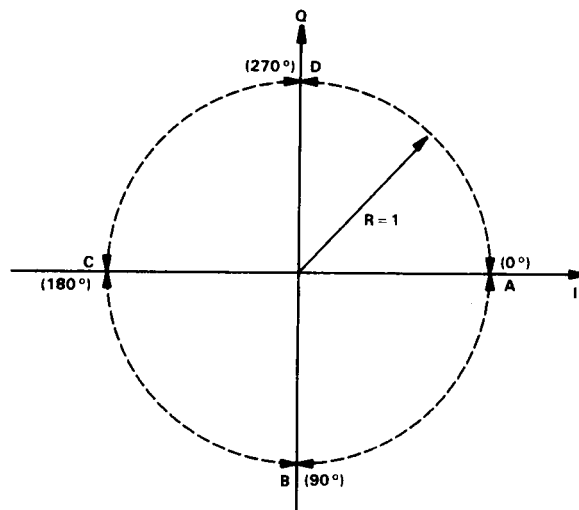


Figure 2. Signal Constellation for the Bell 212A/V.22

Table 2. Total-Phase-to-Coder-Output Mapping Rule

Total Phase	Encoder Output { I , Q }	Point in Constellation Diagram of Figure 2
0 degrees	{ 1 , 0 }	A
90 degrees	{ 0 , -1 }	B
180 degrees	{ -1 , 0 }	C
270 degrees	{ 0 , 1 }	D

The calculation of the total phase from the incoming dibits is accomplished in two steps. First, each incoming dibit is mapped to a unique phase change. Second, this phase change is added to the previous total phase to obtain the new total phase. The mapping rule used to uniquely map each dibit to a phase change is shown in Table 3.

Table 3. Dibit-to-Phase Change Correspondence

Dibit	Phase Change
00	90 degrees
01	0 degrees
10	180 degrees
11	270 degrees

To illustrate the two-step procedure used to calculate the total phase, consider the following example. The previous total phase is 90 degrees, and the incoming dibit is 10. From Table 3, the phase change corresponding to a 10 dibit is 180 degrees. Therefore, the new total phase is

$$\begin{aligned} \text{new total phase} &= \text{modulo } 360 (\text{previous total phase} + \text{phase change}) \\ &= (90 \text{ degrees}) + (180 \text{ degrees}) = 270 \text{ degrees} \end{aligned}$$

Using Table 2, for this value of total phase (270 degrees), the encoder output is $\{I, Q\} = \{0, 1\}$. For the next incoming dibit, the above procedure is repeated with a 270-degree previous total phase.

At the receiver, the total phase is determined from the received $\{I, Q\}$ value. This total phase is subtracted from the previous total phase (the one transmitted during the previous baud), and the difference is the phase change. Since the phase-change-to-dibit mapping is unique, using the calculated value of the phase change results in the transmitted dibit being uniquely recovered at the receiver.

This differential approach (i.e., the calculation of the phase change instead of an absolute phase) is used because if the dibits were to correspond to an absolute phase, then a common-phase reference for both the receiver and the transmitter would be required. This in turn implies the need of a training sequence between the transmitter and the receiver so that a common-phase reference can be established. This training sequence, however, is not provided for the Bell 212A/V.22.

An overall block diagram for the modem transmitter is shown in Figure 3.3 The basic structural blocks are the scrambler, encoder, digital lowpass filter, and digital modulator.

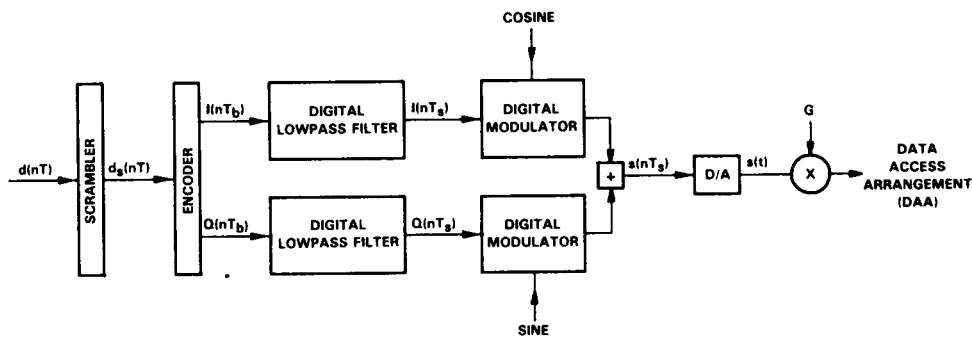


Figure 3. Modem Transmitter Block Diagram

Scrambler

The scrambler scrambles the bits sent by the DTE. To understand the need for a scrambler, consider the situation where the DTE sends a series of 01 dibits. From Table 3, each 1 dibit corresponds to a 0-degree phase change. Therefore, the total phase transmitted is the same. From the geometrical point of view, this results in transmitting the same constellation point (same total phase). At the receiver end, however, phase changes are required for correct clock recovery (see the clock recovery discussion in the Modem Receiver subsection). Therefore, the transmission of a series of 01 dibits generates problems for the receiving modem, such as losing carrier lock. To avoid this, the scrambler is introduced to minimize the probability that such 'ill-conditioned' dibits occur. With $d(nT)$ input to the scrambler, the output $d_s(nT)$ is given by

$$d_s(nT) = d(nT) \text{ XOR } d_s((n-14)T) \text{ XOR } d_s((n-17)T) \quad (4)$$

where XOR indicates the exclusive-OR operation and T is the data period, i.e., the time between two successive bits sent by the Data Terminal Equipment. The signal flowgraph of the modem transmitter scrambler is shown in Figure 4 in which z^{-n} is used to indicated an n -sample delay.

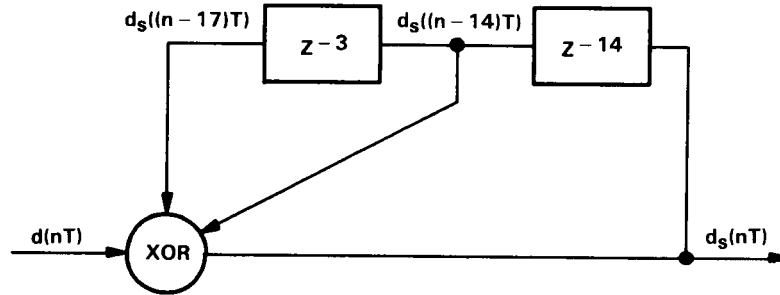


Figure 4. Signal Flowgraph of Transmitter Scrambler

Encoder

The function of the encoder, i.e., the mapping of the incoming sequence $d_s(n)$ to the values of the sequence $\{I(n), Q(n)\}$, was discussed earlier. However, there is one more related issue associated with the encoder, i.e., the change of the sampling frequency at the encoder output. Every two bits that the modem transmitter accepts from the DTE correspond to a unique phase to be transmitted. Therefore, at the encoder output, the sampling period changes from T (sampling period of incoming data) to T_b , i.e., from $1/1200$ s to $1/600$ s. The subscript b in T_b represents baud since the encoder output (I and Q channels) changes at the baud rate. The above discussion implies that $T_b = 2T$; i.e., the I and Q channels are updated after every pair of bits received from the DTE.

Digital Modulators and Lowpass Filters

Since the telephone network behaves as a bandpass filter with the passband starting around 300 Hz and ending around 3200 Hz, the baseband encoder outputs, $I(nT_b)$ and $Q(nT_b)$, cannot be directly transmitted through the communications medium. They first must be modulated up in frequency. The modulation is not attempted directly on the encoder outputs for two reasons. First, as discussed at the end of this subsection, the sampling frequency must increase from $1/T_b$ to $1/T_s$, with $1/T_s$ being at least 6.4 kHz. This increase in the sampling frequency is accomplished by interpolation. Second, if the modulation is attempted directly on the encoder outputs, the instantaneous changes of the $I(nT_b)$ and $Q(nT_b)$ generate higher-order harmonics. Some of these harmonics fall in the frequency region reserved for the receiver. To eliminate the harmonics and to also increase the sampling frequency by interpolation, the encoder outputs must be digitally lowpass-filtered. The characteristics and the implementation of the digital lowpass filters are discussed in detail in the Transmit Filters subsection of "Functions Implemented in the TMS32010."

At the output of the lowpass filters, the I-channel modulates a cosine wave and the Q-channel a sine wave. The modulating frequency is 1200 Hz for an originate modem and 2400 Hz for a answer modem. Finally, the two channels are summed before they are transformed into the analog signal transmitted through the telephone network. The output of the digital transmitter (before the D/A converter) is given by equation (3), repeated below for convenience.

$$s(nT_s) = I(nT_s) \cos(\omega nT_s) + Q(nT_s) \sin(\omega nT_s)$$

The sampling period T_s is $T_s = 1/f_s$ where f_s is the sampling frequency. This frequency must be at least twice the highest frequency component of the transmitted information (Nyquist rate) to satisfy the sampling theorem. Since the telephone network cuts off at approximately 3.2 kHz, the sampling frequency must be at least 6.4 kHz. Practical considerations (integer number of samples per baud, etc.) impose the necessity of higher sampling rates. For the present application, the sampling frequency used was 9.6 kHz. Since the baud frequency is 600 Hz, 16 (9600/600) samples correspond to each baud interval.

Modem Receiver

This subsection discusses the issues associated with the functional blocks required to implement a Bell 212A/V.22 modem receiver. The receiver structure is more sophisticated than that of the transmitter. For a low bit-error-rate performance (percentage of error bits received), an Automatic Gain Control (AGC) subsystem, adaptive equalization of the overall transmitting system, and a noise-independent carrier recovery and clock recovery are required. Since the issues associated with the carrier recovery and the clock recovery are critical in a modem design and difficult to understand, a good portion of this subsection is devoted to their discussion.

The adaptive equalizer is an adaptive filter that compensates for intersymbol interference and Doppler spread effects introduced during transmission over the telephone lines.⁴ The magnitude of these effects depends on the bit rate and the quality of the telephone line. The effects are more severe at high bit rates (2400 bps and above) and over a worst-case telephone line, which is commonly represented by the 3002 line simulator.⁵ The Bell 212A/V.22 protocol does not require the presence of an adaptive equalizer; therefore, this implementation does not include one. However, for increased performance on a 3002 line where even at medium speeds, such as 1200 bps, intersymbol interference and Doppler spread effects become severe, an adaptive equalizer is recommended. An important point here is that the addition of an adaptive equalizer in the current TMS32010 implementation of the Bell 212A/V.22 modem does not require any hardware changes. Increased performance results from an increase in the algorithmic sophistication.

An overall block diagram of the modem receiver is shown in Figure 5. The basic structural blocks of the modem receiver are the input bandpass filters, the automatic gain control (AGC), the demodulator, the decision block, the decoder, the descrambler, the carrier recovery, and the clock recovery.

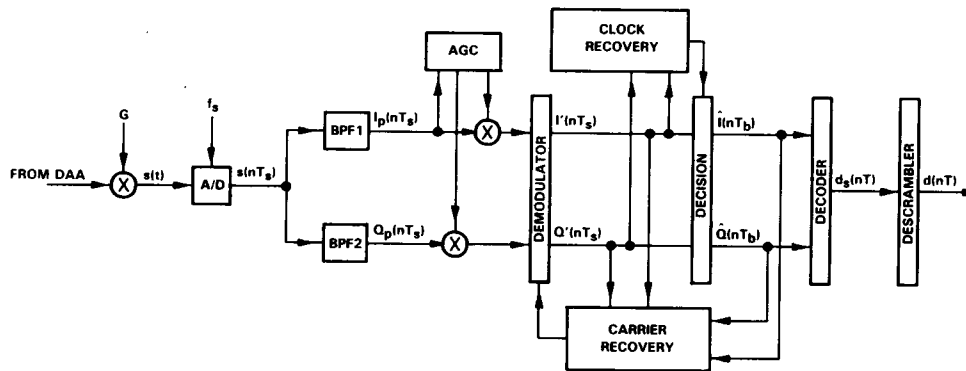


Figure 5. Modem Receiver Block Diagram

Input Bandpass Filters

The incoming analog signal $s(t)$ is digitized at the sampling frequency f_s to obtain its digital counterpart $s(nT_s)$. This signal is then bandpass-filtered for three reasons:

1. Rejection of out-of-band noise, including the rejection of the transmit signal spectrum due to the near-end echo path,
2. Introduction of 90-degree relative phase shift required for the I and Q channel separation (see Appendix A), and
3. Fixed equalization for line distortion.

The second reason mentioned above implies the need of receiver bandpass filters that achieve a 90-degree relative phase shift. It is theoretically justified in Appendix B that if the two bandpass filters, denoted by BPF 1 and BPF 2 in Figure 5, achieve an exact 90-degree relative phase shift, there are no harmonics at the output of the receiver demodulator. If this condition is not met, harmonics appear at twice the carrier frequency. These harmonics were observed in the modem implementation when a set of bandpass filters not meeting the above condition was used. Elimination of the harmonics due to an inexact 90-degree relative phase shift involves the use of lowpass filters at the output of the demodulator. However, the group delay and the possible phase distortion introduced by the lowpass filters affect the carrier recovery and decision algorithms. Compensation for these side-effects of the lowpass filters results in a more complicated modem receiver design.

In the analog domain, where component drift is due to aging and/or temperature, it is virtually impossible to design bandpass filters or Hilbert transformers that achieve an exact 90-degree relative phase shift. Hilbert transformers, a special class of filters, are discussed in Appendices A and B. In the digital domain, however, the design of bandpass filters or Hilbert transformers that achieve an exact 90-degree relative phase shift is relatively easy. Digital filter design packages, such as the Digital Filter Design Package (DFDP) offered by the Atlanta Signal Processors Incorporated (ASPI), can be used to design modem receiver input filters on the TMS32010 that meet the exact 90-degree relative phase shift requirement. The characteristics and implementation of the modem receiver input bandpass filters are discussed in detail in the Receive Filters subsection of "Functions Implemented in the TMS32010."

Automatic Gain Control (AGC)

Because of the attenuation introduced by the telephone lines, the peak-to-peak voltage of the incoming analog signal $s(t)$ ranges between 2 mV and 700 mV. However, signal levels in the receiver must be independent of the attenuation introduced by the communications channel. This is of paramount importance because the carrier recovery and clock recovery algorithms use error signals and thresholds dependent on the I and Q channel values. Therefore, the Automatic Gain Control subsystem is required to adjust the envelope of the I and Q channels so that they are of the same magnitude. The AGC algorithm used and its implementation is discussed in the Automatic Gain Control Implementation subsection of "Functions Implemented in the TMS32010."

Demodulator

The demodulator translates the passband information back to the baseband. With $I_p(nT_s)$ and $Q_p(nT_s)$ inputs to the demodulator (see Figure 5), the outputs $I'(nT_s)$ and $Q'(nT_s)$ are given by (see derivation in Appendix A)

$$I'(nT_s) = I_p(nT_s) \cos(\omega' nT_s) + Q_p(nT_s) \sin(\omega' nT_s) \quad (5)$$

$$Q'(nT_s) = I_p(nT_s) \sin(\omega' nT_s) - Q_p(nT_s) \cos(\omega' nT_s) \quad (6)$$

where ω' is the local carrier frequency. Figure 6 shows the demodulator structure that implements (5) and (6).

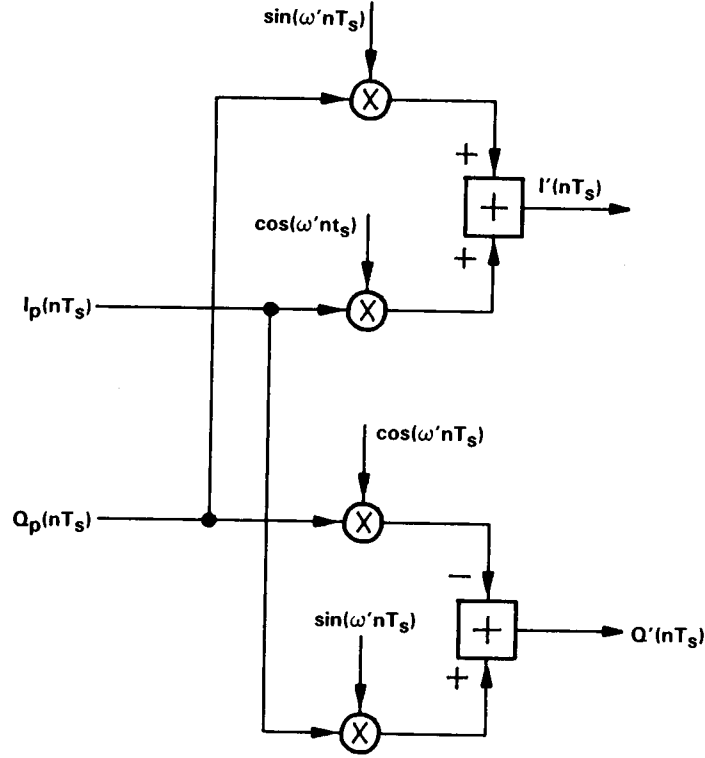


Figure 6. Demodulator Structure

Even with an ideal receiver, the $I'(nT_s)$ and $Q'(nT_s)$ channels shown in Figure 6 are 'noisy' replicas of the baseband I and Q channels at the output of the transmitter digital lowpass filters. The 'noise' has been injected by the telephone network as group delay, frequency jitter, and Gaussian noise.⁴

Decision Block and Descrambler

The decision block in Figure 5 calculates the total phase from the values of the baseband I and Q channels. By subtracting it from the previous total phase (the phase transmitted during the previous baud interval), the phase change is computed. Each phase change (total of four) has a corresponding unique dibit (see Modem Transmitter subsection). This dibit is fed into the descrambler (see Figure 5) to recover the originally transmitted dibit. The output of the descrambler is described by

$$d(nT) = d_s(nT) \text{ XOR } d_s((n-14)T) \text{ XOR } d_s((n-17)T) \quad (7)$$

where T is the data period (1/1200 s for the Bell 212A). The signal flowgraph of the receiver descrambler is shown in Figure 7.

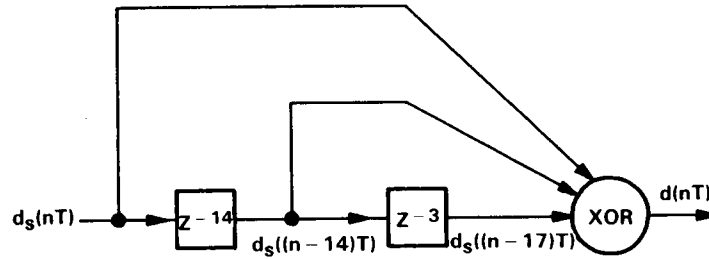


Figure 7. Signal Flowgraph of Receiver Descrambler

Carrier Recovery

A very important task of the modem receiver is the generation of a carrier that has the same frequency and phase with the incoming carrier. This receiver-generated carrier, called the local carrier, is used by the demodulator of Figure 6 to demodulate the incoming signal and therefore retrieve the baseband information. The process of generating this carrier is called carrier recovery. The standard approach to this is to use a phase-locked loop.⁶ Figure 8 shows the basic blocks of a phase-locked loop: the phase detector (PD), loop filter and Voltage Controlled Oscillator (VCO).

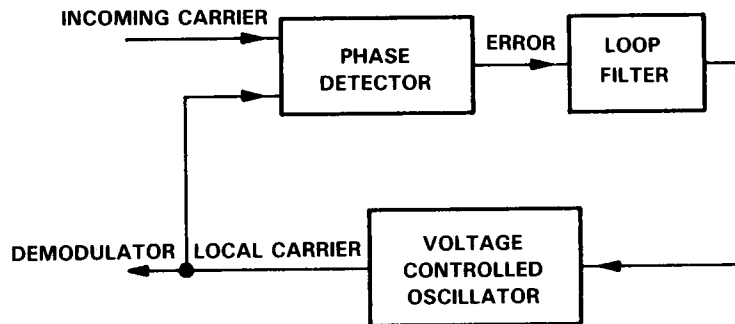


Figure 8. Carrier Recovery Phase-Locked Loop

For a microprocessor implementation, the blocks in Figure 6 are implemented digitally. The digital implementation is discussed in the Carrier Recovery Implementation subsection of "Functions Implemented in the TMS32010." Only the issues associated with the carrier recovery phase-locked loop are considered here.

The phase detector (PD) generates an error signal that is used to synchronize the local carrier to the incoming carrier. This error signal must contain the information about the phase and frequency difference between the local and the incoming carriers. To implement the correct carrier recovery algorithm, it is critical to know the exact dependence of the phase detector output on the frequency and phase difference between the two carriers (discussed later in this subsection). The phase detector output is of the form^{7,8}

$$E(nT_b) = \hat{Q}(nT_b) I'(nT_b) - \hat{I}(nT_b) Q'(nT_b) \quad (8)$$

where \hat{I} and \hat{Q} are the I and Q channel decisions and T_b is the baud period (1/600 s). If the decisions are correct, then

$$\hat{Q}(nT_b) = Q(nT_b) \quad (9)$$

$$\hat{I}(nT_b) = I(nT_b) \quad (10)$$

i.e., the outputs of the transmitter coder (see Figure 3) have been successfully recovered. The probability that these decisions are correct is maximum in the middle of each baud because the incoming signal energy is maximum here. Based on the error signal $E(nT_b)$, the local carrier is corrected once every baud, i.e., at a 600-Hz frequency. Geometrically, the error $E(nT_b)$ is a measure of the geometrical distance between the point used to make the decision and the optimum one. The optimum decision points are the constellation points. It is shown later that when the local carrier has the same phase and frequency with the incoming carrier, the error $E(nT_b) = 0$. In this case, the point used to make the decision coincides with a constellation point. The optimality of the receiver constellation points is discussed next.

Optimality in the receiver, in terms of low probability of error, is determined only by the geometrical distance between the constellation points.⁹ The four constellation points of Figure 2, notated as A, B, C, and D, are optimum. The following intuitive argument helps to illustrate this. The four points lie on a circle of normalized unity radius. In the configuration of Figure 2, point A is equidistant from points B and D. This means that the probability of error p when deciding between points A or B, i.e., deciding point A when point B is correct and vice versa, is equal to the probability of error when deciding between points A or D. If point A moves counterclockwise, it moves away from point B but closer to point D. Since at the new location, point A is farther from point B, the probability of error p_1 when deciding between points A or B decreases, i.e., $p_1 < p$. However, at this new location, point A is closer to point D, and therefore, the probability of error p_2 when deciding between points A or D increases, i.e., $p_2 > p$. Using the analytical tools discussed in [9], it can be shown that $p_1 + p_2 > 2p$. Since the overall probability of error increases ($p_1 + p_2 > 2p$) if point A moves away from the location indicated in Figure 2, the resulting structure is no longer optimum. This is not true, however, if all four constellation points are equally rotated by an arbitrary amount in the clockwise or counterclockwise direction. Therefore, an infinite set of constellation points that preserve optimality in the receiver exist. The final choice depends on implementation considerations.

For the modem implementation described in this report, two considerations lead to a 45-degree rotation (see Figure 9) of the transmitter constellation diagram of Figure 2.

1. For the constellation points of Figure 9, the decision boundaries are the I and Q axes. That is, the decision region for point A is the first quadrant, the decision region for point D the second quadrant, and so on. Therefore, a decision can be made based only on the sign of the demodulated I ($I'(nT_s)$) and Q ($Q'(nT_s)$) channels.
2. For this set of constellation points, the products $\hat{Q}(nT_b) I'(nT_b)$ and $\hat{I}(nT_b) Q'(nT_b)$, required to calculate the phase error $E(nT_b)$ (see equation (8)), obtain on the average maximum values. Therefore, an optimum utilization of the dynamic range is achieved, and the error function calculated by (8) is the least-noise sensitive.

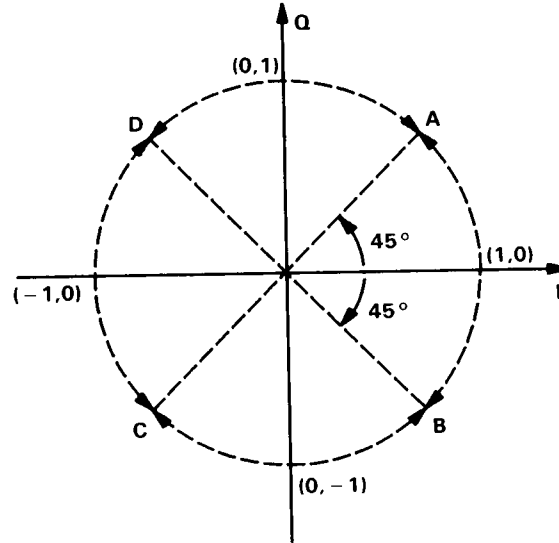


Figure 9. Modem Receiver Decision Points

The error $E(nT_b)$, the output of the phase detector, as given by (8) shows no apparent dependence on the phase or frequency difference between the local and incoming carriers. The discussion that follows shows the dependence of $E(nT_b)$ on the phase difference between the two carriers. This discussion is then extended to include the case of frequency as well as phase difference.

The inputs $I_p(nT_s)$ and $Q_p(nT_s)$ of the receiver demodulator (see Figure 6) are given by (see Appendix A)

$$I_p(nT_s) = I(nT_s) \cos(\omega nT_s + \theta_r) + Q(nT_s) \sin(\omega nT_s + \theta_r) \quad (11)$$

$$Q_p(nT_s) = I(nT_s) \sin(\omega nT_s + \theta_r) - Q(nT_s) \cos(\omega nT_s + \theta_r) \quad (12)$$

where ω and θ_r are the incoming (received) carrier frequency and phase, respectively. The outputs $I'(nT_s)$ and $Q'(nT_s)$ of the demodulator are described by equations (5) and (6), respectively. Introducing an arbitrary phase θ_l in the local carrier, (5) and (6) can be rewritten as

$$I'(nT_s) = I_p(nT_s) \cos(\omega' nT_s + \theta_l) + Q_p(nT_s) \sin(\omega' nT_s + \theta_l) \quad (13)$$

$$Q'(nT_s) = I_p(nT_s) \sin(\omega' nT_s + \theta_l) - Q_p(nT_s) \cos(\omega' nT_s + \theta_l) \quad (14)$$

where ω' is the local carrier frequency.

Assuming no frequency difference ($\omega' = \omega$), substitution of (11) and (12) into (13) and (14) gives

$$I'(nT_b) = I(nT_b) \cos(\theta_e) + Q(nT_b) \sin(\theta_e) \quad (15)$$

$$Q'(nT_b) = Q(nT_b) \cos(\theta_e) - I(nT_b) \sin(\theta_e) \quad (16)$$

where $\theta_e = \theta_r - \theta_l$ is the phase difference between the two carriers. Note that if $\theta_r = \theta_l$, then $\theta_e = 0$. From (15) and (16),

$$I'(nT_b) = I(nT_b)$$

$$Q'(nT_b) = Q(nT_b)$$

i.e., the output of the receiver demodulator at the middle of the baud is the same as the output of the transmitter coder (baseband information).

Assuming the decisions are correct, equations (9) and (10) hold. Substitution of (9), (10), (15), and (16) into the error signal defined by (8) gives

$$E(nT_b) = \{I^2(nT_b) + Q^2(nT_b)\} \sin(\theta_e) \quad (17)$$

The quantity $I^2(nT_b) + Q^2(nT_b)$ is a positive quantity (sum of squares). With $I^2(nT_b) + Q^2(nT_b) = K$,

(17) can be rewritten as

$$E(nT_b) = K \sin(\theta_e) \quad \text{where } K > 0 \quad (18)$$

Equation (18) is the same as (8) under the assumption of correct decisions ((9) and (10)). However, the phase information is more apparent in (18) than in (8), and leads to the following algorithm for the carrier recovery: If the phase of the received carrier is greater than the phase of the local carrier ($\theta_r > \theta_l$), the phase error θ_e is positive ($\theta_e > 0$). From (18), this implies that the output of the phase detector is also positive ($E(nT_b) > 0$). Therefore, if $E(nT_b) > 0$, the phase of the local carrier must be advanced, resulting in a smaller phase error. On the other hand, if the phase of the received carrier is less than the phase of the local carrier ($\theta_r < \theta_l$), the phase error is negative ($\theta_e < 0$). From (18), this implies that the output of the phase detector is also negative ($E(nT_b) < 0$). Therefore, if $E(nT_b) < 0$, the phase of the local carrier must be delayed.

In the case of frequency as well as phase difference, a similar development leads to

$$E(nT_b) = K \sin(\omega_e nT_b + \theta_e) \quad \text{where } K > 0 \quad (19)$$

where $\omega_e = \omega - \omega'$ is the frequency difference between the incoming and local carriers. Since this frequency is very small (on the order of a few Hz) and the phase error correction is applied every baud (600 Hz), the term $\omega_e nT_b$ can be considered to be constant and the term $\omega_e nT_b + \theta_e$ in (19) an overall phase error. Therefore, using the algorithm discussed earlier, the frequency difference is compensated for as phase difference. Also note that in (19), $E(nT_b) = 0$ when $\omega_e = 0$ and $\theta_e = 0$; i.e., the local carrier is completely synchronized with the incoming carrier. Therefore, the error signal $E(nT_b)$ generated by the phase detector contains the information about the frequency and phase difference between the incoming and local carriers.

The error signal $E(nT_b)$ generated by the phase detector is processed by the loop filter as shown in Figure 8. Only the DC and low-frequency components of this signal must drive the Voltage Controlled Oscillator (VCO).⁶ Therefore, the loop filter is basically a lowpass filter, whose most important characteristic is its bandwidth.

A large bandwidth of the loop filter implies that high-frequency components pass through the filter. Since the high-frequency information is applied to the VCO, the local carrier quickly locks-on to the incoming carrier. However, noise also passes through the filter, and the Bit Error Rate (BER) of the receiver increases. A narrow bandwidth decreases the BER but the lock-on time increases. An intelligent solution consists of starting with a wide bandwidth and, after the receiver is locked-on to the incoming carrier, narrow it down. This approach is used in this implementation and is described further in the Carrier Recovery Implementation subsection of "Functions Implemented in the TMS32010."

Clock Recovery

The purpose of the Clock Recovery block in Figure 5 is to detect the middle of each baud. Once this is known, the decision block can make decisions with minimum probability of error because the energy of the incoming signal is maximum at the middle of the baud. The following paragraphs discuss a robust clock recovery approach.

As the demodulation point moves from one constellation point to another, at least one of the two channels is expected to cross zero (see Figure 9). This zero crossing indicates the beginning of a new baud interval. Therefore, one approach is to look at the zero crossings of the $I'(nT_s)$ and/or $Q'(nT_s)$ channels. However, there are two problems with that approach:

1. From (15) and (16), it can be seen that the presence of a phase difference θ_e between the two carriers can cause severe distortion of the zero crossings. To illustrate this point, consider the first of the two equations, repeated here for convenience.

$$I'(nT_b) = I(nT_b) \cos(\theta_e) + Q(nT_b) \sin(\theta_e)$$

The correct zero crossing information lies in $I(nT_b)$. Multiplication by $\cos(\theta_e)$ scales the $I(nT_b)$ curve, but does not change the location of the zero crossings. This is accomplished by the second additive term $Q(nT_b) \sin(\theta_e)$, which moves the scaled curve up or down depending on the term's sign.

2. The quantization noise in a digital implementation may result in undesirable nonlinearities and mislocation of the zero crossings. This is because finding the zero crossings involves monitoring the change of the sign of a particular variable (I channel and/or Q channel). A zero crossing occurs when this variable changes from a small positive value to a small negative value, and vice versa. Since the quantization noise can seriously affect small quantities (numbers), mislocation of the zero crossings may result.

The first of the above problems indicates that a clock recovery approach is required that is independent of the phase or frequency difference between the two carriers. This becomes clearer by considering the operation of the modem. The first task that the receiver must perform is to adjust the baud clock. During this adjustment, the two carriers are most likely to have a phase and/or frequency difference. Then, once the baud clock is adjusted, the carrier recovery algorithm places the local carrier in phase and in frequency with the incoming carrier.

Consider the energy of the incoming signal

$$\text{Energy} = I'^2(nT_s) + Q'^2(nT_s) \quad (20)$$

Substitution of (15) and (16) into (20), and the use of the identity

$$\sin^2(\theta_e) + \cos^2(\theta_e) = 1$$

gives

$$\text{Energy} = I^2(nT_s) + Q^2(nT_s) \quad (21)$$

This is the energy sent out by the transmitting modem. Equation (21) shows that the energy is independent of any phase and/or frequency difference between the two carriers. Geometrically, the energy is the square of the length of the vector that has its beginning at the intersection of the I and Q axis of Figure 9 and its tip at the demodulation point plotted on the constellation diagram. The path traced by the tip of the energy vector for a series of four consecutive baud intervals, each corresponding to a 90-degree phase change, is shown in Figure 10.

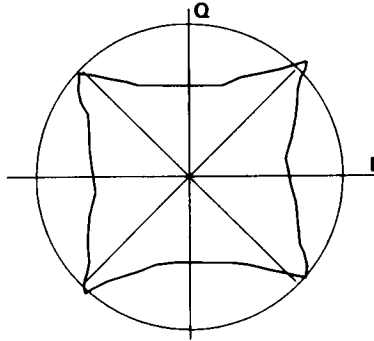


Figure 10. Trace of Demodulation Point Plotted on the Constellation Diagram

The plot shown in Figure 10 was obtained using a simulator. It can be seen that the signal energy ($I^2 + Q^2$) achieves its maximum value at the middle of each baud. Before and after the middle of the baud, the length of this vector is less than maximum. If there is a transition from one quadrant to another, this vector goes through a minimum, thus indicating the beginning of a new baud interval. Only if the same constellation point is transmitted because of a zero-degree phase change does such a transition not occur. It is easy to explain now why a series of zero-degree phase changes can create problems for the receiver. Zero-degree phase changes imply that the transmitter keeps sending the same constellation point. Therefore, the energy vector at the receiver does not go through a minimum for a series of baud intervals; i.e., during these intervals the receiver cannot adjust the baud clock and therefore may lose lock. This situation is avoided with the inclusion of the scrambler in the transmitter structure.

The frequency of the energy minima is discussed next. From Figure 9, it can be seen that there are four possible transitions for each constellation point. For example, consider constellation point A. The four possible transitions are: from point A to B, from A to C, from A to D, and from point A to A (i.e., receiving a zero-degree phase change). Three out of the four possible transitions result in a quadrant change (transitions from point A to points B, C, or D). For these transitions, the energy vector goes through a minimum. The fourth transition (from point A to itself), does not result in a quadrant change, but due to the presence of the scrambler, the probability of its occurrence is less than 0.25 (one out of four). Therefore, the average frequency of these minima is greater than 450 Hz for a 600-Hz baud frequency.

For the baud clock adjustment, the advantages of the energy-based approach over the zero crossings-based approach are:

1. The energy-based approach is independent of the phase and frequency difference between the two carriers, and therefore it gives the correct information about the incoming baud boundaries.

2. The average frequency of the energy minima is greater than 450 Hz while the average frequency of the zero crossings of the I or Q channels is between 300 and 400 Hz. The explanation follows. In the four possible transitions for each constellation point, two of them result in a zero crossing for a particular channel. Considering, for example, the transitions of constellation point A of Figure 9, the transitions from point A to points C or D result in a zero crossing for channel I. The transitions from point A to points B or C result in a zero crossing for channel Q. This implies that for a baud frequency of 600 Hz, the frequency of the zero crossings of a particular channel (I or Q) is on the average 300 Hz (two out of four). Because of the scrambler, the probability of retransmitting the same constellation point (zero-degree phase change) is minimized. This implies that on the average the frequency of the zero crossings of a particular channel increases. In the limit (no zero-degree phase changes), the average frequency of the zero crossings approaches 400 Hz (two out of three). Therefore, the average frequency of the zero crossings of a particular channel is between 300 and 400 Hz. To obtain more information from the zero crossings (greater average zero crossings frequency), the zero crossings of both the I and Q channels must be considered. However, this approach involves monitoring two quantities (I channel and Q channel) compared to monitoring only one (energy) if the energy-based approach is used.
3. Using the energy-based clock recovery technique described in the Baud Clock Alignment Implementation subsection of "Functions Implemented in the TMS32010," the quantization noise effects are less severe compared to those of a zero crossing-based approach.

Modem Hardware Description

A brief description of the hardware used for the implementation of the Bell 212A/V.22 modem is covered in this section. Most of the signal processing required for the implementation of the modem functional blocks described in the previous section is performed digitally by the TMS32010 digital signal processor (see "Functions Implemented in the TMS32010"). The DTE interface and the protocol are handled by the TMS7742^{10,11}, an 8-bit EPROM microcomputer with an on-chip UART. Therefore, the hardware required for the system is minimal and consists primarily of the TMS32010 and TMS7742 processors, their memory, the A/D and D/A converters, and the associated antialiasing and smoothing filters.

To aid in the development and prototyping of this project, off-the-shelf development tools were used to build the modem hardware. The TMS32010 and the TMS7742 were emulated using Extended Development Systems (XDS) (part # TMDS3262211 for the TMS32010, and part # TMDS7062230 for the TMS7742). For the A/D and D/A conversions, the TMS32010 Analog Interface Board (AIB) (part # RTC/EVM320C/06) was used. The Cermetek CH1810, Data Access Arrangement (DAA) approved by the Federal Communications Commission (FCC), is used for the telephone-line interface. A block diagram of the modem system hardware is shown in Figure 11.

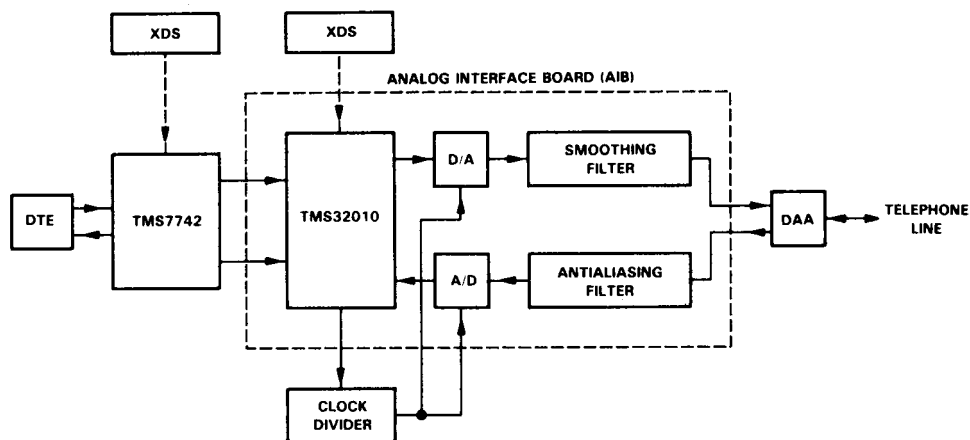


Figure 11. Modem Hardware Block Diagram

Analog Interface

Compensative gain circuits have been placed between the DAA and the analog-to-digital converter. The gain circuit on the receiver side (see Figure 5) was added to match the peak amplitude of the signal from the phone line (-9 dbm or 0.77 V peak-to-peak) with the maximum range of the analog-to-digital converter (10 V peak-to-peak). This allows as much as possible of the A/D's dynamic range to be used without causing saturation. The gain circuit on the transmitter side (see Figure 3) is designed to attenuate the output of the digital-to-analog converter (10 V peak-to-peak) to a level consistent with the phone system signal strength limits (-12 dbm or 0.55 V peak-to-peak).

Filters

The analog antialiasing and smoothing filters used by the A/D and D/A converters are sixth-order lowpass filters existing on the AIB, implemented using cascaded second-order opamp filter sections. These filters are designed with cutoff frequencies around 4.7 kHz in order to satisfy the Nyquist criterion requirements of the system.

Data Converters

Analog Devices' AD565A and ADC80, monolithic A/D and D/A converters on the AIB, are configured for a ± 10 V full-scale range and are interfaced to I/O port 2 of the TMS32010. The sampling rate for the conversions is determined by a set of presettable counters configured as frequency dividers. These counters are driven by the TMS32010's CLKOUT signal and produce a periodic sampling clock that initiates A/D and D/A conversions. The sampling frequency used is 9.6 kHz.

TMS32010/TMS7742 Interface

The TMS32010 interfaces to the TMS7742 in parallel as shown in Figure 12.

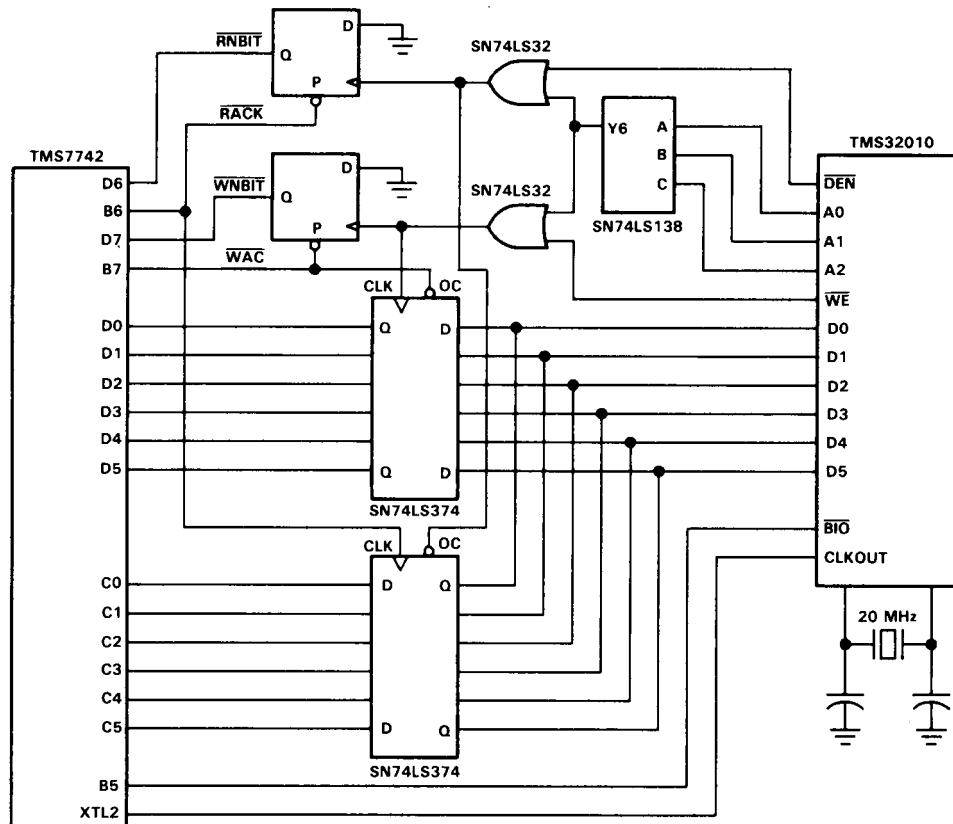


Figure 12. TMS7742 and TMS32010 Interface

The TMS7742 is mapped as an I/O device at Port 6 of the TMS32010. When the TMS32010 writes to Port 6, the WNBIT line goes active (D7). The TMS7742 polls this line and when active, reads data and status bits from the buffer into Port D. It also resets the WNBIT by sending a low pulse to the write acknowledge (WACK) line. Similarly, when the TMS32010 reads from Port 6, the RNBIT line goes active. The TMS7742 immediately writes the new data to Port C and resets the read acknowledge (RACK) line.

Six bits are used to interface the TMS32010 to the TMS7742. Two of these are used to pass the dibit data, two are used to send commands or status, and the other two are reserved to pass additional data if implementing the V.22 bis. The V.22 bis is a 2400-bps splitband modem that uses quad (four) bits instead of dibits. Table 4 lists the commands. The symbol X is used to indicate a don't care condition.

Table 4. Modem Controller Commands

Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Command Description
0	0	X	X	X	X	Idle
0	1	X	X	X	X	Run local digital loopback
1	0	X	X	D1	D0	Run modem
1	1	D3	D2	D1	D0	Configure TMS32010 according to D3-D0

In the idle mode, the TMS32010 continues to monitor the commands from the TMS7742. In the local digital loopback mode, the TMS32010 reads the scrambled dibits from the TMS7742 and sends them back to the TMS7742. In the run mode, the TMS32010 reads the scrambled dibits from the TMS7742 and does the required encoding and modulation for the transmission through the telephone network. It also decodes the demodulated data and sends it to the TMS7742 for descrambling. Bits D0 and D1 are used to carry the dibit information. Bits D2 and D3 can be used when implementing the V.22 bis. In the configuration mode, the TMS32010 configures the transmit and receive filters for the originate or answer mode, depending on the data on D3-D0 (see TMS7742 source code provided in Appendix E).

Functions Implemented in the TMS32010

The functions discussed in this section include all of the functional blocks described in the Modem Transmitter and Modem Receiver subsections with the exception of the transmitter scrambler and the receiver descrambler, which are implemented in the TMS7742. Table 5 shows the modem functions that are implemented on each device.

Table 5. Modem Functions Implemented in the TMS32010 and TMS7742

Modem Transmitter	Implemented
Guard Tone Generator	No
Scrambler	TMS7742
Encoder	TMS32010
Digital Lowpass Filters	TMS32010
Originate Mode Modulator	TMS32010
Answer Mode Modulator	No
DTE Interface	TMS7742
Modem Receiver	Implemented
Notch Filter	No
Originate Mode Bandpass Filters	TMS32010
Answer Mode Bandpass Filters	No
Automatic Gain Control	TMS32010
Demodulator	TMS32010
Decision Block	TMS32010
Decoder	TMS32010
Descrambler	TMS7742
Clock Recovery	TMS32010
Carrier Recovery	TMS32010
DTE Interface	TMS7742

Each variable used in this section is referred to by its name in the TMS32010 program enclosed in parentheses (see Appendix D).

Transmit Filters

The transmit lowpass filters are implemented using 48-tap FIR structures, whose frequency responses exhibit a raised-cosine shape. A raised-cosine response is a filter response whose pass and stopbands are flat and whose rolloff characteristic is defined as a constant times a $(1 + \cos)$ term. The $(1 + \cos)$ term results in the rolloff shape being a portion of a cosine wave raised above the X-axis by one, hence the term 'raised-cosine response'. The raised-cosine response is used since it has been shown that it minimizes the intersymbol interference.¹²

The response shape of the transmit filters is actually defined by the square root of a raised-cosine response since the raised-cosine characteristic is split equally between the transmitter and receiver; i.e., both the transmitter and receiver filters are designed to exhibit the square root of the raised-cosine response. This results in the combined, end-to-end response of the path from transmitter to receiver being the full raised-cosine response.

The frequency-response characteristic of the transmit lowpass filters, as shown in Figure 13, rolls off smoothly to approximately -40 dB at 600 Hz.

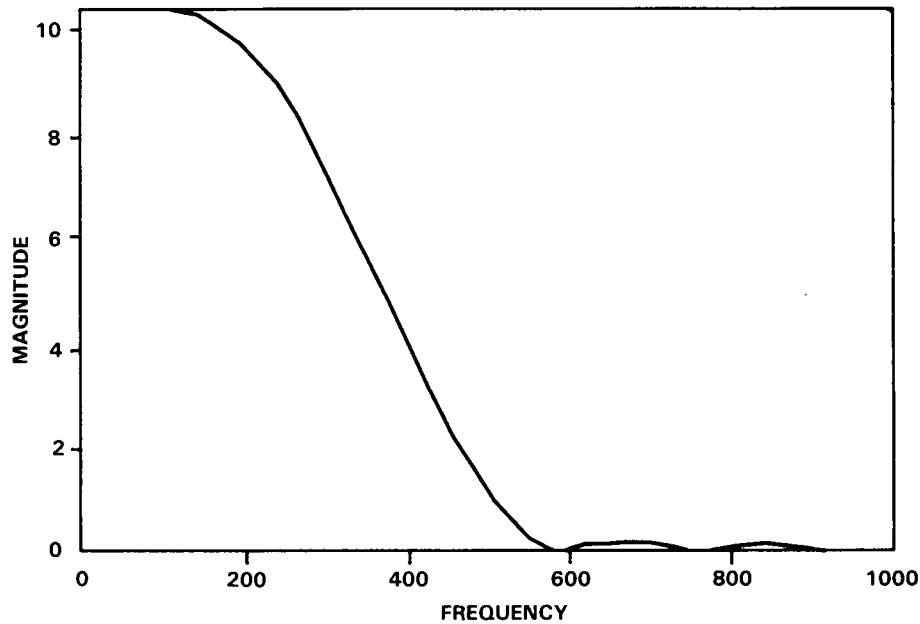


Figure 13. Frequency Response Characteristics of Transmit Lowpass Filters

The FIR structure is well suited to implementation of these filters, because FIR filters are stable, simple in structure, and can be designed to exhibit linear phase. These filters are easily implemented on the TMS32010 since the processor provides special instructions and architectural features that facilitate this type of algorithm. A signal flowgraph of the FIR filter structure is shown in Figure 14.

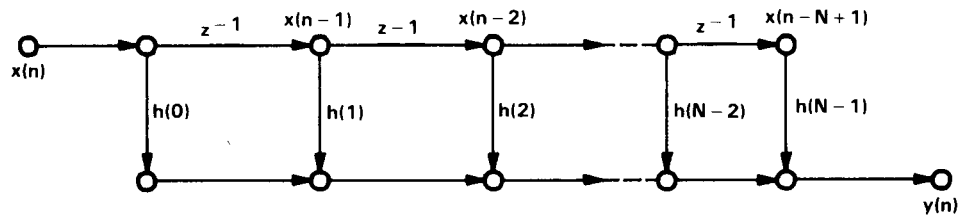


Figure 14. Signal Flowgraph of the FIR Filter Structure

As the flowgraph illustrates, this type of filter uses no feedback, which accounts for its stable behavior. FIR filters implement a transfer function of the form

$$H(z) = B_0 + B_1 z^{-1} + B_2 z^{-2} + B_3 z^{-3} + \dots + B_n z^{-n} \quad (22)$$

The parameters in (22) that determine the characteristics of the specific filter implemented are the B coefficients B_0 - B_n . In the case of the modem filters, the primary task in designing the filter is the determination of these coefficients so that the filter has the desired response shape, in this case, the raised-cosine response shape. For a detailed description of FIR and IIR filter design for the TMS32010, refer to "Implementation of FIR/IIR Filters with the TMS32010/TMS32020," an application report¹³, and to *Digital Filter Design*, a book by T.W. Parks and C.S. Burrus.¹⁴

The raised-cosine response shape is defined by

$$H(f) = \begin{cases} \frac{1}{2 B_t} & |f| < f_1 \\ \frac{1}{4 B_t} \left\{ 1 + \cos \left[\frac{\pi (|f| - f_1)}{2 B_t - 2 f_1} \right] \right\} & f_1 < |f| < 2 B_t - f_1 \\ 0 & |f| > 2 B_t - f_1 \end{cases} \quad (23)$$

where $f_1 = (1 - p) B_t$.

For this design, $B_t = 300$ Hz and $p = 0.75$. Note that (23) describes the ideal zero-phase version of the raised-cosine response.

The actual frequency response of the transmit filters, shown in Figure 13, is the square root of the raised-cosine response described by (16).

To calculate the B coefficients required to implement this response in an FIR filter, the square root of (23) is first calculated. The Inverse Fourier Transform of the response is then used to generate the time-domain representation of the filter transfer function (the impulse response of the filter). In an FIR filter, the impulse response of the filter corresponds directly to the filter coefficients. Therefore, obtaining the coefficients requires merely shifting the impulse response in time to obtain a linear-phase version of the filter, and then sampling the impulse response at the system sampling rate.

After the filter coefficients are obtained, implementation of the filter digitally in the TMS32010 is accomplished by directly translating the signal flowgraph of Figure 14 into assembly language code.

As shown in Figure 14, the output of the filter is defined to be the sum of each of the delayed versions of the input, multiplied by the appropriate coefficient. In the TMS32010, the delayed versions of the previous input samples are stored in a table with the oldest sample stored at the highest address and the newest sample stored at the lowest address.

In the TMS32010 implementation, the transmit filters are arranged in a somewhat different manner from that which is commonly used for digital filters. In many digital filters, the input sample rate is the same as the output sample rate. In the transmit filters, however, the input sample rate is reduced because the rate of change of the information

entering the filter is known to be slower than the filter sample rate. Filters of this type are known as interpolating filters, and the ratio of the output sample rate to the input sample rate is referred to as the interpolation factor. In the modem transmit filters, the input sample rate is 600 Hz (the baud rate), and the output sample rate is 9.6 kHz, resulting in an interpolation factor of 16. As a result, the input of the filter is updated only after every 16 output samples, and is zero otherwise. Thus, the effective input $a(nT_s)$ to the transmit filters can be described for the I channel as

$$a(nT_s) = \begin{cases} I\left(\frac{nT_b}{L}\right) & \text{for } n = 0, \pm L, \pm 2L, \text{ etc.} \\ 0 & \text{otherwise} \end{cases} \quad (24)$$

and for the Q channel as

$$a(nT_s) = \begin{cases} Q\left(\frac{nT_b}{L}\right) & \text{for } n = 0, \pm L, \pm 2L, \text{ etc.} \\ 0 & \text{otherwise} \end{cases} \quad (25)$$

This technique reduces the number of multiplications required to compute the filter output from N to N/L where N is the length of the filter and L is the interpolation factor.

In both the transmit and receive filters, sampling of the nonzero portion of the filter impulse response at the system sample rate results in only 37 taps required for proper implementation of the filters. However, since the transmit filters are interpolating filters with an interpolation factor of 16, 16 taps are processed for each sample of the input. As a result, the number of taps in the filter must be an integer multiple of 16. In this case, 48 actual taps are used.

With a 48-tap filter and an interpolation factor of 16, only three multiplies are required to calculate the output of the filter. Because of this, these filters are coded on the TMS32010 somewhat differently than FIR filters that are not interpolated. In most filters, the data is shifted each time a sample is processed. With interpolation, the data is shifted only when a new input is processed, i.e., every 16 samples. During the remaining samples (when a new input is not being received), instead of shifting the data, a pointer (XPTR) is shifted through the table of coefficients so that effectively the coefficients are shifted. Thus, the complete filter output can be calculated with the following short section of code:

ZAC		* CLEAR ACCUMULATOR
LT	XIBUF2	* LOAD OLDEST SAMPLE
MPY	CX2	* MPY BY COEFF 2
LTD	XIBUF1	* LOAD NEXT SAMPLE
MPY	CX1	* MPY BY COEFF 1
LTD	XIBUF0	* LOAD NEWEST SAMPLE
MPY	CX0	* MPY BY COEFF 0
APAC		* MAKE FINAL SUM
SACH	XIOUT,1	* STORE OUTPUT

This code calculates the output of the I channel filter when a new input sample is being processed. The code that implements the filter output calculation when a new sample is not being input is similar to this code except that LTA instructions are used in place of LTD instructions.

During samples in which new inputs are being received, the inputs and the coefficients are shifted. This results in savings in data RAM space since only three data values must be stored.

Receive Filters

The receiver bandpass filters are implemented using 37-tap FIR structures, which also exhibit a raised-cosine frequency response characteristic. These filters are virtually identical in structure to the transmit lowpass filters, with the exceptions that the cutoff frequencies are different and the receive bandpass filters do not interpolate since the input and output sample rates are the same. Like the transmit lowpass filters, the actual response implemented in these filters is the square root of the raised-cosine response since this response is split equally between the transmit and receive sections. The receive filters are centered around the carrier frequency f_c (1200 Hz for originate and 2400 Hz for answer), and roll off smoothly to approximately -40 dB at $f_c \pm 600$ Hz. The frequency response characteristic of these filters is shown in Figure 15.

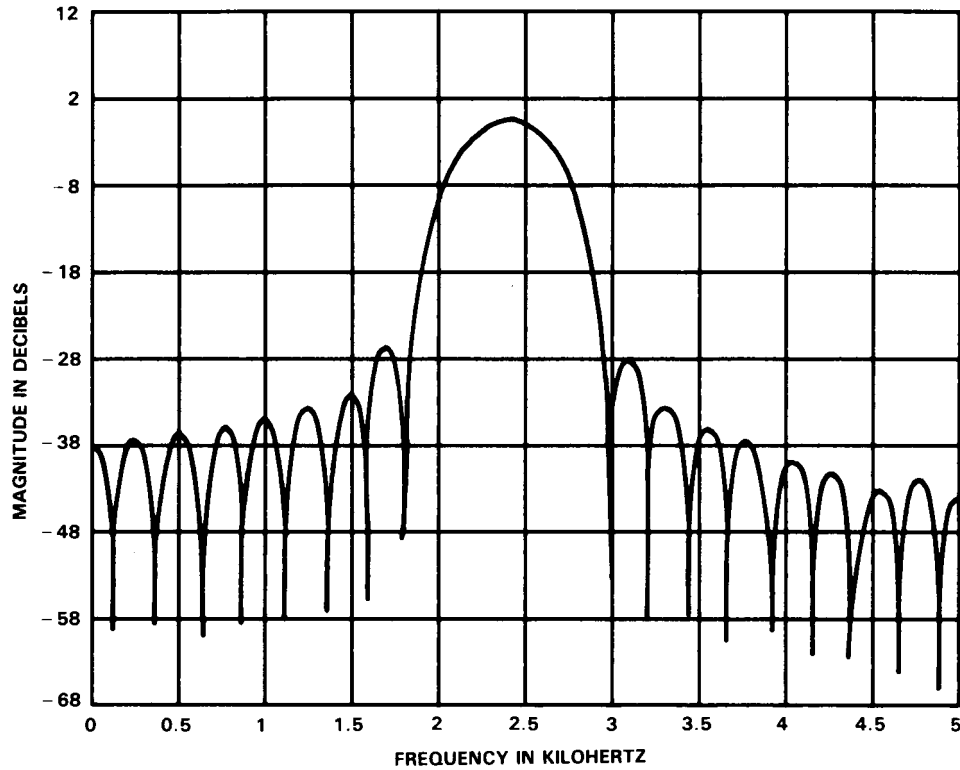


Figure 15. Frequency Response of Receiver Bandpass Filters

Except for the difference in filter order (the number of taps in the filter), the signal flowgraph and transfer function equation for the receive filters are identical to those of the transmit lowpass filters shown in Figure 13 and described by equation (22), respectively.

Besides being similar in structure to the transmit lowpass filters, the receive filters are actually designed directly from the transmit filters by simply shifting the filters' center frequencies. This is possible because the bandwidth of the transmit filters is the same as that required for the receive filters, and the transmit filters exhibit the raised-cosine response also required for the receive filters.

In order to generate the proper coefficients to implement the receive filters, the coefficients of the transmit filter are multiplied by a sine wave to obtain the I channel coefficients and by a cosine wave to obtain the Q channel coefficients. Specifically, if $h(nT_s)$ are the transmit filter coefficients, the receive filter coefficients $h_1(nT_s)$ and $h_2(nT_s)$ are obtained by

$$\begin{aligned}
h_1(nT_s) &= 2 h(nT_s) \cos(n\omega T_s) & (\text{I channel}) \\
h_2(nT_s) &= 2 h(nT_s) \sin(n\omega T_s) & (\text{Q channel})
\end{aligned}
\tag{26}$$

where T_s is the sampling period. Note that the factor of two must be included for the original frequency spectrum to be translated to the new frequency with the same magnitude.

The result of multiplying the transmit filter coefficients by sine and cosine is to effectively modulate their frequency response characteristics by a carrier at the frequency of the sine and cosine waves. This translates the frequency spectrum of the resultant filter up in frequency to a point centered around the frequency of the modulating signal, which is precisely what is required for the receive bandpass filters. Accordingly, bandpass filters for the originate mode are multiplied by sine and cosine functions at 1200 Hz and those for the answer mode are multiplied by sine and cosine functions at 2400 Hz, thus yielding the exact filters required.

In addition to shifting the frequency spectrum of the filters to the appropriate center frequencies, the fact that the I channel filter is multiplied by a sine function and the Q by a cosine function results in another important characteristic of these filters; the outputs of these filters are exactly 90 degrees out of phase with respect to each other. This provides a convenient method for implementing the phase shift required for proper demodulation of the I and Q channels. Also, since the filters are symmetric FIR structures, their phase response is linear, and the difference in phase shift between the two filters is precisely 90 degrees. This is beneficial because deviations from a precise 90-degree phase shift can cause serious distortion in other parts of the modem receiver.

In a direct implementation of this type of filter on the TMS32010, the filter output is calculated by repeatedly using the following two-instruction sequence:

LTD	* LOAD T, ACCUMULATE, DATA SHIFT
MPY	* MULTIPLY BY NEW COEFFICIENT

This sequence performs the following four operations:

1. Loads the T register with the input value,
2. Multiplies the input value by the appropriate coefficient,
3. Adds the product to the accumulator, and
4. Shifts the input data one place in the table, making room for the next input sample.

For FIR filters, a sequence of pairs of the LTD and MPY instructions is all that is required to implement the complete filter.

In the TMS32010, the receive filters are implemented in a somewhat more conventional manner than the transmit filters. The receive filters do not interpolate; however, due to careful choice of sample points on the impulse response, every second

coefficient in each filter is zero, reducing by a factor of two the number of LTD/MPY instruction pairs that must be executed to calculate the filter output.

Another feature of the FIR structure, which simplifies the implementation of these filters, is that since there is no feedback, the delay path $(x(n-1), x(n-2), \dots)$, in Figure 14) for the two filters contains the same values for input samples in data RAM. Because of this, the same delay path can be used for both the I and the Q channel filters. This reduces by a factor of two the RAM required for data storage. As a result, the code that implements the I channel filter (processed first) uses LTA instructions instead of LTDs, performing no shift of the input data table within memory. A one-position shift of the data table is then performed when the Q channel filter output is calculated.

Even though every other coefficient is zero, each sample in the delay table must still be shifted by one memory location during each pass through the filter. Since the Q channel filter performs this shifting but only operates on every second data point, an additional DMOV instruction is coded between each LTD/MPY instruction pair in order to shift the even-numbered data table entries.

The assembly code that implements the Q channel bandpass filter is shown below.

```
*
* FIRST (Nth) TAP SETS UP FOR REST OF FILTER
*
      ZAC                                * CLEAR ACCUMULATOR
      LT      RBUF35                     * LOAD T REGISTER
      DMOV    RBUF35                     * SHIFT OLD VALUE
      MPYK    QCF35                      * MPY BY COEFFICIENT
      DMOV    RBUF34                     * PERFORM EXTRA SHIFT
*
* SECOND TAP
*
      LTD     RBUF33                     * LOAD T, ACCUMULATE, DATA SHIFT
      MPYK    QCF33                      * MULTIPLY BY COEFFICIENT
      DMOV    RBUF32                     * PERFORM EXTRA SHIFT
      .
      .
      .
*
* LAST TAP
*
      LTD     RBUF1                      * LOAD T, ACCUMULATE, DATA SHIFT
      MPYK    QCF1                       * MULTIPLY BY COEFFICIENT
      DMOV    RBUF0                      * PERFORM EXTRA SHIFT
*
      APAC                                * ADD LAST SUM
      SACH    QSUM,4                     * STORE FILTER OUTPUT
```

Automatic Gain Control Implementation

To better control the signal strength of the receiver, a software Automatic Gain Control (AGC) algorithm was added. The need of an AGC stems from the use of thresholds in both the carrier recovery and clock recovery algorithms. For increased performance, these thresholds (discussed in the following two subsections) must remain valid (unchanged) for different levels of the incoming signal. This is achieved with the use of the software AGC.

The arrangement of the AGC with respect to the other functional blocks of the modem receiver was shown in Figure 5. The AGC monitors the I channel of the receiver and calculates a gain correction factor. Both the I and Q channels are then multiplied by this gain correction factor so that the signal maxima remain within a certain range. This range is narrow compared to the range of the incoming signal maxima. The peak-to-peak voltage of the incoming signal is between 2 mV and 700 mV. In 16-bit hexadecimal Q15 format,¹⁵ this range is from $>5C$ to >5999 . However, with the use of the software AGC, the signal maxima are in the range 780 mV (>6400) to 820 mV (>6900).

The gain correction factor is calculated once every three bauds by a two-step process. First, the three maximum values of the signal (BSMAX), each one corresponding to one baud (16 samples), are monitored and added to each other. A counter (AGCNT) is used to keep the count of the signal maxima. The previous running average is then added to this sum, and the result is divided by four to obtain the new running average (AGCRA). The division by four is accomplished by shifting the final sum, contained in the accumulator, two locations to the right before storing it in the memory as the new running average (AGCRA). The section of code that implements this step is listed below.

```
*
*   DETECT MAX SIGNAL STRENGTH OF I CHANNEL PER BAUD
*   (THIS CODE IS EXECUTED EVERY CYCLE)
*
AGCAL    EQU    $
          LAC    ISUM      * AGC VALUE CALCULATED USING ISUM
          ABS          * GET MAGNITUDE OF SIGNAL
          SUB     BSMAX    * COMPARE TO PREVIOUS MAX VALUE
          BLZ     OVRMAX   * IF LESS THAN, THEN SKIP UPDATE
          ADD     BSMAX    * RESTORE VALUE AND
          SACL    BSMAX    * STORE AS NEW MAX
*
*   MULTIPLY I AND Q CHANNELS BY AGC FACTOR
*
OVRMAX
          .
          .
          .
```

```

*
*   UPDATE THE RUNNING AVERAGE ONCE EVERY THREE BAUDS
*   (THIS CODE IS EXECUTED ONCE EVERY BAUD)
*
AGCUPT   ZALH   AGCRA   * ADD THE NEW BSMAX VALUE
          ADD    BSMAX,14 * TO THE RUNNING AVERAGE
          SACH   AGCRA   * AND SAVE IT
          LAC    AGCNT   * DECREMENT RUNNING AVERAGE
          SUB    ONE     * SAVE IT AND
          SACL   AGCNT   * CHECK FOR ZERO
          SACH   BSMAX   * ZERO OUT RUNNING SIGNAL MAX
          BZ     OVROUT  * IF ZERO, THEN UPDATE AGC
          RET                      * ELSE RETURN TO CALLING SEQUENCE
OVROUT   LACK   3        * RESET RUNNING AVERAGE COUNT
          SACL   AGCNT   * TO THREE
          LAC    AGCRA   * MOVE AGCRA
          SACL   AGCLEV  * TO THE CALCULATION LEVEL
          LAC    AGCRA,14 * DIVIDE RUNNING AVERAGE SUM
          SACH   AGCRA   * BY 4 TO GET NEW RUNNING AVERAGE

```

At the second step, the gain correction factor (AGC) is calculated, based on the running average. A brute force approach is to divide the maximum-allowed signal level by the running average and obtain the gain correction factor as the result of this division. The maximum value of the product of the signal times the gain correction factor should then remain close to the maximum-allowed signal level. However, since divisions are costly in processing time, the second step is implemented by using the running average as an index (AGCLEV) to a 32-word lookup table. The offset to this table (AGCOFF) is added to the index (AGCLEV) to calculate the table entry on which the gain correction factor (AGC) is located. The TBLR instruction is then used to transfer the gain correction factor from program memory to data memory. To lessen the code space required to handle the AGC lookup table, the code uses only the six most significant bits of the running average. This requires a 64-word lookup table. However, since the most significant bit of the six bits is always one, only 32 entries of the table are needed. The gain correction factor, obtained by the table lookup, is shifted so that the product of the gain correction factor times the incoming signal is in Q14 format (designer's choice). The shift factor is provided by the BASIC program used to generate the AGC lookup table (see Appendix C). The TMS32010 code that implements the calculation of the gain correction factor is shown below.

```

          LAC    AGCLEV   * GET AVERAGE MAX SIGNAL LEVEL
          SUB    ONE,14   * COMPARE TO 16384
          BLZ    ASHF1    * IF LESS THAN SHIFT TABLE LOOKUP
          LAC    AGCLEV,7 * GET LOOKUP VALUE
          SACH   TEMP     * MOVE LOOKUP VALUE TO

```


	LAC	TEMP	* THE LOW HALF OF THE ACC
	ADD	AGCOFF	* ADD IN TABLE OFFSET
	TBLR	AGC	* AND GET AGC VALUE
	LAC	AGC,15	* DIVIDE THE AGC VALUE
	SACH	AGC	* BY 2 TO FORCE TO Q14 MODE
	RET		* RETURN TO CALLING SEQUENCE
ASHF1	ADD	ONE,13	* COMPARE TO 8192
	BLZ	ASHF2	* IF LESS THAN SHIFT TABLE LOOKUP
	LAC	AGCLEV,8	* GET LOOKUP VALUE
	SACH	TEMP	* MOVE LOOKUP VALUE TO
	LAC	TEMP	* THE LOW HALF OF THE ACC
	ADD	AGCOFF	* ADD IN TABLE OFFSET
	TBLR	AGC	* AND GET AGC VALUE
	RET		* RETURN TO CALLING SEQUENCE
ASHF2	ADD	ONE,12	* COMPARE TO 4096
	BLZ	ASHF3	* IF LESS THAN SHIFT TABLE LOOKUP
	LAC	AGCLEV,9	* GET LOOKUP VALUE
	SACH	TEMP	* MOVE LOOKUP VALUE TO
	LAC	TEMP	* THE LOW HALF OF THE ACC
	ADD	AGCOFF	* ADD IN TABLE OFFSET
	TBLR	AGC	* AND GET AGC VALUE
	LAC	AGC,1	* AGC VALUE*2 TO ADJUST
	SACL	AGC	* FOR LOWER SIGNAL STRENGTH
	RET		* RETURN TO CALLING SEQUENCE
	.		
	.		
	.		
ASHF6	ADD	ONE,5	* COMPARE TO 32
	BLZ	NOEDT	* LOST MINIMUM ENERGY LEVEL
	LAC	AGCLEV,13	* GET LOOKUP VALUE
	SACH	TEMP	* MOVE LOOKUP VALUE TO
	LAC	TEMP	* THE LOW HALF OF THE ACC
	ADD	AGCOFF	* ADD IN TABLE OFFSET
	TBLR	AGC	* AND GET AGC VALUE
	LAC	AGC,5	* AGC VALUE*32 TO ADJUST
	SACL	AGC	* FOR LOWER SIGNAL STRENGTH
	RET		* RETURN TO CALLING SEQUENCE

The AGC table was generated by the BASIC program listed in Appendix C. This program is written to execute on any MS-DOS operating system. The program prompts the user for the table size and gain range factor, and then generates and stores the AGC table. The table is stored in a format that allows insertion directly into the user's code.

Carrier Recovery Implementation

The carrier recovery is implemented with a phase-locked loop, as explained in the Modem Receiver subsection. In Figure 8, the functional blocks that must be digitally implemented are the phase detector, loop filter, and Voltage Controlled Oscillator (VCO).

Phase Detector

In the middle of each baud, the phase detector block calculates an equation equivalent to (8), repeated below for convenience,

$$E(nT_b) = \hat{Q}(nT_b) I'(nT_b) - \hat{I}(nT_b) Q'(nT_b)$$

where I' (RECI) and Q' (RECQ) are the baseband (demodulated) I and Q channels, and \hat{I} and \hat{Q} are the I and Q channel decisions. The derivation of the equivalent equation to (8) is discussed next.

In Figure 9, the I channel decision for constellation point A is the length of the projection of the vector \vec{OA} on the I axis. Similarly, the Q channel decision for constellation point A is the length of the projection of the vector \vec{OA} on the Q axis. Since the four constellation points A, B, C, and D are located on the 45 and -45 degree lines, the lengths of these projections are the same. With this common length denoted by L, the I channel decisions can be expressed as

$$\hat{I}(nT_b) = \begin{cases} +L & \text{for points A and B} \\ -L & \text{for points C and D} \end{cases} \quad (27)$$

The value of L depends on the radius of the circle on which the four constellation points are located. Equation (27) can equivalently be expressed as

$$\hat{I}(nT_b) = \text{sgn}(I'(nT_b)) L \quad (28)$$

where sgn is the sign function defined as

$$\text{sgn}(I'(nT_b)) = \begin{cases} +1 & \text{if } I'(nT_b) > 0 \text{ (points A and B)} \\ -1 & \text{if } I'(nT_b) < 0 \text{ (points C and D)} \end{cases} \quad (29)$$

Similarly,

$$\hat{Q}(nT_b) = \text{sgn}(Q'(nT_b)) L \quad (30)$$

Substitution of (28) and (30) into (8) gives

$$E(nT_b) = L \{ \text{sgn}(Q'(nT_b)) I'(nT_b) - \text{sgn}(I'(nT_b)) Q'(nT_b) \} \quad (31)$$

Equations (31) and (8) are identical. However, (31) is the final step towards the equation implemented in the TMS32010. Since L in (31) is a positive constant, an equivalent error function that contains the phase and frequency information is

$$E'(nT_b) = \text{sgn}(Q'(nT_b)) I'(nT_b) - \text{sgn}(I'(nT_b)) Q'(nT_b) \quad (32)$$

Equation (32) is the one implemented in the TMS32010 as part of the carrier recovery algorithm. In this equation, $\text{sgn}(I')$ (SIGNI) and $\text{sgn}(Q')$ (SIGNQ) are the I and Q channel decisions, respectively. The TMS32010 code used to implement (32) is shown below.

```

*
* COMPUTE CARRIER ERROR SIGNAL
*
* e(t) = RECI*SIGNQ - RECQ*SIGNI
*
COMERR LT      RECI      * T = RECI
      MPY      SIGNQ     * P = RECI*SIGNQ
      LTP      RECQ     * T = RECQ, ACC = RECI*SGNQ
      MPY      SIGNI     * P = RECQ*SIGNI
      SPAC     * ACC = RECI*SIGNQ - RECQ*SIGNI
      SACH     ERROR,1  * STORE IN ERROR

```

Loop Filter

The error signal $E'(nT_b)$ (ERROR), generated by the phase detector (equation (32)), is filtered by the carrier recovery loop filter (see Figure 8). The filter was implemented as a first-order Infinite Impulse Response structure. In other words, the loop filter is just an integrator with transfer function

$$H_1(z) = \frac{B_1}{1 - A_1 z^{-1}} \quad (33)$$

where A_1 (PLL1) and B_1 (PLL2) are the filter coefficients.

A higher-order filter was not used, because high-order filter structures usually introduce more phase delay than first-order sections. Phase delays¹⁶ are critical in the operation of a phase-locked loop, and their effects are difficult to analyze.

The time-domain equivalent of (33) is

$$y(n) = B_1 x(n) + A_1 y(n-1) \quad (34)$$

where $x(n)$ is the input to the filter and $y(n)$ the output. The signal flowgraph of the carrier recovery loop filter is shown in Figure 16.

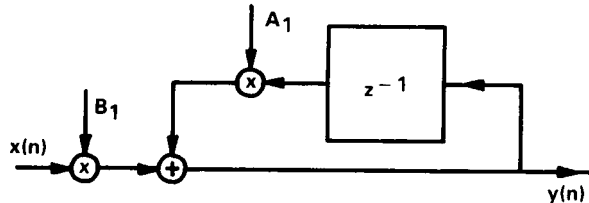


Figure 16. Carrier Recovery Loop Filter

The TMS32010, with a hardware on-chip multiplier, is most efficient in the implementation of such filter structures.¹³ The code used to implement the carrier recovery loop filter (equation (34)) is shown below. The filter's input $x(n)$ is stored in ERROR, and the filter's output $y(n)$ is stored in ERRSIG.

```
*
* LOOP FILTER
*
      LT          PLL2          * T = PLL2
      MPY          ERROR         * P = PLL2*ERROR
      LTP          PLL1          * ACC = PLL2 ERROR, T = PLL1
      MPY          ERRSIG        * P = PLL1*ERRSIG
      APAC         * ACC = PLL2*ERROR + PLL1*ERRSIG
      SACH          ERRSIG,1     * STORE IN ERRSIG
```

The effect of the loop filter's bandwidth in the modem performance is considered in the following discussion where the bandwidth of the loop filter is defined as the frequency at which the magnitude of the filter's transfer function is 3 db below its maximum value. Therefore, the bandwidth of the loop filter is the frequency ω_b at which

$$20 \log \frac{|H_1|_{\max}}{|H_1(\omega_b)|} = 3 \quad (35)$$

where $|H_1|_{\max}$ is the maximum value of the magnitude of the filter's transfer function. Substituting $z = e^{j\omega}$ in (33) gives

$$|H_1(\omega)| = \frac{|B_1|}{\{1 + A_1^2 - 2A_1 \cos(\omega)\}^{1/2}} \quad (36)$$

Equation (36) is maximum when the denominator is minimum. This is true for $\omega=0$, i.e., at DC. Substituting $\omega=0$ in (36) gives

$$|H_1|_{\max} = \frac{|B_1|}{1 - A_1} \quad \text{where } 0 < A_1 < 1 \quad (37)$$

Substitution of (36) and (37) into (35) gives the following quadratic equation that relates the bandwidth of the loop filter ω_b to the coefficient A_1 .

$$A_1^2 + 2A_1 \{\cos(\omega_b) - 2\} + 1 = 0 \quad (38)$$

Therefore, the value of the coefficient A_1 determines the bandwidth of the loop filter. Figure 17 shows a plot of the values of A_1 versus the bandwidth ω_b , i.e., a plot of (38).

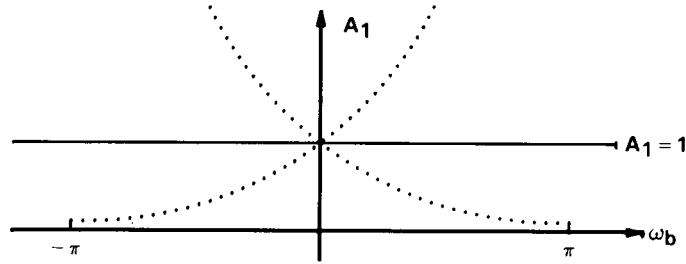


Figure 17. Parameter A_1 vs. the Bandwidth of $H_1(z) = \frac{B_1}{1 - A_1 z^{-1}}$

The two curves in Figure 17 represent the solutions of the quadratic equation (38). From this figure, it can be seen that the closer A_1 is to unity, the narrower the bandwidth of the filter. If $A_1 = 1$, the magnitude response begins rolling off at zero frequency ($\omega = 0$). However, this situation must be avoided since $A_1 = 1$ results in placing a pole on the unit circle in the z -domain, thereby causing the filter to oscillate. Since only values less than unity of the coefficient A_1 result in a stable filter structure, Q15 format¹⁵ was used to represent A_1 .

The bandwidth ω_b is expressed in radians. Since the sampling frequency f_b corresponds to 2π radians, the bandwidth of the loop filter in Hz is given by

$$\text{Bandwidth} = \frac{\omega_b f_b}{2\pi} \text{ Hz} \quad (39)$$

Since the loop filter runs once every baud, the sampling frequency f_b is

$$f_b = \frac{1}{T_b} = 600 \text{ Hz}$$

with T_b as the baud interval. This frequency should not be confused with the A/D and D/A sampling frequency designated by f_s and having the value of 9600 Hz.

The bandwidth of the loop filter affects the Bit Error Rate (BER) and the time it takes for the modem receiver to lock-on to the incoming carrier. Initially, a large bandwidth results in a fast lock-on while a narrow bandwidth provides a good BER. Therefore, the ability to switch from a large bandwidth to a narrow one results in a better modem design. With the TMS32010, this is easily implemented using the TBLR instruction that transfers data from program memory to data memory.¹⁵ On startup, A_1 (PLL1) is 0.539 or >4500 in Q15 format. This corresponds to a bandwidth of approximately 63 Hz. Once locked-on with the use of the TBLR instruction, the value of A_1 (PLL1) is changed to 0.953 or >7A00 in Q15 format. This corresponds to a bandwidth of approximately 6 Hz. Lock-on criterion is based on the magnitude of the error function

calculated by (32) being less than a certain threshold. The need and calculation of this threshold is covered later in this subsection. The TMS32010 code used to switch the loop filter's bandwidth is given below. The fifth bit of RECST is used as a flag, which if set indicates that the local carrier is locked-on to the incoming carrier.

LAC	ONE,4	* CHECK IF LOCAL CARRIER
AND	RECST	* IS LOCKED. IF SO, SWITCH
BNZ	CARLCK	* PLL FILTERS' BANDWIDTH
B	NORMAL	* EXECUTE NORMAL SEQUENCE
*		
CARLCK	LACK	PLLC
	TBLR	PLL1
		* CHANGE CARRIER PLL COEF. 1

Voltage-Controlled Oscillator

Both the carrier used in the transmitter to modulate the data and the one used in the receiver for the demodulation (local carrier) were implemented in the TMS32010 using a 128-point sine table and a routine to drive it.¹⁷ The voltage-controlled oscillator in the phase-locked loop for the carrier recovery generates the local carrier using this 128-point sine table. The frequency of this digital sine wave is 2400 Hz for an originate modem and 1200 Hz for an answer modem.

Carrier Recovery Threshold

The lowpass-filtered value of the error signal generated by the phase detector contains the information about the phase and frequency difference between the local and incoming carriers. If this value (ERRSIG) is positive, the local carrier must be advanced in phase. If negative, the local carrier must be delayed (see the Modem Receiver subsection). Since there are 128 points in the sine table, there is a 360/128 or 2.8125-degree jump going from one table entry to the next. This implies that corrections should not be made unless the magnitude of the error signal is greater than one table entry because redundant corrections introduce inaccuracies and noise. Therefore, the value of this threshold should correspond to the magnitude of the error signal when there is a 2.8125-degree phase error.

An estimate of the threshold can be obtained as described below. The relation of the phase error signal $E(nT_b)$ to the phase error θ_e is given by (18). Substituting 2.8125 for θ_e in (18) and taking the magnitude of both sides gives

$$|E(nT_b)| = |K \sin(2.8125)| \quad (40)$$

$K(I^2 + Q^2)$ is the signal energy, i.e., the maximum value of the I and Q channels. This value is set by the Automatic Gain Control. Since the software AGC used in this implementation of the Bell 212A/V.22 limits the signal maxima between 0.78 and 0.82 (see Automatic Gain Control Implementation in the Modem Receiver subsection), K is between 0.78 and 0.82. Using the average value of 0.80 for K, (40) gives

$$|E(nT_b)| = 0.039.$$

The threshold level should be at 0.039 if the gain of the loop filter given by (33) is unity. For DC, the gain G_1 of the loop filter is given by (37), repeated below for convenience.

$$G_1 = |H_1|_{\max} = \frac{B_1}{1 - A_1} \quad \text{where } 0 < A_1 < 1$$

The coefficient B_1 (PLL2) was chosen to be 0.0039 (or >50 in the Q15 format). As explained earlier, once the receiver is locked, the value of coefficient A_1 (PLL1) is 0.953. From (37), the gain G_1 of the loop filter is $G_1 = 0.082$. Therefore, the threshold is scaled down to

$$\text{Effective Threshold} = 0.039 \times 0.082 = 0.0032.$$

This corresponds to $>D$ in Q12, the format used for the threshold (designer's choice). After this initial estimate of the threshold was obtained, the final value of the carrier recovery threshold (TRSHD1), >7 , was determined by trial and error. The calculated threshold is greater than the one obtained by trial and error, because of the use of the maximum value of the loop filter's gain in the threshold calculation.

To improve the lock-on characteristics of the modem, a two-level correction was used for the carrier recovery. If the magnitude of the error (ERRSIG) is less than the threshold (TRSHD1), no correction is applied. If the magnitude of the error is greater than the threshold but less than twice the threshold, one sine-table entry correction is applied by incrementing or decrementing the table entry pointer (RALPHA) by one. If the magnitude of the error is greater than twice the threshold value, then a two-table entry correction is applied by incrementing or decrementing the table entry pointer (RALPHA) by two. All of the corrections are applied to advance or delay the local carrier according to the algorithm described in the Modem Receiver subsection.

Baud Clock Alignment Implementation

The purpose of the clock recovery is to identify the baud boundaries and inform the decision block when the middle of each baud occurs and therefore the optimum time to make an error-free decision (see Figure 5). As explained in the Modem Receiver subsection, one approach for clock recovery (adjustment of the baud clock) is to use the energy of the incoming signal. The energy is the sum of the squares of the demodulated I and Q channels (see equation (20)). As implied by (21), this quantity is independent of any phase and/or frequency difference between the incoming and local carriers.

The minima of the signal energy indicate the beginning of a new baud. This can be seen in Figure 18 where the signal energy is plotted every sample for several consecutive baud intervals.

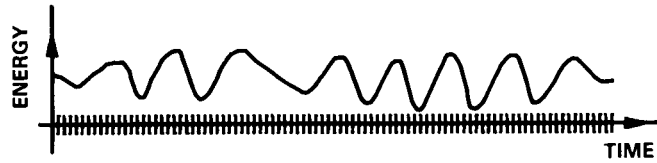


Figure 18. Signal Energy Plotted Every Sample For Several Baud Intervals

Each of the short vertical lines along the horizontal axis in Figure 18 corresponds to a sample time. This data was obtained using the XDS/22 emulator for the TMS32010. The block diagram for the clock recovery algorithm is shown in Figure 19. The functional blocks to be implemented are the error signal generator, loop filter, and baud clock.

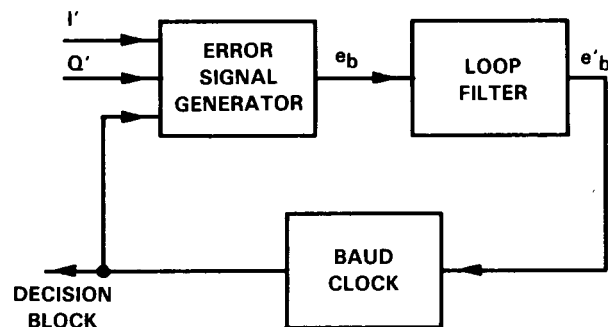


Figure 19. Baud Clock Alignment Block Diagram

Error Signal Generator

The error signal generator calculates the signal energy and from it generates an error signal e_b . This error signal contains the information about how close the local baud boundaries are to the incoming baud boundaries. The error signal is then lowpass-filtered so that noise and high-frequency components are removed. The output of the loop filter corrects the local baud clock.

The critical issue is how to calculate this error signal. Figure 20 shows the signal energy for a single baud interval. This figure was motivated from the realtime data of Figure 18. The 16 energy samples for this baud are indicated as $E(0), E(1), \dots, E(15)$.

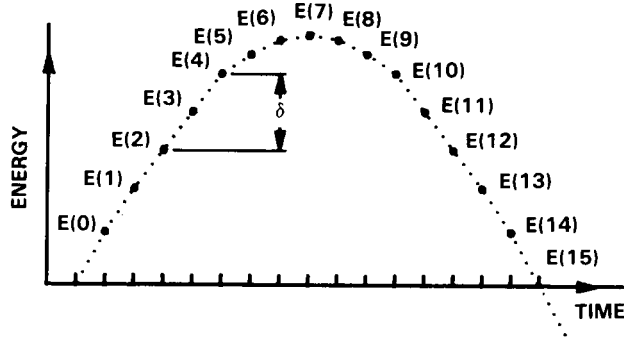


Figure 20. Signal Energy Samples over a Baud

From Figure 20, it can be seen that the energy sample E(7) is located at the middle of the baud (top of the 'energy hill'), and the rest of the samples are located symmetrically around it, i.e., $E(6) = E(8)$, $E(5) = E(9)$, and so on. Therefore, E(7) is taken to be the middle of the local baud. Consider now the difference between the energy sample E(11) that is four samples after E(7) and the energy sample E(3) that is four samples before E(7). If the local baud clock is correctly aligned so that E(7) corresponds to the middle of the incoming baud, then

$$E(11) - E(3) = 0.$$

If

$$E(11) - E(3) > 0,$$

then the sample E(7) is located to the left of the middle of the baud. This means that the middle of the local baud occurred earlier than the middle of the incoming baud. Therefore, the local baud clock must be delayed. On the other hand, if

$$E(11) - E(3) < 0,$$

the middle of the local baud occurred later than the middle of the incoming baud. Therefore, the local baud clock must be advanced.

In summary, the error signal generator computes the signal energy at sample points 3 (PENRGY) and 11 (ENRGY). The sample count information (SAMPLE) is provided by the baud clock shown in Figure 19. The error signal generator then calculates the error signal e_b (BERROR) defined by

$$e_b = E(11) - E(3) \quad (41)$$

The subscript b represents baud since this signal is calculated once every baud.

Loop Filter

Perturbations that may occur in the communications medium pass onto the demodulated I and Q channels. This can be seen from the data of Figure 18 where even with the presence of the automatic gain control, the energy levels are not exactly the same for every baud. Also, the duration of each baud in Figure 18 is not exactly sixteen samples (sixteen short vertical lines) as it theoretically should be. These perturbations result in abrupt changes of the signal generated by the error signal generator. Therefore, the error signal is not directly fed into the baud clock. Instead, it is lowpass-filtered by the loop filter. This removes noise and high-frequency components and results in a stable clock recovery.

The loop filter was implemented as a first-order recursive filter. The transfer function is of the same form as (33).

$$H_2(z) = \frac{B_2}{1 - A_2 z^{-1}} \quad (42)$$

Just as with the loop filter used for the carrier recovery, the most important characteristic of the loop filter used for the clock recovery is its bandwidth. A wide bandwidth results in a quick adjustment of the local baud boundaries to the incoming baud boundaries. A narrow bandwidth results in a more stable clock recovery. A good approach for this filter's design is to start with a wide bandwidth and then switch to a narrow one. All of the information provided in the Carrier Recovery Implementation subsection relating the coefficient A_1 to the loop filter's bandwidth apply here as well. With the use of the TBLR instruction, after the receiver is locked-on to the incoming carrier, the initial wide bandwidth is switched to a narrow one. The initial value of A_2 is 0.5, which is >4000 in Q15 format and corresponds to approximately a 70-Hz bandwidth. After the receiver is locked, this value changes to 0.91 (>7500 in Q15 format), which corresponds to a bandwidth of approximately 10 Hz. The criterion used for the receiver being locked-on is the magnitude of the error function calculated by (32) being less than the threshold used for the carrier recovery (TRSHD1).

Baud Clock

The output of the loop filter, designated by e'_b in Figure 19, drives the local baud clock. The baud clock tracks the sample count (SAMPLE) and thus informs:

1. The decision block when it is the middle of the baud (sample 7) and thus the optimum time for demodulation, and
2. The error signal generator when the sample count is 3 and 11 so that the error signal e_b can be calculated.

These two objectives are achieved with the use of a 16-entry table in the program memory. Each table entry contains the address of a subroutine task to be performed between two consecutive samples. The tasks are numbered 0, 1,..., 15. Table 6 shows the memory map of the 16 tasks performed by the modem receiver.

Table 6. Memory Map of Tasks Performed by the Modem Receiver

*				
* TASK MASTER SEQUENCE TABLE (RECEIVE)				
* TASKS ARE EXECUTED FROM BOTTOM TO TOP				
*				
TSKSEQ	EQU	\$		
	DATA	DUMMY	* UNUSED CYCLE	15
	DATA	DUMMY	* UNUSED CYCLE	14
	DATA	DUMMY	* UNUSED CYCLE	13
	DATA	DUMMY	* UNUSED CYCLE	12
	DATA	BDCLK2	* COMPUTE ENERGY E(11)	11
	DATA	DUMMY	* UNUSED CYCLE	10
	DATA	OUT	* COMMUNICATE WITH TMS7742	9
	DATA	DECODE	* DECODE/GET SCRAMBLED DIBIT	8
	DATA	DEMODB	* DEMODULATE IN MIDDLE OF BAUD	7
	DATA	DUMMY	* UNUSED CYCLE	6
	DATA	AGCUPT	* UPDATE THE AGC EVERY 3RD BAUD	5
	DATA	DUMMY	* UNUSED CYCLE	4
	DATA	BDCLK1	* COMPUTE ENERGY E(3)	3
	DATA	DUMMY	* UNUSED CYCLE	2
	DATA	DUMMY	* UNUSED CYCLE	1
	DATA	DUMMY	* UNUSED CYCLE	0

Task 3 (BDCLK1) calculates the signal energy E(3) (PENRGY). Task 5 updates (once every three bauds) the automatic gain control value. Task 7 (DEMODB) implements the demodulation in the middle of the baud. Task 8 (DECODE) makes the channel decisions based on the demodulated (from Task 7) I and Q values, and decodes the decisions to obtain the scrambled dibits. Task 9 (OUT) performs the data exchange between the TMS32010 and the TMS7742. Task 11 calculates the signal energy E(11) (ENRGY). The TMS32010 code used to drive the table of the modem receiver tasks is shown below.

*			
* RECEIVER TASK SEQUENCE DRIVER ROUTINE			
*			
	LAC	SAMPLE	* DECREMENT THE SAMPLE COUNT
	SUB	ONE	* TO CHECK FOR END OF BAUD
	BGEZ	OVRSAM	* IF NOT, THEN SKIP COUNT RESET
	LACK	15	* RESTART THE SAMPLE COUNTER AT 15

OVRSAM	SACL	SAMPLE	* SAVE NEW COUNT VALUE
	LACK	TSKSEQ	* GET ADDRESS OF TOP OF TABLE
	ADD	SAMPLE	* ADD IN OFFSET
	TBLR	TEMP	* GET THE PROGRAM ADDRESS
	LAC	TEMP	* FOR THE TASK CALL
	CALA		* EXECUTE THE APPROPRIATE TASK

Initially, the sample count (SAMPLE) contains the task number of the previous task performed. This number is decremented so that the next task in the sequence is performed. If the sample count becomes negative, it is reset to 15. The sample count is then added to the address of the top of the task table (TSKSEQ). With the use of the TBLR instruction, the table entry is transferred to the data memory. Each table entry is the address of the subroutine task to be performed. Using the CALA instruction, the equivalent of the 'computed GOTO' used in FORTRAN, the program control transfers to the selected subroutine. For a 9.6-kHz sampling rate, the TMS32010 with a 200-ns cycle time has 512 cycles available to implement each of these tasks. This number of cycles is more than enough since the worst-case task takes approximately 300 cycles. Also, since only 6 out of the 16 tasks are used, 10 more tasks are available for the designer to incorporate additional functions such as an adaptive equalizer, scrambling/descrambling, and synchronous-to-asynchronous and asynchronous-to-synchronous conversions.

The algorithm of adjusting the baud clock based on the filtered error signal e'_b (BEROUT) is the same as the one described earlier for the unfiltered error signal e_b (BERROR), and is summarized below.

$$\begin{aligned}
 e'_b > 0 & \quad \text{delay local baud clock} \\
 e'_b < 0 & \quad \text{advance local baud clock}
 \end{aligned}
 \tag{43}$$

The advance or delay of the baud clock is implemented by changing the sample count (SAMPLE) appropriately. In the case of delaying the clock, the middle of the local baud clock (sample 7) occurs earlier than the middle of the incoming baud. Geometrically, sample 7 is located on the left side of the 'energy hill' of Figure 20 instead of at the top. If the sample count does not change, then 16 samples later, sample 7 of the next local baud will again be located on the left side of the 'energy hill' of the next incoming baud. Therefore, the sample count must be decremented by one. Instead of 16 samples, the middle of the next baud is taken to be 17 samples later. Hopefully then, the middle of the local baud is on or at least closer to the top of the 'energy hill.'

The case of advancing the clock is similar, except that the sample count is incremented by one, and thus the middle of the next baud is taken 15 samples after the middle of the current baud.

Clock Recovery Threshold

One more issue, the clock recovery threshold, is associated with the alignment of the baud clock. Since there is a finite number of samples in each baud interval, the

baud clock has a finite resolution. Therefore, if the middle of the local baud (sample 7) is within one sample of the middle of the incoming baud, no correction must be applied. A threshold can be used so that corrections are applied only if the magnitude of the filtered error signal is greater than the threshold value. An initial estimate of this threshold is obtained by computing the magnitude of the error signal that corresponds to a one-sample change in the local baud clock. Consider the effect of a one-sample change in Figure 20. The middle of the local baud clock E(7) is translated to E(6) (or E(8)); E(3) is translated to E(2) (or E(4)); and E(11) is to E(10) (or E(12)). Therefore, a one sample change results in an error signal e_b given by (41) of magnitude δ as indicated in Figure 20. Approximating the 'energy hill' with the positive half of a sine wave (see Figure 20), results in $\delta = 0.12$. This would be the threshold if the gain of the clock recovery loop filter were unity. For DC, the gain of this filter is (see equation (37))

$$G_2 = |H_2|_{\max} = \frac{B_2}{1 - A_2} \quad \text{where } 0 < A_2 < 1$$

The value chosen for the coefficient B_2 (BPLL2) is 0.0024 or >50 in Q15 format. After the receiver is locked-on to the incoming carrier, the coefficient A_2 (BPLL1) is 0.91. The gain G_2 of the loop filter is computed to be $G_2 = 0.026$.

The gain G_2 results in an 'effective threshold' of $\delta = 0.00312$. This corresponds to >33 in Q14 format used for the clock recovery threshold by designer's choice. However, this is just an initial estimate since the mathematical model used is only an approximation. After this estimate was obtained, the final value of the clock recovery threshold (TRSHD2), >8 , was determined by trial and error.

The calculation of the thresholds for both the clock and carrier recoveries was performed based on the DC gain of the loop filters. A reason why the calculated thresholds are greater than those obtained by trial and error is that the filter gain is maximum at DC.

Just as in the carrier recovery, a two-level correction is used for the baud clock. If the magnitude of the error signal is less than the threshold, no correction is applied. If the magnitude of the error signal (BERROUT) is greater than the threshold (TRSHD2) but less than twice the threshold, the baud clock is advanced or delayed by one sample. If the magnitude of the error is greater than twice the threshold, then the baud clock is adjusted by two samples.

Functions Implemented in the TMS7742

The Texas Instruments TMS7742 is a microcomputer with an on-chip UART and 4K bytes of internal EPROM. It was included in the modem design to increase its flexibility and upgradability. With the use of the TMS7742, both serial and parallel interfaces with the DTE can be efficiently implemented. The TMS7742 can also perform some of the modem functions, thus allowing the TMS32010 to do more complicated tasks. This flexibility allows the hardware design to be upgradable to 2400-bps splitband modems

(V.22 bis). The TMS7742 acts as a modem controller and performs the asynchronous-to-synchronous and synchronous-to-asynchronous data conversions. It also scrambles the data from the DTE and descrambles the decoded dibits received from the TMS32010 before sending them to the DTE. The TMS7742 code is given in Appendix E.

Asynchronous-to-Synchronous and Synchronous-to-Asynchronous Conversions

Asynchronous data received from the DTE may include a start bit, seven or eight data bits, and one or more stop bits. When the DTE is not sending any data, the modem must still continue to transmit scrambled marks. Even though the DTE can send faster than 1200 bits per second, the modem must transmit only 1200 bits per second to the telephone line. This means that the modem must delete some of the bits received from the DTE. The Bell 212A protocol permits deleting one stop bit every nine characters. The data received from the TMS32010 demodulator may have characters with a deleted stop bit. The TMS7742 must detect the deleted stop bit and add it to the character before sending it to the DTE. The TMS7742 assembles the descrambled dibits into a character, checks for missing stop bits, and adds the missing stop bit if detected. The speed of the UART is set to enable inserting one stop bit in every nine characters; i.e., when transmitting 10 bits per character, adding one bit in nine characters (a total of 90 bits) should not change the speed. Thus, the UART is set to 1/90th of a bit interval faster.

Scrambler/Descrambler

The data that has been converted into synchronous dibits is scrambled using equation (2), which is repeated below.

$$d_s(n) = d(n) \text{ XOR } d_s(n - 14) \text{ XOR } d_s(n - 17)$$

The TMS7742 holds the previous 17 scrambler outputs in its internal registers and uses the XOR instruction to exclusively-OR the proper bits to generate the new scrambled output. After scrambling each bit, these registers are shifted by one and saved to provide the $(n - 7)$ outputs for the next bit.

A similar routine is used to descramble the decoded data received from the TMS32010. The descrambling is performed using equation (3) repeated below.

$$d(n) = d_s(n) \text{ XOR } d_s(n - 14) \text{ XOR } d_s(n - 17)$$

Performance

The performance of the modem implemented using the TMS32010 was evaluated using automatic modem testing equipment. A block diagram of this testing equipment is shown in Figure 21.

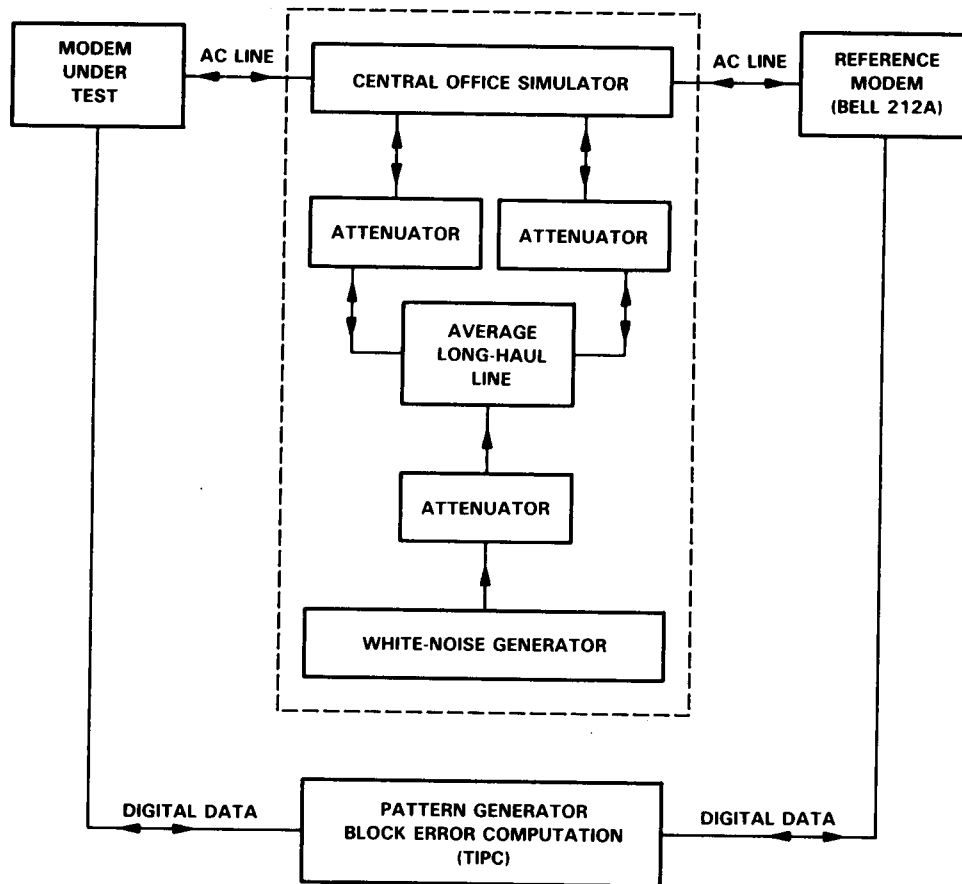
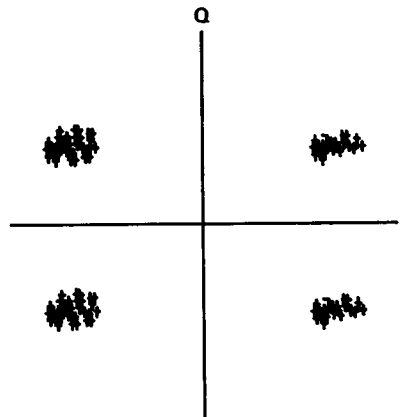


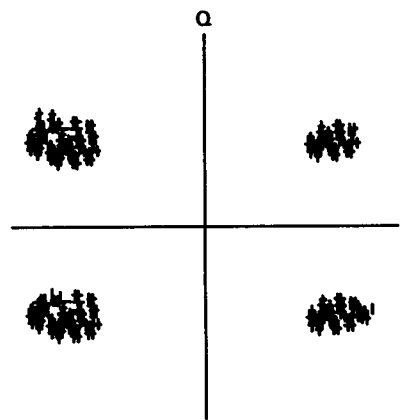
Figure 21. Modem Testing Equipment

The testing environment in Figure 21 provides a Central Office simulator, an average long-haul line simulator, and a C-notched white-noise generator. The attenuators provide signal-level and noise-level attenuation. The testing is performed under full-duplex and maximum data throughput conditions.

The average long-haul line effects are evident from the differences between the signal constellation diagrams of Figures 22(a) and 22(b). Figure 22(a) shows the signal constellation with the TMS32010 modem in the analog loop-back mode. Figure 22(b) shows the signal constellation with the TMS32010 modem operating over an average long-haul line at a 14-db signal-to-noise ratio. The presence of the average long-haul line results in a 'spreading' of the signal constellation points. This spreading implies a higher probability of error since the signal points used to make the decisions approach the decision boundaries.



(a) SIGNAL CONSTELLATION IN ANALOG LOOP-BACK MODE



(b) SIGNAL CONSTELLATION OVER AVERAGE LONG-HAUL LINE

Figure 22. Signal Constellation Diagrams

Referring to Figure 21, the Texas Instruments Professional Computer generates random characters. These characters are sent to the reference modem and the modem under testing. The modems transmit the characters they receive to each other, and each modem sends the characters received to the Professional Computer. The computer then compares the received characters with the ones originally created to determine the error rate. The error rate is determined in terms of percent error-free blocks. Each block consists of 512 characters (5120 bits) and is considered to be error-free only if all of the bits in the block are received with no error.

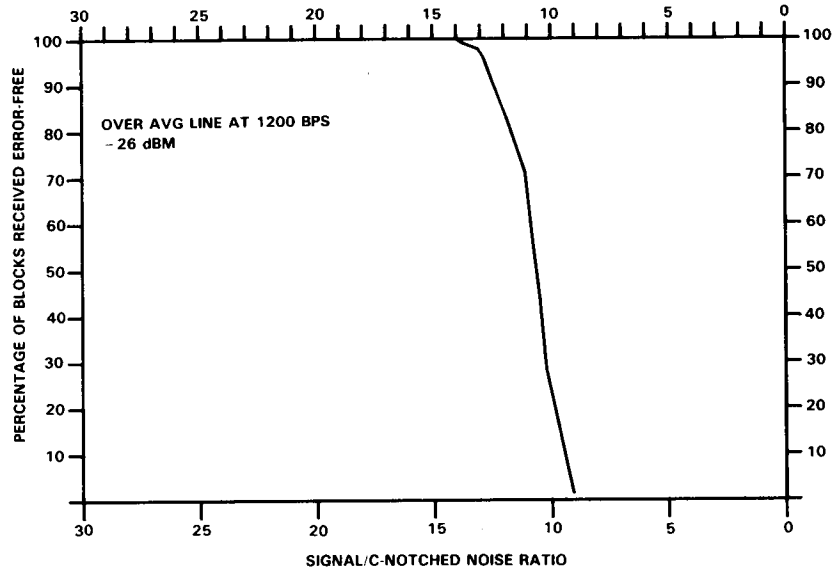
In all the tests performed, the Bell 212A modem was the reference modem configured in the answer mode. The reason for this is that only an originate

TMS32010-based modem is implemented. The answer mode is not included because, as mentioned in the "Introduction," this is beyond the purpose of this report. To incorporate the answer mode, two tables must be added in the TMS32010 code presented in Appendix D. The first table should contain the coefficients of the two receiver input bandpass filters with a passband centered around 1200 Hz. The second table should contain the increments used by the sine-table driver routine so that a 2400-Hz carrier is generated for the transmitter and a 1200-Hz carrier is generated for the receiver. When the TMS7742 configures the TMS32010 in the answer mode, the filter coefficients and the sine-table increments can be transferred from the program memory to the data memory with the use of the TBLR instruction. No performance difference is expected between the answer and originate modes.

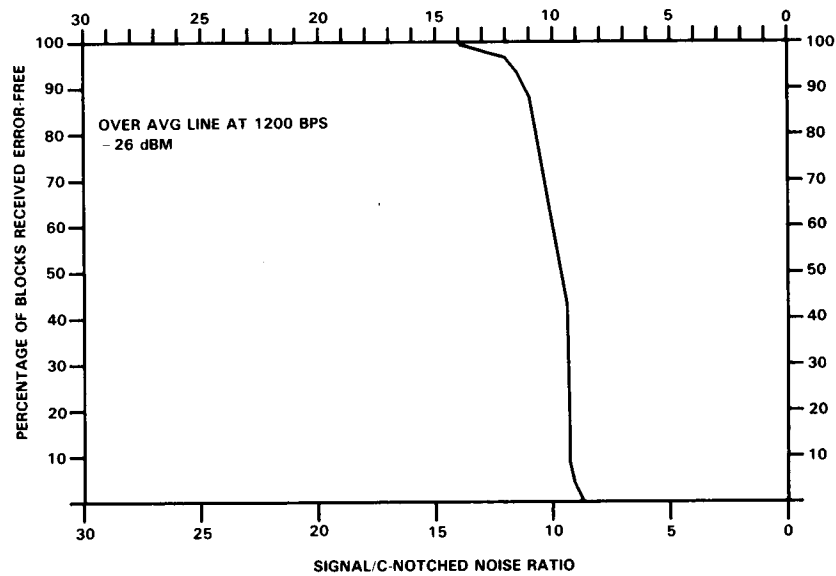
In Figure 23, the vertical axis indicates the percentage of blocks received error-free and the horizontal axis is the signal-to-noise ratio in db. The percentage of error-free blocks is calculated at each signal-to-noise ratio level (30, 29, 28,...) based on the number of error-free blocks received out of 1024 transmitted. All tests were performed at a -26 dbm (0.1 V) signal level. Figure 23(a) shows the test results with the TMS32010-based modem as the modem under testing. The vertical axis of Figure 23(a) is the percentage of blocks received error-free by the Bell modem. Figure 23(b) shows the results when the AT&T Dataphone II is used instead of the TMS32010-based modem.

Since the Bell modem is used as a reference modem, the above results indicate how well the transmitters of the TMS32010-based modem and the AT&T modem are performing. From Figures 23(a) and 23(b), it can be seen that for both the TMS32010 and AT&T modems, block errors start occurring at a signal-to-noise ratio of approximately 13 db and that the curve corresponding to the TMS32010 modem falls slightly faster. Therefore, the performance of both modem transmitters is approximately the same with the AT&T transmitter performing slightly better than the TMS32010 transmitter. Figure 24(a) shows the percentage of blocks received error-free by the TMS32010-based modem. The Bell modem (reference modem) is used to transmit these blocks. Figure 24(b) shows the percentage of blocks received error-free by the AT&T modem with the Bell modem transmitting.

It can be seen that the AT&T receiver performs approximately 2 db better than the TMS32010 receiver. The performance of the TMS32010 modem receiver could be improved with the inclusion of more filter taps in the receiver input bandpass filters.

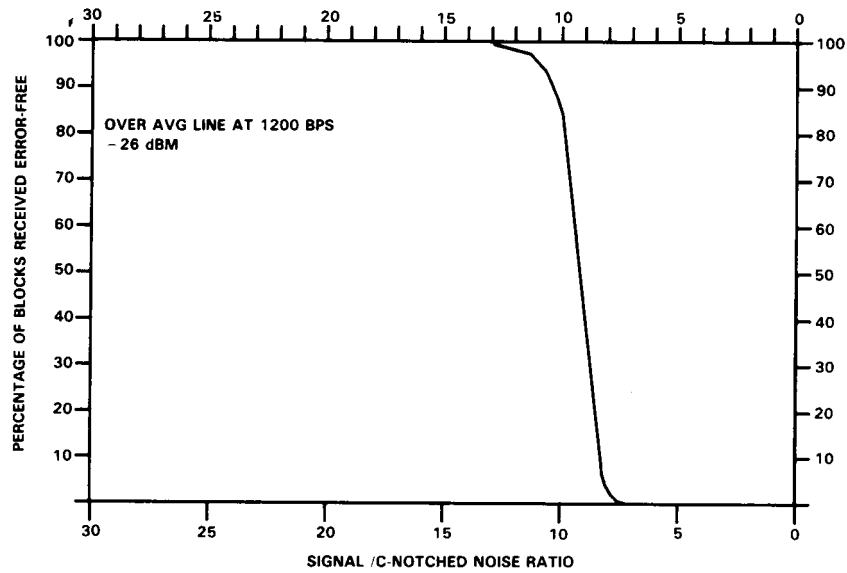


(a) PERCENTAGE OF BLOCKS RECEIVED ERROR-FREE BY THE BELL 212A MODEM VS. SNR WITH THE TMS32010 MODEM ORIGINATING

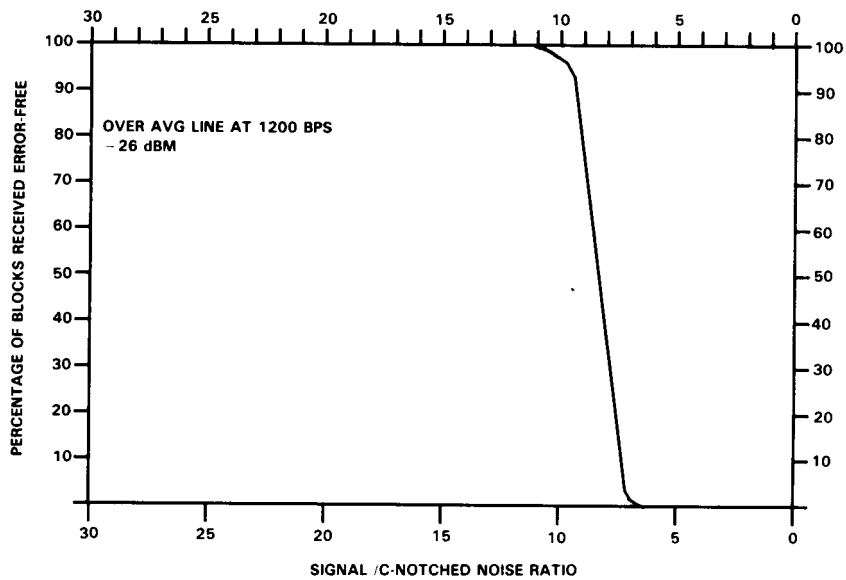


(b) PERCENTAGE OF BLOCKS RECEIVED ERROR-FREE BY THE BELL 212A MODEM VS. SNR WITH THE AT&T MODEM ORIGINATING

Figure 23. Performance of TMS32010 and AT&T Modem Transmitters



(a) PERCENTAGE OF BLOCKS RECEIVED ERROR-FREE BY THE TMS32010 MODEM VS. SNR WITH THE BELL MODEM TRANSMITTING



(b) PERCENTAGE OF BLOCKS RECEIVED ERROR-FREE BY THE AT&T MODEM VS. SNR WITH THE BELL MODEM TRANSMITTING

Figure 24. Performance of TMS32010 and AT&T Modem Receivers

Other Implementation Considerations

The implementation approach of the Bell 212A/V.22 modem presented in the previous sections is not unique. There are other and possibly more efficient ways of implementing the modem.

Drastic reduction of the hardware cost results from the use of a codec for the A/D and D/A conversions instead of the 12-bit linear A/D and D/A converters used in this implementation. This approach becomes even more attractive with the use of the TMS32011 digital signal processor in place of the TMS32010. The TMS32011 is a microcomputer (no external memory expansion) having the same architecture as the TMS32010 with the additional feature of containing the necessary logic for interfacing to a codec. In this implementation, the necessary input bandpass filtering for the modem receiver can be performed with an AMI S35212A analog filter chip. The modem hardware block diagram of this implementation is shown in Figure 25.

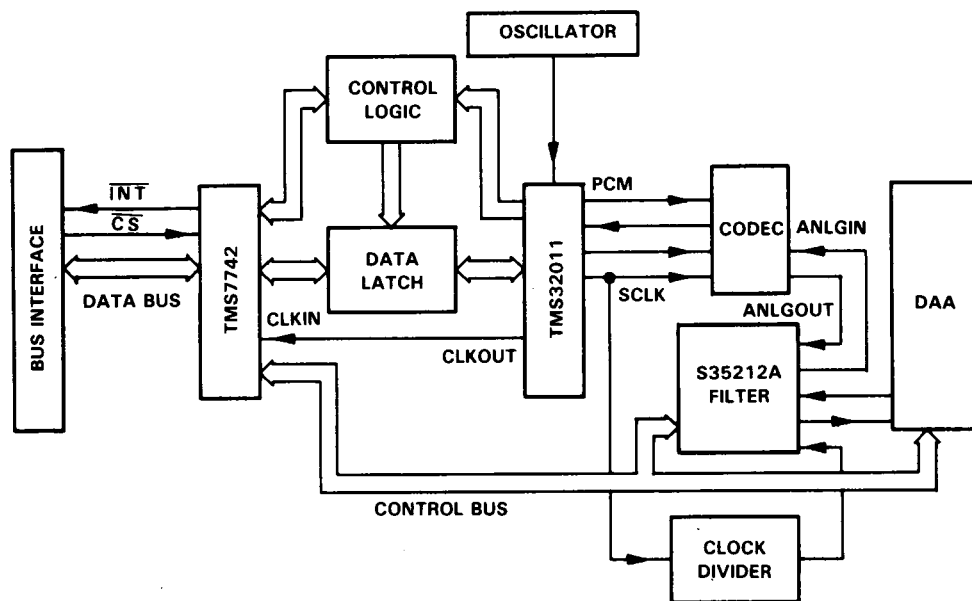


Figure 25. Modem Hardware Block Diagram Using a Codec for the A/D and D/A Conversions

If this approach is used, the receiver input has the configuration shown in Figure 26. The bandpass filtering is implemented in the analog domain and the Automatic Gain Control and Hilbert Transformer Pair implemented in the digital domain inside the TMS32011. Implementing the bandpass filtering in the analog domain should save adequate program memory, data memory, and processing power to allow the design to be upgraded to the V.22 bis specification. If only the Bell 212A is of interest, the bandpass filtering could be performed digitally within the TMS32011.

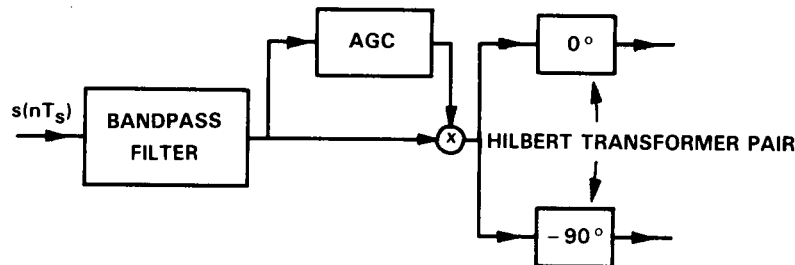


Figure 26. Alternative Modem Receiver Input Configuration

Conclusions

This application report discussed the digital implementation of splitband modems using the TMS32010 general-purpose high-speed digital signal processor. The theory and implementation of the Bell 212A/V.22 full-duplex modem was covered in detail. With a modification of some of the functional blocks of the Bell 212A/V.22, 2400-bps splitband modems (V.22 bis) can be implemented.

Modems are sophisticated devices, consisting of many functional blocks. This implies the need of special features for the microprocessor to be used. The TMS32010 with a 200-ns cycle, an on-board single-cycle multiplier, and a special instruction set tailored for digital signal processing is able to implement the modem functional blocks (see Table 5) with approximately 60-percent use of the available processing power. The modem program utilizes 103 words of data memory out of the 144 words available. This corresponds to approximately 71 percent of the data memory. The program also utilizes 954 words of program memory out of the 1536 words available, corresponding to approximately 62 percent of the on-chip program memory. Therefore, the use of the full-speed off-chip memory feature of the TMS32010 was not utilized. Since a large portion of the power of the TMS32010 is still available, additional functions, such as an adaptive equalizer and the Data Encryption Standard (DES)¹, can be implemented with the inclusion of new code. With a 6-percent loading of the TMS32010, the DES can provide secure communication between 1200-bps full-duplex modems.

The TMS32010 is one of many digital signal processors in the TMS320 family. The flexibility and processing power of the TMS320 family provide high performance, high reliability, and cost-effective solutions for medium- and high-speed modems.

References

1. P.E. Papamichalis and J. Reimer, "Implementation of the Data Encryption Standard with the TMS32010," *Digital Signal Processing Applications with the TMS320 Family*, Texas Instruments (1986).
2. C.F. Foschini, R.D. Gitlin, and S.B. Weinstein, "On the Selection of a Two-Dimensional Signal Constellation in the Presence of Phase Jitter and Gaussian Noise," *Bell System Technical Journal*, Vol. 52, 927-965 (July-August 1973).
3. P.J. Van Gerwen, N.A.M. Verhoeckx, H.A. Van Essen, and F.A.M. Snijders, "Microprocessor Implementation of High Speed Data Modems," *IEEE Trans. on Communications*, Vol. COM-25, No. 2, 238-250 (February 1977).
4. M.J. Di Toro, "Communication in Time Frequency Spread Using Adaptive Equalization," *Proceedings of the IEEE*, Vol. 56, No. 10, 1653-1679 (October 1968).
5. "Data Communications Using Voice Band Private Line Channels," *Bell Systems Technical Reference*, No. 41004 (1973).
6. F.M. Gardner, *Phaselock Techniques*, John Wiley & Sons (1979).
7. M.K. Simon and J.D. Smith, "Carrier Synchronization and Detection of QASK Signal Sets," *IEEE Trans. on Communications*, Vol. COM-22, 98-106 (February 1974).
8. W.C. Lindsey and M.K. Simon, "Carrier Synchronization and Detection of Polyphase Signals," *IEEE Trans. on Communications*, Vol. COM-20, 441-454 (June 1972).
9. H.L. Van Trees, *Detection, Estimation and Modulation Theory*, John Wiley and Sons (1968).
10. *TMS7742 Data Sheet*, Texas Instruments (1985).
11. *TMS7000 Family Data Manual*, Texas Instruments (1986).
12. M. Schwartz, *Information Transmission, Modulation, and Noise*, McGraw-Hill (1970).
13. A. Lovrich and R. Simar, "Implementation of FIR/IIR Filters with the TMS32010/TMS32020," *Digital Signal Processing Applications with the TMS320 Family*, Texas Instruments (1986).
14. T.W. Parks and C.S. Burrus, *Digital Filter Design*, John Wiley and Sons (1987).
15. *TMS32010 User's Guide*, Texas Instruments (1983).
16. A. Papoulis, *The Fourier Integral and Its Applications*, McGraw-Hill (1962).
17. D. Garcia, "Precision Digital Sine-Wave Generation with the TMS32010," *Digital Signal Processing Applications with the TMS320 Family*, Texas Instruments (1986).
18. H. Stark and F.B. Tuteur, *Modern Electrical Communications*, Prentice-Hall (1979).

Appendix A

Derivation of Demodulator Structure Equations

The equations that describe the demodulator structure (see Figure 6) of the modem receiver are derived in this Appendix. The background material required for this derivation is presented first. The following discussion requires a basic knowledge of complex variables.

The baseband signal, at the output of the transmitter digital lowpass filters (see Figure 3), can be expressed as a complex value

$$c(nT_s) = I(nT_s) - j Q(nT_s) \quad (A-1)$$

For transmission through the telephone network, this signal is modulated to the voice frequencies. Modulation involves multiplication by a complex exponential.¹⁸ The modulated signal is then given by

$$m(nT_s) = c(nT_s) e^{j\omega_c nT_s} \quad (A-2)$$

where ω_c is the carrier frequency. Substitution of (A-1) into (A-2), and the use of the identity $e^{j\omega_c nT_s} = \cos(\omega_c nT_s) + j \sin(\omega_c nT_s)$

give

$$\begin{aligned} m(nT_s) = & \{I(nT_s) \cos(\omega_c nT_s) + Q(nT_s) \sin(\omega_c nT_s)\} \\ & + j \{I(nT_s) \sin(\omega_c nT_s) - Q(nT_s) \cos(\omega_c nT_s)\} \end{aligned} \quad (A-3)$$

The real and imaginary parts of (A-3) are later shown to be a Hilbert transform pair. Two signals are referred to as a Hilbert transform pair if they are related with a Hilbert transform. A Hilbert transform is implemented with a filter called a Hilbert transformer. The Hilbert transform pair property that relates the real and imaginary parts of (A-3) allows the transmission of the real part of (A-3) only. The imaginary part is recovered at the receiver by Hilbert transforming the incoming signal. Figure A-1 shows the spectrum of the complex baseband information $c(nT_s)$. Figure A-2 shows the spectrum after modulation by the complex exponential (see equation (A-2)). This is the spectrum of $m(nT_s)$. Figure A-3 shows the spectrum of the transmitted signal, i.e., the spectrum of the real part of $m(nT_s)$.

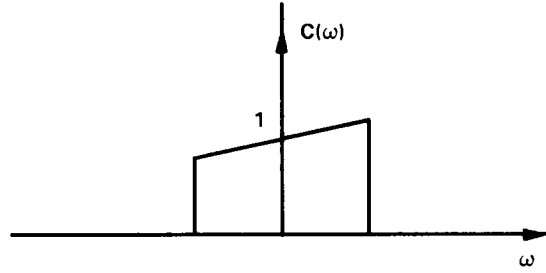


Figure A-1. Spectrum of Complex Baseband Information

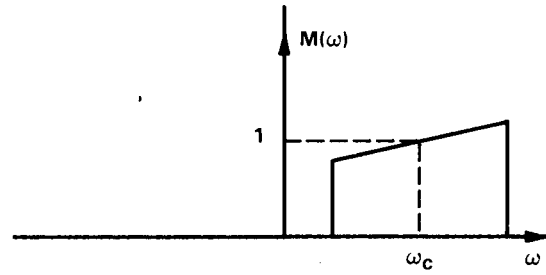


Figure A-2. Spectrum after Modulation

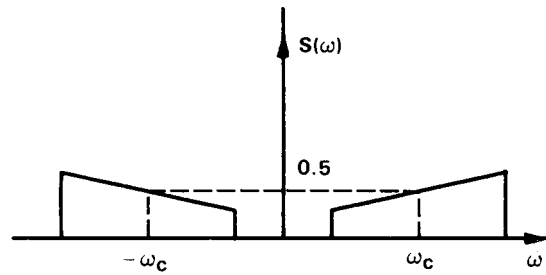


Figure A-3. Transmitted Spectrum

A Hilbert transformer is defined to be a filter with the transfer function¹⁸

$$H_t(\omega) = -e^{j\frac{\pi}{2}} \text{sgn}(\omega) = -j \text{sgn}(\omega) \quad (\text{A-4})$$

where sgn is the sign function defined by equation (29). The transfer function characteristics of the Hilbert transformer are shown in Figure A-4, where it is seen that the Hilbert transformer introduces a -90 degree phase shift for positive frequencies ($\omega > 0$), and a $+90$ degree phase shift for negative frequencies ($\omega < 0$).

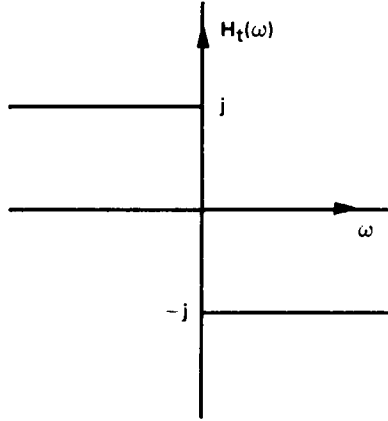


Figure A-4. Hilbert Transformer Transfer Function

The Hilbert transform pair relationship between the real and imaginary parts of (A-3) is discussed next. It is shown that the imaginary part of $m(nT_s)$ is the output of a Hilbert transformer with the input being the real part of $m(nT_s)$. The analysis is performed in the frequency domain where multiplication is replaced by convolution. Let $S(\omega)$ and $\hat{S}(\omega)$ be the Fourier transforms of the real and imaginary parts of $m(nT_s)$, respectively. Then (see equation (A-3))

$$S(\omega) = \frac{1}{2} \{I(\omega - \omega_c) + I(\omega + \omega_c)\} + \frac{j}{2} \{Q(\omega + \omega_c) - Q(\omega - \omega_c)\} \quad (\text{A-5})$$

$$\hat{S}(\omega) = \frac{j}{2} \{I(\omega + \omega_c) - I(\omega - \omega_c)\} - \frac{1}{2} \{Q(\omega - \omega_c) + Q(\omega + \omega_c)\} \quad (\text{A-6})$$

where $I(\omega)$ and $Q(\omega)$ are the Fourier transforms of $I(nT_s)$ and $Q(nT_s)$, respectively.

With $S(\omega)$ as the input to the Hilbert transformer (transfer function $H_t(\omega)$), the output in the frequency domain is given by

$$O(\omega) = S(\omega) H_t(\omega) = -j S(\omega) \text{sgn}(\omega) \quad (\text{A-7})$$

Substitution of (A-5) into (A-7) gives

$$O(\omega) = -j \left\{ \frac{1}{2} \{I(\omega - \omega_c) + I(\omega + \omega_c)\} + \frac{j}{2} \{Q(\omega + \omega_c) - Q(\omega - \omega_c)\} \right\} \text{sgn}(\omega) \quad (\text{A-8})$$

Since for positive frequencies ($\omega > 0$),

$$I(\omega + \omega_c) = 0$$

$$Q(\omega + \omega_c) = 0 \quad (\text{A-9})$$

and for negative frequencies ($\omega < 0$),

$$\begin{aligned} I(\omega - \omega_c) &= 0 \\ Q(\omega - \omega_c) &= 0 \end{aligned} \quad (\text{A-10})$$

equation (A-8) simplifies to

$$O(\omega) = \begin{cases} -\frac{j}{2} I(\omega - \omega_c) - \frac{1}{2} Q(\omega - \omega_c) & \text{where } \omega > 0 \\ \frac{j}{2} I(\omega + \omega_c) - \frac{1}{2} Q(\omega + \omega_c) & \text{where } \omega < 0 \end{cases} \quad (\text{A-11})$$

Substitution of (A-9) and (A-10) into (A-6) and comparison of the result with (A-11) shows that $S(\omega) = O(\omega)$.

Therefore, the real and imaginary parts of $m(nT_s)$ (see equation (A-3)) represent a Hilbert transform pair. With $s(nT_s)$ and $\hat{s}(nT_s)$ denoting the real and imaginary parts of $m(nT_s)$, respectively, (A-3) can be written as

$$m(nT_s) = s(nT_s) + j \hat{s}(nT_s) \quad (\text{A-12})$$

At the receiver end, recovery of the imaginary part $\hat{s}(nT_s)$ involves Hilbert transforming the real part $s(nT_s)$ (incoming signal), as shown in Figure A-5.

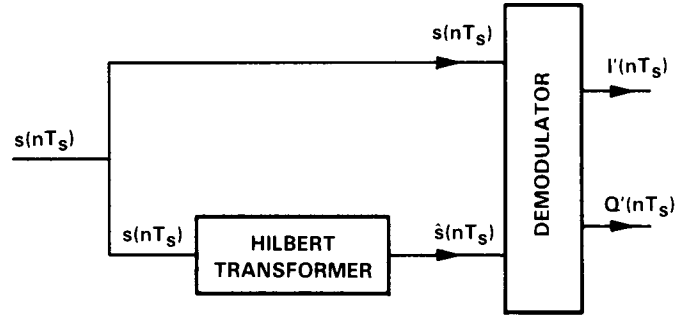


Figure A-5. Recovery of Complex Information by Hilbert Transforming the Incoming Signal

Consider the Fourier transform of (A-12)

$$\begin{aligned} M(\omega) &= S(\omega) + j \hat{S}(\omega) \\ &= S(\omega) + j \{ -j S(\omega) \operatorname{sgn}(\omega) \} \\ &= S(\omega) + S(\omega) \operatorname{sgn}(\omega) \end{aligned}$$

Therefore,

$$M(\omega) = \begin{cases} 2 S(\omega) & \text{where } \omega > 0 \\ 0 & \text{where } \omega < 0 \end{cases} \quad (\text{A-13})$$

i.e., the spectrum of $m(nT_s)$ is zero for negative frequencies (see Figure A-2). If this property does not hold due to the use of a nonideal Hilbert transformer, harmonics appear at the output of the receiver demodulator (see Appendix B).

The equations that describe the receiver demodulator are derived next. The demodulator translates the recovered complex modulated information back to the baseband. This is accomplished by multiplying the passband information with a complex exponential.

$$c'(nT_s) = m(nT_s) e^{-j\omega_c nT_s} \quad (\text{A-14})$$

where $c'(nT_s)$ is the recovered baseband signal, $m(nT_s)$ is the passband signal given by (A-12), and ω_c is the carrier frequency recovered at the receiver by the carrier recovery algorithm.

Substitution of (A-12) into (A-14) gives

$$\begin{aligned} c'(nT_s) = & \{s(nT_s) \cos(\omega_c nT_s) + \hat{s}(nT_s) \sin(\omega_c nT_s)\} \\ & + j \{ \hat{s}(nT_s) \cos(\omega_c nT_s) - s(nT_s) \sin(\omega_c nT_s) \} \end{aligned} \quad (\text{A-15})$$

The complex baseband information $c'(nT_s)$ is also given by (see equation (A-1) and Figure 5)

$$c'(nT_s) = I'(nT_s) - j Q'(nT_s) \quad (\text{A-16})$$

Equating the real and imaginary parts of (A-15) to those of (A-16) results in

$$I'(nT_s) = s(nT_s) \cos(\omega_c nT_s) + \hat{s}(nT_s) \sin(\omega_c nT_s) \quad (\text{A-17})$$

$$Q'(nT_s) = s(nT_s) \sin(\omega_c nT_s) - \hat{s}(nT_s) \cos(\omega_c nT_s) \quad (\text{A-18})$$

Equations (A-17) and (A-18) describe the receiver demodulator of Figure 6.

Appendix B

Effects of Nonideal Hilbert Transformers

The effect of nonideal Hilbert Transformers in modem design is studied in this Appendix. The following discussion requires a basic knowledge of complex variables.

The nonideal Hilbert transformer characteristics differ from the ideal ones shown in Figure 28 and described by equation (A-4) in Appendix A. The phase shift introduced by the nonideal filter is not exactly 90 degrees. The transfer function characteristics of such a filter are given by

$$H'(\omega) = -e^{j(\frac{\pi}{2} + \alpha)} \text{sgn}(\omega) = -j e^{j\alpha} \text{sgn}(\omega) \quad (\text{B-1})$$

where ' α ' is a nonzero constant indicating the deviation from the ideal filter.

Consider the effect of a nonideal Hilbert transformer described by equation (B-1). The incoming signal $s(nT_s)$ is the real part of $m(nT_s)$. This signal is filtered by the nonideal Hilbert transformer to generate at the output a signal $\hat{s}'(nT_s)$ different from $\hat{s}(nT_s)$ (see Appendix A). With $\hat{S}'(\omega)$ as the Fourier transform of $\hat{s}'(nT_s)$, the output of the nonideal Hilbert transformer can be described in the frequency domain by

$$\hat{S}'(\omega) = H'(\omega) S(\omega) = -j e^{j\alpha} \text{sgn}(\omega) S(\omega) \quad (\text{B-2})$$

The complex signal at the input of the receiver demodulator is described by

$$m'(nT_s) = s(nT_s) + j \hat{s}'(nT_s) \quad (\text{B-3})$$

The frequency-domain equivalent of (B-3) is

$$M'(\omega) = S(\omega) + j \hat{S}'(\omega) \quad (\text{B-4})$$

Substitution of (B-2) into (B-4) gives

$$M'(\omega) = S(\omega) + e^{j\alpha} \text{sgn}(\omega) S(\omega) \quad (\text{B-5})$$

Equation (B-5) can be written as

$$M'(\omega) = \begin{cases} S(\omega) \{ 1 + e^{j\alpha} \} & \text{where } \omega > 0 \\ S(\omega) \{ 1 - e^{j\alpha} \} & \text{where } \omega < 0 \end{cases} \quad (\text{B-6})$$

For a nonideal Hilbert transformer, the parameter ' α ' is nonzero. This results in $M'(\omega)$ having nonzero components at negative frequencies as indicated by (B-6). The spectrum of the signal at the input of the receiver demodulator is shown in Figure B-1. Comparison of Figures A-2 and B-1 indicates that the effect of the nonideal Hilbert transformer is the generation of nonzero spectral components at negative frequencies.

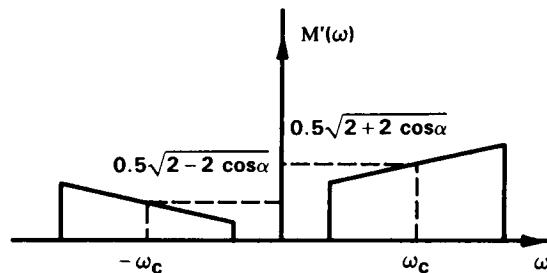


Figure B-1. Effect of Nonideal Hilbert Transformer on the Spectrum of the Complex Signal at the Input of the Demodulator

The effect of the receiver demodulator on the spectrum of Figure B-1 is shown in Figure B-2.

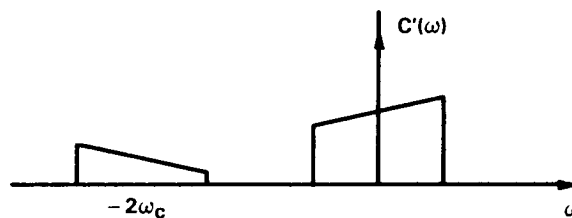


Figure B-2. Effect of Nonideal Hilbert Transformer on the Spectrum of the Baseband Complex Signal

Figure B-2 indicates that harmonics appear at the output of the demodulator. The frequency of these harmonics is twice the carrier frequency. Their elimination involves the use of lowpass filters at the output of the demodulator. These filters, however, introduce group delay and possibly phase delay effects that affect the carrier recovery and decision algorithms. Compensation for the lowpass filter side-effects results in a more complicated modem receiver design. Such nonideal Hilbert transformers are encountered in analog modems. This appendix has demonstrated one more advantage of a digital implementation of a modem using the TMS32010 digital signal processor.

Appendix C
Automatic Gain Control Table Generator Code

```

10 ' THIS PROGRAM GENERATES THE GAIN TABLE FOR THE AUTOMATIC
20 ' GAIN CONTROL ALGORITHM IN THE MODEM CODE
30 '
40 ' THE PROGRAM PROMPTS THE USER IN THE FOLLOWING MANNER:
50 '
60 ' AGC TABLE ADJUST FACTOR ?
70 ' This feature allows the AGC to gain to a level lower
80 ' than unity. The entry for unity gain is 256, to set
90 ' the gain lower than unity enter the appropriate per-
100 ' centage of 256.
110 '
120 ' ENTER NAME OF OUTPUT FILE =
130 ' This prompt request the name of a MSDOS format file
140 ' name to store the generated table.
150 '
160 ' TABLE LENGTH =
170 ' This feature allows the user to generate different
180 ' length AGC tables. This allows the accuracy of the
190 ' table to vary by the number of entries. The number
200 ' of entries is tied to the number of bits used in the
210 ' table lookup. In the modem algorithm six bits were
220 ' used in the lookup, therefore the table length will be
230 ' 64 words.
240 '
250 ' THE TABLE GENERATED WILL INCLUDE DESCRIPTIVE COMMENTS AND WILL
260 ' BE IN A FORM READY TO BE ADDED DIRECTLY INTO THE ASSEMBLY CODE
270 ' FOR AN ALGORITHM. SINCE THE AGC SOFTWARE SHIFTS THE LOOKUP
280 ' VALUE TO THE MOST SIGNIFICANT BIT THE FIRST HALF OF THE AGC TABLE
290 ' (THE LESS ACCURATE HALF) WILL NOT BE USED. THEREFORE THE USER
300 ' CAN DELETE THE FIRST HALF AND SAVE A CONSIDERABLE AMOUNT OF PROGRAM
310 ' MEMORY SPACE.
320 '
330 ' THIS PROGRAM WAS WRITTEN BY PETER EHLIG FOR USE ON A
340 ' TEXAS INSTRUMENTS PROFESSIONAL COMPUTER
350 ' THE CODE TO MY KNOWLEDGE IS WRITTEN IN STANDARD MS-BASIC AND
360 ' SHOULD OPERATE ON ANY MSDOS SYSTEM.
370 '
380 PRINT "PROGRAM STARTED"
390 DIM TBLD(500),HTB$(500)
400 OPEN "LPT1:" FOR OUTPUT AS #1
410 INPUT "AGC TABLE ADJUSTMENT FACTOR ? ",GAINADJ
420 INPUT "ENTER NAME OF OUTPUT FILE = ",OUTFILE$
430 OPEN OUTFILE$ FOR OUTPUT AS #3
440 P1 = 3.1415927#
450 P12 = P1 * 2
460 INPUT "TABLE LENGTH = ",TBLEN
470 GOSUB 820 ' GENERATE TABLE HEADER
480 DELTA = 32768 / TBLEN
490 FOR I = 1 TO TBLEN
500 TBL = INT(32767 / (I * DELTA) * GAINADJ)
510 TBLD(I) = TBL
520 HTB$(I) = HEX$(TBL)
530 HTB$(I) = HTB$(I)
540 GOSUB 690 ' DISPLAY RANGE ACCURACY (OPTIONAL)
550 NEXT
560 GOTO 650
570 ' SAVE AGC TABLE TO DISK
580 PRINT#3, " DATA ";
590 PRINT#3, USING ">\ \";HTB$(I);
600 PRINT#3, " ";
610 TBLD(I) = TBLD(I) / 256
620 PRINT#3, USING "###.#####";TBLD(I)
630 RETURN
640 ' END OF AGC TABLE SAVE ROUTINE
650 GOSUB 940 ' DISPLAY SECOND LEVEL LOOKUP
660 GOSUB 880 ' GENERATE TABLE TERMINATION COMMENTS
670 PRINT "PROGRAM FINISHED"
680 END

```

```

690 ' THIS ROUTINE DISPLAYS INFORMATION ABOUT THE RANGE
700 ' ACCURACY OF EACH STEP OF THE TABLE
710 TBLRL = (I - 1) * DELTA - 256
720 IF TBLRL < 0 THEN TBLRL = 0
730 SH1$ = HEX$(TBLRL)
740 SH1A$ = HEX$(TBLRL * TBL / 256)
750 TBLRH = (I - 1) * DELTA + 255
760 SH2$ = HEX$(TBLRH)
770 SH2A$ = HEX$(TBLRH * TBL / 256)
780 PRINT I;TBL;HTBL$;" ";SH1$;" ";SH1A$;" ";SH2$;" ";SH2A$
790 ' PRINT#1,I;TBL;HTBL$;" ";SH1$;" ";SH1A$;" ";SH2$;" ";SH2A$
800 RETURN
810 ' END OF RANGE INFORMATION
820 ' THE ROUTINE GENERATES THE HEADER COMMENTS FOR THE TABLE
830 PRINT#3,"....."
840 PRINT#3,"AGCTBL EQU $ AGC TABLE LENGTH = ";
850 PRINT#3, USING "###";TBLN
860 RETURN
870 ' END OF HEADER ROUTINE
880 ' THIS ROUTINE GENERATES THE TABLE TERMINATION COMMENTS
890 PRINT#3,"....."
900 PRINT#3, " PAGE"
910 CLOSE
920 RETURN
930 ' END OF TERMINATOR ROUTINE
940 ' TRY SECOND LEVEL LOOKUP
950 DELTA1 = DELTA * 8
960 FOR I = 1 TO 64
970 GOSUB 570 ' SAVE AGC TABLE TO DISK
980 TBLRL = (I - 1) * DELTA - 256
990 IF TBLRL < 0 THEN TBLRL = 0
1000 TBLRH = (I - 1) * DELTA + 255
1010 SH1$ = HEX$(TBLRL)
1020 SH2$ = HEX$(TBLRH)
1030 GOSUB 1100 ' CALCULATE ACCURACY STEPS
1040 SH1A$ = HEX$(TBLRL * TBLD(TBLR1) / SHF1)
1050 SH2A$ = HEX$(TBLRH * TBLD(TBLR2) / SHF1)
1060 PRINT I;TBL;HTBL$;" ";SH1$;" ";SH1A$;" ";SH2$;" ";SH2A$;TBLR1;TBLR2;SHF1
1070 ' PRINT#1,I;TBL;HTBL$;" ";SH1$;" ";SH1A$;" ";SH2$;" ";SH2A$;TBLR1;TBLR2;SHF1
1080 NEXT I
1090 RETURN
1100 'TABLE LOOKUP SHIFTER
1110 TBLEV = TBLRH - 4096
1120 IF TBLEV > 0 GOTO 1180
1130 TBLEV = TBLEV + 2048
1140 IF TBLEV > 0 GOTO 1220
1150 TBLEV = TBLEV + 1024
1160 IF TBLEV > 0 GOTO 1260
1170 GOTO 1300
1180 TBLR1 = I
1190 TBLR2 = I
1200 SHF1 = 256
1210 RETURN
1220 TBLR2 = FIX(TBLRH / 64) + 1
1230 TBLR1 = FIX(TBLRL / 64) + 1
1240 SHF1 = 32
1250 RETURN
1260 TBLR2 = FIX(TBLRH / 32) + 1
1270 TBLR1 = FIX(TBLRL / 32) + 1
1280 SHF1 = 16
1290 RETURN
1300 TBLR2 = FIX(TBLRH / 16) + 1
1310 TBLR1 = FIX(TBLRL / 16) + 1
1320 SHF1 = 8
1330 RETURN

```


The following 64-point table was generated using the 124 for the AGC table adjust factor.

```

*****
AGCTBL EQU $          AGC TABLE LENGTH = 64
DATA >1EFF             30.9961000
DATA >F7F              15.4960900
DATA >A55              10.3320300
DATA >7BF              7.7460940
DATA >633              6.1992190
DATA >52A              5.1640630
DATA >46D              4.4257820
DATA >3DF              3.8710940
DATA >371              3.4414060
DATA >319              3.0976560
DATA >2D1              2.8164060
DATA >295              2.5820310
DATA >262              2.3828130
DATA >236              2.2109380
DATA >211              2.0664060
DATA >1EF              1.9335940
DATA >1D2              1.8203130
DATA >1B8              1.7187500
DATA >1A1              1.6289060
DATA >18C              1.5468750
DATA >179              1.4726560
DATA >168              1.4062500
DATA >159              1.3476560
DATA >14A              1.2890630
DATA >13D              1.2382810
DATA >131              1.1914060
DATA >125              1.1445310
DATA >11B              1.1054690
DATA >111              1.0664060
DATA >108              1.0312500
DATA >FF              0.9960938
DATA >F7              0.9648438
DATA >F0              0.9375000
DATA >E9              0.9101562
DATA >E2              0.8828125
DATA >DC              0.8593750
DATA >D6              0.8359375
DATA >D0              0.8125000
DATA >CB              0.7929688
DATA >C6              0.7734375
DATA >C1              0.7539063
DATA >BC              0.7343750
DATA >B8              0.7187500
DATA >B4              0.7031250
DATA >B0              0.6875000
DATA >AC              0.6718750
DATA >A8              0.6562500
DATA >A5              0.6445313
DATA >A1              0.6289063
DATA >9E              0.6171875
DATA >9B              0.6054688
DATA >98              0.5937500
DATA >95              0.5820313
DATA >92              0.5703125
DATA >90              0.5625000
DATA >8D              0.5507813
DATA >8B              0.5429688
DATA >88              0.5312500
DATA >86              0.5234375
DATA >84              0.5156250
DATA >82              0.5078125
DATA >7F              0.4960938

```

DATA >7D 0.4882813
DATA >7B 0.4804688

.....
PAGE

```
10 /
20 /   This program generates sine table in a format compatible
30 /   to the 320 assembler. This allows the user to generate
40 /   any length sine table and this program will calculate the
50 /   table entries, configure them in a format compatible to
60 /   the assembler, and document the code.
70 /
80 /   The program prompts the user in the following manner:
90 /
100 /       ENTER NAME OF OUTPUT FILE =
110 /           This prompt request the name of a MSDOS format file
120 /           name to store the generated table.
130 /
140 /       TABLE LENGTH =
150 /           This feature allows the user to select the length of
160 /           the sine table to be generated and therefore the
170 /           accuracy of the table steps.
180 /
190 /
200 /   This program was written by Peter Ehlig for use on a
210 /   Texas Instruments Professional Computer
220 /   The code to my knowledge is written in standard MS-BASIC and
230 /   should operate on any MSDOS system.
240 /
250 PRINT "PROGRAM STARTED"
260 INPUT "ENTER NAME OF OUTPUT FILE = ",OUTFILE$
270 OPEN OUTFILE$ FOR OUTPUT AS #3
280 PI = 3.1415927#
290 PI2 = PI * 2
300 INPUT "TABLE LENGTH = ",TBLEN
310 DELTA = PI2 / TBLEN
320 INDX1 = -DELTA
330 NETDEG = 360 / TBLEN
340 PRINT#3,"....."
350 PRINT#3,"SINE EQU $          SINE TABLE LENGTH = ";
360 PRINT#3, USING "###";TBLEN
370 FOR I = 1 TO TBLEN
380 INDX1 = INDX1 + DELTA
390 TBL = SIN(INDX1)
400 HTBL$ = HEX$(TBL*16384)
410 RADS = INDX1 / PI
420 DEGR = NETDEG * (I - 1)
430 PRINT#3, "      DATA ";
440 PRINT#3, USING ">\ ";HTBL$;
450 PRINT#3, "      ANGLE = ";
460 PRINT#3, USING "###.####";DEGR;
470 PRINT#3, "      SINE = ";
480 PRINT#3, USING "#.#####";TBL
490 NEXT I
500 PRINT#3,"....."
510 PRINT#3, "      PAGE"
520 CLOSE
530 PRINT "PROGRAM FINISHED"
540 END
```

```

.....
SINE EQU $ SINE TABLE LENGTH = 32
DATA >0 ANGLE = 0.0000 SINE = 0.000000
DATA >C7C ANGLE = 11.2500 SINE = 0.195090
DATA >187E ANGLE = 22.5000 SINE = 0.382683
DATA >238E ANGLE = 33.7500 SINE = 0.555570
DATA >2D41 ANGLE = 45.0000 SINE = 0.707107
DATA >3537 ANGLE = 56.2500 SINE = 0.831470
DATA >3B21 ANGLE = 67.5000 SINE = 0.923880
DATA >3EC5 ANGLE = 78.7500 SINE = 0.980785
DATA >4000 ANGLE = 90.0000 SINE = 1.000000
DATA >3EC5 ANGLE = 101.2500 SINE = 0.980785
DATA >3B21 ANGLE = 112.5000 SINE = 0.923880
DATA >3537 ANGLE = 123.7500 SINE = 0.831470
DATA >2D41 ANGLE = 135.0000 SINE = 0.707107
DATA >238E ANGLE = 146.2500 SINE = 0.555570
DATA >187E ANGLE = 157.5000 SINE = 0.382683
DATA >C7C ANGLE = 168.7500 SINE = 0.195090
DATA >0 ANGLE = 180.0000 SINE = -.000000
DATA >F384 ANGLE = 191.2500 SINE = -.195091
DATA >E782 ANGLE = 202.5000 SINE = -.382684
DATA >DC72 ANGLE = 213.7500 SINE = -.555571
DATA >D2BF ANGLE = 225.0000 SINE = -.707107
DATA >CAC9 ANGLE = 236.2500 SINE = -.831470
DATA >C4DF ANGLE = 247.5000 SINE = -.923880
DATA >C13B ANGLE = 258.7500 SINE = -.980786
DATA >C000 ANGLE = 270.0000 SINE = -1.000000
DATA >C13B ANGLE = 281.2500 SINE = -.980785
DATA >C4DF ANGLE = 292.5000 SINE = -.923879
DATA >CAC9 ANGLE = 303.7500 SINE = -.831469
DATA >D2BF ANGLE = 315.0000 SINE = -.707106
DATA >DC72 ANGLE = 326.2500 SINE = -.555569
DATA >E782 ANGLE = 337.5000 SINE = -.382682
DATA >F384 ANGLE = 348.7500 SINE = -.195089
.....
PAGE

```

Appendix D
TMS32010 Source Code

```

*
*****
***          - DSP MODEM PROGRAM -          ***
*****
*** THIS CODE IMPLEMENTS A BELL 212A / V.22 MODEM ***
*** ON THE TMS32010.                             ***
***
*** SCRAMBLING AND DESCRAMBLING ARE IMPLEMENTED ***
*** ON THE TMS7742.                             ***
*****

IDT      'TASK6212'
OPTION   XREF
AORG     0
B        START

*****-----*****
*****          DATA MEMORY USED.          *****
*****-----*****

XDELTA EQU 0      * SWAVE MACRO CARRIER RATE
XALPHA EQU 1      * SWAVE MACRO CARRIER ANGLE
SINA EQU 2        * XMIT SIN CARRIER MAGNETUDE
COSA EQU 3        * XMIT COS CARRIER MAGNETUDE
ONE EQU 4         * VALUE 1 HELD FOR MASKING
MASK1 EQU 5       * SWAVE MACRO TBL RANGE ADJ >7F
MASK2 EQU 6       * SWAVE MACRO TBL RANGE ADJ >7FFF
MASK3 EQU 7       * XMIT PHASE ENCODE MASK >0006
OFSET0 EQU 8      * SWAVE MACRO POINT TO COS TABLE
OFSET1 EQU 9      * XMIT POINT TO DIBIT ENCODE TABLE
XPTR EQU 10       * XMIT POINT TO RAISED COS TABLE
CX0 EQU 11        * XMIT COEF FOR RAISED COS
CX1 EQU 12        * XMIT COEF FOR RAISED COS
CX2 EQU 13        * XMIT COEF FOR RAISED COS
XIBUF0 EQU 14     * XMIT STORE DATA FOR RAISED COS
XIBUF1 EQU 15     * XMIT STORE DATA FOR RAISED COS
XIBUF2 EQU 16     * XMIT STORE DATA FOR RAISED COS
XQBUF0 EQU 17     * XMIT STORE DATA FOR RAISED COS
XQBUF1 EQU 18     * XMIT STORE DATA FOR RAISED COS
XQBUF2 EQU 19     * XMIT STORE DATA FOR RAISED COS
XIOUT EQU 20      * XMIT HOLD FILTERED I VALUE
XQOUT EQU 21      * XMIT HOLD FILTERED Q VALUE
XMTOUT EQU 22     * XMIT HOLD FOR TRANSMIT OUTPUT
XOLDPH EQU 23     * XMIT HOLD LAST PHASE
XNEWPH EQU 24     * XMIT HOLD NEW PHASE
RDIBIT EQU 25     * DECODED DIBIT
INDXPH EQU 26     * XMIT POINT TO PHASE ENCODE TABLE
XDIBIT EQU 27     * XMIT DIBIT ISOLATION MASK
PLUS1 EQU 28      * +1 Q12 >FFF & MASK VALUE
XMTD EQU 29       * XMIT HOLD DTE INPUT
RBUF0 EQU 30      * HOLD LOWPASS FILTERED SAMPLE
RBUF1 EQU 31      % RECEIVE BPF COEFFICIENT
RBUF2 EQU 32      % RECEIVE BPF COEFFICIENT
RBUF3 EQU 33      % RECEIVE BPF COEFFICIENT
RBUF4 EQU 34      % RECEIVE BPF COEFFICIENT
RBUF5 EQU 35      % RECEIVE BPF COEFFICIENT
RBUF6 EQU 36      % RECEIVE BPF COEFFICIENT
RBUF7 EQU 37      % RECEIVE BPF COEFFICIENT
RBUF8 EQU 38      % RECEIVE BPF COEFFICIENT
RBUF9 EQU 39      % RECEIVE BPF COEFFICIENT
RBUF10 EQU 40     % RECEIVE BPF COEFFICIENT
RBUF11 EQU 41     % RECEIVE BPF COEFFICIENT
RBUF12 EQU 42     % RECEIVE BPF COEFFICIENT
RBUF13 EQU 43     % RECEIVE BPF COEFFICIENT
RBUF14 EQU 44     % RECEIVE BPF COEFFICIENT
RBUF15 EQU 45     % RECEIVE BPF COEFFICIENT
RBUF16 EQU 46     % RECEIVE BPF COEFFICIENT
RBUF17 EQU 47     % RECEIVE BPF COEFFICIENT
RBUF18 EQU 48     % RECEIVE BPF COEFFICIENT
RBUF19 EQU 49     % RECEIVE BPF COEFFICIENT
RBUF20 EQU 50     % RECEIVE BPF COEFFICIENT
RBUF21 EQU 51     % RECEIVE BPF COEFFICIENT

```

```

RBUF22 EQU 52      % RECEIVE BPF COEFFICIENT
RBUF23 EQU 53      % RECEIVE BPF COEFFICIENT
RBUF24 EQU 54      % RECEIVE BPF COEFFICIENT
RBUF25 EQU 55      % RECEIVE BPF COEFFICIENT
RBUF26 EQU 56      % RECEIVE BPF COEFFICIENT
RBUF27 EQU 57      % RECEIVE BPF COEFFICIENT
RBUF28 EQU 58      % RECEIVE BPF COEFFICIENT
RBUF29 EQU 59      % RECEIVE BPF COEFFICIENT
RBUF30 EQU 60      % RECEIVE BPF COEFFICIENT
RBUF31 EQU 61      % RECEIVE BPF COEFFICIENT
RBUF32 EQU 62      % RECEIVE BPF COEFFICIENT
RBUF33 EQU 63      % RECEIVE BPF COEFFICIENT
RBUF34 EQU 64      % RECEIVE BPF COEFFICIENT
RBUF35 EQU 65      % RECEIVE BPF COEFFICIENT
RBUF36 EQU 66      % RECEIVE BPF COEFFICIENT
RBUF37 EQU 67      % RECEIVE BPF COEFFICIENT
AGC EQU 68         * AUTOMATIC GAIN FACTOR
AGCRA EQU 69       * SIGNAL MAX RUNNING AVERAGE FOR AGC
RECST EQU 70       * RECEIVER STATUS
AGCOFF EQU 71      * AGC CALCULATION LOOKUP TABLE
BSMAX EQU 72       * BAUD SIGNAL MAX
AGCNT EQU 73       * BAUD SAMPLE COUNT
AGCLEV EQU 74      * TEMPORARY AGC LEVEL (AGCUPT)
SAMPLE EQU 75      * BAUD LIMIT SAMPLE COUNT
SAMXMT EQU 76      * TRANSMITTER SAMPLE COUNT
BITOUT EQU 77      * DIBIT POSITIONED TO XMIT TO 7041
RPHSE EQU 78       * OFFSET FOR RECEIVE PHASE DECODE
TRSHD1 EQU 79      * THRESHOLD FOR CARRIER RECOVERY
RALPHA EQU 80      * RECEIVE CARRIER POINTER
RDELTA EQU 81      * DELTA TO GENERATE RECEIVE CARRIER
ISUM EQU 82        * FILTERED/PHASE SHIFTED SAMPLE
QSUM EQU 83        * FILTERED/PHASE SHIFTED SAMPLE
RECI EQU 84        * BASEBAND I CHANNEL
ROLOPH EQU 85      * PREVIOUS ABSOLUTE PHASE (QUADRANT)
RNEWPH EQU 86      * CURRENT ABSOLUTE PHASE (QUADRANT)
ERRSIG EQU 87      * FILTERED CARRIER ERROR SIGNAL
MINUS1 EQU 88      * MINUS 1 IN THE Q12 FORMAT
PLL1 EQU 89        * CARRIER RECOVERY PLL FILTER COEFFICIENT 1
PLL2 EQU 90        * CARRIER RECOVERY PLL FILTER COEFFICIENT 2
FOUR EQU 91        * >4 ( MASK VALUE FOR PHASE CODE/DECODE)
SIGNI EQU 92       * SIGN OF I CHANNEL (TO COMPUTE CARRIER ERROR)
SIGNQ EQU 93       * SIGN OF Q CHANNEL (TO COMPUTE CARRIER ERROR)
ERROR EQU 94       * CARRIER PHASE ERROR
TEMP EQU 95        * MISC. TEMPORARY REGISTER
RECQ EQU 96        * BASEBAND Q CHANNEL
*-----DEFINE REGISTERS FOR BAUD CLOCK
ENRGY EQU 97       * CURRENT ENERGY
PENRGY EQU 98      * PREVIOUS ENERGY
BERROR EQU 99      * BAUD CLOCK ERROR
BEROUT EQU 100     * OUTPUT OF BAUD PLL LOOP FILTER
BPLL1 EQU 101      * CLOCK RECOVERY PLL FILTER COEFFICIENT 1
BPLL2 EQU 102      * CLOCK RECOVERY PLL FILTER COEFFICIENT 2
TRSHD2 EQU 103     * CLOCK RECOVERY TRESHOLD
*-----
***** TRANSMITTER DIBIT ENCODER TABLE. *****
*****
ENCODE DATA >0002  * DIBIT '01' = 90 deg.
DATA >0000  * DIBIT '00' = 0 deg.
DATA >0004  * DIBIT '10' = 180 deg.
DATA >0006  * DIBIT '11' = 270 deg.
XPHASE DATA >7FFF  * 0 deg. I CHANNEL = 1
DATA >0000  * Q CHANNEL = 0
DATA >0000  * 90 deg. I CHANNEL = 0
DATA >8000  * Q CHANNEL = -1
DATA >8000  * 180 deg. I CHANNEL = -1
DATA >0000  * Q CHANNEL = 0
DATA >0000  * 270 deg. I CHANNEL = 0
DATA >7FFF  * Q CHANNEL = 1
*****

```

```

*****      RECEIVER DIBIT ENCODER TABLE.      *****
*****      DIBITS are formed as 'MSB,LSB'.      *****
*****

```

```

RPHASE  DATA  >0001      * 0 deg., DIBIT = '01'
          DATA  >0000      * 90 deg., DIBIT = '00'
          DATA  >0002      * 180 deg., DIBIT = '10'
          DATA  >0003      * 270 deg., DIBIT = '11'
*
M1       DATA  >7FFF      * MASK 1
M2       DATA  >007F      * MASK 2
M3       DATA  >0006      * MASK 3
CK       DATA  >0208      * CLOCK FOR A1B
MD       DATA  >000A      * MODE FOR A1B
ST       DATA  >1800
          DATA  >0000
DT       DATA  >1000      * TRANSMIT DELTA.
          DATA  >2000      * RECEIVE DELTA.
TH1      DATA  >0007      * 0.01 Q12 TRSHD FOR CARRIER
TH2      DATA  >0008      * 0.01 Q12 TRSHD FOR BAUD CLOCK
MIN1     DATA  >F000      * -1 Q12
PLS1     DATA  >0FFF      * 1 Q12
*

```

```

*****      PLL LOOP FILTER COFFECIENTS.      *****
*****

```

```

PLLC1   DATA  >4500      * Q15 CARRIER PLL INITIAL COEF. 1
PLLC2   DATA  >80        * Q15 CARRIER PLL COEFFICIENT 2
BPLLC1  DATA  >4000      * Q15 BAUD CLOCK PLL INITIAL COEF. 1
BPLLC2  DATA  >50        * Q15 BAUD CLOCK PLL COEFFICIENT 2
PLLC    DATA  >7A00      * Q15 CARRIER PLL STEADY STATE COEF. 1
BPLLC   DATA  >7500      * Q15 BAUD CLOCK PLL STEADY STATE COEF. 1
*

```

```

*****      TASK MASTER SEQUENCE TABLE (RECEIVE)      *****
*****      TASKS ARE EXECUTED FROM BOTTOM TO TOP      *****
*****

```

```

TSKSEQ  EQU  $
          DATA DUMMY      UNUSED CYCLE      15
          DATA DUMMY      UNUSED CYCLE      14
          DATA DUMMY      UNUSED CYCLE      13
          DATA DUMMY      UNUSED CYCLE      12
          DATA BDCLK2     COMPUTE ENERGY E(11) 11
          DATA DUMMY      UNUSED CYCLE      10
          DATA OUT        COMMUNICATE WITH TMS7742 9
          DATA DECODE     DECODE/GET SCRAMBLED DIBIT 8
          DATA DEMODB     DEMODULATE IN THE MIDDLE OF BAUD 7
          DATA DUMMY      UNUSED CYCLE      6
          DATA AGCUPT     UPDATE THE AGC (IF NECESSARY) 5
          DATA DUMMY      UNUSED CYCLE      4
          DATA BDCLK1     COMPUTE ENERGY E(3) 3
          DATA DUMMY      UNUSED CYCLE      2
          DATA DUMMY      UNUSED CYCLE      1
          DATA DUMMY      UNUSED CYCLE      0
*

```

```

*****      TASK MASTER SEQUENCE TABLE (TRANSMIT)      *****
*****      TASKS ARE EXECUTED FROM BOTTOM TO TOP      *****
*****

```

```

TSKXMT  EQU  $
          DATA GETDBT     GET THE NEXT DIBIT      16
          DATA DUMXMT     NO CYCLE      15
          DATA DUMXMT     NO CYCLE      14
          DATA DUMXMT     NO CYCLE      13
          DATA DUMXMT     NO CYCLE      12
          DATA DUMXMT     NO CYCLE      11
          DATA DUMXMT     NO CYCLE      10

```

DATA DUMXMT	NO CYCLE	9
DATA DUMXMT	NO CYCLE	8
DATA DUMXMT	NO CYCLE	7
DATA DUMXMT	NO CYCLE	6
DATA DUMXMT	NO CYCLE	5
DATA DUMXMT	NO CYCLE	4
DATA DUMXMT	NO CYCLE	3
DATA DUMXMT	NO CYCLE	2
DATA DUMXMT	NO CYCLE	1
PAGE		

 ***** RAISED COSINE COEFFICIENT TABLE. *****

COEF	DATA	>1
	DATA	>49A
	DATA	>394
	DATA	>FFD9
	DATA	>5A2
	DATA	>29A
	DATA	>FFAB
	DATA	>6A0
	DATA	>1B5
	DATA	>FF7A
	DATA	>789
	DATA	>ED
	DATA	>FF4C
	DATA	>853
	DATA	>45
	DATA	>FF27
	DATA	>8F4
	DATA	>FFC3
	DATA	>FF11
	DATA	>963
	DATA	>FF65
	DATA	>FF10
	DATA	>99C
	DATA	>FF2A
	DATA	>FF2A
	DATA	>99C
	DATA	>FF10
	DATA	>FF65
	DATA	>963
	DATA	>FF11
	DATA	>FFC3
	DATA	>8F4
	DATA	>FF27
	DATA	>45
	DATA	>853
	DATA	>FF4C
	DATA	>ED
	DATA	>789
	DATA	>FF7A
	DATA	>1B5
	DATA	>6A0
	DATA	>FFAB
	DATA	>29A
	DATA	>5A2
	DATA	>FFD9
	DATA	>394
	DATA	>49A
	DATA	>1

 ***** AGC DIVIDE LOOKUP TABLE *****
 ***** STANDARD GAIN RANGE -- >3CC3 - >3F79 *****
 ***** WITH 5% SIGNAL VARIATION -- >3966 - >41D6 *****

```

AGCTBL EQU $-32      AGC TABLE LENGTH = 32
DATA >F8              0.9687500
DATA >F0              0.9375000
DATA >EA              0.9140625
DATA >E3              0.8867188 35 -
DATA >DD              0.8632812
DATA >D7              0.8398438
DATA >D2              0.8203125
DATA >CC              0.7968750 39 -
DATA >C7              0.7773438
DATA >C3              0.7617188
DATA >8E              0.7421875
DATA >8A              0.7265625 43 -
DATA >86              0.7109375
DATA >82              0.6953125
DATA >AE              0.6796875
DATA >AA              0.6640625 47 -
DATA >A7              0.6523438
DATA >A3              0.6367188
DATA >A0              0.6250000
DATA >9D              0.6132813 51 -
DATA >9A              0.6015625
DATA >97              0.5898438
DATA >94              0.5781250
DATA >92              0.5703125 55 -
DATA >8F              0.5585938
DATA >8D              0.5507813
DATA >8A              0.5390625
DATA >88              0.5312500 59 -
DATA >86              0.5234375
DATA >84              0.5156250
DATA >82              0.5078125
DATA >7F              0.4960938 63 -
PAGE

```

```

*
*****-----*****
*****      SINE(COSINE) TABLE      *****
*****-----*****
SINE DATA >0
DATA >648
DATA >C8C
DATA >12C8
DATA >18F9
DATA >1F1A
DATA >2528
DATA >2B1F
DATA >30FC
DATA >36BA
DATA >3C57
DATA >41CE
DATA >471D
DATA >4C40
DATA >5134
DATA >55F6
DATA >5A82
DATA >5E07
DATA >62F2
DATA >66D0
DATA >6A6E
DATA >6DCA
DATA >70E3
DATA >73B6
DATA >7642
DATA >7885
DATA >7A7D
DATA >7C2A
DATA >7D8A
DATA >7E9D
DATA >7F62

```

	DATA	>7FD9
COSINE	DATA	>7FFF
	DATA	>7FD9
	DATA	>7F62
	DATA	>7E9D
	DATA	>708A
	DATA	>7C2A
	DATA	>7A7D
	DATA	>7885
	DATA	>7642
	DATA	>73B6
	DATA	>70E3
	DATA	>6DCA
	DATA	>6A6E
	DATA	>66D0
	DATA	>62F2
	DATA	>5ED7
	DATA	>5A82
	DATA	>55F6
	DATA	>5134
	DATA	>4C40
	DATA	>471D
	DATA	>41CE
	DATA	>3C57
	DATA	>36BA
	DATA	>30FC
	DATA	>2B1F
	DATA	>2528
	DATA	>1F1A
	DATA	>18F9
	DATA	>12CB
	DATA	>C8C
	DATA	>648
	DATA	>0
	DATA	>F9B8
	DATA	>F374
	DATA	>ED38
	DATA	>E707
	DATA	>E0E6
	DATA	>DAD8
	DATA	>D4E1
	DATA	>CF04
	DATA	>C946
	DATA	>C3A9
	DATA	>BE32
	DATA	>B8E3
	DATA	>B3C0
	DATA	>AECC
	DATA	>AA0A
	DATA	>A57E
	DATA	>A129
	DATA	>9D0E
	DATA	>9930
	DATA	>9592
	DATA	>9236
	DATA	>8F1D
	DATA	>8C4A
	DATA	>89BE
	DATA	>877B
	DATA	>8583
	DATA	>8306
	DATA	>8276
	DATA	>8163
	DATA	>809E
	DATA	>8027
	DATA	>8000
	DATA	>8027
	DATA	>809E
	DATA	>8163

```

DATA      >8276
DATA      >83D6
DATA      >8583
DATA      >8778
DATA      >89BE
DATA      >8C4A
DATA      >8F10
DATA      >9236
DATA      >9592
DATA      >9930
DATA      >9D0E
DATA      >A129
DATA      >A57E
DATA      >AA0A
DATA      >AECC
DATA      >B3C0
DATA      >B8E3
DATA      >BE32
DATA      >C3A9
DATA      >C946
DATA      >CF04
DATA      >D4E1
DATA      >DAD8
DATA      >E0E6
DATA      >E707
DATA      >ED38
DATA      >F374
DATA      >F9B8

```

```

*****
***** RECEIVER 1 CHANNEL BPASS FILTER COEFFICIENTS *****
*****

```

```

ICF0      EQU      58      * 3A 0.014064
*ICF1      EQU      0      * 0 0.000000
ICF2      EQU     -58      * FFC6 -0.014067
*ICF3      EQU      0      * 0 0.000000
ICF4      EQU      28      * 1C 0.006883
*ICF5      EQU      0      * 0 0.000000
ICF6      EQU      37      * 25 0.009069
*ICF7      EQU      0      * 0 0.000000
ICF8      EQU     -137     * FF77 -0.033477
*ICF9      EQU      0      * 0 0.000000
ICF10     EQU      262     * 106 0.063862
*ICF11     EQU      0      * 0 0.000000
ICF12     EQU     -393     * FE77 -0.095882
*ICF13     EQU      0      * 0 0.000000
ICF14     EQU      509     * 1FD 0.124198
*ICF15     EQU      0      * 0 0.000000
ICF16     EQU     -588     * FDB4 -0.143676
*ICF17     EQU      0      * 0 0.000000
ICF18     EQU      617     * 269 0.150616
*ICF19     EQU      0      * 0 0.000000
ICF20     EQU     -588     * FDB4 -0.143676
*ICF21     EQU      0      * 0 0.000000
ICF22     EQU      509     * 1FD 0.124198
*ICF23     EQU      0      * 0 0.000000
ICF24     EQU     -393     * FE77 -0.095882
*ICF25     EQU      0      * 0 0.000000
ICF26     EQU      262     * 106 0.063862
*ICF27     EQU      0      * 0 0.000000
ICF28     EQU     -137     * FF77 -0.033477
*ICF29     EQU      0      * 0 0.000000
ICF30     EQU      37      * 25 0.009069
*ICF31     EQU      0      * 0 0.000000
ICF32     EQU      28      * 1C 0.006883
*ICF33     EQU      0      * 0 0.000000
ICF34     EQU     -58      * FFC6 -0.014067
*ICF35     EQU      0      * 0 0.000000
ICF36     EQU      58      * 3A 0.014064

```

 ***** RECEIVER Q CHANNEL BPASS FILTER COEFFICIENTS *****

*QCF0	EQU	0	* 0 0.000000
QCF1	EQU	61	* 3D 0.014809
*QCF2	EQU	0	* 0 0.000000
QCF3	EQU	-47	* FFD1 -0.011510
*QCF4	EQU	0	* 0 0.000000
QCF5	EQU	0	* 0 0.000034
*QCF6	EQU	0	* 0 0.000000
QCF7	EQU	83	* 53 0.020321
*QCF8	EQU	0	* 0 0.000000
QCF9	EQU	-197	* FF3B -0.048158
*QCF10	EQU	0	* 0 0.000000
QCF11	EQU	328	* 148 0.079991
*QCF12	EQU	0	* 0 0.000000
QCF13	EQU	-454	* FE3A -0.110844
*QCF14	EQU	0	* 0 0.000000
QCF15	EQU	554	* 22A 0.135320
*QCF16	EQU	0	* 0 0.000000
QCF17	EQU	-610	* FD9E -0.148859
*QCF18	EQU	0	* 0 0.000000
QCF19	EQU	610	* 262 0.148859
*QCF20	EQU	0	* 0 0.000000
QCF21	EQU	-554	* FDD6 -0.135320
*QCF22	EQU	0	* 0 0.000000
QCF23	EQU	454	* 1C6 0.110844
*QCF24	EQU	0	* 0 0.000000
QCF25	EQU	-328	* FEB8 -0.079991
*QCF26	EQU	0	* 0 0.000000
QCF27	EQU	197	* C5 0.048158
*QCF28	EQU	0	* 0 0.000000
QCF29	EQU	-83	* FFAD -0.020321
*QCF30	EQU	0	* 0 0.000000
QCF31	EQU	0	* 0 -0.000034
*QCF32	EQU	0	* 0 0.000000
QCF33	EQU	47	* 2F 0.011510
*QCF34	EQU	0	* 0 0.000000
QCF35	EQU	-61	* FFC3 -0.014809
*QCF36	EQU	0	* 0 0.000000

PAGE

 ***** Initialization routine *****

START	DINT	0	
	LDPK		
	ROVM		
	LACK	1	
	SACL	ONE	
	LACK	M1	* INITIALIZE MASK 1
	TBLR	MASK1	
	LACK	M2	* INITIALIZE MASK 2
	TBLR	MASK2	
	LACK	M3	* INITIALIZE MASK 3
	TBLR	MASK3	
	LACK	MD	* AIB BOARD INITIALIZATION.
	TBLR	TEMP	* MD IS MODE CNTRL FOR AIB.
	OUT	TEMP,PA0	
	LACK	CK	* CK IS SAMPLE RATE FOR AIB.
	TBLR	TEMP	
	OUT	TEMP,PA1	* SENT CLOCK VALUE TO PORT 1(NEW AIB)
	LACK	SINE	* TABLE OFFSET INITIALIZATION.
	SACL	OFSET0	* SINE TABLE OFFSET
	LACK	ENCODE	

	SACL	OFSET1	* <DIBIT TO PHASE> TABLE
	LACK	COEF	
	SACL	XPTR	* RAISED COS COEF. TABLE.
	LACK	XPHASE	
	SACL	INDXPH	* OFFSET FOR XMIT PHASE TABLE
	LACK	RPHASE	
	SACL	RPHSE	* OFFSET FOR RCVR PHASE TABLE
	LACK	4	
	SACL	FOUR	
	ZAC		
	SACL	ROLDPH	* MISC. INITIALIZATIONS.
*			* INITIALIZE PREVIOUS TOTAL PHASE
	SACL	RALPHA	
	SACL	XALPHA	* SWAVE INITIALIZATIONS.
	LACK	DT	
	TBLR	XDELTA	* READ SWAVE DELTAs
	ADD	ONE	
	TBLR	RDELTA	
	LACK	TH1	* CARRIER PLL THRESHOLD
	TBLR	TRSHD1	
	ADD	ONE	
	TBLR	TRSHD2	* BAUD CLOCK PLL THRESHOLD
	LACK	MIN1	
	TBLR	MINUS1	* -1 IN Q12
	LACK	PLS1	
	TBLR	PLUS1	* +1 IN Q12
	LACK	PLLC1	* CARRIER PLL INITIAL COEF. 1
	TBLR	PLL1	
	ADD	ONE	
	TBLR	PLL2	* CARRIER PLL COEF. 2
	ADD	ONE	
	TBLR	BPLL1	* BAUD CLOCK PLL INITIAL COEF. 1
	ADD	ONE	
	TBLR	BPLL2	* BAUD CLOCK COEF. 2
	LACK	AGCTBL	* SET THE AGC TABLE LOOKUP
	SACL	AGCOFF	* OFF SET VALUE
	LAC	ONE,13	* INITIALIZE RUNNING AVERAGE
	SACL	AGCRA	* TO >2000
	LACK	>FF	* INITIALIZE THE AGC FACTOR
	SACL	AGC	* TO ONE
	ZAC		* INITIALIZE THE
	SACL	BSMAX	* BAUD SIGNAL MAX TO ZERO
	LACK	3	* RUNNING AVERAGE COUNT
	SACL	AGCNT	* TO THREE
	LACK	>20	* SET THE ENERGY DETECT
	SACL	RECS	* BIT IN THE STATUS FLAG WORD
	LACK	15	* SET THE REC SAMPLE COUNT
	SACL	SAMPLE	* TO 16
	ZAC		* SET THE XMT SAMPLE COUNT
	SACL	SAMXMT	* TO ZERO

PAGE

THE FOLLOWING CODE HANDLES COMMANDS FROM THE 7042

	LAC	ONE,4	SET COUNTER VALUE TO RUN
	SACL	TEMP	DLB AT 600 BAUD
COMD	BIOZ	LOOK	WAIT FOR 9600HZ SAMPLE PULSE
	B	COMD	
LOOK	NOP		
	IN	RBUF0,PA2	*--- DUMMY READ TO GET COUNTER GOING
LOOK1	IN	XMTD,PA6	LOOK FOR COMMAND
	LACK	>30	MASK OFF ALL BUT COMMAND BITS
	AND	XMTD	CHECK COMMAND BITS FOR NEW COMMAND

```

      BZ  COMD      IF ZERO THEN NO COMMAND YET
      SUB  ONE,4    CHECK FOR DIGITAL LOOP BACK TEST
      BZ  LDLB      IF SO THEN EXECUTE TEST
      SUB  ONE,4    CHECK FOR MODEM RUN COMMAND
      BZ  WAIT      IF SO THEN RUN MODEM
      LACK >F      MASK OFF COMMAND BITS
      AND  XMTD     TO GET SPECIFIC CONFIGURATION
*   THIS IS FOR CONFIGURATION CODES
      BZ  COMD      ZERO IS NOT VALID COMMAND
      SUB  ONE      CHECK FOR COMMAND ONE
      BZ  SETALB    SETUP THE MODEM TO RUN ALB
      SUB  ONE      CHECK FOR COMMAND TWO
      BZ  SETORG    SETUP THE MODEM TO RUN ORIGINATE
      SUB  ONE      CHECK FOR COMMAND THREE
      BZ  SQTREC    SHUT DOWN RECEIVER TO RUN XMIT ONLY
      B  COMD       CHECK FOR NEXT COMMAND
      SETALB LAC ONE,13  LOAD ACC WITH 2000 TO PUT
      SACL XDELTA     XMIT IN SAME BAND AS RECEIVE
      B  COMD       CHECK FOR NEXT COMMAND
      SETORG LAC ONE,12  LOAD ACC WITH 1000 TO PUT
      SACL XDELTA     XMIT IN ORIGINATE MODE
      B  COMD       CHECK FOR NEXT COMMAND
      SQTREC LAC ONE,8   SET RECEIVER SQUELCH BIT
      OR  RECST       IN THE RECEIVE STATUS REG
      SACL RECST      TO DISABLE RECEIVER CODE
      B  COMD       CHECK FOR NEXT COMMAND
*
      LDLB BIOZ DLBOUT  WAIT FOR NEXT SAMPLE PERIOD
      B  LDLB          LOOP ON TIMER
      DLBOUT IN RBUF0,PA2 *--- DUMMY READ TO GET COUNTER GOING
      LAC TEMP         GET 16 SAMPLE BAUD COUNTER
      SUB  ONE          DECREMENT IT
      SACL TEMP         SAVE COUNT
      BNZ  LDLB         COUNT ANOTHER SAMPLE PERIOD
      LAC ONE,4         RESET COUNTER VALUE TO RUN
      SACL TEMP         DLB AT 600 BAUD
      LAC XMTD,10       ADJUST FOR OUTPUT RANGE
      OR  MINUS1        * MASK COMMAND BITS(15-12) TO 1'S
      SACL XMTD         STORE IT FOR OUTPUT
      OUT XMTD,PA6      ECHO INPUT
      B  LOOK1          REPEAT LOOP BACK TEST
*-----*
*****
***** THE FOLLOWING SECTION IMPLEMENTS MODEM FUNCTIONS *****
*****
*-----*
      WAIT  BIOZ  GO      * WAIT FOR 9600HZ SAMPLE PULSE
      B  WAIT
      GO  NOP
      OUT XMTOUT,PA2 * OUTPUT TO D/A
      IN  RBUF0,PA2 * INPUT FROM A/D
*****
***** TRANSMITTER SECTION STARTS HERE. *****
*****
      XMITER EQU $
*-----*
*****
***** SINE(COSINE) WAVE GENERATION *****
*****
      SWAVE EQU $
      LAC XALPHA,8 * DELTA IS THE INCREMENT.
      SACH TEMP * ISOLATE INTEGER PORTION.
      LAC TEMP
      ADD OFFSET0 * ADD INDEX TO SINE TABLE.
      TBLR SINA * SINE VALUE, (Q15).
      LACK >20 * OFFSET TO COSINE VALUE (Q15).
      ADD TEMP
      AND MASK2
      ADD OFFSET0 * ADD INDEX TO COSINE TABLE.

```

TBLR	COSA	* COSINE VALUE, (Q15).
LAC	XALPHA	* COMPUTE ADDRESS OF NEXT
ADD	XDELTA	* POINT FOR TABLE.
AND	MASK1	* KEEP MOD128, MASK=>7FFF.
SACL	XALPHA	* SAVE NEXT ADDRESS

```

*****
***** TRANSMITTER 48 TAP RAISED COSINE FILTER. *****
***** INPUTS UPDATED AT 600HZ RATE. *****
***** OUTPUT UPDATED AT 9600HZ RATE. *****
*****

```

```

RACS EQU $
      LAC XPTR
      TBLR CX0      * RETRIEVE COEFFICIENTS
      ADD ONE
      TBLR CX1
      ADD ONE
      TBLR CX2
      ADD ONE
      SACL XPTR

      ZAC
      LT XIBUF2      * COMPUTE FILTER TAPS ICHAN.
      MPY CX2
      LTA XIBUF1
      MPY CX1
      LTA XIBUF0
      MPY CX0
      APAC
      SACH XIOUT,1

      ZAC
      LT XQBUF2      * COMPUTE FILTER TAPS QCHAN.
      MPY CX2
      LTA XQBUF1
      MPY CX1
      LTA XQBUF0
      MPY CX0
      APAC
      SACH XQOUT,1

      ZAC
      LT XIOUT      * ICHAN*cos(wt)+ QCHAN*sin(wt)
      MPY COSA
      LTA XQOUT
      MPY SINA
      APAC
      SACH XMTOUT,1

```

PAGE

```

*****
***** RECEIVER 1 CHANNEL BANDPASS FILTER. *****
***** SAMPLING RATE IS 9600HZ. *****
*****

```

```

CONT6 ZAC
      LT RBUF36
      MPYK ICF36
      LTA RBUF34
      MPYK ICF34
      LTA RBUF32
      MPYK ICF32
      LTA RBUF30
      MPYK ICF30
      LTA RBUF28
      MPYK ICF28
      LTA RBUF26
      MPYK ICF26
      LTA RBUF24

```

MPYK	ICF24
LTA	RBUF22
MPYK	ICF22
LTA	RBUF20
MPYK	ICF20
LTA	RBUF18
MPYK	ICF18
LTA	RBUF16
MPYK	ICF16
LTA	RBUF14
MPYK	ICF14
LTA	RBUF12
MPYK	ICF12
LTA	RBUF10
MPYK	ICF10
LTA	RBUF8
MPYK	ICF8
LTA	RBUF6
MPYK	ICF6
LTA	RBUF4
MPYK	ICF4
LTA	RBUF2
MPYK	ICF2
LTA	RBUF0
MPYK	ICF0
APAC	
SACH	ISUM,4

* OUTPUT OF I CHAN.

 ***** RECEIVER Q CHANNEL BANDPASS FILTER. *****
 ***** SAMPLING RATE IS 9600HZ. *****

LTD	RBUF35
ZAC	
MPYK	QCF35
DMOV	RBUF34
LTD	RBUF33
MPYK	QCF33
DMOV	RBUF32
LTD	RBUF31
MPYK	QCF31
DMOV	RBUF30
LTD	RBUF29
MPYK	QCF29
DMOV	RBUF28
LTD	RBUF27
MPYK	QCF27
DMOV	RBUF26
LTD	RBUF25
MPYK	QCF25
DMOV	RBUF24
LTD	RBUF23
MPYK	QCF23
DMOV	RBUF22
LTD	RBUF21
MPYK	QCF21
DMOV	RBUF20
LTD	RBUF19
MPYK	QCF19
DMOV	RBUF18
LTD	RBUF17
MPYK	QCF17
DMOV	RBUF16
LTD	RBUF15
MPYK	QCF15
DMOV	RBUF14
LTD	RBUF13
MPYK	QCF13
DMOV	RBUF12


```

LTD      RBUF11
MPYK     QCF11
DMOV     RBUF10
LTD      RBUF9
MPYK     QCF9
DMOV     RBUF8
LTD      RBUF7
MPYK     QCF7
DMOV     RBUF6
LTD      RBUF5
MPYK     QCF5
DMOV     RBUF4
LTD      RBUF3
MPYK     QCF3
DMOV     RBUF2
LTD      RBUF1
MPYK     QCF1
DMOV     RBUF0
APAC
SACH     QSUM,4      * OUTPUT OF Q CHAN.

```

PAGE

```

* .....
* DETECT MAXIMUM SIGNAL STRENGTH OF RECI PER BAUD
AGCAL    EQU      $
LAC      ISUM      * AGC VALUE CALCULATED USING ISUM
ABS      * GET MAGNETUDE OF SIGNAL
SUB      BSMAX     * COMPARE TO PREVIOUS MAX VALUE
BLZ      OVRMAX    * IF LESS THAN THEN JUMP OVER UPDATE
ADD      BSMAX     * RESTORE VALUE AND
SACL     BSMAX     * STORE AS NEW MAX
* MULTIPLY IN AGC FACTOR TO FILTERED SIGNAL
OVRMAX   LT       AGC      * MULTIPLY THE AGC FACTOR
MPYI     MPY      ISUM     * BY THE FILTERED DATA ELEMENT
PAC      * MOVE THE PRODUCT TO THE ACC
SACH     TEMP,4    * SAVE TOP HALF OF ACC
AND      PLUS1     * MASK OFF UNUSABLE BITS
SACL     ISUM      * SAVE BOTTOM HALF OF ACC
ZALH     TEMP      * RELOAD HIGH ACC VALUE
ADD      ISUM,4    * SHIFT LOW HALF INTO POSITION
SACH     ISUM,4    * STORE Q15 GAINED FILTERED DATA
MPYQ     LT       AGC      * MULTIPLY THE AGC FACTOR
MPY      QSUM      * BY THE FILTERED DATA ELEMENT
PAC      * MOVE THE PRODUCT TO THE ACC
SACH     TEMP,4    * SAVE TOP HALF OF ACC
AND      PLUS1     * MASK OFF UNUSABLE BITS
SACL     QSUM      * SAVE BOTTOM HALF OF ACC
ZALH     TEMP      * RELOAD HIGH ACC VALUE
ADD      QSUM,4    * SHIFT LOW HALF INTO POSITION
SACH     QSUM,4    * STORE Q15 GAINED FILTERED DATA
* .....

```

PAGE

```

* .....
* The following code is the time sliced code task master.
* The routine monitors the status of the modem operations
* and sequences the code appropriately.
* .....
MASTER   EQU      $
LAC      ONE,5      * CHECK OPERATING STATUS FOR
AND      RECST      * ENERGY DETECT
BZ       HANGUP     * IF NO ENERGY DETECT THEN HANG UP
LAC      ONE,4      * CHECK IF LOCAL CARRIER
AND      RECST      * IS LOCKED. IF SO SWITCH
BNZ      CARLCK     * PLL FILTERS BANDWIDTH
B        NORMAL     * EXECUTE NORMAL SEQUENCE
* .....
CARLCK   LACK PLLC   * CHANGE CARRIER PLL COEF. 1
          TBLR PLL1
          LACK BPLLC  * CHANGE BAUD CLOCK PLL COEF. 1

```

```

NORMAL    TBLR BPLL1
          EQU $
          LAC SAMPLE      * DECREMENT THE SAMPLE COUNT
          SUB ONE          * TO CHECK FOR END OF BAUD
          BGEZ OVRSAM      * IF NOT THEN SKIP COUNT RESET
          LACK 15          * RESTART THE SAMPLE COUNTER AT 15
OVRSAM    SACL SAMPLE      * SAVE NEW COUNT VALUE
          LACK TSKSEQ      * GET ADDRESS OF TOP OF TABLE
          ADD SAMPLE      * ADD IN OFFSET
          TBLR TEMP        * GET THE PROGRAM ADDRESS
          LAC TEMP         * FOR THE TASK CALL
          CALA             * EXECUTE THE APPROPRIATE TASK

```

```

*
*   UPDATE CARRIER ANGLE AT SAMPLE RATE
*

```

```

          LAC RALPHA      * COMPUTE ADDRESS OF NEXT
          ADD RDELTA      * POINT FOR TABLE.
          AND MASK1       * KEEP MOD128, MASK=>7FFF.
          SACL RALPHA     * SAVE NEXT ADDRESS

```

```

MASXMT    EQU $          * EXECUTE TRANSMIT TASK SEQUENCE
          LAC SAMXMT      * DECREMENT THE SAMPLE COUNT
          SUB ONE          * TO CHECK FOR END OF BAUD
          BGEZ OVRSM1     * IF NOT THEN SKIP COUNT RESET
          LACK 15          * RESTART THE SAMPLE COUNTER AT 15
OVRSM1    SACL SAMXMT     * SAVE NEW COUNT VALUE
          LACK TSKXMT     * GET ADDRESS OF TOP OF TABLE
          ADD SAMXMT      * ADD IN OFFSET
          TBLR TEMP        * GET THE PROGRAM ADDRESS
          LAC TEMP         * FOR THE TASK CALL
          CALA             * EXECUTE THE APPROPRIATE TASK
          B WAIT          * WAIT FOR NEXT SAMPLE TIMEOUT

```

PAGE

```

*****
* This is the software automatic gain control factor update. *
* The routine keeps a running average plus three baud max's *
* to generate each new AGC update. Once the value is gained *
* the routine uses a table lookup device to force the filter *
* data max's into a tight range.
*****

```

```

AGCUPT    ZALH AGCRA      ADD THE NEW BSMAX VALUE
          ADD BSMAX,14    TO THE RUNNING AVERAGE
          SACH AGCRA      AND SAVE IT
          LAC AGCNT       DECREMENT RUNNING AVERAGE COUNT
          SUB ONE         SAVE IT AND
          SACL AGCNT      CHECK FOR ZERO
          SACH BSMAX      ZERO OUT RUNNING SIGNAL MAX
          BZ OVRROUT      IF ZERO THEN UPDATE AGC
          RET             ELSE RETURN TO CALLING SEQUENCE
OVRROUT    LACK 3         RESET RUNNING AVERAGE COUNT
          SACL AGCNT      TO THREE
          LAC AGCRA       MOVE AGCRA
          SACL AGCLEV     TO THE CALCULATION LEVEL
          LAC AGCRA,14    DIVIDE RUNNING AVERAGE SUM
          SACH AGCRA      BY 4 TO GET NEW RUNNING AVERAGE
          LAC AGCLEV      GET AVERAGE MAX SIGNAL LEVEL
          SUB ONE,14      COMPARE TO 16384
          BLZ ASHF1       IF LESS THAN SHIFT TABLE LOOKUP
          LAC AGCLEV,7    GET LOOKUP VALUE
          SACH TEMP       MOVE LOOKUP VALUE TO
          LAC TEMP        THE LOW HALF OF THE ACC
          ADD AGCOFF      ADD IN TABLE OFFSET
          TBLR AGC        AND GET AGC VALUE
          LAC AGC,15      DIVIDE THE AGC VALUE
          SACH AGC        BY 2 TO FORCE TO Q14 MODE
          RET             RETURN TO CALLING SEQUENCE
ASHF1     ADD ONE,13      COMPARE TO 8192
          BLZ ASHF2       IF LESS THAN SHIFT TABLE LOOKUP

```

	LAC AGCLEV,8	GET LOOKUP VALUE
	SACH TEMP	MOVE LOOKUP VALUE TO
	LAC TEMP	THE LOW HALF OF THE ACC
	ADD AGCOFF	ADD IN TABLE OFFSET
	TBLR AGC	AND GET AGC VALUE
	RET	RETURN TO CALLING SEQUENCE
ASHF2	ADD ONE,12	COMPARE TO 4096
	BLZ ASHF3	IF LESS THAN SHIFT TABLE LOOKUP
	LAC AGCLEV,9	GET LOOKUP VALUE
	SACH TEMP	MOVE LOOKUP VALUE TO
	LAC TEMP	THE LOW HALF OF THE ACC
	ADD AGCOFF	ADD IN TABLE OFFSET
	TBLR AGC	AND GET AGC VALUE
	LAC AGC,1	AGC VALUE * 2 TO ADJUST
	SACL AGC	FOR LOWER SIGNAL STRENGTH
	RET	RETURN TO CALLING SEQUENCE
ASHF3	ADD ONE,11	COMPARE TO 2048
	BLZ ASHF4	IF LESS THAN SHIFT TABLE LOOKUP
	LAC AGCLEV,10	GET LOOKUP VALUE
	SACH TEMP	MOVE LOOKUP VALUE TO
	LAC TEMP	THE LOW HALF OF THE ACC
	ADD AGCOFF	ADD IN TABLE OFFSET
	TBLR AGC	AND GET AGC VALUE
	LAC AGC,2	AGC VALUE * 4 TO ADJUST
	SACL AGC	FOR LOWER SIGNAL STRENGTH
	RET	RETURN TO CALLING SEQUENCE
ASHF4	ADD ONE,10	COMPARE TO 1024
	BLZ ASHF5	IF LESS THAN SHIFT TABLE LOOKUP
	LAC AGCLEV,11	GET LOOKUP VALUE
	SACH TEMP	MOVE LOOKUP VALUE TO
	LAC TEMP	THE LOW HALF OF THE ACC
	ADD AGCOFF	ADD IN TABLE OFFSET
	TBLR AGC	AND GET AGC VALUE
	LAC AGC,3	AGC VALUE * 8 TO ADJUST
	SACL AGC	FOR LOWER SIGNAL STRENGTH
	RET	RETURN TO CALLING SEQUENCE
ASHF5	ADD ONE,9	COMPARE TO 512
	BLZ ASHF5	IF LESS THAN SHIFT TABLE LOOKUP
	LAC AGCLEV,12	GET LOOKUP VALUE
	SACH TEMP	MOVE LOOKUP VALUE TO
	LAC TEMP	THE LOW HALF OF THE ACC
	ADD AGCOFF	ADD IN TABLE OFFSET
	TBLR AGC	AND GET AGC VALUE
	LAC AGC,4	AGC VALUE * 16 TO ADJUST
	SACL AGC	FOR LOWER SIGNAL STRENGTH
	RET	RETURN TO CALLING SEQUENCE
ASHF6	ADD ONE,5	COMPARE TO 32
	BLZ NOEDT	LOST MINIMUM ENERGY LEVEL
	LAC AGCLEV,13	GET LOOKUP VALUE
	SACH TEMP	MOVE LOOKUP VALUE TO
	LAC TEMP	THE LOW HALF OF THE ACC
	ADD AGCOFF	ADD IN TABLE OFFSET
	TBLR AGC	AND GET AGC VALUE
	LAC AGC,5	AGC VALUE * 32 TO ADJUST
	SACL AGC	FOR LOWER SIGNAL STRENGTH
	RET	RETURN TO CALLING SEQUENCE
NOEDT	LACK >DF	PASSBAND SIGNAL TOOL LOW
	AND RECST	DISABLE SIGNAL ENERGY DETECT
	SACL RECST	AND CARRIER DETECT SIGNAL
	RET	RETURN TO CALLING SEQUENCE
.		
	PAGE	
HANGUP	B WAIT	
DUMXMT	EQU \$	
	RET	
SMARK	EQU \$	
	RET	
.		

```

.....
*
GETDBT EQU $
IN XMTD,PA6 * GET NEW DIBIT
LACK >30
AND XMTD * CHECK COMMAND BITS
BZ COMD * IF ZERO SQT MODEM, IDLE
*
LACK COEF * RECYCLE IF FINISHED
SACL XPTR
DMOV XIBUF1 * SHIFT UP THE FILTER
DMOV XIBUF0 * TO MAKE ROOM FOR
DMOV XQBUF1 * FOR THE NEW DATA VALUE
DMOV XQBUF0 * JUST INPUT
*
LACK 3
AND XMTD * NEW DIBIT FROM 7000
ADD OFSET1 * LOOKUP NEWPHASE
TBLR XNEWPH
LAC XOLDPH * GET OLDPHASE.
ADD XNEWPH * ADD NEW PHASE.
AND MASK3 * MASK WITH >0006.
SACL XOLDPH * STORE BACK 'NEW' OLDPHASE.
ADD INDXPH * LOOKUP I & Q INPUTS.
TBLR XIBUF0
ADD ONE
TBLR XQBUF0
RET
*
*
DUMMY CALL DEMOD ATTEMPT DEMODULATION
RET RETURN TO TASK MASTER
*
*
DEMODB EQU $ MIDDLE OF THE BAUD
LACK >FE RESET THE CURRENT BAUD CLOCK
AND RECST CORRECTION FLAG IN THE STATUS
SACL RECST REGISTER AND SAVE IT
.....
***** DEMODULATE THE PASSBAND SIGNAL. *****
***** RCVR. CARRIER SINE(COSINE) WAVE GENERATOR *****
*****
DEMOD EQU $
LAC RALPHA,8 * DELTA IS THE INCREMENT.
SACH TEMP * ISOLATE INTEGER PORTION.
LAC TEMP
ADD OFSET0 * ADD INDEX TO SINE TABLE.
TBLR SINA * SINE VALUE, (Q15).
LACK >20
ADD TEMP
AND MASK2
ADD OFSET0 * ADD INDEX TO COSINE TABLE.
TBLR COSA * COSINE VALUE, (Q15).
*
*
*****
CONT1 LT ISUM * DEMOD. I CHANNEL
MPY COSA * A=(YI * cosA)/2
PAC
LT QSUM
MPY SINA
APAC * A=(YI * cosA)/2 + (Yq * sin A)/2
SACH RECI,1 * RECI= (YI * cosA) + (Yq * sinA)
*
LT ISUM * DEMOD. Q CHANNEL
MPY SINA * A = (Yi * sinA)/2
PAC
LT QSUM
MPY COSA

```

```

      SPAC      * A = [(Yi * s
      SACH RECQ,I * RECQ = (Yi
*
*----MUST DETERMINE ENERGY FOR BAUD CLOC
*
      LT      RECI
      MPY      RECI      * FIND I**2
      PAC
      LT      RECQ
      MPY      RECQ      * FIND Q**2
      APAC
      SACH ENRGY      * ENERGY = (I**
*
*----MUST DETERMINE SIGN OF I AND Q FOR
*
      LAC      RECI      * DETERMINE I
      BGZ      DM1
      LAC      MINUS1
      B        DM2
DM1      LAC      PLUS1
DM2      SACL     SIGNI      * SAVE SIGN I
*
      LAC      RECQ      * DETERMINE Q
      BGZ      DM3
      LAC      MINUS1
      B        DM4
DM3      LAC      PLUS1
DM4      SACL     SIGNQ
      RET              RETURN TO CALL
*****
* INOUT GET DIBIT FROM 7000 AND XMIT N
* TO THE 7000
*****
OUT      EQU      $
      LAC      RDIBIT,10
      OR      MINUS1      * MASK D15-D1
      SACL     BITOUT      *AND SAVE THE
      OUT      BITOUT,PA6 *XMIT TO 7000
      RET              * BACK TO CALL
*****
***** PHASE DECODING - BINARY TO GR
* THIS ROUTINE CALCULATES PHASE SHIFT
* CURRENT ABSOLUTE PHASE, GREY CODE RE
*****
DECODE   LAC      RECI      * DETERMINE ABS
      BGZ      ABS1
      LAC      RECQ
      BGZ      ABS2
      LACK     2          * PHASE IS 2 (0
      B        DIFFER
ABS2     LACK     3          * PHASE IS 3 (2
      B        DIFFER
ABS1     LAC      RECQ
      BGZ      ABS3
      LACK     1          * PHASE IS 1 (1
      B        DIFFER
ABS3     LACK     0          * PHASE IS 0 (0
*
DIFFER   SACL     TEMP
      SUB      ROLDPH      * SUBTRACT PREVIOUS
      BGEZ     DF1          * LATE PHASE (E
      ADD      FOUR
DF1      ADD      RPHSE      * MAP PHASE CHANGE
      TBLR     RDIBIT
      LAC      TEMP
      SACL     ROLDPH
*****
***** COMPUTE CARRIER ERROR SIGNAL.
***** e(t) = RECI*SIGNQ - RECQ*SIGNI
*****

```

```

COMERR ZAC
      LT      RECI
      MPY     SIGNQ
      LTA     RECQ
      MPY     SIGNI
      SPAC
      SACH    ERROR,1      * ERROR IS IN Q12
*****
***** LOOP FILTER *****
*****
      ZAC
      LT      PLL2
      MPY     ERROR
      LTA     PLL1
      MPY     ERRSIG
      APAC
      SACH    ERRSIG,1     * ERRSIG IS IN Q12
*****
* CORRECT PHASE ERROR ONLY AT MIDDLE OF BAUD
*****
* Adjust carrier phase +/-
* one table entry if      - (2*trshld) > error > trshld
* two table entries if    - (2*trshld) < error >> trshld
* RALPHA is current local carrier table index.(in MSB )
*****
CKEROR LAC  ERRSIG
      BGZ  ERR1
      ADD  TRSHD1
      BGZ  ERRETN
      ADD  TRSHD1
      BGZ  SUB1A
      LAC  RALPHA
      SUB  ONE,9
      B    ERR2
SUB1A   LAC  RALPHA
      SUB  ONE,8
      B    ERR2
ERR1    SUB  TRSHD1
      BLZ  ERRETN
      SUB  TRSHD1
      BLZ  ADD1A
      LAC  RALPHA
      ADD  ONE,9
      B    ERR2
ADD1A   LAC  RALPHA
      ADD  ONE,8
ERR2     AND  MASK1
      SACL  RALPHA
      RET
ERRETN   LAC  ONE,4
      OR    RECST
      SACL  RECST
      RET
*****
***** BAUD CLOCK ALIGNMENT *****
*****
BDCLK1  CALL  DEMOD
      LAC  ENRGY
      SACL  PENRGY
      RET
      * ENRGY = E(3)
      * STORE IT IN PENRGY
BDCLK2  CALL  DEMOD
      LAC  RECST
      AND  ONE
      BNZ  RETB
      LAC  ENRGY
      SUB  PENRGY
      SACL  BERROR
      * TEST IF CORRECTION OF THE
      * BAUD CLOCK IS MADE
      * IF SO THEN RETURN
      * ENRGY = E(11), PENRGY = E(3)
      * FORM ERROR SIGNAL
      * BERROR = E(11)-E(3)
*****
***** LOOP FILTER *****

```

```

*****
      ZAC
      LT   BPLL2
      MPY  BERROR
      LTA  BPLL1
      MPY  BEROUT
      APAC
      SACH BEROUT,1      * BEROUT IN Q14
*
*---APPLY CORRECTION
*
      LAC  BEROUT
      BGEZ POS           * TEST BERROUT SIGN.
      ADD  TRSHD2
      BGEZ RETB          * IF ;BERROUT;<TRSHD RETURN.
      ADD  TRSHD2        * BERROUT IS NEGATIVE. THEREFORE
      BGEZ SUB1B         * ADJUST CLOCK BY DELAYING SAMPLE COUNT.
      LAC  SAMPLE        * IF ;BERROUT;>2*TRSHD
      SUB  ONE,1         * MAKE TWO SAMPLE ADJUSTMENT
      SACL SAMPLE        * OF THE SAMPLE (BAUD CLOCK)
      B    RETB          * COUNT.
SUB1B  LAC  SAMPLE      * IF TRSHD<;BERROUT;<2*TRSHD
      SUB  ONE          * MAKE ONE SAMPLE ADJUSTMENT
      SACL SAMPLE      * OF THE SAMPLE (BAUD CLOCK)
      B    RETB        * COUNT.
POS    SUB  TRSHD2     * BERROUT IS POSITIVE. THEREFORE
      BLZ  RETB        * ADJUST CLOCK BY ADVANCING SAMPLE
      SUB  TRSHD2     * COUNT.
      BLZ  ADD1B      * IF ;BERROUT;>2*TRSHD
      LAC  SAMPLE    * MAKE TWO SAMPLE ADJUSTMENT
      ADD  ONE,1     * OF THE SAMPLE (BAUD CLOCK)
      SACL SAMPLE    * COUNT.
      B    RETB
ADD1B  LAC  SAMPLE    * IF TRSHD<;BERROUT;<2*TRSHD
      ADD  ONE        * MAKE ONE SAMPLE ADJUSTMENT
      SACL SAMPLE    * OF THE SAMPLE (BAUD CLOCK) COUNT.
RETB   LAC  RECST     * SET FLAG TO INDICATE THAT THE BAUD
      OR   ONE        * CLOCK ADJUSTMENT IS MADE.
      SACL RECST
      RET
*****
      END

```

Appendix E
TMS7742 Source Code


```

TITL      TMS 7742 MODEM INTERFACE PROGRAM'
OPTION    XREF,TUNLST
7042 PORT ASSIGNMENTS

```

```

* APORT

```

```

      A7  A6  A5  A4  A3  A2  A1  A0
OHR_ N.C. RCVD ATE_ A_/O SQT DTR DCD
(O)  (X)  (1)  (0)  (0)  (0)  (1)  (1)

```

```

* BPORT

```

```

      B7  B6  B5  B4  B3  B2  B1  B0
NB8  NB4  NB2  NB1  TXD  DP  DSR  CTS
(O)  (0)  (0)  (0)  (0)  (0)  (0)  (0)

```

```

* CPORT

```

```

      C7  C6  C5  C4  C3  C2  C1  C0
ACKW ACKR CMD2 CMD1 TDB3 TDB2 TDB1 TDB0
(O)  (0)  (0)  (0)  (0)  (0)  (0)  (0)

```

```

* DPORT

```

```

      D7  D6  D5  D4  D3  D2  D1  D0
NEW0 NEW1 CDT  ENB  RDB3 RDB2 RDB1 RDB0
(1)  (1)  (1)  (1)  (1)  (1)  (1)  (1)

```

```

-----
* SWSTAT  +-----+
          | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
          +-----+

```

```

* BIT7: modem type      0= B103 mode
                        1= B212 mode
* BIT6: timer flag      0= carrier wait timer enabled
                        1= 1200 Hz timer enabled
* BIT5: 1st dibit flag  0= flag reset
                        1= flag set
-----

```

```

      IDT 'DSPMODM'
OPTION  XREF
AORG    >F006

```

```

* 7041 Peripheral Memory Symbols

```

```

IOCNT0 EQU P0
TIDATA EQU P2
TICNTL EQU P3
APORT  EQU P4
ADDR   EQU P5
BPORT  EQU P6
CPORT  EQU P8
CDDR   EQU P9
DPORT  EQU P10
DDDR   EQU P11
IOCNT1 EQU P16
SMODE  EQU P17
SCTL0  EQU P17
SSTAT  EQU P17
T2DATA EQU P18
T2CNTL EQU P19
T3DATA EQU P20
SCTL1  EQU P21
RXBUF  EQU P22
TXBUF  EQU P23
MPRTC  EQU >108
MPRTD  EQU >10A

```

```

* Bit Masks.

```

```

BIT0 EQU >01
BIT1 EQU >02
BIT2 EQU >04
BIT3 EQU >08
BIT4 EQU >10
BIT5 EQU >20
BIT6 EQU >40
BIT7 EQU >80
*
NOT0 EQU >FE
NOT1 EQU >FD
NOT2 EQU >FB
NOT3 EQU >F7
NOT4 EQU >EF
NOT5 EQU >DF
NOT6 EQU >BF
NOT7 EQU >7F
*
* Ascii constants
*
TAB EQU >09 ; tab character
BLANK EQU >20 ; space character
COMMA EQU >2C ; ','
LF EQU 10
CR EQU 13
BS EQU 8 ; BACKSPACE CHARACTER
POUND EQU >23 ; '#'
STAR EQU >2A ; '*'
*
ISA EQU >41 ; 'A'
ISZ EQU >5A ; 'Z'
*
* 7041 RAM map
*
RDIBIT EQU R2
DIBIT2 EQU RDIBIT+1 ; receiver input sequence
DIBIT1 EQU DIBIT2+1 ; from the 32010
TEMP1 EQU DIBIT1+1 ; temporary register (receiver)
TEMP2 EQU TEMP1+1 ; temporary register (xmitter)
RBIT14 EQU TEMP2+1
RBIT17 EQU RBIT14+1
DESREG EQU RBIT17+1
FLAG EQU DESREG+1
COUNT EQU FLAG+1
BITCNT EQU COUNT+1
CHRCNT EQU BITCNT+1
TRNMIT EQU CHRCNT+1 ; character to be transmitted
STPFLG EQU TRNMIT+1 ; stop bit deleted flag
MCOUNT EQU STPFLG+1 ; mark counter
XDBIT2 EQU MCOUNT+1
XDBIT1 EQU XDBIT2+1
XBIT17 EQU XDBIT1+1
XBIT14 EQU XBIT17+1
SCMREG EQU XBIT14+1
XDIBIT EQU SCMREG+1 ; xmitter input dibit
CONTER EQU XDIBIT+1 ; counter register
RBTCNT EQU CONTER+1 ; character bit counter (xmt)
XMTCHR EQU RBTCNT+1 ; input character buffer
ADDRES EQU XMTCHR+2 ; command buffer address pointer
PNTR EQU ADDRES+1 ; counter register
SWSTAT EQU PNTR+1 ; software status flag
LOCH1 EQU SWSTAT+1
LOCLO EQU LOCH1+1
ADDR1 EQU LOCLO
INDEX1 EQU ADDR1+1 ; general purpose register
INDEX2 EQU INDEX1+1 ; general purpose register
COUNT1 EQU INDEX2+1
COUNT EQU COUNT1+1 ; general use double register counter

```

```

COMBUF EQU COUNT+1          ; beginning of the command buffer
S0      EQU COMBUF+40        ; carriage return register
S1      EQU S0+1            ; line feed register
S2      EQU S1+1            ; backspace register
S3      EQU S2+1            ; # of rings to answer on
S4      EQU S3+1            ; # of rings detected
S5      EQU S4+1            ; escape code character
INT5TM  EQU S5+1            ; interrupt 5 timer register
VALUE   EQU INT5TM+1        ; contains numerical value of parameters
MSGM    EQU VALUE+1
MSGL    EQU MSGM+1          ; message address register
CWT1    EQU MSGL+1
CWT2    EQU CWT1+1          ; carrier wait abort timer
MSTIME  EQU CWT2+2          ; millisec timing register
DELYR1  EQU MSTIME+1
STACK   EQU R100

```

```

*
ALL      EQU >FF
ZERO     EQU >00
ONE      EQU >01
TWO      EQU >02
THREE    EQU >03
EIGHT    EQU >08
NINE     EQU >09
TEN      EQU >0A
CNTVAL   EQU >DC
ADDTOP   EQU S0-1
ADDBOT   EQU COMBUF

```

```

*****
***** Initialization *****
*****

```

```

INIT     MOVP    %>0F,IOCNT0    ; s.c. mode, enable int2 and int1
          MOVP    %>00,IOCNT1    ; disable int4 and int5
          MOVP    %>0C,APORT
          MOVP    %>9C,ADDR      ; set direction of APORT
          MOVP    %>FB,BPORT
          MOVP    %>CF,CPORT
          MOVP    %>FF,CDDR      ; set direction of CPORT
          MOVP    %>CF,DPORT
          MOVP    %>00,DDDR      ; set direction of DPORT
          MOVP    %155,T2DATA
          MOVP    %>81,T2CNTL
          ORP     %BIT2,BPORT    ; reset the 99531 dialer
          ANDP    %NOT2,BPORT
          MOV     %STACK,B
          LDSP
          MOV     %ALL,DIBIT1
          MOV     %ALL,DIBIT2
          MOV     %ONE,RBIT17
          MOV     %ONE,RBIT14
          MOV     %ALL,XDBIT1
          MOV     %ALL,XDBIT2
          MOV     %ONE,XBIT17
          MOV     %ONE,XBIT14
*
          MOV     %CR,S0        ; carriage return character
          MOV     %LF,S1        ; line feed character
          MOV     %BS,S2        ; backspace character
          MOV     %ONE,S3       ; # of rings to answer on
          CLR     S4            ; # of rings detected
          MOV     %'+',S5       ; escape code character
          MOV     %>C0,SWSTAT    ; software flag default conditions

```

```

* main routine

```

```

          ANDP    %NOT0,BPORT    ; set CTS_
          EINT

```

```

TOP    CALL    @AUTOBD      ; Autobaud to terminal speed
      MOVD    %HELLO,MSGL   ; Send hello message
      CALL    @PRINT
      .
      * look for input commands.
      .
LOOK   CALL    @CLEAR        ; clear the command buffer
      MOVD    %ADDTOP,ADDRES ; point to top of the buffer
      CLR     PNTR           ; clear buffer command pointer
LK4COM BTJZP    %BIT1,SSTAT,LK4COM ; command received?
      MOVP    RXBUF,A
      MOVP    A,TXBUF        ; echo
WAIT4  BTJZP    %BIT2,SSTAT,WAIT4
      CMP     %CR,A          ; last character?
      JEQ     EXEC           ; yes, go execute command
      .
      CMP     %'(',A         ; ignore
      JEQ     LK4COM
      CMP     %')',A         ; ignore
      JEQ     LK4COM
      CMP     %'-',A         ; ignore
      JEQ     LK4COM
      CMP     %' ',A         ; ignore
      JEQ     LK4COM
      CMP     %'/',A         ; ignore
      JEQ     LK4COM
      CMP     %BS,A          ; backspace?
      JNE     NXTSTG         ; yes, go get new command
      DEC     PNTR           ; decrement pointer
      CLR     A              ; CLEAR OUT THE BUFFER
      STA     *ADDRES         ; AT THE CURRENT LOCATION
      INC     ADDRES          ; point to the previous location
      JMP     LK4COM
      .
NXTSTG INC     PNTR          ; command buffer pointer
      STA     *ADDRES         ; location for command
      DECD    ADDRES          ; location for next command
      CMP     %40,PNTR        ; allow 40 chars maximum
      JEQ     ERR             ; more than 40..clear buffer
      JMP     LK4COM          ; keep going till <CR>
      .
ERR    CALL    @CLEAR        ; clear command buffer
      MOVD    %ERROR,MSGL     ; send error message
      CALL    @PRINT
      MOV     %STACK,B        ; reset the stack pointer
      LDSP    BR
      BR      @LOOK
      .
EXEC   MOVD    %ADDTOP,ADDRES ; initialize address point
      LDA     *ADDRES         ; get command
      CMP     %'A',A          ;
      JL      ERR             ; Check for A thru Z
      CMP     %'Z'+1,A        ;
      JHS     ERR             ;
      CLR     B               ; Parameter buffer pointer
      DECD    ADDRES
      SUB     %'A',A
      MOV     A,B
      RL      B               ; B*2
      LDA     @COMLIS(B)      ;
      MOV     A,LOCHI         ; MSB address
      INC     B
      LDA     @COMLIS(B)      ;
      MOV     A,LOCLO         ; LSB address
      BR      *ADDR1          ; execute command
      .

```

PAGE

```

*****-----*****
*****      Local Digital Loopback Test      *****
*****-----*****
*
LDLB  EQU  $
      MOVD  %LDLB,MSGL          ; RESPOND TO COMMAND TO DTE
      CALL  @PRINT              ; BY PRINTING TEST CODE
      MOV   %>10,R23            ; SET COMMAND TO LDLB MODE
      BR    @GO320              ; AND RUN THE 320
*
      PAGE
*****-----*****
*****      Dial Blind!      *****
*****-----*****
*
DB    OR     %BIT6,SWSTAT
      MOVP   %>2A,IOCNT0        ; disable RI interrupt
      ORP    %BIT0,BPORT        ; turn off CTS_
      ORP    %>8C,APORT        ; originate mode, squelch 532,
*                               ; and go off hook
      CALL   @DIAL              ; dial
      MOV    %18,CWT1           ; initialize carrier abort timer.
      CLR    CWT2
      ORP    %BIT2,IOCNT0       ; enable carrier abort interrupt
CHKDCD BTJOP  %BIT0,APORT,CHKDCD ; wait for DCD_
      AND    %NOT6,SWSTAT
      BTJZ   %BIT7,SWSTAT,B103  ; check for modem type
      ORP    %BIT3,CPORT        ; 32010 in B212 originate mode
      ORP    %BIT2,CPORT
      BR     @B212
*****-----*****
*****      Bell 103 Call Initiation.      *****
*****-----*****
B103  BTJOP  %BIT0,APORT,B103    ; Wait for DCD_
*
* Send originate tone.
*
      ORP    %BIT4,APORT        ; ATE = 1
      ANDP   %NOT2,APORT        ; unsquelch 532.
*
      MOVP   %>4A,IOCNT0        ; got DCD_, disable abort interrupt
*
* Wait 800ms
*
      MOVD   %800,MSTIME
      CALL   @MSDLY
      ANDP   %NOT0,BPORT        ; activate CTS_
      MOVD   %CONN3,MSGL        ; send connect 300 message
      CALL   @PRINT
      BR     @DAT103            ; enter data mode
*****-----*****
*****      DIAL - Dial number stored in ADDRES.      *****
*****-----*****
DIAL  ANDP   %NOT4,APORT        ; ATE_ = 0, enable EX1 mode
*
* Execute dialing.
*
      MOVD   %4000,MSTIME        ; Initial dial tone wait of 2 second
      CALL   @MSDLY
NEXTDIG LDA  *ADDRES            ; Load subcommand
      CMP    %ZERO,A            ; is it the last command?
      JNE    NOTEND
*
* End of dialing.
*
      RETS
*

```

```

* Case statement to determine subcommand.
*
NOTEND DECD  ADDRES          ; update address
      CMP    %'0',A          ; check less than '0'
      JL     NOTNUM
      CMP    %'9'+1,A        ; check greater than '9'
      JHS    NOTSPC
      BR     @ISANUM
NOTNUM  CMP    %',',A          ; ', ' - dial tone wait
      JEQ    DPAUSE
      CMP    %STAR,A         ; '*' - tone dial *
      JEQ    ISSTAR
      CMP    %POUND,A        ; '#' - tone dial #
      JEQ    APOUND
NOTSPC  BR     @NXTDIG
*
* Wait for a dial tone.
*
DPAUSE  MOV    %TWO,VALUE     ; Blind delay
      CALL   @SECDLY
      BR     @NXTDIG
*
* Dial a digit.
*
ISSTAR  MOV    %TEN,A         ; dial * if tone dial
      JMP    OUTDIG
APOUND  MOV    %11,A          ; dial # if tone dial
      JMP    OUTDIG
ISANUM  SUB    %'0',A         ; dial a number
OUTDIG  ANDP   %>0F,BPORT     ; clear old digit
      RL     A               ; get the correct value
      RL     A
      RL     A
      RL     A
      ORP    A,BPORT         ; send new digit
PNDWTO  BTJZP  %BIT1,APORT,PNDWTO ; wait for acceptance
      ORP    %BIT2,BPORT     ; set DP
PNDWT1  BTJOP  %BIT1,APORT,PNDWT1 ; wait for PND low
      ANDP   %NOT2,BPORT     ; clear DP
PNDWT2  BTJZP  %BIT1,APORT,PNDWT2 ; wait for PND high
      BR     @NXTDIG
*
      PAGE
*****-----*****
*****  BELL 1200 BPS MODEM ALGORITHM  *****
*****-----*****
*
B212    EQU    $
      MOV    %>20,R23        ; SET COMMAND TO MODEM RUN
GO320   CLR    R2            ; CLEAR COM STATUS REG
      CLR    R11             ; INITIALIZE SCRAMBLER HISTORY
      CLR    R12             ; AS ALL ZEROS
      CLR    R18             ; INITIALIZE DESCRAMBLER HISTORY
      CLR    R19             ; AS ALL ZEROS
      CLR    R13             ; INITIALIZE DESCRAMBLER HISTORY
      CLR    R20             ; AS ALL ZEROS
      ANDP   %NOT0,BPORT     ; ACTIVATE CTS TO DTE
      CLR    A               ; CYCLE THE CLEAR LINES
      STA    @MPRTC          ; OF THE I/O CONTROL
      ORP    %>C0,CPORT       ; RESET 320 ACK LINES
*
* START UP MODEM OR DLB TEST
*
STOPB2  MOV    %3,R10         ; SET DIBIT TO MARKS
      CALL   @SCRAM          ; AND SCRAMBLE IT
      MOV    R10,A           ; HOLD IT FOR TRANSMIT
      OR     R23,A           ; OR IN COMMAND BITS

```

```

        ANDP    %>C0,CPORT        ; CLEAR OFF CURRENT BITS
        ORP     A,CPORT            ; SEND OUT SCRAM MARKS
*   TRANSMIT UNSCRAMBLD MARKS AND RECEIVE
MRC1    BTJZP   %BIT7,DPORT,MRC2   ; WAIT FOR WRITE FROM 320
CHKTCH  BTJOP   %BIT6,DPORT,RECDTE ; WAIT FOR READ FROM 320
        BR      @MRC3              ; PROCESS READ FROM 320
RECDTE  BTJOP   %BIT1,SSTAT,DTEGET ; IS DTE REC BUF FULL
XMTDTE  BTJOP   %BIT0,SSTAT,DTEPUT ; IS DTE TRANS BUF EMPTY
        JMP     MRC1              ; LOOK AGAIN
*
*   CODE INTERFACE TO DTE
*
DTEGET  EQU     $
        MOVP    RXBUF,A           ; YES, GET THE CHARACTER?
        CMP     %>1B,A            ; IF A <> ESCAPE
        JMP     OVRSQT            ; THEN CONTINUE
        CLR     A                 ; ELSE SQUELCH THE
        STA     @MPRTC            ; THE 320 MODEM AND
        BR      @TOP              ; AND RETURN TO MONITOR
OVRSQT  INC     R24               ; INCREMENT BYTE COUNT
        BTJO    %BIT5,R2,DTEGER    ; CHECK FOR BUF2 FULL
        BTJO    %BIT3,R2,DTEGI     ; CHECK IF 1ST CHAR
        OR      %BIT7,R2          ; FLAG FOR START BIT
        MOV     A,R7              ; IF SO THEN RESTART
        MOV     %>A,R21           ; RESET XMT COUNT
        OR      %BIT3,R2          ; SET TRANS ACTIVE
        JMP     XMTDTE            ; CHECK OUTPUT
DTEGI   MOV     A,R28             ; SAVE IT IN THE BUF2
        OR      %BIT5,R2          ; SET BUF2 FULL FLAG
        JMP     XMTDTE            ; CHECK OUTPUT
*
DTEGER  CLR     A                 ; SQUELCH THE
        STA     @MPRTC            ; 320 MODEM
        MOVD    %BUFERR,MSGL      ; SEND ERROR MESSAGE
        CALL    @PRINT            ; TO USER TERMINAL
        BR      @TOP              ; EXIT ROUTINE
*
DTEPUT  EQU     $
        BTJZ    %BIT4,R2,MRC1     ; CHECK FOR CHARACTER READY
        MOV     R29,A             ; GET BUFFERED CHARACTER
        MOVP    A,TXBUF           ; SEND IT TO THE DTE
        AND     %NOT4,R2          ; RESET BUFFER FULL FLAG
        JMP     MRC1              ; RETURN TO FLAG LOOP
*
*   PAGE
*
*   RECEIVE DIBITS FROM THE 320
*
MRC2    ANDP    %>7F,CPORT        ; RESET WRITE ACKNOWLEDGE
        ORP     %>80,CPORT        ; BY TOGGING LINES
        MOVP    DPORT,A           ; GET THE RETURNED DATA
        MOV     A,R10             ; AND HOLD IT IN R10
        BTJZ    %BIT5,A,CHKTCH    ; IF NO CARRIER THEN DONE
        AND     %3,R10            ; AND OFF STATUS
        CALL    @DESCRAM          ; DESCRAMBLE IT
        BTJO    %BIT2,R2,RCHAR1   ; CHECK FOR REC CHAR ACTIVE
        RRC     R10               ; CHECK DIBIT0
        JC      RNB               ; IF HIGH THEN CHECK NEXT
        RRC     R10               ; SAVE LSB OF RECEIVE CHAR
        RRC     R5                ; IN CHAR HOLD REG
        MOV     %7,R22            ; SET REC BIT COUNT REG
        JMP     RCHAR0            ; SKIP OVER NEXT CHECK
*
RNB     RRC     R10               ; CHECK DIBIT1
        JC      CHKTCH            ; IF HIGH THEN CHECK XMTCHAR
        MOV     %8,R22            ; SET REC BIT COUNT REG
RCHAR0  OR      %BIT2,R2          ; SET REC CHAR ACTIVE
        BR      @RECDTE          ; CHECK DTE

```

```

*
RCHAR1 SUB    %2,R22          ; CHECK BIT POSITION
        JP     RCHAR3         ; IF > 0 GET 2 BITS
        JZ     RCHAR2         ; IF = 0 GET 1 BIT
        RRC    R10            ; PUT BIT7 INTO
        RRC    R5             ; REC CHAR HOLD REG
        MOV    R5,R29         ; PUT CHAR IN OUT BUFFER
        OR     %BIT4,R2       ; SET BUFFER FULL FLAG
        CLR    R5             ; CLEAR BUFFER FOR NEXT CHAR
        AND    %NOT2,R2       ; RESET REC CHAR ACTIVE
        JMP    RNB            ; CHECK DIBIT1 FOR START BIT
*
RCHAR2 RRC    R10            ; SAVE MSB OF RECEIVE CHAR
        RRC    R5             ; INTO REC CHAR HOLD REG
        RRC    R10            ; PUT BIT7 INTO
        RRC    R5             ; REC CHAR HOLD REG
        MOV    R5,R29         ; PUT CHAR IN OUT BUFFER
        OR     %BIT4,R2       ; SET BUFFER FULL FLAG
        CLR    R5             ; CLEAR BUFFER FOR NEXT CHAR
        AND    %NOT2,R2       ; RESET REC CHAR ACTIVE
        BR     @RECDTE        ; CHECK DTE
*
RCHAR3 RRC    R10            ; MOVE DIBIT0 TO
        RRC    R5             ; REC CHAR HOLD REG
        RRC    R10            ; MOVE DIBIT1 TO
        RRC    R5             ; REC CHAR HOLD REG
        BR     @RECDTE        ; CHECK DTE
*
        PAGE
*
        SEND DIBITS TO THE 320
*
MRC3 ANDP    %>BF,CPORT      ; RESET ACKNOWLEDGE
        ORP    %>40,CPORT     ; BY TOGGILING LINES
        BTJO   %BIT3,R2,TCHAR0 ; CHECK FOR TRANS CHAR ACTIVE
        BR     @STOPB2        ; IF NOT SEND STOPBITS
TCHAR0 CLR    R10            ; CLEAR OUT DIBIT REG
        SUB    %2,R21         ; CHECK POSITION
        JP     TCHAR6         ; > 2 MEANS TRANSMIT BITS
        JNZ    TCHAR3         ; IF PATTERN ONE THEN ODD
        RRC    R7             ; GET BIT 7 FROM CHAR
        JNC    TCH00          ; IF NO CARRY DIBIT0=0
        OR     %BIT0,R10       ; ELSE DIBIT0=1
TCH00 BTJO   %BIT5,R2,TCHAR1   ; IF BUF2 EMPTY
        AND    %NOT3,R2       ; RESET TRAN ACTIVE BIT
        OR     %BIT1,R10       ; SET DIBIT1 TO STOP
        JMP    TCHSND          ; AND SEND DIBIT
TCHAR1 CMP    %9,R24          ; CHECK CHAR COUNT
        JL     TCHAR2         ; IF < DON'T DELETE STOPBIT
        CLR    R24            ; CLEAR BYTE COUNT
        AND    %NOT1,R10       ; SEND DIBIT1 TO START
        MOV    R28,R7          ; LOAD IN NEW CHAR
        MOV    %9,R21         ; SET BIT COUNT
        AND    %NOT5,R2       ; RESET BUF2 FULL FLAG
        JMP    TCHSND          ; SEND THE DIBIT
TCHAR2 OR     %BIT1,R10        ; SEND DIBIT1 TO STOP
        MOV    R28,R7          ; LOAD IN NEW CHAR
        OR     %BIT7,R2        ; FLAG IN START BIT
        MOV    %>A,R21         ; SET BIT COUNT
        AND    %NOT5,R2       ; RESET BUF2 FULL FLAG
        JMP    TCHSND          ; SEND THE DIBIT
TCHAR3 BTJO   %BIT5,R2,TCHAR4   ; IF BUF2 EMPTY
        AND    %NOT3,R2       ; RESET TRAN ACTIVE BIT
        BR     @STOPB2        ; AND SEND MARKS
TCHAR4 CMP    %9,R24          ; CHECK CHAR COUNT
        JL     TCHAR5         ; IF < DON'T DELETE STOPBIT
        CLR    R24            ; CLEAR BYTE COUNT

```



```

        MOV     R28,R7           ; LOAD IN NEW CHAR
        OR      %BIT7,R2        ; FLAG IN START BIT
        MOV     %B,R21          ; SET BIT COUNT
        AND     %NOT5,R2        ; RESET BUF2 FULL FLAG
        JMP     TCHAR6          ; SEND THE DIBIT
TCHAR5  MOV     %1,R10           ; SEND STOP THEN START
        MOV     R28,R7           ; LOAD IN NEW CHAR
        MOV     %9,R21          ; SET BIT COUNT
        AND     %NOT5,R2        ; RESET BUF2 FULL FLAG
        JMP     TCHSND          ; SEND THE DIBIT
TCHAR6  BTJZ    %BIT7,R2,TCHAR7 ; START BIT NEEDED
        AND     %NOT7,R2        ; RESET START BIT FLAG
        JMP     TCHO1           ; SKIP DIBIT1
TCHAR7  RRC     R7               ; GET NEXT BIT OF CHAR
        JNC     TCHO1           ; IF LOW SKIP BIT SET
        OR      %1,R10          ; ELSE SET DIBIT0 TO ONE
TCHO1   RRC     R7               ; GET NEXT BIT OF CHAR
        JNC     TCHSND          ; IF LOW SKIP BIT SET
        OR      %2,R10          ; ELSE SET DIBIT1 TO ONE
TCHSND  EQU     $
        CALL    @SCRAM          ; AND SCRAMBLE IT
        MOV     R10,A           ; HOLD IT FOR TRANSMIT
        ANDP    %>F0,CPORT      ; CLEAR OUT DIBIT VALUE
        ORP     A,CPORT         ; SEND TO PORT
        BR      @RECDTE        ; WAIT FOR RETURN LOOP

```

```

*
PAGE
*****
Receiver descrambler
*****
X(N) = Y(N-17) XOR Y(N-14) XOR Y(N)
*****

```

```

DSCRAM  EQU     $
*
        MOV     R10,B           ; SAVE SCRAMBLED DIBIT
        CLR     R16             ; CLEAR THE Y(N-14) REFERENCE
        CLR     R17             ; CLEAR THE Y(N-17) REFERENCE
        MOV     R11,A           ; GET THE DESCRAMBLER HISTORY
        RL      A               ; SHIFT OUT Y(N-18)
        RLC     A               ; GET HISTORY Y(N-17)
        RLC     R17             ; AND PUT INTO REFERENCE
        RLC     A               ; SHIFT OFF TWO MORE BITS
        RLC     R17             ; SAVE Y(N-16) REFERENCE
        RLC     A               ; TO GET TO THE Y(N-14)
        RLC     A               ; AND GET HISTORY
        RLC     R16             ; AND PUT INTO REFERENCE
        RLC     A               ; GET HISTORY Y(N-13)
        RLC     R16             ; AND PUT INTO REFERENCE
        XOR     R16,R10         ; R10=X(N) XOR Y(N-14)
        XOR     R17,R10         ; R10=X(N) XOR Y(N-14) XOR Y(N-17)
        CLRC                    ; CLEAR OUT THE CARRY BIT
        RRC     R10             ; REVERSE THE DIBITS FOR
        JNC     OVRSW1          ; ALLIGNMENT WITH SCRAMBLER
        OR      %2,R10          ; IF CARRY THEN BIT HIGH
OVRSW1  EQU     $
*
        RLC     R13             ; SHIFT UP THE LSB HISTORY BITS
        RLC     R12             ; AND CARRY TO CSB HISTORY BITS
        RLC     R11             ; AND CARRY TO MSB HISTORY BITS
        CLRC                    ; CLEAR THE CARRY BIT
        RLC     R13             ; SHIFT UP THE LSB HISTORY BITS
        RLC     R12             ; AND CARRY TO CSB HISTORY BITS
        RLC     R11             ; AND CARRY TO MSB HISTORY BITS
        RRC     B               ; GET DIBIT0 AND
        RRC     R13             ; AND SHIFT IT INTO R13
        RRC     B               ; GET DIBIT0 AND
        RRC     R13             ; AND SHIFT IT INTO R13
*
RETS

```

```

*
***** PAGE *****
***** Transmitter Scrambler *****
***** Y(N) = Y(N-17) XOR Y(N-14) XOR X(N) *****
*****
SCRAM EQU $
      CLRC          ; CLEAR OUT THE CARRY BIT
      RRC R10       ; REVERSE THE DIBITS FOR
      JNC OVRSW2    ; ALIGNMENT WITH SCRAMBLER
      OR  %2,R10    ; IF CARRY THEN BIT HIGH
OVRSW2 EQU $
      CLR R16       ; CLEAR THE Y(N-14) REFERENCE
      CLR R17       ; CLEAR THE Y(N-17) REFERENCE
      MOV R18,A      ; GET THE SCRAMBLER HISTORY
      RL A          ; SHIFT OUT Y(N-18)
      RLC A          ; GET HISTORY Y(N-17)
      RLC R17        ; AND PUT INTO REFERENCE
      RLC A          ; SHIFT OFF TWO MORE BITS
      RLC R17        ; SAVE Y(N-16) REFERENCE
      RLC A          ; TO GET TO THE Y(N-14)
      RLC A          ; AND GET HISTORY
      RLC R16        ; AND PUT INTO REFERENCE
      RLC A          ; GET HISTORY Y(N-13)
      RLC R16        ; AND PUT INTO REFERENCE
      XOR R16,R10    ; R10=X(N) XOR Y(N-14)
      XOR R17,R10    ; R10=X(N) XOR Y(N-14) XOR Y(N-17)
*
      MOV R10,B      ; HOLD SCRAMBLED DIBIT FOR HISTORY
      RLC R20        ; SHIFT UP THE LSB HISTORY BITS
      RLC R19        ; AND CARRY TO CSB HISTORY BITS
      RLC R18        ; AND CARRY TO MSB HISTORY BITS
      CLRC          ; CLEAR CARRY BIT
      RLC R20        ; SHIFT UP THE LSB HISTORY BITS
      RLC R19        ; AND CARRY TO CSB HISTORY BITS
      RLC R18        ; AND CARRY TO MSB HISTORY BITS
      RRC B          ; GET DIBIT0 AND
      RRC R20        ; AND SHIFT IT INTO R20
      RRC B          ; GET DIBIT0 AND
      RRC R20        ; AND SHIFT IT INTO R20
*
      RETS
*
***** PAGE *****
***** MSDLY - Wait MTIME number of milliseconds *****
*****
MSDLY EQU $
      MOV %CNTVAL,DELYR1 ; load the inner counter (9)
HERE2 DJNZ DELYR1,HERE2  ; (9+2)
      DECD MTIME         ; (11)
      JC  MSDLY          ; (7)
      RETS
*****
***** SECDLY - Wait VALUE number of seconds *****
*****
SECDLY CMP %0,VALUE
      JEQ NODLY
NXTSEC MOVD %1001,MTIME
      CALL @MSDLY
      DJNZ VALUE,NXTSEC
NODLY RETS
*****
***** PRINT subroutine *****
*****
* MSGM and MSGL contain the address of text to print
* for messages to the screen
*

```

```

PRINT CALL @CRLF
PRINT1 LDA *MSG1
      JZ WAIT6
      MOVP A, TXBUF ; print each character in text statement
WAIT5 BTJZP %BIT0, SSTAT, WAIT5 ; wait for txbuf ready
      INC MSG1
      ADC %0, MSG1
      JMP PRINT1
WAIT6 CALL @CRLF
      RETS
*
* send carriage return/line feed
*
CRLF MOV S0, A
      MOVP A, TXBUF ; send carriage return
CRWAIT BTJZP %BIT0, SSTAT, CRWAIT
      MOV S1, A
      MOVP A, TXBUF ; send line feed
LFWAIT BTJZP %BIT2, SSTAT, LFWAIT
      RETS
*
PAGE
*****-----*****
***** PRINT subroutine *****
*****-----*****
AUTOBD EQU $
      MOV %>20, A ; SET BAUD CLOCK FOR
      MOVP A, T3DATA ; FOR OVERSPEED DTE
SETMOD MOVP %0, P17 ; Write to P17 to guarantee
      MOVP %>60, SCTL0 ; we are talking to SCTL0, then reset
* ; serial port
      MOVP B, SMODE ;
      MOVP %>15, SCTL0 ;
      MOVP %>40, SCTL1 ;
*
      MOVP %BIT6, SCTL0 ; Parity error, parity is disabled in DTE.
      MOVP %>6E, SMODE ; Disable parity of port
      MOVP %>15, SCTL0
      MOVP %>40, SCTL1
      RETS
*
*****-----*****
***** screen messages - text statements *****
*****-----*****
ERROR TEXT 'ERROR'
      BYTE 0
*
BUFERR TEXT 'DTE BUFFER OVERFLOW ERROR'
      BYTE 0
*
CONN12 TEXT 'CONNECT 1200'
      BYTE 0
*
CONN3 TEXT 'CONNECT 300'
      BYTE 0
*
NOCAR TEXT 'NO CARRIER'
      BYTE 0
*
RCALL TEXT 'RING'
      BYTE 0
*
RESET TEXT 'OK'
      BYTE 0
*
LDLBM TEXT 'EXECUTE LDLB, ENTER CHARACTERS'
      BYTE 0
*

```

```

IALBM TEXT 'INITIALIZE 320 FOR ALB TEST'
    BYTE 0
*
IORGM TEXT 'INITIALIZE 320 FOR ORIGINATE MODE'
    BYTE 0
*
IENBM TEXT 'INITIALIZE 320 TO REENABLE RECEIVER'
    BYTE 0
*
ISQTM TEXT 'INITIALIZE 320 TO SQUELCH RECEIVER'
    BYTE 0
*
ANSM TEXT 'INITIALIZE 320 TO ANSWER MODE'
    BYTE 0
*
HONM TEXT 'PUT LINE ON HOOK'
    BYTE 0
*
HOFFM TEXT 'TAKE LINE OFF HOOK'
    BYTE 0
*
HELPM TEXT 'TABLE OF COMMANDS'
    BYTE >0D,>0A
    TEXT 'A ==> PUT MODEM IN ANSWER MODE'
    BYTE >0D,>0A
    TEXT 'D ==> BLIND DIAL FOLLOWING DIGITS'
    BYTE >0D,>0A
    TEXT 'E ==> ENABLE 320 RECEIVER'
    BYTE >0D,>0A
    TEXT 'H ==> DISPLAY HELP LIST'
    BYTE >0D,>0A
    TEXT 'J ==> PUT LINE ON HOOK'
    BYTE >0D,>0A
    TEXT 'K ==> TAKE LINE OFF HOOK'
    BYTE >0D,>0A
    TEXT 'L ==> RUN DIGITAL LOOP BACK TEST'
    BYTE >0D,>0A
    TEXT 'M ==> RUN ANALOG LOOP BACK TEST'
    BYTE >0D,>0A
    TEXT 'O ==> PUT MODEM IN ANSWER MODE'
    BYTE >0D,>0A
    TEXT 'R ==> RUN THE 320 MODEM'
    BYTE >0D,>0A
    TEXT 'S ==> SQUELCH THE 320 RECEIVER'
    BYTE >0D,>0A
    TEXT 'Z ==> RESTART THE 7000'
    BYTE >0D,>0A
    BYTE 0
*
HELLO TEXT 'DSP MODEM. VERSION 1.0'
    BYTE 0
*****-----*****
*****      command address table      *****
*****-----*****
COMLIS DATA ANSMDM          ; INITIALIZE TO ANSWER
        DATA ERR
        DATA ERR
        DATA DB             ; dial command
        DATA ENBREC         ; REENABLE RECEIVER ON 320
        DATA ERR
        DATA ERR
        DATA HELP           ; HELP LIST
        DATA ERR
        DATA HOOKON          ; TAKE LINE ON HOOK
        DATA HOOKOF         ; TAKE LINE OFF HOOK
        DATA LDLB           ; LOCAL DIGITAL LOOP BACK
        DATA IALB           ; INITIALIZE TO ALB MODE

```

```

DATA ERR
DATA IORIG                ; INITIALIZE TO ORIGINATE
DATA ERR
DATA ERR
DATA B2:2                ; RUN MODEM ROUTINE
DATA SQTREC              ; SQUELCH THE RECEIVER
DATA ERR
DATA ERR
DATA ERR
DATA ERR
DATA ERR
DATA INIT                ; reset command
*****-----*****
***** INITIALIZE TO ALB MODE *****
*****-----*****
IALB EQU $
CLR A                    ; CYCLE THE CLEAR LINES
STA @MPRTC              ; OF THE I/O CONTROL
ORP %BIT0,CPORT         ; SET ALB INIT COMMAND
ORP %>F0,CPORT          ; PUT 320 IN INIT COMMAND MODE
IALB1 BTJOP %BIT6,DPORT,IALB1 ; CHECK 320 RESPONSE
MOVD %IALBM,MSGL        ; GET CONFIRMATION MSG
CALL @PRINT             ; AND SEND IT
CLR A                   ; CLEAR OUT THE COMMAND
STA @MPRTC              ; FROM I/O LINES
BR @TOP                 ; EXIT ROUTINE
*****-----*****
***** INITIALIZE TO ORIGINATE MODE *****
*****-----*****
IORIG EQU $
CLR A                    ; CYCLE THE CLEAR LINES
STA @MPRTC              ; OF THE I/O CONTROL
ORP %BIT1,CPORT         ; SET ORIG INIT COMMAND
ORP %>F0,CPORT          ; PUT 320 IN INIT COMMAND MODE
IORIG1 BTJOP %BIT6,DPORT,IORIG1 ; CHECK 320 RESPONSE
MOVD %IORGM,MSGL        ; GET CONFIRMATION MSG
CALL @PRINT             ; AND SEND IT
CLR A                   ; CLEAR OUT THE COMMAND
STA @MPRTC              ; FROM I/O LINES
BR @TOP                 ; EXIT ROUTINE
*****-----*****
***** INITIALIZE TO RECEIVER SQUELCHED *****
*****-----*****
SQTREC EQU $
CLR A                    ; CYCLE THE CLEAR LINES
STA @MPRTC              ; OF THE I/O CONTROL
ORP %3,CPORT            ; SET SQT INIT COMMAND
ORP %>F0,CPORT          ; PUT 320 IN INIT COMMAND MODE
ISQT1 BTJOP %BIT6,DPORT,ISQT1 ; CHECK 320 RESPONSE
MOVD %ISQTM,MSGL        ; GET CONFIRMATION MSG
CALL @PRINT             ; AND SEND IT
CLR A                   ; CLEAR OUT THE COMMAND
STA @MPRTC              ; FROM I/O LINES
BR @TOP                 ; EXIT ROUTINE
*****-----*****
***** INITIALIZE TO REENABLE RECEIVER *****
*****-----*****
ENBREC EQU $
CLR A                    ; CYCLE THE CLEAR LINES
STA @MPRTC              ; OF THE I/O CONTROL
ORP %4,CPORT            ; SET ENB INIT COMMAND
ORP %>F0,CPORT          ; PUT 320 IN INIT COMMAND MODE
IENB1 BTJOP %BIT6,DPORT,IENB1 ; CHECK 320 RESPONSE
MOVD %IENBM,MSGL        ; GET CONFIRMATION MSG
CALL @PRINT             ; AND SEND IT
CLR A                   ; CLEAR OUT THE COMMAND

```

```

        STA @MPRTC          ; FROM I/O LINES
        BR @TOP             ; EXIT ROUTINE
*****
***** INITIALIZE TO ANSWER MODE *****
*****
ANSMDM EQU $
        CLR A               ; CYCLE THE CLEAR LINES
        STA @MPRTC          ; OF THE I/O CONTROL
        ORP %5,CPORT        ; SET ANS INIT COMMAND
        ORP %>F0,CPORT       ; PUT 320 IN INIT COMMAND MODE
IANS1 BTJOP %BIT6,DPORT,IANS1 ; CHECK 320 RESPONSE
        MOVD %ANSM,MSGL      ; GET CONFIRMATION MSG
        CALL @PRINT          ; AND SEND IT
        CLR A               ; CLEAR OUT THE COMMAND
        STA @MPRTC          ; FROM I/O LINES
        BR @TOP             ; EXIT ROUTINE
*****
***** PUT LINE ON HOOK *****
*****
HOOKON EQU $
        ANDP %NOT7,APOINT    ; PUT MODEM BACK ON HOOK
        MOVD %HONM,MSGL      ; GET CONFIRMATION MSG
        CALL @PRINT          ; AND SEND IT
        BR @TOP             ; EXIT ROUTINE
*****
***** TAKE LINE OFF HOOK *****
*****
HOOKOF EQU $
        ORP %BIT7,APOINT     ; TAKE OFF HOOK
        MOVD %HOFFM,MSGL     ; GET CONFIRMATION MSG
        CALL @PRINT          ; AND SEND IT
        BR @TOP             ; EXIT ROUTINE
*****
***** DISPLAY HELP LIST *****
*****
HELP EQU $
        MOVD %HELPM,MSGL     ; GET CONFIRMATION MSG
        CALL @PRINT          ; AND SEND IT
        BR @TOP             ; EXIT ROUTINE
*****
***** Clear command buffer *****
*****
CLEAR CLR A
        CLR B
MORE STA @ADDBOT(B)         ; zero command register
        INC B
        CMP %40,B           ; are we done yet?
        JNE MORE
        RETS
*****
***** Auto-answer routine *****
*****
INT1 BTJZP %BIT1,APOINT,ANSMOD ; DTR_ must be active, else return
    RETI
*
ANSMOD CLR S4
        ORP %BIT0,BPORT      ; Turn off CTS_
        MOVP %>2A,IOCNT0     ; activate timer interrupt
        EINT
RIHIGH ORP %BIT1,IOCNT0
        BTJOP %BIT1,IOCNT0,RIHIGH ; Wait RI to fall
        ORP %BIT1,IOCNT0
        MOVD %50,COUNT1
STALOW MOVD %10,MSTIME
        CALL @MSDLY
        BTJOP %BIT1,IOCNT0,RIHIGH ; separate rings
        DJNZ COUNT1,STALOW

```

```

*
*      MOVD    %RCALL,MSGL      ; send ring message
*      CALL    @PRINT
*      ORP     %BIT1,IOCNT0
*
*
* LABEL0 INC    S4              ; increment ring counter
*      CMP     S3,S4
*      JZ      PICKUP
* NXTRNG MOVD    %100,COUNT1
* RILOW  MOVD    %100,MSTIME
*      CALL    @MSDLY
*      BTJOP   %BIT1,IOCNT0,RIHIGH ; check RI_ every 100 msec
*      DJNZ    COUNT1,RILOW
*
*
* * no rings, abort answer
*
*      ANDP    %NOT0,BPORT
*      RETI
*
*
* * Pickup the phone and go through answer procedures.
*
* PICKUP ORP     %BIT7,APORT      ; Go off hook
*      ORP     %BIT1,BPORT      ; DSR is active
*
*
* * wait at least 2 seconds for billing delay
*
*      MOV     %2,VALUE          ; must wait at least 2 secds
* BDELAY CALL    @SECDLY          ; Wait 2 seconds
*      MOV     %18,CWT1          ; Initialize carrier abort timer.
*      CLR     CWT2
*      ORP     %BIT2,IOCNT0      ; Enable carrier abort interrupt
*
*
* * determine if B212A or B103J mode
*
*      ANDP    %BIT4,CPORT      ; answer mode (to 32010)
*      ORP     %BIT4,APORT      ; ATE=1
*      ANDP    %NOT2,APORT      ; Unsquell 532, send 2225hz tone
*
*
*      MOVD    %600,INT5TM      ; load timer
*
*
* ORGWT0 BTJZP   %BIT5,DPORT,BE212 ; check for EDT_
*      BTJZP   %BIT0,APORT,BE103 ; check for DCD_
*      JMP     ORGWT0           ; keep looping till carrier timer aborts
* BE212 MOVP     %>0C,IOCNT1      ; enable INT5
*      JMP     GOTEDT           ; BELL 212 selected
* BE103 MOVP     %>0C,IOCNT1      ; enable INT5
*
*
* * Bell 103J selected
*
*      MOV     %150,COUNT
* DCDWT0 BTJOP   %BIT0,APORT,ORGWT0 ; check for DCD_
*      MOVD    %1,MSTIME
*      CALL    @MSDLY
*      DJNZ    COUNT,DCDWT0
*
*
*      MOVP     %>00,IOCNT0      ; Got DCD_, disable abort interrupt
*      MOVP     %>00,IOCNT1
*
*
*      MOVD    %CONN3,MSGL
*      CALL    @PRINT
*
*
*      MOVD    %765,MSTIME
*      CALL    @MSDLY
*      BR      @DAT103
*
*
* GOTEDT MOV     %150,COUNT      ; EDT_ active for at least 150 ms
* EDTWT2 BTJOP   %BIT5,DPORT,ORGWT0

```

```

        MOVD    %1,MSTIME
        CALL    @MSDLY
        DJNZ    COUNT,EDTWT2
*
        MOVP    %>00,IOCNT0      ; Got EDT_, disable abort interrupt
        MOVP    %>00,IOCNT1
        ORP     %BIT2,APORT      ; Squelch 532
        ANDP    %NOT4,APORT      ; ATE=0 (EX1 MODE)
        MOVD    %CONN12,MSG1
        CALL    @PRINT           ; CONNECT 1200
        ANDP    %NOT0,BPORT      ; CPORT is active (CTS_=0)
*
        MOVD    %765,MSTIME      ; Wait 765 ms
        CALL    @MSDLY
        BR      @B212           ; 212A mode, act as 32010 to DTE interface
*
*   Call Initiation Routines.
*
*   We are now in data mode.  Wait for a disconnect.
*
DAT103 ANDP    %NOT0,BPORT      ; Activate CTS_
*
*   look for escape character
*
LP103A MOV     %3,TEMP1
LP103B EQU     $
*       BTJOP   %BIT1,APORT,NODTR0 ; no DTR_
*       BTJOP   %BIT0,APORT,DIS103 ; no DCD_
LP103E BTJZP   %BIT1,SCTL0,LP103E ; received char?
        MOVP    RXBUF,A
        CMP     S5,A           ; escape character?
        JNE     LP103A
        DJNZ    TEMP1,LP103B
*
*   we now have three escape characters.  start escape code timer
*
        MOV     %50,COUNT1
LP103C MOVD    %20,COUNT
LP103D MOVD    %1,MSTIME
        CALL    @MSDLY
        BTJOP   %BIT1,APORT,NODTR0 ; no DTR_
        BTJOP   %BIT0,APORT,DIS103 ; no DCD_
        BTJOP   %BIT1,SCTL0,LP103A
        DJNZ    COUNT,LP103D
        DJNZ    COUNT1,LP103C
*
*   everything checked out O.K.
*
        JMP     CM103
*
NODTR0 MOV     %5,COUNT          ; 5 m/s check of DTR_
NODTR1 MOVD    %1,MSTIME
        CALL    @MSDLY
        BTJZP   %BIT1,APORT,LP103B
        DJNZ    COUNT,NODTR1
*
*   Disconnect from 103 data mode
*
DIS103 ORP     %BIT2,APORT      ; Squelch 532
        ANDP    %NOT7,APORT     ; Go on hook
        MOVP    %>03,IOCNT0     ; Enable interrupt 1
        MOVD    %NOCAR,MSG1
        CALL    @PRINT           ; Send disconnect message
TCODE2 BTJZP   %BIT0,SSTAT,TCODE2
        EINT
        BR      @INIT
*

```



```

* 103 COMMAND MODE
*
CM103  ANDP  %NOT0,BPORT      ; Activate CTS_
        MOVD  %RESET,MSGL
        CALL  @PRINT
        BR    @LOOK          ; look for new command
*
*****
****  TIMEOUT INTERRUPT OF CARRIER DETECT  ****
*****
INT2   EQU    $
        DECD  CWT2            ; DECREMENT SECONDARY COUNTER
        JNC   CABORT          ; IF COUNTED OUT THEN APORT
        RETI
CABORT ANDP  %NOT7,APORT      ; TIMEOUT NOT COMPLETE CONTINUE
        ORP   %BIT2,APORT     ; GO ON HOOK
        ANDP  %NOT0,BPORT     ; SQUELCH 532
        ORP   %BIT3,IOCNT0    ; ACTIVATE CTS
        EINT                                ; DISBLE TIMER
        MOVD  %NOCAR,MSGL     ; SEND NO CARRIER
        CALL  @PRINT          ; MESSAGE TO DTE
        BR    @LOOK          ; LOOK FOR NEXT COMMAND
*
INT3   RETI
INT4   RETI
INT5   RETI
        AORG  >FFF4
VECT5  DATA  INT5
VECT4  DATA  INT4
VECT3  DATA  INT3
VECT2  DATA  INT2
VECT1  DATA  INT1
VECT0  DATA  INT1

```