Implementation of an FSK Modem Using the TMS320C17

APPLICATION REPORT: SPRA080

Phil Evans
Regional Technology Center
Ottawa, Canada
Al Lovrich
Digital Signal Processor Products
Semiconductor Group
Texas Instruments

Digital Signal Processing Solutions

Texas Instruments

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.

Copyright © 1997, Texas Instruments Incorporated

TRADEMARKS

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

Implementation of an FSK Modem Using the TMS320C17

Abstract

This report presents a complete hardware design for a splitband modem, and the software to implement a V.21/Bell 103 300-bps modem, using a TMS320C17 DSP.

- ☐ The first section reviews basic modem concepts and definitions and introduces frequency shift keying (FSK) data modulation.
- ☐ The second section describes the major functional blocks of the FSK modem system.
  - Host Interface
  - Modem Controller
  - Digital Signal Processor
  - Analog Front End
- ☐ The third section discusses DSP software implementation of the V.21/Bell 103 300-bps modem, using a TMS320C17 DSP.
- ☐ The fourth section reviews some issues involved with incorporating additional code into DSP software provided in Appendix B.
- ☐ The fifth section summarizes conclusions.
- ☐ Appendix A is a derivation of the filter coefficient value required for the sample fraction time delay.
- ☐ Appendix B is the source code listing for the TMS320C17 modem implementation.

The report also includes flow chart and tabular frequency and phase step data.

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

This application report presents an implementation of a 300-bit-per-second (BPS) modem conforming to the V.21 and Bell 103 standards, using a TMS320C17 Digital Signal Processor (DSP).

The purpose of this application report is, with references [1], [2], [3], to provide a complete hardware design for a splitband modem and the software to implement a V.21/Bell 103 300-bps modem. The designer can then concentrate on developing value-added functions, such as V.22bis or V.22 standard modems, encryption algorithms, etc. These value-added functions are implemented in software and can be easily incorporated into the TMS320C17 software provided in Appendix B.

The structure of this report is as follows:

- The first section reviews basic modem concepts and definitions and introduces the reader to frequency shift keying (FSK) data modulation.
- The second section describes the major functional blocks of the FSK modem system presented in this report:
  - — Host interface,
  - — Modem controller,
  - — Digital signal processor, and
  - — Analog front end.
- References to documents describing the actual hardware implementation are provided.
- The third section discusses the DSP software implementation of the V.21/Bell 103 modulator/demodulator using the TMS320C17 DSP.
- The fourth section reviews some of the issues involved with incorporating additional code into DSP software provided in Appendix B.
- The fifth section concludes this report.
- Appendix A is a derivation of the filter coefficient value required for the sample fraction time delay.
- Appendix B is the source code listing for the TMS320C17 modulator and demodulator implementation.

Background

Over the past decade there has been a proliferation in the number and the use of computer systems. Accompanying this growth, there has been an increased demand for data communications between the various computer systems and terminals.

One of the most convenient and frequently used methods of data communications between geographically separated computer equipment is via the Public Switched Telephone Network (PTSN). The essential element for this method of data communication is the modem.

The modem converts the digital data it receives from the computer system or terminal into a modulated analog signal that is transmitted via the telephone network to the destination computer system or terminal. At the destination, the receive modem demodulates the received signal and transfers the digital data to the receiving terminal or computer system.

Table 1 shows a number of popular modem standards as specified by either the International Telegraph and Telephone Consultive Committee (CCITT) or the Bell System.

Table 1. Bell and CCITT Modem Standards

Modem	Standard	Type*	Modulation	Data Rate (BPS)	Duplex
Bell	103	S/B	FSK	300	Full
	202	S/B	FSK	1200	Half
	212A	S/B	DPSK	1200	Full
	201	S/B	DPSK	2400	Half
CCITT	V.21	S/B	FSK	300	Full
	V.22	S/B	DPSK	1200	Full
	V.22bis	S/B	QAM	2400	Full
	V.32	E/C	QAM	9600	Full

* S/B = Split band E/C = Echo Cancelling

Modems can be either half-duplex or full-duplex. In a half-duplex system, the transmission can be in either direction; however, only one direction is possible at a time. A half-duplex modem cannot simultaneously transmit and receive information. At the end of its transmission sequence, the modem must advise the receiving modem that the sequence is complete. The receiving modem may then begin transmitting data.

In a full-duplex system, the data transmission is bidirectional. Both modems may simultaneously transmit and receive data. Bidirectional (simultaneous data transmission) is achieved by either splitband or echo cancellation techniques.

Figure 1 shows the spectral response of a typical telephone channel. A splitband modem uses a filtering scheme to separate the telephone channel into two distinct frequency bands. One band is dedicated to the transmissions of the originate modem, the other band is dedicated to transmissions of the answer modem. To separate the received signal from the received and transmitted signal that is detected on the two-wire telephone line, the modem removes the transmitted signal frequency band using a splitband filter [1], [4], or by other means (such as software implemented on the DSP). Dividing the telephone channel into two separate non-overlapping frequency bands limits the maximum baud rate.

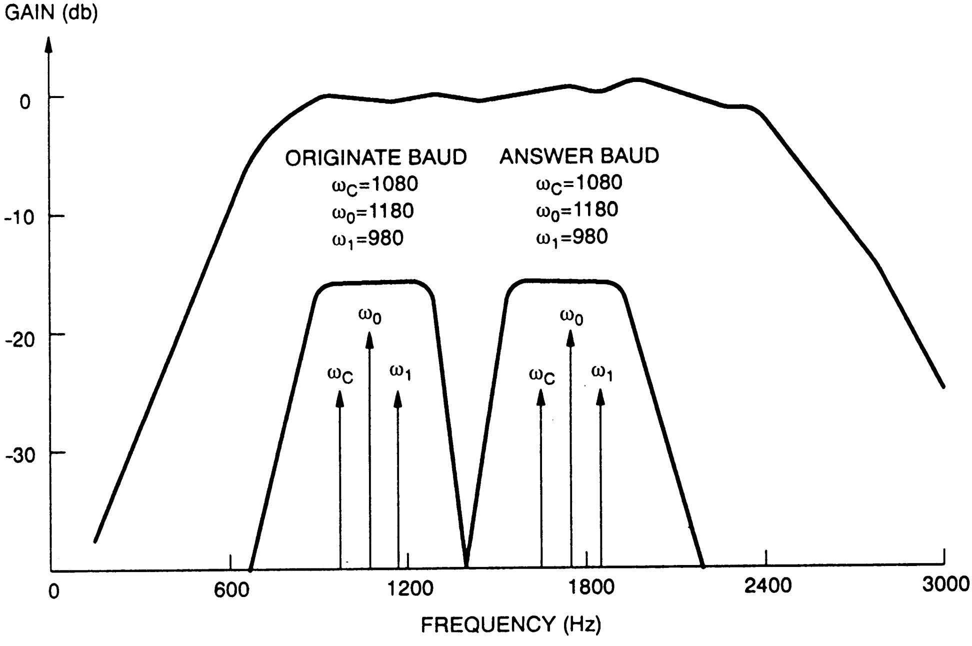


Figure 1. Spectral Response of a Typical Telephone and a V.21 Splitband Modem

The actual bit rate of the channel is determined by the baud rate and the data modulation scheme that is employed. Splitband type modems are typically used in low- to moderate-speed applications. As shown in Table 1, each modem standard uses a particular modulation scheme. For example, CCITT V.21, V.22, and V.22bis standards specify the frequency shift keyed (FSK), phase shift keyed (PSK) and quadrature amplitude modulation (QAM) schemes respectively.

Echo cancellation type modems, such as V.32, transmit both the originate and answer signals on the same channel. This allows both the originate and answer modems to utilize the complete bandwidth of the channel and to maximize the data baud rate. It is still necessary to separate the receive signal from the receive and transmit signal detected on the two-wire telephone line. However, the originate and answer signals are superimposed on the same channel band, and separating techniques that are more sophisticated than those found in splitband-type modems are required. The fact that transmit signal is typically 20 dB stronger than the receive signal, as measured on the transmit Tip and Ring, further complicates the extraction of the receive signal.

Echo cancellation type modems use algorithms that subtract an estimate of the transmit signal from the signal sampled from the two-wire telephone line, to determine the receive signal. Refer to [5] and [6] for further information on Echo cancellation type modems.

Table 2 shows the transmission frequencies for answer and originate modes for both the binary FSK modulated 300-bps V.21 and Bell 103 standards. It also shows details of the V.23 and Bell 202 1200-bps half-duplex standards.

Table 2. Binary FSK Transmission Frequencies

Modem Standard		Carrier (Hz)	1(Mark) (Hz)	0(Space) (Hz)
V.21	Originate Answer	1080 1750	980 1650	1180 1850
BELL 103	Originate Answer	1170 2125	1270 2225	1070 2025
V.23		1700	1300	2100
BELL 202		1700	1200	2200

Since this report is primarily concerned with the 300-bps V.21 and Bell 103 standard modems, it is worthwhile to review FSK data communication.

These are the primary advantages of an FSK system:

1. There is no requirement for carrier phase recovery; this reducing system complexity.

2. Increased immunity to amplitude nonlinearities. FSK is a constant envelope signal, with the information transmitted in the zero crossings. It is less affected by amplitude nonlinearities than amplitude modulated schemes, and

3. The modulator and demodulator architectures are easily implemented in software.

The primary disadvantage of FSK modulation is its low spectral efficiency. Because the telephone network is bandlimited to 4KHz, only moderate data transmission rates over the telephone network are supported by an FSK modulation scheme. As a consequence, FSK is often the favored modulation scheme for very low cost, low-to-moderate speed data communication systems.

Subsequent sections of this report discuss FSK modulation and demodulation in some detail. It is important that you understand the mathematical representations of FSK signals. FSK modulation is represented in the following manner:

$$S(t) = \cos((\omega_c \pm \delta\omega)*t + \phi) \qquad (1)$$

where $S(t)$ = Transmitted signal
ω_c = Carrier frequency
$\delta\omega$ = Frequency shift
t = Time
ϕ = Phase shift

For a given baud period T, S(t) is at a frequency $f_1=(f_c+\delta f)$ or $f_0=(f_c-\delta f)$, corresponding to the transmission of a 1 or 0, respectively, for the duration of the baud period. In some cases, it is convenient to represent

$$\begin{aligned} \omega_0 &= \omega_c - \delta_\omega \\ \omega_1 &= \omega_c - \delta_\omega \end{aligned} \qquad (2)$$

Thus the following identities are true:

$$\begin{aligned} \omega_c &= (\omega_1 + \omega_0)/2 \\ \delta\omega &= (\omega_1 - \omega_0)/2 \end{aligned} \qquad (3)$$

Some binary FSK modulation schemes, such as V.21, have ω_0 greater than ω_1; so by (3), $\delta\omega$ would be negative. Figure 2 shows an FSK signal transmission.

Note that the telephone channel provides limited spectral bandwidth. To achieve progressively higher data rates, more spectrally efficient modulation schemes, such as PSK and QAM, must be used. As spectral efficiency increases, typically, the complexity of the signal modulation and demodulation schemes increase. Additional information on modulation schemes can be found in references [4], [5], [6] and [7].

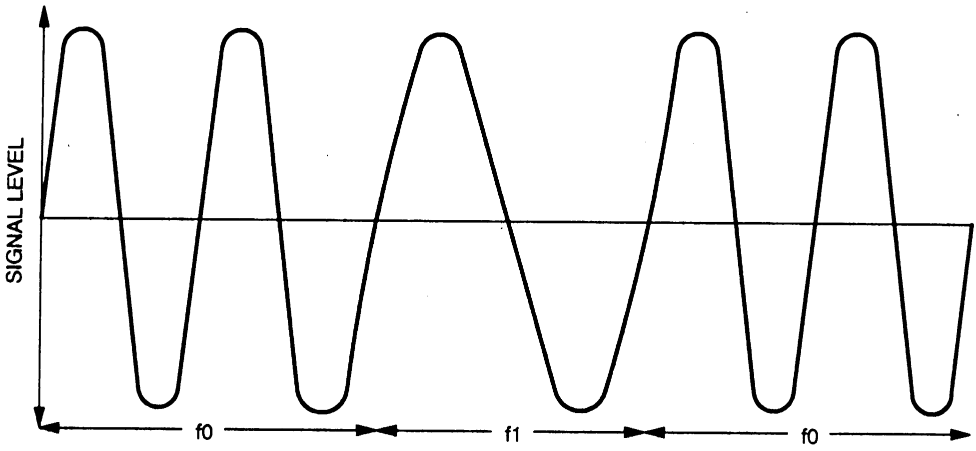


Figure 2. FSK Signal Transmission

System Description

As discussed in the introduction, this application report presents the implementation of a V.21/Bell 103 300-bps FSK modem using a TMS320C17 Digital Signal Processor. The system hardware is identical to that of the Texas Instruments DSP2400 modem [1].

There are significant functional differences between the modem design provided here and the DSP2400 modem. These result from the differences between the TMS320 code provided in Appendix B and the DSP2400 code. The software found in Appendix B implements a V.21/Bell 103 FSK modem. The DSP2400 also implements V.22, Bell 212A, and V.22bis standard modems that implement PSK and QAM modulation/demodulation and the associated carrier recovery, clock recovery, and adaptive equalization functions.

The software in Appendix B provides all the necessary hooks so that the designer can easily incorporate his own custom value-added features (such as V.22 and V.22bis standard modems). Nevertheless, the reader should be aware of the difference between the DSP2400 software implementation and the software in Appendix B, particularly when referring to any DSP2400 related literature [1], [2], [3].

Figure 3 is a block diagram showing the components of the modem system. The modem consists of the following subsystems:

1. Host interface
2. Modem controller
3. Digital signal processor
4. Analog front end

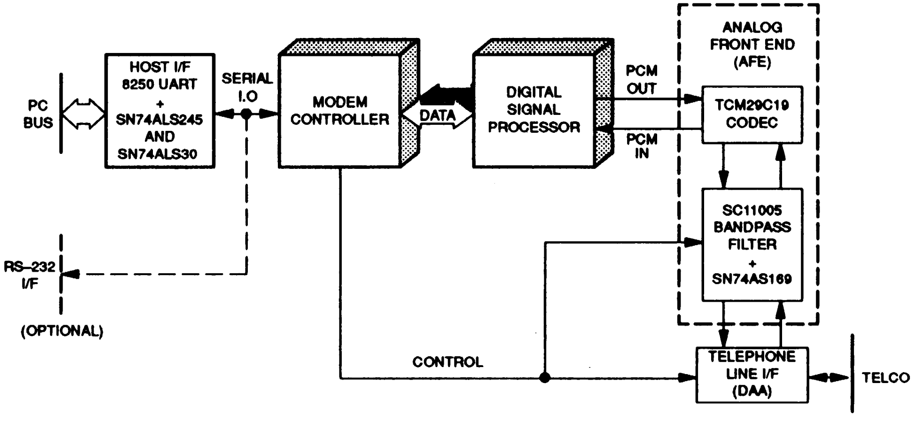


Figure 3. Block Diagram of Modem System Components

The designer must provide an interface between the host data terminal equipment and the modem controller. The DSP2400 uses an 8250 UART (plus a 74LS245 buffer and a 74ALS30 NAND Gate) to interface between a standard PC-AT and the modem controller. A standard RS-232C interface is used between the UART and the modem controller. The circuit diagram and additional information on the host interface used for the DSP2400 Modem can be found in [1].

The modem controller (80C51, TMS70C42, etc.) handles the overall modem control [3], directs the handshaking sequences, etc. It specifically performs the following functions:

1. AT command set interpretation
2. Scrambling/descrambling
3. Pulse dialing
4. Synchronous/asynchronous conversion
5. Modem configuration control
6. Protocol initialization

The modem controller sends a command to the DSP once per baud. Table 3 is a complete list of the commands, showing the structure and functions that are implemented.

Table 3. Modem Controller Commands for the DSP

Command	Code	Description
Protocol Select	Fxh	Select protocol Bits 1, 0 – Speed select 0 0 = 300 BPS 0 1 = Reserved 1 0 = Reserved 1 1 = Reserved Bit 2 – CCITT/Bell 0 = CCITT 1 = Bell Bit 3 – Answer/originate 0 = Answer 1 = Originate
Reserved	Exh	Reserved command
Operation Select	Dxh	Select operating mode (bits 3, 2 reserved) 0 0 = Line mode 0 1 = Analog loopback 1 0 = Reserved 1 1 = Reserved
Reserved	Cxh	Reserved command
Reserved	Bxh	Reserved command
Reserved	Axh	Reserved command
Transmit DTMF Tones	9xh	Dial DTMF and return to configuration mode xxxx = D3-D0; numbers 0-9, A, B, C, D, *, and #

Table 3. Modem Controller Commands for the DSP (Concluded)

Command	Code	Description
Transmit Mode Select	8xh	Enable answer tone/data select Bits 1, 0 = Transmit select 0 0 = Transmit idle 0 1 = Transmit answer tone 1 0 = Transmit data mode enable 1 1 = Reserved Bits 3, 2 = Select answer tone frequency 0 0 = 2100 Hz answer tone (V.21) 0 1 = 2225 Hz answer mark (Bell 103) 1 0 = 2025 Hz answer space (Bell 103) 1 1 = Reserved
Receive Mode Select	7xh	Select receive configuration (bits 3,2 reserved) 0 0 = Receive idle mode 0 1 = Reserved 1 0 = Receive data mode 1 1 = Reserved
Reserved	6xh	Reserved command
FSK Mode	5xh	Select 300 BPS mode (bits 3,2,1 reserved) 0 = 300 BPS mode deselect 1 = 300 BPS mode select
Reserved	4xh	Reserved command
Reserved	3xh	Reserved command
Reserved	2xh	Reserved command
Reserved	1xh	Reserved command
Reserved	0xh	Reserved command

As an example, the DSP2400 uses a masked ROM version of the TMS70C42 microcontroller (denoted as a TMS70C2400A) as the modem controller. The TMS70C2400A source code is available from Texas Instruments and includes provisions for the V.22bis and V.22 standard modems.

One noteworthy advantage of the TMS70C42/TMS320C17 interface is that it requires no external glue logic [7]. For complete information on the TMS70C2400 Modem Controller, including the call originate and answer sequences, refer to [2].

The TMS320 Digital Signal Processor performs the computationally intensive tasks such as modulation, demodulation, and tone generation and detection. It does not perform any control functions. Specifically, the TMS320 DSP performs the following functions:

1. Modulation/demodulation (V.21/Bell 103)
2. Data encoding/decoding
3. Filtering
4. Automatic gain control
5. Tone dialing
6. Call progress monitoring

The DSP is discussed in further detail in the next section of this application report. The DSP source code in Appendix B was originally part of the code developed for the TMS320A2400 Modem Digital Signal Processor (a ROM coded TMS320C17 DSP). The TMS320A2400 source code also includes V.22bis, V.22, and Bell 212A standard modems, with the software implementing the QAM and PSK modulation and demodulation schemes, carrier recovery, clock recovery, automatic gain control, and adaptive equalization functions. The TMS320A2400 and the source code is available from Texas Instruments.

Despite the differences between the code provided in Appendix B and the TMS320A2400 code, [1] and [3] are useful references, providing technical information about TMS320C17 modem applications.

The analog front end is composed of a TCM29C19 combo codec [9], a SC11005 bandpass filter [10] and a data access arrangement (DAA) telephone line interface composed of discrete components. The codec converts an 8-bit μ-law companded bit stream to an analog waveform and vice versa, at a 9.6-KHz sampling frequency. The SC11005 is a splitband filter that separates the transmit and receive carriers and performs the required signal shaping to the analog waveform. The DAA section is composed of a number of discrete components and is required to interface the modem to the public telephone network as dictated by FCC Rules Part 68. The analog front end circuit diagram is found in [1]. Further technical details are found in [2].

The DSP Software Implementation

The code provided in Appendix B is written specifically for a Texas Instruments TMS320C17 Digital Signal Processor. The key architectural features of the TMS320C17 are these:

1. 4 Kwords (8 Kbytes) of on-chip maskable ROM
2. 256 words of on-chip data RAM
3. Two full-duplex serial ports
4. On-chip companding hardware (μ- or A-Law)
5. On-chip sign magnitude/two's complement conversion hardware
6. A coprocessor port
7. 6.25-MIPS maximum execution speed

TMS320E17, with 4 Kwords of on-chip EPROM substituted for the 4 Kwords of maskable ROM, is also available for development and prototyping purposes. Refer to [8] and [11] for additional information on the TMS320C17 and TMS320E17.

The TMS320C17 source code listing file is found in Appendix B. The code requires approximately 50 words of data RAM and occupies 1100 words of program ROM. Of the 1100 words of program memory, 390 are coefficients, and the remaining 710 words are the program instructions. The software consists of a main program that references various subroutines. These are the main subroutines found in the program:

1. Command control interpreter (CCI)
2. FSK transmitter (FSKTX)
3. Dual-tone multifrequency transmitter (Part of FSKTX)
4. Automatic gain control (AGC)
5. FSK receiver (RSTSK)

The next section of text describes the main program. The subroutines are discussed in subsequent sections.

Figure 4 is a block diagram of the main program (code starting at beginning of main program label and ending at start of subroutines label) in Appendix B. Once the initialization of the data RAM and control registers (code beginning at start of additional tables label and ending at start of main program sequencer label) is complete, the main program loop is executed. The device remains in the WAIT loop (first four lines of code of main program sequencer routine) until the FR flag in the control register is raised. Control register bits 27-24 and 23-16 are set so the main program and data samples are transmitted/received to/from the TCM2919 codec at a rate of 9.6 KHz.

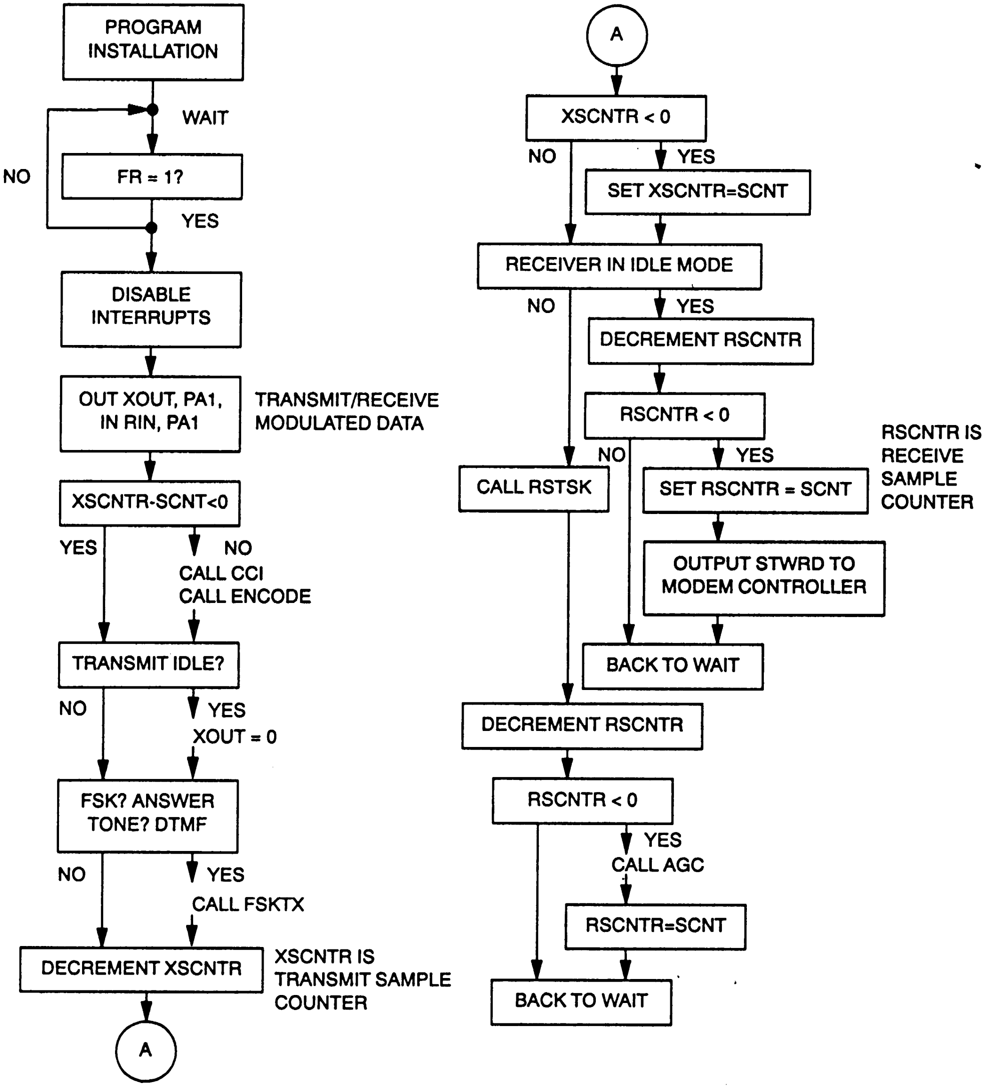


Figure 4. Flowchart of Main Program (Appendix B)

As the V.21/Bell 103 standard modems transmit data at 300 bps, a 9.6-KHz sampling rate results in 32 samples/baud. The 9.6-KHz sampling rate is very practical for several reasons:

- It is higher than the Nyquist sampling frequency of approximately 8 KHz for a telephone channel, and
- It is a convenient multiple of the popular modem transmission frequencies (300, 1200, and 2400 bps).

The TMS320C17 is clocked by a 18.432-MHz oscillator. To satisfy the 9.6-KHz sampling frequency, the number of instructions executed per sample must be less than 480. To implement the various functions required by the FSK modulator/demodulator, it is necessary to distribute the tasks among the various samples within the baud. The command control interpreter (CCI) is executed during the first sample of the baud, and the AGC routine is implemented during the final sample baud.

When the raised FR flag is detected, the processor exits the WAIT loop and executes the main program. Refer to [8], Sections 3.8 and 3.9 for additional details on the FR flag, interrupts, and serial port. Table 4 describes the variables that are referenced in the main program.

Table 4. Variables Referenced in Main Program Variable

Variable Name	Description
XSCNTR	The transmit counter; equals the number of samples that have been transmitted in the current baud.
SCNT	Number of samples in a baud, i.e., 9.6 KHZ/300 HZ = 32 samples/baud.
XOUT	Output sample sent via the TX serial port to the combo codec.
RIN	Input sample sent via RX serial port from the combo codec.
STATUS	An 8-bit number used internally by the DSP. Indicates present operating mode of the modem.
STWRD	8-bit status word sent to the modem controller by the DSP. See Table 5.
OAFLAG	Indicates if modem is in originate or answer mode. OAFLAG = 0 → originate mode.
DTFLAG	Indicates if the modem is transmitting DTMF tones. DTFLAG = 1 → transmitting DTMF data.

Table 5 shows the organization of STWRD (the DSP status word that is written to the microcontroller).

Table 5 STWRD - DSP Status Word Written to the Modem Controller

Bit No.	Description
7	Enable/disable automatic gain control. 0 = Enable 1 = Disable
6	EDT (in band energy) 0 = Not detected 1 = Detected
5	Reserved
4	Reserved
3	Received data bit (0,1)
2	Reserved, set to 1
1	Reserved, set to 1
0	Reserved, set to 1

When the program exits the wait loop, it disables all interrupts and reads a data sample RIN from the receive buffer or writes a data sample XOUT to the transmit buffer of serial port #1.

At the first sample of a baud, when XSCNTR = SCNT (=31), the program implements the command control interpreter (CCI) subroutine as shown in the following code. Note that SCNT = 31, and XSCNTR is initially set at 31 and decremented by 1 every sample. When XSCNTR equals 0, it is reset to 31, for a total of 32 samples.

```
        LAC     XSCNTR
        SUB     SCNT        ; ACCUM = XSCNTR-SCNT
        BLZ     SEQU        ; BRANCH TO SEQU IF ACCUM <0
        CALL    CCI
SEQU:   LACK    030h
```

The CCI subroutine reads the next 8-bit command from the modem controller (TMS70C42400A or equivalent), performs the required program control functions, and returns to the main program.

If the DSP is in transmit idle mode, the data sample XOUT is set to 0 and sent to serial port #1 transmit buffer.

If the DSP is not in transmit idle, the FSK transmit subroutine FSKTX is called. Depending on the present value of STATUS as determined by the modem controller and the CCI subroutine, the FSKTX subroutine will transmit FSK encoded data, DTMF tones, or an answer tone. Upon completing the FSKTX subroutine, the program decrements the transmit sample counter XSCNTR by 1 and checks to see if it is less than 0. If so, XSCNTR is reset to 31. Otherwise, the program proceeds without any further modifications to XSCNTR.

At this point, the main program checks to see if the receiver is in idle mode. If the receiver is in idle mode, the receive sample counter RSCNTR is decremented. If RSCNTR is not less than 0, the program returns to the WAIT loop. If RSCNTR is now less than 0, it is reset to 31, and the program then returns to the WAIT loop.

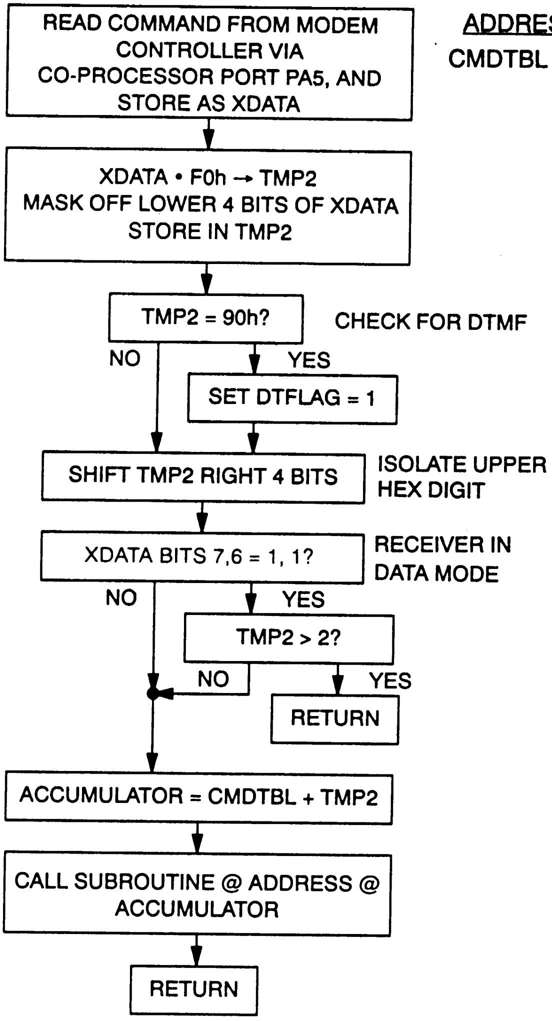
If the receiver is not in idle mode, the receiver decode/demodulation subroutine RSTSK (receiver per sample task) is called. This subroutine demodulates the receiver signal and estimates the value of the received data. When the subroutine is complete, the main program decrements RSCNTR and resets it to 31, if required.

After the RSTSK subroutine is complete, the program decrements RSCNTR. If RSCNTR is greater or equal to 0, the program returns to the wait loop. For the sample, when RSCNTR is less than 0, the automatic gain control subroutine (AGC) is called once per baud. The AGC subroutine monitors and compensates for any significant variation of the received signal level caused by telephone line fluctuations and other dynamic effects. RSCNTR is then RESET to 31, and the program returns to the WAIT loop.

The main program calls the following subroutines:

- CCI—Command control interpreter
- DTMF—DTMF setup
- FSKSET—Set up FSK transmit frequency
- FSKTX—Transmitter mode select
- OPER—Set operating mode
- PROTO—Protocol select
- RESET—Reset and equalizer enable
- RMODE—Receiver mode select
- RSTSK—FSK demodulation
- XMODE—Transmitter mode select

Figure 5 shows a block diagram of the CCI subroutine. The CCI reads the setup command from the modem controller (through the co-processor port PA5) and stores it in data RAM location XDATA (The structure of XDATA is shown in Table 3). The CCI then calls the appropriate subroutine to modify the system control bits (OAFLAG and DTFLAG) and status register (STATUS). The CCI, depending whether the modem configuring the DSP is in answer, originate, or transmit DTMF, loads the required nominal frequency values into TXFRQ and RXFRQ. Table 6 shows the organization of the STATUS register.



ADDRESS		SUBROUTINE	DESCRIPTION
CMDTBL +	0h	RESERVED	
	1h	RESERVED	
	2h	RESERVED	
	3h	RESET	
	4h	RESERVED	
	5h	FSKSET	FSK MODE
	6h	RESERVED	
	7h	RMODE	RECEIVED MODE SELECT
	8h	XMODE	TRANSMIT MODE SELECT
	9h	DTMF	DTMF TONES
	Ah	RESERVED	
	Bh	RESERVED	
	Ch	RESERVED	
	Dh	OPER	OPERATION SELECT
	Eh	RESERVED	
	Fh	PROTO	PROTOCOL SELECT

Figure 5. Flowchart of the CCI Sub

Table 6. The Status Register Organization

Bits	Description
7,6	Indicate Receiver Mode: 00 = Receiver in Idle Mode 01 = Call Progress Monitoring Mode 10 = Data Mode 11 = Reserved
5,4	Indicate Transmitter Mode: 00 = Transmitter in Idle Mode 01 = Transmit Answer Tone 10 = Data Mode 11 = Reserved
3	Answer/Originate Mode: 0 = Originate Mode 1 = Answer mode
2	CCITT/Bell Mode: 0 = CCITT (V.21) 1 = Bell (103)
1,0	Speed status: 00 = 300 BPS 01 = Reserved 10 = Reserved 11 = Reserved

The setup commands from the modem controller and subroutines called by the CCI subroutine are shown in Table 3.

The RESET subroutine loads 81h into the STWRD word that is sent to the modem controller via the co-processor port PA5. This advises the modem controller that the DSP has been reset. The DSP program then branches to START, and the DSP is reinitialized.

The FSKSET subroutine reads the XDATA word to determine if the next bit to be transmitted is 0 or 1 and then loads the appropriate 0 or 1 frequency F0ADD or F1ADD into the TXFRQ register.

When setup in answer mode, XDATA bits 3 and 2 are loaded into the STATUS register bits 7 and 6, respectively, by the RMODE subroutine. These bits determine what tasks the FSK receiver subroutine RSTSK will perform, as shown in Table 3 and Figure 5.

The XMODE subroutine reads XDATA bits 0 and 1. These bits determine what tasks the FSK transmitter subroutine FSKTX will perform as shown in Figure 4. If the transmit answer tone function is selected, bits 2 and 3 of XDATA indicate what the answer tone frequency will be:

XDATA Bits 3,2 = 0,0 2100 Hz
0,1 2225 Hz
1,0 Reserved
1,1 Reserved

The program loads the appropriate answer tone value into register TXFRQ. XMODE then loads XDATA bits 1 and 0 into STATUS bits 5 and 4, respectively. STATUS bits 5 and 4 determine what tasks the transmitter subroutine FSKTX will perform.

The DTMF subroutine determines what number or symbol needs to be transmitted by reading XDATA bits 3 through 0. DTMF then loads the appropriate high-frequency phase step, low-frequency phase step, high-frequency gain, and low-frequency gain into the RXFRQ, TXFRQ, DTMFH, and DTMFL registers, respectively, from the Table TONTBL.

The OPER subroutine checks bits 1 and 0 of XDATA. If bits 1 and 0 equal 0 and 1 bit 3 of STATUS is set to 1, indicating that the modem is in analog loopback mode. If bits 1 and 0 are not equal to 0 and 1, OPER returns without performing any operations.

The PROTO subroutine selects the mode and protocol of the DSP based on XDATA bits 3 through 0. PROTO first sets bits 1 and 0 of STATUS (indicating the modem data rate), based on the value of bits 1 and 0 of XDATA (see Figure 7).

While the software provided in Appendix B supports only a 300-bps data rate, it does provide the necessary hooks so that different standard modems (ie V.22, V.22bis) can easily be incorporated into the code.

Next, PROTO checks XDATA bits 3 and 2 to determine if the modem should be in originate/answer mode and Bell/CCITT mode.

Bit 3: 0 = Originate
1 = Answer
Bit 2: 0 = Bell
1 = CCITT

As shown in Table 2, the transmission frequencies of the Bell 103 and V.21 originate and answer modes are unique. PROTO loads registers used by the FSK transmitter subroutine (FSKTX) and the FSK receiver subroutine (RSTSK) with values stored in table TONTBL in data ROM and corresponding to transmit and receive frequencies.

PROTO then uses the XDATA bits 3 and 2 to determine which constants are transfered from table FSKTBL into addresses F1ADD (transmit 1 phase step), F0ADD (transmit 0 phase step), B1FSK (FSK delay filter coefficient), and GAIN (FSK mode gain). PROTO also loads addresses SCNT (baud counter=32), TRANS (FSK data transmition N=15), A1FSK (A1 demodulator filter coefficient), A2FSK (A2 demodulator filter coefficient), and DZONE (dead zone of window comparator) with the appropriate values.

If bit 3 of the STATUS word equals 1, the modem is set to analog loopback mode, and the modem should receive the information that it transmits. PROTO checks to see if bit 3 of STATUS equals 1; if so, the receiver parameters are modified to be the same band as the transmitter.

The FSK modulator is implemented in the FSKTX subroutine. Figure 6 is a block diagram of the FSKTX subroutine. The primary function of the FSK modulator is the following: Given a stream of binary data a_0, a_1, a_2, . . ., a_{k-1}, ak for each data element $a_k = \{0,1\}$, generate a corresponding signal of frequency f_0 or f_1 for the duration of a_k's baud period.

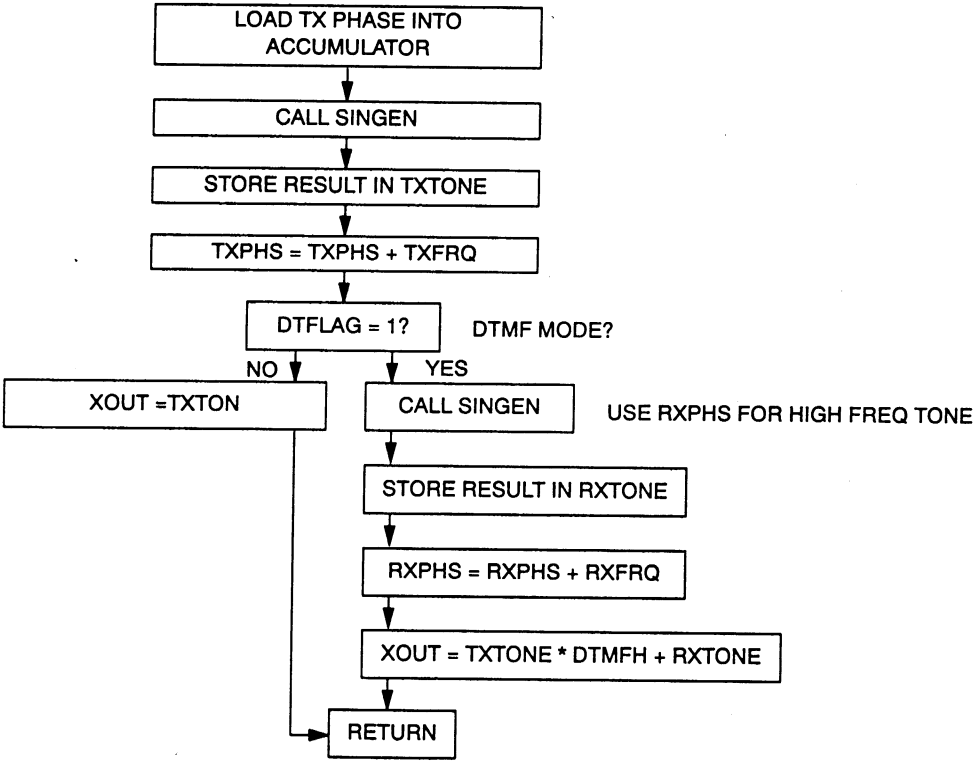


Figure 6. Flowchart of Subroutine FSKTX

Figure 7 shows a functional model of the FSK modulator. The TMS320 software implementation of the FSK modulator generates tones by stepping through a cosine table. The size of the phase step determines the output signal frequency. You should pay particular attention how phase angles, phase steps, cosines, and sines are represented as 16- and 32-bit integer numbers.

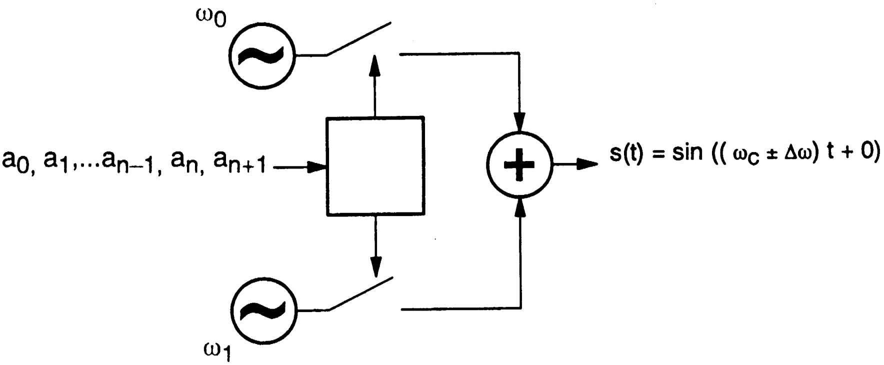


Figure 7. Functional Model of an FSK Modulator

Table 7 describes the significant variables used in the FSKTX subroutine.

Table 7. Variables Referenced in the FSK Transmitter Subroutine FSKTX

Variable Name	Description
TXPHS	Present value of the transmit signal phase. Also used as present phase of the low frequency DTMF tones.
TXFRQ	Phase step between consecutive TXPHS samples.
RXPHS	Normally used in the FSK demodulator subroutine RXTSK. Used as present phase for the high frequency DTMF tone.
RXFRQ	Normally used in RXTSK subroutine. Also used as phase step for high-frequency tone when transmitting DTMF tones.
DTMFL	Scaling factor for low-frequency DTMF tones.
DTMFH	Scaling factor for high-frequency DTMF tones.
SINGEN	A subroutine called by FSKTX. Given a 16-bit number representing an angle from 0 to Pi, the SINGEN routine determines the sine of the angle and stores the result at address TMP3.

The software FSK Modulation routine receives data at a rate of 300 bps and generates 12-bit, two's complement data samples at a rate of 9.6 KHz. The TMS320C17's on-chip hardware compander reduces the sample to 8 bits before it is sent to the Codec via the serial port.

The most recent phase of the output signal is stored in data memory location TXPHS, and the amplitude is read from the COSOFF table by the SINGEN subroutine. The frequency of the transmitted signal is determined by the size of the phase step TXFRQ between successive output samples:

$$\mathrm{TXPHS}[(N+1)T] = \mathrm{TXPHS}[NT] + \mathrm{TXFRQ}[NT]$$

The value of TXFRQ is determined by the FSKSET subroutine referenced by the CCI subroutine. Recall that, depending on the instruction received from the modem controller at the beginning of the baud, the CCI subroutine loaded data memory location TXPHS with either F0ADD or F1ADD. Table 8 shows the FSK frequencies and phase steps (TXFRQ) for the V.21 and Bell 103 modem standards.

Table 8. Frequencies and Phase steps for V.21 and Bell 103 Modems

Modem Standard		Frequency (Hz)	Phase Step @9.6 KHz	Phase Step TXFRQ, Q15 hex
V.21	Originate 1	980	0.2042*Pi	1A22h
	Originate 0	1180	0.2458*Pi	1F77h
	Answer 1	1650	0.3428*Pi	2C00h
	Answer 0	1850	0.3854*Pi	3155h
Bell 103	Originate 1	1270	0.2646*Pi	21DDh
	Originate 0	1070	0.2229*Pi	1C89h
	Answer 1	2225	0.4635*Pi	3B55h
	Answer 0	2025	0.4219*Pi	3600h

The magnitude of the phase step is determined by

$$[(\text{Desired Frequency})/(\text{Sampling Frequency})] * 2\pi$$

In the case of the originate 1 of the V.21 modem, the phase step equals

$$(1270/9600) * 2\pi = .2646\ \pi \text{ Radians}$$

Both TXPHS and TXFRQ data memory locations are 16-bit binary numbers in Q15 two's complement notation equal to

$$(\text{Output Signal Phase})/\pi.$$

Thus TXPHS hex values

$$\begin{aligned} 2000h &= \pi/4 \\ 4000h &= \pi/2 \\ 6000h &= 3\pi/4 \\ 8000h &= -\pi \\ A000h &= -3\pi/4 \end{aligned}$$

An advantage of this approach is that the phase of the output signal is continuous. This provides a higher spectral efficiency than that of a discontinuous phase FSK implementation.

The sine generation subroutine SINGEN subtracts $\pi/2$ (4000h) from TXPHS and uses this phase to read the amplitude from the COSOFF table. The symmetry of the cosine function has been used to reduce the table size from 513 to 257 elements, with data memory addresses COSOFF, COSOFF+128, and COSOFF+256 corresponding to 0, $\pi/2$, and π radians, repectively. To determine the cosine of an angle outside the 0-to-π range, the program utilizes the two's complement format of the data and the absolute value function ABS. As an example, assume that the present phase TXPHS is

$$\text{TXPHS(N)} = (-170/256) * \pi = -.6640625 * \pi = \text{A600h}$$

If we are transmitting a 1 in V.21 Originate mode, the phase step is

$$\text{TXFRQ} = .26448 * \pi = \text{21DDh}$$

The next value of:

$$\begin{aligned} \text{TXPHS(N+1)} &= \text{TXPHS(N)} + \text{TXFRQ} \\ &= -.6640625\ \pi + .26448\ \pi \\ &= -.3995825\ \pi \\ &= \text{A600h} + \text{21DDh} = \text{C7DDh} \end{aligned}$$

The subroutine then subtracts $\pi/2$ (4000h) from TXPHS, so the sine of angle TXPHS can be determined from the Cosine table:

$$\begin{aligned} \text{ANGLE} &= \text{TXPHS(N+1)} - \pi/2 \\ &= -.3995825\ \pi - .5\ \pi \\ &= -.8995825\ \pi \\ &= \text{C7DDh} - \text{4000h} = \text{87DDh} \end{aligned}$$

Note that TXRFQ is added to TXPHS(N), and $\pi/2$ is subtracted from TXPHS(N+1) with the sign extension suppressed, so TXPHS(N+1) = 87DDh. This represents 1.06143 π as an unsigned number or $-.93857\ \pi$ as a signed number. If we now consider TXPHS(N+1) a signed and take the absolute value:

$$\text{ABS[TXPHS]} = \text{ABS[87DDh]} = \text{7823h representing } .93857\ \pi$$

Note that:

$$\text{Cos}(1.06143\pi) = \text{Cos}(.93857\pi) = -.98144$$

The cosine table address is generated:

$$\text{COSSOFF} + (\text{7823h}/\text{80h}) = \text{COSOFF} + \text{F0h}$$

The value at Data Memory address COSOFF + F0h is

$$\text{Cos}((240/256)\pi) = -.980786 = \text{8276h, Q15 2's complement notation}$$

Within the limits of the cosine table precision, the calculated output value equals the value read from the table.

The structure of the FSK Demodulator is shown in Figure 8.

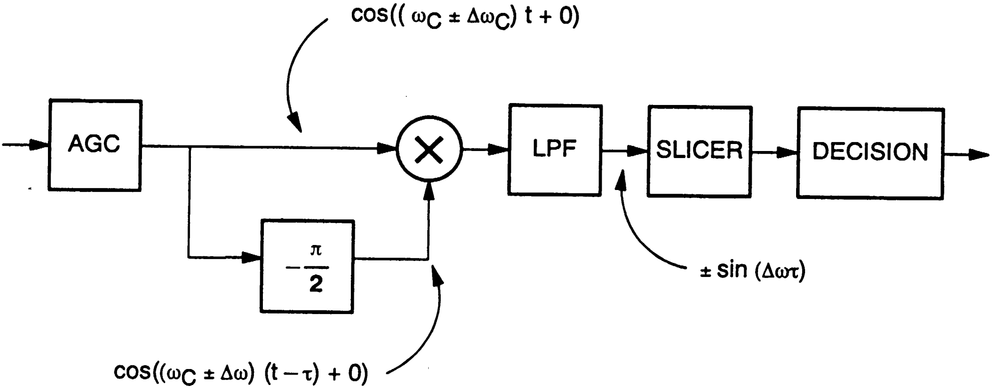


Figure 8. FSK Demodulator

The received FSK signal is sent to the DSP from the Codec via the serial port. The on-chip companding hardware expands the signal from an 8- to 13-bit value. The automatic gain control routine compensates for transient signal level variations and sends the amplitude adjusted received signal R(t) to the software demodulator.

$$R(t) = \cos[(\omega_c \pm \delta\omega) * t + \phi] \qquad (4)$$

As this is a binary FSK system, the frequency of this signal is either $\omega_c - \delta\omega$ or $\omega_c + \delta\omega$, depending on whether a 0 or 1 was sent. (Recall from the V.21 signal that $\delta\omega$ is less than 0.)

The received signal R(t) is multiplied by a delayed version of itself:

$$R(t - \tau) = \cos[(\omega_c \pm \delta\omega) * (t - \tau) + \phi] \qquad (5)$$

Where τ is the signal delay.

The product of the received signal (4) and delayed received signal (5) is

$$2 * \cos[(\omega_c \pm \delta\omega) * t + \phi] * \cos[(\omega_c \pm \delta\omega) * (t - \tau)] \tag{6}$$

$$= \cos[2(\omega_c \pm \delta\omega) * t - (\omega_c \pm \delta\omega) * \tau + 2 * \phi] + \cos[(\omega_c \pm \delta\omega) * \tau] \tag{7}$$

If $\omega_c\ \tau$ is set to equal $\pi/2$, and (7) is lowpass filtered to remove the double frequency component, the resulting signal is

$$\cos(\pi/2 \pm \delta\omega * \tau) = \sin(\pm \delta\omega\tau) = \pm \sin(\delta\omega) \tag{8}$$

If $\delta\omega$ is greater than 0, then the sign of the lowpass filter output will be positive or negative, depending on whether $\omega_c + \delta\omega$ or $\omega_c - \delta\omega$ is originally transmitted. If $\delta\omega$ is less than 0, obviously the opposite relationship is true. The sign of the lowpass filter output indicates the value of the received data.

The TMS320 software implementation of the 300-bps FSK Demodulator is found in Subroutine RSTSK, Subroutine CCITT, and Subroutine FDEM20 in Appendix B.

The AGC subroutine provides the RSTSK subroutine with a Q11 two's complement format received signal sample at a rate of 9.6 K samples per second.

As previously discussed, the data is extracted from the received signal by multiplying the received signal by a $\pi/2$ delayed version of itself, $\cos[(\omega_c \pm \delta\omega) * t + \phi - \pi/2 \pm \delta\omega * \tau]$. The product is then passed through a lowpass filter to remove the high frequency components.

If the desired phase delay is

$$\omega_c * \tau = \pi/2, \tag{9}$$

then

$$\tau = 1/(4 * f_c) \tag{10}$$

The sample rate is 9.6 KHz, or a period T = 104.167 μs. Table 9 shows the carrier frequencies, for both the V.21 and Bell 103 standards, the time delays corresponding to a $\pi/2$ phase delay and the equivalent number of 9.6-Khz samples. Note that none of the delays are exact multiples of the 9.6-KHz sampling period; each delay has an integer and fractional part.

Table 9. Carrier Frequency and Time Delays

Modem Standard		Frequency (Hz)	$\tau(\mu s)$	# of 9.6-KHz Samples
V.21	Originate	1080	231.481	2.2222
	Answer	1750	142.857	1.3714
Bell 103	Originate	1170	213.675	2.0513
	Answer	2125	117.647	1.1294

To minimize the probability of error, it is necessary that the phase delay be as close to $\pi/2$ as possible. An accurate estimate of the fractional part of the delay must be total phase delay. This is achieved by using a single zero FIR filter.

$$R((n - \alpha)T) = GAIN * [R(nT) + B1FSK * R((n-1)T)] \qquad (11)$$

where $R(nT)$ is the nth sample of the received signal $R(t)$
$R((n - \alpha)T)$ is the estimate of the fractionally delayed signal
n is an integer
α is the desired fractional delay , $0< \alpha <1$

The filter coefficient B1FSK and GAIN for the fractional delay filter of each V.21 and Bell 103 carrier are shown in Table 10. The derivation of the gain and filter coefficients are shown in Appendix A.

Table 10. Time Delay and FIR Filter Coefficients

Modem Standard		Frequency	Fractional Delay 9.6-KHz Sample(__)	Gain	B1FSK
V.21	Originate	1080	.2222	.69753	.32796
	Answer	1750	.3714	1.00000	.68889
Bell 103	Originate	1170	.0518	.57731	.07175
	Answer	2125	.1294	1.00000	.31678

B1 and GAIN are stored in data memory locations B1FSK and GAIN, resepectively. The actual implementation is

$$PDEL1 = AGCOUT + B1FSK * PDEL0$$

where AGCOUT is the received signal after the signal level has been compensated by the automatic gain control routine.

$$
\begin{aligned}
\text{AGCOUT} &= \cos[(\omega_c \pm \delta\omega) * nT + \phi\ ] \\
\text{PDEL0} &= \cos[(\omega_c \pm \delta\omega) * (n-1)T + \phi\ ] \\
\text{PDEL1} &= \cos[(\omega_c \pm \delta\omega) * (n-1-\alpha)T + \phi\ ],\ 0< \alpha <1 \\
\text{PDEL2} &= \cos[(\omega_c \pm \delta\omega) * (n-2-\alpha)T + \phi\ ]
\end{aligned}
$$

Since AGCOUT, PDEL0, PDEL1, and PDEL2 are consecutive data memory locations, the integer multiples of the 9.6-KHz sample delays are easily achieved by using the data move (DMOV) instruction. PDEL1 is calculated after the demodulator product operation and is not used until the next sample period, a delay of one sample period.

For the low-frequency carriers of the V.21 and Bell 103 standards, a second delay is required and is implemented as DMOV PDEL1, moving the contents of PDEL1 into data memory PDEL2.

When the sample delayed signal (PDEL1 or PDEL2 for the high- or low-frequency carriers, respectively) is generated, it is multiplied by the most recent sample AGCOUT. The product of the multiply is stored in data memory location PROD. PROD is multiplied by GAIN and then filtered by a second-order direct-form, lowpass IIR filter, and the result is stored in location LPFOUT. Further information on digital filters can be found in [12], [13].

Given the lowpass filter output LPFOUT, the FSK demodulator must now estimate the value of the received signal.

In the Data Estimation routine, the following memory location addresses are called:

BDATA — The data estimation for the previous baud.

FSKDAT — Data estimation of the current sample.

BAUDCK — A record of the number of samples presently taken in the current baud. Recall that the sample rate is 9.6 KHz and the baud rate is 300 Hz; so there are 32 samples/baud.

COUNTR — The data estimations of each sample in the current baud are compared to the decision of the previous baud. If these are different, then COUNTR is incremented. If COUNTR reaches 32 before BAUDCK reaches 32, it is assumed that a data transition has occurred, and BDATA is set to the opposite value:

$$\text{BDATA}(N+1) = \text{ABS}[\text{BDATA}(N) - 1]$$

Figure 9 is a flowchart of the data decision source code implementation.

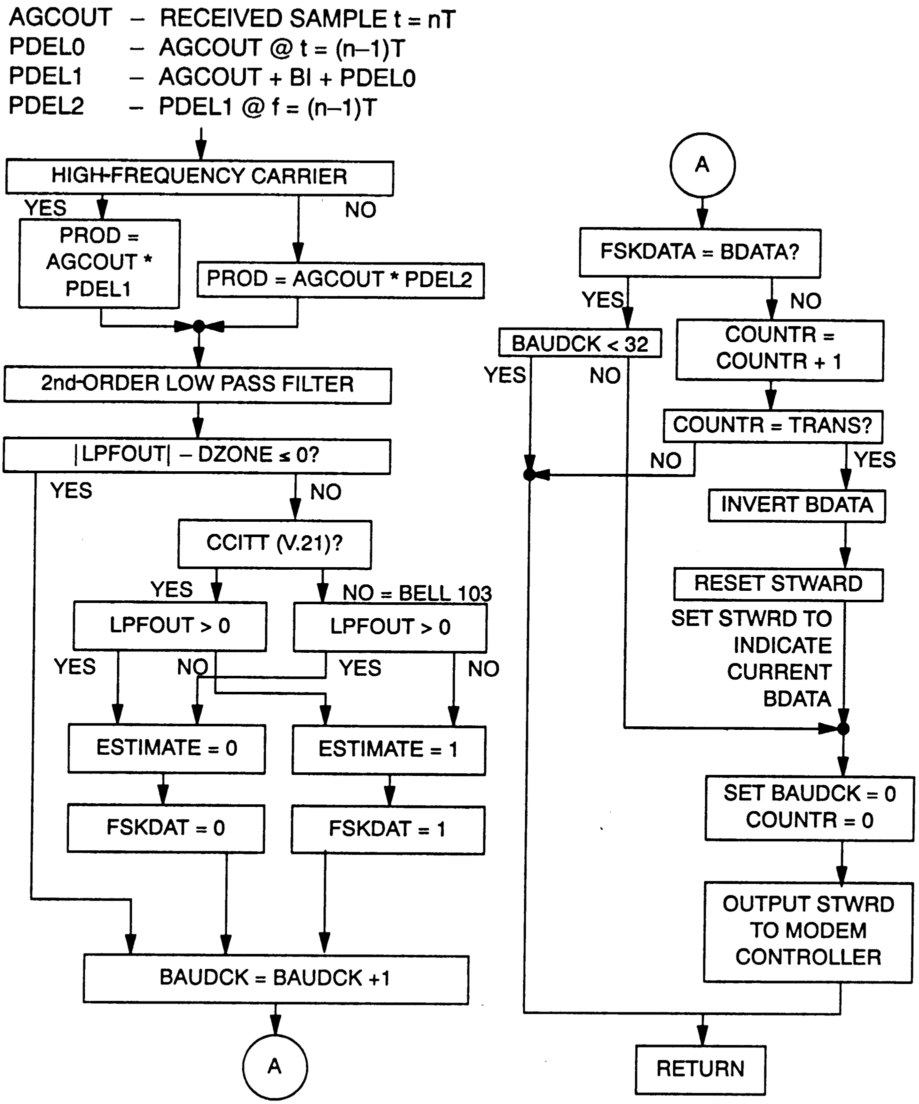


Figure 9. Data Decision Algorithm Flowchart

The function of the automatic gain control subroutine AGC is to compensate for amplitude distortions introduced by the telephone system, etc. References [5], [14] provide additional information on AGC.

Incorporating Additional Functions into the DSP

One of the important tasks the designer faces is incorporating value-added software functions into the DSP source code found in Appendix B.

The software presented here uses only 1.1 Kwords of the 4 Kwords of maskable ROM available on the TMS320C17. This provides you with a significant amount of code space to implement value-added functions.

This software offers a number of hooks that facilitate the easy inclusion of additional software. Note in Table 3 (Modem Controller Commands for the DSP), that the following commands are presently reserved : E, C, B, A, 6, 4, 2, 1, and 0. Each of these commands have bits 0 through 3 undefined. All of these commands can be used by the designer to call additional functions.

You must ensure that the correct modifications are made to the modem controller and modem DSP software. The DSP control command interpreter (CCI) must be modified to recognize and respond to the new commands. The additional functions should be implemented in either a new or the appropriate existing subroutine. The option indicating to the main program that the new subroutine should be called, needs to be provided. This can be done using the STATUS register, or you can define a new register.

You must also ensure that the XDATA word will indicate the present status of the DSP to the modem controller. There are presently a number of unused bits in the XDATA word, so incorporating the modifications in the DSP is straightforward.

Finally, you must ensure that the additional software functions do not exceed the timing requirements imposed by the 9600-KHz sampling frequency.

Conclusions

This application report presented you with the information required to implement a 300-bps V.21/Bell 103 FSK modem based on a TMS320C17 Digital Signal Processor. Both hardware and software issues were discussed. A summary of the FSK modulation and demodulation algorithms and a basic review of modems were also provided.A discussion about incorporation of additional functions and software into the code provided concluded this report.

Appendix A is a derivation of the FSK demodulator fractional delay filter coefficients. Appendix B is the TMS320C17 source code listing.

Acknowledgements

The author wishes to acknowledge the contribution of Dr. Amin Haoni of Technekron, Inc., and George Troullinos, and Raj Chirayil of Texas Instruments. This report is based on their work.

Glossary of Symbols and Abbreviations

bps — Bits per second
FSK — Frequency shift keying
ω_c — Carrier signal angular velocity
$\delta\omega$ — Modulation shift of angular velocity
t — Time
ϕ — Phase shift
ω_0 — Angular velocity transmitted to indicate a 0
ω_1 — Angular velocity transmitted to indicate a 1
τ — The amount of time the received signal is delayed in the FSK demodulator
f_0 — Frequency transmitted to indicate a 0
f_1 — Fequency transmitted to indicate a 1
f_c — Carrier frequency
α — Sample fractional delay created by the single FIR filter

References

[1] *DSP2400 Modem User's Guide*, Texas Instruments Inc. (1988).

[2] "TMS320A2400A Modem Digital Signal Processor Data Sheet", Texas Instruments Inc., (1988).

[3] "TMS70A2400A Modem Controller Data Sheet", Texas Instruments Inc., (1988).

[4] Lee, E.A., and Messerschmitt, D.G., "Digital Communications", Kluwer Academic Publishers (1988).

[5] Bingham, J.A.C., "The Theory and Practice of Modem Design", John Wiley and Sons, (1988).

[6] Proakis, J.G., "Digital Communications", McGraw-Hill (1983).

[7] Troullinos, G., et al., "Theory and Implementation of a Splitband Modem Using the TMS32010" (document number SPRA013), Texas Instruments Inc. (1986).

[8] "TCM29C18,'C19 PCM Codec Data Sheet" (document number SCTS021), Texas Instruments Inc., (1987).

[9] "SC11005 Splitband Filter Data Sheet", Sierra Semiconductors (1986).

[10] *First-Generation TMS320 User's Guide* (document number SPRA013A), Texas Instruments Inc., (1988).

[11] "First-Generation Digital Signal Processors Data Sheet" (document number SPRS009), Texas Instruments Inc., (1987).

[12] "Digital Signal Processing Applications with the TMS320 Family" (document number SPRA012A), pp 27-69, Texas Instruments Inc., (1986).

[13] Parks,T.W., and Burrus, C.S., "Digital Filter Design", John Wiley and Sons Inc. (1987).

[14] Lovrich, A., Troullinos, G., Chirayil, R. "An All Digital Automatic Gain Control", Texas Instruments Inc., *ICASSP Conference Proceedings*, (1988).

Appendix A

Calculation of Phase Delay Filter Coefficients

A key element of the FSK demodulator implementation is the $\pi/2$ phase delay of the carrier signal. The effectiveness of the demodulator is highly dependent on the accuracy of the $\pi/2$ phase delay.

In a digital system, it is highly unlikely that the time delay required for the $\pi/2$ phase delay is an exact multiple of the signal sampling period. It will be necessary to introduce phase delays that are a fraction of the sampling period.

To accurately generate the fractional delay, the digital signal processor uses a single zero FIR filter. This appendix derives the coefficients for the single zero FIR.

Given the one zero FIR filter shown in Figure A-1:

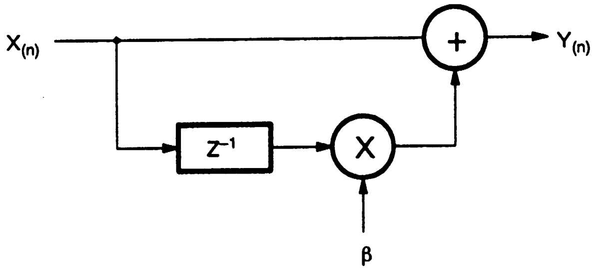


Figure A-1. One Zero FIR Filter.

$$Y(n) = X(n) + \beta X(n-1) \tag{A1}$$

therefore

$$\begin{aligned} Y(z) &= X(z) + \beta * z^{-1} X(z) \\ &= X(z) * (1 + \beta z^{-1}) \end{aligned} \tag{A2}$$

The transform of the filter is F(z)

$$F(z) = Y(z)/X(z) = (1 + \beta z^{-1}) \tag{A3}$$

The purpose of this filter is to introduce a precise group delay τ (delay of the signal envelope) to the received signal τ. is defined as

$$\tau = \frac{-d\theta(\omega)}{d\omega} \overset{\Delta}{=} \text{group delay} \tag{A4}$$

Evaluate F(z) at $z = e^{j\omega}$ to obtain the frequency response.

$$F'(\omega) = F(e^{j\omega}) = 1 + \beta\, e^{-j\omega} \tag{A5}$$

$$F'(\omega) = R(\omega) + jI(\omega) = A(\omega)e^{j\phi(\omega)} \tag{A6}$$

Where $R(\omega)$, $I(\omega)$, $A(\omega)$, and $\phi\,(\omega)$ are real functions of ω.

$$A(\omega) = |F'(\omega)| = [R(\omega)^2 + I(\omega)^2]^{1/2} \tag{A7}$$

and

$$\phi\,(\omega) = \arctan\,(I(\omega)/R(\omega)) \tag{A8}$$

Given

$$e^{-j\omega} = \cos\omega - j\sin\omega \tag{A9}$$

Substituting (A9) into (A5)

$$F'(\omega) = 1 + \beta\,\cos\omega - j\beta\sin\omega \tag{A10}$$

From (A6), (A8), and (A10)

$$\phi(\omega) = \left(\arctan\,\frac{-\beta\sin\omega}{1 + \beta\,\cos\omega}\right) \tag{A11}$$

Substituting (A11) into (A5)

$$\frac{-d\,\phi(\omega)}{d} \qquad \frac{-d}{d\omega}\Big(\qquad \Big(\frac{-\beta\sin\omega}{1 + \beta\cos\omega}\Big)\Big) \tag{A12}$$

now

$$\frac{d}{dx}\,(\arctan\,(u)) = \frac{1}{1 + u^2} * \frac{du}{dx} \tag{A13}$$

therefore

$$\tau = \frac{-1}{1 + \left(\dfrac{-\beta\sin\omega}{1 + \beta\cos\omega}\right)^2} * \frac{d}{d\omega}\left(\frac{-\beta\sin\omega}{1 + \beta\cos\omega}\right) \tag{A14}$$

$$= \left(\frac{-(1 + \beta\cos\omega)^2}{1 + \beta^2 + 2\beta\cos\omega}\right) * \left(\frac{-\beta\cos\omega - \beta}{(1 + \beta\cos\omega)^2}\right) \tag{A15}$$

$$= \frac{+\,\beta\,(\beta + \cos\omega)}{1 + \beta^2 + 2\,\beta\cos\omega}$$

Assuming τ is expressed in terms of sample delays D

$$D = \frac{+\ \beta\,(\beta + \cos\omega)}{(1 + \beta^2 + 2\,\beta\cos\omega)} \tag{A17}$$

Rearranging (A17) and using the quadratic equation to solve for

$$\beta = -\ \frac{(1-2D)\cos\omega \pm ((1-2D)^2\cos^2\omega + 4D(1-D))^{1/2}}{2(1-D)} \tag{A18}$$

Given the desired group delay D, and the frequency $f=\omega/2\pi$ (where $\omega=(1080/9600)2\pi$), the filter coefficient β can be determined using equation (A18).

Appendix B

```
#######################################################################
*
*    APPENDIX B
*
#######################################################################
*
*    V21/BELL 103 MODEM
*
*    ASSEMBLY LANGUAGE SOFTWARE FOR TMS320C17 IMPLEMENTATION.
*
*    ALL RIGHTS RESERVED BY TEXAS INSTRUMENTS  (C)
#######################################################################
*
*    VERSION 1.0        01/SEP/89
*
#######################################################################
*
        .option   x
*
#######################################################################
*
*    CONSTANT DEFINITIONS
*
#######################################################################
*
RMMSK:    .set     0C0h
ANMSK:    .set     08h
LMMSK:    .set     0F3h
LDMSK:    .set     04h
NTXMSK:   .set     0CFh
NRCMSK:   .set     03Fh
NSPMSK:   .set     0FCh
TXSH:     .set     04h
RCSH:     .set     06h
*
#######################################################################
*
*    AGC EQUATES
*
#######################################################################
*
AGCREF:   .set     05B6h
*
#######################################################################
*
*    HIGH PASS FILTER CONSTANT
*
#######################################################################
*
TAU:     .set    14                 ; *16 - 14
*
```

```
#######################################################################
*
*    DATA MEMORY (RAM) ASSIGNMENTS
*
#######################################################################
*
STATUS:   .set     0
XDATA:    .set     1
DTFLAG:   .set     2                 ; DTMF FLAG
ONE:      .set     3                 ; CONSTANT 1
S1CNT:    .set     16                ; S1 DETECTION COUNTER
STWRD:    .set     ONE+1
XSCNTR:   .set     STWRD+1
RSCNTR:   .set     XSCNTR+1          ; REC. BAUD COUNTER
SCNT:     .set     RSCNTR+1          ; NOMINAL BAUD COUNTER
XBITS:    .set     SCNT+1
XOUT:     .set     XBITS+1           ; OUTPUT SAMPLE
RIN:      .set     XOUT+1            ; INPUT SAMPLE
TXFRQ:    .set     RIN+1
TXPHS:    .set     TXFRQ+1
RXFRQ:    .set     TXPHS+1
RXPHS:    .set     RXFRQ+1           ; RECEIVE DEMODULATION ANGLE (KH)
TMP4:     .set     RXPHS+1
TMP5:     .set     TMP4+1
*
TMP2:     .set     TMP5+1            ; PARTIAL FILTERED SIGNAL w(n)
TMP0:     .set     TMP2+1
*
TMP3:     .set     TMP0+1
TMP1:     .set     TMP3+1
*
IRCNT:    .set     TMP1+1
TIMIND:   .set     IRCNT+1
*
IRO:      .set     TIMIND+1
*
#######################################################################
*
*    IN THE FSK MODE, THE TX/RX PARAMETERS SHARE THE SAME MEMORY AS THE
*    EQUALIZER DELAY LINE
*
#######################################################################
*
OD:       .set     IRO               ; OUTPUT OF PRODUCT DEMODULATOR
AGCOUT:   .set     PROD+1
PDEL0:    .set     AGCOUT+1          ; PRODUCT DEMODULATOR DELAY LINE
PDEL1:    .set     PDEL0+1
PDEL2:    .set     PDEL1+1
LPDEL0:   .set     PDEL2+1           ; FSK LOWPASS DEMOD DELAY LINE
LPDEL1:   .set     LPDEL0+1
LPDEL2:   .set     LPDEL1+1
GAIN:     .set     LPDEL2+1          ; GAIN OF FSK DEMOD FILTER (0.5 OR 1.0)
FSKDAT:   .set     GAIN+1            ; OUTPUT OF FSK SLICER (X111)
BAUDCK:   .set     FSKDAT+1          ; BAUD CLOCK FOR FSK TIMING RECOVERY
```

```
BDATA:    .set      BAUDCK+1        ; CURRENT BAUD FSK DATA (X111)
TRANS:    .set      BDATA+1         ; = 20 USED IN TIMING DECISION
COUNTR:   .set      TRANS+1         ; TRANSITION COUNTER FOR RX TIMING
B1FSK:    .set      COUNTR+1        ; COEF B1 OF PHASE ADJUST FIR
A1FSK:    .set      B1FSK+1         ; COEF A1 OF FSK DEMOD FILTER
A2FSK:    .set      A1FSK+1         ; COEF A2 OF FSK DEMOD FILTER
F0ADD:    .set      A2FSK+1         ; ADDRESS OF 0 FREQUENCY
F1ADD:    .set      F0ADD+1         ; ADDRESS OF 1 FREQUENCY
FSKFLG:   .set      F1ADD+1         ; FLAG TO INDICATE FSK OPERATION
OAFLAG:   .set      FSKFLG+1        ; ORIGINATE_/ANSWER MODE FLAG
DZONE:    .set      OAFLAG+1        ; DEAD ZONE OF SLICER
LPFOUT:   .set      DZONE+1
CCITT:    .set      LPFOUT+1
*
******************************************************************************
*
*    AGC RAM
*
******************************************************************************
*
ALPHA:    .set      CCITT+1
AVESQR:   .set      ALPHA+1
GN:       .set      AVESQR+1
HYST:     .set      GN+1
*
******************************************************************************
*
*    BAUD COUNTER
*
******************************************************************************
*
BDCNTR:   .set      HYST+1
*
******************************************************************************
*
*    PAGE 1 RAM ASSIGNMENTS
*
******************************************************************************
*
X1:       .set      0
X2:       .set      X1+1
ST:       .set      X2+1
STLSB:    .set      ST+1
POSSM:    .set      STLSB+1
NEGSM:    .set      POSSM+1
*
```

```
******************************************************************************
*
*    DIAGNOSTICS!
*
******************************************************************************
*
DTMFL:    .set      IRCNT
DTMFH:    .set      TIMIND
*
          .text
          B         START
*
******************************************************************************
*
*    COEFFICIENTS STORED IN PROGRAM ROM
*
******************************************************************************
*    PHASE ANGLE LOOK-UP TABLE
*
*    CONTAINS THE INCREMENT USED AS XDELTA IN THE CARRIER GENERATION. THE
*    TABLE GIVES THE ANGLES FOR 300 BPS, (V21, BELL 103) ORIG/ANSW MODES.
*
*    NOTE: INCREMENTS ARE IN UNITS OF PI/128 TIMES 256 (UPPER 8 BITS OF DATA
*          EVENTUALLY REPRESENT TABLE INDEX)
*
******************************************************************************
*
FSKTBL:   .set      $
          .word     021ddh          ; FSK, 103, ORIGINATE, 1  1270 HZ
          .word     01c89h          ; FSK, 103, ORIGINATE, 0  1070 HZ
          .word     01852h          ; COEFF B1 FOR 2125 HZ FREQ.(0.4293)
          .word     07fffh          ; GAIN FOR FSK DEMOD LPF (1)
*
          .word     01a22h          ; FSK, V.21, ORIGINATE, 1  1180 HZ
          .word     01f77h          ; FSK, V.21, ORIGINATE, 0  980 HZ
          .word     22574h          ; COEFF B1 FOR 1750 HZ FREQ.(0.63)
          .word     07fffh          ; GAIN FOR FSK DEMOD LPF (0.5)
*
          .word     03b55h          ; FSK, 103, ANSWER, 1  2225 HZ
          .word     03600h          ; FSK, 103, ANSWER, 0  2025 HZ
          .word     0666h           ; COEFF B1 FOR 1170 HZ FREQ.(0.3891)
          .word     049e5h          ; GAIN FOR FSK DEMOD LPF (0.8323)
*
          .word     02c00h          ; FSK, V.21, ANSWER, 1  1850 HZ
          .word     03155h          ; FSK, V.21, ANSWER, 0  1650 HZ
          .word     10747           ; COEFF B1 FOR 1080 HZ FREQ.(0.3)
          .word     22857           ; GAIN FOR FSK DEMOD LPF (0.5)
*
F21:      .word     03800h          ; 2100 HZ ANSWER TONE
F22:      .word     03b55h          ; 2225 HZ ANSWER TONE
ZONE:     .word     50              ; DEAD ZONE OF FSK DEMOD SLICER
FSKA1:    .word     04989h          ; COEF A1 OF FSK DEMOD FILTER
FSKA2:    .word     0aadbh          ; COEF A2 OF FSK DEMOD FILTER
*
```

```
***********************************************************************
*
*    COMMAND MODE SUBROUTINE LOOK-UP TABLE
*
*    IN COMMAND MODE, EACH COMMAND BYTE CORRESPONDS TO AN ENTRY HERE WHICH
*    CONTAINS THE NAME OF THE APPROPRIATE SUBROUTINE TO CALL TO EXECUTE THE
*    COMMAND. THE NUMBER OF UNDEFINED SUBROUTINES (AT ADDRESSES 00, 01, 02,
*    04, 06, 0A, 0B, 0C, AND 0Eh) MEANS INCREASED SYSTEM EXPANSION IS EASILY
*    ACCOMODATED.
*
***********************************************************************
*
CMDTBL:  .set    $
         .word   NONE           ; 00h NOT DEFINED
         .word   NONE           ; 1Xh NOT DEFINED
         .word   NONE           ; 2Xh NOT DEFINED
         .word   RESET          ; 3Xh  RESET THE DSP
         .word   NONE           ; 4Xh NOT DEFINED
         .word   FSKSET         ; 5Xh  FSK DATA MODE
         .word   NONE           ; 6Xh NOT DEFINED
         .word   RMODE          ; 7Xh  RECVR. MODE SETUP
         .word   XMODE          ; 8hX  TRANS. MODE SETUP
         .word   DTMF           ; 9Xh  TRANSMIT DTMF CHARACTER
         .word   NONE           ; AXh NOT DEFINED
         .word   NONE           ; BXh NOT DEFINED
         .word   NONE           ; CXh NOT DEFINED
         .word   OPER           ; DXh  SET OPERATING MODE
         .word   NONE           ; EXh NOT DEFINED
         .word   PROTO          ; FXh  PROTOCOL SELECT
*
***********************************************************************
*
*    RECEIVER SUBROUTINE TABLE
*
*    USED TO SELECT DISTRIBUTED RECEIVER TASKS. THIS IS IMPORTANT IN HIGHER
*    FUNCTIONALITY SYSTEMS THAT CANNOT IMPLEMENT TASKS IN A SINGLE SAMPLE
*    PERIOD.
*
***********************************************************************
*
RSEQTB:  .set    $
         .word   NONE
         .word   NONE
         .word   NONE
         .word   NONE
         .word   NONE
         .word   NONE
         .word   NONE
         .word   NONE
         .word   NONE
         .word   RTSK10
         .word   NONE
         .word   NONE
         .word   NONE
         .word   NONE
         .word   NONE
*
***********************************************************************
*
*    SERIAL PORT CONTROL REGISTER DATA
*
***********************************************************************
*
DA1:     .word   0390Fh
DA2:     .word   02CBEh          ; MODIFIED TO USE H/W COMPANDING
DA3:     .word   0380Fh
*
***********************************************************************
*
*    AGC DATA
*
*    THE FOLLOWING VALUES ARE THE CALCULATED VALUES WITH NO WINDOW
*    IMPLEMENTATION AROUND THE THEORETICAL ENERGY BAND OF QAM SIGNALS.
*    HOWEVER, DUE TO ISI EFFECTS, WINDOWS ARE REQUIRED. FOR PROPER OPERATION,
*    THESE VALUES MUST BE ADJUSTED FOR APPROXIMATELY 3 DB DIFFERENCE IN SIGNAL
*    ENERGY LEVELS TO COMPENSATE FOR THE WINDOWS.
*
***********************************************************************
*
THRES1   .word   23BBh           ; -48.0 DBM REC. LEV. (A=35.73)
THRES2   .word   1548h           ; -43.5 DBM REC. LEV. (A=21.28)
THRES3   .word   240h            ; -24 DBM REC. LEV. (AFE=ON,A=2.25)
THRES4   .word   8FBh            ; -24 DBM REC. LEV. (AFE=OFF,A=8.98)
THRES5   .word   1417h           ; -31 DBM REC. LEV. (AFE=OFF,A=20.09)
THRES6   .word   50Dh            ; -31 DBM REC. LEV. (AFE=ON,A=5.05)
*
MAXALP:  .word   038A1h          ; -52.0 DBM RECEIVE LEVEL
PSM:     .word   01EA6h          ; BAUD ENERGY ERROR LEVEL 10628
NSM:     .word   0113h           ; BAUD ENERGY ERROR LEVEL 1
THRES1:  .word   0194Ch          ; -48.0 DBM REC. LEV. (A=25.30)
THRES2:  .word   0FF5h           ; -43.5 DBM REC. LEV. (A=15.96)
THRES3:  .word   0199h           ; -24 DBM REC. LEV. (AFE=ON,A=1.60)
THRES4:  .word   06B0h           ; -24 DBM REC. LEV. (AFE=OFF,A=6.69)
THRES5:  .word   0E00h           ; -31 DBM REC. LEV. (AFE=OFF,A=14.00)
THRES6:  .word   0490h           ; -31 DBM REC. LEV. (AFE=ON,A=4.56)
HYSINC:  .word   0F0Fh           ; 1/17 OF 65536
*
```

```
******************************************************************************
*
*   DTMF TONE TABLE:
*
*                       FIRST ENTRY REPRESENTS LOW FREQUENCY
*                       SECOND ENTRY REPRESENTS HIGH FREQUENCY
*
*   DELTA = (F / F ) * N
*                 S
*
*   WITH N = 256  TABLE SIZE
*        F = 9600 HZ
*         S
*        F = FREQUENCY OF INTEREST
*
******************************************************************************
*
*   DATA FORMAT IS S7.8 TO BE AS STEP SIZE. THE TABLE ENTRIES ARE HOWEVER,
*   TREATED AS 16 BIT UNSIGNED INTEGERS. A MULTIPLICATION OF DELTA BY 256
*   DOES THE NECESSARY CONVERSION IN FORMATS.
*
******************************************************************************
*
TONTBL:  .word    01918h          ; 0 LOW FREQ
         .word    023A0h
         .word    02A0h           ; LOW FREQ GAIN
         .word    03A0h           ; HI FREQ GAIN
*
         .word    01296h          ; 1
         .word    0203Eh
         .word    0800h           ; LOW FREQ GAIN
         .word    0493h           ; HI FREQ GAIN
*
         .word    01296h          ; 2
         .word    023A0h
         .word    0800h           ; LOW FREQ GAIN
         .word    0493h           ; HI FREQ GAIN
*
         .word    01296h          ; 3
         .word    02762h
         .word    08A0h           ; LOW FREQ GAIN
         .word    0493h           ; HI  FREQ GAIN
*
         .word    01488h          ; 4
         .word    0203Eh
         .word    02E0h           ; LOW FREQ GAIN
         .word    03A0h           ; HI FREQ GAIN
*
         .word    01488h          ; 5
         .word    023A0h
         .word    0340h           ; LOW FREQ GAIN
         .word    0420h           ; HI FREQ GAIN
*
         .word    01488h          ; 6
```

```
         .word    02762h
         .word    0340h           ; LOW FREQ GAIN
         .word    03A0h           ; HI FREQ GAIN
*
         .word    016B8h          ; 7
         .word    0203Eh
         .word    02A0h           ; LOW FREQ GAIN
         .word    0370h           ; HI FREQ GAIN
*
         .word    016B8h          ; 8
         .word    023A0h
         .word    02A0h           ; LOW FREQ GAIN
         .word    03D0h           ; HI FREQ GAIN
*
         .word    016B8h          ; 9
         .word    02762h
         .word    0290h           ; LOW FREQ GAIN
         .word    0300h           ; HI FREQ GAIN
*
         .word    01296h          ; A
         .word    02B8Ch
         .word    0493h           ; LOW FREQ GAIN
         .word    0493h           ; HI FREQ GAIN
*
         .word    01488h          ; B
         .word    02B8Ch
         .word    0493h           ; LOW FREQ GAIN
         .word    0493h           ; HI FREQ GAIN
*
         .word    016B8h          ; C
         .word    02B8Ch
         .word    0493h           ; LOW FREQ GAIN
         .word    0493h           ; HI FREQ GAIN
*
         .word    01918h          ; D
         .word    02B8Ch
         .word    0493h           ; LOW FREQ GAIN
         .word    0493h           ; HI FREQ GAIN
*
         .word    01918h          ; E (*)
         .word    0203Eh
         .word    0300h           ; LOW FREQ GAIN
         .word    0300h           ; HI FREQ GAIN
*
         .word    01918h          ; F (#)
         .word    02762h
         .word    0300h           ; LOW FREQ GAIN
         .word    0300h           ; HI FREQ GAIN
*
```

```
*******************************************************************
*
*    ADDITIONAL TABLES
*
*******************************************************************
*
        .copy     "COSTBL.A00"   ; COSINE FUNCTION TABLE
*
*******************************************************************
*
*    COSINE LOOKUP TABLE:
*
*    257 ENTRIES OVER THE RANGE [0,PI]. THE RESOLUTION OF THE TABLE IS
*    THEREFORE:
*
*    (180 / 256 ) = 0.703125 DEGREES
*
*******************************************************************
*
COSOFF:  .set     $             ; COSINE TABLE LENGTH = 512
         .word    07FFFh        ; ANGLE =   0.0000  COSINE = 1.000000
         .word    07FFEh        ; ANGLE =   0.7031  COSINE = 0.999925
         .word    07FF6h        ; ANGLE =   1.4063  COSINE = 0.999699
         .word    07FEAh        ; ANGLE =   2.1094  COSINE = 0.999322
         .word    07FD9h        ; ANGLE =   2.8125  COSINE = 0.998795
         .word    07FC2h        ; ANGLE =   3.5156  COSINE = 0.998118
         .word    07FA7h        ; ANGLE =   4.2188  COSINE = 0.997291
         .word    07F87h        ; ANGLE =   4.9219  COSINE = 0.996313
         .word    07F62h        ; ANGLE =   5.6250  COSINE = 0.995185
         .word    07F38h        ; ANGLE =   6.3281  COSINE = 0.993907
         .word    07F0Ah        ; ANGLE =   7.0313  COSINE = 0.992480
         .word    07ED6h        ; ANGLE =   7.7344  COSINE = 0.990903
         .word    07E9Dh        ; ANGLE =   8.4375  COSINE = 0.989177
         .word    07E60h        ; ANGLE =   9.1406  COSINE = 0.987301
         .word    07E1Eh        ; ANGLE =   9.8438  COSINE = 0.985278
         .word    07dDDh        ; ANGLE =  10.5469  COSINE = 0.983106
         .word    07D8Ah        ; ANGLE =  11.2500  COSINE = 0.980785
         .word    07D3Ah        ; ANGLE =  11.9531  COSINE = 0.978317
         .word    07CE4h        ; ANGLE =  12.6563  COSINE = 0.975702
         .word    07C89h        ; ANGLE =  13.3594  COSINE = 0.972940
         .word    07C2Ah        ; ANGLE =  14.0625  COSINE = 0.970031
         .word    07BC6h        ; ANGLE =  14.7656  COSINE = 0.966976
         .word    07B5Dh        ; ANGLE =  15.4688  COSINE = 0.963776
         .word    07AEFh        ; ANGLE =  16.1719  COSINE = 0.960431
         .word    07A7Dh        ; ANGLE =  16.8750  COSINE = 0.956940
         .word    07A06h        ; ANGLE =  17.5781  COSINE = 0.953306
         .word    0798Ah        ; ANGLE =  18.2813  COSINE = 0.949528
         .word    0790Ah        ; ANGLE =  18.9844  COSINE = 0.945607
         .word    07885h        ; ANGLE =  19.6875  COSINE = 0.941544
         .word    077FBh        ; ANGLE =  20.3906  COSINE = 0.937339
         .word    0776Ch        ; ANGLE =  21.0938  COSINE = 0.932993
         .word    076D9h        ; ANGLE =  21.7969  COSINE = 0.928506
         .word    07642h        ; ANGLE =  22.5000  COSINE = 0.923880
         .word    075A6h        ; ANGLE =  23.2031  COSINE = 0.919114
         .word    07505h        ; ANGLE =  23.9063  COSINE = 0.914210
         .word    07460h        ; ANGLE =  24.6094  COSINE = 0.909168
         .word    073B6h        ; ANGLE =  25.3125  COSINE = 0.903989
         .word    07308h        ; ANGLE =  26.0156  COSINE = 0.898675
         .word    07255h        ; ANGLE =  26.7188  COSINE = 0.893224
         .word    0719Eh        ; ANGLE =  27.4219  COSINE = 0.887640
         .word    070E3h        ; ANGLE =  28.1250  COSINE = 0.881921
         .word    07023h        ; ANGLE =  28.8281  COSINE = 0.876070
         .word    06F5Fh        ; ANGLE =  29.5313  COSINE = 0.870087
         .word    06E97h        ; ANGLE =  30.2344  COSINE = 0.863973
         .word    06DCAh        ; ANGLE =  30.9375  COSINE = 0.857729
         .word    06CF9h        ; ANGLE =  31.6406  COSINE = 0.851355
         .word    06C24h        ; ANGLE =  32.3438  COSINE = 0.844854
         .word    06B4Bh        ; ANGLE =  33.0469  COSINE = 0.838225
         .word    06A6Eh        ; ANGLE =  33.7500  COSINE = 0.831470
         .word    0698Ch        ; ANGLE =  34.4531  COSINE = 0.824589
         .word    068A7h        ; ANGLE =  35.1563  COSINE = 0.817585
         .word    067BDh        ; ANGLE =  35.8594  COSINE = 0.810457
         .word    066D0h        ; ANGLE =  36.5625  COSINE = 0.803208
         .word    065DEh        ; ANGLE =  37.2656  COSINE = 0.795837
         .word    064E9h        ; ANGLE =  37.9688  COSINE = 0.788347
         .word    063EFh        ; ANGLE =  38.6719  COSINE = 0.780737
         .word    062F2h        ; ANGLE =  39.3750  COSINE = 0.773011
         .word    061F1h        ; ANGLE =  40.0781  COSINE = 0.765168
         .word    060ECh        ; ANGLE =  40.7813  COSINE = 0.757209
         .word    05FE4h        ; ANGLE =  41.4844  COSINE = 0.749137
         .word    05ED7h        ; ANGLE =  42.1875  COSINE = 0.740951
         .word    05DC8h        ; ANGLE =  42.8906  COSINE = 0.732655
         .word    05CB4h        ; ANGLE =  43.5938  COSINE = 0.724247
         .word    05B9Dh        ; ANGLE =  44.2969  COSINE = 0.715731
         .word    05A82h        ; ANGLE =  45.0000  COSINE = 0.707107
         .word    05964h        ; ANGLE =  45.7031  COSINE = 0.698377
         .word    05843h        ; ANGLE =  46.4063  COSINE = 0.689541
         .word    0571Eh        ; ANGLE =  47.1094  COSINE = 0.680601
         .word    055F6h        ; ANGLE =  47.8125  COSINE = 0.671559
         .word    054CAh        ; ANGLE =  48.5156  COSINE = 0.662416
         .word    0539Bh        ; ANGLE =  49.2188  COSINE = 0.653173
         .word    05269h        ; ANGLE =  49.9219  COSINE = 0.643832
         .word    05134h        ; ANGLE =  50.6250  COSINE = 0.634394
         .word    04FFBh        ; ANGLE =  51.3281  COSINE = 0.624860
         .word    04EC0h        ; ANGLE =  52.0313  COSINE = 0.615232
         .word    04D81h        ; ANGLE =  52.7344  COSINE = 0.605512
         .word    04C40h        ; ANGLE =  53.4375  COSINE = 0.595700
         .word    04AFBh        ; ANGLE =  54.1406  COSINE = 0.585799
         .word    049B4h        ; ANGLE =  54.8438  COSINE = 0.575809
         .word    0486Ah        ; ANGLE =  55.5469  COSINE = 0.565733
         .word    0471Dh        ; ANGLE =  56.2500  COSINE = 0.555571
         .word    045CDh        ; ANGLE =  56.9531  COSINE = 0.545326
         .word    0447Bh        ; ANGLE =  57.6563  COSINE = 0.534998
         .word    04326h        ; ANGLE =  58.3594  COSINE = 0.524590
         .word    041CEh        ; ANGLE =  59.0625  COSINE = 0.514103
         .word    04074h        ; ANGLE =  59.7656  COSINE = 0.503539
         .word    03F17h        ; ANGLE =  60.4688  COSINE = 0.492899
         .word    03DB8h        ; ANGLE =  61.1719  COSINE = 0.482184
```

```
        .word     03C57h        ; ANGLE =  61.8750  COSINE = 0.471397
        .word     03AF3h        ; ANGLE =  62.5781  COSINE = 0.460539
        .word     0398Dh        ; ANGLE =  63.2813  COSINE = 0.449612
        .word     03825h        ; ANGLE =  63.9844  COSINE = 0.438617
        .word     036BAh        ; ANGLE =  64.6875  COSINE = 0.427556
        .word     0354Eh        ; ANGLE =  65.3906  COSINE = 0.416430
        .word     033DFh        ; ANGLE =  66.0938  COSINE = 0.405242
        .word     0326Eh        ; ANGLE =  66.7969  COSINE = 0.393992
        .word     030FCh        ; ANGLE =  67.5000  COSINE = 0.382684
        .word     02F87h        ; ANGLE =  68.2031  COSINE = 0.371318
        .word     02E11h        ; ANGLE =  68.9063  COSINE = 0.359895
        .word     02C99h        ; ANGLE =  69.6094  COSINE = 0.348419
        .word     02B1Fh        ; ANGLE =  70.3125  COSINE = 0.336890
        .word     029A4h        ; ANGLE =  71.0156  COSINE = 0.325310
        .word     02827h        ; ANGLE =  71.7188  COSINE = 0.313682
        .word     026A8h        ; ANGLE =  72.4219  COSINE = 0.302006
        .word     02528h        ; ANGLE =  73.1250  COSINE = 0.290285
        .word     023A7h        ; ANGLE =  73.8281  COSINE = 0.278520
        .word     02224h        ; ANGLE =  74.5313  COSINE = 0.266713
        .word     0209Fh        ; ANGLE =  75.2344  COSINE = 0.254866
        .word     01F1Ah        ; ANGLE =  75.9375  COSINE = 0.242980
        .word     01D93h        ; ANGLE =  76.6406  COSINE = 0.231058
        .word     01C0Ch        ; ANGLE =  77.3438  COSINE = 0.219101
        .word     01A83h        ; ANGLE =  78.0469  COSINE = 0.207111
        .word     018F9h        ; ANGLE =  78.7500  COSINE = 0.195090
        .word     0176Eh        ; ANGLE =  79.4531  COSINE = 0.183040
        .word     015E2h        ; ANGLE =  80.1563  COSINE = 0.170962
        .word     01455h        ; ANGLE =  80.8594  COSINE = 0.158858
        .word     012C8h        ; ANGLE =  81.5625  COSINE = 0.146730
        .word     0113Ah        ; ANGLE =  82.2656  COSINE = 0.134580
        .word     0FABh         ; ANGLE =  82.9688  COSINE = 0.122410
        .word     0E1Ch         ; ANGLE =  83.6719  COSINE = 0.110222
        .word     0C8Ch         ; ANGLE =  84.3750  COSINE = 0.098017
        .word     0AFBh         ; ANGLE =  85.0781  COSINE = 0.085797
        .word     096Bh         ; ANGLE =  85.7813  COSINE = 0.073564
        .word     07D9h         ; ANGLE =  86.4844  COSINE = 0.061320
        .word     0648h         ; ANGLE =  87.1875  COSINE = 0.049067
        .word     04B6h         ; ANGLE =  87.8906  COSINE = 0.036807
        .word     0324h         ; ANGLE =  88.5938  COSINE = 0.024541
        .word     0192h         ; ANGLE =  89.2969  COSINE = 0.012271
SINE0:  .set      $
        .word     00h           ; ANGLE =  90.0000  COSINE = -.000001
        .word     0fE6Eh        ; ANGLE =  90.7031  COSINE = -.012272
        .word     0fCDCh        ; ANGLE =  91.4063  COSINE = -.024542
        .word     0fB4Ah        ; ANGLE =  92.1094  COSINE = -.036808
        .word     0F9B8h        ; ANGLE =  92.8125  COSINE = -.049069
        .word     0F827h        ; ANGLE =  93.5156  COSINE = -.061322
        .word     0F695h        ; ANGLE =  94.2188  COSINE = -.073566
        .word     0F505h        ; ANGLE =  94.9219  COSINE = -.085798
        .word     0F374h        ; ANGLE =  95.6250  COSINE = -.098018
        .word     0F1E4h        ; ANGLE =  96.3281  COSINE = -.110223
        .word     0F055h        ; ANGLE =  97.0313  COSINE = -.122412
        .word     0EEC6h        ; ANGLE =  97.7344  COSINE = -.134582
        .word     0ED38h        ; ANGLE =  98.4375  COSINE = -.146732
```

```
        .word     0EBAAh        ; ANGLE =  99.1406  COSINE = -.158859
        .word     0EA1Eh        ; ANGLE =  99.8438  COSINE = -.170963
        .word     0E892h        ; ANGLE = 100.5469  COSINE = -.183041
        .word     0E707h        ; ANGLE = 101.2500  COSINE = -.195092
        .word     0E57Dh        ; ANGLE = 101.9531  COSINE = -.207113
        .word     0E3F4h        ; ANGLE = 102.6563  COSINE = -.219103
        .word     0E26Dh        ; ANGLE = 103.3594  COSINE = -.231060
        .word     0E0E6h        ; ANGLE = 104.0625  COSINE = -.242982
        .word     0DF61h        ; ANGLE = 104.7656  COSINE = -.254867
        .word     0DDDCh        ; ANGLE = 105.4688  COSINE = -.266714
        .word     0DC59h        ; ANGLE = 106.1719  COSINE = -.278521
        .word     0DAD8h        ; ANGLE = 106.8750  COSINE = -.290286
        .word     0D958h        ; ANGLE = 107.5781  COSINE = -.302008
        .word     0D7D9h        ; ANGLE = 108.2813  COSINE = -.313683
        .word     0D65Ch        ; ANGLE = 108.9844  COSINE = -.325312
        .word     0D4E1h        ; ANGLE = 109.6875  COSINE = -.336892
        .word     0D367h        ; ANGLE = 110.3906  COSINE = -.348420
        .word     0D1EFh        ; ANGLE = 111.0938  COSINE = -.359897
        .word     0D079h        ; ANGLE = 111.7969  COSINE = -.371319
        .word     0CF04h        ; ANGLE = 112.5000  COSINE = -.382685
        .word     0CD92h        ; ANGLE = 113.2031  COSINE = -.393994
        .word     0CC21h        ; ANGLE = 113.9063  COSINE = -.405243
        .word     0CAB2h        ; ANGLE = 114.6094  COSINE = -.416431
        .word     0C946h        ; ANGLE = 115.3125  COSINE = -.427557
        .word     0C7DBh        ; ANGLE = 116.0156  COSINE = -.438618
        .word     0C673h        ; ANGLE = 116.7188  COSINE = -.449613
        .word     0C50Dh        ; ANGLE = 117.4219  COSINE = -.460541
        .word     0C3A9h        ; ANGLE = 118.1250  COSINE = -.471399
        .word     0C248h        ; ANGLE = 118.8281  COSINE = -.482186
        .word     0C0E9h        ; ANGLE = 119.5313  COSINE = -.492900
        .word     0BF8Ch        ; ANGLE = 120.2344  COSINE = -.503540
        .word     0BE32h        ; ANGLE = 120.9375  COSINE = -.514105
        .word     0BCDAh        ; ANGLE = 121.6406  COSINE = -.524592
        .word     0BB85h        ; ANGLE = 122.3438  COSINE = -.535000
        .word     0BA33h        ; ANGLE = 123.0469  COSINE = -.545327
        .word     0B8E3h        ; ANGLE = 123.7500  COSINE = -.555572
        .word     0B796h        ; ANGLE = 124.4531  COSINE = -.565734
        .word     0B64Ch        ; ANGLE = 125.1563  COSINE = -.575810
        .word     0B505h        ; ANGLE = 125.8594  COSINE = -.585800
        .word     0B3C0h        ; ANGLE = 126.5625  COSINE = -.595701
        .word     0B27Fh        ; ANGLE = 127.2656  COSINE = -.605513
        .word     0B140h        ; ANGLE = 127.9688  COSINE = -.615234
        .word     0B005h        ; ANGLE = 128.6719  COSINE = -.624862
        .word     0AECCh        ; ANGLE = 129.3750  COSINE = -.634395
        .word     0AD97h        ; ANGLE = 130.0781  COSINE = -.643834
        .word     0AC65h        ; ANGLE = 130.7813  COSINE = -.653175
        .word     0AB36h        ; ANGLE = 131.4844  COSINE = -.662418
        .word     0AA0Ah        ; ANGLE = 132.1875  COSINE = -.671561
        .word     0A8E2h        ; ANGLE = 132.8906  COSINE = -.680603
        .word     0A7BDh        ; ANGLE = 133.5938  COSINE = -.689543
        .word     0A69Ch        ; ANGLE = 134.2969  COSINE = -.698378
        .word     0A57Dh        ; ANGLE = 135.0000  COSINE = -.707109
        .word     0A463h        ; ANGLE = 135.7031  COSINE = -.715733
        .word     0A34Ch        ; ANGLE = 136.4063  COSINE = -.724249
```

```
.word    0A238h         ; ANGLE = 137.1094  COSINE = -.732656
.word    0A128h         ; ANGLE = 137.8125  COSINE = -.740953
.word    0A01Ch         ; ANGLE = 138.5156  COSINE = -.749138
.word    09F14h         ; ANGLE = 139.2188  COSINE = -.757211
.word    09E0fh         ; ANGLE = 139.9219  COSINE = -.765169
.word    09D0Eh         ; ANGLE = 140.6250  COSINE = -.773012
.word    09C11h         ; ANGLE = 141.3281  COSINE = -.780739
.word    09B17h         ; ANGLE = 142.0313  COSINE = -.788348
.word    09A22h         ; ANGLE = 142.7344  COSINE = -.795839
.word    09930h         ; ANGLE = 143.4375  COSINE = -.803210
.word    09843h         ; ANGLE = 144.1406  COSINE = -.810459
.word    09759h         ; ANGLE = 144.8438  COSINE = -.817587
.word    09674h         ; ANGLE = 145.5469  COSINE = -.824591
.word    09592h         ; ANGLE = 146.2500  COSINE = -.831472
.word    094B5h         ; ANGLE = 146.9531  COSINE = -.838227
.word    093DCh         ; ANGLE = 147.6563  COSINE = -.844856
.word    09307h         ; ANGLE = 148.3594  COSINE = -.851357
.word    09236h         ; ANGLE = 149.0625  COSINE = -.857730
.word    09169h         ; ANGLE = 149.7656  COSINE = -.863975
.word    090A1h         ; ANGLE = 150.4688  COSINE = -.870089
.word    08fDDh         ; ANGLE = 151.1719  COSINE = -.876072
.word    08F1Dh         ; ANGLE = 151.8750  COSINE = -.881923
.word    08E62h         ; ANGLE = 152.5781  COSINE = -.887641
.word    08DABh         ; ANGLE = 153.2813  COSINE = -.893226
.word    08CF8h         ; ANGLE = 153.9844  COSINE = -.898676
.word    08C4Ah         ; ANGLE = 154.6875  COSINE = -.903991
.word    08BA0h         ; ANGLE = 155.3906  COSINE = -.909170
.word    08AFBh         ; ANGLE = 156.0938  COSINE = -.914211
.word    08A5Ah         ; ANGLE = 156.7969  COSINE = -.919115
.word    089BEh         ; ANGLE = 157.5000  COSINE = -.923881
.word    08927h         ; ANGLE = 158.2031  COSINE = -.928508
.word    08894h         ; ANGLE = 158.9063  COSINE = -.932994
.word    08805h         ; ANGLE = 159.6094  COSINE = -.937341
.word    0877Bh         ; ANGLE = 160.3125  COSINE = -.941546
.word    086F6h         ; ANGLE = 161.0156  COSINE = -.945609
.word    08676h         ; ANGLE = 161.7188  COSINE = -.949530
.word    085FAh         ; ANGLE = 162.4219  COSINE = -.953307
.word    08583h         ; ANGLE = 163.1250  COSINE = -.956942
.word    08511h         ; ANGLE = 163.8281  COSINE = -.960432
.word    084A3h         ; ANGLE = 164.5313  COSINE = -.963777
.word    0843Ah         ; ANGLE = 165.2344  COSINE = -.966978
.word    083D6h         ; ANGLE = 165.9375  COSINE = -.970032
.word    08377h         ; ANGLE = 166.6406  COSINE = -.972941
.word    0831Ch         ; ANGLE = 167.3438  COSINE = -.975703
.word    082C6h         ; ANGLE = 168.0469  COSINE = -.978318
.word    08276h         ; ANGLE = 168.7500  COSINE = -.980786
.word    0822Ah         ; ANGLE = 169.4531  COSINE = -.983106
.word    081E2h         ; ANGLE = 170.1563  COSINE = -.985278
.word    081A0h         ; ANGLE = 170.8594  COSINE = -.987302
.word    08163h         ; ANGLE = 171.5625  COSINE = -.989177
.word    0812Ah         ; ANGLE = 172.2656  COSINE = -.990903
.word    080F6h         ; ANGLE = 172.9688  COSINE = -.992480
.word    080C8h         ; ANGLE = 173.6719  COSINE = -.993907
.word    0809Eh         ; ANGLE = 174.3750  COSINE = -.995185
```

```
        .word    08079h         ; ANGLE = 175.0781  COSINE = -.996313
        .word    08059h         ; ANGLE = 175.7813  COSINE = -.997291
        .word    0803Eh         ; ANGLE = 176.4844  COSINE = -.998118
        .word    08027h         ; ANGLE = 177.1875  COSINE = -.998796
        .word    08016h         ; ANGLE = 177.8906  COSINE = -.999323
        .word    0800Ah         ; ANGLE = 178.5938  COSINE = -.999699
        .word    08002h         ; ANGLE = 179.2969  COSINE = -.999925
        .word    08000h         ; ANGLE = 180.0000  COSINE = -1.000000
*
******************************************************************************
*
*   MAIN PROGRAM
*
******************************************************************************
*
*   INITIALIZATION CODE
*
******************************************************************************
*
START   DINT
*
******************************************************************************
*
*   CLEAR ALL RAM
*
******************************************************************************
*
        LDPK     0
        LARP     AR1
        LARK     AR1,143
        ZAC
CRAM    SACL     *
        BANZ     CRAM
        SACL     0              ; CLEAR RAM 0
*
******************************************************************************
*
*   START INITIALIZATION CODE - FIRST INITIALIZE PAGE 1 DATA
*
******************************************************************************
*
        LDPK     1              ; WINDOW FOR SLEW MODE IN
        LACK     PSM
        TBLR     POSSM          ; POSITIVE DIRECTION
        LACK     NSM
        TBLR     NEGSM          ; WINDOW FOR SLEW MODE IN
*
```

```
************************************************************************
*
*    INITIALIZE PAGE 0 DATA
*
*    SYSTEM IS ORIGINALLY INITIALIZED AT A PSEUDO 1200 BPS, TX, RX IDLE LINE
*    MODE, 16 SAMPLES/BAUD TO ACCOMODATE THE START-UP CONDITION OF THE
*    TMS70A2400 MODEM CONTROLLER.
*
************************************************************************
*
         LDPK      0
         LACK      8
         SACL      IRCNT
         LACK      2
         SACL      STATUS
         LACK      15
         SACL      SCNT
         SACL      XSCNTR
         SACL      RSCNTR
         LACK      1
         SACL      ONE
*
************************************************************************
*
*    INITIALIZE SERIAL PORT CONTROL REGISTERS
*
************************************************************************
*
         LACK      DA1
         TBLR      TMPO
         OUT       TMPO,PAO
         LACK      DA2
         TBLR      TMPO
         OUT       TMPO,PA1
         LACK      DA3
         TBLR      TMPO
         OUT       TMPO,PAO
*
************************************************************************
*
*    INITIALIZE AGC GAIN AND LOCK INDICATOR. (MOD. 5/29)
*
************************************************************************
*
         LACK      MAXALP
         TBLR      ALPHA          ; SET ALPHA TO ITS MAX POSSIBLE VALUE
*
************************************************************************
*
*    INITIALIZE STATUS WORD TO SET AFE GAIN STAGE ON
*
************************************************************************
*
         LACK      080h
         SACL      STWRD
*
************************************************************************
*
*    INITIALIZE HYSTERESIS COUNTER TO 8000h
*
************************************************************************
*
         LAC       ONE,15
         SACL      HYST
*
************************************************************************
*
*    MAIN PROGRAM SEQUENCER
*
*    THE MAIN PROGRAM SEQUENCER PROVIDES THE TIMING FOR THE MAIN PROGRAM LOOP
*    AND CALLS THE VARIOUS SUBROUTINES AT THE APPROPRIATE TIMES. THE MAIN LOOP
*    IS COMPLETED ONCE EVERY BAUD INTERVAL OR EQUIVALENTLY 300 TIMES/SEC FOR
*    FSK.
*
*    THE PCM CODEC HAS A FIXED SAMPLING RATE OF 9.6 KHZ, WHICH MEANS THAT IN
*    THE FSK MODE THE DSP BAUD PERIOD CORRESPONDS TO 32 PCM SAMPLES.
*
************************************************************************
*
WAIT     IN        TMPO,PAO       ; WAIT FOR FR INTERRUPT FLAG
         LACK      8              ; IF NOT LOOP HERE
         AND       TMPO           ; IF YES TRANSMIT AND RECEIVE FROM PORT 1
         BZ        WAIT           ; AND RESET THE INTERRUPT FLAG
*
SEQU     OUT       XOUT,PA1
         IN        RIN,PA1
         LACK      DA3
         TBLR      TMPO
         OUT       TMPO,PAO
*
************************************************************************
*
*    EXECUTE TRANSMITTER TASK FOR TIME SLOT AND UPDATE SAMPLE COUNTER.
*
************************************************************************
*
         LAC       XSCNTR
         SUB       SCNT
         BLZ       SEQU2          ; IF ZERO CALL CCI
*
SEQU01   CALL      CCI            ; CALL COMMAND INTERPRETER
*
************************************************************************
*
*    EXECUTE THE SAMPLE TASKS
*
************************************************************************
*
```

```
SEQU2     LACK      030h            ; CHECK FOR ANSWER TONE
          AND       STATUS
*
********************************************************************************
*
*    CHECK IF IDLE: IF BITS 4 & 5 OF STATUS ARE 0 THEN TX IN IDLE MODE =>
*    TRANSMIT UNMODULATED
*
********************************************************************************
*
          BZ        SEQU20
*
          SUB       ONE,4           ; IF 0 => ANSWER TONE
          BZ        SEQU3           ; ANSWER TONE => CALL FSK
*
          LACK      3
          AND       STATUS          ; CHECK BITS 1 AND 0
          BZ        SEQU3
*
          LAC       DTFLAG          ; CHECK FOR DTMF DIAL MODE DUAL TONE
*                                   ; TRANSMISSION
          BNZ       SEQU3
          B         SEQU4
*
SEQU3     CALL      FSKTX           ; FSK OR ANSWER TONE OR DTMF
*
          B         SEQU4
*
********************************************************************************
*
*    IN IDLE MODE TRANSMIT A 0
*
********************************************************************************
*
SEQU20    ZAC
          SACL      XOUT
*
SEQU4     LAC       XSCNTR
          SUB       ONE
          BGEZ      SEQU5
          LAC       SCNT
SEQU5     SACL      XSCNTR
*
```

```
********************************************************************************
*
*    EXECUTE RECEIVER TASK FOR TIME SLOT AND UPDATE RECEIVER SAMPLE COUNTER.
*    IF RECEIVER IN IDLE MODE RETURN TO WAIT STATE
*
********************************************************************************
*
          LACK      RMMSK
          AND       STATUS
          BNZ       SEQU9
          LAC       RSCNTR
          SUB       ONE
          BGEZ      SEQU22
          LAC       SCNT
          OUT       STWRD,PA5
SEQU22    SACL      RSCNTR
          B         WAIT
*
********************************************************************************
*
*    USE ROUTINE 'RSTSK' TO PERFORM FSK DEMODULATION
*
********************************************************************************
*
SEQU9     CALL      RSTSK
          LACK      3
          AND       STATUS          ; CHECK FSK OPERATION
          BZ        DECRS           ; IF SO, JUST DECREMENT RX SAMPLE COUNTER
*
          LACK      RSEQTB          ; IN HANDSHAKING MODE CALL RTASK
          ADD       RSCNTR          ; SUBROUTINE ONCE PER SAMPLE. ONLY ONE
          TBLR      TMP1            ; NON-TRIVIAL FUNCTION, RTSK10, IS
          LAC       TMP1            ; ACTUALLY CALLED.
          CALA
*
DECRS     LAC       RSCNTR
          SUB       ONE
          BGEZ      SEQU6           ; IF NOT THE END OF BAUD, JUST CONTINUE
          CALL      AGC             ; ELSE DO AGC ONCE PER BAUD.
*
SEQU66    LAC       SCNT            ; RESET BAUD COUNTER
SEQU6     SACL      RSCNTR
*
          B         WAIT
*
********************************************************************************
*
*    SUBROUTINES
*
********************************************************************************
*
FSKTX:    .set      $
          ZALS      TXPHS           ; BRING IN TX ANGLE
          CALL      SINGEN          ; GENERATE TONE AT APPROPRIATE FREQ
```

```
*                               ; RESULT RETURNED IN TMP3
        LAC     DTFLAG
        BNZ     ISDTMF
*
        LAC     TMP3,13         ; 4S12 FORMAT
        SACH    XOUT
        LT      XOUT
        MPYK    0700h
        PAC
        SACH    XOUT,4
        B       NODTMF
*
ISDTMF: .set    $
        LAC     TMP3,15         ; 2S30 FORMAT - LOWER FREQUENCY
        SACH    TMP5            ; 2S14 FORMAT - LOWER FREQUENCY
*
        ZALS    RXPHS           ; IN DTMF MODE, HIGH FREQ IS HANDLED BY
*                               ; BY RXPHS
        CALL    SINGEN          ; GENERATE HIGH FREQUENCY TONE
        ZALS    RXPHS           ; INCREMENT HIGH FREQUENCY PHASE ANGLE
        ADDS    RXFRQ           ; BY SECOND TONE FREQUENCY
        SACL    RXPHS           ; STORE AWAY
*
        LAC     TMP3,15
        SACH    TMP3
        LT      TMP5
        MPY     DTMFL
        PAC
        LT      TMP3
        MPY     DTMFH
        APAC
        SACH    XOUT,4
*
NODTMF: .set    $
        ZALS    TXPHS           ; INCREMENT TX ANGLE BY APPROPRIATE 0 OR 1
        ADDS    TXFRQ           ; FREQUENCY AND STORE IN TXPHS
        SACL    TXPHS
*
        NONE    RET
*
        .copy   "CCIDTM.A00"    ; INCLUDES CODE FOR DTMF
*
*******************************************************************
*
*   CONTROLLER COMMAND INTERPRETER (CCI)
*
*   THE FOLLOWING CODE READS A COMMAND FROM THE TMS70A2400 ON PORT 5 AND
*   INTERPRETS IT ACCORDING TO THE RULES SPECIFIED IN THE CONTROLLER-DSP
*   INTERFACE DOCUMENT. THE 320 READS ONE COMMAND EVERY BAUD PERIOD. THE BAUD
*   RATE IS INITIALLY SET TO 600, AND THE BAUD CLOCK IS DERIVED FROM THE
*   SERIAL PORT FR SIGNAL.
*
*******************************************************************
*
```

```
CCI     IN      XDATA,PA5       ; READ COMMAND
*
        LACK    0F0h
        AND     XDATA           ; MASK OFF 4 LSBS OF COMMAND
        SACL    TMP2
        LACK    090h            ; CHECK FOR DTMF
        SUB     TMP2
        BZ      DTCONT
*
        ZAC                     ; IF NOT,
        SACL    DTFLAG          ; CLEAR DTMF DIAL FLAG
*
CONT    LAC     TMP2,12         ; COMMAND BITS TO ACCH LSBS
        SACH    TMP2
*
*******************************************************************
*
*   NOW THE ACCUMULATOR VALUE CORRESPONDS TO THE FOLLOWING COMMANDS AND
*   CORRESPONDING SERVICE SUBROUTINES
*
*******************************************************************
*
*   ACC         COMMAND                 SUBROUTINE
*
*   Fh          PROTOCOL SELECT         PROTO
*   Dh          SET OPERATING MODE      OPER
*
*   9h          DIAL DTMF               DTMF
*   8h          XMIT MODE               XMODE
*   7h          RECV MODE               RMODE
*
*   5h          FSK DATA MODE           FSKSET
*   4h          RESET                   RESET
*
*******************************************************************
*
*   CHECK FIRST IF RECEIVER IS IN DATA MODE. IN WHICH CASE IGNORE ALL
*   COMMANDS EXCEPT 2Xh, 1Xh AND 00h.
*
*******************************************************************
*
        LACK    RMMSK           ; REC. MODE MASK
        AND     STATUS
        SUB     ONE,7           ; CHECK IF BITS 7 AND 6 ARE ONES
        SUB     ONE,6
        BNZ     CCI1
*
*******************************************************************
*
*   REC. IN DATA MODE => IGNORE COMMANDS >2
*
*******************************************************************
*
        LAC     TMP2
```

```
        SUB     ONE,2
        BLEZ    CCI1            ; IF COMMAND LARGER THAN >2 EXIT COMMAND
        RET                     ; INTERPRETER.
*
*****************************************************************************
*
*   CALL THE APPROPRIATE SERVICE SUBROUTINE (REFER TO CMDTBL TABLE).
*
*****************************************************************************
*
CCI1    LACK    CMDTBL          ; BASE OF COMMAND TABLE
        ADD     TMP2            ; ADD COMMAND OPCODE
        TBLR    TMP2            ; READ ADDRESSS FROM TABLE
        LAC     TMP2            ; LOAD SUB. ADDR. INTO ACC.
        CALA                    ; CALL SERVICE SUBROUTINE
*
        RET                     ; EXIT COMMAND INTERPRETER.
*
*****************************************************************************
*
*   COMMAND INTERPRETER SUBROUTINES
*
*****************************************************************************
*
*   PROTOCOL SELECT COMMAND
*
*****************************************************************************
*
PROTO:  .set    $
*
*****************************************************************************
*
*   EXTRACT TWO LSB'S OF COMMAND
*
*****************************************************************************
*
        LACK    03h             ; MASK OFF BITS 2 AND 3 OF COMMAND
        AND     XDATA
*
*****************************************************************************
*
*   SET SPEED BITS STATUS REGISTER:
*
*   BITS 1 AND 0 = 00 FOR 300 BPS (FSK)
*                = 01 RESERVED
*                = 10 RESERVED
*                = 11 RESERVED
*
*****************************************************************************
*
        SACL    TMP1
        LACK    0FCh
        AND     STATUS          ; ZERO BITS 0 AND 1
        SACL    STATUS
```

```
*****************************************************************************
*
*   DETERMINE FSK FREQUENCIES AND SET BAUD COUNTER. ALSO SET OTHER FSK
*   SIGNAL PROCESSING PARAMETERS
*
*****************************************************************************
*
PROTO2  LACK    31
        SACL    SCNT            ; BAUD COUNTER IS 32
*
        LAC     ONE,2
        AND     XDATA
        SACL    CCITT
*
        LAC     ONE,3
        SACL    FSKFLG
*
*****************************************************************************
*
*   ACCUMULATOR NOW CONTAINS THE NUMERAL 8 LOGICAL AND HENCE IDENTIFIES
*   ORIGINATE_/ANSWER MODES
*
*****************************************************************************
*
        AND     XDATA           ; ISOLATE ORIGINATE_/ANSWER BIT
        SACL    OAFLAG          ; SET OAFLAG <> 0 IN ANSWER MODE = 0 IN
*                               ; ORIGINATE MODE
        LACK    0Ch             ; MASK TWO LSBS OF COMMAND
        AND     XDATA
        SACL    TMP1
        LACK    FSKTBL          ; ADD BASE OF FSK TABLE
        ADD     TMP1
        SACL    F1ADD           ; 1 FREQUENCY
        ADD     ONE
        SACL    F0ADD           ; 0 FREQUENCY ADDRESS
        ADD     ONE
        TBLR    B1FSK           ; FSK LOWPASS FILTER COEFFICIENT
        ADD     ONE
        TBLR    GAIN            ; FSK MODE GAIN
        SUB     ONE
        SACL    TMP1            ; TMP1 NOW POINTS TO B1FSK
*
*****************************************************************************
*
*   SET FSK TIMING RECOVERY PARAMETERS.
*
*****************************************************************************
*
        LACK    0Ch             ; ACC = 12
        SACL    TRANS           ; TRANS = 12
*
*****************************************************************************
*
*   SET FSK RECEIVE FILTER COEFFICIENTS AND SLICER DEAD ZONE
```

```
*
***********************************************************************
*
        LACK    FSKA1           ; LOAD ADDRESS OF A1 COEFFICIENT
        TBLR    A1FSK
        ADD     ONE
        TBLR    A2FSK           ; A2FSK CONTAINS A2 COEFFICIENT
        LACK    ZONE            ; ACC = ADDRESS OF DEAD ZONE
        TBLR    DZONE           ; DEAD ZONE OF WINDOW COMPARATOR FOR FSK
*                               ; DECISION
*
***********************************************************************
*
*   NOW CHECK FOR ANALOG LOOPBACK. IF IT IS IN ANALOG LOOPBACK MODE, THEN WE
*   NEED TO MODIFY THE RECEIVER PARAMETERS TO CONFIGURE RECEIVER IN THE SAME
*   BAND AS THE TRANSMITTER.
*
***********************************************************************
*
        LACK    ANMSK
        AND     STATUS          ; STATUS BIT IS 1 FOR ANALOG
        BZ      PROTO5          ; AND ZERO OTHERWISE
*
***********************************************************************
*
*   CHECK IF ANSWER OR ORIGINATE. IF IN ORIGINATE MODE THE THE RECEIVER MUST
*   ALSO BE CONFIGURED FOR LOW BAND. IN WHICH CASE THE PARAMETERS REQUIRED
*   ARE AT ADDRESS TMP1+8 IF IN ANSWER MODE, THE RECEIVER MUST BE PUT IN HIGH
*   BAND AND THE PARAMETERS ARE AT ADDRESS TMP1-8.(PLEASE REFER TO THE
*   FSKTBL)
*
***********************************************************************
*
        LACK    08h             ; CHECK ORIG/ANS BIT 3 OF XDATA
        AND     XDATA
        BZ      PROTO6
*
        LACK    0
        SACL    OAFLAG
        LAC     TMP1            ; IN ANSWER MODE SUBTRACT 8 FROM TMP1
        SUB     ONE,3
        B       PROTO7
*
PROTO6  LACK    8               ; WE'RE IN ORIGIN MODE,
        SACL    OAFLAG          ; SET OAFLAG <> 0 FOR LOWBAND
        LAC     TMP1            ; IN ORIGINATE MODE ADD 8
        ADD     ONE,3           ; TO TMP1 (THIS ONLY HAPPENS IN ANALOOP)
PROTO7  TBLR    B1FSK           ; READ B1 COEFF
        ADD     ONE
        TBLR    GAIN            ; READ GAIN
PROTO5  RET
*
***********************************************************************
*
```

```
*   SELECT GUARD TONE
*
*   GUARD   RET
*
***********************************************************************
*
*   SET OPERATING MODE
*
***********************************************************************
*
OPER    LACK    LMMSK           ; ZERO ANALOG LOOPBACK
        AND     STATUS          ; LOC. DIG. LOOPBACK BITS
*
***********************************************************************
*
*   CHECK OPERATING MODE
*
***********************************************************************
*
        LACK    03h             ; MASK OFF BITS 2 AND 3 OF COMMAND
        AND     XDATA
        BZ      OPER1           ; IF ZERO => LINE MODE (RET)
        SUB     ONE
        BZ      ANLB            ; IF ONE => ANALOG LOOPBACK
*
OPER1   RET
*
ANLB    LACK    ANMSK           ; SET ANALOG LOOPBACK
        OR      STATUS          ; STATUS BIT
        SACL    STATUS
        RET
*
***********************************************************************
*
*   DIAL DTMF
*
*   DTMF SET-UP ROUTINE : LOOKUP THE LOW AND HIGH FREQUENCIES CORRESPONDING
*   TO EVERY DIGIT AND PLACE IN TXFRQ AND RXFRQ RESPECTIVELY.
*
***********************************************************************
*
DTMF    LACK    15              ; MASK FOR ISOLATING THE DIGIT
        AND     XDATA
        SACL    TMP0            ; STORE AWAY TEMPORARILY
        LACK    TONTBL          ; BRING IN BASE ADDRESS OF TONE TABLE
        ADD     TMP0,2          ; LEFT SHIFT IS REQUIRED AS THERE ARE FOUR
*                               ;   ENTRIES PER DIGIT
        TBLR    TXFRQ           ; READ LOW FREQ INTO TXFRQ
        ADD     ONE             ; INCREMENT POINTER TOWARDS HIGH FREQ
        TBLR    RXFRQ           ; READ HIGH FREQ INTO RXFRQ
        ADD     ONE             ; READ IN LO FRQ GAIN
        TBLR    DTMFL
        ADD     ONE             ; READ IN HI FRQ GAIN
        TBLR    DTMFH
```

```
*
        LAC     ONE             ; SET DTMF DIAL MODE FLAG
        SACL    DTFLAG
*
        RET
*
****************************************************************************
*
*   TRANSMITTER MODE SELECT
*
****************************************************************************
*
XMODE   LACK    03h             ; MASK OFF BITS 2 AND 3
        AND     XDATA
        SACL    TMP1            ; SAVE MODE BITS TO SET STATUS
        SUB     ONE             ; CHECK IF TRANSMIT ANSWER TONE
        BNZ     XMODE1
*
****************************************************************************
*
*   FOR ANSWER TONE, LOAD TONE FREQ IN TXFRQ LOCATION AND 0001h IN XBITS
*   LOCATION IN RAM, AND CALL FSK SUBROUTINE WHICH WILL TRANSMIT TXFRQ (I.E.
*   (THE ANSWER TONE).
*
****************************************************************************
*
        LACK    01h
        SACL    XBITS
        LACK    F21             ; ADDRESS OF 2100 ANS. TONE
        TBLR    TXFRQ
        LACK    0Ch             ; DETERMINE ANS. TONE FREQ.
        AND     XDATA
        BZ      XMODE2          ; IF 0 => 2100 IS RIGHT
*
        LACK    F22             ; OTHERWISE LOAD TXFREQ REG WITH 2225
        TBLR    TXFRQ           ; ANSWER TONE PHASE INC.
        B       XMODE2
*
****************************************************************************
*
*   CHECK FOR REMOTE DIGITAL LOOPBACK
*
****************************************************************************
*
XMODE1  SUB     ONE,1           ; SUBTRACT 2 MORE FROM TMP1
        BNZ     XMODE2          ; IF ZERO => REM. DIG. LOOPBACK
*
****************************************************************************
*
*   HANDLE RDL: (ASSUME THAT THE RECEIVER IS ENABLED)
*
*   PLACE RECEIVED QUADBITS IN XBITS. CHECK FOR 1200 BPS OPERATION. IF SO
*   FORCE A 1 INTO LSB OF XBITS. SET BITS 5 & 4 IN STATUS = 10. PLACE
*   RECEIVED QUADBITS IN XBITS
```

```
*
****************************************************************************
*
        LACK    0Fh             ; MASK OFF QUADBITS
        AND     STWRD
        SACL    XBITS           ; STORE AWAY
*
****************************************************************************
*
*   CHECK FOR 1200 BPS OPERATION
*
****************************************************************************
*
        LACK    0FCh            ; MASK FOR SPEED BITS
        AND     STATUS          ; SPEED BITS
        SUB     ONE,1           ; CHECK FOR 10
        BNZ     XMODE2          ; IF NON-ZERO, JUST CONTINUE
*
        LAC     XBITS           ; ADD A 1 TO LSB FOR 1200 BPS OPERATION
        ADD     ONE
*
****************************************************************************
*
*   SET THE TRANSMITTER MODE BITS IN STATUS REGISTER.
*
****************************************************************************
*
XMODE2  LACK    NTXMSK          ; NEGATION OF TRANSMISSION BITS MASK
        AND     STATUS          ; ZERO THE TX STATUS BITS
        ADD     TMP1,TXSH       ; ADD TX STATUS BITS IN RIGHT POS.
        SACL    STATUS
        RET
*
****************************************************************************
*
*   RECEIVER MODE SELECT
*
*   SET THE RECEIVER STATUS BITS (BITS 6 AND 7) OF STATUS REGISTER TO:
*
*       00 IF RECEIVER IS IDLE
*       01 FOR CALL PROGRESS MONITORING
*       10 FOR DATA MODE
*
****************************************************************************
*
RMODE   LACK    03h             ; MASK OFF BITS 2 AND 3
        AND     XDATA
        SACL    TMP1
        LACK    NRCMSK          ; NEG. OF REC. BITS MASK
        AND     STATUS          ; ZERO REC. STATUS BITS
        ADD     TMP1,RCSH       ; ADD REC STATUS IN RIGHT POS.
        SACL    STATUS
RMODE1  RET
*
```

```
################################################################################
*
*    FSK DATA MODE
*
*    SET UP FSK TRANSMIT FREQ ACCORDING TO THE TX DATA
*
################################################################################
*
FSKSET   LACK     8              ; CHECK THE TRANSMITTED BIT
         AND      XDATA
         BZ       DATA0          ; IF ZERO, DATA MUST BE 0
         LAC      F1ADD          ; POINT ACC TO 1 FREQ
         TBLR     TXFRQ          ; SET TX FREQ TO APPROPRIATE 1 FREQ
         RET
*
DATA0    LAC      F0ADD          ; POINT ACC TO 0 FREQ
         TBLR     TXFRQ          ; SET TX FREQ TO APPROPRIATE 0 FREQ
*
NOFSK    RET
*
################################################################################
*
*    RESET AND EQUALIZER ENABLE ROUTINES
*
################################################################################
*
RESET    LACK     081h
         SACL     STWRD
         OUT      STWRD,PA5
*
         B        START
*
################################################################################
*
*    END CONTROLLER COMMAND INTERPRETER SUBROUTINES
*
################################################################################
```

```
*
         .copy    "SINGEN.A00"
*
################################################################################
*
*    SUBROUTINE : SINGEN
*
*    PURPOSE  : SINE GENERATION
*
*    TASK : GIVEN A COSINE TABLE WITH 257 VALUES AND START ADDRESS COSOFF, AND
*    GIVEN AN ANGLE INDEX IN THE ACCUMULATOR, DETERMINE THE SINE OF THE ANGLE.
*
*    ENTRY CONDITION : THE ANGLE INDEX MUST BE IN THE LOWER ACCUMULATOR.
*
*    EXIT CONDITION : THE SINE OF THE ANGLE IS RETURNED IN TEMPORARY LOCATION
*                     TMP3.
*
*    DESCRIPTION : THE COSINE LOOKUP TABLE CONTAINS 257 VALUES WITH;
*
*        COS[0] = 1.0 AND COS[256] = -1.0
*
*    HENCE ANGLE INDEX 0 MAPS TO  ANGLE 0 AND ANGLE INDEX 256 MAPS TO PI. THE
*    SINE VALUE IS GENERATED BY SUBTRACTING FROM THE ANGLE INDEX THE INDEX
*    CORRESPONDING TO PI/2, TAKING THE ABSOLUTE VALUE, AND HENCE FORMING AN
*    ADDRESS INTO THE LOOKUP TABLE.
*
*    NO OF CYCLES: 17
*
*    NO OF STACK LEVELS USED: 1
*
*    THE ANGLE INDEX IS THE LOWER ACCUMULATOR
*
*    ANGLE INDEX HAS S15.0 FORMAT. MUST SUBTRACT PI/2 VALUE WHICH LAYS AT THE
*    MIDDLE OF THE TABLE AND HAS SS14.0 FORMAT AS VIEWED IN S15.0 FORMAT
*
################################################################################
*
SINGEN   SUB      ONE,14         ; SUBTRACT INDEX OF PI/2
         SACL     TMP3           ; PUT AWAY TEMPORARILY
         ZALH     TMP3           ; PREPARE FOR ABSOLUTE VALUE
         ABS                     ; TAKE ABSOLUTE VALUE
         SACH     TMP3           ; PUT AWAY BEFORE RIGHT SHIFT
*
################################################################################
*
*    THE VALUE STORED IN TMP3 HAS S15.0 FORMAT -- ALBEIT A POSITIVE NUMBER
*
*    A LEFT SHIFT OF 9 BITS CORRESPONDS TO 8S24.0 FORMAT AND SAVING THE HIGH
*    ACCUMULATOR HAS A 8S8.0 FORMAT
*
################################################################################
*
         LAC      TMP3,9         ; ISOLATE 8 MSB'S IN HIGH ACC
         SACH     TMP3           ; PUT AWAY THE 8 MSB'S TEMPORARILY
```

```
*
************************************************************************
*
*   THE NEXT THREE INSTRUCTIONS ELIMINATE ANY SIGN EXTENSION BITS THAT MIGHT
*   HAVE PROPAGATED
*
************************************************************************
*
        LAC     TMP3
        ABS
        SACL    TMP3
*
************************************************************************
*
*   FORM THE FINAL LOOK-UP ADDRESS
*
*   BRING IN THE ADDRESS OFFSET.
*
*   THE BASE ADDRESS IS IN 8S8.0 FORMAT, WHILE THE INDEX IS ALSO IN 8S8.0
*   FORMAT.
*
************************************************************************
*
        LACK    COSOFF
        ADD     TMP3            ; FORM FINAL LOOK-UP ADDRESS
        TBLR    TMP3            ; READ SINE VALUE INTO TMP3
*
        RET
*
************************************************************************
*
*   FSK DEMODULATION FILES
*
************************************************************************
*
        .copy   "RSTSKF.A00"
*
************************************************************************
*
*   DATE: 5-29-86
*
*   SUBROUTINE: RSTSK
*
*   INCLUDES FSK RECEIVER/TIMING RECOVERY
*
*   PURPOSE: RECEIVER PER SAMPLE TASK
*
*   TASK: THIS SUBROUTINE COMBINES SMALLER MODULES TO PERFORM THE SIGNAL
*         PROCESSING FUNCTIONS THAT ARE REQUIRED ON A PER SAMPLE BASIS
*         (9600 Hz).
*
*   ENTRY CONDITION: THE RECEIVED S/M SAMPLE IS IN RAM LOCATION RIN.
*
************************************************************************
```

```
*
RSTSK:  .set    $
        LAC     RIN             ; INPUT 14-BIT S/M SAMPLE
*       SACL    TMP1            ; ARL INTO TMP1
*
************************************************************************
*
*   USE HIGH PASS FILTER. MAKE SURE INCOMING SAMPLE HAS NO SHIFT.
*
************************************************************************
*
*       LAC     TMP1,0          ; ARL
*
************************************************************************
*
*   HIGH PASS FILTER THE INPUT
*
************************************************************************
*
MLHP:   .set    $
        LDPK    1
        SACL    X1
        ZALS    STLSB
        ADDH    ST
        SUB     ST,TAU
        ADDH    X1
        SUB     X1,TAU-1
        SUBH    X2
        ADD     X2,TAU-1
        SACL    STLSB
        SACH    ST
        DMOV    X1
        LDPK    0
        SACH    TMP1
*
************************************************************************
*
*   MLHP LEAVES THE SAMPLE IN TMP1. MULTIPLY IT BY AGC GAIN ALPHA. THE OUTPUT
*   FORMAT IS S4.11 REQUIRING SOME MANIPULATIONS
*
************************************************************************
*
        LT      TMP1            ; MULTIPLY BY AGC WORD
        MPY     ALPHA
        PAC
*
************************************************************************
*
*   SHIFT ACCUMULATOR EIGHT 4 TIMES BEFORE STORING
*
************************************************************************
*
        SACL    TMP0
        SACH    TMP1
```

```
        LAC     TMP0,8
        SACH    TMP0
        LAC     ONE,8           ; MASK OFF ANY SIGN EXTENSION
        SUB     ONE             ; 00FF -> ACC
        AND     TMP0
        SACL    TMP0
        LAC     TMP1,8
        ADD     TMP0
        SACL    TMP1
*
########################################################################
*
*   IN BOTH TMP1 AND TMP2 UPDATE THE SIGNAL POWER ESTIMATE AVESQR
*
*   AVESQR = AVESQR + TMP1^2
*
*   AVESQR IS ZEROED BY THE AGC ROUTINE ONCE PER BAUD.
*
########################################################################
*
        LAC     TMP1,15
        SACH    TMP0            ; TMP0 IN S5.10
        LT      TMP0
        MPY     TMP1
        PAC
        ADDH    AVESQR          ; AVESQR IN S10.5
        SACH    AVESQR
*
########################################################################
*
*   INCREASE SIGNAL ENERGY BY A FACTOR OF 4 FOR COMPATIBILITY WITH THE REST
*   OF THE RECEIVER
*
########################################################################
*
        LAC     TMP1,1          ; MULTIPLY TIMES 2
        SACL    TMP1
        SACL    TMP2
*
########################################################################
*
*   CHECK FOR FSK OPERATION
*
*       LACK    3
*       AND     STATUS
*
*       LAC     TMP1,0
*       SACL    AGCOUT          ; PLACE RECEIVE SIGNAL IN AGCOUT
*       CALL    RXFSK           ; CALL FSK RECEIVER/TIMING RECOVERY
*       RET
        .copy   CCITT.A00
*
########################################################################
*
```

```
*   CHECK FOR FSK OPERATION
*
########################################################################
*
        LACK    3
        AND     STATUS
        LT      TMP1            ; ASSUME V.21
        MPYK    013h            ; MULTIPLY BY 2.5
        LAC     CCITT
        BNZ     V21ORG
        MPYK    018h            ; MULTIPLY BY 3.0
V21ORG  PAC
*
        SACL    TMP0
        SACH    TMP1
        LAC     TMP0,13
        SACH    TMP0
        LAC     ONE,13
        SUB     ONE
        AND     TMP0
        SACL    TMP0
        LAC     TMP1,13
        ADD     TMP0
        SACL    AGCOUT
        CALL    RXFSK
        RET
        .copy   "FDEM20.A00"
*
########################################################################
*
*   DEMODULATOR SECTION
*
*   THIS DESIGN IMPLEMENTS A DELAY OF 5(PI)/2
*
*   AGC
*
*   FSK DEMODULATOR
*
*   MEMORY CONFIGURATION: (CONSECUTIVE ADDRESSES)
*       AGCOUT
*       PDEL0
*       PDEL1
*       PDEL2
*       LPDEL0
*       LPDEL1
*       LPDEL2
*
########################################################################
*
RXFSK   SOVM                    ; SET OVERFLOW MODE
*
########################################################################
*
*   TAKE PRODUCT FOR PRODUCT DEMOD SCHEME. ASSUME ANSWER MODE (2 SAMPLES)
```

```
*
************************************************************************
*
        LT      AGCOUT          ; LOAD T WITH AGC STAGE OUTPUT
        MPY     PDEL2           ; TAKE PRODUCT WITH DELAY LINE OUTPUT
*
************************************************************************
*
*    ORIGINATE/ANSWER FLAG = 0 WHEN TX ORIGINATES THE CALL HENCE RX RECEIVES
*    IN THE HIGH BAND
*
************************************************************************
*
        LAC     OAFLAG          ; CHECK FOR HIGH BAND/LOW BAND
        BNZ     ANSWER          ; LOW BAND, DONE.
        MPY     PDEL1           ; HIGH BAND, USE ONE LESS SAMPLE DELAY
ANSWER  PAC                     ; ACCUMULATE THE PRODUCT OUTPUT
        SACH    PROD,1          ; STORE IN PROD (1,15)
*
************************************************************************
*
*    PRODUCT DEMOD DELAY LINE                   ¦-------------> FOR HIGH BAND
*               -----------------      -------¦    ------
*     AGCOUT--->¦ 1 ZERO FIR FILTER ¦-->¦ Z**-1 ¦-->¦ Z**-1 ¦---> FOR LOW band
*               -----------------      -------     ------
*
*    PRODUCT = AGCOUT * OUTPUT OF DELAY LINE
*    PRODUCT IS LOWPASS FILTERED AND IT'S SIGN INDICATES 0 OR 1 DATA...
*
************************************************************************
*
*    ONE ZERO FILTER TO MOP-UP PHASE SHIFT OF FLAT DELAY LINE TO BE PI/2 IN
*    PRODUCT FSK DEMODULATOR:
*
*    COEFFICIENTS FOR ZERO:  (B1)
*
*            BELL 103:          GAIN 103:
*
*
*    1170 HZ  0.07175           0.57731
*    2125 HZ  0.31678           1.0
*
*            V.21 :             GAIN V21:
*
*    1080 HZ  0.32796           0.69753
*    1750 HZ  0.68889           1.0
*
*    START UPDATING THE DELAY LINE
*
************************************************************************
*
        DMOV    PDEL1           ; START SHIFTING DATA IN DELAY LINE
*
************************************************************************
```

```
*
*    GENERATE THE EXACT DELAY REQUIRED FOR RECEIVER.
*
************************************************************************
*
        LAC     AGCOUT,15       ; PREPARE FOR FIR ACCUMULATION
        LT      PDEL0           ; STATE OF ONE ZERO FIR FILTER
        MPY     B1FSK           ; MULT BY COEFFICIENT
        APAC                    ; OUTPUT OF ONE ZERO FIR FILTER
        BV      RSFLAG          ; RESET OVERFLOW FLAG
*
RSFLAG  .set    $
        SACH    PDEL1,1         ; STORE AT NEXT STAGE OF DELAY FILTER
        DMOV    AGCOUT          ; SHIFT SAMPLE IN FILTER
*
************************************************************************
*
*    PREPARE TO LOW PASS FILTER THE PRODUCT.
*
************************************************************************
*
        ZAC                     ; 2ND ORDER DIRECT FORM LP FILTER
        LT      PROD            ; 500 HZ, 0.25 DB RIPPLE
        MPY     GAIN            ; COEFFICIENTS:
        ADD     LPDEL1,15       ;   A1 = 1.392 = 1.0 + 0.392
        LTA     LPDEL1          ;   A2 = -.562
        MPY     A1FSK           ;   B0 = 1
        LTA     LPDEL2          ;   B1 = 2
        MPY     A2FSK           ;   B2 = 1
        APAC                    ;   GAIN = 1 HIGH BAND
        SACH    LPDEL0,1        ;   GAIN = 0.5 LOW BAND
*
        LAC     LPDEL0,14
        ADD     LPDEL1,15       ; B1 * LPDEL1
        ADD     LPDEL2,14       ; B2 * LPDEL2
        BV      OVRFLW
        B       NOVRFL          ; CHECK FOR OVERFLOW
OVRFLW  .set    $
        ZAC                     ; CLEAR DELAY LINE
        SACL    LPDEL1
        SACL    LPDEL2
NOVRFL  .set    $
        SACH    LPFOUT,1        ; STORE FILTERED
        OUT     LPFOUT,PA6
*
        DMOV    LPDEL1
        DMOV    LPDEL0
```

```
##################################################################
*
*    DECISION: SLICER WITH DEAD ZONE IN MIDDLE (WINDOW COMPARATOR)
*
##################################################################
*
         ABS
         SUB       DZONE
         BLZ       DONE
*
         LAC       CCITT          ; CHECK FOR BELL VS CCITT
         BNZ       ITSV21
         LAC       LPFOUT
         BLEZ      HIGH
         LACK      0              ; SET BAUD DATA = 0 (0xxx)
         SACL      FSKDAT
         B         DONE
HIGH     LACK      8              ; SET BAUD DATA = 1 (1xxx)
         SACL      FSKDAT
         B         DONE
*
##################################################################
*
*    FOR V.21, NEGATIVE SIGNAL MEANS THAT WE HAVE RECEIVED A 0 AND A POSITIVE
*    SIGNAL MEANS THAT WE HAVE A 1.
*
##################################################################
*
ITSV21   .set      $
         LAC       LPFOUT
         BLEZ      LOW
         LACK      8              ; SET BAUD_DATA TO 1
         SACL      FSKDAT
         B         DONE
LOW      LACK      0              ; SET BAUD_DATA TO 0
         SACL      FSKDAT
*
##################################################################
*
*    THE FOLLOWING IMPLEMENTS FSK TIMING RECOVERY
*    ADDED BY K.H 10-3-86
*
##################################################################
*
DONE     LAC       BAUDCK         ; INCREMENT BAUD CLOCK
         ADD       ONE
         SACL      BAUDCK
         LAC       FSKDAT         ; CHECK FOR TRANSITIONS
         SUB       BDATA          ; COMPARE TO PREVIOUS DATA
         BNZ       TRAN           ; IF NOT THE SAME, HANDLE TRANSITION
         LAC       BAUDCK         ; ELSE CHECK FOR END OF BAUD
         SUB       ONE,5          ; PERIOD = 32 AND END OF BAUD
         BLEZ      ERFSK          ; IF NOT END OF BAUD, JUST RETURN
```

```
         B         CLEAR          ; ELSE OUTPUT PREVIOUS DATA AND CLEAR
*                                 ; COUNTERS
*
TRAN     LAC       COUNTR         ; LOAD TRANSITION COUNTER
         ADD       ONE            ; INCREMENT
         SACL      COUNTR
         SUB       TRANS          ; CHECK FOR 15 TRANSITIONS
         BNZ       ERFSK          ; IF NOT ENOUGH TRANSITIONS, RETURN
         LAC       BDATA          ; ELSE INVERT PREVIOUS DATA
         SUB       ONE,3
         ABS
         SACL      BDATA          ; STORE BAUD_DATA
         LACK      0F0h
         AND       STWRD          ; GET RID OF WHAREVER WAS IN LSBs
         ADD       BDATA          ; FORCE BAUD_DATE INTO BIT 3
         SACL      STWRD
*
CLEAR    OUT       STWRD,PA5      ; OUTPUT TO MICROCONTROLLER
         ZAC
         SACL      BAUDCK         ; CLEAR BAUD CLOCK
         SACL      COUNTR         ; CLEAR TRANSITION COUNTER
ERFSK    RET                      ; END OF FSK DEMOD
*
         .copy     "RTASKC.A00"
*
##################################################################
*
*    RTASKS.A00
*
*    RTASK's ALLOW LARGE TIME INTENSIVE TASKS TO BE SPLIT BETWEEN THE SAMPLING
*    PERIODS OF A BAUD. THIS IS A PRACTICAL MEANS OF IMPLEMENTING COMPLEX
*    FUNCTIONS ON THE DSP, WITH THE TIME RESTRICTION PLACED BY THE 9.6 KHZ
*    SAMPLING TRACE
*
##################################################################
*
*    RTSK10
*
##################################################################
*
RTSK10   LACK      0F0h
*
##################################################################
*
*    OUTPUT THE STATUS WORD TO THE CONTROLLER BIT 6 (IN-BAND ENERGY DETECT)
*    INDICATES DIALTONE, BUSY SIGNAL, ETC.
*
##################################################################
*
         LACK      0F0h
         AND       STWRD
         ADD       ONE
         SACL      STWRD
         OUT       STWRD,PA5
```

```
        RET
*
########################################################################
*
*   AGC FILES
*
########################################################################
*
        .copy     "AGC.A00"
*
########################################################################
*
*   AGC.ASM
*
*   FRONT END AGC FUNCTION.
*
*   THIS AGC WAS REDESIGNED TO INCORPORATE THE FREEZE OF EQUALIZATION 5/29/87
*
*   THE AVERAGE SIGNAL SQUARED IS COMPUTED BY THE MAIN PROGRAM AND STORED IN
*   AVESQR, WHICH IS CLEARED BY THIS ROUTINE AFTER AVESQR IS USED. THE
*   ROUTINE USES A WINDOW WHOSE WIDTH DEPENDS ON THE MODULATION (1200, 2400)
*   AND AN ERROR WEIGHTING WHICH ALSO DEPENDS ON THAT RATE. WE FIRST SET
*   THOSE VALUES:
*
########################################################################
*
AGC     SOVM
        LAC       GN              ; CHECK FOR AFE SWITCHING
        BNZ       SWITCH
*
########################################################################
*
*   CHECK IF 2400 AND CHANGE THOSE VALUES
*
########################################################################
*
        LACK      3
        AND       STATUS          ; IF STATUS BITS 0 AND 1 > 2 => 2400
        SUB       ONE,1
        BLEZ      AGC0            ; IF <= 2 , DO NOT MODIFY TMPO AND TMP1
*
########################################################################
*
*   FOR 2400 , 2 -> TMPO AND 1320 -> TMP1
*
########################################################################
*
        LACK      2               ; IT IS 2400
        SACL      TMPO
        LT        ONE
        MPYK      1320
        PAC
        SACL      TMP1
        B         AGC1
```

```
*
AGC0    LACK      1               ; IT IS 1200
        SACL      TMPO            ; WEIGHTING FACTOR -> TMPO
        LT        ONE
        MPYK      950
        PAC
        SACL      TMP1            ; WINDOW -> TMP1
*
########################################################################
*
*   NOW SUBTRACT REFERENCE FROM BAUD ENERGY TO GET ERROR. THE BAUD ENERGY IS
*   IN S10.5 FORMAT. THE AGC MAINTAINS THAT LEVEL AT 2.86*16 = 46.7  (H'5B6
*   IN S10.5). THE AGCREF IS THEREFORE H'5B6
*
########################################################################
*
AGC1    LAC       AVESQR
        BGEZ      CONT1           ; FOR NEGATIVE ENERGY SET TO MAX POSITIVE
        LAC       ONE,15          ; ENERGY LEVEL - FORCED SLEW MODE
        SUB       ONE
CONT1   LT        ONE
        MPYK      AGCREF          ; AGCREF = H'5B6
        SPAC                      ; AVESQR - AGCREF -> ACC
*
########################################################################
*
*   COMPARE THE ERROR TO WINDOW (TMP1).
*
*       IF ERROR > WINDOW => ERROR - WINDOW -> ERROR
*       IF -WINDOW < ERROR < WINDOW => 0 -> ERROR
*       IF ERROR < -WINDOW  => TMPO * (ERROR + WINDOW) -> ERROR
*
*   IF THE AVERAGE BAUD ENERGY IS A, THE PEAK BAUD ENERGY FOR QAM SIGNALS IS
*   1.8 A AND THE MINIMUM IS 0.2 A. THE WINDOW IS THEREFORE CHOSEN TO BE 0.8
*   A IN EITHER DIRECTION. WITH AGCREF = H'5B6, THE WINDOW IS H'492. FOR DPSK
*   SIGNALS, THE VARIATIONS IN BAUD ENERGY ARE ENTIRELY DUE TO ISI AND
*   DISTORTION AND THEREFORE THE WINDOW IS MUCH SMALLER (H'A). FIRST CHECK IF
*   ERROR > WINDOW
*
########################################################################
*
        SUB       TMP1
        SACL      TMP3            ; ERROR - WINDOW -> TMP3
        BGEZ      AGC2
*
```

```
**********************************************************************
*
*   ERROR < WINDOW => CHECK IF ERROR > -WINDOW. IN WHICH CASE, ZERO THE
*   ERROR. FIRST, ZERO THE ERROR (I.E. ASSUME ERROR > -WINDOW) AND MODIFY IF
*   WRONG ASSUMPTION.
*
**********************************************************************
*
        LARK     AR1,0
        SAR      AR1,TMP3        ; ASSUME ERROR > -WINDOW
*
**********************************************************************
*
*   CHECK ASSUMPTION
*
**********************************************************************
*
        ADD      TMP1,1          ; ERROR + WINDOW -> ACC
        BGEZ     AGC2            ; ASSUMPTION IS RIGHT
*
**********************************************************************
*
*   ERROR < -WINDOW => TMP0*(ERROR+WINDOW) -> TMP3
*
**********************************************************************
*
        SACL     TMP3
        LT       TMP3
        MPY      TMP0
        PAC
        SACL     TMP3
*
**********************************************************************
*
*   AT THIS POINT, THE WEIGHTED WINDOWED ERROR IS CONTAINED IN TMP3. WE
*   CONSIDER IT AN S.15 NUMBER AND USE IT TO UPDATE THE AGC GAIN ALPHA.
*   FIRST, WE DETERMINE WHETHER TO SLEW OR NOT. IF THE ERROR IS LARGER THAN
*   1EA6h OR SMALLER THAN F5E7h, GO INTO SLEWING MODE BY SETTING ERROR TO
*   7FFFh or 8000h RESPECTIVELY. OTHERWISE, LEAVE IT UNCHANGED.
*
*   ALSO SET STAT2[7] APPROPRIATELY TO FREEZE THE UPDATE OF THE EQUALIZER
*
*        STAT2[7]=1 UPDATE EQUALIZER
*
*        STAT2[7]=0 FREEZE EQUALIZER
*
**********************************************************************
*
*AGC2    LAC      ONE,7
*        OR       STAT2           ; ASSUME EQUALIZER UPDATE
*        SACL     STAT2
*
AGC2     LAC      TMP3
         LDPK     1
```

```
         SUB      POSSM
         BLEZ     AGC3            ; DO NOT SLEW
*
         LDPK     0               ; ENTER SLEW MODE
         LAC      ONE,15
         SUB      ONE
         SACL     TMP3            ; TMP3 <- 7FFF
*
*        LACK     7Fh
*        AND      STAT2
*        SACL     STAT2           ; FREEZE EQUALIZER UPDATE
         B        AGC4
*
AGC3     ADD      POSSM           ; ACC <- TMP3
         ADD      NEGSM
         LDPK     0
         BGEZ     AGC4            ; DO NOT SLEW
*
         LAC      ONE,15          ; ENTER SLEW MODE
         ADD      ONE
         SACL     TMP3            ; TMP3 <- 8000
*
*        LACK   7Fh
*        AND    STAT2
*        SACL   STAT2             ; FREEZE EQUALIZER
*
**********************************************************************
*
*   THE FOLLOWING LINES UPDATE THE GAIN ALPHA USING AN EXPODENTIAL  INTEGRATOR
*
*        ALPHA = ALPHA(1-K*ERROR)   (ERROR = TMP3)
*
*   WHERE ALPHA IS OF FORMAT S7.8, ERROR IS S0.15, AND K = 0.5.
*
*   ALPHA * ERROR: S7.8 * S.15 = S7.24.
*
*   BY KEEPING ACCH WITHOUT LEFT SHIFT THE MULTIPLICATION BY K IS
*   ACCOMPLISHED.
*
*   ALPHA IS UPPERBOUNDED TO 35.73 IN S7.8
*
**********************************************************************
*
AGC4     LACK     MAXALP
         TBLR     TMP0
         ZALH     ALPHA
         LT       TMP3            ; ERROR -> T
         MPY      ALPHA
         SPAC                     ; ALPHA (1 - 0.5*ERROR) -> ACC
         SACH     ALPHA
*
**********************************************************************
*
*   CHECK IF ALPHA > MAX ALPHA
```

```
*
##########################################################################
*
         SUBH      TMPO
         BLEZ      AGC5
*
         LAC       TMPO
         SACL      ALPHA
*
##########################################################################
*
*    ZERO BAUD ENERGY REGISTER
*
##########################################################################
*
AGC5:    .set      $
*ARL     OUT       ALPHA,PA2       ; ARL DIAGNOSTIC
         ZAC
         SACL      AVESQR
*
##########################################################################
*
*    ENERGY DETECT LOOP
*
*    START BY READING IN HYATERESIS COUNTER INCREMENT CONSTANT
*
##########################################################################
*
         LACK      HYSINC
         TBLR      TMP5
*
##########################################################################
*
*    CHECK IF AFE GAIN IS ON OR OFF
*
##########################################################################
*
EDT      LACK      080h
         AND       STWRD
         BZ        EDT1
*
##########################################################################
*
*    AFE GAIN IS ON, CHECK IF ENERGY DETECT IS ON (STWRD[6] = 1)
*
##########################################################################
*
```

```
         LAC       ONE,6
         AND       STWRD
         BZ        EDT01           ; IF ZERO => ENERGY IS NOT DETECTED =>
*                                  ; CHECK IF LEVEL IS LARGER THAN -43.5 DBM
*                                  ; If STWRD[6] IS ONE, CHECK IF LEVEL LESS
*                                  ; THAN -48 DBM
         LACK      THRES1
         TBLR      TMPO
         LAC       TMPO
         SUB       ALPHA
         BLZ       EDT2            ; IF < 0 THEN NO ENERGY DETECT
*
##########################################################################
*
*    CHECK IF AFE GAIN STAGE SHOULD BE BYPASSED
*
##########################################################################
*
         LACK      THRES3
         TBLR      TMPO
         LAC       TMPO
         SUB       ALPHA
         BLZ       EDT3
*
##########################################################################
*
*    IS AFE GAIN ON ?
*
##########################################################################
*
         LACK      080h
         AND       STWRD
         BZ        EDT3            ; IF GAIN IS OFF, EXIT
*
##########################################################################
*
*    BYPASS AFE GAIN
*
##########################################################################
*
         LACK      07Fh
         AND       STWRD
         SACL      STWRD
         LACK      04h
         SACL      GN
         RET
*
```

```
************************************************************************
*
*   DECREMENT HYSTERESIS COUNTER
*
************************************************************************
*
EDT2     BV       EDT21          ; CLEAR OVERFLOW BIT
EDT21    ZALH     HYST           ; HYSTERESIS COUNTER
         SUBH     TMP5           ; TMP5 = 1927 = 32768/15
         SACH     HYST
*
************************************************************************
*
*   IN CASE OF OVERFLOW, DECLARE LOSS OF ENERGY DETECT
*
************************************************************************
*
         BV       EDT02
         RET
EDT02    LACK     0BFh
         AND      STWRD
         SACL     STWRD
         RET
*
************************************************************************
*
*   FOLLOWING LINES ARE EXECUTED IF AFE GAIN IS HIGH, BUT NO ENERGY DETECT.
*   CHECK IF ALPHA < 21.28 (I.E., RECEIVE LEVEL >-43.5 DBM) AND INCREMENT
*   HYSTERESIS COUNTER IF IT IS. OTHERWISE, EXIT.
*
************************************************************************
*
EDT01    LACK     THRES2         ; 21.28 IN S7.8
         TBLR     TMP0
         LAC      TMP0
         SUB      ALPHA
         BLZ      EDT3
*
************************************************************************
*
*   ALPHA < 21.28 => INCREMENT HYSTERESIS COUNTER
*
************************************************************************
*
         BV       EDT011         ; CLEAR OVERFLOW BIT
EDT011   ZALH     HYST
         ADDH     TMP5           ; TMP5 CONTAINS INC. >FOF
         SACH     HYST
*
```

```
************************************************************************
*
*   DETECT BIT STWRD[6]  = 1.
*
************************************************************************
*
         BV       EDT04          ; IN CASE OF OVERFLOW, SET ENERGY
         RET
EDT04    LAC      ONE,6
         OR       STWRD
         SACL     STWRD
         RET
*
************************************************************************
*
*   IF AFE GAIN STAGE IS BYPASSED, CHECK LEVEL OF ALPHA
*
************************************************************************
*
EDT1     LACK     THRES5
         TBLR     TMP0
         LAC      TMP0
         SUB      ALPHA
         BGZ      EDT3
*
************************************************************************
*
*   IF ALPHA > THRES5 (20.09 IN S7.8) THEN TURN AFE GAIN STATUS WORD BIT ON.
*
************************************************************************
*
         LACK     080h
         OR       STWRD
         SACL     STWRD
         LACK     014h
         SACL     GN
EDT3     RET
*
************************************************************************
*
*   ROUTINE FOR SWITCHING THE AFE ON/OFF
*
************************************************************************
```

```
****************************************************************************
*
*    ZERO BAUD ENERGY REGISTER
*
****************************************************************************
*
SWITCH    ZAC
          SACL      AVESQR
*
          LACK      010h          ; MASK OFF UNWANTED BITS
          AND       GN
          BZ        AFEOFF
*
****************************************************************************
*
*    CHECK IF THE GAIN SHOULD BE ON
*
****************************************************************************
*
          LACK      0Fh           ; MASK OFF THE AFE ON BIT
          AND       GN
          SUB       ONE           ; DECREMENT THE COUNTER
          BZ        SWTCH1
*
          SACL      GN            ; SAVE GN VALUE
          LACK      010h          ; LOAD THE AFE ON BIT
          OR        GN
          SACL      GN            ; RESTORE AFE ON BIT
          RET
*
SWTCH1    SACL      GN            ; RESET THE GN VALUE TO ZERO
          LACK      THRES6        ; LOAD THE NEW ALPHA VALUE
          TBLR      ALPHA         ; RESET ALPHA TO 5.05
          RET
*
AFEOFF    LAC       GN
          SUB       ONE           ; DECREMENT THE COUNTER
          BZ        SWTCH2
*
          SACL      GN
          RET
*
SWTCH2    SACL      GN
          LACK      THRES4        ; LOAD NEW ALPHA VALUE
          TBLR      ALPHA         ; RESET ALPHA TO 8.98 IN S7.8
          RET
*
```