IBM

Personal Computer
Hardware Reference
Library

IBM PC*jr* Speech Attachment Technical Reference

6138761

IBM PC*jr* SPEECH ATTACHMENT

First Edition (June 1984)

This product could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of this publication.

It is possible that this material may contain references to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Products are not stocked at the address below. Requests for copies of this product and for technical information about the system should be made to your Authorized IBM Personal Computer Dealer.

THE FOLLOWING PARAGRAPH APPLIES ONLY TO THE UNITED STATES AND PUERTO RICO: A Reader's Comment Form is provided at the back of this publication. If this form has been removed, address comments to IBM Corporation, Personal Computer, P.O. Box 1328-C, Boca Raton, Florida 33432. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligations whatever.

All specifications subject to change without notice.

Contents

Notes:

Speech Attachment

The Speech Attachment is a side mounted attachment that adds speech capability to the PC*jr*. It contains a program accessible vocabulary of words, phrases, and sound effects and accepts audio input from external sources.

Description

The Speech Attachment provides two technologies for speech reproduction:

- Speech encoding (speech-to-data) and decoding (data-to-speech) using a continuously variable slope delta (CVSD) modulation technique.
- Speech synthesis using linear predictive coding (LPC).

An internal ROM module contains the BIOS necessary to control the Speech Attachment.

CVSD

CVSD allows the user to encode speech using a microphone and store the resulting uncompressed speech data in system memory (RAM), on diskette, or another storage device. The stored speech data may then be decoded with the resulting speech output available through the audio channel of the PC*jr*.

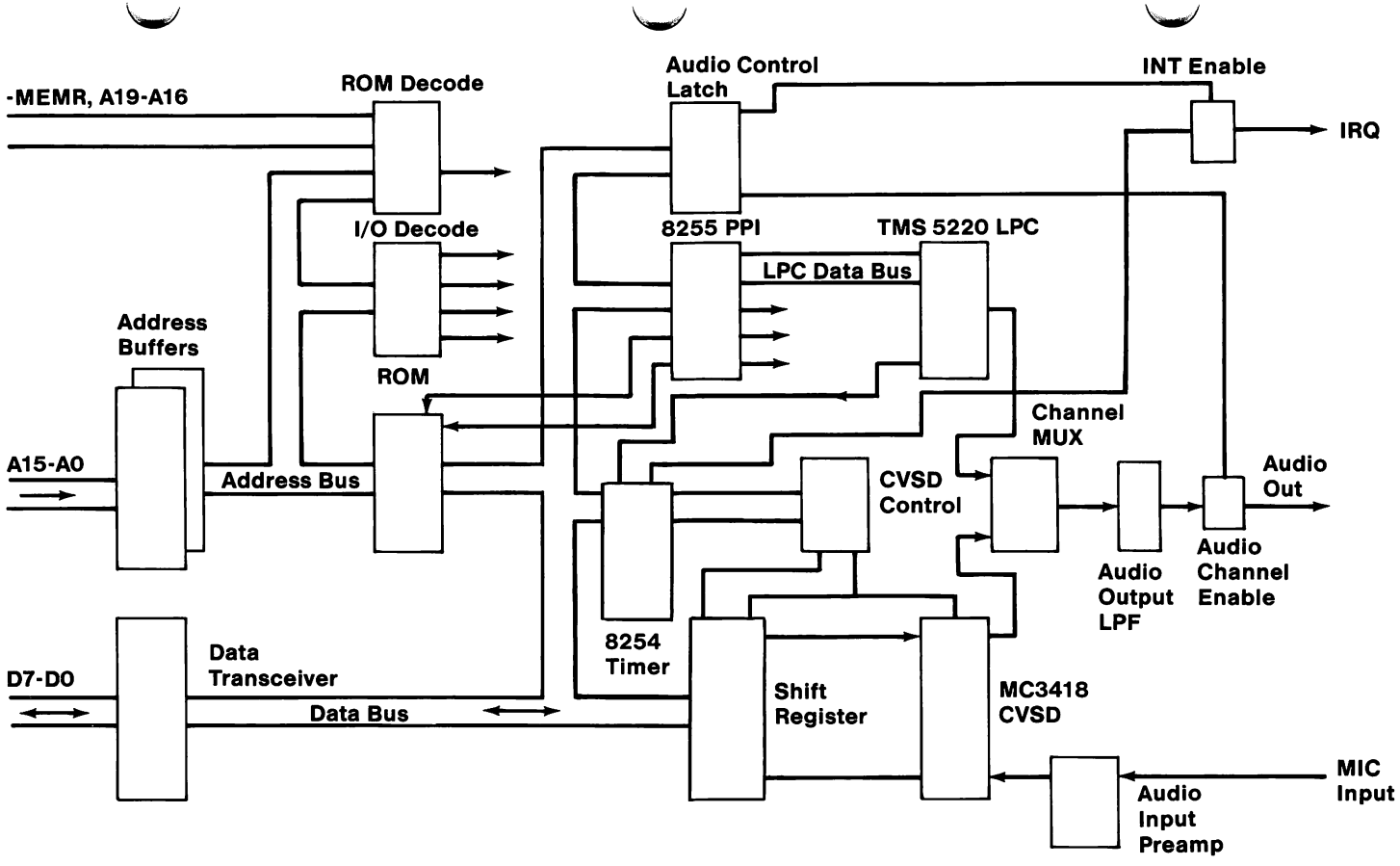
LPC

LPC synthesizes speech from compressed speech data on the internal ROM module. LPC speech data may also reside on program cartridges or may be placed in RAM from a diskette or another storage device.

Microphone

An external microphone jack is provided at the rear of the attachment.

The following is a block diagram of the Speech Attachment.


-MEMR, A19-A16
ROM Decode
Audio Control Latch
INT Enable
IRQ
I/O Decode
8255 PPI
TMS 5220 LPC
LPC Data Bus
Address Buffers
ROM
Channel MUX
A15-A0
Address Bus
CVSD Control
Audio Out
Audio Output LPF
Audio Channel Enable
8254 Timer
Shift Register
MC3418 CVSD
D7-D0
Data Transceiver
Data Bus
MIC Input
Audio Input Preamp

ROM

The Speech Attachment uses a 32K by 8 bit ROM module, which contains the standard vocabulary and BIOS support. This module appears as normal system memory at hex CE000 through CFFFF.

Vocabulary

There are 196 words, phrases, and sound effects encoded in the standard vocabulary on the ROM module of the Speech Attachment. The following is a list of these showing their corresponding index numbers.

1 danger	67 cent	131 -teen
2 time has expired	68 control	132 true
3 laughing	69 date	133 to
4 get ready	70 disk	134 -ty
5 go	71 day	135 this
6 up	72 dollar	136 twelve
7 down	73 down	137 thousand
8 left	74 do	138 that
9 right	75 excellent	139 than
10 warning	76 eleven	140 then
11 well done	77 -ez	141 time
12 gotcha	78 -ed (past tense morpheme)	142 type
13 zero		143 thing
14 one	79 echo	144 try
15 two	80 equals	145 turn
16 three	81 enter	146 the
17 four	82 end	147 twenty
18 five	83 first	148 word
19 six	84 from	149 white
20 seven	85 false	150 wait
21 eight	86 file	151 wrong
22 nine	87 fif--	152 what
23 ten	88 function	153 yes
24 a	89 go	154 you
25 b	90 green	155 yellow
26 c	91 good	156 year
27 d	92 hundred	157 your
28 e	93 hold	158 space
29 f	94 hour	159 delete
30 g	95 home	160 page
31 h	96 is	161 cursor
32 i	97 it	162 name
33 j	98 key	163 letter
34 k	99 last	164 board
35 l	100 lose	165 any
36 m	101 list	166 sign
37 n	102 less	167 spell
38 o	103 left	168 win
39 p	104 ok	169 pause
40 q	105 or	170 bar
41 r	106 period	171 insert
42 s	107 plus	172 look
43 t	108 please	173 lock
44 u	109 program	174 3 frames of silence
45 v	110 press	175 minus
46 w	111 p.m.	176 million
47 x	112 per	177 month
48 y	113 point	178 minute
49 z	114 run	179 move

(Part 1 of 2) **Standard Vocabulary**

50 an	115 read	180 no
51 again	116 red	181 negative
52 alt	117 right	182 number
53 add	118 release	183 not
54 am	119 start	184 alternate
55 are	120 stop	185 up
56 a.m.	121 -s (plural	186 -ing
57 ahead	morpheme)	187 chime 1
58 answer	122 save	188 bat hitting ball
59 back	123 second	189 ball being caught
60 by	124 sorry	190 gunshot
61 brake	125 screen	191 laser
62 at	126 score	192 phaser
63 as	127 select	193 buzz
64 and	128 -th	194 tic
65 code	129 third	195 toc
66 computer	130 thir-	196 fast chime

(Part 2 of 2) **Standard Vocabulary**

I/O Address Decode

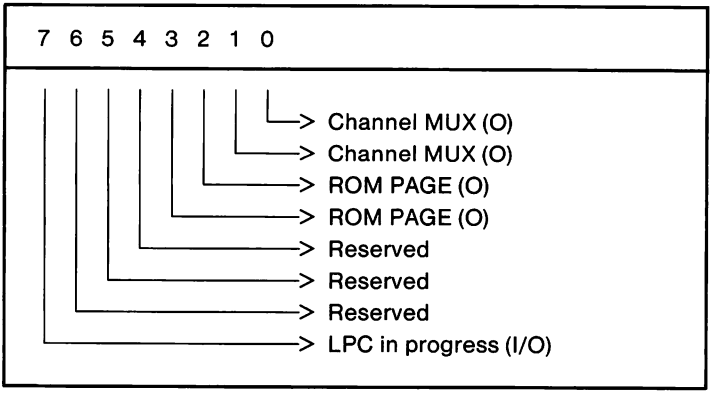
The Speech Attachment uses the following ports:

Device	Address	Port
8255 PPI	FB98	Port A Data
	FB99	Port B Data
	FB9A	Port C Data
	FB9B	Mode Register
8254 Timer	FB9C	Channel 0
	FB9D	Channel 1
	FB9E	Channel 2
	FB9F	Control Word Register
Shift Register	FF98	Shift Register
Audio Control Latch	FF9F	Audio Control Latch

I/O Port Addresses

Programmable Peripheral Interface (8255)

The Speech Attachment uses an 8255 Programmable Peripheral Interface (PPI) for control and status. The following figures show the bit definitions for ports A, B, and C of the PPI.



Port A

Bit 7 A 1 on this bit indicates that LPC is currently running in the background.

Bits 6-4 Reserved

Bits 3-2 ROM PAGE

00 Page 0

01 Page 1

10 Page 2

11 Page 3

Note: Page 0 is the default.

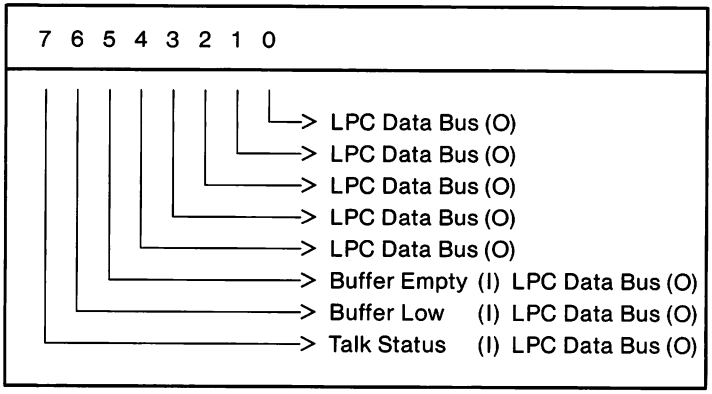
Bits 1-0 CHANNEL MUX

00 LPC

01 CVSD

10 8254 Audio

11 Test Signal



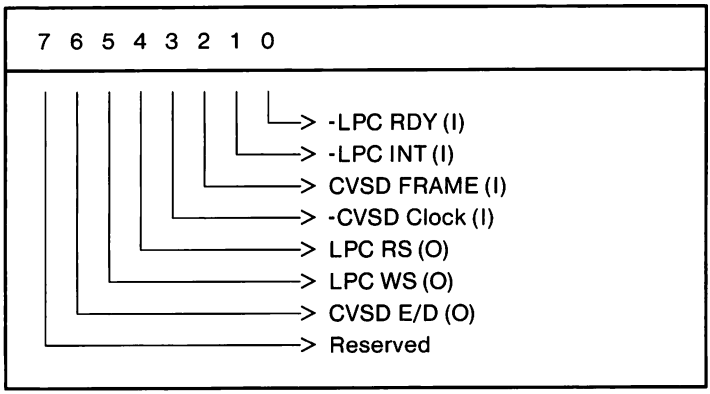
Port B

Bits 7-0 Bits 7 through 0 are used to send commands to the LPC chip. LPC status is returned in bits 7 through 5.

Port B is used as the LPC data bus. Its direction (input or output) is changed by issuing Mode commands to the PPI as follows.

Function	Port Definition	Mode	
LPC Output	A=Output	81H	Normal State
	B=Output C0-C3=Input C4-C7=Output		
LPC Input	A=Input	83H	Used when reading LPC status
	B=Input C0-C3=Input C4-C7=Output		

Note: All output signals are reset when a Mode command is issued. Mode is normally hex 81. It is only changed during LPC speech. If a particular line is needed in a non-reset state, it must be explicitly set. ROM PAGE should be set after a mode change.



Port C

Bit 7 Reserved

Bit 6 CVSD E/D—A 0 on this bit indicates CVSD decode (out, playback) and a 1 indicates CVSD encode (in, record).

Bit 5 LPC WS—A 0 on this bit indicates LPC write is inactive and a 1 indicates that it is active.

Bit 4 LPC RS—A 0 on this bit indicates LPC read is inactive and a 1 indicates that it is active.

Bit 3 -CVSD CLOCK—This signal is the inverted form of the clock used by CVSD for the bit sample rate. It is used to clock serial data into and out of the Shift Register.

Bit 2 CVSD FRAME—The negative going edge of this bit is used to read the Shift Register (S/R) during CVSD encode and the positive going edge is used to write to S/R during CVSD decode. CVSD FRAME is -CVSD CLK divided by 8.

Bit 1 -LPC INT—A 0 on this bit indicates an interrupt.

Bit 0 -LPC RDY—A 1 on this bit indicates a busy state and a 0 indicates a completed state.

Timer

The Speech Attachment uses an 8254 Timer to create the various clock signals required. CVSD circuits use channels 0 and 1 and channel 2 creates the LPC interrupt pulse. Channel 2 may also be gated onto the audio channel.

All clock signals are derived from the 4.77MHz system clock (XCLK).

Channel 0 (CVSD CLOCK)

Channel 0 has the following functions:

- Channel 0 divides the system clock signal to provide the CVSD bit sample rate. The positive going edge of this signal is used by the MC3418 to latch the digital serial data. The shift register uses the positive edge of the inverted CVSD CLOCK to clock the serial data.

- Channel 0 is inverted and is used by channel 1 to generate the CVSD FRAME signal.

  Note: The Speech Attachment initializes channel 0 in the square wave mode and holds the channel 0 gate active.

Channel 1 (CVSD FRAME)

This channel divides the inverted CVSD CLOCK by eight. It counts the CVSD CLOCK periods and goes low for one period every eight clocks. Programs use the positive edge of this signal to write data to the Shift Register during CVSD decoding. Programs poll this channel for sync signals during both CVSD encoding and decoding.

Note: The Speech Attachment initializes channel 1's divisor to 8. It also initializes channel 1 in the rate generator mode and holds the channel 1 gate active.

Channel 2 (INT CLOCK)

Channel 2 has two functions:

- Channel 2 creates an interrupt pulse during LPC operations.

- Channel 2 can be routed to the audio channel and heard on external audio devices.

These functions are selected by the state of the interrupt enable signal on the Audio Control Latch (ACL) port as follows.

-INT ENA	Channel 2 Function
1	Interrupt Mode
0	Audio Mode

When used for interrupts, the Speech Attachment initializes channel 2's divisor to 8 and channel 2 to the hardware retriggerable one-shot mode. Then the channel 2 gate goes high when -INT goes low and interrupts are enabled.

> **Note:** The -INT ENA signal on the ACL port is set active for channel 2 to function in this manner.

Interrupt Mode (Interrupt Enabled)

During LPC operations, channel 2 transforms the positive edge of the LPC interrupt signal into a short negative-going pulse. This negative-going pulse is applied to the IRQ1 line and the system senses an interrupt on the positive edge of this signal. Use of this pulse allows sharing of the system interrupt line and prevents the disabling of local interrupts from causing a false interrupt.

Audio Mode (Interrupt Disabled)

The channel 2 output may be multiplexed onto the audio channel. When the -INT ENA bit on the ACL port is cleared, the channel 2 gate is held active.

The Speech Attachment initializes channel 2 to be used in the interrupt mode.

Linear Predictive Coding (LPC)

The Speech Attachment uses a TMS5220 for LPC synthesis. This device operates at an 8kHz sample rate. Programs, driving this device, may be interrupt driven or may poll the hardware.

Interrupt The interrupt signal, -LPC INT, is enabled and is used to generate interrupts.

Polled -LPC INT is disabled.

Continuously Variable Slope Delta (CVSD) Modulation

The Speech Attachment uses a Motorola MC3418 for CVSD modulation and demodulation. This device, along with two low-pass filters, a shift register, discrete CODEC filter elements, and appropriate clock signals provides for both encode and decode CVSD functions.

Shift Register

The Speech Attachment uses the shift register to serialize and deserialize CVSD data. It is a tri-stated device capable of both serial-to-parallel and parallel-to-serial conversions.

Decode (Playback) Mode

The following is a typical programming procedure:

- Set CVSD E/D low (decode).
- Activate audio channel.

- Do for all bytes
  - Wait for positive edge of CVSD FRAME.
  - Output data byte to the shift register.
  - Do any "housekeeping" needed.
- End do

Encode (Record) Mode

The following is a typical programming procedure:

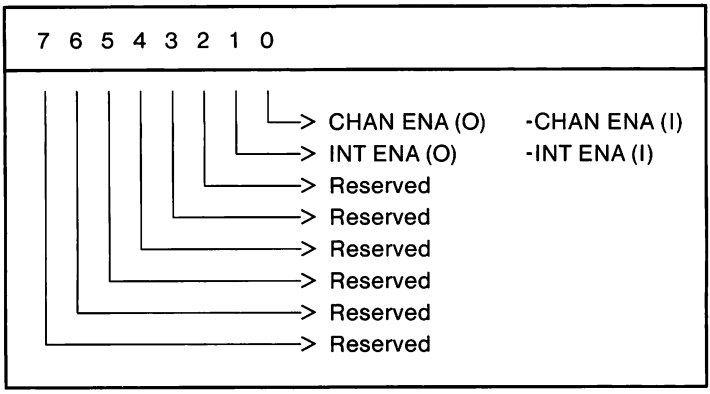
- Set CVSD E/D high (encode).
- Do for all bytes
  - Wait for the negative edge of CVSD FRAME.
  - Input data byte from the shift register.
  - Do any "housekeeping" needed.
- End do

Audio Filters

The Speech Attachment has two audio circuits: output and input. The audio output low-pass filter provides a signal compatible with the system's audio channel. The input preamp provides the amplification and filtering needed to attach a low-level microphone to the Speech Attachment.

Audio Control Latch (ACL)

The following are bit definitions for the Audio Control Latch.



Audio Control Latch

Bits 7-2 Reserved

Bit 1 INT ENA (O)

0 Disabled

1 Enabled

-INT ENA (I)

0 Enabled

1 Disabled

Bit 0 CHAN ENA (O)

0 Disabled

1 Enabled

-CHAN ENA (I)

0 Enabled

1 Disabled

Programs that use the Speech Attachment are responsible for sharing the audio channel. Before using the audio channel, the Speech Attachment BIOS must perform the following steps:

1 Issue 32 Disable Channel commands (00H) to each of the possible 32 audio control latches as shown in the following figure.

2 Read the Speech Attachment's audio control latch. -CHAN ENA should be inactive.

3 Enable the Speech Attachment's audio control channel by setting +CHAN ENA active.

4 When read, -CHAN ENA should be active.

The following shows Audio Control Latch addresses.

Device	ACL (hex)	Device	ACL (hex)
1	079F	17	879F
2	0F9F	18	8F9F
3	179F	19	979F
4	1F9F	20	9F9F
5	279F	21	A79F
6	2F9F	22	AF9F
7	379F	23	B79F
8	3F9F	24	BF9F
9	479F	25	C79F
10	4F9F	26	CF9F
11	579F	27	D79F
12	5F9F	28	DF9F
13	679F	29	E79F
14	6F9F	30	EF9F
15	779F	31	F79F
16	7F9F	32	FF9F

A program must read the Speech Attachment's ACL each time it needs the channel. If the channel is not enabled, another device has control. The program should either post an error or regain control of the channel.

Audio Multiplexers

Before the Speech Attachment begins speech synthesis, it's BIOS sets the following control devices so that audio, generated by the Speech Attachment, will be heard on the PC*jr*'s audio output.

- The Audio Channel Enable bit in the ACL
- The Audio Channel Multiplexer (points to the intended speech source)
- The PC*jr* Sound Multiplexer (points to the external audio channel)

Note: It is the responsibility of the program to restore the state of these devices.

Linear Predictive Coding (LPC)

There are two possible modes of LPC speech synthesis: background and foreground.

Background

This mode returns control to the calling program while speech synthesis is in progress with the following restrictions:

- The system cannot perform diskette or other operations that disable hardware interrupts for an extended period during speech synthesis.
- The system must not change environments during LPC background; for instance, changing from DOS to BASIC.

Foreground

In this mode control is not returned to the system until after the speech synthesis is completed.

Note: BIOS continuously polls the system during speech synthesis and updates when necessary.

Interrupt Hex 04D

Software interrupt hex 04D provides low level BIOS support for CVSD and LPC. The following lists the uses of this interrupt.

AH = 0 Reset Adapter

AH = 1 CVSD

AL = 0 CVSD Record (using speed table)

DS:SI segment:offset

BL Table speed

0 = 1800 Bytes/Sec

1 = 2400 Bytes/Sec

2 = 3000 Bytes/Sec

3 = 3600 Bytes/Sec

4 = 4200 Bytes/Sec

5 = 4800 Bytes/Sec

CX Byte count

AL = 1 CVSD Playback (using speed table)

DS:SI segment:offset

BL Table speed

0 = 1800 Bytes/Sec

1 = 2400 Bytes/Sec

2 = 3000 Bytes/Sec

3 = 3600 Bytes/Sec

4 = 4200 Bytes/Sec

5 = 4800 Bytes/Sec

CX Byte count

AL = 2 CVSD Record (using user speed)

DS:SI segment:offset

BX User speed divisor

CX Byte count

AL = 3 CVSD Playback (using user speed)

DS:SI segment:offset

BX User speed divisor

CX Byte count

AH = 2 LPC (Background)

AL = 0 LPC Status

AL = 1 LPC Speak - INTR (index)

BX Word number from index (BX≥1)

AL = 2 LPC Speak - INTR (buffer)

DS:SI Beginning of buffer (segment:offset)

CX Number of bytes in the LPC word to be spoken. CX must not be larger than 4095 bytes.

AH = 3 Polled LPC (foreground)

AL = 0 LPC Status

AL = 1 LPC Speak - INTR (index)

BX Word number from index (BX≥1)

AL = 2 LPC Speak - INTR (buffer)

DS:SI Beginning of buffer (segment:offset)

CX Number of bytes in the LPC word to be spoken. CX must not be larger than 4095 bytes.

Note: During this call, all registers except AX are preserved.

AL returns:

00H OK

01H Undefined command

02H LPC Speak in progress

03H Speech Attachment ACL error (stuck)

04H LPC index out of range

05H CVSD speed out of range

06H Timeout waiting for LPC READY

Specifications

The following are specifications of the Speech Attachment:

- Size

  Width 32 millimeters (1.26 inches)

  Depth 290 millimeters (11.42 inches)

  Height 96.5 millimeters (3.80 inches)

- Environment

  – Temperature

  System On 15.6 to 32.2 degrees C (60 to 90 degrees F)

  System Off 10 to 43 degrees C (50 to 110 degrees F)

  – Humidity

  System On 8 to 80%

  System Off 8 to 80%

- Power

  – +5Vdc with 150 milliamps maximum current

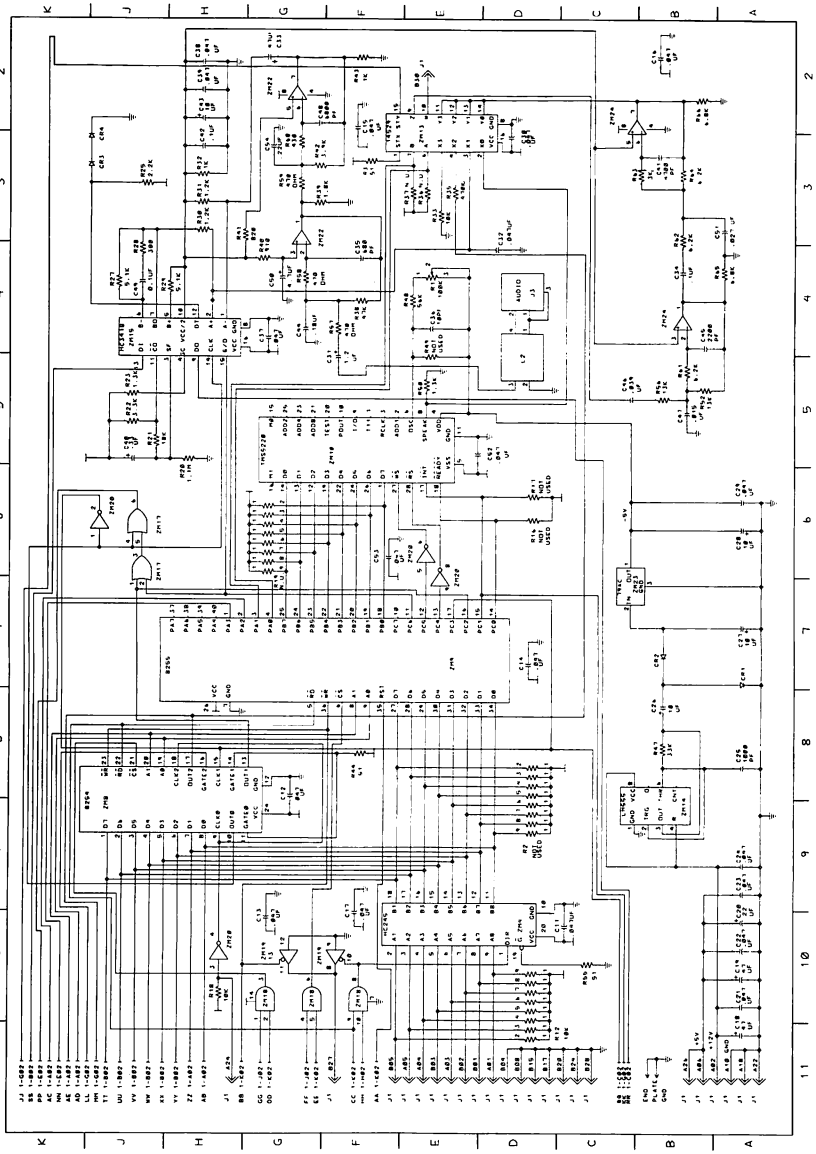
  – +12Vdc with 60 milliamps maximum current

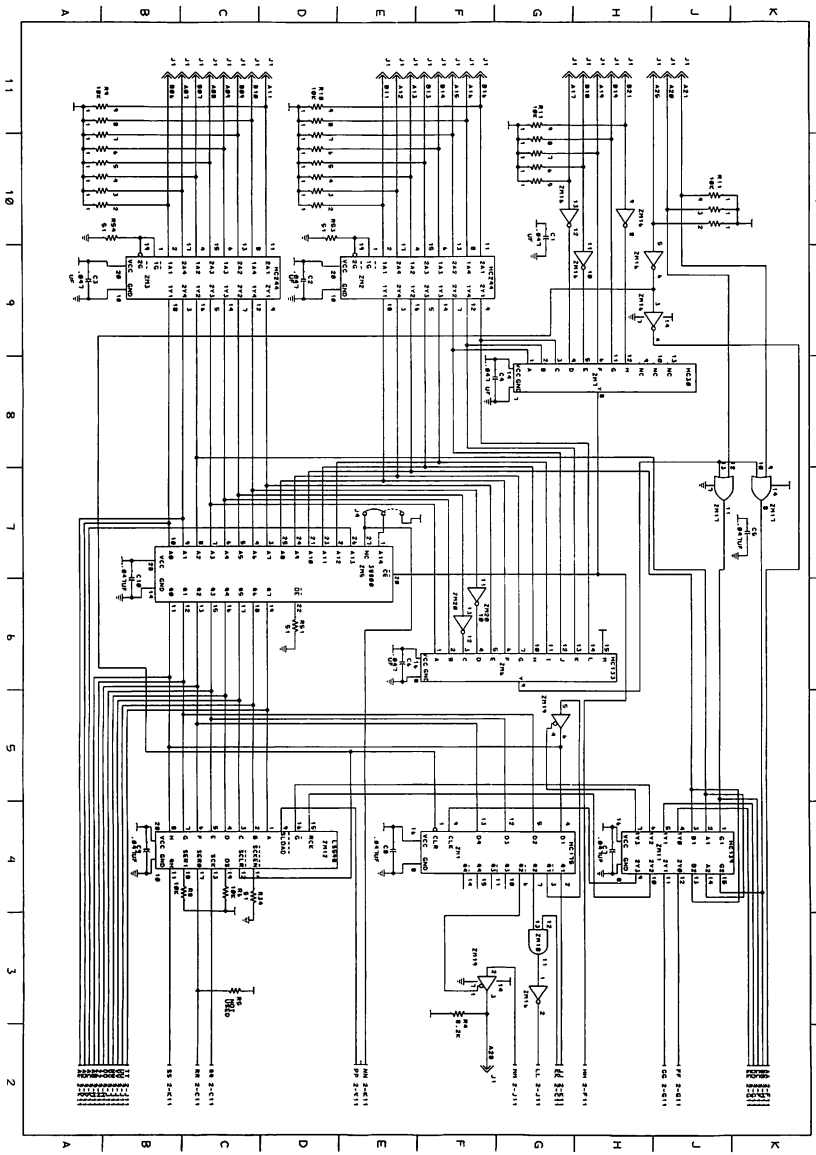
- Microphone Input

  – Miniature phone jack

  – 500 ohm nominal impedance

Logic Diagrams





BIOS Listing

```
                        ;--------------------------------------------------------
                        ; <CAVEAT EMPTOR>:
                        ;
                        ;       THE  BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH
                        ;       SOFTWARE INTERRUPTS ONLY.  ANY ADDRESSES PRESENT IN
                        ;       THE LISTINGS  ARE INCLUDED  ONLY FOR  COMPLETENESS,
                        ;       NOT FOR  REFERENCE.   APPLICATIONS WHICH  REFERENCE
                        ;       ABSOLUTE  ADDRESSES  WITHIN THIS  CODE  VIOLATE THE
                        ;       STRUCTURE AND DESIGN OF BIOS.
                        ;--------------------------------------------------------
                        ;-----------------------------
                        ;               EQUATES
                        ;-----------------------------
= 0060                  PORT_A          EQU     60H     ; 8255 PORT A ADDR
= 0038                  CPUREG          EQU     38H     ; MASK FOR CPU REG BITS
= 0007                  CRTREG          EQU     7       ; MASK FOR CRT REG BITS
= 0061                  PORT_B          EQU     61H     ; 8255 PORT B ADDR
= 0062                  PORT_C          EQU     62H     ; 8255 PORT C ADDR
= 0063                  CMD_PORT        EQU     63H
= 0089                  MODE_8255       EQU     10001001B
= 0020                  INTA00          EQU     20H     ; 8259 PORT
= 0021                  INTA01          EQU     21H     ; 8259 PORT
= 0020                  EOI             EQU     20H
= 0040                  TIMER           EQU     40H
= 0043                  TIM_CTL         EQU     43H     ; 8253 TIMER CONTROL PORT ADDR
= 0040                  TIMER0          EQU     40H     ; 8253 TIMER/CNTER 0 PORT ADDR
= 0061                  KB_CTL          EQU     61H     ; CONTROL BITS FOR KEYBOARD
= 03DA                  VGA_CTL         EQU     3DAH    ; VIDEO GATE ARRAY CONTROL PORT
= 00A0                  NMI_PORT        EQU     0A0H    ; NMI CONTROL PORT
= 00A1                  SWPORT          EQU     0A1H    ; MULTIPLEXING PORT C FOR 8255
= 0001                  SWMASK          EQU     001H    ; ENABLE SWITCHES
= 00B0                  PORT_B0         EQU     0B0H
= 03DF                  PAGREG          EQU     03DFH   ; CRT/CPU PAGE REGISTER
= 0060                  KBPORT          EQU     060H    ; KEYBOARD PORT
= 4000                  DIAG_TABLE_PTR  EQU     4000H


= 0012                  NODIAGI         EQU     18      ; NUMBER OF ENTRIES
= 2000                  MINI            EQU     2000H


                        ;-------------------------------------
                        ;       DISKETTE EQUATES
                        ;-------------------------------------
= 00F2                  NEC_CTL         EQU     0F2H    ; CONTROL PORT FOR THE DISKETTE
= 0080                  FDC_RESET       EQU     80H     ; RESETS THE NEC (FLOPPY DISK
                                                        ; CONTROLLER).  0 RESETS,
                                                        ; 1 RELEASES THE RESET
= 0020                  WD_ENABLE       EQU     20H     ; ENABLES WATCH DOG TIMER IN NEC
= 0040                  WD_STROBE       EQU     40H     ; STROBES WATCHDOG TIMER
= 0001                  DRIVE_ENABLE    EQU     01H     ; SELECTS AND ENABLES DRIVE

= 00F4                  NEC_STAT        EQU     0F4H    ; STATUS REGISTER FOR THE NEC
= 0020                  BUSY_BIT        EQU     20H     ; BIT = 0 AT END OF EXECUTION PHASE
= 0040                  DIO             EQU     40H     ; INDICATES DIRECTION OF TRANSFER
= 0080                  RQM             EQU     80H     ; REQUEST FOR MASTER

= 00F5                  NEC_DATA        EQU     0F5H    ; DATA PORT FOR THE NEC
                        ;-------------------------------------
                        ;       8088 INTERRUPT LOCATIONS
                        ;-------------------------------------
0000                    ABS0    SEGMENT AT 0
0008                            ORG     2*4
0008                    NMI_PTR         LABEL   WORD
000C                            ORG     3*4
000C                    INT3_PTR        LABEL   WORD
0014                            ORG     5*4
0014                    INT5_PTR        LABEL   WORD
0020                            ORG     8*4
0020                    INT_PTR         LABEL   DWORD
0040                            ORG     10H*4
0040                    VIDEO_INT       LABEL   WORD
0070                            ORG     1CH*4
0070                    INT1C_PTR       LABEL   WORD
0074                            ORG     1DH*4
0074                    PARM_PTR        LABEL   DWORD   ; POINTER TO VIDEO PARMS
0060                            ORG     18H*4
0060                    BASIC_PTR       LABEL   WORD    ; ENTRY POINT FOR CASSETTE BASIC
0078                            ORG     01EH*4          ; INTERRUPT 1EH
0078                    DISK_POINTER    LABEL   DWORD
007C                            ORG     01FH*4          ; LOCATION OF POINTER
007C                    EXT_PTR LABEL   DWORD           ; POINTER TO EXTENSION
0110                            ORG     044H*4
0110                    CSET_PTR        LABEL   DWORD   ; POINTER TO DOT PATTERNS
011C                            ORG     047H*4
011C                    KEYBRD_PTR      LABEL   DWORD   ; POINTER TO KEYBOARD TABLES
0120                            ORG     048H*4
0120                    KEY62_PTR       LABEL   WORD    ; POINTER TO 62 KEY KEYBOARD CODE
0124                            ORG     049H*4
0124                    EXST            LABEL   WORD    ; POINTER TO EXT. SCAN TABLE
0214                            ORG     085H*4
0214                    EMPTR   LABEL   WORD


0400                            ORG     400H
0400                    DATA_AREA       LABEL   BYTE    ; ABSOLUTE LOCATION OF DATA SEGMENT
0400                    DATA_WORD       LABEL   WORD
7C00                            ORG     7C00H
7C00                    BOOT_LOCN       LABEL   FAR
7C00                    ABS0    ENDS
                        ;--------------------------------------------
                        ; STACK -- USED DURING INITIALIZATION ONLY
```

```
                        ;--------------------------------------------
0000                    STACK   SEGMENT AT 30H
0000    80 [                    DW      128 DUP(?)
        ????
           ]

0100                    TOS     LABEL   WORD
0100                    STACK   ENDS
                        ;--------------------------------------------
                        ;       ROM BIOS DATA AREAS
                        ;--------------------------------------------

0000                    DATA    SEGMENT AT 40H
0000    04 [            RS232_BASE      DW      4 DUP(?) ; ADDRESSES OF RS232 ADAPTERS
        ????
           ]

0008    04 [            PRINTER_BASE    DW      4 DUP(?) ; ADDRESSES OF PRINTERS
        ????
           ]

0010  ????              EQUIP_FLAG      DW      ?       ; INSTALLED HARDWARE
0012  ??                KBD_ERR         DB      ?       ; COUNT OF KEYBOARD TRANSMIT ERRORS
0013  ????              MEMORY_SIZE     DW      ?       ; USABLE MEMORY SIZE IN K BYTES
0015  ????              TRUE_MEM        DW      ?       ; REAL MEMORY SIZE IN K BYTES
                        ;--------------------------------------------
                        ;       KEYBOARD DATA AREAS
                        ;--------------------------------------------
0017  ??                KB_FLAG         DB      ?
                        ;----- SHIFT FLAG EQUATES WITHIN KB_FLAG
= 0040                  CAPS_STATE      EQU     40H     ; CAPS LOCK STATE HAS BEEN TOGGLED
= 0020                  NUM_STATE       EQU     20H     ; NUM LOCK STATE HAS BEEN TOGGLED
= 0008                  ALT_SHIFT       EQU     08H     ; ALTERNATE SHIFT KEY DEPRESSED
= 0004                  CTL_SHIFT       EQU     04H     ; CONTROL SHIFT KEY DEPRESSED
= 0002                  LEFT_SHIFT      EQU     02H     ; LEFT SHIFT KEY DEPRESSED
= 0001                  RIGHT_SHIFT     EQU     01H     ; RIGHT SHIFT KEY DEPRESSED

0018  ??                KB_FLAG_1       DB      ?       ; SECOND BYTE OF KEYBOARD STATUS
= 0080                  INS_SHIFT       EQU     80H     ; INSERT KEY IS DEPRESSED
= 0040                  CAPS_SHIFT      EQU     40H     ; CAPS LOCK KEY IS DEPRESSED
= 0020                  NUM_SHIFT       EQU     20H     ; NUM LOCK KEY IS DEPRESSED
= 0010                  SCROLL_SHIFT    EQU     10H     ; SCROLL LOCK KEY IS DEPRESSED
= 0008                  HOLD_STATE      EQU     08H     ; SUSPEND KEY HAS BEEN TOGGLED
= 0004                  CLICK_ON        EQU     04H     ; INDICATES THAT AUDIO FEEDBACK IS
                                                        ; ENABLED
= 0002                  CLICK_SEQUENCE  EQU     02H     ; OCURRNCE OF ALT-CTRL-CAPSLOCK HAS


                                                        ; OCCURED

0019  ??                ALT_INPUT       DB      ?       ; STORAGE FOR ALTERNATE KEYPAD
                                                        ; ENTRY
001A  ????              BUFFER_HEAD     DW      ?       ; POINTER TO HEAD OF KEYBOARD BUFF
001C  ????              BUFFER_TAIL     DW      ?       ; POINTER TO TAIL OF KEYBOARD BUFF
001E    10 [            KB_BUFFER       DW      16 DUP(?) ; ROOM FOR 15 ENTRIES
        ????
           ]

                        ;------ HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY

= 0045                  NUM_KEY         EQU     69      ; SCAN CODE FOR NUMBER LOCK

= 0046                  SCROLL_KEY      EQU     70      ; SCROLL LOCK KEY
= 0038                  ALT_KEY         EQU     56      ; ALTERNATE SHIFT KEY SCAN CODE
= 001D                  CTL_KEY         EQU     29      ; SCAN CODE FOR CONTROL KEY
= 003A                  CAPS_KEY        EQU     58      ; SCAN CODE FOR SHIFT LOCK
= 002A                  LEFT_KEY        EQU     42      ; SCAN CODE FOR LEFT SHIFT
= 0036                  RIGHT_KEY       EQU     54      ; SCAN CODE FOR RIGHT SHIFT
= 0052                  INS_KEY         EQU     82      ; SCAN CODE FOR INSERT KEY
= 0053                  DEL_KEY         EQU     83      ; SCAN CODE FOR DELETE KEY
                        ;--------------------------------------------
                        ;       DISKETTE DATA AREAS
                        ;--------------------------------------------
003E  ??                SEEK_STATUS     DB      ?       ; DRIVE RECALIBRATION STATUS
                                                        ; BIT 0 = DRIVE NEEDS RECAL BEFORE
                                                        ;   NEXT SEEK IF BIT IS = 0
003F  ??                MOTOR_STATUS    DB      ?       ; MOTOR STATUS
                                                        ; BIT 0 = DRIVE 0 IS CURRENTLY
                                                        ;   RUNNING
0040  ??                MOTOR_COUNT     DB      ?       ; TIME OUT COUNTER FOR DRIVE
                                                        ; TURN OFF
= 0025                  MOTOR_WAIT      EQU     37      ; 2 SECS OF COUNTS FOR MOTOR
                                                        ; TURN OFF
0041  ??                DISKETTE_STATUS DB      ?       ; RETURN CODE STATUS BYTE
= 0080                  TIME_OUT        EQU     80H     ; ATTACHMENT FAILED TO RESPOND
= 0040                  BAD_SEEK        EQU     40H     ; SEEK OPERATION FAILED
= 0020                  BAD_NEC         EQU     20H     ; NEC CONTROLLER HAS FAILED
= 0010                  BAD_CRC         EQU     10H     ; BAD CRC ON DISKETTE READ
= 0009                  DMA_BOUNDARY    EQU     09H     ; ATTEMPT TO DMA ACROSS 64K
                                                        ; BOUNDARY
= 0008                  BAD_DMA         EQU     08H     ; DMA OVERRUN ON OPERATION
= 0004                  RECORD_NOT_FND  EQU     04H     ; REQUESTED SECTOR NOT FOUND
= 0003                  WRITE_PROTECT   EQU     03H     ; WRITE ATTEMPTED ON WRITE
                                                        ; PROTECTED DISK
= 0002                  BAD_ADDR_MARK   EQU     02H     ; ADDRESS MARK NOT FOUND
= 0001                  BAD_CMD         EQU     01H     ; BAD COMMAND GIVEN TO DISKETTE I/O

0042    07 [            NEC_STATUS      DB      7 DUP(?) ; STATUS BYTES FROM NEC
           ??
              ]



= 0020                  SEEK_END        EQU     20H
= 012C                  THRESHOLD       EQU     300     ; NUMBER OF TIMER-0 TICKS TILL
                                                        ; ENABLE
= 00AF                  PARM0           EQU     0AFH    ; PARAMETER 0 IN THE DISK_PARM
                                                        ; TABLE
= 0003                  PARM1           EQU     3       ; PARAMETER 1
= 0019                  PARM9           EQU     25      ; PARAMETER 9
= 0004                  PARM10          EQU     4       ; PARAMETER 10
                        ;--------------------------------------------
                        ;       VIDEO DISPLAY DATA AREA
                        ;--------------------------------------------
0049  ??                CRT_MODE        DB      ?       ; CURRENT CRT MODE
004A  ????              CRT_COLS        DW      ?       ; NUMBER OF COLUMNS ON SCREEN
004C  ????              CRT_LEN         DW      ?       ; LENGTH OF REGEN IN BYTES
004E  ????              CRT_START       DW      ?       ; STARTING ADDRESS IN REGEN BUFFER
0050    08 [            CURSOR_POSN     DW      8 DUP(?) ; CURSOR FOR EACH OF UP TO 8 PAGES
        ????
           ]

0060  ????              CURSOR_MODE     DW      ?       ; CURRENT CURSOR MODE SETTING
```

```
0062  ??                    ACTIVE_PAGE     DB      ?       ; CURRENT PAGE BEING DISPLAYED
0063  ????                  ADDR_6845       DW      ?       ; BASE ADDRESS FOR ACTIVE DISPLAY
                                                            ; CARD
0065  ??                    CRT_MODE_SET    DB      ?       ; CURRENT SETTING OF THE
                                                            ; CRT MODE REGISTER
0066  ??                    CRT_PALLETTE    DB      ?       ; CURRENT PALETTE MASK SETTING
                            ;--------------------------------------
                            ;       CASSETTE DATA AREA
                            ;--------------------------------------
0067  ????                  EDGE_CNT        DW      ?       ; TIME COUNT AT DATA EDGE
0069  ????                  CRC_REG         DW      ?       ; CRC REGISTER
006B  ??                    LAST_VAL        DB      ?       ; LAST INPUT VALUE
                            ;--------------------------------------
                            ;       TIMER DATA AREA
                            ;--------------------------------------
006C  ????                  TIMER_LOW       DW      ?       ; LOW WORD OF TIMER COUNT
006E  ????                  TIMER_HIGH      DW      ?       ; HIGH WORD OF TIMER COUNT
0070  ??                    TIMER_OFL       DB      ?       ; TIMER HAS ROLLED OVER SINCE LAST
                                                            ; READ
                            ;--------------------------------------
                            ;       SYSTEM DATA AREA
                            ;--------------------------------------
0071  ??                    BIOS_BREAK      DB      ?       ; BIT 7=1 IF BREAK KEY HAS BEEN HIT
0072  ????                  RESET_FLAG      DW      ?       ; WORD=1234H IF KEYBOARD RESET
                                                            ; UNDERWAY
                            ;--------------------------------------
                            ;       EXTRA DISKETTE DATA AREAS
                            ;--------------------------------------
0074  ??                    TRACK0          DB      ?
0075  ??                    TRACK1          DB      ?
0076  ??                    TRACK2          DB      ?
0077  ??                                    DB      ?
= 0020                      NL62            EQU     20H     ; 62 KEY NUM LOCK STATE



                                            40H     ; RESERVED
                                            80H     ; RESERVED FOR FUTURE USE
                            ;--------------------------------------
                            ;       PRINTER AND RS232 TIME-OUT VARIABLES
                            ;--------------------------------------
0078      04 [              PRINT_TIM_OUT   DB      4 DUP(?)
            ??
          ]

007B                                        ORG     $-1
007B  ??                    DK_INDEX        DB      ?
= 0080                      MATCH_BIT       EQU     80H     ; INDICATES FIRST KEYSTROKE IN
                                                            ; ..A DEAD KEY SEQUENCE
007C      04 [              RS232_TIM_OUT   DB      4 DUP(?)
            ??
          ]


                            ;--------------------------------------
                            ;       ADDITIONAL KEYBOARD DATA AREA
                            ;--------------------------------------
0080  ????                  BUFFER_START    DW      ?
0082  ????                  BUFFER_END      DW      ?
0084  ??                    INTR_FLAG       DB      ?       ; FLAG TO INDICATE AN INTERRUPT
                                                            ; HAPPENED
                            ;--------------------------------------
                            ;  62 KEY KEYBOARD DATA AREA
                            ;--------------------------------------
0085  ??                    CUR_CHAR        DB      ?       ; CURRENT CHARACTER FOR TYPAMATIC
0086  ??                    VAR_DELAY       DB      ?       ; DETERMINES WHEN INITIAL DELAY IS
                                                            ; OVER
= 000F                      DELAY_RATE      EQU     0FH     ; INCREASES INITIAL DELAY
0087  ??                    CUR_FUNC        DB      ?       ; CURRENT FUNCTION
0088  ??                    KB_FLAG_2       DB      ?       ; 3RD BYTE OF KEYBOARD FLAGS
= 0004                      RANGE           EQU     4       ; NUMBER OF POSITIONS TO SHIFT
                                                            ; DISPLAY
                            ;--------------------------------------
                            ;  BIT ASSIGNMETS FOR KB_FLAG_2
                            ;--------------------------------------
= 0080                      FN_FLAG         EQU     80H
= 0040                      FN_BREAK        EQU     40H
= 0020                      FN_PENDING      EQU     20H
= 0010                      FN_LOCK         EQU     10H
= 0008                      TYPE_OFF        EQU     08H
= 0004                      HALF_RATE       EQU     04H
= 0002                      INIT_DELAY      EQU     02H
= 0001                      PUTCHAR         EQU     01H
0089  ??                    HORZ_POS        DB      ?       ; CURRENT VALUE OF HORIZONTAL
                                                            ; START PARM
008A  ??                    PAGDAT          DB      ?       ; IMAGE OF DATA WRITTEN TO PAGREG
008B                        DATA    ENDS
                            ;--------------------------------------
                            ;       EXTRA DATA AREA
                            ;--------------------------------------



0000                        XXDATA  SEGMENT AT 50H
0000  ??                    STATUS_BYTE     DB      ?
                            ; THE FOLLOWING AREA IS USED ONLY DURING DIAGNOSTICS
                            ; (POST AND ROM RESIDENT)
0001  ??                    DCP_MENU_PAGE   DB      ?       ; TO CURRENT PAGE FOR DIAG. MENU
0002  ????                  DCP_ROW_COL     DW      ?       ; CURRENT ROW/COLUMN COORDINATES
                                                            ; FOR DIAG MENU
0004  ??                    WRAP_FLAG       DB      ?       ; INTERNAL/EXTERNAL 8250 WRAP
                                                            ; INDICATOR
0005  ??                    MFG_TST         DB      ?       ; INITIALIZATION FLAG
0006  ????                  MEM_TOT         DW      ?       ; WORD EQUIV. TO HIGHEST SEGMENT IN
                                                            ; MEMORY
0008  ????                  MEM_DONES       DW      ?       ; CURRENT SEGMENT VALUE FOR
                                                            ; BACKGROUND MEM TEST
000A  ????                  MEM_DONEO       DW      ?       ; CURRENT OFFSET VALUE FOR
                                                            ; BACKGROUND MEM TEST
000C  ????                  INT1CO          DW      ?       ; SAVE AREA FOR INTERRUPT 1C
                                                            ; ROUTINE
000E  ????                  INT1CS          DW      ?
0010  ??                    MENU_UP         DB      ?       ; FLAG TO INDICATE WHETHER MENU IS
                                                            ; ON SCREEN (FF=YES, 0=NO)
0011  ??                    DONE128         DB      ?       ; COUNTER TO KEEP TRACK OF 128 BYTE
                                                            ; BLOCKS TESTED BY BGMEM
0012  ????                  KBDONE          DW      ?       ; TOTAL K OF MEMORY THAT HAS BEEN
                                                            ; TESTED BY BACKGROUND MEM TEST
                            ;--------------------------------------
                            ;       POST DATA AREA
                            ;--------------------------------------
0014  ????                  IO_ROM_INIT     DW      ?       ; POINTR TO OPTIONAL I/O ROM INIT
                                                            ; ROUTINE
0016  ????                  IO_ROM_SEG      DW      ?       ; POINTER TO IO ROM SEGMENT
0018  ??                    POST_ERR        DB      ?       ; FLAG TO INDICATE ERROR OCCURRED
```

```
                                                        ; DURING POST
0019     09 [                   MODEM_BUFFER    DB      9 DUP(?) ; MODEM RESPONSE BUFFER
              ??
                 ]

                                                        ; (MAX 9 CHARS)
0022  ????                      MFG_RTN         DW      ?       ; POINTER TO MFG. OUTPUT ROUTINE
0024  ????                                      DW      ?
                                ;-----------------------------------------
                                ;       SERIAL PRINTER DATA
                                ;-----------------------------------------
0026  ????                      SP_FLAG         DW      ?
0028  ??                        SP_CHAR         DB      ?
0029  ??                        DCP_RUNNING     DB      ?       ; FLAG TO TELL E_MSG WHERE IT IS
                                                                ; BEING CALLED FROM
002A                            XXDATA  ENDS
                                ;-----------------------------------------
                                ;       DISKETTE DATA AREA
                                ;-----------------------------------------
0000                            DKDATA  SEGMENT AT 60H
                                ;       FORMAT ID


0000     08 [                   TK_HD_SC        DB      8 DUP(0,0,0,0)  ;TRACK,HEAD,SECTOR
              00
              00
              00
              00
                 ]

                                                        ; SECTOR
                                ;       BUFFER FOR READ AND WRITE OPERATION
= 0200                          DK_BUF_LEN      EQU     512     ; 512 BYTES/SECTOR
0020  ??                        DIAG_RETRY      DB      ?
0021  0200 [                   READ_BUF        DB      DK_BUF_LEN DUP(0)
              00
                 ]

0221  0100 [                   WRITE_BUF       DB      (DK_BUF_LEN/2) DUP(6DH,0BH)
              6D
              0B
                 ]

0421     07 [                   DNEC_STATUS     DB      7 DUP(?)
              ??
                 ]

0428                            DKDATA  ENDS
                                ;-----------------------------------------
                                ;       VIDEO DISPLAY BUFFER
                                ;-----------------------------------------
0000                            VIDEO_RAM       SEGMENT AT 0B800H
0000  4000 [                           DB      16384 DUP(?)
              ??
                 ]

4000                            VIDEO_RAM       ENDS

                                ;*******************************************************
                                ; TKR_SEG1.INC
                                ;       THIS MODULE CONTAINS SEGMENT DEFINITION 1.

                                ;       DUMMY IS THE SEGMENT LOCATED AT ABLOLUTE
                                ;       LOCATION 0 HOLDING THE INTERRUPT VECTORS.
                                ;*******************************************************

0000                            DUMMY   SEGMENT AT 0

0024                                    ORG     09H*4           ;INT 9H
0024                            LPC_PTR LABEL   WORD            ;POINTER TO LPC CODE
0024                            OLDKBD_PTR LABEL WORD           ;OLD POINTER TO KBD CODE

0134                                    ORG     04DH*4          ;INT 4DH
0134                            TALKER_PTR LABEL WORD           ;POINTER TO BIOS CODE

0138                                    ORG     04EH*4          ;INT 4EH
0138                            KBD_PTR LABEL   WORD            ;NEW POINTER TO KBD CODE



013C                                    ORG     04FH*4          ;INT 4FH
013C                            BFR_PTR LABEL   WORD            ;POINTER TO LPC BUFR, COUNTER

0248                                    ORG     092H*4          ;INT 92H
0248                            TALKER_DIAG_PTR LABEL   WORD    ;POINTER FOR DIAG CODE ENTRY

0248                            DUMMY   ENDS

                                ;8255 PORTS - I/O ADDRESSES

= FB98                          PORTA   EQU     0FB98H          ;8255 PORT A
= FB99                          PORTB   EQU     0FB99H          ;8255 PORT B
= FB9A                          PORTC   EQU     0FB9AH          ;8255 PORT C
= FB9B                          CWREG   EQU     0FB9BH          ;8255 CONTROL WORD REGISTER

                                ;---------------------------------------------------------------
                                ;PORT A  (OUTPUT)
                                ;---------------------------------------------------------------
                                ;
                                ;       ┌─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┐
                                ;PORT A: │ PA7 │ PA6 │ PA5 │ PA4 │ PA3 │ PA2 │ PA1 │ PA0 │
                                ;       └─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┘

                                ;PA7 - LPC SPEAK IN PROGRESS FLAG

= 0080                          TALK_LPC        EQU     10000000B       ;LPC SPEAK IN PROGRESS FLAG

                                ;PA6 - UNUSED

                                ;PA5,PA4 - RESERVED

                                ;PA3,PA2 - ROM PAGE

= 0000                          PAGE0   EQU     00000000B       ;ROM PAGE 0
= 0004                          PAGE1   EQU     00000100B       ;ROM PAGE 1
= 0008                          PAGE2   EQU     00001000B       ;ROM PAGE 2
= 000C                          PAGE3   EQU     00001100B       ;ROM PAGE 3

                                ;PA1,PA0 - CHANNEL MUX

= 0000                          LPC     EQU     00000000B       ;LPC
```

```
= 0001                  CVSD        EQU     00000001B   ;CVSD
= 0002                  AUDIO8254   EQU     00000010B   ;8254 AUDIO

= 00F3                  CLR_PAGE    EQU     11110011B   ;CLEAR ROM PAGE
= 00FC                  CLR_MUX     EQU     11111100B   ;CLEAR CHANNEL MUX
= 00F0                  CLR_MXPG    EQU     11110000B   ;CLEAR ROM PAGE & CH MUX

                        ;-----------------------------------------------------------------



                        ;PORT B  (INPUT/OUTPUT)
                        ;-----------------------------------------------------------------
                        ;
                        ;                   ____ ____ ____ ____ ____ ____ ____ ____
                        ;PORT B:           | PB7| PB6| PB5| PB4| PB3| PB2| PB1| PB0|
                        ;                  |____|____|____|____|____|____|____|____|

                        ;PB7-PB0  - LPC BUS

                        ;                   ____ ____ ____ ____ ____ ____ ____ ____
                        ;PORT C:           | PC7| PC6| PC5| PC4| PC3| PC2| PC1| PC0|
                        ;                  |____|____|____|____|____|____|____|____|

                        ; ---> PORT C - UPPER  (OUTPUT)

                        ;PC7 - UNUSED

                        ;PC6 - CVSD ENCODE/DECODE

                        ;PC5 - LPC WRITE

                        ;PC4 - LPC READ

                        ; ---> PORT C - LOWER  (INPUT)

                        ;PC3 - UNUSED

                        ;PC2 - CVSD ENCODE/DECODE

= 0004                  FRAME_HI    EQU     00000100B   ;CVSD FRAME HI              (+)

                        ;PC1 - LPC INTERRUPT

= 0002                  LPC_INT     EQU     00000010B   ;LPC INTERRUPT              (-)

                        ;PC0 - LPC READY

= 0001                  LPC_READY   EQU     00000001B   ;LPC BUSY  (0 => READY)     (-)

                        ;MODE DEFINITION FORMAT
                        ;                   ____ ____ ____ ____ ____ ____ ____ ____
                        ;CONTROL WORD REG:  | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
                        ;                  |____|____|____|____|____|____|____|____|

                        ;D7 - MODE SET FLAG
= 0080                  MODE_SET    EQU     10000000B   ;MODE SET FLAG (1 = ACTIVE)

                        ;D6,D5 - PORT A - MODE SELECTION
= 0000                  MODE0_A     EQU     00000000B   ;PORT A MODE 0
= 0020                  MODE1_A     EQU     00100000B   ;PORT A MODE 1


= 0040                  MODE2_A     EQU     01000000B   ;PORT A MODE 2

                        ;D4 - PORT A
= 0000                  PORTA_OUT   EQU     00000000B   ;PORT A OUTPUT
= 0010                  PORTA_IN    EQU     00010000B   ;PORT A INPUT

                        ;D3 - PORT C (UPPER)
= 0000                  PORTCU_OUT  EQU     00000000B   ;PORT C - UPPER OUTPUT
= 0008                  PORTCU_IN   EQU     00001000B   ;PORT C - UPPER INPUT

                        ;D2 - MODE SELECTION - PORT B
= 0000                  MODE0_B     EQU     00000000B   ;PORT B MODE 0
= 0004                  MODE1_B     EQU     00000100B   ;PORT B MODE 1

                        ;D1 - PORT B
= 0000                  PORTB_OUT   EQU     00000000B   ;PORT B OUTPUT
= 0002                  PORTB_IN    EQU     00000010B   ;PORT B INPUT

                        ;D0 - PORT C (LOWER)
= 0000                  PORTCL_OUT  EQU     00000000B   ;PORT C - LOWER OUTPUT
= 0001                  PORTCL_IN   EQU     00000001B   ;PORT C - LOWER INPUT

= 0080                  LPC_IN0     EQU     MODE_SET+MODE0_A+PORTA_OUT+PORTCU_OUT
= 0003                  LPC_IN1     EQU     MODE0_B+PORTB_IN+PORTCL_IN
= 0083                  LPC_IN      EQU     LPC_IN0+LPC_IN1

= 0080                  LPC_OUT0    EQU     MODE_SET+MODE0_A+PORTA_OUT+PORTCU_OUT
= 0001                  LPC_OUT1    EQU     MODE0_B+PORTB_OUT+PORTCL_IN
= 0081                  LPC_OUT     EQU     LPC_OUT0+LPC_OUT1

                        ;BIT SET/RESET FORMAT
                        ;                   ____ ____ ____ ____ ____ ____ ____ ____
                        ;CONTROL WORD REG:  | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
                        ;                  |____|____|____|____|____|____|____|____|

                        ;D7 - BIT SET/RESET FLAG     (0 = ACTIVE)

                        ;D6,D5,D4 - UNUSED

                        ;D3,D2,D1 - BIT SELECT
                        ;           100  =>  BIT 4 - LPC READ
                        ;           101  =>  BIT 5 - LPC WRITE
                        ;           110  =>  BIT 6 - CVSD DECODE/ENCODE
                        ;           111  =>  BIT 7 - UNUSED

                        ;D0 - BIT SET/RESET   (1 = BIT SET)

= 0008                  LPCR_OFF    EQU     00001000B   ;LPC READ OFF
= 0009                  LPCR_ON     EQU     00001001B   ;LPC READ ON

= 000A                  LPCW_OFF    EQU     00001010B   ;LPC WRITE OFF
```

```
= 000B                  LPCW_ON     EQU     00001011B     ;LPC WRITE ON

= 000D                  ENCODE      EQU     00001101B     ;ENCODE (RECORD)
= 000C                  DECODE      EQU     00001100B     ;DECODE (SPEAK)


                        ;8254 PORTS - I/O ADDRESSES

= FB9C                  CVSD_CLK    EQU     0FB9CH        ;8254 CTR 0 - CVSD BIT CLOCK
= FB9D                  CVSD_FRAME  EQU     0FB9DH        ;8254 CTR 1 - CVSD FRAME
= FB9E                  INTR_CTR    EQU     0FB9EH        ;8254 CTR 2 - INTR PULSE CTR
= FB9F                  CWR_8254    EQU     0FB9FH        ;8254 CONTROL WORD REGISTER

                        ;CONTROL WORD FORMAT
                        ;
                        ;
                        ;CONTROL WORD REG:     | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
                        ;
                        ;                      |____|____|____|____|____|____|____|____|


                        ;D7,D6 - SELECT COUNTER

= 0000                  CTR0        EQU     00000000B     ;SELECT COUNTER 0
= 0040                  CTR1        EQU     01000000B     ;SELECT COUNTER 1
= 0080                  CTR2        EQU     10000000B     ;SELECT COUNTER 2
= 00C0                  RD_BACK     EQU     11000000B     ;READ BACK COMMAND

                        ;D5,D4 - READ/WRITE

= 0000                  CTR_LATCH   EQU     00000000B     ;COUNTER LATCH COMMAND
= 0010                  RW_LSB      EQU     00010000B     ;READ/WRITE LEAST SIG BYTE ONLY
= 0020                  RW_MSB      EQU     00100000B     ;READ/WRITE MOST SIG BYTE ONLY
= 0030                  RW_LSBMSB   EQU     00110000B     ;READ/WRITE LSB FIRST,THEN MSB

                        ;D3,D2,D1 - MODE

= 0000                  MD0         EQU     00000000B     ;MODE 0 - INTR ON TERM CNT
= 0002                  MD1         EQU     00000010B     ;MODE 1 - HARDWARE ONE-SHOT
= 0004                  MD2         EQU     00000100B     ;MODE 2 - RATE GENERATOR
= 0006                  MD3         EQU     00000110B     ;MODE 3 - SQUARE WAVE MODE
= 0008                  MD4         EQU     00001000B     ;MODE 4 - SOFTWARE TRIG. STROBE
= 000A                  MD5         EQU     00001010B     ;MODE 5 - HARDWARE TRIG. STROBE

                        ;D0 - BINARY/BCD COUNTER

= 0000                  BINARY      EQU     00000000B     ;BINARY COUNTER
= 0001                  BCD         EQU     00000001B     ;BCD COUNTER

                        ;ATTACHMENT ENABLE PORT

= FF9F                  TKR_ACL     EQU     0FF9FH        ;TALKER II AUDIO CONTROL LATCH

= 0001                  CHN_ON      EQU     01H           ;ENABLE CHANNEL   (OUTPUT)



= 0000                  CHN_OFF     EQU     00H           ;DISABLE CHANNEL  (OUTPUT)

= 0001                  ACL_OFF     EQU     001H          ;ACL DISABLED     (INPUT)

                        ;NOTE: NOTICE THERE IS A DIFFERENCE IN POLARITIES BETWEEN


                        ;       ENABLING/DISABLING THE ACL (OUTPUT) AND READING THE
                        ;       STATUS OF THE ACL (INPUT).

                        ;CVSD SHIFT REGISTER

= FF98                  SHIFTREG    EQU     0FF98H        ;CVSD SHIFT REGISTER

                        ;SYSTEM'S 8255 PORT B - PORT 61H

= 0061                  PORT_61H    EQU     061H          ;8255 PORT B
= 009F                  CLR_SPKSW   EQU     10011111B     ;CLEAR SPKR SWITCH BITS (PB6,PB5)
= 0040                  AUDIO_CHN   EQU     01000000B     ;I/O AUDIO CHANNEL IN

                        ;SYSTEM'S 8259

= 0020                  PORT_20H    EQU     020H          ;8259 OPERATION CNTL PORT
= 0021                  PORT_21H    EQU     021H          ;8259 MASK REGISTER
= 00FD                  INT1_ON     EQU     0FDH          ;ENABLE INTR 1  (AND)
= 0002                  INT1_OFF    EQU     002H          ;DISABLE INTR 1 (OR)
= 0061                  INT1_EOI    EQU     01100001B     ;SPECIFIC EOI CMD

                        ;SYSTEM NMI PORT

= 00A0                  NMI_PORT    EQU     0A0H          ;NMI PORT

                        ;CVSD SPEED EQUATES

= 0000                  SPEED_0     EQU     00H           ;SPEED = 1800 BYTES/SEC  (1802)
= 0001                  SPEED_1     EQU     01H           ;SPEED = 2400 BYTES/SEC  (2396)
= 0002                  SPEED_2     EQU     02H           ;SPEED = 3000 BYTES/SEC  (2997)
= 0003                  SPEED_3     EQU     03H           ;SPEED = 3600 BYTES/SEC  (3593)
= 0004                  SPEED_4     EQU     04H           ;SPEED = 4200 BYTES/SEC  (4201)
= 0005                  SPEED_5     EQU     05H           ;SPEED = 4800 BYTES/SEC  (4811)
= 0005                  SPEED_MAX   EQU     05H           ;MAXIMUM VALUE FOR SPEED DECODE

                        ; FUNCTION DECODES & ERRORS

                        ; FUNCTION VALUE IN AH

= 0000                  RST_FN      EQU      00H          ;RESET CARD FUNCTION
= 0001                  CVSD_FN     EQU      01H          ;CVSD FUNCTION
= 0002                  LPC_FN      EQU      02H          ;LPC FUNCTION (BACKGROUND)
= 0003                  LPC_FN_FORE EQU       03H         ;LPC FOREGROUND FUNCTION

                        ; SUB-FUNCTION VALUE IN AL

= 0000                  LPC_STATUS  EQU      00H          ;LPC STATUS
= 0001                  LPC_INDEX   EQU      01H          ;LPC SPEAK - INDEX



= 0002                  LPC_BUFFER  EQU      02H          ;LPC SPEAK - BUFFER

= 0000                  CVSDR       EQU      00H          ;CVSD RECORD INDICATOR
= 00FF                  CVSDW       EQU      0FFH         ;CVSD PLAYBACK INDICATOR

= 0000                  CVSDR_TBL   EQU      00H          ;CVSD RECORD USING TABLE SPEED
= 0001                  CVSDW_TBL   EQU      01H          ;CVSD PLAYBACK USING TABLE SPEED
= 0002                  CVSDR_USER  EQU      02H          ;CVSD RECORD USING USER SPEED
= 0003                  CVSDW_USER  EQU      03H          ;CVSD PLAYBACK USING USER SPEED

                        ; RETURN CODES (VALUE IN AL)
```

```
= 0000                          OK          EQU     00H         ;NO ERRORS
= 0001                          BAD_CMD     EQU     01H         ;UNDEFINED COMMAND
= 0002                          LPC_INPROG  EQU     02H         ;LPC SPEAK IN PROGRESS
= 0003                          ACL_ERROR   EQU     03H         ;ACL STUCK ON CARD
= 0004                          INDEX_ERR   EQU     04H         ;LPC INDEX OUT OF RANGE
= 0005                          SPEED_ERR   EQU     05H         ;SPEED OUT OF RANGE
= 0006                          LPCRDY_ERR  EQU     06H         ;TIMEOUT WAITING FOR LPC READY

                                ; USED INTERRUPTS

= 004D                          TALKER      EQU     04DH        ;BIOS INTERRUPT
= 004E                          KBD         EQU     04EH        ;KBD INTR MOVED TO 04EH

                                ; 5220 COMMANDS

= 0060                          SPK_EXT     EQU     01100000B   ;SPEAK EXTERNAL CMD (X110XXXX)
= 00FF                          RST_5220    EQU     11111111B   ;RESET CMD (X111XXXX)

                                ; 5220 STATUS

= 0080                          TALK_ON     EQU     80H         ;TS - TALK STATUS ACTIVE
= 0040                          BFR_LOW     EQU     40H         ;BL - BUFFER LOW
= 0020                          BFR_EMPTY   EQU     20H         ;BE - BUFFER EMPTY

= 00FF                          STOP_CODE   EQU     0FFH        ;5220 SPEAK STOP CODE

                                ;       POST ERROR CODE

= 004A                          CUST_ER         EQU     'J'     ;CUSTOMER ER. CODE
= 0028                          SERV_ER         EQU     28H     ;SERVICE ER. CODE
= 2833                          RESET_ER        EQU     2833H   ;SERVICE LEVEL ERROR
                                                                ;WHEN CARD RESET FAILS
= 0001                          ER_CODE8255     EQU     01H     ;ER. CODE ON THE 8255
= 0002                          ER_CODE8254     EQU     02H     ;ER. CODE ON THE 8254
                                ;
                                ;       DIAGNOSTIC ERROR CODE FOR LPC AND CVSD
                                ;
                                ;               CUSTOMER LEVEL
                                ;
= 0042                          ER_LPC_C1       EQU     'B'     ;RESET FAIL, PROBABLY BAD CARD
= 0043                          ER_LPC_C2       EQU     'C'     ;LPC ERROR



= 0044                          ER_CVSD_C1      EQU     'D'     ;CVSD PLAYBACK ERROR
= 0045                          ER_CVSD_C2      EQU     'E'     ;CVSD RECORD ERROR
                                ;
                                ;               SERVICE LEVEL
                                ;
= 0010                          ER_LPC_S1       EQU     10H     ;RESET FAIL, PROBABLY BAD CARD
= 0011                          ER_LPC_S2       EQU     11H     ;LPC ERROR
= 0012                          ER_CVSD_S1      EQU     12H     ;CVSD PLAYBACK ERROR
= 0013                          ER_CVSD_S2      EQU     13H     ;CVSD RECORD ERROR
= 0014                          ER_CVSD_S3      EQU     14H     ;CVSD PLAYBACK AFTER RECORD ERROR
                                ;
= 0007                          TLK_WIDTH        EQU     07H     ;TALKER ICON WIDTH
                                ;
                                ;       CURSOR POSITION TO PUT TALKER ICON, SELECTION ...
                                ;
= 070E                          SPEAKER_POS     EQU     070EH   ;CURSOR AT ROW 8 COL 16
= 0A15                          WAVE_POS        EQU     0A15H   ;       "       11   " 21

= 0812                          MIC_POS         EQU     0812H   ;       "       8    " 18
= 890D                          ARROW_POS1      EQU     890DH   ;       "       9    " 13 BIT
                                                                ; SET FOR SPECIAL ATTRIBUTE
= 890D                          ARROW_POS2      EQU     0890DH  ;       "       9    " 13 BIT
                                                                ; SET FOR SPECIAL ATTRIBUTE AND BEEP
                                ;
                                ;       ROUTINE USED FROM SYSTEM BIOS
                                ;
= 0081                          LOCATE          EQU     81H     ;LOCATE ROUTINE TO PUT ICON
= 0082                          PRINT           EQU     82H     ;PRINT ROUTINE TO PUT ICON

0000                            TKRSEG  SEGMENT

                                        ASSUME  CS:TKRSEG,DS:DUMMY

0000                                    ORG     0
0000  55 AA                     X       DB      055H,0AAH       ;ROS INDICATOR
0002  10                               DB      010H            ;LENGTH

0003                            INIT    PROC    FAR
0003  EB 2A                             JMP     SHORT INIT1     ;GO TO BEG OF INIT CODE
0005  00                               DB      PAGE0           ;BANK 0 IDENTIFIER
0006  00                               DB      00H             ;CMD NAME LENGTH
                                ;                               ;COPYRIGHT INFORMATION
0007  36 31 38 31 37 33                 DB      '6181736 COPR. IBM 1984'
      36 20 43 4F 50 52
      2E 20 49 42 4D 20
      31 39 38 34


                                ;TABLE OF DIVISORS FOR DIFFERENT CVSD RATES

001D  014B                      SPEED_TBL       DW      331     ;SPEED = 1800 BYTES/SEC
001F  00F9                              DW      249             ;SPEED = 2400 BYTES/SEC
0021  00C7                              DW      199             ;SPEED = 3000 BYTES/SEC
0023  00A6                              DW      166             ;SPEED = 3600 BYTES/SEC
0025  008E                              DW      142             ;SPEED = 4200 BYTES/SEC



0027  007C                              DW      124             ;SPEED = 4800 BYTES/SEC

0029  0C90 R                            DW      OFFSET WORDS_BEGIN ;POINTER TO THE END OF
                                                                ;DUPLICATED CODE

                                ;*************************************************************
                                ; DESCRIPTION:
                                ;       TEST CODE IS LOADED INTO RAM AT SEGMENT 100H AND
                                ;       THE SAME OFFSET IT HAS IN THIS ROM MODULE.
                                ;       CONTROL IS PASSED TO THIS RAM CODE.
                                ;       THE 8255 PPI IS TESTED.
                                ;       BANK SWITCHING IS TESTED, AND ALL FOUR BANKS ARE
                                ;       CHECKSUMMED.
                                ;       IF AN ERROR IS ENCOUNTERED, IT IS PROCESSED AND
                                ;       CONTROL IS RETURNED DIRECTLY TO THE SYSTEM (RATHER
                                ;       THAT TO THE FEATURE ROM).
                                ;       IF NO ERROR IS ENCOUNTERED, RAM IS RESTORED TO ZEROS UPON
                                ;       RETURN.
                                ;
                                ; ENTRY CONDITIONS:
                                ;       BC_LEN MUST EQUAL THE # OF WORDS
                                ;       TO BE MOVED. BC_START MUST EQUAL THE OFFSET OF THE
                                ;       BEGINNING OF THE CODE TO MOVE.
```

```
                        ;
                        ; ON EXIT:
                        ;       ALL REGS BUT BX,DX,SP, AND SS ARE DESTROYED.
                        ;
                        ;*****************************************************************
002B  01A6 R            BCLOC   DW      OFFSET BANK_TEST_START
002D  0100              INIT1:  DW      0100H           ;RAM STARTING LOCATION OF THE CODE.
002F
002F  B8 0100                   MOV     AX,0100H                ;LOAD CODE ON THE 4K BOUND
0032  8E C0                     MOV     ES,AX                   ;
0034  BF 0172 R                 MOV     DI,OFFSET BC_START
0037  57                        PUSH    DI                      ;SAVE DI FOR LATER
0038  8B F7                     MOV     SI,DI                   ;ES:DI=LOCATION TO PUT CODE
003A  0E                        PUSH    CS
003B  1F                        POP     DS                      ;DS:SI=LOC OF CODE TO LIFT
003C  B9 00B1 90                MOV     CX,BCLEN                ;NUMBER OF WORDS TO MOVE
0040  51                        PUSH    CX                      ;SAVE CX FOR LATER USE
0041  F3/ A5                   REP     MOVSW                   ;MOVE THE CODE TO RAM

0043  06                        PUSH    ES                      ;SAVE REGS ADDRESSING RAM

0044  2E: FF 1E 002B R          CALL    DWORD PTR BCLOC         ;CALL RAM CODE

0049  07                        POP     ES                      ;RESTORE REGS ADDRESSING
004A  59                        POP     CX                      ;RAM
004B  5F                        POP     DI
004C  33 C0                     XOR     AX,AX                   ;AX=0

004E  F3/ AB                    REP     STOSW                   ;RESTORE USED RAM TO ZEROS



                        ;*************************************************
                        ;       THIS CODE LOADS THE NEEDED INTERRUPT VECTORS
                        ;*************************************************
0050  33 C0                      XOR     AX,AX
0052  8E D8                      MOV     DS,AX
0054  C7 06 0134 R 02D4 R        MOV     WORD PTR TALKER_PTR,OFFSET START
005A  8C 0E 0136 R               MOV     WORD PTR TALKER_PTR+2,CS
005E  C7 06 0248 R 07FF R        MOV     WORD PTR TALKER_DIAG_PTR,OFFSET TALKER2_DIAG
0064  8C 0E 024A R               MOV     WORD PTR TALKER_DIAG_PTR+2,CS

                        ;*****************************************************************
                        ; POWER ON SELF TEST                                            |
                        ;                                                              |
                        ; DESCRIPTION:                                                 |
                        ;       TIMER CHANNELS ON THE 8254 ARE TESTED FOR STUCK BITS.  |
                        ;       TIMER 1'S RESPONSE TO TIMER 0 IS CHECKED.              |
                        ;       HARDWARE ON THE CARD IS RESET (SEE BIOS RESET COMMAND) |
                        ;                                                              |
                        ; ERROR CODES:  (SOME MAY BE PASSED BY CODE PREVIOUSLY EXECUTED|
                        ;               FROM RAM)                                      |
                        ;       CUSTOMER LEVEL: J                                      |
                        ;       SERVICE LEVEL:  28XX                                   |
                        ;                         XX = 01 PORT A FAIL MODE 83H         |
                        ;                              02  "  B  "    "     "          |
                        ;                              03  "  C  "    "     "          |
                        ;                              04  "  A  "    "    81H         |
                        ;                              05  "  C  "    "     "          |
                        ;                                                              |
                        ;                              10 STUCK BIT IN TIMER CHANNEL 0 |
                        ;                              11 STUCK BIT IN TIMER CHANNEL 1 |
                        ;                              12 STUCK BIT IN TIMER CHANNEL 2 |
                        ;                              13 CVSD FRAME NOT CHANGING      |
                        ;                              14 CVSD CLOCK NOT CHANGING      |
                        ;                              15 CVSD FRAME NOT RESPONDING TO |
                        ;                                 CVSD CLOCK AS EXPECTED       |
                        ;                              23 ACL ERROR DURRING CARD RESET |
                        ;                              26 TIMEOUT WAITING FOR  LPC     |
                        ;                                 COMPLETION DURRING CARD      |
                        ;                                 RESET                        |
                        ;                                                              |
                        ;       PORT A = FB98H                                         |
                        ;       PORT B = FB99H                                         |
                        ;       PORT C = FB9AH                                         |
                        ;*****************************************************************

                        ;-----------------------------------------------------------------
                        ; TEST:                                                          |
                        ;       8254 PROGRAMMABLE INTERVAL TIMER TEST                   |
                        ;                                                               |
                        ; DESCRIPTION:                                                  |
                        ;       TEST FOR STUCK BITS IN TIMER CHANNELS 0, 1, AND 2.      |
                        ;       TEST TO SEE THAT THE OUTPUT OF TIMER 0 IS WORKING       |
                        ;       VERIFY THAT VIMER 1 DIVIDES TIMER 0 BY 8                |



                        ; NOTES:                                                        |
                        ;       COUNTER 0 = CVSD BIT CLOCK                              |
                        ;       COUNTER 1 = CVSD FRAME                                  |
                        ;       COUNTER 2 = LPC INTERRUPT CLOCK                         |
                        ;-----------------------------------------------------------------

0068                    POST    PROC    FAR

                        ;-----------------------------------------------------------------
                        ;       RESET HARDWARE INTO A KNOWN STATE                       :
                        ;-----------------------------------------------------------------

0068  E8 0308 R                 CALL    RST_TALKER      ;INIT 8255, 8254, ACL
006B  0A C0                     OR      AL,AL           ;AL = 00  PASSED, ELSE FAILED
006D  74 03                     JZ      T8254           ;PASSED
006F  E9 0110 R                 JMP     CARD_RESET_ER   ;REPORT CARD RESET ERROR

                        ; SET INITIAL COUNT FOR CTRS 0, 1, AND 2 TO TEST FOR STUCK BITS

0072                    T8254:
0072  B0 36                     MOV     AL,CTR0+RW_LSBMSB+MD3+BINARY    ;FOR CWR_8254
0074  B9 FB9C                   MOV     CX,CVSD_CLK     ;COUNTER 0
0077  BB FFFF                   MOV     BX,0FFFFH       ;INITIAL COUNT FOR COUNTER 0
007A  E8 011D R                 CALL    INIT_TIMER      ;SET INITIAL COUNT

007D  B0 74                     MOV     AL,CTR1+RW_LSBMSB+MD2+BINARY    ;FOR CWR_8254
007F  41                        INC     CX              ;COUNTER 1 HAS CVSD_FRAME ADDR
0080  B7 00                     MOV     BH,00           ;INITIAL COUNT FOR CTR 1 IS 00FF
0082  E8 011D R                 CALL    INIT_TIMER      ;SET INITIAL COUNT

0085  B0 B4                     MOV     AL,CTR2+RW_LSBMSB+MD2+BINARY    ;FOR LPC INT TIMER
0087  41                        INC     CX              ;COUNTER 2 HAS INTR_CTR ADDR
                                                        ;INITIAL COUNT IS 00FF
0088  E8 011D R                 CALL    INIT_TIMER      ;SET INITIAL COUNT
```

```
                                ;
                                ; CHECK IF ALL BITS GO ON/OFF IN COUNTER 0 (CVSD_CLK)
                                ;
008B  B0 06                             MOV     AL,CTR0+CTR_LATCH+MD3+BINARY    ;FOR CWR_8254
008D  B9 FB9C                           MOV     CX,CVSD_CLK     ;COUNTER 0
0090  E8 012C R                         CALL    BITS_ON_OFF     ;SEE THAT ALL BITS GO ON AND OFF
0093  B3 10                            MOV     BL,10H          ;ERROR CODE FOR COUNTER 0 IS 10
0095  72 0C                            JC      TIMER_ERROR     ;POST MESSAGE IF ERROR FOUND
                                ;
                                ; CHECK IF ALL BITS GO ON/OFF IN TIMER 1 (CVSD_FRAME)
                                ;
0097                            COUNTER_CK:
0097  B0 44                             MOV     AL,CTR1+CTR_LATCH+MD2+BINARY    ;FOR CWR_8254
0099  B9 FB9D                           MOV     CX,CVSD_FRAME   ;COUNTER 1
009C  E8 012C R                         CALL    BITS_ON_OFF     ;CHECK BITS
009F  B3 11                            MOV     BL,11H          ;ERROR CODE COUNTER 1 IS 11
00A1  73 07                            JNC     CHECK_COUNTER_2 ;IF NO ERROR GO ON
                                                                ;OTHERWISE FALL THROUGH AND
                                                                ;POST AN ERROR
00A3                            TIMER_ERROR:
00A3  B4 4                              MOV     AH,CUST_ER      ;CUSTOMER ER. CODE FOR IS "J"
00A5  B7 2.                             MOV     BH,SERV_ER      ;SERVICE ERROR CODE IS 28XX
00A7  E9 0' 1                           JMP     NEAR PTR E_MSG_B ;DISPLAY ERROR MESSAGE
                                ;
                                ; CHECK IF ALL BITS GO ON/OFF IN TIMER 2 (INTR_CTR )
                                ;
00AA                            CHECK_COUNTER_2:
00AA  B0 84                             MOV     AL,CTR2+CTR_LATCH+MD2+BINARY    ;FOR LPC INT TIMER
00AC  B9 FB9E                           MOV     CX,INTR_CTR     ;COUNTER 2
00AF  E8 012C R                         CALL    BITS_ON_OFF     ;CHECK BITS
00B2  B3 12                            MOV     BL,12H          ;ERROR CODE COUNTER 2 IS 12
00B4  72 ED                            JC      TIMER_ERROR     ;POST ERROR MESSAGE
                                ;
                                ; SET INITIAL COUNT FOR COUNTERS 0 AND 1
                                ;
00B6  B0 36                             MOV     AL,CTR0+RW_LSBMSB+MD3+BINARY
00B8  B9 FB9C                           MOV     CX,CVSD_CLK     ;COUNTER 0 INIT
00BB  BB 03FF                           MOV     BX,03FFH        ;COUNT DOWN FROM 03FFH
00BE  E8 011D R                         CALL    INIT_TIMER

00C1  B0 74                             MOV     AL,CTR1+RW_LSBMSB+MD2+BINARY
00C3  41                                INC     CX              ;COUNTER 1 INIT
00C4  BB 0008                           MOV     BX,8            ;DIVIDE BY 8
00C7  E8 011D R                         CALL    INIT_TIMER

00CA  BA FB9A                           MOV     DX,PORTC        ;PORT ADDRESS OF PORT C
                                                                ;THE OUTPUT OF TIMERS 0 AND 1 CAN
                                                                ;BE READ ON PORT C
00CD  B3 13                             MOV     BL,13H          ;ERROR CODE
00CF  33 C9                             XOR     CX,CX           ;TIMEOUT

00D1                            TEST_FRAME_HI:
00D1  EC                                IN      AL,DX           ;GET COUNTER OUTPUT VALUES
00D2  A8 04                             TEST    AL,00000100B    ;TEST CVSD FRAME BIT
00D4  E1 FB                             LOOPZ   TEST_FRAME_HI   ;IF FRAME IS LOW LOOP BACK

00D6  E3 CB                             JCXZ    TIMER_ERROR     ;TIMEOUT, NO CVSD FRAME
                                                                ;ERROR CODE 13
00D8  33 C9                             XOR     CX,CX

00DA                            TEST_FRAME_LO:
00DA  EC                                IN      AL,DX
00DB  A8 04                             TEST    AL,00000100B    ;TEST CVSD FRAME BIT
00DD  E0 FB                             LOOPNZ  TEST_FRAME_LO   ;LOOP BACK UNTIL FRAME GOES LOW

00DF  E3 C2                             JCXZ    TIMER_ERROR     ;IF FRAME DOESN'T GO LOW, ERROR
                                                                ;ERROR CODE 13
00E1  43                                INC     BX              ;INCREMENT ERROR TO 14

00E2  B9 0008                           MOV     CX,8
00E5                            TRANSIT:
00E5  51                                PUSH    CX              ;WE WILL WATCH 8 CYCLES OF TIMER 0

00E6  33 C9                             XOR     CX,CX
00E8                            LTH:
00E8  EC                                IN      AL,DX
00E9  A8 08                             TEST    AL,00001000B    ;LOOK AT OUTPUT OF TIMER 0
00EB  E1 FB                             LOOPZ   LTH             ;LOOP UNTIL LO TO HI TRANSIT MADE
00ED  E3 13                             JCXZ    CT_EP           ;IF TIMEOUT, TIMER IS TOO SLOW
                                                                ;ERROR CODE 14
00EF  33 C9                             XOR     CX,CX
00F1                            HTL:
00F1  EC                                IN      AL,DX
00F2  A8 08                             TEST    AL,00001000B    ;LOOK AT OUTPUT OF TIMER 0
00F4  E0 FB                             LOOPNZ  HTL             ;LOOP UNTIL HI TO LO TRANSIT MADE
00F6  E3 0A                             JCXZ    CT_EP           ;ERROR 14 IF TIMEOUT

00F8  59                                POP     CX
00F9  A8 04                             TEST    AL,00000100B    ;IS THE CVSD FRAME BIT HIGH?
00FB  E0 E8                             LOOPNZ  TRANSIT         ;IT SHOULD BE FOR 8 CLOCK CYCLES

00FD  43                                INC     BX              ;INCREMENT ERROR BYTE TO 15
00FE  E3 05                             JCXZ    LD              ;IF CX IS NOT 0 A TIMER IS BROKEN
0100  EB A1                             JMP     SHORT TIMER_ERROR

0102                            CT_EP:
0102  58                                POP     AX              ;BALANCE STACK FOR RETURN
0103  EB 9E                             JMP     TIMER_ERROR

0105                            LD:
0105  A8 04                             TEST    AL,00000100B    ;IS THE CVSD FRAME HIGH?
0107  75 9A                             JNZ     TIMER_ERROR     ;IF SO, IT IS BROKEN

                                ;-----------------------------------------------------------------
                                ;       RESET HARDWARE INTO A KNOWN STATE                       :
                                ;-----------------------------------------------------------------
0109  E8 0308 R                         CALL    RST_TALKER      ;INIT 8255, 8254, ACL
010C  0A C0                             OR      AL,AL           ;AL = 00  PASSED, ELSE FAILED
010E  74 0C                             JZ      EXIT_POST
0110                            CARD_RESET_ER:
0110  B4 4A                             MOV     AH,CUST_ER      ;SET ERROR CODES IN CASE OF FAILURE
0112  B7 28                             MOV     BH,28H          ;HIGH PART OF SERVICE ERROR CODE
0114  8A D8                             MOV     BL,AL           ;AL IS ERROR CODE RETURNED BY RESET
0116  80 CB 20                          OR      BL,20H          ;BL = 23 OR 26,
0119  E9 0211 R                         JMP     NEAR PTR E_MSG_B ;PUT ERROR MESSAGE
011C                            EXIT_POST:
011C  CB                                RET

011D                            POST    ENDP
```

```
                        ;**********************
                        ;SUBROUTINES FOR POST  I
                        ;**********************

                        ;-------------------------------------------------------------------------
                        ; INIT_TIMER       SUBROUTINE -                                          :


                        ;       SET COUNTER TO INITIAL COUNT                                    :
                        ; ENTRY:                                                                 :
                        ;       CX = COUNTER 0 OR 1 OR 2 ADDRESS                                :
                        ;       AL = CONTROL WORD REGISTER (CWR)                                :
                        ;       BX = INITIAL COUNT                                              :
                        ;               BH = MSB COUNT                                          :
                        ;               BL = LSB COUNT                                          :
                        ; EXIT:                                                                  :
                        ;       DX = COUNTER ADDRESS                                            :
                        ;       AL = HI BYTE OF INITIAL COUNT                                   :
                        ;       OTHER REGISTERS ARE UNCHANGED                                   :
                        ;-------------------------------------------------------------------------
011D                    INIT_TIMER      PROC    NEAR
011D  51                        PUSH    CX              ;SAVE COUNTER ADDRESS
011E  BA FB9F                   MOV     DX,CWR_8254     ;CONFIGURE CWR FOR COUNTER
0121  EE                        OUT     DX,AL
0122  5A                        POP     DX              ;RESTORE COUNTER ADDRESS
0123  8A C3                    MOV     AL,BL           ;LOAD LSB
0125  EE                        OUT     DX,AL
0126  50                        PUSH    AX              ;PAUSE
0127  58                        POP     AX
0128  8A C7                    MOV     AL,BH           ;LOAD MSB
012A  EE                        OUT     DX,AL
012B  C3                        RET
012C                    INIT_TIMER      ENDP

                        ;-------------------------------------------------------------------------
                        ; BITS_ON_OFF    SUBROUTINE -                                           :
                        ;       USED TO DETERMININE IF A COUNTER'S BITS GO ON                   :
                        ;       AND OFF AS THEY SHOULD.                                         :
                        ; ENTRY:                                                                 :
                        ;       CX = COUNTER 0 OR 1 OR 2 ADDRESS                                :
                        ;       AL = CONTROL WORD REGISTER (CWR)                                :
                        ; EXIT:                                                                  :
                        ;       CF = 1 IF FAILED                                                :
                        ;       CF = 0 IF PASSED                                                :
                        ; REGISTER AX,BX,CX,DX,SI ARE ALTERED                                   :
                        ;-------------------------------------------------------------------------
012C                    BITS_ON_OFF     PROC    NEAR
012C  51                        PUSH    CX              ;SAVE COUNTER ADDRESS
012D  BA FB9F                   MOV     DX,CWR_8254     ;CONFIGURE CWR FOR COUNTER
0130  EE                        OUT     DX,AL
0131  33 DB                    XOR     BX,BX           ;INITIALIZE REGISTER
0133  33 F6                    XOR     SI,SI           ;1ST PASS - SI = 0
0135  5A                        POP     DX              ;RESTORE COUNTER ADDRESS
0136                    OUTER_LOOP:
0136  B9 0008                  MOV     CX,8            ;OUTER LOOP COUNTER
0139                    INNER_LOOP:
0139  51                        PUSH    CX
013A  33 C9                    XOR     CX,CX           ;INNER LOOP COUNTER
013C                    TEST_BITS:
013C  EC                        IN      AL,DX           ;READ COUNTER LSB
013D  0B F6                    OR      SI,SI

013F  75 19                    JNE     SECOND          ;SECOND PASS
0141  0C 01                    OR      AL,01H          ;TURN LS BIT ON
0143  0A D8                    OR      BL,AL           ;TURN 'ON' BITS ON
0145  EC                        IN      AL,DX           ;READ COUNTER MSB
0146  0A F8                    OR      BH,AL           ;TURN 'ON' BITS ON
0148  81 FA FB9C               CMP     DX,CVSD_CLK     ;
014C  75 06                    JNE     CNT1_TEST_BITS  ;TEST BITS OF COUNTER 1
014E  81 FB FFFF               CMP     BX,0FFFFH       ;ARE ALL COUNTER BITS ON?
0152  EB 0D                    JMP     SHORT TST_CMP   ;DON'T CHANGE FLAGS
0154                    CNT1_TEST_BITS:
0154  81 FB 00FF               CMP     BX,00FFH        ;LOW NIBBLE BITS ON?
0158  EB 07                    JMP     SHORT TST_CMP   ;DON'T CHANGE FLAGS
015A                    SECOND:
015A  22 D8                    AND     BL,AL           ;CHECK FOR ALL BITS OFF
015C  EC                        IN      AL,DX           ;READ MSB
015D  22 F8                    AND     BH,AL           ;TURN OFF BITS
015F  0B DB                    OR      BX,BX           ;ALL OFF?
0161                    TST_CMP:
0161  74 07                    JE      CHK_END         ;YES, SEE IF DONE
0163  E2 D7                    LOOP    TEST_BITS       ;TRY AGAIN
0165  59                        POP     CX              ;RESTORE OUTER LOOP COUNTER
0166  E2 D1                    LOOP    INNER_LOOP      ;TRY AGAIN
0168  F9                        STC                     ;ALL TRIES EXHAUSTED - FAILED
0169  C3                        RET
016A                    CHK_END:
016A  59                        POP     CX              ;FORMER OUTER LOOP COUNTER
016B  46                        INC     SI
016C  83 FE 02                 CMP     SI,2
016F  75 C5                    JNE     OUTER_LOOP      ;CHECK FOR ALL BITS TO GO OFF
0171  C3                        RET                     ;CARRY FLAG IS RESET
0172                    BITS_ON_OFF     ENDP

                        ;

0172                    BC_START:
                        ;-------------------------------------------------------------------------
                        ; PORT_TST                                                               :
                        ;       THIS PROC DOES A WRITE READ TEST TO A PORT                      :
                        ; ENTRY:                                                                 :
                        ;       DX = PORT TO TEST                                               :
                        ;       BL IS ERROR CODE. IT IS INCREMENTED.                            :
                        ;-------------------------------------------------------------------------
0172                    PORT_TST        PROC    NEAR
0172  43                        INC     BX
0173  2B C0                    SUB     AX,AX           ;TEST PATTERN SEED = 0000
0175  8A C4                    PA:     MOV     AL,AH           ;SAVE PATTERN TO COMPARE
0177  EE                        OUT     DX,AL           ;WRITE PATTERN TO PORT A
0178  EB 00                    JMP     $+2             ;PAUSE
017A  EC                        IN      AL,DX           ;READ PATTERN FROM PORT A
017B  3A C4                    CMP     AL,AH           ;DATA EXPECTED?
017D  75 05                    JNE     PA_E            ;NO, ERROR
017F  FE C4                    INC     AH              ;NEW PATTERN
0181  75 F2                    JNZ     PA              ;LOOP TILL 255 PATTERNS DONE



0183  C3                        RET                     ;CARRY FLAG IS RESET
0184  F9                        PA_E:   STC             ;ERROR RETURN
```

```
0185  C3                        RET
0186                    PORT_TST        ENDP

                        ;--------------------------------------------------------------
                        ; PORTC_TST                                                   :
                        ;       THIS PROC DOES A WRITE READ TEST TO PORT C. (FB9AH)   :
                        ;--------------------------------------------------------------
0186                    PORTC_TST       PROC    NEAR
0186  BA FB9A                  MOV     DX,PORTC        ;PORT C ADDRESS
0189  EC                      IN      AL,DX           ;READ PORT C
018A  24 0F                   AND     AL,00001111B    ;MASK LOWER PORT C
                                                      ;(C0 - C3 = INPUT)
018C  8A F8                   MOV     BH,AL           ;SAVE LOWER BITS IN BH
018E  B4 00                   MOV     AH,0            ;BEGINNING PATTERN TO WRITE
0190  43                      INC     BX              ;INCREMENT ERROR INDICATOR

0191  8A C4             PC:  MOV     AL,AH           ;OUTPUT PATTERN FOR PORT C
0193  0A C7                   OR      AL,BH           ;TURN ON LOWER BITS AS APPROPRIATE
                                                      ;(LOWER C IS THE SAME)
0195  EE                      OUT     DX,AL           ;WRITE TO PORT C
0196  50                      PUSH    AX
0197  58                      POP     AX              ;TIME DELAY
0198  EC                      IN      AL,DX           ;READ PORT
0199  24 F0                   AND     AL,11110000B    ;TURN OFF UNNEEDED BITS
019B  3A C4                   CMP     AL,AH           ;DATA EXPECTED?
019D  75 E5                   JNE     PA_E            ;NO, ERROR

019F  80 C4 10                ADD     AH,00010000B    ;GET NEXT TEST PATTERN
01A2  73 ED                   JNC     PC
01A4  F8                      CLC                     ;GOOD RETURN
01A5  C3                      RET
01A6                    PORTC_TST       ENDP


                        ;--------------------------------------------------------------
                        ; TEST:                                                       :
                        ;       8255 PROGRAMMABLE PERIPHERAL INTERFACE TEST           :
                        ;                                                            :
                        ; DESCRIPTION:                                               :
                        ;       PERFORM WRITE/READ TEST TO PORT A, B, AND C IN       :
                        ;       MODE 83H. DO THE SAME TEST FOR PORTS A & C IN MODE 81H.:
                        ;           MODE 83H: PORT A = OUTPUT                        :
                        ;                     B = OUTPUT                             :
                        ;                     C0-C3 = INPUT                          :
                        ;                     C4-C7 = OUTPUT                         :
                        ;           MODE 81H: PORT A = OUTPUT                        :
                        ;                     B = INPUT                              :
                        ;                     C0-C3 = INPUT                          :
                        ;                     C4-C7 = OUTPUT                         :
                        ;           PORT A = FB98H                                   :
                        ;           PORT B = FB99H                                   :


                        ;           PORT C = FB9AH                                   :
                        ;--------------------------------------------------------------

01A6                    BC      PROC    FAR
01A6                    BANK_TEST_START:
01A6  33 DB                     XOR     BX,BX           ;INITIALIZE ERROR FLAG
01A8  BA FB9B                   MOV     DX,CWREG        ;CONTROL WORD REGISTER
01AB  B0 81                     MOV     AL,LPC_OUT      ;MODE 81H. PORT A,B=OUTPUT,

                                                        ;C(LOW)=INPUT, C(HI)=OUTPUT
01AD  EE                        OUT     DX,AL           ;CONFIGURES I/O PORT
                        ;
01AE  BA FB98                   MOV     DX,PORTA
01B1  E8 0172 R                 CALL    PORT_TST        ;TEST PORT A
01B4  72 54                     JC      EX              ;ERROR 01 IF TEST FAILS
                        ;
01B6  42                        INC     DX              ;DX=PORT B ADDRESS
01B7  E8 0172 R                 CALL    PORT_TST        ;TEST PORT B
01BA  72 4E                     JC      EX              ;ERROR 02 IF TEST FAILS
                        ;
01BC  E8 0186 R                 CALL    PORTC_TST       ;TEST PORT C
01BF  72 49                     JC      EX              ;ERROR 03 IF TEST FAILS
                        ;
01C1  BA FB9B                   MOV     DX,CWREG        ;CONTROL WORD REGESTER
01C4  B0 83                     MOV     AL,LPC_IN       ;MODE 83H. PORT A=OUTPUT,B=INPUT
                                                        ;C(LOW)=INPUT, C(HI)=OUTPUT
01C6  EE                        OUT     DX,AL           ;CONFIGURES I/O PORT
                        ;
01C7  BA FB98                   MOV     DX,PORTA
01CA  E8 0172 R                 CALL    PORT_TST        ;TEST PORT A
01CD  72 3B                     JC      EX              ;ERROR 04 IF TEST FAILS

01CF  E8 0186 R                 CALL    PORTC_TST       ;TEST PORT C
01D2  72 36                     JC      EX              ;ERROR 05 IF TEST FAILS
                        ;********************
                        ; BANK SWITCH TEST
                        ;********************
01D4  BA FB98                   MOV     DX,PORTA        ;GET PORT A ADDR
01D7  FC                        CLD                     ;CLEAR DIRECTION FLAG
01D8  B3 0C                     MOV     BL,PAGE3        ;START WITH PAGE 3
01DA                    BTL:
01DA  EC                ;---------SELECT BANK---------
01DB  24 F3                     IN      AL,DX           ;READ PORT
01DD  0A C3                     AND     AL,CLR_PAGE     ;CLEAR PAGE BITS
01DF  EE                        OR      AL,BL           ;SET BITS TO SELECT DESIRED PAGE
01E0  EB 00                     OUT     DX,AL           ;SELECT PAGE
                                JMP     $+2             ;DELAY FOR HARDWARE RESPONSE
01E2  BE 0005           ;-------VERIFY CORRECT BANK-------
01E5  8A 04                     MOV     SI,5            ;ADDRESS BANK IDENTIFIER BYTE
01E7  3A C3                     MOV     AL,[SI]         ;READ IT
01E9  75 19                     CMP     AL,BL           ;IS IT AS EXPECTED?
                                JNZ     BSE             ;IF NOT, BANK SWITCH ERROR
01EB  B9 2000           ;---------CHECKSUM BANK---------
                                MOV     CX,8192         ;8K BYTES



01EE  33 F6                     XOR     SI,SI           ;INIT POINTERS
01F0  8B C6                     MOV     AX,SI
01F2                    ADD:
01F2  AC                        LODSB                   ;READ BYTE
01F3  02 E0                     ADD     AH,AL           ;RUNNING TOTAL IN AH
01F5  E2 FB                     LOOP    ADD
01F7  75 0F                     JNZ     BSUE            ;IF CHECKSUM <> 0 BANK SUM ERROR
                        ;----SEE IF BOTH BANKS HAVE BEEN TESTED---
01F9  80 FB 00                  CMP     BL,PAGE0        ;DID WE TEST PAGE 0?
01FC  74 05                     JZ      BTD             ;IF SO BANK TEST DONE
01FE  80 EB 04                  SUB     BL,PAGE1        ;IF NOT TEST NEXT BANK
0201  EB D7                     JMP     SHORT BTL       ;
0203                    BTD:
0203  CB                        RET                     ;RETURN TO ROM
```

```
0204                    BSE:
0204  B3 30                     MOV     BL,30H          ;SERVICE ERROR 2830 IF BANK
                                                        ;SWITCH ERROR
0206  EB 02                     JMP     SHORT EX
0208                    BSUE:
0208  B3 31                     MOV     BL,31H          ;SERVICE ERROR 2831 IF BANK
                                                        ;SUM ERROR
020A                    EX:
020A  B4 4A                     MOV     AH,CUST_ER      ;ERROR J IN CUSTOMER MODE.
020C  B7 28                     MOV     BH,SERV_ER
020E  83 C4 0A                  ADD     SP,10           ;ADJUST STACK. WE ARE GOING TO
                                                        ;FALL INTO THE ERROR MESSAGE CODE
                                                        ;AND RETURN TO SYSTEM FROM THERE
0211                    BC      ENDP

                        ;-----------------------------------------------------------------
                        ;       THIS SUBROUTINE IS THE GENERAL ERROR HANDLER FOR THE POST
                        ;
                        ; ENTRY REQUIREMENTS:
                        ;       AH = ASCII CUSTOMER LEVEL ERROR CODE
                        ;       BX= ERROR CODE FOR MANUFACTURING OR SERVICE MODE
                        ;       REGISTERS ARE NOT PRESERVED
                        ;       LOCATION "POST_ERR" IS SET NON-ZERO IF AN ERROR OCCURS IN
                        ;       CUSTOMER MODE
                        ;       SERVICE/MANUFACTURING FLAGS AS FOLLOWS: (HIGH NIBBLE OF
                        ;       PORT 201)
                        ;       0000 = MANUFACTURING (BURN-IN) MODE
                        ;       0001 = MANUFACTURING (SYSTEM TEST) MODE
                        ;       0010 = SERVICE MODE (LOOP POST)
                        ;       0100 = SERVICE MODE (SYSTEM TEST)
                        ;-----------------------------------------------------------------
                        ; FOLLOWON FEATURES MUST BE SURE TO SETUP MEMORY LOCATIONS
                        ; AS DESCRIBED BELOW:
                        ;
                        ;       XXDATA  SEGMENT AT      50H
                        ;               ORG     18H
                        ;       POST_ERR DB     ?
                        ;       XXDATA  ENDS



                        ;-----------------------------------------------------------------
= 0040                  TIMER   EQU     40H
= 0061                  PORT_B  EQU     61H

0211                    E_MSG_B         PROC    FAR

0211  BA 0011                   MOV     DX,11H          ;
0214  8A C7                     MOV     AL,BH           ;
0216  EE                        OUT     DX,AL           ; SEND HI BYTE ERROR CODE
0217  42                        INC     DX              ;
0218  8A C3                     MOV     AL,BL           ;
021A  EE                        OUT     DX,AL           ; SEND LO BYTE ERROR CODE

021B  BA 0201                   MOV     DX,201H
021E  EC                        IN      AL,DX           ; GET MODE BITS
021F  24 F0                     AND     AL,0F0H         ; ISOLATE BITS OF INTEREST
0221  8B E8                     MOV     BP,AX           ; SAVE MODE

0223  53                        PUSH    BX              ; SAVE ERROR AND MODE FLAGS
0224  50                        PUSH    AX
0225  52                        PUSH    DX
0226  B7 07                     MOV     BH,7            ; PAGE 7
0228  B4 02                     MOV     AH,2            ; SET CURSOR
022A  BA 1521                   MOV     DX,1521H        ; ROW 21, COL.33
022D  CD 10                     INT     10H
022F  BE 0293 R                 MOV     SI,OFFSET ERROR_ERR
0232  B9 0005                   MOV     CX,5            ; PRINT WORD "ERROR"
0235  2E: 8A 04         EM_0:   MOV     AL,CS:[SI]
0238  46                        INC     SI
0239  E8 02CB R                 CALL    PRT_HEX
023C  E2 F7                     LOOP    EM_0
                        ; LOOK FOR A BLANK SPACE TO POSSIBLY PUT CUSTOMER LEVEL ERRORS (IN
                        ; CASE OF MULTI ERROR)
023E  B6 16                     MOV     DH,16H
0240  B4 02             EM_1:   MOV     AH,2            ; SET CURSOR
0242  CD 10                     INT     10H             ; ROW 22, COL33 (OR ABOVE, IF
                                                        ; MULTIPLE ERRS)
                                                        ; DIFFERENT FOR MANUF MODE
0244  B4 08                     MOV     AH,8            ; READ CHARACTER THIS POSITION
0246  CD 10                     INT     10H
0248  FE C2                     INC     DL              ; POINT TO NEXT POSTION
024A  3C 20                     CMP     AL,' '          ; BLANK?
024C  75 F2                     JNE     EM_1            ; GO CHECK NEXT POSITION, IF NOT
024E  5A                        POP     DX              ; RECOVER ERROR POINTERS
024F  58                        POP     AX
0250  5B                        POP     BX
0251  8B D5                     MOV     DX,BP           ; RETRIEVE RUN MODE
0253  80 FA 40                  CMP     DL,01000000B    ; SERVICE MODE ?
0256  75 28                     JNZ     CUST_OUT        ; OUTPUT BYTE TO SCREEN
0258                    SERV_OUT:
0258  8A C7                     MOV     AL,BH           ; PRINT MSB
025A  53                        PUSH    BX



025B  E8 02BA R                 CALL    XPC_BYTE        ; DISPLAY IT
025E  5B                        POP     BX
025F  8A C3                     MOV     AL,BL           ; PRINT LSB
0261  E8 02BA R                 CALL    XPC_BYTE
0264  FA                        CLI
0265  B2 02                     MOV     DL,2            ; 2 BEEPS
0267  B3 01             EB:     MOV     BL,1            ; SHORT BEEP
0269  E8 0298 R                 CALL    BEEP
026C  E2 FE             EB0:    LOOP    EB0             ; WAIT (BEEPER OFF)
026E  FE CA                     DEC     DL              ; DONE YET?
0270  75 F5                     JNZ     EB              ; LOOP IF NOT
0272                    TOTLTPO:
0272  FA                        CLI                     ; DISABLE INTS.
0273  E4 61                     IN      AL,PORT_B       ;
0275  24 FC                     AND     AL,0FCH         ;
0277  E6 61                     OUT     PORT_B,AL       ; KILL HEARTBEAT
0279  2A C0                     SUB     AL,AL           ;
027B  E6 F2                     OUT     0F2H,AL         ; STOP DISKETTE MOTOR
027D  E6 A0                     OUT     0A0H,AL         ; DISABLE NMI
027F  F4                        HLT                     ; HALT

0280                    CUST_OUT:                       ;
0280  8A C4                     MOV     AL,AH           ; GET ERROR CHARACTER
0282  E8 02CB R                 CALL    PRT_HEX         ; DISPLAY IT
                                ASSUME  DS:XXDATA
0285  1E                        PUSH    DS
```

```
0286  50                        PUSH    AX
0287  B8  ---- R                MOV     AX,XXDATA
028A  8E D8                     MOV     DS,AX
028C  FE 06 0018 R              INC     POST_ERR        ; SET ERROR FLAG NON-ZERO
0290  58                        POP     AX
0291  1F                        POP     DS
                                ASSUME  DS:NOTHING
0292  CB                        RET                     ; RETURN TO CALLER
0293                    E_MSG_B ENDP

0293  45 52 52 4F 52    ERROR_ERR       DB      "ERROR"

                        ;---------------------------------------
                        ;       ROUTINE TO SOUND BEEPER
                        ; THIS PROC WILL SOUND THE BEEPER FOR A
                        ; TIME DETERMINED BY THE CONTENTS OF BL.
                        ; ON EXIT: AX AND BL ARE DESTROYED
                        ;---------------------------------------
0298                    BEEP    PROC    NEAR
0298  B0 B6                     MOV     AL,10110110B    ; SEL TIM 2,LSB,MSB,BINARY
029A  E6 43                     OUT     TIMER+3,AL      ; WRITE THE TIMER MODE REG
029C  B8 0533                   MOV     AX,533H         ; DIVISOR FOR 1000 HZ
029F  E6 42                     OUT     TIMER+2,AL      ; WRITE TIMER 2 CNT - LSB
02A1  8A C4                     MOV     AL,AH
02A3  E6 42                     OUT     TIMER+2,AL      ; WRITE TIMER 2 CNT - MSB
02A5  E4 61                     IN      AL,PORT_B       ; GET CURRENT SETTING OF PORT
02A7  8A E0                     MOV     AH,AL           ; SAVE THAT SETTING


02A9  0C 03                     OR      AL,03           ; TURN SPEAKER ON
02AB  E6 61                     OUT     PORT_B,AL
02AD  33 C9                     XOR     CX,CX           ; SET DELAY COUNT
02AF  E2 FE                G7:  LOOP    G7              ; DELAY BEFORE TURNING OFF
02B1  FE CB                     DEC     BL              ; DELAY CNT EXPIRED?
02B3  75 FA                     JNZ     G7              ; NO - CONTINUE BEEPING SPK
02B5  8A C4                     MOV     AL,AH           ; RECOVER VALUE OF PORT
02B7  E6 61                     OUT     PORT_B,AL
02B9  C3                        RET                     ; RETURN TO CALLER
02BA                    BEEP    ENDP

                        ;-------------------------------------------------
                        ; XPC_BYTE
                        ;       PROC TO PRINT A HEX BYTE TO THE SCREEN.
                        ;       THE CURSOR POSITION MUST BE SET ALREADY.
                        ;       PASS:
                        ;               AX      = BYTE TO PRINT
                        ;       RETURNS:
                        ;               CX AND AX DESTROYED, FLAGS TOO
                        ;-------------------------------------------------

02BA                    XPC_BYTE        PROC    NEAR
02BA  50                                PUSH    AX
02BB  B1 04                             MOV     CL,4
02BD  D2 E8                             SHR     AL,CL
02BF  E8 02C5 R                         CALL    XLAT_PR
02C2  58                                POP     AX
02C3  24 0F                             AND     AL,0FH
02C5                    XLAT_PR PROC NEAR
02C5  04 90                             ADD     AL,090H
02C7  27                                DAA
02C8  14 40                             ADC     AL,40H

02CA  27                                DAA
02CB                    PRT_HEX PROC NEAR
02CB  53                                PUSH    BX
02CC  B4 0E                             MOV     AH,14
02CE  B7 00                             MOV     BH,0
02D0  CD 10                             INT     10H
02D2  5B                                POP     BX
02D3  C3                                RET
02D4                    PRT_HEX ENDP
02D4                    XLAT_PR ENDP
02D4                    XPC_BYTE        ENDP

= 00B1                  BCLEN   =       ($-BC_START+1)/2

02D4                    INIT    ENDP

                        ;*****************************************************
                        ;*                                                   *
                        ;*              SOFTWARE INTERRUPT - 04DH            *
                        ;*****************************************************
                        ;*                                                   *
                        ;* PURPOSE:  To provide low-level BIOS support for   *


                        ;*          CVSD and LPC                             *
                        ;*                                                   *
                        ;*****************************************************
                        ;*                                                   *
                        ;*      AH = 0    RESET CARD                         *
                        ;*                                                   *
                        ;*      AH = 1    CVSD (Continously Variable Slope Delta)  *
                        ;*                                                   *
                        ;*                AL = 0  - CVSD RECORD (using speed table)  *
                        ;*                          DS:SI  - segment:offset  *
                        ;*                                   (note 1 below)  *
                        ;*                             BL  - table speed     *
                        ;*                                   = 0 => 1800 bytes/sec  *
                        ;*                                   = 1 => 2400 bytes/sec  *
                        ;*                                   = 2 => 3000 bytes/sec  *
                        ;*                                   = 3 => 3600 bytes/sec  *
                        ;*                                   = 4 => 4200 bytes/sec  *
                        ;*                                   = 5 => 4800 bytes/sec  *
                        ;*                          CX  - byte count (note 2 below)  *
                        ;*                                                   *
                        ;*                AL = 1  - CVSD PLAYBACK  (using speed table)  *
                        ;*                          DS:SI  - segment:offset  *
                        ;*                                   (note 1 below)  *
                        ;*                             BL  - table speed     *
                        ;*                                   = 0 => 1800 bytes/sec  *
                        ;*                                   = 1 => 2400 bytes/sec  *
                        ;*                                   = 2 => 3000 bytes/sec  *
                        ;*                                   = 3 => 3600 bytes/sec  *
                        ;*                                   = 4 => 4200 bytes/sec  *
                        ;*                                   = 5 => 4800 bytes/sec  *
                        ;*                          CX  - byte count (note 2 below)  *
                        ;*                                                   *
                        ;*                AL = 2  - CVSD RECORD (using user speed)  *
                        ;*                          DS:SI  - segment:offset  *
                        ;*                                   (note 1 below)  *
                        ;*                             BX  - user speed divisor  *
                        ;*                                   (note 3 below)  *
```

```
;*                                  CX  - byte count (note 2 below) *
;*                                                                  *
;*                  AL = 3  - CVSD PLAYBACK  (using user speed)     *
;*                            DS:SI  - segment:offset               *
;*                                     (note 1 below)               *
;*                               BX  - user speed divisor           *
;*                                     (note 3 below)               *
;*                               CX  - byte count (note 2 below)    *
;*                                                                  *
;*                                                                  *
;*                  LPC  (Linear Predictive Coding)                 *
;*                                                                  *
;*      AH = 2      Interrupt Driven LPC (Background)               *
;*                                                                  *
;*                  AL = 0  - LPC STATUS                            *
;*                                                                  *
;*                                                                  *
;*                  AL = 1  - LPC SPEAK - INTR  (INDEX)             *
;*                            BX  - word number from index          *
;*                            ( 1 <= BX <= 196 )                    *
;*                                                                  *
;*                  AL = 2  - LPC SPEAK - INTR  (BUFFER)            *
;*                            DS:SI  - beg of bfr (seg:offset)      *
;*                                     (note 1 below)               *
;*                            CX  - number of bytes in the LPC      *
;*                            word to be spoken. CX must not be     *
;*                            lagrer than 4095 bytes.               *
;*                                                                  *
;*      AH = 3      Polled LPC (Foreground)                         *
;*                                                                  *
;*                  AL = 0  - LPC STATUS                            *
;*                                                                  *
;*                  AL = 1  - LPC SPEAK - INTR  (INDEX)             *
;*                            BX  - word number from index          *
;*                            ( 1 <= BX <= 196 )                    *
;*                                                                  *
;*                  AL = 2  - LPC SPEAK - INTR  (BUFFER)            *
;*                            DS:SI  - beg of bfr (seg:offset)      *
;*                                     (note 1 below)               *
;*                            CX  - number of bytes in the LPC      *
;*                            word to be spoken. CX must not be     *
;*                            lagrer than 4095 bytes.               *
;*                                                                  *
;******************************************************************

;******************************************************************
;* Note 1 - DS:SI must be set up by the user to address valid     *
;*          memory locations. Checking for valid parameters       *
;*          is not done in the BIOS.                              *
;*                                                                *
;* Note 2 - CX is the byte count in CVSD. If CX = 0 then 64K      *
;*          bytes will be processed.                              *
;*                                                                *
;* Note 3 - BX = User speed divisor when AH = 1 and AL = 2 or     *
;*          3 (CVSD).  Here BX is the number which the timer      *
;*          counts down from between CVSD samples.  The clock     *
;*          frequency is 4.77MHz.  This frequency divided by      *
;*          the (divisor*8) gives the byte sampling rate.         *
;*          Speeds slower than 1800 bytes per second (BX=14BH)    *
;*          or faster than 4800 bytes per second (BX=7CH)         *
;*                                                                *
;*          are not supported.                                    *
;*                                                                *
;* Note 4 - registers preserved during this call:                 *
;*          CS,SS,DS,ES,SI,DI,DX,CX,BX,BP                         *
;*          All other registers destroyed.                        *
;*                                                                *
;* Note 5 - AL returns:                                           *
;*          00H - if everything o.k.                              *
;*          01H - if undefined command                            *
;*                                                                *
;*                                                                *
;*          02H - if LPC speak in progress                        *
;*          03H - if ACL error (stuck)                            *
;*          04H - if LPC index out of range                       *
;*          05H - if CVSD speed out of range                      *
;*          06H - if timeout waiting for LPC READY                *
;*                                                                *
;******************************************************************

02D4                    START   PROC    FAR

                        ; -> CLEAR DIRECTION FLAG & SAVE REGISTERS

02D4  FC                        CLD                     ;CLEAR DIRECTION FLAG
02D5  55                        PUSH    BP              ;
02D6  56                        PUSH    SI              ;SAVE REGISTERS
02D7  57                        PUSH    DI              ;
02D8  52                        PUSH    DX              ;
02D9  51                        PUSH    CX              ;
02DA  53                        PUSH    BX              ;


                        ; -> DECODE REQUESTED FUNCTION & BRANCH TO APPROPIATE CODE

02DB  33 ED                     XOR     BP,BP           ;BP INDICATES LPC BACK/FOREGROUND
02DD  80 FC 03                  CMP     AH,LPC_FN_FORE  ;LPC FOREGROUND FUNCTION ?
02E0  75 03                     JNE     F00             ;NO. CONTINUE DECODE OF FUNCTION
02E2  E9 04AD R                 JMP     LPC000          ;YES. GO TO HANDLE LPC
02E5                    F00:
02E5  80 FC 02                  CMP     AH,LPC_FN       ;LPC FUNCTION ?
02E8  75 03                     JNE     F0              ;NO. CONTINUE DECODE OF FUNCTION
02EA  E9 04AE R                 JMP     LPC00           ;YES. GO TO HANDLE LPC

02ED  80 FC 01          F0:     CMP     AH,CVSD_FN      ;CVSD FUNCTION ?
02F0  75 03                     JNE     F1              ;NO. CONTINUE DECODE OF FUNCTION
02F2  E9 0388 R                 JMP     CVSD00          ;YES. GO TO HANDLE CVSD

02F5  80 FC 00          F1:     CMP     AH,RST_FN       ;RESET CARD FUNCTION ?
02F8  75 05                     JNE     ERROR           ;NO. GO TO SET ERROR CODE
02FA  E8 0308 R                 CALL    RST_TALKER      ;YES. GO TO HANDLE CARD RESET
02FD  EB 02                     JMP     SHORT EXIT      ;GO TO EXIT CODE

02FF  B0 01             ERROR:  MOV     AL,BAD_CMD      ;SET AL = BAD COMMAND

                        ; -> RESTORE REGISTERS & EXIT

0301  5B                EXIT:   POP     BX              ;RESTORE REGISTERS
0302  59                        POP     CX              ;
0303  5A                        POP     DX              ;
```

```
0304  5F                        POP     DI              ;
0305  5E                        POP     SI              ;
0306  5D                        POP     BP              ;


0307  CF                        IRET

0308                      START   ENDP

                          ;*****************************************************************
                          ;*                                                               *
                          ;*       NAME:  RESET CARD                                       *
                          ;*                                                               *
                          ;*    PURPOSE:  SET HARDWARE INTO A KNOWN STATE                  *
                          ;*                                                               *
                          ;*    LINKAGE:  SOFTWARE INTERRUPT   (INT 04DH  WITH AH = 0)     *
                          ;*              - OR -                                           *
                          ;*              BY SUBROUTINE CALL   (CALL  RST_TALKER)          *
                          ;*                                                               *
                          ;*     INPUTS:  AH = 0 IF USING 04DH INTERRUPT LINKAGE           *
                          ;*                                                               *
                          ;*    OUTPUTS:  AL CONTAINS A RETURN CODE                        *
                          ;*              00H - IF EVERYTHING O.K.                         *
                          ;*              03H - IF ACL ERROR (STUCK)                       *
                          ;*              06H - TIMEOUT WAITING FOR LPC READY              *
                          ;*                                                               *
                          ;*       EXIT:  RETURN FROM SUBROUTINE                           *
                          ;*              WITH RETURN CODE SET IN AL                       *
                          ;*                                                               *
                          ;*    PROCESS:  (1) - MASK OFF INTR 1 ON THE SYSTEM'S 8259       *
                          ;*              (2) - DISABLE ALL 32 ATTACHMENT$ ACLS &          *
                          ;*                    ENABLE ACL                                 *
                          ;*              (3) - SET 8255 MODE:  PORT A  - OUT              *
                          ;*                                    PORT B  - OUT              *
                          ;*                                    PORT CL - IN               *
                          ;*                                    PORT CU - OUT              *
                          ;*              (4) - SET ROS PAGE 0 & SET CHANNEL MUX = LPC     *
                          ;*              (5) - SET CVSD DECODE ON                         *
                          ;*              (6) - WRITE 10 RESET CMDS TO 5220                *
                          ;*              (7) - SET CVSD SPEED COUNTER MODE TO:            *
                          ;*                    SQUARE WAVE MODE (MODE 3)                  *
                          ;*                    READ/WRITE LSB FIRST, THEN MSB             *
                          ;*                    BINARY COUNTER                             *
                          ;*              (8) - INITIALIZE CVSD SPEED COUNTER TO:          *
                          ;*                    3000 BYTES/SEC                             *
                          ;*              (9) - SET CVSD FRAME COUNTER MODE TO:            *
                          ;*                    RATE GENERATOR (MODE 2)                    *
                          ;*                    READ/WRITE LSB FIRST, THEN MSB             *
                          ;*                    BINARY COUNTER                             *
                          ;*             (10) - INITIALIZE CVSD FRAME COUNTER TO:          *
                          ;*                    DIVIDE-BY-8                                *
                          ;*             (11) - SET INTR PULSE COUNTER MODE TO:            *
                          ;*                    HARDWARE ONE-SHOT (MODE 1)                 *
                          ;*                    READ/WRITE LSB FIRST, THEN MSB             *
                          ;*                    BINARY COUNTER                             *
                          ;*             (12) - INITIALIZE INTR PULSE COUNTER TO 8         *
                          ;*                                                               *
                          ;*             (13) - SET AL = RETURN CODE & EXIT                *
                          ;*                                                               *
                          ;*****************************************************************
                          ;*     NOTES: - THE FOLLOWING REGISTERS ARE DESTROYED:           *
                          ;*                IF SUBROUTINE LINKAGE:  AX, CX & DX            *
                          ;*                IF INTERRUPT LINKAGE:   AX                     *
                          ;*****************************************************************

0308                      RST_TALKER PROC NEAR

                          ; -> MASK OFF INTR 1 ON THE SYSTEM'S 8259

0308  E4 21                        IN      AL,PORT_21H     ;MASK OFF INTR 1
030A  0C 02                        OR      AL,INT1_OFF     ;
030C  E6 21                        OUT     PORT_21H,AL     ;


                          ; -> DISABLE ALL 32 ATTACHMENTS ACLS & ENABLE CARD ACL

030E  E8 0357 R                    CALL    ATTACH_ACL      ;DISABLE ALL ATTACHMENTS ACLS
                                                           ;& ENABLE CARD ACL
                                                           ;ANY ERRORS ?
0311  72 43                        JC      RST_XX          ;YES. RETURN WITH ERROR CODE


                          ; -> SET 8255 MODE

0313  B0 81                        MOV     AL,LPC_OUT      ;SET 8255:  PORT A  - OUT
0315  BA FB9B                      MOV     DX,CWREG        ;           PORT B  - OUT
0318  EE                          OUT     DX,AL           ;           PORT CU - OUT
                          ;                                 ;           PORT CL - IN


                          ; -> SET ROS PAGE 0 & SET CHANNEL MUX =  LPC

0319  52                          PUSH    DX              ;SAVE CWREG
031A  BA FB98                      MOV     DX,PORTA        ;SET CHANNEL MUX = LPC
031D  B0 00                        MOV     AL,LPC+PAGE0    ;& SELECT ROS PAGE 0
031F  EE                          OUT     DX,AL           ;
0320  5A                          POP     DX              ;RESTORE CWREG


                          ; -> SET 8255 PORT C - CVSD DECODE ON

0321  B0 0C                        MOV     AL,DECODE       ;SET CVSD DECODE ON
0323  EE                          OUT     DX,AL           ;

                          ; -> WAIT FOR LPC TO FINISH PROCESSING DATA IN BUFFER

0324  E8 037A R                   CALL    WAIT_FOR_LPC
0327  75 0C                       JNZ     LPC_RDY_ERR

                          ; -> WRITE 10 RESET CMDS TO 5220



0329  B9 000A                      MOV     CX,10           ;SET RESET CMD CNT TO 10
032C  B0 FF                        MOV     AL,RST_5220     ;ISSUE RESET COMMAND TO 5220
032E  E8 065D R           WRT_FF:  CALL    LPCW_10         ;REPEAT IF NO ERRORS (LOOPZ)
0331  E1 FB                        LOOPZ   WRT_FF          ;
```

```
0333  74 03                     JZ      RST20           ;IF NO ERRORS, CONTINUE
0335                      LPC_RDY_ERR:
0335  B0 06                     MOV     AL,LPCRDY_ERR   ;SET ERROR CODE IN AL & EXIT
0337  C3                       RET                     ;(ERROR WAITING FOR LPC READY)

                          ; -> SET MODE FOR SPEED COUNTER (CVSD_CLK)

0338                      RST20:
0338  B9 FB9C                   MOV     CX,CVSD_CLK     ;SET CX=CLOCK TO BE INITIALIZED
033B  8B 0021 R                 MOV     AX,CS:[OFFSET SPEED_TBL+4]      ;SPEED FOR INIT
033E  8B D8                     MOV     BX,AX           ;INTO BX
0340  B0 36                     MOV     AL,CTR0+RW_LSBMSB+MD3+BINARY    ;MODE INTO AL
0342  E8 011D R                 CALL    INIT_TIMER      ;CALL ROUTINE TO INIT TIMER

0345  41                        INC     CX              ;SET CX=CVSD FRAME
0346  BB 0008                   MOV     BX,8            ;DIVISOR
0349  B0 74                     MOV     AL,CTR1+RW_LSBMSB+MD2+BINARY
034B  E8 011D R                 CALL    INIT_TIMER      ;INIT TIMER CHANNEL 1

034E  41                        INC     CX              ;SET CX=LPC INTERRUPT TIMER
                                                        ;BX = 8 FROM LAST INIT
034F  B0 B2                     MOV     AL,CTR2+RW_LSBMSB+MD1+BINARY
0351  E8 011D R                 CALL    INIT_TIMER      ;INIT TIMER CHANNEL 2

                          ; -> SET AL = RETURN CODE & RETURN

0354  B0 00                     MOV     AL,OK           ;SET O.K. RETURN CODE IN AL
0356  C3                  RST_XX:  RET                     ;RETURN

0357                      RST_TALKER ENDP

                          ;*****************************************************************
                          ;*                                                               *
                          ;*  ATTACH_ACL: THIS PROCEDURE ACCOMPLISHES THE FOLLOWING:       *
                          ;*              - DISABLES ALL ATTACHMENTS ACLS                  *
                          ;*              - MAKES SURE CARD ACL IS DISABLED                *
                          ;*              - ENABLES ACL                                    *
                          ;*              - MAKES SURE CARD ACL IS ENABLED                 *
                          ;*              - IF ERROR, 'ACL_ERROR' IS RETURNED IN AL        *
                          ;*                AND CARRY FLAG IS SET                          *
                          ;*              THIS PROCEDURE DESTROYS REGISTERS: AL, CX & DX   *
                          ;*                                                               *
                          ;*****************************************************************
                          ;*                                                               *
                          ;*  CHKTKR_ACL: THIS PROCEDURE ACCOMPLISHES THE FOLLOWING:       *
                          ;*              - MAKES SURE CARD ACL IS ENABLED                 *
                          ;*              - IF ERROR, 'ACL_ERROR' IS RETURNED IN AL        *
                          ;*                AND CARRY FLAG IS SET                          *



                          ;*              THIS PROCEDURE DESTROYS REGISTERS: AL & DX       *
                          ;*                                                               *
                          ;*****************************************************************

0357                      ATTACH_ACL PROC NEAR

                          ; -> DISABLE ALL 32 ATTACHMENTS ACLS

0357  B9 0020                   MOV     CX,32           ;DISABLE ALL ATTACHMENTS ACLS
035A  BA FF9F                   MOV     DX,TKR_ACL      ;
035D  B0 00                     MOV     AL,CHN_OFF      ;

035F  EE                  NXT_ACL: OUT     DX,AL           ;
0360  81 EA 0800                SUB     DX,800H         ;
0364  E2 F9                     LOOP    NXT_ACL         ;

                          ; -> MAKE SURE ACL IS DISABLED

0366  EC                        IN      AL,DX           ;
0367  A8 01                     TEST    AL,ACL_OFF      ;ACL DISABLED ?
0369  74 0B                     JZ      ERROR_0         ;NO. ERROR

                          ; -> ENABLE ACL

036B  24 FD                     AND     AL,0FDH         ;
036D  EE                        OUT     DX,AL           ;ENABLE ACL

036E                      CHKTKR_ACL PROC NEAR

                          ; -> MAKE SURE ACL IS ENABLED

036E  BA FF9F                   MOV     DX,TKR_ACL      ;READ ACL
0371  EC                        IN      AL,DX           ;
0372  A8 01                     TEST    AL,ACL_OFF      ;ACL ENABLED ?
0374  74 03                     JZ      RET_0           ;YES. RETURN

0376  B0 03               ERROR_0: MOV     AL,ACL_ERROR    ;SAVE ERROR CODE IN CL
0378  F9                        STC                     ;SET CARRY (ERROR) FLAG ON
0379  C3                  RET_0:   RET                     ;RETURN

037A                      CHKTKR_ACL ENDP

037A                      ATTACH_ACL ENDP

                          ;*****************************************************************
                          ;* WAIT_FOR_LPC                                                  *
                          ;*      THIS PROC WAITS FOR TS ON THE 5220 TO INDICATE LPC       *
                          ;*      SPEECH PROCESSING COMPLETION. IT WILL RETRY ONLY A       *
                          ;*      LIMITED NUMBER OF TIMES.                                 *
                          ;*      ON ENTRY: NO REQUIREMENTS                                *
                          ;*      ON EXIT: AX,BH,CX,DX ARE DESTROYED                       *



                          ;*              ZERO FLAG SET IF LPC DID NOT COMPLETE IN TIME.   *
                          ;*              ZERO FLAG RESET IF LPC COMPLETED.                *
                          ;*****************************************************************

037A                      WAIT_FOR_LPC    PROC    NEAR
037A  B9 1000                   MOV     CX,1000H        ;LOOP COUNT
037D                      LPC_BUF_NOT_EMPTY:
037D  E9 0A9C R                 JMP     SETUP_FLAG
0380                      FLAG_SETUP:
0380  75 05                     JNZ     LPC_CHIP_FAIL
0382  F6 C4 80                 TEST    AH,TALK_ON      ;WAIT FOR TS TO GO INACTIVE LOW
0385  E0 F6                     LOOPNE  LPC_BUF_NOT_EMPTY
0387                      LPC_CHIP_FAIL:
0387  C3                       RET
0388                      WAIT_FOR_LPC    ENDP
                          ;*****************************************************************
                          ;*                                                               *
```

```
;*      NAME:  CVSD DRIVER                                              *
;*                                                                      *
;*   PURPOSE:  TO PROVIDE LOW-LEVEL BIOS SUPPORT                        *
;*             CVSD                                                     *
;*                                                                      *
;*   LINKAGE:  SOFTWARE INTERRUPT   (INT 04DH  WITH AH = 1)            *
;*                                                                      *
;*    INPUTS:  AL - CONTAINS THE CVSD FUNCTION                          *
;*                  =0  FOR CVSD RECORD (SPEED TABLE)                   *
;*                  =1  FOR CVSD PLAYBACK (SPEED TABLE)                 *
;*                  =2  FOR CVSD RECORD (USER SPEED)                    *
;*                  =3  FOR CVSD PLAYBACK (USER SPEED)                  *
;*             BX - USER SPEED DIVISOR (IF AL = 2 OR 3)                 *
;*                       HERE BX IS THE NUMBER WHICH THE TIMER          *
;*                       COUNTS DOWN FROM BETWEEN CVSD SAMPLES.         *
;*                       THE A CLOCK FREQUENCY IS 4.77MHz. THIS         *
;*                       FREQUENCY DIVIDED BY THE (DIVISOR*8)           *
;*                       GIVES THE BYTE SAMLPING RATE.                  *
;*             BL - TABLE SPEED  (IF AL = 0 OR 1)                       *
;*                  = 0 => 1800 BYTES/SEC                               *
;*                  = 1 => 2400 BYTES/SEC                               *
;*                  = 2 => 3000 BYTES/SEC                               *
;*                  = 3 => 3600 BYTES/SEC                               *
;*                  = 4 => 4200 BYTES/SEC                               *
;*                  = 5 => 4800 BYTES/SEC                               *
;*             CX  - BYTE COUNT (LENGTH) OF SPEECH BUFFER               *
;*          DS:SI  - SEGMENT:OFFSET OF SPEECH BUFFER                    *
;*                                                                      *
;*   OUTPUTS:  AL CONTAINS A RETURN CODE                                *
;*             00H - IF EVERYTHING O.K.                                 *
;*             01H - IF UNDEFINED COMMAND                               *
;*             02H - IF LPC SPEAK IN PROGRESS                           *
;*             03H - IF CARD ACL ERROR (STUCK)                          *
;*             05H - IF CVSD SPEED OUT OF RANGE                         *
;*                                                                      *
;*      EXIT:  INTERRUPT RETURN WITH RETURN CODE SET IN AL              *
```

```
;*                                                                      *
;*   PROCESS:  (1) - DECODE CVSD FUNCTION AND SET CVSD FLAG             *
;*                   IN DI=0000H IF CVSD RECORD (AL = 0 OR 2)           *
;*                   00FFH IF CVSD PLAYBACK (AL = 1 OR 3)               *
;*                 - IF INVALID FUNCTION, EXIT WITH RETURN              *
;*                   CODE IN AL = 01H                                   *
;*             (2) - CHECK FOR LPC IN PROGRESS.  IF SO, EXIT            *
;*                   WITH RETURN CODE IN AL = 02H                       *
;*             (3) - IF CVSD PLAYBACK (AL = 1 OR 3), MAKE SURE          *
;*                   ACL IS ENABLED.  IF NOT, EXIT                      *
;*                   WITH RETURN CODE IN AL = 03H                       *
;*             (4) - SET CVSD SPEED.  IF SPEED OUT OF RANGE,            *
;*                   EXIT WITH RETURN CODE IN AL = 05H                  *
;*             (5) - SET CHANNEL MUX = CVSD                             *
;*             (6) - SET SYSTEM SPEAKER SWITCH (PORT 61H)               *
;*                   TO AUDIO CHANNEL                                   *
;*             (7) - DISABLE ALL INTERRUPTS AND SAVE TIME OF            *
;*                   DAY                                                *
;*             (8) - SEE IF CVSD RECORD OR PLAYBACK:                    *
;*                   * IF CVSD RECORD:                                  *
;*                     (A) - TURN OFF AUDIO CHANNEL                     *
;*                     (B) - SET CVSD ENCODE ON                         *
;*                     (C) - WAIT FOR FRAME 1 -> 0                      *
;*                     (D) - READ DATA BYTE                             *
;*                     (E) - CHECK FOR SYNC CHARACTER                   *
;*                     (F) - DO STEPS (B) - (D) WHILE SYNC              *
;*                           SEQUENCE FOUND. WAIT FOR AT MOST           *
;*                           9600 SAMPLES.                              *
;*                     (G) - WAIT FOR FRAME 1 -> 0                      *
;*                     (H) - READ DATA BYTE & SAVE IN BUFFER            *
;*                     (I) - POINT TO NEXT BUFFER LOCATION              *
;*                     (J) - DO STEPS (F) - (H) UNTIL COUNT             *
;*                           EXHAUSTED                                  *
;*                   * IF CVSD PLAYBACK:                                *
;*                     (A) - SET CVSD DECODE ON                         *
;*                     (B) - WAIT FOR FRAME 0 -> 1                      *
;*                     (C) - WRITE DATA BYTE                            *
;*                     (D) - POINT TO NEXT DATA BYTE                    *
;*                     (E) - DO STEPS (B) - (D) UNTIL COUNT             *
;*                           EXHAUSTED                                  *
;*                     (F) - WAIT FOR FRAME 0 -> 1                      *
;*                     (G) - WRITE A BYTE OF 55H (SILENCE)              *
;*                     (H) - WAIT FOR FRAME 0 -> 1                      *
;*             (9) - SET CVSD DECODE ON                                 *
;*            (10) - ENABLE INTERRUPTS AND RESTORE TIME OF DAY          *
;*            (11) - EXIT WITH RETURN CODE IN AL = 00H                  *
;*                                                                      *
;************************************************************************
;*      NOTES: - REGISTERS PRESERVED DURING THIS CALL:                  *
;*               CS,SS,DS,ES,SI,DI,DX,CX,BX                             *
;*               ALL OTHER REGISTERS DESTROYED.                         *
;************************************************************************
```

```
                        ; -> SAVE FUNCTION
0388                    CVSD00:
0388  55                         PUSH    BP              ;SAVE BP
0389  06                         PUSH    ES              ;SAVE ES
038A  50                         PUSH    AX              ;SAVE CVSD FUNCTION TEMPORARILY

                        ; DECODE CVSD FN & SET DI = 0000H FOR CVSD RECORD (AL = 0 OR 2)
                        ;                           00FFH FOR CVSD PLAYBACK (AL = 1 OR 3)

038B  BF 0000                    MOV     DI,CVSDR        ;SET DI = CVSD RECORD
038E  3C 00                      CMP     AL,CVSDR_TBL    ;CVSD RECORD USING TABLE SPEED ?
0390  74 15                      JE      CVSD20          ;YES. CONTINUE
0392  3C 02                      CMP     AL,CVSDR_USER   ;CVSD RECORD USING USER SPEED ?
0394  74 11                      JE      CVSD20          ;YES. CONTINUE
0396  BF 00FF                   MOV     DI,CVSDW        ;SET DI = CVSD PLAYBACK
0399  3C 01                      CMP     AL,CVSDW_TBL    ;CVSD PLAYBACK USING TABLE SPEED ?
039B  74 0A                      JE      CVSD20          ;YES. CONTINUE
039D  3C 03                      CMP     AL,CVSDW_USER   ;CVSD PLAYBACK USING USER SPEED ?
039F  74 06                      JE      CVSD20          ;YES. CONTINUE
03A1  58                         POP     AX              ;RE-ADJUST STACK
03A2  B0 01                      MOV     AL,BAD_CMD      ;SET AL = BAD COMMAND
03A4  E9 04AB R                  JMP     CVSDX0          ;GO TO EXIT

                        ; -> CHECK FOR `LPC IN PROGRESS'

03A7  BA FB98           CVSD20:  MOV     DX,PORTA        ;READ 8255 PORT A
03AA  EC                         IN      AL,DX           ;
```

```
03AB  A8 80                     TEST    AL,TALK_LPC     ;LPC IN PROGRESS ?
03AD  74 06                     JZ      CVSD25          ;NO. CONTINUE
03AF  58                        POP     AX              ;ADJUST STACK
03B0  80 02                     MOV     AL,LPC_INPROG   ;SET AL = LPC IN PROGRESS (02H)
03B2  E9 04A8 R                 JMP     CVSDXO          ;GO TO EXIT

                        ; -> MAKE SURE ACL IS ENABLED IF CVSD PLAYBACK

03B5  83 FF 00          CVSD25: CMP     DI,CVSDR        ;CVSD RECORD ?
03B8  74 10                     JE      CVSD30          ;YES. CONTINUE
03BA  E8 036E R                 CALL    CHKTKR_ACL      ;ACL ENABLED ?
03BD  73 0B                     JNC     CVSD30          ;YES. CONTINUE
03BF  51                        PUSH    CX              ;SAVE SPEECH BYTE CNT
03C0  E8 0357 R                 CALL    ATTACH_ACL      ;RESET ALL ACLS
                        ;                               ;& ENABLE ACL
03C3  59                        POP     CX              ;RESTORE SPEECH BYTE CNT
                        ;                               ;ERRORS ?
03C4  73 04                     JNC     CVSD30          ;NO. CONTINUE
03C6  5B                        POP     BX              ;ADJUST STACK
03C7  E9 04A8 R                 JMP     CVSDXO          ;YES. EXIT


                        ; -> SET SPEED



03CA  58                CVSD30: POP     AX              ;GET CVSD FUNCTION IN AX
03CB  3C 01                     CMP     AL,CVSDW_TBL    ;CVSD SPEED FROM TABLE ?
03CD  8B C3                     MOV     AX,BX           ;PICK UP USER SPEED IN AX
03CF  77 13                     JA      CVSD50          ;CU TO SET USER SPEED

03D1  80 FB 05          CVSD40: CMP     BL,SPEED_MAX    ;SPEED WITHIN RANGE ?
03D4  76 05                     JBE     CVSD45          ;YES. GO TO SET SPEED
03D6  B0 05                     MOV     AL,SPEED_ERR    ;SET SPEED OUT OF RANGE ERROR
03D8  E9 04A8 R                 JMP     CVSDXO          ;EXIT CODE

03DB  B7 00             CVSD45: MOV     BH,0            ;PICK UP SPEED IN AX
03DD  D0 E3                     SHL     BL,1            ;
03DF  2E: 8B 87 001D R          MOV     AX,CS:[BX+OFFSET SPEED_TBL]
03E4  BA FB9C           CVSD50: MOV     DX,CVSD_CLK     ;OUTPUT SPEED (LSB, THEN MSB)
03E7  EE                        OUT     DX,AL           ;
03E8  EB 00                     JMP     $+2             ;(DELAY)
03EA  8A C4                     MOV     AL,AH           ;
03EC  EE                        OUT     DX,AL           ;


                        ; -> SET 8255 PORT A CVSD ON

03ED  BA FB98                   MOV     DX,PORTA        ;READ PORT A
03F0  EC                        IN      AL,DX           ;
03F1  24 FC                     AND     AL,CLR_MUX      ;CLEAR CHANNEL MUX
03F3  0C 01                     OR      AL,CVSD         ;TURN CVSD ON
03F5  EE                        OUT     DX,AL           ;OUTPUT TO PORT A

                        ; -> SET SYSTEM SPEAKER SWITCH TO AUDIO CHANNEL

03F6  E4 61                     IN      AL,PORT_61H     ;READ & SAVE PORT 61H
03F8  24 9F                     AND     AL,CLR_SPKSW    ;CLEAR SPEAKER SWICH BITS
03FA  0C 40                     OR      AL,AUDIO_CHN    ;OR IN 'AUDIO CHANNEL' BITS
03FC  E6 61                     OUT     PORT_61H,AL     ;OUTPUT TO PORT 61H

                        ; -> DISABLE ALL INTERRUPTS

03FE  53                        PUSH    BX
03FF  B3 FF                     MOV     BL,0FFH         ;MASK ALL INTERRUPTS ON 8259
0401  E8 0980 R                 CALL    NMIOFF          ;MASK EVERYTHING
0404  8B D3                     MOV     DX,BX           ;SAVE ORIG. 8259 MASK
0406  5B                        POP     BX
0407  50                        PUSH    AX              ;SAVE TIMER VALUE ON STACK
0408  52                        PUSH    DX              ;SAVE 8259 MASK


                        ; -> CHECK FOR CVSD RECORD/PLAYBACK

                        ;    NOTE: PLAYBACK => DECODE
                        ;          RECORD  => ENCODE

0409  8B C7                     MOV     AX,DI           ;GET CVSD FN INDICATOR IN AX



040B  3C 00                     CMP     AL,CVSDR        ;CVSD RECORD ?
040D  74 40                     JE      READ            ;YES. GO TO CVSD RECORD CODE
040F  EB 1A                     JMP     SHORT WRITE     ;NO. GO TO CVSD PLAYBACK CODE


                        ; NOTES: - FRAME_01 IS A PROCEDURE TO WAIT FOR A 0 TO 1
                        ;          TRANSITION ON CVSD FRAME.
                        ;        - BP = 8255 PORT C
                        ;        - AH = FRAME_HI
                        ;        - AX & DX REGISTERS ARE DESTROYED BY THIS CALL

0411                    FRAME_01 PROC   NEAR

0411  8B D5                     MOV     DX,BP           ;SET DX = 8255 PORT C
0413  EC                WAIT0:  IN      AL,DX           ;READ CVSD FRAME
0414  22 C4                     AND     AL,AH           ;FRAME LOW ?
0416  75 FB                     JNZ     WAIT0           ;NO. WAIT FOR FRAME LOW
0418  EC                WAIT1:  IN      AL,DX           ;READ CVSD FRAME
0419  22 C4                     AND     AL,AH           ;FRAME HIGH ?
041B  74 FB                     JZ      WAIT1           ;NO. WAIT FOR FRAME HIGH
041D  C3                        RET

041E                    FRAME_01 ENDP



                        ; NOTES: - FRAME_10 IS A PROCEDURE TO WAIT FOR A 1 TO 0
                        ;          TRANSITION ON CVSD FRAME.
                        ;        - BP = 8255 PORT C
                        ;        - AH = FRAME_HI
                        ;        - AX & DX REGISTERS ARE DESTROYED BY THIS CALL

041E                    FRAME_10 PROC   NEAR

041E  8B D5                     MOV     DX,BP           ;SET DX = 8255 PORT C
0420  EC                WAIT_1: IN      AL,DX           ;READ CVSD FRAME
0421  22 C4                     AND     AL,AH           ;FRAME HIGH ?
0423  74 FB                     JZ      WAIT_1          ;NO. WAIT FOR FRAME HI
0425  EC                WAIT_0: IN      AL,DX           ;READ CVSD FRAME
0426  22 C4                     AND     AL,AH           ;FRAME LOW ?
```

```
0428  75 FB                         JNZ     WAIT_0          ;NO. WAIT FOR FRAME LO
042A  C3                            RET

0428                          FRAME_10 ENDP

                              ; NOTE: - CAUTION MUST BE TAKEN WHEN CHANGING THIS PART OF
                              ;         THE CODE SINCE IT IS VERY TIME DEPENDENT

0428                          WRITE:

0428  B0 0C                         MOV     AL,DECODE       ;SET CVSD DECODE ON



042D  BA FB9B                       MOV     DX,CWREG        ;
0430  EE                            OUT     DX,AL           ;

0431  BD FB9A                       MOV     BP,PORTC        ;SET BP = 8255 PORT C
0434  B4 04                         MOV     AH,FRAME_HI     ;SET AH = FRAME_HI
0436  BF FF98                       MOV     DI,SHIFTREG     ;SET DI = SHIFTREG

0439  E8 0411 R               WRITEX: CALL    FRAME_01        ;WAIT FOR FRAME 0 -> 1
043C  8B D7                         MOV     DX,DI           ;SET DX = SHIFT REG
043E  AC                            LODSB                   ;GET DATA BYTE IN AL & INCR SI
043F  EE                            OUT     DX,AL           ;WRITE DATA BYTE
0440  E2 F7                         LOOP    WRITEX          ;CONTINUE UNTIL CNT EXHAUSTED

0442  E8 0411 R                     CALL    FRAME_01        ;WAIT FOR FRAME 0 -> 1
0445  8B D7                         MOV     DX,DI           ;SET DX = SHIFT REG
0447  B0 55                         MOV     AL,055H         ;SET AL = 055H (LAST BYTE)
0449  EE                            OUT     DX,AL           ;WRITE DATA BYTE

044A  E8 0411 R                     CALL    FRAME_01        ;WAIT FOR FRAME 0 -> 1

044D  EB 46                         JMP     SHORT CVSDXA    ;GO TO EXIT CVSD CODE

                              ; NOTE: - CAUTION MUST BE TAKEN WHEN CHANGING THIS PART OF
                              ;         THE CODE SINCE IS VERY TIME DEPENDENT

044F                          READ:

044F  E4 61                         IN      AL,PORT_61H     ;TURN OFF AUDIO CHANNEL
0451  24 9F                         AND     AL,CLR_SPKSW    ;
0453  E6 61                         OUT     PORT_61H,AL     ;

0455  B0 0D                         MOV     AL,ENCODE       ;SET CVSD ENCODE ON
0457  BA FB9B                       MOV     DX,CWREG        ;
045A  EE                            OUT     DX,AL           ;

045B  1E                            PUSH    DS              ;SET ES = DS
045C  07                            POP     ES              ;
045D  BD FB9A                       MOV     BP,PORTC        ;SET BP = 8255 PORT C
0460  B4 04                         MOV     AH,FRAME_HI     ;SET AH = FRAME_HI
0462  8B FE                         MOV     DI,SI           ;SAVE OFFSET TEMP IN DI
0464  BE FF98                       MOV     SI,SHIFTREG     ;SET SI = SHIFTREG
0467  BB 2580                       MOV     BX,4800*2       ;WAIT WITH QUIET BUS FOR
                                                            ;AT MOST 2 SECONDS WHEN
                                                            ;RUNNING AT 4800 BYTES PER
                                                            ;SECOND. (5.33 SEC AT 1800)
046A  4B                      FSYNC:  DEC     BX              ;DECRIMENT COUNTER
046B  74 10                         JZ      Q_TIM_OUT       ;QUIET TIME OUT IF ZERO
046D  E8 041E R                     CALL    FRAME_10        ;WAIT FOR FRAME 1 -> 0
0470  8B D6                         MOV     DX,SI           ;SET DX = SHIFT REG
0472  EC                            IN      AL,DX           ;READ DATA BYTE



0473  88 05                         MOV     [DI],AL         ;STORE DATA BYTE

0475  3C 55                         CMP     AL,055H         ;WAIT FOR SYNC
0477  74 F1                         JE      FSYNC           ;
0479  3C AA                         CMP     AL,0AAH
047B  74 ED                         JE      FSYNC
047D                          Q_TIM_OUT:
047D  47                            INC     DI
047E  EB 10                         JMP     SHORT SFIRST

0480                          TLOOP:                          ;WAIT FOR FRAME 1 -> 0
0480  8B D5                         MOV     DX,BP           ;SET DX = 8255 PORT C
0482  EC                      WAITX1: IN      AL,DX           ;READ CVSD FRAME
0483  22 C4                         AND     AL,AH           ;FRAME HIGH ?
0485  74 FB                         JZ      WAITX1          ;NO. WAIT FOR FRAME HI
0487  EC                      WAITX0: IN      AL,DX           ;READ CVSD FRAME
0488  22 C4                         AND     AL,AH           ;FRAME LOW ?
048A  75 FB                         JNZ     WAITX0          ;NO. WAIT FOR FRAME LO
048C  8B D6                         MOV     DX,SI           ;SET DX = SHIFT REG
048E  EC                            IN      AL,DX           ;READ DATA BYTE
048F  AA                            STOSB                   ;STORE DATA BYTE & INCR DI
0490  E2 EE                   SFIRST: LOOP    TLOOP           ;CONTINUE UNTIL CNT EXHAUSTED

0492  E8 041E R                     CALL    FRAME_10        ;WAIT FOR FRAME 1 -> 0
0495                          CVSDXA:
0495  B1 00                         MOV     CL,OK           ;SET OK RETURN CODE IN CL

                              ; NOTE: - BEFORE COMING TO CVSDX, WE MUST SET RETURN CODE IN CL
                              ;         BEFORE COMING TO CVSDXO, WE MUST SET RETURN CODE IN AL

0497  B0 0C                   CVSDX:  MOV     AL,DECODE       ;SET CVSD DECODE ON
0499  BA FB9B                       MOV     DX,CWREG        ;
049C  EE                            OUT     DX,AL           ;

049D  8A C1                         MOV     AL,CL           ;SET AL = RETURN CODE

                              ; ENABLE NMI AND 8259 INTERRUPTS

049F  5B                            POP     BX              ;RECOVER 8259 MASK
04A0  5E                            POP     SI              ;RECOVER TIMER VALUE
04A1  50                            PUSH    AX
04A2  8B C6                         MOV     AX,SI           ;TIMER VALUE INTO AX
04A4  E8 09A5 R                     CALL    NMION           ;ENABLE ALL INTERRUPTS
04A7  58                            POP     AX              ;AND RESTORE TIME OF DAY CLOCK

04A8                          CVSDXO:
04A8  07                            POP     ES              ;RESTORE ES
04A9  5D                            POP     BP              ;RESTORE BP
04AA  E9 0301 R                     JMP     EXIT            ;EXIT

                              ;*************************************************************
                              ;*                                                           *
```

```
                        ;*      NAME:  LPC DRIVER                                       *
                        ;*                                                              *
                        ;*   PURPOSE:  TO PROVIDE LOW-LEVEL BIOS SUPPORT FOR           *
                        ;*             LPC                                            *
                        ;*                                                              *
                        ;*   LINKAGE:  SOFTWARE INTERRUPT (INT 4DH WITH AH = 2 OR 3)   *
                        ;*             IF AH = 3, ENTRY WILL BE MADE AT LPC000. THIS   *
                        ;*             IS FOR LPC FOREGROUND.                         *
                        ;*             IF AH = 2, ENTRY WILL BE MADE AT LPC00. THIS IS *
                        ;*             FOR LPC BACKGROUND.                            *
                        ;*                                                            *
                        ;*    INPUTS:  AL - CONTAINS THE LPC FUNCTION                 *
                        ;*                  = 0  FOR LPC STATUS                       *
                        ;*                  = 1  FOR LPC SPEAK (INDEX)                *
                        ;*                  = 2  FOR LPC SPEAK (BUFFER)               *
                        ;*             BX - WORD NUMBER FROM INDEX  (FOR AL = 1)       *
                        ;*             CX - NUMBER OF BYTES IN ENCODED WORD            *
                        ;*          DS:SI - SEG:OFFSET OF SPEECH BFR  (FOR AL = 2)     *
                        ;*                                                            *
                        ;*   OUTPUTS:  AL CONTAINS A RETURN CODE                      *
                        ;*             00H - IF EVERYTHING O.K.                       *
                        ;*             01H - IF UNDEFINED COMMAND                     *
                        ;*             02H - IF LPC SPEAK IN PROGRESS                 *
                        ;*             03H - IF ACL ERROR (STUCK)                     *
                        ;*             04H - IF LPC INDEX OUT OF RANGE                *
                        ;*             06H - IF TIMEOUT WAITING FOR LPC READY         *
                        ;*                                                            *
                        ;*     EXIT:  INTERRUPT RETURN WITH RETURN CODE SET IN AL     *
                        ;*                                                            *
                        ;*  PROCESS: (1) - IF THIS IS A STATUS REQUEST, THEN CHECK    *
                        ;*                 STATUS OF LPC (IS IT CURRENTLY RUNNING?)   *
                        ;*                 EXIT WITH STATUS INFO IN AL                *
                        ;*           (2) - MASK ALL INTERRUPTS (NOT STATUS REQ.)       *
                        ;*                 READ LPC IN PROGRESS BIT. IF LPC RUNNING,  *
                        ;*                 REENABLE INTERRUPTS AND EXIT WITH STATUS.  *
                        ;*           (3) - SET LPC IN PROGRESS FLAG AND REENABLE      *
                        ;*                 INTERRUPTS.                                *
                        ;*           (4) - MAKE SURE  ACL IS ENABLED.  IF             *
                        ;*                 NOT, EXIT WITH RETURN CODE IN AL = 03H     *
                        ;*           (5) - CHECK TO SEE IF LPC INTR VECTOR HAS BEEN   *
                        ;*                 SET.  IF NOT, MOVE KBD VECTOR INTR TO      *
                        ;*                 04EH AND SET LPC INTR VECTOR. NMI AND      *
                        ;*                 OTHER HARDWARE INTERRUPTS ARE MASKED FOR   *
                        ;*                 THIS DURRATION. THIS IS NOT DONE IF THE    *
                        ;*                 REQUEST IS FOR LPC FOREGROUND.             *
                        ;*           (6) - DECODE LPC FUNCTION. IF INVALID FUNCTION,  *
                        ;*                 EXIT WITH RETURN CODE IN AL = 01H          *
                        ;*           (7) - IF SPEAK LPC INDEX FUNCTION, SET PROPER    *
                        ;*                 ROS PAGE.  IF INDEX OUT OF RANGE, EXIT     *
                        ;*                 WITH RETURN CODE IN AL = 04H               *
                        ;*           (8) - SET DS:SI TO POINT TO BUFFER               *
                        ;*           (9) - ISSUE SPEAK EXTERNAL COMMAND TO THE 5220   *
                        ;*          (10) - SET SYSTEM SPEAKER SWITCH (PORT 61H) BITS  *



                        ;*                 TO AUDIO CHANNEL                           *
                        ;*          (11) - SET CHANNEL MUX = LPC                      *
                        ;*                                                            *
                        ;*          (12) - ENABLE INTR 1 ON SYSTEM 8259 AND ENABLE    *
                        ;*                 LPC INTR ON CARD. THIS IS NOT DONE IF THE  *
                        ;*                 REQUEST IS FOR LPC FOREGROUND.             *
                        ;*          (13) - DISABLE NMI INTR (KBD) & OTHER INTRS       *
                        ;*          (14) - CALL LOAD BUFFER ROUTINE TO LOAD 16 BYTES  *
                        ;*                 OF DATA, SAVE THE COUNT AND POINTERS.      *
                        ;*          (15) - REENABLE INTERRUPTS.                       *
                        ;*          (16) - IF LPC FOREGROUND, REPEAT THE FOLLOWING    *
                        ;*                 STEPS UNTIL THE WORD IS COMPLETE           *
                        ;*                 -TEST BUFFER LOW UNTIL LOW TO HI TRANSIT   *
                        ;*                 -SEND 8 MORE BITS TO LPC BUFFER            *
                        ;*                 WHEN WORD IS DONE, WAIT TO RETURN UNTIL    *
                        ;*                 THE TALK STATUS BIT GOES HI TO LOW.        *
                        ;*                                                            *
                        ;*                 IF LPC BACKGROUND, RETURN TO USER WHILE    *
                        ;*                 INTERRUPT HANDLER UPDATES LPC BUFFER       *
                        ;*                                                            *
                        ;******************************************************************
                        ;*     NOTES: - REGISTERS PRESERVED DURING THIS CALL:             *
                        ;*              CS,SS,DS,ES,SI,DI,DX,CX,BX                        *
                        ;*              ALL OTHER REGISTERS DESTROYED.                    *
                        ;******************************************************************

                        ; -> SAVE AX & DS
04AD                    LPC000:
04AD  45                        INC     BP              ;BP=1 INDICATES FOREGROUND LPC
04AE  1E                LPC00:  PUSH    DS              ;SAVE DS
04AF  50                        PUSH    AX              ;SAVE AX

                        ; -> CHECK TO SEE TYPE OF INTERRUPT, IF STATUS, NO NMI MASK.

04B0  BA FB98                   MOV     DX,PORTA        ;8255 PORT A ADDRESS
04B3  0A C0                     OR      AL,AL           ;STATUS UPDATE?
04B5  75 0B                     JNZ     MASK_NMI        ;IF NOT, JUMP

                        ; -> HANDLE STATUS INQUIRY

04B7  EC                        IN      AL,DX           ;READ PORT A
04B8  A8 80                     TEST    AL,TALK_LPC     ;CHECK LPC IN PROGRESS BIT
04BA  58                        POP     AX              ;RESTORE STACK
04BB  75 17                     JNZ     LPC02A          ;REPORT LPC IN PROGRESS
04BD  B0 00                     MOV     AL,OK           ;REPORT ALL OK WITH LPC
04BF  E9 064A R                 JMP     LPCX            ;EXIT LPC BIOS

04C2                    ; -> MASK NMI AND HARDWARE INTERRUPTS
04C2                    MASK_NMI:
04C2  B0 10                     MOV     AL,10H
04C4  E6 A0                     OUT     NMI_PORT,AL     ;MASK NMI
04C6  FA                        CLI                     ;MASK HARDWARE INT'S

                        ; -> CHECK FOR LPC IN PROGRESS



04C7  EC                        IN      AL,DX           ;
04C8  A8 80                     TEST    AL,TALK_LPC     ;LPC IN PROGRESS ?

04CA  58                        POP     AX              ;RESTORE REQUEST
04CB  74 0C                     JZ      LPC03           ;IF LPC NOT IN PROG, SPEAK

04CD  FB                        STI                     ;ENABLE HARDWARE INT'S
```

```
04CE  E4 A0                             IN      AL,NMI_PORT
04D0  B0 80                             MOV     AL,80H
04D2  E6 A0                             OUT     NMI_PORT,AL     ;ENABLE NMI
04D4                            LPC02A:
04D4  B0 02                             MOV     AL,LPC_INPROG   ;SET AL = LPC IN PROGRESS (02H)
04D6  E9 064A R                         JMP     LPCX            ;GO TO EXIT

04D9                            LPC03:

04D9  50                                PUSH    AX              ;SAVE LPC FN REQUEST

                                ; -> SET LPC IN PROGRESS FLAG (8255 PORT A)

04DA  BA FB98                           MOV     DX,PORTA        ;SET LPC IN PROG FLAG
04DD  EC                                IN      AL,DX           ;
04DE  0C 80                             OR      AL,TALK_LPC     ;
04E0  EE                                OUT     DX,AL           ;

                                ; -> REENABLE NMI AND HARDWARE INTERRUPTS

04E1  FB                                STI                     ;ENABLE HARDWARE INT'S
04E2  E4 A0                             IN      AL,NMI_PORT
04E4  B0 80                             MOV     AL,80H
04E6  E6 A0                             OUT     NMI_PORT,AL     ;ENABLE NMI

                                ; -> MAKE SURE ACL IS ENABLED

04E8  E8 036E R                         CALL    CHKTKR_ACL      ;ACL ENABLED ?
04EB  73 0B                             JNC     LPC04           ;YES. CONTINUE
04ED  51                                PUSH    CX
04EE  E8 0357 R                         CALL    ATTACH_ACL      ;RESET ALL ACLS
04F1  59                                POP     CX
                                ;                               ;& ENABLE CARD ACL
04F2  73 04                             JNC     LPC04           ;IF NO ERRORS THEN CONTINUE
04F4  5B                                POP     BX              ;ADJUST STACK
04F5  E9 059A R                         JMP     LPC_ERR_EXIT    ;ACL ERROR EXIT

04F8                            LPC04:
04F8  0B ED                             OR      BP,BP           ;FORE OR BACKGROUND?
04FA  75 36                             JNZ     LPC10           ;IF FORE, DON'T TOUCH VECTORS

                                ; -> SET DS = 0

04FC  33 C0                             XOR     AX,AX           ;SET DS = 0
04FE  8E D8                             MOV     DS,AX



                                        ASSUME DS:DUMMY

                                ; -> CHECK TO SEE IF LPC INTR VECTOR HAS BEEN SET

0500  A1 0024 R                         MOV     AX,WORD PTR LPC_PTR     ;LOOK AT INT 9 VECTOR
0503  3D 06D2 R                         CMP     AX,OFFSET LPC_XX        ;POINTING AT LPC CODE?
0506  75 08                             JNE     LPC05                   ;IF NOT, SETUP INT 9
0508  8C C8                             MOV     AX,CS                   ;SEGMENT ADDR CORRECT?
050A  3B 06 0026 R                      CMP     AX,WORD PTR LPC_PTR+2   ;IF NOT, SETUP INT 9
050E  74 22                             JE      LPC10

                                ; -> KBD INTR VECTOR ->  04EH INTR
                                ; -> DISABLE INTERRUPTS (NMI & OTHERS)

0510                            LPC05:
0510  B0 10                             MOV     AL,10H          ;DISABLE NMI & HOLD REQUEST
0512  E6 A0                             OUT     NMI_PORT,AL     ;
0514  FA                                CLI                     ;DISABLE INTERRUPTS

0515  A1 0024 R                         MOV     AX,WORD PTR OLDKBD_PTR
0518  A3 0138 R                         MOV     WORD PTR KBD_PTR,AX     ;SAVE OLD KBD PTR
051B  A1 0026 R                         MOV     AX,WORD PTR OLDKBD_PTR+2
051E  A3 013A R                         MOV     WORD PTR KBD_PTR+2,AX   ;SETUP NEW INT 9 PTR

                                ;-> SET LPC INTR VECTOR

0521  C7 06 0024 R 06D2 R              MOV     WORD PTR LPC_PTR,OFFSET LPC_XX
0527  8C 0E 0026 R                      MOV     WORD PTR LPC_PTR+2,CS


                                ;-> ENABLE INTERRUPTS (NMI & OTHERS)

052B  FB                                STI                     ;ENABLE INTERRUPTS
052C  E4 A0                             IN      AL,NMI_PORT     ;RESET LATCH
052E  B0 80                             MOV     AL,80H          ;MASK TO ENABLE NMI
0530  E6 A0                             OUT     NMI_PORT,AL     ;ENABLE NMI
0532                            LPC10:

                                ; -> DECODE LPC FUNCTION

0532  58                                POP     AX              ;RESTORE AX  (LPC FUNCTION)
0533  3C 01                             CMP     AL,LPC_INDEX    ;SPEAK LPC INDEX FUNCTION ?
0535  74 09                             JE      LPC20           ;YES. GO TO SPEAK LPC INDEX CODE
0537  3C 02                             CMP     AL,LPC_BUFFER   ;SPEAK LPC BUFFER FUNCTION ?
0539  74 51                             JE      LPC25           ;YES. GO TO SPEAK LPC BFR CODE
053B  B0 01                             MOV     AL,BAD_CMD      ;SET ERROR CODE IN AL
053D  EB 5B 90                          JMP     LPC_ERR_EXIT    ;EXIT LPC CODE

                                ; SET PROPER ROS PAGE  (SPEAK LPC INDEX FUNCTION)

0540  0B DB                     LPC20:  OR      BX,BX           ;INDEX = 0 ? (INVALID)
0542  74 28                             JE      LPC22           ;YES. EXIT



0544  80 FF 00                          CMP     BH,0            ;BX < 256?
0547  75 23                             JNZ     LPC22           ;IF NOT, INDEX ERROR

0549  B1 00                             MOV     CL,PAGE0        ;SET CL = ROS PAGE 0
054B  81 FB 0029                        CMP     BX,PG0_MAX      ;IS WORD IN PAGE 0 ?
054F  72 20                             JB      LPC23           ;YES. GO TO SET ROS PAGE
0551  43                                INC     BX              ;INCREMENT BX TO ADJUST FOR
                                                                ;END_PAGE ENTRIES IN TABLE

0552  B1 04                             MOV     CL,PAGE1        ;SET CL = ROS PAGE 1
0554  81 FB 005D                        CMP     BX,PG1_MAX      ;IS WORD IN PAGE 1 ?
0558  72 17                             JB      LPC23           ;YES. GO TO SET ROS PAGE
055A  43                                INC     BX              ;INCREMENT BX TO ADJUST FOR
                                                                ;END_PAGE ENTRIES IN TABLE

055B  B1 08                             MOV     CL,PAGE2        ;SET CL = ROS PAGE 2
055D  81 FB 0091                        CMP     BX,PG2_MAX      ;IS WORD IN PAGE 2 ?
0561  72 0E                             JB      LPC23           ;YES. GO TO SET ROS PAGE
0563  43                                INC     BX              ;INCREMENT BX TO ADJUST FOR
```

```
                                                        ;END_PAGE ENTRIES IN TABLE

0564  B1 0C                          MOV     CL,PAGE3        ;SET CL = ROS PAGE 3
0566  81 FB 00C8                     CMP     BX,PG3_MAX      ;IS WORD IN PAGE 3 ?
056A  72 05                          JB      LPC23           ;YES. GO TO SET ROS PAGE

056C  B0 04                    LPC22:MOV     AL,INDEX_ERR    ;SET AL = INDEX ERROR
056E  EB 2A 90                       JMP     LPC_ERR_EXIT    ;GO TO EXIT CODE

0571  BA FB98                  LPC23:MOV     DX,PORTA        ;SET PROPER ROS PAGE
0574  EC                            IN      AL,DX           ;
0575  24 F3                          AND     AL,CLR_PAGE     ;
0577  0A C1                          OR      AL,CL           ;
0579  EE                            OUT     DX,AL           ;

                               ; SET DS:SI TO POINT TO BUFFER  (SPEAK LPC INDEX FUNCTION)

057A  0E                             PUSH    CS              ;SET DS:SI -> SPEECH BFR
057B  1F                             POP     DS              ;
057C  D1 E3                          SHL     BX,1            ;
057E  8B B7 0AFE R                   MOV     SI,[BX+OFFSET TABIDX-2]
0582  8B CE                          MOV     CX,SI
0584  F7 D9                          NEG     CX              ;WE ARE GETTING THE WORD'S BYTE
0586  03 8F 0B00 R                   ADD     CX,[BX+OFFSET TABIDX] ;COUNT FROM THE DIFFERENCE
                                                             ;BETWEEN TABLE OFFSETS
058A  EB 02                          JMP     SHORT LPC30     ;CONTINUE

                               ; SET DS:SI TO POINT TO BUFFER   (SPEAK LPC BUFFER FUNCTION)

058C  1F                       LPC25:POP     DS              ;SET DS:SI -> SPEECH BFR
058D  1E                             PUSH    DS              ;
                                                             ;CX = LENGTH OF WORD
                               ; SET 5220 SPEAK EXTERNAL COMMAND



058E  B0 60                    LPC30:MOV     AL,SPK_EXT      ;SET SPEAK EXTERNAL MODE
0590  E8 065D R                      CALL    LPCW_10         ;WRITE COMMAND TO 8255
                                                             ;ERROR WAITING FOR LPC READY ?
0593  74 4B                          JZ      LPC35           ;NO. CONTINUE
0595                           LPC33:
0595  E8 037A R                      CALL    WAIT_FOR_LPC    ;BE SURE LPC SPEECH IS COMPLETE
0598  B0 06                       MOV     AL,LPCRDY_ERR   ;YES. SET ERROR CODE IN AL

                               ; THIS IS THE GENERAL EXIT PATH FOR LPC BIOS

059A                           LPC_ERR_EXIT:
059A  50                             PUSH    AX

059B  E4 21                          IN      AL,PORT_21H
059D  0C 02                          OR      AL,INT1_OFF
059F  E6 21                          OUT     PORT_21H,AL
05A1  BA FF9F                        MOV     DX,TKR_ACL      ;DISSABLE INTERRUPTS BOTH
05A4  EC                             IN      AL,DX           ;ON SYSTEM AND FEATURE CARD
05A5  24 FD                          AND     AL,11111101B
05A7  0C 01                          OR      AL,00000001B
05A9  EE                             OUT     DX,AL

05AA  B0 10                          MOV     AL,10H
05AC  E6 A0                          OUT     NMI_PORT,AL     ;MASK NMI
05AE  FA                             CLI                     ;MASK HARDWARE INT'S

                               ; -> CHECK TO SEE IF LPC INTR VECTOR HAS BEEN SET

05AF  33 C0                            XOR     AX,AX           ;RESTORE KEYBOARD INT VECTOR
05B1  8E D8                            MOV     DS,AX           ;ONLY IF NEEDED

05B3  A1 0024 R                        MOV     AX,WORD PTR LPC_PTR
05B6  3D 06D2 R                        CMP     AX,OFFSET LPC_XX ; DOES INT 9 POINT AT LPC?
05B9  74 08                            JE      LPC34           ;IF SO, RESTORE WITH SAVED
05BB  8C C8                            MOV     AX,CS           ;VECTOR
05BD  3B 06 0026 R                     CMP     AX,WORD PTR LPC_PTR+2
05C1  75 0C                            JNE     LPC34A
05C3                           LPC34:
05C3  A1 0138 R                        MOV     AX,KBD_PTR
05C6  A3 0024 R                        MOV     WORD PTR OLDKBD_PTR,AX
05C9  A1 013A R                        MOV     AX,KBD_PTR+2    ;RESTORE INT 9 WITH SAVED
05CC  A3 0026 R                        MOV     WORD PTR OLDKBD_PTR+2,AX        ;VECTOR
05CF                           LPC34A:
05CF  BA FB98                          MOV     DX,PORTA
05D2  EC                               IN      AL,DX
05D3  24 7F                            AND     AL,0FFH-TALK_LPC
05D5  EE                               OUT     DX,AL           ;TURN OFF LPC IN PROG FLAG

05D6  FB                               STI                     ;ENABLE HARDWARE INT'S
05D7  E4 A0                            IN      AL,NMI_PORT
05D9  B0 80                            MOV     AL,80H
05DB  E6 A0                            OUT     NMI_PORT,AL     ;ENABLE NMI



05DD  58                               POP     AX
05DE  EB 6A                            JMP     SHORT LPCX      ;GO TO EXIT

                               ; SET SYSTEM SPEAKER SWITCH TO AUDIO CHANNEL

05E0  E4 61                    LPC35:  IN      AL,PORT_61H     ;READ SYSTEM'S PORT 61H
05E2  24 9F                            AND     AL,CLR_SPKSW    ;CLEAR SPEAKER SWITCH BITS
05E4  0C 40                            OR      AL,AUDIO_CHN    ;OR IN 'AUDIO CHANNEL' BITS
05E6  E6 61                            OUT     PORT_61H,AL     ;OUTPUT TO PORT 61H

                               ; SETUP CHANNEL MUX FOR LPC SPEECH

05E8  BA FB98                          MOV     DX,PORTA        ;ADDRESS PORT A
05EB  EC                               IN      AL,DX           ;READ IT
05EC  24 FC                            AND     AL,CLR_MUX      ;SET CHANNEL MUX BITS TO 00
05EE  EE                               OUT     DX,AL           ;OUTPUT TO PORT

05EF  0B ED                            OR      BP,BP           ;FORE OR BACKGROUND
05F1  75 0D                            JNZ     LPC40           ;IF FORE, DON'T ENABLE INT'S

                               ; ENABLE INTR 1 ON SYSTEM 8259

05F3  E4 21                            IN      AL,PORT_21H     ;ENABLE INTR 1
05F5  24 FD                            AND     AL,INT1_ON      ;
05F7  E6 21                            OUT     PORT_21H,AL     ;

                               ; ENABLE LPC INTR W/O DISABLING THE CHANNEL

05F9  BA FF9F                          MOV     DX,TKR_ACL      ;ENABLE LPC INTR
05FC  EC                               IN      AL,DX           ;
05FD  0C 03                            OR      AL,3            ;
```

```
05FF  EE                        OUT     DX,AL           ;

0600                        LPC40:
0600  BB 0010                MOV     BX,16           ;LOAD BUFFER WITH 16 DATA BYTES
0603                        LPC45:

                            ; LOAD LPC BUFFER WITH OF DATA

0603  B0 10                   MOV     AL,10H
0605  E6 A0                   OUT     NMI_PORT,AL     ;MASK NMI INTERRUPT
0607  FA                      CLI                     ;MASK HARDWARE INTERRUPTS

0608  E8 0675 R               CALL    LOAD_BFR_HNDLR ;SEND BYTES TO LPC, SAVE PTR
                                                      ;AND COUNT INFO

060B  FB                      STI                     ;ENABLE OTHER INTERRUPTS
060C  E4 A0                   IN      AL,NMI_PORT     ;RESET NMI LATCH
060E  B0 80                   MOV     AL,80H
0610  E6 A0                   OUT     NMI_PORT,AL     ;ENABLE NMI

                                                      ;ERROR WAITING FOR LPC READY ?



0612  72 81                   JC      LPC33           ;YES. GO TO SET ERROR & EXIT

0614  0B ED                   OR      BP,BP           ;FORE OR BACKGROUND
0616  74 30                   JZ      LPC_BACKGROUND ;EXIT, LET BACKGROUND TAKE OVER

                            ; -> FOREGROUND LPC IS PROCESSED HERE

0618  0B C9                   OR      CX,CX           ;ARE ALL BYTES SENT TO LPC?
061A  74 18                   JZ      FOREGROUND_COMPLETE ;IF SO, GO ON

061C  51                      PUSH    CX
061D  B9 2000                 MOV     CX,2000H
0620                        TEST_HALF_BUF_BIT:
0620  E8 075C R               CALL    READ_PORTB      ;READ LPC STATUS
0623  F6 C4 40                TEST    AH,01000000B    ;LOOK AT BUF HALF FULL BIT
0626  E1 F8                   LOOPZ   TEST_HALF_BUF_BIT ;GO ON WHEN BUF HALF FULL
0628  E3 06                   JCXZ    FGND_ERR
062A  59                      POP     CX

062B  BB 0008                 MOV     BX,8            ;SEND 8 BYTES TO LPC
062E  EB D3                   JMP     LPC45           ;LOOP BACK FOR ANOTHER ROUND

0630                        FGND_ERR:
0630  59                      POP     CX
0631                        LPC33_LINK:
0631  E9 0595 R               JMP     LPC33

0634                        FOREGROUND_COMPLETE:
0634  BA FB98                 MOV     DX,PORTA
0637  EC                      IN      AL,DX
0638  24 7F                   AND     AL,0FFH-TALK_LPC
063A  EE                      OUT     DX,AL           ;TURN OFF LPC IN PROGRESS FLAG
063B  B9 5000                 MOV     CX,5000H
063E                        FOR_COMP:
063E  E8 075C R               CALL    READ_PORTB      ;
0641  F6 C4 80                TEST    AH,10000000B    ;READ LPC SPEAKING BIT
0644  E0 F8                   LOOPNZ  FOR_COMP        ;LOOP BACK UNTIL LPC
                                                      ;HAS PROCESSED ALL DATA
0646  E3 E9                   JCXZ    LPC33_LINK

                            ; -> EXIT LPC CODE

0648                        LPC_BACKGROUND:

0648  B0 00                   MOV     AL,OK           ;SET LPC STATUS TO O.K.
064A  1F                    LPCX:   POP     DS              ;RESTORE ORIGINAL DS
064B  E9 0301 R               JMP     EXIT            ;

                            ; WAIT_RDY: - THIS PROCEDURE WAITS FOR LPC READY (LOW ACTIVE)
                            ;           - IT DESTROYS REGISTERS AL & DX
                            ;           - THIS PROCEDURE MUST BE FOLLOWED BY A CHECK OF
                            ;             THE ZERO FLAG:
                            ;             IF ON, => NO ERRORS



                            ;             IF OFF => ERROR WAITING FOR LPC READY
                            ;                       (SET AL = LPCRDY_ERR & EXIT)

064E                        WAIT_RDY PROC   NEAR

064E  51                      PUSH    CX              ;SAVE CX
064F  33 C9                   XOR     CX,CX           ;CLEAR CX
0651  51                    WAIT00:   PUSH    CX              ;DELAY
0652  59                      POP     CX              ;DELAY
0653  BA FB9A                 MOV     DX,PORTC        ;READ READY
0656  EC                      IN      AL,DX           ;
0657  24 01                   AND     AL,LPC_READY    ;TURN OFF ALL BITS EXCEPT READY
                                                      ;READY ?
0659  E0 F6                   LOOPNZ  WAIT00          ;NO. KEEP CHECKING
065B  59                      POP     CX              ;RESTORE CX
065C  C3                      RET                     ;RETURN

065D                        WAIT_RDY ENDP

                            ; LPCW_10  THIS PROCEDURE WRITES TO PORT B THE VALUE
                            ;          CONTAINED IN AL BY TURNING LPC WRITE LINE
                            ;          ON & THEN OFF
                            ;        - AL SHOULD CONTAIN VALUE TO BE WRITTEN TO
                            ;          PORT B.
                            ;        - IT DESTROYS REGISTERS AL & DX
                            ;        - THIS PROCEDURE MUST BE FOLLOWED BY A CHECK OF
                            ;          THE ZERO FLAG:
                            ;          IF ON, => NO ERRORS
                            ;          IF OFF => ERROR WAITING FOR LPC READY
                            ;                    (SET AL = LPCRDY_ERR & EXIT)

065D                        LPCW_10  PROC   NEAR

065D  BA FB99                 MOV     DX,PORTB        ;LPC WRITE (ON -> OFF)
0660  EE                      OUT     DX,AL           ;
0661  B0 0B                   MOV     AL,LPCW_ON      ;
0663  BA FB9B                 MOV     DX,CWREG        ;
0666  EE                      OUT     DX,AL           ;
0667  E8 064E R               CALL    WAIT_RDY
066A  75 08                   JNZ     LPCW_X          ;(TIMEOUT WAITING FOR LPC RDY)
066C  B0 0A                   MOV     AL,LPCW_OFF     ;
066E  BA FB98                 MOV     DX,CWREG        ;
0671  EE                      OUT     DX,AL           ;
```

```
0672  32 C0                     XOR     AL,AL           ;SET THE ZERO FLAG
0674  C3                  LPCW_X: RET                     ;
0675                      LPCW_10 ENDP

                          ;*************************************************************
                          ; LOAD_BFR_HANDLER
                          ; DESCRIPTION



                          ;       THE REMAINING NUMBER OF BYTES IN THE LPC WORD TO BE OUTPUT
                          ;       IS COMPARED TO THE MAXIMUM NUMBER OF OUTPUT BYTES ALLOWED.
                          ;       IF THERE ARE MORE BYTES LEFT TO BE OUTPUT THAN CAN BE SENT
                          ;       OUT WITH THIS CALL THEN
                          ;          THE MAXIMUM ALLOWED NUMBER OF BYTES IS SENT OUT TO THE
                          ;          LPC CHIP, AND
                          ;          THE POINTER TO THE NEXT BYTE TO OUTPUT (SEGMENT AND
                          ;          OFFSET) AND
                          ;          THE REMAINING COUNT ARE FOLDED INTO TWO WORDS AND SAVED,
                          ;          AND
                          ;          THE CARRY FLAG IS RESET TO INDICATE NO ERROR.
                          ;       IF ALL REMAINING BYTES TO BE OUTPUT CAN BE HANDLED THIS
                          ;       TIME THEN
                          ;          THEY ARE SENT,
                          ;          A BYTE OF 00 IS SENT, AND
                          ;          THE CARRY FLAG IS SET TO INDICATE END OF SPEECH DATA
                          ;          OUTPUT.
                          ;
                          ; ON ENTRY
                          ;       BX = MAXIMUM ALLOWED NUMBER OF BYTES TO BE OUTPUT.
                          ;       CX = REMAINING NUMBER OF BYTES TO BE OUTPUT IN THE LPC
                          ;       WORD.
                          ;       DS:SI = POINTER TO LPC DATA
                          ;       DFL = 0 (DIRECTION FLAG RESET TO INCREMENT)
                          ;
                          ; ON EXIT
                          ;       INTERRUPT VECTOR LOCATION 4FH = THE COMPRESSED VERSION OF
                          ;       THE POINTER AND COUNT INFORMATION DESCRIBING THE REMAINING
                          ;       DATA TO BE OUTPUT FOR THE LPC WORD BEING PROCESSED.
                          ;          0:13C = LMMM WHERE L IS A HEX DIGIT REPRESENTING AN
                          ;             OFFSET, MMM IS A 3 HEX DIGIT COUNT OF REMAINING BYTES
                          ;          0:13E = PPPP WHERE PPPP IS THE SEGMENT ADDRESS OF THE
                          ;             NEXT LPC DATA TO BE OUTPUT.
                          ;             PPPP:L POINTS AT THE NEXT LPC DATA TO BE OUTPUT
                          ;             MMM IS THE REMAINING NUMBER OF BYTES TO BE OUTPUT
                          ;
                          ;       REGISTERS AX, BX,CX, DX, AND SI ARE ALTERED.
                          ;
                          ;*************************************************************

0675                      LOAD_BFR_HNDLR  PROC    NEAR
0675  3B CB                      CMP     CX,BX           ;IS REMAINING COUNT LESS THAN MAX?
0677  7C 16                      JL      COMPLETE_OUTPUT ;IF SO, FINISH WORD.
0679  7F 03                      JG      CONTINUE_OUTPUT ;IF MORE THAN THE LIMIT,SEND LIMIT
067B  80 EB 04                   SUB     BL,04           ;IF EXACTLY THE LIMIT REMAINS,
                                                          ;OUTPUT 4 LESS THAN LIMIT TO AVOID
067E                                                      ;OVER RUNNING THE BUFFER.
                          CONTINUE_OUTPUT:
067E  51                          PUSH    CX
067F  8B CB                       MOV     CX,BX           ;LOAD MAXIMUM BYTE COUNT

0681  E8 069F R                   CALL    LOAD_BFR        ;LOAD BYTES INTO 5220
0684  59                          POP     CX
0685  75 16                       JNZ     LOAD_BFR_ERR    ;JUMP IF ERROR ENCOUNTERED



0687  2B CB                       SUB     CX,BX           ;ADJUST COUNT FOR BYTES OUTPUT
0689                      UPDATE_COMPLETE:                ;THIS TIME.
0689  E8 06AA R                   CALL    SAVE_POINTER

068C  F8                          CLC                     ;CLEAR CARRY INDICATES NO ERRORS
                                                          ;SPEAKING
068D  EB 0F                       JMP     SHORT EXIT_BFR_HNDLR

068F                      COMPLETE_OUTPUT:
068F  E8 069F R                   CALL    LOAD_BFR        ;LOAD LAST BYTES
0692  75 09                       JNZ     LOAD_BFR_ERR    ;BRANCH IF ERROR
0694  B0 00                       MOV     AL,0
0696  E8 065D R                   CALL    LPCW_10         ;SEND BYTE OF 0 TO 5220
0699  33 C9                       XOR     CX,CX           ;REMAINING COUNT = 0
069B  EB EC                       JMP     UPDATE_COMPLETE ;SAVE POINTER, CLEAR CARRY, RETURN
069D                      LOAD_BFR_ERR:
069D  F9                          STC                     ;SET CARRY TO INDICATE SPEECH END
069E                      EXIT_BFR_HNDLR:
069E  C3                          RET
069F                      LOAD_BFR_HNDLR  ENDP


                          ; NOTES: - PRIOR TO CALLING THE LOAD_BFR PROCEDURE, WE MUST HAVE:
                          ;          CX = # OF BYTES TO LOAD
                          ;          DS:SI = SEGMENT:OFFSET OF DATA
                          ;        - THIS PROCEDURE DESTROYS REGISTERS AL & DX
                          ;        - THIS PROCEDURE MUST BE FOLLOWED BY A CHECK OF
                          ;          THE ZERO FLAG:
                          ;          IF ON, => NO ERRORS
                          ;          IF OFF => ERROR WAITING FOR LPC READY
                          ;                    (SET AL = LPCRDY_ERR & EXIT)

069F                      LOAD_BFR PROC   NEAR

069F  AC                  LOAD00:  LODSB                   ;LOAD BFR WITH CX BYTES OF DATA
06A0  E8 065D R                    CALL    LPCW_10         ;
06A3  75 04                        JNZ     LOADXX          ;(TIMEOUT WAITING FOR LPC RDY)
06A5  E2 F8                        LOOP    LOAD00          ;
06A7  32 C0                        XOR     AL,AL           ;SET THE ZERO FLAG
06A9  C3                  LOADXX:  RET                     ;

06AA                      LOAD_BFR ENDP

                          ;*************************************************************
                          ; SAVE_POINTER
                          ; THIS PROC FOLDS THE OFFSET AND COUNT FOR LPC INTO TWO WORDS.
                          ; SEE DOCUMENTATION ON LOAD_BFR_HNDLR FOR MORE INFO.
                          ;*************************************************************

06AA                      SAVE_POINTER    PROC    NEAR
06AA  0B ED                       OR      BP,BP           ;ARE WE IN FOREGROUND?
06AC  75 23                       JNZ     NO_POINTER_SAVE ;IF SO, DON'T DO THIS SAVE
```

```
06AE  06                        PUSH    ES
06AF  33 C0                     XOR     AX,AX
06B1  8E C0                     MOV     ES,AX
                                ASSUME  ES:DUMMY        ;ADDRESS PTR AND COUNT SAVE LOCS

06B3  8B C1                     MOV     AX,CX           ;SAVE COUNT IN LOW 3 NIBS OF AX
06B5  B1 04                     MOV     CL,4
06B7  D3 CE                     ROR     SI,CL           ;ROTATE NIBS OF SI
                                                        ;(4,3,2,1 --> 1,4,3,2)
06B9  56                        PUSH    SI              ;SAVE THIS NEW VERSION

06BA  81 E6 F000                AND     SI,0F000H       ;MASK LOW 3 NIBS(MOST SIGNIFICANT)
06BE  0B C6                     OR      AX,SI           ;OR IN WITH COUNT
06C0  26: A3 013C R             MOV     ES:WORD PTR BFR_PTR,AX  ;STORE PACKED WORD FOR USE
                                                        ;NEXT TIME
06C4  8C D8                     MOV     AX,DS
06C6  5B                        POP     BX              ;RESTORE ALTERED VERSION OF SI
06C7  80 E7 0F                  AND     BH,0FH          ;MASK LEFTMOST
                                                        ;(LOWEST SIGNIFICANCE)
06CA  03 C3                     ADD     AX,BX           ;ADD INTO SEGMENT
06CC  26: A3 013E R             MOV     ES:WORD PTR BFR_PTR+2,AX ;STORE NEW VERSION OF
                                                        ;SEGMENT ADDR

06D0  07                        POP     ES              ;RESTORE ES
06D1                    NO_POINTER_SAVE:
06D1  C3                        RET
06D2                    SAVE_POINTER    ENDP
                        ;*****************************************************************
                        ;*                                                               *
                        ;*       NAME:  LPC INTERRUPT HANDLER                            *
                        ;*                                                               *
                        ;*    PURPOSE:  TO PROVIDE SUPPORT FOR LPC HARDWARE INTR         *
                        ;*                                                               *
                        ;*    LINKAGE:  HARDWARE INTR 1 (8259)                           *
                        ;*                                                               *
                        ;*     INPUTS:  VECTOR AREA FOR SOFTWARE INTR 04FH MUST          *
                        ;*              CONTAIN THE LPC BUFFER POINTER                   *
                        ;*                                                               *
                        ;*    OUTPUTS:  NONE                                             *
                        ;*                                                               *
                        ;*       EXIT:  INTERRUPT RETURN                                 *
                        ;*                                                               *
                        ;*    PROCESS:  (1) - CHECK TO SEE IF LPC OR KBD INTERRUPT.      *
                        ;*                    IF KBD, ISSUE KBD INTERRUPT & RETURN       *
                        ;*                    IF LPC, CONTINUE                           *
                        ;*              (2) - MASK NMI AND OTHER HARDWARE INTERRUPTS.    *
                        ;*              (3) - CHECK TYPE OF LPC INTERRUPT.  IF STRAY,    *
                        ;*                    GO ON.  IF BUFFER LOW, CALL ROUTINE TO     *
                        ;*                    SEND 8 MORE BYTES TO LPC CHIP.  IF OTHER.  *
                        ;*                    TURN OFF LPC AND LPC_IN_PROGRESS FLAG.     *
                        ;*              (4) - ISSUE EOI CMD TO INT 1 OF THE 8259         *
                        ;*              (5) - REENABLE NMI AND OTHERS                    *
                        ;*              (6) - ISSUE IRET                                 *



                        ;*                                                               *
                        ;*****************************************************************

                        ; -> SAVE SOME REGISTERS

06D2  50                LPC_XX:  PUSH    AX             ;SAVE AX & DX
06D3  52                         PUSH    DX             ;

                        ; -> CHECK TO SEE IF LPC OR KBD INTERRUPT
                        ;     IF LPC, GO TO LPC INTR HANDLER CODE
                        ;     IF KBD, ISSUE KBD INTERRUPT & RETURN

06D4  B0 0B                      MOV     AL,0BH
06D6  E6 20                      OUT     PORT_20H,AL    ;SYSTEM 8259 CONTROL PORT
06D8  EB 00                      JMP     $+2
06DA  E4 20                      IN      AL,PORT_20H
06DC  A8 02                      TEST    AL,02          ;LPC INTR ?
06DE  75 05                      JNZ     LPCX00         ;YES, CONTINUE

06E0  5A                         POP     DX             ;RESTORE AX & DX
06E1  58                         POP     AX             ;
06E2  CD 4E                      INT     KBD            ;ISSUE KBD INTR
06E4  CF                         IRET                   ;RETURN

                        ;*****************************************************************
                        ;*****************   LPC INTERRUPT HANDLER   *********************
                        ;*****************************************************************

                        ; -> SAVE OTHER REGISTERS

06E5  B0 10             LPCX00: MOV      AL,10H
06E7  E6 A0                     OUT      NMI_PORT,AL    ;DISSABLE NMI
06E9  FA                        CLI                     ;DISSABLE HARDWARE INTERRUPTS

06EA  E9 0A95 R                 JMP      SETUP_FLAG2
06ED                    FLAG_SETUP2:
06ED  56                        PUSH     SI             ;SAVE REGISTERS
06EE  57                        PUSH     DI             ;
06EF  55                        PUSH     BP

06F0  33 ED                     XOR      BP,BP          ;BP=0 MEANS LPC BACKGROUND

                        ; -> READ 8255 PORT B (5220 STATUS REG) & SAVE IN AH

06F2  E8 075C R                 CALL     READ_PORTB

                        ; -> DECODE TYPE OF LPC INTERRUPT

06F5  80 E4 E0                  AND      AH,0E0H        ;ZERO OUT BITS OF NO INTEREST
06F8  80 FC 80                  CMP      AH,10000000B   ;STRAY INTERRUPT?
06FB  74 4B                     JZ       LPCX95         ;IF SO END INTERRUPT AND RETURN
06FD  80 FC C0                  CMP      AH,11000000B   ;BUFFER LOW INTERRUPT?
0700  75 1D                     JNZ      LPCX90         ;IF NOT, EITHER LPC WORD IS DONE



                                                        ;OR SOMETHING'S WRONG.IN ANY CASE,
                                                        ;DISCONTINUE LPC PROCESSING.

                        ; -> BUFFER LOW INTR  => LOAD 8 MORE BYTES

0702  33 C0             LPCX30:  XOR     AX,AX          ;SET DS:SI TO POINT TO LPC DATA
0704  8E D8                      MOV     DS,AX          ;
0706  C5 36 013C R               LDS     SI,DWORD PTR[BFR_PTR] ;GET PTR AND COUNT INTO
                                                        ;DS:SI
070A  56                         PUSH    SI             ;SAVE SI
070B  B1 0C                      MOV     CL.12
```

```
070D  D3 EE                     SHR     SI,CL           ;EXTRACT OFFSET VALUE FROM SI
070F  59                        POP     CX
0710  80 E5 0F                  AND     CH,0FH          ;EXTRACT COUNT VALUE FROM SI
0713  0B C9                     OR      CX,CX
0715  74 31                     JZ      LPCX95          ;IF COUNT IS 0, DO NOT UPDATE
0717  BB 0008                   MOV     BX,8            ;MAXIMUM OUTPUT OF 8 BYTES
071A  E8 0675 R                 CALL    LOAD_BFR_HNDLR  ;PROCESS UPDATE TO 5220
071D  73 29                     JNC     LPCX95          ;IF NO ERROR, JUMP
                                                        ;IF ERROR OR END OF SPEECH, GO ON
                        ; -> TURN OFF 'LPC IN PROGRESS' FLAG

071F                    LPCX90:
071F  E8 037A R                 CALL    WAIT_FOR_LPC    ;WAIT FOR LPC TO COMPLETE
0722  BA FB98                   MOV     DX,PORTA        ;TURN OFF LPC IN PROG FLAG
0725  EC                        IN      AL,DX
0726  24 7F                     AND     AL,0FFH-TALK_LPC
0728  EE                        OUT     DX,AL           ;TURN OFF LPC IN PROGRESS FLAG

0729  E4 21                     IN      AL,PORT_21H
072B  0C 02                     OR      AL,INT1_OFF
072D  E6 21                     OUT     PORT_21H,AL
072F  BA FF9F                   MOV     DX,TKR_ACL      ;DISSABLE INTERRUPTS BOTH
0732  EC                        IN      AL,DX           ;ON SYSTEM AND FEATURE CARD
0733  24 FD                     AND     AL,11111101B
0735  0C 01                     OR      AL,00000001B
0737  EE                        OUT     DX,AL

                                ASSUME  DS:DUMMY
0738  33 C0                     XOR     AX,AX           ;RESTORE KEYBOARD INTERRUPT VECTOR
073A  8E D8                     MOV     DS,AX
073C  A1 0138 R                 MOV     AX,KBD_PTR
073F  A3 0024 R                 MOV     WORD PTR OLDKBD_PTR,AX
0742  A1 013A R                 MOV     AX,KBD_PTR+2
0745  A3 0026 R                 MOV     WORD PTR OLDKBD_PTR+2,AX

                        ; -> ISSUE INTR 1 EOI (END OF INTR) CMD

0748  B0 61             LPCX95: MOV     AL,INT1_EOI     ;INT 1 EOI CMD
074A  E6 20                     OUT     PORT_20H,AL     ;

074C  E4 A0                     IN      AL,NMI_PORT     ;RESET NMI LATCH
074E  B0 80                     MOV     AL,80H


0750  E6 A0                     OUT     NMI_PORT,AL     ;ENABLE NMI
0752  FB                        STI                     ;ENABLE OTHER HARDWARE INTERRUPTS

                        ; -> RESTORE REGISTERS & RETURN

0753  5D                        POP     BP              ;RESTORE REGISTERS
0754  5F                        POP     DI              ;
0755  5E                        POP     SI              ;
0756  1F                        POP     DS              ;
0757  59                        POP     CX              ;
0758  5B                        POP     BX              ;
0759  5A                        POP     DX              ;
075A  58                        POP     AX              ;
075B  CF                        IRET                    ;RETURN

                        ;*************************************************************
                        ; READ_PORTB
                        ;       THIS PROC READS PORT B AND SAVES IT IN AH
                        ; DESCRIPTION
                        ;       FIRST PORT A IS SAVED (WHEN 8255 MODE SET OCCURRS ALL
                        ;       OUTPUTS GO HIGH)
                        ;       THE 8255 MODE IS SET TO LPC IN MODE (PORT B INPUT)
                        ;       THE LPC READ BIT IS SET
                        ;       WE WAIT FOR READY TO GO HIGH
                        ;       WE SAVE PORT B IN AH
                        ;       RESET LPC READ BIT
                        ;       RETURN 8255 TO PORT B OUTPUT MODE
                        ;       RESTORE PORT A AND RETURN
                        ; ON EXIT:
                        ;       DX,AX,BH ARE DESTROYED
                        ;       AH HOLDS VALUE OF LPC STATUS REGISTER
                        ;*************************************************************
075C                    READ_PORTB      PROC    NEAR
075C  BA FB98                   MOV     DX,PORTA        ;READ 8255 PORT A
075F  EC                        IN      AL,DX           ;
0760  8A F8                     MOV     BH,AL           ;SAVE IN BH

0762  B0 83                     MOV     AL,LPC_IN       ;SET PORT B AS INPUT
0764  BA FB9B                   MOV     DX,CWREG        ;
0767  52                        PUSH    DX              ;SAVE CWREG ADDRESS
0768  EE                        OUT     DX,AL           ;
0769  B0 09                     MOV     AL,LPCR_ON      ;READ PORT B
076B  EE                        OUT     DX,AL           ;
076C  E8 064E R                 CALL    WAIT_RDY        ;
076F  BA FB99                   MOV     DX,PORTB        ;
0772  EC                        IN      AL,DX           ;
0773  8A E0                     MOV     AH,AL           ;SAVE VALUE IN AH
0775  B0 08                     MOV     AL,LPCR_OFF     ;
0777  5A                        POP     DX              ;RESTORE CWREG
0778  EE                        OUT     DX,AL           ;

0779  B0 81                     MOV     AL,LPC_OUT      ;SET PORT B AS OUTPUT
077B  EE                        OUT     DX,AL           ;


077C  8A C7                     MOV     AL,BH           ;RE-WRITE 8255 PORT A
077E  BA FB98                   MOV     DX,PORTA        ;
0781  EE                        OUT     DX,AL           ;
0782  C3                       RET
0783                    READ_PORTB      ENDP

                        ;*************************************************************
                        ;*                                                           *
                        ;*                      DIAGNOSTIC CODE                      *
                        ;*                                                           *
                        ;* THIS CODE IS ACCESSED VIA INTERRUPT 92H                   *
                        ;*************************************************************
                        ; ERRORS:                                                    *
                        ;       CUSTOMER LEVEL  |  SERVICE LEVEL  | SYMPTOM        *
                        ;       ----------------------------------------------------*
                        ;               B       |       10XX      | RESET FAILURE  *
                        ;               C       |       11XX      | LPC ERROR      *
                        ;               D       |       12XX      | CVSD PLAYBACK ERROR *
                        ;               E       |       13XX      | CVSD RECORD ERROR *
                        ;               D       |       14XX      | CVSD PLAYBACK-AFTER-*
                        ;                                         | RECORD ERROR   *
                        ;                                 XX IS THE BIOS RETURN CODE *
```

```
                                ;*************************************************************
                                ;
                                ; DATA
                                ;
0783  17                       T2_ICON         DB      T2_I-T2_ICON
0784  0A 0A 0B 05 2F BA                       DB      10,10,11,5,47,186,11,-4
      0B FC
078C  C9 CD 20 BA 0B FC                      DB      201,205,32,186,11,-4
0792  C8 CD 20 BA 0B FE                      DB      200,205,32,186,11,-2
0798  5C BA                                 DB      92,186
= 079A                          T2_I            =       $
079A  FF                                     DB      -1
079B  07                        T2_ABC          DB      T2_A-T2_ABC
079C  41 0B FD                               DB      'A',11,-3
079F  42 0A                                  DB      'B',10
07A1  43                                     DB      'C'
= 07A2                          T2_A            =       $
07A2  FF                                     DB      -1
07A3  0B                        T2_SELECT       DB      T2_S-T2_SELECT
07A4  87 4E 07 2C 87 4F                      DB      87H,'N',07H,',',87H ,'O',07,',',87H,'P'
      07 2C 87 50
= 07AE                          T2_S            =       $
07AE  A2 A2 A2 A2 A2 FD                      DB      161+1,161+1,161+1,161+1,161+1,-3,40-9
      1F
07B5  7C FC                                  DB      121+3,-4
07B7  17                        T2_WAVE         DB      T2_W-T2_WAVE
07B8  2D 2D 2D 3E 0B FC                      DB      '--->',11,-4
07BE  2D 2D 2D 3E 0B FC                      DB      '--->',11,-4
07C4  2D 2D 2D 3E 0B FC                      DB      '--->',11,-4
07CA  2D 2D 2D 3E                            DB      '--->'


= 07CE                          T2_W            =       $
07CE  FF                                     DB      -1
07CF  23                        MIC_ICON        DB      MIC_I-MIC_ICON
07D0  B0 B0 0B FE                            DB      176,176,11,-2
07D4  B0 B0 0B FE                            DB      176,176,11,-2
07D8  B0 B0 0B FE                            DB      176,176,11,-2
07DC  B3 B3 0B FE                            DB      179,179,11,-2
07E0  B3 B3 0B FE                            DB      179,179,11,-2
07E4  B3 B3 0B FE                            DB      179,179,11,-2
07E8  B3 B3 0B FE                            DB      179,179,11,-2
07EC  B3 B3 0B FE                            DB      179,179,11,-2
07F0  C0 D9                                  DB      192,217
= 07F2                          MIC_I           =       $
07F2  FF                        ARROW           DB      -1
07F3  04                                     DB      A-ARROW
07F4  2D 3E 3C                               DB      '-><'
= 07F7                          A               =       $
07F7  03 01 FD 27 7D 01                      DB      3,1,-3,40-1,121+4,1,3,-4
      03 FC


                                ;*****************************************
                                ;       DIAGNOSTIC ENTRY POINT
                                ;*****************************************
07FF                            TALKER2_DIAG    PROC    FAR
07FF  80 FC 01                          CMP     AH,01           ;CALL FOR SCREEN SETUP?
0802  76 12                             JBE     TALKER_ICON
0804  80 FC 4E                          CMP     AH,'N'          ;CALL FOR LPC TEST?
0807  74 2D                             JZ      LPC_TEST

0809  80 FC 4F                          CMP     AH,'O'          ;CALL FOR CVSD PLAYBACK?
080C  74 75                             JZ      CVSD_TEST
080E  80 FC 50                          CMP     AH,'P'          ;CALL FOR CVSD RECORD?
0811  75 22                             JNZ     GOODBYE         ;
0813  E9 08D3 R                         JMP     CVSD_REC

                                ;*****************
                                ; SCREEN SETUP
                                ;*****************
                                ; PUT THE ICON AND ITS SELECTION CHARATER ON THE DCP MENU
                                ;
0816                            TALKER_ICON:
0816  B4 07                             MOV     AH,TLK_WIDTH    ;WIDTH OF THE ICON
0818  BD 0783 R                         MOV     BP,OFFSET T2_ICON
081B  CD 81                             INT     LOCATE          ;LOCATE POSITION ON MENU
081D  52                                PUSH    DX              ;SAVE IT
081E  CD 82                             INT     PRINT           ;PUT ICON ON SCREEN
0820  BD 079B R                         MOV     BP,OFFSET T2_ABC
0823  5A                                POP     DX              ;TO ADJUST ROW & COL FOR 'ABC'
0824  52                                PUSH    DX              ;SAVE AGAIN
0825  81 C2 0309                       ADD     DX,0309H        ;
0829  CD 82                             INT     PRINT           ;PUT 'ABC'
082B  BD 07A3 R                         MOV     BP,OFFSET T2_SELECT
082E  5A                                POP     DX              ;ROW & COL FOR THE SELECTION ID



082F  81 C2 0803                        ADD     DX,0803H        ;
0833  CD 82                             INT     PRINT           ;PUT SELECTION ID ON SCREEN
0835                            GOODBYE:
0835  CF                                IRET                    ;RETURN TO DCP

                                ;*************************************************************
                                ; LPC DIAGNOSTIC
                                ;       SPEAK THE FIRST 10 WORDS IN THE ROM VOCABULARY
                                ;*************************************************************
0836                            LPC_TEST:
0836  E8 0966 R                       CALL    RESET           ;RESET CARD
0839  72 4D                             JC      DIAG_RESET_ERR  ;ERROR
083B                            LPC_SPEAK:
083B  BD 0783 R                         MOV     BP,OFFSET T2_ICON
083E  BA 070E                           MOV     DX,SPEAKER_POS  ;START CURSOR AT ROW 8 COL 16
0841  CD 82                             INT     PRINT           ;PUT ICON ON SCREEN
0843  BD 07B7 R                         MOV     BP,OFFSET T2_WAVE
0846  BA 0A15                           MOV     DX,WAVE_POS     ;ROW & COL FOR SOUND WAVE
0849  CD 82                             INT     PRINT           ;PUT SOUND WAVE
084B  B9 000A                           MOV     CX,10           ;COUNTER FOR SPEAKING 10 WORDS
084E  BB 0005                           MOV     BX,5            ;BEGIN WITH WORD 6
0851                            SPEAK:
0851  51                               PUSH    CX
0852  B8 0201                          MOV     AX,0201H        ;LPC SPEAK WITH WORD INDEX
0855  43                               INC     BX              ;NEXT WORD INDEX
0856  CD 4D                           INT     TALKER          ;SPEAK ....
0858  3C 00                           CMP     AL,OK           ;PASSED?
085A  74 09                           JE      STATUS_CK       ;YES, WAIT TILL LPC SPEAK DONE
085C                            LPC_ERR:
085C  59                                POP     CX
085D  B6 43                             MOV     DH,ER_LPC_C2    ;ERROR 'C' IN CUSTOMER LEVEL
085F  B7 11                             MOV     BH,ER_LPC_S2    ;ERROR 1100 IN SERVICE LEVEL
0861  8A D8                             MOV     BL,AL
0863  EB 15                             JMP     SHORT TK_EX_LINK1
```

```
0865                            STATUS_CK:
0865  33 C9                             XOR     CX,CX           ;SET FOR MAXIMUM # OF LOOPS
0867                            STATUS_CK2:
0867  B0 00                             MOV     AL,0
0869  E6 A0                             OUT     NMI_PORT,AL     ;MASK NMI
086B  B4 02                             MOV     AH,02           ;LPC STATUS FUNCTION (AX=0200)
086D  CD 4D                             INT     TALKER          ;BIOS CALL
086F  3C 00                             CMP     AL,OK           ;LPC READY TO SPEAK?
0871  E0 F4                             LOOPNE  STATUS_CK2      ;NO, WAIT...
0873  E3 E7                             JCXZ    LPC_ERR         ;IF CX=0 THEN TIMEOUT ERROR
0875  59                               POP     CX
0876  E2 D9                             LOOP    SPEAK           ;SPEAK NEXT WORD
0878                            GOOD_TEST_END:
0878  32 F6                             XOR     DH,DH           ;ZERO DH FOR GOOD RETURN
087A                            TK_EX_LINK1:
087A  E4 A0                             IN      AL,NMI_PORT
087C  B0 80                             MOV     AL,80H
087E  E6 A0                             OUT     NMI_PORT,AL     ;ENABLE NMI



0880  E9 0960 R                         JMP     TK_EX           ;EXIT

                                ;******************
                                ; CVSD PLAYBACK
                                ;******************
0883                            CVSD_TEST:
0883  E8 0966 R                         CALL    RESET           ;RESET CARD
0886  73 09                             JNC     CONTINUE_CVSD_TEST
0888                            DIAG_RESET_ERR:
0888  B6 42                             MOV     DH,ER_LPC_C1    ;RESET ERROR "B"
088A  B7 10                             MOV     BH,ER_LPC_S1    ;SERVICE LEVEL RESET ERR
088C  8A D8                             MOV     BL,AL           ;BIOS RETURN CODE
088E  E9 0960 R                         JMP     TK_EX

0891                            CONTINUE_CVSD_TEST:
0891  BD 0783 R                         MOV     BP,OFFSET T2_ICON
0894  BA 070E                           MOV     DX,SPEAKER_POS
0897  CD 82                             INT     PRINT           ;PRINT SPEAKER

0899  BD 07B7 R                         MOV     BP,OFFSET T2_WAVE
089C  BA 0A15                           MOV     DX,WAVE_POS
089F  CD 82                             INT     PRINT           ;PRINT WAVE

08A1  B8 0100                           MOV     AX,100H         ;START ON 4K BOUNDARY
08A4  8E D8                             MOV     DS,AX           ;THAT IS 100:0
08A6  8E C0                             MOV     ES,AX
08A8  33 FF                             XOR     DI,DI           ;BEGIN AT OFFSET 0
08AA  33 F6                             XOR     SI,SI
08AC  FC                                CLD
08AD  B9 0640                           MOV     CX,4800*4/12    ;LOAD ENOUGH BYTES TO SOUND A TONE
                                                                ;FOR 4 SECONDS
08B0  33 C0                             XOR     AX,AX
08B2                            LOAD_LOOP:
08B2  AB                                STOSW                   ;FILL RAM WITH PATTERN FOR TONE
08B3  AB                                STOSW                   ;THIS LOADS 6 BYTES OF 00'S
08B4  AB                                STOSW
08B5  48                                DEC     AX
08B6  AB                                STOSW                   ;THIS LOADS 6 BYTES OF FF'S
08B7  AB                                STOSW
08B8  AB                                STOSW
08B9  40                                INC     AX

08BA  E2 F6                             LOOP    LOAD_LOOP

08BC  B9 4B00                           MOV     CX,4800*4       ;SOUND A TONE FOR 4 SECONDS
08BF  BB 0101                           MOV     AX,0101H        ;CVSD WRITE
08C2  B3 05                             MOV     BL,5            ;AT 4800 BPS
08C4  CD 4D                             INT     TALKER          ;GO AND TALK

08C6  0A C0                             OR      AL,AL           ;ERROR OCCURRED?
08C8  74 AE                             JZ      GOOD_TEST_END   ;IF NOT, GO ON AND EXIT

08CA  B6 44                             MOV     DH,ER_CVSD_C1   ;SETUP FOR ERROR RETURN
08CC  B7 12                             MOV     BH,ER_CVSD_S1



08CE  8A D8                             MOV     BL,AL
08D0                            TK_EX_LINK2:

08D0  E9 0960 R                         JMP     TK_EX           ;EXIT TALKER DIAGNOSTIC

                                ;************************
                                ; CVSD RECORD TEST
                                ;************************
08D3                            CVSD_REC:
08D3  E8 0966 R                         CALL    RESET           ;RESET CARD
08D6  72 B0                             JC      DIAG_RESET_ERR  ;ERROR

08D8  BA 0812                           MOV     DX,MIC_POS
08DB  BD 07CF R                         MOV     BP,OFFSET MIC_ICON
08DE  CD 82                             INT     PRINT           ;PUT UP MICROPHONE ON SCREEN
08E0  BA 890D                           MOV     DX,ARROW_POS1
08E3  B3 87                             MOV     BL,87H          ;BLINKING NORMAL INTENSITY ATTRIB.
08E5  BD 07F3 R                         MOV     BP,OFFSET ARROW
08E8  CD 82                             INT     PRINT           ;PUT UP BLINKING ARROWS ON SCREEN

08EA  B9 003C                           MOV     CX,60           ;DELAY
08ED  E8 0976 R                         CALL    DELAY

08F0  E4 61                             IN      AL,PORT_61H
08F2  24 9F                             AND     AL,CLR_SPKSW
08F4  E6 61                             OUT     PORT_61H,AL     ;POINT SYSTEM SOUND MUX AT BEEPER
08F6  BA 890D                           MOV     DX,ARROW_POS2
08F9  B3 00                             MOV     BL,00
08FB  CD 82                             INT     PRINT           ;BLANK ARROW
08FD  B3 01                             MOV     BL,1            ;SETUP FOR SHORT BEEP
08FF  E8 0298 R                         CALL    BEEP

0902  B8 0100                           MOV     AX,0100H        ;SELECT CVSD READ
0905  B3 05                             MOV     BL,5            ;4800 BYTES PER SECOND
0907  B9 5DC0                           MOV     CX,4800*5       ;5 SECONDS RECORDING TIME
090A  BE 0100                           MOV     SI,100H         ;BEGIN ON THE 4K BOUNDARY
090D  8E DE                             MOV     DS,SI
090F  33 F6                             XOR     SI,SI           ;OFFSET OF ZERO
0911  56                                PUSH    SI
0912  1E                               PUSH    DS
0913  51                               PUSH    CX
0914  53                               PUSH    BX
0915  CD 4D                             INT     TALKER

0917  0A C0                             OR      AL,AL           ;ERROR OCCURRED?
0919  74 09                             JZ      RECORD_OK       ;IF NO ERROR OCCURRED, GO ON
```

```
091B  83 C4 08                ADD     SP,8                    ;OTHERWISE FALL THROUGH
091E  B6 45                   MOV     DH,ER_CVSD_C2           ;ADJUST STACK FOR EXIT
0920  B7 13                   MOV     BH,ER_CVSD_S2           ;SETUP ERROR RETURN CODES
0922  EB 36                   JMP     SHORT TK_EX1
0924                    RECORD_OK:



0924  33 C0                   XOR     AX,AX
0926  CD 10                   INT     10H                     ;CLEAR SCREEN
0928  B4 01                   MOV     AH,1
092A  B5 20                   MOV     CH,20H
092C  CD 10                   INT     10H                     ;TURN OFF CURSOR

092E  BD 0783 R               MOV     BP,OFFSET T2_ICON
0931  BA 070E                 MOV     DX,SPEAKER_POS
0934  CD 82                   INT     PRINT                   ;PUT UP SPEAKER ON THE SCREEN
0936  B3 01                   MOV     BL,1                    ;SETUP FOR SHORT BEEP
0938  E8 0298 R               CALL    BEEP

093B  B9 0028                 MOV     CX,40                   ;DELAY .8 OF SECOND
093E  E8 0976 R               CALL    DELAY                   ;DELAY BEFORE PLAYBACK

0941  BD 07B7 R               MOV     BP,OFFSET T2_WAVE
0944  BA 0A15                 MOV     DX,WAVE_POS
0947  CD 82                   INT     PRINT                   ;PUT UP ARROWS COMING FROM SPEAKER

0949  5B                     POP     BX
094A  59                     POP     CX
094B  1F                     POP     DS
094C  5E                     POP     SI
094D  B8 0101                MOV     AX,0101H
0950  CD 4D                   INT     TALKER                  ;PLAYBACK

0952  0A C0                   OR      AL,AL
0954  74 08                   JZ      PLAYBACK_OK             ;ERROR OCCURRED ON PLAYBACK?
                                                              ;IF NOT GO ON
0956  B6 44                   MOV     DH,ER_CVSD_C1           ;IF SO FALL THROUGH
0958  B7 14                   MOV     BH,ER_CVSD_S3
095A                    TK_EX1:
095A  8A D8                   MOV     BL,AL
095C  EB 02                   JMP     SHORT TK_EX
095E                    PLAYBACK_OK:
095E  B6 00                   MOV     DH,0                    ;SETUP NO ERROR RETURN

                        ;---------------------------------------------------------------
                        ; RETURN TO DCP                                                :
                        ;       TEST PASSED:                                           :
                        ;               DH = 0                                         :
                        ;       TEST FAILED:                                           :
                        ;               DH = ASCII ERROR CODE IN CUSTOMER LEVEL        :
                        ;               BX = ERROR CODE IN SERVICE LEVEL               :
                        ;---------------------------------------------------------------

0960                    TK_EX:
0960  B2 00                   MOV     DL,0
0962  F9                     STC
0963  CA 0002                 RET     2
0966                    TALKER2_DIAG    ENDP

                        ;---------------------------------------------------------------
                        ; RESET                                                        :
                        ;       RESET CARD TO NORMAL CONDITION                         :
                        ;---------------------------------------------------------------

0966                    RESET   PROC    NEAR
0966  33 C0                   XOR     AX,AX                   ;CLEAR  AH
0968  CD 4D                   INT     TALKER
                                                              ;TO RESET CARD
096A  3C 00                   CMP     AL,OK                   ;RESET OK?
096C  74 07                   JZ      RESET_OK                ;YES
096E  B6 42                   MOV     DH,ER_LPC_C1            ;ERROR 'B' IN CUSTOMER LEVEL
0970  B7 10                   MOV     BH,ER_LPC_S1            ;ERROR 10XX IN SERVICE LEVEL
0972  8A D8                   MOV     BL,AL
0974  F9                     STC
0975                    RESET_OK:
0975  C3                     RET
0976                    RESET   ENDP

                        ;---------------------------------------------------------------
                        ; DELAY                                                        :
                        ;       THIS ROUTINE WAITS APPROXIMATELY CX * .10 SECONDS      :
                        ;       BEFORE RETURNING                                       :
                        ; ON ENTRY:                                                    :
                        ;       CX = DELAY TIME                                        :
                        ; ON EXIT:                                                     :
                        ;       DX = 0                                                 :
                        ;---------------------------------------------------------------

0976                    DELAY   PROC    NEAR
0976                    DEL10:
0976  51                     PUSH    CX
0977  B9 3340                MOV     CX,13120                ;DECIMAL VALUE TO GIVE WAIT TIME
                                                              ;OF .1 SECOND
097A                    DEL20:
097A  E2 FE                   LOOP    DEL20
097C  59                     POP     CX
097D  E2 F7                   LOOP    DEL10
097F  C3                     RET
0980                    DELAY   ENDP
                                ASSUME  DS:DATA
                        ;-----------------------------------------
                        ;NMIOFF
                        ;       THIS PROCEDURE IS CALLED TO DISABLE NMI AND SELECTED
                        ;       INTERRUPTS ON THE 8259.
                        ;INPUT
                        ;       BL=MASK TO DISABLE 8259 INTERRUPTS
                        ;OUTPUT
                        ;       AX=INITIAL TIMER1 VALUE
                        ;       BL=ORIGIONAL 8259 MASK
                        ;-----------------------------------------
0980                    NMIOFF  PROC    NEAR

                        ;---------------------------------------------------------------


                        ;***NOTE***
                        ;ALL INTERRUPTS ARE ABOUT TO BE DISABLED.  THERE IS A POTENTIAL
```

```
                        ;       THAT THIS TIME PERIOD WILL BE LONG ENOUGH TO MISS TIME OF
                        ;       DAY INTERRUPTS.  FOR THIS REASON, TIMER1 WILL BE USED TO
                        ;       KEEP TRACK OF THE NUMBER OF TIME OF DAY INTERRUPTS WHICH
                        ;       WILL BE MISSED. THIS INFORMATION IS USED AFTER THE
                        ;       OPERATION TO UPDATE THE TIME OF DAY.
                        ;-------------------------------------------------------------------
0980  B0 10                     MOV     AL,10H          ; DISABLE NMI
0982  E6 A0                     OUT     NMI_PORT,AL     ; NO KEYBOARD INTERRUPT
0984  B0 70                     MOV     AL,70H          ; SELECT TIMER1,LSB-MSB, MODE 0,
                                                        ; BINARY COUNTER
0986  E6 43                     OUT     TIM_CTL,AL      ; INIT THE COUNTER
0988  50                       PUSH    AX              ; DELAY
0989  58                       POP     AX
098A  B0 FF                     MOV     AL,0FFH         ; INITIAL COUNT
098C  E6 41                     OUT     TIMER+1,AL      ; OUTPUT LEAST SIGNIFICANT BYTE
098E  50                       PUSH    AX              ; DELAY
098F  58                       POP     AX
0990  E6 41                     OUT     TIMER+1,AL      ; OUTPUT MOST SIGNIFICANT BYTE
0992  E8 0A2D R                 CALL    CLOCK_WAIT      ; WAIT IF TIMER0 IS ABOUT TO
                                                        ; INTERRUPT

0995  B0 30                     MOV     AL,30H          ; SELECT TIMER1 INPUT FROM TIMER0
                                                        ; OUTPUT
0997  E6 A0                     OUT     NMI_PORT,AL
                        ;------ READ TIMER1 NOW AND SAVE THE INITIAL VALUE
0999  E8 0A41 R                 CALL    READ_TIME       ; GET TIMER1 VALUE IN AX
099C  50                       PUSH    AX              ; SAVE INITIAL VALUE FOR CLOCK
099D  E4 21                     IN      AL,INTA01       ; READ CURRENT MASK
099F  86 C3                     XCHG    AL,BL           ; SWAP OLD AND NEW MASKS
09A1  E6 21                     OUT     INTA01,AL       ; OUTPUT MASK TO THE 8259
09A3  58                       POP     AX
09A4  C3                       RET
09A5                    NMIOFF  ENDP

                        ;-----------------------------------------------
                        ;NMION
                        ;       THIS PROCEDURE IS CALLED TO ENABLE NMI AND SELECTED
                        ;       INTERRUPTS ON THE 8259 AND UPDATE TOD INFO
                        ;INPUT
                        ;       AX=INITIAL TIMER1 VALUE FROM NMIOFF
                        ;       BL=8259 MASK
                        ;OUTPUT
                        ;       NMI & 8259 ARE ENABLED AND TOD IS UPDATED
                        ;-----------------------------------------------
09A5                    NMION   PROC    NEAR
09A5  1E                       PUSH    DS
09A6  53                       PUSH    BX
09A7  50                       PUSH    AX
09A8  B8 0040                  MOV     AX,40H          ; POINT DS AT BIOS DATA SEGMENT
09AB  8E D8                    MOV     DS,AX
09AD  E8 0A2D R                CALL    CLOCK_WAIT      ; WAIT IF TIMER0 IS CLOSE TO



                                                        ; WRAPPING
09B0  E8 0A41 R                 CALL    READ_TIME
09B3  5B                       POP     BX              ; GET THE INITIAL VALUE OF TIMER1
09B4  2B D8                    SUB     BX,AX           ; UPDATE NUMBER OF INTERRUPTS
                                                        ; MISSED
09B6  B0 0A                    MOV     AL,0AH          ; SEE IF THERE IS A PENDING
09B8  E6 20                    OUT     INTA00,AL       ; INTERRUPT FROM TIMER 0
09BA  E4 20                    IN      AL,INTA00

09BC  24 01                    AND     AL,1
09BE  74 15                    JZ      J16_2
09C0  83 EB 01                 SUB     BX,1            ; YES ACCOUNT FOR IT
09C3  73 10                    JAE     J16_2           ; OK IF RESULT >= 0
09C5  33 DB                    XOR     BX,BX
09C7  53                       PUSH    BX              ; SAVE 0 AS COUNT IN ISSUING USER
                                                        ; TIMER INTERRUPTS
09C8  83 2E 006C R 01          SUB     TIMER_LOW,1     ; SUBTRACT 1 FROM TIMER
09CD  73 31                    JNC     J16_5           ; OK IF NO BORROW
09CF  FF 0E 006E R             DEC     TIMER_HIGH
09D3  EB 2B                    JMP     SHORT   J16_5

09D5  53                J16_2: PUSH    BX              ; SAVE IT FOR REUSE IN ISSUING USER
                                                        ; TIMER INTERRUPTS
09D6  01 1E 006C R             ADD     TIMER_LOW,BX    ; ADD NUMBER OF TIMER INTERRUPTS TO
                                                        ; TIME
09DA  73 04                    JNC     J16_4           ; JUMP IF TIMER_LOW DID NOT SPILL
                                                        ; OVER TO TIMER_HI
09DC  FF 06 006E R             INC     TIMER_HIGH
09E0  83 3E 006E R 18   J16_4: CMP     TIMER_HIGH,018H ; TEST FOR COUNT TOTALING 24 HOURS
09E5  75 19                    JNZ     J16_5           ; JUMP IF NOT 24 HOURS
09E7  81 3E 006C R 00B0        CMP     TIMER_LOW,0B0H  ; LOW VALUE = 24 HOUR VALUE?
09ED  7C 11                    JL      J16_5           ; NOT 24 HOUR VALUE?
                        ;------ TIMER HAS GONE 24 HOURS
09EF  C7 06 006E R 0000        MOV     TIMER_HIGH,0    ; ZERO OUT TIMER_HIGH VALUE
09F5  81 2E 006C R 00B0        SUB     TIMER_LOW,0B0H  ; VALUE REFLECTS CORRECT TICKS PAST
                                                        ; 00B0H
09FB  C6 06 0070 R 01          MOV     TIMER_OFL,1     ; INDICATES 24 HOUR THRESHOLD
0A00  58                J16_5: POP     AX              ; RECOVER COUNT

0A01  5B                       POP     BX              ;RECOVER 8259 MASK
0A02  50                       PUSH    AX              ; SAVE COUNT
0A03  E8 0A52 R                CALL    ENABLE          ; ENABLE ALL INTERRUPTS
0A06  59                       POP     CX              ; CX:=AX, COUNT FOR NUMBER OF USER
                                                        ; TIME INTERRUPTS
0A07  E3 0A                    JCXZ    J16_61          ; IF ZERO DO NOT ISSUE ANY
                                                        ; INTERRUPTS
0A09  1E                       PUSH    DS              ; SAVE ALL REGISTERS SAVED PRIOR TO
                                                        ; INT 1C CALL FROM TIMERINT
0A0A  50                       PUSH    AX              ; THIS PROVIDES A COMPATIBLE
                                                        ; INTERFACE TO 1C
0A0B  52                       PUSH    DX              ;
0A0C                    J16_6:
0A0C  CD 1C                    INT     1CH             ; TRANSFER CONTROL TO USER
                                                        ; INTERRUPT



0A0E  E2 FC                    LOOP    J16_6           ; DO ALL USER TIMER INTERRUPTS
0A10  5A                       POP     DX
0A11  58                       POP     AX
0A12  1F                       POP     DS              ; RESTORE REGISTERS
                        ;------ CLOCK IS UPDATED AND USER INTERRUPTS 1C HAVE BEEN ISSUED.
                        ;       CHECK IF KEYSTROKE OCCURED
0A13  0A C0             J16_61: OR      AL,AL           ; AL WAS SET DURING CALL TO ENABLE
0A15  74 14                    JZ      J16_7           ; NO KEY WAS PRESSED WHILE SYSTEM
                                                        ; WAS MASKED
                        ;-------CLEAR SHIFT STATES-DONT LEAVE POSSIBILTY OF DANGLING STATES
                        ;       OF MISSED BREAKS, AND NOTIFY USER OF MISSED KEYBOARD INPUT
                        ;       CLEAR ALT,CLRL,LEFT AND RIGHT SHIFTS
                        ;       CLEAR POTENTIAL BREAK OF INS,CAPS,NUM AND SCROLL SHIFT
0A17  81 26 0017 R 0FF0        AND     WORD PTR KB_FLAG,0FF0H
```

```
0A1D  80 26 0088 R 1F           AND     KB_FLAG_2,1FH   ;CLEAR FUNCTION STATES
0A22  BB 0080                   MOV     BX,80H          ;BEEP DURATION
0A25  B9 0048                   MOV     CX,48H          ;BEEP HALF CYCLE FREQUENCY
0A28  E8 0A74 R                 CALL    KB_NOISE        ;INDICATE MISSED KEY
0A2B                    J16_7:
0A2B  1F                        POP     DS
0A2C  C3                        RET
0A2D                    NMION   ENDP

                        ;--------------------------------------------
                        ;CLOCK_WAIT
                        ;       THIS PROCEDURE IS CALLED WHEN THE TIME OF DAY
                        ;       IS BEING UPDATED.  IT WAITS IF TIMER0 IS ALMOST
                        ;       READY TO WRAP UNTIL IT IS SAFE TO READ AN ACCURATE
                        ;       TIMER1.
                        ;INPUT
                        ;       NONE.
                        ;OUTPUT
                        ;       NONE. AX IS DESTROYED.
                        ;--------------------------------------------
0A2D                    CLOCK_WAIT      PROC    NEAR
0A2D  32 C0                     XOR     AL,AL           ; READ MODE TIMER0 FOR 8253
0A2F  E6 43                     OUT     TIM_CTL,AL      ; OUTPUT TO THE 8253
0A31  50                        PUSH    AX
0A32  58                        POP     AX              ; WAIT FOR 8253 TO INITIALIZE
                                                        ; ITSELF
0A33  E4 40                     IN      AL,TIMER0       ; READ LEAST SIGNIFICANT BYTE
0A35  86 C4                     XCHG    AL,AH           ; SAVE IT
0A37  E4 40                     IN      AL,TIMER0       ; READ MOST SIGNIFICANT BYTE
0A39  86 C4                     XCHG    AL,AH           ; REARRANGE FOR PROPER ORDER
0A3B  3D 012C                   CMP     AX,THRESHOLD    ; IS TIMER0 CLOSE TO WRAPPING?
0A3E  72 ED                     JC      CLOCK_WAIT      ; JUMP IF CLOCK IS WITHIN THRESHOLD
0A40  C3                        RET                     ; OK TO READ TIMER1
0A41                    CLOCK_WAIT      ENDP

                        ;--------------------------------------------------------------------
                        ;THIS ROUTINE WILL READ TIMER1.  THE VALUE READ IS RETURNED IN AX.
                        ;--------------------------------------------------------------------
0A41                    READ_TIME PROC NEAR


0A41  B0 40                     MOV     AL,40H          ; LATCH TIMER1
0A43  E6 43                     OUT     TIM_CTL,AL
0A45  50                        PUSH    AX              ; WAIT FOR 8253 TO INIT ITSELF
0A46  58                        POP     AX
0A47  E4 41                     IN      AL,TIMER+1      ; READ LSB
0A49  8A E0                     MOV     AH,AL           ; SAVE IT IN HIGH BYTE
0A4B  50                        PUSH    AX              ; WAIT FOR 8253 TO INIT ITSELF
0A4C  58                        POP     AX
0A4D  E4 41                     IN      AL,TIMER+1      ; READ MSB
0A4F  86 C4                     XCHG    AL,AH           ; PUT BYTES IN PROPER ORDER
0A51  C3                        RET
0A52                    READ_TIME       ENDP
                        ;--------------------------------------------
                        ; ENABLE
                        ;       THIS PROC ENABLES ALL INTERRUPTS.  IT ALSO SETS THE 8253 TO
                        ;       THE MODE REQUIRED FOR KEYBOARD DATA DESERIALIZATION.
                        ;       BEFORE THE LATCH FOR KEYBOARD DATA IS RESET, BIT 0 OF THE
                        ;       8255 IS READ TO DETERMINE WHETHER ANY KEYSTROKES OCCURED
                        ;       WHILE THE SYSTEM WAS MASKED OFF.
                        ; INPUT
                        ;       BL=8259 MASK
                        ; OUTPUT
                        ;       AL=1 MEANS A KEY WAS STRUCK DURING DISKETTE I/O.  (OR NOISE
                        ;            ON THE LINE)
                        ;       AL=0 MEANS THAT NO KEY WAS PRESSED.
                        ;       AX IS DESTROYED.  ALL OTHER REGISTERS REMAIN INTACT.
                        ;--------------------------------------------
0A52                    ENABLE          PROC    NEAR
0A52  52                        PUSH    DX              ; SAVE DX
                        ;------ RETURN TIMER1 TO STATE NEEDED FOR KEYBOARD I/O
0A53  B0 76                     MOV     AL,01110110B    ;
0A55  E6 43                     OUT     TIM_CTL,AL
0A57  50                        PUSH    AX
0A58  58                        POP     AX              ; WAIT FOR 8253 TO INITIALIZE
                                                        ; ITSELF
0A59  B0 FF                     MOV     AL,0FFH         ; INITIAL VALUE FOR 8253
0A5B  E6 41                     OUT     TIMER+1,AL      ; LSB
0A5D  50                        PUSH    AX
0A5E  58                        POP     AX              ; WAIT
0A5F  E6 41                     OUT     TIMER+1,AL      ; MSB
                        ;------ CHECK IF ANY KEYSTROKES OCCURED DURING DISKETTE TRANSFER
0A61  E4 62                     IN      AL,62H          ; READ PORT C OF 8255
0A63  24 01                     AND     AL,01H          ; BIT=1 MEANS KESTROKE HAS OCCURED
0A65  50                        PUSH    AX              ; SAVE IT ON THE STACK
                        ;-----  ENABLE ALL INTERRUPTS WHICH WERE ENABLED BEFORE TRANSFER
0A66  8A C3                     MOV     AL,BL           ; GET MASK
0A68  E6 21                     OUT     INTA01,AL
0A6A  FB                        STI
                        ;------ ENABLE NMI INTERRUPTS
0A6B  E4 A0                     IN      AL,NMI_PORT     ; RESET LATCH
0A6D  B0 80                     MOV     AL,80H          ; MASK TO ENABLE NMI
0A6F  E6 A0                     OUT     NMI_PORT,AL     ; ENABLE NMI
0A71  58                        POP     AX              ; PASS BACK KEY STROKE FLAG



0A72  5A                        POP     DX
0A73  C3                        RET
0A74                    ENABLE          ENDP
                        ;--------------------------------------------------------------------
                        ;KB_NOISE
                        ;       THIS ROUTINE IS CALLED WHEN GENERAL BEEPS ARE REQUIRED FROM
                        ;       THE SYSTEM.
                        ;INPUT
                        ;       BX=LENGH OF THE TONE
                        ;       CX=CONTAINS THE FREQUENCY
                        ;OUTPUT
                        ;       ALL REGISTERS ARE MAINTAINED.
                        ;HINTS
                        ;       AS CX GETS LARGER THE TONE PRODUCED GETS LOWER IN PITCH.
                        ;
                        ;--------------------------------------------------------------------
0A74                    KB_NOISE        PROC    NEAR
0A74  FB                        STI
0A75  50                        PUSH    AX
0A76  53                        PUSH    BX
0A77  51                        PUSH    CX
0A78  E4 61                     IN      AL,061H         ; GET CONTROL INFO
0A7A  50                        PUSH    AX              ; SAVE
0A7B                    LOOP01:
0A7B  24 FC                     AND     AL,0FCH         ; TURN OFF TIMER GATE AND SPEAKER
                                                        ; DATA
```

```
0A7D  E6 61                     OUT     061H,AL         ; OUTPUT TO CONTROL
0A7F  51                        PUSH    CX              ; HALF CYCLE TIME FOR TONE
0A80  E2 FE                     LOOP    $               ; SPEAKER OFF
0A82  0C 02                     OR      AL,2            ; TURN ON SPEAKER BIT
0A84  E6 61                     OUT     061H,AL         ; OUTPUT TO CONTROL
0A86  59                        POP     CX
0A87  51                        PUSH    CX              ; RETRIEVE FREQUENCY
0A88  E2 FE                     LOOP    $               ; ANOTHER HALF CYCLE
0A8A  4B                        DEC     BX              ; TOTAL TIME COUNT
0A8B  59                        POP     CX              ; RETRIEVE  FREQ.
0A8C  75 ED                     JNZ     LOOP01          ; DO ANOTHER CYCLE
0A8E  58                        POP     AX              ; RECOVER CONTROL
0A8F  E6 61                     OUT     061H,AL         ; OUTPUT THE CONTROL
0A91  59                        POP     CX
0A92  5B                        POP     BX
0A93  58                        POP     AX
0A94  C3                        RET
0A95                    KB_NOISE        ENDP

0A95                    SETUP_FLAG2:
0A95  FC                        CLD                     ;CLEAR DIRECTION FLAG
0A96  53                        PUSH    BX
0A97  51                        PUSH    CX              ;SAVE REGISTERS
0A98  1E                        PUSH    DS
0A99  E9 06ED R                 JMP     FLAG_SETUP2     ;RETURN

0A9C                    SETUP_FLAG:


0A9C  FA                        CLI
0A9D  E8 075C R                 CALL    READ_PORTB
0AA0  FB                        STI
0AA1  E9 0380 R                 JMP     FLAG_SETUP

0B00                            ORG     0B00H
= 0B00                  TAB1DX  EQU     $
0B00  0C90 R                    DW      OFFSET A0006
0B02  0D01 R                    DW      OFFSET A0019
0B04  0EAE R                    DW      OFFSET A0020
0B06  10DC R                    DW      OFFSET A0022
0B08  116C R                    DW      OFFSET A0023
0B0A  11D7 R                    DW      OFFSET A0024
0B0C  1219 R                    DW      OFFSET A0025
0B0E  12A8 R                    DW      OFFSET A0026
0B10  130F R                    DW      OFFSET A0027
0B12  137D R                    DW      OFFSET A0028
0B14  13F8 R                    DW      OFFSET A0033
0B16  14A9 R                    DW      OFFSET A0034
0B18  1505 R                    DW      OFFSET B0001
0B1A  158B R                    DW      OFFSET B0002
0B1C  15F5 R                    DW      OFFSET B0003
0B1E  1643 R                    DW      OFFSET B000N
0B20  16B9 R                    DW      OFFSET B0005
0B22  1705 R                    DW      OFFSET B0006
0B24  1781 R                    DW      OFFSET B0007
0B26  17C9 R                    DW      OFFSET B0008
0B28  1849 R                    DW      OFFSET B0009
0B2A  18BB R                    DW      OFFSET B0010
0B2C  18F7 R                    DW      OFFSET B0011
0B2E  192D R                    DW      OFFSET B0012
0B30  1999 R                    DW      OFFSET B0013
0B32  19CB R                    DW      OFFSET B0014
0B34  1A27 R                    DW      OFFSET B0015
0B36  1A5B R                    DW      OFFSET B0016
0B38  1AC7 R                    DW      OFFSET B0017
0B3A  1B07 R                    DW      OFFSET B0018
0B3C  1B4B R                    DW      OFFSET B0019
0B3E  1BB5 R                    DW      OFFSET B0020
0B40  1C27 R                    DW      OFFSET B0021
0B42  1CA1 R                    DW      OFFSET B0022
0B44  1D17 R                    DW      OFFSET B0023
0B46  1DA1 R                    DW      OFFSET B0024
0B48  1E2D R                    DW      OFFSET B0025
0B4A  1EC7 R                    DW      OFFSET B0026
0B4C  1EF5 R                    DW      OFFSET B0027
0B4E  1F83 R                    DW      OFFSET B0028
0B50  1FFB R                    DW      OFFSET END_PGO
= 0B52                  TAB0    EQU     $
0B52  0C90 R                    DW      OFFSET B0029-2000H
0B54  0D20 R                    DW      OFFSET B0030-2000H
0B56  0D50 R                    DW      OFFSET B0031-2000H


0B58  0DAA R                    DW      OFFSET B0032-2000H
0B5A  0E1C R                    DW      OFFSET B0033-2000H
0B5C  0E82 R                    DW      OFFSET B0034-2000H
0B5E  0F1A R                    DW      OFFSET B0035-2000H
0B60  0F70 R                    DW      OFFSET B0036-2000H
0B62  1000 R                    DW      OFFSET B0037-2000H
0B64  1034 R                    DW      OFFSET B0038-2000H
0B66  1062 R                    DW      OFFSET B0039-2000H
0B68  10D8 R                    DW      OFFSET B0040-2000H
0B6A  1116 R                    DW      OFFSET B0041-2000H
0B6C  117C R                    DW      OFFSET B0042-2000H
0B6E  11F0 R                    DW      OFFSET B0043-2000H
0B70  122A R                    DW      OFFSET B0044-2000H
0B72  12D0 R                    DW      OFFSET B0045-2000H
0B74  1352 R                    DW      OFFSET B0046-2000H
0B76  13CA R                    DW      OFFSET B0047-2000H
0B78  141A R                    DW      OFFSET B0048-2000H
0B7A  1488 R                    DW      OFFSET B0049-2000H
0B7C  14E0 R                    DW      OFFSET B0050-2000H
0B7E  150E R                    DW      OFFSET B0051-2000H
0B80  1584 R                    DW      OFFSET B0052-2000H
0B82  1604 R                    DW      OFFSET B0053-2000H
0B84  1670 R                    DW      OFFSET B0054-2000H
0B86  170C R                    DW      OFFSET B0055-2000H
0B88  1766 R                    DW      OFFSET B0056-2000H
0B8A  17FE R                    DW      OFFSET C0001-2000H
0B8C  1844 R                    DW      OFFSET C0002-2000H
0B8E  18AA R                    DW      OFFSET C0003-2000H
0B90  18FB R                    DW      OFFSET C0004-2000H
0B92  1953 R                    DW      OFFSET C0005-2000H
0B94  19A2 R                    DW      OFFSET C0006-2000H
0B96  19F3 R                    DW      OFFSET C0007-2000H
0B98  1A57 R                    DW      OFFSET C0008-2000H
0B9A  1AD2 R                    DW      OFFSET C0009-2000H
0B9C  1B0F R                    DW      OFFSET C0010-2000H
0B9E  1B49 R                    DW      OFFSET C0011-2000H
0BA0  1B8B R                    DW      OFFSET C0012-2000H
0BA2  1BF9 R                    DW      OFFSET C0013-2000H
```

```
0BA4  1C4B R                          DW      OFFSET C0014-2000H
0BA6  1C9E R                          DW      OFFSET C0015-2000H
0BA8  1CF1 R                          DW      OFFSET C0016-2000H
0BAA  1D4C R                          DW      OFFSET C0017-2000H
0BAC  1DBE R                          DW      OFFSET C0018-2000H
0BAE  1E25 R                          DW      OFFSET C0019-2000H
0BB0  1E4C R                          DW      OFFSET C0020-2000H
0BB2  1EBB R                          DW      OFFSET C0021-2000H
0BB4  1F12 R                          DW      OFFSET C0022-2000H
0BB6  1F98 R                          DW      OFFSET C0023-2000H
0BB8  1FFE R                          DW      OFFSET END_PG1-2000H
= 0B8A                        TAB1    EQU     $
0BBA  0C90 R                          DW      OFFSET C0024-4000H
0BBC  0D15 R                          DW      OFFSET C0025-4000H
0BBE  0D6F R                          DW      OFFSET C0026-4000H


0BC0  0DD6 R                          DW      OFFSET C0027-4000H
0BC2  0E18 R                          DW      OFFSET C0028-4000H
0BC4  0E80 R                          DW      OFFSET C0029-4000H
0BC6  0ECA R                          DW      OFFSET C0030-4000H
0BC8  0F0E R                          DW      OFFSET C0031-4000H
0BCA  0F9C R                          DW      OFFSET C0032-4000H
0BCC  100D R                          DW      OFFSET C0033-4000H
0BCE  1079 R                          DW      OFFSET C0034-4000H
0BD0  10DA R                          DW      OFFSET C0035-4000H
0BD2  1118 R                          DW      OFFSET C0036-4000H
0BD4  117E R                          DW      OFFSET C0037-4000H
0BD6  11D7 R                          DW      OFFSET C0038-4000H
0BD8  123D R                          DW      OFFSET C0039-4000H
0BDA  12A5 R                          DW      OFFSET C0040-4000H
0BDC  1332 R                          DW      OFFSET C0041-4000H
0BDE  13DE R                          DW      OFFSET C0042-4000H
0BE0  1421 R                          DW      OFFSET C0043-4000H
0BE2  14D3 R                          DW      OFFSET C0044-4000H
0BE4  151E R                          DW      OFFSET C0045-4000H
0BE6  1567 R                          DW      OFFSET C0046-4000H
0BE8  15A7 R                          DW      OFFSET C0047-4000H
0BEA  1629 R                          DW      OFFSET C0048-4000H
0BEC  1689 R                          DW      OFFSET C0049-4000H
0BEE  16E9 R                          DW      OFFSET C0050-4000H
0BF0  175D R                          DW      OFFSET C0051-4000H
0BF2  17A8 R                          DW      OFFSET C0052-4000H
0BF4  17FF R                          DW      OFFSET C0053-4000H
0BF6  1821 R                          DW      OFFSET C0054-4000H
0BF8  189B R                          DW      OFFSET C0055-4000H
0BFA  18F1 R                          DW      OFFSET C0056-4000H
0BFC  1969 R                          DW      OFFSET C0057-4000H
0BFE  19D4 R                          DW      OFFSET C0058-4000H
0C00  1A20 R                          DW      OFFSET C0059-4000H
0C02  1A87 R                          DW      OFFSET D0001-4000H
0C04  1AAA R                          DW      OFFSET D0002-4000H
0C06  1B26 R                          DW      OFFSET D0003-4000H
0C08  1B5E R                          DW      OFFSET D0004-4000H
0C0A  1B9F R                          DW      OFFSET D0005-4000H
0C0C  1BF7 R                          DW      OFFSET D0006-4000H
0C0E  1C26 R                          DW      OFFSET D0007-4000H
0C10  1C4E R                          DW      OFFSET D0008-4000H
0C12  1C93 R                          DW      OFFSET D0009-4000H
0C14  1D15 R                          DW      OFFSET D0010-4000H
0C16  1DBD R                          DW      OFFSET D0011-4000H
0C18  1E23 R                          DW      OFFSET D0012-4000H
0C1A  1EB2 R                          DW      OFFSET D0013-4000H
0C1C  1F20 R                          DW      OFFSET D0014-4000H
0C1E  1F8B R                          DW      OFFSET D0015-4000H
0C20  1FD9 R                          DW      OFFSET END_PG2-4000H
= 0C22                        TAB2    EQU     $
0C22  0C90 R                          DW      OFFSET D0016-6000H
0C24  0D08 R                          DW      OFFSET D0017-6000H
0C26  0D6B R                          DW      OFFSET D0018-6000H


0C28  0DAE R                          DW      OFFSET D0019-6000H
0C2A  0E27 R                          DW      OFFSET D0020-6000H
0C2C  0E73 R                          DW      OFFSET D0021-6000H
0C2E  0EBE R                          DW      OFFSET D0022-6000H
0C30  0F04 R                          DW      OFFSET D0023-6000H
0C32  0F6C R                          DW      OFFSET D0024-6000H
0C34  0FC7 R                          DW      OFFSET D0025-6000H
0C36  1010 R                          DW      OFFSET D0026-6000H
0C38  107B R                          DW      OFFSET D0027-6000H
0C3A  10C2 R                          DW      OFFSET D0028-6000H
0C3C  1114 R                          DW      OFFSET D0029-6000H
0C3E  1193 R                          DW      OFFSET D0030-6000H
0C40  11F9 R                          DW      OFFSET D0031-6000H
0C42  1263 R                          DW      OFFSET D0032-6000H
0C44  12BD R                          DW      OFFSET D0033-6000H
0C46  1316 R                          DW      OFFSET D0034-6000H
0C48  137E R                          DW      OFFSET D0035-6000H
0C4A  13E7 R                          DW      OFFSET D0036-6000H
0C4C  1440 R                          DW      OFFSET D0037-6000H
0C4E  14BF R                          DW      OFFSET D0038-6000H
0C50  14F6 R                          DW      OFFSET D0039-6000H
0C52  1576 R                          DW      OFFSET D0040-6000H
0C54  15EE R                          DW      OFFSET D0041-6000H
0C56  1663 R                          DW      OFFSET D0042-6000H
0C58  16EA R                          DW      OFFSET D0043-6000H
0C5A  1763 R                          DW      OFFSET D0044-6000H
0C5C  17CF R                          DW      OFFSET D0045-6000H
0C5E  180F R                          DW      OFFSET D0046-6000H
0C60  186E R                          DW      OFFSET D0047-6000H
0C62  1879 R                          DW      OFFSET D0048-6000H
0C64  18F8 R                          DW      OFFSET D0049-6000H
0C66  1980 R                          DW      OFFSET D0050-6000H
0C68  19EF R                          DW      OFFSET D0051-6000H
0C6A  1A58 R                          DW      OFFSET D0052-6000H
0C6C  1ABE R                          DW      OFFSET D0053-6000H
0C6E  1B38 R                          DW      OFFSET D0054-6000H
0C70  1BCF R                          DW      OFFSET D0055-6000H
0C72  1C5A R                          DW      OFFSET D0056-6000H
0C74  1C9E R                          DW      OFFSET D0057-6000H
0C76  1D24 R                          DW      OFFSET D0058-6000H
0C78  1D4F R                          DW      OFFSET D0059-6000H
0C7A  1D97 R                          DW      OFFSET E0001-6000H
0C7C  1E02 R                          DW      OFFSET E0002-6000H
0C7E  1E31 R                          DW      OFFSET E0003-6000H
0C80  1E59 R                          DW      OFFSET E0004-6000H
0C82  1EDD R                          DW      OFFSET E0006-6000H
0C84  1F16 R                          DW      OFFSET E0007-6000H
0C86  1F37 R                          DW      OFFSET E0008-6000H
0C88  1F80 R                          DW      OFFSET E0010-6000H
0C8A  1F90 R                          DW      OFFSET E0011-6000H
```

```
0C8C  1FA0 R                          DW      OFFSET EQ012-6000H
0C8E  1FE9 R                          DW      OFFSET END_PG3-6000H
= 0C90                          TAB3  EQU     $


= 0029                          PG0_MAX EQU   ((TAB0-TABIDX)/2)
= 005D                          PG1_MAX EQU   ((TAB1-TABIDX)/2)
= 0091                          PG2_MAX EQU   ((TAB2-TABIDX)/2)
= 00C8                          PG3_MAX EQU   ((TAB3-TABIDX)/2)
= 0C90                          WORDS_BEGIN EQU   $
= 0C90                          A0006 EQU     $     ; DANGER
0C90  0E E8 C0 ED 18 D3               DB      0EH,0E8H,0C0H,0EDH,018H,0D3H,08CH,093H,0EBH
      8C 93 EB
0C99  96 53 6D 33 41 2E               DB      096H,053H,06DH,033H,041H,02EH,05BH,04EH,0B1H
      5B 4E B1
0CA2  DD 3E BA B4 39 D9               DB      0DDH,03AH,0BAH,0B4H,039H,0D9H,056H,089H,0DCH
      56 89 DC
0CAB  B1 68 64 83 28 C6               DB      0B1H,068H,064H,083H,028H,0C6H,0E6H,0ACH,092H
      E6 AC 92
0CB4  45 41 71 D2 51 53               DB      045H,041H,071H,0D2H,051H,053H,0E6H,024H,0D5H
      E6 24 D5
0CBD  45 84 19 D5 74 C4               DB      045H,084H,019H,0D5H,074H,0C4H,02AH,0D1H,074H
      2A D1 74
0CC6  B5 BB 4E 25 91 D4               DB      0B5H,0BBH,04EH,025H,091H,0D4H,035H,0CCH,0D8H
      35 CC D8
0CCF  C6 22 4E 56 DF 62               DB      0C6H,022H,04EH,056H,0DFH,062H,0A1H,084H,075H
      A1 84 75
0CD8  5D 7D 0D 8D 9A 92               DB      05DH,07DH,0DH,08DH,09AH,092H,079H,0DH,035H
      79 0D 35
0CE1  2E 7A 48 D6 35 D6               DB      02EH,07AH,048H,0D6H,035H,0D6H,088H,068H,029H
      B8 68 29
0CEA  59 DB 50 9A 61 86               DB      059H,0DBH,050H,09AH,061H,086H,06CH,09H,043H
      6C 09 43
0CF3  B4 8E B3 92 C1 F6               DB      0B4H,08EH,0B3H,092H,0C1H,0F6H,062H,0C8H,09H
      62 C8 09
0CFC  A9 52 16 FE 3F                  DB      0A9H,052H,016H,0FEH,03FH

= 0D01                          A0019 EQU     $     ; TIME HAS EXPIRED
0D01  0E 68 C4 24 02 63               DB      0EH,068H,0C4H,024H,02H,063H,0E8H,05AH,092H
      E8 5A 92
0D0A  29 63 9F B1 B8 4C               DB      029H,063H,09FH,0B1H,0B8H,04CH,089H,034H,07DH
      89 34 7D
0D13  A6 94 33 A5 B3 E9               DB      0A6H,094H,033H,0A5H,0B3H,0E9H,059H,072H,0E8H
      59 72 E8
0D1C  52 EF 8E 67 CD A9               DB      052H,0EFH,08EH,067H,0CDH,0A9H,04BH,023H,0D6H
      4B 23 D6
0D25  9C 35 E5 49 F5 EE               DB      09CH,035H,0E5H,049H,0F5H,0EEH,074H,08EH,0E2H
      74 8E E2
0D2E  6B CC 75 F3 39 73               DB      06BH,0CCH,075H,0F3H,039H,073H,0ACH,076H,08BH
      AC 76 8B
0D37  4D E7 2C D9 CB 22               DB      04DH,0E7H,02CH,0D9H,0CBH,022H,056H,0AFH,02BH
      56 AF 2B
0D40  17 2D 8F 9E B5 EE               DB      017H,02DH,08FH,09EH,0B5H,0EEH,0D2H,0A4H,0B4H
      D2 A4 B4
0D49  7A 55 B9 63 D6 EE               DB      07AH,055H,0B9H,063H,0D6H,0EEH,0AEH,018H,0E3H
      AE 18 E3
0D52  2E 41 32 3A 28 95               DB      02EH,041H,032H,03AH,028H,095H,0ABH,07AH,08AH
      AB 7A 8A
0D5B  28 9D E4 AE 68 A5               DB      028H,09DH,0E4H,0AEH,068H,0A5H,05CH,033H,0B5H
      5C 33 B5
0D64  3B A3 91 32 ED 34               DB      03BH,0A3H,091H,032H,0EDH,034H,0EEH,08CH,056H
      EE 8C 56
0D6D  32 35 DB B8 23 7B               DB      032H,035H,0DBH,0B8H,023H,07BH,09H,091H,0CCH
      09 91 CC
0D76  E2 8E 62 39 54 22               DB      0E2H,08EH,062H,039H,054H,022H,069H,0D8H,093H
      69 D8 93
0D7F  E3 62 8F 34 6A 2B               DB      0E3H,062H,08FH,034H,06AH,02BH,086H,0D2H,0C4H
      86 D2 C4
0D88  93 11 20 47 2D 09               DB      093H,011H,020H,047H,02DH,09H,018H,0A0H,079H
      18 A0 79
0D91  8B 33 35 6F 61 E5               DB      08BH,033H,035H,06FH,061H,0E5H,061H,0CEH,05AH
      61 CE 5A
0D9A  BC 87 6F C5 39 6B               DB      0BCH,087H,06FH,0C5H,039H,06BH,02DH,0E6H,0B6H
      2D E6 B6
0DA3  39 E5 AC 25 6B F8               DB      039H,0E5H,0ACH,025H,06BH,0F8H,0E6H,0E4H,0B3H
      E6 E4 B3
0DAC  D6 A4 69 D9 73 CE               DB      0D6H,0A4H,069H,0D9H,073H,0CEH,0DAH,0BCH,085H
      DA BC 85
0DB5  47 DD 3E 7B 73 69               DB      047H,0DDH,03EH,07BH,073H,069H,0EAH,0FEH,064H
      EA FE 64
0DBE  ED D5 A9 A9 D5 99               DB      0EDH,0D5H,0A9H,0A9H,0D5H,099H,072H,094H,020H
      72 94 20
0DC7  26 DA 57 CA 51 8D               DB      026H,0DAH,057H,0CAH,051H,08DH,0AAH,072H,02EH
      AA 72 2E
0DD0  0E D7 50 4A A2 EA               DB      0EH,0D7H,050H,04AH,0A2H,0EAH,0C6H,00H,053H
      C6 00 53
0DD9  B9 6A 80 01 8D BB               DB      0B9H,06AH,080H,01H,08DH,0BBH,022H,020H,05H
      22 20 05
0DE2  C1 36 B5 64 EE 91               DB      0C1H,036H,0B5H,064H,0EEH,091H,089H,0CFH,0D4H
      89 CF D4
0DEB  82 B9 77 CF 39 63               DB      082H,0B9H,077H,0CFH,039H,063H,08BH,066H,0DEH
      8B 66 DE
0DF4  3D FB F4 AD AA 5A               DB      03DH,0FBH,0F4H,0ADH,0AAH,05AH,0E7H,0ECH,0D0H
      E7 EC D0
0DFD  37 EC 12 1B 12 51               DB      037H,0ECH,012H,01BH,012H,051H,01FH,05CH,0BAH
      1F 5C BA
0E06  AB D8 06 01 D4 26               DB      0ABH,0D8H,06H,01H,0D4H,026H,011H,080H,0A6H
      11 80 A6
0E0F  5C 0D F0 6C BB 01               DB      05CH,0DH,0F0H,06CH,0BBH,01H,09EH,049H,075H
      9E 49 75
0E18  C0 33 AE 06 78 7C               DB      0C0H,033H,0AEH,06H,078H,07CH,0CH,01H,09BH
      0C 01 9B
0E21  8F 21 20 31 33 0C               DB      08FH,021H,020H,031H,033H,0CH,060H,060H,064H
      60 60 64
0E2A  2D 68 BA 56 9C D5               DB      02DH,068H,0BAH,056H,09CH,0D5H,046H,0D7H,013H
      46 D7 13
0E33  98 8D D7 1C DB A4               DB      098H,08DH,0D7H,01CH,0DBH,0A4H,06AH,0CFH,019H
      6A CF 19
0E3C  7B 2C 98 2E F9 E8               DB      07BH,02CH,09BH,02EH,0F9H,0E8H,0ECH,025H,076H
      EC 25 76
0E45  85 7A 97 73 56 97               DB      085H,07AH,097H,073H,056H,097H,0E9H,0A9H,05BH
      E9 A9 5B
0E4E  CE 55 BD BB 97 6F               DB      0CEH,055H,0BDH,0BBH,097H,06FH,05EH,057H,02BH
      5E 57 2B
0E57  6A 5A BE 64 5C 35               DB      06AH,05AH,0BEH,064H,05CH,035H,02AH,0E7H,054H
      2A E7 54
```

```
0E60  E7 71 17 CF 96 1B                    DB      0E7H,071H,017H,0CFH,096H,01BH,0ABH,0DBH,05DH
      AB DB 5D
0E69  22 79 AC B7 1A 57                    DB      022H,079H,0ACH,0B7H,01AH,057H,0A9H,0A4H,095H
      A9 A4 95
0E72  BE A6 5D 29 53 74                    DB      0BEH,0A6H,05DH,029H,053H,074H,0C7H,0E6H,076H
      C7 E6 76
0E7B  27 C7 11 9B 89 DB                    DB      027H,0C7H,011H,09BH,089H,0DBH,093H,0A5H,0A5H
      93 A5 A5
0E84  4C CE 11 8F 91 9A                    DB      04CH,0CEH,011H,08FH,091H,09AH,063H,095H,080H
      63 95 80
0E8D  3D 42 78 76 49 02                    DB      03DH,042H,078H,076H,049H,02H,0F6H,02BH,05AH
      F6 2B 5A
0E96  31 11 89 CA DF AD                    DB      031H,011H,089H,0CAH,0DFH,0ADH,0BBH,079H,028H
      BB 79 28
0E9F  2A 7F 76 6A DE 75                    DB      02AH,07FH,076H,06AH,0DEH,075H,031H,03DH,039H
      31 3D 39
0EA8  88 E4 54 AC FF 07                    DB      088H,0E4H,054H,0ACH,0FFH,07H

= 0EAE                              A0020  EQU     $       ; LAUGHING - 2.5 SECONDS
0EAE  0C 68 BA 52 00 2D                    DB      0CH,068H,0BAH,052H,00H,02DH,079H,0AH,0A0H
      79 0A A0
0EB7  C4 AC A6 14 C9 E6                    DB      0C4H,0ACH,0A6H,014H,0C9H,0E6H,0E9H,0C9H,086H
      E9 C9 86
0EC0  10 85 79 96 25 2B                    DB      010H,085H,079H,096H,025H,02BH,052H,052H,0ECH
      52 52 EC
```

Segments and groups:

N a m e	Size	align	combine class
ABS0	7C00	AT	0000
DATA	0088	AT	0040
DKDATA	0428	AT	0060
DUMMY	0248	AT	0000
STACK	0100	AT	0030
TKRSEG	7FFF	PARA	NONE
VIDEO_RAM	4000	AT	B800
XXDATA	002A	AT	0050

Symbols:

N a m e	Type	Value	Attr	
A.	Number	07F7	TKRSEG	
A0006.	Number	0C90	TKRSEG	
A0019.	Number	0D01	TKRSEG	
A0020.	Number	0EAE	TKRSEG	
A0022.	Number	10DC	TKRSEG	
A0023.	Number	116C	TKRSEG	
A0024.	Number	11D7	TKRSEG	
A0025.	Number	1219	TKRSEG	
A0026.	Number	12A8	TKRSEG	
A0027.	Number	130F	TKRSEG	
A0028.	Number	137D	TKRSEG	
A0033.	Number	13F8	TKRSEG	
A0034.	Number	14A9	TKRSEG	
ACL_ERROR.	Number	0003		
ACL_OFF.	Number	0001		
ACTIVE_PAGE.	L BYTE	0062	DATA	
ADD.	L NEAR	01F2	TKRSEG	
ADDR_6845.	L WORD	0063	DATA	
ALT_INPUT.	L BYTE	0019	DATA	
ALT_KEY.	Number	0038		
ALT_SHIFT.	Number	0008		
ARROW.	L BYTE	07F3	TKRSEG	
ARROW_POS1	Number	890D		
ARROW_POS2	Number	890D		
ATTACH_ACL	N PROC	0357	TKRSEG	Length =0023
AUDIO8254.	Number	0002		
AUDIO_CHN.	Number	0040		
B0001.	Number	1505	TKRSEG	
B0002.	Number	1588	TKRSEG	
B0003.	Number	15F5	TKRSEG	
B0004.	Number	1643	TKRSEG	
B0005.	Number	16B9	TKRSEG	
B0006.	Number	1705	TKRSEG	
B0007.	Number	1781	TKRSEG	
B0008.	Number	17C9	TKRSEG	
B0009.	Number	1849	TKRSEG	
B0010.	Number	1888	TKRSEG	
B0011.	Number	18F7	TKRSEG	
B0012.	Number	192D	TKRSEG	
B0013.	Number	1999	TKRSEG	
B0014.	Number	19CB	TKRSEG	
B0015.	Number	1A27	TKRSEG	
B0016.	Number	1A5B	TKRSEG	
B0017.	Number	1AC7	TKRSEG	
B0018.	Number	1B07	TKRSEG	
B0019.	Number	1B4B	TKRSEG	
B0020.	Number	1BB5	TKRSEG	
B0021.	Number	1C27	TKRSEG	
B0022.	Number	1CA1	TKRSEG	
B0023.	Number	1D17	TKRSEG	
B0024.	Number	1DA1	TKRSEG	
B0025.	Number	1E2D	TKRSEG	
B0026.	Number	1EC7	TKRSEG	
B0027.	Number	1EF5	TKRSEG	
B0028.	Number	1F83	TKRSEG	
B0029.	Number	2090	TKRSEG	
B0030.	Number	2D20	TKRSEG	
B0031.	Number	2D50	TKRSEG	
B0032.	Number	2DAA	TKRSEG	
B0033.	Number	2E1C	TKRSEG	
B0034.	Number	2E82	TKRSEG	
B0035.	Number	2F1A	TKRSEG	
B0036.	Number	2F70	TKRSEG	
B0037.	Number	3000	TKRSEG	
B0038.	Number	3034	TKRSEG	
B0039.	Number	3062	TKRSEG	
B0040.	Number	3008	TKRSEG	
B0041.	Number	3116	TKRSEG	
B0042.	Number	317C	TKRSEG	
B0043.	Number	31F0	TKRSEG	
B0044.	Number	322A	TKRSEG	
B0045.	Number	32D0	TKRSEG	
B0046.	Number	3352	TKRSEG	
B0047.	Number	33CA	TKRSEG	
B0048.	Number	341A	TKRSEG	

```
B0049. . . . . Number  3488   TKRSEG
B0050. . . . . Number  34E0   TKRSEG
B0051. . . . . Number  350E   TKRSEG
B0052. . . . . Number  3584   TKRSEG
B0053. . . . . Number  3604   TKRSEG
B0054. . . . . Number  3670   TKRSEG
B0055. . . . . Number  370C   TKRSEG
B0056. . . . . Number  3766   TKRSEG
BAD_ADDR_MARK. . . . . Number  0002
BAD_CMD. . . . . Number  0001
BAD_CRC. . . . . Number  0010
BAD_DMA. . . . . Number  0008
BAD_NEC. . . . . Number  0020
BAD_SEEK . . . . . Number  0040


BANK_TEST_START. . . . . L NEAR  01A6   TKRSEG
BASIC_PTR. . . . . L WORD  0060   ABS0
BC . . . . . F PROC  01A6   TKRSEG  Length =006B
BCD. . . . . Number  0001
BCLEN. . . . . Number  00B1
BCLOC. . . . . L WORD  002B   TKRSEG
BC_START . . . . . L NEAR  0172   TKRSEG
BEEP . . . . . N PROC  0298   TKRSEG  Length =0022
BFR_EMPTY. . . . . Number  0020
BFR_LOW. . . . . Number  0040
BFR_PTR. . . . . L WORD  013C   DUMMY
BINARY . . . . . Number  0000
BIOS_BREAK . . . . . L BYTE  0071   DATA
BITS_ON_OFF. . . . . N PROC  012C   TKRSEG  Length =0046
BOOT_LOCN. . . . . L FAR  7C00   ABS0
BSE. . . . . L NEAR  0204   TKRSEG
BSUE . . . . . L NEAR  0208   TKRSEG
BTD. . . . . L NEAR  0203   TKRSEG
BIL. . . . . L NEAR  01DA   TKRSEG
BUFFER_END . . . . . L WORD  0082   DATA
BUFFER_HEAD. . . . . L WORD  001A   DATA
BUFFER_START . . . . . L WORD  0080   DATA
BUFFER_TAIL. . . . . L WORD  001C   DATA
BUSY_BIT . . . . . Number  0020
C0001. . . . . Number  37FE   TKRSEG
C0002. . . . . Number  3844   TKRSEG
C0003. . . . . Number  38AA   TKRSEG
C0004. . . . . Number  38FB   TKRSEG
C0005. . . . . Number  3953   TKRSEG
C0006. . . . . Number  39A2   TKRSEG
C0007. . . . . Number  39F3   TKRSEG
C0008. . . . . Number  3A57   TKRSEG
C0009. . . . . Number  3AD2   TKRSEG
C0010. . . . . Number  3B0F   TKRSEG
C0011. . . . . Number  3B49   TKRSEG
C0012. . . . . Number  3B8B   TKRSEG
C0013. . . . . Number  3BF9   TKRSEG
C0014. . . . . Number  3C4B   TKRSEG
C0015. . . . . Number  3C9E   TKRSEG
C0016. . . . . Number  3CF1   TKRSEG
C0017. . . . . Number  3D4C   TKRSEG
C0018. . . . . Number  3DBE   TKRSEG
C0019. . . . . Number  3E25   TKRSEG
C0020. . . . . Number  3E4C   TKRSEG
C0021. . . . . Number  3EBA   TKRSEG

C0022. . . . . Number  3F12   TKRSEG
C0023. . . . . Number  3F98   TKRSEG
C0024. . . . . Number  4C90   TKRSEG
C0025. . . . . Number  4D15   TKRSEG
C0026. . . . . Number  4D6F   TKRSEG
C0027. . . . . Number  4DD6   TKRSEG
C0028. . . . . Number  4E18   TKRSEG
C0029. . . . . Number  4E80   TKRSEG


C0030. . . . . Number  4ECA   TKRSEG
C0031. . . . . Number  4F0E   TKRSEG
C0032. . . . . Number  4F9C   TKRSEG
C0033. . . . . Number  500D   TKRSEG
C0034. . . . . Number  5079   TKRSEG
C0035. . . . . Number  50DA   TKRSEG
C0036. . . . . Number  5118   TKRSEG
C0037. . . . . Number  517E   TKRSEG
C0038. . . . . Number  51D7   TKRSEG
C0039. . . . . Number  523D   TKRSEG
C0040. . . . . Number  52A5   TKRSEG
C0041. . . . . Number  5332   TKRSEG
C0042. . . . . Number  53DE   TKRSEG
C0043. . . . . Number  5421   TKRSEG
C0044. . . . . Number  54D3   TKRSEG
C0045. . . . . Number  551E   TKRSEG
C0046. . . . . Number  5567   TKRSEG
C0047. . . . . Number  55A7   TKRSEG
C0048. . . . . Number  5629   TKRSEG
C0049. . . . . Number  5689   TKRSEG
C0050. . . . . Number  56E9   TKRSEG
C0051. . . . . Number  575D   TKRSEG
C0052. . . . . Number  57A8   TKRSEG
C0053. . . . . Number  57FF   TKRSEG
C0054. . . . . Number  5821   TKRSEG
C0055. . . . . Number  589B   TKRSEG
C0056. . . . . Number  58F1   TKRSEG
C0057. . . . . Number  5969   TKRSEG
C0058. . . . . Number  59D4   TKRSEG
C0059. . . . . Number  5A20   TKRSEG
CAPS_KEY . . . . . Number  003A
CAPS_SHIFT . . . . . Number  0040
CAPS_STATE . . . . . Number  0040
CARD_RESET_ER. . . . . L NEAR  0110   TKRSEG
CHECK_COUNTER_2. . . . . L NEAR  00AA   TKRSEG
CHKTKR_ACL . . . . . N PROC  036E   TKRSEG  Length =000C
CHK_END. . . . . L NEAR  016A   TKRSEG
CHN_OFF. . . . . Number  0000
CHN_ON . . . . . Number  0001
CLICK_ON . . . . . Number  0004
CLICK_SEQUENCE . . . . . Number  0002
CLOCK_WAIT . . . . . N PROC  0A2D   TKRSEG  Length =0014
CLR_MUX. . . . . Number  00FC
CLR_MXPG . . . . . Number  00F0
CLR_PAGE . . . . . Number  00F3
CLR_SPKSW. . . . . Number  009F
CMD_PORT . . . . . Number  0063
CNTT_TEST_BITS . . . . . L NEAR  0154   TKRSEG
COMPLETE_OUTPUT. . . . . L NEAR  068F   TKRSEG
CONTINUE_CVSD_TEST . . . . . L NEAR  0891   TKRSEG
CONTINUE_OUTPUT. . . . . L NEAR  067E   TKRSEG
```

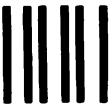
```
COUNTER_CK . . . . . L NEAR  0097  TKRSEG
CPUREG . . . . . Number  0038


CRC_REG. . . . . L WORD  0069  DATA
CRTREG . . . . . Number  0007
CRT_COLS . . . . . L WORD  004A  DATA
CRT_LEN. . . . . L WORD  004C  DATA
CRT_MODE . . . . . L BYTE  0049  DATA
CRT_MODE_SET . . . . . L BYTE  0065  DATA
CRT_PALLETTE . . . . . L BYTE  0066  DATA
CRT_START. . . . . L WORD  004E  DATA
CSET_PTR . . . . . L DWORD 0110  ABS0
CTL_KEY. . . . . Number  001D
CTL_SHIFT. . . . . Number  0004
CTR0 . . . . . Number  0000
CTR1 . . . . . Number  0040
CTR2 . . . . . Number  0080
CTR_LATCH. . . . . Number  0000
CI_EP. . . . . L NEAR  0102  TKRSEG
CURSOR_MODE. . . . . L WORD  0060  DATA
CURSOR_POSN. . . . . L WORD  0050  DATA  Length =0008
CUR_CHAR . . . . . L BYTE  0085  DATA
CUR_FUNC . . . . . L BYTE  0087  DATA
CUST_ER. . . . . Number  004A
CUST_OUT . . . . . L NEAR  0280  TKRSEG
CVSD . . . . . Number  0001
CVSD00 . . . . . L NEAR  0388  TKRSEG
CVSD20 . . . . . L NEAR  03A7  TKRSEG
CVSD25 . . . . . L NEAR  03B5  TKRSEG
CVSD30 . . . . . L NEAR  03CA  TKRSEG
CVSD40 . . . . . L NEAR  03D1  TKRSEG
CVSD45 . . . . . L NEAR  03DB  TKRSEG
CVSD50 . . . . . L NEAR  03E4  TKRSEG
CVSDR. . . . . Number  0000
CVSDR_TBL. . . . . Number  0000
CVSDR_USER . . . . . Number  0002
CVSDW. . . . . Number  00FF
CVSDW_TBL. . . . . Number  0001
CVSDW_USER . . . . . Number  0003
CVSDX. . . . . L NEAR  0497  TKRSEG
CVSDX0 . . . . . L NEAR  04A8  TKRSEG
CVSDXA . . . . . L NEAR  0495  TKRSEG
CVSD_CLK . . . . . Number  FB9C
CVSD_FN. . . . . Number  0001
CVSD_FRAME . . . . . Number  FB9D
CVSD_REC . . . . . L NEAR  08D3  TKRSEG
CVSD_TEST. . . . . L NEAR  0883  TKRSEG
CWREG. . . . . Number  FB9B
CWR_8254 . . . . . Number  FB9F
D0001. . . . . Number  5A87  TKRSEG
D0002. . . . . Number  5AAA  TKRSEG
D0003. . . . . Number  5B26  TKRSEG
D0004. . . . . Number  5B5E  TKRSEG
D0005. . . . . Number  5B9F  TKRSEG
D0006. . . . . Number  5BF7  TKRSEG
D0007. . . . . Number  5C26  TKRSEG
D0008. . . . . Number  5C4E  TKRSEG
D0009. . . . . Number  5C93  TKRSEG
D0010. . . . . Number  5D15  TKRSEG
D0011. . . . . Number  5DBD  TKRSEG
D0012. . . . . Number  5E23  TKRSEG
D0013. . . . . Number  5EB2  TKRSEG
D0014. . . . . Number  5F20  TKRSEG
D0015. . . . . Number  5F8B  TKRSEG
D0016. . . . . Number  6C90  TKRSEG
D0017. . . . . Number  6D08  TKRSEG
D0018. . . . . Number  6D6B  TKRSEG
D0019. . . . . Number  6DAE  TKRSEG
D0020. . . . . Number  6E27  TKRSEG
D0021. . . . . Number  6E73  TKRSEG
D0022. . . . . Number  6EBE  TKRSEG
D0023. . . . . Number  6F04  TKRSEG
D0024. . . . . Number  6F6C  TKRSEG
D0025. . . . . Number  6FC7  TKRSEG
D0026. . . . . Number  7010  TKRSEG
D0027. . . . . Number  707B  TKRSEG
D0028. . . . . Number  70C2  TKRSEG
D0029. . . . . Number  7114  TKRSEG
D0030. . . . . Number  7193  TKRSEG
D0031. . . . . Number  71F9  TKRSEG
D0032. . . . . Number  7263  TKRSEG
D0033. . . . . Number  72BD  TKRSEG
D0034. . . . . Number  7316  TKRSEG
D0035. . . . . Number  737E  TKRSEG
D0036. . . . . Number  73E7  TKRSEG
D0037. . . . . Number  744D  TKRSEG
D0038. . . . . Number  74BF  TKRSEG
D0039. . . . . Number  74F6  TKRSEG
D0040. . . . . Number  7576  TKRSEG
D0041. . . . . Number  75EE  TKRSEG
D0042. . . . . Number  7663  TKRSEG
D0043. . . . . Number  766A  TKRSEG
D0044. . . . . Number  7763  TKRSEG
D0045. . . . . Number  77CF  TKRSEG
D0046. . . . . Number  780F  TKRSEG
D0047. . . . . Number  786E  TKRSEG
D0048. . . . . Number  7879  TKRSEG
D0049. . . . . Number  78FB  TKRSEG
D0050. . . . . Number  7980  TKRSEG
D0051. . . . . Number  79EF  TKRSEG
D0052. . . . . Number  7A5B  TKRSEG
D0053. . . . . Number  7ABE  TKRSEG
D0054. . . . . Number  7B38  TKRSEG
D0055. . . . . Number  7BCF  TKRSEG
D0056. . . . . Number  7C5A  TKRSEG
D0057. . . . . Number  7C9E  TKRSEG
D0058. . . . . Number  7D24  TKRSEG
D0059. . . . . Number  7D4F  TKRSEG
DATA_AREA. . . . . L BYTE  0400  ABS0


DATA_WORD. . . . . L WORD  0400  ABS0
DCP_MENU_PAGE. . . . . L BYTE  0001  XXDATA
DCP_ROW_COL. . . . . L WORD  0002  XXDATA
DCP_RUNNING. . . . . L BYTE  0029  XXDATA
DECODE . . . . . Number  000C
DEL10. . . . . L NEAR  0976  TKRSEG
```

```
DEL20. . . . . L NEAR  097A   TKRSEG
DELAY. . . . . N PROC  0976   TKRSEG  Length =000A
DELAY_RATE . . . . . Number  000F
DEL_KEY. . . . . Number  0053
DIAG_RESET_ERR . . . . . L NEAR  0888   TKRSEG
DIAG_RETRY . . . . . L BYTE  0020   DKDATA
DIAG_TABLE_PTR . . . . . Number  4000
DIO. . . . . Number  0040
DISKETTE_STATUS. . . . . L BYTE  0041   DATA
DISK_POINTER . . . . . L DWORD 0078   ABS0
DK_BUF_LEN . . . . . Number  0200
DK_INDEX . . . . . L BYTE  007B   DATA
DMA_BOUNDARY . . . . . Number  0009
DNEC_STATUS. . . . . L BYTE  0421   DKDATA  Length =0007
DONE128. . . . . L BYTE  0011   XXDATA
DRIVE_ENABLE . . . . . Number  0001
E0001. . . . . Number  7D97   TKRSEG
E0002. . . . . Number  7E02   TKRSEG
E0003. . . . . Number  7E31   TKRSEG
E000N. . . . . Number  7E59   TKRSEG
E0006. . . . . Number  7EDD   TKRSEG
E0007. . . . . Number  7F16   TKRSEG
E0008. . . . . Number  7F37   TKRSEG
E0010. . . . . Number  7F80   TKRSEG
E0011. . . . . Number  7F90   TKRSEG
E0012. . . . . Number  7FA0   TKRSEG
EB . . . . . L NEAR  0267   TKRSEG
EBO. . . . . L NEAR  026C   TKRSEG
EDGE_CNT . . . . . L WORD  0067   DATA
EMPTR. . . . . L WORD  0214   ABS0
EM_0 . . . . . L NEAR  0235   TKRSEG
EM_1 . . . . . L NEAR  0240   TKRSEG
ENABLE . . . . . N PROC  0A52   TKRSEG  Length =0022
ENCODE . . . . . Number  000D
END_PG0. . . . . Number  1FFB   TKRSEG
END_PG1. . . . . Number  3FFE   TKRSEG
END_PG2. . . . . Number  5FD9   TKRSEG
END_PG3. . . . . Number  7FE9   TKRSEG
EOI. . . . . Number  0020
EQUIP_FLAG . . . . . L WORD  0010   DATA
ERROR. . . . . L NEAR  02FF   TKRSEG
ERROR_0. . . . . L NEAR  0376   TKRSEG
ERROR_ERR. . . . . L BYTE  0293   TKRSEG
ER_CODE8254. . . . . Number  0002
ER_CODE8255. . . . . Number  0001
ER_CVSD_C1 . . . . . Number  0044
ER_CVSD_C2 . . . . . Number  0045

ER_CVSD_S1 . . . . . Number  0012
ER_CVSD_S2 . . . . . Number  0013
ER_CVSD_S3 . . . . . Number  0014
ER_LPC_C1. . . . . Number  0042
ER_LPC_C2. . . . . Number  0043
ER_LPC_S1. . . . . Number  0010
ER_LPC_S2. . . . . Number  0011
EX . . . . . L NEAR  020A   TKRSEG
EXIT . . . . . L NEAR  0301   TKRSEG
EXIT_BFR_HNDLR . . . . . L NEAR  069E   TKRSEG
EXIT_POST. . . . . L NEAR  011C   TKRSEG
EXST . . . . . L WORD  0124   ABS0
EXT_PTR. . . . . L DWORD 007C   ABS0
E_MSG_B. . . . . F PROC  0211   TKRSEG  Length =0082
F0 . . . . . L NEAR  02ED   TKRSEG
F00. . . . . L NEAR  02E5   TKRSEG
F1 . . . . . L NEAR  02F5   TKRSEG
FDC_RESET. . . . . Number  0080
FDND_ERR . . . . . L NEAR  0630   TKRSEG
FLAC_SETUP . . . . . L NEAR  0380   TKRSEG
FLAG_SETUP2. . . . . L NEAR  06E0   TKRSEG
FN_BREAK . . . . . Number  0040
FN_FLAG. . . . . Number  0080
FN_LOCK. . . . . Number  0010
FN_PENDING . . . . . Number  0020
FOREGROUND_COMPLETE. . . . . L NEAR  063A   TKRSEG
FOR_COMP . . . . . L NEAR  063E   TKRSEG
FRAME_01 . . . . . N PROC  0411   TKRSEG  Length =000D
FRAME_10 . . . . . N PROC  041E   TKRSEG  Length =000D
FRAME_HI . . . . . Number  0004
FSYNC. . . . . L NEAR  046A   TKRSEG
G7 . . . . . L NEAR  02AF   TKRSEG
GOODBYE. . . . . L NEAR  0835   TKRSEG
GOOD_TEST_END. . . . . L NEAR  0878   TKRSEG
HALF_RATE. . . . . Number  0004
HOLD_STATE . . . . . Number  0008
HORZ_POS . . . . . L BYTE  0089   DATA
HTL. . . . . L NEAR  00F1   TKRSEG
INDEX_ERR. . . . . Number  0004
INIT . . . . . F PROC  0003   TKRSEG  Length =02D1
INIT1. . . . . L NEAR  002F   TKRSEG
INIT_DELAY . . . . . Number  0002
INIT_TIMER . . . . . N PROC  011D   TKRSEG  Length =000F
INNER_LOOP . . . . . L NEAR  0139   TKRSEG
INS_KEY. . . . . Number  0052
INS_SHIFT. . . . . Number  0080
INT1C0 . . . . . L WORD  000C   XXDATA
INT1CS . . . . . L WORD  000E   XXDATA
INT1C_PTR. . . . . L WORD  0070   ABS0
INT1_EOI . . . . . Number  0061
INT1_OFF . . . . . Number  0002
INT1_ON. . . . . Number  00FD
INT3_PTR . . . . . L WORD  000C   ABS0

INT5_PTR . . . . . L WORD  0014   ABS0
INTA00 . . . . . Number  0020
INTA01 . . . . . Number  0021
INTR_CTR . . . . . Number  FB9E
INTR_FLAG. . . . . L BYTE  0084   DATA
INT_PTR. . . . . L DWORD 0020   ABS0
IO_ROM_INIT. . . . . L WORD  0014   XXDATA
IO_ROM_SEG . . . . . L WORD  0016   XXDATA
J16_2. . . . . L NEAR  09D5   TKRSEG
J16_4. . . . . L NEAR  09E0   TKRSEG
J16_5. . . . . L NEAR  0A00   TKRSEG
J16_6. . . . . L NEAR  0A0C   TKRSEG
J16_61 . . . . . L NEAR  0A13   TKRSEG
J16_7. . . . . L NEAR  0A2B   TKRSEG
KBD. . . . . Number  004E
KBDONE . . . . . L WORD  0012   XXDATA
KBD_ERR. . . . . L BYTE  0012   DATA
KBD_PTR. . . . . L WORD  0138   DUMMY
```

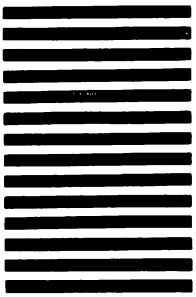
```
KBPORT . . . . . Number  0060
KB_BUFFER. . . . . L WORD  001E   DATA     Length =0010
KB_CTL . . . . . Number  0061
KB_FLAG. . . . . L BYTE  0017   DATA
KB_FLAG_1. . . . . L BYTE  0018   DATA
KB_FLAG_2. . . . . L BYTE  0088   DATA
KB_NOISE . . . . . N PROC  0A74   TKRSEG   Length =0021
KEY62_PTR. . . . . L WORD  0120   ABS0
KEYBRD_PTR . . . . . L DWORD 011C   ABS0
LAST_VAL . . . . . L BYTE  006B   DATA
LD . . . . . L NEAR  0105   TKRSEG
LEFT_KEY . . . . . Number  002A
LEFT_SHIFT . . . . . Number  0002
LOADDO . . . . . L NEAR  069F   TKRSEG
LOADXX . . . . . L NEAR  06A9   TKRSEG
LOAD_BFR . . . . . N PROC  069F   TKRSEG   Length =000B
LOAD_BFR_ERR . . . . . L NEAR  069D   TKRSEG
LOAD_BFR_HNDLR . . . . . N PROC  0675   TKRSEG   Length =002A
LOAD_LOOP. . . . . L NEAR  08B2   TKRSEG
LOCATE . . . . . Number  0081
LOOPD1 . . . . . L NEAR  0A7B   TKRSEG
LPC. . . . . Number  0000
LPC00. . . . . L NEAR  04AE   TKRSEG
LPC000 . . . . . L NEAR  04AD   TKRSEG
LPC02A . . . . . L NEAR  04D4   TKRSEG
LPC03. . . . . L NEAR  04D9   TKRSEG
LPC04. . . . . L NEAR  04F8   TKRSEG
LPC05. . . . . L NEAR  0510   TKRSEG
LPC10. . . . . L NEAR  0532   TKRSEG
LPC20. . . . . L NEAR  0540   TKRSEG
LPC22. . . . . L NEAR  056C   TKRSEG
LPC23. . . . . L NEAR  0571   TKRSEG
LPC25. . . . . L NEAR  058C   TKRSEG
LPC30. . . . . L NEAR  058E   TKRSEG
LPC33. . . . . L NEAR  0595   TKRSEG

LPC33_LINK . . . . . L NEAR  0631   TKRSEG
LPC34. . . . . L NEAR  05C3   TKRSEG
LPC34A . . . . . L NEAR  05CF   TKRSEG
LPC35. . . . . L NEAR  05E0   TKRSEG
LPC40. . . . . L NEAR  0600   TKRSEG
LPC45. . . . . L NEAR  0603   TKRSEG
LPCRDY_ERR . . . . . Number  0006
LPCR_OFF . . . . . Number  0008
LPCR_ON. . . . . Number  0009
LPCW_10. . . . . N PROC  065D   TKRSEG   Length =0018
LPCW_OFF . . . . . Number  000A
LPCW_ON. . . . . Number  000B
LPCW_X . . . . . L NEAR  0674   TKRSEG
LPCX . . . . . L NEAR  064A   TKRSEG
LPCX00 . . . . . L NEAR  06E5   TKRSEG
LPCX30 . . . . . L NEAR  0702   TKRSEG
LPCX90 . . . . . L NEAR  071F   TKRSEG
LPCX95 . . . . . L NEAR  0748   TKRSEG
LPC_BACKGROUND . . . . . L NEAR  0648   TKRSEG
LPC_BUFFER-. . . . . Number  0002
LPC_BUF_NOT_EMPTY. . . . . L NEAR  037D   TKRSEG
LPC_CHIP_FATL. . . . . L NEAR  0387   TKRSEG
LPC_ERR. . . . . L NEAR  085C   TKRSEG
LPC_ERR_EXIT . . . . . L NEAR  059A   TKRSEG
LPC_FN . . . . . Number  0002
LPC_FN_FORE. . . . . Number  0003
LPC_IN . . . . . Number  0083
LPC_INO. . . . . Number  0080
LPC_IN1. . . . . Number  0003
LPC_INDEX. . . . . Number  0001
LPC_INPROG . . . . . Number  0002
LPC_INT. . . . . Number  0002
LPC_OUT. . . . . Number  0081
LPC_OUT0 . . . . . Number  0080
LPC_OUT1 . . . . . Number  0001
LPC_PTR. . . . . L WORD  0024   DUMMY
LPC_RDY_ERR. . . . . L NEAR  0335   TKRSEG
LPC_READY. . . . . Number  0001
LPC_SPEAK. . . . . L NEAR  083B   TKRSEG
LPC_STATUS . . . . . Number  0000
LPC_TEST . . . . . L NEAR  0836   TKRSEG
LPC_XX . . . . . L NEAR  06D2   TKRSEG
LTH. . . . . L NEAR  00E8   TKRSEG
MASK_NMI . . . . . L NEAR  04C2   TKRSEG
MATCH_BIT. . . . . Number  0080
MD0. . . . . Number  0000
MD1. . . . . Number  0002
MD2. . . . . Number  0004
MD3. . . . . Number  0006
MD4. . . . . Number  0008
MD5. . . . . Number  000A
MEMORY_SIZE. . . . . L WORD  0013   DATA
MEM_DONE0. . . . . L WORD  000A   XXDATA

MEM_DONES. . . . . L WORD  0008   XXDATA
MEM_TOT. . . . . L WORD  0006   XXDATA
MENU_UP. . . . . L BYTE  0010   XXDATA
MFG_RTN. . . . . L WORD  0022   XXDATA
MFG_TST. . . . . L BYTE  0005   XXDATA
MIC_I. . . . . Number  07F2   TKRSEG
MIC_ICON . . . . . L BYTE  07CF   TKRSEG
MIC_POS. . . . . Number  0812
MINI . . . . . Number  2000
MODE0_A. . . . . Number  0000
MODE0_B. . . . . Number  0000
MODE1_A. . . . . Number  0020
MODE1_B. . . . . Number  0004
MODE2_A. . . . . Number  00A0
MODEM_BUFFER . . . . . L BYTE  0019   XXDATA   Length =0009
MODE_8255. . . . . Number  0089
MODE_SET . . . . . Number  0080
MOTOR_COUNT. . . . . L BYTE  0040   DATA
MOTOR_STATUS . . . . . L BYTE  003F   DATA
MOTOR_WAIT . . . . . Number  0025
NEC_CTL. . . . . Number  00F2
NEC_DATA . . . . . Number  00F5
NEC_STAT . . . . . Number  00F4
NEC_STATUS . . . . . L BYTE  0042   DATA     Length =0007
NL62 . . . . . Number  0020
NMIOFF . . . . . N PROC  0980   TKRSEG   Length =0025
NMION. . . . . N PROC  09A5   TKRSEG   Length =0088
NMI_PORT . . . . . Number  00A0
NMI_PTR. . . . . L WORD  0008   ABS0
NODTAG!. . . . . Number  0012
```

```
NO_POINTER_SAVE. . . . . L NEAR  0601   TKRSEG
NUM_KEY. . . . . Number  0045
NUM_SHIFT. . . . . Number  0020
NUM_STATE. . . . . Number  0020
NXT_ACL. . . . . L NEAR  035F   TKRSEG
OFF1 . . . . . Number  136F
OFF2 . . . . . Number  336F
OFF3 . . . . . Number  536F
OK . . . . . Number  0000
OLDKBD_PTR . . . . . L WORD  0024   DUMMY
OUTER_LOOP . . . . . L NEAR  0136   TKRSEG
PA . . . . . L NEAR  0175   TKRSEG
PAGDAT . . . . . L BYTE  008A   DATA
PAGE0. . . . . Number  0000
PAGE1. . . . . Number  0004
PAGE2. . . . . Number  0008
PAGE3. . . . . Number  000C
PAGREG . . . . . Number  03DF
PARM0. . . . . Number  00AF
PARM1. . . . . Number  0003
PARM10 . . . . . Number  0004
PARM9. . . . . Number  0019
PARM_PTR . . . . . L DWORD 0074   ABSO

PA_E . . . . . L NEAR  0184   TKRSEG
PC . . . . . L NEAR  0191   TKRSEG
PG0_MAX. . . . . Number  0029
PG1_MAX. . . . . Number  005D
PG2_MAX. . . . . Number  0091
PG3_MAX. . . . . Number  00C8
PLAYBACK_OK. . . . . L NEAR  095E   TKRSEG
PORTA. . . . . Number  FB98
PORTA_IN . . . . . Number  0010
PORTA_OUT. . . . . Number  0000
PORTB. . . . . Number  FB99
PORTB_IN . . . . . Number  0002
PORTB_OUT. . . . . Number  0000
PORTC. . . . . Number  FB9A
PORTCL_IN. . . . . Number  0001
PORTCL_OUT . . . . . Number  0000
PORTCU_IN. . . . . Number  0008
PORTCU_OUT . . . . . Number  0000
PORTC_TST. . . . . N PROC  0186   TKRSEG  Length =0020
PORT_20H . . . . . Number  0020
PORT_21H . . . . . Number  0021
PORT_61H . . . . . Number  0061
PORT_A . . . . . Number  0060
PORT_B . . . . . Number  0061
PORT_B0. . . . . Number  00B0
PORT_C . . . . . Number  0062
PORT_TST . . . . . N PROC  0172   TKRSEG  Length =0014
POST . . . . . F PROC  0068   TKRSEG  Length =00B5
POST_ERR . . . . . L BYTE  0018   XXDATA
PRINT. . . . . Number  0082
PRINTER_BASE . . . . . L WORD  0008   DATA    Length =0004
PRINT_TIM_OUT. . . . . L BYTE  0078   DATA    Length =0004
PRT_HEX. . . . . N PROC  02CB   TKRSEG  Length =0009
PUTCHAR. . . . . Number  0001
Q_TIM_OUT. . . . . L NEAR  047D   TKRSEG
RANGE. . . . . Number  0004
RD_BACK. . . . . Number  00C0
READ . . . . . L NEAR  044F   TKRSEG
READ_BUF . . . . . L BYTE  0021   DKDATA  Length =0200
READ_PORTB . . . . . N PROC  075C   TKRSEG  Length =0027
READ_TIME. . . . . N PROC  0A41   TKRSEG  Length =0011
RECORD_NOT_FND . . . . . Number  0004
RECORD_OK. . . . . L NEAR  0924   TKRSEG
RESET. . . . . N PROC  0966   TKRSEG  Length =0010
RESET_ER . . . . . Number  2833
RESET_FLAG . . . . . L WORD  0072   DATA
RESET_OK . . . . . L NEAR  0975   TKRSEG
RET_O. . . . . L NEAR  0379   TKRSEG
RIGHT_KEY. . . . . Number  0036
RIGHT_SHIFT. . . . . Number  0001
RQM. . . . . Number  0080
RS232_BASE . . . . . L WORD  0000   DATA    Length =0004
RS232_TIM_OUT. . . . . L BYTE  007C   DATA    Length =0004

RST20. . . . . L NEAR  0338   TKRSEG
RST_5220 . . . . . Number  00FF
RST_FN . . . . . Number  0000
RST_TALKER . . . . . N PROC  0308   TKRSEG  Length =004F
RST_XX . . . . . L NEAR  0356   TKRSEG
RW_LSB . . . . . Number  0010
RW_LSBMSB. . . . . Number  0030
RW_MSB . . . . . Number  0020
SAVE_POINTER . . . . . N PROC  06AA   TKRSEG  Length =0028
SCROLL_KEY . . . . . Number  0046
SCROLL_SHIFT . . . . . Number  0010
SECOND . . . . . L NEAR  015A   TKRSEG
SEEK_END . . . . . Number  0020
SEEK_STATUS. . . . . L BYTE  003E   DATA
SERV_ER. . . . . Number  0028
SERV_OUT . . . . . L NEAR  0258   TKRSEG
SETUP_FLAG . . . . . L NEAR  0A9C   TKRSEG
SETUP_FLAG2. . . . . L NEAR  0A95   TKRSEG
SFIRST . . . . . L NEAR  0490   TKRSEG
SHIFTREG . . . . . Number  FF98
SPEAK. . . . . L NEAR  0851   TKRSEG
SPEAKER_POS. . . . . Number  070E
SPEED_0. . . . . Number  0000
SPEED_1. . . . . Number  0001
SPEED_2. . . . . Number  0002
SPEED_3. . . . . Number  0003
SPEED_4. . . . . Number  0004
SPEED_5. . . . . Number  0005
SPEED_ERR. . . . . Number  0005
SPEED_MAX. . . . . Number  0005
SPEED_TBL. . . . . L WORD  001D   TKRSEG
SPK_EXT. . . . . Number  0060
SP_CHAR. . . . . L BYTE  0028   XXDATA
SP_FLAG. . . . . L WORD  0026   XXDATA
START. . . . . F PROC  02D6   TKRSEG  Length =0034
STATUS_BYTE. . . . . L BYTE  0000   XXDATA
STATUS_CK. . . . . L NEAR  0865   TKRSEG
STATUS_CK2 . . . . . L NEAR  0867   TKRSEG
STOP_CODE. . . . . Number  00FF
SWMASK . . . . . Number  0001
SWPORT . . . . . Number  00A1
T2_A . . . . . Number  07A2   TKRSEG
```

```
T2_ABC . . . . . L BYTE  079B    TKRSEG
T2_I . . . . . Number  079A    TKRSEG
T2_ICON. . . . . L BYTE  0783    TKRSEG
T2_S . . . . . Number  07AE    TKRSEG
T2_SELECT. . . . . L BYTE  07A3    TKRSEG
T2_W . . . . . Number  07CE    TKRSEG
T2_WAVE. . . . . L BYTE  07B7    TKRSEG
T8254. . . . . L NEAR  0072    TKRSEG
TAB0 . . . . . Number  0B52    TKRSEG
TAB1 . . . . . Number  0BBA    TKRSEG
TAB2 . . . . . Number  0C22    TKRSEG


TAB3 . . . . . Number  0C90    TKRSEG
TAB1DX . . . . . Number  0B00    TKRSEG
TALKER . . . . . Number  004D
TALKER2_DIAG . . . . . F PROC  07FF    TKRSEG  Length =0167
TALKER_DIAG_PTR. . . . . L WORD  0248    DUMMY
TALKER_ICON. . . . . L NEAR  0816    TKRSEG
TALKER_PTR . . . . . L WORD  0134    DUMMY
TALK_LPC . . . . . Number  0080
TALK_ON. . . . . Number  0080
TEST_BITS. . . . . L NEAR  013C    TKRSEG
TEST_FRAME_HI. . . . . L NEAR  00D1    TKRSEG
TEST_FRAME_LO. . . . . L NEAR  00DA    TKRSEG
TEST_HALF_BUF_BIT. . . . . L NEAR  0620    TKRSEG
THRESHOLD. . . . . Number  012C
TIMER. . . . . Number  0040
TIMER0 . . . . . Number  0040
TIMER_ERROR. . . . . L NEAR  00A3    TKRSEG
TIMER_HIGH . . . . . L WORD  006E    DATA
TIMER_LOW. . . . . L WORD  006C    DATA
TIMER_OFL. . . . . L BYTE  0070    DATA
TIME_OUT . . . . . Number  0080
TIM_CTL. . . . . Number  0043
TKR_ACL. . . . . Number  FF9F
TK_EX. . . . . L NEAR  0960    TKRSEG
TK_EX1 . . . . . L NEAR  095A    TKRSEG
TK_EX_LINK1. . . . . L NEAR  087A    TKRSEG
TK_EX_LINK2. . . . . L NEAR  08D0    TKRSEG
TK_HD_SC . . . . . L BYTE  0000    DKDATA  Length =0008
TLR_WIDTH. . . . . Number  0007
TLOOP. . . . . L NEAR  0480    TKRSEG
TOS. . . . . L WORD  0100    STACK
TOTLTPO. . . . . L NEAR  0272    TKRSEG
TRACK0 . . . . . L BYTE  0074    DATA
TRACK1 . . . . . L BYTE  0075    DATA
TRACK2 . . . . . L BYTE  0076    DATA
TRANSIT. . . . . L NEAR  00E5    TKRSEG
TRUE_MEM . . . . . L WORD  0015    DATA
TST_CMP. . . . . L NEAR  0161    TKRSEG
TYPE_OFF . . . . . Number  0008
UPDATE_COMPLETE. . . . . L NEAR  0689    TKRSEG
VAR_DELAY. . . . . L BYTE  0086    DATA
VGA_CTL. . . . . Number  03DA
VIDEO_INT. . . . . L WORD  0040    ABS0
WAIT0. . . . . L NEAR  0413    TKRSEG
WAIT00 . . . . . L NEAR  0651    TKRSEG
WAIT1. . . . . L NEAR  0418    TKRSEG
WAITX0 . . . . . L NEAR  0487    TKRSEG
WAITX1 . . . . . L NEAR  0482    TKRSEG
WAIT_0 . . . . . L NEAR  0425    TKRSEG
WAIT_1 . . . . . L NEAR  0420    TKRSEG
WAIT_FOR_LPC . . . . . N PROC  037A    TKRSEG  Length =000E
WAIT_RDY . . . . . N PROC  064E    TKRSEG  Length =000F
WAVE_POS . . . . . Number  0A15


WD_ENABLE. . . . . Number  0020
WD_STROBE. . . . . Number  0040
WORDS_BEGIN. . . . . Number  0C90    TKRSEG
WRAP_FLAG. . . . . L BYTE  0004    XXDATA
WRITE. . . . . L NEAR  042B    TKRSEG
WRITEX . . . . . L NEAR  0439    TKRSEG
WRITE_BUF. . . . . L BYTE  0221    DKDATA  Length =0100
WRITE_PROTECT. . . . . Number  0003
WRT_FF . . . . . L NEAR  032E    TKRSEG
X. . . . . L BYTE  0000    TKRSEG
XLAT_PR. . . . . N PROC  02C5    TKRSEG  Length =000F
XPC_BYTE . . . . . N PROC  02BA    TKRSEG  Length =001A
```



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 321 BOCA RATON, FLORIDA 33432

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432

Fold here

Tape Please do not staple Tape

The Personal Computer
Hardware Library

Reader's Comment Form

Speech Attachment
Technical Reference 6138761

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments: