```
;*****************************************************************
;*****************************************************************
;**
;**  MODULE NAME:  VDEMO.ASM
;**
;**  CONTENTS:     CALC_TIME    - Calculate sample time
;**                CALC_SPACE   - Calculate memory requirements
;**                START        - Main program start of VOICE DEMO
;**
;**
;**     THIS IS PROPRIETARY AND CONFIDENTIAL INFORMATION OF
;**            M I N D S E T  C O R P O R A T I O N
;**
;****************** WRITTEN BY WILLIAM C. JORDAN *********************
;*****************************************************************

TRUE                    EQU             1               ;boolean true
FALSE                   EQU             0               ;boolean false
CR                      EQU             0DH             ;carriage return
LF                      EQU             0AH             ;line feed
BUFSIZE                 EQU             65535           ;size of data segment
THOLD                   EQU             8               ;voice threshold
PRINT_DELAY             EQU             3FFFH           ;printing delay

;***
;*** STACK SEGMENT
;***

STACK                   SEGMENT         STACK
                        DW              256 DUP (?)
STACK_BTM               LABEL           WORD
STACK                   ENDS

;***
;*** VARIABLE SEGMENT
;***

VARIAB                  SEGMENT
RFLAG                   DB              FALSE           ;True if voice in mem
SSEG                    DB              1               ;Number of segments
SSIZE                   DW              15000           ;size of last segment
SDELAY                  DW              14              ;Offset in delay table
FILE_NAME               DB              'B:VOICE.SMP',0 ;Name of voice file
MENU_NAME               DB              'B:VOICE.MNU',0 ;Name of menu file
FIRST_TIME              DB              TRUE
VARIAB                  ENDS

;***
;*** DATA SEGMENTS
;***

DATA                    SEGMENT

VBUFFER                 DB              BUFSIZE DUP(00)

DATA                    ENDS
DATA1                   SEGMENT
```

```
VBUFFER1                DB              BUFSIZE DUP(?)

DATA1                   ENDS
DATA2                   SEGMENT

VBUFFER2                DB              BUFSIZE DUP (?)

DATA2                   ENDS
DATA3                   SEGMENT

VBUFFER3                DB              6000 DUP (?)

DATA3                   ENDS
                        ASSUME          CS:CODE,SS:STACK,DS:NOTHING,ES:VARIAB
                        EXTRN           SCANC:NEAR, PRINT:NEAR
                        EXTRN           PUTC:NEAR, GETC:NEAR
                        EXTRN           PUTNUM:NEAR, GETNUM:NEAR
                        EXTRN           INIT_D_TO_A:NEAR, INIT_A_TO_D:NEAR
                        EXTRN           SEND_D_TO_A:NEAR,SAMPLE_A_TO_D:NEAR
                        EXTRN           ENABLE:NEAR, DISABLE:NEAR
                        EXTRN           CREATE:NEAR, OPEN:NEAR
                        EXTRN           READ:NEAR, WRITE:NEAR
                        EXTRN           CLOSE:NEAR, EXIT:NEAR

CODE                    SEGMENT


;*************************************************************************
;*************************************************************************
;**
;**   PROCEDURE NAME:  CALC_TIME
;**
;**   DESCRIPTION:     This procedure calculates the time in seconds
;**                    it will take to record or playback voice
;**                    sampled at the specified sample rate and
;**                    using the specified memory size.
;**
;**   PARAMETERS:      None
;**
;**   RETURNS:         AL = Number of seconds it will take to record
;**                         or play back voice samples using formula:
;**                         #sec = (#segs * 65536 + lastbuf) / #khz
;**
;**   DESTROYS:        AX, CX, DX
;**
;*************** WRITTEN BY WILLIAM C. JORDAN ****************
;*************************************************************************

CALC_TIME               PROC            NEAR

                        MOV             AX,SDELAY       ;get sample rate
                        SHR             AX,1            ;divide by 2 and
                        INC             AX              ;increment
                        MOV             CX,1000         ;convert to khz
                        MUL             CX
                        MOV             CX,AX           ;save in CX
                        MOV             DL,SSEG         ;get #segs
                        DEC             DL              ;actually one less
```

```
                MOV             DH,0            ;zero high byte
                MOV             AX,SSIZE        ;load remaning size
                DIV             CX              ;divide by khz
                RET
CALC_TIME       ENDP

;*************************************************************
;*************************************************************
;**
;**  PROCEDURE NAME: CALC_SPACE
;**
;**  DESCRIPTION:    This procedure computes the amount of space
;**                  required to sample voice at a given sample
;**                  rate and duration in seconds.
;**                  SSEG  = (#secs * KHZ)  /  65535
;**                  SSIZE = (#secs * KHZ) mod 65535
;**
;**  PARAMETERS:     BX - Number of seconds to sample for.
;**
;**  RETURNS:        Nothing
;**
;**  DESTROYS:       AX, DX
;**
;***************** WRITTEN BY WILLIAM C. JORDAN ******************
;*************************************************************

CALC_SPACE      PROC            NEAR

                MOV             AX,BX           ;save seconds in AX
                MOV             DX,SDELAY       ;get sample rate
                SHR             DX,1
                INC             DX
                MUL             DL              ;mult #sec by rate
                MOV             DX,1000         ;convert to khz
                MUL             DX
                INC             DL              ;one more seg
                MOV             SSEG,DL         ;save #segs in mem
                MOV             SSIZE,AX        ;save remainder in mem
                RET

CALC_SPACE      ENDP


;*************************************************************************
;***  VOICE DEMONSTRATION - MAIN PROGRAM
;*************************************************************************

START:
                MOV             AX,STACK
                MOV             SS,AX           ;STACK SEGMENT
                MOV             SP,OFFSET STACK_BTM     ;STACK POINTER
                MOV             AX,VARIAB
                MOV             ES,AX           ;EXTRA SEGMENT

                CALL            INIT_D_TO_A
                CALL            INIT_A_TO_D

                MOV             RFLAG,FALSE     ;NOTHING RECORDED
```

```
                        MOV             SSEG,1          ;FIRST SEGMENT
                        MOV             SSIZE,15000     ;2 SECONDS
                        MOV             SDELAY,14       ;INITIAL DELAY 9HHZ

                        MOV             DX,OFFSET MENU_NAME
                        JMP             FAR PTR VOICE_START
BEGIN:
                        MOV             SI,OFFSET INTRO ;OUTPUT BANNER
RE_LIST:
                        CALL            PRINT
                        CALL            GETC            ;GET A CHOICE
                        PUSH            AX
                        CALL            PUTC            ;ECHO IT
                        POP             AX

                        CMP             AL,'A'          ;'A' FOR SET ATTRIBUTES
                        JNE             NOT_A
                        JMP             FAR PTR SET_ATTR

NOT_A:                  CMP             AL,'E'          ;'E' FOR ECHO
                        JNE             NOT_E
                        JMP             FAR PTR EFFECTS
NOT_E:                  CMP             AL,'P'          ;'P' FOR PLAY
                        JNE             NOT_P
                        JMP             FAR PTR PLAYBACK

NOT_P:                  CMP             AL,'R'          ;'R' FOR RECORD
                        JNE             NOT_R
                        JMP             RECORDX

NOT_R:                  CMP             AL,'S'          ;'S' FOR SAVE ON DISK
                        JNE             NOT_S
                        JMP             FAR PTR WRITEX

NOT_S:                  CMP             AL,'L'          ;'L' FOR LOAD DISK
                        JNE             NOT_L
                        JMP             FAR PTR LOADX

NOT_L:                  CMP             AL,'Q'          ;'Q' FOR QUIT
                        JNE             NOT_Q
                        JMP             FAR PTR QUIT

NOT_Q:                  MOV             SI,OFFSET QQPS  ;BAD CHOICE
                        CALL            PRINT
                        MOV             SI,OFFSET MENU  ;RE-DISPLAY MENU
                        JMP             RE_LIST

;************************ RECORD SOME VOICE ********************************

RECORDX:
                        MOV             SI,OFFSET RECORD_MSG    ;RECORD BANNER
                        CALL            PRINT
                        CALL            GETC            ;WAIT FOR KEY

                        MOV             RFLAG,TRUE      ;VOICE IS RECORDED

                        MOV             SI,SDELAY       ;GET DELAY FROM TABLE
```

```
                        MOV             DI,CS:DELAYTABLE[SI]

                        CALL            DISABLE         ;TURN OFF INTERRUPTS

                        MOV             AX,DATA         ;VOICE DATA SEGMENT
                        MOV             DS,AX           ;DATA SEGMENT
                        MOV             CL,SSEG         ;NUMBER OF SEGMENTS
                        MOV             CH,0
SEGLOOP:                PUSH            CX              ;SAVE #SEGS ON STACK
                        CMP             CX,1            ;LAST SEG?
                        JE              LASTSEG
                        MOV             CX,0000         ;NO, 64K WHEN LOOPED
                        JMP             NOTLAST
LASTSEG:                MOV             CX,SSIZE        ;USE REMAINDER
NOTLAST:                MOV             SI,0            ;START AT BYTE 0
LOOP1:

                        CALL            SAMPLE_A_TO_D   ;GET A SAMPLE FROM A/D
                        PUSH            AX
                        CALL            SEND_D_TO_A     ;SEND TO SPEAKER
                        POP             AX
                        MOV             DS:VBUFFER[SI],AL ;PUT SAMPLE IN BUFFER
                        PUSH            CX
                        MOV             CX,DI           ;DELAY LOOP
WOW:                    LOOP            WOW
                        POP             CX
                        INC             SI              ;POINT TO NEXT BYTE
                        LOOP            LOOP1           ;IS SEGMENT FULL?

                        POP             CX              ;YES, NEXT SEG
                        MOV             AX,DS
                        ADD             AX,1000H        ;ADD 64K
                        MOV             DS,AX
                        LOOP            SEGLOOP         ;LAST SEGMENT?

                        CALL            ENABLE          ;RE-ENABLE INTS

                        MOV             SI,OFFSET COMMANDS
                        JMP             RE_LIST         ;GO GET NEXT COMMAND

;********************** PLAY BACK RECORDED VOICE ******************

PLAYBACK:
                        MOV             AL,RFLAG        ;IS VOICE RECORDED?
                        CMP             AL,FALSE
                        JNE             GOT_VOICE       ;YES, PLAY IT BACK

                        MOV             SI,OFFSET NO_VOICE ;NO, OOPS!
                        CALL            PRINT
                        MOV             SI,OFFSET COMMANDS ;GO GET ANOTHER
                        JMP             RE_LIST         ;COMMAND

GOT_VOICE:
                        CALL            DISABLE         ;DISABLE INTS

                        MOV             SI,SDELAY       ;GET DELAY FROM TABLE
                        MOV             DI,CS:DELAYTABLE[SI]
```

```
                        MOV             AX,DATA
                        MOV             DS,AX           ;DATA SEGMENT
                        MOV             CL,SSEG         ;LOAD NUMBER OF SEGS
                        MOV             CH,0
SEGLOOP1:               PUSH            CX              ;SAVE ON STACK
                        CMP             CX,1            ;LAST ONE?
                        JE              LASTSEG1
                        MOV             CX,0000         ;NO, 64K WHEN LOOPED
                        JMP             NOTLAST1
LASTSEG1:               MOV             CX,SSIZE        ;YES, USE REMAINDER
NOTLAST1:               MOV             SI,0            ;START AT BYTE 0

LOOP2:                  MOV             AL,DS:UBUFFER[SI]       ;GET SAMPLE
;*** DEBUG
                        MOV             DX,8140H        ;USE OSCILLOSCOPE
                        OUT             DX,AL           ;TO COMPARE PLAYBACK
;*** DEBUG              ;RATE
                        CALL            SEND_D_TO_A     ;OUTPUT IT
                        PUSH            CX
;****************** DELAY TO MAKE UP FOR RECORDING ******************
                        MOV             CX,7
OUCH:                   LOOP            OUCH
;****************** DELAY TO MAKE UP FOR RECORDING ******************
                        MOV             CX,DI           ;DELAY SAMPLE TIME
WOW1:                   LOOP            WOW1
                        POP             CX
                        INC             SI              ;POINT TO NEXT BYTE
                        LOOP            LOOP2           ;LAST BYTE IN SEG?

                        POP             CX              ;RESTORE SEG NUMBER
                        MOV             AX,DS
                        ADD             AX,1000H        ;ADD 64K
                        MOV             DS,AX
                        LOOP            SEGLOOP1        ;LAST SEGMENT?
                        CALL            ENABLE          ;YES, ENABLE INTS
                        MOV             AL,FIRST_TIME
                        CMP             AL,TRUE
                        JNE             OLD_HAT
                        MOV             FIRST_TIME,FALSE
                        JMP             FAR PTR BEGIN

OLD_HAT:                MOV             SI,OFFSET COMMANDS
                        JMP             RE_LIST         ;GO GET NEXT COMMAND

;********************** ECHO EFFECT *******************************

EFFECTS:                MOV             AL,SSEG         ;ECHO CAN ONLY USE
                        CMP             AL,1            ;FIRST SEGMENT
                        JE              OK_TO_ECHO
                        MOV             SI,OFFSET PLEASE_SET    ;TELL USER
                        CALL            PRINT           ;TO CHANGE TIME
                        JMP             FAR PTR DISSALLOW

OK_TO_ECHO:             MOV             SI,OFFSET ESCAPE        ;ECHO BANNER
                        CALL            PRINT
```

```
                        MOV     CX,PRINT_DELAY  ;WAIT A LITTLE
PXD:                    LOOP    PXD
                        MOV     RFLAG,FALSE     ;ECHO ERASES RECORDING
                        MOV     SI,SDELAY       ;GET DELAY RATE
                        MOV     DI,CS:DELAYTABLE[SI]
                        CALL    DISABLE         ;DISABLE INTS
                        MOV     AX,DATA         ;USER FIRST SEGMENT
                        MOV     DS,AX
LOOPX:
                        CALL    SAMPLE_A_TO_D   ;GET A SAMPLE
                        PUSH    AX
                        CALL    SEND_D_TO_A     ;OUTPUT IT
                        MOV     CX,DI
XX:                     LOOP    XX              ;DELAY AT SAMPLE RATE
                        POP     AX
                        CMP     AL,THOLD        ;OVER THRESHOLD?
                        JL      LOOPX           ;NO, KEEP LOOPING
ECHO:                   MOV     CX,SSIZE        ;#SAMPLES IN SEG
                        MOV     SI,0

LOP1:                   CALL    SAMPLE_A_TO_D   ;GET A SAMPLE
                        MOV     DS:VBUFFER[SI],AL       ;PUT IN MEM
                        PUSH    CX
                        CALL    SEND_D_TO_A     ;SEND TO SPEAKER
                        MOV     CX,DI
D1:                     LOOP    D1              ;DELAY SAMPLE TIME
;******************** DELAYS TO MAKE UP FOR SHIFTING DATA ********
                        NOP
                        NOP
                        NOP
;******************** DELAYS TO MAKE UP FOR SHIFTING DATA ********
                        POP     CX
                        INC     SI
                        LOOP    LOP1            ;DONE WITH SAMPLING?

                        CALL    ENABLE          ;YES, OUTPUT A '*'
                        MOV     AL,'*'
                        CALL    PUTC

                        MOV     CX,PRINT_DELAY  ;WAIT FOR IT
PXY:                    LOOP    PXY

                        CALL    DISABLE         ;TURN OFF INTS AGAIN

                        MOV     CX,8            ;ECHO 8 TIMES
ECH:                    PUSH    CX

                        MOV     CX,SSIZE        ;SIZE OF SAMPLE
                        MOV     SI,0
LOP3:
                        MOV     AL,DS:VBUFFER[SI]       ;GET A SAMPLE
                        PUSH    CX
                        PUSH    AX
                        CALL    SEND_D_TO_A     ;SEND IT TO SPEAKER
                        MOV     CX,DI
```

```
D2:             LOOP    D2              ;DELAY SAMPLE TIME
                POP     AX
                SAR     AL,1            ;DIVIDE BY 2
                MOV     DS:VBUFFER[SI],AL       ;SAVE IN MEM

                CALL    SAMPLE_A_TO_D   ;LOOK TO START OVER
                POP     CX

                CMP     AL,THOLD        ;IS SOMEONE MAKING
                JGE     XXXX            ;NOISE?

                INC     SI              ;POINT TO NEXT SAMPLE
                LOOP    LOP3            ;ANY LEFT?

                CALL    SCANC           ;HAS A KEY BEEN HIT
                JNZ     YYYY

                POP     CX
                LOOP    ECH             ;NO, LAST TIME TO ECHO?
                JMP     LOOPX           ;YES, LOOK FOR NEW ONE

XXXX:           POP     CX
                JMP     ECHO            ;GET A NEW SAMPLE

YYYY:           POP     CX              ;DONE WITH ECHO...
                CALL    GETC            ;GET KEY
                CALL    ENABLE          ;ENABLE INTS
DISSALLOW:      MOV     SI,OFFSET COMMANDS      ;GET NEXT
                JMP     RE_LIST                 ;COMMAND

;********************* SET SAMPLE/PLAYBACK PARAMETERS ***************************

SET_ATTR:

                MOV     SI,OFFSET SET_MSG       ;BANNER FOR SET
                CALL    PRINT
                MOV     AX,SDELAY               ;OUTPUT DEFAULT
                SHR     AX,1                    ;KHZ
                INC     AX
                CALL    PUTNUM
                MOV     SI,OFFSET END_PAREN
                CALL    PRINT
                CALL    GETNUM          ;GET KHZ
                CMP     BX,0            ;IF ZERO USE DEFUALT
                JE      NO_CHANGE
                CMP     BX,8            ;1-8 KHZ ONLY
                JG      WRONG
                DEC     BX
                SHL     BX,1
                MOV     SDELAY,BX       ;SAVE IN MEM

NO_CHANGE:      MOV     SI,OFFSET LEN_MSG       ;GET DURATION
                CALL    PRINT

                CALL    CALC_TIME       ;OUTPUT DEFUALT
                CALL    PUTNUM
                MOV     SI,OFFSET END_PAREN
                CALL    PRINT
```

```
                        CALL    GETNUM          ;GET TIME IN SECS
                        CMP     BX,0            ;IF ZERO, USE DEFAULT
                        JE      USE_OLD
                        CMP     BX,25           ;1-25 SECS ONLY
                        JG      WRONG
                        CALL    CALC_SPACE      ;COMPUTE MEMORY NEEDED

USE_OLD:                MOV     SI,OFFSET CR_LF
                        CALL    PRINT
                        MOV     SI,OFFSET COMMANDS      ;GET NEXT
                        JMP     RE_LIST                 ;COMMAND
WRONG:
                        MOV     SI,OFFSET BE_NICE       ;BLEW IT
                        CALL    PRINT
                        MOV     SI,OFFSET COMMANDS      ;GET NEXT
                        JMP     RE_LIST                 ;COMMAND

;********************* WRITE VOICE SAMPLE TO DISK *******************************

WRITEX:
                        MOV     AL,RFLAG        ;IS VOICE RECORDED?
                        CMP     AL,FALSE
                        JNE     OK_SAVE         ;YES, PLAY IT BACK

                        MOV     SI,OFFSET NO_VOICE      ;NO, OOPS!
                        CALL    PRINT
                        JMP     OH_WELL
OK_SAVE:
                        MOV     DX,OFFSET FILE_NAME
                        MOV     AX,VARIAB
                        MOV     DS,AX
                        CALL    CREATE
                        MOV     BX,AX

                        PUSH    BX
                        MOV     CX,8
                        MOV     AX,VARIAB
                        MOV     DS,AX
                        MOV     DX,OFFSET RFLAG
                        CALL    WRITE
                        POP     BX

                        MOV     CL,SSEG
                        MOV     CH,0
                        MOV     AX,DATA
                        MOV     DS,AX
WRITE_LOOP:             CMP     CX,1
                        JE      LAST_BLOCK
                        MOV     DX,65535
                        JMP     PUT_DISK
LAST_BLOCK:             MOV     DX,SSIZE
PUT_DISK:               PUSH    CX
                        PUSH    BX
                        MOV     CX,DX
                        PUSH    CX
                        MOV     DX,0
                        CALL    WRITE
                        POP     CX
                        CMP     AX,CX
                        POP     BX
```

```
                    POP             CX
                    JNE             NO_ROOM

                    MOV             AX,DS
                    ADD             AX,1000H
                    MOV             DS,AX
                    LOOP            WRITE_LOOP

                    CALL            CLOSE

OH_WELL:            MOV             SI,OFFSET COMMANDS
                    JMP             RE_LIST         ;GO GET NEXT COMMAND
NO_ROOM:            MOV             SI,OFFSET SPACE_PROB
                    CALL            PRINT
                    JMP             OH_WELL

;******************** LOAD VOICE SAMPLE FROM DISK *******************************

LOADX:
                    MOV             DX,OFFSET FILE_NAME
VOICE_START:        MOV             AX,VARIAB
                    MOV             DS,AX
                    MOV             AL,0            ;READ ONLY
                    CALL            OPEN
                    JNC             OPENED

                    MOV             AL,FIRST_TIME
                    CMP             AL,TRUE
                    JNE             NOT_FIRST
                    MOV             FIRST_TIME,FALSE
                    JMP             FAR PTR BEGIN

NOT_FIRST:          MOV             SI,OFFS CANT_OPEN
                    CALL            PRINT
                    JMP             LEAVE
OPENED:             MOV             BX,AX

                    MOV             AX,VARIAB
                    MOV             DS,AX
                    MOV             DX,OFFSET BFLAG
                    MOV             CX,6
                    PUSH            BX
                    CALL            READ
                    ;DOES AX == 6
                    POP             BX
                    MOV             CL,SSEG
                    MOV             CH,0
                    MOV             AX,DATA
                    MOV             DS,AX
LOAD_LOOP:          CMP             CX,1
                    JE              LAST_BLCK
                    MOV             DX,65535
                    JMP             GET_DISK
LAST_BLCK:          MOV             DX,SSIZE
GET_DISK:           PUSH            CX
                    PUSH            BX
                    MOV             CX,DX
                    MOV             DX,0
                    CALL            READ
```

```
                        ;CHECK IF AX == CX
                        POP             BX
                        POP             CX
                        MOV             AX,DS
                        ADD             AX,1000H
                        MOV             DS,AX
                        LOOP            LOAD_LOOP

                        CALL            CLOSE

                        MOV             AL,FIRST_TIME
                        CMP             AL,TRUE
                        JNE             LEAVE
                        JMP             FAR PTR PLAYBACK

LEAVE:                  MOV             SI,OFFSET COMMANDS
                        JMP             RE_LIST         ;GO GET NEXT COMMAND

;*********************RETURN TO DOS ******************************

QUIT:                   MOV             SI,OFFSET BYEBYE
                        CALL            PRINT
                        CALL            EXIT
                        HLT


;***** STRINGS *****

INTRO           DB                                              CR,LF
                DB              'M I N D  S E T  V O I C E  U T I L I T Y  V1.0'
MENU            DB                                              CR,LF
                DB                                              CR,LF
                DB              'OPTIONS :',                    CR,LF
                DB              'R - RECORD VOICE',             CR,LF
                DB              'P - PLAYBACK RECORDED VOICE',  CR,LF
                DB              'S - SAVE VOICE SAMPLE ON DISK',CR,LF
                DB              'L - LOAD VOICE SAMPLE FROM DISK',CR,LF
                DB              'A - SET SAMPLE/PLAYBACK ATTRIBUTES',CR,LF
                DB              'E - ECHO DEMONSTRATION',       CR,LF
                DB              'Q - QUIT',                     CR,LF
COMMANDS        DB                                              CR,LF
                DB              'CHOOSE ONE (R,P,S,L,A,E,Q) > ',0

OOPS            DB                                      CR,LF,CR,LF
                DB              '### BAD CHOICE ###',   CR,LF,0
RECORD_MSG      DB                                      CR,LF,CR,LF
                DB              'PRESS ANY KEY TO START RECORDING'
                DB                                      CR,LF,0
ESCAPE          DB                                      CR,LF,CR,LF
                DB              'ECHO DEMONSTRATION',   CR,LF
                DB              '(PRESS A KEY TO EXIT)',CR,LF,0
SPACE_PROB      DB                                      CR,LF,CR,LF
                DB              '!!! NOT ENOUGH ROOM ON DISK !!!'
                DB                                      CR,LF,0
CANT_OPEN       DB                                      CR,LF,CR,LF
                DB              'SORRY, NO VOICE FILE EXISTS !!!'
                DB                                      CR,LF,0
```

```
NO_VOICE        DB                                      CR,LF,CR,LF
                DB          'SORRY, NO VOICE RECORDED !!!'
                DB                                      CR,LF,0
SET_MSG         DB                                      CR,LF,CR,LF
                DB          'SET SAMPLE/PLAYBACK RATE',  CR,LF,CR,LF
                DB          'ENTER SAMPLE/PLAYBACK RATE  (1-8KHZ'
                DB          ',  DEFAULT = ',                 0
LEN_MSG         DB                                      CR,LF,CR,LF
                DB          'ENTER SAMPLE/PLAYBACK LENGTH (1-255ED'
                DB          ', DEFAULT = ',                  0
END_PAREN       DB          ') ? ',                      0
BYEBYE          DB                                      CR,LF,CR,LF
                DB          'BYE BYE',                   CR,LF,CR,LF
                DB                                       0
BE_NICE         DB                                      CR,LF,CR,LF
                DB          'SORRY, YOU BLEW IT!!!!'
CR_LF           DB                                      CR,LF,0
PLEASE_SET      DB                                      CR,LF,CR,LF
                DB          'PLEASE SET LESS TIME FOR ECHO!!'
                DB                                      CR,LF,0

;*** DELAY TABLE ***
DELAYTABLE      LABEL       WORD
                DW          366             ;1KHZ
                DW          180             ;2KHZ
                DW          117             ;3KHZ
                DW          85              ;4KHZ
                DW          65              ;5KHZ
                DW          53              ;6KHZ
                DW          45              ;7KHZ
                DW          39              ;8KHZ

CODE            ENDS
                END         START
```

```
;*****************************************************************
;*****************************************************************
;**
;**   MODULE NAME:   TTY.ASM
;**
;**   CONTENTS:      PUTC   - Put a single character on the screen
;**                  PRINT  - Print a string on the screen
;**                  SCANC  - Scan keyboard for a character
;**                  GETC   - Get a single character from the keyboard
;**                  GETNUM - Get a decimal number from the keyboard
;**                  PUTNUM - Put a decimal number on the screen
;**
;**      THIS IS PROPRIETARY AND CONFIDENTIAL INFORMATION OF
;**
;**             M I N D S E T   C O R P O R A T I O N
;**
;****************** WRITTEN BY WILLIAM C. JORDAN ******************
;*****************************************************************

SCAN_CHR        EQU     1               ;SCAN FOR A CHAR
GET_CHR         EQU     0               ;GET A SINGLE CHAR
WRITE_TELE      EQU     14              ;WRITE SINGLE CHAR
PAGE_NUM        EQU     0               ;PAGE IN ALPHA MODE
CR              EQU     0DH             ;CARRIAGE RETURN
LF              EQU     0AH             ;LINE FEED

                ASSUME  CS:CODE, DS:NOTHING

CODE            SEGMENT BYTE PUBLIC

                PUBLIC  PRINT, PUTC, SCANC, GETC
                PUBLIC  GETNUM, PUTNUM

;*****************************************************************
;*****************************************************************
;**  PROCEDURE NAME: PUTC
;**
;**  DESCRIPTION:    This procedure places a single character at
;**                  the current cursor position of the CRT screen.
;**
;**  PARAMETERS:     AL - Character to print
;**
;**  RETURNS:        NOTHING
;**
;**  DESTROYS:
;**
;****************** WRITTEN BY WILLIAM C. JORDAN ******************
;*****************************************************************

PUTC            PROC    NEAR

                MOV     AH,WRITE_TELE
                MOV     BL,0
                MOV     BH,PAGE_NUM
                INT     10H
                RET

PUTC            ENDP
```

```
;*****************************************************************
;*****************************************************************
;**
;**   PROCEDURE NAME: PRINT
;**
;**   DESCRIPTION:    This procedure outputs a string, terminated
;**                   with a zero, to the CRT screen.
;**
;**   PARAMETERS:     CS:[SI] - Address of first character in string
;**                             that is terminated by zero.
;**
;**   RETURNS:        NOTHING
;**
;**   DESTROYS:       ----
;**
;****************** WRITTEN BY WILLIAM C. JORDAN ******************
;*****************************************************************

PRINT   PROC    NEAR

NCHAR:  MOV     AL,CS:BYTE PTR[SI]      ;get char
        CMP     AL,0                    ;terminator?
        JE      ESTRING                 ;yes, done.
        CALL    PUTC                    ;output a char
        INC     SI                      ;next
        JMP     NCHAR
ESTRING: RET

PRINT   ENDP

;*****************************************************************
;*****************************************************************
;**
;**   PROCEDURE NAME: SCANC
;**
;**   DESCRIPTION:    This procedure scans the keyboard for character
;**                   input. If a key has been depressed, the ZF flag
;**                   will be zero.
;**
;**   PARAMETERS:     None
;**
;**   RETURNS:        AL - Character from keyboard if ZF=0
;**
;**   DESTROYS:       ----
;**
;****************** WRITTEN BY WILLIAM C. JORDAN ******************
;*****************************************************************

SCANC   PROC    NEAR

        MOV     AH,SCAN_CHR
        INT     15H
        RET

SCANC   ENDP
```

```
;*****************************************************************
;*****************************************************************
;**
;**   PROCEDURE NAME: GETC
;**
;**   DESCRIPTION:    This procedure gets a single character from the
;**                   keyboard.
;**
;**   PARAMETERS:     None
;**
;**   RETURNS:        AL = Character typed from keyboard
;**
;**   DESTROYS:       ----
;*****************  WRITTEN BY WILLIAM C. JORDAN *********************
;*****************************************************************

GETC                    PROC    NEAR

                        MOV     AH,GET_CHR
                        INT     16H
                        RET

GETC                    ENDP

;*****************************************************************
;*****************************************************************
;**
;**   PROCEDURE NAME: GETNUM
;**
;**   DESCRIPTION:    This procedure gets a decimal number from
;**                   the keyboard.  If a carriage return is
;**                   entered only, a zero is returned.
;**
;**   PARAMETERS:     None
;**
;**   RETURNS:        BX = Decimal number entered in from keyboard.
;**
;**   DESTROYS:       ----
;**
;*****************  WRITTEN BY WILLIAM C. JORDAN *********************
;*****************************************************************

GETNUM                  PROC    NEAR

                        MOV     BX,0            ;initial value of BX
GETLOOP:                CALL    GETC            ;get a char
                        CMP     AL,CR           ;If CR or LF, done
                        JZ      EOL
                        CMP     AL,LF
                        JZ      EOL
                        CMP     AL,'0'          ;Must be 0 thru 9
                        JL      GETLOOP
                        CMP     AL,'9'
                        JG      GETLOOP
                        PUSH    AX              ;Assume PUTC destroys
```

```
                        PUSH    BX              ;AX and BX
                        CALL    PUTC            ;Output the character
                        POP     AX
                        MOV     BX,10           ;Mult by 10 for decimal
                        MUL     BX
                        MOV     BX,AX
                        POP     AX
                        SUB     AL,'0'
                        CBW
                        ADD     BX,AX           ;Add in next char
                        JMP     GETLOOP
EOL:                    RET                     ;all done...

GETNUM                  ENDP

;*****************************************************************
;*****************************************************************
;**  PROCEDURE NAME: PUTNUM
;**
;**  DESCRIPTION:    This procedure converts a number to decimal
;**                  and outputs it on the CRT screen.
;**
;**  PARAMETERS:     AX - Decimal number to output to screen
;**
;**  RETURNS:        Nothing
;**
;**  DESTROYS:       ----
;**
;***************** WRITTEN BY WILLIAM C. JORDAN ******************
;*****************************************************************

PUTNUM                  PROC    NEAR

                        MOV     CX,0
                        MOV     DL,10           ;10 for decimal
CNURT:                  DIV     DL
                        PUSH    AX              ;Put chars on stack
                        INC     CX
                        XOR     AH,AH
                        CMP     AX,0            ;Done when zero...
                        JNZ     CNURT
DISPLAY:                POP     AX              ;Get chars off stack
                        MOV     AL,AH
                        ADD     AL,'0'          ;Convert to ascii
                        CALL    PUTC            ;Output it
                        LOOP    DISPLAY         ;more?
                        RET                     ;done.

PUTNUM                  ENDP


CODE                    ENDS

                        END
```

CCAT.ASM

```
DELAY_IN        EQU     40
DELAY_OUT       EQU     DELAY_IN
BUFSIZE         EQU     5000

STACK           SEGMENT STACK
                DW      1000 DUP (?)
STACK_BTM       LABEL   WORD
STACK           ENDS

DATA            SEGMENT
VBUFFER         DB      BUFSIZE DUP (?)
DATA            ENDS

                ASSUME  CS:CODE, SS:STACK, DS:DATA

CODE            SEGMENT

                EXTRN   INIT_D_TO_A:NEAR, INIT_A_TO_D:NEAR
                EXTRN   SAMPLE_A_TO_D:NEAR, SEND_D_TO_A:NEAR
                EXTRN   DISABLE:NEAR

START:

                MOV     AX,STACK        ;INIT STACK PTR
                MOV     SS,AX
                MOV     SP,OFFSET STACK_BTM
                MOV     AX,DATA         ;INIT DATA PTR
                MOV     DS,AX

                MOV     AX,0            ;TURN OFF DISPLAY
                INT     0EFH

                CALL    DISABLE
                CALL    INIT_D_TO_A
                CALL    INIT_A_TO_D

;
;*** RECORD VOICE ***
;
GETVOICE:       MOV     CX,BUFSIZE      ;LOAD INITIAL COUNT
                MOV     SI,0            ;START AT BYTE 0
LOOP1:

                CALL    SAMPLE_A_TO_D   ;GET A SAMPLE FROM A/D
                PUSH    CX
                MOV     CX,DELAY_IN
WOW:            LOOP    WOW
                POP     CX
                MOV     VBUFFER[SI],AL  ;PUT SAMPLE IN BUFFER
                INC     SI              ;POINT TO NEXT BYTE
                LOOP    LOOP1           ;IS BUFFER FULL?

;
;*** PLAYBACK ***
;
                MOV     CX,BUFSIZE      ;LOAD INITIAL COUNT
                MOV     SI,0            ;START AT BYTE 0
```

```
LOOP2:      PUSH    CX
            PUSH    SI
            MOV     AL,VBUFFER[SI]  ;GET SAMPLE

            MOV     DX,8140H
            OUT     DX,AL

            CALL    SEND_D_TO_A     ;OUTPUT IT
            MOV     CX,DELAY_OUT
WOW1:       LOOP    WOW1
            POP     SI
            POP     CX
            INC     SI              ;POINT TO NEXT BYTE
            LOOP    LOOP2           ;ALL OFF BUFFER?

            MOV     CX,08FFFH
DEL:        LOOP    DEL
            JMP     GETVOICE

CODE        ENDS

            END     START
```

```
;*************************************************************************
;*************************************************************************
;**
;**   MODULE NAME:  VOICE.ASM
;**
;**   CONTENTS:     This module contains assembly language routines
;**                 to drive the Analog-to-Digital converter and
;**                 the Custom Sound Processor chip in see-through
;**                 mode to serve as a Digital-to-Analog converter.
;**
;**                 INIT_A_TO_D   - Send initial OUT command to A/D
;**                                 converter to start a conversion
;**                                 and wait 125 usec before returning.
;**                 INIT_D_TO_A   - Put CSP in see-through mode.
;**                 SAMPLE_A_TO_D - Get sample from A/D converter and
;**                                 send an OUT to start another
;**                                 conversion.
;**                 SEND_D_TO_A   - Send data to D/A convert through
;**                                 the CSP.
;**                 DISABLE       - Disable interrupts so that voice
;**                                 samples will not be distorted.
;**                 ENABLE        - Re-enable interrupts.
;**
;**
;**      THIS IS PROPRIETARY AND CONFIDENTIAL INFORMATION OF
;**
;**              M I N D S E T   C O R P O R A T I O N
;**
;****************** WRITTEN BY WILLIAM C. JORDAN ******************
;*************************************************************************


;*** EQUATES FOR THE DEVICES ***
ADC0804                 EQU             8140H           ;ADDRESS OF A/D
D_TO_A                  EQU             82A2H           ;ADDRESS OF D/A
MAGIC                   EQU             0FF2AH          ;ISR MASK

;*** DELAY CONSTANT ***
I_DELAY                 EQU             41              ;125 uSECs

;*** EQUATES FOR THE CSP ***
SET_MODE                EQU             24H             ;SET UP CSP
SOUND_DATA              EQU             26H             ;D/A DATA COMMAND
STEREO                  EQU             3               ;BOTH CHANNELS, FOR AL
SEE_THRU                EQU             0               ;MODE 4,        FOR BL


                        ASSUME          CS:CODE, DS:NOTHING

CODE                    SEGMENT         BYTE PUBLIC

                        PUBLIC          INIT_A_TO_D, INIT_D_TO_A
                        PUBLIC          SAMPLE_A_TO_D, SEND_D_TO_A
                        PUBLIC          DISABLE, ENABLE

;*************************************************************************
;*************************************************************************
;**
;**   PROCEDURE NAME: INIT_A_TO_D
;**
```

```
;**  DESCRIPTION:    This procedure starts the Analog-to-Digital
;**                  converter sampling. It starts the A/D by
;**                  outputting to it. Before returning, 125uSec
;**                  of time passes to garentee that the next
;**                  sample will not be before the conversion has
;**                  completed.
;**
;**  PARAMETERS:     None
;**
;**  RETURNS:        Nothing
;**
;**  DESTROYS:       CX, DX
;**
;****************** WRITTEN BY WILLIAM C. JORDAN ******************
;******************************************************************

INIT_A_TO_D             PROC            NEAR

                        MOV             DX,ADC0804      ;ADDRESS OF A/D
                        OUT             DX,AL           ;START CONVERSION
                        MOV             CX,I_DELAY      ;DON'T RETURN UNTIL
REFUSE:                 LOOP            REFUSE          ;125 uSECs ELAPSED
                        RET

INIT_A_TO_D             ENDP
```

```
;******************************************************************
;******************************************************************
;**
;**  PROCEDURE NAME: INIT_D_TO_A
;**
;**  DESCRIPTION:    This procedure sets up the CSP for See-through
;**                  mode (ie, used as a Digital-to-Analog converter.
;**
;**  PARAMETERS:     None
;**
;**  RETURNS:        Nothing
;**
;**  DESTROYS:       AX, BL
;**
;****************** WRITTEN BY WILLIAM C. JORDAN ******************
;******************************************************************

INIT_D_TO_A             PROC            NEAR

                        MOV             AH,SET_MODE     ;SET SOUND MODE
                        MOV             AL,STEREO       ;SEND TO BOTH CSP'S
                        MOV             BL,SEE_THRU     ;SEE THRU MODE
                        INT             0EEH            ;TELL OS
                        RET

INIT_D_TO_A             ENDP


;******************************************************************
;******************************************************************
;**
;**  PROCEDURE NAME: SAMPLE_A_TO_D
;**
```

```
;**  DESCRIPTION:     This procedure gets a sample from the A/D
;**                   and places it in the AL register.
;**                   Do not call this procedure before 114uSecs
;**                   have elapsed since the previous call or
;**                   the sample is not garanteed to be valid.
;**
;**  PARAMETERS:      None
;**
;**  RETURNS:         AL - 8 bit sample from A/D (signed number)
;**
;**  DESTROYS:        AL, DX
;**
;****************** WRITTEN BY WILLIAM C. JORDAN ******************
;******************************************************************

SAMPLE_A_TO_D       PROC        NEAR

                    MOV         DX,ADC0804      ;ADDRESS OF A/D
                    IN          AL,DX           ;GET SAMPLE
                    OUT         DX,AL           ;START NEXT CONVERSION
                    XOR         AL,7FH          ;COMPLIMENT DATA
                    RET

SAMPLE_A_TO_D       ENDP

;******************************************************************
;******************************************************************
;**
;**  PROCEDURE NAME: SEND_D_TO_A
;**
;**  DESCRIPTION:     This procedure sends data to the D/A
;**                   for outputting to the loud speaker.
;**
;**  PARAMETERS:      AL - Signed data to send to D/A converter
;**
;**  RETURNS:         Nothing
;**
;**  DESTROYS:        AL, DX
;**
;****************** WRITTEN BY WILLIAM C. JORDAN ******************
;******************************************************************

SEND_D_TO_A         PROC        NEAR

                    XOR         AL,80H          ;COMPLIMENT SINE BIT
                    MOV         DX,D_TO_A
                    OUT         DX,AL
                    RET

SEND_D_TO_A         ENDP

;******************************************************************
;******************************************************************
;**
;**  PROCEDURE NAME: DISABLE
;**
;**  DESCRIPTION:     This procedure disables the interrupts so
;**                   voice samples will not be distorted by the
```

```
;**                    timing inconsistencies.
;**
;**  PARAMETERS:       None
;**
;**  RETURNS:          Nothing
;**
;**  DESTROYS:         AX, DX
;**
;****************** WRITTEN BY WILLIAM C. JORDAN ******************
;******************************************************************

DISABLE                 PROC            NEAR

                        MOV             DX,MASIC
                        MOV             AX,2
                        OUT             DX,AX
                        RET

DISABLE                 ENDP


;******************************************************************
;******************************************************************
;**
;**  PROCEDURE NAME: ENABLE
;**
;**  DESCRIPTION:    This procedure re-enables the interupts.
;**                  It is called so that output can be done
;**                  to the terminal after sampling voice.
;**
;**  PARAMETERS:     None
;**
;**  RETURNS:        Nothing
;**
;**  DESTROYS:       AX, DX
;**
;****************** WRITTEN BY WILLIAM C. JORDAN ******************
;******************************************************************

ENABLE                  PROC            NEAR

                        MOV             DX,MASIC
                        MOV             AX,7
                        OUT             DX,AX
                        RET

ENABLE                  ENDP

CODE                    ENDS

                        END
```

```
;*************************************************************************
;*************************************************************************
;**
;**   MODULE NAME:   DOSDRIVR.ASM
;**
;**   CONTENTS:      CREATE -
;**                  OPEN   -
;**                  READ   -
;**                  WRITE  -
;**                  CLOSE  -
;**                  EXIT   - Return to DOS
;**
;**
;**       THIS IS PROPRIETARY AND CONFIDENTIAL INFORMATION OF
;**
;**              M I N D S E T   C O R P O R A T I O N
;**
;****************** WRITTEN BY WILLIAM C. JORDAN ******************
;*************************************************************************

CREATE_FILE     EQU             3CH             ;CREATE A FILE
OPEN_FILE       EQU             3DH             ;OPEN A FILE
READ_FILE       EQU             3FH             ;READ FROM A FILE
WRITE_FILE      EQU             40H             ;WRITE TO A FILE
CLOSE_FILE      EQU             3EH             ;CLOSE A FILE
EXIT_TO_DOS     EQU             4CH             ;EXIT TO DOS
ARCHIVE         EQU             20H             ;ARCHIVE ATTRIBUTE

                ASSUME          CS:CODE, DS:NOTHING

CODE            SEGMENT         BYTE PUBLIC

                PUBLIC          CREATE, OPEN, READ, WRITE, CLOSE, EXIT

CREATE          PROC            NEAR

                MOV             AH,CREATE_FILE  ;CREATE A NEW FILE
                MOV             CX,ARCHIVE      ;ARCHIVE BIT IS SET
                INT             21H
                RET

CREATE          ENDP


OPEN            PROC            NEAR

                MOV             AH,OPEN_FILE    ;OPEN EXISTING FILE
                INT             21H
                RET

OPEN            ENDP

READ            PROC            NEAR

                MOV             AH,READ_FILE    ;READ CONTENTS OF A FILE
                INT             21H
                RET
```

```
READ            ENDP


WRITE           PROC            NEAR

                MOV             AH,WRITE_FILE   ;WRITE TO A FILE
                INT             21H
                RET

WRITE           ENDP


CLOSE           PROC            NEAR

                MOV             AH,CLOSE_FILE   ;CLOSE FILE
                INT             21H
                RET

CLOSE           ENDP


;*****************************************************************
;*****************************************************************
;**
;**  PROCEDURE NAME: EXIT
;**
;**  DESCRIPTION:    This procedure forces the main program to
;**                  return to dos.
;**
;**  PARAMETERS:     None
;**
;**  RETURNS:        Does not return (kiss your program goodbye)
;**
;**  DESTROYS:       Everything
;**
;***************** WRITTEN BY WILLIAM O. JORDAN ******************
;*****************************************************************

EXIT            PROC            NEAR

                MOV             AH,EXIT_TO_DOS          ;RETURN TO DOS
                MOV             AL,0                    ;NO ERRORS
                INT             21H
                HLT

EXIT            ENDP


CODE            ENDS

                END
```