ARI Research Note 94-13

Micro Computer FeedBack Report for the Strategic Leader Development Inventory—Source Code

James E. Hopkins
Independent Contractor

Strategic Leadership Technical Area
T. Owen Jacobs, Chief

Manpower and Personnel Research Division
Zita M. Simutis, Director

March 1994

United States Army
Research Institute for the Behavioral and Social Sciences

U.S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency Under the Jurisdiction of the Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON
Director

Research accomplished under contract
for the Department of the Army

James Hopkins
Independent Contractor

Technical review by

T. Owen Jacobs

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1994, March	Final Jun 92-Sep 92

4. TITLE AND SUBTITLE
Micro Computer Feedback Report for the Strategic Leader Development Inventory--Source Code

5. FUNDING NUMBERS
DAAL03-91-C-0034
62785A
791
1111
TCN 93-304

6. AUTHOR(S)
Hopkins, James E. (Independent)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
--

8. PERFORMING ORGANIZATION REPORT NUMBER
--

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Institute for the Behavioral and Social Sciences
ATTN: PERI-RO
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
ARI Research Note 94-13

11. SUPPLEMENTARY NOTES
Task was performed under a Scientific Services Agreement issued by Battelle, Research Triangle Park Office, NC 27709.
Contracting Officer's Representative, T. Owen Jacobs (U.S. Army Research Institute)

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE
--

13. ABSTRACT (Maximum 200 words)

In 1990, individuals at the U.S. Army War College (USAWC) saw the need for a tool to provide leadership developmental feedback to incoming students to help with planning for the resident year and progress following that year. The U.S. Army Research Institute for the Behavioral and Social Sciences (ARI) developed and pilot-tested the Strategic Leader Development Inventory (SLDI) to satisfy that need. As a part of the concept, an automated feedback printing program was developed to enable user organizations to print graphic feedback that facilitates student interpretation of SLDI scale scores. That program was developed for the academic year (AY) 91-92 pilot test and was revised for the academic year 92-93 field test of the SLDI.

The AY92-93 version of the FeedBack program produced reports for a fixed set of questions and evaluation factors. If any changes were made in the SLDI, the program had to be rewritten. The AY93-94 upgraded version of the FeedBack program is flexible. It allows the survey questions and the evaluation factors to be redefined through the use of a look-up table, thereby enabling continuous user product

(Continued)

14. SUBJECT TERMS

ASM	Assembler
PCL	Computer
SLDI	Leadership

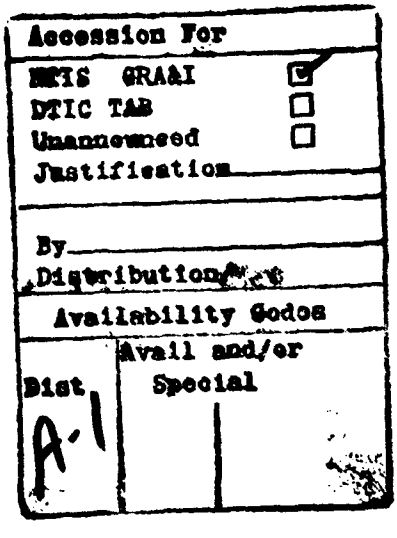
15. NUMBER OF PAGES
164

16. PRICE CODE
--

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

13. ABSTRACT (Continued)

improvement over time, and adaptation of the SLDI to different subject populations as needed. This report contains the source code for the FeedBack program. A user's guide, Micro Computer FeedBack Program for the Strategic Leader Development Inventory: User's Guide, has been published separately.



ACKNOWLEDGMENTS

As a member of the Largo High School Mathematics Department, I participated in the U.S. Army Summer Associateship Program for High School Science and Mathematics Faculty. My summer associateship was supported by the U.S. Army Research Institute for the Behavioral and Social Sciences, Strategic Leadership Technical Area, T. O. Jacobs, Chief, under the auspices of the U.S. Army Research Office Scientific Services Program administered by Battelle.

I wish to express my appreciation to T. Owen Jacobs and Steven R. Stewart for their continued assistance and support in creating the FeedBack micro computer program. They provided me with the insights, responsibility, and flexibility to complete my assignment.

INTRODUCTION

The Strategic Leader Development Inventory (SLDI) is being developed by the U.S. Army Research Institute for the Behavioral and Social Sciences (ARI), Strategic Leadership Technical Area (SLTA), in collaboration with the U.S. Army War College (USAWC) and the Industrial College of the Armed Forces (ICAF). The SLDI allows the student to assess himself/herself and to obtain assessments from former superiors, peers, and subordinates on a number of attributes important for effective strategic leader performance.

In addition to completing a self-assessment form, students select as many as three superiors, three peers, and four subordinates to provide assessments. After all the surveys are completed, the answers are scanned into an ASCII text file. The FeedBack micro computer program uses the ASCII text files to print a graphic report for each participants that summarizes the results from all sources. The participants can use this report to compare their self-descriptions with those of fellow students and with those provided by former peers, subordinates, and superiors.

In addition to the FeedBack graphic reports, each participant receives a written description of the research that led to development of the SLDI and the items defining each of the factors.

Because of time limitations, the academic year 92-93 version of the FeedBack program produced reports for a static set of predefined factors. Alteration of any of the SLDI factors required that the FeedBack computer program be rewritten. It was known at the outset that "tailorable" feedback would be necessary for a final operational system, and that was a primary goal for revision of the program. The program was rewritten to allow the factors to be defined using a look-up table and now is substantially more flexible than the previous version. In addition, it will now print on any Hewlet Packard LaserJet printer, though the font derivation mechanism is different among the various generations of these printers.

This report contains the source code for the FeedBack program. It exists in compiled form and is available to qualified requesters from the U.S. Army Research Institute for the Behavioral and Social Sciences. A user's guide, Micro Computer FeedBack Program for the Strategic Leader Development Inventory: User's Guide, has been published separately and is available to qualified requestors through the Defense Technical Information Center.

Appendix A:

SOURCE CODE

```
;FeedBack.ASM                    Summer 1993                    James K. Hopkins
;A program to print Strategic Leader Development Inventory feedback forms.
;
        .MODEL small
;
        STACKSIZE       EQU     3328
        ;3328 = up to 2304 bytes of heap space for CFG data
        ;and not less than 1024 for stack space.
        .STACK  STACKSIZE
;
        .DATA
;---- HP PCL strings used to position a point on the graph
NextUn  db      32,1Bh,'&k2S',1Bh,'&a+19C',1Bh,'&k0S',0 ;space+19 comp. spaces
NextTn  db      1Bh,'&k2S',1Bh,'&a+2C',1Bh,'&k0S',0     ;2 compressed spaces
HalfSp  db      1Bh,'&k2S',1Bh,'&a+1C',1Bh,'&k0S',0     ;1 compressed spaces
BackSp  db      1Bh,'&k2S',1Bh,'&a-1C',1Bh,'&k0S',0     ;1 compressed spaces
FullBk  db      1Bh,'&k2S',1Bh,'&a-2C',1Bh,'&k0S',0     ;1 compressed spaces

;
;---- HP PCL strings used by printing procedures
Heder   db      1Bh,'(s3BSTRATEGIC  LEADER  DEVELOPMENT  INVENTORY',0
;
IDstr   db      'ID: '
ID      db      20 DUP (32),0
;
PosT    db      '|',1Bh,'&k2S',1Bh,'&a-1C',1Bh,'&k0S^Q'
        db      1Bh,'&k2S',1Bh,'&a-1C',1Bh,'&k0S-'
        db      1Bh,'&k2S',1Bh,'&a-1C',1Bh,'&k0S-'
        db      'Below Average    Better Than Most     The Best '
        db      1Bh,'&k2S',1Bh,'&a-1C',1Bh,'&k0S-'
        db      1Bh,'&k2S',1Bh,'&a-1C',1Bh,'&k0S-'
        db      1Bh,'&k2S',1Bh,'&a-1C',1Bh,'&k0S^P'
        db      1Bh,'&k2S',1Bh,'&a-1C',1Bh,'&k0S|',0

NegT    db      '|',1Bh,'&k2S',1Bh,'&a-1C',1Bh,'&k0S^Q'
        db      1Bh,'&k2S',1Bh,'&a-1C',1Bh,'&k0S-'
        db      1Bh,'&k2S',1Bh,'&a-1C',1Bh,'&k0S-'
        db      'Never              Occasionally          Always '
        db      1Bh,'&k2S',1Bh,'&a-1C',1Bh,'&k0S-'
        db      1Bh,'&k2S',1Bh,'&a-1C',1Bh,'&k0S-'
        db      1Bh,'&k2S',1Bh,'&a-1C',1Bh,'&k0S^P'
        db      1Bh,'&k2S',1Bh,'&a-1C',1Bh,'&k0S|',0
;
DTstr   db      'Scored: '
Date    db      '07/24/92',0                            ;file date
;
DIstr   db      'FACTORS:',1Bh,'(s0B',0                 ;unBold, EndStMarker
;
FFeed   db      0Ch,0                                   ;formfeed string
Median  db      186,0                                   ;179 = "‖"
Left    db      179,0                                   ;179 = "│"
Right   db      179,0                                   ;179 = "│"
TenLt   db      177,0                                   ;177 = "▒"
TenDk   db      178,0                                   ;178 = "▓"
UntLT   db      13 DUP (177),0
UntDk   db      13 DUP (178),0

BON     db    1Bh,'(s3B',0                              ;bold ON
BOFF    db    1Bh,'(s0B',0                              ;bold OFF

Other   db    1Bh,'(s3B',6,1Bh,'(s0B',0                  ;bold ON,"^F",bold OFF
Point   db    1Bh,'(s3B',4,1Bh,'(s0B',0                  ;bold ON,"^D",bold OFF
```

```
;----An HP PCL string used by Initialize_HP procedure
Init    db      1Bh,'E'                                 ;reset printer
        db      1Bh,'&l1O'                              ;landscape
        db      1Bh,'&k0S'                              ;10.0 cpi
        db      1Bh,'(10U'                              ;PC-8 symbol set
        db      1Bh,'(s0P',0                            ;Fixed spacing

;----An HP PCL string used by Restore_HP procedure
Rest    db      1Bh,'&l0O'                              ;portrait
        db      1Bh,'(8U'                               ;Roman-8 symbol set
        db      1Bh,'(s1P'                              ;Proportional spacing
        db      1Bh,'E'                                 ;reset printer
        db      1Ah,0                                   ;end of file marker

;----An HP PCL string used by HPGOTOYX procedure
loc     db      1Bh,'&a'                                ;set hp laser to
col     dw      0                                       ;ASC II column number
        db      'C',1Bh,'&a'                            ;set hp laser to
row     dw      0                                       ;ASC II row number
        db      'R',0                                   ;end of string marker


;----An HP PCL string for drawing scale and top line of factor box.
Top     db      1Bh,'&f0S Raw Scores:  1'                   ;starting Push
      ;ticks and "2"
        db      1Bh,'&k2S - - - - - - - - - ',1Bh,'&k0S'    ;compressed mode
        db      1Bh,'&f0S',0Ah,194,1Bh,'&f1S2'              ;top tick mark
      ;ticks and "3"
        db      1Bh,'&k2S - - - - - - - - - ',1Bh,'&k0S'    ;compressed mode
        db      1Bh,'&f0S',0Ah,194,1Bh,'&f1S3'               ;top tick mark
      ;ticks and "4"
        db      1Bh,'&k2S - - - - - - - - - ',1Bh,'&k0S'    ;compressed mode
        db      1Bh,'&f0S',0Ah,194,1Bh,'&f1S4'              ;top tick mark
      ;ticks and "5"
        db      1Bh,'&k2S - - - - - - - - - ',1Bh,'&k0S5'   ;compress and "5"
        db      1Bh,'&f1S',0Ah              ;ending Pop + line feed = next line
      ;top line
        db      218,13 DUP (196),194,49 DUP (196),191,0      ;EndOfString

;----An HP PCL string for drawing scale ticks and bottom line of factor box.
Bot     db      1Bh,'&f0S                   '               ;starting Push
        db      1Bh,'&k2S                       ',1Bh,'&k0S'    ;compressed mode
        db      193                                             ;bottom tick mark
        db      1Bh,'&k2S                       ',1Bh,'&k0S'    ;compressed mode
        db      193                                             ;bottom tick mark
        db      1Bh,'&k2S                       ',1Bh,'&k0S'    ;compressed mode
        db      193                                             ;bottom tick mark
        db      1Bh,'&f1S'                                      ;ending Pop
      ;bottom line
        db      192,13 DUP (196),193, 49 DUP (196),217,0    ;EndOfString


;--- HP PCL strings to used to draw one line of the factor box.
self    db      179,'         Self ',179,1Bh,'&a+49C',179,0  ;EndOfString
peer    db      179,'        Peers ',179,1Bh,'&a+49C',179,0  ;EndOfString
super   db      179,'   Superiors ',179,1Bh,'&a+49C',179,0  ;EndOfString
subor   db      179,'Subordinates ',179,1Bh,'&a+49C',179,0  ;EndOfString

;----An HP PCL string to draw a information box and the present cursor position
Inform  db      1Bh,'&f0S',218,63 DUP (196),191
        db      1Bh,'&f1S',0Ah                              ;next line
```

```
        db      1Bh,'&f0S',179
        db      '  ^D = Self   ^F = Others    | = 25% or 75%   || = 50%   ▒ = Range
'
        db      179,1Bh,'&f1S',0Ah                              ;next line
        db      192, 63 DUP (196),217,0                 ;EndOfString marker

        .CODE

;       Is the data file open, unranked, and ID > 0
;       Input = None
;       Output = Carry flag if Not Ready.
PROC    IS_RANK
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------is data file open?
        MOV     BX,[DATHd]                              ;get file handle
        CMP     BX,0                                    ;is a file open?
        JNZ     RK1
        CALL    FILE_ERR
        JMP     SHORT RK4
;------were the percentiles in the rank file?
RK1:    CMP     WORD PTR [RNKHd],0                      ;are percentiles set?
        JZ      RK5
        CALL    PERCT_ERR                               ;Rerank file ?
        JNC     RK5                                     ;carry flag = NO
RK4:    STC                                             ;set error flag
RK5:    POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    IS_RANK
;
;
;       Input = none
;       Output = carry flag = NO
PROC    PERCT_ERR
        CALL    CLEAR_MESSAGE
        MOV     AL,[Warning]                            ;warning color
        MOV     CL,[Color]                              ;save original color
        MOV     [Color],AL                              ;set color
        MOV     AX,020Bh                                ;row 3/Col 8
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      ' This data file is already ranked.  '
        db      'Rank it again?  Y/[N] ',0
        MOV     [Color],CL                               ;restore original color
        CALL    HIDE_CUR
        CALL    ERR_SOUND
PER1:   CALL    GET_CHAR
        AND     AL,5Fh                                   ;turn off bits 6 & 8
        CMP     AL,'N'                                   ;is it No?
        JZ      PER3                                     ;if yes exit
        CMP     AL,0Dh                                   ;is it <Enter>?
        JZ      PER3                                     ;if not continue
        CMP     AL,'Y'                                   ;is it Yes?
        JNZ     PER1                                      ;if not get another
PER2:   CLC
        JMP     SHORT PER4
PER3:   STC                                              ;clear carry flag
```

```
PER4:   RET
ENDP    PERCT_ERR
;
;  Release the memory variable block.
;       Input = None
;       Output = Carry flag if DOS error
;       [VarSeg] = starting segment address for variable block.
;
PROC    RELEASE_VAR_BLK
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        XOR     AX,AX                           ;zero AX
        CMP     [VarSeg],AX                     ;is VarSeg assigned?
        JZ      REL1                            ;if not assigned go on
;------release assigned memory block
        MOV     AX,[VarSeg]                     ;get memory segment
        MOV     ES,AX                           ;place in ES register
        MOV     AX,4900h                        ;release funcdtion no
        INT     21h                             ;release memory block
        JC      REL1                            ;if No error continue
;------initialize  variable
        XOR     AX,AX                           ;zero to register
        MOV     [VarSeg],AX                     ;set memory bock to 0
        CLC                                     ;clear carry flag
REL1:   POP     ES
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        RET
ENDP    RELEASE_VAR_BLK
;
;
;  Create a byte array to be used to rank each variable.
;       Input = [MaxID] > 0
;       Output = Carry flag if DOS error
;       [VarSeg] = Starting segment address of memory block.
;       [MaxNo] = total number ID's in the file.
;
PROC    GET_VAR_BLK
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        CALL    RELEASE_VAR_BLK
        JNC     CRV0                            ;continue if no error
        JMP     CRV9                            ;exit on DOS error
CRV0:   MOV     AX,[MaxID]                      ;get number of ID's
        MOV     CL,4                            ;no. bits to shift
        SHR     AX,CL                           ;paragraph = ID/16 + 1
        MOV     BX,AX                           ;paragraph count to BX
        INC     BX                              ;get an extra paragraph
        MOV     AH,48h                          ;allocate men function
        INT     21h                             ;request memory block
        JC      CRV2                            ;jump if memory error.
        MOV     [VarSeg],AX                     ;base address of seg
        JMP     SHORT CRV8                      ;normal exit of proc.
CRV2:   MOV     CL,[Color]                      ;save original color
        MOV     AL,[Warning]                    ;warning color
        MOV     [Color],AL                      ;set color
        MOV     AX,0207h                        ;row/Col
```

```
        CALL    GOTOYX                                  ;position cursor
        CALL    CSTR_OUT                                ;send string to screen
        db      ' Not enough memory to rank the variables. '
        db      'Press Any Key to Continue. ', 0
        MOV     [Color],CL                              ;restore original color
        CALL    HIDE_CUR                                ;hide cursor off screen
        CALL    ERR_SOUND
        CALL    GET_CHAR                                ;wait for key is pressed
        STC                                             ;set carry flag = error
        JMP     SHORT  CRV9
CRV8:   CLC                                             ;clear carry flag
CRV9:   POP     ES
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        RET
ENDP    GET_VAR_BLK
;
;
;  Clear Percentilte variables.
;       Input = None
;       Output = None  192 hex 0 to [OutBuf]
;
PROC    CLEAR_PERCNT
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
;------fill [FilBuf] with 192 hex 0's
        MOV     AX,DS                                   ;Make ES = DS
        MOV     ES,AX
        MOV     CX,96                                   ;max number of words
        MOV     BX,Offset OutBuf                        ;pointer to ASCIIZ str
        XOR     AX,AX                                   ;hex 0's to AX register
        MOV     [BX],AX                                 ;0 to first word of str
        MOV     DI,BX                                   ;DI = pointer to next
        INC     DI                                      ;ptr to next word
        INC     DI
        MOV     SI,BX                                   ;SI = ptr to hex 0's
        CLD                                             ;auto inc DI and SI
        REP     MOVSW                                   ;fill string with 0's
        CLC                                             ;clear carry flag
        POP     ES
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    CLEAR_PERCNT
;
;
;------fill the following answer buffers with hex 0's
;  SF1Buf  db      256 DUP (0h),0h                       ;buff for Self I data
;  SF2Buf  db      256 DUP (0h),0h                       ;buff for Self II data
;  PERBuf  db     2048 DUP (0h),0h                       ;buffer for 8 Peer lines
;  SUPBuf  db     2048 DUP (0h),0h                       ;buf 8 Superior lines
;  SUBBuf  db     2048 DUP (0h),0h                       ;buf 8 Subordinate lines

;       Input = None
;       Output = None  6660 hex 0 to fill answer buffers
;
```

```
PROC    CLEAR_ANSWERS
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        MOV     AX,DS                           ;Make ES = DS
        MOV     ES,AX
        MOV     CX,3329                         ;max number of words
        MOV     BX,Offset SF1Buf                ;pointer to ASCIIZ str
        XOR     AX,AX                           ;hex 0's to AX register
        MOV     [BX],AX                         ;0 to first word of str
        MOV     DI,BX                           ;DI = pointer to next
        INC     DI                              ;ptr to next word
        INC     DI
        MOV     SI,BX                           ;SI = ptr to hex 0's
        CLD                                     ;auto inc DI and SI
        REP     MOVSW                           ;fill string with 0's
        CLC                                     ;clear carry flag
        POP     ES
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    CLEAR_ANSWERS
;
;
;   Move byte hex number in data file to memory [VarSeg]
;       Input = BX = Offset in [VarSeg]
;               DX = Offset in [FilBuf]
;       Output = None
PROC    VAR_TO_BLK
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        MOV     AX,[VarSeg]                     ;ptr to base of memblk
        MOV     ES,AX                           ;ES set to memory blk
        MOV     SI,DX                           ;source pointer
        MOV     DI,BX                           ;destination pointer
;------copy byte to memory for sorting
        MOVSB
        CLC                                     ;clear carry flag
        POP     ES
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    VAR_TO_BLK
;
;
;-------Sort the Word Variables in [SegVar].
;
;       Input = AX = count of word variables.
;       Output = None
;       Note: this routine reassigns the DS and ES registers to [SegVar]
;       Special Note: It does not sort the first word of [SegVar] so
;                     ranking variables begin at [SegVar] + 2 Offset and
;                     go to (2 x numbers found) Offset
```

```
;       This sort is based on the following TPASCAL procedure:
;       PROCEDURE Sort;             {A Shell Sort}
;       VAR
;          Gap,J : Integer;
;          Temp : string[13];
;          TempNo : Integer;
;       Begin
;          Gap := MaxRec Div 2;
;          While gap > 0  Do
;          Begin
;             For I := (Gap + 1) to MaxRec Do
;             Begin
;                J := I-Gap;
;                While J > 0 Do
;                Begin
;                   If A[J] > A[J+Gap] then
;                   Begin
;                      Temp := A[J];
;                      A[J] := A[J+Gap];
;                      A[J+Gap] := Temp;
;                      J := J-Gap;
;                   End
;                      Else J := 0;
;                   End;
;             End;
;             Gap :=Gap DIV 2;
;          End;
;       End;
;       The follow registers hold the above variables:
;       AX = Gap;  BX = J;  CX = I;  DX = MaxRec; and  BP = temp storage
;
PROC    VAR_SORT
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    DS
        PUSH    ES
        PUSH    BP
        MOV     DX,AX                                   ;store MaxRec in DX
        MOV     AX,[VarSeg]                             ;get Index base segment
        MOV     DS,AX                                   ;reassign the  DS & ES
        MOV     ES,AX                                   ;to ptr to the Index.
        XOR     BX,BX                                   ;zero buffer pointer
        MOV     [BX],BL                                 ;first byte unused
        MOV     AX,DX                                   ;Gap = MaxRec
        SHR     AX,1                                    ;Gap = Gap Div by 2
VARS1:  CMP     AX,0                                    ;when Gap = 0 exit.
        JLE     VARS4                                   ;exit if <= 0
        MOV     CX,AX                                   ;I is stored in CX
        INC     CX                                      ;I = Gap + 1
VARS2:  MOV     BX,CX                                   ;J in BX
        SUB     BX,AX                                   ;J = I - Gap
        JZ      VARS3                                   ;skip if J = 0
        JC      VARS3                                   ;skip if J is < 0.
        CALL    COMPARE_VAR                             ;repeat until J = 0
VARS3:  INC     CX                                      ;I = I + 1
        CMP     DX,CX                                   ;Is I < or = MaxRec
        JNC     VARS2                                   ;If yes then loop.
        SHR     AX,1                                    ;Gap = Gap Div by 2
        JMP     SHORT VARS1
VARS4:  POP     BP                                      ;restore registers
```

```
        POP     ES
        POP     DS
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET                                             ;sort is complete.
;
;-------Compare and swap words if needed.
;
;       Input = AX = Gap;  BX = J;   DS & ES point to the base of index file.
;       Output = [none]      items swaped in memory if needed
;
PROC    COMPARE_VAR
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     DX,AX                                   ;save Gap in DX
;       Compare the byte of each pointer
COMV1:  MOV     BP,BX                                   ;save J in BP
        ADD     AX,BX                                   ;AX = J + Gap
      ;  SHL     AX,1                                    ;ptr to J+Gap in mem
      ;  SHL     BX,1                                    ;ptr to J in mem
        CLD                                             ;auto-inc SI, DI
        MOV     DI,AX                                   ;offset of J + Gap
        MOV     SI,BX                                   ;offset of J
        CMPSB                                           ;compare bytes
        JLE     COMV3                                   ;exit if < or =.
;       Swap the 1 bytes of index record if string A > string A+Gap
        MOV     DI,AX                                   ;offset of J + Gap
        MOV     SI,BX                                   ;offset of J
        MOV     AL,[SI]                                 ;read word each str.
        MOV     AH,[DI]
        MOV     [SI],AH                                 ;write word each str.
        MOV     [DI],AL
        MOV     AX,DX                                   ;restore gap to AX
        MOV     BX,BP                                   ;restore J to BX
        SUB     BX,AX                                   ;J = J - gap
        JZ      COMV3                                   ;exit if J = 0.
        JNC     COMV1                                   ;continue if J > 0.
COMV3:  POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET                                             ;return to Shell_Sort
ENDP    COMPARE_VAR
ENDP    VAR_SORT
;
;
;
;  Copy first,last, median, 25th and 75th percentiles the [OutBuf] data string.
;       Input = AX = Number of variables found
;               Round AX to and even number = ptr to 50%    50/2=25%
50%+25%=75%
;       Output =  First, Last, Median  and 25% and 75% stored in [OutBuf]
;
PROC    STORE_VAR
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
```

```
        PUSH    ES
        MOV     DX,AX                   ;save number found
        MOV     BX, Offset OutBuf       ;begin of var string
        MOV     AX,[VarSeg]             ;base of memory block
        MOV     ES,AX                   ;ES ptr to block seg
;------get the lowest
        MOV     DI,1                    ;ptr to lowest score
        MOV     AL,[ES:DI]              ;get lowest score
        MOV     [BX],AL                 ;store lowest score
        INC      BX                     ;word pointer.
;------get the highest
        MOV     DI,DX                   ;ptr to highest score
        MOV     AL,[ES:DI]              ;get last score
        MOV     [BX],AL                 ;store last score
        INC      BX                     ;advance [OutBuf]
;------get the 50%
        INC     DX                      ;count + 1
        TEST    DL,01h                  ;is the number even?
        JZ      STV1                    ;if Yes goto next test
        INC     DL                      ;if NO make it even
STV1:   SHR     DX,1                    ;div by 2
        MOV     DI,DX                   ;50% ptr
        MOV     AL,[ES:DI]              ;get 50% value
        MOV     [BX],AL                 ;store 50% value
        INC      BX                     ;advance [OutBuf]
;------get the 25% and 27%
        MOV     AX,DX                   ;restore 50% ptr
        DEC     DX
        INC     AX
        TEST    AL,01h                  ;is the number even?
        JZ      STV2                    ;if Yes goto next test
        INC     AX                      ;if NO make it even
STV2:   SHR     AX,1                    ;50%/2 =ptr to 25%
        MOV     DI,AX                   ;25% ptr in DI
        ADD     DX,AX                   ;75% ptr in DX
        MOV     AL,[ES:DI]              ;get 25% value
        MOV     [BX],AL                 ;stroe 25% value
        INC      BX                     ;word pointer.
        MOV     DI,DX                   ;75% ptr to DI
        MOV     AL,[ES:DI]              ;get 25% value
        MOV     [BX],AL                 ;stroe 25% value
        CLC                             ;clear carry flag
        POP     ES
        POP     DX                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    STORE_VAR
;
;
;       Input = none
;       Output = none
PROC    RANK_WAIT_MESS
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------please wait message to screen.
        XOR     AX,AX                   ;clear menu area
        CALL    MENU_BOX
        MOV     CL,[Color]              ;save original attri
```

```
        MOV     AL,[Warning]                                    ;warning color
        MOV     [Color],AL                                      ;set color
        MOV     AX,010Bh                                        ;row 3/Col 12
        CALL    GOTOYX                                          ;set cursor
        CALL    CSTR_OUT                                        ;display warning
        db      ' Please wait ........      Ranking the data file: ',0
        MOV     AX, Offset FileNa
        CALL    DSTR_OUT
        CALL    CSTR_OUT
        db      ' ',0
        MOV     [Color],CL                                      ;restore original attri
        CALL    HIDE_CUR
        CLC
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    RANK_WAIT_MESS
;
;   Please wait message to screen.
;       Input = AX = number of scores
;               CX = loop count  = variable number
;
;       Output = message to the screen
PROC    PROGRESS_MESS
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     DL,[Color]                                      ;save original attri
        MOV     AL,[Menu]                                       ;menu color
        MOV     [Color],AL                                      ;set color
        MOV     AX,0108h                                        ;row 3/Col 12
        CALL    GOTOYX                                          ;set cursor
        CALL    CSTR_OUT                                        ;display warning
        db      ' Please wait ........      Ranking variable ',0
        MOV     AX,[DRecLn]                                     ;ID + means =rec ln
        SUB     AX,20                                           ;subtract ID length
        MOV     BX,AX                                           ;store no of means
        SUB     AX,CX                                           ;subtract loop counter
        INC     AX                                              ;start count from 1
        CALL    BIN_OUT                                         ;number of current mean
        CALL    CSTR_OUT                                        ;to screen
        db      ' of ',0
        MOV     AX,BX                                           ;get no of means
        CALL    BIN_OUT                                         ;no of means to screen
        CALL    CSTR_OUT                                        ;in one record
        db      ' variables.    ',0
        MOV     [Color],DL                                      ;restore original attri
        CALL    HIDE_CUR
        CLC
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PROGRESS_MESS
;
;
;       Input = none
;       Output = carry flag = abort printing
```

```
PROC    ESC_YN
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    CLEAR_MESSAGE
        MOV     CL,[Color]                          ;store original Color
        MOV     AL,[Warning]                        ;warning color
        MOV     [Color],AL                          ;set color
        MOV     AX,020Dh                            ;row 3/Col 12
        CALL    GOTOYX                              ;set cursor
        CALL    CSTR_OUT                            ;display warning
        db      " Do you want to ABORT the ranking process ? "
        db      " Y/N ",0
        MOV     [Color],CL                           ;restore original color
ESY1:   CALL    HIDE_CUR
        CALL    GET_CHAR
        AND     AL,0DFh                              ;turn off bit 6
        CMP     AL,'N'                               ;is it No?
        JZ      ESY4                                 ;if yes exit
ESY2:   CMP     AL,'Y'                               ;is it Yes?
        JNZ     ESY3                                 ;if not continue
        STC                                          ;set carry flag = abort
        JMP     SHORT ESY5                           ;exit
ESY3:   CALL    ERR_SOUND
        JMP     SHORT ESY1
ESY4:   CALL    CLEAR_MESSAGE                        ;empty message line
        CLC                                          ;clear cf = continue
ESY5:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    ESC_YN
;
;
;-------Instructions for rank command.
;       Input = None
;       Output = None
;
PROC    RANK_INSTRU
        PUSH    AX                                   ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,1500h                             ;row 21,column 0
        CALL    MENU_BOX                             ;draw menu box
        MOV     CL,[Color]                           ;get assigned color
        MOV     AL,[Menu]                            ;get menu color
        MOV     [Color],AL                           ;set menu color
        MOV     AX,160Ah                             ;row 22,column 12
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'Press the <Esc> key to pause or cancel the '
        db      'ranking of scores.',0
        CALL    HIDE_CUR
        MOV     [Color],CL                           ;restore assigned color
        POP     DX                                   ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
```

```
ENDP    RANK_INSTRU

;
;------Read the next record form the DAT file.
;       Input =  None (assumes file ptr is in correct position)
;       Output = Carry flag = file closes or file ptr already at EndOfFile.
;
PROC    READ_DAT
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     BX,[DATHd]                          ;get file handle
        CMP     BX,0                                ;is file open ?
        JZ      RDT2                                ;if NO Exit
;------read next record from the data file
        MOV     CX,[DRecLn]                         ;no. of bytes to read
        MOV     AX,3F00h                            ;read file function no
        MOV     DX,Offset FilBuf                    ;buffer ptr to DX
        INT     21h                                 ;get bytes
        JC      RDT2                                ;end of file?
        CMP     AX,CX                               ;was read complete ?
        JZ      RDT3                                ;if Yes normal return
RDT2:   STC                                         ;else set carry flag
RDT3:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    READ_DAT
;

;
;  Clear Input Buffer.
;       Input = None
;       Output = None  6 hex 0 to [FilBuf]
;
PROC    SET_DEFAULT
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------fill rank buffer with 5 hex 0's
        MOV     BX,Offset OutBuf                    ;ptr to buffer
        XOR     AX,AX                               ;hex zero to ax
        MOV     [BX],AX                             ;make first word 0
        INC     BX                                  ;ptr to next word
        INC     BX
        MOV     [BX],AX                             ;make second word 0
        INC     BX                                  ;ptr to next word
        INC     BX
        MOV     [BX],AL                             ;make 5th byte 0
;------place offset into DAT record for matching mean score.
        INC     BX                                  ;ptr to next word
        MOV     AX,[DRecLn]                         ;get DAT rec length
        SUB     AX,CX                               ;subtract loop counter
        MOV     [BX],AX                             ;store DAT ptr
        CLC                                         ;clear carry flag
        POP     DX                                  ;restore registers
        POP     CX
        POP     BX
        POP     AX
```

```
        RET
ENDP    SET_DEFAULT
;
;
;       Input = none
;       Output = carry flag = DOS error
;                File handle is in [RNKHd]
;                File contains zero bytes.
;    WARNING: this proceedure will erase an existing file with the same name.
;
PROC    OPEN_NEW_RANK_FILE
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        XOR     AX,AX                           ;zero to ax register
        MOV     [RNKHd],AX                      ;set file handle to 0
        MOV     AX,DS                           ;assign ES to the
        MOV     ES,AX                           ;data section
        MOV     AX,Offset Search                ;ptr to path + file name
        MOV     CX,AX                           ;save str beginning ptr
        CALL    STR_LENGTH                      ;get length of string
        SUB     AX,4                            ;length - 4 = "."
        ADD     AX,CX                           ;ptr to the period
        MOV     DI,AX                           ;destination ptr
        MOV     SI,Offset RNKTyp                ;ptr to file type name
        CLD                                     ;auto inc SI & DI
        MOV     CX,5                            ;number byte to move
        REP     MOVSB                           ;move type to Search
        MOV     AX, Offset Search               ;ptr to name of file
        CALL    CREATE                          ;open an empty file
        JC      ONR1                            ;exit on DOS error
        MOV     [RNKHd],BX                      ;save file handle
        CLC                                     ;clear carry = data OK!
ONR1:   POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    OPEN_NEW_RANK_FILE
;
;       Input = none
;       Output = no error message if file not found or DOS error.
;    WARNING: this proceedure will erase an existing file.
;
PROC    DELETE_RANK_FILE
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,DS                           ;assign ES to the
        MOV     ES,AX                           ;data section
        MOV     AX,Offset Search                ;ptr to path + file name
        MOV     CX,AX                           ;save str beginning ptr
        CALL    STR_LENGTH                      ;get length of string
        SUB     AX,4                            ;length - 4 = "."
        ADD     AX,CX                           ;ptr to the period
        MOV     DI,AX                           ;destination ptr
        MOV     SI,Offset RNKTyp                ;ptr to file type name
        CLD                                     ;auto inc SI & DI
        MOV     CX,5                            ;number byte to move
        REP     MOVSB                           ;move type to Search
```

```
        MOV     DX,Offset Search                ;ptr to file name string
        MOV     AH,41h                          ;delete file function no
        INT     21h                             ;try to delete file.
        XOR     AX,AX                           ;zero to ax
        MOV     [RNKHd],AX                      ;zero file handle
        CLC                                     ;clear carry = data OK!
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DELETE_RANK_FILE
;
;
;       Input = None
;       Output = None  6 hex 0 to [FilBuf]
;
PROC    DISPLAY_SORT
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        MOV     DI,AX
        MOV     AX,0700h
        CALL    GOTOYX
        MOV     CX,22                           ;number found counter
        MOV     AX,[VarSeg]
        MOV     ES,AX
        SUB     DI,20
DS1:    XOR     AX,AX
        MOV     AL,[ES:DI]
        CALL    BIN_OUT
        CALL    CSTR_OUT
        db      ' ',0
        INC     DI
        LOOP    DS1
        CALL    GET_CHAR
        CLC                                     ;clear carry flag
        POP     ES
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DISPLAY_SORT
;
;
;  Read mean scores for one variable for each ID number into DOS memory block.
;       Input = None
;       Output = AX = No of mean scores to sort
;
;       [VarSeg] = starting segment address for variable block.
;       BX = stores offset ptr in memory blk or No of mean score found.
;       DX = stores offset in [FilBuf]
;
PROC    READ_VAR
        PUSH    BX
        PUSH    CX
        PUSH    DX
        XOR     AX,AX                           ;zero AX
        MOV     BX,AX                           ;destination pointer
```

```
        CMP     [VarSeg],AX                             ;is VarSeg assigned?
        JZ      RVR9                                    ;if not assigned go on
;------compute offset in [FilBuf] based on CX loop counter
        MOV     AX,[DRecLn]                             ;ID + means =rec ln
        SUB     AX,CX                                   ;subtract loop counter
        MOV     DX, Offset FilBuf                       ;set base address
        ADD     DX,AX                                   ;add offset to base ptr
        MOV     AX,[DATHd]                              ;get file handle
        CALL    GOTO_TOP                                ;reset file pointer
        JC      RVR9                                    ;exit on DOS error
;------read a ID record from the data file.
RVR1:   CALL    READ_DAT                                ;read next rec from file
        JC      RVR9                                    ;exit if EndOfFile
;------is data out of range ?
        MOV     SI,DX                                   ;ptr to buffer section
        MOV     AX,0A33h                                ;1.0 and 5.1
        CMP     [SI],AH                                 ;is score < 1.0 ?
        JC      RVR1                                    ;if yes skip mean
        CMP     [SI],AL                                 ;is it > 5.0 ?
        JNC     RVR1                                    ;if yes skip means
        INC     BX                                      ;count = count + 1
        CALL    VAR_TO_BLK                              ;move variable to block
        JMP     SHORT RVR1
RVR9:   MOV     AX,BX                                   ;return no of means
        CLC                                             ;clear carry flag
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        RET
ENDP    READ_VAR
;
;
;       Rank the data to compute lowest, 25th, 50th, 75th percentiles, and last
;       Input = None
;       Output = rank file open if randed
;
PROC    RANK_DATA
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    IS_RANK                                 ;is file already ranked?
        JC      RD7                                     ;if Yes exit
        CALL    GET_VAR_BLK                             ;create var mem block
        JC      RD7                                     ;exit on DOS error
        CALL    IS_FULL                                 ;is room available?
        JC      RD6                                     ;exit on disk full
        CALL    OPEN_NEW_RANK_FILE                      ;open new rank file
        JC      RD6                                     ;exit on DOS error
        CALL    RANK_WAIT_MESS                          ;estimate how much time
        CALL    RANK_INSTRU                             ;bottom message box
        MOV     CX,[DRecLn]                             ;number var to rank + 20
        SUB     CX,20                                   ;number of var to rank
RD1:    CALL    SET_DEFAULT                             ;set default value = 0
        CALL    READ_VAR                                ;one var for all ID's
        CMP     AX,3                                    ;were < 2 found?
        JC      RD3                                     ;is Yes write default
RD2:    CALL    PROGRESS_MESS                           ;tell user of progress
        CALL    VAR_SORT                                ;sort in DOS mem bock
        CALL    STORE_VAR                               ;get median,25% & 75%
;------Check keyboard buffer to see if the <Esc> key been pressed?
        MOV     AX,0600h                                ;DOS function # 6
```

```
        MOV     DL,0FFh                         ;read char from key-
        INT     21h                             ;board buffer.
        JZ      RD3                             ;NO key pressed continue
        CMP     AL,1Bh                          ;was it the <ESC> key?
        JNZ     RD3                             ;if NO continue
        CALL    ESC_YN                          ;if YES inform user
        JC      RD5                             ;carry flag = abort
;------Write record to RANK file.
RD3:    PUSH    CX                              ;save loop counter
        MOV     CX,7                            ;no. of bytes to write
        MOV     BX,[RNKHd]                      ;file handle to BX
        MOV     AX,4000h                        ;write to file: func.
no.
        MOV     DX,Offset OutBuf                ;ptr to data to write
        INT     21h                             ;write to the file
        POP     CX                              ;restore loop counter
        JC      RD5                             ;exit on Dos error
;------loop until each column is ranked.
RD4:    LOOP    RD1                             ;loop until cx = 0
        JMP     SHORT RD6                       ;OK! file is complete
RD5:    CALL    DELETE_RANK_FILE                ;erase rank file
;------normal exit point
RD6:    CALL    RELEASE_VAR_BLK                 ;release mem var block
        CLC
RD7:    POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    RANK_DATA
;
        .CODE
;
;       Index the IDs and place them in a memory block buffer.
;       Input = None
;       Output = Carry flag = No IDs in memory block
;
PROC    GET_INDEX
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    GET_INDEX_BLK                   ;create var mem block
        JC      IF7                             ;exit on DOS error
        CALL    READ_INDEX_FILE                 ;index file to memory
        JNC     IF6                             ;Exit if NO errors
;------create an index in memory               ;else create a new one
        CALL    INDEX_WAIT_MESS                 ;estimate how much time
        CALL    IDS_TO_MEMORY                   ;read all ID's to memory
        JC      IF7                             ;exit on Error
        CALL    SORT_INDEX                      ;sort in DOS mem bock
;------write the index to the index file
        CALL    WRITE_INDEX_FILE                ;save index to disk
        JNC     IF6                             ;exit if NO Dos error
IF5:    CALL    DELETE_NDX_FILE                 ;else erase index file
IF6:    CLC
IF7:    POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    GET_INDEX
```

```
;
;       Input = none
;       Output = no error message if file not found or DOS error.
;    WARNING: this proceedure will erase an existing file.
;
PROC    DELETE_NDX_FILE
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,DS                                   ;assign ES to the
        MOV     ES,AX                                   ;data section
        MOV     AX,Offset Search                        ;ptr to path + file name
        MOV     CX,AX                                   ;save str beginning ptr
        CALL    STR_LENGTH                              ;get length of string
        SUB     AX,4                                    ;length - 4 = "."
        ADD     AX,CX                                   ;ptr to the period
        MOV     DI,AX                                   ;destination ptr
        MOV     SI,Offset NDXTyp                        ;ptr to file type name
        CLD                                             ;auto inc SI & DI
        MOV     CX,5                                    ;number byte to move
        REP     MOVSB                                   ;move type to Search
        MOV     DX,Offset Search                        ;ptr to file name string
        MOV     AH,41h                                  ;delete file function no
        INT     21h                                     ;try to delete file.
        XOR     AX,AX                                   ;zero to ax
        MOV     [NDXHd],AX                              ;zero file handle
        CLC                                             ;clear carry = data OK!
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DELETE_NDX_FILE
;
;
;       Input = none
;       Output = none
PROC    INDEX_WAIT_MESS
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------please wait message to screen.
        XOR     AX,AX                                   ;clear menu area
        CALL    MENU_BOX
        MOV     CL,[Color]                              ;save original attri
        MOV     AL,[Warning]                            ;warning color
        MOV     [Color],AL                              ;set color
        MOV     AX,010Eh                                ;row /Col
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      " Please wait ........      Indexing the IDs. ",0
        MOV     [Color],CL                              ;restore original attri
        CALL    HIDE_CUR
        CLC
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    INDEX_WAIT_MESS
```

```
;
;
;
;  Create a memory block array to be used to index the IDs.
;       Input = [MaxID] > 0
;       Output = Carry flag if DOS error
;       [VarSeg] = Starting segment address of memory block.
;       [MaxNo] = total number ID's in the file.
;       reserving 32 byte per ID plus 2 extra entries.
;       (an extra blank at the bottom and rounding to next higher value)
;
PROC    GET_INDEX_BLK
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        CALL    RELEASE_VAR_BLK
        JNC     GIB0                            ;continue if no error
        JMP     GIB9                            ;exit on DOS error
GIB0:   MOV     AX,[MaxID]                      ;get number of ID's
        MOV     CL,4                            ;no. bits to shift
        SHL     AX,1                            ;paragraph = 2(ID) + 4
        ADD     AX,4                            ;two extra entries
        MOV     BX,AX                           ;paragraph count to BX
        MOV     AH,48h                          ;allocate men function
        INT     21h                             ;request memory block
        JC      GIB2                            ;jump if memory error.
        MOV     [VarSeg],AX                     ;base address of seg
        JMP     SHORT GIB8                      ;normal exit of proc.
GIB2:   MOV     CL,[Color]                      ;save original color
        MOV     AL,[Warning]                    ;warning color
        MOV     [Color],AL                      ;set color
        MOV     AX,0207h                        ;row/Col
        CALL    GOTOYX                          ;position cursor
        CALL    CSTR_OUT                        ;send string to screen
        db      ' Not enough memory to index the IDs. '
        db      'Press Any Key to Continue. ', 0
        MOV     [Color],CL                      ;restore original color
        CALL    HIDE_CUR                        ;hide cursor off screen
        CALL    ERR_SOUND
        CALL    GET_CHAR                        ;wait for key is pressed
        STC                                     ;set carry flag = error
        JMP     SHORT  GIB9
GIB8:   CLC                                     ;clear carry flag
GIB9:   POP     ES
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        RET
ENDP    GET_INDEX_BLK
;

;
;       Input = none
;       Output = carry flag = DOS error
;    WARNING: this proceedure will erase an existing file.
;
PROC    DELETE_INDEX_FILE
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
```

```
        PUSH    DX
        MOV     AX,DS                           ;assign ES to the
        MOV     ES,AX                           ;data section
        MOV     AX,Offset Search                ;ptr to path + file name
        MOV     CX,AX                           ;save str beginning ptr
        CALL    STR_LENGTH                      ;get length of string
        SUB     AX,4                            ;length - 4 = "."
        ADD     AX,CX                           ;ptr to the period
        MOV     DI,AX                           ;destination ptr
        MOV     SI,Offset NDXTyp                ;ptr to file type name
        CLD                                     ;auto inc SI & DI
        MOV     CX,5                            ;number byte to move
        REP     MOVSB                           ;move type to Search
        MOV     AX, Offset Search               ;ptr to name of file
        CALL    DELETE_FILE                     ;erase file
        JC      DIF1                            ;exit on DOS error
        XOR     AX,AX                           ;zero to ax
        MOV     [NDXHd],AX                      ;zero file handle
        CLC                                     ;clear carry = data OK!
DIF1:   POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DELETE_INDEX_FILE
;
;
;       Input = none
;       Output = carry flag = DOS error
;                File handle is in [NDXHd]
;                File contains zero bytes.
;    WARNING: this proceedure will erase an existing file with the same name.
;
PROC    OPEN_NEW_INDEX_FILE
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        XOR     AX,AX                           ;zero to ax register
        MOV     [NDXHd],AX                      ;set file handle to 0
        MOV     AX,DS                           ;assign ES to the
        MOV     ES,AX                           ;data section
        MOV     AX,Offset Search                ;ptr to path + file name
        MOV     CX,AX                           ;save str beginning ptr
        CALL    STR_LENGTH                      ;get length of string
        SUB     AX,4                            ;length - 4 = "."
        ADD     AX,CX                           ;ptr to the period
        MOV     DI,AX                           ;destination ptr
        MOV     SI,Offset NDXTyp                ;ptr to file type name
        CLD                                     ;auto inc SI & DI
        MOV     CX,5                            ;number byte to move
        REP     MOVSB                           ;move type to Search
        MOV     AX, Offset Search               ;ptr to name of file
        CALL    CREATE                          ;open an empty file
        JC      OIF1                            ;exit on DOS error
        MOV     [NDXHd],BX                      ;save file handle
        CLC                                     ;clear carry = data OK!
OIF1:   POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
```

```
ENDP    OPEN_NEW_INDEX_FILE
;
;     Read unsorted ID's to memory from the Data file.
;       Input = none
;       Output = carry flag = DOS error
;
; Format for the memory index:
;  word: ID number,4 spaces,20 character ID string, 4 spaces,2 hex zeros
;  Note: Record number 0 exists and is marked as record 0 but is never sorted.
;        Record number MaxId + 1 exist but is never marked or sorted.
;        Note: rec 0 and MaxId + 1 are add to allow quick binary searches.
;
PROC    IDS_TO_MEMORY
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        MOV     AX,[VarSeg]                     ;set register to
        MOV     ES,AX                           ;memory block address
        PUSH    DS                              ;save data seg. register
        MOV     DS,AX                           ;set data to mem blk
        XOR     BX,BX                           ;memory offset ptr
        MOV     DX,BX                           ;zero record counter
        MOV     DI,BX                           ;zero mem blk offset
;------set first record as a blank line.
        MOV     [DI],DX                         ;mark first rec no = 0
        INC     DI                              ;adv mem word pointer
        INC     DI
        MOV     AX,2020h                        ;spaces to ax register
        MOV     [DI],AX                         ;place spaces in string
        MOV     SI,DI                           ;set source pointer
        INC     DI                              ;adv mem word pointer
        INC     DI
        CLD                                     ;auto inc SI & DI
        MOV     CX,13                           ;number words to move
        REP     MOVSW                           ;move type to Search
        MOV     [DI],DX                         ;mark end of string
        POP     DS                              ;restore data seg reg.
;------is the data file open ?
        MOV     AX,[DATHd]                      ;get data file handle
        CMP     AX,0                            ;is file open ?
        JNZ     RDI1                            ;if Yes continue
        STC                                     ;else set carry flag
        JMP     SHORT RDI4                      ;asn exit error.
RDI1:   CALL    GOTO_TOP                        ;reset file pointer
RDI2:   CALL    READ_DAT                        ;read one ID line
        JC      RDI3                            ;exit if EndOfFile
        ADD     BX,32                           ;ptr to next mem line
        MOV     DI,BX                           ;set ptr in DI register
        INC     DX                              ;advance record no
        MOV     [ES:DI],DX                      ;record no to ID string
        INC     DI                              ;advance mem  ptr past
        INC     DI                              ;id number
        MOV     AX,2020h                        ;two first two spaces
        MOV     [ES:DI],AX
        INC     DI                              ;advance mem ptr
        INC     DI                              ;
        MOV     [ES:DI],AX                      ;add second two spaces
        INC     DI                              ;at the beginning of
        INC     DI                              ;the ID string
        CLD                                     ;auto inc SI & DI
```

```
        MOV     SI,Offset FilBuf                ;source offset ptr
        MOV     CX,10                           ;move the ID to mem
        REP     MOVSW
        MOV     [ES:DI],AX                      ;place two spaces
        INC     DI                              ;after the end of
        INC     DI                              ;the ID string
        MOV     [ES:DI],AX                      ;place two spaces
        INC     DI                              ;after the end of
        INC     DI                              ;the ID string
        XOR     AX,AX                           ;hex zero to ax
        MOV     [ES:DI],AX                      ;endofstring marker
        JMP     SHORT RDI2
RDI3:   CLC                                     ;clear carry = data OK!
RDI4:   POP     ES
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    IDS_TO_MEMORY
;
;------Write memory block to NDX file.
;       Note: no errors are reported because this index file will
;             be computed each time if it is not stored on the disk.
;             If it is on the disk it is assumed to be sorted.
;             The file command will delete a index file if it is not
;             the correct size:  32(IDs + 2) = index size in bytes.
;
PROC    WRITE_INDEX_FILE
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,[MaxID]                      ;get number of ID's
        INC     AX                              ;ID + 2
        INC     AX
        MOV     CX,32                           ;each entry is 32 bytes
        MUL     CX                              ;AX = total bytes
        MOV     CX,AX                           ;no. of bytes to write
        MOV     AX,[VarSeg]                     ;get memory segment
        CMP     AX,0                            ;is memory allocated?
        JZ      WTI1                            ;if NO exit
        CALL    OPEN_NEW_INDEX_FILE             ;open a new rnk file.
        JC      WTI1
        MOV     BX,[NDXHd]
        PUSH    DS                              ;save data seg. address
        MOV     DS,AX                           ;memory blk addr. to ds
        MOV     AX,4000h                        ;write to file: func.
no.
        XOR     DX,DX                           ;segment offset = 0
        INT     21h                             ;write to the file
        POP     DS                              ;restore data seg. addr.
        JNC     WTI1                            ;exit if NO error
        CALL    DELETE_INDEX_FILE               ;else erase the file
WTI1:   CLC                                     ;clear carry flag
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    WRITE_INDEX_FILE
;
```

```
;
;------Read the index file into memonry block.
;       Input = None
;       Output = Carry flag if Dos error.
;
PROC    READ_INDEX_FILE
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,[NDXHd]                      ;get file handle
        CMP     AX,0                            ;is file open ?
        JZ      RIF2                            ;if NO Exit
        CALL    GOTO_TOP                        ;file ptr to beg of file
        JC      RIF2                            ;exit on DOS error
        MOV     BX,AX                           ;file handle to bx
        MOV     AX,[MaxID]                      ;get number of ID's
        INC     AX                              ;ID + 2
        INC     AX
        MOV     CX,32                           ;each entry is 32 bytes
        MUL     CX                              ;AX = total bytes
        MOV     CX,AX                           ;no. of bytes to read
        MOV     AX,[VarSeg]                     ;get memory segment
        CMP     AX,0                            ;is memory allocated?
        JZ      RIF2                            ;if NO exit
        PUSH    DS                              ;save data seg. address
        MOV     DS,AX                           ;memory blk addr. to ds
        MOV     AX,3F00h                        ;read file function no
        MOV     DX,0                            ;buffer ptr to DX
        INT     21h                             ;get bytes
        POP     DS
        JC      RIF2                            ;exit on DOS error?
        CMP     AX,CX                           ;was read complete ?
        JZ      RIF3                            ;if Yes normal return
RIF2:   STC                                     ;else set carry flag
RIF3:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    READ_INDEX_FILE
;
;       The follow registers hold the above variables:
;       AX = Gap;  BX = J;  CX = I;  DX = MaxRec; and  BP = temp storage
;
PROC    SORT_INDEX
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    DS
        PUSH    ES
        PUSH    BP
        MOV     DX,[MaxID]                      ;store MaxRec in DX
        MOV     AX,[VarSeg]                     ;get Index base segment
        MOV     DS,AX                           ;reassign the  DS & ES
        MOV     ES,AX                           ;to ptr to the Index.
        MOV     AX,DX                           ;Gap = MaxRec
        SHR     AX,1                            ;Gap = Gap Div by 2
STID1:  CMP     AX,0                            ;when Gap = 0 exit.
        JLE     STID4                           ;exit if <= 0
        MOV     CX,AX                           ;I is stored in CX
```

```
        INC     CX                                      ;I = Gap + 1
STID2:  MOV     BX,CX                                   ;J in BX
        SUB     BX,AX                                   ;J = I - Gap
        JZ      STID3                                   ;skip if J = 0
        JC      STID3                                   ;skip if J is < 0.
        CALL    COMPARE_IDS                             ;repeat until J = 0
STID3:  INC     CX                                      ;I = I + 1
        CMP     DX,CX                                   ;Is I < or = MaxRec
        JNC     STID2                                   ;If yes then loop.
        SHR     AX,1                                    ;Gap = Gap Div by 2
        JMP     SHORT STID1
STID4:  POP     BP                                      ;restore registers
        POP     ES
        POP     DS
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET                                             ;sort is complete.
;
;-------Compare and swap Index strings if needed.
;       Note: This is a subroutine of SORT_INDEX.  The index file record
;             length is 16 bytes.  The sort is made on the first 6 bytes.
;
;       Input = AX = Gap;  BX = J;   DS & ES point to the base of index file.
;       Output = AX = Gap; CX = I; and DX = MaxRec are returned on changed.
;                BX = J is discarded.
;
PROC    COMPARE_IDS
        PUSH    AX                                      ;save registers
        PUSH    CX
        PUSH    DX
        MOV     DX,AX                                   ;save Gap in DX
;       Compare the first 20 bytes of each index record
CMID1:  MOV     BP,BX                                   ;save J in BP
        ADD     AX,BX                                   ;AX = J + Gap
        MOV     CL,5                                    ;multiply by 32
        SHL     AX,CL                                   ;ptr to J+Gap in mem
        SHL     BX,CL                                   ;ptr to J in mem
        CLD                                             ;auto-inc SI, DI
        MOV     DI,AX                                   ;offset of J + Gap
        ADD     DI,6                                    ;skip rec no and spaces
        MOV     SI,BX                                   ;offset of J
        ADD     SI,6                                    ;skip rec no and spaces
        MOV     CX,20                                   ;byte counter
        REPE    CMPSB                                   ;compare strings
        JLE     CMID3                                   ;exit if < or =.
;       Swap the 32 bytes of index record if string A > string A+Gap
        MOV     DI,AX                                   ;offset of J + Gap
        MOV     SI,BX                                   ;offset of J
        MOV     CX,16                                   ;word counter
CMID2:  MOV     AX,[SI]                                 ;read word each str.
        MOV     BX,[DI]
        MOV     [SI],BX                                 ;write word each str.
        MOV     [DI],AX
        INC     DI                                      ;point to next word
        INC     DI
        INC     SI                                      ;point to next word
        INC     SI
        LOOP    CMID2                                   ;loop five times
        MOV     AX,DX                                   ;restore gap to AX
        MOV     BX,BP                                   ;restore J to BX
```

```
        SUB     BX,AX                                   ;J = J - gap
        JZ      CMID3                                   ;exit if J = 0.
        JNC     CMID1                                   ;continue if J > 0.
CMID3:  POP     DX                                      ;restore registers
        POP     CX
        POP     AX
        RET                                             ;return to Shell_Sort
ENDP    COMPARE_IDS
ENDP    SORT_INDEX
;
;
;-------Display IDs in Memory Directory
;       Input   none
;        byte [HiBar] = which bar is highlighed bar (1 to 14)
;        word [Topbar] = index item at top to screen (1 to MaxId)
;
;       Output  a 14 line of file names to the screen.
;       Note:  local variables:         AX = Id at top of screen display
;              BX = row/col             DH = hilite bar color attribute
;              CX = loop counter        DL = non hilite color attribute
;
PROC    DISPLAY_IDS
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,[TopBar]                             ;get TopBar
        MOV     DL,[Menu]                               ;normal attribute
        MOV     [Color],DL                              ;set default color
        MOV     DH,[Warning]                            ;hilite bar attirbute
        MOV     BX,0519h                                ;row 6/ col 32
        MOV     CX,15                                   ;number of rows
DID0:   CMP     CL,[HiBar]                              ;is this the hilite bar
        JNZ     DID1                                    ;<> 0 = no color change
        MOV     [Color],DH                              ;if yes color = warning
DID1:   CALL    ID_STR_OUT                              ;display one ID string
        MOV     [Color],DL                              ;if yes color = menu
DID2:   INC     BH                                      ;ptr to next row
        INC     AX                                      ;ptr to next dir entry
        LOOP    DID0                                    ;loop 14 times
        CALL    HIDE_CUR
        CLC                                             ;clear carry flag
DID3:   POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DISPLAY_IDS
;
;
;-------Send a 24 byte memory block asciiz string to the Screen
;       Input =  AX = memory index number  (0 to MaxID) 0 = blank entry
;                BX = row /col
;                [MaxID] = the number of IDs in the menory block
;                [VarSeg] = segment address of the base of the memory directory
;       Output = ASCIIZ string sent to the screen
;
PROC ID_STR_OUT
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
```

```
;-------compute memmory index offset
        CMP     [MaxID],AX                      ;is DirFile # OK ?
        JNC     ID80                            ;if in bounds jump
        XOR     AX,AX                           ;else make blank file
ID80:   MOV     CL,5                            ;shift 5 = times 32
        SHL     AX,CL                           ;multi by 32
        INC     AX                              ;skip record no word
        INC     AX
        MOV     CX,AX                           ;store memory offset
        MOV     AX,BX                           ;row/cow to ax;
        CALL    GOTOYX                          ;position cursor
        MOV     AX,[VarSeg]                     ;get the memory blk seg
        PUSH    DS                              ;save current data seg
        MOV     DS,AX                           ;set data seg to mem blk
        MOV     AX,CX                           ;restore memory offset
        CALL    DSTR_OUT                        ;send ID to the screen
        POP     DS                              ;restore orig. data seg
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    ID_STR_OUT
;
;
;
;     Mark all ID's in memory for printing.
;       Input = none
;       Output = none
;
; Format for the memory index:
;  word: ID number,4 spaces,20 character ID string, 4 spaces,2 hex zeros
;  Note: Record number 0 exists and is marked as record 0 but is never sorted.
;        Record number MaxId + 1 exist but is never marked or sorted.
;        Note: rec 0 and MaxId + 1 are add to allow quick binary searches.
;
PROC    TAG_ALL
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        MOV     AX,[VarSeg]                     ;set register to
        MOV     ES,AX                           ;memory block address
        MOV     CX,[MaxID]                      ;loop counter
        MOV     DI,4                            ;offset in record
        MOV     AL,'*'                          ;tag marker
        MOV     BX,32                           ;record length
TA1:    ADD     DI,BX
        MOV     [ES:DI],AL
        LOOP    TA1
        CLC                                     ;clear carry = data OK!
        POP     ES
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    TAG_ALL
;
; Format for the memory index:
;  word: ID number,4 spaces,20 character ID string, 4 spaces,2 hex zeros
```

```
;  Note: Record number 0 exists and is marked as record 0 but is never sorted.
;        Record number MaxId + 1 exist but is never marked or sorted.
;        Note: rec 0 and MaxId + 1 are add to allow quick binary searches.
;
PROC    UNTAG_ALL
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        MOV     AX,[VarSeg]                             ;set register to
        MOV     ES,AX                                   ;memory block address
        MOV     CX,[MaxID]                              ;loop counter
        MOV     DI,4                                    ;offset in record
        MOV     AL,' '                                  ;untag marker
        MOV     BX,32                                   ;record length
UT1:    ADD     DI,BX
        MOV     [ES:DI],AL
        LOOP    UT1
        CLC                                             ;clear carry = data OK!
        POP     ES
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    UNTAG_ALL
;
; Format for the memory index:
;  word: ID number,4 spaces,20 character ID string, 4 spaces,2 hex zeros
;  Note: Record number 0 exists and is marked as record 0 but is never sorted.
;        Record number MaxId + 1 exist but is never marked or sorted.
;        Note: rec 0 and MaxId + 1 are add to allow quick binary searches.
;       Tag highlighted bar ID in index file.
;        byte [HiBar] = which bar is highlighed bar (1 to 15)
;        word [TopBar] = index item at top to screen (1 to MaxId)
;
PROC    TAG
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        MOV     AX,[VarSeg]                             ;set register to
        MOV     ES,AX                                   ;memory block address
        MOV     CX,15                                   ;used to convert [HiBar]
        SUB     CL,[HiBar]                              ;off of highlight bar
        MOV     AX,[TopBar]                             ;get top of screen
        ADD     AX,CX                                   ;index record no to ax
        MOV     CL,32                                   ;record length
        MUL     CL                                      ;ax = memory blk offset
        ADD     AX,4                                    ;ptr to 4th byte
        MOV     DI,AX                                   ;offset in mem blk
        MOV     AL,'*'                                  ;tag marker
        MOV     [ES:DI],AL                              ;mark the record
        CALL    DISPLAY_IDS                             ;send to the screen
        CLC                                             ;clear carry = data OK!
        POP     ES
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
```

```
        RET
ENDP    TAG
;
; Format for the memory index:
;  word: ID number,4 spaces,20 character ID string, 4 spaces,2 hex zeros
;  Note: Record number 0 exists and is marked as record 0 but is never sorted.
;        Recond number MaxId + 1 exist but is never marked or sorted.
;        Note: rec 0 and MaxId + 1 are add to allow quick binary searches.
;       Tag highlighted bar ID in index file.
;        byte [HiBar] = which bar is highlighed bar (1 to 15)
;        word [TopBar] = index item at top to screen (1 to MaxId)
;
PROC    UNTAG
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        MOV     AX,[VarSeg]                     ;set register to
        MOV     ES,AX                           ;memory block address
        MOV     CX,15                           ;used to convert [HiBar]
        SUB     CL,[HiBar]                      ;off of highlight bar
        MOV     AX,[TopBar]                     ;get top of screen
        ADD     AX,CX                           ;index record no to ax
        MOV     CL,32                           ;record length
        MUL     CL                              ;ax = memory blk offset
        ADD     AX,4                            ;ptr to 4th byte
        MOV     DI,AX                           ;offset in mem blk
        MOV     AL,' '                          ;tag marker
        MOV     [ES:DI],AL                      ;mark the record
        CALL    DISPLAY_IDS                     ;send to the screen
        CLC                                     ;clear carry = data OK!
        POP     ES
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    UNTAG
;
;       Are any of the memory index IDs tagged ?
;       Input = none
;       Output = carry flag if no ID's are tagged
;
PROC    IS_TAG
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        MOV     AX,[VarSeg]                     ;set register to
        MOV     ES,AX                           ;memory block address
        MOV     CX,[MaxID]                      ;loop counter
        MOV     DI,4                            ;offset in record
        MOV     AL,' '                          ;untag marker
        MOV     BX,32                           ;record length
IT1:    ADD     DI,BX
        CMP     [ES:DI],AL
        LOOPZ   IT1
        JNZ     IT2
        CALL    CLEAR_MESSAGE
        MOV     CL,[Color]                      ;store original Color
```

```
        MOV     AL,[Warning]                            ;warning color
        MOV     [Color],AL                              ;set color
        MOV     AX,020Ah                                ;row/Col
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      ' Use the "Tag" command to mark the IDs to print.'
        db      ' Press Any Key ',0
        MOV     [Color],CL                               ;restore original color
        CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR
        STC
        JMP     SHORT IT3
IT2:    CLC                                              ;clear cf = continue
IT3:    POP     ES
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    IS_TAG
;
;       Find the next tagged Id in the memory index
;       Input = none
;       Output = AX = Dat record no. (1 to MaxId)
;                or carry flag in NO tagged found.
;
PROC    FIND_TAG
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        MOV     AX,[VarSeg]                             ;set register to
        MOV     ES,AX                                   ;memory block address
        MOV     CX,[MaxID]                              ;loop counter
        MOV     DI,4                                    ;offset in record
        MOV     AL,'*'                                  ;untag marker
        MOV     BX,32                                   ;record length
        XOR     DX,DX                                   ;index counter to 0
FT1:    ADD     DI,BX
        INC     DX                                      ;count = count + 1
        CMP     [ES:DI],AL                              ;is it tagged ?
        JZ      FT2                                     ;if yes exit loop
        LOOP    FT1                                     ;look at next ID
        CALL    DISPLAY_IDS                             ;show IDs
        STC                                             ;carry flag = not found
        JMP     FT3                                     ;exit NOT found
FT2:    SUB     DI,4                                    ;ptr to dat rec no
        MOV     AX,[ES:DI]                              ;get DAT RecNo
        MOV     [TopBar],DX                             ;set display variable
        MOV     BYTE PTR [HiBar],15                     ;move bar to top
        ADD     DI,4                                    ;ptr to tag position
        MOV     BYTE PTR [ES:DI],' '                    ;untag the ID
        CALL    DISPLAY_IDS                             ;show IDs
        CLC
FT3:    POP     ES
        POP     DX
        POP     CX
        POP     BX
        RET
ENDP    FIND_TAG
;
```

```
        .CODE
;
;-------Print all factors for the SLDI.
;       Input = BX = pointer to Type 6 (factor name) in CFGTbl.
;               CX = row position on page
;       Output = AX = new row position on page
;       Local variables:
;       BX = Start Ptr to CFGTbl
;       CX = page row in hex (0 - 44)
;
PROC    PRINT_FACTOR
        PUSH    BX
        PUSH    CX
        PUSH    DX
        XOR     AL,AL                                   ;zero to al
        MOV     [Tint],AL                               ;set tint light
;------draw scale line and top line of box
        MOV     AL,22h                                  ;col in hex
        MOV     AH,CL                                   ;row in hex
        CALL    HPGOTOYX                                ;set positon
        JC      PRF4                                    ;exit on error
        MOV     AX,Offset Top                           ;ptr to draw top string
        CALL    PRINT_STRING                            ;draw first 2 lines
        JC      PRF4                                    ;exit on printer error
        INC     CX                                      ;adv row counter
        INC     CX                                      ;adv row counter
        CALL    PRINT_NAME                              ;prt factor name
        JC      PRF5                                    ;exit on abort cmd
        JMP     SHORT PRF2                              ;do not adv. row count
PRF1:   INC     CX                                      ;adv row counter
PRF2:   ADD     BX,4                                    ;adv config pointer
        CMP     WORD PTR [BX],6                         ;is this an answer file?
        JNC     PRF4                                    ;is NO then done.
        CALL    READ_RNK_REC                            ;read 7 bytes to OutBuf
        JC      PRF5
        XOR     AL,AL                                   ;zero to al
        MOV     [Others],AL                             ;mark Others bar TRUE
;------print type of line:  Self, Peers, Superiors, or Subordinates
        MOV     AL,22h                                  ;col in hex
        MOV     AH,CL                                   ;row in hex
        CALL    HPGOTOYX                                ;set positon
        JC      PRF5                                    ;exit on error
        MOV     AX,Offset Subor                         ;ptr to subordinate line
        CMP     WORD PTR [BX],1                         ;is it a subordinate?
        JZ      PRF3                                    ;if YES print it
        MOV     AX,Offset Super                         ;ptr to superior line
        CMP     WORD PTR [BX],2                         ;is it a superior?
        JZ      PRF3                                    ;if YES print it
        MOV     AX,Offset Peer                          ;ptr to peer line
        CMP     WORD PTR [BX],3                         ;is it a peer?
        JZ      PRF3                                    ;if YES print it
;------is this a self bar or self data for an Other bar ?
        MOV     AX,[BX + 4]                             ;get the next file type
        CMP     AX,[BX]                                 ;are they both self's
        JNZ     PRF2                                    ;if NO then skip self
        MOV     AL,0FFh                                 ;
        MOV     [Others],AL                             ;mark Others bar FALSE
        MOV     AX,Offset Self                          ;else must be a self bar
PRF3:   CALL    PRINT_STRING                            ;ptr box side and name
        JC      PRF5                                    ;exit on printer error
        CALL    PRINT_RANGE                             ;print range of means
        JC      PRF5                                    ;exit of abort cmd
```

```
        CALL    PRINT_PERCENTILES                       ;25th, 50th, and 75th
        JC      PRF5
        CALL    PRINT_MEANS                             ;print mean scores
        JC      PRF5                                    ;exit of abort cmd
        JMP     SHORT PRF1
;------print the bottom line of the box
PRF4:   MOV     AL,22h                                  ;col in hex
        MOV     AH,CL                                   ;row in hex
        CALL    HPGOTOYX                                ;set positon
        JC      PRF5                                    ;exit on error
        MOV     AX,Offset Bot                           ;ptr to draw top string
        CALL    PRINT_STRING                            ;draw first 2 lines
        JC      PRF5                                    ;exit on printer error
        INC     CX                                      ;adv row counter
        INC     CX                                      ;adv row counter
        CLC
PRF5:   MOV     AX,CX                                   ;return new row count
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        RET
ENDP    PRINT_FACTOR
;
;
;-------Print the Factor Name.
;       Input = BX = ptr to Factor Name data type
;               CX = row
;       Output = Factor name to the screen
;       This routine will split name if larger the 24 characters.
;       It aborts the name if it is longer than 36 characters
;       or a single word with more then 24 letters.
;
PROC    PRINT_NAME
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,Offset BON                           ;Bold ON string
        CALL    PRINT_STRING                            ;turn bold on
        JC      PFN6
        MOV     AX,[BX + 2]                             ;ptr to factor name
        CALL    COPY_HEAP_STR                           ;move to [FilBuf]
        CMP     AX,37                                   ;is name length > 36?
        JNC     PFN5                                    ;if YES do not display
        CMP     AX,25                                   ;is name length > 24 ?
        JC      PFN4                                    ;if NO display string
;------look for a place to divide the factor name into two lines
        DEC     AX                                      ;lenth = length -1
PFN1:   MOV     BX, Offset FilBuf                       ;ptr to beg if string
        MOV     DX,BX                                   ;save start ptr in dx
        ADD     BX,AX                                   ;ptr to last char in str
        CMP     BYTE PTR [BX],' '                       ;is this a space ?
        JZ      PFN2                                    ;if yes goto next test
        CMP     BYTE PTR [BX],'/'                       ;is it a backslash ?
        JZ      PFN2                                    ;if YES goto next test
        DEC     AX                                      ;lenth = length -1
        JZ      PFN5                                    ;exit if can not divide
        JMP     SHORT PFN1                              ;else look a next char
;------have we shorten it enough ?
PFN2:   CMP     AX,25                                   ;is name length > 24 ?
        JC      PFN3                                    ;if YES split name
        DEC     AX                                      ;lenth = length -1
```

```
        JMP     PFN1                                    ;if NO keep looking
;------split the factor name
PFN3:   XOR     AL,AL                                   ;0 = endofstring marker
        MOV     [BX],AL                                 ;mark end of first str
;------display 2nd half of string
        MOV     AL,8h                                   ;col in hex
        MOV     AH,CL                                   ;row in hex
        INC     AH                                      ;print on next row
        CALL    HPGOTOYX                                ;set positon
        JC      PFN6                                    ;exit on error
        INC     BX                                      ;ptr to beg. of 2nd half
        MOV     AX,BX                                   ;ptr to string
        CALL    PRINT_STRING                            ;prt wnd half of name
        JC      PFN6
;------display 1st half of string
PFN4:   MOV     AL,8h                                   ;col in hex
        MOV     AH,CL                                   ;row in hex
        CALL    HPGOTOYX                                ;set positon
        JC      PFN6                                    ;exit on error
        MOV     AX,Offset FilBuf                        ;ptr to string
        CALL    PRINT_STRING                            ;prt wnd half of name
        JC      PFN6
PFN5:   MOV     AX,Offset BOFF                          ;Bold Off string
        CALL    PRINT_STRING                            ;turn bold off
PFN6:   POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRINT_NAME
;

;
;-------Print a report for each tagged ID number in the memory index file.
;       Input = (one or more ID's are tagged)
;       Output = None
;       Local variables:
;       BX = Start Ptr in the CFGTbl
;       CX = starting page row in hex (0, 1, or 2)
;       DX = Stop Ptr in the CfgTbl (next start ptr)
;       Note: [EOF] is being used to mark the end of the config table.
PROC    PRINT_REPORTS
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    INITIALIZE_HP                           ;hp to portrait mode
        JC      PRA4                                    ;exit if printer error
        CALL    PRINT_INSTRU                            ;message box to screen
;------locate next tagged ID
PRA1:   CALL    FIND_TAG                                ;get DAT RecNo in ax
        JC      PRA3                                    ;cf = no more tags
        CALL    READ_DAT_REC                            ;ax=RecNo (1 to MaxId)
        JC      PRA5                                    ;exit on DOS error
        MOV     AX,[RNKHd]                              ;set rank file ptr
        CALL    GOTO_TOP                                ;to top of file.
        JC      PRA5                                    ;exit on DOS error
        XOR     AX,AX                                   ;zero to ax
        MOV     [EOF],AL                                ;mark EndOfFile False
        CALL    PROGRESS_MESSAGE                        ;inform user of prog.
        MOV     DX,Offset CFGTbl                        ;assign start ptr
PRA2:   CALL    GET_PAGE_INFO                           ;get row & stop ptr
```

```
        JC      PRA4                                    ;exit on config error
        MOV     CX,AX                                   ;starting row in cx
        XCHG    BX,DX                                   ;bx=start dx=stop
        CALL    SET_TIME                                ;set [time] ticks
        CALL    PRINT_PAGE                              ;print this page
        JC      PRA4                                    ;exit on abort cmd
        CALL    CHECK_TIME                              ;check timing ticks
        CMP     BYTE PTR [EOF],0                        ;end of config table ?
        JZ      PRA2                                    ;if NO print next page
        JMP     SHORT PRA1                              ;goto next ID report
;------normal exit point
PRA3:   CALL    RESTORE_HP                             ;reset to normal defaults
        CLC                                             ;normal exit
        JMP     SHORT PRA5
;------abort exit point
PRA4:   CALL    RESTORE_HP
        STC                                             ;abort or error exit
PRA5:   POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRINT_REPORTS
;
;
;-------Print one page use the LPT port and a HP LaserJet printer.
;       Input =  BX = Start Ptr in the CFGTbl
;                CX = starting row in hex (0, 1, 2, or 3)
;                DX = Stop Ptr in the CfgTbl
;       Output = Cary flag = abort printing
;
PROC    PRINT_PAGE
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    PRINT_TITLE                             ;print first 5 lines
        JC      PP4                                     ;exit on abort cmd
        ADD     CX,5                                    ;advance row count
;------advance BX until factor name or BX = DX
PP1:    CMP     BX,DX                                   ;is this the stop ptr
        JNC     PP3                                     ;if Yes then EndOfPage
        CMP     WORD PTR [BX],6                         ;is factor name ?
        JZ      PP2                                     ;if YES print factor
        ADD     BX,4                                    ;advance config ptr
        JMP     SHORT  PP1                              ;check next type
PP2:    CALL    PRINT_FACTOR                            ;ret. new row ct in AX
        JC      PP4                                     ;exit on abort cmd
        MOV     CX,AX                                   ;save new row count
        ADD     BX,4                                    ;adv past factor name
        JMP     SHORT PP1                               ;loop until BX = DX
PP3:    CALL    PRINT_BOTTOM
        JC      PP4                                     ;exit on abort cmd
        CALL    EJECT
        JC      PP4                                     ;exit of abort cmd
        CLC                                             ;clear carry flag
PP4:    POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRINT_PAGE
```

```
;
;
;-------Print a title on the HP laser.
;       Input = CX = starting row
;       Output = AX = new starting row
PROC    PRINT_TITLE
        PUSH    BX                              ;save registers
        PUSH    CX
        PUSH    DX
        MOV     AH,CL                           ;row position
        MOV     AL,21h                          ;column position
        CALL    HPGOTOYX
        JNC     PTA
        JMP     PT5                             ;exit on printer error
PTA:    MOV     AX,Offset Header                ;ptr Title
        CALL    PRINT_STRING                    ;string holds Bold ON
        JNC     PTB
        JMP     PT5                             ;exit on printer error
PTB:    INC     CL
        INC     CL
        MOV     AH,CL                           ;row position
        MOV     AL,8h                           ;column position
        CALL    HPGOTOYX
        JNC     PTC
        JMP     PT5                             ;exit on printer error
PTC:    CALL    COPY_ID                         ;place ID in print str
        MOV     AX,Offset IDstr                 ;ptr to data string
        CALL    PRINT_STRING                    ;print "ID :"
        JC      PT5
;------print group title
        MOV     SI,BX                           ;config ptr to si
        ADD     SI,4
PT4:    SUB     SI,4                            ;decreass type ptr
        CMP     WORD PTR [SI],7                 ;is it group title?
        JNZ     PT4                             ;if NO loop
        INC     SI                              ;adv ptr to get
        INC     SI                              ;offset in heap
        MOV     AX,[SI]                         ;load pt to group title
        CALL    COPY_HEAP_STR                   ;move to [FilBuf]
        SHR     AL,1                            ;divide length by 2
        MOV     AH,AL                           ;save in ah register
        MOV     AL,53                           ;center column position
        SUB     AL,AH                           ;column position
        MOV     AH,CL                           ;row position
        CALL    HPGOTOYX
        JC      PT5                             ;exit on printer error
        MOV     AX,Offset FilBuf
        CALL    PRINT_STRING
        JC      PT5                             ;exit on printer error
;------print scored date
        MOV     AL,53h                           ;col hex
        MOV     AH,CL
        CALL    HPGOTOYX
        JC      PT5                             ;exit on printer error
        MOV     AX,Offset DTStr
        CALL    PRINT_STRING
        JC      PT5
;-----print "FACTORS:" and turn off bold print
        INC     CL
        INC     CL
        MOV     AH,CL                           ;row position
        MOV     AL,8h                           ;column position
```

```
        CALL    HPGOTOYX
        JC      PT5                     ;exit on printer error
        MOV     AX,Offset DIStr
        CALL    PRINT_STRING            ;string holds Bold OFF
        JC      PT5
        MOV     AH,CL                   ;row position
        MOV     AL,48                   ;column position
        CALL    HPGOTOYX
        JC      PT5                     ;exit on printer error
;------ is it a POS or NEG report
        MOV     SI,BX                   ;config ptr to si
        SUB     SI,4
PT1:    ADD     SI,4                    ;advance type ptr
        CMP     WORD PTR [SI],5         ;is it positive ?
        JZ      PT2                     ;if YES stop loop
        CMP     WORD PTR [SI],4         ;is it negative ?
        JNZ     PT1                     ;if NO loop
        MOV     AX,Offset NegT          ;ptr to derailment str
        JMP     SHORT PT3               ;jump to print the str
PT2:    MOV     AX,Offset PosT          ;ptr to positive str
PT3:    CALL    PRINT_STRING            ;print the string
        JC      PT5                     ;exit on printer error
PT5:    MOV     AX,CX                   ;return row position
        POP     DX                      ;restore registers
        POP     CX
        POP     BX
        RET
ENDP    PRINT_TITLE
;
;
;-------Print the bottom instruction box.
;       Input = CX = starting row
;       Output = AX = new starting row
PROC    PRINT_BOTTOM
        PUSH    BX                      ;save registers
        PUSH    CX
        PUSH    DX
        MOV     AH,CL                   ;row position
        MOV     AL,22h                  ;column position
        CALL    HPGOTOYX
        JC      PB1                     ;exit on printer error
        MOV     AX,Offset Inform        ;bottom information box
        CALL    PRINT_STRING
PB1:    MOV     AX,CX                   ;return row position
        POP     DX                      ;restore registers
        POP     CX
        POP     BX
        RET
ENDP    PRINT_BOTTOM
;
;-------Copy ID from [F1Buf] to [ID] in print string
;       Input = none
;       Output = none
PROC    COPY_ID
        PUSH    AX                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        MOV     AX,DS                   ;make ES = DS
        MOV     ES,AX
        MOV     SI,Offset SF1Buf        ;source ptr
```

```
        MOV     DI,Offset ID                            ;destination ptr
        MOV     CX,10                                   ;number of words
        CLD                                             ;auto inc SI & DI
        REP     MOVSW                                   ;copy all 20 bytes
        CLC
        POP     ES
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    COPY_ID
;
         .CODE
;
;-------Send ASCII string to the Line Printer at port [LPT]
;       Input = AX pointer to beginning of string in data section
;               CH = number of tries if busy   CL = store char
;       Output = Carry flag = abort printing
PROC    PRINT_STRING
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     BX,AX                                   ;ptr to ASCII string
        XOR     CH,CH                                   ;zero loop counter
;------Check keyboard buffer to see if the <Esc> key been pressed?
PS1:    MOV     AX,0600h                                ;DOS function # 6
        MOV     DL,0FFh                                 ;read char from key-
        INT     21h                                     ;board buffer.
        JZ      PS2                                     ;NO key pressed continue
        CMP     AL,1Bh                                  ;was it the <ESC> key?
        JNZ     PS2                                     ;if NO continue
        CALL    PRT_ERROR3                              ;if YES inform user
        JC      PS8                                     ;carry flag = abort
;------get character to be sent to LPT port
PS2:    MOV     AL,[BX]                                 ;load Char to send
        CMP     AL,0                                    ;is this end of string?
        JZ      PS8                                     ;if yes normal exit.
;------send character to assigned LPT port
        MOV     AH,0                                    ;BIOS function number
        MOV     DX,[LPT]                                ;get LPT port assign.
        INT     17h                                     ;get port status
        CMP     BYTE PTR [DeBug],0                      ;is debug ON ?
        JZ      PS3                                     ;if NO goto next test
        CALL    SHOW_AH                                 ;bitmap of AH to screen
;------test bit 5 of 8.  If bit 5 = 0 then no power.
PS3:    TEST    AH,10h                                  ;Is printer powered up?
        JNZ     PS5                                     ;OK!<>0 goto next test
        CALL    PRT_ERROR1                              ;display error message
        JC      PS8                                     ;carry flag = abort
        CALL    PROGRESS_MESSAGE                        ;inform user of progress
        JMP     SHORT PS1                               ;send same char again
;------test bit 4 & 6 of 8.  bit 4 = I/O error; 6 = printer out of paper.
PS5:    XOR     CH,CH                                   ;loop counter to zero
        TEST    AH,28h                                  ;I/O or out of paper?
        JZ      PS6                                     ;if NO send char
        CALL    PRT_ERROR2                              ;if YES tell user.
        JC      PS8                                     ;cf = abort
        CALL    PROGRESS_MESSAGE                        ;inform user of progress
        JMP     SHORT PS1                               ;send same char again
;------test bit 1 of 8.  If bit 1 = 1 then printer time-out
```

```
PS6:    TEST    AH,01                   ;is printer time-out?
        JZ      PS7                     ;if NO send next char
        CALL    PRT_ERROR4              ;inform user or timeout
        JC      PS8                     ;cf = abort else
        CALL    PROGRESS_MESSAGE        ;inform user of progress
        JMP     SHORT PS1               ;send same char again
PS7:    INC     BX                      ;point to next char
        JMP     SHORT PS1               ;loop until finished
PS8:    POP     DX                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
;
;       Input = none
;       Output = carry flag = abort printing
PROC    PRT_ERROR1
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    CLEAR_MESSAGE           ;empty message line
        MOV     CL,[Color]              ;store original Color
        MOV     AL,[Warning]            ;warning color
        MOV     [Color],AL              ;set color
        MOV     AX,020Bh                ;row 3/Col 12
        CALL    GOTOYX                  ;set cursor
        CALL    CSTR_OUT                ;display warning
        db      " Printer is off line.  Do you want to try again ? "
        db      " Y/N ",0
        MOV     [Color],CL              ;restore original color
PRE1:   CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR
        AND     AL,5Fh                  ;turn off bits 6 & 8
        CMP     AL,'N'                  ;is it No?
        JZ      PRE4                    ;if yes exit
PRE2:   CMP     AL,'Y'                  ;is it Yes?
        JNZ     PRE3                    ;if not continue
        CALL    CLEAR_MESSAGE           ;empty message line
        CLC                             ;clear carry flag
        JMP     SHORT PRE5              ;exit
PRE3:   CALL    ERR_SOUND
        JMP     SHORT PRE1
PRE4:   CALL    CLEAR_MESSAGE           ;empty message line
        STC                             ;set carry flag
PRE5:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRT_ERROR1
;
;       Input = none
;       Output = carry flag = abort printing
PROC    PRT_ERROR2
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    CLEAR_MESSAGE
        MOV     CL,[Color]              ;store original Color
```

```
        MOV     AL,[Warning]                            ;warning color
        MOV     [Color],AL                              ;set color
        MOV     AX,0207h                                ;row 3/Col 12
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      " Printer Error. Check the paper. Do you want to continue ? "
        db      " Y/N ",0
        MOV     [Color],CL                               ;restore original color
PRR1:   CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR
        AND     AL,5Fh                                  ;turn off bit 6 & 8
        CMP     AL,'N'                                  ;is it No?
        JZ      PRR4                                    ;if yes exit
PRR2:   CMP     AL,'Y'                                  ;is it Yes?
        JNZ     PRR3                                    ;if not continue
        CALL    CLEAR_MESSAGE                           ;empty message line
        CLC                                             ;clear carry flag
        JMP     SHORT PRR5                              ;exit
PRR3:   CALL    ERR_SOUND
        JMP     SHORT PRR1
PRR4:   CALL    CLEAR_MESSAGE                           ;empty message line
        STC                                             ;set carry flag
PRR5:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRT_ERROR2
;
;
;       Input = none
;       Output = carry flag = abort printing
PROC    PRT_ERROR3
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    CLEAR_MESSAGE
        MOV     CL,[Color]                              ;store original Color
        MOV     AL,[Warning]                            ;warning color
        MOV     [Color],AL                              ;set color
        MOV     AX,020Bh                                ;row 3/Col 12
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      " Do you want to ABORT the print instructions ? "
        db      " Y/N ",0
        MOV     [Color],CL                               ;restore original color
PEE1:   CALL    HIDE_CUR
        CALL    GET_CHAR
        AND     AL,5Fh                                  ;turn off bit 6 & 8
        CMP     AL,'N'                                  ;is it No?
        JZ      PEE4                                    ;if yes exit
PEE2:   CMP     AL,'Y'                                  ;is it Yes?
        JNZ     PEE3                                    ;if not continue
        CALL    CLEAR_MESSAGE                           ;empty message line
        STC                                             ;set carry flag = abort
        JMP     SHORT PEE5                              ;exit
PEE3:   CALL    ERR_SOUND
        JMP     SHORT PEE1
PEE4:   CALL    CLEAR_MESSAGE                           ;empty message line
        CLC                                             ;clear cf = continue
```

```
PRES:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRT_ERROR3
;
;
;       Input = none
;       Output = carry flag = abort printing
PROC    PRT_ERROR4
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    CLEAR_MESSAGE
        MOV     CL,[Color]                      ;store original Color
        MOV     AL,[Warning]                    ;warning color
        MOV     [Color],AL                      ;set color
        MOV     AX,0207h                        ;row 3/Col 12
        CALL    GOTOYX                          ;set cursor
        CALL    CSTR_OUT                        ;display warning
        db      " Printer Time-Out.  Press any key to try again or <Esc> "
        db      "to abort. ",0
        MOV     [Color],CL                       ;restore original color
        CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR
        CMP     AL,1Bh                          ;is it <Esc>
        JNZ     RPP1                            ;if yes exit
        STC                                     ;set carry flag = abort
        JMP     SHORT RPP2                      ;exit
RPP1:   CALL    CLEAR_MESSAGE                   ;empty message line
        CLC                                     ;clear cf = continue
RPP2:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRT_ERROR4
ENDP    PRINT_STRING
;
;
;-------Print the data points on each bar.
;       Input = assumes byte value in SF1Buf.
;               offset in SF1Buf can be found in word [Outbuf + 5]
;               if this is a Self bar [Others] will = 0
;               CL = row position
;       Output = Cary flag = abort printing
;
PROC    PRINT_MEANS
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------get pointer to mean position if SF1Buf
        MOV     BX,Offset OutBuf + 5            ;offset to get offset
        MOV     AX,[BX]                         ;load offset
        MOV     BX,Offset SF1Buf                ;load base ptr
        ADD     BX,AX                           ;bx points to mean
        CMP     BYTE PTR [Others],0             ;is this a self bar?
        JNZ     CP1                             ;if yes goto plot_self
```

```
;------plot the Others value
        CALL    PLOT_OTHERS                     ;if No plot others
        JC      CP2                             ;exit on abort cmd
        DEC     BX                              ;pointer to self mean
CP1:    CALL    PLOT_SELF                       ;plot the self mean
        JC      CP2                             ;exit on abort cmd
        CLC
CP2:    POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRINT_MEANS
;
;
;-------Print the Others data point.
;       Input = BX = ptr to mean score.
;               CL = row postion
;       Output = Cary flag = abort printing
;
PROC    PLOT_OTHERS
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------set cursor
        MOV     AL,30h                          ;column position
        MOV     AH,CL                           ;row position
        CALL    HPGOTOYX                        ;set positon
        JC      PLO6                            ;exit on printer error
        MOV     AL,[BX]                         ;get mean score
        CMP     AL,0Ah                          ;is it to small?
        JC      PLO5                            ;if yes exit no error
        CMP     AL,51                           ;if it to large ?
        JNC     PLO5                            ;if yes exit no error
        XOR     AH,AH                           ;convert to 16 bits
        MOV     CL,10                           ;divisor
        DIV     CL                              ;ah=tenths  al=units
        MOV     DX,AX                           ;save values in DX
        XOR     CX,CX                           ;zero to loop counter
        MOV     CL,DL                           ;units to cl
        DEC     CX                              ;scale starts at 1 not 0
        JZ      PLO2                            ;skip if units = 0
        MOV     AX,Offset NextUn                ;ptr to next unit str
PLO1:   CALL    PRINT_STRING                    ;advance to next tens
        JC      PLO6                            ;exit on printer error
        LOOP    PLO1                            ;loop until tens = 0
PLO2:   XOR     CX,CX                           ;zero to loop counter
        ADD     CL,DH                           ;get units digit
        JZ      PLO4                            ;if zero skip units
        MOV     AX,Offset NextTn                ;ptr to next units str
PLO3:   CALL    PRINT_STRING                    ;advance  to next unit
        JC      PLO6                            ;exit on printer error
        LOOP    PLO3                            ;loop until units = 0
PLO4:   MOV     AX,Offset Other                 ;ptr to Other string
        CALL    PRINT_STRING                    ;plot point in chart
        JC      PLO6                            ;exit on printer error
PLO5:   CLC
PLO6:   POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
```

```
        RET
ENDP    PLOT_OTHERS
;

;-------Print the Self data point.
;       Input = BX = ptr to mean score.
;               CL = row postion
;       Output = Cary flag = abort printing
;
PROC    PLOT_SELF
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------set cursor
        MOV     AL,30h                          ;column position
        MOV     AH,CL                           ;row position
        CALL    HPGOTOYX                        ;set positon
        JC      PLS6                            ;exit on printer error
;------get data point
        MOV     AL,[BX]                         ;get mean score
        CMP     AL,0Ah                          ;is it to small?
        JC      PLS5                            ;if yes exit no error
        CMP     AL,51                           ;if it to large ?
        JNC     PLS5                            ;if yes exit no error
        XOR     AH,AH                           ;convert to 16 bits
        MOV     CL,10                           ;divisor
        DIV     CL                              ;ah=tenths  al=units
        MOV     DX,AX                           ;save values in DX
        XOR     CX,CX                           ;zero to loop counter
        MOV     CL,DL                           ;units to cl
        DEC     CX                              ;scale starts at 1 not 0
        JZ      PLS2                            ;skip if units = 0
        MOV     AX,Offset NextUn                ;ptr to next unit str
PLS1:   CALL    PRINT_STRING                    ;advance to next tens
        JC      PLS6                            ;exit on printer error
        LOOP    PLS1                            ;loop until tens = 0
PLS2:   XOR     CX,CX                           ;zero to loop counter
        ADD     CL,DH                           ;get units digit
        JZ      PLS4                            ;if zero skip units
        MOV     AX,Offset NextTn                ;ptr to next units str
PLS3:   CALL    PRINT_STRING                    ;advance  to next unit
        JC      PLS6                            ;exit on printer error
        LOOP    PLS3                            ;loop until units = 0
PLS4:   MOV     AX,Offset Point                 ;ptr to Point string
        CALL    PRINT_STRING                    ;plot point in chart
        JC      PLS6                            ;exit on printer error
PLS5:   CLC
PLS6:   POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PLOT_SELF
;




;       Is the data file, report type and LPT port ready?
;       Input = None
```

```
;       Output = Carry flag if Not Ready.
PROC    IS_PRINT
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------is data file open ?
        MOV     BX,[DATHd]                              ;get file handle
        CMP     BX,0                                    ;is a file open?
        JNZ     PR1
        CALL    FILE_ERR
        JMP     SHORT PR4
;------were the rank file open ?
PR1:    CMP     WORD PTR [RNKHd],0                      ;is rank file open ?
        JNZ     PR2                                     ;if Yes goto next test
        CALL    RANK_ERR                                ;else inform user
        JMP     SHORT PR4
;------is the printer on line?
PR2:    CALL    ON_LINE
        JNC     PR3
        CALL    LPT_ERR
        JMP     SHORT PR4
;------read index file into memory or create the index file.
PR3:    CALL    GET_INDEX                               ;read index into memory
        JNC     PR5                                     ;jump if NO error.
        CALL    INDEX_ERR                               ;should not happen!!!
PR4:    STC                                             ;set error flag
PR5:    POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    IS_PRINT
;
;       Input = none
;       Output = none
PROC    LPT_ERR
        CALL    CLEAR_MESSAGE
        MOV     AL,[Warning]                            ;warning color
        MOV     CL,[Color]                              ;save original color
        MOV     [Color],AL                              ;set color
        MOV     AX,0207h                                ;row 3/Col 8
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      ' Printer Not On Line!  Check power or LPT assignment.'
        db      '  Press Any Key. ',0
        MOV     [Color],CL                               ;restore original color
        CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR
        RET
ENDP    LPT_ERR
;
;       Input = none
;       Output = none
PROC    INDEX_ERR
        CALL    CLEAR_MESSAGE
        MOV     AL,[Warning]                            ;warning color
        MOV     CL,[Color]                              ;save original color
        MOV     [Color],AL                              ;set color
        MOV     AX,0207h                                ;row 3/Col 8
        CALL    GOTOYX                                  ;set cursor
```

```
        CALL    CSTR_OUT                                        ;display warning
        db      ' Error Indexing the IDs!  Call 703-751-3027 for help.'
        db      '  Press Any Key. ',0
        MOV     [Color],CL                                      ;restore original color
        CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR
        RET
ENDP    INDEX_ERR
;
;       Input = none
;       Output = none
PROC    RANK_ERR
        CALL    CLEAR_MESSAGE
        MOV     AL,[Warning]                                    ;warning color
        MOV     CL,[Color]                                      ;save original color
        MOV     [Color],AL                                      ;set color
        MOV     AX,0207h                                        ;row 3/Col 8
        CALL    GOTOYX                                          ;set cursor
        CALL    CSTR_OUT                                        ;display warning
        db      " Use the 'Rank' command to compute the percentiles."
        db      '  Press Any Key. ',0
        MOV     [Color],CL                                      ;restore original color
        CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR
        RET
ENDP    RANK_ERR
;
;       Input = none
;       Output = none
PROC    PROGRESS_MESSAGE
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AL,[Menu]                                       ;menu color
        MOV     CL,[Color]                                      ;save original color
        MOV     [Color],AL                                      ;set color
        MOV     AX,0107h                                        ;row 3/Col 8
        CALL    GOTOYX                                          ;set cursor
        CALL    CSTR_OUT                                        ;display string
        db      ' Please wait .....   Printing report for ID : ',0
        MOV     AX, Offset SF1Buf                               ;ptr to string
        MOV     CX,20                                           ;number of bytes
        CALL    SUB_DSTR_OUT                                    ;display 20 chars
        MOV     [Color],CL                                      ;restore original color
        CALL    HIDE_CUR
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PROGRESS_MESSAGE
;

;
;-------Request Printer Port Status
;       Input = Assign port in [LPT]  0 - 2
;       Output = Carry Flag = port not ready
PROC    ON_LINE
        PUSH    AX                                              ;save registers
```

```
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,0200h                ;get status function no
        MOV     DX,[LPT]                ;ptr to port
        INT     17h                     ;request status
        AND     AH,10h                  ;Is printer ready ?
        JNZ     ISR1                    ;0 means printer error
        STC                             ;set carry flag
ISR1:   POP     DX                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    ON_LINE

;-------Eject paper on HP laser.
;       Input = None
;       Output = None
PROC    EJECT
        PUSH    AX                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,Offset FFeed
        CALL    PRINT_STRING
        POP     AX                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    EJECT
;
;-------Initialize the HP laser.
;       Input = None
;       Output = None
PROC    INITIALIZE_HP
        PUSH    AX                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,Offset Init
        CALL    PRINT_STRING
        POP     DX                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    INITIALIZE_HP
;
;-------Restore default setting to the HP laser.
;       Input = None
;       Output = None
PROC    RESTORE_HP
        PUSH    AX                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    RESTORE_MESS            ;inform user
        MOV     SI,Offset Rest          ;ptr to ASCII string
;------get character to be sent to LPT port
RR1:    MOV     AL,[SI]                 ;load Char to send
```

```
        CMP     AL,0                                    ;is this end of string?
        JZ      RR4                                     ;if yes normal exit.
;------send character to assigned LPT port
        XOR     AH,AH                                   ;0 = BIOS function No.
        MOV     DX,[LPT]                                ;get LPT port assign.
        INT     17h                                     ;send char to printer
;------test bit 5 of 8.  If bit 5 = 0 then no power.
        TEST    AH,10h                                  ;Is printer powered up?
        JZ      RR4                                     ;exit if NO
;------test bit 1 of 8.  If bit 1 = 1 then printer time-out
        TEST    AH,01                                   ;is printer time-out?
        JNZ     RR4                                     ;if YES then exit
;------pause 1/3 second or up to 1/6 second once each hour.
RR2:    MOV     AH,0                                    ;function number
        INT     1Ah                                     ;get DOS clock ticks
        MOV     BX,DX                                   ;save ticks in CX
        MOV     AX,3                                    ;18.2 ticks per second
        ADD     BX,AX                                   ;add 15 seconds
        JC      RR2                                     ;loop if over flow
RR3:    MOV     AH,0                                    ;function number
        INT     1Ah                                     ;get DOS clock ticks
        CMP     DX,BX                                   ;has time run out ?
        JC      RR3                                     ;if not loop until done
        INC     SI                                      ;point to next char
        JMP     SHORT RR1                               ;loop until finished
RR4:    CLC
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
;
;       Input = none
;       Output = none
PROC    RESTORE_MESS
        CALL    CLEAR_MESSAGE
        MOV     AL,[Menu]                               ;menu color
        MOV     CL,[Color]                              ;save original color
        MOV     [Color],AL                              ;set color
        MOV     AX,0107h                                ;row 3/Col 8
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      ' Please wait . . . . . .  while resetting the '
        db      'HP Printer.           ',0
        MOV     [Color],CL                               ;restore original color
        CALL    HIDE_CUR
        RET
ENDP    RESTORE_MESS
ENDP    RESTORE_HP
;
;-------Place printer cursor in row/col position
;       Input = AX = row/col in hex
;       Output = None
PROC    HPGOTOYX
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     BX,AX                                   ;save row/col
        CMP     AL,100                                  ;is col < 100 ?
        JC      GOT1                                    ;if yes Ok continue
        XOR     AL,AL                                   ;if NO column = 0
```

```
GOT1:   XOR     AH,AH                                   ;zero to high byte
        MOV     CL,10                                   ;divisor to CL
        DIV     CL                                      ;convert to decimal
        OR      AX,3030h                                ;convert to ASCII digit
        MOV     [Col],AX                                ;save digit
        MOV     AL,BH                                   ;move row to AL
        CMP     AL,100                                  ;is row < 100 ?
        JC      GOT2                                    ;if yes Ok continue
        XOR     AL,AL                                   ;if NO row = 0
GOT2:   XOR     AH,AH                                   ;zero to high byte
        MOV     CL,10                                   ;divisor to CL
        DIV     CL                                      ;convert to decimal
        OR      AX,3030h                                ;convert to ASCII digit
        MOV     [Row],AX                                ;save digit
        MOV     AX,Offset loc
        CALL    PRINT_STRING
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    HPGOTOYX
;
;-------Show the contents of the AH register to screen.
;       Used for showing feedback from LPT port using INT 17h calls
;       Input = None
;       Output = None
;      Called from: PRINT_STRING if [Debug] is ON
;
PROC    SHOW_AH
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------display contents of AH register in binary
        MOV     BL,AH                                   ;save input in BX
        MOV     AX,0734h                                ;row/colm
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'low ',0
        MOV     CX,8                                    ;loop counter
AH0:    MOV     AX,BX
        AND     AX,1                                    ;zero all bit but first
        CALL    BIN_OUT
        CMP     CL,5
        JNZ     AH1
        CALL    CSTR_OUT
        db      ' to ',0
AH1:    SHR     BL,1
        LOOP    AH0                                     ;loop 8 times
        CALL    CSTR_OUT
        db      ' high',0
        CLC
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    SHOW_AH
;
;-------Instructions for the Print command.
```

```
;       Input = None
;       Output = None
;
PROC    PRINT_INSTRU
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        XOR     AX,AX                                   ;row 0,column 0
        CALL    MENU_BOX                                ;clear menu box
        MOV     AX,1500h                                ;row 21,column 0
        CALL    MENU_BOX                                ;draw menu box
        MOV     CL,[Color]                              ;get assigned color
        MOV     AL,[Menu]                               ;get menu color
        MOV     [Color],AL                              ;set menu color
        MOV     AX,1609h                                ;row 22,column 12
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'Press the <Esc> key to pause or cancel the '
        db      'printing of reports.',0
        CALL    HIDE_CUR
        MOV     [Color],CL                              ;restore assigned color
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRINT_INSTRU
;
PROC    ERR_SOUND
        PUSH    AX
        MOV     AX,Offset Beep
        CALL    SOUND
        POP     AX
        RET
ENDP    ERR_SOUND

;-------Place the cursor in the disired positon in the chart.
;       Input = AL = Data value in hex to position cursor.
;               CL = current row
;       Output = Cary flag = abort printing
;               CX = loop counter
;
PROC    POSITION_YX
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     DX,AX                                   ;save point
;------set cursor
        MOV     AL,30h                                  ;column position
        MOV     AH,CL                                   ;row position
        CALL    HPGOTOYX                                ;set positon
        JC      PX6                                     ;exit on printer error
;------change to units and tenths
        MOV     AX,DX                                   ;restore point value
        XOR     AH,AH                                   ;convert 16 bit number
        MOV     CL,10                                   ;divisor
        DIV     CL                                      ;al = units ah = tenths
        MOV     DX,AX                                   ;save numbers in DX
        XOR     CX,CX                                   ;zero to loop counter
        ADD     CL,DL                                   ;get units digit
```

```
        JZ      PX5                                     ;if zero out of bounds
        DEC     CX                                      ;beg. scale at 1 not 0
        JZ      PX2                                     ;if zero skip units
        CMP     CX,5                                    ;is data in bounds?
        JNC     PX5                                     ;exit if out of bounds
        MOV     AX,Offset NextUn                        ;ptr to next unit str
PX1:    CALL    PRINT_STRING                            ;advance to next unit
        JC      PX6                                     ;exit on printer error
        LOOP    PX1                                     ;loop until units = 0
PX2:    XOR     CX,CX                                   ;zero loop counter
        ADD     CL,DH                                   ;get tenths digit
        JZ      PX6                                     ;if zero OK! exit
        MOV     AX,Offset NextTn                        ;ptr to next tenths str
PX3:    CALL    PRINT_STRING                            ;advance  to next unit
        JC      PX6                                     ;exit on printer error
        LOOP    PX3                                     ;loop until units = 0
        CLC                                             ;normal exit
        JMP     SHORT PX6
PX5:    CALL    DATA_ERROR                              ;mark data error
PX6:    POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
;
;       Input = none
;       Output = none
PROC    DATA_ERROR
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    CLEAR_MESSAGE
        MOV     AL,[Warning]                            ;warning color
        MOV     CL,[Color]                              ;save original color
        MOV     [Color],AL                              ;set color
        MOV     AX,0207h                                ;row 3/Col 8
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      ' The POSITION_YX proceedure has OutOfRange data.'
        db      '  Press Any Key. ',0
        MOV     [Color],CL                               ;restore original color
        CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR
        CLC
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DATA_ERROR
ENDP    POSITION_YX
;
;
;-------place percentiles in the chart.
;       Input = CL = current row
;       Output = Cary flag = abort printing
;       Note: do not distroy the row counter in cl because POSITION_YX
;             requires the information to set the correct row position.
;
PROC    PRINT_PERCENTILES
```

```
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
;-------chart the median
        MOV     SI, Offset OutBuf + 2                   ;ptr to median in buf
        MOV     AL,[SI]                                 ;get value from chart
        CMP     AL,0Ah                                  ;is it less than 1.0?
        JC      CHM1                                    ;if yes exit no error
        CMP     AL,33h                                  ;is it larger then 5.0?
        JNC     CHM1                                    ;if yes exit no error
        CALL    POSITION_YX                             ;set cursor in chart
        JC      CHM2                                    ;exit if error
        MOV     AX, Offset Median                       ;ptr to median string
        CALL    PRINT_STRING                            ;print the median
        JC      CHM2                                    ;exit if error
;------chart the 25%
        INC     SI                                      ;ptr 25 percentile
        MOV     AL,[SI]                                 ;get value from chart
        CMP     AL,0Ah                                  ;is it less than 1.0?
        JC      CHM1                                    ;if yes exit no error
        CMP     AL,33h                                  ;is it larger then 5.0?
        JNC     CHM1                                    ;if yes exit no error
        CALL    POSITION_YX                             ;set cursor in chart
        JC      CHM2                                    ;exit if error
        MOV     AX, Offset Left                         ;ptr to Left string
        CALL    PRINT_STRING                            ;print the median
        JC      CHM2                                    ;exit if error
;------chart the 75%
        INC     SI
        MOV     AL,[SI]                                 ;get value from chart
        CMP     AL,0Ah                                  ;is it less than 1.0?
        JC      CHM1                                    ;if yes exit no error
        CMP     AL,33h                                  ;is it larger then 5.0?
        JNC     CHM1                                    ;if yes exit no error
        CALL    POSITION_YX                             ;set cursor in chart
        JC      CHM2                                    ;exit if error
        MOV     AX, Offset Right                        ;ptr to right string
        CALL    PRINT_STRING                            ;print the median
        JC      CHM2                                    ;exit if error
CHM1:   CLC
CHM2:   POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRINT_PERCENTILES

;       Input =  (assumes [OutBuf] contains the rank values)
;                CL = row position need by POSITION_YX
;       Note:  The cusor is adjusted before printing the shaded area because
;              the bar does not look smooth if compressed characters are used.
;
PROC    PRINT_RANGE
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AL,0FFh
        XOR     [Tint],AL                               ;toggle tint of bar
;------get lowest values from file PerCnt buffer
CPT0:   MOV     SI, Offset OutBuf
```

```
        MOV     AL,[SI]                                 ;get lowest score
        CMP     AL,0Ah                                  ;is it to small ?
        JC      CPT5                                    ;if yes exit no error
        CMP     AL,51                                   ;is it to large ?
        JNC     CPT5                                    ;if yes exit no error
;------position the cursor to the starting position for the bar.
        CALL    POSITION_YX                             ;set cursor
        JC      CPT6                                    ;exit on printer error
        CMP     AL,0Ah                                  ;is the score 1.0 ?
        JNZ     CPT1                                    ;if NO jump to CPT1
        MOV     DL,AL                                   ;save value lowest score
        MOV     AX,Offset HalfSp                        ;ptr to 1/2 space str
        CALL    PRINT_STRING                            ;move cursor 1/2 space
        JC      CPT6                                    ;exit on printer error
        MOV     AL,DL                                   ;restore value to AL
;------get high values from file OutBuf
CPT1:   MOV     SI, Offset OutBuf + 1                   ;ptr to highest score
        MOV     AH,[SI]                                 ;get high score
        CMP     AH,0Ah                                  ;is it to small?
        JC      CPT5                                    ;if yes exit no error
        CMP     AH,51                                   ;is it to large ?
        JNC     CPT5                                    ;if yes exit no error
        CMP     AH,50                                   ;is it a 5.0?
        JNZ     CPT4                                    ;if NO goto next test.
;------high value is 5.0!  Check to see if low value is also 0 tenths.
        MOV     DX,AX                                   ;store high/low values
        XOR     AH,AH                                   ;low number to 16 bits
        MOV     CL,10                                   ;divisor
        DIV     CL                                      ;remainder = tenths
        CMP     AH,0                                    ;is remainder = 0 ?
        MOV     AX,Offset FullBk                        ;use fullbk if Yes
        JZ      CPT2                                    ;if Yes back 2 compress
        MOV     AX,Offset BackSp                        ;if NO back 1 compress
CPT2:   CALL    PRINT_STRING                            ;move cursor
        JC      CPT6                                    ;exit on printer error
        MOV     AX,DX                                   ;restore values to AX
;------high - low = length of shaded bar
CPT4:   SUB     AH,AL                                   ;high - low
        JC      CPT5                                    ;exit if out of bounds
        MOV     AL,AH                                   ;put answer in al
        XOR     AH,AH                                   ;convert to 16 bit no.
        MOV     CL,10                                   ;divisor = 10
        DIV     CL                                      ;ah=tenths and al=units
        CALL    PRINT_UNITS                             ;print units value
        JC      CPT6                                    ;exit on abort cmd
        CALL    PRINT_TENTHS                            ;print tenths value
        JC      CPT6                                    ;exit on abort cmd
CPT5:   CLC
CPT6:   POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRINT_RANGE
;
;       Input = AL = units and AH = tenths
;       Output = Carry Flag if Printer error
;
PROC    PRINT_TENTHS
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
```

```
        PUSH    DX
        MOV     AL,AH                   ;tenths to al register
        CMP     AL,0                    ;is units = 0 ?
        JZ      PTH3                    ;if YES exit
        CMP     AL,10                   ;is it less than 10 ?
        JNC     PTH3                    ;if NO exit to large
        MOV     CL,AL                   ;move units to cl
        XOR     CH,CH                   ;make a 16 bit number
        CMP     CL,4                    ;is units 1,2 or 3 ?
        JC      PTH1                    ;if yes draw units
        INC     CL                      ;add 1 to units
        CMP     CL,7                    ;orig unit 4 5 or 6?
        JC      PTH1                    ;if yes draw units
        INC     CL                      ;add 1 to units
PTH1:   MOV     AX,Offset TenDk         ;ptr to next tenths str
        CMP     BYTE PTR [Tint],0       ;is it ON or OFF ?
        JZ      PTH2                    ;if Yes print dark
        MOV     AX,Offset TenLt         ;else print light
PTH2:   CALL    PRINT_STRING            ;advance to next tenths
        JC      PTH4                    ;exit on printer error
        LOOP    PTH2                    ;loop until units = 0
PTH3:   CLC
PTH4:   POP     DX                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRINT_TENTHS
;
;
;       Input = AL = units and AH = tenths
;       Output = Carry Flag if Printer error
;
PROC    PRINT_UNITS
        PUSH    AX                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CMP     AL,0                    ;is units value = 0
        JZ      PU2                     ;if yes then exit
        CMP     AL,5                    ;is it less than 5 ?
        JNC     PU2                     ;if NO exit to large
        MOV     CL,AL                   ;move units to cl
        XOR     CH,CH                   ;make a 16 bit number
        MOV     AX,Offset UntDk         ;ptr to next unit str
        CMP     BYTE PTR [Tint],0       ;is it ON or OFF ?
        JZ      PU1                     ;if Yes print dark
        MOV     AX,Offset UntLt         ;else print light
PU1:    CALL    PRINT_STRING            ;advance to next unit
        JC      PU3                     ;exit on printer error
        LOOP    PU1                     ;loop until units = 0
PU2:    CLC
PU3:    POP     DX                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRINT_UNITS
;
;
;       Count the number of rows that will fill one page.  (45 max)
;       Input = DX start ptr in CFGTbl
```

```
;       Output = AX = starting row =  0, 1, or 2
;                BX = stop ptr in CFGTbl
;                carry flag if error is found.
;        Note: calls its own subroutines FACTOR_ROW_COUNT
;              and PAGE_INFO_ERR.
;
PROC    GET_PAGE_INFO
        PUSH    CX
        PUSH    DX
        MOV     BX,DX
        CMP     WORD PTR [BX],7                         ;is this a group title
        JNZ     GR1                                     ;if YES adv ptr to
        ADD     DX,4                                    ;skip passed title.
GR1:    MOV     CX,8                                    ;5 title and 3 bottom
GR2:    MOV     BX,DX                                   ;get starting offset
        ADD     DX,4                                    ;ptr to next entry
        CMP     WORD PTR [BX],6                         ;is this a file name?
        JC      GR6                                     ;if yes exit error
        CMP     WORD PTR [BX],8                         ;is it EndOfTable ?
        JNZ     GR3                                     ;if NO continue else
        MOV     AL,0FFh                                 ;nonzero = EndOfTable
        MOV     [EOF],AL                                ;mark [EOF] True
        JMP     SHORT GR4                               ;exit EndOfPage
GR3:    CMP     WORD PTR[BX],7                          ;is it a group title?
        JZ      GR4                                     ;if YES exit EndOfPage
        CMP     WORD PTR [BX],6                         ;is it a factor title?
        JNZ     GR6                                     ;if NO exit error
        CALL    FACTOR_ROW_COUNT                        ;add next factor to
;------test for page length here                       ;the row count
        CMP     AX,46                                   ;is row ct > 45 ?
        JNC     GR4                                     ;if YES exit overflow
        MOV     CX,AX                                   ;else save new total
        MOV     DX,BX                                   ;and new stop pointer
        JMP     GR2                                     ;jmp forever
;------compute start row value: 45 lines = row 0, 44-43 = 1, 42-41 = 2, else 3
GR4:    MOV     AX,3                                    ;default setting
        CMP     CX,41                                   ;is it 40 or less ?
        JC      GR5                                     ;if Yes row 3 is good.
        DEC     AX                                      ;row = 2
        CMP     CX,43                                   ;is it 42 or 41 ?
        JC      GR5                                     ;if Yes row 2 is good.
        DEC     AX                                      ;row = 1
        CMP     CX,45                                   ;is it 43 or 44
        JC      GR5                                     ;if Yes row 1 is good.
        DEC     AX                                      ;row = 0
GR5:    MOV     BX,DX                                   ;return new stop ptr
        CLC                                             ;no error
        JMP     SHORT GR7                               ;normal exit point
GR6:    CALL    PAGE_INFO_ERR                           ;tell user of error
        STC                                             ;mark  error
GR7:    POP     DX
        POP     CX
        RET
;
;       Adds the number of rows need for one factor.
;       Input =  BX ptr to factor name
;                DX ptr to start of factor
;       Output = AX = new total of lines.
;                BX = ptr to next entry after factor
;       Note:  the factor box takes 4 rows
;              each bar takes one row
;              the number of bar = the number of self files
```

```
;               SF1 is type 5 and SF2 is type 4
;
PROC    FACTOR_ROW_COUNT
        PUSH    CX
        PUSH    DX
        MOV     BX,DX                                   ;ptr to first file
        MOV     AX,4                                    ;factor box = 4 rows
        ADD     CX,AX                                   ;add box to total
FRC1:   CMP     WORD PTR [BX],4                         ;is it a self file?
        JC      FRC2                                    ;if NO don' count it
        INC     CX                                      ;else count = count + 1
FRC2:   ADD     BX,AX                                   ;ptr to next type
        CMP     WORD PTR [BX],6                         ;is it a file name ?
        JC      FRC1                                    ;if Yes loop again
        MOV     AX,CX                                   ;place new total in ax
        CLC
        POP     DX
        POP     CX
        RET
ENDP    FACTOR_ROW_COUNT
;
; The error message should never be called!!
;       Input = none
;       Output = none
PROC    PAGE_INFO_ERR
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    CLEAR_MESSAGE
        MOV     CL,[Color]                              ;store original Color
        MOV     AL,[Warning]                            ;warning color
        MOV     [Color],AL                              ;set color
        MOV     AX,0207h                                ;row 3/Col 12
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      " ERROR:  GET_PAGE_INFO found a mistake when reading the "
        db      "config table. ",0
        MOV     [Color],CL                               ;restore original color
        CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR
        CALL    CLEAR_MESSAGE                           ;empty message line
        CLC                                             ;clear cf = continue
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PAGE_INFO_ERR
ENDP    GET_PAGE_INFO
;
;       Set [Time] with DOS timer ticks.  There are 18.2 ticks per second.
;       Input = none
;       Output = none
PROC    SET_TIME
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------get DOS timer ticks
ST1:    MOV     AH,0                                    ;function number
```

```
        INT     1Ah                                     ;get DOS clock ticks
        MOV     AX,455                                  ;18.2 ticks per second
        ADD     DX,AX                                   ;add 25 seconds
        JC      ST1                                     ;loop if over flow
        MOV     [Time],DX                               ;save stop time count
        CLC
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    SET_TIME
;
;  Has the DOS timer ticks in [Time] runout yet?  If NO wait until they do.
;       Input = none
;       Output = none
PROC    CHECK_TIME
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------is time up yet?
CKT1:   MOV     BX,[Time]                               ;get stop time count
        MOV     AH,0                                    ;function number
        INT     1Ah                                     ;get DOS clock ticks
        CMP     DX,BX                                   ;has time run out ?
        JC      CKT1                                    ;if not loop until done
        CLC
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    CHECK_TIME
;
;Data used by the Menu System
        .DATA
;Menu0data structure
menu0   dw      Offset menu0HK                          ;ptr to menu HotKeys
        db      '     File          Rank          Print          LPT'
        db      '          Quit          ',0
        db      05,6            ;lightbar:position in string and number of bytes
        dw      Offset m01      ;pointer to lightbar message
        db      18,6
        dw      Offset m02
        db      30,7
        dw      Offset m03
        db      44,5
        dw      Offset m04
        db      56,6
        dw      Offset m05
;
;the menu menu0HK string contains the hot keys that will activate the Choice.
;the letters in the string should include the first letters of each menu item.
;these letters must be in the same order as the menu items.  Additional
;hot keys maybe added to the string if needed.  The calling program must
;be able to filter these additional HotKeys.  The hot key string must
;end with a zero.
menu0HK db      'FRPLQ',0                               ;Hotkey ASCIIZ string.
;
;messages can be up to 72 character in length.  The length does not have to
;be the same.  The previous message is cleared before the new message is
```

```
;written.  The messages can be anywhere in the data section.  The numbering
;system for messages: "m01" stands for menu0 message0
m01     db      ' Select a configuration file describing the SLDI feedback '
        db      'reports.',0
m02     db      '     Compute the 25th and 75th percentiles for the data'
        db      ' file.',0
m03     db      '     Send the SLDI FeedBack reports to the HP LaserJet
Printer.',0
m04     db      '  Select the parallel port assigned to the HP LaserJet '
        db      'Printer.',0
m05     db      '                Exit the program and return to DOS.',0
;end of menu0 structure
;
;Menu1data structure
menu1   dw      Offset menu1HK                          ;ptr to menu HotKeys
        db      '          First LPT          Second LPT          Third '
        db      'LPT     ',0
        db       9,11           ;lightbar:position in string and number of bytes
        dw      Offset m11      ;pointer to lightbar message
        db      28,12
        dw      Offset m12
        db      48,11
        dw      Offset m13
;
;the menu menu1HK string contains the hot keys that will activate the Choice.
;the letters in the string should include the first letters of each menu item.
;these letters must be in the same order as the menu items.  Additional
;hot keys maybe added to the string if needed.  The calling program must
;be able to filter these additional HotKeys.  The hot key string must
;end with a zero.
menu1HK db      'FST',0                         ;Hotkey ASCIIZ string.
;
;messages can be up to 72 character in length.  The length does not have to
;be the same.  The previous message is cleared before the new message is
;written.  The messages can be anywhere in the data section.  The numbering
;system for messages: "m11" stands for menu1 message1
m11     db      "       Send the SLDI data to the computer's LPT 1 "
        db      "output port.",0
m12     db      "       Send the SLDI data to the computer's LPT 2 "
        db      "output port.",0
m13     db      "       Send the SLDI data to the computer's LPT 3 "
        db      "output port.",0
;end of menu1 structure
;
;
;Menu3data structure
menu3   dw      Offset menu3HK                          ;ptr to menu HotKeys
        db      '       Tag          UnTag          Print          Main '
        db      'Menu    ',0
        db      07,5            ;lightbar:position in string and number of bytes
        dw      Offset m31      ;pointer to lightbar message
        db      21,7
        dw      Offset m32
        db      37,7
        dw      Offset m33
        db      53,11
        dw      Offset m34
;
;the menu menu3HK string contains the hot keys that will activate the Choice.
;the letters in the string should include the first letters of each menu item.
;these letters must be in the same order as the menu items.  Additional
;hot keys maybe added to the string if needed.  The calling program must
```

```
;be able to filter these additional HotKeys.  The hot key string must
;end with a zero.
menu3HK db      'TUPM',0                        ;Hotkey ASCIIZ string.
;
;messages can be up to 72 character in length.  The length does not have to
;be the same.  The previous message is cleared before the new message is
;written.  The messages can be anywhere in the data section.  The numbering
;system for messages: "m11" stands for menu1 message1
m31     db      "   Tag highlighted ID.  Press Ctrl and <Enter> to "
        db      "tag all IDs.",0
m32     db      "    Untag highlighted ID.  Press Ctrl and <Enter> to "
        db      "untag all IDs.",0
m33     db      "              Print reports for all the tagged IDs.",0
m34     db      "     Untag IDs and return to the FeedBack programs "
        db      "opening menu.",0


;end of menu3 structure
;
        .CODE
;
;       Present the Main Menu and Title screen
;       Input = None
;       Output = If critical DOS error.  Error number is in AL
;              CH = last choice  CL = max number of choices
PROC    MAIN_MENU
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
;-------draw playing screen
        XOR     AX,AX                                   ;row 0 and column 0
        CALL    MENU_BOX                                ;draw top menu box
        CALL    DRAW_TITLE
        CALL    DISPLAY_STATUS                          ;display name & LPT port
        CALL    MENU_INSTRU                             ;draw bottom box
        MOV     CH,01                                   ;starting menu selection
MA1:    MOV     AX, Offset Menu0
        MOV     CL,5                                    ;max choice for menu1
        CALL    GET_CHOICE                              ;get a menu selections
;------is it Esc key ?
        CMP     AH,0h                                   ;was <ESC> key pressed?
        JNZ     MA2                                     ;if not goto next test
        CALL    EXIT_YN                                 ;exit to Dos Y/N ?
        JNC     MA1                                     ;if no get next choice
        JMP     MA8                                     ;Exit on <Esc> key
MA2:    MOV     CH,AH                                   ;save current choice
;------is it Select a data file ?
        DEC     AH                                      ;is this the choice?
        JNZ     MA3                                     ;if not goto next test
        CALL    GET_PATH                                ;get file path
        JC      MA2B                                    ;main menu if Esc key
        CALL    SELECT_FILE                             ;pick config file
        CALL    MENU_INSTRU
        CALL    DRAW_TITLE                              ;redraw title screen
        CALL    DISPLAY_STATUS                          ;display name & LPT port
MA2B:   XOR     AX,AX                                   ;row/column
        CALL    MENU_BOX                                ;clear menu box
        JMP     SHORT MA1                               ;get another choice
;------is it Rank the variables?
MA3:    DEC     AH                                      ;is this the choice?
        JNZ     MA4                                     ;if not goto next test
```

```
        CALL    RANK_DATA                       ;compute presentiles
        JC      MA1
        CALL    MENU_INSTRU                     ;restore bottom box
        CALL    DISPLAY_STATUS
        JMP     SHORT MA1                       ;get another choice
;------is it Print the data file?
MA4:    DEC     AH                              ;is this the choice?
        JNZ     MA5                             ;if not goto next test
        CALL    PRINT_MENU                      ;
        JC      MA1                             ;loop if not ready
        MOV     CH,5                            ;set main menu to Quit
        CALL    MENU_INSTRU
        CALL    DRAW_TITLE                      ;redraw title screen
        CALL    DISPLAY_STATUS                  ;display name & LPT port
      JMP     SHORT MA1                        ;get another choice
;------is it choose a Laser port?
MA5:    DEC     AH
        JNZ     MA6
        CALL    PORT_MENU                       ;Select LPT port
        JMP     SHORT MA7                       ;return to main menu
;------is it the Exit command ?
MA6:    DEC     AH
        JNZ     MA7                             ;go get another choice
        CALL    EXIT_YN                         ;exit to Dos Y/N ?
        JC      MA8                             ;if yes exit. else
MA7:    CALL    DISPLAY_STATUS                  ;display status report
        JMP     SHORT MA1                       ;get next choice.
MA8:    CLC                                     ;clear cf = normal exit
MA9:    POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    MAIN_MENU
;       Print Menu for selecting a single or all reports.
;       Input = None
;       Output = If critical DOS error  Error number is in AL
;              CH = last choice  CL = max number of choices
PROC    PRINT_MENU
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    IS_PRINT                        ;is data ready to print?
        JC      PI8                             ;exit if not ready.
        MOV     WORD PTR [TopBar],1             ;default (1 to MaxId)
        MOV     BYTE PTR [HiBar],15             ;default (15 to 1)
        CALL    PRINT_MENU_INSTRU               ;display print instru
        CALL    DISPLAY_IDS
        MOV     CH,1                            ;starting menu selection
PI1:    MOV     AX, Offset Menu3
        MOV     CL,4                            ;max choice for menu1
        CALL    GET_PRINT_CHOICE                ;get a menu selections
;------is it Esc key ?
        CMP     AH,0h                           ;was <ESC> key pressed?
        JZ      PI7                             ;if yes exit menu
        MOV     CH,AH                           ;save current choice
;------is it Tag ID ?
        DEC     AH                              ;is this the choice?
        JNZ     PI2                             ;if not goto next test
        CALL    TAG
        JMP     SHORT PI1                       ;get next choice
```

```
;------is it Untag ID ?
PI2:    DEC     AH                              ;is this the choice?
        JNZ     PI3                             ;if not goto next test
        CALL    UNTAG
        JMP     SHORT PI1                       ;get next choice
;------is it Print ?
PI3:    DEC     AH                              ;is this the choice?
        JNZ     PI4                             ;if not goto next test
        CALL    IS_TAG                          ;are any ID tagged ?
        JC      PI1                             ;if NO abort command
        CALL    PRINT_REPORTS                   ;for all tagged IDs
        CALL    MENU_INSTRU                     ;restore bottom box
        JMP     SHORT PI1                       ;get next choice
;------is it return to Main Menu ?
PI4:    DEC     AH                              ;is this the choice?
        JNZ     PI1                             ;NO go get next char
PI7:    CALL    MENU_INSTRU                     ;draw bottom box
        CALL    RELEASE_VAR_BLK                 ;release mem var block
        CLC                                     ;clear cf = normal exit
PI8:    POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRINT_MENU
;
;       Menu for selecting the LPT output port.
;       Input = None
;       Output = If critical DOS error  Error number is in AL
;              CH = last choice  CL = max number of choices
PROC    PORT_MENU
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     CX,[LPT]                        ;current port assignment
        MOV     CH,CL                           ;place in CH register
        INC     CH                              ;starting menu selection
MI1:    MOV     AX, Offset Menu1
        MOV     CL,3                            ;max choice for menu1
        CALL    GET_CHOICE                      ;get a menu selections
;------is it Esc key ?
        CMP     AH,0h                           ;was <ESC> key pressed?
        JZ      MI4                             ;if yes exit menu
        XOR     AL,AL                           ;zero LPT choice
        MOV     CH,AH                           ;save current choice
;------is it Select LPT 1 ?
        DEC     AH                              ;is this the choice?
        JNZ     MI2                             ;if not goto next test
        XOR     AH,AH                           ;zero AH register
        MOV     [LPT],AX                        ;assign 0 to [LPT]
        JMP     SHORT MI4                       ;exit procedure
;------is it Select LPT 2 ?
MI2:    INC     AL                              ;assign 1 to AL
        DEC     AH                              ;is this the choice?
        JNZ     MI3                             ;if not goto next test
        XOR     AH,AH                           ;zero AH register
        MOV     [LPT],AX                        ;assign 1 to [LPT]
        JMP     SHORT MI4                       ;exit procedure
;------is it Select LPT 3 ?
MI3:    INC     AL                              ;assign 2 to AL
        DEC     AH                              ;is this the choice?
```

```
        JNZ     MI1                             ;NO =  get next choice
        XOR     AH,AH                           ;zero AH register
        MOV     [LPT],AX                        ;assign 2 to [LPT]
MI4:    CLC                                     ;clear cf = normal exit
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PORT_MENU
;
;
;-------Get a Choice from the Keyboard from the menu system pointed to by AX.
;       Input = AX points the desired Menu data structure
;               CH = Starting Choice (menu item to highlight)
;               CL = max number of choices in this menu.
;       Output = AH = Choice is returned to the calling program
;                AL = Char from the keyboard. (Return key or Esc key)
;       Calls  'Display_Menu' to display the menu on the screen.
;              'Hot_keys' to see is Char is a HotKey for this menu
;
PROC    GET_CHOICE
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     DX,AX
GE0:    CALL    DISPLAY_MENU                    ;menu to the screen
        MOV     BL,CH                           ;save old Choice in BL
GE1:    CALL    GET_CHAR                        ;get keyboard input.
        CMP     AL,'1'                          ;is Char < '1'
        JC      GE4                             ;if yes goto next test
        CMP     AL,';'                          ;is Char a digit ?
        JNC     GE4                             ;if not goto next test
        CALL    NUM_LOCK                        ;covert NumLock pad
GE4:    CMP     AL,4                            ;Is it a right arrow?
        JNZ     GE5                             ;jump if not
        INC     CH                              ;Choice = Choice + 1
GE5:    CMP     AL,13h                          ;Is it a left arrow?
        JNZ     GE6                             ;jump if not
        DEC     CH                              ;Choice = Choice -1
        JNZ     GE6                             ;if CH = 0 then
        MOV     CH,CL                           ;set CH = maxmenu.
GE6:    CMP     CL,CH                           ;Is CH > maxmenu?
        JNC     GE7                             ;If yes then
        MOV     CH,1                            ;set CH = 1
GE7:    CMP     AL,0Dh                          ;is it a return key?
        JZ      GE10                            ;if yes return
        CMP     AL,1Bh                          ;is it an escape key?
        JNZ     GE8                             ;if no goto next test
        SUB     CH,CH                           ;if yes, choice = 0
        JMP     SHORT GE10                      ;exit; save new Choice
GE8:    CALL    HOT_KEYS                        ;is Char a hot key?
        JC      GE9                             ;carry = No; next char
        CMP     AH,CH                           ;has Choice changed?
        JZ      GE11                            ;if not then exit
        MOV     CH,AH                           ;save new Choice
        CALL    DISPLAY_MENU                    ;display menu on Exit
        JMP     SHORT GE11                      ;exit; save new choice
GE9:    CMP     CH,BL                           ;new Choice=old Choice?
        JZ      GE1                             ;yes = no menu display
        JMP     SHORT GE0                       ;no = call display menu
GE10:   MOV     AH,CH                           ;save choice on Exit
```

```
GE11:   POP     DX
        POP     CX
        POP     BX
        RET
;
PROC    NUM_LOCK
        AND     AL,0Fh                          ;convert to hex
        DEC     AL
        JNZ     NML0                            ;if not = 1 continue
        MOV     AL,6                            ;convert to End
        JMP     SHORT NML7                      ;exit
NML0:   DEC     AL
        JNZ     NML1                            ;if not = 2 continue
        MOV     AL,24                           ;convert to DnArrow
        JMP     SHORT NML7                      ;exit
NML1:   DEC     AL
        JNZ     NML2                            ;if not = 3 continue
        MOV     AL,3                            ;convert to PageDn
        JMP     SHORT NML7                      ;exit
NML2:   DEC     AL
        JNZ     NML3                            ;if not = 4 continue
        MOV     AL,19                           ;convert to LeftArrow
        JMP     SHORT NML7                      ;exit
NML3:   DEC     AL
        DEC     AL
        JNZ     NML4                            ;if not = 6 continue
        MOV     AL,4                            ;convert to RtArrow
        JMP     SHORT NML7                      ;exit
NML4:   DEC     AL
        JNZ     NML5                            ;if not = 7 continue
        MOV     AL,1                            ;convert to Home Key
        JMP     SHORT NML7                      ;exit
NML5:   DEC     AL
        JNZ     NML6                            ;if not = 8 continue
        MOV     AL,5                            ;convert to UpArrow
        JMP     SHORT NML7                      ;exit
NML6:   DEC     AL
        JNZ     NML7                            ;if not = 9 continue
        MOV     AL,18                           ;convert to PageUp
NML7:   RET
ENDP    NUM_LOCK
;
;       Examine the Hot Key ASCIIZ string to find out if Char is a Hot Key.
;       Input = AL = Char
;               CH = Choice
;               CL = MaxChoice
;               DX = pointer to the menu structure in data segment.
;                    the first word in the data structure is a pointer to
;                    the Hot Key ASCIIZ string.
;       Output = Carry Flag  if Char in AL is not a HotKey
;               AH = Choice
;               AL = menu match  AL = 0Dh
;                    nonmenu match AL = Char
;       Notes: Called by GET_CHOICE. Menu data must be in an exact format.
;              See Menuldata structure for an example of the corrcet format.
PROC    HOT_KEYS
        PUSH    BX                              ;save registers
        PUSH    CX
        PUSH    DX
        MOV     SI,DX                           ;ptr to HotKey string pointer
        MOV     BX,[SI]                         ;load ptr to ASCIIZ HotKey str.
        AND     AL,7Fh                          ;make 0 - 127 ASCII char.
```

```
        CMP     AL,'a'                          ;is char a small letter?
        JC      HOT1                            ;if not, Ok continue.
        AND     AL,0DFh                         ;change to capital char
HOT1:   MOV     DX,AX                           ;save Char in DX
        XOR     AX,AX                           ;zero to AX
        MOV     SI,AX                           ;new Choice counter
        MOV     AL,DL                           ;Char returns to AL
HOT2:   CMP     [BX+SI],AH                      ;is this the End of String?
        JZ      HOT5                            ;exit; no match found
        CMP     [BX+SI],AL                      ;is Char a Hot Key?
        JZ      HOT3                            ;0 = found a Hot Key
        INC     SI                              ;choice = choice + 1
        JMP     SHORT HOT2                      ;check the next Char in string.
        JMP     HOT4
HOT3:   INC     SI                              ;choice = choice + 1
        MOV     BX,SI                           ;choice counter to BL
        MOV     AH,CH                           ;original Choice to AH
        CMP     CL,BL                           ;is choice a menu item?
        JC      HOT4                            ;carry = not a menu item
        MOV     AH,BL                           ;set new Choice
HOT4:   CLC                                     ;clear carry = HotKey found
        JMP     SHORT HOT6                      ;Exit  (found)
HOT5:   MOV     AH,CH                           ;restore original Choice
        STC                                     ;set carry flag = not HotKey
HOT6:   POP     DX                              ;restore registers
        POP     CX
        POP     BX
        RET
ENDP    HOT_KEYS
;
;       Display menu string; highlight one menu item; and write message string.
;       Input = DX pointer to the menu structure in data segment.
;               CH = Choice
;       Ouput = None
;       Notes: Called by GET_CHOICE. Menu data must be in an exact format.
;              See Menldata structure for an example of the corrcet format.
PROC    DISPLAY_MENU
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        INC     DX                              ;skip HotKey string offset
        INC     DX                              ;ptr to beginning menu string
        MOV     AL,[Menu]                       ;menu color attribute
        MOV     [Color],AL                      ;change color attribute
        MOV     AX,0107h                        ;starting position for cursor
        CALL    GOTOYX                          ;place cursor
        MOV     AX,DX                           ;Offset menu to AX
        CALL    DSTR_OUT                        ;Display menu
        MOV     AL,[Warning]                    ;color for lightbar
        MOV     [Color],AL                      ;change color attribute
        SUB     AX,AX                           ;zero AX register
        ADD     AL,CH                           ;Choice to AL
        JZ      DIP1                            ;abort if Choice = 0
        DEC     AL                              ;(Choice-1) * 4 = offset
        MOV     CL,2                            ;number of shifts to CL
        SHL     AX,CL                           ;shift twice = ax*4
        INC     SI                              ;SI points to base of table
        ADD     SI,AX                           ;add offset
        MOV     BX,[SI]                         ;get 2 bytes from table
        MOV     CL,BH                           ;number of char to copy
        SUB     BH,BH                           ;zero BH
```

```
        DEC     BL                                      ;BX = position in menu string
        MOV     AX,0107h                                ;starting cursor postion
        ADD     AX,BX                                   ;add offset to choice
        PUSH    SI                                      ;save SI register
        CALL    GOTOYX                                  ;position cursor
        POP     SI                                      ;restore SI register
        MOV     AX,DX                                   ;start of menu string to AX
        ADD     AX,BX                                   ;add offset to choice
        MOV     DX,[SI + 2]                             ;ptr to message from table.
        CALL    SUB_DSTR_OUT                            ;highlight choice
        MOV     AL,[Menu]                               ;set menu color
        MOV     [Color],AL                              ;change color attribute
        MOV     AX,0207h                                ;position cursor in 2rd row
        CALL    GOTOYX                                  ;column 7 for message string.
        MOV     BX,024Eh                                ;row 2 and column 78
        CALL    CLEAR_WINDOW                            ;clear out old message.
        MOV     AL,[MenuMes]                            ;set message color
        MOV     [Color],AL                              ;change color attribute
        MOV     AX,DX                                   ;pointer to message string
        CALL    DSTR_OUT                                ;display message string
        MOV     AL,[Normal]                             ;set normal color
        MOV     [Color],AL                              ;change color attribute
        CALL    HIDE_CUR                                ;hide the cursor.
DIP1:   POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DISPLAY_MENU
ENDP    GET_CHOICE
;

;
;-------Instructions for use of the menu system highlight bar.
;       Input = None
;       Output = None
;
PROC    MENU_INSTRU
        PUSH    AX                                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,1500h                                        ;row 21,column 0
        CALL    MENU_BOX                                        ;draw menu box
        MOV     AL,[Menu]
        MOV     [Color],AL
        MOV     AX,160Ch                                        ;row 22,column 12
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'Use the ',17,205,' or ',205,16,' arrow keys to '
        db      'position the Highlight Bar.',0
        MOV     AX,1709h                                        ;row 23,column 12
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'Press the <Enter> key to begin the highlighted Menu Bar '
        db      'command.',0
        MOV     AL,[Normal]
        MOV     [Color],AL
        POP     DX                                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
```

```
        RET
ENDP    MENU_INSTRU
;
;-------Display the users selections.
;       Input = AX = none
;       Output = none
;       AX-DX register saved.
;
PROC    DISPLAY_STATUS
        PUSH    AX
        PUSH    BX                              ;save registers
        PUSH    CX
        PUSH    DX
        MOV     CL,[Color]                      ;save original color
        MOV     AL,[Normal]
        MOV     [Color],AL
        MOV     AX,0506h                        ;row/col
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'Data File = ',0
        XOR     AX,AX                           ;zero to AX
        MOV     BX,[DATHd]                      ;file handle
        CMP     BX,AX                           ;Is the file open?
        JZ      FLE1                            ;if no clear line
        MOV     AX,DS                           ;assign ES to the
        MOV     ES,AX                           ;data section
        MOV     AX,Offset Search                ;ptr to path + file name
        MOV     CX,AX                           ;save str beginning ptr
        CALL    STR_LENGTH                      ;get length of string
        SUB     AX,4                            ;length - 4 = "."
        ADD     AX,CX                           ;ptr to the period
        MOV     DI,AX                           ;destination ptr
        MOV     SI,Offset DATTyp                ;ptr to file type name
        CLD                                     ;auto inc SI & DI
        MOV     CX,5                            ;number byte to move
        REP     MOVSB                           ;move type to Search
        MOV     AX, Offset Search               ;ptr to name of file
        CALL    DSTR_OUT                        ;send to the screen
        JMP     SHORT FLE2
FLE1:   CALL    CSTR_OUT
        db      'NOT Selected',0
FLE2:   MOV     AX,0534h
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      "Number of IDs = ",0
        MOV     AX,[MaxID]                      ;report type
        CALL    BIN_OUT
        MOV     AX,0606h                        ;row/col
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'Percentiles are ',0
        XOR     AX,AX                           ;zero to AX
        CMP     WORD PTR [RNKHd],0              ;is the rank file open?
        JZ      FLE3                            ;if not goto next text
        CALL    CSTR_OUT                        ;send string to screen
        db      'computed.     ',0
        JMP     SHORT FLE7                      ;exit routine
FLE3:   CALL    CSTR_OUT                        ;send string to screen
        db      'NOT computed.',0
FLE7:   CALL    ON_LINE                         ;is printer on_line?
        JNC     FLE8                            ;carry flag means NO
        MOV     AX,0634h                        ;row/col
```

```
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT
        db      'LPT ',0
        XOR     AX,AX
        MOV     AX,[LPT]
        INC     AX
        CALL    BIN_OUT
        CALL    CSTR_OUT                                ;send string to screen
        db      ' is NOT Ready.',0
        JMP     SHORT FLE9
FLE8:   MOV     AX,0634h                                ;row/col
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT
        db      'LPT ',0
        XOR     AX,AX
        MOV     AX,[LPT]
        INC     AX
        CALL    BIN_OUT
        CALL    CSTR_OUT                                ;send string to screen
        db      ' is Ready.    ',0
FLE9:   MOV     [Color],CL                              ;restore color value
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DISPLAY_STATUS
;
;
;-------Instructions for use of the print menu system .
;       Input = None
;       Output = None
;
PROC    PRINT_MENU_INSTRU
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    PRINT_WINDOW
        MOV     AX,1500h                                ;row 21,column 0
        CALL    MENU_BOX                                ;draw menu box
        MOV     AL,[Menu]
        MOV     [Color],AL
        MOV     AX,160Fh                                ;row 22,column 12
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'Use the arrow keys to position the Highlight Bars.',0
        MOV     AX,1709h                                ;row 23,column 12
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'Press the <Enter> key to begin the highlighted Menu Bar '
        db      'command.',0
        MOV     AL,[Normal]
        MOV     [Color],AL
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRINT_MENU_INSTRU
;
;
```

```
;--- draw ID diplay windows and key instructions
;       Input = None
;       Output = None
PROC    PRINT_WINDOW
        MOV     AL,[Normal]                             ;set Color
        MOV     [Color],AL
        CALL    CLEAR_TITLE
        MOV     AX,0802h                                ;row 8 column 4
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      "<Up Arrow> = Move Up                                   "
        db      "<Dn Arrow> = Move Dn",0
        MOV     AX,0A02h                                ;row 10,column 4
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      "<PageUp> = Scroll Up                                   "
        db      "<Home> = First ID",0
        MOV     AX,0C02h                                ;row 12,column 4
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      "<PageDn> = Scroll Dn                                   "
        db      "<End> =  Last ID",0
;------draw display windows
        MOV     AL,[System]
        MOV     [Color],AL
        MOV     AX,061Bh
        MOV     BX,1436h
        CALL    CLEAR_WINDOW
        CALL    HIDE_CUR
        MOV     [Color],CL                              ;restore original Color
        CLC                                             ;clear carry flag
        RET
ENDP    PRINT_WINDOW
;
;
;-------Special version if Get_Choice to control menu and ID display on screen.
;       Input = AX points the desired Menu data structure
;               CH = Starting Choice (menu item to highlight)
;               CL = max number of choices in this menu.
;       Output = AH = Choice is returned to the calling program
;                AL = Char from the keyboard. (Return key or Esc key)
;       Calls  'Display_Menu' to display the menu on the screen.
;              'Hot_keys' to see is Char is a HotKey for this menu
;
PROC    GET_PRINT_CHOICE
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     DX,AX
GP0:    CALL    DISPLAY_MENU                            ;menu to the screen
        MOV     BL,CH                                   ;save old Choice in BL
GP1:    CALL    GET_CHAR                                ;get keyboard input.
        CALL    IS_ID_KEY                               ;change display?
        JC      GP1                                     ;loop if YES.
        CMP     AL,'1'                                  ;is Char < '1'
        JC      GP4                                     ;if yes goto next test
        CMP     AL,';'                                  ;is Char a digit ?
        JNC     GP4                                     ;if not goto next test
        CALL    NUM_LOCK                                ;covert NumLock pad
GP4:    CMP     AL,4                                    ;Is it a right arrow?
        JNZ     GP5                                     ;jump if not
        INC     CH                                      ;Choice = Choice + 1
```

```
GP5:    CMP     AL,13h                          ;Is it a left arrow?
        JNZ     GP6                             ;jump if not
        DEC     CH                              ;Choice = Choice -1
        JNZ     GP6                             ;if CH = 0 then
        MOV     CH,CL                           ;set CH = maxmenu.
GP6:    CMP     CL,CH                           ;Is CH > maxmenu?
        JNC     GP7                             ;If yes then
        MOV     CH,1                            ;set CH = 1
GP7:    CMP     AL,0Dh                          ;is it a return key?
        JZ      GP10                            ;if yes return
        CMP     AL,1Bh                          ;is it an escape key?
        JNZ     GP8                             ;if no goto next test
        SUB     CH,CH                           ;if yes, choice = 0
        JMP     SHORT GP10                      ;exit; save new Choice
GP8:    CALL    HOT_KEYS                        ;is Char a hot key?
        JC      GP9                             ;carry = No; next char
        CMP     AH,CH                           ;has Choice changed?
        JZ      GP11                            ;if not then exit
        MOV     CH,AH                           ;save new Choice
        CALL    DISPLAY_MENU                    ;display menu on Exit
        JMP     SHORT GP11                      ;exit; save new choice
GP9:    CMP     CH,BL                           ;new Choice=old Choice?
        JZ      GP1                             ;yes = no menu display
        JMP     SHORT GP0                       ;no = call display menu
GP10:   MOV     AH,CH                           ;save choice on Exit
GP11:   POP     DX
        POP     CX
        POP     BX
        RET
ENDP    GET_PRINT_CHOICE
;
;  Is this character a key that controls the display of IDs?
;       Input = AL contain ascii character
;            Local variables:
;               BX = [TopBar] is index number at top of screen (1 to MaxId)
;               CL = [HiBar] = hilite bar position # 1 to 15
;               Note: 15 = top and 1 = bottom
;
;       Output = Carry Flag = Yes
;
PROC    IS_ID_KEY
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     BX,[TopBar]                     ;Id at top of screen
        MOV     CL,[HiBar]                      ;hight bar poition
;------is it Control Enter ?
        CMP     AL,0Ah                          ;is it Ctr-Return ?
        JNZ     IIK2                            ;if NO goto next test
        CMP     CH,2                            ;is choice = untag all?
        JNZ     IIK1                            ;if NO goto next test
        CALL    UNTAG_ALL                       ;else untag all and
        JMP     IIK13                           ;display results
IIK1:   CMP     CH,1                            ;is choice = tag all ?
        JNZ     IIK2                            ;if NO goto next test
        CALL    TAG_ALL                         ;else tag all IDs and
        JMP     IIK13                           ;display results
;-------is it a Down arrow ?
IIK2:   CMP     AL,24                           ;is it Down arrow?
        JNZ     IIK5                            ;if not goto next test
        MOV     AX,BX                           ;get top of screen
```

```
        ADD     AX,16                   ;add length of screen+1
        XOR     CH,CH                   ;make a 16 bit number
        SUB     AX,CX                   ;top of screen+16-bar
        CMP     [MaxId],AX              ;is bar at EndOfIndex?
        JNC     IIK3                    ;if NO continue else
        JMP     IIK14                   ;if Yes exit do nothing
IIK3:   CMP     CL,1                    ;is bottom of window ?
        JZ      IIK4                    ;if yes inc starting
        DEC     CL                      ;else inc bar number
        JMP     IIK13                   ;exit change
IIK4:   INC     BX                      ;inc starting number
        JMP     SHORT IIK13             ;display new directory
;-------is it an Up arrow ?
IIK5:   CMP     AL,5                    ;is it Up arrow ?
        JNZ     IIK7                    ;if no goto next test
        CMP     CL,15                   ;is cursor at the top
        JZ      IIK6                    ;if yes check index no.
        INC     CL                      ;if NO move hilite up
?
        JMP     SHORT IIK13             ;exit change
IIK6:   CMP     BX,1                    ;is this top to list?
        JZ      IIK14                   ;if yes Exit no change
        DEC     BX                      ;if NO decrease index
        JMP     SHORT IIK13             ;exit change
;-------Is it a Home Key ?             ;go to Top of Directroy
IIK7:   CMP     AL,1                    ;is it Home key?
        JNZ     IIK8                    ;if not goto next test
        MOV     BX,1                    ;set top of index
        MOV     CL,15                   ;set bar at top
        JMP     SHORT IIK13             ;display new directory
;-------Is it a End Key ?              ;go Bottom of Directory
IIK8:   CMP     AL,6                    ;is it End key ?
        JNZ     IIK10                   ;if not goto next test
        MOV     AX,[MaxId]              ;get number of files
        CMP     AX,15                   ;will it fill the box ?
        JC      IIK9                    ;if NO compute bar pos.
        SUB     AX,14                   ;compute top of screen
        MOV     BX,AX                   ;set top of screen
        MOV     CL,1                    ;bar to bottom screen
        JMP     SHORT IIK13             ;display new directory
IIK9:   SUB     AX,BX                   ;inverse of bar position
        MOV     CL,15                   ;top - inverse = post.
        SUB     CL,AL                   ;set bar position
        JMP     SHORT IIK13             ;display directory
;-------Is it the PageUp Key ?
IIK10:  CMP     AL,18                   ;is it the pageup key ?
        JNZ     IIK11                   ;if goto next test
        MOV     AX,BX                   ;save top of screen
        CMP     BX,16                   ;is this the first page
page + 1
        MOV     BX,1                    ;set top in case YES.
        JC      SHORT IIK13             ;if it is first page
        SUB     AX,15                   ;subtarct a page
        MOV     BX,AX                   ;new addres in bx
        JMP     SHORT IIK13             ;display new page
;-------Is it the PageDn Key ?
IIK11:  CMP     AL,3                    ;is it the pagedn key ?
        JNZ     IIK12                   ;if goto next test
        MOV     AX,BX                   ;number of IDs
        ADD     AX,15                   ;ptr to top of lastpage
        CMP     [MaxID],AX              ;is this the lastpage?
        JC      IIK14                   ;if NO skip bar check
```

```
        ADD     BX,15                   ;else go Page + 1
        MOV     AX,[MaxID]              ;get max index number
        SUB     AX,BX                   ;is this the last page
        CMP     AX,14                   ;<14 means change bar
        JNC     IIK13                   ;exit if not last page
        MOV     AH,15                   ;max lines on the page
        SUB     AH,AL                   ;convert highlit bar no
        CMP     CL,AH                   ;is bar below last line
        JNC     IIK13                   ;if no do not more bar
        MOV     CL,AH                   ;else set new hilite bar
        JMP     SHORT IIK13
IIK12:  CLC                             ;clear flag means not
        JMP     SHORT IIK15             ;a key for this routine
IIK13:  MOV     [TopBar],BX             ;save current bars
        MOV     [HiBar],CL
        CALL    DISPLAY_IDS
IIK14:  STC                             ;set carry flag = change
IIK15:  POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    IS_ID_KEY
;
        .DATA

TKey    db      '
        ',0
        db      '
        ',0
        db      '
',0
        db      '
',0
        db      '
',0
        db      '
',0
        db      '
',0
        db      '
',0

Shade   db      '
        ',0
        db      '
        ',0
        db      '
',0
        db      '
',0
        db      '
',0
        db      '
',0
        db      '
',0
        db      '
',0

        .CODE
;-------Display the programs title on the screen
```

```
;       Input =  None
;               [color] = current attribute from data section
;       Output = None
;       Video mode:  text   row = 25 Col = 80
;
PROC DRAW_TITLE
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    CLEAR_TITLE                             ;clear main window
        MOV     AL,[Warning]
        MOV     [Color],AL
        MOV     AX,0800h
        MOV     BX,134Fh
        CALL    CLEAR_WINDOW
        MOV     AL,[Menu]
        MOV     [Color],AL
        MOV     AX,0902h
        MOV     BX,124Dh
        CALL    CLEAR_WINDOW
;------draw shading
        MOV     AL,[Border]
        MOV     [Color],AL
        MOV     CX,8                                    ;line counter
        MOV     BX,0A05h                                ;row/col
        MOV     AX,Offset Shade
TIT2:   XCHG    AX,BX
        CALL    GOTOYX
        XCHG    AX,BX
        CALL    DSTR_OUT
        MOV     AX,SI
        INC     AX
        INC     BH
        LOOP    TIT2
;------draw title
        MOV     AL,[Normal]
        MOV     [Color],AL
        MOV     CX,7                                    ;line counter
        MOV     BX,0A04h                                ;row/col
        MOV     AX,Offset TKey
TIT1:   XCHG    AX,BX
        CALL    GOTOYX
        XCHG    AX,BX
        CALL    TITLE_OUT
        MOV     AX,SI
        INC     AX
        INC     BH
        LOOP    TIT1
TIT3:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DRAW_TITLE
;
;-------Clear the main display window in the EditKey view program.
;       Input =  None
;       Output = None
;        16 colors   row = 25 Col = 80
;
PROC CLEAR_TITLE
```

```
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        MOV     CL,[Color]                      ;save orig. color attri
        MOV     AL,[Normal]                     ;get background color
        MOV     [Color],AL                      ;assign backgd color
        MOV     AX,0400h                        ;row 4 /column 0
        MOV     BX,144Fh                        ;row 20 /column 79
        CALL    CLEAR_WINDOW                    ;clear display window
        MOV     [Color],CL                      ;restore orig. color att
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    CLEAR_TITLE
;
;
;-------Send an ASCIIZ string to screen and skip all <space> but advance
;       the cursor for each space.
;
;       Input = AX must point to the string. The string must end with
;               a hex zero.  The desired color attribute must be defined
;               in the data segment.
;       Output = None.  All register are saved except SI.
;
PROC TITLE_OUT
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     SI,AX                           ;pointer to string.
        MOV     BH,0                            ;page 0 asummed
        MOV     CX,1                            ;from data segment.
        MOV     BL,[Color]                      ;load color attribute
TITL1:  MOV     AL,[SI]                         ;get char from string
        CMP     AL,0                            ;Is it the end ?
        JZ      TITL2                           ;exit if end of string.
        CMP     AL,20h                          ;is it a space ?
        JNZ     TITL3                           ;if space skip ?
        CALL    INC_CURSOR
        JMP     SHORT TITL4
TITL3:  MOV     AL,20h                          ;write a space
        MOV     DL,AL                           ;store char in DL
        MOV     AX,0920h                        ;write 1 space to
        PUSH    SI                              ;save SI register
        INT     10h                             ;set color attribute
        MOV     AH,0Eh                          ;fun.no. teletype mode
        MOV     AL,DL                           ;get char for DL reg.
        INT     10h                             ;char to the console
        POP     SI                              ;restore SI register
TITL4:  INC     SI                              ;point to next char
        JMP     SHORT TITL1                     ;get next character.
TITL2:  POP     DX
        POP     CX                              ;restore registers.
        POP     BX
        POP     AX
        RET
ENDP    TITLE_OUT
;
        .CODE
;
;------Set the colors variables for the video mode.
```

```
;
;       Input: ES is a ptr to the (PSP) Program Segment Prefix
;       (when DOS programs are loaded the ES register points to PSP)
;       Output: None
;
PROC    COLOR_MODE
        MOV     DI,80h                          ;offset to len COM tail
        XOR     AX,AX                           ;zero register
        ADD     AL,[ES:DI]                      ;get len Com tail
        JZ      VID2                            ;jmp if no parameters
        MOV     CX,AX                           ;loop counter
VID0:   INC     DI                              ;ptr to next byte
        MOV     AL,[ES:DI]                      ;is it the marker?
        AND     AL,5Fh                          ;make a capital letter
        CMP     AL,'M'                          ;is it the mono par?
        JZ      VID1                            ;if Yes jump  to Mono
        LOOP    VID0                            ;look through COM tail
        JMP     SHORT VID2                      ;Not found get dis mode
VID1:   CALL    MONO_VIDEO                      ;set color variables
        JMP     SHORT VID5                      ;exit
VID2:   XOR     AX,AX
        MOV     AH,0Fh                          ;get display mode
        INT     10h                             ;BIOS function.
        CMP     AL,7                            ;is it Text-Mono ?
        JNZ     VID3                            ;No = jmp next test
        CALL    MONO_VIDEO                      ;set color variables
        JMP     SHORT VID5                      ;exit
VID3:   CMP     AL,0Fh                          ;is it Graph-Mono?
        JNZ     VID4                            ;No = jmp next test
        CALL    MONO_VIDEO                      ;set color variables
        JMP     SHORT VID5                      ;exit
VID4:   CALL    COLOR_VIDEO                     ;set color variables
VID5:   RET
;
PROC    COLOR_VIDEO
        MOV     BX,Offset Menu
        MOV     AL,71h                          ;Menu=Blue/Lt White
        MOV     [BX],AL
        INC     BX
        MOV     AL,1Fh                          ;Normal=White/Blue
        MOV     [BX],AL
        INC     BX
        MOV     AL,1Eh                          ;HiLite=Yellow/Blue
        MOV     [BX],AL
        INC     BX
        MOV     AL,7Fh                          ;MenuMes=White/Lt White
        MOV     [BX],AL
        INC     BX
        MOV     AL,4Fh                          ;Warning=White/Red
        MOV     [BX],AL
        INC     BX
        MOV     AL,070h                         ;Border=Black/White
        MOV     [BX],AL
        RET
ENDP    COLOR_VIDEO
PROC    MONO_VIDEO
        MOV     BX,Offset Menu
        MOV     AL,70h                          ;Menu=Black/White
        MOV     [BX],AL
        INC     BX
        MOV     AL,0Fh                          ;Normal= White/Black
        MOV     [BX],AL
```

```
        INC     BX
        MOV     AL,0Fh                          ;HiLite= White/Black
        MOV     [BX],AL
        INC     BX
        MOV     AL,70h                          ;MenuMes=Black/White
        MOV     [BX],AL
        INC     BX
        MOV     AL,0Fh                          ;Warning= White/Black
        MOV     [BX],AL
        INC     BX
        MOV     AL,7Fh                          ;Border= White/Lt White
        MOV     [BX],AL
        RET
ENDP    MONO_VIDEO
ENDP    COLOR_MODE
;
; Save the current users video information to be restored by RESTORE_VIDEO
; set text video mode for this program.
;
;       Input = None
;       Output = set variables:  [vidmode],[vidpage],[vidcurs],[vidfont]
;                                [vidattr] and [vidbord]
;       Note: has no effect if the dos version is less than 3.30.
;
PROC    TEXT_VIDEO
;------test for DOS 3.3 or greater
        MOV     AH,30h                          ;get dos ver number
        INT     21h
        XCHG    AH,AL                           ;high byte to ah
        CMP     AX,031Eh                        ;is dos >= 3.30 ?
        JNC     SV0                             ;if yes continue else
        JMP     SV5                             ;exit assume text mode
;------get video mode
SV0:    MOV     AH,0Fh                          ;get video mode
        INT     10h
        MOV     [vidmode],AL                    ;save videomode
        MOV     [vidpage],BH                    ;save videopage
;------get cursor information
        MOV     AH,03h                          ;get cursor status
        INT     10h
        MOV     [vidcurs],CX                    ;save cursor shape
;------get font size
        MOV     AX,1130h                        ;get font information
        XOR     BH,BH                           ;0 = current font
        INT     10h
        MOV     AX,1112h                        ;assume small font
        CMP     CX,8                            ;is it 8x8 font ?
        JZ      SV1                             ;if yes save font
        MOV     AX,1114h
        CMP     CX,16                           ;is it 8x16 font ?
        JZ      SV1                             ;if yes save font
        MOV     AX,1111h                        ;assume 8x14 font
SV1:    MOV     [vidfont],AX                    ;save current font
;------get current color attributes
        MOV     AH,08h                          ;read char and attri
        MOV     BH,[vidpage]                    ;get current video page
        INT     10h
        MOV     [vidattr],AH                    ;save color attribute
        MOV     CL,4                            ;counter
        SHR     AH,CL                           ;high nibble to low
        MOV     [vidbord],AH                    ;save background color
;------is this a VGA system ?
```

```
        MOV     AX,1A00h                                ;read video config.
        INT     10h
        CMP     AL,1Ah                                  ;is it an VGA ?
        JNZ     SV2                                     ;if no then exit
        MOV     AX,1008h                                ;if yes get border color
        INT     10h
        MOV     [vidbord],BH                            ;save border color
;------set font type
        MOV     AX,1114h                                ;8x16 character set
        XOR     BL,BL
        INT     10h
;------set text video mode
SV2:    MOV     AX,0003                                 ;default video text mode
        MOV     BL,[vidmode]                            ;get current mode
        CMP     BL,7                                    ;is it mono text ?
        JZ      SV5                                     ;if yes no change
        INT     10h                                     ;set color text mode
SV5:    MOV     AL,[Normal]
        MOV     [Color],AL
        CALL    CLEAR_SCREEN
        CLC                                             ;clear carry flag
        RET
ENDP    TEXT_VIDEO
;
;
; Restore the users video information which was save by SAVE_VIDEO
; when the program began.
;
;       Input = None
;       Output = Clears the screen
;       Note: uses variables:  [vidmode],[vidpage],[vidcurs],[vidfont]
;                              [vidattr] and [vidbord]
;
PROC    RESTORE_VIDEO
        PUSH    ES
;------test for DOS 3.3 or greater
        MOV     AH,30h                                  ;get dos ver number
        INT     21h
        XCHG    AH,AL                                   ;high byte to ah
        CMP     AX,031Eh                                ;is dos >= 3.30 ?
        JC      REV2                                    ;default to w/b
;------restore original video mode
        MOV     AL,[vidmode]                            ;get video mode no.
        XOR     AH,AH
        INT     10h
;------return display page to 0
        MOV     AX,0500h
        INT     10h
;------restore original font size
        MOV     AX,[vidfont]
        XOR     BL,BL
        INT     10h
;------read cursor configuration
        MOV     AH,03h
        XOR     BH,BH
        INT     10h
;------restore original video page
        MOV     AL,[vidpage]
        MOV     AH,5h
        INT     10h
;------restore cursor shape
        MOV     CX,[vidcurs]
```

```
        MOV     AH,01h
        INT     10h
;------set border color
        MOV     AX,1001h
        MOV     BH,[vidbord]
        INT     10h
;------clear the screen if dos 3.3 or greater
        MOV     AX,0600h                ;scroll & clear window
        MOV     BH,[vidattr]            ;get color attribute
        XOR     CX,CX                   ;start row 0, col 0
        MOV     DX,40h                  ;bios data area
        MOV     ES,DX                   ;ptr to data area
        MOV     DX,[ES:4Ah]             ;get number of columns
        DEC     DX                      ;convert to 0 start
        MOV     DH,[ES:84h]             ;get number of rows
        INT     10h                     ;clear whole screen
        JMP     SHORT REV3
;------clear screen if not dos 3.3 or greater
REV2:   MOV     AL,[Vidattr]            ;system color W/B
        MOV     [Color],AL              ;set color
        CALL    CLEAR_SCREEN            ;blank the screen
REV3:   CLC                             ;clear carry flag
        POP     ES
        RET
ENDP    RESTORE_VIDEO
;
;
;------ Open a Disk File using the file Handle method.
;       Input =  AX = ptr ASCIIZ name of the file.
;                 shared/read/write access assumed.
;       Output = Carry flag set if an opening error or
;                file size in AX and DX; File Handel in BX.
;                is set to beginning of the file.
;       Note:    Registers are not saved.
;
PROC    OPEN
        MOV     DX,AX                   ;ptr to file name string
        MOV     AH,3Dh                  ;open file with handle
        MOV     AL,42h                  ;share/read/write mode
        INT     21h                     ;try to open file.
        JC      OP1                     ;carry = opening error
        MOV     BX,AX                   ;file handle in BX
        XOR     AX,AX                   ;zero AX
        MOV     CX,AX                   ;off set from EOF
        MOV     DX,AX                   ;= 0 in CX AND DX.
        MOV     AX,4202h                ;position at EOF
        INT     21h                     ;size of file in
        PUSH    AX                      ;bytes returns in
        PUSH    DX                      ;AX and DX.
        XOR     AX,AX                   ;zero AX
        MOV     CX,AX                   ;off set from BOF
        MOV     DX,AX                   ;= 0 in CX AND DX.
        MOV     AX,4200h                ;position file pointer
        INT     21h                     ;file ptr to BOF
        POP     DX                      ;size of file returns
        POP     AX                      ;in AX and DX regs.
        CLC                             ;clear carry flag
        JMP     SHORT OP4               ;return to calling prg.
OP1:    CMP     AX,0Ch                  ;is access code wrong?
        JNZ     OP2                     ;if not skip.
        XOR     AX,AX                   ;zero AX if wrong code
OP2:    CMP     AX,6                    ;is error code > 5 ?
```

```
        JC      OP3                                     ;if not skip.
        MOV     AX,6                                    ;end of error table
OP3:    MOV     [ErrCode],AL                            ;save error code
        SHL     AX,1                                    ;multi err code by 2
        MOV     BX,Offset OpenErr                       ;open error table base
        ADD     BX,AX                                   ;add err. offset to base
        CALL    CLEAR_MESSAGE
        MOV     CH,[Warning]                            ;warning color
        MOV     CL,[Color]                              ;save original color
        MOV     [Color],CH                              ;set color
        MOV     AX,0207h                                ;row 3/Col 8
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;output this line to
        db      ' Error! opening ',0                    ;the screen.
        MOV     AX,DX                                   ;File name pointer.
        CALL    DSTR_OUT                                ;output file name.
        MOV     AX,[BX]                                 ;ptr msg string to AX
        CALL    DSTR_OUT                                ;display type of error
        MOV     [Color],CL                               ;restore original color
        CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR                                ;to stop program
        STC                                             ;set carry flag
OP4:    RET
ENDP    OPEN
;
        .DATA
OpenErr dw      Offset OpE1,Offset OpE2,Offset OpE3
        dw      Offset OpE4,Offset OpE5,Offset OpE6,Offset OpE7
OpE1    db      ': Invalid access code. ',0
OpE2    db      ': Invalid function. ',0
OpE3    db      ': File not found. ',0
OpE4    db      ': Path not found. ',0
OpE5    db      ': No handles available. ',0
OpE6    db      ': Access denied. ',0
OpE7    db      ': Error code beyond table. ',0
;
        .CODE
;
;-------Send a ASCII String of a given length in the Data Seg. to the console.
;       Input = AX must points to the first character to send in the string.
;               CL = number of bytes to send
;       Output = None.  AX-DX registers saved.
PROC SUB_DSTR_OUT
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     SI,AX                                   ;pointer to string.
        MOV     DL,CL                                   ;DL = number of chars
        MOV     BH,0                                    ;page 0 asummed
        MOV     BL,[Color]                              ;load color attribute
        MOV     CX,1                                    ;from data segment.
DSTR3:  MOV     AL,[SI]                                 ;get char from string
        CMP     AL,0                                    ;Is it the end ?
        JZ      DSTR4                                   ;exit if end of string.
        MOV     DH,AL                                   ;store char in DH
        MOV     AX,0920h                                ;write 1 space to
        PUSH    SI                                      ;save SI register
        INT     10h                                     ;set color attribute
        MOV     AH,0Eh                                  ;fun.no. teletype mode
        MOV     AL,DH                                   ;get char for DH reg.
```

```
        INT     10h                                     ;char to the console
        POP     SI                                      ;restore DI register
        INC     SI                                      ;point to next char
        DEC     DL                                      ;character counter
        JNZ     SHORT DSTR3                             ;get next character.
DSTR4:  POP     DX
        POP     CX                                      ;restore registers.
        POP     BX
        POP     AX
        RET
ENDP    SUB_DSTR_OUT
;
;
;------Position the cursor on the screen
;       Input = AH (row) AL (column) position in binary numbers.
;       Output = none.  All registers restored.
;       Notes:  upper left hand cornor = 0,0
;               page 0, 25 rows and 80 columns screen assumed.
;               Calling with DH = 25 will hide the cursor off screen!!!
PROC GOTOYX
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     DX,AX
        CMP     DH,26                                   ;is row > 25 ?
        JC      @@LOC1                                  ;if yes default to 0
        MOV     DH,0                                    ;set row to top line
@@LOC1: CMP     DL,80                                   ;is column > 79 ?
        JC      @@LOC2                                  ;if yes defualt to 0
        MOV     DL,0                                    ;column to far right
@@LOC2: MOV     AH,02h                                  ;set cursor funct, no.
        MOV     BH,0                                    ;page 0 assumed
        INT     10h                                     ;position cursor
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
;
;------Advance cursor one column on the screen
;       Input = none
;       Output = none.  All registers restored.
;               page 0, 25 rows and 80 columns screen assumed.
PROC INC_CURSOR
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,0300h
        MOV     BH,AL
        INT     10h
        CMP     DL,79
        JZ      INC1
        INC     DL
        JMP     SHORT INC2
INC1:   INC     DH
INC2:   MOV     AX,0200h
        INT     10h
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
```

```
        POP     AX
        RET
ENDP    INC_CURSOR
;
;
;-------Hide cursor at row 25, column 0 below the last line of the screen.
;       Input = None
;       Output = None
;       Calls  GOTOYX
;       Notes: Page 0 and 25 line text screen assumed.
;
PROC HIDE_CUR
        MOV     AX,1900h                                ;row=25 col = 0
        CALL    GOTOYX                                  ;place cursor
        RET
ENDP    HIDE_CUR
ENDP GOTOYX
;
;-------Send an ASCII string in the Data Segment to the console.
;
;       Input = AX must point to the string. The string must end with
;               a hex zero.  The desired color attribute must be defined
;               in the data segment.
;       Output = None.  All register are saved except SI.
;
PROC DSTR_OUT
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     SI,AX                                   ;pointer to string.
        MOV     BH,0                                    ;page 0 asummed
        MOV     AX,@DATA
        PUSH    DS
        MOV     DS,AX
        MOV     BL,[Color]                              ;load color attribute
        POP     DS
        MOV     CX,1                                    ;frc. data segment.
DSTR1:  MOV     AL,[SI]                                 ;get char from string
        CMP     AL,0                                    ;Is it the end ?
        JZ      DSTR2                                   ;exit if end of string.
        MOV     DL,AL                                   ;store char in DL
        MOV     AX,0920h                                ;write 1 space to
        PUSH    SI                                      ;save SI register
        INT     10h                                     ;set color attribute
        MOV     AH,0Eh                                  ;fun.no. teletype mode
        MOV     AL,DL                                   ;get char for DL reg.
        INT     10h                                     ;char to the console
        POP     SI                                      ;restore SI register
        INC     SI                                      ;point to next char
        JMP     SHORT DSTR1                             ;get next character.
DSTR2:  POP     DX
        POP     CX                                      ;restore registers.
        POP     BX
        POP     AX
        RET
ENDP    DSTR_OUT
;
;
;------Send a two byte unsigned binary number to the screen in decimal form.
;      Input = binary number in AX
;      Output = decimal number to the screen.  Registers restored on return.
```

```
;       Note: this recursive procedure could use up to 40 bytes of stack space.
;             leading zeros are surpressed and no space padding is used.
;             BIN_OUT and DIGIT_OUT must be NEAR procedures.
PROC BIN_OUT    NEAR
        PUSH    AX                                      ;save dividend
        PUSH    BX
        PUSH    CX                                      ;save CX register
        PUSH    DX                                      ;save remainder
        SUB     DX,DX                                   ;zero DX register
        MOV     CX,0Ah                                  ;divisor is 10
        DIV     CX                                      ;AX/10; answer in AX
        CMP     AX,0                                    ;remainder digit in DL.
        JZ      @@BIN                                   ;if yes stop recursion
        CALL    BIN_OUT                                 ;contine recursive call
@@BIN:  CALL    DIGIT_OUT                               ;display digit in DL.
        POP     DX                                      ;previous digit to DL.
        POP     CX                                      ;restore CX register
        POP     BX
        POP     AX                                      ;restore AX register
        RET             ;NOTE: this RET will point to @@BIN: to display
                        ;each digit of the recursions stored in DL register.
                        ;After all digits are displayed it will return to
                        ;the calling program.
;-------- Send a digit (0 to 9) stored in DL register to the screen
PROC DIGIT_OUT  NEAR
        MOV     BH,0                                    ;page 0 asummed
        MOV     BL,[Color]                              ;load color attribute
        MOV     CX,1                                    ;from data segment.
        MOV     AX,0920h                                ;write 1 space to
        INT     10h                                     ;set color attribute
        MOV     AH,0Eh                                  ;fun.no. teletype mode
        MOV     AL,DL                                   ;get char for DL reg.
        OR      AL,30h                                  ;convert to ASCII digit
        INT     10h                                     ;char to the console
        RET
ENDP DIGIT_OUT
END? BIN_OUT
;
;------Get a Char from the keyboard.  (keyboard buffer not cleared before input)
;       Input = none
;       Output = binary ASCII keyboard code in AL
;       Carry flag = extended code.
;       extended codes are converted to control keys by EXT_CHAR
PROC GET_TEXT
GET1:   MOV     AX,0700h                                ;input function number
        INT     21h                                     ;wait for charactor
        CLC                                             ;clear carry flag
        CMP     AL,0                                    ;Is the Char extended?
        JNZ     @@TEXT                                  ;If not extended return
        MOV     AH,07h                                  ;unfilter char input
        INT     21h                                     ;to get extended char.
        CALL    EXT_CHAR                                ;convert extended codes
        JC      GET1                                    ;cf = not on the list
        STC                                             ;set carry flag
@@TEXT: MOV     AH,0                                    ;zero AH register.
        RET                                             ;end of subroutine
ENDP GET_TEXT
;
;
;------Get a Char from the Standard input device.  (keyboard assumed)
;       Input = none
;       Output = binary ASCII keyboard code in AX
```

```
;       Carry flag = extended code.
PROC GET_CHAR
GET0:   MOV     AX,0C07h                ;clear keyboard buffer
        INT     21h                     ;and wait for char.
        CLC                             ;clear carry flag
        CMP     AL,0                    ;Is the Char extended?
        JNZ     @@CHAR                  ;If not extended return
        MOV     AH,07h                  ;unfilter char input
        INT     21h                     ;to get extended char.
        CALL    EXT_CHAR                ;convert extended codes
        JC      GET0                    ;cf = not on the list
        STC                             ;set carry flag
@@CHAR: MOV     AH,0                    ;zero AH register.
        RET                             ;end of subroutine
;
;       A subroutine to convert extended codes to control codes.
;       Input = extended code in AL
;       Output = converted code in AL
;                Carry flag if not one of the Keys listed below:
;       Key     Extended Code  Converted to:   Ctrl-Char   Ctrl-Value
;       Home        47h                         ^A            1h
;       UpArr       48h                         ^E            5h
;       PgUp        49h                         ^R           12h
;       LtArr       4Bh                         ^S           13h
;       RtArr       4Dh                         ^D            4h
;       End         4Fh                         ^F            6h
;       DnArr       50h                         ^X           18h
;       PgDn        51h                         ^C            3h
;       Ins         52h                         ^V           16h
;       Del         53h                         ^G            7h
PROC    EXT_CHAR
        CMP         AL,47h
        JNZ         EXT0
        MOV         AL,1
        JMP         SHORT EXT10
EXT0:   CMP         AL,48h
        JNZ         EXT1
        MOV         AL,5
        JMP         SHORT EXT10
EXT1:   CMP         AL,49h
        JNZ         EXT2
        MOV         AL,12h
        JMP         SHORT EXT10
EXT2:   CMP         AL,4Bh
        JNZ         EXT3
        MOV         AL,13h
        JMP         SHORT EXT10
EXT3:   CMP         AL,4Dh
        JNZ         EXT4
        MOV         AL,4
        JMP         SHORT EXT10
EXT4:   CMP         AL,4Fh
        JNZ         EXT5
        MOV         AL,6
        JMP         SHORT EXT10
EXT5:   CMP         AL,50h
        JNZ         EXT6
        MOV         AL,18h
        JMP         SHORT EXT10
EXT6:   CMP         AL,51h
        JNZ         EXT7
        MOV         AL,3
```

```
        JMP         SHORT EXT10
EXT7:   CMP         AL,52h
        JNZ         EXT8
        MOV         AL,16h
        JMP         SHORT EXT10
EXT8:   CMP         AL,53h
        JNZ         EXT9
        MOV         AL,7
        JMP         SHORT EXT10
EXT9:   STC                                         ;set carry flag
        JMP         SHORT EXT11
EXT10:  CLC                                         ;clear carry flag
EXT11:  RET
ENDP    EXT_CHAR
;
;
ENDP GET_CHAR

;-------Send an ASCII string in the Code segment to the console.
;       The call must be right before the string. The string must end with
;       a hex zero.  This procedure must be called as a near procedure.
;       The desired [color] attribute must be stored in the Data segment.
;       Note: All registers save except SI.
PROC CSTR_OUT   NEAR
        POP     SI
        PUSH    AX                                  ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     BH,0                                ;page 0 asummed
        MOV     BL,[Color]                          ;load color attribute
        MOV     CX,1                                ;from data segment.
CSTR1:  MOV     AL,[CS:SI]                          ;get char from code seg
        CMP     AL,0
        JZ      CSTR2                               ;0 = end of string
        MOV     DL,AL                               ;store char in DL
        MOV     AX,0920h                            ;write 1 space to
        PUSH    SI                                  ;save SI register
        INT     10h                                 ;set color attribute
        MOV     AH,0Eh                              ;func.no.teletype mode
        MOV     AL,DL                               ;get char for DL
        INT     10h                                 ;char to the console
        POP     SI                                  ;restore SI register.
        INC     SI                                  ;point to next char
        JMP     SHORT CSTR1                         ;get next character.
CSTR2:  INC     SI                                  ;SI points to next
        POP     DX                                  ;instruction in code.
        POP     CX                                  ;restore registers.
        POP     BX
        POP     AX
        PUSH    SI                                  ;store the CS pointer.
        RET                                         ;restore CS register.
ENDP    CSTR_OUT
;
;-------Clear a Window
;       Input =  AX = upperleft corner row/col     row 0 - 24
;                BX = lower right corner row/col   col 0 - 79
;                [color] = current attribute from data section
;                page 0 assumed.
;       Output = abort if row or column are out of bounds.
PROC CLEAR_WINDOW
        PUSH    AX                                  ;save registers
```

```
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CMP     BH,AH                   ;is starting row>ending?
        JC      WIN1                    ;exit if yes.
        CMP     BL,AL                   ;is starting col>ending?
        JC      WIN1                    ;exit if yes.
        CMP     BH,25                   ;is row out of bounds?
        JNC     WIN1                    ;exit if yes.
        CMP     BL,80                   ;is col out of bounds?
        JNC     WIN1                    ;exit if yes.
        MOV     CX,AX                   ;starting row/col to CX
        MOV     DX,BX                   ;ending row/col to DX
        MOV     AX,0600h                ;window function no
        MOV     BH,[Color]              ;get active color
        INT     10h                     ;clear the window
WIN1:   POP     DX                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    CLEAR_WINDOW
;
;-------Draw a 17 line 80 column display box for the Restaurant program.
;       It can also be used to clear the display screen and title screen.
;       Input = None
;       Output = None
;       Calls  CSTR_OUT  procedure
PROC DISPLAY_BOX
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AL,[Border]             ;change color attribute
        MOV     [Color],AL              ;for scren output.
        MOV     AX,0400h                ;row 4, column 0
        CALL    GOTOYX                  ;set cursor position
        CALL    CSTR_OUT                ;draw box
        db      201, 78 DUP (205),187
        db      186, 78 DUP (' '),186
        db      186, 78 DUP (' '),186
        db      186, 78 DUP (' '),186
        db      186, 78 DUP (' '),186
        db      186, 78 DUP (' '),186
        db      186, 78 DUP (' '),186
        db      186, 78 DUP (' '),186
        db      186, 78 DUP (' '),186
        db      186, 78 DUP (' '),186
        db      186, 78 DUP (' '),186
        db      186, 78 DUP (' '),186
        db      186, 78 DUP (' '),186
        db      186, 78 DUP (' '),186
        db      186, 78 DUP (' '),186
        db      186, 78 DUP (' '),186
        db      200, 78 DUP (205),188,0
        MOV     AL,[Normal]
        MOV     [Color],AL
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
```

```
ENDP    DISPLAY_BOX
;
;   Display an error message on the screen in row 5 column 3, Normal colors
;       Input  = AX pointer to ASCIIZ string in Data segment
;       Output = Carry Flag set
;       Note:    sends message to screen and wait for key to be pressed.

PROC  ERROR_MESSAGE
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     CL,[Color]                      ;save assigned color
        MOV     AL,[Normal]                     ;set color to normal
        MOV     [Color],AL                      ;for string output
        MOV     AX,0403h                        ;point row 4, col 3
        CALL    GOTOYX                          ;position cursor
        MOV     AX,BX                           ;load string pointer
        CALL    DSTR_OUT                        ;display error message
        MOV     [Color],CL                      ;restore assigned color
        CALL    HIDE_CUR                        ;hide the cursor
        CALL    GET_CHAR                        ;wait until key pressed
        STC                                     ;carry flag = error
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    ERROR_MESSAGE
;
;       Play a series of notes using the 8253 timer chip to produce sound.
;       Input  AX = pointer to 16 bit data string containing
;                   frequency and duration for each pitch.
;                   data string must end with a Hex 0
;       Output None
;
PROC    SOUND
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    BP
        MOV     SI,AX                           ;place data ptr in SI
        IN      AL,61h                          ;get status port B
        OR      AL,3                            ;enable speaker and
        OUT     61h,AL                          ;timer channel 2.
        MOV     AL,0B6h                         ;initialize  channel 2
        OUT     43h,AL                          ;for mode 3
SOU1:   MOV     DX,[SI]                         ;load frequency
        CMP     DX,0                            ;is it the end of str?
        JZ      SOU3                            ;if yes exit else
        INC     SI                              ;advance ptr to
        INC     SI                              ;point to the duration
        MOV     AL,DL                           ;low lsb of frequency
        OUT     42h,AL                          ;send to latch2 port
        MOV     AL,DH                           ;low msb of frequency
        OUT     42h,AL                          ;send to latch2 port
        MOV     AH,0                            ;int function number
        INT     1Ah                             ;get BIOS timer count
        MOV     BX,DX                           ;move lsword to BX
        ADD     BX,[SI]                         ;add duration to BX
        INC     SI                              ;advance ptr to
```

```
        INC     SI                              ;point to next
frequency.
SOU2:   INT     1Ah                             ;get BIOS timer count
        CMP     DX,BX                           ;is count > duration?
        JC      SOU2                            ;if not check again else
        JMP     SHORT SOU1                      ;jump to get next freq.
SOU3:   IN      AL,61h                          ;get byte from port B
        AND     AL,0FCh                         ;turn off speaker bits
        OUT     61h,AL                          ;replace byte in port B
        MOV     DX,1282                         ;default setting
        MOV     AL,DL                           ;get lsb of count
        OUT     42h,AL                          ;send to port 42h
        MOV     AL,DH                           ;get msb of count
        OUT     42h,AL                          ;send to port 42h
        POP     BP
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    SOUND
;
;       Force the numlock key ON by turning on bit 5 in the BIOS data area
PROC    NUM_LOCK_ON
        PUSH    AX
        PUSH    DS
        XOR     AX,AX
        MOV     DS,AX
        MOV     AL,20h
        MOV     SI,417h
        OR      [SI],AL
        POP     DS
        POP     AX
        RET
ENDP    NUM_LOCK_ON
;
;
;-------Clear Display Message.
;       Input = None
;       Output = None
;
PROC    CLEAR_MESSAGE
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AL,[Menu]                       ;Menu color
        MOV     [Color],AL                      ;set color
        MOV     AX,0206h                        ;row 3; col 8
        CALL    GOTOYX                          ;position cursor
        CALL    CSTR_OUT
        db      73 DUP(20h),0
        MOV     AL,[Normal]                      ;normal color
        MOV     [Color],AL                       ;set color
        CALL    HIDE_CUR
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    CLEAR_MESSAGE
;
```

```
;
;
;------ Create a Disk File using the file Handle method.
;       Input = AX = ptr ASCIIZ name of the file.
;                shared/read/write access.
;       Output = Carry flag set if a creating error or
;                File Handel in BX and a file of 0 bytes is open
;
;  **** Caution: This procedure will erase and existing file. ********
;                if the two files have the same name.
;
PROC    CREATE
        MOV     DX,AX                                   ;ptr to file name string
        MOV     AH,3Ch                                  ;create file with handle
        XOR     CX,CX                                   ;normal attributes
        INT     21h                                     ;try to open file.
        JC      CT1                                     ;carry = opening error
        MOV     BX,AX                                   ;file handle in BX
        CLC                                             ;clear carry flag
        JMP     SHORT CT4                               ;return to calling prg.
CT1:    CMP     AX,0Ch                                  ;is access code wrong?
        JNZ     CT2                                     ;if not skip.
        XOR     AX,AX                                   ;zero AX if wrong code
CT2:    CMP     AX,6                                    ;is error code > 5 ?
        JC      CT3                                     ;if not skip.
        MOV     AX,6                                    ;end of error table
CT3:    MOV     [ErrCode],AL                            ;save error code
        SHL     AX,1                                    ;multi err code by 2
        MOV     BX,Offset OpenErr                       ;open error table base
        ADD     BX,AX                                   ;add err. offset to base
        MOV     AX,0107h                                ;row 2, column 8
        CALL    GOTOYX                                  ;position cursor
        MOV     CL,[Color]                              ;save color attribute
        MOV     AL,[Warning]                            ;get new attribute
        MOV     [Color],AL                              ;assign attri to color
        CALL    CSTR_OUT                                ;output this line to
        db      ' Error! creating ',0                    ;the screen.
        MOV     AX,DX                                   ;File name pointer.
        CALL    DSTR_OUT                                ;output file name.
        MOV     AX,[BX]                                 ;ptr msg string to AX
        CALL    DSTR_OUT                                ;display type of error
        MOV     [Color],CL                              ;restore original attri
        STC                                             ;set carry flag
CT4:    RET
ENDP    CREATE
;
;------ Delete a Disk File.
;       Input = AX = ptr ASCIIZ name of the file.
;                shared/read/write access.
;       Output = Carry flag set if a delete error.
;
;  **** Caution: This procedure will erase and existing file. ********
;
PROC    DELETE_FILE
        MOV     DX,AX                                   ;ptr to file name string
        MOV     AH,41h                                  ;delete file function no
        INT     21h                                     ;try to open file.
        JC      DT1                                     ;carry = delete error
        CLC                                             ;clear carry flag
        JMP     SHORT DT4                               ;return to calling prg.
DT1:    CMP     AX,0Ch                                  ;is access code wrong?
        JNZ     DT2                                     ;if not skip.
```

```
        XOR     AX,AX                                   ;zero AX if wrong code
DT2:    CMP     AX,6                                    ;is error code > 5 ?
        JC      DT3                                     ;if not skip.
        MOV     AX,6                                    ;end of error table
DT3:    MOV     [ErrCode],AL                            ;save error code
        SHL     AX,1                                    ;multi err code by 2
        MOV     BX,Offset OpenErr                       ;open error table base
        ADD     BX,AX                                   ;add err. offset to base
        MOV     AX,0107h                                ;row 2, column 8
        CALL    GOTOYX                                  ;position cursor
        MOV     CL,[Color]                              ;save color attribute
        MOV     AL,[Warning]                            ;get new attribute
        MOV     [Color],AL                              ;assign attri to color
        CALL    CSTR_OUT                                ;output this line to
        db      ' Error! deleting ',0                    ;the screen.
        MOV     AX,DX                                   ;File name pointer.
        CALL    DSTR_OUT                                ;output file name.
        MOV     AX,[BX]                                 ;ptr msg string to AX
        CALL    DSTR_OUT                                ;display type of error
        MOV     [Color],CL                              ;restore original attri
        STC                                             ;set carry flag
DT4:    RET
ENDP    DELETE_FILE
;
;
;------Send a 16 bit unsigned binary number to the screen in decimal form
;               in EGA and VGA Graphics Mode 10h page 0
;       Input =  AX = binary number     CX = total number of digits
;               the number is padded with leading zeros until CX digits .
;               [color] = current attribute from data section
;       Output = None
;       Video mode:  10h  640 x 350  16 colors   row = 25 Col = 80
;
;      Note: the calling procedure must make sure that the number is CX is
;            large enough to display all the digits of the number in AX.  This
;            procedure can be used when leading zeros are needed.
;
PROC BIN_DIG_OUT    NEAR
        PUSH    AX                                      ;save dividend
        PUSH    BX
        PUSH    CX                                      ;save CX register
        PUSH    DX                                      ;save remainder
        SUB     DX,DX                                   ;zero DX register
        MOV     BX,0Ah                                  ;divisor is 10
        DIV     BX                                      ;AX/10; answer in AX
        DEC     CX                                      ;remainder digit in DL.
        JZ      @BIN                                    ;if yes stop recursion
        CALL    BIN_DIG_OUT                             ;contine recursive call
@BIN:   CALL    DIGIT_OUT                               ;display digit in DL.
        POP     DX                                      ;previous digit to DL.
        POP     CX                                      ;restore CX register
        POP     BX
        POP     AX                                      ;restore AX register
        RET                 ;NOTE: this RET will point to @@BIN: to display
ENDP BIN_DIG_OUT            ;each digit of the recursions stored in DL register.
                            ;After all digits are displayed it will return to
                            ;the calling program.
;
;
;-------Ask a yes or no question.
;       Input = None
;       Output = Carry Flag = YES
```

```
;
PROC    EXIT_YN
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    CLEAR_MESSAGE
        MOV     AL,[Warning]                            ;warning color
        MOV     [Color],AL                              ;set color
        MOV     AX,020Bh                                ;row 3/Col 12
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      ' Exit to DOS ?  [Y]/N ',0
        MOV     AL,[Normal]                              ;normal color
        MOV     [Color],AL                               ;set color
EX1:    CALL    HIDE_CUR
        CALL    GET_CHAR
        AND     AL,5Fh                                  ;turn off bits 6 & 8
        CMP     AL,'N'                                   ;is it No?
        JZ      EX4                                      ;if yes exit
        CMP     AL,0Dh                                   ;is it <Enter>?
        JNZ     EX2                                      ;if not continue
        STC                                              ;set carry flag
        JMP     SHORT EX5                                ;exit
EX2:    CMP     AL,'Y'                                   ;is it Yes?
        JNZ     EX1                                      ;if not get another
        STC                                              ;set carry flag
        JMP     SHORT EX5                                ;exit
EX4:    CLC                                              ;clear carry flag
EX5:    POP     DX                                       ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    EXIT_YN
;
;
;-------Clear the screen and place the cursor in position 0,0
;       Input = None    Color - current [color] attribute from data section
;       Output = None   Border color is also set the same as the screen.
;       Notes: All registers saved.  25 rows and 80 columns page 0 assumed.
;
PROC CLEAR_SCREEN
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     BH,[Color]                              ;color attribute
        MOV     AX,0700h                                ;initialize window func
        SUB     CX,CX                                   ;row/col = 0,0
        MOV     DX,184Fh                                ;row/col = 24,79
        INT     10h                                     ;clear screen window
        MOV     BH,[Color]                              ;color attribute
        MOV     CL,4                                    ;shift background color
        SHR     BH,CL                                   ;to the lower 4 bytes.
        MOV     AX,1001h                                ;function number
        INT     10h                                     ;set screen border
        MOV     AH,2h                                   ;set cursor position
        MOV     BH,0                                    ;page 0, row,col to DX
        MOV     DX,CX                                   ;position cursor at the
        INT     10h                                     ;the top left cornor.
        POP     DX                                      ;restore registers
```

```
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    CLEAR_SCREEN
;
;
;
;-------Draw a 4 line 80 column menu box.  Starting at row 0-21, column 0.
;       Input = AX = Row,Columns cursor position.  Column must be 0
;       Output = None
;       Calls  CSTR_OUT  procdeures
PROC MENU_BOX
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        XOR     AL,AL                           ;set column to 0
        CALL    GOTOYX                          ;set cursor position
        MOV     DL,[Color]                      ;save original Color
        MOV     AL,[Menu]                       ;clange color attribute
        MOV     [Color],AL                      ;for scren output.
        CALL    CSTR_OUT                        ;draw menu box
        db      201, 78 DUP (205),187           ;all except last byte.
        db      186, 78 DUP (' '),186           ;CSTRU_OUT will cause
        db      186, 78 DUP (' '),186           ;the screen to scroll
        db      200, 78 DUP (205),0             ;in row 25, col 80.
        MOV     BL,AL                           ;color to BL
        MOV     AH,09h                          ;write char funct no.
        MOV     AL,188                          ;last character of box
        MOV     BH,0                            ;page 0 assumed
        MOV     CX,1                            ;number of bytes
        INT     10h                             ;write last byte
        MOV     [Color],DL                      ;restore original Color
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    MENU_BOX
;
;-------Draw a 4 line 80 column menu box.  Starting at row 0-21, column 0.
;       Input = AX = Row,Columns cursor position.  Column must be 0
;       Output = None
;       Calls  CSTR_OUT  procdeures
PROC  CLEAR_BOX
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     DL,[Color]                      ;save original Color
        MOV     AL,[Normal]                     ;clange color attribute
        MOV     [Color],AL                      ;for scren output.
        XOR     AL,AL                           ;set column to 0
        MOV     BL,79
        MOV     BH,AH
        ADD     BH,4
        CALL    CLEAR_WINDOW
        MOV     [Color],DL                      ;restore original Color
        POP     DX
        POP     CX
        POP     BX
```

```
        POP     AX
        RET
ENDP  CLEAR_BOX
;
;
; Display the DOS extended error message return
; by calling Int 21h function 59h - Get extended error information.  If the
; error code is less then 36 the error string is presented.  If the error
; number is 36 or larger the number is print to the screen.
;   Input = None     data bytes [ErrCode],[Color], and [Normal] assumed.
;   Output = Error number in [ErrCode]
;
PROC    DISPLAY_ERROR
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        MOV     [ErrCode],AL                    ;save AL register
        CALL    CLEAR_MESSAGE
        MOV     CH,[Warning]                    ;warning color
        MOV     CL,[Color]                      ;save original color
        MOV     [Color],CH                      ;set color
        MOV     AX,0207h                        ;row 3/Col 8
        CALL    GOTOYX                          ;set cursor
;------request extended error information
        XOR     BX,BX                           ;get extended error
        MOV     AX,5900h                        ;information  from
        INT     21h                             ;DOS system
        XOR     AH,AH                           ;zero hi byte
        CMP     AL,0                            ;was an error found?
        JZ      ERR01                           ;if NO display message
        CMP     AL,37                           ;is code < 37 ?
        JC      ERR01                           ;if YES display mess
        CALL    CSTR_OUT
        db      ' DOS Error Number: ',0
        CALL    BIN_OUT
        MOV     AX,37                           ;37 = unknown DOS error
ERR01:  SHL     AX,1                            ;error number x 2
        MOV     BX,Offset ErrStr                ;pointer to table base
        ADD     BX,AX                           ;BX ptr to error ptr.
        CALL    CSTR_OUT
        db      ' DOS Error: ',0
        MOV     AX,[BX]                         ;load ptr to error str.
        CALL    DSTR_OUT                        ;send string to screen
        CALL    CSTR_OUT
        db      '  Press Any Key. ',0
        MOV     [Color],CL                      ;restore original color
        CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR
        POP     ES
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DISPLAY_ERROR
        .DATA
ErrStr  dw      Err00,Err01,Err02,Err03,Err04,Err05,Err06,Err07,Err08,Err09
        dw      Err10,Err11,Err12,Err13,Err14,Err15,Err16,Err17,Err18,Err19
        dw      Err20,Err21,Err22,Err23,Err24,Err25,Err26,Err27,Err28,Err29
```

```
        dw      Err30,Err31,Err32,Err33,Err34,Err35,Err36,Err37
Err00   db      'no error found',0
Err01   db      'function number invalid',0
Err02   db      'file not found',0
Err03   db      'path not found',0
Err04   db      'to many open files',0
Err05   db      'access denied',0
Err06   db      'handle invalid',0
Err07   db      'memory control blocks destroyed',0
Err08   db      'insufficient memory',0
Err09   db      'memory block address invalid',0
Err10   db      'environment invalid',0
Err11   db      'format invalid',0
Err12   db      'access code invalid',0
Err13   db      'data invalid',0
Err14   db      'unkown unit',0
Err15   db      'disk drive invalid',0
Err16   db      'attempted to remove current directory',0
Err17   db      'not same device',0
Err18   db      'no more files',0
Err19   db      'disk write-protected',0
Err20   db      'unkown unit',0
Err21   db      'drive not ready',0
Err22   db      'unkown command',0
Err23   db      'data error (crc)',0
Err24   db      'bad request structure length',0
Err25   db      'seek error',0
Err26   db      'unkown media type',0
Err27   db      'sector not found',0
Err28   db      'printer out of paper',0
Err29   db      'write fault',0
Err30   db      'read fault',0
Err31   db      'general failure',0
Err32   db      'sharing violation',0
Err33   db      'lock violation',0
Err34   db      'disk change invalid',0
Err35   db      'FCB unavailable',0
Err36   db      'sharing buffer exceeded',0
Err37   db      'check DOS documentation',0

        .CODE
;
;
; INT24h  Substitute critical-error handler to tell DOS to Retry or Fail errors
and
; return to the calling program.  This subroutine will redirect DOS's
;attempt back to the calling program.
; Note : The Abort is converted to what DOS calls a Fail and will return
;        control back to the calling program with an error code in AL.
; INT23h Ignore the control C break command from the keyboard.
;
PROC    INTERRUPT_HANDLER
;------install critical-error handler
        PUSH    DS
        MOV     DX,Seg INT24
        MOV     DS,DX
        MOV     DX,Offset INT24
        MOV     AX,2524h
        INT     21h
;------install ^C error handler (ignore ^C breaks)
        MOV     DX,Seg INT23
        MOV     DS,DX
```

```
        MOV     DX,Offset INT23
        MOV     AX,2523h
        INT     21h
        POP     DS
        RET
;------substitute interrupt 23h
PROC    INT23   FAR
        XOR     AX,AX
        IRET
ENDP    INT23
;------substitute interrupt 23h
PROC    INT24   FAR
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    SI
        PUSH    DI
        PUSH    BP
        PUSH    DS
        PUSH    ES
        MOV     DX,AX                                   ;save in DX
        MOV     AX,@DATA                                ;get data segment
        MOV     DS,AX                                   ;assign data segment
        MOV     CL,[Color]                              ;get current attribute
        MOV     AL,[Warning]                            ;get warning color
        MOV     [Color],AL                              ;assign to color
        MOV     AX,0220h                                ;row 4 / col 0
        CALL    GOTOYX                                  ;position cursor
        CALL    CSTR_OUT                                ;send string to screen
        db      7,7,' Error: Press R to Retry or A to Abort. ',0
        CALL    HIDE_CUR
CRT1:   MOV     AH,6                                    ;Dos function number
        MOV     DL,0FFh                                 ;get char input from
        INT     21h                                     ;the keyboard.
        JZ      CRT1                                    ;if not char try again
        AND     AL,5Fh                                  ;make capital letter
        MOV     AH,AL                                   ;input to AH
        MOV     AL,3                                    ;3 = fail the DOS call
        CMP     AH,'A'                                  ;is it an Abort ?
        JZ      CRT2                                    ;exit if an Abort
        MOV     AL,1                                    ;1 = retry the DOS call
        CMP     AH,'R'                                  ;is it a Retry ?
        JNZ     CRT1                                    ;if not get new char
CRT2:   MOV     DL,AL                                   ;save return code
        MOV     AL,[Menu]                               ;get menu color
        MOV     [Color],AL                              ;assign to color
        MOV     AX,0220h                                ;row 4 / col 0
        CALL    GOTOYX                                  ;position cursor
        CALL    CSTR_OUT                                ;send string to screen
        db      '                                        ',0
        MOV     [Color],CL                              ;assign original attri
        MOV     AX,DX                                   ;restore DOS return code
        POP     ES
        POP     DS
        POP     BP
        POP     DI
        POP     SI
        POP     DX
        POP     CX
        POP     BX
        IRET
ENDP    INT24
```

```
ENDP    INTERRUPT_HANDLER
;
;
;-------The Pause for set for 1/2 second.
;       Input = None
;       Output = None
PROC  PAUSE
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        XOR     AX,AX                           ;Get ticks function no.
        INT     1Ah                             ;get Dos timer ticks
        MOV     BX,DX                           ;low byte ticks to BX
        ADD     BX,9                            ;9 = 1/2 second
PA1:    XOR     AX,AX                           ;Get ticks function no.
        INT     1Ah                             ;get Dos timer ticks
        CMP     BX,DX                           ;is time run out?
        JNC     PA1                             ;if not loop again
        CLC                                     ;clear carry flag
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP  PAUSE
;
;
;------Open the configuration file for use by the FeedBack program
;      Input =  name of file in [FileNa]
;      Output = Carry flag set = critical DOS error.
;
PROC    OPEN_CONFIG
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
;------locate end of search string
        MOV     AX,DS
        MOV     ES,AX
        MOV     CX,68                           ;max length of string
        MOV     BX,Offset Search                ;ptr to first byte
        MOV     AL,0                            ;looking for EndOfStr
KE1:    INC     BX                              ;ptr to next byte
        CMP     [BX],AL                         ;if this it ?
        JZ      KE2                             ;continue! I found it.
        LOOP    KE1                             ;if NO look at next byte
        STC
        JMP     SHORT KE9                       ;if no match exit
;------backup until finding the last \
KE2:    MOV     CX,12
        MOV     AL,'\'
        DEC     BX
        CMP     [BX],AL
        JZ      KE3
        LOOP    KE2
        STC
        JMP     SHORT KE9                       ;if no match exit
;------copy file name to end of path
KE3:    INC     BX
        MOV     DI,BX                           ;destination ptr
        MOV     SI,Offset FileNa                ;source ptr
```

```
        MOV     CX,13                                   ;number of bytes
        CLD                                             ;auto inc SI & DI
        REP     MOVSB                                   ;copy all 13 bytes
;------open file and save file handle
        MOV     AX,Offset Search                        ;ptr to path + file name
        CALL    OPEN                                    ;open key file
        JC      KE9                                     ;goto main menu on error
        MOV     [CFGHd],BX                              ;save data.fil handle
;------set disk drive of open file
        XOR     AX,AX                                   ;zero to [DiskDr] =
        MOV     [FileDr],AL                             ;default drive
        MOV     BX,Offset Search                        ;ptr to path + filename
        MOV     AX,[BX]                                 ;get first two bytes
        CMP     AH,':'                                  ;is a drive given?
        JNZ     KE8                                     ;if NO will use default
        SUB     AL,64                                   ;convert to hex value
        JC      KE8                                     ;if error continue
        MOV     [FileDr],AL                             ;save drive of file
KE8:    CLC                                             ;clear carry flag
KE9:    POP     ES
        POP     DX
        POP     CX
        POP     BX
        RET
ENDP    OPEN_CONFIG
;
;
;-------Read the configuration file into memory.  The data strings are
;       stored on the heap and a ptr table to the entries is in the data
;       section. In the table the first word is the type of data and the
;       second word is a ptr to the offset in the heap.
;
;       Input = none.
;       Output = carry flag if error in the file.
;       BX = FulBuf pointer
;       BP = heap pointer
;       DX = end of CFGTbl
;
PROC    READ_CONFIG
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,DS                                   ;assign ES = DS
        MOV     ES,AX
        XOR     AX,AX
        MOV     [EOF],AL                                ;set EndOfFile = FALSE
        MOV     BP,AX                                   ;ptr to bottom heap
        MOV     BX,Offset CFGTbl                        ;ptr to table
        MOV     DX,Offset CFGTbl + 636                  ;ptr to end of table
        MOV     AX,[CFGHd]                              ;get file handle
        CALL    GOTO_TOP                                ;file ptr to BegOfFile
        JNC     FZR0                                    ;continue if No error
        JMP     FXR14                                   ;exit on DOS error
;------locate a data line in the config file.
FZR0:   CALL    PRINT_WAIT_MESS                         ;inform user of search
FZR1:   MOV     AX,[CFGHd]                              ;get file handle
        CALL    READ_LINE                               ;1 line from data file
        JNC     FZR2                                    ;not EndOfFile
        MOV     AL,0FFh                                 ;mark EndOfFile true
        MOV     [EOF],AL                                ;<> 0 = True
FZR2:   CALL    REMOVE_SPACES                           ;remove leading spaces
```

```
;------is it and SF1 data line?
        MOV     CX,3                            ;loop counter
        MOV     DI,Offset SF1Typ +1             ;ptr to SF1
        MOV     SI,Offset FilBuf                ;ptr to data file line
        CLD                                     ;auto inc DI and SI
        REPZ    CMPSB                           ;are the bytes = ?
        JNZ     FZR3                            ;if NO goto next test
        MOV     WORD PTR [BX],5                 ;identify as SF1
        JMP     SHORT FZR7                      ;store in memory
;------is it and SF2 data line?
FZR3:   MOV     CX,3                            ;loop counter
        MOV     DI,Offset SF2Typ +1             ;ptr to SF2
        MOV     SI,Offset FilBuf                ;ptr to data file line
        CLD                                     ;auto inc DI and SI
        REPZ    CMPSB                           ;are the bytes = ?
        JNZ     FZR4                            ;if NO goto next test
        MOV     WORD PTR [BX],4                 ;identify as SF2
        JMP     SHORT FZR7                      ;store in memory
;------is it and PER data line?
FZR4:   MOV     CX,3                            ;loop counter
        MOV     DI,Offset PERTyp +1             ;ptr to PER
        MOV     SI,Offset FilBuf                ;ptr to data file line
        CLD                                     ;auto inc DI and SI
        REPZ    CMPSB                           ;are the bytes = ?
        JNZ     FZR5                            ;if NO goto next test
        MOV     WORD PTR [BX],3                 ;identify as PER
        JMP     SHORT FZR7                      ;store in memory
FZR5:   MOV     CX,3                            ;loop counter
        MOV     DI,Offset SUPTyp +1             ;ptr to SUP
        MOV     SI,Offset FilBuf                ;ptr to data file line
        CLD                                     ;auto inc DI and SI
        REPZ    CMPSB                           ;are the bytes = ?
        JNZ     FZR6                            ;if NO goto next test
        MOV     WORD PTR [BX],2                 ;identify as SUP
        JMP     SHORT FZR7                      ;store in memory
;------is it a SUB data line?
FZR6:   MOV     CX,3                            ;loop counter
        MOV     DI,Offset SUBTyp +1             ;ptr to SUB
        MOV     SI,Offset FilBuf                ;ptr to data file line
        CLD                                     ;auto inc DI and SI
        REPZ    CMPSB                           ;are the bytes = ?
        JNZ     FZR8                            ;if NO goto next test
        MOV     WORD PTR [BX],1                 ;identify as SUB
;------store a string of word integers
FZR7:   INC     BX                              ;point to the next
        INC     BX                              ;word in table
        MOV     [BX],BP                         ;ptr to the table
        INC     BX                              ;point to the next
        INC     BX                              ;word in table
        CALL    COPY_NUMBERS                    ;copy data into memory
        JMP     SHORT FXR11                     ;read the next line
;------is it a group title?
FZR8:   MOV     SI,Offset FilBuf                ;ptr to data file line
        CMP     BYTE PTR [SI],"{"               ;is this a group title?
        JNZ     FZR9                            ;if NO goto next test
        MOV     WORD PTR [BX],7                 ;identify group title
        JMP     SHORT FXR10                     ;copy to memory
;------is it a factor title?
FZR9:   MOV     SI,Offset FilBuf                ;ptr to data file line
        CMP     BYTE PTR [SI],"["               ;is this a factor title?
        JNZ     FXR11                           ;if NO goto next test
        MOV     WORD PTR [BX],6                 ;identify group title
```

```
;------store an asciiz string in memory
FXR10:  INC     BX                              ;point to the next
        INC     BX                              ;word in table
        MOV     [BX],BP                         ;ptr to the table
        INC     BX                              ;point to the next
        INC     BX                              ;word in table
        CALL    COPY_TO_HEAP                     ;copy data into memory
;------is this the last line ?
FXR11:  XOR     AL,AL                           ;zero AX register
        CMP     AL,[EOF]                        ;is EndOfFile TRUE?
        JNZ     FXR13                           ;not zero = EndOfFile
;------is this the end of the CFGTbl ?
        CMP     BX,DX                           ;is it end of table?
        JNC     FXR12                           ;if yes display error
        JMP     FZR1                            ;else read next line
FXR12:  XOR     AX,AX                           ;0 = to many factors
        CALL    CFG_ERR                         ;to many CFG items
        STC                                     ;set error flag
        JMP     SHORT FXR14                     ;exit on error
FXR13:  MOV     WORD PTR [BX],8                 ;mark EndOfTable
        CALL    CHECK_CFG                       ;look for file errors
        JC      FXR14                           ;exit if file error
        CALL    SET_DAT_VAR                     ;set DRecLn & RNKLn
        CALL    SHOW_YN                         ;show config Y/N?
        JNC     FXR14                           ;carryflag = YES
        CALL    DISPLAY_CFG                     ;show user the data
        CLC                                     ;clear cf = no error
FXR14:  PUSHF                                   ;save flag register
        MOV     AX,[CFGHd]                      ;get file handle
        CALL    CLOSE                           ;close config file
        JC      FXR15                           ;exit on DOS error
        XOR     AX,AX                           ;zero to register
        MOV     [CFGHd],AX                      ;mark file closed
FXR15:  POPF                                    ;restore flag register
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    READ_CONFIG
;
;
PROC    SET_DAT_VAR
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------the number of data points = number of answer files in config.
        XOR     CX,CX                           ;zero counter
        MOV     SI, Offset CFGTbl - 4           ;ptr to table
        MOV     AX,4                            ;length 2 table entries
SDV1:   ADD     SI,AX                           ;ptr to next type
        CMP     WORD PTR [SI],8                 ;is it EndOfTable?
        JZ      SDV2                            ;OK! exit loop counter
;------is this an answer file ?
        CMP     WORD PTR [SI],6                 ;is it an answer file?
        JNC     SDV1                            ;if NO continue loop
        INC     CX                              ;count = count + 1
        JMP     SHORT SDV1                      ;continue loop
;------DRecLn = no of answer files + 20
SDV2:   MOV     BX,CX                           ;save number of points
        ADD     CX,20                           ;add ID length
```

```
        MOV     [DRecLn],CX                             ;save data rec length
;------ RNKLn = no of answer files * 7
        MOV     AX,BX                                   ;bar count to AX
        MOV     CL,7                                    ;size of rank record
        MUL     CL
        MOV     [RNKLn],AX                              ;save size of rank file
        CLC
        POP     DX
        POP     CX
        POP     BX
        PCP     AX
        RET
ENDP    SET_DAT_VAR
;
; Copy number string into memory from the configuration file.
;       Input = BP ptr to heap storage area
;               line of text in [FilBuf]
;       Output = byte integers stored in heap
;                unsigned integer from 1 - 200
;                FF = number larger than 200
;                hex 0 marks the end of the integer string
;
PROC    COPY_NUMBERS
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    DI
        XOR     AX,AX                                   ;zero number holder
        MOV     DI,10                                   ;multipler
        MOV     BX, Offset FilBuf +2                    ;ptr past file name
NUM0:   XOR     AX,AX                                   ;zero the value
NUM1:   INC     BX                                      ;ptr to next char
        CMP     BYTE PTR [BX],0h                        ;is it endofline?
        JZ      NUM6                                    ;if yes exit
        CMP     BYTE PTR [BX],'l'                       ;is it lower case "l"
        JNZ     NUM2                                    ;if NO goto next test
        MOV     BYTE PTR [BX],"1"                       ;if YES convert to a 1
NUM2:   CMP     BYTE PTR [BX],'O'                       ;is it upper case "O"
        JNZ     NUM2A                                   ;if NO goto next test
        MOV     BYTE PTR [BX],"0"                       ;if YES convert to zero
NUM2A:  CMP     BYTE PTR [BX],';'                       ;is it comment marker?
        JZ      NUM6                                    ;if yes exit
        CMP     BYTE PTR [BX],":"                       ;is it a digit?
        JNC     NUM4                                    ;if no get next char
        CMP     BYTE PTR [BX],"0"                       ;is it a digit?
        JC      NUM4                                    ;if not get next char
;------process the digit
NUM3:   MUL     DI                                      ;multiply by 10
        XOR     DX,DX                                   ;zero register
        MOV     DL,[BX]                                 ;get digit
        SUB     DL,"0"                                  ;concert to hex
        ADD     AX,DX                                   ;save value in AX
        JMP     SHORT NUM1                              ;get next digit
;------process the number if any and continue
NUM4:   CMP     AX,0                                    ;is number = 0 ?
        JZ      NUM1                                    ;if YES skip it
NUM5:   MOV     [BP],AL                                 ;store value in memory
        INC     BP
        JMP     SHORT NUM0                              ;get next char
;------process the number if any and exit
NUM6:   CMP     AX,0                                    ;is number = 0 ?
```

```
        JZ      NUM8                                    ;if YES skip it
        CMP     AX,201                                  ;is it larger than 200?
        JC      NUM7                                    ;if NO Store number
        MOV     AL,0FFh                                 ;else mark overflow
NUM7:   MOV     [BP],AL                                 ;store value in memory
        INC     BP
NUM8:   XOR     AX,AX
        MOV     [BP],AL                                 ;place a zero integer
        INC     BP                                      ;to mark the EndOfStr
        CLC
        POP     DI
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    COPY_NUMBERS
;
;
; Copy a title string into memory from the configuration file.
;       Input = BP ptr to heap storage area
;               line of text in [FilBuf]
;       Output = asciiz text string stored in the heap
;                byte = hex 0 marks the end of the string
;
PROC    COPY_TO_HEAP
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        XOR     AX,AX                                   ;zero number holder
        MOV     BX, Offset FilBuf                       ;ptr past file name
        MOV     AH,[BX]                                 ;get staring marker
        MOV     AL,"]"                                  ;close bracket
        CMP     AH,"["                                  ;is open bracket?
        JZ      CTH0                                    ;if yes goto next test
        MOV     AL,"}"                                  ;else close prentheses
CTH0:   MOV     AH,AL                                   ;endOfString marker
CTH1:   INC     BX                                      ;ptr to next char
        CMP     BYTE PTR [BX],0h                        ;is it endofline?
        JZ      CTH2                                    ;if yes exit
        CMP     BYTE PTR [BX],AH                        ;is end of string?
        JZ      CTH2                                    ;if yes exit
;------place the character in the text string
        MOV     AL,[BX]                                 ;get char from buffer
        MOV     [BP],AL                                 ;place char in heap
        INC     BP                                      ;advance heap pointer
        JMP     SHORT CTH1                              ;get next char
;------place a hex 0 at the end of the string
CTH2:   XOR     AX,AX                                   ;0 = EndOfString
        MOV     [BP],AL                                 ;store value in memory
        INC     BP                                      ;advance heap ptr
        CLC
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    COPY_TO_HEAP
;
;  Remove all leading spaces from the FilBuf string.
;       Input = none  (asciiz string in FilBuf)
```

```
;       Output = carry flag = buffer overflow
;
PROC    REMOVE_SPACES
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        MOV     AX,DS                           ;set register ES
        MOV     ES,AX                           ;to data segment
;------is there a leading space ?
REM1:   MOV     BX,Offset FilBuf                ;ptr to buffer
        CMP     BYTE PTR [BX], ' '              ;is 1st char a space?
        JNZ     REM4                            ;if NO then exit
;------compute the length of the string
        MOV     AX,Offset FilBuf                ;address of buffer
        CALL    STR_LENGTH                      ;get string length
        JC      REM5                            ;exit if overflow
        MOV     CX,AX                           ;length to CX
;------overwrite the leading space
REM3:   INC     CX                              ;so hex 0 gets moved
        MOV     SI, Offset FilBuf + 1           ;source ptr
        MOV     DI, Offset FilBuf               ;destination ptr
        CLD                                     ;auto inc DI and SI
        REP     MOVSB                           ;move chars to the left
        JMP     SHORT REM1                      ;check for more spaces
REM4:   CLC                                     ;exit no error
REM5:   POP     ES
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    REMOVE_SPACES
;
;  computer the length of the ASCii string in the DS segment
;       Input = AX = string offset address
;       Output = AX = number of characters in the string
;             assumed string less than 256 bytes
;             carry flag if overflow
;
PROC    STR_LENGTH
        PUSH    BX
        PUSH    CX
        MOV     BX,AX                           ;ptr to buffer
        XOR     CX,CX                           ;zero counter
LEN1:   CMP     BYTE PTR [BX],0h                ;is it end of str?
        JZ      LEN2                            ;if yes then done
        INC     BX                              ;ptr to next byte
        INC     CL                              ;counter + 1
        JNC     LEN1                            ;loop if not overflow
LEN2:   MOV     AX,CX                           ;place count in AX
        POP     CX
        POP     BX
        RET
ENDP    STR_LENGTH
;
;
;       Input = AX = error type
;       Output = display message on the screen
PROC    CFG_ERR
        PUSH    AX
```

```
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CMP     AX,8                                    ;is data out of range?
        JC      GER1                                    ;if NO continue
        MOV     AX,8                                    ;if YES error = 8
GER1:   SHL     AX,1                                    ;multi err code by 2
        MOV     BX,Offset CFGErr                        ;open error table base
        ADD     BX,AX                                   ;add err. offset to base
        MOV     AL,[Warning]                            ;warning color
        MOV     CL,[Color]                              ;save original color
        MOV     [Color],AL                              ;set color
        MOV     AX,0207h                                ;row 3/Col 8
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;output this line to
        db      ' CFG file error!  ',0                  ;the screen.
        MOV     AX,[BX]                                 ;ptr msg string to AX
        CALL    DSTR_OUT                                ;display type of error
        MOV     AX,1500h                                ;row 21,column 0
        CALL    MENU_BOX                                ;draw menu box
        MOV     AL,[Menu]                               ;set color = menu
        MOV     [Color],AL
        MOV     AX,1607h                                ;row 23,column 5
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'This configuration file can not be used until the error is '
        db      'corrected.',0
        MOV     AX,1711h                                ;row 23,column 5
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      "Press any key to return to the FeedBack Main Menu.",0
        MOV     [Color],CL                               ;restore original color
        CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    CLOSE_ALL_FILES
        CALL    GET_CHAR
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    CFG_ERR
;

        .DATA
CFGErr  dw      Offset CfE0,Offset CfE1,Offset CfE2,Offset CfE3
        dw      Offset CfE4,Offset CfE5,Offset CfE6,Offset CfE7,Offset CfE8
CfE0    db      ': To many Factors.  Divide into two files.',0
CfE1    db      ': The first data line is not a Group name. ',0
CfE2    db      ': Data line after Group name is not a Factor name. ',0
CfE3    db      ': Data line after Factor name is not a Self file type. ',0
CfE4    db      ': An answer file name is not preceded by a Self file. ',0
CfE5    db      ': An answer file name is missing question numbers. ',0
CfE6    db      ': A question number is greater than 200. ',0
CfE7    db      ': A Group contains both SF1 and SF2 file types. ',0
CfE8    db      ': Error code beyond table. ',0
;
        .CODE
;
;   Check the CFG information entered in the heap for errors.
;       Input = none
;       Output = carry flag = data error
```

```
;
PROC    CHECK_CFG
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,4                                    ;offset factor in table
        MOV     BX,Offset CFGTbl                        ;ptr to CFG table
        MOV     DX,BX                                   ;save the ptr in dx
;------is the first data member a group name?
        MOV     CX,1                                    ;type error number
        CMP     WORD PTR [BX],7                         ;is it a group name?
        JZ      CKE0                                    ;if YES goto next test
        JMP     CKE15                                   ;else display error
;------is each group name followed by a factor name ?
CKE0:   INC     CX                                      ;type error number
        MOV     BX,DX                                   ;ptr to CFG table
        SUB     BX,AX                                   ;adjust ptr for an add
CKE1:   ADD     BX,AX                                   ;ptr to next entry
        CMP     WORD PTR [BX],8                         ;is it end of table?
        JZ      CKE2                                    ;if yes next test
        CMP     WORD PTR [BX],7                         ;is this a group name?
        JNZ     CKE1                                    ;if NO look somemore
        CMP     WORD PTR [BX + 4],6                     ;is next a factor name?
        JZ      CKE1                                    ;if YES loop
        JMP     CKE15                                   ;else display error
;------is each factor name followed by a Self file name ?
CKE2:   INC     CX                                      ;type error number
        MOV     BX,DX                                   ;ptr to CFG table
        SUB     BX,AX                                   ;adjust ptr for an add
CKE3:   ADD     BX,AX                                   ;ptr to next entry
        CMP     WORD PTR [BX],8                         ;is it end of table?
        JZ      CKE4                                    ;if yes next test
        CMP     WORD PTR [BX],6                         ;is this a group name?
        JNZ     CKE3                                    ;if NO look somemore
        CMP     WORD PTR [BX + 4],5                     ;is next a SF1 name?
        JZ      CKE3                                    ;if YES OK next test
        CMP     WORD PTR [BX + 4],4                     ;is next a SF2 name?
        JZ      CKE3                                    ;if YES OK next test
        JMP     SHORT CKE15                              ;else display error
;------is each Other file name preceded by a self file name ?
CKE4:   INC     CX                                      ;type error number
        MOV     BX,DX                                   ;ptr to CFG table
        SUB     BX,AX                                   ;adjust ptr for an add
CKE5:   ADD     BX,AX                                   ;ptr to next entry
        CMP     WORD PTR [BX],8                         ;is it end of table?
        JZ      CKE6                                    ;if yes next test
        CMP     WORD PTR [BX],4                         ;is this an other name?
        JNC     CKE5                                    ;if NO look somemore
        CMP     WORD PTR [BX - 4],5                     ;is preced SF1 name?
        JZ      CKE5                                    ;if YES OK next test
        CMP     WORD PTR [BX - 4],4                     ;is preced SF2 name?
        JZ      CKE5                                    ;if YES OK next test
        JMP     SHORT CKE15                              ;else display error
;------is each file name follow by an answer string of one or more numbers ?
CKE6:   INC     CX                                      ;type error number
        MOV     BX,DX                                   ;ptr to CFG table
        SUB     BX,AX                                   ;adjust ptr for an add
CKE7:   ADD     BX,AX                                   ;ptr to next entry
        CMP     WORD PTR [BX],8                         ;is it end of table?
        JZ      CKE10                                   ;if yes goto next test
        CMP     WORD PTR [BX],6                         ;is this a file name?
```

```
        JNC     CKE7                                    ;if NO look somemore
        MOV     BP,[BX + 2]                             ;get ptr heap address
        CMP     BYTE PTR [BP],0                         ;is it End of String?
        JZ      CKE15                                   ;YES = number error
;------look for integers > 200 marked as FFh
        INC     CX                                      ;marker for overflow
CKE8:   CMP     BYTE PTR [BP],0                         ;is this EndOFString?
        JZ      CKE9                                    ;if Yes exit loop
        CMP     BYTE PTR [BP],0FFh                      ;is it overflow error
        JZ      CKE15                                    ;if YES goto error
        INC     BP                                      ;ptr to next integer
        JMP     SHORT CKE8                              ;loop until EndOfStr
CKE9:   DEC     CX                                      ;marker for No numbers
        JMP     SHORT CKE7                              ;loop until EndOfTable
;------look for SF1 and SF2 files in the same group.
;       Note: must be last test because ptr to CFG table in DX is overwritten
CKE10:  INC     CX                                      ;advance 2 because last
        INC     CX                                      ;test checked to errers
        MOV     BX,DX                                   ;ptr to CFG table
        SUB     BX,AX                                   ;adjust ptr for an add
CKE11:  ADD     BX,AX                                   ;ptr to next entry
        CMP     WORD PTR [BX],8                         ;is it end of table?
        JZ      CKE16                                   ;if yes exit CFG OK!
        CMP     WORD PTR [BX],7                         ;is this a group name?
        JNZ     CKE11                                   ;if NO look somemore
CKE12:  ADD     BX,8                                    ;ptr to 1st self file
        MOV     DX,0504h                                ;5 = SF1  4 = SF2
        CMP     DL,[BX]                                 ;is file = 4?
        JNZ     CKE13                                   ;if NO then dl=4 file=5
        MOV     DL,DH                                   ;else dl = 5 file =4
;------look until end of table or next group name.
CKE13:  XOR     DH,DH                                   ;make a 16 bit number
CKE14:  ADD     BX,AX                                   ;ptr to next entry
        CMP     WORD PTR [BX],8                         ;is it end of table?
        JZ      CKE16                                   ;if yes exit CFG OK!
        CMP     WORD PTR [BX],7                         ;is this a group name?
        JZ      CKE12                                   ;if YES loop
        CMP     [BX],DX                                 ;is it the wrong file?
        JNZ     CKE14                                   ;if NO loop
CKE15:  MOV     AX,CX                                   ;err type to ax
        CALL    CFG_ERR                                 ;display the error
        STC                                             ;set error flag
CKE16:  POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    CHECK_CFG
;


;
;  Copy a heap integer string or an ascii test string to the FilBuf.
;  It will work on the integer strings because they are byte intergers
;  and 0 is not used as an integer but as an endofstring marker.
;
;       Input = AX = ptr to string in heap
;       Output = length of string in AX
;
PROC    COPY_HEAP_STR
        PUSH    BX
        PUSH    CX
        PUSH    DX
```

```
        PUSH    DS
        PUSH    ES
        MOV     SI,AX                           ;source ptr
        MOV     AX,DS                           ;set register ES
        MOV     ES,AX                           ;to data segment
        MOV     AX,SS                           ;set register DS
        MOV     DS,AX                           ;to the stack segment
        XOR     CX,CX                           ;so hex 0 gets moved
        MOV     DI, Offset FilBuf               ;destination ptr
        CLD                                     ;auto inc DI and SI
CPP1:   CMP     BYTE PTR [SI],0                 ;is it endofstring?
        JZ      CPP2                            ;if yes exit else
        INC     CX                              ;inc line ptr
        MOVSB                                   ;move the byte
        JMP     SHORT CPP1                      ;check the next byte
CPP2:   MOVSB                                   ;move hex 0 also
        MOV     AX,CX                           ;line length to ax
        POP     ES
        POP     DS
        POP     DX
        POP     CX
        POP     BX
        RET
ENDP    COPY_HEAP_STR
;
;
;  display the configuration file data to the screen
;       Input = [CGFTbl]  data table points to heap data strings
;       Output = display heap data to the screen
;
PROC    DISPLAY_CFG
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     BX,Offset CFGTbl                ;ptr to group in table
        MOV     DX,BX                           ;save last group ptr
;note all group names or new pages start here
DSS0:   MOV     AL,[Normal]                     ;get normal color var
        MOV     [Color],AL                      ;set set color to normal
        MOV     CX,0027h                        ;set row/col ptr = 0
        CALL    CLEAR_SCREEN
        PUSH    BX                              ;store current ptr
        CMP     WORD PTR [BX],7                 ;is this a group title?
        JZ      DSS1                            ;if Yes continue else
        MOV     BX,DX                           ;get last group ptr
DSS1:   MOV     AX,[BX + 2]                     ;ptr to group name
        POP     BX                              ;restore current ptr
        CALL    COPY_HEAP_STR                   ;move to [FilBuf]
        SHR     AL,1                            ;divide length by 2
        SUB     CL,AL                           ;center group title
        MOV     AX,CX                           ;row/col ptr
        CALL    GOTOYX                          ;place cursor
        MOV     AX,Offset FilBuf                ;ptr to string
        CALL    DSTR_OUT                        ;show the group title
        CMP     WORD PTR [BX],7                 ;is this a group title?
        JNZ     DSS2                            ;if NO it is Factor
        ADD     BX,4                            ;ptr to Factor title
        XOR     CX,CX                           ;row 0/column 0
;Note all factor names start hear
DSS2:   ADD     CH,2                            ;row = row + 2
        XOR     CL,CL                           ;set column = 0
```

```
        CALL    FACTOR_NAME             ;display factor name
        ADD     BX,4                    ;ptr to file type
        DEC     CH                      ;row=row - 1
;all file types start here
DSS3:   MOV     CL,19h                  ;start in col 25
        INC     CH                      ;row=row+1
        MOV     AX,CX                   ;row/col to ax
        CALL    GOTOYX                  ;move cursor
        MOV     AX,[BX]                 ;get file type
        DEC     AX                      ;convert 0 - 5
        SHL     AX,1                    ;multiply by 2
        MOV     SI,AX                   ;store offset in SI
        MOV     AX,Offset FilTbl        ;ptr to base of table
        ADD     SI,AX                   ;add offset to base
        MOV     AX,[SI]                 ;get ptr from table
        CALL    DSTR_OUT                ;display the type name
        MOV     AX,[BX + 2]             ;ptr to question no
        CALL    COPY_HEAP_STR           ;move to [FilBuf]
        MOV     SI,Offset FilBuf        ;ptr to file buffer
DSS4:   MOV     AL,[SI]                 ;get number
        CMP     AL,0                    ;is it endofstring?
        JZ      DSS5                    ;if YES exit loop
        PUSH    SI                      ;save SI pointer
        CALL    CSTR_OUT                ;space before number
        db      " ",0                   ;this routine does not
        POP     SI                      ;restore SI pointer
        XOR     AH,AH                   ;convert to 16 bits
        CALL    BIN_OUT                 ;display number
        INC     SI                      ;ptr to next byte
        JMP     SHORT DSS4
DSS5:   ADD     BX,4                    ;point to next entry
        MOV     AX,[BX]                 ;get data type in ax
        SUB     AX,6                    ;is it a file type?
        JC      DSS3                    ;if yes do again
        JNZ     DSS5B                   ;<> 0 is group or end
;------will this factor fit on the screen
        PUSH    BX                      ;save current pointer
        MOV     CL,CH                   ;place row count in cl
DSS5A:  ADD     BX,4                    ;ptr to next data type
        INC     CL                      ;row+1 in case we loop
        CMP     WORD PTR [BX],6         ;is it a file type?
        JC      DSS5A                   ;loop if YES
        POP     BX                      ;restore current ptr
        CMP     CL,20                   ;will it fit on page?
        JC      DSS2                    ;if YES display factor
;------page break
DSS5B:  CALL    CFG_MESSAGE             ;stop at page break
        JC      DSS6                    ;exit on escape key
        CMP     AX,0                    ;is it a factor name?
        JZ      DSS5C                   ;if YES goto the top
        MOV     DX,BX                   ;update the group name?
        DEC     AX                      ;is it a group name?
        JNZ     DSS6                    ;if NO then exit
DSS5C:  JMP     DSS0                    ;if YES goto the top
DSS6:   POP     DX                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DISPLAY_CFG
;
        .DATA
```

```
FILTbl   dw      Offset FILS4,Offset FILS3,Offset FILS2
         dw      Offset FILS1, Offset FILS1
FILS1    db      '        Self:',0
FILS2    db      '       Peers:',0
FILS3    db      '   Superiors:',0
FILS4    db      'Subordinates:',0
         .CODE
;
;
;-------Display a Factor Name to the screen.
;        Input = BX = ptr to Factor Name data type
;                CX = row/col
;        Output = Factor name to the screen
;        This routine will split name if larger the 24 characters.
;        It aborts the name if it is longer than 36 characters
;        or a single word with more then 24 letters.
;
PROC     FACTOR_NAME
         PUSH    AX                                      ;save registers
         PUSH    BX
         PUSH    CX
         PUSH    DX
         MOV     AX,[BX + 2]                             ;ptr to factor name
         CALL    COPY_HEAP_STR                           ;move to [FilBuf]
         CMP     AX,37                                   ;is name length > 36?
         JNC     FAN5                                    ;if YES do not display
         CMP     AX,25                                   ;is name length > 24 ?
         JC      FAN4                                    ;if NO display string
;------look for a place to divide the factor name into two lines
         DEC     AX                                      ;lenth = length -1
FAN1:    MOV     BX, Offset FilBuf                       ;ptr to beg if string
         MOV     DX,BX                                   ;save start ptr in dx
         ADD     BX,AX                                   ;ptr to last char in str
         CMP     BYTE PTR [BX],' '                       ;is this a space ?
         JZ      FAN2                                    ;if yes goto next test
         CMP     BYTE PTR [BX],'/'                       ;is it a backslash ?
         JZ      FAN2                                    ;if YES goto next test
         DEC     AX                                      ;lenth = length -1
         JZ      FAN5                                    ;exit if can not divide
         JMP     SHORT FAN1                              ;else look a next char
;------have we shorten it enough ?
FAN2:    CMP     AX,25                                   ;is name length > 24 ?
         JC      FAN3                                    ;if YES split name
         DEC     AX                                      ;lenth = length -1
         JMP     FAN1                                    ;if NO keep looking
;------split the factor name
FAN3:    XOR     AL,AL                                   ;0 = endofstring marker
         MOV     [BX],AL                                 ;mark end of first str
;------display 2nd half of string
         INC     CH                                      ;pointer to next row
         MOV     AX,CX                                   ;row/col ptr
         CALL    GOTOYX                                  ;place cursor
         INC     BX                                      ;ptr to beg. of 2nd half
         MOV     AX,BX                                   ;ptr to string
         CALL    DSTR_OUT                                ;show the group title
         DEC     CH                                      ;restore original row
;------display 1st half of string
FAN4:    MOV     AX,CX                                   ;row/col ptr
         CALL    GOTOYX                                  ;place cursor
         MOV     AX,Offset FilBuf                        ;ptr to string
         CALL    DSTR_OUT                                ;show the group title
FAN5:    POP     DX                                      ;restore registers
```

```
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    FACTOR_NAME
;
;-------Instructions for use of the display configuration.
;       Input = None
;       Output = None
;
PROC    CFG_MESSAGE
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     CL,[Color]                              ;get current color
        MOV     AX,1600h                                ;row 22/column 0
        CALL    GOTOYX                                  ;set cursor position
        MOV     AL,[Menu]                               ;clange color attribute
        MOV     [Color],AL                              ;for scren output.
        CALL    CSTR_OUT                                ;draw menu box
        db      201, 78 DUP (205),187                   ;all except last byte.
        db      186, 78 DUP (' '),186                   ;the screen to scroll
        db      200, 78 DUP (205),0                     ;in row 25, col 80.
        MOV     BL,AL                                   ;color to BL
        MOV     AH,09h                                  ;write char funct no.
        MOV     AL,188                                  ;last character of box
        MOV     BH,0                                    ;page 0 assumed
        MOV     CX,1                                    ;number of bytes
        INT     10h                                     ;write last byte
        MOV     AX,1718h                                ;row 22,column 12
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'Press   Any   Key  to   Continue',0
        MOV     [Color],CL                              ;restore orig color
        CALL    HIDE_CUR
        CALL    GET_CHAR
        CMP     AL,1Bh                                  ;is it an escape key?
        JNZ     CFF2                                    ;if NO exit
        STC                                             ;if YES set carry flag
        JMP     SHORT CFF3                              ;and exit.
CFF2:   CLC                                             ;clear carry flag
CFF3:   POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    CFG_MESSAGE
;
;
;
;-------Ask a yes or no question.
;       Input = None
;       Output = Carry Flag = YES
;
PROC    SHOW_YN
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    CLEAR_MESSAGE
        MOV     AL,[Warning]                            ;warning color
```

```
        MOV     [Color],AL                              ;set color
        MOV     AX,0212h                                ;row 3/Col 12
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      ' View the configuration information ?  Y/[N] ',0
        MOV     AL,[Normal]                              ;normal color
        MOV     [Color],AL                               ;set color
SH1:    CALL    HIDE_CUR
        CALL    GET_CHAR
        AND     AL,5Fh                                   ;turn off bits 6 & 8
        CMP     AL,'N'                                   ;is it No?
        JZ      SH4                                      ;if yes exit
        CMP     AL,0Dh                                   ;is it <Enter>?
        JNZ     SH2                                      ;if not continue
        STC                                              ;set carry flag
        JMP     SHORT SH4                                ;exit
SH2:    CMP     AL,'Y'                                   ;is it Yes?
        JNZ     SH1                                      ;if not get another
        STC                                              ;set carry flag
        JMP     SHORT SH5                                ;exit
SH4:    CLC                                              ;clear carry flag
SH5:    POP     DX                                       ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    SHOW_YN
;
;
;------ Open auxiliary files.  DO NOT DISPLAY ERROR MESSAGE
;
;       Input =  AX = ptr ASCIIZ file type.
;         Assumes [Search] already has a correct path and file name.
;
;       Output = Carry flag set if an opening error AX = error
;                file size in AX and DX; File Handel in BX.
;                File ptr at the beginning of the file.
;       Note:    Registers are not saved.
;
PROC    OPEN_QUIT
        MOV     BX,AX                                   ;save file type offset
        MOV     AX,DS                                   ;assign ES to the
        MOV     ES,AX                                   ;data section
        MOV     AX,Offset Search                        ;ptr to path + file name
        MOV     CX,AX                                   ;save str beginning ptr
        CALL    STR_LENGTH                              ;get length of string
        SUB     AX,4                                    ;length - 4 = "."
        ADD     AX,CX                                   ;ptr to the period
        MOV     DI,AX                                   ;destination ptr
        MOV     SI,BX                                   ;source ptr
        CLD                                             ;auto inc SI & DI
        MOV     CX,5                                    ;number byte to move
        REP     MOVSB                                   ;move type to Search
;------open file and save file handle
         MOV    DX,Offset Search                         ;ptr to path + file
name
         MOV    AH,3Dh                                   ;open file & get handle
         MOV    AL,42h                                   ;share/read/write mode
         INT    21h                                      ;try to open file.
         JC     OA1                                      ;carry = opening error
         MOV    BX,AX                                    ;file handle in BX
         XOR    AX,AX                                    ;zero AX
```

```
          MOV     CX,AX                                   ;off set from EOF
          MOV     DX,AX                                   ;= 0 in CX AND DX.
          MOV     AX,4202h                                ;position at EOF
          INT     21h                                     ;size of file to
          CLC                                             ;AX and DX
OA1:      RET
ENDP      OPEN_QUIT
;
;
;-------Ask a yes or no question.
;         Input = None
;         Output = Carry Flag = YES
;
PROC      DATA_YN
          PUSH    AX                                      ;save registers
          PUSH    BX
          PUSH    CX
          PUSH    DX
          CALL    CLEAR_MESSAGE
          MOV     AL,[Warning]                            ;warning color
          MOV     [Color],AL                              ;set color
          MOV     AX,020Ch                                ;row 3/Col 12
          CALL    GOTOYX                                  ;set cursor
          CALL    CSTR_OUT                                ;display warning
          db      ' Do you want to create a FeedBack data file ?  [Y]/N ',0
          MOV     AL,[Normal]                             ;normal color
          MOV     [Color],AL                              ;set color
DH1:      CALL    HIDE_CUR
          CALL    GET_CHAR
          AND     AL,5Fh                                  ;turn off bits 6 & 8
          CMP     AL,'N'                                  ;is it No?
          JZ      DH4                                     ;if yes exit
          CMP     AL,0Dh                                  ;is it <Enter>?
          JZ      DH2                                     ;if not continue
          CMP     AL,'Y'                                  ;is it Yes?
          JNZ     DH1                                     ;if not get another
DH2:      STC                                             ;set carry flag
          JMP     SHORT DH5                               ;exit
DH4:      CLC                                             ;clear carry flag
DH5:      POP     DX                                      ;restore registers
          POP     CX
          POP     BX
          POP     AX
          RET
ENDP      DATA_YN
;
;
;-------Does the Data file size match the configuration information?
;         Input = AX & DX = size of data file
;         Output = Carry Flag = error if DAT file is not the correct size
;                  Else:
;                  Sets [MaxID]  number of ID numbers in the DAT file
;                  Sets [DNXLn]  expected no of bytes in the index file.
;
PROC      IS_DATA_CORRECT
          PUSH    AX                                      ;save registers
          PUSH    BX
          PUSH    CX
          PUSH    DX
          CMP     AX,0                                    ;is the file empty?
          JZ      ISD3                                    ;if yes display error
;------MaxId = Data File size in bytes / DataRecordLength
```

```
        MOV     CX,[DRecLn]                                     ;get data record ln
        DIV     CX                                              ;File ln/Rec ln
        CMP     DX,0                                            ;is remainder = 0
        JNZ     ISD3                                            ;error in size
        MOV     [MaxID],AX                                      ;number of students
;------Index size in bytes = MaxID * 22                        ;
        MOV     CL,22                                           ;22=IDfield + DRecNo
        MUL     CL                                              ;times MaxID =
        MOV     [NDXLn],AX                                      ;size of index file
        JMP     SHORT ISD4                                      ;exit no error
ISD3:   MOV     AL,[Warning]                                    ;warning color
        MOV     [Color],AL                                     ;set color
        MOV     AX,0107h                                       ;row 3/Col 12
        CALL    GOTOYX                                         ;set cursor
        CALL    CSTR_OUT                                       ;display warning
        db      ' The data file is not compatible with the'
        db      ' configuration file. ',0
        MOV     AL,[Normal]                                     ;normal color
        MOV     [Color],AL                                      ;set color
        STC                                                     ;set carry flag
        JMP     SHORT ISD5                                      ;exit
ISD4:   CLC                                                     ;clear carry flag
ISD5:   POP     DX                                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    IS_DATA_CORRECT
;
;
;       Open DAT file and check to see it is the correct size
;        input = none
;        output = carry flag if DAT file is not correct size
;           CALLS  IS_DATA_CORRECT that sets the following variables
;                [MaxID]  number of ID numbers in the DAT file
;                [DNXLn]  expected no of bytes in the index file.
;
PROC    OPEN_DATA_FILE
        PUSH    AX                                             ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AL,[Normal]                                    ;get normal color var
        MOV     [Color],AL                                     ;set set color to normal
        XOR     AX,AX                                          ;row 0 and column 0
        CALL    MENU_BOX                                       ;draw top menu box
        MOV     AX,Offset DATTyp
        CALL    OPEN_QUIT
        JC      OPD3
        MOV     [DATHd],BX                                      ;save file handle
        CALL    IS_DATA_CORRECT                                 ;data size = config?
        JC      OPD3                                            ;exit on size error
;------Is a rank file the correct size ?
        MOV     AX,Offset RNKTyp                                ;open rank file if
        CALL    OPEN_QUIT                                       ;it is present?
        JC      OPD1                                            ;If not found next test
        MOV     [RNKHd],BX                                      ;save rank handle
        CMP     AX,[RNKLn]                                      ;is size correct?
        JZ      OPD1                                            ;if yes look for index
        CALL    DELETE_RANK_FILE
;------Is the index file the correct size ?
OPD1:   MOV     AX,Offset NDXTyp
```

```
        CALL    OPEN_QUIT                               ;look for index file
        JC      OPD4                                    ;exit if not found
        MOV     [NDXHd],BX                              ;save file handle
        MOV     BX,AX                                   ;save file size
        MOV     AX,[MaxID]                              ;get number of records
        INC     AX                                      ;correct size =
        INC     AX                                      ;32(IDs + 2) bytes
        MOV     CX,32                                   ;index record length
        MUL     CX                                      ;AX = index file size
        CMP     AX,BX                                   ;is it correct size?
        JZ      OPD4                                    ;if yes exit OK!
        CALL    DELETE_NDX_FILE
        JMP     SHORT OPD4                              ;also close rand file
OPD3:   MOV     AX,[DATHd]                              ;get data file handle
        CALL    CLOSE                                   ;close bad data file
        JC      OPD3A                                   ;exit on DOS error
        XOR     AX,AX                                   ;zero to register
        MOV     [DATHd],AX                              ;mark file closed
OPD3A:  CALL    DATA_YN                                 ;create the file ?
        JNC     OPD5                                    ;if NO then exit
        CALL    MAKE_DATA_FILE                          ;build a data file
        JC      OPD5                                    ;exit on error
OPD4:   CLC                                             ;clear carry = data OK!
OPD5:   POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    OPEN_DATA_FILE
;
      .CODE
;
;------ Close an open file
;       Input =  AX = File Handle
;       Output = None  (message displayed and carry flag set on error)
;       Note:    Major registers saved.
;
PROC    CLOSE
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     BX,AX                                   ;file handle
        CMP     BX,0                                    ;Is the file open?
        JZ      CLO3                                    ;exit if file closed.
        MOV     AH,3Eh                                  ;close file function no
        INT     21h                                     ;close REST.FIL
        JNC     CLO3                                    ;exit if successful.
CLO1:   CALL    CLEAR_MESSAGE
        MOV     CL,[Color]                              ;save current color
        MOV     AL,[Warning]                            ;warning color
        MOV     [Color],AL                              ;set color
        MOV     AX,0207h                                ;row 3/Col 12
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      'Error closing file.  Press Any Key to Continue. ',0
        MOV     [Color],CL                               ;restore color
        CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR
        STC                                             ;set carry flag for ret
CLO3:   POP     DX                                      ;restore registers
```

```
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    CLOSE
;
;
;------ Close the data files used by the FeedBack program
;       Input =  None
;       Output = None  (message displayed and carry flag set on error)
;       Note:    Major registers saved.
;
PROC    CLOSE_ALL_FILES
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
;-----clear variables
        CALL    CLEAR_PERCNT                    ;set percentiles = 0
        XOR     AX,AX                           ;zero to AX
        MOV     CX,6                            ;loop counter
        MOV     SI, Offset EOF                  ;pointer to file vars.
CLA0:   MOV     [SI], AX                        ;set all file variables
        INC     SI                              ;except [FileDr] to 0
        INC     SI
        LOOP    CLA0
        MOV     CX,9                            ;loop counter
        MOV     SI,Offset CFGHd                 ;ptr to 1st file hd
CLA1:   MOV     AX,[SI]                         ;get file handle
        CALL    CLOSE                           ;close the file
        JC      CLA2                            ;on error leave handle
        XOR     AX,AX                           ;0 = file is closed
        MOV     [SI],AX                         ;mark the file closed
CLA2:   INC     SI                              ;ptr to next file hdl
        INC     SI
        LOOP    CLA1                            ;loop thru all files
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    CLOSE_ALL_FILES
;
;
;-------Check to make sure a feedback file is in the directory.
;       Input = None
;       Output = Carry Flag if no file is open.
;
PROC    IS_SLD
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        XOR     AX,AX                           ;zero to AX register
        CMP     [MaxFile],AL                    ;were data files found?
        JZ      NCF1                             ;0 means NO files
        JMP     NCF2                             ;exit if found
NCF1:   MOV     CL,[Color]
        MOV     AL,[Warning]                    ;warning color
        MOV     [Color],AL                      ;set color
        MOV     AX,020Bh                        ;row 3/Col 12
        CALL    GOTOYX                          ;set cursor
```

```
        CALL    CSTR_OUT                                        ;display warning
        db      ' No *.CFG files found in the directory!  Press Any Key '
        db      'to Continue. ',0
        MOV     [Color],CL                                      ;restore original color
        CALL    HIDE_CUR
        CALL    GET_CHAR
        STC                                                     ;set carry flag
NCF2:   POP     DX                                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    IS_SLD

;
;-------Inform the user the file is being opened.
;       Input = None
;       Output = None
;
PROC    READ_MESS
        PUSH    AX
        PUSH    CX
        MOV     CL,[Color]                                      ;save orig. color attr
        MOV     AL,[Warning]                                    ;warning color
        MOV     [Color],AL                                      ;set color
        MOV     AX,0209h                                        ;row 3/Col 12
        CALL    GOTOYX                                          ;set cursor
        CALL    CSTR_OUT                                        ;display warning
        db      '   Reading File   ',0
        MOV     [Color],CL                                      ;restore orig. color att
        CALL    HIDE_CUR
        POP     CX
        POP     AX
        RET
ENDP    READ_MESS
;
;
;-------Clear the second line of the meu box
;       Input = None
;       Output = None
;
PROC    CLEAR_MESS
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     CL,[Color]                                      ;save orig. color attr
        MOV     AL,[Menu]                                       ;set menu color
        MOV     [Color],AL                                      ;change color attribute
        MOV     AX,0207h                                        ;row 2 and column 7
        MOV     BX,024Eh                                        ;row 2 and column 78
        CALL    CLEAR_WINDOW                                    ;clear out old message.
        MOV     AX,020Fh                                        ;row 2,column 7
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'Press the <Enter> key to open the highlighted file.',0
        MOV     [Color],CL                                      ;restore orig.color attr
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
```

```
ENDP    CLEAR_MESS
;
;
;------Read a line from a file into the 256 byte 'FilBuf' buffer.
;       Input =  valid file handle must be in AX
;       Output = sets [EOF] <> 0 when EndOfFile is reached.
;                Carry flag = file closes or file ptr already at EndOfFile.
;       NOTES:
;       LineFeeds (0Ah) and WP page break (D4h) are convert to hex 0.
;       Only the lower set ASCII characters are placed in the buffer.
;       No control codes etc.
;       Only the first 256 bytes of the line are saved in the buffer but the
;       procedure will keep reading until EndOfFile or an 0Ah or D4h is reached.
;       0Ah is a line feed and WP EndOfLine marker. DOS text files end each line
;       with 0Dh a carrage return followed by a line feed 0Ah.
;       (D4h is a WP page break marker)
;
PROC    READ_LINE
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    SI
        PUSH    BP
        CMP     AX,0                            ;is the file open?
        JZ      REE2                            ;if not Exit
        MOV     BX,AX                           ;store file handle
        XOR     AX,AX
        MOV     SI,Offset FilBuf                ;mark position in buffer
        MOV     BP,Offset FilBuf + 255          ;mark end of buffer
        CMP     [EOF],AL                        ;is ptr at endoffile
        JNZ     REE2                            ;if yes Exit
;------read 1  byte from data file
      MOV   CX,1                        ;read 1 byte
REE1:   MOV     AX,3F00h                        ;read file function no
        MOV     DX,SI                           ;buffer ptr to DX
        INT     21h                             ;get byte
        JC      REE3                            ;end of file?
      CMP   AX,CX                       ;did it read a byte?
        JNZ     REE3                            ;if no then EndOfFile
        MOV     AL,[SI]                         ;get char in AL
        CMP     AL,0Ah                          ;is it the endofline ?
        JZ      REE4                            ;if YES exit
        CMP     AL,0D4h                         ;is it WP page break ?
        JZ      REE4                            ;if YES exit
        CMP     AL,128                          ;is 8th bit on?
        JNC     REE1                            ;if yes read next char?
      CMP   AL,32                       ;is it a control char?
        JC      REE1                            ;if yes read next char
        CMP     BP,SI                           ;is buffer full ?
        JC      REE1                            ;if yes read until 0Dh
        INC     SI                              ;if no advance buffer
        JMP     SHORT REE1                      ;ptr & get another char
REE2:   STC                                     ;set carry flag
        JMP     SHORT REE5                      ;exit finished file.
REE3:   MOV     AL,0FFh                         ;non zero = end of file
        MOV     [EOF],AL                        ;mark endoffile true
REE4:   XOR     AL,AL                           ;place endofline
        MOV     [SI],AL                         ;in data file buffer
        CLC                                     ;clear carry flag
REE5:   POP     BP
        POP     SI
```

```
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    READ_LINE
;
;------Read a line from a file into the 256 byte 'SF1Buf' buffer.
;       Input = none
;       Output = sets [SF1EOF]<> 0 when EndOfFile is reached.
;                Carry flag = file closes or file ptr already at EndOfFile.
;       NOTES:
;       LineFeeds (0Ah) and WP page break (D4h) are convert to hex 0.
;       Only the lower set ASCII characters are placed in the buffer.
;       No control codes etc.
;       Only the first 256 bytes of the line are saved in the buffer but the
;       procedure will keep reading until EndOfFile or an 0Ah or D4h is reached.
;
PROC    READ_SF1
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    SI
        PUSH    BP
        MOV     AX,[SF1Hd]
        CMP     AX,0                            ;is the file open?
        JZ      RSF2                            ;if not Exit
        MOV     BX,AX                           ;store file handle
        XOR     AX,AX
        MOV     SI,Offset SF1Buf                ;mark position in buffer
        MOV     BP,Offset SF1Buf + 255          ;mark end of buffer
        CMP     [SF1EOF],AL                     ;is ptr at endoffile
        JNZ     RSF2                            ;if yes Exit
;------read 1  byte from data file
      MOV   CX,1                        ;read 1 byte
RSF1:   MOV     AX,3F00h                        ;read file function no
        MOV     DX,SI                           ;buffer ptr to DX
        INT     21h                             ;get byte
        JC      RSF3                            ;end of file?
      CMP   AX,CX                       ;did it read a byte?
        JNZ     RSF3                            ;if no then EndOfFile
        MOV     AL,[SI]                         ;get char in AL
        CMP     AL,0Ah                          ;is it the endofline ?
        JZ      RSF4                            ;if YES exit
        CMP     AL,0D4h                         ;is it WP page break ?
        JZ      RSF4                            ;if YES exit
        CMP     AL,128                          ;is 8th bit on?
        JNC     RSF1                            ;if yes read next char?
      CMP   AL,32                       ;is it a control char?
        JC      RSF1                            ;if yes read next char
        CMP     BP,SI                           ;is buffer full ?
        JC      RSF1                            ;if yes read until 0Dh
        INC     SI                              ;if no advance buffer
        JMP     SHORT RSF1                      ;ptr & get another char
RSF2:   STC                                     ;set carry flag
        JMP     SHORT RSF5                      ;exit finished file.
RSF3:   MOV     AL,0FFh                         ;non zero = end of file
        MOV     [SF1EOF],AL                     ;mark endoffile true
RSF4:   XOR     AL,AL                           ;place endofline
        MOV     [SI],AL                         ;in data file buffer
        CLC                                     ;clear carry flag
```

```
RSF5:   POP     BP
        POP     SI
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    READ_SF1
;
;
;Place the file pointer at the begining of the open file.
;       Input = a valid file handle in AX
;       Output = Carry flag = error
PROC    GOTO_TOP
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    SI
        PUSH    DI
        PUSH    BP
        CMP     AX,0                                    ;is file open?
        JNZ     TOP1                                    ;if open goto next test
        STC                                             ;else set error flag
        JMP     SHORT TOP2                              ;exit on error
;------place file pointer to the beginning of the file
TOP1:   MOV     BX,AX                                   ;store file handle
        XOR     AX,AX                                   ;zero register
        MOV     CX,AX                                   ;set offset = 0
        MOV     DX,AX                                   ;set offset = 0
        MOV     AX,4200h                                ;set file pointer no.
        INT     21h                                     ;set to beg. of file
        JC      TOP2                                    ;exit if error.
        XOR     AL,AL                                   ;zerp to register
        MOV     [EOF],AL                                ;set EndOfFile = False
        CLC
TOP2:   POP     BP
        POP     DI
        POP     SI
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    GOTO_TOP
;

;
;Checks [FileDr] to make sure there is room for data
;       Input = none
;               assumes [FileDr] is pointing the desired drive
;                       0 = default,  1 = A,  2 = B, etc
;       Output = Carry flag = if not enough room
PROC    IS_FULL
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    BP
        MOV     DL,[FileDr]                             ;get file drive no.
        MOV     AX,3600h                                ;disk space function
        INT     21h                                     ;get disk space
```

```
        CMP     AX,0FFFFh                               ;is drive valid?
        JZ      ISF3                                    ;if NO exit error
        CMP     BX,2                                    ;avail cluster > 2
        JNC     ISF4                                    ;yes OK! lots of room
        CALL    FULL_ERR                                ;else inform user
ISF3:   STC
        JMP     SHORT ISF5
ISF4:   CLC
ISF5:   POP     BP
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
;
;       Input = none
;       Output = none
PROC    FULL_ERR
        CALL    CLEAR_MESSAGE
        MOV     AL,[Warning]                            ;warning color
        MOV     CL,[Color]                              ;save original color
        MOV     [Color],AL                              ;set color
        MOV     AX,0209h                                ;row 3/Col 8
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      ' Not enough Disk Space! '
        db      '  Press Any Key to Continue. ',0
        MOV     [Color],CL                               ;restore original color
        CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR
        RET
ENDP    FULL_ERR
ENDP    IS_FULL
;
;       Input = none
;       Output = carry flag = DOS error
;                File handle is in [DATHd]
;                File contains zero bytes.
;    WARNING: this proceedure will erase an existing file with the same name.
;
PROC    OPEN_NEW_DATA_FILE
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        XOR     AX,AX                                   ;zero to ax register
        MOV     [DATHd],AX                              ;set file handle to 0
        MOV     AX,DS                                   ;assign ES to the
        MOV     ES,AX                                   ;data section
        MOV     AX,Offset Search                        ;ptr to path + file name
        MOV     CX,AX                                   ;save str beginning ptr
        CALL    STR_LENGTH                              ;get length of string
        SUB     AX,4                                    ;length - 4 = "."
        ADD     AX,CX                                   ;ptr to the period
        MOV     DI,AX                                   ;destination ptr
        MOV     SI,Offset DATTyp                        ;ptr to file type name
        CLD                                             ;auto inc SI & DI
        MOV     CX,5                                    ;number byte to move
        REP     MOVSB                                   ;move type to Search
        MOV     AX, Offset Search                       ;ptr to name of file
        CALL    CREATE                                  ;open an empty file
```

```
        JC      CDF1                                    ;exit on DOS error
        MOV     [DATHd],BX                              ;save file handle
        CLC                                             ;clear carry = data OK!
CDF1:   POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    OPEN_NEW_DATA_FILE
;
;       Input = none
;       Output = carry flag = DOS error
;    WARNING: this proceedure will erase an existing file.
;
PROC    DELETE_DATA_FILE
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,DS                                   ;assign ES to the
        MOV     ES,AX                                   ;data section
        MOV     AX,Offset Search                        ;ptr to path + file name
        MOV     CX,AX                                   ;save str beginning ptr
        CALL    STR_LENGTH                              ;get length of string
        SUB     AX,4                                    ;length - 4 = "."
        ADD     AX,CX                                   ;ptr to the period
        MOV     DI,AX                                   ;destination ptr
        MOV     SI,Offset DATTyp                        ;ptr to file type name
        CLD                                             ;auto inc SI & DI
        MOV     CX,5                                    ;number byte to move
        REP     MOVSB                                   ;move type to Search
        MOV     AX, Offset Search                       ;ptr to name of file
        CALL    DELETE_FILE                             ;erase file
        JC      DDF1                                    ;exit on DOS error
        XOR     AX,AX                                   ;zero to ax
        MOV     [DATHd],AX                              ;zero file handle
        MOV     [MaxID],AX                              ;zero no of ID's
        CLC                                             ;clear carry = data OK!
DDF1:   POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DELETE_DATA_FILE
;
;
;    Does the config information call for this answer file.
;       input = AX = file type number
;            5=POS1 4=POS2 3=PER 2=SUP and 1=SUB
;       output = cf = NOT NEEDED
;
PROC    IS_FILE_NEEDED
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     CX,AX                                   ;save type in cx
        MOV     BX,Offset CFGTbl                        ;ptr to beg of table
        MOV     AX,4                                    ;offset to next entry
ISN1:   CMP     WORD PTR [BX],8                         ;is it end of table?
        JZ      ISN2                                    ;if YES exit not needed
        CMP     WORD PTR [BX],CX                        ;is the file needed?
```

```
        JZ      ISN3                                    ;if YES exit needed
        ADD     BX,AX                                   ;point to next entry
        JMP     SHORT ISN1
ISN2:   STC                                             ;carry flag = NO
ISN3:   POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    IS_FILE_NEEDED
;
;
;------ Open files.  DISPLAY ERROR MESSAGE
;
;       Input =  AX = ptr ASCIIZ file type.
;         Assumes [Search] already has a correct path and file name.
;
;       Output = Carry flag set if an opening error AX = error
;                file size in AX and DX; File Handel in BX.
;                File ptr at the beginning of the file.
;       Note:    Registers are not saved.
;
PROC    OPEN_FILE
        MOV     BX,AX                                   ;save file type offset
        MOV     AX,DS                                   ;assign ES to the
        MOV     ES,AX                                   ;data section
        MOV     AX,Offset Search                        ;ptr to path + file name
        MOV     CX,AX                                   ;save str beginning ptr
        CALL    STR_LENGTH                              ;get length of string
        SUB     AX,4                                    ;length - 4 = "."
        ADD     AX,CX                                   ;ptr to the period
        MOV     DI,AX                                   ;destination ptr
        MOV     SI,BX                                   ;source ptr
        CLD                                             ;auto inc SI & DI
        MOV     CX,5                                    ;number byte to move
        REP     MOVSB                                   ;move type to Search
        MOV     AX, Offset Search
        CALL    OPEN
        RET
ENDP    OPEN_FILE
;
; File type numbers:  5=POS1 4=POS2 3=PER 2=SUP and 1=SUB
;
PROC    OPEN_ANSWER_FILES
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        XOR     AX,AX                                   ;row 0 and column 0
        CALL    MENU_BOX                                ;draw top menu box
;------the SF1 file must exist for the program to build a new data file
        MOV     AX,Offset SF1Typ                        ;open SF1 anser file
        CALL    OPEN_FILE
        JC      OAF5                                    ;if error exit
        MOV     [SF1Hd],BX                              ;save file handle
        XOR     AL,AL                                   ;zero to al and
        MOV     [EOF],AL                                ;set EndOfFile false
;------open SF2 if needed
        MOV     AX,4                                    ;look for SF2 type
        CALL    IS_FILE_NEEDED                          ;is SF2 needed ?
        JC      OAF1                                    ;if NO check next file
        MOV     AX,Offset SF2Typ                        ;open SF1 anser file
```

```
        CALL    OPEN_FILE
        JC      OAF5                            ;if error exit
        MOV     [SF2Hd],BX                      ;save file handle
;------open PER if needed
OAF1:   MOV     AX,3                            ;look for PER type
        CALL    IS_FILE_NEEDED                  ;is PER needed ?
        JC      OAF2                            ;if NO check next file
        MOV     AX,Offset PERTyp                ;open PER anser file
        CALL    OPEN_FILE
        JC      OAF2                            ;if not avail. no error
        MOV     [PERHd],BX                      ;save file handle
;------open SUP if needed
OAF2:   MOV     AX,2                            ;look for SUP type
        CALL    IS_FILE_NEEDED                  ;is SUP needed ?
        JC      OAF3                            ;if NO check next file
        MOV     AX,Offset SUPTyp                ;open SUP answer file
        CALL    OPEN_FILE
        JC      OAF3                            ;if not avail. no error
        MOV     [SUPHd],BX                      ;save file handle
;------open SUB if needed
OAF3:   MOV     AX,1                            ;look for SUB type
        CALL    IS_FILE_NEEDED                  ;is SUB needed ?
        JC      OAF4                            ;if NO exit no error
        MOV     AX,Offset SUBTyp                ;open SUB answer file
        CALL    OPEN_FILE
        JC      OAF4                            ;if not avail. no error
        MOV     [SUBHd],BX                      ;save file handle
OAF4:   CLC                                     ;clear carry = data OK!
OAF5:   POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    OPEN_ANSWER_FILES
;
;
;Add up the answers for a given factor for the ID stored in the BX buffer
;
;       Input = SI = ptr into Config table    DI = ptr into OutBuf
;               BP = ptr into question no string in heap
;               BX = ptr into answer buffer: SF1, SF2 ,PER, SUP, or SUB.
;                NOTE: PerBuf, SupBuf and SubBuf hold up to 5 records
;               CX = stores answer total in CH and number of answers in CL
;       Output = AH = total of answers
;                AL = no. of questions answered.
;                If a 50% were not answered then AX = 0
PROC    GET_ANSWERS
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    SI
        PUSH    DI
        PUSH    BP
;------is this a matching ID to SF1Buffer ?
        MOV     CX,20                           ;loop counter
        MOV     DI,Offset SF1Buf                ;ptr to SF1
        MOV     SI,BX                           ;ptr to data file line
        CLD                                     ;auto inc DI and SI
        REPZ    CMPSB                           ;are the bytes = ?
        JZ      GAN2                            ;if YES continue
        XOR     AX,AX                           ;return 0,0
        STC                                     ;else EXIT no more
```

```
        JMP     GAN7                                    ;ID's were found.
;------get the answers from one answer sheet for one data point
GAN2:   XOR     CX,CX                                   ;zero new totals
        XOR     DX,DX                                   ;counter: no. of ques.
        ADD     BX,20                                   ;skip ID string
GAN3:   CMP     BYTE PTR [BP],0                         ;is it end of string ?
        JZ      GAN5                                    ;normal exit of loop
        MOV     AL,[BP]                                 ;get question number
        DEC     AL                                      ;question no - 1
        XOR     AH,AH                                   ;convert to 16 bits
        PUSH    BX                                      ;save starting position
        ADD     BX,AX                                   ;ptr to answer in buf
        MOV     AH,[BX]                                 ;get answer
        POP     BX                                      ;restore starting posit.
        AND     AH,0CFh                                 ;ascii to hex number
        JZ      GAN4                                    ;if zero skip to small
        CMP     AH,6                                    ;is answer < 6 ?
        JNC     GAN4                                    ;if NO skip to large
        INC     CL                                      ;no of answers counter
        ADD     CH,AH                                   ;add answer to total
GAN4:   INC     BP                                      ;ptr next config answer
        INC     DX                                      ;no of questions counter
        JMP     SHORT GAN3                              ;get the next answer
;------were 50% of the question answered?
GAN5:   XOR     AX,AX                                   ;zero ax register
        INC     DX                                      ;for rounding
        SHR     DL,1                                    ;no of ques./2
        CMP     CL,DL                                   ;did answ. 50% ?
        JC      GAN6                                    ;if NO don't use answers
        MOV     AX,CX                                   ;return totals
GAN6:   CLC                                             ;clear carry flag
GAN7:   POP     BP
        POP     DI
        POP     SI
        POP     DX
        POP     CX
        POP     BX
        RET
ENDP    GET_ANSWERS
;
;       Compute the means score and code the output value
;
;       Input   CX = stores answer total in CH and number of answers in CL
;       Output = AL = coded data point
;                 example: 15h =  decimal 21 which means a score of 2.1
PROC    CODE_MEAN
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,CX                                   ;counts to AX register
        MOV     BX,CX                                   ;save a copy in BX
        CMP     CL,0                                    ;were answers found?
        JZ      COM2                                    ;if NO answer = 0
        MOV     CH,0Ah                                  ;divsor & multiplier
        MOV     AL,BH                                   ;total to ax
        XOR     AH,AH                                   ;convert to 16 bits
        DIV     CL                                      ;total/no. of items
        MOV     DL,AH                                   ;save remander in ch
        MUL     CH                                      ;ax = mean * 10
        XCHG    AX,DX                                   ;remainder to al
        MUL     CH                                      ;ax = remainder * 10
        DIV     CL                                      ;al = tenths
```

```
        SHR     CL,1                    ;divsor/2 + 1
        INC     CL                      ;is used for rounding
        CMP     AH,CL                   ;is remainder <.05 ?
        JC      COM1                    ;if YES do not round
        INC     AL                      ;round answer remainder
COM1:   ADD     AL,DL                   ;add units to tenths
        XOR     AH,AH                   ;convert to 16 bits
COM2:   CLC                             ;clear carry flag
        POP     DX
        POP     CX
        POP     BX
        RET
ENDP    CODE_MEAN
;
;
;
; Find an Id number that matches the ID in SF1Buf
;       input = AX = file handle
;       output  Carry flag = NO FIND
;
PROC    FIND_ID
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     BX,AX                   ;save file handle
FII1:   MOV     AX,BX                   ;restore file handle
        CALL    READ_LINE               ;1 line from data file
        JNC     FII2                    ;not EndOfFile
        MOV     AL,0FFh                 ;mark EndOfFile true
        MOV     [EOF],AL                ;<> 0 = True
;------has the ESC key been pressed ?
FII2:   CALL    IS_ESC                  ;is ESC key presses
        JC      FII3                    ;cf means abort process
;------is it the correct ID?
        MOV     CX,20                   ;loop counter
        MOV     DI,Offset SF1Buf        ;ptr to SF1
        MOV     SI,Offset FilBuf        ;ptr to data file line
        CLD                             ;auto inc DI and SI
        REPZ    CMPSB                   ;are the bytes = ?
        CLC                             ;clear carry flag
        JZ      FII3                    ;if YES exit FOUND
        CMP     BYTE PTR [EOF],0        ;is it EndOfFile ?
        JZ      FII1                    ;if NO read next line
        STC                             ;exit NOT FOUND
FII3:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    FIND_ID
;
;
;
;   Copy ASCIIZ string from FilBuf to another location in the data segment
;       Input:  AX = destination offset
;       Output: None
;
PROC    COPY_STRING
        PUSH    AX
        PUSH    BX
        PUSH    CX
```

```
        PUSH    DX
        MOV     DI,AX                           ;di = destination
        MOV     AX,DS                           ;set segments ES = DS
        MOV     ES,AX
        MOV     SI,Offset FilBuf                ;si = source
CPY1:   MOV     AL,[SI]                         ;get byte
        CMP     AL,0                            ;is it endof string
        JZ      CPY2                            ;is yes exit
        MOV     [DI],AL
        INC     SI
        INC     DI
        JMP     CPY1
CPY2:   CLC                                     ;clear carry flag
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    COPY_STRING
;
;   Check each answer file for all ID's matching the current ID in SF1Buf
;       Input:  If a file is needed it will be open thus its handle <> 0
;       Output: None
;
PROC    LOCATE_ALL_IDS
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,[SF2Hd]                      ;get self II handle
        CMP     AX,0                            ;is it open?
        JZ      LAI0                            ;if NO goto next file
        MOV     BYTE PTR [EOF],0                ;set EndOfFile false
        CALL    GOTO_TOP                        ;file ptr to TopOfFile
        CALL    FIND_ID                         ;search for the ID
        JC      LAI0                            ;if not found next file
        MOV     AX,Offset SF2Buf                ;ptr to buffer
        CALL    COPY_STRING                     ;copy to self 2 buffer
LAI0:   CMP     WORD PTR[DatHd],0               ;abort the proccess?
        JNZ     LAI1                            ;if NO continue
        JMP     LAI7                            ;else exit abort
LAI1:   MOV     AX,[PERHd]                      ;get peer handle
        CMP     AX,0                            ;is it open?
        JZ      LAI3                            ;if NO goto next file
        MOV     BX,AX                           ;save handle in bx
        MOV     DX,Offset PERBuf                ;ptr to buffer
        MOV     CX,8                            ;loop counter
        MOV     BYTE PTR [EOF],0                ;set EndOfFile false
        CALL    GOTO_TOP                        ;file ptr to TopOfFile
LAI2:   MOV     AX,BX                           ;get peer handle
        CALL    FIND_ID                         ;search for the ID
        JC      LAI3                            ;if not found next file
        MOV     AX,DX                           ;ptr into peer buffer
      CALL  COPY_STRING                 ;copy to per buffer
      INC   DH                          ;ptr+256=next peer buf
        LOOP    LAI2                            ;stop loop after 5 fines
LAI3:   CMP     WORD PTR[DatHd],0               ;abort the proccess?
        JZ      LAI7                            ;Exit if abort
        MOV     AX,[SUPHd]                      ;get superior handle
        CMP     AX,0                            ;is it open?
      JZ    LAI5                        ;if NO goto next file
        MOV     BX,AX                           ;save handle in bx
```

```
        MOV     DX,Offset SUPBuf                        ;ptr to buffer
        MOV     CX,8                                    ;loop counter
        MOV     BYTE PTR [EOF],0                        ;set EndOfFile false
        CALL    GOTO_TOP                                ;file ptr to TopOfFile
LAI4:   MOV     AX,BX                                   ;get superior handle
        CALL    FIND_ID                                 ;search for the ID
      JC    LAI5                            ;if not found next file
        MOV     AX,DX                                   ;ptr into superior buf
        CALL    COPY_STRING                             ;copy to sup buffer
        INC     DH                                      ;ptr+256=next sup buf
        LOOP    LAI4                                    ;stop loop after 5 fines
LAI5:   CMP     WORD PTR[DatHd],0                       ;abort the proccess?
        JZ      LAI7                                    ;Exit if abort
        MOV     AX,[SUBHd]                              ;get subordinate handle
        CMP     AX,0                                    ;is it open?
        JZ      LAI7                                    ;if NO goto exit
        MOV     BX,AX                                   ;save handle in bx
        MOV     DX,Offset SUBBuf                        ;ptr to buffer
        MOV     CX,8                                    ;loop counter
        MOV     BYTE PTR [EOF],0                        ;set EndOfFile false
        CALL    GOTO_TOP                                ;file ptr to TopOfFile
LAI6:   MOV     AX,BX                                   ;get subordinate handle
        CALL    FIND_ID                                 ;search for the ID
        JC      LAI7                                    ;if not found exit
        MOV     AX,DX                                   ;ptr into superior buf
        CALL    COPY_STRING                             ;copy to sup buffer
        INC     DH                                      ;ptr+256=next sub buf
        LOOP    LAI6                                    ;stop loop after 5 fines
LAI7:   CMP     WORD PTR[DatHd],0                       ;abort the proccess?
        JNZ     LAI8                                    ;exit clear carry flag
        STC                                             ;else exit abort
        JMP     SHORT LAI9
LAI8:   CLC
LAI9:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    LOCATE_ALL_IDS
;
;
;    Do you want to continue ?
;       Input = none
;       Output = carry flag = NO
PROC    WARNING_MESS
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    CLEAR_MESSAGE
        MOV     CL,[Color]                              ;store original Color
        MOV     AL,[Warning]                            ;warning color
        MOV     [Color],AL                              ;set color
        MOV     AX,0109h                                ;row 3/Col 12
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      " WARNING:  Creating a new data file could take "
        db      "several hours. ",0
        MOV     AX,0214h                                ;row 3/Col 12
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      " Do you want to begin the process ? "
```

```
        db      " Y/N ",0
        MOV     [Color],CL                      ;restore original color
        CALL    HIDE_CUR
WAR1:   CALL    ERR_SOUND
        CALL    GET_CHAR
        AND     AL,5Fh                          ;turn off bit 6 & 8
        CMP     AL,'N'                          ;is it No?
        JNZ     WAR2                            ;if NO goto next test
        STC                                     ;else set carry flag
        JMP     SHORT WAR3                      ;and exit
WAR2:   CMP     AL,'Y'                          ;is it Yes?
        JNZ     WAR1                            ;if not YES loop
        CALL    CLEAR_MESSAGE                   ;empty message line
        CLC                                     ;clear cf = continue
WAR3:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    WARNING_MESS
;
;    Do you want to abort the data file ?
;       Input = none
;       Output = carry flag = NO
PROC    ABORT_MESS
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    SI
        PUSH    DI
        PUSH    BP
        CALL    CLEAR_MESSAGE
        MOV     CL,[Color]                      ;store original Color
        MOV     AL,[Warning]                    ;warning color
        MOV     [Color],AL                      ;set color
        MOV     AX,020Ah                        ;row 3/Col 12
        CALL    GOTOYX                          ;set cursor
        CALL    CSTR_OUT                        ;display warning
        db      " Do you want to cancel the create data file command ? "
        db      " Y/N ",0
        MOV     [Color],CL                      ;restore original color
        CALL    HIDE_CUR
ABO1:   CALL    ERR_SOUND
        CALL    GET_CHAR
        AND     AL,5Fh                          ;turn off bit 6 & 8
        CMP     AL,'Y'                          ;is it Yes?
        JNZ     ABO2                            ;if not Yes goto next
        CALL    DELETE_DATA_FILE                ;erase the data file
        STC                                     ;else set clear carry
        JMP     SHORT ABO3                      ;flag and exit
ABO2:   CMP     AL,'N'                          ;is it NO?
        JNZ     ABO1                            ;if not NO loop
        CALL    CLEAR_MESSAGE                   ;empty message line
        CLC                                     ;clear cf = continue
ABO3:   POP     BP
        POP     DI
        POP     SI
        POP     DX
        POP     CX
        POP     BX
        POP     AX
```

```
        RET
ENDP    ABORT_MESS
;
;
;-------Instructions for the Create data command.
;       Input = None
;       Output = None
;
PROC    DATA_INSTRU
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,1500h                                ;row 21,column 0
        CALL    MENU_BOX                                ;draw menu box
        MOV     CL,[Color]                              ;get assigned color
        MOV     AL,[Normal]                             ;get color
        MOV     [Color],AL                              ;set color
        MOV     AX,0A0Bh                                ;row/column
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'Searching the answer files for ID: ',0
        MOV     AX,0C0Bh                                ;row/column
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      "The number of Self ID's already processed = ",0
        MOV     AL,[Menu]                               ;get menu color
        MOV     [Color],AL                              ;set menu color
        MOV     AX,160Bh                                ;row/column
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'Press the <Esc> key to cancel the create '
        db      'data file command.',0
        CALL    HIDE_CUR
        MOV     [Color],CL                              ;restore assigned color
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DATA_INSTRU
;
;    Has the Esc key been pressed?
;      Input = none
;      Output = carry flag if abort = yes
PROC    IS_ESC
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------Check keyboard buffer to see if the <Esc> key been pressed?
        MOV     AX,0600h                                ;DOS function # 6
        MOV     DL,0FFh                                 ;read char from key-
        INT     21h                                     ;board buffer.
        JZ      IES1                                    ;NO key pressed continue
        CMP     AL,1Bh                                  ;was it the <ESC> key?
        JNZ     IES1                                    ;if NO continue
        CALL    ABORT_MESS                              ;if YES inform user
        JC      IES2                                    ;carry flag = abort
IES1:   CLC                                             ;clear carry flag
IES2:   POP     DX
        POP     CX
```

```
        POP     BX
        POP     AX
        RET
ENDP    IS_ESC
;
;-------Instructions for the Create data command.
;       Input = None
;       Output = None
;
PROC    DISPLAY_PROGRESS
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     CL,[Color]                              ;get assigned color
        MOV     AL,[Normal]                             ;get color
        MOV     [Color],AL                              ;set color
        MOV     AX,0A2Fh                                ;row/column
        CALL    GOTOYX
        MOV     AX,Offset OutBuf                        ;ptr to ID string
        MOV     DX,AX                                   ;store start of buffer
        ADD     AX,20                                   ;ptr to EndOfString
        MOV     BX,AX                                   ;ptr to bx register
        MOV     BYTE PTR [BX],0                         ;place EndOfStr marker
        MOV     AX,DX                                   ;restore start ptr
        CALL    DSTR_OUT                                ;display ID string
        MOV     AX,0C38h                                ;row/column
        CALL    GOTOYX
        MOV     AX,[MaxID]                              ;get maximum ID
        MOV     BX,AX                                   ;store maximum ID
        CALL    BIN_OUT                                 ;display value to
        CALL    HIDE_CUR                                ;the screen
        INC     BX                                      ;add 1 to maximun
        MOV     [MaxID],BX                              ;save new value
        MOV     [Color],CL                              ;restore assigned color
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DISPLAY_PROGRESS
;
;
;       Input = none
;       Output = none
PROC    FILE_ERR
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    CLEAR_MESSAGE
        MOV     AL,[Warning]                            ;warning color
        MOV     CL,[Color]                              ;save original color
        MOV     [Color],AL                              ;set color
        MOV     AX,0207h                                ;row 3/Col 8
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      ' Use the "File" command to open the data file.'
        db      '  Press Any Key. ',0
        MOV     [Color],CL                              ;restore original color
        CALL    HIDE_CUR
```

```
        CALL    ERR_SOUND
        CALL    GET_CHAR
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    FILE_ERR
;
;
; Close all answer files.
;       input = none
;       output = dos error message
;       NOTE: This subroutine always clears the carry flag
; Answer File type names:  POS1 POS2 PER SUP and SUB
;
PROC    CLOSE_ANSWER_FILES
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     BX,Offset SF1HD                 ;ptr to file handle
        MOV     AX,[BX]                         ;get file handle
        CMP     AX,0                            ;is file open
        JZ      CAF1                            ;if NO goto next file
        CALL    CLOSE                           ;close the file
        JC      CAF1                            ;if Dos error next file
        XOR     AX,AX                           ;zero to ax
        MOV     [BX],AX                         ;mark file closed
CAF1:   MOV     BX,Offset SF2Hd                 ;ptr to file handle
        MOV     AX,[BX]                         ;get file handle
        CMP     AX,0                            ;is file open
        JZ      CAF2                            ;if NO goto next file
        CALL    CLOSE                           ;close the file
        JC      CAF2                            ;if Dos error next file
        XOR     AX,AX                           ;zero to ax
        MOV     [BX],AX                         ;mark file closed
CAF2:   MOV     BX,Offset PERHd                 ;ptr to file handle
        MOV     AX,[BX]                         ;get file handle
        CMP     AX,0                            ;is file open
        JZ      CAF3                            ;if NO goto next file
        CALL    CLOSE                           ;close the file
        JC      CAF3                            ;if Dos error next file
        XOR     AX,AX                           ;zero to ax
        MOV     [BX],AX                         ;mark file closed
CAF3:   MOV     BX,Offset SUPHd                 ;ptr to file handle
        MOV     AX,[BX]                         ;get file handle
        CMP     AX,0                            ;is file open
        JZ      CAF4                            ;if NO goto next file
        CALL    CLOSE                           ;close the file
        JC      CAF4                            ;if Dos error next file
        XOR     AX,AX                           ;zero to ax
        MOV     [BX],AX                         ;mark file closed
CAF4:   MOV     BX,Offset SUBHd                 ;ptr to file handle
        MOV     AX,[BX]                         ;get file handle
        CMP     AX,0                            ;is file open
        JZ      CAF5                            ;if NO goto next file
        CALL    CLOSE                           ;close the file
        JC      CAF5                            ;if Dos error next file
        XOR     AX,AX                           ;zero to ax
        MOV     [BX],AX                         ;mark file closed
CAF5:   CLC                                     ;clear carry = data OK!
```

```
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    CLOSE_ANSWER_FILES
;

;
;  Are the first 20 characters of the line valid ID characters?
;  That means digits,spaces,or upper case letters.
;       input = AX = ptr to input buffer
;       output = cf if not a valid ID or if all 20 characters are spaces.
PROC    IS_ID
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     CX,20                                   ;loop counter
        MOV     BX,AX                                   ;ptr to beg of table
        XOR     AX,AX                                   ;set char count = 0
ISI1:   CMP     BYTE PTR [BX],0                         ;is it end of line?
        JZ      ISI8                                    ;if YES exit FALSE
        CMP     BYTE PTR [BX],20h                       ;is it a space?
        JZ      ISI7                                    ;if YES goto next byte
        CMP     BYTE PTR [BX],'0'                       ;is it < 0
        JC      ISI8                                    ;if YES exit FALSE
        CMP     BYTE PTR [BX],'9'+1                     ;is it a ascii digit?
        JC      ISI6                                    ;if YES goto next byte
        CMP     BYTE PTR [BX],'A'                       ;is it < A
        JC      ISI8                                    ;if YES exit FALSE
        CMP     BYTE PTR [BX],'Z'+1                     ;is upper case letter?
        JNC     ISI8                                    ;if NO exit False
ISI6:   INC     AX                                      ;count character
ISI7:   INC     BX                                      ;ptr to next byte
        LOOP    ISI1                                    ;loop 20 times
        CMP     AX,0                                    ;were any chars counted?
        JNZ     ISI9                                    ;if <> 0 exit OK!
ISI8:   STC                                             ;carry flag = NO ID
ISI9:   POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    IS_ID
;
;
;Read a record from the DAT file to the SF1Buf.
;       Input = AX = record number (1  to MaxID)
;       Output = Carry flag = error
;       Note: reads to the [SF1Buf] because the normal input [FilBuf]
;             is being used to hold the strings to be printed.
PROC    READ_DAT_REC
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    BP
        MOV     BP,AX                                   ;save Rec Number
;------is record number in range ?
        DEC     AX                                      ;is rec number = 0 ?
        JC      RDR2                                    ;if Yes exit error
```

```
        CMP     AX,[MaxId]                              ;is rec no. to large ?
        JC      RDR1                                    ;if NO goto next test
        STC                                             ;else set error flag
        JMP     SHORT RDR2                              ;exit on error
;------position the DATA file pointer.
RDR1:   MOV     CX,[DRecLn]                             ;get length of each rec
        MUL     CX                                      ;compute offset in file
        MOV     CX,DX                                   ;high word of offset
        MOV     DX,AX                                   ;low word of offset
        MOV     BX,[DATHd]                              ;get file handle
        MOV     AX,4200h                                ;set file ptr function
        INT     21h                                     ;set pointer
        JC      RDR2                                    ;exit if error.
;------read the record into the file buffer.
        MOV     AX,3F00h                                ;read file function no
        MOV     CX,[DRecLn]                             ;number of bytes to read
        MOV     DX,Offset SF1Buf                        ;ptr to destination
        INT     21h                                     ;read the record
        CMP     AX,CX                                   ;was the read complete?
        JZ      RDR3                                    ;exit if NO error
RDR2:   CALL    CLEAR_MESSAGE
        MOV     CL,[Color]                              ;store original Color
        MOV     AL,[Warning]                            ;warning color
        MOV     [Color],AL                              ;set color
        MOV     AX,0207h                                ;row /Col
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      " ERROR reading Data file record: ",0
        MOV     AX,BP                                   ;get record number
        CALL    BIN_OUT                                 ;send number to screen
        CALL    CSTR_OUT
        db      ".  Press any key to Abort. ",0
        MOV     [Color],CL                              ;restore original color
        CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR
        STC
RDR3:   POP     BP
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    READ_DAT_REC
;
;Read a seven byte record from the rank file.
;       Input = none  (assumes file ptr is in the correct position)
;       Output = Carry flag = error
;       Note: reads to the [OutBuf] because the normal input [FilBuf]
;             is being used to hold the strings to be printed.
;
PROC    READ_RNK_REC
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------read five bytes from the rank file.
        MOV     AX,3F00h                                ;read file function no
        MOV     BX,[RNKHd]                              ;get rank file handle
        MOV     CX,7                                    ;number of bytes to read
        MOV     DX,Offset OutBuf                        ;ptr to destination
        INT     21h                                     ;read the record
```

```
        CMP     AX,CX                                   ;was the read complete?
        JZ      RRR3                                    ;exit if NO error
RRR2:   CALL    CLEAR_MESSAGE
        MOV     CL,[Color]                              ;store original Color
        MOV     AL,[Warning]                            ;warning color
        MOV     [Color],AL                              ;set color
        MOV     AX,0207h                                ;row /Col
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      " ERROR reading Rank file record. "
        db      "  Press any key to Abort. ",0
        MOV     [Color],CL                              ;restore original color
        CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR
        STC
RRR3:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    READ_RNK_REC
;

;
;
; Create a new data file from the scanned answer file.
;       input = none  (files described in Config Table assumed)
;       output = cf = error in creating the data file.
;       The inner loop beginning a MAD6: uses the following registers:
;       SI = ptr into Config table    DI = ptr into OutBuf
;       BP = ptr into question no string in heap
;       BX = ptr into Input buffer: FilBuf or SF1Buf for a SF1 file.
;       CX = stores answer total in CH and number of answers in CL
;       DX = temp storage of current input file handle.
;
PROC    MAKE_DATA_FILE
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        XOR     AX,AX                                   ;row 0 and column 0
        CALL    MENU_BOX                                ;draw top menu box
        CALL    OPEN_ANSWER_FILES                       ;open answer files
        JC      MAD1                                   ;exit on Dos error
;------At this point all answer files are open with the read ptr is at the top
        CALL    WARNING_MESS                            ;Do you want to cont ?
        JC      MAD1                                    ;if NO then exit
        CALL    CLEAR_TITLE
        CALL    DATA_INSTRU                             ;display data mess box
        CALL    IS_FULL                                 ;is room available?
        JC      MAD1                                    ;exit on disk full
        MOV     WORD PTR [MaxID],0                      ;set counter = 0
        CALL    OPEN_NEW_DATA_FILE                      ;open an empty file
        JNC     MAD2                                    ;continue if NO error
MAD1:   JMP     MAD12                                   ;exit on open error
MAD2:   CALL    DELETE_NDX_FILE
        CALL    DELETE_RANK_FILE
;------beginning of outer loop to read one ID from the SP1 answer file.
MAD3:   CALL    CLEAR_ANSWERS                           ;hex 0's to ans buffers
        CALL    READ_SF1                                ;get first ID & answers
        JNC     MAD4                                    ;if no error contnue
```

```
        JMP     MAD11                                   ;file closed or EOF
MAD4:   MOV     AX,Offset SF1Buf                        ;ptr to input buffer
        CALL    IS_ID                                   ;is this a data line?
        JNC     MAD5                                    ;if YES continue
        JMP     MAD10                                   ;if NO read next line
;------copy ID to output buffer                        ;else process data
MAD5:   MOV     AX,DS                                   ;assign ES to the
        MOV     ES,AX                                   ;data section
        MOV     SI,Offset SF1Buf                        ;ptr to ID name
        MOV     DI,Offset OutBuf                        ;ptr to output buffer
        MOV     CX,10                                   ;no of words to move
        CLD                                             ;auto inc SI & DI
        REP     MOVSW                                   ;move type to Search
        CALL    DISPLAY_PROGRESS                        ;inform user of progress
;------fine matching ID's in all the answer files.
        CALL    LOCATE_ALL_IDS
        JC      MAD11                                   ;abort on ESC key
        MOV     DI,Offset OutBuf + 20                   ;ptr for 1st mean
        MOV     SI, Offset CFGTbl - 4                   ;ptr to table
;------beginning of inter loop to compute means for one ID number
MAD6:   ADD     SI,4                                    ;ptr to next type
        CMP     WORD PTR [SI],8                         ;is it EndOfTable?
        JZ      MAD9                                   ;OK! exit inter loop
;------is this an answer file ?
        CMP     WORD PTR [SI],6                         ;is it an answer file?
        JNC     MAD6                                    ;if NO continue loop
;------set answer number pointer
        MOV     BP,[SI + 2]                             ;ptr to heap string
;------set default values
        XOR     AL,AL
        MOV     [DI],AL                                 ;set means = 0
        XOR     CX,CX                                   ;CH = total CL = items
;------is this a self part 1 file ?
        MOV     BX,Offset SF1Buf                        ;in case it is self 1
        CMP     WORD PTR [SI],5                         ;is it a SF1 file?
        JZ      MAD7                                    ;if YES compute mean
;------is this a self part 2 file ?
        MOV     BX,Offset SF2Buf                        ;in case it is self 2
        CMP     WORD PTR [SI],4                         ;is it a SF2 file?
        JZ      MAD7                                    ;if YES compute mean
;------is this a peer file ?
        MOV     BX,Offset PERBuf                        ;in case it is peer
        CMP     WORD PTR [SI],3                         ;is it a PER file?
        JZ      MAD7                                    ;if YES compute mean
;------is this a superior file ?
        MOV     BX,Offset SUPBuf                        ;in case it is superior
        CMP     WORD PTR [SI],2                         ;is it a superior file?
        JZ      MAD7                                    ;if YES compute mean
;------is this a subordinate file ?
        MOV     BX,Offset SUBBuf                        ;in case its subordinate
        CMP     WORD PTR [SI],1                         ;is a subordinate file ?
        JZ      MAD7                                    ;if YES compute mean
        STC                                             ;mark programming error
        JMP     SHORT MAD12                             ;exit this subroutine.
;------get answer total and number of questions answered
MAD7:   CALL    GET_ANSWERS                             ;ans. from data buffer
        JC      MAD8                                    ;no matching ID in buf
        ADD     CL,AL                                   ;update answer total
        ADD     CH,AH                                   ;update no of answers
        INC     BH                                      ;buffer+256= next buff
        CMP     WORD PTR [SI],4                         ;is PER,SUB or SUP file?
        JC      MAD7                                    ;YES look for next ID
```

```
;------compute and code the data points mean              ;CH = total CL = items
MAD8:   CALL    CODE_MEAN                                  ;compute the data point
;------store mean score in output buffer (example: hex 15 = bin 21 or 2.1)
        MOV     [DI],AL                                    ;means to out buffer
        INC      DI
        JMP      SHORT  MAD6                               ;loop thru config table
;------Write record to DAT file.
MAD9:   MOV     CX,[DRecLn]                                ;no. of bytes to write
        MOV     BX,[DATHd]                                 ;file handle to BX
        MOV     AX,4000h                                   ;write to file: func.
no.
        MOV     DX,Offset OutBuf                           ;ptr to data to write
        INT     21h                                        ;write to the file
MAD10:  CMP     BYTE PTR [SF1EOF],0                        ;is it EndOf SF1 File ?
        JNZ     MAD11                                      ;if YES normal exit
        JMP     MAD3                                       ;else loop thru SF1 file
MAD11:  CLC                                                ;clear carry = data OK!
MAD12:  PUSHF                                              ;save flag register
        CALL    CLOSE_ANSWER_FILES
        POPF                                               ;restore flag resister
        POP     DX                                         ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    MAKE_DATA_FILE
;
        .CODE
PROC    GET_PATH
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CMP     BYTE PTR [Path],0                          ;is the Path empty
        JNZ     GEP1                                       ;if NO then display Path
;------get default drive                                  ;if YES get default path
        MOV     AH,19h                                     ;default function
        INT     21h                                        ;get default drive
        ADD     AL,65                                      ;convert to cap letter
        MOV     AH,':'                                     ;place ':' in path
        MOV     SI,Offset Path                             ;ptr to [Path] string
        MOV     [SI],AX                                    ;place drive letter
        INC     SI                                         ;in path.
        INC     SI                                         ;ptr to 3rd byte
        MOV     AL,'\'                                     ;place backslash in
        MOV     [SI],AL                                    ;3rd byte of string
        INC     SI                                         ;point to 4th byte
;------get default path
        MOV     AH,47h                                     ;get current path
        MOV     DL,0                                       ;on default drive
        INT     21h                                        ;get path
GEP1:   CALL    EDIT_PATH
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    GET_PATH

;
;       Input = last path entered or default path in [Path]
;       Output = current path in Input
```

```
;
PROC    EDIT_PATH
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     BX,DS
        MOV     ES,BX
        CALL    PATH_TO_INPUT                   ;move Path str to Input
        CALL    PATH_MESS_TOP
        CALL    PATH_MESS_BTM                   ;edit message
EDT1:   MOV     AX,0106h
        CALL    PATH_EDITOR                     ;edit this field
        JC      EDT2                            ;exit on <Esc> key
        CALL    CHECK_PATH                      ;if valid save path
        JC      EDT1                            ;if Not valid loop
        JMP     SHORT EDT3                      ;exit path OK!
EDT2:   CALL    MENU_INSTRU                     ;draw bottom box
        STC                                     ;carry flag = Esc key
EDT3:   POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    EDIT_PATH
;

PROC    PATH_MESS_TOP
        PUSH    AX
        PUSH    CX
        XOR     AX,AX
        CALL    MENU_BOX
        MOV     CL,[Color]
        MOV     AX,0206h
        CALL    GOTOYX
        MOV     AL,[Menu]
        MOV     [Color],AL
        CALL    CSTR_OUT
        db      'Enter the directory path.',0
        MOV     [Color],CL
        POP     CX
        POP     AX
        RET
ENDP    PATH_MESS_TOP
;
;
;-------remove all but letters from the field and convert into and ASCIIZ string
;       Input = None
;       Output = None
;       Note: fields are 14 bytes long but the last byte is always a hex 0
;             therefore the name fields can only have 13 letters.
PROC    FILTER_FIELD
        PUSH    AX
        PUSH    BX                              ;save original str ptr
        PUSH    CX
        PUSH    DX
        MOV     BX, Offset Input                ;pointer to input str
        MOV     CX,12                           ;string length - 1
TRI1:   MOV     AL,'A'                          ;is character less than
        CMP     [BX],AL                         ;the letter "A" ?
        JNC     TRI3                            ;if yes remove character
TRI2:   CALL    DELETE_CHAR                     ;shift string left
```

```
        DEC     BX                                      ;check same byte again
        JMP     SHORT TRI4
TRI3:   MOV     AL,'Z'                                  ;is character greater
        CMP     AL,[BX]                                 ;than letter "Z" ?
        JC      TRI2                                    ;if yes remove character
TRI4:   INC     BX                                      ;ptr to next byte
        LOOP    TRI1                                    ;check next byte
;-------convert trailing spaces to hex 0.
        MOV     CX,13                                   ;loop counter
        MOV     BX,Offset Input + 12                    ;ptr to LastByte
        MOV     AX,20h                                  ;AH = hex 0  AL = space
TRI5:   CMP     AL,[BX]                                 ;is char a <space> ?
        JNZ     TRI6                                    ;if no exit.
        MOV     [BX],AH                                 ;mark as end of string
        DEC     BX                                      ;ptr to last byte
        LOOP    TRI5                                    ;loop until beg of str
TRI6:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
;-------- delete a character at the cursor
PROC    DELETE_CHAR
        PUSH    AX
        PUSH    BX                                      ;save original str ptr
DEP1:   MOV     AX,[BX]                                 ;read ptr BX and BX+1
        CMP     AH,0                                    ;Is it the end of str?
        JZ      DEP2                                    ;if yes then done.
        MOV     [BX],AH                                 ;place BX+1 in BX
        INC     BX                                      ;point to next byte
        JMP     SHORT DEP1                              ;loop until end of str.
DEP2:   MOV     AH,' '                                  ;place a <space> at
        MOV     [BX],AH                                 ;end of the string.
        POP     BX                                      ;restore original ptr
        POP     AX
        RET
ENDP    DELETE_CHAR
ENDP    FILTER_FIELD
;
;-------Instructions for entering the path name.
;       Input = None
;       Output = None
;
PROC    PATH_MESS_BTM
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,1500h                                ;row 21,column 0
        CALL    MENU_BOX                                ;draw menu box
        MOV     AX,160Ah                                ;row 22,column 13
        CALL    GOTOYX
        MOV     AL,[Color]                              ;get current color
        MOV     CL,AL                                   ;store in CL
        MOV     AL,[Menu]                                 ;set color = menu
        MOV     [Color],AL
        CALL    CSTR_OUT
        db      'Type the complete path name for the directory to be
searched.',0
        MOV     AX,1708h                                ;row 23,column 13
        CALL    GOTOYX
        CALL    CSTR_OUT
```

```
        db      'Press the <Enter> key to continue or the <Esc> key for '
        db      'the Menu.',0
        MOV     [Color],CL                              ;restore orig. color
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PATH_MESS_BTM
;
;
;-------Copy path ASCIIZ string in [Path] to [Input].
;       Input = none
;       Output = none
;       AX-DX register saved.
;
PROC    PATH_TO_INPUT
        PUSH    AX
        PUSH    BX                                      ;save registers
        PUSH    CX
        PUSH    DX
;------fill [input] with 68 spaces
        MOV     AX,DS                                   ;Make ES = DS
        MOV     ES,AX
        MOV     CX,67                                   ;max number of chars
        MOV     BX,Offset Input                         ;pointer to ASCIIZ str
        MOV     AL,' '                                  ;space Al register
        MOV     [BX],AL                                 ;0 to first byte of str
        MOV     DI,BX                                   ;DI = pointer to next
        INC     DI                                      ;byte of string
        MOV     SI,BX                                   ;SI = pointer to str
        CLD                                             ;auto inc DI and SI
        REP     MOVSB                                   ;fill string with 0's
;------copy current [path] to [input]
        MOV     DI,Offset Input                         ;destination offset
        MOV     SI,Offset Path                          ;source offset
        CLD                                             ;auto inc DI and SI
EDT0:   MOVSB                                           ;copy one byte
        CMP     BYTE PTR [SI],0                         ;is next char = 0
        JNZ     EDT0
        CLC
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PATH_TO_INPUT
;
;------ Get an ASCIIZ string input from the keyboard.
;       Input = AX = Row/Column position on the screen
;               [Input] must contain the string to be edited before
;               calling this subroutine.
;               [insert] <> 0 places the editor in the insert mode.
;       Output = AL = Exit 'Char'
;                ASCIIZ string at [Input] in the data section.
;                BX-DX register saved
;       Note: the follow register hold the following local variables.
;                AL = Input Character
;                BX = ptr in [Input] string
;                CX = Row/Col cursor position
;                DX = Starting Row/Col position
;
```

```
PROC    PATH_EDITOR
        PUSH    BX                      ;save registers
        PUSH    CX
        PUSH    DX
        MOV     DX,AX                   ;save row/column in DX
        CALL    GOTOYX                  ;set cursor position
        MOV     BX,Offset Input         ;ptr to [Input] strr.
        MOV     AX,BX                   ;[Input] ptr to AX
        CALL    DSTR_OUT                ;Display blanks
;------fine first space in string
        MOV     SI,0                    ;zero to SI
PATA:   INC     SI                      ;ptr to next char
        CMP     SI,67                   ;stop if no spaces
        JZ      PATB                    ;safty value
        CMP     BYTE PTR [BX + SI],' '  ;is this a space ?
        JNZ     PATA                    ;if NO check next char
PATB:   MOV     CX,SI                   ;offset to CX
        ADD     CX,DX                   ;row/column ptr to CX
        ADD     BX,SI                   ;advance BX pointer
;-------beginning of input loop
        CALL    PATH_INSERT             ;display insert status.
PAT0:   MOV     AX,CX                   ;cursor position to AX
        CALL    GOTOYX                  ;set cursor position
        CALL    GET_TEXT
;-------Return key
        CMP     AL,0Dh                  ;is it a <return> ?
        JZ      PAT10                   ;if yes exit.
;-------Is it any other control character?
        CMP     AL,1Bh                  ;is it a control char?
        JNC     PAT5                    ;jmp = not control char
        CALL    PATH_CONTROL_CHAR       ;handel control char
        JMP     SHORT PAT0              ;get next character
;-------is it the <Esc> key  ?
PAT5:   STC                             ;set carry flag
        JZ      PAT11                   ;exit on <Esc> key
;-------filter unwanted characters
        CALL    CHAR_FILTER
        JC      PAT0                    ;carry flag = not char
;-------check the insert mode
PAT7:   MOV     AH,[Insert]             ;get Insert flag
        CMP     AH,0                    ;Is insert OFF? = 0
        JZ      PAT8                    ;skip if turned off
        CALL    SHIFT_STR_RT            ;move rest of str right
;-------place the character in the [Input] string.
PAT8:   MOV     [BX],AL                 ;place char in [Input]
        MOV     AX,BX                   ;prt rest of string
        CALL    DSTR_OUT                ;display string from
;-------see if 'end of string' is true.
        XOR     AH,AH
        CMP     AH,[BX+1]               ;Is 'end of string' ?
        JZ      PAT0                    ;yes=do not move cursor
        INC     BX                      ;advance [Input] ptr.
        INC     CX                      ;advance cursor
        JMP     PAT0                    ;if not continue input.
PAT10:  CLC                             ;clear carry flag
PAT11:  POP     DX
        POP     CX
        POP     BX
        RET
;
;-------Display the status of the [Insert] flag to screen.
;       Input = None
```

```
;       Output = None
;       AX - DX registers saved
PROC    PATH_INSERT
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     DL,[Color]                      ;save current color
        MOV     AL,[HiLite]                     ;set color for insert
        MOV     [Color],AL                      ;string.
        MOV     AX,0420h                        ;row  col
        CALL    GOTOYX                          ;set cursor
        XOR     AX,AX                           ;zero AX
        ADD     AL,[Insert]                     ;get Insert flag
        JNZ     PAH1                            ;<> 0 = Insert mode
        CALL    CSTR_OUT                        ;clear insert from
        db      '          ',0                  ;the screen.
        JMP     SHORT PAH2                     ;exit.
PAH1:   CALL    CSTR_OUT                        ;send the following
        db      '<Insert On>',0                    ;string to the screen
PAH2:   MOV     [Color],DL                      ;restore current color.
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PATH_INSERT
;
;-------Check Control Characters
;        Input AL = Control Character
;              BX = ptr in [Input] string
;              CX = Row/Col cursor position
;              DX = Starting Row/Col position
;        OutPut jumps back to get another character.
PROC    PATH_CONTROL_CHAR
;-------Backspace key
        CMP     AL,08h                          ;is it a Backspace key
        JNZ     CNN0                            ;if not continue.
        CALL    BACKSPACE                       ;del char left of cur.
;-------Insert key
CNN0:   CMP     AL,16h                          ;is it the insert key?
        JNZ     CNN1                            ;if not continue.
        PUSH    AX                              ;save Char
        XOR     AX,AX                           ;zero AX
        ADD     AL,[Insert]                     ;get Insert flag
        JZ      CNN0A                           ;if zero jump
        MOV     AL,AH                           ;make flag = 0
        JMP     SHORT CNN0B                     ;replace flag
CNN0A:  DEC     AL                              ;make flag = FFh
CNN0B:  MOV     [Insert],AL                     ;replace flag
        CALL    PATH_INSERT                     ;display insert status.
        POP     AX                              ;restore Char
;-------Home key
CNN1:   CMP     AL,1h                           ;is it the Home key?
        JNZ     CNN2                            ;if not continue.
        MOV     BX,Offset Input
        MOV     CX,DX
;-------End key
CNN2:   CMP     AL,6h                           ;is it the End key?
        JNZ     CNN3                            ;if not continue.
        CALL    END_STR
;-------Delete key
```

```
CNN3:   CMP     AL,07h                  ;is it the delete key?
        JNZ     CNN4                    ;if not continue.
        CALL    DELETE                  ;delete char at cursor.
;-------left arrow key
CNN4:   CMP     AL,13h                  ;is it a left arrow key
        JNZ     CNN6                    ;if not continue.
        CMP     CX,DX                   ;beginnig of the string?
        JZ      CNN6                    ;yes = beg. of line
        DEC     BX                      ;so loop will continue.
        DEC     CX
;-------right arrow key
CNN6:   CMP     AL,4                    ;Is it Rt Arrow key?
        JNZ     CNN8                    ;if not jump.
        XOR     AH,AH
        CMP     BYTE PTR [BX+1],0       ;Is 'end of string' ?
        JZ      CNN8                    ;if = 0 no right
        INC     BX                      ;advance pointer
        INC     CX
CNN8:   RET
ENDP    PATH_CONTROL_CHAR
;
;-------move cursor to end of string
PROC    END_STR
        PUSH    AX
CON2A:  MOV     AX,[BX]                 ;check for end of str.
        CMP     AH,0                    ;zero = end of string
        JZ      CON2B                   ;ret on end of string
        INC     BX                      ;advance pointer
        INC     CX                      ;advance cursor
        JMP     SHORT CON2A
CON2B:  POP     AX
        RET
ENDP    END_STR
;
; ------insert a character at the cursor.
PROC    SHIFT_STR_RT
        PUSH    AX                      ;save new character
        PUSH    BX                      ;save str pointer
        MOV     AL,[BX]                 ;load char to be moved
        INC     BX                      ;ptr to the next char
SHI1:   MOV     AH,[BX]                 ;load next char.
        CMP     AH,0                    ;is it the end of str?
        JZ      SHI2                    ;if yes then Exit.
        MOV     [BX],AL                 ;last char in the str.
        MOV     AL,AH                   ;next char to last char
        INC     BX                      ;ptr for new next char
        JMP     SHORT SHI1              ;loop until end of str.
SHI2:   POP     BX                      ;restore str pointer
        POP     AX                      ;restore original ptr
        RET
ENDP    SHIFT_STR_RT
;
;------- delete a character at the cursor
PROC    DELETE
        PUSH    AX
        PUSH    BX                      ;save original str ptr
DEL1:   MOV     AX,[BX]                 ;read ptr BX and BX+1
        CMP     AH,0                    ;Is it the end of str?
        JZ      DEL2                    ;if yes then done.
        MOV     [BX],AH                 ;place BX+1 in BX
        INC     BX                      ;point to next byte
        JMP     SHORT DEL1              ;loop until end of str.
```

```
DEL2:   MOV     AH,' '                          ;place a <space> at
        MOV     [BX],AH                         ;end of the string.
        POP     BX                              ;restore original ptr
        MOV     AX,BX                           ;str pointer to AX
        CALL    DSTR_OUT                        ;display string
        POP     AX
        RET
ENDP    DELETE
;
;------- delete a character to the left of the cursor
PROC    BACKSPACE
        PUSH    AX
        MOV     AX,Offset Input                 ;is the cursor at the
        CMP     AX,BX                           ;beginnig of the string?
        JZ      BA3                             ;if yes ignor backspace
        DEC     BX                              ;line pointer left
        DEC     CX                              ;cursor left
        PUSH    BX                              ;save original str ptr
BA1:    MOV     AX,[BX]                         ;read ptr BX and BX+1
        CMP     AH,0                            ;Is it the end of str?
        JZ      BA2                             ;if yes then done.
        MOV     [BX],AH                         ;place BX+1 in BX
        INC     BX                              ;point to next byte
        JMP     SHORT BA1                       ;loop until end of str.
BA2:    MOV     AH,' '                          ;move <space> to AH
        MOV     [BX],AH                         ;place in last position
        POP     BX                              ;restore original ptr
        MOV     AX,CX                           ;row/column to AX
        CALL    GOTOYX                          ;set cursor position
        MOV     AX,BX                           ;str pointer to AX
        CALL    DSTR_OUT                        ;display string
BA3:    POP     AX
        RET
ENDP    BACKSPACE
;
;-------Filter out unwanted ASCII characters and capitalize letters
;       Input = Char in AL
;       Output = Carry  Flag = not a good character. get another!
PROC    CHAR_FILTER
        AND     AL,7Fh                          ;make 0 - 127 ASCII.
        CMP     AL,' '                          ;is it a control char?
        JC      CHAR1                           ;if yes, get next char.
        CMP     AL,'a'                          ;is char a small letter
        JC      CHAR0                           ;if not, Ok continue.
        AND     AL,0DFh                         ;change to capital char
CHAR0:  CLC                                     ;clear carry flag
CHAR1:  RET
ENDP    CHAR_FILTER
ENDP    PATH_EDITOR
;
;
;-------Check [Input] to see if the path is Ok!
;       Input = AX = Assumed [Input] hold a Path
;       Output = Carry flag is not a valid path name
;       AX-DX register saved.
;
PROC    CHECK_PATH
        PUSH    AX
        PUSH    BX                              ;save registers
        PUSH    CX
        PUSH    DX
        MOV     AX,DS                           ;Make ES = DS
```

```
        MOV     ES,AX
;------remove all leading spaces
        MOV     BX,Offset Input                 ;ptr to Input string
CHE0:   CMP     BYTE PTR [BX],' '               ;is leading space ?
        JNZ     CHE1                            ;if NO continue
        MOV     CX,68                           ;else remove space
        MOV     DI,BX                           ;offset to 1st byte
        MOV     SI,BX
        INC     SI                              ;offset to 2nd byte
        CLD                                     ;auto inc DI and SI
        REP     MOVSB                           ;shift line left
        JMP     SHORT CHE0                      ;check for leading space
;------convert first ASCII space to a hex zero EndofStr marker
CHE1:   MOV     CX,68                           ;max string length
        MOV     BX,Offset Input - 1
CHE2:   INC     BX
        CMP     BYTE PTR [BX],' '               ;is it a space?
        JC      CHE4                            ;exit if char < ' '
        LOOPNZ  CHE2                            ;is NO loop
;------remove trailing back slash
        DEC     BX                              ;ptr to last char
        CMP     BYTE PTR [BX],'\'               ;is it a back slash ?
        JZ      CHE3                            ;if Yes remove from str
        INC     BX                              ;if NO leave in str
 ------place : after drive name?
CHE3:   MOV     AX,BX                           ;ptr in str to ax
        SUB     AX,Offset Input                 ;string length in AX
        JZ      CHE4                            ;if OK! goto next test
        CMP     AX,3                            ;is less than 3?
        JNC     CHE4                            ;if NO goto next test
        MOV     AX,003Ah                        ;3Ah = ':'
        MOV     BX,Offset Input + 1             ;ptr to 2nd byte
        MOV     [BX],AX
        INC     BX
CHE4:   MOV     [BX],CH                         ;mark EndofStg = 0
;------is the path valid
        CALL    IS_PATH                         ;is path valid ?
        JNC     CHE5                            ;NOT carry = OK!
        CALL    PATH_ERROR                      ;display error message
        STC                                     ;set cf = error
        JMP     SHORT CHE7
;------save valid path string in [Path]
CHE5:   MOV     SI,Offset Input                 ;source offset
        MOV     DI,Offset Path                  ;destination offset
        CLD                                     ;auto inc DI and SI
CHE6:   MOVSB                                   ;copy one byte
        CMP     BYTE PTR [SI],0                 ;is next char = 0
        JNZ     CHE6                            ;copy bytes
        XOR     AL,AL
        MOV     [DI],AL                         ;zero = EndOfString
        CLC                                     ;clear carry flag
CHE7:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
;
;-------Is this a Valid path?
;       Input = ASCIIZ drive/directory string in [Input]
;       Output = carry flag in not a valid path
;       AX - DX registers saved
PROC    IS_PATH
```

```
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------copy string to [Search]
        MOV     SI,Offset Input                         ;source offset
        MOV     DI,Offset Search                        ;destination offset
        CLD                                             ;auto inc DI and SI
ISP1:   MOVSB                                           ;copy one byte
        CMP     BYTE PTR [SI],0                         ;is next char = 0
        JNZ     ISP1                                    ;copy bytes
;------place \*.* at end of string
        MOV     AX,'*\'
        MOV     [DI],AX
        INC     DI
        INC     DI
        MOV     AX,'*.'
        MOV     [DI],AX
        INC     DI
        INC     DI
        XOR     AX,AX
        MOV     [DI],AX
;------see if path is OK!
        MOV     DX,Offset Search                        ;ptr to ASCIIZ string
        MOV     AX,4E00h                                ;Find function no.
        MOV     CX,0010h                                ;directory search
        INT     21h                                     ;do search
        CMP     AL,3                                    ;is path BAD ?
        CLC                                             ;clear carry flag
        JNZ     ISP2                                    ;OK! if not 3
        STC                                             ;set error flag
ISP2:   POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    IS_PATH
;
;-------Display Path error message.
;       Input = None
;       Output = None
;       AX - DX registers saved
PROC    PATH_ERROR
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AL,' '                                  ;replace hex 0 with
        MOV     [BX],AL                                 ;a space
        MOV     CL,[Color]                              ;save current color
        MOV     AL,[Warning]                            ;warning color
        MOV     [Color],AL                              ;set color
        MOV     AX,0222h                                ;row 5 Col 7
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      ' Error: Invalid path.  Press Any Key. ',0
        CALL    HIDE_CUR
        CALL    ERR_SOUND
        CALL    GET_CHAR                                ;wait for keyboard key
        MOV     AL,[Menu]                               ;menu color
        MOV     [Color],AL                              ;set color
        MOV     AX,0222h                                ;row 5 Col 7
```

```
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;clear warning
        db      '                                      ',0
        MOV     [Color],CL                              ;restore original Color
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PATH_ERROR
ENDP    CHECK_PATH
;
;
;-------Select a Select a Key file.
;
;       Input = None
;       Output = Carry Flag if DOS Error
;       Local variables:
;        BH = hilite bar position # 1 to 14
;        BL = starting directory number # 1 to MaxFile
;
PROC    SELECT_FILE
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CALL    CLOSE_ALL_FILES                         ;close any open file
        CALL    SET_TYPE                                ;POS or NEG variables
        CALL    CREATE_MEM_DIR                          ;make a memory directory
        JC      SEL5                                    ;exit if error
        CALL    SELECT_SCREEN                           ;display select screen
        JC      SEL4                                    ;no files in directory
;------display the files
        MOV     BX,[BarPos]                             ;starting position
SEL0:   MOV     AX,BX                                   ;file variables to AX
        CALL    DISPLAY_FILES                           ;files to screen
SEL1:   CALL    GET_CHAR                                ;get keyboard input
        CMP     AL,1Bh                                  ;is it an Esc key ?
        JNZ     SEL2                                    ;if not goto next test
        JMP     SEL4                                    ;Exit
SEL2:   CMP     AL,0Dh                                  ;is it a pick ?
        JZ      SEL3                                    ;if YES exit loop
;-------see if an active control key was pressed
        CALL    CONTROL_KEYS
        JC      SEL1                                    ;no change get char
        JMP     SHORT SEL0                              ;redraw file window
;------Open configuration file.
SEL3:   CALL    MOVE_NAME                               ;file name to data seg
        CALL    OPEN_CONFIG                             ;open configuration file
        JC      SEL5                                    ;Exit on DOS error
        MOV     [BarPos],BX                             ;save current position
        CALL    READ_DATE                               ;get DOS date of file
        CALL    READ_CONFIG                             ;read file into memory
        JC      SEL5                                    ;exit to main menu
        CALL    OPEN_DATA_FILE                          ;is data available?
SEL4:   CALL    RELEASE_MEM_DIR                         ;release mem block
        CLC                                             ;clear carry flag
        JMP     SHORT SEL6
SEL5:   CALL    RELEASE_MEM_DIR                         ;release mem block
        STC                                             ;error set carry flag
SEL6:   POP     DX
        POP     CX
```

```
        POP     BX
        POP     AX
        RET
;
;    Adjust the highlight bar position variables in BX register
;       Input = AL = Char for the keyboard
;               BH = hilite bar position # 1 to 14
;               BL = starting directory number # 1 to MaxFile
;       Output  Carry Flag = no change in BX
;
PROC    CONTROL_KEYS
;-------is it a Down arrow ?
        CMP     AL,24                           ;is it Down arrow?
        JNZ     KYS2                            ;if not goto next test
        MOV     AX,BX                           ;get current variables
        ADD     AH,AL                           ;inc hilllite bar
        CMP     [MaxFile],AH                    ;is it end of file ?
        JC      KYS10                           ;if yes Exit no changes
        CMP     BH,14                           ;is bottom of window ?
        JZ      KYS1                            ;if yes inc starting
        INC     BH                              ;else inc bar number
        JMP     SHORT KYS9                      ;exit
KYS1:   INC     BL                              ;inc starting number
        JMP     SHORT KYS9                      ;display new directory
;-------is it an Up arrow ?
KYS2:   CMP     AL,5                            ;is it Up arrow ?
        JNZ     KYS4                            ;if no goto next test
        CMP     BX,0101h                        ;is it begining of file
?
        STC                                     ;set carry for ret
        JZ      SHORT KYS10                     ;if yes Exit no changes
        CMP     BH,1                            ;is top of window ?
        JZ      KYS3                            ;if yes Dec starting
        DEC     BH                              ;else dec bar number
        JMP     SHORT KYS9                      ;exit
KYS3:   DEC     BL                              ;dec starting number
        JMP     SHORT KYS9                      ;display new directory
;-------Is it a Home Key ?                      ;go to Top of Directroy
KYS4:   CMP     AL,1                            ;is it Home key?
        JNZ     KYS5                            ;if not goto next test
        MOV     BX,0101h                        ;set top of file
        JMP     SHORT KYS9                      ;display new directory
;-------Is it a End Key ?                       ;go Bottom of Directory
KYS5:   CMP     Al,6                            ;is it End key ?
        JNZ     KYS7                            ;if not goto next test
        MOV     AH,[MaxFile]                    ;get number of files
        CMP     AH,15                           ;are # of files > 14 ?
        JNC     KYS6                            ;if yes jump to lastpage
        MOV     BL,1                            ;starting position
        MOV     BH,AH                           ;hilite last entry.
                                                ;
        JMP     SHORT KYS9
KYS6:   SUB     AH,13                           ;maxfiles-13 = starting
        MOV     BL,AH                           ;store in BL
        MOV     BH,14                           ;hilite bottom of window
        JMP     SHORT KYS9                      ;display directory
;-------Is it the PageUp Key ?
KYS7:   CMP     AL,18                           ;is it the pageup key ?
        JNZ     KYS8                            ;if goto next test
        MOV     AH,BL                           ;starting # to AL
        SUB     AH,14                           ;subtract 14 = pageup
        MOV     AL,1                            ;char for homekey
```

```
        JLE     KYS4                                    ;no goto homekey
        MOV     BL,AH                                   ;change page
        JMP     SHORT KYS9                              ;display new page
;-------Is it the PageDn Key ?
KYS8:   CMP     AL,3                                    ;is it the pagedn key ?
        STC                                             ;set carry flag for ret
        JNZ     KYS10                                   ;if goto next test
        MOV     AH,BL                                   ;starting # to AL
        MOV     AL,[MaxFile]                            ;number of dir files
        SUB     AL,14                                   ;ptr to top of lastpage
        ADD     AH,14                                   ;page = page + 1
        CMP     AL,AH                                   ;is this the lastpage?
        MOV     AL,6                                    ;char for End key
        JLE     KYS5                                    ;if yes goto End
        MOV     BL,AH                                   ;else go Page + 1
KYS9:   CLC
KYS10:  RET
ENDP    CONTROL_KEYS
;
;-------Display Files in Memory Directory
;       Input   AL = starting directory number (1 to MaxFile)
;               AH = hilite bar number (1 to 14)
;       Output  a 14 line of file names to the screen.
;       Note:  local variables:         AH = non hilite color attribute
;              BX = row/col             DH = hilite bar color attribute
;              CX = loop counter        DL = reverse hilite bar number (14 to 1)
;       Note: the hilite bar counter stored in DL is reversed from 1 to 14
;             into 14 to 1 so it can be compaired to the loop counter in
;             CX to select the correct row to hilite.
;
PROC    DISPLAY_FILES
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     DL,15                                   ;convert 1 to 14 into
        SUB     DL,AH                                   ;14 to 1
        MOV     AH,[Menu]                               ;normal attribute
        MOV     [Color],AH                              ;set default color
        MOV     DH,[Warning]                            ;hilite bar attirbute
        MOV     BX,0520h                                ;row 6/ col 32
        MOV     CX,14                                   ;number of rows
DIS0:   CMP     DL,CL                                   ;is this the hilite bar
        JNZ     DIS1                                    ;<> 0 = no color change
        MOV     [Color],DH                              ;if yes color = warning
DIS1:   CALL    DIR_STR                                 ;display one file name
        CMP     DL,CL                                   ;is the the hilite bar
        JNZ     DIS2                                    ;<> 0 = no color change
        MOV     [Color],AH                              ;if yes color = menu
DIS2:   INC     BH                                      ;ptr to next row
        INC     AL                                      ;ptr to next dir entry
        LOOP    DIS0                                    ;loop 14 times
        CALL    HIDE_CUR
        CLC                                             ;clear carry flag
DIS3:   POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DISPLAY_FILES
;
;-------Move Selected file Name from memory block to FileNa in data segment.
```

```
;       Input   BL = starting directory number (1 to MaxFile)
;               BH = hilite bar number (1 to 14)
;       Output  an ASCIIZ file name string in FileNa in the data section.
;
PROC    MOVE_NAME
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,BX                                   ;memblk ptr to AX
        ADD     AL,AH                                   ;start dir no + hilite
        DEC     AL                                      ;minus 1 = memblk recno
        XOR     AH,AH                                   ;convert 16 bit number
        MOV     CL,4                                    ;shift counter
        SHL     AX,CL                                   ;2^4  = times 16
        INC     AX                                      ;skip 2 leading spaces
        INC     AX
        MOV     SI,AX                                   ;memblk offset to SI
        MOV     DI,Offset FileNa                        ;destination offset
        MOV     AX,DS                                   ;assign ES to the
        MOV     ES,AX                                   ;data section
        MOV     AX,[VarSeg]                             ;memblk base ptr
        MOV     DS,AX                                   ;assign memblk to DS
        CLD                                             ;auto inc SI & DI
MOV0:   MOV     AL,[SI]                                 ;get first byte
        CMP     AL,'.'                                  ;is beginning of type?
        JZ      MOV1                                    ;exit loop if yes
        MOVSB                                           ;move byte
        JMP     SHORT MOV0                              ;loop until beg of type
MOV1:   MOV     AX,ES                                   ;restore DS to
        MOV     DS,AX                                   ;point to the data seg
        MOV     CX,5                                    ;number byte to move
        MOV     SI,Offset FilTyp                        ;point to 5 byte string
        REP     MOVSB                                   ;move type to FileNa
        CLC                                             ;clear carry flag
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    MOVE_NAME
;
;-------Display the Select a file screen.
;       Input = None
;       Output = Carry Flag if no FIL files in current directory
;
PROC    SELECT_SCREEN
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        XOR     AX,AX                                   ;row 0,column 0
        CALL    MENU_BOX                                ;draw menu box
        MOV     AX,010Bh                                ;row 1,column 4
        CALL    GOTOYX
        MOV     CL,[Color]                              ;save current color attr
        MOV     AL,[Menu]                               ;set color = menu
        MOV     [Color],AL
        CALL    CSTR_OUT
        db      'Use the ',24,' and ',25,' arrow keys to highlight the '
        db      'desired data file.',0
        MOV     AX,020Fh                                ;row 2,column 7
```

```
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'Press the <Enter> key to select the highlighted file.',0
        MOV     AL,[Normal]                             ;set Color
        MOV     [Color],AL
        CALL    CLEAR_TITLE
        MOV     AX,0405h
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'Directory Path :  ',0
        MOV     AX,Offset Path
        CALL    DSTR_OUT
        MOV     AX,1500h                                ;row 21,column 0
        CALL    MENU_BOX                                ;draw menu box
        MOV     AL,[Menu]                               ;set color = menu
        MOV     [Color],AL
        MOV     AX,1606h                                ;row 23,column 5
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'Press the <Esc> key to return to the menu without '
        db      'selecting a file.',0
        MOV     AX,1702h                                ;row 23,column 5
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      'The current directory contains: ',0
        XOR     AX,AX                                   ;zero AX register
        MOV     AL,[Maxfile]                            ;load number FIL files
        CMP     AL,1                                    ;is it only one ?
        JZ      CEE1                                    ;is yes singular text
        CMP     AL,0                                    ;is it zero ?
        JZ      CEE2                                    ;if yes display error
        CALL    BIN_OUT                                 ;else display number
        CALL    CSTR_OUT                                ;of files.
        db      " files with a type of '",0
        JMP     CEE4
CEE1:   CALL    CSTR_OUT                                ;singular text mess.
        db      "1 file with a type of '",0
        JMP     CEE4
CEE2:   CALL    CSTR_OUT                                ;zero files statement
        db      "no files with a type of '",0
        MOV     AX,Offset FilTyp
        CALL    DSTR_OUT
        CALL    CSTR_OUT
        db      "'.",0
        MOV     AL,[Warning]                            ;warning color
        MOV     [Color],AL                              ;set color
        MOV     AX,0506h                                ;row 5 Col 7
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      ' No ',0
        MOV     AX,Offset FilTyp
        CALL    DSTR_OUT
        CALL    CSTR_OUT
        db      ' files found in directory!  Press Any Key for '
        db      'previous Menu. ',0
        CALL    HIDE_CUR
        CALL    GET_CHAR                                ;wait for keyboard key
        MOV     [Color],CL                              ;restore original Color
        STC                                             ;set carry flag
        JMP     SHORT CEE5                              ;exit no files found
;-------draw background boxes and key discriptions
CEE4:   MOV     AX,Offset FilTyp
```

```
        CALL    DSTR_OUT
        CALL    CSTR_OUT
        db      "'.",0
        CALL    SELECT_WINDOW
CKE5:   POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
;--- draw file diplay windows and key instructions
;       Input = None
;       Output = None
PROC    SELECT_WINDOW
        MOV     AL,[Normal]                             ;set Color
        MOV     [Color],AL
        MOV     AX,0804h                                ;row 8 column 4
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      "<Up Arrow> = Move Up                           "
        db      "<Down Arrow> = Move Down",0
        MOV     AX,0A04h                                ;row 10,column 4
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      "<PageUp> = Scroll Up                           "
        db      "<Home> = First File.",0
        MOV     AX,0C04h                                ;row 12,column 4
        CALL    GOTOYX
        CALL    CSTR_OUT
        db      "<PageDn> = Scroll Down                         "
        db      "<End> =  Last File.",0
;------draw display windows
        MOV     AL,[System]
        MOV     [Color],AL
        MOV     AX,0621h
        MOV     BX,1332h
        CALL    CLEAR_WINDOW
        MOV     AL,[Menu]
        MOV     [Color],AL
        MOV     AX,051Fh
        MOV     BX,1230h
        CALL    CLEAR_WINDOW
        CALL    HIDE_CUR
        MOV     [Color],CL                              ;restore original Color
        CLC                                             ;clear carry flag
        RET
ENDP    SELECT_WINDOW
ENDP    SELECT_SCREEN
ENDP    SELECT_FILE
;


;
;  Create a directory of the FeedBack files in memory.
;       Input = None
;       Output = Carry flag if DOS error,  AL = FFh is to many files
;       [VarSeg] = Starting segment address of memory block.
;       [MaxFile] = total number of FeedBack files.
;
PROC    CREATE_MEM_DIR
        PUSH    BX
        PUSH    CX
        PUSH    DX
```

```
        PUSH    ES
        CALL    RELEASE_MEM_DIR
        JNC     CRE0                            ;continue if no error
        JMP     CRE9                            ;exit on DOS error
CRE0:   CALL    COUNT_FILES                     ;how many SLDI files?
        CMP     AX,251                          ;is files found < 251 ?
        JNC     CRE1                            ;if no display err mess
        MOV     [MaxFile],AL                    ;save number of files
        JMP     CRE2                            ;if yes OK! continue
CRE1:   MOV     AX,030Bh                        ;row 3 / col 1
        CALL    GOTOYX                          ;position cursor
        CALL    CSTR_OUT                        ;string to screen
        db      'There are to more than 250 SLD files in this directory.',0
        MOV     AX,0511h                        ;row 3 / col 1
        CALL    GOTOYX                          ;position cursor
        CALL    CSTR_OUT                        ;string to screen
        db      'Please move some of them to another directory.',0
        MOV     AX,071Ah                        ;row 3 / col 1
        CALL    GOTOYX                          ;position cursor
        CALL    CSTR_OUT                        ;string to screen
        db      'Press Any Key to Exit to DOS.',0
        CALL    HIDE_CUR                        ;hide cursor
        CALL    GET_CHAR                        ;wait for keypressed
        MOV     AL,0FFh                         ;to many files marker
        JMP     CRE8                            ;exit to many files
CRE2:   CMP     AL,0                            ;were any files found ?
        JZ      CRE8                            ;if no files Exit
CRE3:   CALL    GET_DIR_BLK                     ;allocate memory blk
        JC      CRE9
        CALL    MAKE_DIR
        JC      CRE9
        CALL    SHELL_SORT
CRE8:   CLC                                     ;clear carry flag
CRE9:   POP     ES
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        RET
ENDP    CREATE_MEM_DIR
;
;
;   Make a directory of FeedBack files in memory block [VarSeg]
;   Each entry is 16 bytes.  Format: 2 spaces + File Name + padding spaces = 16
;       Input = [VarSeg] and [Search] in data section
;       Output = None
PROC    MAKE_DIR
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    DS
        PUSH    ES
        MOV     AX,DS
        MOV     ES,AX
        CMP     WORD PTR[VarSeg],0              ;is memblk allocated ?
        STC                                     ;set carry if error
        JZ      COP4                            ;if no memblk EXIT
;-------find first match
        XOR     BX,BX                           ;zero file counter
        MOV     AX,4E00h                        ;find a first file
        XOR     CX,CX                           ;ordinary files only
        MOV     DX,Offset Search                ;ptr file name ASCIIZ
```

```
        INT     21h                                     ;do the first search
        JC      COP4                                    ;if no match exit
;-------set up ES and DS segment registers
        MOV     AH,62h                                  ;get the current PSP
        INT     21h                                     ;segment address.
        JC      COP4                                    ;exit on error
        MOV     AX,[VarSeg]                             ;ptr to base of memblk
        MOV     ES,AX                                   ;ES set to memory blk
        MOV     DS,AX                                   ;DS set to memory blk
;-------set directory entry 0 = a blank ASCII string (16 spaces)
        MOV     AX,2020h                                ;two spaces in ASCII
        MOV     DI,2                                    ;distination ptr
        MOV     SI,0                                    ;source ptr
        MOV     CX,7                                    ;loop counter
        MOV     [SI],AX                                 ;place 1st 2 bytes
        CLD                                             ;auto inc SI & DI
        REP     MOVSW                                   ;place next 14 bytes
;-------copy directory entries loop
        MOV     DS,BX                                   ;DS set to PSP
;-------place leading 2 spaces
COP0:   MOV     AX,2020h                                ;two ASCII spaces
        MOV     [ES:DI],AX                              ;place in directory
        INC     DI                                      ;advance directory ptr
        INC     DI
;-------move one file name                              ;80h = DTA; 1Eh = offset
        MOV     SI,9Eh                                  ;DTA + offset = 9Eh
      MOV     CX,12                                   ;max length of Name
        MOV     CX,12                                   ;max file name length
COP1:   MOV     AL,[SI]                                 ;load byte to be moved
        CMP     AL,0                                    ;is it end of string ?
        JZ      COP2                                    ;if end exit loop
        CLD                                             ;auto inc SI & DI
        MOVSB
        LOOP    COP1                                    ;copy file name
;-------pad end of file name with spaces.
COP2:   ADD     CX,2                                    ;number of bytes
        MOV     AL,20h                                  ;space to AL
COP3:   MOV     [ES:DI],AL                              ;place space in dir
        INC     DI                                      ;ptr to next byte
        LOOP    COP3                                    ;loop until CX = 0
;-------find next match
        MOV     AX,4F00h                                ;fine next file function
        INT     21h                                     ;do next search
        JNC     COP0                                    ;loop until all found
        CLC                                             ;clear carry flag
COP4:   POP     ES
        POP     DS
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    MAKE_DIR
;
;------ Allocate memory block for the Director of files ([MaxFile]+2 paragraphs)
;       Input =  None
;       Output = Carry flag set if memory block is not available.
;                Index file seg address stored in [VarSeg]
;       Note:    The binary SEARCH procedure needs a blank record before
;                the memory index records.  The number of pargraphs
;                needed is [MaxFile] + 1.
;
```

```
PROC    GET_DIR_BLK
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
MEM1:   MOV     BL,[MaxFile]                    ;get number of files
        XOR     BH,BH                           ;zero high byte
        INC     BX                              ;get an extra paragraph
        MOV     AH,48h                          ;allocate mem function
        INT     21h                             ;request memory block
        JC      MEM2                            ;jump if memory error.
        MOV     [VarSeg],AX                     ;base address of seg
        JMP     SHORT MEM3                      ;normal exit of proc.
MEM2:   MOV     CL,[Color]                      ;save original color
        MOV     AL,[Warning]                    ;warning color
        MOV     [Color],AL                      ;set color
        MOV     AX,0101h                        ;row 1/Col 1
        CALL    GOTOYX                          ;position cursor
        CALL    CSTR_OUT                        ;send string to screen
        db      ' Not enough memory for the directory of files.  '
        db      'Press Any Key to Continue. ', 0
        MOV     [Color],CL                      ;restore original color
        CALL    HIDE_CUR                        ;hide cursor off screen
        CALL    GET_CHAR                        ;wait for key is pressed
        STC                                     ;set carry flag = error
MEM3:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    GET_DIR_BLK
;
;
;------ Adjust the DOS memory block size allocation to the minimum amount.
;       Input =  None
;       Output = Carry flag set if memory block error.
;       Note:    Assumes the programs memory is in a single block
;                and the stack segment is at the end of the program.
;
PROC    RELEASE_MEM
        PUSH    AX                              ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        MOV     AX,STACKSIZE                    ;current stack size
        MOV     CL,4                            ;convert to paragraphs
        SHR     AX,CL                           ;divide by 2^4 or 16
        INC     AX                              ;round up 2 paragraphs
        INC     AX                              ;to protect top of stack
        MOV     CX,AX                           ;save in register CX
        MOV     AX,SS                           ;get stack seg address
        ADD     CX,AX                           ;ptr to end of stack
        MOV     AH,62h                          ;get the current PSP
        INT     21h                             ;segment address.
        JC      REL0                            ;exit on error
        MOV     ES,BX                           ;ptr to current PSP
        SUB     CX,BX                           ;program size in
        MOV     BX,CX                           ;paragraphs to BX.
        MOV     AH,4Ah                          ;release mem function
        INT     21h                             ;release previous block.
REL0:   POP     ES                              ;restore registers
```

```
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    RELEASE_MEM
;
;   Count the number of FeedBack data files in the currect directory
;       Input = None
;       Output = AX = total number of FeedBack files found.
;       assumed any file ending with a .SLD ext is a FeedBack data file.
;       When the file is opened the data will be validated.
;
PROC    COUNT_FILES
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,DS
        MOV     ES,AX
;------copy Path to [Search]
        MOV     SI,Offset Path                          ;source offsmt
        MOV     DI,Offset Search                        ;destination offset
        CLD                                             ;auto inc DI and SI
COU1:   MOVSB                                           ;copy one byte
        CMP     BYTE PTR [SI],0                         ;is next char = 0
        JNZ     COU1                                    ;copy bytes
;------place "\" after path name
        MOV     AL,'\'
        MOV     [DI],AL
        INC     DI
;------copy search name to end of path
        MOV     SI,Offset SearNa                        ;source ptr
        MOV     CX,13                                   ;number of bytes
        CLD                                             ;auto inc SI & DI
        REP     MOVSB                                   ;copy all 13 bytes
;------search for key files
        XOR     BX,BX                                   ;zero file counter
        MOV     AX,4E00h                                ;find a first file
        XOR     CX,CX                                   ;ordinary files only
        MOV     DX,Offset Search                        ;ptr file name ASCIIZ
        INT     21h                                     ;do the first search
        JC      COU6                                    ;if no match exit
COU5:   INC     BX                                      ;found:increase counter
        MOV     AX,4F00h                                ;fine next file function
        INT     21h                                     ;do next search
        JNC     COU5                                    ;loop until not found
COU6:   MOV     AX,BX                                   ;file count to AX
        CLC                                             ;clear carry flag
        POP     DX                                      ;restore registers
        POP     CX
        POP     BX
        RET
ENDP    COUNT_FILES
;
;-------Fill the name field with 13 spaces in the data section.
;       Input = AX = pointer to field
;       Output = None
;
PROC    CLEAR_FIELD
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
```

```
        PUSH    DX
        MOV     BX,AX                           ;ptr to field.
        MOV     AX,DS                           ;Make ES = DS
        MOV     ES,AX
        MOV     CX,12                           ;restore length of str
        MOV     AL,' '                          ;place a space in first
        MOV     [BX],AL                         ;byte of [Input] string.
        MOV     DI,BX                           ;DI = pointer to next
        INC     DI                              ;byte of string
        MOV     SI,BX                           ;SI = pointer to str
        CLD                                     ;auto inc DI and SI
        REP     MOVSB                           ;fill str with spaces
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    CLEAR_FIELD
;
;
;
;-------Sort the Memory Index Records.
;
;       Input = expects the 16 byte index records to be located at address
;               pointer [IdxSeg] and the number of record to be [MaxRec]
;       Output = None
;       Note: this routine reassigns the DS and ES registers to point to the
;             Index File in memory.   Record 0 is not sorted.  The sort is
;             from record 1 to MaxRec.  A blank record in record 0 is needed
;             for an ASCII string when performing a binary search.
;             The memory index record length is 16 bytes.
;             The sort is based on the first 10 bytes.
;
;       This sort is based on the following TPASCAL procedure:
;       PROCEDURE Sort;             {A Shell Sort}
;       VAR
;          Gap,J : Integer;
;          Temp : string[13];
;          TempNo : Integer;
;       Begin
;          Gap := MaxRec Div 2;
;          While gap > 0  Do
;          Begin
;             For I := (Gap + 1) to MaxRec Do
;             Begin
;                J := I-Gap;
;                While J > 0 Do
;                Begin
;                   If A[J] > A[J+Gap] then
;                   Begin
;                      Temp := A[J];
;                      A[J] := A[J+Gap];
;                      A[J+Gap] := Temp;
;                      J := J-Gap;
;                   End
;                      Else J := 0;
;                   End;
;             End;
;             Gap :=Gap DIV 2;
;          End;
;       End;
;       The follow registers hold the above variables:
```

```
;       AX = Gap;  BX = J;  CX = I;  DX = MaxRec; and  BP = temp storage
;
PROC    SHELL_SORT
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    DS
        PUSH    ES
        PUSH    BP
        MOV     DL,[MaxFile]                            ;store MaxRec in DX
        XOR     DH,DH                                   ;zero high byte
        MOV     AX,[VarSeg]                             ;get Index base segment
        MOV     DS,AX                                   ;reassign the  DS & ES
        MOV     ES,AX                                   ;to ptr to the Index.
        MOV     AX,DX                                   ;Gap = MaxRec
        SHR     AX,1                                    ;Gap = Gap Div by 2
SHEL1:  CMP     AX,0                                    ;when Gap = 0 exit.
        JLE     SHEL4                                   ;exit if <= 0
        MOV     CX,AX                                   ;I is stored in CX
        INC     CX                                      ;I = Gap + 1
SHEL2:  MOV     BX,CX                                   ;J in BX
        SUB     BX,AX                                   ;J = I - Gap
        JZ      SHEL3                                   ;skip if J = 0
        JC      SHEL3                                   ;skip if J is < 0.
        CALL    COMPARE_SWAP                            ;repeat until J = 0
SHEL3:  INC     CX                                      ;I = I + 1
        CMP     DX,CX                                   ;Is I < or = MaxRec
        JNC     SHEL2                                   ;If yes then loop.
        SHR     AX,1                                    ;Gap = Gap Div by 2
        JMP     SHORT SHEL1
SHEL4:  POP     BP                                      ;restore registers
        POP     ES
        POP     DS
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET                                             ;sort is complete.
;
;-------Compare and swap Index strings if needed.
;       Note: This is a subroutine of SHELL_SORT.  The index file record
;             length is 16 bytes.  The sort is made on the first 6 bytes.
;
;       Input = AX = Gap;  BX = J;   DS & ES point to the base of index file.
;       Output = AX = Gap; CX = I; and DX = MaxRec are returned on changed.
;                BX = J is discarded.
;
PROC    COMPARE_SWAP
        PUSH    AX                                      ;save registers
        PUSH    CX
        PUSH    DX
        MOV     DX,AX                                   ;save Gap in DX
;       Compare the first six bytes of each index record
COMP1:  MOV     BP,BX                                   ;save J in BP
        ADD     AX,BX                                   ;AX = J + Gap
        MOV     CL,4                                    ;shift counter
        SHL     AX,CL                                   ;ptr to J+Gap in mem
        SHL     BX,CL                                   ;ptr to J in mem
        CLD                                             ;auto-inc SI, DI
        MOV     DI,AX                                   ;offset of J + Gap
        MOV     SI,BX                                   ;offset of J
```

```
        MOV     CX,10                                   ;byte counter
        REPE    CMPSB                                   ;compare strings
        JLE     COMP3                                   ;exit if < or =.
;       Swap the 16 bytes of index record if string A > string A+Gap
        MOV     DI,AX                                   ;offset of J + Gap
        MOV     SI,BX                                   ;offset of J
        MOV     CX,8                                    ;word counter
COMP2:  MOV     AX,[SI]                                 ;read word each str.
        MOV     BX,[DI]
        MOV     [SI],BX                                 ;write word each str.
        MOV     [DI],AX
        INC     DI                                      ;point to next word
        INC     DI
        INC     SI                                      ;point to next word
        INC     SI
        LOOP    COMP2                                   ;loop five times
        MOV     AX,DX                                   ;restore gap to AX
        MOV     BX,BP                                   ;restore J to BX
        SUB     BX,AX                                   ;J = J - gap
        JZ      COMP3                                   ;exit if J = 0.
        JNC     COMP1                                   ;continue if J > 0.
COMP3:  POP     DX                                      ;restore registers
        POP     CX
        POP     AX
        RET                                             ;return to Shell_Sort
ENDP    COMPARE_SWAP
ENDP    SHELL_SORT
;
;-------Send a 16 byte memory directory entry to the Screen
;       Input =  AL = DirFile number  (0 to Maxfile) 0 = blank directory entry
;                BX = row /col
;                [MaxFile] = the number of directory entries in the memory dir
;                [VarSeg] = segment address of the base of the memory directory
;       Output = ASCIIZ string sent to the screen
;
PROC DIR_STR
        PUSH    AX                                      ;save registers
        PUSH    BX
        PUSH    CX
        PUSH    DX
;-------compute dirfile offset
        XOR     AH,AH                                   ;AX = DirFile number
        CMP     [MaxFile],AL                            ;is DirFile # OK ?
        JNC     DIR0                                    ;if in bounds jump
        MOV     AL,AH                                   ;else make blank file
DIR0:   MOV     CL,4                                    ;shift 4 = times 16
        SHL     AX,CL                                   ;multi by 16
        MOV     SI,AX                                   ;ASCIIZ message ptr SI
        MOV     AX,BX                                   ;row/col to AX
        CALL    GOTOYX                                  ;position cursor
        MOV     DI,Offset Input                         ;ptr to input string
        MOV     AX,DS                                   ;place data seg
        MOV     ES,AX                                   ;in the ES register.
        MOV     AX,[VarSeg]                             ;place the memory blk
        MOV     DS,AX                                   ;seg in DS.
        MOV     CX,8                                    ;8 words = 16 bytes
        CLD
        REP     MOVSW
        MOV     AX,ES                                   ;restore reg DS to
        MOV     DS,AX                                   ;point to data segment.
        XOR     AL,AL                                   ;place zero in string
        MOV     [DI],AL                                 ;as EndOfString marker
```

```
        MOV     AX,Offset Input                         ;ptr to input string
        CALL    DSTR_OUT                                ;send name to the screen
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    DIR_STR
;
;       Input = none
;       Output = none
PROC    PRINT_WAIT_MESS
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
;------please wait message to screen.
        CALL    CLEAR_MESSAGE
        MOV     CL,[Color]                              ;save original attri
        MOV     AL,[Warning]                            ;warning color
        MOV     [Color],AL                              ;set color
        MOV     AX,020Bh                                ;row 3/Col 12
        CALL    GOTOYX                                  ;set cursor
        CALL    CSTR_OUT                                ;display warning
        db      ' Please wait ........     Reading file: ',0
        MOV     AX, Offset FileNa
        CALL    DSTR_OUT
        CALL    CSTR_OUT
        db      ' ',0
        MOV     [Color],CL                              ;restore original attri
        CALL    HIDE_CUR
        CLC
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    PRINT_WAIT_MESS
;
;  Release the memory directory and variable blocks.
;       Input = None
;       Output = Carry flag if DOS error
;       [VarSeg] = Starting segment address of directory block.
;       [VarSeg] = starting segment address for variable block.
;       [MaxFile] = total number of FeedBack files.
;
PROC    RELEASE_MEM_DIR
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    ES
        XOR     AX,AX                                   ;zero AX
        CMP     [VarSeg],AX                             ;is VarSeg assigned?
        JZ      REL2                                    ;if not assigned go on
;------release assigned memory block
        MOV     AX,[VarSeg]                             ;get memory segment
        MOV     ES,AX                                   ;place in ES register
        MOV     AX,4900h                                ;release funcdtion no
        INT     21h                                     ;release memory block
        JC      REL2                                    ;if No error continue
;------initialize  variables
        MOV     AX,0101h                                ;set barposition to
```

```
        MOV     [BarPos],AX                     ;start = 1 hilight = 1
        XOR     AX,AX                           ;zero to register
        MOV     [VarSeg],AX                     ;set memory bock to 0
        MOV     [MaxFile],AL                    ;set maxfiles to 0
        CLC                                     ;clear carry flag
REL2:   POP     ES
        POP     DX                              ;restore registers
        POP     CX
        POP     BX
        RET
ENDP    RELEASE_MEM_DIR
;
;
;-------Read the files DOS date to the [Date] string
;       Input = None
;       Output =  files date to [Date]
;
PROC    READ_DATE
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     BX,[CFGHd]                      ;load file handle
        CMP     BX,0                            ;is a file open
        JZ      DOS2                            ;if not exit
        MOV     AX,5700h                        ;get date stamp funct.
        INT     21h                             ;get stamp
        JC      DOS2                            ;if DOS error Exit
        MOV     BX,DX                           ;composite to get day
        AND     BX,01Fh                         ;isolate day
        MOV     CL,5                            ;shift counter
        SHR     DX,CL                           ;month to bits 0 to 3
        MOV     AX,DX                           ;composite to get month
        AND     AX,0Fh                          ;isolate month
        MOV     CL,4                            ;shift counter
        SHR     DX,CL                           ;year to bits 0 to 5
        AND     DX,03Fh                         ;isolate year
        ADD     DX,80                           ;add base year
        MOV     CX,BX                           ;store day in CX
;------convert to ASCII                        ;AX=Mon,CX=day,DX=year
        MOV     BX,Offset Date                  ;ptr to Date string
        CALL    CONVERT_ASCII                   ;place month in string
        MOV     AX,CX                           ;day of month to AX
        MOV     BX,Offset Date + 3              ;ptr to day section
        CALL    CONVERT_ASCII                   ;place day in Date str
        MOV     AX,DX                           ;place year in AX
        MOV     BX,Offset Date + 6              ;ptr to year section
        CALL    CONVERT_ASCII                   ;place year in Date str
DOS2:   CLC                                     ;clear carry flag
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
;
;-------Convert hex number into 2 digit ASCII number.
;       Input =  AX = hex number
;                BX = ptr in [Date]
;       Output =  two byte number into [Date] string
;
PROC    CONVERT_ASCII
        PUSH    AX
```

```
        PUSH    BX
        PUSH    CX
        PUSH    DX
        CMP     AX,100                          ;is it a 2 digit number?
        JC      COV1                            ;if yes continue else
        XOR     AX,AX                           ;set number to 00
COV1:   MOV     CL,10                           ;divisor
        DIV     CL                              ;AX/10
        OR      AX,3030h                        ;convert to ASCII
COV2:   MOV     [BX],AX                         ;place in Date string
        CLC                                     ;clear carry flag
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    CONVERT_ASCII
ENDP    READ_DATE
;

;-------Set search file name variables for file type .NEG or .POS
;       Input =  AX = none
;       Output =  Adjust the following strings [FilTyp], [FileNa] and [SearNa]
;
PROC    SET_TYPE
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AX,DS                           ;set ES = DS
        MOV     ES,AX
        MOV     BX,Offset CFGtyp                ;ptr to CFG string
STT1:   MOV     DX,2                            ;save byte counter
        CLD                                     ;auto inc SI & DI
        MOV     DI,Offset FilTyp                ;destination pointer
        MOV     SI,BX                           ;source pointer
        MOV     CX,DX                           ;loop counter = 2
        REP     MOVSW                           ;move two Words
                                ;clear carry flag
        MOV     DI,Offset FileNa + 8            ;destination pointer
        MOV     SI,BX                           ;source pointer
        MOV     CX,DX                           ;loop counter = 2
        REP     MOVSW                           ;move two Words
                                ;clear carry flag
        MOV     DI,Offset SearNa + 8            ;destination pointer
        MOV     SI,BX                           ;source pointer
        MOV     CX,DX                           ;loop counter = 2
        REP     MOVSW                           ;move two Words
                                ;clear carry flag
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ENDP    SET_TYPE
;
;

;
        .DATA                                   ;the data segment.
ErrCode db      0                               ;ret error msg to DOS
;note: if Debug is ON the printing time will be twice as long.
```

```
Debug   db      0                                       ;0 = OFF   Other = ON
Tint    db      0                                       ;0 = dark  Other = light
Others  db      0                                       ;0 = FALSE Other = TRUE
;
;Video data
Vidmode db      0                                       ;video mode
vidpage db      0                                       ;video page
vidcurs dw      0                                       ;cursor type
vidfont dw      0                                       ;font size
vidattr db      07h                                     ;default Lt White/Black
vidbord db      07h                                     ;border color
;
;Color variables
Color   db      07h                                     ;active color
System  db      07h                                     ;default Lt White/Black
Menu    db      0                                       ;Menu main color
Normal  db      07h                                     ;Main display screen
HiLite  db      0                                       ;display screen titles
MenuMes db      0                                       ;menu messages line
Warning db      0                                       ;accent color
Border  db      0                                       ;display screen box
;
;Memory Block variables
VarSeg  dw      0                                       ;seg of var mem block
TopBar  dw      1                                       ;used to display ids
HiBar   db     15                                       ;which id to highlite
MaxFile db      0                                       ;number of files 0-250
MaxDim  db      0                                       ;number of dimensions
BarPos  dw      0101h                                   ;position of hilite bar
;
;file types
CFGTyp  db      '.CFG',0                                ;config file type
DATTyp  db      '.DAT',0                                ;data file type
RNKTyp  db      '.RNK',0                                ;rank file type
NDXTyp  db      '.NDX',0                                ;index file type
SF1Typ  db      '.SF1',0                                ;self part 1 answers
SF2Typ  db      '.SF2',0                                ;self part 2 answers
PERTyp  db      '.PER',0                                ;peer answers
SUPTyp  db      '.SUP',0                                ;superior answers
SUBTyp  db      '.SUB',0                                ;subordinate answers
FilTyp  db      '.???',0                                ;File type holder
;file names
FileNa  db      '????????.SLD',0                        ;ASCIIZ file name
SearNa  db      '????????.SLD',0                        ;ASCIIZ file name
;
;file handles
CFGHd   dw      0                                       ;config file handle
DATHd   dw      0                                       ;data file handle
RNKHd   dw      0                                       ;rank file handle
NDXHd   dw      0                                       ;index file handle
;note do not change the order of the following five answer file handles
;Procedure: MAKE_DATA_FILE in FBF.ASM uses these handles as a lookup file
SUBHd   dw      0                                       ;subordinates handle
SUPHd   dw      0                                       ;superiors file handle
PERHd   dw      0                                       ;peer file handle
SF2Hd   dw      0                                       ;self part 2 handle
SF1Hd   dw      0                                       ;self part 1 handle

;file variables
FileDr  db      0                                       ;0 = default, 1 = A etc
;
EOF     db      0                                       ;0 = FALSE Other = TRUE
```

```
SF1EOF  db      0                                       ;0 = FALSE Other = TRUE
Time    dw      0                                       ;store clock tick count
MaxID   dw      0                                       ;num of ID's in DAT file
DRecLn  dw      0                                       ;DAT record length
NDXLn   dw      0                                       ;expected size NDX file
RNKLn   dw      0                                       ;expected size RNK file

;table used to point to configuration data strings stored on the heap
;each entry is 4 bytes.  The first word identifies the type of information and
;the next word is a pointer to its location on the heap.  All the data in the
;heap is stored in asciiz format.  The question numbers are stored as byte
;integer strings ending with a hex 0.   Type identification are:
;8 = end of table, 7 = group title, 6 = factor title
;5 = SF1 hex question numbers, 4 = SF2 hex question numbers,
;3 = PER hex question numbers, 2 = SUP hex question numbers,
;and 1 = SUB hex question numbers
CFGTbl  dw      320 DUP (0h),0h                         ;

;Order of values in the Rank file: bytes: Lowest, Highest, Median, 25%, 75% and
;word: offset in buffer of the mean score. (add offset to base to locate mean)
;
;buffers used for file reads and writes:
Filbuf  db      256 DUP (0h),0h                         ;buff for reading files
OutBuf  db      256 DUP (0h),0h                         ;buff for writing files
SF1Buf  db      256 DUP (0h),0h                         ;buff for Self I data
SF2Buf  db      256 DUP (0h),0h                         ;buff for Self II data
PERBuf  db     2048 DUP (0h),0h                         ;buffer for 8 Peer lines
SUPBuf  db     2048 DUP (0h),0h                         ;buf 8 Superior lines
SUBBuf  db     2048 DUP (0h),0h                         ;buf 8 Subordinate lines

;Printer port (the program expects an HP Laser Jet assigned to a parallel port)
LPT     dw      0                                       ;default = LPT1
;0 = LPT1, 1 = LPT2, & 3 = LPT3
;
;Path Editor variables
Path    db      82 DUP (0h)
Input   db      82 DUP (0h)                             ;input ASCIIZ string.
Search  db      82 DUP (0h)
Digit   db      1                                       ;0 = OFF  Other = ON
Insert  db      0                                       ;0 = OFF  Other = ON
EndFld  db      0                                       ;0 = OFF  Other = ON
;
;Sound string
Beep     dw      6000,2,4500,2,0
        .CODE                                           ;the code segment
MAIN:
;------Determine Color and Graphics Mode
        MOV     AX,@data                                ;get data segment
        MOV     DS,AX                                   ;put in data segment reg
        CALL    COLOR_MODE                              ;define default colors
        CALL    TEXT_VIDEO                              ;save default settings
;-------Main procedure for FeedBack
        CALL    INTERRUPT_HANDLER                       ;INT23 & INT24 handlers
        CALL    RELEASE_MEM                             ;release unused memory
        JC      Error                                   ;display Dos error
        CALL    MAIN_MENU                               ;Program's MAIN LOOP
        JC      Error                                   ;display Dos error
        CALL    CLOSE_ALL_FILES                         ;close any open files
        JC      Error                                   ;display DOS error.
;-------Exit to DOS                                    ;program always ends
Exit:   CALL    RESTORE_VIDEO                           ;restore users settings
        MOV     AL,[ErrCode]                            ;load errorlevel number
```

```
        MOV     AH,4Ch                          ;Exit function number
        INT     21h                             ;return to DOS
;-------End of Main procedure for FeedBack
Error:  CALL    DISPLAY_ERROR                   ;show DOS extend error
        JMP     SHORT Exit

;-------End of the source code
        END     MAIN
```