```
************************************************************
*                                                          *
*            File and Aux Typer - An AppleWorks Init       *
*               *******************************            *
*           - Expands the 'Change file type'               *
*              selection in the File Activities            *
*              menu of AppleWorks 5.1 by:                  *
*                (i) adding several menu-selectable        *
*                     popular file types from which        *
*                     to chose                             *
*               (ii) displaying both the current file      *
*                     type and the aux type of the         *
*                     selected file                        *
*              (iii) displaying both the 3-character       *
*                     file type (e.g. AWP) and its         *
*                     hexadecimal representation           *
*                     (e.g. $1A)                           *
*               (iv) allowing entry of a file's            *
*                     type and aux type in hexadecimal     *
*                     (e.g. $FC / $8001)                   *
*                                                          *
*           Version 5.1 (for AppleWorks Version 5.1)       *
*                     (c) 2024 by Hugh Hood                *
*                                                          *
************************************************************
*                                                          *
*     - Seg $2A (Disk/File activities) is patched to       *
*         add the new/changed features to the File         *
*         Activities menu                                  *
*                                                          *
*     - NOTE: Since the length of the new code for $8200+  *
*              exceeds the $400 byte maximum for the       *
*              SEG Patch Manager memory ($BB00-$BEFF),     *
*              the code is first stored in Desktop         *
*              memory when the init is run and its         *
*              pointer is saved within the patch code      *
*              so that it may be retrieved and run at      *
*              the proper address ($8200+).                *
*                                                          *
************************************************************

               TR             ADR             ; truncate bank address

               XC                             ; enable 65C02 code


* Internal AppleWorks Equates
ErrFlag        EQU            $03             ; ProDOS error occur?

FoundCmd       EQU            $86             ;
FoundRtn       EQU            $87             ;
FoundEsc       EQU            $88             ; >0 if <esc> pressed
MReg           EQU            $91             ; holds result of math operations
AArg           EQU            $9A             ; used for ZP pointers
CurrMode       EQU            $CC7            ; top center msg
StackP1        EQU            $CDC            ; 1st routine in stack (top right)
NewStr         EQU            $D85            ; 128 bytes for string storage
OldStr         EQU            $E05            ; 128 bytes for string storage
```

```

* AppleWorks Host Equates
AWVersion      EQU            $1003          ; $33/51 = 5.1 / $28/40 = 4.0  /
                                             ;   $1E/30 = 3.0
AfterReadTest  EQU            $100C          ; JSR here after getting key
                                             ;  (normally $60/RTS)
TOActive       EQU            $10E6          ; $00 = enabled / $01 = disabled <OA-ESC>
DoBell         EQU            $1115          ;
GetVBar        EQU            $1130          ; returns selection from vertical menu
MvLeftRtn      EQU            $1148          ; move in memory routine (negative/left)
                                             ; - followed by destination/source/length
MvRightRtn     EQU            $114B          ; move in memory routine (positive/right)
                                             ; - followed by destination/source/length
PopStack       EQU            $114E          ; removes last escape level
PushStack      EQU            $1157          ; add level to escape menu
RestCursor     EQU            $115D          ; restore cursor position
SetUCC         EQU            $1163          ; convert accum char a-z to uppercase
StrMvRtn       EQU            $116C          ; follow with to/from
StrWrRtn       EQU            $116F          ; follow with column/row/string address
WriteCom       EQU            $118A          ; writes a string on command line
Write?         EQU            $118D          ; are you sure?

* AppleWorks SEG $2A (File and Disk Activities) Equates



NewCodeAdr     EQU            $8200          ; new code to be placed here
                                             ;  (SEG $2A ends at $8100)

* AppleWorks Subhost Equates
WriteText      EQU            $AB37          ; writes multiple strings
                                             ;  - follow with address of string 'block'
                                             ;  - block contains:   (i) pString for 1
                                             ;                      (ii) column for 1
                                             ;                     (iii) row for 1
                                             ;                      (iv) repeat i/ii/iii
                                             ;                       (v) $00 = endbyte
ManyVBar       EQU            $AB3A          ; routine to build a vertical menu
                                             ;  (repeatedly calls VBarWrRtn)
                                             ; - followed by address of string 'block'
                                             ; - then: Column for #1 (+$80 if work area)
                                             ;         Row for #1 (+$80 if work area)
                                             ;         Spacing (e.g. 1/2/3)
                                             ;         end byte ($00)
                                             ; - block contains:    (i) pString for 1
                                             ;                      (ii) pString for 2
                                             ;                     (iii) pString for n
                                             ;                      (iv) $00 = endbyte
newMultByte    EQU            $AB70          ; Multiply X x Y {result to MReg - $91/$91}



SMGetBlock     EQU            $D005          ; load a block into main memory
                                             ; (pointer in main / 'gets' to main)
                                             ; - follow with:
                                             ;    (i) address in main to block pointer
                                             ;   (ii) address in main to write data
                                             ;  [CArg/ZReg00 $9E/$9F is returned with
```

```
                                                ;     number of bytes retrieved]
SMPutBlock     EQU            $D011             ; puts a block into desktop memory from main
                                                ; - follow with:
                                                ;    (i) address in main to block pointer
                                                ;     [if possible, uses existing pointer;
                                                ;       otherwise new pointer stored at
                                                ;       this address]
                                                ;   (ii) address in main to read data
                                                ;  (iii) length of area to read

DispFE         EQU            $D02C             ; display function and escape map
GetStr         EQU            $D038             ; LDA with maxlength before calling
GetYN          EQU            $D03B             ; menu bar to get a yes or no

* AppleWorks Init Manager Equates
imSavePatch    EQU            $3006             ; Patch Manager save routine in SEG.IM
InitAdr        EQU            $4000             ; load address for Init files
PatchAdr       EQU            $BB00             ; initial load address for patch code
                                                ;  (NOTE: uses ProDOS I/O buffer -
                                                ;          1K max length -
                                                ;           $BB00 - $BEFF)


* Literals

* BaseColumn   EQU            #$06              ; example only - not used here


************************************************************

               ORG            InitAdr           ; ($4000)
               TYP            $06               ; create binary file


********************************
*          Init Header         *
********************************
START
               JMP            IStart            ; skip over header

**----------------------------------------------------------------

               ASC            'mb'              ; Init ID Bytes (AW 5.1)
               DB             $33               ; Init Version - programmer assigned
                                                ;  e.g. - $0A/1.0 $0B/1.1 $33/5.1
               STR            'FileAuxTyper'    ; Init Screen Name

               HEX            0000              ; Init Header End Bytes

**----------------------------------------------------------------

IStart

               LDA            AWVersion         ; AppleWorks version #
               CMP            #$33              ; Is it Version 5.1?
               BNE            Done              ; disregard - wrong version


```

```
StoreStrings   JSR             SMPutBlock     ; store new code in DT memory
               DA              NewCodePtrAdr  ; save pointer here for retrieval ($4xxx)
               DA              Code1End       ; start of new code to store ($4xxx)
               DA              NewCodeEnd-Code1End          ; length of new code to store


PatchH2C       JSR             imSavePatch    ; call patch manager
               DW              Code1          ; beginning of patch1 code ($40xx)
               DW              Code1End-Code1 ; length of patch code
               DW              $002A          ; SEG number to patch
                                              ;  ($2A = File and Disk Activities SEG)

Done           RTS                            ; back to Init Manager

**--------------------------------------------------------------

Code1          EQU             *              ; (will be $40xx)


* Begin $BB00+ run location addressing
               ORG             PatchAdr       ; (Patching Code is moved and run
                                              ;   @ $BB00 by Init Manager)

               HEX             0000           ; Init begin bytes for SEG $2A Patch


               JSR             SMGetBlock     ; retrieve new code from DT memory
               DA              NewCodePA      ; address of pointer stored by SMPutBlock
                                              ;  at running address ($BBxx+)
               DA              NewCodeAdr     ; place New Code at $8200

* Write any new code and tables at end rather than moving existing code and tables

               LDA             #$20           ; JSR
               STA             $6E22          ;
               LDA             #<NewMenu      ;
               STA             $6E23          ;
               LDA             #>NewMenu      ;
               STA             $6E24          ;


               LDA             #$0B           ; new row #
               STA             $6E1F          ;
               LDA             #<Prompt1      ;
               STA             $6E20          ;
               LDA             #>Prompt1      ;
               STA             $6E21          ;


* Modify File activities selection to show 'Change file and aux types'
               LDA             #<Block5       ;
               STA             $6318          ;
               LDA             #>Block5       ;
               STA             $6319

* Modify command line prompt message to show $ *

               LDA             #<CmdStr1      ;
               STA          $6E46          ;
               LDA          #>CmdStr1      ;
               STA          $6E47          ;

* Modify Max Number Used for GetNum ($FF/255) to Max Length used by GetStr
               LDA          #$02
               STA          $6E49

* Modify GetNum to use OurGetStr Instead
               LDA          #<OurGetStr    ;
               STA          $6E4B          ;
               LDA          #>OurGetStr    ;
               STA          $6E4C          ;


* select ProDOS hex file type for menu selection chosen using NewTable1

               LDA          #<NewTable1-1  ;
               STA          $6E32          ;
               STA          $6E60          ;
               LDA          #>NewTable1-1  ;
               STA          $6E33          ;
               STA          $6E61          ;

* increase # of items to check in larger NewTable1

               LDA          #$17           ; 22 items + 1
               STA          $6E66          ; 10 items + 1

* experimental: redirect ESC at enter custom back to selection menu
               LDA          #$4C           ; JMP
               STA          $6E5A          ;
               LDA          #$22           ;
               STA          $6E5B          ;
               LDA          #$6E           ;
               STA          $6E5C          ;

* to replace routine lost when $6E5A was patched to return
*  to NewMenu when <ESC> is pressed in enter Custom File option
               LDA          #$4C           ; JMP
               STA          $6E2B          ;
               LDA          #<MyGetVBar    ;
               STA          $6E2C          ;
               LDA          #>MyGetVBar    ;
               STA          $6E2D          ;

* experimental: change JMP after ESC address from $6D8A to $6D8E to
*                 prevent double count of ESC
*              {Seems to be AppleWorks bug that also effects other options}
               LDA          #$8E           ;
               STA          $6E16          ;

* experimental: require confirmation before changing type
               LDA          #$4C           ; JMP
               STA          $6E39          ; was LDA #$01
               LDA          #<MyConfirmRtn ;
               STA          $6E3A          ;
               LDA          #>MyConfirmRtn ;
               STA          $6E3B        ; was RTS

               RTS                       ;

*------------------------------------------------------------


PatchEnd       EQU          *            ; ($BBxx=+)


*------------------------------------------------------------

               ORG                       ; ($4xxx+)
NewCodePtrAdr                            ; pointer stored here by SMPutBlock above

               ORG          PatchEnd     ; return to $BBxx addressing

NewCodePA      DA           $0000        ; pointer to stored code block ($BBxx+)

**----------------------------------------------------------------

               ORG                       ; ($4xxx+)
Code1End

**----------------------------------------------------------------

               ORG          NewCodeAdr   ; $8200

**----------------------------------------------------------------
NewCode

NewMenu
               STZ          ChangeFlag   ; initialize ChangeFlag


NewMenu1
               JSR          ManyVBar     ;
               DA           Block0       ;
               DB           $10          ; column
               DB           $0D          ; row
               DB           $01          ; spacing
               DB           $00          ; end byte

               JSR          ManyVBar     ;
               DA           Block1       ;
               DB           $10          ; column
               DB           $10          ; row
               DB           $01          ; spacing
               DB           $00          ; end byte

               JSR          ManyVBar     ;
               DA           Block2       ;
               DB           $1F          ; column
               DB           $10          ; row
               DB           $01          ; spacing
               DB           $00          ; end byte

               JSR          ManyVBar     ;
```

```
                DA      Block3          ;
                DB      $2E             ; column
                DB      $10             ; row
                DB      $01             ; spacing
                DB      $00             ; end byte

                JSR     ManyVBar        ;
                DA      Block4          ;
                DB      $3D             ; column
                DB      $10             ; row
                DB      $01             ; spacing
                DB      $00             ; end byte

                LDA     ChangeFlag      ;
                BEQ     :NM2            ; nothing changed yet / 1st time through

                RTS                     ; goes back to $6E25 {one after JSR}


:NM2            JSR     StrWrRtn        ;
                DB      $30             ; column
                DB      $0D             ; row
                DA      Line1and2       ;

                JSR     StrWrRtn        ;
                DB      $30             ; column
                DB      $0E             ; row
                DA      Line1and2       ;


* Read hex bytes in File Info parm table and show in ASCII on screen

                LDA     $807A           ; file type from parm table

                JSR     Hex2Asc         ; convert to ASCII 0-9/A-F
                STX     CrntTypeStr+11 ;
                STA     CrntTypeStr+12 ;

                JSR     HexTo3Char      ; match hex type to 3-character code

                JSR     StrWrRtn        ;
                DB      $34             ; column
                DB      $0D             ; row
                DA      CrntTypeStr     ;

                JSR     StrWrRtn        ;
                DB      $40             ; column
                DB      $0D             ; row
                DA      TypeStr         ;

                LDA     $807C           ; aux type (HB) from parm table
                JSR     Hex2Asc         ; convert to ASCII 0-9/A-F
                STX     CrntAuxStr+11   ;
                STA     CrntAuxStr+12   ;

                LDA     $807B           ; aux type (LB) from parm table
                JSR     Hex2Asc         ; convert to ASCII 0-9/A-F
                STX     CrntAuxStr+13   ;
```

```
                 STA        CrntAuxStr+14  ;

                 JSR        StrWrRtn       ;
                 DB         $34            ; column
                 DB         $0E            ; row
                 DA         CrntAuxStr     ;

***

                 JSR        StrWrRtn       ;
                 DB         $31            ; column
                 DB         $0C            ; row
                 DA         TopLine        ;

                 JSR        StrWrRtn       ;
                 DB         $31            ; column
                 DB         $0F            ; rown
                 DA         BottomLine     ;

                 RTS                       ; goes back to $6E25 {one after JSR}

*                JMP        $6E25          ; back to regular code

Block0           STR        'Enter new hex file type'      ;
                 STR        'Enter new hex aux type'       ;
                 DB         $00            ; end byte

Block1
                 STR        '$19/ADB'      ;
                 STR        '$1A/AWP'      ;
                 STR        '$1B/ASP'      ;
                 STR        '$04/TXT'      ;
                 STR        '$06/BIN'      ;
                 DB         $00            ; end byte

Block2
                 STR        '$C8/FON'      ;
                 STR        '$FC/BAS'      ;
                 STR        '$FF/SYS'      ;
                 STR        '$50/GWP'      ;
                 STR        '$51/GSS'      ;
                 DB         $00            ; end byte

Block3
                 STR        '$52/GDB'      ;
*                STR        'S16/$B3'      ;
*                STR        'NDA/$B8'      ;
                 STR        '$07/FNT'      ;
                 STR        '$08/FOT'      ;
                 STR        '$B0/SRC'      ;
                 STR        '$E0/LBR'      ;
                 DB         $00            ; end byte

Block4
                 STR        '$B9/CDA'      ;
                 STR        '$C1/PIC'      ;
                 STR        '$CA/ICN'      ;
                 STR        '$F9/OS '      ;
```

```
               STR            '$0F/DIR'      ;
               DB             $00            ; end byte

Prompt1        STR            'Enter new file/aux type or select:'

CrntTypeStr    STR            'Current: $  ' ;
CrntAuxStr     STR            'Current: $    '              ;
PendingStr     STR            'Pending'
PendTypeStr    STR            'Pending: $  ' ;
PendAuxStr     STR            'Pending: $    '              ;
TypeStr        STR            '/XXX'         ;

TopLine        STR            '______________________'      ; Underscore for Top Line

BottomLine     STR            "LLLLLLLLLLLLLLLLLLLLLL"      ; MouseText 'L' is Top Bar

Line1and2      DB             :L12-*-1       ; Leading Length Byte
               ASC            "Z"            ; MouseText Right Bar
               ASC            '                    '
               ASC            "_"            ; MouseText Left Bar
:L12

CmdStr1        STR            'Enter new 2-digit hex file type: $'
CmdStr2        STR            'Enter new 4-digit hex aux type: $'

AnotherStr     STR            'Make another change to this file'

ChangeFlag     DB             #$00           ; not zero if change made

FileNameStr    DS             16             ; temp space for stack move

**-------------------------------------------------------------
*  Replacement menu block for $5CE7

Block5
               STR            'Lock files'   ;
               STR            'Unlock files' ;
B53            STR            'Change file and aux types'   ;
               DB             $00            ; end byte

* ChangeTitle  STR            'Change file/aux types'       ; note too long for stack
                                                            ; just use default


**-------------------------------------------------------------
*  New file type table for $6222

NewTable1
               DB             $00            ; enter new file type
               DB             $00            ; enter new aux type
               DB             $19            ; ADB
               DB             $1A            ; AWP
               DB             $1B            ; ASP
               DB             $04            ; TXT
               DB             $06            ; BIN
               DB             $C8            ; FON
               DB             $FC            ; BAS
               DB             $FF            ; SYS
```

```
             DB          $50          ; GWP
             DB          $51          ; GSS
             DB          $52          ; GDB
*            DB          $B3          ; S16
*            DB          $B8          ; NDA
             DB          $07          ; FNT
             DB          $08          ; FOT
             DB          $B0          ; SRC
             DB          $E0          ; LBR
             DB          $B9          ; CDA
             DB          $C1          ; PIC
             DB          $CA          ; ICN
             DB          $F9          ; OS
             DB          $0F          ; DIR

**------------------------------------------------------------

*  Hex2Asc Routine - Input 'A' and Output 'X' (HNib) and 'A' (LNib)

Hex2Asc
             PHA                      ;
             LSR                      ;
             LSR                      ;
             LSR                      ;
             LSR                      ;
             JSR         :A           ;
             TAX                      ;
             PLA                      ;
             AND         #$0F         ;
:A           ORA         #$30         ;
             CMP         #$3A         ;
             BCC         :B           ;
             ADC         #$06         ;
:B           RTS                      ;

**------------------------------------------------------------


* Patch GetStr to just get 0-9 and A-F before calling GetStr
*  and then restore GetStr to default when done

OurGetStr
             STA         StrSize      ; save entry accumulator / 2 or 4
             STZ         FoundEsc     ; clear ESC on 2nd trip through

:PP          INC         TOActive     ; disable TimeOut until we un-patch
                                      ;  after GetStr
             LDA         #$4C         ; JMP
             STA         $B245        ;
             LDA         #<HexFilter  ;
             STA         $B245+1      ;
             LDA         #>HexFilter  ;
             STA         $B245+2      ;

             STZ         NewStr       ; init Newstr

             LDA         StrSize      ; 2 or 4 characters
             JSR         GetStr       ;
```

```

               LDA         #$C9         ; un-patch JMP to HexFilter
               STA         $B245        ;
               LDA         #$20         ;
               STA         $B245+1      ;
               LDA         #$90         ;
               STA         $B245+2      ;

               STZ         TOActive     ; re-enable TimeOut after un-patches

               LDA         FoundEsc     ; if <ESC> key, exit
               BNE         :HH          ; Yes, escaped out of EnterTypeinHex

* Test if both (or 4) digits entered
               LDA         NewStr       ; length of string
               CMP         StrSize      ;
               BNE         :RR          ; does not have both (or 4) Hex Digits

               LDA         FoundCmd     ; if first character is not #, try again
               BEQ         :QQ          ; Cmd not set -- IS a number, proceed

:RR            JSR         RestCursor   ; restore cursor position
               JSR         DoBell       ; ring error bell and return 'A' = 1 / true
               BNE         :PP          ; try again



* now, convert 2-digit ascii hex string in NewStr to hex# and put in accumulator
:QQ            LDA         NewStr+1     ;
               LDY         NewStr+2     ;
               JSR         AscHex2Hex   ; accumulator has result
               STA         HexByte1     ;
               LDA         StrSize      ; for aux type instead of file type?
               CMP         #$04
               BNE         :TT          ; retrive file type byte and rts

               LDA         NewStr+3     ;
               LDY         NewStr+4     ;
               JSR         AscHex2Hex   ; accumlator has result
               STA         HexByte2     ;
               RTS                      ; back to enter custom aux type

:TT            LDA         HexByte1     ;
               RTS                      ; back to enter custom file type

*-------------------------------------------------------------

StrSize        DB          #$02         ; either 2 or 4 characters

HexByte1       DB          #$00         ;
HexByte2       DB          #$00         ;
*-------------------------------------------------------------

:HH
               RTS                      ; back to custom filetype routine
                                        ;  ($6E4D)

**-------------------------------------------------------------
```

```
* Routine to filter all characters except 0-9 and A-F

HexFilter      JSR            SetUCC         ; make a-z uppercase
               CMP            #$30           ; 0
               BCC            :Else          ;
               CMP            #$47           ; G
               BCS            :Else          ;
               CMP            #$3A           ; :
               BCC            :Good          ;
               CMP            #$41           ; A
               BCS            :Good          ;

:Else          JMP            $B2BE          ;

:Good          JMP            $B24D          ;                              ;

**---------------------------------------------------------------

*  AscHex2Hex Routine - Input 'A' (first ASCII Hex digit) and 'Y' (second
*   ASCII Hex digit ) / Output 'A' (Binary/Hex byte)

AscHex2Hex
               PHA                           ; save high digit
               TYA                           ; get low digit
               JSR            :A             ; convert
               STA            Temp           ; save low nibble
               PLA                           ; get high digit
               JSR            :A             ; convert
               ASL                           ; shift high nibble to upper 4 bits
               ASL                           ;
               ASL                           ;
               ASL                           ;
               ORA            Temp           ; 'or' in the low nibble
               RTS

:A             SEC                           ;
               SBC            #$30           ; subtract ASCII '0'
               CMP            #$10           ; greater than 0-9?
               BCC            :B             ; not a letter - go back
               SBC            #$7            ; subtract ASCII offset for letters
:B             RTS                           ;

Temp           DB             #$00           ;


**---------------------------------------------------------------

* to replace routine lost when $6E5A was patched to return
*  to NewMenu when <ESC> is pressed in enter Custom File option
MyGetVBar      JSR            GetVBar        ;
               BEQ            :ZZ            ; goes back to first menu if ESC
               CMP            #$01           ; Selection 1 - Change file type
               BEQ            :YY            ;
               CMP            #$02           ; Selection 2 - Change aux type
               BEQ            :XX
               JMP            $6E30          ; proceeds with changing file type to that
                                             ;  selection from menu choices
```

```
:ZZ            LDA          #$00           ; ESC pressed;
               RTS                         ; go back next file and/or
                                           ;   to File Activities menu

:YY            JMP          $6E3C          ; Enter Custom File Type Routine

* Change aux type routine - duplicate (with changes) routine @ $6E3C
:XX            JSR          StrMvRtn       ; preserve OldStr
                                           ;  (contains pathname of selected file)
               DA           $BB00          ; temp storage area
               DA           OldStr         ; $0E05
               JSR          WriteCom       ;
               DA           CmdStr2        ; enter new 4-digit hex aux type: $
               LDA          #$04           ; max # of hex digits allowed
               JSR          OurGetStr

               JSR          StrMvRtn       ; restore OldStr
                                           ;  (contains pathname of selected file)
               DA           OldStr         ; restore OldStr ($E05) [file pathname]
               DA           $BB00          ;

* check for <esc> pressed
               LDX          FoundEsc       ;
               BNE          :XXZ           ;

* no <esc> - take two aux bytes and put in parm table
               LDA          HexByte2       ;
               STA          $807B          ; SetFileInfo Parms Aux Type (LB)
               LDA          HexByte1       ;
               STA          $807C          ; SetFileInfo Parms Aux Type (HB)

               JMP          $6E39          ; LDA #$01 (success) and RTS to next file
                                           ;  and/or to File Activities menu

:XXZ           LDA          #$00           ; Escape pressed in entry; try again
               JMP          $6E22          ; re-build menu and try again

**------------------------------------------------------------

* Read hex bytes in File Info parm table and show in ASCII on screen and confirm

MyConfirmRtn   LDA          $807A          ; file type from parm table
               JSR          Hex2Asc        ; convert to ASCII 0-9/A-F
               STX          PendTypeStr+11 ;
               STA          PendTypeStr+12 ;

               JSR          HexTo3Char     ; match hex type to 3-character code

               JSR          StrWrRtn       ;
               DB           $34            ; column
               DB           $0D            ; row
               DA           PendTypeStr    ;

               JSR          StrWrRtn       ;
               DB           $40            ; column
               DB           $0D            ; row
               DA           TypeStr        ;

               LDA          $807C        ; aux type (HB) from parm table
               JSR          Hex2Asc      ; convert to ASCII 0-9/A-F
               STX          PendAuxStr+11 ;
               STA          PendAuxStr+12 ;

               LDA          $807B        ; aux type (LB) from parm table
               JSR          Hex2Asc      ; convert to ASCII 0-9/A-F
               STX          PendAuxStr+13 ;
               STA          PendAuxStr+14 ;

               JSR          StrWrRtn     ;
               DB           $34          ; column
               DB           $0E          ; row
               DA           PendAuxStr   ;

               JSR          WriteCom     ;
               DA           AnotherStr   ; Make another change to this file?
               JSR          GetYN        ;
               CMP          #$59         ;
               BNE          :ABC         ;
               INC          ChangeFlag   ;
               JSR          NewMenu1     ; Try Again with NewMenu, but
                                         ; don't initialize ChangeFlag

               JMP          $6E25        ; skip past JSR to NewMenu


:ABC           JSR          Write?       ;
               CMP          #$59         ;
               BEQ          :ABA         ;


               LDA          #$00         ; DISCARD change
               BRA          :ABB         ;

:ABA           LDA          #$01         ; MAKE change

:ABB           RTS                       ; goes back to $6DEE to write/discard change

**-------------------------------------------------------------

* Lookup 3-Character File Type Code given Hex Type
HexTo3Char
               LDA          $807A        ; File Type from Parm Table
               LDX          #$62         ; 98 types assigned 3-char codes

:Loop3Char     CMP          HexTable,X   ; Look for match starting at $FF and
                                         ;   working backwards down to $00
               BEQ          :CalcIndex   ; Yes, HAS code - go look up

               BCS          :No3Code     ; Hex is greater than next lower type
                                         ;  - no 3 character code
               DEX                       ; decrement and check next lower type
               BNE          :Loop3Char   ; try next

:CalcIndex     LDY          #$03         ; each code is 3 characters
               JSR          newMultByte  ; 'X' index times '3'
                                         ; - result in MReg / $91/$92
```

```
                CLC                             ; prepare to add to start of 3CharTable
                LDA             #<3CharTable-1  ; (LB)
                ADC             MReg            ; (LB) / $91
                TAX                             ;
                LDA             #>3CharTable    ; (HB)
                ADC             MReg+1          ; (HB) / $92
                BRA             :CCC            ; will always branch

* If no 3-character File Type Code, just display '$' plus Hex Bytes
:No3Code        JSR             Hex2Asc         ; enter with File Type in 'A'
                STX             TypeStr+3       ; high nibble
                STA             TypeStr+4       ; low nibble
                LDA             #'$'            ; Hex symbol
                STA             TypeStr+2       ;

                RTS

:CCC            STX             AArg            ; (LB) of 3CharTable / $9A
                STA             AArg+1          ; (HB) of 3CharTable / $9B
                LDY             #$03            ; length of character string

:DDD            LDA             (AArg),Y        ;
                STA             TypeStr+1,Y     ;
                DEY                             ;
                BNE             :DDD            ; loop to get all 3 characters

                RTS

HexTable
                DB              $00
                DB              $01
                DB              $02
                DB              $03
                DB              $04
                DB              $05
                DB              $06
                DB              $07
                DB              $08
                DB              $09
                DB              $0A
                DB              $0B
                DB              $0C
                DB              $0F
                DB              $10
                DB              $11
                DB              $12
                DB              $13
                DB              $14
                DB              $15
                DB              $16
                DB              $19
                DB              $1A
                DB              $1B
                DB              $20
                DB              $2A
                DB              $2B
                DB              $2C
                DB              $2D
```

```
            DB          $2E
            DB          $40
            DB          $41
            DB          $42
            DB          $50
            DB          $51
            DB          $52
            DB          $53
            DB          $54
            DB          $55
            DB          $56
            DB          $57
            DB          $58
            DB          $59
            DB          $5A
            DB          $5B
            DB          $5C
            DB          $5D
            DB          $5E
            DB          $5F
            DB          $6B
            DB          $6D
            DB          $6E
            DB          $6F
            DB          $A0
            DB          $AB
            DB          $AC
            DB          $AD
            DB          $B0
            DB          $B1
            DB          $B2
            DB          $B3
            DB          $B4
            DB          $B5
            DB          $B6
            DB          $B7
            DB          $B8
            DB          $B9
            DB          $BA
            DB          $BB
            DB          $BC
            DB          $BD
            DB          $BF
            DB          $C0
            DB          $C1
            DB          $C2
            DB          $C3
            DB          $C5
            DB          $C6
            DB          $C7
            DB          $C8
            DB          $C9
            DB          $CA
            DB          $D5
            DB          $D6
            DB          $D7
            DB          $D8
            DB          $DB
```

```
                DB              $E0
                DB              $E2
                DB              $EE
                DB              $EF
                DB              $F0
                DB              $F9
                DB              $FA
                DB              $FB
                DB              $FC
                DB              $FD
                DB              $FE
                DB              $FF

3CharTable
                ASC             'NON'
                ASC             'BAD'
                ASC             'PCD'
                ASC             'PTX'
                ASC             'TXT'
                ASC             'PDA'
                ASC             'BIN'
                ASC             'FNT'
                ASC             'FOT'
                ASC             'BA3'
                ASC             'DA3'
                ASC             'WPF'
                ASC             'SOS'
                ASC             'DIR'
                ASC             'RPD'
                ASC             'RPI'
                ASC             'AFD'
                ASC             'AFM'
                ASC             'AFR'
                ASC             'SCL'
                ASC             'PFS'
                ASC             'ADB'
                ASC             'AWP'
                ASC             'ASP'
                ASC             'TDM'
                ASC             '8SC'
                ASC             '8OB'
                ASC             '8IC'
                ASC             '8LD'
                ASC             'P8C'
                ASC             'DIC'
                ASC             'OCR'
                ASC             'FTD'
                ASC             'GWP'
                ASC             'GSS'
                ASC             'GDB'
                ASC             'DRW'
                ASC             'GDP'
                ASC             'HMD'
                ASC             'EDU'
                ASC             'STN'
                ASC             'HLP'
                ASC             'COM'
                ASC             'CFG'
                ASC          'ANM'
                ASC          'MUM'
                ASC          'ENT'
                ASC          'DVU'
                ASC          'FIN'
                ASC          'BIO'
                ASC          'TDR'
                ASC          'PRE'
                ASC          'HDV'
                ASC          'WP '
                ASC          'GSB'
                ASC          'TDF'
                ASC          'BDF'
                ASC          'SRC'
                ASC          'OBJ'
                ASC          'LIB'
                ASC          'S16'
                ASC          'RTL'
                ASC          'EXE'
                ASC          'PIF'
                ASC          'TIF'
                ASC          'NDA'
                ASC          'CDA'
                ASC          'TOL'
                ASC          'DVR'
                ASC          'LDF'
                ASC          'FST'
                ASC          'DOC'
                ASC          'PNT'
                ASC          'PIC'
                ASC          'ANI'
                ASC          'PAL'
                ASC          'OOG'
                ASC          'SCR'
                ASC          'CDV'
                ASC          'FON'
                ASC          'FND'
                ASC          'ICN'
                ASC          'MUS'
                ASC          'INS'
                ASC          'MDI'
                ASC          'SND'
                ASC          'DBM'
                ASC          'LBR'
                ASC          'ATK'
                ASC          'R16'
                ASC          'PAS'
                ASC          'CMD'
                ASC          'OS '
                ASC          'INT'
                ASC          'IVR'
                ASC          'BAS'
                ASC          'VAR'
                ASC          'REL'
                ASC          'SYS'
                DB           $00


```

```
**---------------------------------------------------------------
                ORG                             ; ($4xxx+)

NewCodeEnd      EQU             *               ; ($4xxx+)

**---------------------------------------------------------------

                SAV             I.FILEAUXTYPE   ;
                LST             OFF

                END

*============================================================


```