RAF Commander#

a Norton Commander Clone for the Atari 8bit

RAFCMAIN.ASM

```
; $Id: rafcmain.asm , cstrotm $
;
;  RAF Commander - A free File Manager for Atari 8bit
;  Copyright (C) 1999-2000 Regionalgruppe Atari Frankfurt / RAF
;
;  This program is free software; you can redistribute it and/or
;  modify it under the terms of the GNU General Public License
;  as published by the Free Software Foundation; either version 2
;  of the License, or (at your option) any later version.
;
;  This program is distributed in the hope that it will be useful,
;  but WITHOUT ANY WARRANTY; without even the implied warranty of
;  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;  GNU General Public License for more details.
;
;  You should have received a copy of the GNU General Public License
;  along with this program; if not, write to the Free Software
;  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
;  or visit http://www.gnu.org

FORTH = 0
.IF .NOT FORTH
DEBUG = 0
.ENDIF

BETA      = 1

ATARI_800 = 0
ATARI_XL  = 1

ATARI_DOS = 1
MY_DOS    = 0
SPARTA23  = 0
SPARTA_X  = 0

.INCLUDE MACROS.ASM
.INCLUDE EQU.ASM

.IF .NOT FORTH
  * = $4000
  JMP RAFCINIT
.ENDIF

; Padding for FORTH, aligning indirekt Jumps

.IF FORTH
NFA_PREV .= NFA_MON
NOP

; >>>>>>> RAFCINIT <<<<<<<
; --
NFA_RAFCINIT .CBYTE $88, "RAFCINIT"
LFA_RAFCINIT .WORD NFA_PREV  ; LINK TO "SCRCLR"
```

```
CFA_RAFCINIT .WORD PFA_RAFCINIT
NFA_PREV .= NFA_RAFCINIT
PFA_RAFCINIT
    LDY #0
    STY LMARGN
    INY
    STY CRSINH
    M_CHOUT clrscr
    JMP NEXT

.ELSE

RAFCINIT
    LDX #$E8            ; Init Stack Pointer
    STX CRSINH
    M_CHOUT clrscr
    LDA #0
    STA LMARGN

.IF BETA
    JSR BETAMESSAGE
.ENDIF

;    JMP RAFC            ; and go!

.ENDIF

; >>>>>>> RAFC <<<<<<
; --
.IF FORTH
NFA_PREV .= NFA_PROCESS

NFA_RAFC .CBYTE $84, "RAFC"
LFA_RAFC .WORD NFA_PREV
CFA_RAFC .WORD DOCOL
PFA_RAFC
    .WORD CFA_RAFCINIT
    .WORD CFA_0 ; 0
    .WORD CFA_READDIR
    .WORD CFA_1 ; 1
    .WORD CFA_READDIR
    .WORD CFA_REFSCR
    .WORD CFA_0             ; Start Action = 0
L1100
    .WORD CFA_RAFCDEBUG     ; DEBUG : show stack pointer
    .WORD CFA_DROP

    .WORD CFA_GETKEY
    .WORD CFA_GETAC
    .WORD CFA_DUP     ; DUP
    .WORD CFA_0BRANCH       ; 0BRANCH
    .WORD L1100-*
    .WORD CFA_ACTION
    .WORD CFA_BRANCH      ; BRANCH
    .WORD L1100-*
    .WORD CFA_EXIT     ; S;

NFA_LASTWORD = NFA_RAFC
```

```
.ELSE

RAFC
    JSR GETDIR
    JSR REFSCR
    DEX
    DEX
RAFC1
.IF DEBUG
    JSR RAFCDEBUG
.ENDIF
    JSR GETKEY
    STA 0,X
    JSR GETAC
    LDA 0,X
    BEQ RAFC1
    JSR ACTION
    JMP RAFC1
.ENDIF

; >>>>>> ACTIONTABLE <<<<<<<<

ACTAB
.INCBIN "ac.tab"

; Action Code and Process handling

.INCLUDE ACTION.ASM

; Screen IO and Keyboard Input Routines

.INCLUDE SCREENIO.ASM

; File IO Routines

.INCLUDE FILEIO.ASM

; Sort and Framehandling Routines

.INCLUDE SORT.ASM

; DEBUG Routines

.INCLUDE DEBUG.ASM

; >>>>>> Variable <<<<<

.INCLUDE VARTAB.ASM

; >>>>>> STRINGS <<<<<<

.INCLUDE STRTAB.ASM


; $Id: action.asm , cstrotm $
;
;  RAF Commander - A free File Manager for Atari 8bit
;  Copyright (C) 1999-2000 Regionalgruppe Atari Frankfurt / RAF
;
;  This program is free software; you can redistribute it and/or
```

```
;  modify it under the terms of the GNU General Public License
;  as published by the Free Software Foundation; either version 2
;  of the License, or (at your option) any later version.
;
;  This program is distributed in the hope that it will be useful,
;  but WITHOUT ANY WARRANTY; without even the implied warranty of
;  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;  GNU General Public License for more details.
;
;  You should have received a copy of the GNU General Public License
;  along with this program; if not, write to the Free Software
;  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
;  or visit http://www.gnu.org

; Action Code and List processing Routines

; >>>>>>> DOSSTR <<<<<<<
; n -- addr
; ( creates DOS Filename )
.IF FORTH

NFA_DOSSTR .CBYTE $86, "DOSSTR"
LFA_DOSSTR .WORD NFA_PREV  ; UPLINK
CFA_DOSSTR .WORD PFA_DOSSTR
NFA_PREV .= NFA_DOSSTR
PFA_DOSSTR

    JSR DOSSTR
    JMP NEXT

.ENDIF

DOSSTR
    JSR DOSSTR0
    LDA P_DSTR
    STA 0,X
    LDA P_DSTR+1
    STA 1,X
    RTS

DOSSTR0
    JSR CALCLINE0
    JSR ADDONE

    LDA P_DSTR
    STA N
    LDA P_DSTR+1
    STA N+1

    LDY ACTFR
    LDA DNU,Y
    LDY #$0
    STY N+3             ; charcount
    INY
    JSR PUTCHAR1
    INY

DOSSTR1
    JSR ADDONE
```

```
    INC N+3
    LDA N+3
    CMP #9
    BNE DOSSTR2
    LDA #'.             ; dot
    JSR PUTCHAR1
DOSSTR2
    LDA N+3
    CMP #12             ; ende?
    BEQ DOSSTR4

    LDA (0,X)
    CMP #$20            ; leerzeichen ?
    BEQ DOSSTR1
    JSR PUTCHAR         ; nein = Zeichen setzen
    JMP DOSSTR1

DOSSTR4
    LDA #155            ; lineend
    BNE PUTCHAR1

PUTCHAR
    LDA (0,X)
    AND #$7F
PUTCHAR1
    STA (N),Y
    INY

DOSSTR3
    RTS


; >>>>>>> MARKENTRY <<<<<<<
; n --
.IF FORTH
NFA_MARKENTRY .CBYTE $89, "MARKENTRY"
LFA_MARKENTRY .WORD NFA_PREV  ; UPLINK
CFA_MARKENTRY .WORD PFA_MARKENTRY
NFA_PREV .= NFA_MARKENTRY
PFA_MARKENTRY
    JSR MARKENTRY
    JMP POP
.ENDIF

MARKENTRY
    JSR GETMARK
    ORA CURC
    STA (0,X)
    RTS

DELETEENTRY
    JSR GETMARK
    AND #bl
    STA (0,X)
    RTS

TOGGLEENTRY
    JSR GETMARK
    EOR CURC
```

```
    STA (0,X)
    RTS

GETMARK
    JSR CALCLINE0
    JSR ADDONE
    LDA (0,X)
    RTS

GETLOCK
    JSR CALCLINE0
    LDA (0,X)
    RTS

; >>>>>>> COUNTMARK <<<<<<<
; -- n
; ( count all marked direntries )
.IF FORTH
NFA_COUNTMARK .CBYTE $89, "COUNTMARK"
LFA_COUNTMARK .WORD NFA_GETKEY  ; LINK TO "GETKEY"
CFA_COUNTMARK .WORD PFA_COUNTMARK
PFA_COUNTMARK
    DEX
    DEX
    JSR COUNTMARK
    JMP NEXT
.ENDIF

merk        .BYTE 0
COUNTMARK
    LDA #0
    STA CABSA

    LDY NUMDA
COUNTMARK1
    STY 0,X
    TYA
    STA merk

    JSR GETMARK
    CMP #$20     ; Mark = Blank?
    BEQ COUNTMARK2
    INC CABSA
COUNTMARK2
    LDA merk            ; weg faktorieren WORK!!!!
    TAY
    DEY
    BPL COUNTMARK1
    LDA CABSA
    STA 0,X
    LDA #0
    STA 1,X
    JSR CALCCABS
    RTS

; >>>>>>> GETACTIONCODE <<<<<<<
; keycode -- n
.IF FORTH
NFA_GETAC .CBYTE $85, "GETAC"
```

```
LFA_GETAC .WORD NFA_PREV  ; UPLINK
CFA_GETAC .WORD PFA_GETAC
NFA_PREV .= NFA_GETAC
PFA_GETAC
    JSR GETAC
    JMP NEXT
.ENDIF
GETAC
    LDY 0,X
    LDA ACTAB-1,Y
    STA 0,X
    RTS

; >>>>>>> ACTION <<<<<<<
; n -- rc
.IF FORTH
NFA_ACTION .CBYTE $86, "ACTION"
LFA_ACTION .WORD NFA_PREV  ; UPLINK
CFA_ACTION .WORD PFA_ACTION
NFA_PREV .= NFA_ACTION
PFA_ACTION
    JSR ACTION
    JMP NEXT
.ENDIF

ACTION
    STX XSAVE
    JSR CALCCABS
    JSR SETACTFR

    LDA 0,X
    CMP #10         ; Quit ?
    BNE L1200

.IF FORTH
    LDY #0
    LDA #<CFA_QUIT  ; QUIT LOBYTE
    STA 0,X
    LDA #>CFA_QUIT  ; QUIT HIBYTE
    STA 1,X
    JMP PFA_EXECUTE        ; EXECUTE
    JMP NEXT
.ELSE
    JMP (DOSVEC)
.ENDIF


L1200
    CMP #61         ; Cursor down
    BNE L1220

    LDA #1
    JSR CRSDOWN
    JMP L1299

L1220
    CMP #60         ; Cursor up
    BNE L1221
```

```
    LDA #1
    JSR CRSUP
    JMP L1299

L1221
    CMP #62
    BNE L1223       ; Cursor 5 up
    JSR CRSHIDE
    LDA #5
    JSR CRSUP
    JMP L1299

L1223
    CMP #63
    BNE L1230       ; Cursor 5 down ?
    JSR CRSHIDE
    LDA #5
    JSR CRSDOWN
    JMP L1299

L1230
    LDA 0,X
    CMP #64             ; TAB - change Panel
    BNE L1240

    JSR TOGGLEACT
    JSR SETACTFR
    JSR CALCCABS

    JSR L_UPDLEFT
    JSR L_UPDRIGHT
    JMP L1299

L1240
    LDA 0,X
    CMP #30             ; sort filename
    BNE L1250

    LDA #8
    STA FLENKEY         ; 1st key = filename
    LDA #3
    STA FLENSEC         ; 2nd key = extender
    LDA #2
    STA OFFSETKEY
    LDA #10
    STA OFFSETSEC

    JSR SORTACT
    JMP L1299           ; and exit

L1250
    LDA 0,X
    CMP #31             ; sort externder
    BNE L1252

    LDA #3
    STA FLENKEY         ; 1st key = externder
    LDA #8
    STA FLENSEC         ; 2nd key = filename
```

```
    LDA #10
    STA OFFSETKEY
    LDA #2
    STA OFFSETSEC

    JSR SORTACT
    JMP L1299           ; end exit

L1252
    LDA 0,X
    CMP #32             ; sort size
    BNE L1270


    LDA #3
    STA FLENKEY         ; 1st key = size
    LDA #11
    STA FLENSEC         ; 2nd key = filename
    LDA #14
    STA OFFSETKEY
    LDA #2
    STA OFFSETSEC

    JSR SORTACT
    JMP L1299           ; end exit

L1270
    LDA 0,X
    CMP #11             ; left panel new dir
    BNE L1271

    LDA #5
    STA COLCRS
    JSR GETDRIVE
    BEQ L1270a          ; valid drive ?
    STA DNU0
    JSR L_READLEFT
L1270a
    JSR REFFRM
    JSR CLRFRMLEFT
    JSR L_UPDLEFT
    JMP L1299

L1271
    LDA 0,X
    CMP #12             ; right panel new dir
    BNE L1273

    LDA #25
    STA COLCRS
    JSR GETDRIVE
    BEQ L1271a          ; valid drive ?
    STA DNU1
    JSR L_READRIGHT
L1271a
    JSR CLRFRMRIGHT
    JSR REFFRM
    JSR L_UPDRIGHT
    JMP L1299
```

```
L1273
    LDA 0,X
    CMP #13
    BNE L1275

    JSR L_READACT
    JSR CLRFRMACT
    JSR L_UPDACT
    JMP L1299

L1275
    LDA 0,X
    CMP #20             ; mark entry
    BNE L1276

    LDA CABSA
    STA 0,X
    JSR TOGGLEENTRY
    LDA #1
    JSR CRSDOWN
    JSR L_UPDCRS
    JMP L1299

L1276
    LDA 0,X
    CMP #21             ; mark all entries
    BNE L1278

    LDY NUMDA

L1277
    STY 0,X
    TYA
    PHA
    JSR MARKENTRY
    PLA
    TAY
    DEY
    BNE L1277

    STY 0,X
    JSR MARKENTRY
    JSR L_UPDACT
    JMP L1299

L1278
    LDA 0,X
    CMP #22             ; unmark all entries
    BNE L1280

    LDY NUMDA

L1279
    STY 0,X
    TYA
    PHA
    JSR DELETEENTRY
    PLA
```

```
    TAY
    DEY
    BNE L1279

    STY 0,X
    JSR DELETEENTRY
    JSR L_UPDACT
    JMP L1299

L1280
    LDA 0,X
    CMP #23             ; toggle all entries
    BNE L1282

    LDY NUMDA

L1281
    STY 0,X
    TYA
    PHA
    JSR TOGGLEENTRY
    PLA
    TAY
    DEY
    BNE L1281

    STY 0,X
    JSR TOGGLEENTRY
    JSR L_UPDACT
    JMP L1299

L1282
    LDA 0,X
    CMP #15         ; lock / unlock
    BNE L1285

    LDA P_LOC0
    STA P_PROC
    LDA P_LOC0+1
    STA P_PROC+1

    LDA LOCVEC
    STA AJMVEC
    LDA LOCVEC+1
    STA AJMVEC+1
    JSR PROCESS
    JMP L1299

L1285
    LDA 0,X
    CMP #8          ; delete
    BNE L1286

    LDA P_DEL0
    STA P_PROC
    LDA P_DEL0+1
    STA P_PROC+1

    LDA DELVEC
```

```
    STA AJMVEC
    LDA DELVEC+1
    STA AJMVEC+1
    JSR PROCESS
    JMP L1299

L1286
    LDA 0,X
    CMP #17         ; start COM Programm
    BNE L1287

    LDA P_LOA0
    STA P_PROC
    LDA P_LOA0+1
    STA P_PROC+1

    LDA RUNVEC
    STA AJMVEC
    LDA RUNVEC+1
    STA AJMVEC+1
    JSR PROCESS
    JMP L1299

L1287
    LDA 0,X
    CMP #5          ; copy file
    BNE L1288
    LDA P_COP0
    STA P_PROC
    LDA P_LOC0+1
    STA P_PROC+1

    LDA COPVEC
    STA AJMVEC
    LDA COPVEC+1
    STA AJMVEC+1
    JSR PROCESS

    JMP L1299

L1288
    LDA 0,X
    CMP #6          ; move file
    BNE L1289
    LDA P_MOV0
    STA P_PROC
    LDA P_MOV0+1
    STA P_PROC+1

    LDA MOVVEC
    STA AJMVEC
    LDA MOVVEC+1
    STA AJMVEC+1
    JSR PROCESS
    JMP L1299

L1289

L1299
```

```
    LDA #0
    STA 0,X
    STA 1,X         ; Returncode --> WORK!!!

    LDX XSAVE
    RTS

; >>>>>>> PROCESS <<<<<<<
; --
.IF FORTH
NFA_PROCESS .CBYTE $87, "PROCESS"
LFA_PROCESS .WORD NFA_PREV  ; UPLINK
LFA_PROCESS .WORD NFA_PREV
CFA_PROCESS .WORD PFA_PROCESS
PFA_PROCESS
    JSR PROCESS
    JMP NEXT
.ENDIF

count .BYTE 0

PROCESS
    M_FRAME 4,5,28,4

    LDA #0
    STA ERRORCD
    JSR COUNTMARK
    LDA 0,X
    BNE PROCESS3
    LDA CABSA
    STA 0,X
    JSR TOGGLEENTRY

PROCESS3
    LDY NUMDA

PROCESS1
    STY COUNT
    STY 0,X
    JSR GETMARK
    CMP #$20    ; blank ?
    BEQ PROCESS4

    LDA COUNT
    STA 0,X
    JSR GETLOCK
    STA ACTLOCK

    LDA COUNT
    STA 0,X
    JSR DOSSTR
    JSR PROCESSTEXT

    LDA ERRORCD
    BEQ PROCESS2    ; Abort
    CMP #1          ; Skip File ?
    BEQ PROCESS4

    STX XSAVE
```

```
    JSR DOVEC       ; jump through vector
    LDX XSAVE
PROCESS4
    LDY COUNT
    BEQ PROCESS2
    DEY
    JMP PROCESS1

PROCESS2
    JSR REFFRM

    JSR L_READLEFT
    JSR L_READRIGHT

PROCESS5
    JSR SETACTFR
    JSR CALCCABS
    SEC
    LDA NUMDA
    SBC CABSA
    BPL PROCESS6
    LDA #1
    JSR CRSUP
    JMP PROCESS5

PROCESS6
    JSR CLRFRMLEFT
    JSR CLRFRMRIGHT

    JSR L_UPDLEFT
    JSR L_UPDRIGHT
    RTS

PROCESSTEXT
    M_POS 5,6
    M_PRINT P_PROC
    M_POS 5,7
    LDA #bl
    LDY #17
    JSR XTIMES
    M_POS 5,7
    M_PRINT P_DSTR
    LDA ERRORCD
    CMP #3              ; Yes to all?
    BEQ PROCESSTEXT1
    M_POS 5,8
    M_PRINT P_ASTR
    JSR GETCH
    LDY #4
PROCESSTEXT2
    DEY
    BEQ PROCESSTEXT3
    CMP QSTR,Y
    BNE PROCESSTEXT2
PROCESSTEXT3
    TYA
    STA ERRORCD
PROCESSTEXT1
    RTS
```

```
DOVEC
    LDA AJMVEC      ; get current action vector
    STA N
    LDA AJMVEC+1
    STA N+1
    JMP (N)         ; and jump through vector

; $Id: screenio.asm , cstrotm $
;
;  RAF Commander - A free File Manager for Atari 8bit
;  Copyright (C) 1999-2000 Regionalgruppe Atari Frankfurt / RAF
;
;  This program is free software; you can redistribute it and/or
;  modify it under the terms of the GNU General Public License
;  as published by the Free Software Foundation; either version 2
;  of the License, or (at your option) any later version.
;
;  This program is distributed in the hope that it will be useful,
;  but WITHOUT ANY WARRANTY; without even the implied warranty of
;  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;  GNU General Public License for more details.
;
;  You should have received a copy of the GNU General Public License
;  along with this program; if not, write to the Free Software
;  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
;  or visit http://www.gnu.org

; Screen and Keyboard IO Routines

; some definitions for Screen IO

ro = 5          ;  FRAME "TOP RIGHT"
lo = 17         ;  FRAME "TOP LEFT"
hr = 18         ;  FRAME "HORIZONTAL"
vr = 124        ;  FRAME "VERTICAL"
lu = 26         ;  FRAME "BOTTOM LEFT"
ru = 3          ;  FRAME "BOTTOM RIGHT"
cross = 19      ;  FRAME "CROSS"
cr = 155        ;  CARRIAGE RETURN
bl = 32         ;  BLANK
clrscr = 125    ;  CLEAR SCREEN

GETCH
    STX XSAVE-1
    TYA         ; Y Register retten
    PHA
    JSR KGETCH  ; Zeichen holen (in accu)
    TAX
    PLA         ; Y Register wiederherstellen
    TAY
    TXA
    LDX XSAVE-1
    RTS

OUTCH
    STX XSAVE-1
    TAX
    TYA         ; Y Register retten
    PHA
```

```
    TXA
    JSR EOUTCH  ; Zeichen ausgeben
    PLA         ; Y Register wiederherstellen
    TAY
    LDX XSAVE-1
    RTS

XSTROUT
    LDA (0,X)
    TAY
    INC 0,X
    LDA (0,X)
    STY 0,X
    STA 1,X
XSTROUT1
    LDA (0,X)

    AND #$7F
    ORA INVFLG
    JSR OUTCH
    LDA (0,X)
    PHA
    JSR ADDONE
XSTROUT2
    PLA
    AND #$80
    BEQ XSTROUT1
RTSVEC
    RTS

XFRAME      ; prints frame
            ; N = XLEN, N+1 = YLEN
            ; N+2 = XPOS
    LDA COLCRS
    STA N+2
    M_CHOUT lo
    LDA #hr
    LDY N
    JSR XTIMES
    M_CHOUT ro
    M_CHOUT cr

    DEC N+1
L0010
    LDA N+2
    STA COLCRS
    M_CHOUT vr
    LDA #bl
    LDY N
    JSR XTIMES
    M_CHOUT vr
    M_CHOUT cr
    DEC N+1
    BNE L0010

    LDA N+2
    STA COLCRS
    M_CHOUT lu
    LDA #hr
```

```
    LDY N
    JSR XTIMES
    M_CHOUT ru
    RTS

; >>>>>>> XTIMES <<<<<<<

.IF FORTH

NFA_XTIMES .CBYTE $86, "XTIMES"
LFA_XTIMES .WORD NFA_PREV  ; UPLINK
CFA_XTIMES .WORD PFA_XTIMES
NFA_PREV .= NFA_XTIMES
PFA_XTIMES
    LDY 0,X ; GET TIMES
    LDA 2,X ; GET CHARACTER
    JSR XTIMES
    JMP POPTWO

.ENDIF

XTIMES
    PHA
    JSR OUTCH
    PLA
    DEY
    BNE XTIMES
    RTS

; >>>>>>> POS <<<<<<<
.IF FORTH
NFA_POS .CBYTE $83, "POS"
LFA_POS .WORD NFA_PREV  ; UPLINK
CFA_POS .WORD PFA_POS
NFA_PREV .= NFA_POS
PFA_POS
    LDA 0,X ; GET XPOS
    STA COLCRS
    LDA 2,X ; GET YPOS
    STA ROWCRS
    JMP POPTWO
.ENDIF

; >>>>>>> REFRESH_SCREEN <<<<<<<
.IF FORTH
NFA_REFSCR .CBYTE $86, "REFSCR"
LFA_REFSCR .WORD NFA_PREV  ; UPLINK
CFA_REFSCR .WORD DOCOL
NFA_PREV .= NFA_REFSCR
PFA_REFSCR
    .WORD CFA_REFFRM        ; Frame
    .WORD CFA_0 ; 0
    .WORD CFA_REFDIR        ; show left dir
    .WORD CFA_1 ; 1
    .WORD CFA_REFDIR        ; show right dir
    .WORD CFA_EXIT          ; S;    ; end of word
.ELSE
REFSCR
    JSR REFFRM              ; Frame
```

```
    DEX
    DEX
    LDA #0
    STA 0,X
    JSR _REFDIR             ; left dir
    LDA #1
    STA 0,X
    JSR _REFDIR             ; right dir
    INX
    INX
    RTS
.ENDIF

.IF FORTH
; >>>>>>> REFRESH_FRAME <<<<<<<
NFA_REFFRM .CBYTE $86, "REFFRM"
LFA_REFFRM .WORD NFA_PREV  ; UPLINK
CFA_REFFRM .WORD PFA_REFFRM
NFA_PREV .= NFA_REFFRM
PFA_REFFRM
    JSR REFFRM
    JMP NEXT
.ENDIF

REFFRM
    M_POS 8,0
    M_PRINT P_0RAFM
    M_CHOUT cr
    M_CHOUT lo

    JSR HRLOOP      ; Horizontal line

    M_CHOUT ro

    M_POS 8,1
    M_PRINT P_0DSKM
    M_POS 10,1
    LDA DNU0
    JSR OUTCH

    M_POS 19,1
    M_CHOUT 23

    M_POS 27,1
    M_PRINT P_0DSKM
    M_POS 29,1
    LDA DNU1
    JSR OUTCH

    LDY #20
L1003
    STY ROWCRS
    LDA #0
    STA COLCRS
    M_CHOUT vr
    LDA #19
    STA COLCRS
    M_CHOUT vr
    LDA #38
```

```
    STA COLCRS
    M_CHOUT vr
    DEY
    CPY #1
    BNE L1003

    M_POS 0,21
    M_CHOUT 1

    JSR HRLOOP

    M_CHOUT 4
    M_POS 19,21
    M_CHOUT cross
    M_POS 0,22

    JSR DSKDSP
    JSR DSKDSP

    M_CHOUT vr
    M_POS 0,23
    M_CHOUT lu

    JSR HRLOOP

    M_CHOUT ru
    M_POS 19,23
    M_CHOUT 24

    M_POS 0,0

    RTS

;   -----------
HRLOOP
    LDY #37
L1001
    M_CHOUT hr
    DEY
    BNE L1001
    RTS

DSKDSP
    M_CHOUT vr

    M_PRINT P_1DSKM
    M_PRINT P_SFREM
    LDA #31
    LDY #4
    JSR XTIMES
    RTS

CLRFRMACT
    LDA ACTFR
    BNE CLRFRMRIGHT

CLRFRMLEFT
    LDA #1
    STA XPOS
```

```
    BNE CLRFRM

CLRFRMRIGHT
    LDA #20
    STA XPOS

CLRFRM
    LDA #20
CLRFRM1
    STA ROWCRS
    LDA XPOS
    STA COLCRS
    LDA #bl
    LDY #18
    JSR XTIMES
    DEC ROWCRS
    LDA ROWCRS
    CMP #1
    BNE CLRFRM1
    RTS

; >>>>>>> REFRESH_DIR <<<<<<<
; n --
.IF FORTH
NFA_REFDIR .CBYTE $86, "REFDIR"
LFA_REFDIR .WORD NFA_PREV  ; UPLINK
CFA_REFDIR .WORD PFA_REFDIR
NFA_PREV .= NFA_REFDIR

PFA_REFDIR
    STX XSAVE
    JSR _REFDIR
    LDX XSAVE
    JMP POP
.ENDIF

DIRCNT = TIB      ; Counter for act. entry
YPOS   = TIB+1    ; Y-Position
XPOS   = TIB+2    ; X-Position
MAXN   = TIB+3    ; max Entries

_REFDIR
    LDA #2
    STA YPOS
    LDA #0
    STA ACTCF

    LDY 0,X         ;  left (0) or right (1) panel
    CPY ACTFR       ;  act. panel?
    BNE L1400
    LDA #$80        ; then show cursor
    STA ACTCF

L1400
    LDA XPS0,Y
    STA XPOS

    LDA COFS0,Y
    STA DIRCNT
```

```
    LDA NUMD0,Y
    BMI L1405           ; no entries !
    STA MAXN
    LDA CPOS0,Y
    STA CPOSA

    CLC
    ASL 0,X           ; two byte values
    LDY 0,X
    LDA DIRMEM0,Y
    STA ACTDM
    LDA DIRMEM0+1,Y
    STA ACTDM+1

    INC MAXN
L1401
    LDA YPOS
    STA ROWCRS
    LDA XPOS
    STA COLCRS
    LDA DIRCNT
    STA 0,X
    JSR PRINTDIRENTRY

    INC YPOS
    INC DIRCNT
    LDA DIRCNT
    CMP MAXN            ; all entries ?
    BEQ L1403
    LDA YPOS
    CMP #21             ; all 19 Screen lines ?
    BEQ L1403
    BNE L1401

L1403
    LDA MAXN
    STA 0,X             ; last line = free Sectors
    LDA #22
    STA ROWCRS
    CLC
    LDA XPOS
    ADC #14
    STA COLCRS
    JSR PRINTDIRENTRY
L1405
    RTS

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

PRINTDIRENTRY
    LDA 0,X
    CMP CABSA
    BNE PRINTDIRENTRY1
    LDA ACTCF
    STA INVFLG

PRINTDIRENTRY1
    JSR CALCLINE0   ; calculate dir entry memory location
    JSR XSTROUT1    ; print dir entry
```

```
    LDA #0
    STA INVFLG
    RTS

; >>>>>>> SCRCLR <<<<<<<
; --
.IF FORTH
NFA_SCRCLR .CBYTE $86, "SCRCLR"
LFA_SCRCLR .WORD NFA_PREV  ; UPLINK
CFA_SCRCLR .WORD PFA_SCRCLR
NFA_PREV .= NFA_SCRCLR
PFA_SCRCLR
    M_CHOUT clrscr
    JMP NEXT
.ENDIF

; >>>>>>> GETKEY <<<<<<<
; -- n
.IF FORTH
NFA_GETKEY .CBYTE $86, "GETKEY"
LFA_GETKEY .WORD NFA_PREV  ; UPLINK
CFA_GETKEY .WORD PFA_GETKEY
NFA_PREV .= NFA_GETKEY
PFA_GETKEY
    JSR GETKEY
    PHA
    LDA #0
    JMP PUSH
.ENDIF

GETKEY
    LDA #$FF
    STA CH
LAGK1
    LDA CH
    CMP #$FF
    BEQ LAGK1

    LDY #$FF
    STY CH
    RTS

; >>>>>>> CRSUP <<<<<<<
; --
.IF FORTH
NFA_CRSUP .CBYTE $85, "CRSUP"
LFA_CRSUP .WORD NFA_PREV  ; UPLINK
CFA_CRSUP .WORD PFA_CRSUP
NFA_PREV .= NFA_CRSUP
PFA_CRSUP
    LDA #1
    JSR CRSUP
    JMP NEXT
.ENDIF

CRSUP
    STA N
    LDA #0
    STA N+1
```

```
    JSR CRSHIDE

CRSUP3
    JSR CALCCABS
    LDY ACTFR
    LDA CPOS0,Y
    BEQ CRSUP1      ; Cursor Position Top ?

    LDA CPOS0,Y     ; no, decrement cursor position
    STA CPOSA
    DEC CPOSA
    LDA CPOSA
    STA CPOS0,Y
    JMP CRSUP2

CRSUP1          ; CPOS = 0, Offset ?
    LDA COFS0,Y
    BEQ CRSUP2  ; Offset 0?

    STA COFSA   ; decrement Cursor Offset
    DEC COFSA
    LDA COFSA
    STA COFS0,Y
    INC N+1

CRSUP2
    DEC N
    BNE CRSUP3
    JSR CALCCABS
    LDA N+1
    BEQ CRSUP4
    JSR L_UPDACT

CRSUP4
    JMP CRSSHOW

CALCCABS        ; Calculates absolute Cursor positions
    LDY #2
CALCCABS1
    CLC
    LDA CPOS0,Y
    ADC COFS0,Y
    STA CABS0,Y
    DEY
    BPL CALCCABS1
    RTS

; >>>>>>> CRSDOWN <<<<<<<
; --
.IF FORTH
NFA_CRSDOWN .CBYTE $87, "CRSDOWN"
LFA_CRSDOWN .WORD NFA_PREV  ; UPLINK
CFA_CRSDOWN .WORD PFA_CRSDOWN
NFA_PREV .= NFA_CRSDOWN
PFA_CRSDOWN
    LDA #1
    JSR CRSDOWN
    JMP NEXT
.ENDIF
```

```
CRSDOWN
    STA N           ; Counter
    LDA #0
    STA N+1         ; Flag for Update
    JSR CRSHIDE

CRSDOWN3
    JSR CALCCABS
    LDY ACTFR

    LDA CABS0,Y
    CMP NUMD0,Y
    BEQ CRSDOWN2    ; max dir entries ?
    LDA CPOS0,Y
    CMP #18         ; 19 Screen entries ?
    BEQ CRSDOWN1

    LDA CPOS0,Y
    STA CPOSA
    INC CPOSA
    LDA CPOSA
    STA CPOS0,Y
    BNE CRSDOWN2

CRSDOWN1
    LDA COFS0,Y     ; increment cursor offset
    STA COFSA
    INC COFSA
    LDA COFSA
    STA COFS0,Y
    INC N+1

CRSDOWN2
    DEC N
    BNE CRSDOWN3
    JSR CALCCABS
    LDA N+1
    BEQ CRSDOWN4
    JSR L_UPDACT

CRSDOWN4
    JMP CRSSHOW

; >>>>>>> GETDRIVE <<<<<<<
; -- n
.IF FORTH
NFA_GETDRV .CBYTE $88, "GETDRIVE"
LFA_GETDRV .WORD NFA_PREV  ; UPLINK
CFA_GETDRV .WORD PFA_GETDRV
NFA_PREV .= NFA_GETDRV
PFA_GETDRV
    STX XSAVE
    JSR GETDRIVE
    LDX XSAVE
    PHA
    LDA #0
    JMP PUSH
.ENDIF
```

```
GETDRIVE
    LDA COLCRS
    PHA             ; save xpos
    LDA #6
    STA ROWCRS
    LDA #9
    STA N
    LDA #2
    STA N+1
    JSR XFRAME
    PLA
    STA COLCRS
    INC COLCRS
    LDA #7
    STA ROWCRS
    M_PRINT P_DRVS
    JSR GETCH
    PHA             ; save number
    SEC
    SBC #49
    BMI GETDRVERROR ; drive <= 0 ? -> error
    CMP #8
    BPL GETDRVERROR ; drive > 8 ? -> error
    TAY
    LDA DRVS,Y
    CMP #'x
    BNE GETDRV1
GETDRVERROR
    PLA
    LDA #0          ; error !
    PHA
GETDRV1
    PLA
    RTS

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

CRSHIDE

    LDA #0
    STA ACTCF
    BEQ L_UPDCRS3

L_UPDCRS            ; Update act. Cursor
CRSSHOW
    LDA #$80
    STA ACTCF
L_UPDCRS3
    LDA CPOSA
L_UPDCRS2
    PHA
    CLC
    ADC COFSA
    STA 0,X
    PLA
    CLC
    ADC #2
    STA ROWCRS
```

```
    LDY ACTFR
    LDA XPS0,Y
    STA COLCRS

    JSR PRINTDIRENTRY
    RTS

L_UPDACT
    LDA ACTFR
    BNE L_UPDRIGHT

L_UPDLEFT
    LDA #0
    BEQ L_1

L_UPDRIGHT
    LDA #1
L_1
    STA 0,X
    JMP _REFDIR

; $Id: fileio.asm , cstrotm $
;
;  RAF Commander - A free File Manager for Atari 8bit
;  Copyright (C) 1999-2000 Regionalgruppe Atari Frankfurt / RAF
;
;  This program is free software; you can redistribute it and/or
;  modify it under the terms of the GNU General Public License
;  as published by the Free Software Foundation; either version 2
;  of the License, or (at your option) any later version.
;
;  This program is distributed in the hope that it will be useful,
;  but WITHOUT ANY WARRANTY; without even the implied warranty of
;  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;  GNU General Public License for more details.
;
;  You should have received a copy of the GNU General Public License
;  along with this program; if not, write to the Free Software
;  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
;  or visit http://www.gnu.org

; DOS File IO Routines


; >>>>>>> READDIR <<<<<<<
; n --
.IF FORTH
NFA_READDIR .CBYTE $87, "READDIR"
LFA_READDIR .WORD NFA_PREV  ; UPLINK
CFA_READDIR .WORD PFA_READDIR
NFA_PREV .= NFA_READDIR
PFA_READDIR
    JSR _READDIR
    JMP POP
.ENDIF

_READDIR
    STX XSAVELOCAL
```

```
    LDA 0,X
    PHA         ; save stack parameter
    BNE L1300   ; 0 = left, 1 = right dir-panel

    LDA DNU0    ; Set Diskdrive Number
    STA DFI0+1
    BNE L1301
L1300
    LDA DNU1    ; Set Diskdrive Number
    STA DFI1+1

L1301
    CLC
    ASL 0,X
    LDY 0,X

    LDA P_0DFIL,Y     ; load 2 Byte parameter
    STA ICBAL+$60
    LDA P_0DFIL+1,Y
    STA ICBAH+$60
    LDA DIRMEM0,Y
    STA ACTDM
    LDA DIRMEM0+1,Y
    STA ACTDM+1

    LDA #0
    STA N

    LDX #$60
    LDA #CIOOPEN
    STA ICCOM,X
    LDA #6              ; Aux1=6 = Read Dir
    STA ICAX1,X
    JSR CIOV
    BMI ERROR
L1302
    LDA #CIOGETREC
    STA ICCOM,X

    LDA ACTDM
    STA ICBAL,X
    LDA ACTDM+1
    STA ICBAH,X
    LDA #<$12
    STA ICBLL,X
    LDA #>$12
    STA ICBLH,X
    JSR CIOV

    LDA ICSTA+$60
    BMI ERROR

    LDY #1
    LDA (ICBALZ),Y      ; FILENAME OR FREE SECTORS ?
    CMP #$20
    BEQ L1304

    LDY #3              ; IT'S THE FREE SECTORS LINE
    LDA (ICBALZ),Y
```

```
    ORA #$80
    STA (ICBALZ),Y
    JMP CLOSEDIR

L1304                   ; ENTRY IS FILENAME
    INC N               ; NUMBER OF ENTRIES

    LDY #$10
    LDA (ICBALZ),Y
    ORA #$80
    STA (ICBALZ),Y
    DEY
L1305
    LDA (ICBALZ),Y
    AND #$7F
    STA (ICBALZ),Y
    DEY
    BNE L1305

L1310
    CLC
    LDA ACTDM
    ADC RLEN
    STA ACTDM
    BCC L1302
    INC ACTDM+1
    JMP L1302

ERROR
CLOSEDIR
    LDX #$60
    JSR CLOSECHANNEL    ; close channel 6

    DEC N
    PLA                 ; get stack parameter (left or right)
    TAY

    LDA N
    STA NUMD0,Y

    TYA
    CLC
    ASL
    TAY
    LDA ACTDM
    STA DIRMEE0,Y
    LDA ACTDM+1
    STA DIRMEE0+1,Y

    JSR CALCCABS
    LDX XSAVELOCAL
    RTS
;   ---------
L_READLEFT
    LDY #0
    BEQ L_READ
L_READRIGHT
    LDY #1
    BNE L_READ
```

```
L_READACT
    LDY ACTFR
L_READ
    STY 0,X
    JMP _READDIR

GETDIR
    DEX
    DEX
    JSR L_READLEFT
    JSR L_READRIGHT
    INX
    INX
    RTS

LOCKFILE
    LDA ACTLOCK
    CMP #'*
    BEQ UNLOCKFILE

    LDA #ciolock
    PHA
    JMP DOCIO

UNLOCKFILE
    LDA #ciounlock
    PHA
    JMP DOCIO

DELETEFILE
    LDA #ciodelete
    PHA
    JMP DOCIO

RUNFILE
    LDA #cioload
    PHA

DOCIO
    LDA 0,X
    PHA
    LDA 1,X
    PHA
    LDX #$60
    PLA
    STA ICBAH,X
    PLA
    STA ICBAL,X
    PLA
    STA ICCOM,X
    JSR CIOV
    JSR CHECKERROR
    RTS

COPYFILE
    JSR GETDESTFILENAME
    JSR OPENSOURCEFILE
    JSR OPENDESTFILE
```

```
    JSR DOCOPY
    LDX #$50
    JSR CLOSECHANNEL
    LDX #$60
    JSR CLOSECHANNEL
    RTS

MOVEFILE
    JSR COPYFILE
    LDA P_DSTR
    STA 0,X
    LDA P_DSTR+1
    STA 1,X
    JSR DELETEFILE
    RTS

DOCOPY
    LDX #$60
    LDA #ciogetchar
    STA ICCOM,X
    LDA DIRMEE1
    PHA
    STA ICBAL,X
    LDA DIRMEE1+1
    PHA
    STA ICBAH,X
    LDA COPBUF
    STA ICBLL,X
    LDA COPBUF+1
    STA ICBLH,X
    JSR CIOV        ; Read in Buffer
    JSR CHECKERROR

    LDA ICBLL,X
    PHA
    LDA ICBLH,X
    PHA             ; Get number of bytes in buffer

    LDX #$50
    LDA #cioputchar
    STA ICCOM,X
    PLA
    STA ICBLH,X
    PLA
    STA ICBLL,X
    PLA
    STA ICBAH,X
    PLA
    STA ICBAL,X
    JSR CIOV
    JSR CHECKERROR

    LDX #$60
    LDA ICSTA,X
    BPL DOCOPY

DOCOPYEND
    RTS
```

```
OPENSOURCEFILE
    LDX #$60        ; channel 6
    LDA #cioopen
    STA ICCOM,X
    LDA P_DSTR
    STA ICBAL,X
    LDA P_DSTR+1
    STA ICBAH,X
    LDA #cioread
    STA ICAX1,X
    LDA #0
    STA ICAX2,X
    JSR CIOV
    RTS

OPENDESTFILE
    LDX #$50        ; channel 5
    LDA #cioopen
    STA ICCOM,X
    LDA P_CSTR
    STA ICBAL,X
    LDA P_CSTR+1
    STA ICBAH,X
    LDA #ciowrite
    STA ICAX1,X
    LDA #0
    STA ICAX2,X
    JSR CIOV
    RTS

CLOSECHANNEL
    LDA #cioclose
    STA ICCOM,X
    JSR CIOV
    RTS

GETDESTFILENAME
    LDA P_DSTR
    STA N
    LDA P_DSTR+1
    STA N+1
    LDA P_CSTR
    STA N+2
    LDA P_CSTR+1
    STA N+3
    LDY #15
GDF1
    LDA (N),Y
    STA (N+2),Y
    DEY
    BNE GDF1

    LDA ACTFR
    EOR #1      ; get number of inactive frame
    TAY
    LDA DNU,Y
    LDY #1
    STA (N+2),Y ; set opposite Drive number
    RTS
```

```
CHECKERROR
    RTS

; $Id: sort.asm , cstrotm $
;
;  RAF Commander - A free File Manager for Atari 8bit
;  Copyright (C) 1999-2000 Regionalgruppe Atari Frankfurt / RAF
;
;  This program is free software; you can redistribute it and/or
;  modify it under the terms of the GNU General Public License
;  as published by the Free Software Foundation; either version 2
;  of the License, or (at your option) any later version.
;
;  This program is distributed in the hope that it will be useful,
;  but WITHOUT ANY WARRANTY; without even the implied warranty of
;  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;  GNU General Public License for more details.
;
;  You should have received a copy of the GNU General Public License
;  along with this program; if not, write to the Free Software
;  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
;  or visit http://www.gnu.org

; Sort and Framehandling Routines

; >>>> BUBBLESORT <<<<
FLAST       = N     ; End outer loop
FLENKEY   .BYTE 0   ; Key length
RLEN      .BYTE $11 ; Record Length
OFFSETKEY .BYTE 0   ; Key offset
FIRST       = N+2   ; 1st element pointer
SECOND      = N+4   ; 2nd element pointer
LAST      .WORD 0   ; End, inner loop
FENDKEY   .BYTE 0   ; end, key loop
ORDER     .BYTE 0   ; Order, sort 0,1
FLENSEC   .BYTE 0   ; 2nd field length
OFFSETSEC .BYTE 0   ; 2nd field offset
FENDSEC   .BYTE 0   ; End 2nd field

BUBBLESORT
; set first (2 Byte), last (2 Byte)
; set FLENKEY (1 Byte), OFFSETKEY (1 Byte)
; set FLENSEC (1 Byte), OFFSETSEC (1 Byte)
; set RLEN (1 Byte), set ORDER (1 Byte)
    LDA ORDER
    EOR #01
    STA ORDER

    LDA OFFSETKEY
    CLC             ; Find end of
    ADC FLENKEY     ; first key
    STA FENDKEY
    LDA OFFSETSEC
    CLC             ; Find end of
    ADC FLENSEC     ; second key
    STA FENDSEC
;
; All parms. now in zero page.
; Next, set pointer to the end
```

```
; of the outer loop
    LDA LAST+1
    STA FLAST+1
    SEC
    LDA LAST
    SBC RLEN
    STA FLAST
    BCS SETSECOND
;
    DEC FLAST+1
; Start of outer loop.
; Adjust the second pointer to
; point to the first pointer
; plus the record length.
;
SETSECOND
    CLC
    LDA FIRST+1
    STA SECOND+1
    LDA FIRST
    ADC RLEN
    STA SECOND
    BCC SORTKEY
    INC SECOND+1
; Start of the inner loop.
; 1. Compare the Key field of
;    the two sort elements.
; 2. If we find a mismatch,
;    do we need to swap them?
SORTKEY
    LDY OFFSETKEY
;
KEYLOOP
;
    LDA (FIRST),Y ;get a byte
    CMP (SECOND),Y ;of each.
    BEQ CHKMORE ;If = Continue.
;
    BCC NOSWAP  ;If F<S no swap.
;
    BCS SWAP    ;If F>S swap.
;
CHKMORE
;
    INY         ;Adjust pointer
    CPY FENDKEY ;All done?
    BNE KEYLOOP ;No.  Continue.
;
; At this point, all bytes in
; the key field of both sort
; elements are equal.  Drop to
; SORTSECOND and check the
; secondary fields.
;   If their lengths = 0, then
; we don't have 2ndry fields.
; We are only doing a one-field
; sort--goto the Noswap routine.
; If the key fields are equal,
; there is no need to swap them.
```

```
;
SORTSEC
    LDA FLENSEC ;A second field?
    BEQ NOSWAPBD ;No.
;
    LDY OFFSETSEC
SECLOOP
    LDA (FIRST),Y ;Compare byte
    CMP (SECOND),Y ;by byte...
    BEQ CHKMORE2 ;If = do more.
;
    BCC NOSWAP  ;If F<S Noswap.
;
    BCS SWAP    ;If F>s Swap.
;
CHKMORE2
    INY         ;Point to next.
    CPY FENDSEC ;End of 2nd?
    BNE SECLOOP ;No, do more.
;
; Now, both the key and the
; secondary fields of both sort
; elemtents are equal. Goto
; the noswap routine through the
; "back door." No need to check
; the order, no need to swap.
    BEQ NOSWAPBD
;
; This is where the swapping
; occurs. First, check the
; swapping order.
; (Assume swapping in ascending
; order.) If order<>0, then
; sort in descending order.
;
SWAP
;
    LDA ORDER   ;Get order
    BNE NOSWAPBD ;Not 0, No swap
;
; Swap routine's back door.
; If NOSWAP decides we need to
; swap by checking the order,
; we need to come here (instead
; of SWAP) or we would go into
; a continuous loop.
;
SWAPBD
    LDY #0
SWAPLOOP
    LDA (FIRST),Y ;Key byte
    PHA         ;  to stack.
    LDA (SECOND),Y ; 2ndry byte
    STA (FIRST),Y ;  to key.
    PLA         ;Key from stack
    STA (SECOND),Y ;to 2ndry.
    INY         ;Next byte.
    CPY RLEN    ; More?
    BNE SWAPLOOP ;Yes. Continue.
```

```
;
; All bytes have been swapped.
; Now adjust pointers to the
; next elements for the sort.
;  Goto the noswap back door.
;
    BEQ NOSWAPBD
;
NOSWAP
;
    LDA ORDER   ;Is ORDER=1?
    BNE SWAPBD  ;Yes. Swap them
;
NOSWAPBD
    CLC
    LDA SECOND
    ADC RLEN
    STA SECOND
    LDA SECOND+1
    ADC #0
    STA SECOND+1
    CMP LAST+1
    BNE SORTKEY
;
    LDA SECOND
    CMP LAST
    BNE SORTKEY
;
; We've made one pass through
; the sort's inner loop.  Now,
; adjust the outer loop and
; check if we're done with it.
; If not, readjust the inner
; loop pointer to the outer loop
; pointer + the record length.
;
    CLC
    LDA FIRST
    ADC RLEN
    STA FIRST
    LDA FIRST+1
    ADC #0
    STA FIRST+1
    CMP FLAST+1
    BEQ CHECK2
    JMP SETSECOND
;
CHECK2
    LDA FIRST
    CMP FLAST
    BEQ ENDSORT
    JMP SETSECOND
;
ENDSORT
    RTS         ; RETURN

SORTACT
    LDA ACTFR
    BNE SORTRIGHT
```

```
SORTLEFT
    LDA DIRMEM0         ; left panel
    STA FIRST
    LDA DIRMEM0+1
    STA FIRST+1
    LDA DIRMEE0
    STA LAST
    LDA DIRMEE0+1
    STA LAST+1
    JSR BUBBLESORT      ; sort
    JSR L_UPDLEFT       ; update screen
    RTS

SORTRIGHT
    LDA DIRMEM1         ; left panel
    STA FIRST
    LDA DIRMEM1+1
    STA FIRST+1
    LDA DIRMEE1
    STA LAST
    LDA DIRMEE1+1
    STA LAST+1
    JSR BUBBLESORT      ; sort
    JSR L_UPDRIGHT      ; update screen
    RTS

SETACTFR
    LDY ACTFR
    LDA NUMD0,Y
    STA NUMDA

    LDA COFS0,Y
    STA COFSA

    LDA CPOS0,Y
    STA CPOSA

    TYA             ; Accu * 2 for 2 Byte parameter
    CLC
    ASL
    TAY

    LDA DIRMEM0,Y
    STA ACTDM
    LDA DIRMEM0+1,Y
    STA ACTDM+1
    RTS

TOGGLEACT
    LDA ACTFR
    EOR #$01
    STA ACTFR
    RTS

; >>>>>>> ADD ONE <<<<<<
; n -- n+1

ADDONE
```

```
    CLC
    LDA 0,X
    ADC #1
    STA 0,X
    BCC ADDONE1
    INC 1,X
ADDONE1
    RTS

; >>>>>>> CALC LINE <<<<<<<
; n -- addr
;
.IF FORTH
NFA_CALCLINE .CBYTE $88, "CALCLINE"
LFA_CALCLINE .WORD NFA_PREV  ; UPLINK
CFA_CALCLINE .WORD PFA_CALCLINE
NFA_PREV .= NFA_CALCLINE
PFA_CALCLINE

    JSR CALCLINE0
    JMP NEXT
.ENDIF

CALCLINE0
    LDA 0,X

    LDY ACTDM
    STY 0,X
    LDY ACTDM+1
    STY 1,X

    TAY
    CPY #0
    BEQ CALCLINE3   ; 0 ?

CALCLINE1       ; * 16
    CLC
    LDA 0,X
    ADC #$11
    STA 0,X
    LDA 1,X
    ADC #0
    STA 1,X
    DEY
    BNE CALCLINE1

CALCLINE3
    RTS

; $Id: debug.asm , cstrotm $
;
;  RAF Commander - A free File Manager for Atari 8bit
;  Copyright (C) 1999-2000 Regionalgruppe Atari Frankfurt / RAF
;
;  This program is free software; you can redistribute it and/or
;  modify it under the terms of the GNU General Public License
;  as published by the Free Software Foundation; either version 2
;  of the License, or (at your option) any later version.
;
;  This program is distributed in the hope that it will be useful,
```

```
;  but WITHOUT ANY WARRANTY; without even the implied warranty of
;  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;  GNU General Public License for more details.
;
;  You should have received a copy of the GNU General Public License
;  along with this program; if not, write to the Free Software
;  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
;  or visit http://www.gnu.org

; Debugging Routines


; >>>>>>> RAFCDEBUG <<<<<<<
; --
.IF FORTH
NFA_RAFCDEBUG .CBYTE $82, "##"
LFA_RAFCDEBUG .WORD NFA_PREV  ; UPLINK
CFA_RAFCDEBUG .WORD PFA_RAFCDEBUG
NFA_PREF = NFA_RAFCDEBUG
PFA_RAFCDEBUG
    JSR RAFCDEBUG
    JMP NEXT
.ENDIF

RAFCDEBUG
    STX XSAVELOCAL
    LDA #0
    STA COLCRS
    STA ROWCRS
    TXA
    JSR HEX2
    TSX
    TXA
    JSR HEX2
    LDX XSAVELOCAL
    RTS

; print accum as two hex digits
HEX2    PHA
        LSR     A
        LSR     A
        LSR     A
        LSR     A
        JSR     HEX2A
        PLA
HEX2A   AND     #$0F
        JSR     HXDGT
        JMP     OUTCH
;
;convert hex digit to ASCII
;
HXDGT   CMP     #$0A
        BCC     HXDGT1
        CLC
        ADC     #7
HXDGT1  ADC     #'0
        RTS
```

```
.IF BETA
BETAMSG1    .CBYTE " RAF Commander 2000 BETA V1 "
BETAMSG2    .CBYTE "  25.3.2000 NF ATARI DOS    "
BETAMSG3    .CBYTE "Report Bugs / Suggestions to"
BETAMSG4    .CBYTE "   rafcbugs@strotmann.de    "
BETAMSG5    .CBYTE "  RAF Commander Homepage    "
BETAMSG6    .CBYTE "http://www.strotmann.de/rafc"
BETAMSG7    .CBYTE "Released u. GPL, www.gnu.org"
P_BETAMSG1  .WORD BETAMSG1
P_BETAMSG2  .WORD BETAMSG2
P_BETAMSG3  .WORD BETAMSG3
P_BETAMSG4  .WORD BETAMSG4
P_BETAMSG5  .WORD BETAMSG5
P_BETAMSG6  .WORD BETAMSG6
P_BETAMSG7  .WORD BETAMSG7

BETAMESSAGE
    M_FRAME 4,5,28,9
    M_POS 5,6
    M_PRINT P_BETAMSG1
    M_POS 5,7
    M_PRINT P_BETAMSG2
    M_POS 5,9
    M_PRINT P_BETAMSG3
    M_POS 5,10
    M_PRINT P_BETAMSG4
    M_POS 5,11
    M_PRINT P_BETAMSG5
    M_POS 5,12
    M_PRINT P_BETAMSG6
    M_POS 5,13
    M_PRINT P_BETAMSG7
    JSR GETCH
    M_CHOUT clrscr
    RTS

.ENDIF

; $Id: vartab.asm , cstrotm $
;
;  RAF Commander - A free File Manager for Atari 8bit
;  Copyright (C) 1999-2000 Regionalgruppe Atari Frankfurt / RAF
;
;  This program is free software; you can redistribute it and/or
;  modify it under the terms of the GNU General Public License
;  as published by the Free Software Foundation; either version 2
;  of the License, or (at your option) any later version.
;
;  This program is distributed in the hope that it will be useful,
;  but WITHOUT ANY WARRANTY; without even the implied warranty of
;  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;  GNU General Public License for more details.
;
;  You should have received a copy of the GNU General Public License
;  along with this program; if not, write to the Free Software
;  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
;  or visit http://www.gnu.org

; Variable Memory
```

```
VARMEM

    XPS0  .BYTE 1               ; XPos left panel
    XPS1  .BYTE 20              ; XPos right panel

    DNU
    DNU0  .BYTE '1              ; Drive Number left
    DNU1  .BYTE '2              ; Drive Number right

    CPOS0 .BYTE 0               ; Cursor Position left
    CPOS1 .BYTE 0               ; Cursor Position right
    CPOSA .BYTE 0               ; Cursor Porition act. Panel

    COFS0 .BYTE 0               ; Cursor Offset left
    COFS1 .BYTE 0               ; Cursor Offset right
    COFSA .BYTE 0               ; Cursor Offset act. Panel

    CABS0 .BYTE 0               ; Cursor absolute Position left
    CABS1 .BYTE 0               ; Cursor absolute Position right
    CABSA .BYTE 0               ; Cursor absolute Position act

    NUMD0 .BYTE 0               ; Number of Dir entries left panel
    NUMD1 .BYTE 0               ; Number of Dir entries right panel
    NUMDA .BYTE 0               ; Number of Dir entries act. panel

    ACTFR .BYTE 0               ; act. Panel (0=left, 1=right)
    ACTDM .WORD 0               ; act. Panel dir Memory start
    ACTCF .BYTE 0               ; act. Panel Cursor Flag

    ACTLOCK .BYTE 0             ; lock marker act. entry
    ERRORCD .BYTE 0             ; Errorcode
    XSAVELOCAL .BYTE 0

; >>>>>> VECTORS <<<<<<

VECTAB
    FREMEM      .WORD $6000     ; begin free memory
    DIRMEM0     .WORD $6000     ; begin memory for left panel
    DIRMEM1     .WORD $6500     ; begin memory for right panel
    DIRMEE0     .WORD $64FF     ; end memory for left panel
    DIRMEE1     .WORD $6FFF     ; end memory for right panel
    COPBUF      .WORD $4000     ; length of copy buffer
    ACTVEC      .WORD RTSVEC    ; action table chain vector

    AJMVEC      .WORD RTSVEC        ; active jump vector
    LOCVEC      .WORD LOCKFILE      ; jump vector lock file
    DELVEC      .WORD DELETEFILE    ; jump vector delete file
    COPVEC      .WORD COPYFILE      ; jump vector copy file
    MOVVEC      .WORD MOVEFILE      ; jump vector move file
    RUNVEC      .WORD RUNFILE       ; jimp vector run file

; $Id: strtab.asm , cstrotm $
;
;  RAF Commander - A free File Manager for Atari 8bit
;  Copyright (C) 1999-2000 Regionalgruppe Atari Frankfurt / RAF
;
;  This program is free software; you can redistribute it and/or
;  modify it under the terms of the GNU General Public License
;  as published by the Free Software Foundation; either version 2
;  of the License, or (at your option) any later version.
```

```
;
;  This program is distributed in the hope that it will be useful,
;  but WITHOUT ANY WARRANTY; without even the implied warranty of
;  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;  GNU General Public License for more details.
;
;  You should have received a copy of the GNU General Public License
;  along with this program; if not, write to the Free Software
;  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
;  or visit http://www.gnu.org

; Fixed Srings

STRFIX
    QSTR .BYTE  "ANYL"       ; String  for  Question  Dialog.
                             ; 1st letter for "(A)bort",
                             ; 2nd letter for "(N)o",
                             ; 3rd letter for "(Y)es",
                             ; 4th letter for "A(L)l"

; String table for relocating text
; table of pointers to text strings
; String should never accessed direct in this
; program, but through pointers to the text

STRTAB
    P_0RAFM .WORD RAF0

    P_0DSKM .WORD DSK0
    P_1DSKM .WORD DSK1

    P_0FREM .WORD FRES
    P_1FREM .WORD FRES
    P_SFREM .WORD FRES

    P_0DFIL .WORD DFI0
    P_1DFIL .WORD DFI1
    P_DRVS  .WORD DRVS
    P_DSTR  .WORD DSTR
    P_CSTR  .WORD CSTR

    P_LOC0  .WORD LOC0
    P_COP0  .WORD COP0
    P_MOV0  .WORD MOV0
    P_DEL0  .WORD DEL0
    P_LOA0  .WORD LOA0
    P_PROC  .WORD 0         ; Pointer to Process Text
    P_ASTR  .WORD ASTR

; text srings for screen output
; Hi-Bit set is EOL marker
; (done by .CBYTE dircetive

STRMEM

    RAF0 .CBYTE "RAF Disk Commander 2000"

    LOC0 .CBYTE "(Un)Locking"
    COP0 .CBYTE "Copy"
```

```
    MOV0 .CBYTE "Move"
    DEL0 .CBYTE "Delete"
    LOA0 .CBYTE "Loading"

    DRVS .CBYTE "12xxxxx8x"
    ASTR .CBYTE "(Y)es (N)o (A)bort A(l)l"

    DSK0 .CBYTE "[Dx:]"
    DSK1 .CBYTE " D       "

    FRES .CBYTE "free:"
    CURC .BYTE 30                   ; Marker Char

    DFI
    DFI0 .BYTE "D1:*.*         "    ; FILTER LEFT
    DFI1 .BYTE "D1:*.*         "    ; FILTER RIGHT
    DSTR .CBYTE "D1:             "    ; String for Disk Operations
    CSTR .CBYTE "D1:             "    ; String for Copy/Move Operations

; $Id: equ.asm , cstrotm $
;
;  RAF Commander - A free File Manager for Atari 8bit
;  Copyright (C) 1999-2000 Regionalgruppe Atari Frankfurt / RAF
;
;  This program is free software; you can redistribute it and/or
;  modify it under the terms of the GNU General Public License
;  as published by the Free Software Foundation; either version 2
;  of the License, or (at your option) any later version.
;
;  This program is distributed in the hope that it will be useful,
;  but WITHOUT ANY WARRANTY; without even the implied warranty of
;  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;  GNU General Public License for more details.
;
;  You should have received a copy of the GNU General Public License
;  along with this program; if not, write to the Free Software
;  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
;  or visit http://www.gnu.org

; Equates and Definitions

cioopen     = 3

ciogetrec   = 5
ciogetchar  = 7
cioputrec   = 8
cioputchar  = 11
cioclose    = 12
ciodelete   = 33
ciolock     = 35
ciounlock   = 36
ciopoint    = 37
cionote     = 38
cioload     = 40
ciosave     = 41
cioformat   = 254

.IF MY_DOS
ciocreatedir = 34
ciochangedir = 41
```

```
.ENDIF

.IF SPARTA23 .OR SPARTA_X
ciolockdisk  = 34
ciogetfilelen= 39
ciocreatedir = 42
ciodeletedir = 43
ciochangedir = 44
ciosetboot   = 45
ciounlockdisk= 46
.ENDIF

.IF SPARTA_X
ciosetattr   = 49
.ENDIF

cioread      = 4
ciowrite     = 8
ciodir       = 6

; >>>>> Zero Page <<<<<

DOSVEC = $0A

ICHIDZ = $20
ICDNOZ = $21
ICCOMZ = $22
ICSTAZ = $23
ICBALZ = $24
ICBAHZ = $25
ICPTLZ = $26
ICPTHZ = $27
ICBLLZ = $28
ICBLHZ = $29
ICAX1Z = $2A
ICAX2Z = $2B
ICAX3Z = $2C
ICAX4Z = $2D
ICAX5Z = $2E
ICAX6Z = $2F

ROWCRS = $54
COLCRS = $55
LMARGN = $52

N       = $F0       ; $F0-$FE Free Bytes for RAF Commander Modules
XSAVE   = $FF       ; Save Place for X Register = Datastack Pointer
TIB     = $100      ; Terminal Input Buffer (80 Bytes)

; >>>>> PAGE 2-5 <<<<<<

INVFLG = $2B6
CRSINH = $2F0
CH     = $2FC

; Input/Output Control Block (IOCB)

ICHID  = $340
ICDNO  = $341
```

```
ICCOM  = $342
ICSTA  = $343
ICBAL  = $344
ICBAH  = $345
ICPTL  = $346
ICPTH  = $347
ICBLL  = $348
ICBLH  = $349
ICAX1  = $34A
ICAX2  = $34B
ICAX3  = $34C
ICAX4  = $34D
ICAX5  = $34E
ICAX6  = $34F

; Atari OS Vectors

CIOV   = $E456

.IF ATARI_800
  EOUTCH = $F6A4            ; put Value in Accu on Screen  (Atari 800)
  KGETCH = $F6E2            ; get Key from Keyboard and place in Accu (Atari 800)
.ENDIF

.IF ATARI_XL
  EOUTCH = $F2B0            ; put Value in Accu on Screen  (Atari XL)
  KGETCH = $F2F8            ; get Key from  Keyboard  and place in Accu (Atari XL)
.ENDIF


; $Id: macros.asm , cstrotm $
;
;  RAF Commander - A free File Manager for Atari 8bit
;  Copyright (C) 1999-2000 Regionalgruppe Atari Frankfurt / RAF
;
;  This program is free software; you can redistribute it and/or
;  modify it under the terms of the GNU General Public License
;  as published by the Free Software Foundation; either version 2
;  of the License, or (at your option) any later version.
;
;  This program is distributed in the hope that it will be useful,
;  but WITHOUT ANY WARRANTY; without even the implied warranty of
;  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;  GNU General Public License for more details.
;
;  You should have received a copy of the GNU General Public License
;  along with this program; if not, write to the Free Software
;  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
;  or visit http://www.gnu.org

; Macros

; >>>>>> MACROS <<<<<<

.MACRO M_FRAME
    LDA #%2
    STA ROWCRS
    LDA #%1
    STA COLCRS
    LDA #%3
```

```
    STA N
    LDA #%4
    STA N+1
    JSR XFRAME
.ENDM

.MACRO M_CHOUT
    LDA #%1
    JSR OUTCH
.ENDM

.MACRO M_POS
    LDA #%2
    STA ROWCRS
    LDA #%1
    STA COLCRS
.ENDM

.MACRO M_PRINT
    INX
    INX
    LDA # <%1
    STA 0,X
    LDA # >%1
    STA 1,X
    JSR XSTROUT
    DEX
    DEX
.ENDM
```