```
; ============================================================
;  geoGopherCvt: read file from network
; ============================================================
.if             Pass1
                .noeqin
                .include    geoGopherSym
                .include    geoGopherMac
                .include    geoGopher.inc
                .include    ultimate.inc
                .eqin
.endif
; ============================================================
;  Read file from network, convert if CVT, and write to disk.
;               pass:       a0, selector
;               return:     carry set on error, clear otherwise
;                           on success, file written
;               destroyed: a0, a1
; ============================================================
readCvt:        LoadW       r0,readMsg
                jsr         showStat
                MoveW       a0,a1           ;selector, for basename
                LoadW       a6,fileHeader   ;temp
                jsr         ultRdBlk        ;first block: dir entry, signature
                bcc         10$
                rts
10$             stx         eof
                sty         bytes           ;in case not a CVT file
                AddVW       2,a6            ;for length
                MoveW       a6,a0
                lda         bytes
                cmp         #254
                beq         ckCvt
                jsr         isItem          ;.X points past end, .A = type
                bcc         ckCvt           ;small, non-CVT file
                cmp         #TYP_ERR        ;error item?
                bne         50$
                MoveW       a0,a1
                MoveW       itemsBuf,a0
                txa                         ;from isItem
                clc
                adc         itemsBuf
                sta         itemsEnd
                lda         itemsBuf+1
                adc         #0
                sta         itemsEnd+1
                txa
                tay
40$             lda         (a6),y          ;copy from fileHeader (temp) to itemsBuf
                sta         (a0),y
                dey
                bne         40$
                jsr         cntItems
                jsr         clrStat
                ldx         #0
                jsr         doItems
                lda         #CNCL_ERR
                sec
                rts
50$             lda         #CVTLNERR
                sec
                rts
```

```
ckCvt:      ldx     #0              ;check CVT signature
70$         lda     cvtSig,x
            beq     90$
            cmp     fileHeader+36,x ;allow for length bytes
            beq     80$
            jsr     basename        ;name to saveName (a1 holds selector)
            lda     itemType
            cmp     #TYP_TEXT
            bne     75$
            LoadW   a8,saveName     ;from basename
            LoadW   r0,textDB       ;view/save dialog
            LoadW   RecoverVector,rstrDone  ;vwTextDB will cover it
            jsr     DoDlgBox
            lda     r0
            cmp     #VIEW
            bne     75$
            jsr     viewFile
            rts
75$         jsr     getName
            bcs     140$
            jsr     getPlain        ;download non-CVT file
            jsr     rstrDrv         ;restore original drive
            rts
80$         inx
            bne     70$
;           ================================================
90$         ldx     #2              ;allow for length bytes
            ldy     #0
100$        lda     fileHeader,x    ;save directory entry
            sta     dirBuf,y
            inx
            iny
            cpy     #30
            bne     100$
            ldx     #0
110$        lda     dirBuf+3,x      ;copy filename to saveName
            cmp     #$a0
            beq     120$
            sta     saveName,x
            inx
            cpx     #16
            bne     110$
120$        lda     #0
            sta     saveName,x
            jsr     getName         ;ask where to put file
            bcs     140$
            jsr     readHdr         ;read first sector
            bcs     140$
            lda     dirBuf+21       ;VLIR file?
            beq     130$
            jsr     readVlir
            jsr     rstrDrv         ;restore original drive
            rts
130$        jsr     readSeq
            jsr     rstrDrv         ;restore original drive
140$        rts
;           ================================================
rstrDrv:    php
            lda     drvSave
            cmp     curDrive
            beq     10$
            jsr     SetDevice
            jsr     OpenDisk
10$         plp
            rts
```

```
; ========================================================
;  View text file read from server.
;               pass:       a6,diskBlkBuf (first 254 bytes already read)
;               return:     carry set on error, clear otherwise
;               destroyed:  a1
; ========================================================
viewFile:       LoadB       tooBig,0
                LoadW       r0,textMsg
                jsr         showStat
                MoveW       itemsEnd,a1     ;start of text
10$             ldy         #0
20$             lda         (a6),y
                jsr         canonify
                sta         (a1),y
                iny
                cpy         bytes           ;bytes read
                bne         20$
                tya
                clc
                adc         a1L
                sta         a1L
                lda         a1H
                adc         #0
                sta         a1H
                cmp         #$7f            ;buffer full ($8000 = kernel)
                bne         30$
                LoadB       tooBig,#$ff
                bne         40$
30$             lda         eof
                bne         40$
                LoadW       a6,diskBlkBuf
                jsr         ultRdBlk
                bcc         35$
                rts
35$             stx         eof
                sty         bytes
                tya                         ;could have EOF with bytes
                beq         40$             ;left in disk block buffer
                lda         a6L
                clc
                adc         #2              ;allow for length
                sta         a6L
                lda         a6H
                adc         #0
                sta         a6H
                bra         10$
40$             lda         #0              ;write EOF marker
                tay
                sta         (a1),y
                dec         a1H             ;strip EOT? ('.', $0d, $0a)
                ldy         #$fd
                lda         (a1),y
                cmp         #'.'            ;end of transmission
                bne         50$
                iny
                lda         (a1),y
                cmp         #$0d
                bne         50$
                iny
                lda         (a1),y
                cmp         #$0a
                bne         50$
```

```
                lda     #0              ;write EOF marker
                ldy     #$fd
                sta     (a1),y          ;(replace EOT with null)
                lda     a1L
                clc
                adc     #$fd
                sta     textEnd
                lda     a1H
                adc     #0
                sta     textEnd+1
                bne     60$
50$             inc     a1H             ;restore
                MoveW   a1,textEnd
60$             MoveW   itemsEnd,a0     ;start of text
                jsr     clrStat
                jsr     saveItms        ;save item state
                LoadB   textType,TXT_FILE
                lda     tooBig
                beq     70$
                LoadW   r0,oomDB        ;show out-of-memory dialog
                LoadW   RecoverVector,rstrDone  ;vwTextDB will cover it
                jsr     DoDlgBox
70$             LoadW   r0,vwTextDB
                LoadW   RecoverVector,rstrTDlg
                jsr     DoDlgBox        ;calls showText via DB_USR_ROUT
                sec                     ;so caller doesn't show "download OK" dialog
                lda     r0L             ;return value from vwTextDB
                rts
; ============================================================
;  Force character into ASCII range ($20-$7f) or GEOS go boom!
;               pass:   .A, character to canonify
;               return: .A, canonified character.
;               note:   Does not use X or Y registers.
; ============================================================
canonify:       cmp     #$20
                bcs     40$
                cmp     #$09            ;tab
                bne     10$
                lda     #' '
                rts
10$             cmp     #$0a            ;line feed
                bne     20$
                rts
20$             cmp     #$0d            ;carriage return
                bne     30$
                rts
30$             lda     #'?'
                rts
40$             cmp     #$80
                bcs     30$
                rts
```

```
; ============================================================
;  Download non-GEOS text or binary file.
; ============================================================
getPlain:       ldx     #30             ;build directory entry
                lda     #0
10$             sta     dirBuf-1,x
                dex
                bne     10$
                sta     sectors
                sta     sectors+1
                lda     itemType
                cmp     #TYP_TEXT
                bne     20$
                lda     #SEQ | $80
                bne     30$
20$             lda     #PRG | $80
30$             sta     dirBuf
                jsr     readSeq2        ;first sector already in diskBlkBuf
                rts                     ;(data sector, not CVT header)
; ============================================================
;  Get name to save file under. A DB_USR_ROUT populates the
;  disk name when saveDB is called.
;               pass:   saveName, default save name
;               return: carry clear if name accepted, otherwise set
;                       with error in .A (including CNCL_ERR)
;               destroyed: a9
; ============================================================
getName:        LoadW   a9,saveName
                lda     #TXT_LN_2_Y
                sec
                sbc     baselineOffset
                sta     askName+2
                LoadW   r0,saveDB
                LoadW   RecoverVector,rstrDone
                jsr     DoDlgBox
                jsr     ckSave
                bcc     getName         ;show again until it's right
                pha
                php
                jsr     rstrDlg
                plp
                pla
                beq     10$             ;OK, proceed
                pha
                php
                jsr     j_close         ;error or download cancelled
                plp
                pla
                rts
10$             clc
                rts
```

```
; ============================================================
;  Determine whether to accept the file save name.
;               return:     carry clear: dialog must be shown again
;                           carry set: proceed if .A = 0, else abort
; ============================================================
ckSave:         lda         r0L
                cmp         #CANCEL
                bne         10$
                sec
                lda         #CNCL_ERR       ;caller should not show error
                rts
;               ================================================
10$             cmp         #DISK
                bne         40$
                LoadW       r0,diskDB       ;change disk
                LoadW       RecoverVector,rstrDone
                jsr         DoDlgBox
                lda         r0L
                cmp         #CANCEL
                bne         20$
                clc                         ;show dialog again
                rts
20$             jsr         OpenDisk
                txa
                bne         30$
                clc                         ;new disk, show dialog again
                rts
30$             sec
                rts
;               ================================================
40$             cmp         #DRIVE          ;handled by doDrv
                bne         90$
                rts                         ;error only
;               ================================================
90$             lda         saveName        ;must be DBGETSTRING
                bne         95$
                LoadW       errMsg,noName
                LoadW       r0,errorDB
                LoadW       RecoverVector,rstrDone
                jsr         DoDlgBox
                clc                         ;try again
                rts
95$             LoadW       r6,saveName
                jsr         FindFile        ;file already exists?
                txa
                beq         110$            ;yes (found)
                cmp         #5              ;FILE_NOT_FOUND (OK)
                bne         100$
                lda         #0
100$            sec
                rts
110$            LoadW       r0,existsDB
                LoadW       RecoverVector,rstrDone
                jsr         DoDlgBox
                lda         r0L
                cmp         #YES
                beq         120$
                clc                         ;show dialog again
                rts
120$            LoadW       r0,saveName
                jsr         DeleteFile
                txa
                bne         130$
                lda         #0              ;OK, proceed
130$            sec
                rts
```

```
; ============================================================
;  dispatch routine for DRIVE icon in saveDB
; ============================================================
doDrv:      lda     #DRIVE          ;custom icon ID
            sta     sysDBData
            ldy     #0              ;drive counter
5$          ldx     curDrive
10$         cpx     #11             ;support all four GEOS drives
            bcc     20$
            ldx     #8              ;wrap around
            bne     30$
20$         inx
30$         iny
            cpy     #4              ;checked four drives?
            bne     40$
            jsr     beep            ;only one drive on system
            rts
40$         lda     driveType-8,x   ;drive present?
            beq     10$
            txa                     ;new drive number
            jsr     SetDevice
            txa
            bne     50$
            jsr     OpenDisk
            txa
            bne     5$              ;no disk in drive
            jsr     showDisk
            lda     #DEF_DB_LEFT    ;print disk name "by hand"
            clc                     ;(since we're still in dialog)
            adc     tellDisk+1      ;in saveDB
            sta     r11L
            lda     #0
            sta     r11H
            lda     #DEF_DB_TOP
            clc
            adc     tellDisk+2
            sta     r1H
            MoveW   a8L,r0L
            jsr     PutString
            rts
50$         sec
            jmp     RstrFrmDialog
; ============================================================
;  DB_USR_ROUT for saveDB; also called from doDrv ("Drive" icon)
; ============================================================
showDisk:   jsr     clrName
            ldx     #a8L
            jsr     GetPtrCurDkName
            ldy     #0
10$         lda     (a8),y
            cmp     #$a0
            beq     20$
            sta     diskName,y
            iny
            cpy     #16
            bne     10$
20$         lda     #0
            sta     diskName,y
            LoadW   a8,diskName     ;for DBVARSTR
            rts
```

```
; ============================================================
;  Copy filename from selector to saveName. Assumes selector not
;  longer than 255 bytes.
;               pass:       selector in a1
; ============================================================
basename:   ldy         #0
10$         lda         (a1),y
            cmp         #9              ;tab (find end of selector)
            beq         20$
            iny
            bne         10$
20$         dey                         ;back up to '/' or start
            beq         30$
            lda         (a1),y
            cmp         #'/'
            bne         20$
            iny
            bne         40$             ;assume branch always
30$         lda         (a1),y
            cmp         #'/'            ;first char. is '/' ?
            bne         40$
            iny
40$         ldx         #0
50$         lda         (a1),y
            cmp         #9              ;tab
            beq         60$
            sta         saveName,x
            iny
            inx
            cpx         #16             ;truncate if too long
            bne         50$
60$         lda         #0
            sta         saveName,x
            rts
; ============================================================
;  Clear disk name from saveDB.
;               pass:       carry set: clear file name, else disk name
; ============================================================
clrName:    lda         #TXT_LN_1_Y
            sec
            sbc         baselineOffset
            clc
            adc         #DEF_DB_TOP
            sta         r2L             ;top
            adc         curHeight
            sta         r2H             ;bottom
            lda         #DEF_DB_LEFT
            clc
            adc         askName+1
            sta         r3L
            lda         #0
            sta         r3H
            ldx         #DEF_DB_RIGHT
            dex
            stx         r4L
            lda         #0
            sta         r4H
            lda         #0              ;clear
            jsr         SetPattern
            jsr         Rectangle
            rts
```

```
; ============================================================
;  Read header sector of CVT file.
;                   return:     carry set on error, clear otherwise
; ============================================================
readHdr:        jsr         clrStat
                LoadW       r0,hdrMsg
                jsr         showStat
                LoadW       a6,fileHeader
                jsr         ultRdBlk        ;second block: GEOS header
                bcs         10$
                cpy         #254
                beq         20$
10$             lda         #HDRLNERR
                sec
                rts
20$             lda         #0
                sta         fileHeader
                lda         #$ff
                sta         fileHeader+1
                LoadW       r2,#254
                LoadW       r6,fileTrScTab
                jsr         BlkAlloc
                txa
                beq         40$
                sec
                rts
40$             lda         fileTrScTab
                sta         dirBuf+19       ;header pointer in dir entry
                sta         r1L
                lda         fileTrScTab+1
                sta         dirBuf+20
                sta         r1H
                LoadW       r4,fileHeader
                jsr         EnterTurbo
                jsr         InitForIO
                jsr         WriteBlock      ;write file header
                txa
                pha
                jsr         DoneWithIO
                pla
                beq         50$
                sec
                rts
50$             clc
                rts
```

```
; ============================================================
;  Read data portion of VLIR file.
; ============================================================
readVlir:       jsr     clrStat
                LoadW   r0,vNdxMsg
                jsr     showStat
                LoadW   a6,vlirBuf
                jsr     ultRdBlk        ;read VLIR index
                bcc     10$
                rts
10$             cpy     #254
                beq     20$
                lda     #VLRLNERR
                sec
                rts
20$             lda     #0
                sta     vlirBuf
                lda     #$ff
                sta     vlirBuf+1
                ldx     #2
40$             stx     vlirPtr
                lda     vlirBuf,x       ;VLIR record present?
                beq     60$
                jsr     setVRec
                bcc     50$
                rts
50$             ldx     vlirPtr
60$             inx
                inx
                bne     40$
                jsr     writeNdx        ;write VLIR index
                bcc     70$
                rts
70$             jsr     writeDir        ;write directory entry
                rts
```

```
; ============================================================
;  Setup and read of VLIR record (calls readRec).
;                 pass:       .A, number of full sectors (fake track pointer)
;                             .X, VLIR index pointer
;                 return:     carry set on error, clear otherwise
;                 destroyed:  a1,a2
; ============================================================
setVRec:        pha
                txa                             ;index offset
                clc
                ror     a                       ;convert to VLIR rec. no.
                tax                             ;to display in status message
                dex
                txa
                jsr     byte2asc
                ldx     #0
10$             lda     ascNum,x
                sta     vRecNo,x
                beq     20$
                inx
                bne     10$
20$             LoadW   r0,vRecMsg
                jsr     showStat
                pla
                tay
                dey                             ;full sectors
                sty     a1L
                lda     #0
                sta     a1H
                LoadW   a2,#254
                ldx     #a1L                    ;word (result)
                ldy     #a2L                    ;byte
                jsr     BMult                   ;sectors to bytes
                ldx     vlirPtr
                inx
                lda     vlirBuf,x               ;bytes in last sector
                tay
                dey                             ;bytes to allocate - 1
                tya
                sta     bytes
                clc
                adc     a1L
                sta     r2L
                lda     a1H
                adc     #0
                sta     r2H
                LoadW   r6,fileTrScTab
                jsr     BlkAlloc                ;allocate sectors for VLIR record
                txa
                beq     30$
                sec
                rts
30$             ldx     vlirPtr
                lda     fileTrScTab             ;update VLIR record T/S pointer
                sta     vlirBuf,x
                lda     fileTrScTab+1
                sta     vlirBuf+1,x
                jsr     readRec                 ;read and save a VLIR record
                rts
```

```
; ============================================================
;  Read VLIR record from network and write to disk.
; ============================================================
readRec:        lda     #0
                sta     chainPtr
10$             LoadW   a6,diskBlkBuf
                jsr     ultRdBlk
                bcc     20$
                rts
20$             cpy     #254
                beq     30$
                txa                     ;EOF?
                bne     30$             ;then it's OK
                lda     #VBKLNERR
                sec
                rts
30$             ldx     chainPtr
                lda     fileTrScTab,x
                sta     r1L             ;track
                lda     fileTrScTab+1,x
                sta     r1H             ;sector
                LoadW   r4,diskBlkBuf
                inx
                inx
                stx     chainPtr
                lda     fileTrScTab,x   ;forward pointer
                sta     diskBlkBuf
                lda     fileTrScTab+1,x
                sta     diskBlkBuf+1
                jsr     EnterTurbo
                jsr     InitForIO
                jsr     WriteBlock
                txa
                pha
                jsr     DoneWithIO
                pla
                beq     50$
                sec
                rts
50$             lda     diskBlkBuf      ;next T/S pointer
                bne     10$
                clc
                rts
```

```
; ============================================================
;  Update directory entry and write VLIR index to disk.
; ============================================================
writeNdx:       LoadW   r2,#254
                LoadW   r6,fileTrScTab
                jsr     BlkAlloc
                txa
                beq     10$
                sec
                rts
10$             lda     fileTrScTab     ;track
                sta     r1L
                sta     dirBuf+1
                lda     fileTrScTab+1   ;sector
                sta     r1H
                sta     dirBuf+2
                LoadW   r4,vlirBuf
                jsr     EnterTurbo
                jsr     InitForIO
                jsr     WriteBlock
                txa
                pha
                jsr     DoneWithIO
                pla
                beq     20$
                sec
                rts
20$             clc
                rts
```

```
; ============================================================
;  Read data portion of SEQUENTIAL or non-GEOS file.
; ============================================================
readSeq:    jsr     clrStat
            LoadW   r0,fileMsg
            jsr     showStat
            LoadW   sectors,0
            LoadW   a6,fileHeader   ;temp
            jsr     ultRdBlk        ;read first sector
            stx     eof
            sty     bytes
readSeq2:   LoadW   r2,#254         ;entry point from getPlain
            LoadW   r6,fileTrScTab
            jsr     BlkAlloc
            txa
            beq     20$
            sec
            rts
20$         lda     fileTrScTab
            sta     track
            sta     dirBuf+1
            lda     fileTrScTab+1
            sta     sector
            sta     dirBuf+2
30$         lda     eof             ;did driver read last sector?
            bne     seqEof
            jsr     temp2Blk
            LoadW   a6,fileHeader   ;temp
            jsr     ultRdBlk        ;read next sector
            stx     eof
            sty     bytes
            LoadW   r2,#254
            LoadW   r6,fileTrScTab
            jsr     BlkAlloc
            txa
            beq     50$
            sec
            rts
50$         MoveW   fileTrScTab,diskBlkBuf ;forward T/S pointer
            jsr     wrSeqSec        ;write previous sector
            bcc     60$
            rts
60$         inc     sectors
            bne     70$
            inc     sectors+1
70$         lda     fileTrScTab
            sta     track
            lda     fileTrScTab+1
            sta     sector
            jsr     temp2Blk        ;returns .Z set
            beq     30$             ;read next sector from network
```

```
seqEof:     jsr     temp2Blk        ;returns .Y=0
            sty     diskBlkBuf      ;dummy track pointer
            ldy     bytes
            iny                     ;-1 to last byte, +2 pointer bytes
            sty     diskBlkBuf+1    ;last byte pointer
            jsr     wrSeqSec        ;write last sector
            bcc     10$
            brk                     ;FIXME error handling
10$         inc     sectors
            bne     20$
            inc     sectors+1
20$         lda     itemType        ;non-CVT file?
            cmp     #TYP_TEXT
            beq     30$
            cmp     #TYP_BIN
            bne     40$
30$         lda     sectors         ;yes, fill used block count
            sta     dirBuf+28
            lda     sectors+1
            sta     dirBuf+29
40$         jsr     writeDir
            bcc     50$
            sec
50$         rts
; ============================================================
;  Copy temp buffer to disk blockbuffer (at offset 2).
;           pass:   fileHeader populated with temp data
;           return  disk buffer populated, .Z set
; ============================================================
temp2Blk:   ldy     #2
10$         lda     fileHeader,y
            sta     diskBlkBuf,y
            iny
            bne     10$
            rts     ;returns .Z set
```

```
; ============================================================
;  Write one sector of SEQUENTIAL file.
;                pass:       disk block buffer populated
;                            track, sector: track/sector to write
; ============================================================
wrSeqSec:       MoveB       track,r1L
                MoveB       sector,r1H
                LoadW       r4,diskBlkBuf
                jsr         EnterTurbo
                jsr         InitForIO
                jsr         WriteBlock
                txa
                pha
                jsr         DoneWithIO
                pla
                clc
                beq         10$
                sec
10$             rts
; ============================================================
;  Write directory entry and update BAM.
; ============================================================
writeDir:       LoadB       r10L,0          ;0 means first page (not 1)
                jsr         GetFreeDirBlk   ;returns dir entry pointer in .Y
                txa
                beq         10$
                sec
                rts
10$             jsr         copyName        ;copy entered name to dir
                ldx         #0
                tya
20$             lda         dirBuf,x        ;add directory entry
                sta         diskBlkBuf,y
                iny
                inx
                cpx         #30
                bne         20$
                LoadW       r4,diskBlkBuf   ;r1 holds T&S from GetFreeDirBlk
                jsr         EnterTurbo
                jsr         InitForIO
                jsr         WriteBlock      ;update directory sector
                txa
                pha
                jsr         DoneWithIO
                pla
                beq         40$
30$             sec
                rts
40$             jsr         PutDirHead      ;update BAM
                txa
                bne         30$
50$             clc
                rts
```

```
; ============================================================
;  Copy user-entered name to directory buffer.
; ============================================================
copyName:   ldx     #0
10$         lda     saveName,x
            bne     20$
            cpx     #16
            bne     30$
            rts
20$         sta     dirBuf+3,x
            inx
            cpx     #16
            bne     10$
            rts
30$         lda     #$a0
40$         sta     dirBuf+3,x
            inx
            cpx     #16
            bne     40$
            rts
```

```
; ============================================================
;  Read 254 bytes from Ultimate to fill a disk block buffer (wrapper around
;  ultRead). This is necessary because Ultimate will sometimes deliver less
;  than 254 bytes even though the server has more than that available.All
;  reads are through dataBuf.
;  Disk block buffer (a6) is filled from offset 2 (beyond T/S pointer).
;                  pass:       connection open, socket set
;                              a6, address of disk block buffer to build
;                  return:     .X, non-zero if EOF
;                              .Y, number of bytes read to disk block buffer
; ============================================================
ultRdBlk:       LoadB   blkPtr,2
                ldx     #254
10$             stx     expected
                PushW   a6              ;used by sendCmd
                LoadW   a7,dataBuf
                jsr     j_read
                PopW    a6
                bcc     20$
                rts
20$             stx     blkEof
                txa                     ;EOF?
                bne     40$
                sty     received        ;byte count from j_read
                ldx     #0              ;dataBuf pointer
                ldy     blkPtr          ;starts at 2
30$             lda     dataBuf+2,x
                sta     (a6),y
                inx
                iny
                beq     50$             ;block full (254 bytes)
                cpx     received
                bne     30$
                sty     blkPtr
                lda     expected
                sec
                sbc     received
                tax                     ;no. bytes remaining to be read
                bra     10$
40$             ldy     blkPtr          ;total bytes read
50$             ldx     blkEof
                dey                     ;allow for count
                dey
                clc
                rts
```