```
; ============================================================
;  geoGopherItm: routines for handling gopher items
; ============================================================
.if             Pass1
                .noeqin
                .include    geoGopherSym
                .include    geoGopherMac
                .include    geoGopher.inc
                .include    ultimate.inc
                .eqin
.endif
; ============================================================
;  Read gopher item list from server.
;               return:     Carry set on error, clear otherwise.
;               destroyed:  a9
; ============================================================
getItems:       lda         itemsBuf
                sta         itemsEnd        ;mark buffer as empty
                sta         a0L
                lda         itemsBuf+1
                sta         itemsEnd+1
                sta         a0H
10$             LoadW       a7,dataBuf
                ldx         #254
                jsr         j_read          ;reads <= 254 bytes to (a7)
                bcc         20$
                jsr         j_close
                sec
                rts
20$             sty         count
                txa                         ;EOF?
                bne         50$
                ldx         #2
                ldy         #0
25$             lda         dataBuf,x
                sta         (a0),y
                inx
                iny
                cpy         count
                bne         25$
                lda         a0L
                clc
                adc         count
                sta         a0L
                lda         a0H
                adc         #0
                sta         a0H
                cmp         memDiv          ;used too much memory?
                bne         10$             ;no, read more
;               ================================================
                LoadW       a7,dataBuf      ;could have hit memDiv at EOF
                ldx         #254
                jsr         j_read
                txa                         ;EOF?
                bne         50$             ;yes, ignore memDiv
                ldx         a0H             ;reached memDiv, back off
                dex                         ;mark end of last complete item
                stx         a9H
                ldx         a0L
                stx         a9L
                ldy         #$ff
30$             lda         (a9),y
                cmp         #$0a            ;end of last complete item
                beq         40$
                dey
                cpy         #$ff
                bne         30$
```

```
                dec     a9H
                bne     30$
40$             iny
                tya
                clc
                adc     a9L
                sta     itemsEnd
                lda     a9H
                adc     #0
                sta     itemsEnd+1
                LoadW   r0,oomDB
                LoadW   RecoverVector,rstrDone  ;items will cover it
                jsr     DoDlgBox
                bra     60$
;               ================================================
50$             MoveW   a0,itemsEnd     ;EOF
60$             jsr     j_close
                jsr     normInfo
                clc
                rts
; ============================================================
;  Count number of items in item buffer. Call before doItems.
;               pass:   itemsBuf loaded
;               return: number of items at numItems
;                       on error/end of buffer: carry set, code in .A
; ============================================================
cntItems:       ldx     #0
                stx     numItems
                CmpW    itemsBuf,itemsEnd
                bne     10$
                lda     #BUF_EMTY
                sec
                rts
10$             MoveW   itemsBuf,a0
20$             jsr     nextItem
                bcc     40$
                cmp     #BUF_END        ;last one?
                bne     30$
                inc     numItems
                clc
                rts
30$             sec
                rts
40$             ldy     #0
                lda     (a0),y
                cmp     #'.'            ;end of transmission
                bne     50$
                iny
                lda     (a0),y
                cmp     #$0d
                bne     50$
                iny
                lda     (a0),y
                cmp     #$0a
                bne     50$
                MoveW   a0,itemsEnd
                clc
                rts
50$             inc     numItems
                bne     20$
                brk                     ;defensive
```

```
; ============================================================
;  Display a screen of gopher items.
;               pass:       .X, first item to show
;               return:     carry clear on success, set otherwise
; ============================================================
doItems:        stx         curItem
                stx         topItem
                jsr         doThumb
                jsr         clrItems        ;clear items area
                ldy         #0
10$             tya                         ;clear existing icon pointers
                pha
                asl         a
                tay
                lda         icnPtrs,y
                sta         a1L
                lda         icnPtrs+1,y
                sta         a1H
                ldy         #0
                tya
                sta         (a1),y
                iny
                sta         (a1),y
                pla
                tay
                iny
                cpy         maxItems
                bne         10$
                MoveW       fontLoad,r0     ;small font in memory
                jsr         LoadCharSet
                ldx         curItem
                jsr         getItem
                bcs         30$             ;FIXME error handling
20$             jsr         showItem        ;sets icon
                lda         curItem
                sec
                sbc         topItem
                tax
                inx
                cpx         maxItems
                bcs         30$
                inc         curItem
                ldx         curItem
                cpx         numItems
                bcs         30$
                jsr         nextItem
                bcc         20$             ;FIXME error handling
30$             jsr         UseSystemFont
                ldx         numItems
                dex
                cpx         maxItems
                bcc         40$
```

```
                LoadW   pageDown,itmPgDn
                LoadW   pageUp,itmPgUp
                php
                sei
                LoadW   otherPressVector,chkMouse
                LoadW   topDspch,itmTop  ;enable top/bottom icons
                LoadW   botDspch,itmBot
                plp
                bra     50$
40$             php
                sei
                LoadW   otherPressVector,0
                plp
50$             LoadW   r0,itmIcons
                jsr     DoIcons
                rts
; ============================================================
;  Collapse info items so that consecutive ones are padded with $a0.
;               pass:   a0, address of items buffer
;               return: items buffer normalized
;                       carry set on error, clear otherwise
;               destroyed: a0,a1
; ============================================================
normInfo:       CmpW    itemsBuf,itemsEnd
                bne     10$
                clc
                rts
10$             MoveW   itemsBuf,a0
                lda     #0
                sta     iType           ;boolean
                beq     40$
20$             MoveW   a0,a1           ;save previous item for padPrev
                jsr     nextItem
                bcc     40$
                cmp     #BUF_END
                beq     30$
                sec
                rts
30$             clc
                rts
40$             lda     iType           ;was previous item type info?
                beq     70$
                ldy     #0
                lda     (a0),y
                cmp     #TYP_INFO
                beq     60$
                LoadB   iType,0
                beq     20$
60$             jsr     padPrev
                bra     20$
70$             ldy     #0
                lda     (a0),y          ;item type
                cmp     #TYP_INFO
                bne     20$
                LoadB   iType,$ff       ;set to true
                bne     20$
```

```
; ============================================================
;  Pad previous item (of type info) to merge two of them.
;                   pass:       a1, item to pad with $a0
; ============================================================
padPrev:    ldx     #$a0        ;padding character
            ldy     #1          ;past item type (see below)
10$         lda     (a1),y
            cmp     #9          ;tab
            beq     20$
            iny
            bne     10$
            inc     a1H
            bne     10$
20$         txa
            sta     (a1),y
            iny
            bne     30$
            inc     a1H
30$         lda     (a1),y
            cmp     #$0d
            bne     60$
            txa
            sta     (a1),y      ;carriage return
            iny
            bne     40$
            inc     a1H
40$         sta     (a1),y      ;line feed
            iny
            bne     50$
            inc     a1H
50$         sta     (a1),y      ;next item type
            rts
60$         txa
            sta     (a1),y
            iny
            bne     30$
            inc     a1H
            bne     30$
```

```
; ============================================================
;  Get next info item in a series of collapsed ones.
;                   pass:       a0, address of current info item
;                   return:     carry clear if another found, set otherwise
;                               r0, pointer to next one if found
;                   destroyed:  a1
; ============================================================
nextInfo:       MoveW   a0,a1
                ldy     #0
10$             lda     (a1),y
                cmp     #$a0
                beq     30$
                cmp     #$0d
                bne     20$
                sec
                rts
20$             iny
                bne     10$
                inc     a1H
                bne     10$
30$             iny
                bne     40$
                inc     a1H
40$             lda     (a1),y
                cmp     #$a0
                beq     30$
                tya
                clc
                adc     a1L
                sta     r0L
                lda     a1H
                adc     #0
                sta     r0H
                clc
                rts
```

```
; ========================================================
;  Show a gopher item.
;                pass:     a0, address of gopher item (call getItem)
;                          curItem: number of gopher item
; ========================================================
showItem:       ldy       #0
                lda       (a0),y          ;item type
                ldx       #0
10$             cmp       itmTypes,x
                beq       20$
                inx
                cpx       #NUM_TYPE
                bne       10$
                dex                       ;last item is unknown type
20$             txa
                asl       a
                tax                       ;index into icon address table
                lda       curItem
                sec
                sbc       topItem
                asl       a
                tay                       ;index into icon pointers table
                lda       icnPtrs,y
                sta       a1L
                lda       icnPtrs+1,y
                sta       a1H
                ldy       #0
                lda       icnAddrs,x
                sta       (a1),y
                lda       icnAddrs+1,x
                iny
                sta       (a1),y
;               ================================================
clrText:        lda       curItem
                sec
                sbc       topItem
                tax
                lda       icnYPsns,x      ;clear area where text will go
                sta       r2L
                clc
                adc       #ITEM_HI-4      ;item icon height-1
                sta       r2H
                LoadW     r3,32
                lda       scrlLeft        ;r4 = scrlLeft-4
                sec
                sbc       #4
                sta       r4L
                lda       scrlLeft+1
                sbc       #0
                sta       r4H
                lda       #0              ;clear
                jsr       SetPattern
                PushW     r2
                PushW     r3
                PushW     r4
                jsr       Rectangle
                PopW      r4
                PopW      r3
                PopW      r2
                lda       #$ff            ;solid line
                jsr       FrameRectangle
                jsr       rndText         ;render descriptive text
                rts
```

```
; ============================================================
;  Render gopher item descriptive text, wrapping if necessary.
;                pass:       gopher item address in a0
;                return:     nothing
;                destroyed:  a1, a9
; ============================================================
rndText:        ldy         #0
                lda         (a0),y
                sta         itemType
                lda         a0L
                clc
                adc         #1
                sta         r0L
                lda         a0H
                adc         #0
                sta         r0H             ;point to item display string
                jsr         ckWrap          ;does it fit?
                bcs         30$             ;no, break string
                jsr         saveChar        ;yes, print on first line
                jsr         doStrng1
                jsr         restChar
                lda         itemType
                cmp         #TYP_INFO
                bne         10$
                jsr         nextInfo        ;another collapsed info item?
                bcs         10$             ;no, we're done
                jsr         ckWrap          ;does next one fit?
                bcs         20$             ;no, truncate with ellipsis
                jsr         saveChar        ;yes, print on second line
                jsr         doStrng2
                jsr         restChar
                jsr         nextInfo        ;is there still more?
                bcs         10$             ;no, we're done
                jsr         doDots          ;hope an ellipsis will fit!
10$             rts
20$             jsr         ellipsis
                rts
30$             jsr         doSplit
                jsr         doStrng1        ;print first line
                jsr         unSplit
                jsr         ckWrap          ;does remainder fit?
                bcc         40$
                jsr         ellipsis        ;no, truncate with ellipsis
                rts
40$             jsr         saveChar        ;remainder fits, print on second line
                jsr         doStrng2
                jsr         restChar
                lda         itemType
                cmp         #TYP_INFO
                bne         50$
                jsr         nextInfo        ;another collapsed info item?
                bcs         50$             ;no, we're done
                jsr         doDots          ;hope an ellipsis will fit!
50$             rts
```

```
; ============================================================
;  Split a display line that won't fit in the item display area.
;                   pass:       r0, address of display string
;                               a9L, index of character that wouldn't fit
;                   return:     a9H, replaced character
; ============================================================
doSplit:    ldy     a9L             ;set by ckWrap
10$         lda     (r0),y
            cmp     #' '            ;look for word break
            beq     20$
            dey
            bne     10$
            ldy     a9L             ;no word break, must split
            lda     (r0),y
20$         sta     a9H             ;save character
            sty     a9L
            lda     #0
            sta     (r0),y
            rts
; ============================================================
;  "Un-split" a display line that doSplit has been called on.
;                   pass:       r0, address of display string
;                               a9L, index of character that wouldn't fit
;                               a9H, character to restore
;                   return:     r0 points to remainder of string
; ============================================================
unSplit:    lda     a9H
            ldy     a9L
            sta     (r0),y          ;replace character
            cmp     #' '
            bne     35$
            iny                     ;don't reprint space
35$         tya
            clc
            adc     r0L
            sta     r0L
            lda     #0
            adc     r0H
            sta     r0H
            rts
```

```
; ============================================================
;  Print what fits on second line with an ellipsis.
;               pass:       r0, address of display string
;                           a9L, index to character that won't fit
;               destroyed:  a9H
; ============================================================
ellipsis:       dec         a9L
                dec         a9L
                dec         a9L             ;make room for ellipsis
                ldy         a9L
                lda         (r0),y          ;wont fit, print w/ellipsis
                sta         a9H
                lda         #0
                sta         (r0),y
                jsr         doStrng2
                jsr         doDots
                ldy         a9L
                lda         a9H
                sta         (r0),y          ;replace character
                rts
;               ================================================
doDots:         lda         #'.'
                jsr         SmallPutChar
                lda         #'.'
                jsr         SmallPutChar
                lda         #'.'
                jsr         SmallPutChar
                rts
; ============================================================
;  Check to see if item descriptor (or second part if a break was
;  performed) will fit on one line.
;               pass:       r0, address of display string
;               return:     carry clear if string fits, set otherwise
;                           if set, a9L points to character that didn't fit
;               destroyed:  a9L
; ============================================================
ckWrap:         ldy         #0
                sty         a9L
10$             lda         (r0),y
                cmp         #9              ;tab (end of display string)
                beq         20$
                cmp         #$a0            ;info padding character
                bne         30$
20$             clc                         ;fits on one line
                rts
30$             cpy         #ITEM_WD-1
                bcc         40$
                rts                         ;won't fit on one line
40$             inc         a9L
                ldy         a9L
                bne         10$
                brk                         ;defensive
```

```
; ============================================================
;  Print first string for current gopher item.
;               pass:       r0, string address
;                           curItem set
;               preserved:  r0
; ============================================================
doStrng1:       lda         curItem
                sec
                sbc         topItem
                tax
                lda         icnYPsns,x
                clc
                adc         #7              ;point size
                sta         r1H             ;baseline
                LoadW       r11,ITXT_X
                PushW       r0
                jsr         PutString
                PopW        r0
                rts
; ============================================================
;  Print second string for current gopher item.
;               pass:       r0, string address
;                           r1H, Y position of first string
;               preserved:  r0
; ============================================================
doStrng2:       lda         r1H
                clc
                adc         #8              ;font height + 1
                sta         r1H
                LoadW       r11,ITXT_X
                PushW       r0
                jsr         PutString
                PopW        r0
                rts
; ============================================================
;  Null-terminate display string; save location of replaced character.
;               pass:       r0, display string (or second portion)
;               return:     a8L, index to character that was replaced
;                           a8H, character replaced
; ============================================================
saveChar:       ldy         #0
10$             lda         (r0),y
                cmp         #9              ;tab
                beq         20$
                cmp         #$a0            ;padding character
                beq         20$
                iny
                bne         10$
20$             sty         a8L
                sta         a8H
                lda         #0              ;null-terminate
                sta         (r0),y
                rts
; ============================================================
;  Restore character into null-terminated display string.
;               pass:       r0, display string (or second portion)
;                           a8L, index to character that was replaced
;                           a8H, character that was replaced
; ============================================================
restChar:       ldy         a8L
                lda         a8H
                sta         (r0),y          ;restore
                rts
```

```
; ============================================================
;  Get address of next gopher item.
;                   pass:       current item's address in a0
;                   return:     next item's address in a0
;                               on error/end of buffer: carry set, code in .A
; ============================================================
nextItem:   ldy     #0
10$         lda     (a0),y
            cmp     #$0d            ;CR/LF
            beq     40$
            cmp     #$a0            ;padding for type 'i'
            beq     20$
            iny
            bne     10$
            inc     a0H
            bne     10$
20$         iny
            bne     30$
            inc     a0H
30$         lda     (a0),y          ;walk past padding
            cmp     #$a0
            beq     20$
            tya
            clc                     ;avoid overflow in .Y
            adc     a0L
            sta     a0L
            lda     a0H
            adc     #0
            sta     a0H
            ldy     #0
            beq     10$
40$         iny                     ;line feed
            bne     50$
            inc     a0H
50$         iny                     ;start of next item
            bne     60$
            inc     a0H
60$         tya
            clc
            adc     a0L
            sta     a0L
            lda     #0
            adc     a0H
            sta     a0H
            CmpW    a0,itemsEnd
            bcc     90$
            beq     70$
            lda     #BUF_OVFL
            bne     80$
70$         lda     #BUF_END
80$         sec
90$         rts
```

```
; ============================================================
;  Get address of a given gopher item.
;               pass:       .X, item number (0-based)
;               return:     a0, address of gopher item
;                           on error/end of buffer: carry set, code in .A
; ============================================================
getItem:    CmpW    itemsBuf,itemsEnd
            bne     10$
            lda     #BUF_EMTY
            sec
            rts
10$         MoveW   itemsBuf,a0
            txa
            bne     20$             ;first item?
            clc
            rts
20$         jsr     nextItem
            bcc     30$
            rts
30$         dex
            bne     20$
            clc
            rts
; ============================================================
;  Save item state for text display.
; ============================================================
saveItms:   MoveW   topDspch,itemSave
            MoveW   botDspch,itemSave+2
            MoveW   otherPressVector,itemSave+4
            php
            sei
            LoadW   otherPressVector,0
            plp
            MoveB   numItems,itemSave+6
            MoveB   topItem,itemSave+7
            MoveB   thumbHi,itemSave+8
            MoveB   thumbSav,itemSave+9
            MoveW   pageDown,itemSave+10
            MoveW   pageUp,itemSave+12
            rts
; ============================================================
;  Restore item state after text display.
; ============================================================
restItms:   php
            sei
            MoveW   itemSave,topDspch
            MoveW   itemSave+2,botDspch
            MoveW   itemSave+4,otherPressVector
            plp
            MoveB   itemSave+6,numItems
            MoveB   itemSave+7,topItem
            MoveB   itemSave+8,thumbHi
            MoveB   itemSave+9,thumbSav
            MoveW   itemSave+10,pageDown
            MoveW   itemSave+12,pageUp
            jsr     setGoph
            rts
```

```
; ============================================================
;  Push a gopher location onto the stack (for BACK icon).
;  If stack index is MAX_BACK, move them all down one,
;  discarding the oldest.
;                   pass:       selector address in a0, hostname/port set
;                   destroyed:  a8, a9
; ============================================================
pushBack:       ldx         backNdx
                cpx         #MAX_BACK       ;back stack full?
                bcc         10$
                dec         backNdx
                LoadW       r0,backStak+BACKSIZE  ;source
                LoadW       r1,backStak         ;destination
                LoadW       r2,BACKSIZE*(MAX_BACK-1)  ;bytes to move
                jsr         MoveData            ;discard oldest
10$             lda         backNdx
                asl         a
                tax
                lda         backPtrs,x
                sta         a9L
                lda         backPtrs+1,x
                sta         a9H
                ldy         #0
20$             lda         hostname,y
                sta         (a9),y
                beq         30$
                iny
                bne         20$
30$             iny
                lda         itemType
                sta         (a9),y
                iny
                lda         #0                  ;placeholder for topItem
                sta         (a9),y
;               ================================================
;               copy current topItem to previously pushed location
;               ================================================
                ldx         backNdx
                beq         35$
                dex
                txa
                asl         a
                tax
                lda         backPtrs,x
                sta         a8L
                lda         backPtrs+1,x
                sta         a8H
                tya
                pha
                ldy         #0
32$             lda         (a8),y
                beq         34$                 ;past hostname
                iny
                bne         32$
34$             iny                             ;past null
                iny                             ;past item type
                lda         topItem
                sta         (a8),y
                pla
                tay
```

```
35$     iny                     ;to selector
        sty     a8L             ;back stack index
        ldy     #0
        sty     a8H             ;selector index
40$     lda     (a0),y          ;selector
        php
        iny
        sty     a8H
        ldy     a8L
        sta     (a9),y
        inc     a8L
        bne     50$
        inc     a9H
50$     plp
        beq     60$             ;end of selector?
        ldy     a8H             ;selector index
        bne     40$
60$     ldx     #0
        ldy     a8L             ;back stack index
70$     lda     port,x
        sta     (a9),y
        beq     80$
        inx
        iny
        bne     70$
        inc     a9H
        bne     70$
80$     inc     backNdx
        jsr     enBack          ;enable BACK icon
        jsr     enHome          ;enable HOME icon
        rts
```

```
; ============================================================
;  Pop a gopher location from the stack (BACK icon handler).
;               return:     selector address in a0, hostname/port set
;               destroyed:  a9
; ============================================================
popBack:    ldx     backNdx
            bne     10$
            brk                     ;shouldn't happen!
10$         dex                     ;where next push will go
            stx     backNdx         ;(overwriting current location)
            dex                     ;to previous location
            txa
            asl     a
            tax
            lda     backPtrs,x
            sta     a9L
            lda     backPtrs+1,x
            sta     a9H
            MoveW   a9,a1           ;for newHost
            ldy     #0
20$         lda     (a9),y          ;skip hostname
            beq     30$
            iny
            bne     20$
30$         iny
            lda     (a9),y
            sta     itemType
            iny
            lda     (a9),y
            sta     topBack         ;to restore topItem
            iny
            tya
            clc
            adc     a9L
            sta     a0L
            lda     a9H
            adc     #0
            sta     a0H             ;selector address to a0
40$         lda     (a9),y          ;find end of selector
            beq     50$
            iny
            bne     40$
            inc     a9H
            bne     40$
50$         iny
            ldx     #0
60$         lda     (a9),y          ;port
            beq     70$
            sta     port,x
            inx
            iny
            bne     60$
            inc     a9H
            bne     60$
70$         ldx     backNdx         ;popped last one?
            dex
            bne     80$
            jsr     disBack         ;disable BACK icon
80$         LoadB   popping,$ff     ;avoid tunnel of mirrors
            jmp     tySelect
```