```
; ============================================================
;  geoGopherUlt (Ultimate network driver for geoGopher)
;  A jump table is provided for the high-level geoGopher API; these routines
;  call into low-level Ultimate code.
; ============================================================
.if             Pass1
                .include    geoGopherSym
                .include    geoGopherMac
                .include    ultimate.inc
.endif
.psect
; ============================================================
;  jump table
; ============================================================
                jmp         ultInit
                jmp         ultProbe
                jmp         ultConn
                jmp         ultRead
                jmp         ultWrite
                jmp         ultClose
; ============================================================
;  Set fontLoad address.
; ============================================================
ultInit:        LoadW       fontLoad,drvEnd  ;font loads after this driver
                rts
; ============================================================
;  Probe for presence of an Ultimate device. If found, get ID string and
;  network addresses.
;               return:     carry clear if device present, set otherwise
; ============================================================
ultProbe:       jsr         probe            ;device present?
                bcc         10$
                rts
10$             jsr         reset
                jsr         identify         ;null-terminates data
                ldx         #0
20$             lda         dataBuf,x
                sta         ultId,x
                beq         30$
                inx
                bne         20$
30$             jsr         getip
                LoadW       a6,dataBuf
                LoadW       a7,ipAddr
                jsr         convAddr
                LoadW       a6,dataBuf+4
                LoadW       a7,netmask
                jsr         convAddr
                LoadW       a6,dataBuf+8
                LoadW       a7,gateway
                jsr         convAddr
                clc
                rts
; ============================================================
;  Convert binary network address to dotted quad.
;               pass:       a6, address of binary data
;                           a7, address of dotted quad
; ============================================================
convAddr:       ldy         #3
10$             lda         (a6),y
                sta         binip,y
                dey
                bpl         10$
                jsr         bindot
                ldy         #0
20$             lda         address,y
                sta         (a7),y
```

```
                beq     30$
                iny
                bne     20$
30$             rts
; ============================================================
;  Convert binary IP to dotted quad.
;               pass:   binary IP at binip
;               return: dotted quad at address
; ============================================================
bindot:         ldx     #0
                ldy     #0
10$             tya
                pha
                lda     binip,x
                jsr     byte2asc        ;trashes .Y
                pla
                tay
                txa
                pha
                ldx     #0
20$             lda     ascNum,x
                beq     30$
                sta     address,y
                inx
                iny
                bne     20$
30$             pla
                tax
                inx
                cpx     #4
                beq     40$
                lda     #'.'
                sta     address,y
                iny
                bne     10$
40$             lda     #0
                sta     address,y
                rts
; ============================================================
;  Connect to gopher host.
;               pass:   hostname, port populated
;               return: carry set on error, clear otherwise
; ============================================================
ultConn:        jsr     port2bin        ;ASCII to binary in command
                lda     #TCPCONN
                ldx     connport
                ldy     connport+1      ;now redundant (port2bin)
                jsr     connect
                bcs     20$
                sta     socket
                lda     statBuf
                cmp     #'0'            ;error code?
                bne     10$
                lda     statBuf+1
                cmp     #'0'
                bne     10$
                clc
                rts
10$             sec
20$             rts
```

```
; ============================================================
;  Read up to 254 bytes from Ultimate.
;                pass:     connection open, socket set
;                          .X, max. no. bytes to read (<= 254)
;                          a7, address of data buffer
;                return:   carry set on error, clear otherwise
;                          (a7), bytes read (first two bytes are length)
;                          .X, non-zero on EOF
;                          .Y, no. bytes read (<= 254)
; ============================================================
ultRead:        lda       socket
                ldy       #0              ;high byte of no. bytes to read
                jsr       sockrd          ;returns bytes read in .Y
                bcs       20$
                lda       statBuf
                cmp       #'0'
                bne       20$             ;error
                lda       statBuf+1
                cmp       #'2'            ;02,NO DATA (yet)
                beq       ultRead         ;retry
                cmp       #'1'            ;01,CONNECTION CLOSED
                bne       10$
                ldx       #$ff            ;EOF marker
                bne       30$
10$             cmp       #'0'
                bne       20$             ;error
                ldx       #0              ;EOF
                tya
                bne       30$
                ldx       #$ff            ;nothing received
                bne       30$
20$             sec
                rts
30$             clc
                rts
; ============================================================
;  Write data to server (no more than 255 bytes).
;                pass:     server connection must be open
;                          a7, address of data to send
;                          .A, socket number
;                          .X, number of bytes to send (max. 254)
;                return:   carry set on error, clear otherwise
; ============================================================
ultWrite:       jsr       sockwr
                rts
; ============================================================
;  Close a network connection.
; ============================================================
ultClose:       lda       socket
                jsr       sockcls
                rts
```

```
; ============================================================
;                LOW-LEVEL ULTIMATE NETWORK API STARTS HERE
; ============================================================
;  probe (check for  Ultimate device)
;                return:     carry clear if device present, set otherwise
; ============================================================
probe:          jsr         enableIO
                lda         ID_REG
                jsr         restoreIO
                cmp         #$c9            ;default read value
                beq         10$
                sec
                rts
10$             clc
                rts
; ============================================================
;  reset (abort pending commands and clear error flag)
; ============================================================
reset:          jsr         enableIO
                lda         #ABRT           ;abort pending command
                sta         CTRL_REG
10$             lda         STA_REG
                and         #ABRT_PND
                bne         10$
                lda         #CLR_ERR        ;clear error status
                sta         CTRL_REG
20$             lda         STA_REG
                and         #ERR
                bne         20$
                jsr         restoreIO
                rts
; ============================================================
;  identify (returns Ultimate ID string at dataBuf)
; ============================================================
identify:       LoadW       a6,cmdid
                jsr         sendcmd
                LoadW       a7,dataBuf
                jsr         readdata        ;returns bytes read in .Y
                iny                         ;two bytes for count
                iny
                iny                         ;past data
                lda         #0
                sta         dataBuf,y       ;null-terminate
                jsr         readstat
                jsr         accept
                rts
; ============================================================
;  getip (get Ultimate's IP address)
;                return:     dataBuf: IP address, netmask, gateway
;                            (4 bytes binary for each)
; ============================================================
getip:          LoadW       a6,cmdgetip
                jsr         sendcmd
                LoadW       a7,dataBuf
                jsr         readdata
                jsr         readstat
                jsr         accept
                rts
```

```
; ============================================================
;  connect (connect to host)
;                   pass:       hostname populated
;                               .A TCPCONN/UDPCONN
;                               .X/.Y port number
;                   return:     carry set on failure, clear otherwise
;                               on success, socket no. in .A
; ============================================================
connect:    sta     conntype
            stx     connport
            sty     connport+1
            ldy     #0
10$         lda     hostname,y
            beq     20$
            sta     connhost,y      ;in command
            iny
            bne     10$
20$         iny                     ;target
            iny                     ;conntype
            iny                     ;port
            iny
            sty     cmdconn         ;no. bytes in cmd
            LoadW   a6,cmdconn
            jsr     sendcmd
            php                     ;carry set on error
            LoadW   a7,dataBuf
            jsr     readdata
            jsr     readstat
            jsr     accept
            lda     dataBuf         ;socket no.
            plp
            rts
; ============================================================
;  sockwr (write to open socket)
;                   pass:       .A socket
;                               .X number of bytes to send (max. 254)
;                               a7, address of data to send
;                   return:     carry set if sendcmd failed, clear otherwise
;                   destroyed:  a7
; ============================================================
sockwr:     sta     cmdsckwr+3      ;socket
            LoadW   a6,cmdsckwr
            jsr     sendcmd         ;also sends data from (a7)
            php                     ;save carry
            LoadW   a7,dataBuf
            jsr     readdata        ;not needed?
            jsr     readstat
            jsr     accept
            plp                     ;we only care about sendcmd
            rts
```

```
; ============================================================
;  sockrd (read from open socket)
;  Note that ultRead always passes a length of 254.
;                   pass:       a7, address of data buffer
;                               .A socket
;                               .X/.Y length of data to read
;                   return:     carry set on error, clear otherwise
;                               (a7), bytes read (first two bytes are length)
;                               .Y, no. bytes read (<= 254)
; ============================================================
sockrd:         sta     cmdsckrd+3
                stx     cmdsckrd+4
                sty     cmdsckrd+5
                LoadW   a6,cmdsckrd
                jsr     sendcmd
                php                     ;save carry flag
                jsr     readdata
                jsr     readstat        ;trashes .X
                jsr     accept
                plp                     ;we only care about sendcmd
                rts
; ============================================================
;  sockcls (close socket)
;                   pass:       .A, socket
; ============================================================
sockcls:        sta     cmdsckcl+3
                LoadW   a6,cmdsckcl
                jsr     sendcmd
                LoadW   a7,dataBuf
                jsr     readdata
                jsr     readstat
                jsr     accept
                rts
; ============================================================
;  readdata (read data from network connection)
;                   pass:       a7, address to read data
;                   return:     data at (a7); first two bytes are length
;                               .Y, no. bytes read (<= 254)
; ============================================================
readdata:       jsr     enableIO
                PushW   a7
                ldy     #0
10$             lda     STA_REG
                and     #DAT_AVL
                beq     20$
                lda     RSP_DREG        ;read data
                sta     (a7),y
                iny
                bne     10$
                inc     a7H             ;happens on data byte 254
                bne     10$
20$             dey
                dey                     ;for count at start of buffer
                jsr     restoreIO
                PopW    a7
                rts
```

```
; ============================================================
;  readstat (populates statBuf)
;                   pass:       carry clear to wait for status available
;                   return:     statBuf populated (null-terminated)
;                               .Y preserved
; ============================================================
readstat:       jsr         enableIO
                ldx         #0
                bcc         20$
10$             lda         STA_REG
                and         #STA_AVL
                beq         10$
20$             lda         STA_REG
                and         #STA_AVL
                beq         30$
                lda         STA_DREG
                sta         statBuf,x
                inx
                bne         20$
30$             lda         #0
                sta         statBuf,x
                jsr         restoreIO
                rts
; ============================================================
;  accept (signal acceptance of data)
; ============================================================
accept:         jsr         enableIO
                lda         #ACC_DATA
                sta         CTRL_REG
10$             lda         STA_REG
                and         #DATA_ACC
                bne         10$
                jsr         restoreIO
                rts
```

```
; ============================================================
;  sendcmd (send command to Ultimate)
; ============================================================
;               pass:   a6, address of command
;                       (first byte is number of bytes in command)
;                       if writing to socket:
;                         a7, address of data buffer
;                         .X, number of bytes to send (max. 254)
;               return: carry set on failure, clear otherwise
;                       a7 preserved
; ============================================================
sendcmd:    txa                     ;for SCKWRITE
            pha
            jsr     enableIO
            lda     STA_REG
            and     #ERR
            beq     20$
            lda     #CLR_ERR
            sta     CTRL_REG
10$         lda     STA_REG
            and     #ERR
            bne     10$
20$         lda     STA_REG
            and     #STA_BITS
            bne     20$
            ldy     #0
            lda     (a6),y
            tax                     ;number of bytes in command
wrcmd:      iny
            lda     (a6),y
            sta     CMD_DREG
            dex
            bne     wrcmd
            pla
            tax                     ;for SCKWRITE
            ldy     #2
            lda     (a6),y          ;command
            cmp     #SCKWRITE
            bne     cmddone
            ldy     #0
snddata:    lda     (a7),y          ;write socket data
            sta     CMD_DREG
            dex
            beq     cmddone
            iny
            bne     snddata
cmddone:    lda     #PUSH_CMD
            sta     CTRL_REG
            lda     STA_REG         ;error check
            and     #ERR
            beq     90$
            lda     #CLR_ERR
            sta     CTRL_REG
85$         lda     STA_REG
            and     #ERR
            bne     85$
            sec
            bcs     100$
90$         lda     STA_REG
            and     #DAT_LAST
            beq     90$
            lda     STA_REG
            and     #CMD_BUSY
            bne     90$             ;thanks, Leif!
            clc
100$        jsr     restoreIO
            rts
```

```
; ============================================================
;  Convert port number from ASCII at port to binary at connport.
; ============================================================
port2bin:       ldx     #0
                lda     port,x
                and     #$0f
                sta     connport
                stx     connport+1
10$             inx
                lda     port,x
                beq     20$
                and     #$0f
                pha                     ;save next digit
                asl     connport
                rol     connport+1      ;times two
                lda     connport
                ldy     connport+1      ;save times two
                asl     connport
                rol     connport+1      ;times four
                asl     connport
                rol     connport+1      ;times eight
                clc
                adc     connport
                sta     connport
                tya
                adc     connport+1
                sta     connport+1      ;plus two equals 10
                pla
                clc
                adc     connport
                sta     connport
                lda     connport+1
                adc     #0
                sta     connport+1      ;add next digit
                bra     10$
20$             rts
; ============================================================
;  Enable access to I/O bank (or restore previous setting).
;               pass:   nothing
;               return: .A and status register preserved
;  Note: these routines are not re-entrant; enableIO saves the current value
;  of $01 and restoreIO restores it.
; ============================================================
enableIO:       php
                sei
                pha
                lda     $01
                sta     ioSave          ;save memory configuration
                and     #$f8
                ora     #$05            ;bank in I/O
                sta     $01
                pla
                plp
                rts
restoreIO:      php
                sei
                pha
                lda     ioSave
                sta     $01             ;restore previous state
                pla
                plp
                rts
```

```
; ============================================================
ioSave:         .byte       0
cmdid:          .byte       $02,TGT_DOS1,IDENT
cmdgetip:       .byte       $03,TGT_NET,GETIPADR,0
cmdconn:        .byte       0,TGT_NET
conntype:       .byte       0                   ;TCP or UDP
connport:       .word       0
connhost:       .block      128
cmdsckwr:       .byte       3,TGT_NET,SCKWRITE,0
cmdsckrd:       .byte       5,TGT_NET,SCKREAD,0,0,0
cmdsckcl:       .byte       3,TGT_NET,SCKCLOSE,0
binip:          .block      4
address:        .block      16
drvEnd:         .byte       0                   ;marker, must come last
```