P.P. Load, save

```
initializing printer for ibm tabs

                initialized!
c1      c2      c3      c4      c5      c6      c7
done!
;
;written 10-19-84 by Jesus and Will IV
;
;

;it might be noted that code.src is "instxt" into this


;[[ used for task #s
;(( used for diagnostics



        instxt  /publics.src/
;maxtask        ;# of next task avail (max+1)
        public  maxtask
        public  nextask
        public  taction
        public  saction
        public  STATALM,ALARMED,JUMPOUT,BEDTIME,ALARMTIME
        public  DAILY,ROUTINE,MORNSIZ,MORNING,ETIME,WPRORDER
        PUBLIC  EVENSIZ,EVENING
        public  OFF,ON
        public  THIRSTY,HUNGRY,POTTY
        public  THIRSTCNT,HUNCNT
        public  COLEV   ;0=no h2o
        public  FOOD    ;0=no,non-0 = yes
        PUBLIC  LUNCH,DINNER
        PUBLIC  STARTSICK,SICK,SICKINC
        PUBLIC  INBED,MOOD
        PUBLIC  HEADOF2,FREE
        PUBLIC  LOADMOOD
        PUBLIC  SUNUP,SUNDOWN,TSTIOR0,UNSPEAK,SPEECH
        PUBLIC  INTFLAG,DFLAGS
        public  FLOAD,MOVEPAGES,ONKERNAL,OFFKERNAL
        public  DAY


trg     fcb     0


nextask
;this routine determines what task is done next
        lda     #0              ;set the task and subtask pointers to zero
        sta     taction
        sta     saction

;this routine decides what task will be done next by PP

;lets see if he's just finished hybernation
        lda     HYBERNATING     ;is he currently hybernating?
        bpl     not             ;if so then not done hybernating,so jump

        lda     INTFLAG         ;is hybernation interrupt pending?
```

```
        bne     not             ;if so then not done hybernating, so jump

;he is in hybernation now

        lda     HYBERNATED      ;set his hybernation status
        ora     #$80
        sta     HYBERNATED

        lda     HYBERNATING     ;clear his "in the process of hybernating" fl
        and     #$7f
        sta     HYBERNATING

        jsr     HYB             ;actually do the hybernation to the disk (sub is
        jmp     enxtask
not

;lets see if a particular task is being requested
        lda     trq     ;this is used by code.src (ctrl-g) and also in this file
        beq     nope
;yes, a task is requested, lets do it
        ldx     #0              ;first zero the task request so it wont keep happ
        stx     trq
;then store the task # requested in taction
        beq     doregg  ;unc jmp        setask
nope
;not a special request, do normal task
;===============================================
;===============================================
;=============       NEEDS       ===============
;===============================================
;===============================================
;NEEDS include hunger,thirst,potty,bed,lunch,dinner.

;needs for hunger and thirst (2-26-85)

        lda     THIRSTY         ;is PP thirsty?
        beq     nothrsty        ;nope, not at all
        cmp     #1              ;yes, but how much?
        beq     litlebit
;he's very thirsty, maybe even sick - if theres h2o he's going there
        lda     COLEV           ;look at cooler level
        bne     gotodrnk        ;if not empty, go drink!
        lda     THIRSTCNT       ;if no H2O then go back every now and then...
        cmp     #2              ;.. to look real.
        bne     nothrsty
        dec     THIRSTCNT       ;dec CNT so that he wont keep goin while=2
        bne     gotodrnk        ;unc
litlebit
;he's a little thirsty, he will get a drink maybe
        jsr     getran
        cmp     #$45
        bne     nothrsty
        lda     random+1
        cmp     #$23
        bcs     nothrsty
gotodrnk
        lda     #$1d    ;[[     drink(only) even if no water!!!
        bne     chkbed  ;unc
nothrsty

;check for hungry
```

```
        lda     HUNGRY  ;is PP hungry?
        beq     nohngry ;nope, not at all!
        cmp     #1      ;yes, but how much?
        beq     ltlebit
;he's very hungry, maybe even sick - if theres food he's going there

        lda     FOOD            ;is there food?
        bne     gotoeat         ;yes, go get it!!!
        lda     HUNCNT  ;if not, then go back every now and then to look real
        cmp     #2
        bne     nohngry
        dec     HUNCNT  ;so that he wont do it for whole time while CNT=2
        bne     gotoeat         ;unc
ltlebit
;he's a little hungry, he will eat maybe
        jsr     getran
        cmp     #$45
        bne     nohngry
        lda     random+1
        cmp     #$23
        bcs     nohngry
gotoeat
        lda     #$4a    ;[[     snack(only) even if no food available
chkbed
;if PP's in bed, get PP out and queue up the eat or drink task
        ldx     INBED   ;is PP in bed?
        beq     doregg  ;no, so just go do the task
        sta     trq     ;yes, PP's in bed, queue up the task and get out first
        lda     ##$if   ;[[ get out of bed
doregg
;doregg only allowed to jmp setask because it is used for that above and below
        jmp     setask
nohngry

;check for time to go potty
        lda     POTTY           ;does PP have to go?
        beq     nopotty         ;nope, not at all
;he's very full, he's going there
;go potty task sets POTTY=0 when he goes
        lda     #$04    ;[[     potty
        bne     chkbed  ;unc
nopotty


;=============================================================================
;=============================================================================
;=====================    BEDTIME FOR BONZO    ===============================
;=============================================================================
;=============================================================================

        jsr     TSTIORQ ;if there a interrupt pending, get out of bed
        beq     noint
        lda     INBED
        bne     getout  ;if in bed, get out
noint
;needs to go to bed if sick or sleeping
        ldy     SICK
        ldx     DAILY   ;if sleeping, goto bed
        beq     gobed
;if not sleeping, then check SICK as to whether he should goto bed or not
        cpy     #3      ;PP only goes to bed when sick is 3 or 4
        bcc     nobed
gobed
;goto bed if not already there
```

```
        lda     INBED   ;is PP in bed already?
        bne     already ;yes so don't set in
        lda     #$1e    ;[[ goto bed (sick or sleep)
        bne     doregg  ;unc jmp setask
already
        txa     ;lda    DAILY
        beq     chkd    ;is PP in sleep part of day if so then continue
        jmp     enxtask ;else PP is sick, so PP won't do anything but lay in bed
nobed
;get out of bed if there and not sick (unless sleeping of course)
        lda     INBED   ;is PP in bed?
        beq     chkd    ;if not then don't call task to get him out again
getout
        lda     #$1f    ;[[ get out of bed now
        bne     doregg  ;unc    jmp     setask
chkd
;PP's "needs" to eat lunch and dinner on schedule (habits?!) (3-5-85)

        lda     hour            ;what hour of the day is it?
        cmp     LUNCH           ;is it lunch time?
        bne     nolunch
        lda     lunc            ;has PP ate his lunch already today?
        beq     nolunch
;it's lunch time and PP has not ate yet, so now he'll eat
        lda     #0              ;clear flag so we know he is going to eat
        sta     lunc
        lda     #$4c            ;[[lunch
        bne     settask ;unc    jmp     setask
lunc    fcb     1       ;0=ate lunch this hour,1=waiting to eat lunch
nolunch
        sta     lunc    ;when not in hour, reset status

        lda     hour    ;is it dinner hour?
        cmp     DINNER
        bne     nodin
        lda     dindin  ;if so, did PP eat dinner in this hour yet?
        beq     nodin
        lda     #0      ;if not, then set ate status and so eat!
        sta     dindin
        lda     #$01            ;[[ dinner
settask
;settask can only jump to setask because it is used above
        jmp     setask
dindin  fcb     1       ;0=ate dinner this hour,1=waiting to eat dinner
nodin
        sta     dindin  ;when not in hour, reset status

;=============================================================
;=============================================================
;=============================      DESIRES      =============
;=============================================================
;=============================================================
;DESIRES are the part of PP that wants to do things if his
;       needs are already satisfied (similar to MASLOW'S hierarchy)
;       PP's desires are determined by the time of the day.
;       PP's day is split into 3 parts, each about 6 hours long
;       and PP's personality dictates which of the 3 parts of the
;       day is used for working,playing and relaxing. On Saturday,
;       PP will play instead of working in his normal work part of
;       the day and on Sunday PP will substitute relaxing for his
;       work portion of the day.
;       PP's desires include also his desire to get ready for
```

```
;       bed (evening routine) and also his morning routine. What
;       exactly is contained in his morning and evening routines
;       and the order in which these tasks are done
;       is determined by his personality (when the present is unwrapped).

;DFLAGS=
;$81 to flag waiting for morning routine
;$80 to flag in morning routine
;$41 to flag waiting for evening routine
;$40 to flag in evening routine
;$00 to flag no routine waiting (in regular day)

        lda     DAILY   ;is PP in sleep mode?
        beq     inday   ;if so, then just skip over all this
        lda     DFLAGS  ;else, chk if PP is first time trying to do an AM or PM r
        lsr     a
        bcc     notist  ;it's not first time through, so jump
        lda     DFLAGS  ;it's first time through so clear 1st time bit (bit0)
        and     #$fe
        sta     DFLAGS
        lda     #$ff    ;and set routine list pointer
        sta     ROUTINE

notist

        lda     DFLAGS  ;check if its' time to do morning or evening routine
        beq     inday   ;if zero then we're just in regular parts of the day
        bpl     even    ;if top bit clear then do evening routine
;top bit set: morning routines are to be done
;-----------------------------------------------------------
        inc     ROUTINE ;point to next task in morning routine
        ldx     ROUTINE ;get the index into morning task table
        cpx     MORNSIZ ;see if we've done all the morning tasks
        beq     endmorn
;continue morning
        lda     MORNING,x       ;get this task within morning routine
        bne     settask ;unc
endmorn
        lda     #0      ;morning routine done, reset daily flags
        sta     DFLAGS
        jmp     enxtask

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

even
;this does the evening stuff
        inc     ROUTINE ;point to next task in evening routine
        ldx     ROUTINE ;have we done all of the evening tasks?
        cpx     EVENSIZ
        beq     endeven
;continue evening
        lda     EVENING,x       ;get this task within evening routine
        bne     settask ;unc
endeven
        lda     #0              ;reset alarm clock
        sta     ALARMED
        sta     DAILY           ;set daily=0 (this puts PP in sleep mode)
        sta     DFLAGS          ;reset daily flags so that routines are flagged as
        jmp     enxtask

inday

;ok, do regular tasks
```

```
        lda     DAILY           ;computed jump using daily as index
        asl     a
        tay
        lda     DAILYTBL+1,y
        pha
        lda     DAILYTBL,y
        pha
        ldx     WPRORDER        ;put WPRORDER into x for use in routines
        rts
DAILYTBL
        fdb     sleep-1         ;if daily=0, sleep
        fdb     wpr1-1          ;if daily=1, first work/play/relax period
        fdb     wpr2-1          ;if daily=2, second work/play/relax period
        fdb     wpr3-1          ;if daily=3, third work/play/relax period

;-----------------------------------------------------------------
sleep
;while in this routine he is to be in bed sleeping.
;this will take care of dreaming, alarm clock going off and
; going into the next DAILY to set him out of bed.

        lda     INBED   ;is PP in bed (0=outofbed, non-0=inbed)
        beq     nodream ;if out of bed, don't dream
        lda     attime  ;dream only every once and a long while
        bne     nodream
        jsr     getran  ;should he dream some?
        cmp     #$f0
        bcc     nodream
        lda     SPEECH  ;not if PP's already dreamin
        bne     nodream
        jsr     DREAM   ;do a dream!
nodream
        jmp     enxtask
;-----------------------------------------------------------------


wprtbl1
;indexed by WPRORDER to set status of what will be done in first...
;                               ... period (DAILY=1):work,play or relax.
;                       0=work,$80=play,$01=relax
        fcb     $00,$00,$01,$01,$80,$80,$00,$80
wprtbl2
;indexed by WPRORDER to set status of what will be one in second...
;                               ... period (DAILY=2):work,play or relax
;                       0=work,$80=play,$01=relax
        fcb     $10,$80,$00,$80,$10,$00,$10,$00
wprtbl3
;indexed by WPRORDER to set status of what will be done in third...
;                               ... period (DAILY=3):work,play or relax
;                       0=work,$80=play,$01=relax
        fcb     $80,$01,$80,$00,$00,$01,$80,$01


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

wpr1
        lda     wprtbl1,x       ;set what is done in first period (daily=1)...
;                               ...using this PP's personality (WPRORDER)
        jmp     wpr
;-----------------------------------------------------------------
wpr2
        lda     wprtbl2,x       ;set what is done in second period (daily=2)...
```

```
;                                   ...using this PP's personality (WPRORDER)
        jmp     wpr
;-----------------------------------------------------------
wpr3
        lda     wprtbl3,x       ;set what is done in third period (daily=3)...
;                               ...using this PP's personality (WPRORDER)
wpr
;now accum has 0(work),$80(play) or $01(relax)

        bne     notwork
;it is work, but on saturdays and Sundays he plays and relaxes instead:
        lda     DAY     ;what day of the week is it?
        and     #7
        cmp     #5      ;is it Saturday?
        beq     play    ;if so, go play instead of working
        cmp     #6      ;is it Sunday?
        beq     relax   ;if so, go relax instead of working

;is work, pick a work task at random
        jsr     getran  ;pick a random task of 8 for working
        and     #7
        tax
        lda     worktbl,x
        bne     setask  ;unc
worktbl
;these are work tasks
        fcb     $0c     ;[[top closet
        fcb     $13     ;[[clean-up
        fcb     $14     ;[[type on computer
        fcb     $16     ;[[typewritter
        fcb     $17     ;[[file
        fcb     $18     ;[[read news
        fcb     $19     ;[[call on phone
        fcb     $14     ;[[confuser again

notwork
;accum has either $80(play) or $01(relax)

        bmi     play
relax
        jsr     getran  ;pick one of 8 relax tasks
        and     #7
        tax
        lda     relaxtbl,x
        bne     setask  ;unc
relaxtbl
;these are relaxing tasks
        fcb     $05     ;[[watch TV
        fcb     $06     ;[[relax in upstairs chair
        fcb     $07     ;[[put log on fire
        fcb     $09     ;[[listen to stereo
        fcb     $0c     ;[[go thru top closet
        fcb     $10     ;[[turn TV on/off
        fcb     $11     ;[[random lamp
        fcb     $18     ;[[read comics
play
        jsr     getran  ;pick one of 8 play tasks
        and     #7
        tax
        lda     playtbl,x
setask
        sta     task
enxtask
```

```
        rts

playtbl
;these are playing tasks
        fcb     $08     ;[[talk
        fcb     $35     ;[[dancing
        fcb     $0e     ;[[go out front door
        fcb     $10     ;[[turn TV on/off
        fcb     $12     ;[[trun stereo on/off
        fcb     $15     ;[[play piano
        fcb     $19     ;[[talk on phone
        fcb     $35     ;[[dancing again




GET
;get is called once per screen by ALPHA.SRC
;get processes the control keys pressed by the user for hybernation
;       and dehybernation and runs during while PP is hybernated so
;       that the keys can be checked for.

;get also does an "instxt" of code.src so that programmer diagnostics
;       are linked in.


        public  HYBERNATED,DEHYB,HYB
;[[are there any control keys running during hyber that shouldn't be?

;       hyber/dehyber

        lda     reskey  ;set key pressed on keyboard
        cmp     #4      ;check for ctrl-d for dehybernate
        bne     nodhyber
;user requests dehybernation sir!
        lda     HYBERNATED      ;is PP already hybernated?
        bpl     nodhyber        ;cant do it if its already done
;dehybernate PP and check if he was hybernated correctly last time
        jsr     DEHYB   ;this routine is in INIT.src, it's gets hyber data fr
        bcc     nodhyber        ;was he hybernated when used last?
;error, he wasnt hybernated correctly before - make him sick
        jsr     STARTSICK       ;start off his sickness,but...
        lda     #0              ;make him getting better and...
        sta     SICKINC
        lda     #4              ;make him as sick as he can be
        sta     SICK
nodhyber


;now check for hybernation


        lda     reskey  ;what was last key pressed?
        cmp     #8      ;was it ctrl-h for hybernate?
```

```
        bne     nosiree
;user requests hybernation sir!
        lda     HYBERNATED      ;is PP already hybernated?
        bmi     nosiree ;cant if PP already is!
        lda     HYBERNATING     ;is PP currently doing hybernation?
        bmi     nosiree ;cant if he curently is doing it!!!
;hybernate him
        lda     HYBERNATING     ;set "currently hybernating" flag
        ora     #$80
        sta     HYBERNATING
        lda     #$ff
        sta     INTFLAG         ;set hybernation interrupt

;       lda     #$21    ;CC hybernate
;       sta     task
;       lda     #0
;       sta     taction
nosiree




;this is being inserted to do programmer diagnostics:
        instxt  /b:code.src/    ;put in all the code to be taken out for producti
        rts

;======== subs ======== subroutines ==========

LOADMOOD
;loadmood loads PP's mood (and current calendar) from disk, this routine is
;       called when PP goes out a door becuase sprites need to be
;       turned off when going to disk

        lda     MOMMY           ;don't change mood if mom loaded
        bne     loadcalen

        lda     MOOD            ;check if mood is different than current one loa
        cmp     curmood
        bne     loait
loadcalen
        jmp     loadcal

MOMMY   fcb     0       ;0=no mom, 1= mom loaded

curmood
        fcb     4       ;this is current mood loaded and is initially set to
;                       invalid mood so it will cause mood to be loaded
;                       at first door used

loait
        cmp     #3      ;if mood is invalid then set to zero
        bcc     alrite
        lda     #0
        sta     MOOD
alrite
        sta     curmood
        clc             ;add $41 to get a character(A,B or C) for filename
```

```
        adc     #$41
        sta     fname   ;put into filename

;now the hard part, where does it go ???

;the crunced sprites have a two byte address table in the beginins to point
;to where the masks are so we'll look for the first pointer for heads and
;use it to tell us where to load these new heads in.

        lda     #HEADOF2        ;headof2 * 2 is index into ptr tbl ...
        sta     tptr            ;...for first head sprite mask
        lda     #0
        sta     tptr+1

        asl     tptr            ;multiply it by two because it's and address tbl
        rol     tptr+1

        lda     tptr            ;add FREE to get absolute addr
        clc
        adc     #<FREE
        sta     tptr
        lda     tptr+1
        adc     #>FREE
        sta     tptr+1          ;now tptr,+1 points to the pointer to the...
;                                       ... first head mask

        ldy     #0      ;set ptr but subtract $5000 (addr where they were
        lda     (tptr),y        ;...crunched) to get offset into FREE
        sta     tptr+2
        iny
        lda     (tptr),y
        sec
        sbc     #$50
        sta     tptr+3

        lda     tptr+2  ;now add FREE to get absolute addr of 1st mood head
        clc
        adc     #<FREE
        sta     tptr+2
        sta     modptr

        lda     tptr+3  ;put the address into the load address
        adc     #>FREE
        sta     tptr+3
        sta     modptr+1

        jsr     FLOAD   ;now actually load the masks
        fcb     1+$80   ;1 byte in filename (+$80 to do load at address)
modptr
        fdb     fname   ;this is modified above to point to address to load at
fname
        fcc     /A/     ;this is modified above to either a,b,or c as filename

;now we have to move the pointers for the 15 mood heads from the last 30...
;... bytes of the load to where the pointers are located
;tptr,1 => dest   tptr+2,3 => source

        inc     tptr+3  ;look at end of second 256 bytes of load

        ldx     #29     ;move 15 pointers (30 bytes)
```

```
movem
        txa     ;set index into loaded data
        clc
        adc     #226    ;29->255,28->254 etc.
        tay
        lda     (tptr+2),y      ;set the pointers from the end of the loaded
        pha
        txa
        tay     ;set the index into the pointer table
        pla
        sta     (tptr),y        ;put the pointers here
        dex
        bpl     movem
getback
        rts

loadcal
;now load calendar if it's changed

        lda     DAY     ;has the day of week changed since last load?
        cmp     day
        beq     getback
;yes, load new calendar
        sta     day     ;store day
        pha
        and     #7      ;get day # in ascii($31 through $37)
        clc
        adc     #$31    ;1-7
        sta     daynum  ;put into filename
        pla
        lsr     a       ;now get week # in ascii ($41 through $42)
        lsr     a
        lsr     a
        clc
        adc     #$41    ;"a" or "b"
        sta     weeknum
        jsr     FLOAD   ;load calendar into...
        fcb     3+$80
        fdb     $ff40   ;...buffer under kernal at end of background screen
        fcc     /C/
weeknum
        fcc     /A/
daynum
        fcc     /1/

;and now move down to screen memory

        jsr     UNSPEAK ;can't have talk bubble under calendar

        jsr     OFFKERNAL       ;turn off kernal so we can read the calendar
;                               graphics which where just loaded into ram bel

        ldx     #31     ;4 tiles * 8 bytes per tile = 32 bytes
movbmp
        lda     $ff40,x         ;get calendar grahics
        sta     scrmem+6016,x   ;(18*40+32)*8=6016    ;put on screen
        lda     $ff60,x         ;get more calendar graphics
        sta     scrmem+6336,x   ;+8*40 (next line)    ;put on screen
        lda     $ff80,x         ;get more calendar graphics
        sta     scrmem+6656,x   ;+8*40 (next line)    ;put on screen
        dex
        bpl     movbmp
```

```
        ldx     #3      ;4 tiles worth of colors
movclrs
        lda     $ffa0,x         ;get colors
        sta     colmap+752,x    ;put on screen
        lda     $ffa4,x         ;get colors
        sta     colmap+792,x    ;put on screen
        lda     $ffa8,x         ;get colors
        sta     colmap+832,x    ;put on screen


        lda     $ffac,x         ;get colors
        pha
        lda     $ffb0,x
        pha
        lda     $ffb4,x
        pha
        jsr     ONKERNAL        ;switch back in the i/o to grahics chip
        pla
        sta     $d800+832,x     ;put colors on screen
        pla
        sta     $d800+792,x
        pla
        sta     $d800+752,x
        jsr     OFFKERNAL       ;switch back to seing the buffer below kernal
        dex
        bpl     movclrs

        jsr     ONKERNAL        ;switch kernal back on

        rts

day     fcb     7       ;this is the current calendar loaded
;this # is initially invalid, so cal will always be...
;        loaded 1st call


HYBERNATING

        fcb     0       ;top bit set when hybernating in process

OFF
;turn lights off on the floor given in accum (0top,1,2bottom)
        sta     mine
        tya     ;save registers
        pha
        txa
        pha
        lda     #3      ;set flag to indicate lights off on all floors
        sta     light
        ldx     darkness        ;what is current degree of darkness?
        beq     noreson ;no reason to do it if its daylight
        dex
        stx     mine+1
        lda     mine
        jsr     darken1         ;save the colors first time through
        lda     mine+1          ;if thats it then get out
        beq     noreson
loop
        lda     mine    ;if it should be darker then darken more.get floor #...
        jsr     darken  ;...and darken that floor some more
        dec     mine+1
        bne     loop
noreson
```

```
        pla     ;restore registers
        tax
        pla
        tay
        rts


ON
;turn lights on for the floor given in accum
        sta     mine
        tya             ;save registers
        pha
        txa
        pha
        lda     mine

        sta     light   ;set flag to indicate lights on this floor
        ldx     darkness        ;what is current degree of darkness?
        beq     noreason        ;no reason to do it if its already
        jsr     flooron         ;turn lights on (restore color) on this floor
noreason
        pla             ;restore registers
        tax
        pla
        tay
        rts

mine    rmb     2

;-----------------------------------------------------------
;darken and lighten subs

darkness
        fcb     0       ;0=color,1=1st grey,2=2nd grey
light
        fcb     0       ;3=no lights on,0=floor 1 (top)lights on,...
lolly
        fcb     1       ;loop count

SUNDOWN
;calling this routine darkens the house 1 degree.
;the first call saves the colors in ram and goes to grey tones.
;successive calls will only darken the grey tones until to lowest level.

        jsr     UNSPEAK         ;dont do it during speech because speech
;                                       restores colors when it's done

;only darken floors where lights are off
        lda     #2              ;so in a loop for all 3 floors
        sta     lolly
down
        lda     lolly           ;are lights on for this floor?
        cmp     light
        beq     nohere
;lights not on here, darken it once
        jsr     darken
nohere
        dec     lolly
        bpl     down
        inc     darkness        ;inc degree of darkness
```

```
        rts

SUNUP
;calling this routine lightens the house 1 degree.
;the first calls lighten the grey tones until the last call which
;restores the colors from ram

        jsr     UNSPEAK         ;dont call this during speech
        lda     darkness        ;dont do anything if light already
        beq     goback
        dec     darkness        ;dec degree of darkness
;only lighten floors where lights are off
        lda     #2
        sta     lolly
up
        lda     lolly   ;does this floor have lights on already?
        cmp     light
        beq     nohere2
        jsr     lighten ;no, lighten it
nohere2
        dec     lolly
        bpl     up
goback
        rts

darken
;accum holds floor # to darken one step (0top,1, 2 bot)
        ldx     darkness        ;what is current degree of darkness?
        bne     oneless         ;if not zero then colors have been saved already
;first darkness call, save colors first
darken1
        pha
        jsr     getptrs ;get addr ptrs for the graphics
        lda     #2      ;move all the data for this floor to a buffer
        sta     bump
        ldy     #$ff
loop2
        lda     (tptr),y        ;move the colmap
        sta     (tptr+4),y
        dey
        cpy     #$ff
        bne     loop2
        dec     bump
        beq     dun2
        inc     tptr+1
        inc     tptr+5
        ldy     #$3f
        bne     loop2
dun2
        lda     #$40
        clc
        adc     tptr+4
        sta     tptr+4
        bcc     dun2b
        inc     tptr+5
dun2b

;now save the nibble map but crunch 2 nibbles into one byte
;(if you know what i mean jelly bean!!!)
        lda     #160    ;160 bytes output (320 half bytes input)
        sta     bump
loop5
        ldy     #0
```

```
        lda     (tptr+2),y      ;nibble lo
        and     #$f
        sta     holdit
        iny
        lda     (tptr+2),y      ;nib hi
        asl     a
        asl     a
        asl     a
        asl     a
        ora     holdit
        dey
        sta     (tptr+4),y      ;put in mem

        inc     tptr+4
        bne     sev7
        inc     tptr+5
sev7
        lda     tptr+2
        clc
        adc     #2
        sta     tptr+2
        bcc     eig8
        inc     tptr+3
eig8
        dec     bump
        bne     loop5
        pla
oneless
;now darken the grey tones for the whole floor
;accum has floor # (0=top,1,2=bottom floor)
        cmp     #0      ;is it top floor
        bne     nochnge
        lda     #6      ;for top floor don't change the blue around the house
        sta     darkentbl+6
        lda     #0
nochnge
        pha
        jsr     getptrs ;get addr pointers to data
        lda     #2
        sta     bump
        ldy     #$ff
loop3
        lda     (tptr),y
        pha
        and     #$f
        tax
        lda     darkentbl,x     ;use the darkentbl to decide how to change colors
        sta     holdit
        pla
        lsr     a
        lsr     a
        lsr     a
        lsr     a
        tax
        lda     darkentbl,x
        asl     a
        asl     a
        asl     a
        asl     a
        ora     holdit
        sta     (tptr),y
        lda     (tptr+2),y
        and     #$f
```

```
        tax
        lda     darkentbl,x
        sta     (tptr+2),y
        dey
        cpy     #$ff
        bne     loop3
        inc     tptr+1
        inc     tptr+3
        ldy     #$3f
        dec     bump
        bne     loop3
        pla
        bne     notrealy
        lda     #$c     ;change the table so blue will be changed next time
        sta     darkentbl+6
notrealy
        rts

darkentbl       ;on floor 1 darkentbl+6 changed to 6 and back to $c
;this table tells how to change the colors to make them darker
        fcb     $0,$f,$c,$f,$f,$c,$c,$1,$f,$c,$f,$b,$b,$f,$f,$c
holdit
        rmb     1
bump    rmb     2
temp    rmb     1

getptrs
;acum=floor # (0,1,2)
;THIS WILL GET POINTERS SET UP FOR THIS FLOOR
        asl     a
        sta     temp
        asl     a
        clc
        adc     temp
        adc     #5
        tax
        ldy     #5
getem
        lda     ptrtab,x
        sta     tptr,y
        dex
        dey
        bpl     getem
        rts
ptrtab
;                               ;floor 1 ptrs
        fdb     colmap+40       ;colmap addr
        fdb     $d800+40        ;nibble map addr
        fdb     floor1          ;addr of buffer to store data

        fdb     colmap+40+320   ;floor 2 ptrs
        fdb     $d800+40+320
        fdb     floor2

        fdb     colmap+40+640   ;floor 3 ptrs
        fdb     $d800+40+640
        fdb     floor3
floor1
;buffer for floor 1 colors
        rmb     320     ;colmap
        rmb     160     ;nibmap
floor2
```

```
        rmb     320     ;colmap
        rmb     160     ;nibmap
floor3
        rmb     320     ;colmap
        rmb     160     ;nibmap

lighten
;accum holds floor # to lighten one step
        ldx     darkness        ;what is current degree of darkness?
        bne     onemore
;last lightness call, restore colors
flooron
        jsr     getptrs

        lda     #2
        sta     bump
        ldy     #$ff
hosana
        lda     (tptr+4),y      ;move the colmap back in from buffer
        sta     (tptr),y
        dey
        cpy     #$ff
        bne     hosana
        dec     bump
        beq     prince
        inc     tptr+1
        inc     tptr+5
        ldy     #$3f
        bne     hosana
prince
        lda     #$40
        clc
        adc     tptr+4
        sta     tptr+4
        bcc     christ
        inc     tptr+5
christ

;now restore the nibble map (it is crunched 2 nibbles into one byte)
;(if you know what i mean Jelly bean!!!)

        lda     #160    ;160 bytes input (320 half bytes output)
        sta     bump
almighty
        ldy     #0
        lda     (tptr+4),y      ;lo=first,hi=second
        sta     (tptr+2),y      ;nib hi
        lsr     a
        lsr     a
        lsr     a
        lsr     a
        iny
        sta     (tptr+2),y
        inc     tptr+4
        bne     lordj
        inc     tptr+5
lordj
        lda     tptr+2
        clc
        adc     #2
        sta     tptr+2
        bcc     savior
        inc     tptr+3
```

```
savior
        dec     bump
        bne     almighty

        rts
onemore
;lighten the grey tones for the whole floor


        jsr     getptrs
        lda     #2
        sta     bump
        ldy     #$ff
king
        lda     (tptr),y
        pha
        and     #$f
        tax
        lda     litentbl,x      ;use litentbl to lighten the colors
        sta     holdit
        pla
        lsr     a
        lsr     a
        lsr     a
        lsr     a
        tax
        lda     litentbl,x
        asl     a
        asl     a
        asl     a
        asl     a
        ora     holdit
        sta     (tptr),y
        lda     (tptr+2),y
        and     #$f
        tax
        lda     litentbl,x
        sta     (tptr+2),y
        dey
        cpy     #$ff
        bne     king
        inc     tptr+1
        inc     tptr+3
        ldy     #$3f
        dec     bump
        bne     king

        rts
litentbl
;used to lighten the colors
        fcb     $0,$1,$2,$3,$4,$5,$6,$7,$8,$9,$a,$c,$f,$d,$e,$1

FLOAD
;this routine does a file load using the commodore 64 kernal routines.
;call as follows:
;       JSR     FLOAD
;       fcb     7               ;length of filename
;       fcc     /FILENAM/       ;filename
;       OR
;       JSR     FLOAD
;       fcb     7+$80           ;length of filename with top bit set
```

```
;       fdb     $8000           ;address to load file at
;       fcc     /FILENAM/       ;filename

        lda     #0              ;turn off sprites or this will crash
        sta     senabl

        pla                     ;get data addr-1 (after jsr)
        clc
        adc     #1              ;add 1 to get true addr
        sta     JTPTR           ;lo
        pla
        adc     #0
        sta     JTPTR+1         ;hi
FL2
        ldy     #0              ;set secondary addr for SETLFS in y reg
        lda     (JTPTR),y
        bmi     set             ;if top bit set, load at addr(use 0 for sec addr)
        iny                     ;use secondary addr=1 if top bit clear
set
;y has secondary addr (0 or 1)
        ldx     #8              ;device in x reg
        lda     #3              ;log. file #
        jsr     SETLFS
;bcs    FL2             ;if error, redo

FL3
        ldy     #0              ;now set y reg to point to filename
        lda     (JTPTR),y
        bpl     reg
        iny                     ;if filename size has top bit set then...
        iny                     ;... inc y twice to skip load address
reg
;now y points to filename-1
        and     #$7f            ;and filename length with $7f to clear top bit
        sta     holda
        sty     holdy
        pha                     ;put length of filename on stack

        tya                     ;and add y to JTPTR so x,y point to fname addr
        sec
        adc     JTPTR
        tax                     ;lo fn addr in x reg
        lda     JTPTR+1
        adc     #0
        tay                     ;hi fn addr in y reg
        pla                     ;len of fn in accum
        jsr     SETNAM
;bcs    FL3

FL4
        ldy     #1
        lda     (JTPTR),y       ;set lo addr in case load@addr
        tax
        iny
        lda     (JTPTR),y
        tay                     ;set hi addr in case load@addr
        lda     #0              ;load (not verify)
        JSR     LOAD
        bcs     FL4             ;in case of error do it again

;now restore stack for return
        lda     holda
        clc
```

```
        adc     holdy
        adc     JTPTR
        tay
        lda     JTPTR+1
        adc     #0
        pha                     ;push hi
        tya
        pha                     ;push lo
        rts
holda   fcb     0
holdy   fcb     0

MOVEPAGES
;to call
;       ldy     #num pages (0=256)
;       lda     #source page
;       ldx     #dest   page
;       JSR     MOVEPAGES
;
;example
;       ldy     #2
;       lda     #$b2
;       ldx     #$d5
;       jsr     MOVEPAGES
;this will move data from $b200 through $b3ff to $d500 through $d6ff

        sta     sores+2
        stx     desti+2
        ldx     #0
hulahoop
sores
        lda     $1000,x ;this address is modified above
desti
        sta     $2000,x ;this address is modified above
        inx
        bne     hulahoop
        inc     sores+2
        inc     desti+2
        dey
        bne     hulahoop
        rts


;INIT.src
        instxt  /publics.src/

        public  DEHYB,HYB
        public  HYBERNATED,HYBERDATA,EHYBERDATA
        public  FREE,SINGLE,LOADMOOD,SICK
        PUBLIC  ATTIME,ONKERNAL,OFFKERNAL,saction,taction,task
        public  FLOAD,MOVEPAGES
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;NAME=          INIT
;PURPOSE=       initializes hardware and software at start-up
;REGISTER USAGE=accum,x,y
;REGISTERS ON ENTRY:
;       accum = don't care
;       x =     don't care
;       y =     don't care
;REGISTERS ON EXIT:
```

```
;         accum = scrambled
;         x =     scrambled
;         y =     scrambled
;CONDITION FLAGS ON EXIT: not relevant
;RAM USAGE:
;         NON ZERO PAGE:
;
;         ZERO PAGE:
;
;
;Effects on grahics and sound: sets up graphics chip,loads and initializes
;                  sound code.
;
;


INIT

         lda      #13      ;((
         sta      SINGLE   ;((for diagnostics

         lda      #0       ;use kernal routine to stop "searching and LOADING" me
         jsr      SETMSG

         lda      #$c1     ;no restor/stop combo to return to basic allowed
;((put this back in!!! sta       NMI
;


;put in Todd's audio by loading it from disk files "ralph" and "ralpha"


         jsr      FLOAD            ;load 1st part of sound software
         fcb      5                ;length of filename
         fcc      /RALPH/          ;filename
;move ralph from $b000-$bfff to $d000-$dfff
         jsr      OFFKERNAL                 ;turn $d000 to RAM
         ldy      #$10     ;$10 pages
         lda      #$b0     ;$b000 is source
         ldx      #$d0     ;$d000 is destination
         jsr      MOVEPAGES        ;move the data
         jsr      ONKERNAL         ;turn KERNAL and I/O back on
;load todd's music  part 2
         jsr      FLOAD            ;load 2nd part of sound software
         fcb      6                ;length of filename
         fcc      /RALPHA/
         jsr      MINIT    ;and initialize it !!!

;put in the graphics

         lda      $2a7     ;if loaded by "person" don't load house graphics
;                         ;       (they were already loaded at unwrap)


;$2a7 will contain $a9 if loaded by "person"
;  ($46 ("F") if loaded by autoloader)
;  (0 if loaded in development invironment)
         cmp      #$a9
         beq      nograp


;load colmap first
         jsr      FLOAD
```

```
        fcb     2+$80
        fdb     colmap
        fcc     /CM/

;load bitmap second
        jsr     FLOAD
        fcb     2+$80
        fdb     scrmem+$140
        fcc     /BM/
;load nibblemap last
        jsr     FLOAD
        fcb     2+$80
        fdb     $d800
        fcc     /NM/
nograp
        lda     #14     ;((diagnostics
        sta     SINGLE

;also we'll init the first line in bitmap mem to blank

        lda     #0      ;change 320 bytes to zero
        tay
loup
        sta     scrmem,y
        sta     scrmem+160,y
        iny
        cpy     #160
        bne     loup

;load sprites to FREE memory (FREE is the end of the object code)

        jsr     FLOAD
        fcb     7+$80   ;length of filename (top bit set for load@addr)
        fdb     FREE    ;address to load at
        fcc     /SPRITES/

;now hybernation data,if its the first time (if loaded from "person")
        lda     #15     ;((
        sta     SINGLE

        lda     $2a7    ;don't dehybernate automatically if loaded by autoloader
;$2a7 will contain $a9 if loaded by "person"
; ($46 ("F") if loaded by autoloader)
; (0 if loaded in development invironment)
        cmp     #$46
        beq     notfirst
;yes it is first time, de-hybernate PP
        jsr     DEHYB   ;((do not under any circumstances put this before loa
notfirst
        lda     #16     ;((
        sta     SINGLE

        sei

        lda     #0      ;zero out vertical blank flag
        sta     vbflag
        sta     bc0     ;background color = black
        lda     #blue1  ;border = dark blue
        sta     bordcol
```

```
;
        lda     vidmem
        and     #$fc
        sta     vidmem  ;vid mem=$c000-$ffff
;
        LDA     #$18
        sta     chmcm   ;character multicolor mode, 40 column display
;
        lda     #$3B    ;BIT MAP MODE ON , 25 lines on screen
        sta     txtctrl ;also rast top bit =0
;
;if the following is changed then change "sptr" in SCB.src and
;       ...change ? in dsprite.src
        LDA     #$38    ;CHAR GEN (MASKS) $e000-$ffff)
        STA     scrnmem ;SCREEN (COLORS) $c000-$c3f8

        lda     #blue1  ;sprite multicolor resisters
        sta     smcr0
        lda     #red2
        sta     smcr1
;
;SET IRQ TO DO RASTER INTERRUPT
;
        lda     IRQ+1   ;take old vector and put into our interrupt routine
        sta     norast+2
        lda     IRQ
        sta     norast+1

        lda     #>VBINT ;put in vector to our ISR (interrupt service routine)
        sta     IRQ+1
        lda     #<VBINT
        sta     IRQ
        lda     #10
        sta     rwrast  ;interrupt at raster line 252
        lda     #$81    ;turn on raster interrupt
        sta     IRQmsk

        cli
        lda     #18     ;((diagnostics
        sta     SINGLE

        jsr     PUTCLOCK        ;init clock


        lda     #19     ;((
        sta     SINGLE

        jsr     INITKEYS        ;init char set



;
;       FIX UP SPRITE STUFF
;
again
        ldx     #4
cleerspt
        lda     update,x        ;wait for all sprites to be updated
        and     #$7f
        beq     next
        jsr     spdate
        jmp     again
```

```
next
        dex
        bpl     cleerspt
;
; x y position set up now time to clear pipe
;
        lda     #$00
;
clearspt
        sta     $c800,x ;clear out the sprite buffers
        sta     $c900,x
        sta     $ca00,x
        sta     $cb00,x
        dex
        bne     clearspt

        lda     #$ff            ;turn on all sprites!!!!!
        sta     senabl          ;enabl sprites to clear c64 hardware
        lda     rwrast          ;wait one frame
hold
        cmp     rwrast
        beq     hold
holdon
        cmp     rwrast
        bne     holdon          ;wait one frame
        lda     #$00            ;disable all sprites!!!!!
        sta     senabl
;
        lda     #$80            ;turn on dog
        sta     update+1
;
        lda     #17     ;((
        sta     SINGLE

        rts                     ;return to main program!!!!!!!!

ATTIME  FCB     0

;VBINT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;NAME=          VBINT
;PURPOSE=       Vertical Blank interrupt - set vbflag,do sounds,inc counter
;REGISTER USAGE=accum,x,y
;REGISTERS ON ENTRY:
;       accum = don't care
;       x =     don't care
;       y =     don't care
;REGISTERS ON EXIT:
;       accum = to be restored from stack
;       x =     to be restored from stack
;       y =     to be restored from stack
;CONDITION FLAGS ON EXIT: not relevant (restored from stack)
;RAM USAGE:
;       NON ZERO PAGE:
;
;       ZERO PAGE:
;
;Effects on grahics and sound: not relevant to graphics, but calls
;       sound routine.
;
;
```

```
;


        lda     intflg
        bpl     norast  ;isnt a raster interrupt
        sta     intflg  ;reset interrupt
;got a raster interrupt

        INC     ATTIME  ;inc frame counter

        jsr     TUNEPLAY        ;do sounds

        lda     #1      ;set flag to tell vblank happened
        sta     vbflag  ;1=time for vblank sync/0=waiting

        pla
        tay
        pla
        tax
        pla
        rti     ;return from interrupt
norast
;THIS JUMP IS MODIFIED ABOVE
        jmp     $0000   ;do rest of IRQ routine in kernal


DEHYB
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;NAME=          DEHYB
;PURPOSE=       Dehybernate PP from the disk
;REGISTER USAGE=accum,x,y
;REGISTERS ON ENTRY:
;       accum = don't care
;       x =     don't care
;       y =     don't care
;REGISTERS ON EXIT:
;       accum = scrambled
;       x =     scrambled
;       y =     scrambled
;CONDITION FLAGS ON EXIT: carry set if PP already had been dehybernated
;RAM USAGE:
;       NON ZERO PAGE:
;       holdhyb
;       ZERO PAGE:
;
;Effects on grahics and sound: not relevant
;
;
;

        lda     #$22    ;[[ de-hybernate
        sta     task    ;have PP do de-hybernation task (get out of piano)
        lda     #0      ;reset task and subtask pointers
        sta     taction
        sta     saction
        jsr     FLOAD   ;load in the hybernation data from disk
        fcb     7+$80
```

```
        fdb     HYBERDATA
hybnam
        fcc     /@:HYBER/
ehybnam
        jsr     LOADMOOD        ;load in current mood and calendar

        lda     HYBERNATED      ;check if PP was hybernated last time
        sta     holdhyb         ;hold status in RAM
        and     #$7f    ;clear "hybernated" flag to show PP currently dehybernate
        sta     HYBERNATED

        ldx     #green1 ;set his facial color to green if sick or flesh if not si
        lda     SICK    ;sick is non-zero if PP is sick
        bne     asadog
        ldx     #red2
asadog
        stx     smcr1
;fall into the hybernation routine so that hybernation status is stored...
;...back onto the disk
HYB
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;NAME=          HYB
;PURPOSE=       Hybernate PP on the disk (also used by DEHYB)
;REGISTER USAGE=accum,x,y
;REGISTERS ON ENTRY:
;       accum = don't care
;       x =     don't care
;       y =     don't care
;REGISTERS ON EXIT:
;       accum = scrambled
;       x =     scrambled
;       y =     scrambled
;CONDITION FLAGS ON EXIT: not relevant
;RAM USAGE:
;       NON ZERO PAGE:
;
;       ZERO PAGE:
;
;Effects on grahics and sound: not relevant
;
;
;


;hybernate data to disk

        lda     #0
        sta     senabl  ;disable sprites during disk io

;set up to call KERNAL SAVE ROUTINE
        ldy     #255            ;secondary addr not needed
        ldx     #8              ;device in x reg
        lda     #3              ;log. file #
        jsr     SETLFS
        lda     #<ehybnam-hybnam        ;set filename
        ldx     #<hybnam
        ldy     #>hybnam
        jsr     SETNAM
FL4
        lda     #<HYBERDATA     ;set pointers to begin and end of hybernation.
```

```
        sta     JTPTR           ;... data
        lda     #>HYBERDATA
        sta     JTPTR+1
        ldx     #<EHYBERDATA
        ldy     #>EHYBERDATA
        lda     #<JTPTR
        JSR     SAVE            ;save hybernation data
        bcs     FL4

        lda     holdhyb         ;set carry if PP was shown dehybernated on disk
        eor     #$80
        asl     a
        rts
holdhyb rmb     1
```

SETLFS: FFBA

A: Log. File # X: Device Y: FF or sec. addr

SAVE: FFD8