```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;      Apple /// RAT Driver     ;
;                               ;
; For Apple ][-//e Mouse Card   ;
;        rls - 11/85            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Rat Driver Specific Error Codes

XNoRat .EQU 30 ;No Mouse Card In Specified Slot
XInvRPrm .EQU 31 ;Invalid Mouse Parameters
XInvROp .EQU 32 ;Invali operation request (such as trying
 ; to read the Mouse when it is inactive)
;Control/Status Equates

ReqCode .EQU 0C0 ;Request code
BufPtr .EQU 0C2 ;Buffer pointer
CtlStat .EQU 0C2 ;Control/status code
CSlist .EQU 0C3 ;Control/Status list pointer
ReqKt .EQU 0C4 ;Requested byte count
BRPtr .EQU 0C8 ;Bytes-read pointer
SltC .EQU 0E0 ;Zpage temp for Mouse firmware check
TmpM10 .EQU 0E0 ;Temp for 16 bit integer to ASCII conv.
TmpDig .EQU 0E2 ;Temp for ASCII digit
RetKt .EQU 0E3 ;Returned byte count temp

PrmTblEP .EQU 1401 ;Parameter Table Extended Page
AllocSIR .EQU 1913 ;Allocate system internal resource
DealcSIR .EQU 1916 ;Deallocate system internal resource
QueEvent .EQU 191F ;Place event in queue
SysErr .EQU 1928 ;System error handler

; RAT Hardware Equates

AIODDR .EQU 0C080 ;6821 I/O - DDR Register A
APIACR .EQU 0C081 ;6821 Control Register A
BIODDR .EQU 0C082 ;6821 I/O - DDR Register B
BPIACR .EQU 0C083 ;6821 Control Register B

; /// Hardware Equates

ASltInt .EQU 0FFDD ;Any slot interrupt flag
Ereg .EQU 0FFDF ;Environment register
Breg .EQU 0FFEF ;RAM bank register
VKIntCtl .EQU 0FFEC ;VIA #2 PCR
VKIntFlg .EQU 0FFED ;VIA #2 IFR
VKIntEn .EQU 0FFEE ;VIA #2 IER

; Macros
 .MACRO Slowdwn
 LDA Ereg
 ORA #80
 STA Ereg
 .ENDM

 .MACRO SpeedUp
 LDA Ereg
 AND #7F
 STA Ereg
 .ENDM
```

```
 .MACRO Set1Mhz ;Shift to granny
 PHP
 SEI
 Slowdwn
 PLP
 .ENDM

 .MACRO Set2Mhz ;Crank it up
 PHP
 SEI
 SpeedUp
 PLP
 .ENDM

 .MACRO IncAdr ;Increment 3 byte address
 INC %1
 BNE $10
 INC %1+1
 BNE $10
 SEC
 ROR %1+1
 INC %1+PrmTblEP ;increment extended page
$10 .ENDM


 .PROC RatDriver

 .WORD 0FFFF ;Comment Flag
 .WORD 22 ;# of chars in comment
 .ASCII "Apple // Mouse driver"
 .ASCII " -- rls 11/85"
DIB .WORD 00 ;Link
.WORD ENTRY
.BYTE 04 ;Length of Driver Name
.ASCII ".RAT           " ;Device Name (pad to 15 char)
 .BYTE 80 ;Activity Flag (80=TRUE)
DIBslt .BYTE 0FF ;Slot # (not currently defined)
 .BYTE 00 ;Unit #
 .BYTE 60 ;Type (char, read/write)
 .BYTE 00 ;Subtype
 .BYTE 00 ;Filler
 .WORD 0000 ;Link
 .WORD 00 ;Manufacturer
 .WORD 1100 ;Version 1.1

 .WORD 00 ;No DCB used

; General driver variables

OpnFlg .BYTE 00 ;<Device is OPEN> flag
Iok .BYTE 00 ;<Mouse card found> flag
NLFlg .BYTE 00 ;NewLine mode in effect flag
NLNxt .BYTE 00 ;NewLine char goes next flag
NLChar .BYTE 00 ;Newline character (for read)
RdFlg .BYTE 00 ;Read in progress flag
RdKt .BYTE 00 ;Counter for bytes read
Ld0Flg .BYTE 00 ;Flag for truncating leading 0's
FldKt .BYTE 00 ;Counter for fields processed
WrtFlg .BYTE 00 ;Write in progress flag
WrtKt .BYTE 00 ;Counter for bytes written
```

```
IntPer .BYTE 00 ;Interrupt period (x/60 second)
SltNum .BYTE 00 ;Slot number (s0)
Rtmp .BLOCK 5,0 ;Temporary storage
TBufr .BLOCK 0E,0 ;Temporary buffer for strings

; RAT Variables

BIStat .BYTE 00 ;Button and interrupt status
YH .BYTE 00 ;Current Y coordinate (high)
YL .BYTE 00 ;Current Y coordinate (low)
XH .BYTE 00 ;Current X coordinate (high)
XL .BYTE 00 ;Current X coordinate (low)
CRMode .BYTE 00 ;Current Rat Mode

ClpXlo .BLOCK 2,0 ;Current clamping values (low byte first)
ClpXmx .BLOCK 2,0
ClpYlo .BLOCK 2,0
ClpYmx .BLOCK 2,0

PLen .EQU *-OpnFlg ;Parameter table length

SClpTbl .BYTE 0,0,0FF,3,0,0,0FF,3 ;Standard clamping values table
 ; (used only to set current value
 ; table to internal mouse defaults)

; SIR table for VIA (synch mouse on VBL)
VSIRaddr .WORD VSIRtbl
VSIRTbl .BYTE 00 ;SIR number
 .BYTE 00 ;ID byte
 .WORD 00 ;No interrupts used here
VSIRXbyt .BYTE 00 ;SIR X byte
VSIRkt .EQU *-VSIRtbl

; SIR table for mouse interrupts
SIRaddr .WORD SIRtbl
SIRtbl .BYTE 00 ;SIR number
 .BYTE 00 ;ID byte
 .WORD Serve ;Interrupt handler address
SIRXbyt .BYTE 00 ;SIR X byte
SIRkt .EQU *-SIRtbl

; Event table for Mouse button event
MBEaddr .WORD MBETbl
MBETbl .BLOCK 5,0 ;Priority, id, address (low,high,x-byte)

; Event table for Mouse movement event
MMEaddr .WORD MMETbl
MMETbl .BLOCK 5,0

; Event table for Mouse timer event
MTEaddr .WORD MTETbl
MTETbl .BLOCK 5,0

; Request table

ReqTbl .WORD DevRead-1
 .WORD DevWrite-1
 .WORD DevStat-1
 .WORD DevCtrl-1
 .WORD ICErr-1
 .WORD ICErr-1
```

```
 .WORD DevOpen-1
 .WORD DevClose-1
 .WORD DevInit-1

ENTRY LDA ReqCode ;Request code check
 CMP #09
 BCS ICErr
 ASL A ;Code ok
 TAY
 LDA ReqTbl+1,Y
 PHA
 LDA ReqTbl,Y
 PHA
 RTS

; System errors

ICErr LDA #20 ;Invalid Request Code Error
 JSR SysErr

DNOErr LDA #23 ;Device Not Open Error
 JSR SysErr

DevInit LDY DIBslt ;First check slot boundary
 DEY
 CPY #04
 BCS DSerr ;Out of bounds- exit w/C=1
 LDA #0 ;Check for mousecard
 STA Iok ;Se card-found flag to false
 STA SltC ;Zero out low address pointer byte
 LDA DIBslt ;Get slot number
 ORA #0C0 ; and stuff in high address pointer
 STA SltC+1
 Set1Mhz
 LDY #0C ;Check firmware sig #1
 LDA (SltC),Y
 CMP #20 ;Sig #1 = $20?
 BNE DSerr ;No- bye
 LDY #0FB ;Check firmware sig #2
 LDA (SltC),Y
 CMP #0D6 ;Sig #2 = $d6?
 BNE DSerr ;No- bye
 LDA #80 ;Ok- Set flag and clear carry
 STA Iok
 CLC
 BCC Rfnd ;Ok - card found
DSerr SEC
Rfnd Set2Mhz
 RTS

DevOpen BIT OpnFlg
 BPL $1
 LDA #24 ;Device Not Available
 JSR SysErr
$1 JSR InitRat ;Set mouse and variables to
 ;default startup state
 LDA #80 ;Set <Device Is Open> flag
 STA OpnFlg
 RTS

DevClose ASL OpnFlg ;Check status and clear flag
```

```
 BCS DoClose
 JMP DNOErr
DoClose LDA CRMode ;Check current mode
 BEQ $1 ;Rat is not running
 JSR RatOff ;Deactivate the Rat
$1 LDA #SIRkt ;Deallocate resources and bug out
 LDX SIRaddr
 LDY SIRaddr+1
 JSR DealcSIR
 RTS

DevRead BIT OpnFlg
 BMI $1
 JMP DNOErr
$1 LDA CRMode ;Check that Mouse card is active
 AND #1
 BNE DoRead
 LDA XInvROp
 JSR SysErr

DoRead JSR FixUp ;Play it safe- check & fix any buf. anomalies
 LDA #0
 STA RetKt
 LDA #0FF ;set 1's compliment
 EOR ReqKt
 STA ReqKt
$1 INC ReqKt
 BEQ RdEnd
 JSR GetByte
 LDY #0
 STA (BufPtr),Y
 PHA
 IncAdr BufPtr
 INC RetKt
 PLA
 BIT NLFlg
 BPL $1
 CMP NLChar
 BNE $1
RdEnd LDY #0
 LDA RetKt
 STA (BRPtr),Y
 TYA ;0-> MSB
 INY
 STA (BRPtr),Y
 RTS

; 'GetByte' does all the work of reading the mouse data, formatting, and
; returning the decimal digit to the main Read routine.
GetByte BIT NLNxt ;Is NewLine char going next?
 BMI XferNL ;Yes- send it
 BIT RdFlg ;Are we in the middle of a read xfer?
 BMI Xfer ;Yes- branch
 LDA #80 ;Set xfer flag True so Mouse won't be read
 STA RdFlg ; until current values are pulled.
 ASL A ;0->A
 STA FldKt
 JSR Read ;Read Mouse and store current values
 JSR B2Str ;Convert binary Mouse data to string format
 LDA #0
 STA RdKt ;Set bytes-read counter to 0
```

```
 STA NLNxt ;Set NewLine char next flag to 0
 BEQ Xfer ;Always
XferNL ASL NLNxt ;Read/clear flag
 LDA NLChar
 BCS XfrEnd ;Always
Xfer LDY RdKt ;Get rat data bytes read counter
 LDA TBufr,Y
 INY
 STY RdKt
 CMP #80 ;End of field?
 BCC Xfrend ;B/no
 AND #7F ;Clear bit 7
 INC FldKt ;Increment field counter
 LDY FldKt ;And check for end of record
 CPY #3
 BNE ChkNl ;B/not end
 LDY #0 ;Else clear read flag
 STY RdFlg
ChkNl BIT NLFlg ;Is NewLine flag set?
 BPL Xfrend ;No- branch
 LDY #80 ;Set NewLine char goes next flag
 STY NLNxt
Xfrend RTS

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; 16 bit to ASCII conversion ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
B2Str LDY #0 ;Clear buffer index
 LDA XL ;Convert X coords
 LDX XH
 JSR Cnvrt
 LDA YL ;Convert Y coords
 LDX YH
 JSR Cnvrt
 LDA BIStat ;Convert status byte
 ROL A ;Shift button flag bits
 ROL A
 ROL A
 AND #03 ;Clear out junk bits
 EOR #03 ;and convert bits 0+1 to 2's comp. (0=4,1=3,2=2,3=1)
 SEC
 ADC #00
 LDX #00
 CLC ;Status is always positive
 JMP PosCon

Cnvrt CPX #80 ;Check for negative number
 BCC PosCon ; and branch if not
;convert negative to psitive by performing 2's complement operation
 EOR #0FF
 ADC #00
 PHA
 TXA
 EOR #0FF
 ADC #00
 TAX
 PLA
 SEC
PosCon STA TmpM10
 STX TmpM10+1
 LDA #"+"
```

```
 BCC PutSgn
 LDA #"-"
PutSgn STA TBufr,Y
 INY
 LDA #80 ;Set Leading 0 flag to True
 STA Ld0Flg
 LDX #5-1 ;Index for 5 digits max. per field
NxDigt LDA #"0"
 STA TmpDig ;Start with "0" and work up
Subtct LDA TmpM10
 CMP LowTbl,X ;C=1 if >=
 LDA TmpM10+1
 SBC HiTbl,X ;See if High and Low >= table
 BCC $1 ;B/less than
 STA TmpM10+1 ;Store result high
 LDA TmpM10
 SBC LowTbl,X
 STA TmpM10
 INC TmpDig ;Increment the digit
 BNE Subtct ;Always
$1 LDA TmpDig
 CPX #0 ;Check if last digit
 BEQ $2 ;B/yes
 CMP #"0"
 BNE $2
 BIT Ld0Flg
 BMI $3
$2 STA TBufr,Y
 ASL Ld0Flg
 INY
$3 DEX
 BPL NxDigt ;Branch if field not finished
 ORA #80 ;Flag last character of field
 STA TBufr-1,Y
 RTS

HiTbl .BYTE 00 ;1
 .BYTE 00 ;10
 .BYTE 00 ;100
 .BYTE 03 ;1000
 .BYTE 27 ;10000
LowTbl .BYTE 01 ;1
 .BYTE 0A ;10
 .BYTE 64 ;100
 .BYTE 0E8 ;1000
 .BYTE 10 ;10000


DevWrite BIT OpnFlg
 BMI DoWrite
 JMP DNOErr

DoWrite JSR FixUp
 LDA #0FF ;set 1's compliment
 EOR ReqKT
 STA ReqKt
 LDA #0FF
 EOR ReqKt+1
 STA ReqKt+1
$1 INC ReqKt
 BNE $2
```

```
 INC ReqKt+1
 BNE $2
 RTS ;Finished
$2 LDY #0
 LDA (BufPtr),Y
 JSR PutByte
 IncAdr BufPtr
 JMP $1

PutByte BNE $1 ;1st check if Rat to be switched off
 LDA CRMode ;Check if Rat is active
 AND #1
 BEQ PutExit ;Already off
 JMP RatOff ;Turn it off now and return to DoWrite
$1 CMP #1
 BNE PutExit ;Just ignore any other characters
; Turn Rat ON
 LDA CRMod
 ORA #1
 JSR Set ;Set Rat mode (and CRMode)
 JSR Clear
PutExit RTS

FixUp LDA BufPtr+1
 BEQ $2
 CMP #0FF
 BEQ $3
 RTS

$2 LDA #80
 STA BufPtr+1
 DEC BufPtr+PrmTblEP
 LDA BufPtr+PrmTblEP
 CMP #7F
 BNE $4
 LDA #20
 STA BufPtr+1
 LDA #8F
 STA BufPtr+PrmTblEP
 BNE $4
$3 CLC
 ROR BufPtr+1
 INC BufPtr+PrmTblEP
$4 RTS

STLen .EQU 0F ;Entries in status table
StatTbl .WORD DS00-1
 .WORD SCtrl-1
 .WORD StatNL-1
 .WORD SStat-1
 .WORD SSet-1
 .WORD SRead-1
 .WORD SClear-1
 .WORD SPos-1
 .WORD SClamp-1
 .WORD SHome-1
 .WORD SButEvt-1
 .WORD SMovEvt-1
 .WORD STmrEvt-1
 .WORD SIntPer-1
 .WORD SR05-1
```

```
DevStat BIT OpnFlg
 BMI $1
 JMP DNOErr

$1 LDA CtlStat ;Status code check
 CMP #STLen
 BCC $2
 LDA #21
 JSR SysErr

$2 ASL A ;Code ok
 TAY
 LDA StatTbl+1,Y
 PHA
 LDA StatTbl,Y
 PHA
 RTS

; do nothing
DS00 RTS

;Return control parameters
SCtrl LDY #0
 LDA #PLen ;Return byte count
 STA (CSList),Y
 PHP
 SEI
 TAY
$2 LDA OpnFlg,Y
 INY ;increment offset and put
 STA (CSList),Y
 DEY ;restore offset
 DEY
 BPL $2
 PLP
 RTS

; Return current NewLine character
StatNl LDY #0
 LDA NLFlg
 STA (CSList),Y
 INY
 LDA NLChar
 STA (CSList),Y
 RTS

; Return and clear the current button/interrupt status byte value
SStat PHP
 SEI
 LDY #0
 LDA BIStat
 STA (CSList),Y
 STY BIStat
 PLP
 RTS

; Return current operation mode
SSet LDY #0
 LDA CRMode
 STA (CSList),Y
```

```
 RTS

; Return mouse X, Y, Status values
SRead JSR Read
 PHP
 SEI
 LDY #0
 LDX #4
$1 LDA BIStat,X
 STA (CSList),Y
 INY
 DEX
 BPL $1
 LDA #0 ;Clear BIStat
 STA BIStat
 PLP
 RTS

; No operation
SClear RTS

; No operation
SPos RTS

; Return current clamping values
SClamp LDY #7
$1 LDA ClpXlo,Y
 STA (CSList),Y
 DEY
 BPL $1 ;Branch until all xferred
 RTS

; No operation
SHome RTS

; Return Button Event parameter table
SButEvt LDY #5
$1 LDA MBETbl,Y
 STA (CSList),Y
 DEY
 BPL $1
 RTS

; Return Mouse Movement Event parameter table
SMovEvt LDY #5
$1 LDA MMETbl,Y
 STA (CSList),Y
 DEY
 BPL $1
 RTS

; Return Mouse Timer Event parameter table
STmrEvt LDY #5
$1 LDA MTETbl,Y
 STA (CSList),Y
 DEY
 BPL $1
 RTS

; Return interrupt interval (x/60 seconds)
SIntPer LDY #0
```

```
 LDA IntPer
 STA (CSList),Y
 RTS

; Read byte from 6805
SR05 PHP
 SEI
 LDY #0
 LDA (CSList),Y ;Low address byte
 STA RTmp+1
 INY
 LDA (CSList),Y ;High address byte
 STA RTmp
 LDA #0F0 ;Mouse code to read byte
 STA RTmp+2
 LDX #2 ;3 bytes to send (code, addrL, addrH)
 JSR SndRDat
 LDX #0 ;1 byte to receive
 JSR GetRDat
 LDA RTmp
 LDY #2
 STA (CSList),Y ;Return in status buffer
 PLP
 RTS


CTLen .EQU 0F ;Entries in status table
CtrlTbl .WORD CReset-1
 .WORD CCtrl-1
 .WORD SetNL-1
 .WORD CStat-1
 .WORD CSet-1
 .WORD CRead-1
 .WORD CClear-1
 .WORD CPos-1
 .WORD CClamp-1
 .WORD CHome-1
 .WORD CButEvt-1
 .WORD CMovEvt-1
 .WORD CTmrEvt-1
 .WORD CIntPer-1
 .WORD CW05-1

DevCtrl BIT OpnFlg
 BMI $1
 JMP DNOErr

$1 LDA CtlStat ;Status code check
 CMP #CTLen
 BCC $2
 JMP SCErr

$2 ASL A ;Code ok
 TAY
 LDA CtrlTbl+1,Y
 PHA
 LDA CtrlTbl,Y
 PHA
 RTS

; Reset- deactivate and set Mouse to initial startup defaults
```

```
CReset JSR RatOff ;Switch Mouse off
 JSR Clear ;Clear internal Mouse X,Y position
 JSR ReStor
 RTS

; Set control parameters
CCtrl LDY #0
 LDA (CSList),Y ;Get parameter count
 CMP #PLen ;and make sure it matches
 BEQ $1
 LDA #XInvRPrm
 JSR SysErr
$1 PHP
 SEI
 TAY
$2 INY ;increment offset and get
 LDA (CSList),Y
 DEY ;restore offset and put
 STA OpnFlg,Y
 DEY
 BPL $2
 PLP
 RTS

; Set NewLine character
SetNL LDY #0
 LDA (CSList),Y
 STA NLFlg
 INY
 LDA (CSList),Y
 STA NLChar
 RTS

; no operation
CStat RTS

; Se Mouse operation mode
CSet LDY #0
 LDA (CSList),Y
 CMP #10 ;Valid mode <= $0F?
 BCC $1
 LDA #XInvRPrm ;Error if not
 JSR SysErr ;No return
$1 JSR Set
 RTS

CRead RTS ;No operation

CClear JSR Clear
 RTS

; Set X,Y internal mouse position to passed values
CPos PHP
 SEI
 LDY #0
 LDX #3
$1 LDA (CSList),Y
 STA RTmp,X
 INY
 DEX ;4 bytes xferred?
 BPL $1 ;B/no
```

```
 LDA #40 ;6805 code to set internal position vars
 STA RTmp+4
 LDX #4
 JSR SndRDat
 PLP
 RTS

CClamp LDY #7 ;Xfer 8 parameters
$1 LDA (CSList),Y ;Gt clamp values from parm list
 STA ClpXlo,Y ;Store in current clamp table
 DEX
 BPL $1
 JSR Clamp
 RTS

CHome JSR Home
 RTS

; Setup Mouse Button event parameters
CButEvt PHP
 SEI
 LDY #5
$1 LDA (CSList),Y
 STA MBETbl,Y
 DEY
 BPL $1
 PLP
 RTS

; Setup Mouse Movement event parameters
CMovEvt PHP
 SEI
 LDY #5
$1 LDA (CSList),Y
 STA MMETbl,Y
 DEY
 BPL $1
 PLP
 RTS

; Setup Mouse Timer event parameters
CTmrEvt PHP
 SEI
 LDY #5
$1 LDA (CSList),Y
 STA MTETbl,Y
 DEY
 BPL $1
 PLP
 RTS

; Set interrupt interval (x/60 seconds)
CIntPer LDY #0
 LDA (CSList),Y
 STA IntPer
 JSR SetIntP
 RTS

; Write byte to 6805
CW05 PHP
 SEI
```

```
 LDY #0
 LDX #2
$1 LDA (CSList),Y
 STA RTmp,X
 INY
 DEX
 BPL $1
 LDA #0F1 ;Mouse code to write byte
 STA RTmp+3
 LDX #3 ;4 bytes to send (code, addrL, addrH, byte)
 JSR SndRDat
 PLP
 RTS


SCErr LDA #21 ;Invalid Control/Status Code
 JSR SysErr

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; RAT 6805/6502 communication routines ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Initialize PIA pot B ;
InitB PHP
 SEI
 SlowDwn
 LDY SltNum
 LDA BPIACR,Y ;Set CRB for DDR active
 AND #0FB
 STA BPIACR,Y
 LDA #3E ;Set DDRB: 0,6,7=input / 1-5=output
 STA BIODDR,Y
 LDA BPIACR,Y ;Set CRB for I/O active
 ORA #4
 STA BPIACR,Y
 SpeedUp
 PLP
 RTS

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Get RAT data from 6805 ;
; Enter with # of parms-1 in X reg ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
GetRDat Slowdwn
 LDY SltNum
 LDA APIACR,Y ;Set CRA for DDR active
 AND #0FB
 STA APIACR,Y
 LDA #00 ;All data lines= INPUT
 STA AIODDR,Y
 LDA APIACR,Y ;Set CRA for I/O active
 ORA #04
 STA APIACR,Y
RLup1 LDA BIODDR,Y ;Wait until PB6=1
 ASL A
 BPL RLup1
 LDA AIODDR,Y ;Read 6805 data byte
 STA Rtmp,X ;and stuff it.
 LDA BIODDR,Y ;Set PB4=1 (flag to 6805)
 ORA #10
 STA BIODDR,Y
```

```
RLup2 LDA BIODDR,Y ;Wait until PB6=0
 ASL A
 BMI RLup2
 LDA BIODDR,Y ;Clear PB4=0
 AND #0EF
 STA BIODDR,Y
 DEX ;Decrement & repeat until done
 BPL RLup1
 SpeedUp
 RTS

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Send Rat data to 6805 ;
; Enter with # of parms-1 in X reg ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SndRDat Slowdwn
 LDY SltNum
$1 LDA BIODDR,Y ;Wait until PB7 = 0
 BMI $1
 LDA APIACR,Y ;Set CRA for DDR active
 AND #0FB
 STA APIACR,Y
 LDA #0FF ;All data lines = OUTPUT
 STA AIODDR,Y
SndNxt LDA APIACR,Y ;Set CRA for I/O active
 ORA #04
 STA APIACR,Y
 LDA Rtmp,X ;Get data and xmit
 STA AIODDR,Y
 LDA BIODDR,Y ;Set PB5 = 1
 ORA #20
 STA BIODDR,Y
SdLup1 LDA BIODDR,Y ;Wait until PB7 = 1
 BPL SdLup1
 AND #0DF ;Clear PB5 = 0
 STA BIODDR,Y
 DEX
 BMI Snd_End ;Branch if done
SdLup2 LDA BIODDR,Y ;Else - Wait until PB7 = 0
 BMI SdLup2
 BPL SndNxt ; and xmit more
Snd_End SpeedUp
 RTS

InitRat BIT Iok ;Check if boot-up INIT went ok
 BMI DoInit ;Yes- goferit
 LDA #XNoRat ;No Mouse Card
 JSR SysErr
; Set driver parameters
DoInit LDA DIBslt ;Get slot
 ORA #10 ;Format for SIR
 STA SIRtbl ;and stuff
 ASL A ;Format for indexing (s0)
 ASL A
 ASL A
 ASL A
 STA SltNum ;Save for future use
 LDA Breg ;Stuff Bank in SIR
 AND #0F
 STA VSIRXbyt
 STA SIRXbyt
```

```
 LDA #SIRkt ;Length of SIR table
 LDX SIRaddr ;Get pointer to SIR table
 LDY SIRaddr+1
 JSR AllocSIR
 BCC $1 ;Clear Mouse position variables
 LDA #25 ;Resource Not Available
 JSR SysErr
$1 JSR InitB ;Initialize PIA port B
 JSR ReStor ;Set mouse/driver variables/defaults
 PHP
 SEI
 LDX #0 ;1 parm to send
 LDA #50 ;1st init call
 STA Rtmp
 JSR SndRDat ;send it
 LDX #0
 JSR GetRDat ;byte returned is NOT used
 PLP
; Useless (at present) attempt to initially synchronize the Mouse
; with the ///'s VBL cycle.
 LDA #VSIRkt ;Set up SIR
 LDX VSIRaddr
 LDY VSIRaddr+1
 JSR AllocSIR
 BCS IExit ;B/unable to allocate
 PHP ;Shut down interrupts
 SEI
 LDA VKIntCtl ;Prepare VIA
 AND #1F ;Retain CB1 and CA1/CA2 status
 ORA #60 ;CB2=independent interrupt on pos. edge
 STA VKIntCtl
 LDA #8 ;Disable CB2 interrupt (will still be flagged)
 STA VKIntEn
 STA VKIntFlg ;Clear CB2 interrupt bit
 PLP ; Reinstate any interrupts
$2 BIT VKIntFlg ;check if VBL has triggered bit
 BEQ $2 ;keep polling until VBL
 LDA #VSIRkt ;Success- deallocate and continue
 LDX VSIRaddr
 LDY VSIRaddr+1
 JSR DealcSIR
 PHP
 SEI
 LDX #0 ;Now signal Mouse to synchronize
 LDA #50
 STA Rtmp
 JSR SndRDat ;Finish up the init call
 PLP
IExit RTS

; Interrupt handler for the mouse driver. Update BIStat, signal events
Serve LDX #0
 LDA #20 ;6805 "Serve" command
 STA RTmp
 JSR SndRDat
 LDX #0 ;Get 1 byte (interrupt flag) from 6805
 JSR GetRDat
 LDA RTmp
; The byte returned flags the cause(s) of the interrupt and only affects bits
; 1-3 of BIStat. Prior (unread) interrupt flag bits are preserved. Events are
; checked and called if active (then cleared).
```

```
 PHA ;Preserve interrupt stat
 ORA BIStat
 STA BIStat ;Strre current button/interrupt status
 PLA
 PHA
 AND #4 ;Check for button interrupt
 BEQ $1
 LDA MBETbl ;Check if button event active
 BEQ $1 ;B/no
 LDX MBEaddr
 LDY MBEaddr+1
 JSR QueEvent
 LDA #0
 STA MBETbl
$1 PLA
 PHA
 AND #2 ;Check for movement interrupt
 BEQ $2
 LDA MMETbl ;Mouse movement event active?
 BEQ $2 ;B/no - check for timer interrupt/event
 LDX MMEaddr
 LDY MMEaddr+1
 JSR QueEvent
 LDA #0
 STA MMETbl
$2 PLA
 AND #8 ;Check for timer event
 BEQ $3
 LDA MTETbl ;Mouse timer event active?
 BEQ $ ;B/no - exit
 LDX MTEaddr
 LDY MTEaddr+1
 JSR QueEvent
 LDA #0
 STA MTETbl
$3 LDA #2 ;Clear VIA 'Any Slot' interrupt flag
 STA ASltInt
 RTS

; Read Mouse values
Read PHP
 SEI
 LDX #0 ;Send Read code to 6805
 LDA #10
 STA RTmp
 JSR SndRDat
 LDX #4
 JSR GetRDat ;Now go read Mouse values
 LDX #3 ;and xfer to current storage
$1 LDA RTmp+1,X
 STA BIStat+1,X
 DEX
 BPL $1 ;repeat until 4 coordinate bytes xferred
 LDA BIStat ;Retrieve current status
 AND #0E ;Preserve interrupt flags
 ORA RTmp ;Update current button/movement flags
 STA BIStat ;And store
 PLP
 RTS

; Set minimum and maximum mouse boundaries for X and Y positions
```

```
Clamp PHP
 SEI
 LDA #60 ;"60" is 6805 code to set X clamp values
 STA RTmp+4
 LDA ClpXlo ;1st= low byte of minimum
 STA Rtmp+3
 LDA ClpXmx ;2nd= low byte of maximum
 STA Rtmp+2
 LDA ClpXlo+1 ;3rd= high byte of minimum
 STA Rtmp+1
 LDA ClpXmx+1 ;4th= high byte of maximum
 STA Rtmp
 LDX #4 ;Send 5 bytes to 6805
 JSR SndRDat

 LDA #61 ;"61" is 6805 code to set Y clamp values
 STA RTmp+4
 LDA ClpYlo
 STA Rtmp+3
 LDA ClpYmx
 STA Rtmp+2
 LDA ClpYlo+1
 STA Rtmp+1
 LDA ClpYmx+1
 STA Rtmp
 LDX #4 ;Send 5 bytes to 6805
 JSR SndRDat
 PLP
 RTS

; Set operating mode of Rat (A= valid mode of 0 - $0F)
Set PHP
 SEI
 STA RTmp
 STA CRMode ;Set current Mode
 LDX #0
 JSR SndRDat
 PLP
 RTS

; Set Mouse internal registers to lower boundaries
Home PHP
 SEI
 LDA #70
 STA RTmp
 LDX #0
 JSR SndRDat
 PLP
 RTS

; Set Mouse and driver X,Y position variables to 0
Clear PHP
 SEI
 LDA #30 ;6805 "clear" command
 STA RTmp
 LDX #0
 JSR SndRDat
 LDY #3 ;Clear driver variables
 LDA #0
$1 STA YH,Y
 DEY
```

```
 BPL $1
 PLP
 RTS

; Set Interrupt period to x/60 second (normally 1/60)
SetIntP PHP
 SEI
 LDA IntPer ;Get the period constant
 STA RTmp
 LDA #0A0 ;Code to set interrupt interval
 STA RTmp+1
 LDX #01 ;Send 2 parameters to Mouse
 JSR SndRDat
 PLP
 RTS

; Miscellaneous routines

RatOff PHP
 SEI
 LDX #0 ;0-> 6805 turns Mouse off
 STX Rtmp
 STX CRMode ;Set current Mode
 JSR SndRDat
 PLP
 RTS

RVrZap LDA #0 ;Clear out current RAT variables
 LDX #4
$1 STA BIStat,X
 DEX
 BPL $1
 STA RdFlg ;Set Flags
 STA WrtFlg
 STA NLNxt
 LDA #01
 STA IntPer ;Set interrupt period
 LDX #7 ;Set current clamp value table to standard
$2 LDA SClpTbl,X
 STA ClpXlo,X
 DEX
 BPL $2
$3 RTS

; Restore startup mouse variables/values
ReStor JSR RVrZap ;Restore variables to startup conditions
 JSR SetIntP ;Restore mouse interrupt period
 JSR Clamp ;Set default clamp values
 RTS

 .END



Capture buffer closed.


[72415,1607]
RATDVR.TXT               15-Jan-86 22410 Accesses: 35

Enter command, N for next file
```

or <CR> for disposition menu !