Subject: Apple IIgs firmware equates
Posted by toinet on Fri, 22 Aug 2014 21:49:06 GMT
View Forum Message <> Reply to Message

If one needs a reference table, there it is!
It describes all the softswitches and important memory locations of the Apple IIgs.

```
; This EdAsm/Asm816 source code file was converted to AsmIIGS
; by EdAsmCvtIIGS version 1.2d4 on 5/7/91 at 4:9:18 PM

  TITLE 'Bank $FF Variables'

*****************************************************
*                                   *
*          Bank $FF Variables           *
*                                   *
*               by                  *
*          Fern Bachman    1985-1987      *
*          Joe Bo         1987-1989       *
*                                   *
*    Copyright Apple Computer, Inc. 1985-1989     *
*          All Rights Reserved.           *
*                                   *
*****************************************************


F8VERSION  EQU $06 ;F8 ROM //e ID byte


*
* ZERO PAGE EQUATES
*
LOC0  EQU $00 ;vector for autostart from disk
LOC1  EQU $01
PRINTTMP  EQU $06 ;$06-$08 saved and restore by print rtne
WNDLFT  EQU $20 ;left edge of text window
WNDWDTH  EQU $21 ;width of text window
WNDTOP  EQU $22 ;top of text window
WNDBTM  EQU $23 ;bottom+1 of text window
CH  EQU $24 ;cursor horizontal position
CV  EQU $25 ;cursor vertical position
GBASL  EQU $26 ;lo-res graphics base addr.
GBASH  EQU $27
BASL  EQU $28 ;text base address
BASH  EQU $29
BAS2L  EQU $2A ;temp base for scrolling
BAS2H  EQU $2B
```

```
H2  EQU $2C ;temp for lo-res graphics
LMNEM  EQU $2C ;temp for mnemonic decoding
V2  EQU $2D ;temp for lo-res graphics
RMNEM  EQU $2D ;temp for mnemonic decoding
MASK  EQU $2E ;color mask for lo-res gr.
FORMAT  EQU $2E ;temp for opcode decode
LENGTH  EQU $2F ;temp for opcode decode
COLOR  EQU $30 ;color for lo-res graphics
MONMODE  EQU $31 ;Monitor mode
INVFLG  EQU $32 ;normal/inverse(/flash)
PROMPT  EQU $33 ;prompt character
YSAV  EQU $34 ;position in Monitor command
YSAV1  EQU $35 ;temp for Y register
CSWL  EQU $36 ;character output hook
CSWH  EQU $37
KSWL  EQU $38 ;character input hook
KSWH  EQU $39
PCL  EQU $3A ;temp for program counter
PCH  EQU $3B
A1L  EQU $3C ;Monitor temp
A1H  EQU $3D ;Monitor temp
A2L  EQU $3E ;Monitor temp
A2H  EQU $3F ;Monitor temp
A3L  EQU $40 ;Monitor temp
A3H  EQU $41 ;Monitor temp
A4L  EQU $42 ;Monitor temp
A4H  EQU $43 ;Monitor temp
*A5L EQU $44 ;Monitor temp
A5H  EQU $45 ;Monitor temp
ACC  EQU $45 ;Acc after break (destroys A5H)
XREG  EQU $46 ;X reg after break
YREG  EQU $47 ;Y reg after break
STATUS  EQU $48 ;P reg after break
SPNT  EQU $49 ;SP after break
RNDL  EQU $4E ;random counter low
RNDH  EQU $4F ;random counter high

*
* Value equates
*

GOODF8  EQU $06 ;//e ID byte
X8  EQU $10 ;Mask for interrupt code
M8  EQU $20 ;Mask for interrupt code
EXMEMCMD  EQU $38 ;Mask for extnd mem designator/sctr # cmd
CHKSMCNST  EQU $AAAA ;BATTERYRAM checksum EOR constant

*
```

```
* Bank 1 / Page 1 permanent storage
*

MNEMSTKPTR EQU $010100 ;Main stack pointer
ALEMSTKPTR EQU $010101 ;Alternate stack pointer

*
* Characters read by GETLN are placed in
* IN, terminated by a carriage return.
*

IN  EQU $0200 ;input buffer for GETLN

*
* Page 3 vectors
*

OSWMSTRT  EQU $3D0 ;QUITMON jumps here
BRKV  EQU $03F0 ;vectors here after break
SOFTEV  EQU $03F2 ;vector for warm start
PWREDUP  EQU $03F4 ;THIS MUST = EOR #$A5 OF SOFTEV+1
AMPERV  EQU $03F5 ;APPLESOFT & EXIT VECTOR
USRADR  EQU $03F8 ;APPLESOFT USR function vector
NMI  EQU $03FB ;NMI vector
IRQLOC  EQU $03FE ;Maskable interrupt vector
LINE1  EQU $0400 ;first line of text screen
PRO8MLI  EQU $00BF00 ;Address of PRODOS8 MLI entry point
  EJECT
* HARDWARE EQUATES

  EQUSECT $C000
IOADR  EQUDS * ;All I/O is at $Cxxx
KBD  EQUDS * ;Bit 7=1 if keystroke
CLR80COL  EQUDS.B 1 ;disable 80 column store
SET80COL  EQUDS.B 1 ;enable 80 column store
RDMAINRAM  EQUDS.B 1 ;read from main 48K RAM
RDCARDRAM  EQUDS.B 1 ;read from alt. 48K RAM
WRMAINRAM  EQUDS.B 1 ;write to main 48K RAM
WRCARDRAM  EQUDS.B 1 ;write to alt. 48K RAM
SETSLOTCXROM EQUDS.B 1 ;use ROMS on cards
SETINTCXROM EQUDS.B 1 ;use internal ROM
SETSTDZP  EQUDS.B 1 ;use main zero page/stack
SETALTZP  EQUDS.B 1 ;use alt. zero page/stack
SETINTC3ROM EQUDS.B 1 ;Enable internal slot 3 ROM
SETSLOTC3ROM EQUDS.B 1 ;Enable external slot 3 ROM
CLR80VID  EQUDS.B 1 ;disable 80 column hardware
SET80VID  EQUDS.B 1 ;enable 80 column hardware
CLRALTCHAR EQUDS.B 1 ;normal LC, flashing UC
```

```
SETALTCHAR EQUDS.B 1 ;normal inverse, LC; no flash
KBDSTRB  EQUDS.B 1 ;turn off key pressed flag
RDLCBNK2  EQUDS.B 1 ;Bit 7=1 if LC bank 2 is in
RDLCRAM  EQUDS.B 1 ;Bit 7=1 if LC RAM read enabled
RDRAMRD  EQUDS.B 1 ;Bit 7=1 if reading alt 48K
RDRAMWRT  EQUDS.B 1 ;Bit 7=1 if writing alt 48K
RDCXROM  EQUDS.B 1 ;Bit 7=1 if using int rom
RDALTZP  EQUDS.B 1 ;Bit 7=1 if slot zp enabled
RDC3ROM  EQUDS.B 1 ;Bit 7=1 if slot c3 space enabled
RD80COL  EQUDS.B 1 ;Bit 7=1 if 80 column store
RDVBLBAR  EQUDS.B 1 ;Bit 7=1 if not VBL
RDTEXT  EQUDS.B 1 ;Bit 7=1 if text (not graphics)
RDMIX  EQUDS.B 1 ;Bit 7=1 if mixed mode on
RDPAGE2  EQUDS.B 1 ;Bit 7=1 if TXTPAGE2 switched in
RDHIRES  EQUDS.B 1 ;Bit 7=1 if HIRES is on
ALTCHARSET EQUDS.B 1 ;Bit 7=1 if alternate char set in use
RD80VID  EQUDS.B 1 ;Bit 7=1 if 80 column hardware in
  EQUDS.B 1 ;Reserved for future system expansion

* __7_____6_____5_____4_____3_____2_____1_____0___
*|     |     |     |     |     |     |     |     |
*|Enable|     |     |     |     |     |     |     |
*|color/|   0  |   0  |   0  |   0  |   0  |   0  |   0  |
*| mono |     |     |     |     |     |     |     |
*|_____|_____|_____|_____|_____|_____|_____|_____|
*         ^^^^^  MONOCOLOR Status Byte ^^^^^

* MONOCOLOR bits defined as follows
*    bit 7= 0 enables color -- 1 disables color
*    bit 6,5,4,3,2,1,0  Must be 0

MONOCOLOR  EQUDS.B 1 ;Monochrome/color select register

* __7_____6_____5_____4_____3_____2_____1_____0___
*|     |     |     |     |     |     |     |     |
*|     |     |     |     |     |     |     |     |
*|      Text Color Bits     |   Background Color Bits   |
*|     |     |     |     |     |     |     |     |
*|_____|_____|_____|_____|_____|_____|_____|_____|
*           ^^^^^  TBCOLOR Byte ^^^^^

* TBCOLOR bits defined as follows
*    bit 7,6,5,4 = Text color bits
*    bits 3,2,1,0 = Background color bits
*
*   Color bits =
*              $0 = Black
*              $1 = Deep Red
```

```
*               $2 = Dark Blue
*               $3 = Purple
*               $4 = Dark green
*               $5 = Dark Gray
*               $6 = Medium Blue
*               $7 = Light Blue
*               $8 = Brown
*               $9 = Orange
*               $A = Light Gray
*               $B = Pink
*               $C = Green
*               $D = Yellow
*               $E = Aquamarine
*               $F = White

TBCOLOR  EQUDS.B 1 ;Text/background color select register

* __7______6______5______4______3______2______1______0___
*|      |      |      |      |      |      |      |      |
*| VGC  | 1sec | Scan | Ext  |      | 1sec | Scan | Ext  |
*|  int |  int |  int | int  |Unused|  int |  int | int  |
*|active|active|active|active|      |enable|enable|enable|
*|______|______|______|______|______|______|______|______|
*          ^^^^^  VGCINT Status Byte ^^^^^

* VGCINT bits defined as follows
*    bit 7= 1 if interrupt generated by VGC
*    bit 6= 1 if 1 second timer interrupt
*    bit 5= 1 if scan line interrupt
*    bit 4= 1 if external interrupt
*    bit 3= unused
*    bit 2= 1 second timer interrupt enable
*    bit 1= scan line interrupt enable
*    bit 0= external interrupt enable

VGCINT  EQUDS.B 1 ;VGC interrupt register

* __7______6______5______4______3______2______1______0___
*|      |      |      |      |      |      |      |      |
*|Button|      |                                 |
*|status|Delta |            Delta movement              |
*| now  | sign |                                  |
*|______|______|______|______|______|______|______|______|
*           ^^^^^  MOUSEDATA Byte ^^^^^

* MOUSEDATA bits defined as follows
*    bit 7= if reading X data = button 1 status
*           if reading Y data = button 0 status
```

```
*    bit 6= sign of delta 0='+' -- 1='-'
*    bit 5,4,3,2,1,0 = delta movement

MOUSEDATA  EQUDS.B 1 ;X or Y mouse data register

* __7______6______5______4______3______2______1______0___
*|      |      |Update|      |      |      |      |      |
*| Open |Closed| mod  |Keypad|Repeat| Caps | Ctrl |Shift |
*| Apple| Apple|no key|  key |active| lock |  key | key  |
*|  key |  key | press|active|      |active|active|active|
*|______|______|______|______|______|______|______|______|
*          ^^^^^  KEYMODREG Status Byte ^^^^^

* KEYMODREG bits defined as follows
*    bit 7= Open Apple key active
*    bit 6= Closed Apple key active
*    bit 5= Updated modifier latch without keypress
*    bit 4= Keypad key active
*    bit 3= Repeat active
*    bit 2= Caps lock active
*    bit 1= Control key active
*    bit 0= Shift key active

KEYMODREG  EQUDS.B 1 ;Key modifier register

* __7______6______5______4______3______2______1______0___
*|      |      |      |      |      |      |      |      |
*|                                    |
*|            Data to/from keyboard micro             |
*|                                    |
*|______|______|______|______|______|______|______|______|
*            ^^^^^  DATAREG Byte ^^^^^

* DATAREG bits defined as follows
*    bit 7,6,5,4,3,2,1,0 = data to/from keyboard micro
*
* Data at interrupt time in this register is defined as
*    bit 7= Response byte if set, otherwise status byte
*    bit 6= ABORT valid if set and all other bits reset
*    bit 5= Desktop manager key sequence pressed
*    bit 4= Flush buffer key sequence pressed
*    bit 3= SRQ valid if set
*    bit 2,1,0 If all bits clear then no FDB data valid,
*              else the bits indicate the number of valid
*              bytes received minus 1. (2-8 bytes total)

DATAREG  EQUDS.B 1 ;Data register in KeyGlu chip
```

```
* __7_____6_____5_____4_____3_____2_____1_____0___
*|     |     |     |     |     |     |     |
*| Mouse| Mouse| Data | Data | Key  | Key  | Mouse| Cmd  |
*|  reg |  int |  reg |  int | data | int  |X/Yreg| reg  |
*| full |enable| full |enable| full |enable| data | full |
*|______|______|______|______|______|______|______|______|
*         ^^^^^  KMSTATUS Status Byte ^^^^^

* KMSTATUS bits defined as follows
*    bit 7= 1 if mouse register full
*    bit 6= mouse interrupt disable/enable
*    bit 5= 1 if data register full
*    bit 4= data interrupt enable
*    bit 3= 1 if key data full  <<<<<------NEVER USE, WON'T WORK
*    bit 2= key data interrupt enable  <<<<<------NEVER USE, WON'T WORK
*    bit 1= 0 = mouse 'X' register data available
*           1 = mouse 'Y' register data available
*    bit 0= Command register full

KMSTATUS  EQUDS.B 1 ;Keyboard/mouse status register
ROMBANK  EQUDS.B 1 ;ROM bank select toggle

* __7_____6_____5_____4_____3_____2_____1_____0___
*|     |     |     |     |     |     |     |     |
*|Enable|Linear| B/W  |     |     |     |     |Enable|
*| super| video|Color |  0   |  0   |  0   |  0   |bank 1|
*|hi-res|      |DHires|     |     |     |     |latch |
*|______|______|______|______|______|______|______|______|
*           ^^^^^  NEWVIDEO Byte ^^^^^

* NEWVIDEO bits defined as follows
*    bit 7= 1 to disable Apple //e video (enables super hi-res)
*    bit 6= 1 to linearize for super hi-res
*    bit 5= 0 for color double hi-res -- 1 for B/W hi-res
*    bit 4,3,2,1= MUST be pregrammed as 0
*    bit 0= Enable bank 1 latch to allow long instructions
*           to access bank 1 directly.

NEWVIDEO  EQUDS.B 1 ;Video/enable read alt mem with long instr
  EQUDS.B 1 ;Reserved for future system expansion

* __7_____6_____5_____4_____3_____2_____1_____0___
*|     |     |     |     |     |     |     |     |
*| Character Generator| NTSC/| Lang |     |     |     |
*|   language select  |  PAL |select|   0  |   0  |   0  |
*|     |     |     |     |  bit |     |     |     |
*|______|______|______|______|______|______|______|______|
*           ^^^^^  LANGSEL Byte ^^^^^
```

```
* LANGSEL bits defined as follows
*    bit 7,6,5= Character generator language select
*                  Primary language    Secondary language
*             $0 =       USA               Dvorak
*             $1 =       UK                USA
*             $2 =     French              USA
*             $3 =     Danish              USA
*             $4 =     Spanish             USA
*             $5 =     Italian           USA
*             $6 =     German              USA
*             $7 =     Swedish             USA
*    bit 4= 0 if NTSC video mode -- 1 if PAL video mode
*    bit 3= LANGUAGE switch bit 0 if primary lang set selected
*    bit 2,1,0 ;MUST be programmed as 0's

LANGSEL  EQUDS.B 1 ;Language/PAL/NTSC select register
CHARROM  EQUDS.B 1 ;Addr for tst mode read of character ROM

* __7______6______5______4______3______2______1______0___
*|      |      |      |      |      |      |      |      |
*|Slot7 |Slot6 |Slot5 |Slot4 |      |Slot2 |Slot1 |      |
*|intext|intext|intext|intext|   0  |intext|intext|   0  |
*|enable|enable|enable|enable|      |enable|enable|      |
*|______|______|______|______|______|______|______|______|
*            ^^^^^  SLTROMSEL Byte ^^^^^

* SLTROMSEL bits defined as follows
*    bit 7= 0 enables internal slot 7 -- 1 enables slot ROM
*    bit 6= 0 enables internal slot 6 -- 1 enables slot ROM
*    bit 5= 0 enables internal slot 5 -- 1 enables slot ROM
*    bit 4= 0 enables internal slot 4 -- 1 enables slot ROM
*    bit 3= MUST be 0
*    bit 2= 0 enables internal slot 2 -- 1 enables slot ROM
*    bit 1= 0 enables internal slot 1 -- 1 enables slot ROM
*    bit 0= Must be 0

SLTROMSEL  EQUDS.B 1 ;Slot ROM select
VERTCNT  EQUDS.B 1 ;Addr for read of video cntr bits V5-VB
HORIZCNT  EQUDS.B 1 ;Addr for read of video cntr bits VA-H0
SPKR  EQUDS.B 1 ;clicks the speaker

* __7______6______5______4______3______2______1______0___
*|      |      |      |      |      |      |      |      |
*| 3.5" | 3.5" |      |      |      |      |      |      |
*| head | drive|  0   |  0   |  0   |  0   |  0   |  0   |
*|select|enable|      |      |      |      |      |      |
*|______|______|______|______|______|______|______|______|
```

```
*          ^^^^^  DISKREG Status Byte ^^^^^

* DISKREG bits defined as follows
*    bit 7= 1 to select head on 3.5" drive to use
*    bit 6= 1 to enable 3.5" drive
*    bit 5,4,3,2,1,0= unused

DISKREG  EQUDS.B 1 ;Used for 3.5" disk drives

* __7______6______5______4______3______2______1______0___
*|      |      |      |      |      |      |      |      |
*|      |Clear |Clear |      |      |      |      |      |
*|   0  |1 sec | scan |   0  |   0  |   0  |   0  |   0  |
*|      | int  |In int|      |      |      |      |      |
*|______|______|______|______|______|______|______|______|
*              ^^^^^  SCANINT Byte ^^^^^

* SCANINT bits defined as follows
*    bit 7= unused
*    bit 6= write a 0 here to reset 1 second interrupt
*    bit 5= write a 0 here to clear scan line interrupt
*    bit 4= unused
*    bit 3= unused
*    bit 2= unused
*    bit 1= unused
*    bit 0= unused

SCANINT  EQUDS.B 1 ;Scan line interrupt register

* __7______6______5______4______3______2______1______0___
*|      |      |      |      |      |      |      |      |
*|                                    |
*|              Clock data register                |
*|                                    |
*|______|______|______|______|______|______|______|______|
*             ^^^^^  CLOCKDATA Byte ^^^^^

* CLOCKDATA bits defined as follows
*    bit 7,6,5,4,3,2,1,0 -- Data passed to/from clock chip

CLOCKDATA  EQUDS.B 1 ;Clock data register

* __7______6______5______4______3______2______1______0___
*|      |      |      |      |      |      |      |      |
*| Clock| Read/|Chip  |      |                    |
*| xfer |Write |enable|Unused|        Border Color       |
*|      | chip |assert|      |                    |
*|______|______|______|______|______|______|______|______|
```

```
*            ^^^^^  CLOCKCTL Byte ^^^^^

* CLOCKCTL bits defined as follows
*    bit 7= Set =1 to start transfer to clock
*           Read =0 when transfer to clock is complete
*    bit 6= 0= write to clock chip -- 1= read from clock chip
*    bit 5= Clock chip enable asserted after transfer 0=no 1=yes
*    bit 4= unused
*    bit 3,2,1,0 = select border color (see TBCOLOR for values)

CLOCKCTL  EQUDS.B 1 ;Clock control register

* __7______6______5______4______3______2______1______0___
*|      |      |      |      |      |      |      |      |
*|      | Stop | Stop | Stop | Stop | Stop | Stop | Stop |
*|   0  |I/O/LC|txtpg2|auxh-r|suprhr|hires2|hires1|txtpg1|
*|      |shadow|shadow|shadow|shadow|shadow|shadow|shadow|
*|______|______|______|______|______|______|______|______|
*            ^^^^^  SHADOW Byte ^^^^^

* SHADOW bits defined as follows
*    bit 7= Must write 0
*    bit 6= 1 to inhibit I/O and language card operation
*    bit 5= 1 to inhibit shadowing text page 2
*    bit 4= 1 to inhibit shadowing aux hi-res page
*    bit 3= 1 to inhibit shadowing 32k video buffer
*    bit 2= 1 to inhibit shadowing hires page 2
*    bit 1= 1 to inhibit shadowing hires page 1
*    bit 0= 1 to inhibit shadowing text page 1
*
* Note: bit 5 only implemented in new CYA chip, if using
*       FPI chip then must write 0.

SHADOW  EQUDS.B 1 ;Shadow register

* __7______6______5______4______3______2______1______0___
*|      |      |      |      |      |      |      |      |
*| Slow/|Power |      |Shadow|Slot 7|Slot 6|Slot 5|Slot 4|
*| fast | up   |   0  |in all|motor |motor | motor| motor|
*| speed|detect|      | RAM  |detect|detect|detect|detect|
*|______|______|______|______|______|______|______|______|
*            ^^^^^  CYAREG Byte ^^^^^

* CYAREG bits defined as follows
*    bit 7= 0=slow system speed -- 1=fast system speed
*    bit 6= 1=cold start -- 0=warm start
*    bit 5= Must write 0
*    bit 4= Shadow in all RAM banks
```

```
*    bit 3= Slot 7 disk motor on detect
*    bit 2= Slot 6 disk motor on detect
*    bit 1= Slot 5 disk motor on detect
*    bit 0= Slot 4 disk motor on detect
*
* Note: bit 6 only implemented in new CYA chip, if using
*       FPI chip then must always write 0.

CYAREG  EQUDS.B 1 ;Speed and motor on detect
DMAREG  EQUDS.B 1 ;Used during DMA as bank address
SCCBREG  EQUDS.B 1 ;SCC channel B cmd register
SCCAREG  EQUDS.B 1 ;SCC channel A cmd register
SCCBDATA  EQUDS.B 1 ;SCC channel B data register
SCCADATA  EQUDS.B 1 ;SCC channel A data register

* __7_____6_____5_____4_____3_____2_____1_____0___
*|     |     |     |     |     |     |     |     |
*| Busy |Access| Auto |     |                     |
*| flag | doc/ | inc  |   0  |       Volume DAC       |
*|      | RAM  |adrptr|     |                     |
*|______|______|______|_____|_____|_____|_____|_____|
*           ^^^^^  SOUNDCTL Byte ^^^^^

* SOUNDCTL bits defined as follows
*    bit 7= 0 if not busy -- 1 if busy
*    bit 6= 0=access doc -- 1=access RAM
*    bit 5= 0=disable auto incrementing of address pointer
*           1=enable auto incrementing of address pointer
*    bit 4= Must be 0
*    bit 3,2,1,0= volume DAC $0=off $F=full volume

SOUNDCTL  EQUDS.B 1 ;Sound control register

* __7_____6_____5_____4_____3_____2_____1_____0___
*|     |     |     |     |     |     |     |     |
*|                                   |
*|            Sound data read/written              |
*|                                   |
*|______|______|______|_____|_____|_____|_____|_____|
*         ^^^^^  SOUNDDATA Byte ^^^^^

* SOUNDDATA bits defined as follows
*    bit 7,6,5,4,3,2,1,0 = Data read from/written to sound RAM

SOUNDDATA  EQUDS.B 1 ;Sound data register

* __7_____6_____5_____4_____3_____2_____1_____0___
*|     |     |     |     |     |     |     |     |
```

```
*|                                       |
*|          Low byte of sound address pointer          |
*|                                       |
*|_______|_______|_______|_______|_______|_______|_______|_______|
*         ^^^^^  SOUNDADRL Byte ^^^^^
```

```
* SOUNDADRL bits defined as follows
*    bit 7,6,5,4,3,2,1,0 = Address into sound RAM low byte

SOUNDADRL  EQUDS.B 1 ;Sound address pointer, low byte
```

```
* __7_______6_______5_______4_______3_______2_______1_______0___
*|      |      |      |      |      |      |      |      |
*|                                       |
*|          High byte of sound address pointer         |
*|                                       |
*|_______|_______|_______|_______|_______|_______|_______|_______|
*         ^^^^^  SOUNDADRH Byte ^^^^^
```

```
* SOUNDADRH bits defined as follows
*    bit 7,6,5,4,3,2,1,0 = Address into sound RAM high byte

SOUNDADRH  EQUDS.B 1 ;Sound address pointer, high byte
  EQUDS.B 1 ;Reserved for future system expansion
```

```
* __7_______6_______5_______4_______3_______2_______1_______0___
*|      |      |      |      |      |      |      |      |
*|      |      |      |Enable|Enable|Enable|Enable|Enable|
*|   0  |   0  |   0  |1/4sec| VBL  |switch| move | mouse|
*|      |      |      | ints | ints | ints | ints |      |
*|_______|_______|_______|_______|_______|_______|_______|_______|
*             ^^^^^  INTEN Byte ^^^^^
```

```
* INTEN bits defined as follows
*    bit 7= Must be 0
*    bit 6= Must be 0
*    bit 5= Must be 0
*    bit 4= 1 to enable 1/4 second interrupts
*    bit 3= 1 to enable VBL interrupts
*    bit 2= 1 to enable Mega // mouse switch interrupts
*    bit 1= 1 to enable Mega // mouse movement interrupts
*    bit 0= 1 to enable Mega // mouse operation

INTEN  EQUDS.B 1 ;Interrupt enable register
  EQUDS.B 1 ;Reserved for future system expansion
  EQUDS.B 1 ;Reserved for future system expansion
```

```
* __7_______6_______5_______4_______3_______2_______1_______0___
```

```
*|     |     |     |     |     |     |     |     |
*|                                   |
*|          Mega // mouse delta movement byte          |
*|                                   |
*|_____|_____|_____|_____|_____|_____|_____|_____|
*           ^^^^^  MMDELTAX Byte ^^^^^
```

```
* MMDELTAX bits defined as follows
*    bit 7,6,5,4,3,2,1,0 = delta movement in 2's complement notation

MMDELTAX  EQUDS.B 1 ;Mega // mouse delta X register
```

```
* __7_____6_____5_____4_____3_____2_____1_____0___
*|     |     |     |     |     |     |     |     |
*|                                   |
*|          Mega // mouse delta movement byte          |
*|                                   |
*|_____|_____|_____|_____|_____|_____|_____|_____|
*           ^^^^^  MMDELTAY Byte ^^^^^
```

```
* MMDELTAY bits defined as follows
*    bit 7,6,5,4,3,2,1,0 = delta movement in 2's complement notation

MMDELTAY  EQUDS.B 1 ;Mega // mouse delta Y register
```

```
* __7_____6_____5_____4_____3_____2_____1_____0___
*|     |     |     |     |     |     |     |     |
*|Self/ |MMouse|Status|Status|Status|Status|Status|Status|
*|burnin| last |  AN3 |1/4sec| VBL  |switch| move |system|
*| diags|button|      | int  | int  | int  | int  | IRQ  |
*|_____|_____|_____|_____|_____|_____|_____|_____|
*           ^^^^^  DIAGTYPE Byte ^^^^^
```

```
* DIAGTYPE bits defined as follows
*    bit 7= 0 if self diagnostics get used if BUTN0=1 and BUTN1=1
*    bit 7= 1 if burn-in diagnostics get used if BUTN0=1 and BUTN1=1
*    bits 6-0 = same as INTFLAG
```

```
* __7_____6_____5_____4_____3_____2_____1_____0___
*|     |     |     |     |     |     |     |     |
*|MMouse|MMouse|Status|Status|Status|Status|Status|Status|
*|  now | last |  AN3 |1/4sec| VBL  |switch| move |system|
*|button|button|      | int  | int  | int  | int  | IRQ  |
*|_____|_____|_____|_____|_____|_____|_____|_____|
*           ^^^^^  INTFLAG Byte ^^^^^
```

```
* INTFLAG bits defined as follows
*    bit 7= 1 if mouse button currently down
```

```
*    bit 6= 1 if mouse button was down on last read
*    bit 5= status of AN3
*    bit 4= 1 if 1/4 second interrupted
*    bit 3= 1 if VBL interrupted
*    bit 2= 1 if Mega // mouse switch interrupted
*    bit 1= 1 if Mega // mouse movement interrupted
*    bit 0= 1 if system IRQ line is asserted

DIAGTYPE  EQUDS * ;0/1 Self/burn-in diagnostics
INTFLAG  EQUDS.B 1 ;Interrupt flag register
CLRVBLINT  EQUDS.B 1 ;Clear the VBL and 3.75Hz interrupt flags
CLRXYINT  EQUDS.B 1 ;Clear Mega // mouse interrupt flags
  EQUDS.B 1 ;Reserved for future system expansion
  EQUDS.B 1 ;Reserved for future system expansion
  EQUDS.B 1 ;Reserved for future system expansion
  EQUDS.B 1 ;Reserved for future system expansion
  EQUDS.B 1 ;Reserved for future system expansion
  EQUDS.B 1 ;Reserved for future system expansion
  EQUDS.B 1 ;Reserved for future system expansion
TXTCLR  EQUDS.B 1 ;switch in graphics (not text)
TXTSET  EQUDS.B 1 ;switch in text (not graphics)
MIXCLR  EQUDS.B 1 ;clear mixed-mode
MIXSET  EQUDS.B 1 ;set mixed-mode (4 lines text)
TXTPAGE1  EQUDS.B 1 ;switch in text page 1
TXTPAGE2  EQUDS.B 1 ;switch in text page 2
LORES  EQUDS.B 1 ;low-resolution graphics
HIRES  EQUDS.B 1 ;high-resolution graphics
SETAN0  EQUDS.B 1 ;Clear annunciator 0
CLRAN0  EQUDS.B 1 ;Set annunciator 0
SETAN1  EQUDS.B 1 ;Clear annunciator1
CLRAN1  EQUDS.B 1 ;Set annunciator 1
SETAN2  EQUDS.B 1 ;Clear annunciator 2
CLRAN2  EQUDS.B 1 ;Set annunciator 2
SETAN3  EQUDS.B 1 ;Clear annunciator 3
CLRAN3  EQUDS.B 1 ;Set annunciator 3
BUTN3  EQUDS.B 1 ;Read switch 3
BUTN0  EQUDS.B 1 ;open apple key
BUTN1  EQUDS.B 1 ;closed apple key
BUTN2  EQUDS.B 1 ;Read switch 2
PADDL0  EQUDS.B 1 ;Read paddle 0
  EQUDS.B 1 ;Read paddle 1
  EQUDS.B 1 ;Read paddle 2
  EQUDS.B 1 ;Read paddle 3

* __7_____6_____5_____4_____3_____2_____1_____0___
*|     |     |     |     |     |     |     |     |
*| ALZP | PAGE2| RAMRD|RAMWRT| RDROM|LCBNK2|ROMBNK| INTCX|
*|status|status|status|status|status|status|status|status|
```

```
*|     |     |     |     |     |     |     |     |
*|_____|_____|_____|_____|_____|_____|_____|_____|
*         ^^^^^  STATEREG Status Byte ^^^^^

* STATEREG bits defined as follows
*    bit 7= ALZP status
*    bit 6= PAGE2 status
*    bit 5= RAMRD status
*    bit 4= RAMWRT status
*    bit 3= RDROM status (Read only RAM/ROM (0/1))

* IMPORTANT NOTE:
*          Do two reads to $C083 then change STATEREG
*          to change LCRAM/ROM banks (0/1) and still
*          have the language card write enabled.

*    bit 2= LCBNK2 status 0=LC bank 0 - 1=LC bank 1
*    bit 1= ROMBANK status
*    bit 0= INTCXROM status

STATEREG  EQUDS.B 1 ;State register
  EQUDS.B 1 ;Reserved for future system expansion
  EQUDS.B 1 ;Reserved for future system expansion
  EQUDS.B 1 ;Reserved for future system expansion
  EQUDS.B 1 ;Reserved for future system expansion
TESTREG  EQUDS.B 1 ;Test mode bit register
CLRTM  EQUDS.B 1 ;Clear test mode
ENTM  EQUDS.B 1 ;Enable test mode
PTRIG  EQUDS.B 1 ;trigger the paddles
  EQUDS.B 15 ;ROM interrupt code jump table

  EQUDS.B 1 ;Sel LC RAM bank2 rd, wrt protect LC RAM
ROMIN  EQUDS.B 1 ;Enable ROM read, 2 reads write enb LC RAM
  EQUDS.B 1 ;Enable ROM read, wrt protect LC RAM
LCBANK2  EQUDS.B 1 ;Sel LC RAM bank2, 2 rds wrt enb LC RAM

  EQUDS.B 1 ;Sel LC RAM bank2 rd, wrt protect LC RAM
  EQUDS.B 1 ;Enable ROM read, 2 reads wrt enb LC RAM
  EQUDS.B 1 ;Enable ROM read, wrt protect LC RAM
  EQUDS.B 1 ;Sel LC RAM bank2, 2 rds wrt enb LC RAM

  EQUDS.B 1 ;Sel LC RAM bank1 rd, wrt protect LC RAM
  EQUDS.B 1 ;Enable ROM read, 2 reads wrt enb LC RAM
  EQUDS.B 1 ;Enable ROM read, wrt protect LC RAM
LCBANK1  EQUDS.B 1 ;Sel LC RAM bank1, 2 rds wrt enb LC RAM

  EQUDS.B 1 ;Sel LC RAM bank1 rd, wrt protect LC RAM
  EQUDS.B 1 ;Enable ROM read, 2 reads wrt enb LC RAM
```

```
  EQUDS.B 1 ;Enable ROM read, wrt protect LC RAM
  EQUDS.B 1 ;Sel LC RAM bank1, 2 rds wrt enb LC RAM

  ENDSECT

CLRROM  EQU $CFFF ;switch out $C8 ROMs


* Applesoft equates

FIRST  EQU $F0 ;Used by Applesoft plotting routines
CHKCOM  EQU $DEBE ;Parse for comma to continue
TOOBIG  EQU $F206 ;If argument is too big jump here
ATFNS2  EQU $F20C ;Return point for HLIN fix
BASIC  EQU $E000 ;BASIC entry point
BASIC2  EQU $E003 ;BASIC warm entry point
GETBYT  EQU $E6F8 ;Get next user entered byte


* Permanent data in screenholes
*
* Note: these screenholes are only used by
* the 80 column firmware if an 80 column card
* is detected or if the user explicitly activates
* the firmware.  If the 80 column card is not
* present, only MODE is trashed on RESET.
*
* The success of these routines rely on the
* fact that if 80 column store is on (as it
* normally is during 80 column operation), that
* text page 1 is switched in.  Do not call the
* video firmware if video page 2 is switched in!!

MSLOT  EQU $7F8 ;=$Cn ;n=slot using $C800

OLDCH  EQU $478+3 ;LAST CH used by video firmware
MODE  EQU $4F8+3 ;video firmware operating mode
OURCH  EQU $578+3 ;80 column CH
OURCV  EQU $5F8+3 ;80 column CV
CHAR  EQU $678+3 ;character to be printed/read
XCOORD  EQU $6F8+3 ;GOTOXY X-coord (pascal only)
TEMP1  EQU $778+3 ;temp
OLDBASL  EQU $778+3 ;last BASL (pascal only)
OLDBASH  EQU $7F8+3 ;last BASH (pascal only)

*
*
* BASIC VMODE BITS
```

```
*
* 0....... - BASIC active
* 1....... - Pascal active
* .0...... - Print mouse chars
* .1...... - Don't print mouse characters
* ..0..... - Print control characters
* ..1..... - Don't print ctrl chars
* ...0.... -
* ...1.... -
* ....0... - Print control characters
* ....1... - Don't print ctrl chars.
* .....0.. -
* .....1.. -
* ......0. - Scroll normally
* ......1. - 1200 baud: home instead of scroll
* .......0 - Mouse text inactive
* .......1 - Mouse text active
*
M_6  EQU $40 ;
M_CTL2  EQU $20 ;Don't print controls
M_4  EQU $10 ;
M_CTL  EQU $08 ;Don't print controls
M_2  EQU $04 ;
M_1  EQU $02 ;
M_MOUSE  EQU $01 ;Don't print mouse chars
*
* Pascal Mode Bits
*
* 0....... - BASIC active
* 1....... - Pascal active
* .0...... - Print mouse chars
* .1...... - Don't print mouse characters
* ..0..... -
* ..1..... -
* ...0.... - Cursor always on
* ...1.... - Cursor always off
* ....0... - GOTOXY n/a
* ....1... - GOTOXY in progress
* .....0.. - Normal Video
* .....1.. - Inverse Video
* ......0. - Scroll normally
* ......1. - 1200 baud: home instead of scroll
* .......0 -
* .......1 -
*
M_PASCAL  EQU $80 ;Pascal active
M_CURSOR  EQU $10 ;Don't print cursor
M_GOXY  EQU $08 ;GOTOXY IN PROGRESS
```

```
M_VMODE  EQU $04
M_PAS1_0  EQU $02 ;Pascal 1.0 mode



 EJECT
 TITLE 'Bank $FF Variables2'

*****************************************************
*                                                   *
*              Bank $FF Variables2                  *
*                                                   *
*                    by                             *
*              Fern Bachman    1985-1987            *
*              Joe Bo          1987-1989            *
*                                                   *
*     Copyright Apple Computer, Inc. 1985-1989      *
*              All Rights Reserved.                 *
*                                                   *
*****************************************************


* Tool set numbers and call numbers used
* either in the Control Panel or monitor
* subroutines.

MTN  EQU $03 ;Miscellaneous Tool Number
RTHFN  EQU $0D ;Read Time Hex Function Number
WTHFN  EQU $0E ;Write Time Hex Function Number
RATFN  EQU $0F ;Read Ascii Time Function Number

RDN  EQU $0D ;RAM Disk tool number
DPCN  EQU $0A ;Do Protocol Converter call number
SRDSZN  EQU $0B ;Set RAM Disk size function number
GRDSZN  EQU $0C ;Get RAM Disk size function number

MMN  EQU $02 ;Memory manager tool number
FMEMN  EQU $1B ;Free memory in system function number
TMEMN  EQU $1D ;Total memory in system function number

EMTN  EQU $06 ;Event manager tool number
ACTFN  EQU $06 ;Active function number
GNEFN  EQU $0A ;Get next event function number
EAFN  EQU $0B ;Event available function number


IMTN  EQU $0B ;Integer math routines tool number
LONGMUL  EQU $0C ;32 bit multiply function number
```

```
LONGDIV  EQU $0D ;32 bit divide function number
LONG2DEC  EQU $27 ;Convert hex to decimal ASCII string
DEC2LONG  EQU $29 ;Convert decimal ASCII string to hex


DSKTN  EQU $05 ;Desk accessory manager tool number
SVESCRNFN  EQU $09 ;Save 80 column screen function number
RSTSCRNFN  EQU $0A ;Restore 80 column screen function number


SETHEART  EQU $1203 ;Put task in heartbeat queue
DELHEART  EQU $1303 ;Remove task from heartbeat queue


INSTALLCDA EQU $0F05 ;Tool # + Function # * 256

GOTOMON  EQU $F7 ;Defined below
RETURNFROM EQU GOTOMON+4 ;Return from monitor code
RETURNBANK EQU GOTOMON+8 ;Modify instruction

* GOTOMON contains the following code
*       CLI             ;Reenable interrupts
*       JMP MONZ        ;Enter the monitor
*       SEI             ;Disable interrupts to return to calling bank
*       JMP =RETURNPOINT ;Return to calling bank

MICROPOWER EQU $A5 ;Power up byte for keyboard micro
DECAY  EQU $0250 ;Decay rate for NEWBELL1 sounds
MICRORAMPWR EQU $51 ;Keyboard micro power up byte location
DURBELL  EQU $2000 ;Duration of bell sound
TOGPATTERN EQU $8888 ;Speaker toggle pattern
INITDISMGR EQU $FE0000 ;Initialize DISPATCH handler
MEMORYPEEKER EQU $FE0004 ;Install memory peeker desk acc
TODESKMGR  EQU $FE0008 ;Private CDA entry point
APTALKINIT EQU $FE000C ;Coldstart vector for AppleTalk
APTALKRESET EQU $FE0010 ;Warmstart vector for AppleTalk
ADBSENDTL  EQU $FE0014 ;Vctr to send COMMAND to Keyuc
BOOTAPTALK1 EQU $C1E0 ;Boot AppleTalk from slot1
BOOTAPTALK2 EQU $C2E0 ;Boot AppleTalk from slot2
DIAGS  EQU $FF6400 ;Diagnostics start here
PROCONINIT EQU $FF5600 ;Initialize protocol converter
SERBASPATCH EQU $FF0000 ;Serial code BASIC patch
SERPASPATCH EQU $FF0003 ;Serial code PASCAL patch
SEREXTPATCH EQU $FF0006 ;Serial code EXTERNAL patch


* The bank $F8 expansion rom must have the following
* information located at the base of the rom.
```

```
*
* $F80000-F80007 ... An ASCII string 'ROMTOOLS'
* $F80008-F80009 ... Version Number
* $F8000A-F8001F ... Pad Area
* $F80020 ..... Entry Point
*
* The entry point is called in 16 bit native mode.
* (e=0, m=0, x=0) The entry point will execute
* code to install any rom based extensions to the
* tool set and return with an 'RTL' instruction
* with the carry flag cleared.  If the rom is found,
* the version number will be returned in the 16 bit
* C register after the 'RTL' is executed.
*
* Note that the string at $F80000 must have the
* msb set to a '1'.
*


EXROMSTR  EQU $F80000 ; Expansion rom string 'ROMTOOLS'
EXROMVER  EQU $F80008 ; Expansion rom version
EXROMPAD  EQU $F8000A ; Pad to $F8001F
EXROMENTRY EQU $F80020 ; Expansion rom entry point


* Bank $E1 equates

E1_VARS  EQUSECT $E102C0
BRAMBUFFER EQUDS * ;1 page image of BATTERYRAM

* Port 1 selections

USERPORT1  EQUDS.B 1 ;Device Connected: Printer
  EQUDS.B 1 ;Line Length: Unlimited
  EQUDS.B 1 ;Delete First LF after CR: No
  EQUDS.B 1 ;Add LF after CR: Yes
  EQUDS.B 1 ;Echo: No
  EQUDS.B 1 ;Buffering: No
  EQUDS.B 1 ;Baud: 9600
  EQUDS.B 1 ;Data/Stop Bits: 8/1
  EQUDS.B 1 ;Parity: None
  EQUDS.B 1 ;DCD Handshake: Yes
  EQUDS.B 1 ;DSR Handshake: Yes
  EQUDS.B 1 ;XON/XOFF Handshake: No

* Port 2 selections

USERPORT2  EQUDS.B 1 ;Device Connected: Modem
```

```
  EQUDS.B 1 ;Line Length: Unlimited
  EQUDS.B 1 ;Delete First LF after CR: No
  EQUDS.B 1 ;Add LF after CR: No
  EQUDS.B 1 ;Echo: No
  EQUDS.B 1 ;Buffering: No
  EQUDS.B 1 ;Baud: 1200
  EQUDS.B 1 ;Data/Stop Bits: 8/1
  EQUDS.B 1 ;Parity: None
  EQUDS.B 1 ;DCD Handshake: Yes
  EQUDS.B 1 ;DSR Handshake: Yes
  EQUDS.B 1 ;XON/XOFF Handshake: No

* Display selections

USERCOLOR  EQUDS.B 1 ;Type: Color
USERCOL  EQUDS.B 1 ;Columns: 40
USERTEXT  EQUDS.B 1 ;Text: (white)
USERBACK  EQUDS.B 1 ;Background: (medium blue)
USERBORDER EQUDS.B 1 ;Border: (medium blue)
USERHERTZ  EQUDS.B 1 ;Hertz: 60

* Sound selections

USERVOL  EQUDS.B 1 ;Volume: |-----*---------|
USERBELL  EQUDS.B 1 ; Pitch: |------*--------|

* Speed selections

USERSPEED  EQUDS.B 1 ;Speed: Fast

* Slots' selections

USERSLT1  EQUDS.B 1 ;Slot 1: Port 1
USERSLT2  EQUDS.B 1 ;Slot 2: Port 2
USERSLT3  EQUDS.B 1 ;Slot 3: Text Display
USERSLT4  EQUDS.B 1 ;Slot 4: Mouse
USERSLT5  EQUDS.B 1 ;Slot 5: Smart Port
USERSLT6  EQUDS.B 1 ;Slot 6: Disk Port
USERSLT7  EQUDS.B 1 ;Slot 7: Your Card
USERSLTST  EQUDS.B 1 ;Startup Slot: Scan

* Language selections

USERDISP  EQUDS.B 1 ;Text Display: English (U.S.)
USERLANG  EQUDS.B 1 ;Keyboard: English (U.S.)
USERKBUFF  EQUDS.B 1 ;Keyboard Buffering: No
USERRSPD  EQUDS.B 1 ;Repeat Speed: |-------*-------|
USERRDLY  EQUDS.B 1 ;Repeat Delay: |-----*---------|
```

```
;DFB $02 ;Double Click: |-----*---------|
USERFLASH  EQUDS.B 1 ;Cursor Flash: |-----*---------|
USERCAPS  EQUDS.B 1 ;Shift Caps/Lower Case: No
USERFSPDE  EQUDS.B 1 ;Fast Space/Delete Keys: No
USERDUAL  EQUDS.B 1 ;Dual Speed Keys: Normal
USERD1EJ  EQUDS.B 1 ;Disk1 ejection option, 0->1, 1->2, ....
USERD2EJ  EQUDS.B 1 ;Disk2 ejection option, 0->1, 1->2, ....

* Clock selections

USERDATE  EQUDS.B 1 ;Format: MM/DD/YY
USERAMPM  EQUDS.B 1 ;Format: AM-PM

* Ramdisk selections

USERMRAM  EQUDS.B 1 ;Minimum RAM for Ramdisk option number
USERXRAM  EQUDS.B 1 ;Maximum RAM for Ramdisk option number
USERRSTRAM EQUDS.B 1 ;Reset Ram Disk size during next reset
;USERXRAM DFB $00

* Mouse Selections

USERHMR  EQUDS.B 1 ;Mouse Tracking: |*----------|
USERDBCK  EQUDS.B 1 ;Double Click:   |-----*--------|
USERMDLY  EQUDS.B 1 ;Start Delay:    |-----*--------|
USERSPRT  EQUDS.B 1 ;Acceleration:   |-----*--------|
USERMAXSP  EQUDS.B 1 ;Maximum Speed:  |-----*--------|

LANGS  EQUDS.B 9 ;Count/Number of languages

LAYOUTS  EQUDS.B 17 ;Count/Numbers of layouts

SPCLSLT1  EQUDS.B 1 ;Keep track of real slt1 setup
BRAMDAACT  EQUDS.B 1 ;$00/<>0 = NOT install/install monitor/memory peeker

  EQUORG E1_VARS+$78 ;Pad to Cntl Panel password

PASSWDBUFF EQUDS *
USRPASSWD0 EQUDS.B 1 ;
  EQUDS.B 1
  EQUDS.B 1
  EQUDS.B 1
  EQUDS.B 1
  EQUDS.B 1
  EQUDS.B 1
USRPASSWD7 EQUDS.B 1 ;=0 then default password

  EQUORG E1_VARS+$80 ;Pad to AppleTalk node number
```

```
APTALKNODE EQUDS.B 1 ;AppleTalk node number

OSVARIABLES EQUDS.B 32 ;Operating system variables

  EQUORG E1_VARS+$FC ;Pad to check sums

BRAMCHKSUM1 EQUDS.W 1 ;Main BATTERYRAM checksum
BRAMCHKSUM2 EQUDS.W 1 ;= BRAMCHKSUM1 EOR BRAMCNST

  ENDSECT
  EJECT
  EQUSECT $E10000 ;Monitor $E1 permanent storage area
E1_VECTORS EQUDS *
* Important vectors the user may change
* RAM space from locations $0000->$02B7 are allocated for
* monitor and RAM vectors use.

DISPATCH1  EQUDS.L 1 ;Jmp vector to Tool Locator1
DISPATCH2  EQUDS.L 1 ;Jmp vector to Tool Locator2
UDISPATCH1 EQUDS.L 1 ;Jmp vector to User's tool locator1
UDISPATCH2 EQUDS.L 1 ;Jmp vector to User's tool locator2
INTMGRV  EQUDS.L 1 ;Jmp vector to Interrupt manager
COPMGRV  EQUDS.L 1 ;Jmp vector to COP manager
ABORTMGRV  EQUDS.L 1 ;Jmp vector to ABORT manager
SYSDMGRV  EQUDS.L 1 ;Jmp vector to system death manager

* Tool/internal interrupt vectors

IRQ_APTLK  EQUDS.L 1 ;Jmp vector to APTLK interrupt handler
IRQ_SERIAL EQUDS.L 1 ;Jmp vector to serial port int handler
IRQ_SCAN  EQUDS.L 1 ;Jmp vector to scan line interrupt handler
IRQ_SOUND  EQUDS.L 1 ;Jmp vector to sound interrupt handler
IRQ_VBL  EQUDS.L 1 ;Jmp vector to VBL interrupt handler
IRQ_MOUSE  EQUDS.L 1 ;Jmp vector to mouse interrupt handler
IRQ_QTR  EQUDS.L 1 ;Jmp vector to 1/4 sec interrupt handler
IRQ_KBD  EQUDS.L 1 ;Jmp vector to keyboard interrupt handler
IRQ_RESPONSE EQUDS.L 1 ;Jmp vctr to FDB response byte int hndlr
IRQ_SRQ  EQUDS.L 1 ;Jmp vector to SRQ int handler
IRQ_DSKACC EQUDS.L 1 ;Jmp vector to desk accessory int handler
IRQ_FLUSH  EQUDS.L 1 ;Jmp vector to flush buffer int handler
IRQ_MICRO  EQUDS.L 1 ;Jmp vector to key micro int handler
IRQ_1SEC  EQUDS.L 1 ;Jmp vector to 1 second interrupt handler
IRQ_EXT  EQUDS.L 1 ;Jmp vector to EXT VGC interrupt handler
IRQ_OTHER  EQUDS.L 1 ;Jmp vector to other interrupt handler

CUPDATE  EQUDS.L 1 ;Jmp vector to cursor update handler
```

```
INCBUSYFLG EQUDS.L 1 ;Jmp vector to increment BUSYFLAG routine
DECBUSYFLG EQUDS.L 1 ;Jmp vector to decrement BUSYFLAG routine

BELLVECTOR EQUDS.L 1 ;Jmp vector to BELL routine in use

BREAKVECTOR EQUDS.L 1 ;Breaks go through here for debuggers

TRACEVECTOR EQUDS.L 1 ;Jmp vector to trace code
STEPVECTOR EQUDS.L 1 ;Jmp vector to step code

ROMTOOLHOOK EQUDS.L 1 ;Jmp vector to ROM expansion tools

  EQUORG E1_VECTORS+$80 ;Pad for more vectors
IRQ_HERE01 EQUDS *

IRQ_LEN  EQU IRQ_HERE01-IRQ_APTLK


* End of user changable interrupt vectors.


TOWRITEBR  EQUDS.L 1 ;Jmp vector to write BATTERYRAM routine
TOREADBR  EQUDS.L 1 ;Jmp vector to read BATTERYRAM routine
TOWRITETIME EQUDS.L 1 ;Jmp vector to write time routine
TOREADTIME EQUDS.L 1 ;Jmp vector to read time routine
TOCTRL_PANEL EQUDS.L 1 ;Jmp vctr to Control Panel desk accessory
TOBRAMSETUP EQUDS.L 1 ;Jmp vector for ext Menu to setup stuff
TOPRINTMSG8 EQUDS.L 1 ;Jmp vector to print a msg- 8 bit entry
TOPRINTMSG16 EQUDS.L 1 ;Jmp vector to print a msg- 16 bit entry


CTRLYVECTOR EQUDS.L 1 ;Jmp vector for CTRL-Y (native mode)
TOTEXTPG2DA EQUDS.L 1 ;Text page 2 desk acc routine


  EQUORG E1_VECTORS+$A8 ;Pad for more vectors


OSVECTORS  EQUDS * ;Operating system 4 byte jump vectors
PRO16MLI  EQUDS.L 1 ;1st OS vector is PRODOS16 MLI entry point
CHAINVECTOR EQUDS.L 1 ;Called before application launch
  EQUDS.B 10 ;10 reserved bytes
  EQUDS.W 1 ;Guaranteed 2 zeros
OS_KIND  EQUDS.B 1 ;0/1-P8/P16 current OS running
OS_BOOT  EQUDS.B 1 ;0/1-P8/P16 booted OS
OS_FLAGS  EQUDS.W 1 ;Prodos 16 flags
```

```
MSGPOINTER EQUDS.B 3 ;Pointer to messages table


* IRQ.SECFLAG, IRQ.INTFLAG, IRQ.DATAREG are setup
* during interrupt time.  They are defined as follows
*
* IRQ.INTFLAG = value of INTFLAG read during interrupt
*               processing.
* IRQ.DATAREG = value of DATAREG read during interrupt
*               processing.
* IRQ.KMSTATUS value of KMSTATUS read during interrupt
*              processing.
* IRQ.SECFLAG =0 if all interrupts processed by handlers
*              =<>0 if any interrupt not processed by handlers
* IRQ.SERIAL1 is set up by serial interrupt manager
*              for the user to read.
* IRQ.SERIAL2 is set up by serial interrupt manager
*              for the user to read.
* IRQ.APTLKHI is set up by AppleTalk interrupt manager
*              for the user to read.
*
* IRQ.SECFLAG, IRQ.INTFLAG, IRQ.DATAREG, IRQ.KMSTATUS are set
* to zero in irq.done2.

IRQ_INTFLAG EQUDS.B 1 ;Status of INTFLAG at interrupt time
IRQ_DATAREG EQUDS.B 1 ;Status of DATAREG at interrupt time
IRQ_KMSTATUS EQUDS.B 1 ;Status of KMSTATUS at interrupt time
IRQ_SECFLAG EQUDS.B 1 ;0/<>0 = int not processed/processed
IRQ_SERIAL1 EQUDS.B 1 ;Interrupt info for users
IRQ_SERIAL2 EQUDS.B 1 ;Interrupt info for users
IRQ_APTLKHI EQUDS.B 1 ;Interrupt info for users
IRQ_VOLUME EQUDS.B 1 ;Volume used during interrupt time
IRQ_ACTIVE EQUDS.B 1 ;<>0 if called during irq
IRQ_SOUNDDATA EQUDS.B 1 ;Sound data register at int time


  EQUORG E1_VECTORS+$CF ;Pad for more irq bytes


SIOFLAG  EQUDS.B 1 ;Slts enabled via init call to Misc Tl Loc
INMASK  EQUDS.B 3 ;AND mask / OR mask / slot #
OUTMASK  EQUDS.B 3 ;AND mask / OR mask / slot #


ENSONIQFLG EQUDS.B 1 ;0=chip installed / 1=no chip installed
IRQ_MOUSVBL EQUDS.B 1 ;0 if mouse not using VBL
```

```
HBPTR  EQUDS.L 1 ;Heart beat pointer to 1st item in list
TICKCNT  EQUDS.L 1 ;Tick count updated by heart beat
MOUSESLOT  EQUDS.B 1 ;Mouse slot in use

EVMGRDATA  EQUDS.B 30 ;Event manager permanent information

BUSYFLAG  EQUDS.W 1 ;System busy flag


SCCSTAT  EQUDS.B 1 ;=0 then not SCC int- <>0 then SCC int
  EQUDS.B 1 ;Reserved
SCCFLAG2  EQUDS.B 1 ;Aptalk int mask -$07=chanB-$38=chanA
SERFLAG  EQUDS.B 1 ;Ser int msk-$07=chanB-$38=chanA-$3F=both
SCC1  EQUDS.B 1 ;1st data byte read in by int handler
SCC2  EQUDS.B 1 ;2nd data byte read in by int handler
  EQUDS.B 1 ;Reserved


* Interrupt uses all variables with 'irq.'
*
* Interrupt buffer is used by monitor to save
* ASCIITIME in during '=T' calls.  Interrupts
* are disabled during this time so it is safe
* to use the same buffer.

ASCIITIME  EQUDS * ;Time in ASCII 'mm/dd/yy hh:mm:ss am/pm'
IRQ_A  EQUDS.B 2
IRQ_X  EQUDS.B 2
IRQ_Y  EQUDS.B 2
IRQ_S  EQUDS.B 2
IRQ_D  EQUDS.B 2
IRQ_P  EQUDS.B 1
IRQ_DB  EQUDS.B 1 ;irq.DB and irq.e canNOT be separated

IRQ_HERE02 EQUDS *
IRQLEN  EQU IRQ_HERE02-IRQ_A

IRQ_E  EQUDS.B 1
IRQ_K  EQUDS.B 1
IRQ_PC  EQUDS.B 2
IRQ_STATE  EQUDS.B 1
IRQ_SHADOW EQUDS.B 2
IRQ_MSLOT  EQUDS.B 1
IRQ_EMULSTKPTR EQUDS.B 1 ;emulation stack pointer, $010100


  EQUORG E1_VECTORS+$120 ;Pad for more interrupt info
```

```
AVAL  EQUDS.W 1 ;A register value
XVAL  EQUDS.W 1 ;X register value
YVAL  EQUDS.W 1 ;Y register value
SVAL  EQUDS.W 1 ;S (stack pointer) value
DVAL  EQUDS.W 1 ;D (direct (zero page) register) value
PVAL  EQUDS.B 1 ;P (processor status) value
BVAL  EQUDS.B 1 ;B (data bank register) value
KVAL  EQUDS.B 1 ;K (program bank register) value
MVAL  EQUDS.B 1 ;M (memory status) value
QVAL  EQUDS.B 1 ;Q (bit7=speed - bits6-0 match shadow reg)
LVAL  EQUDS.B 1 ;L (language card bank 0/1 selected) value
MPVAL  EQUDS.B 1 ;m memory access mode bit value
XPVAL  EQUDS.B 1 ;x index mode bit value
EPVAL  EQUDS.B 1 ;e (emulation mode status) value
DDVAL  EQUDS.B 1 ;d (disassembly option) value
ASCFILTER  EQUDS.B 1 ;F (filter mask for ASCII input in mon)

CURSOR  EQUDS.B 1 ;Cursor
NXTCUR  EQUDS.B 1 ;Cursor
CHGCURFLG  EQUDS.B 1 ;<>0=change cursor/ 0=no change cursor

SYSTEMSPD  EQUDS.B 1 ;$00=slow speed -- $80=high speed mask
SLOTSFLG  EQUDS.B 1 ;SLTROMSEL value for reset
;ORGMEMORY DFB 0 ;Used in executed BRK/COP disassembly

E1PCLH  EQUDS.W 1 ;Program counter for 'G'o routine
GOS  EQUDS.W 1 ;'S' value to return to after a 'G'o cmd

BANKS  EQUDS.B 1 ;Source bank
BANKD  EQUDS.B 1 ;Destination bank

EXTENDBNK  EQUDS.W 1 ;Extend+bank bytes that user entered
DIGITCNT  EQUDS.B 1 ;Used by GETNUM - how many digits entered

C1FLAG  EQUDS.B 1 ;Serial code uses as indicator
C2FLAG  EQUDS.B 1 ;Serial code uses as indicator
CURSHIFTER EQUDS.B 1 ;Shifts to 0 which causes cursor to flash

EXTENDBNKS EQUDS.W 1 ;Saved extend+bank bytes entered


IRQ_3031SOUND EQUDS.L 1 ;Jmp vector to Apple sound irqs


  EQUORG E1_VECTORS+$14C ;Pad to KEYTASKHDR

KEYTASKHDR0 EQUDS.B 12 ;Key press task handler buffer
```

```
EVMKEYBUFF EQUDS.B 16 ;16 bytes required
WHATEVENT  EQU EVMKEYBUFF ;1st byte is event number
EVMKEY  EQU EVMKEYBUFF+2 ;ASCII key pressed is here

RDMEMDATA  EQUDS.B 1 ;Powerup byte read from keyboard micro

TOGGLEPAT  EQUDS.W 1 ;Speaker toggle ptrn for NEWBELL1 sounds
DURATION  EQUDS.W 1 ;Hold time counter for NEWBELL1
PITCH  EQUDS.W 1 ;User chosen pitch value for NEWBELL1
VOLUME  EQUDS.B 1 ;Volume counter for NEWBELL1


TP2THDR  EQUDS.L 1 ;Task header for text page 2 copier
TP2TCNT  EQUDS.W 1 ;VBL count
TP2TSIG  EQUDS.W 1 ;Signature byte
TP2TJMP  EQUDS.L 1 ;Jmp vector
TP2MODE  EQUDS.B 1 ;0/1 text page 2 copier off/on


* ROMDAACT is a flag which is reset on to
* $00 on power up only.  It is set <>0 if
* and only if the user types in the '#'
* command in the monitor.  The '#' command
* installs the monitor desk accessory and
* the memory peeker desk accessory.
*
* If ROMDAACT is <>0 the monitor on coldstart
* only will reinstall the monitor/memory peeker
* desk accessories.

ROMDAACT  EQUDS.B 1 ;0=Do not activate <>0=activate


  EQUORG E1_VECTORS+$180 ;Fill to TOBUSYSTRIP


TOBUSYSTRIP EQUDS.L 1 ;JSL to dec busy flag
TOSTRIP  EQUDS.L 1 ;JMP vector to oErrOut routine


NOIDEA  EQUDS.L 1 ;Ask Steve ?????
STEPSP  EQUDS.W 1 ;Step/Trace return address after each inst
KEYUCVER  EQUDS.B 1 ;KEyUC VERSION #
ORGMEMORY  EQUDS.B 1 ;Used in executed BRK/COP disassembly
LOWABSX  EQUDS.B 1 ;Mouse x/y absolute position
LOWABSY  EQUDS.B 1
HIGHABSX  EQUDS.B 1
```

```
HIGHABSY  EQUDS.B 1
MSEINFO  EQUDS.B 1 ;Bit 6, 7 for Y clamps
MSESPARE  EQUDS.B 1 ;Reserved as temp for mouse routine
MSESTAT  EQUDS.B 1 ;Button 0/1 interrupt status byte
MSEMODE  EQUDS.B 1 ;Mode byte
MDISPATCH  EQUDS.L 1 ;Jmp vector to monitor dispatch
MAINSIDEPATCH EQUDS.L 1 ;Jmp vector to main side dispatch
GUS  EQUDS.L 1 ;????
DEBUGB  EQUDS.B 1 ;Rom debugger uses for bank saving
GSC  EQUDS.B 1 ;Bit7=1, GSc; Bit6=1, not install cntlpanl
CLRHOLES  EQUDS.B 1 ;0=CLEAR, <>0 NOT CLEAR
HIGHBANK  EQUDS.B 1 ;Used to handle bank crossing
FLIPSCRN  EQUDS.B 1 ;Bit7, flip between txtpage1 and 'brk scrn
DEFAULTSLT EQUDS.L 1 ;Jmp vctr to restor default slt assignment
RESTORSLT  EQUDS.L 1 ;Jmp vctr to restor system slot assignment
CYAVER  EQUDS.B 1 ;$00 = FPI, $40 = CYA
MIDIINPUTPOLL EQUDS.B 8 ;For MidiTool vector
JSX  EQUDS.B 1 ;Bit7:real time of jsx, bit6: regs display
SCRNSTAT  EQUDS.B 1 ;Store the display before brk, used in 'r'
LLTOOLBOXVCTR EQUDS.L 1 ;Vctr to patch low level toolbox routines
APTALKPORT EQUDS.B 1 ;0=NONE, 1=PORT1, 2=PORT2
APTALKBOOT EQUDS.B 1 ;0=NONE, 1=$C100, 2=$C200, 3=$C300 .....
APTALKRPMM EQUDS.B 1 ;0=NONE, 1=$C100, 2=$C200, 3=$C300 .....

* The following bytes are mainly used in the step/trace
* monitor commands

USEROPC  EQUDS.B 1 ;User opcode
USEROPRL  EQUDS.B 1 ;User low operand
USEROPRH  EQUDS.B 1 ;User high operand
USEROPRK  EQUDS.B 1 ;User program bank
JMPNOBRA  EQUDS.B 1 ;JML, $5c, to jmp to handle user nobranch opcode
NOBRAPCL  EQUDS.B 1 ;Nobranch address, pcl
NOBRAPCH  EQUDS.B 1 ;Nobranch address, pch
NOBRAPCK  EQUDS.B 1 ;Nobranch address, pck
JMPBRACH  EQUDS.B 1 ;JML, $5c, to jmp to handle user branch opcode
BRACHPCL  EQUDS.B 1 ;Branch address, pcl
BRACHPCH  EQUDS.B 1 ;Branch address, pch
BRACHPCK  EQUDS.B 1 ;Branch address, pck
KPMNEMSP  EQUDS.B 1 ;{MSA 4/28/89} Stores Emul SP in Step/Trace Code


  EQUORG E1_VECTORS+$200 ;Fill to new Jmp vectors, publish out


TOMEMMOVER EQUDS.L 1 ;Jump vector to memory mover
TOSETSPEED EQUDS.L 1 ;Jump vector to set system speed
SLTARBITER EQUDS.L 1 ;Jump vector to do slot arbitration
```

```
SERBASIC  EQUDS.L 1 ;Jump vector for serial BASIC
SERPASCAL  EQUDS.L 1 ;Jump vector for serial PASCAL
SEREXTERL  EQUDS.L 1 ;Jump vector for serial external patch
INDINTVCTR EQUDS.L 1 ;Hardware independent interrupt vctr
IRQ_MIDI  EQUDS.L 1 ;Special vector for Midi Interrupt polling


  EQUORG E1_VECTORS+$2B8 ;Fill to mouse clamps


LOWXCLAMP  EQUDS.W 1 ;Low X clamp
LOWYCLAMP  EQUDS.W 1 ;Low Y clamp
HIXCLAMP  EQUDS.W 1 ;High X clamp
HIYCLAMP  EQUDS.W 1 ;High Y clamp

;
; The battery RAM buffer starts right after the mouse clamps.  It
; is from $e102c0 - $e103bf with a total of 256 bytes.
;
; Be careful, not allowed to fill anything between here !!!
;

  EQUORG E1_VECTORS+$03D0 ;Fill to ADB absolute device scaling

ADBSCALE  EQUDS.B 12 ;Absolute device scale data area
ADBCPLTVCTR EQUDS.L 1 ;Completion vector address


  EQUORG E1_VECTORS+$03E0

TIMEOUT  EQUDS.B 1 ;Timeout counter
CLKRDATA  EQUDS.L 1 ;Time read from clock in seconds
CLKWDATA  EQUDS.L 1 ;Time to write to clock in seconds
TIMEBUFFER EQUDS.B 23 ;Working buffer to time tools


  EQUORG E1_VECTORS+$0FB0 ;// gs uses these area instead of screen holes

SMARTPORT  EQUDS.B 25 ;gs smartport variables, $e10fb0-$e10fc8

  EQUORG E1_VECTORS+$0FD0 ;
MTHANDLE  EQUDS.L 1 ;Storage for ID tag handle
KBDIRQFLAG EQUDS.B 1 ;Keyboard interrupt flag

  EQUORG E1_VECTORS+$0FD6 ;Fill to ADB variables
FLAGS  EQUDS.B 1 ;Flags used in ADB tool, cmd pending...
STATUSTMP  EQUDS.B 1 ;Holds any uC pending status
SRQNUM  EQUDS.B 1 ;# of devices installed in SRQ chain
```

```
SRQCNT  EQUDS.B 1 ;# of devices polled since last data
SRQPTR  EQUDS.B 1 ;Pointer of next device in SRQ list
SRQMRU  EQUDS.B 1 ;Most recently used device (top of list)


SERPWRUP  EQU $E10FFE ;+E10FFF = serial port pwr up bytes
RAMDSKPWRUP EQU $E115FF ;<>0 if powered up before
SYSVOLLOC  EQU $E11DB0 ;Volume for sound tools is in hi byte


  EQUORG E1_VECTORS+$154A ;ADB devices completion vectors
ADBVCTRS  EQUDS.B 64 ;MAX 16 devices * 4 = 64


  EQUORG E1_VECTORS+$1DD0 ;Fill to absolute clamp values
ABSCLAMPS  EQUDS.B 8 ;Absolute device clamp values

  ENDSECT


ERRCODE  EQU $0911 ;System can't synch with ADB keyuc error code
CMDREG  EQU $C026 ;Same as datareg
CMDFULL  EQU $01 ;Command reg full bit (in keygloo status)
DATAFULL  EQU $20 ;Data reg full bit (in keygloo status)


* Command word for the G_MOVE_INFO routine

MOVEBLKCMD EQU $0800 ;Tell hardware to do block move option

* The MOVEVALIDITYMASK is a mask used to determine if a
* user's command word is using any invalid (reserved bits or not)
* This value changes as reserved bits become active !!!

MOVEVALIDITYMASK EQU $F7F0 ;Invalid move if any of these bits = 1

* Most common command

MOVESINCDINC EQU $05+MOVEBLKCMD ;Source incs - destination incs

* Less common commands

MOVESINCDDEC EQU $09+MOVEBLKCMD ;Source incs - destination decs
MOVESDECDINC EQU $06+MOVEBLKCMD ;Source decs - destination incs
MOVESDECDDEC EQU $0A+MOVEBLKCMD ;Source decs - destination decs

MOVESCONDCON EQU $00+MOVEBLKCMD ;Source constant - dest constant
MOVESINCDCON EQU $01+MOVEBLKCMD ;Source incs - destination const
```

```
MOVESDECDCON EQU $02+MOVEBLKCMD ;Source decs - destination const
MOVESCONDINC EQU $04+MOVEBLKCMD ;Source constant - dest constant
MOVESCONDDEC EQU $08+MOVEBLKCMD ;Source constant - destination decs

PARMRANGEERR EQU $FFFF ;Command error code
```

Subject: Re: Apple IIgs firmware equates
Posted by admin on Sat, 23 Aug 2014 12:03:52 GMT
View Forum Message <> Reply to Message

This is awesome Antoine. At some point I was planning on putting this type of information on the site in a searchable engine, but the forums are a great place to share this stuff too.
