P2 BOOT PROCESS

After power-up or hardware reset, the P2 loads 16KB of P2 Boot Code from an internal serial ROM into the top 16KB of HUB RAM ($FC000-$FFFFF) ?? or maybe slightly less ??
COG #0 then executes this “booter program code”. This code has been published, and contains the following code sections:

- Boot code (ROM_Booter.spin2):

This code sets RC FAST ??? clock mode and then tests for external pullup and/or pulldown resistors on Pins P59, P60 & P61 (see table). This code uses the resistor settings to determine which section of code will execute next.

- Serial code with autobaud:

This section waits for the two character sequence “> “ on the serial input pin P63, and utilised this sequence to determine the current baud, and sets the SmartPins for Asynchronous Serial mode (P63 input & P62 output). Timeouts to try FLASH or SD ???

- FLASH Boot Code:

This code uses P61=#CS, P60=CLK, P59=DI(out), P58=DO(in).
Note: these pins are shared with the SD Card if present.
?? Description of checksum/load/run sequence ??

- SD Card Boot Code:

This code uses P61=CLK, P60=#CS, P59=DI(out), P58=DO(in).
Note: these pins are shared with the FLASH if present.
The conditions for SD booting are described in more detail below.

- Monitor/Debug Code:

This code provides a Monitor and a number of support routines for debugging COG/LUT/HUB memory.
This code can be called in a number of ways…

  - From the Boot code with the sequence ?? “ >Ctl-D”
  - From TAQOZ with “Ctl-D” ?? Does this still work ??
  - From the user’s program. These routines execute in HUBEXEC mode using COG register space $1E0-$1EF for parameters and variables. The Monitor and its’ routines can be called as subroutines from the user’s program.

The Monitor/Debug operation is described in more detail below.

- TAQOZ:

This code provides a standalone Forth test environment for the P2.
This code can be called in a number of ways…

  - From the Boot code with the sequence
  - From the Monitor/Debugger prompt by the 2 character sequence “Ctl-D<cr>”.

TAQOZ is described in more detail below.

SD Boot Code
Provided a pull-up is sensed on P61=#CS (provided by the SD Card itself) and the other conditions are met, then the Booter code will pass control to the SD Boot Code. The SD code performs the following steps…

- Initialise the SD Card (in SPI mode). This is often referred to as “Mounting” the card.
- If successful, read the MBR Sector $0000_0000 (512 bytes) into HUB RAM starting at HUB $00000.
- Check if the 4 signature bytes at MBR offset $17C (now in HUB) contain “Prop” ($706F7250). If valid, then the MBR sector now residing at HUB $00000-$001FF (128 longs) is copied into COG RAM $000 and then a JMP $020 is executed. This has the effect of COG #0 executing the code loaded from the MBR starting at byte offset $080. Note the whole 512 byte (128 longs) sector is loaded into COG,

```
so while the code starts at $080 (COG $020), this code can use code below this
address if that code is valid and executable. Note that certain parts of the MBR
are required by FAT32, etc. Any P2 code written into the MBR must maintain any
required fields that the SD card format requires.
•       If the MBR does not contain “Prop” at $17C, then this location is checked
for the signature “ProP” ($506F7250). If this is found, then $174 (4 bytes) is
used as the raw SD Sector to start loading from, and $178 (4 bytes) is used as
the length (in bytes). The program then reads code from the card beginning at the
start sector for the length into HUB RAM beginning at HUB $00000. Upon completion
of the load, 496 longs from HUB $00000 are copied into COG $0 and a JMP #$000 is
executed to run the loaded code in COG.
•       If the MBR does not contain either signature, then the MBR is validated
as follows:
o       $1BE[16] is the Ptn0 Table
o       $1BE+0[1] AND $7F must equal $00
o       $1BE+4[1] must be $0C or $0B
o       $1BE+8[4] is used as the Vol_Begin sector
o       $1BE+C[4] is used at the Ptn Size
o       $1FE[2] must be $55AA
If these checks are met, then continue, else go back to Serial boot code.
•       The VOL sector is now read into HUB $00000, and it is checked for the
signature “Prop” or “ProP” at offset $17C. If either is found, then the operation
will continue the same as described above for the signature found in the MBR
sector.
•       If the VOL does not contain either signature, then the VOL is validated
as follows:
o       $00B[2] must be 512
o       $00D[1] is used at the clustershift
o       $00E[1] used to calculate FAT begin
o       $010[1] must be 2 (number of FAT tables)
o       $020[4] used to calculate PTN size (not checked)
o       $024[4] used to calculate DIR begin
o       $030[2] used to calculate FSI begin
o       $1FE[2] must be $55AA
If these checks are met, then continue, else go back to Serial boot code.
•       The FSI sector is read and validated as follows:
o       $000[4] must be $RRaA
o       $1E4[4] must be $rrAa
o       $1FE[2] must be $55AA
If these checks are met, then continue, else go back to Serial boot code.
•       The FAT DIRectory is now searched for the file “BOOT_P2.BIX”. If it is
found, it is read into HUB $00000 for a maximum length of (512-16)*1024 which is
512-16KB such that it will not overwrite the top 16KB of HUB RAM. Then the first
496 longs will be copied into COG $000 and a JMP #$000 will execute.
WARNING: This file must be contiguous on the SD Card! This is not checked!!
•       If the above file is not found, then the FAT DIRectory is searched for
the file “BOOT_P2.BIY”. If it is found, then it will be read and executed in the
same process as the previous step.
•       If neither file is found, the code returns to the Serial boot code.
```

Reference: http://forums.parallax.com/discussion/170637/p2-sd-boot-code-rev-2-silicon
Cluso99

Why the option to boot from the MBR or VOL sectors?

If the card is not formatted with FAT32 (eg exFAT) then the boot code is unable to search the DIRectory to find a file. This is limited because of ROM space and the exFAT license from Microsoft). To overcome this limitation, the MBR is used to store either
* P2 boot code (max 512 bytes = 128 longs)
* Raw SD sector start and Length in bytes (trimmed max 496KB).

The first option loads this sector (512 bytes = 128 longs) into COG at $000 and jumps to COG $020 (ie offset byte $080 in the sector data). Formatting on SD requires the use of some of this space on MBR/VOL. The middle section seems to be available on most (all?) unformatted/formatted SD cards these days. This is the best option for future proofing the P2 ROM SD boot code. Anything on this sector that is free can be used - you are only limited to the code start being at byte $080, and the signature "Prop" at bytes $17C-$17F. The code stored here could be a tiny booter that uses the SD ROM routines in HUB to load more code at fixed locations, or to support exFAT and locate a file. It's use is entirely up to you.

This second option uses the signature "Prop" at bytes $17C-$17F, plus a sector start pointer at bytes $174-$177 and a length at bytes $178-$17B. The max length is trimmed to 496KB in order to not overwrite the top 16KB in HUB that currently stores the ROM Boot Code, and which is actually doing this loading. The sector(s) used must be contiguous as no checks are done. This sector+length may be a pointer to a file residing on the SD, or it could be in the reserved sectors, or even in its' own partition. It's all up to you. Again, this provides the most flexibility into the future.

FAT32 Files
If the SD card is formatted with FAT32, then the boot code looks for the files "BOOT_P2.BIX" and if not found, then "BOOT_P2.BIY". Note the uppercase. If either of these are found, then they will be read into HUB starting at $00000, for the length of the file which is trimmed to a maximum of 496KB to prevent overwriting the HUB ROM code currently executing. Note the file sectors must be contiguous as no checks are done.

SD Cards
SD, SDHC and SDXC cards are supported in the boot code. BUT you should use the later SDHC or SDXC cards that use sector addressing as sectors which permits cards up to 1TB. The older SD and SDHC cards used sector addressing in bytes which limited their maximum to 2GB or 4GB? (seem to think they were limited to 31 bits). They haven't been produced in many years.
Note: I will be releasing an SD Driver shortly for use in a separate COG and it will not support these older SD cards. I should have removed it from the ROM to save some valuable space. Ain't hindsight wonderful :)

Using the SD ROM Routines
Firstly, not there is no Write routine as it was removed due to limited ROM space.
These routines may be called to Initialise (mount) the card, read the CSD/CID, read a sector(s), search the DIRectory for a filename, and optionally read that file into your designated HUB location. These routines require the use of COG $1C0-$1DF for variables.

WARNING: There are code differences between Rev 1 silicon and Rev 2 silicon !!!

SD Boot Code - Callable Routines (Rev2 silicon)

```
   DAT

 ''#####################################################################################
 ###########
                      ''##    SD Card - HUBEXEC code...
 ##

 ''#####################################################################################
 ###########
 fc560                              orgh

                      '+-------[ SD: Initialise/Locate/Load/Run a file from
 SD ]--------------------+ <--- SD: init/locate/load/run a file --->
                      '+ On Entry:
 +
                      '+      fname[3]:       filename[11] 8+3 without "."  (will be $0
 terminated)  +
                      '+ Call Format:
 +
                     '+              CALL    #@_Run_SDfile                   '
+ < call: init/locate/load/run a file >
                      '+ On Return:
 +
                      '+      "NZ" if error, else does not return
 +

 '+--------------------------------------------------------------------------------+



 '+--------------------------------------------------------------------------------+
 fc560     f6079a00 _Start_SDcard   mov     skiprun,         #0              ' load/run
 MBR/VOL code                        \ --tweek2--
fc564     fdb00050                  call    #@_SDcard_Init0                 '
Init/CSD/CID/MBR/VOL/FSI/FAT (skiprun=0)       | --tweek2--
fc568     adb00290          if_e    call    #@_readDIR                      ' read
directory for filenames
                      ''              mov     skiprun,         #0              ' load/run
 <file>              (already 0)
fc56c     adb00378          if_e    call    #@_readFILE                     '
read/load/run the file
fc570     fdb004ec                  call    #@_releaseDO                    ' /CS=1, send
CLKs, tristate CS/CK/DI/DO        | --tweek2--
 fc574     fd800100                 JMP     #try_serial                     ' failed: so
 go back and try serial

 '+--------------------------------------------------------------------------------+
fc578     fdb00020 _Run_SDfile     call    #@_SDcard_Init1                 '
Init/CSD/CID/MBR/VOL/FSI/FAT (skiprun=1)       | --tweek2--
fc57c     adb002ac          if_e    call    #@_searchDIR                    ' search dir
for <fname>
 fc580     f6079a00                 mov     skiprun,         #0              ' load/run
 <file>              (skiprun=0)
 fc584              _Loadit                                                 '
 \ {{ tweek }}
```

```
fc584     adb00360        if_e    call    #@_readFILE                       ' read/load/
[run] the file                     / {{ tweek }}
fc588     fdb004d4                call    #@_releaseDO                       ' /CS=1, send
CLKs, tristate CS/CK/DI/DO        | --tweek2--
 fc58c     fd64002d                RET                                       ' return
 "NZ" = failed, "Z" if loaded ok

 '+--------------------------------------------------------------------------+
fc590     fdb00008 _Load_SDfile    call    #@_SDcard_Init1                   '
Init/CSD/CID/MBR/VOL/FSI/FAT (skiprun=1)      | --tweek2--
fc594     adb00294        if_e    call    #@_searchDIR                       ' search dir
for <fname>
                      ''              mov     skiprun,         #1             ' load, do
 not run <file>       (already 1)
fc598     fd9fffe8                jmp     #@_Loadit                         ' if Z go
read/load file                       ] {{ tweek }}

 '+--------------------------------------------------------------------------+
```

SD Boot Code - SD Routines (HUBEXEC) Rev 1 silicon

```
'' +--------------------------------------------------------------------------+
 ''   P2 ROM SD ROUTINES (HUBEXEC)  Rev 1 silicon
 '' +--------------------------------------------------------------------------+
   _Start_SDcard     = $fc560     ' initialise & read CSD/CID
   _Run_SDfile       = $fc578     ' initialise & read CSD/CID
   _Load_SDfile      = $fc590     ' initialise & read CSD/CID
   _SDcard_Init      = $fc5a4     ' init hub data ptr=$0
   _readMBR1         = $fc6dc     ' do not load/run MBR/VOL code
   _readMBR          = $fc6e0     ' VBR/MBR = SECTOR 0
   _readDIR          = $fc810     ' copy _fname1 -> fname
   _searchDIR        = $fc840     ' 12th char must be $00
   _readFILE         = $fc900     ' DAT SECTOR#
   _readnxtSECTOR    = $fc944     ' next sector#
   _readnxtSLOT      = $fc948     ' next data slot
   _readSECTOR       = $fc950     ' sector#
   _readREG          = $fc974     ' CMD9,10: CSD,CID register
   _readBLOCK        = $fc97c     ' CMD17: PAR=sector, 512 bytes
 '' +--------------------------------------------------------------------------+
 '' HUB ADDRESSES
 '' +--------------------------------------------------------------------------+
   _HUBROM        = $FC000        ' ROM $FC000
   _HUBBUF        = $FC000        ' overwrite Booter
   _HUBBUFSIZE    = 80            ' RxString default size for _HUBBUF
 '' +--------------------------------------------------------------------------+
```

SD Boot Code - SD Routines (HUBEXEC) Rev 2 silicon

```
 '' +--------------------------------------------------------------------------+
 ''   P2 ROM SD ROUTINES (HUBEXEC)  Rev 2 silicon
 '' +--------------------------------------------------------------------------+
   _Start_SDcard     = $fc560     ' initialise & read CSD/CID
   _Run_SDfile       = $fc578     ' initialise & read CSD/CID
```

```
  _Load_SDfile     = $fc590     ' initialise & read CSD/CID
  _SDcard_Init     = $fc59c     ' init hub data ptr=$0
  _searchDIR       = $fc82c     ' 12th char must be $00
  _readFILE        = $fc8e8     ' DAT SECTOR#
  _readnxtSECTOR   = $fc930     ' next sector#
  _readnxtSLOT     = $fc934     ' next data slot
  _readSECTOR      = $fc93c     ' sector#
  _readREG         = $fc960     ' CMD9,10: CSD,CID register
  _readBLOCK       = $fc968     ' CMD17: PAR=sector, 512 bytes
  _releaseDO       = $fca60     ' sends $FF (8 clocks) to SD card with /CS=1, tristates
P58-61
'' +--------------------------------------------------------------------------+
'' HUB ADDRESSES
'' +--------------------------------------------------------------------------+
  _HUBROM       = $FC000        ' ROM $FC000
  _HUBBUF       = $FC000        ' overwrite Booter
  _HUBBUFSIZE   = 80            ' RxString default size for _HUBBUF
'' +--------------------------------------------------------------------------+
```

SD Boot Code - COG Variables

```
''============[ COG VARIABLES - SD BOOT]=======================================
                 org     $1C0              ' place the variables in cog $1C0-$1DF

cmdout          res     1                       ' The 8b CMDxx | $40
cmdpar          res     1                       ' The 32b parameters
cmdcrc          res     1                       ' The 8b CRC (must be valid for CMD0 &
CMD8)
cmdpar2         res     1                       ' SDV1=$0, SDV2=$40000000
cmdtype         res     1                       ' reply is R1=1, R3=3, R7=7, else 0
reply           res     1                       ' R1 reply (moved to replyR1 when R3/R7
32b reply here)
replyR1         res     1                       ' R1 reply (8b saved when R3/R7 32b
reply follows)
dataout         res     1                       ' 8/32 bit data being shifted out
bytescnt        res     1                       ' #bytes to send/recv
bitscnt         res     1                       ' #bits to be shifted in/out
ctr1            res     1
timeout         res     1                       ' = starttime + delay
_AA55           res     1                       ' used to store $AA55 to validate
MBR/VOL/FSI
skiprun         res     1                       ' 1= skip load/run mbr/vol & load/no-
run fname
                                                      '\ 1=SDV1, 2=SDV2(byte address),
3=SDHC/SDV2(block address)
blocksh         res     1                       '/ block shift 0/9 bits
clustersh       res     1                       ' sectors/cluster SHL 'n' bits

vol_begin       res     1 '$0000_2000           ' Ptn0: first sector of PTN
fsi_begin       res     1 '$0000_2001           ' Ptn0:       sector of file system
info
fat_begin       res     1 '$0000_3122           ' Ptn0: first sector of FAT table
dir_begin       res     1 '$0000_4000           ' Ptn0: first sector of DATA is DIR
table
dat_begin       res     1 '$0000_4580           ' Ptn0: first sector of file's DATA
ptn_size        res     1 '$0008_0000           '       file-size 32KB = 64<<9 sectors

_bufad          res     1
_blocknr        res     1
```

```
_sectors        res     1
_entries        res     1
bufad           res     1                       ' ptr sector buffer
blocknr         res     1                       ' sector#
fname           res     3                       ' 8+3+1
_hubdata        res     1
                  fit     $1E0
```

-OR-

```
''===========[ COG VARIABLES $1CO-$1DF - SD BOOT]===========================
  cmdout        = $1c0          ' The 8b CMDxx | $40
  cmdpar        = $1c1          ' The 32b parameters
  cmdcrc        = $1c2          ' The 8b CRC (must be valid for CMD0 & CMD8)
  cmdpar2       = $1c3          ' SDV1=$0, SDV2=$40000000
  cmdtype       = $1c4          ' reply is R1=1, R3=3, R7=7, else 0
  reply         = $1c5          ' R1 reply (moved to replyR1 when R3/R7 32b reply here)
  replyR1       = $1c6          ' R1 reply (8b saved when R3/R7 32b reply follows)
  dataout       = $1c7          ' 8/32 bit data being shifted out
  bytescnt      = $1c8          ' #bytes to send/recv
  bitscnt       = $1c9          ' #bits to be shifted in/out
  ctr1          = $1ca
  timeout       = $1cb          ' = starttime + delay
  spare         = $1cc
  skiprun       = $1cd          ' 1= skip load/run mbr/vol & load/no-run fname
                                  '\ 1=SDV1, 2=SDV2(byte address), 3=SDHC/SDV2(block
address)
  blocksh       = $1ce          '/ block shift 0/9 bits
  clustersh     = $1cf          ' sectors/cluster SHL 'n' bits

  vol_begin     = $1d0          ' Ptn0: first sector of PTN
  fsi_begin     = $1d1          ' Ptn0:       sector of file system info
  fat_begin     = $1d2          ' Ptn0: first sector of FAT table
  dir_begin     = $1d3          ' Ptn0: first sector of DATA is DIR table
  dat_begin     = $1d4          ' Ptn0: first sector of file's DATA
  ptn_size      = $1d5          '       file-size 32KB = 64<<9 sectors

  _bufad        = $1d6
  _blocknr      = $1d7
  _sectors      = $1d8
  _entries      = $1d9
  bufad         = $1da          ' ptr sector buffer
  blocknr       = $1db          ' sector#
  fname         = $1dc'[3]      ' 8+3+1
  _hubdata      = $1df
'' +-----------------------------------------------------------------------+
```

TAQOZ in RAM

To supplement Cluso's documentation in regards to raw sector boot code, this is what how the full RAM version of TAQOZ is saved by default, in sector 1 onwards. In this case only $F000 bytes are needed. On a factory formatted SD card there are hidden sectors before the start of the partition, and in the case of the 8GB card there is about 4MB hidden, so plenty of room for storing your code etc.

```
TAQOZ# $170 $10 SD DUMPW ---
00170: 0000 0000 0001 0000  F000 0000 7250 506F     '............ProP' ok
```

TAQOZ in RAM includes a proper SD formatter and reporting tool that can format cards >32GB to FAT32. Here is the report for an 8GB SanDisk.

```
TAQOZ# .DISK ---  CARD: SANDISK   SD SL08G REV$80 #1561170528 DATE:2016/2


                     *** OCR ***
    VALUE........................... $C0FF_8000
    RANGE........................... 2.7V to 3.6V

                     *** CSD ***
    CARD TYPE....................... SDHC
    LATENCY......................... 1ms+1400 clocks
    SPEED........................... 50Mbps
    CLASSES......................... 010110110101
    BLKLEN.......................... 512
    SIZE............................ 7,761MB
    Iread Vmin...................... 100ma
    Iread Vmax...................... 25ma
    Iwrite Vmin..................... 1ma
    Iwrite Vmax..................... 45ma

                   *** SPEEDS ***
    SECTOR.......................... 478us,624us,555us,552us,552us,554us,588us,559us,
   BLOCKS.......................... 2,015kB/s @192MHz

                     *** MBR ***
    PARTITION....................... 0 00 INACTIVE
    FILE SYSTEM..................... FAT32 LBA
    CHS START....................... 1023,254,63
    CHS END......................... 0,0,0
    FIRST SECTOR.................... $0000_2000
    TOTAL SECTORS................... 15,515,648 = 7,944MB

00170: 0000_0000 0000_0001 0000_F000 506F_7250     '............ProP'

                    *** FAT32 ***
    OEM............................. TAQOZ P2
    Byte/Sect....................... 512
    Sect/Clust...................... 64 = 32kB
    FATs............................ 2
    Media........................... F8
```

```
     Sect/Track...................... $003F
     Heads........................... $00FF
     Hidden Sectors.................. 8,192 = 4MB
     Sect/Part....................... 15,515,648 = 7,944MB
     Sect/FAT........................ 1,894 = 969kB
     Flags........................... 0
     Ver............................. 00 00
     ROOT Cluster.................... $0000_0002 SECTOR: $0000_2EEC
     INFO Sector..................... $0001 = $0000_2001
     Backup Sector................... $0006 = $0000_2006
     res............................. 00 00 00 00 00 00 00 00 00 00 00 00
     Drive#.......................... 128
     Ext sig......................... $29 OK!
     Part Serial#.................... $50AD_0021 #1353515041
     Volume Name..................... P2 CARD    FAT32    ok
```