So You Want to Write an FST...

Application

↓

GS/OS

↓

FST

↓

Device Driver

History

System 1 (1986)
Mouse Desk

System 3 (1987)
PRODOS 16

System 4 (1988)
PRO.FST (1987)
CHAR.FST (1987)
HS.FST (1988)

System 5 (1989)
+AppleShare.FST (1987)

System 6 (1992)
+DOS 3.3 FST (1988)
+HFS.FST (1990)
+Pascal.FST (1989)

System 6.01 (1993)
+MS-DOS 3.3 FST (1990)

History

System 1 (1986)
Mouse Desk

System 3 (1987)
PRODOS 16

System 4 (1988)
PRO.FST (1987)
CHAR.FST (1987)
HS.FST (1988)
+AppleShare.FST (1987)

System 5 (1989)
+DOS 3.3 FST (1988)
+Pascal.FST (1989)

System 6 (1992)
+HFS.FST (1990)
+MS-DOS 3.3 FST (1990)

System 6.01 (1993)

Resources

Files of type $BD contains file system translators, or FSTs. FSTs do not load if bit 15 of their auxiliary type is set.

GS/OS calls FSTs to interpret the physical file systems stored on block devices. By asking translation software to read the file system, GS/OS can read virtually any file system while having only an abstract file system assumed in the operating system code. Not all released file system translators are required, saving space on disk and in memory.

The format for FSTs is Apple confidential and subject to change with every system software release; Apple will release all future FSTs for GS/OS. Third-party developers may not create GS/OS FSTs -- **no documentation is available, and disassembly of the code for this purpose is not permitted**. This is not an easy decision for Apple, which is a company that was built upon and operates with the goal to empower individuals through computing. Not revealing information isn't exactly consistent with this goal. There are, however, reasons for this policy.

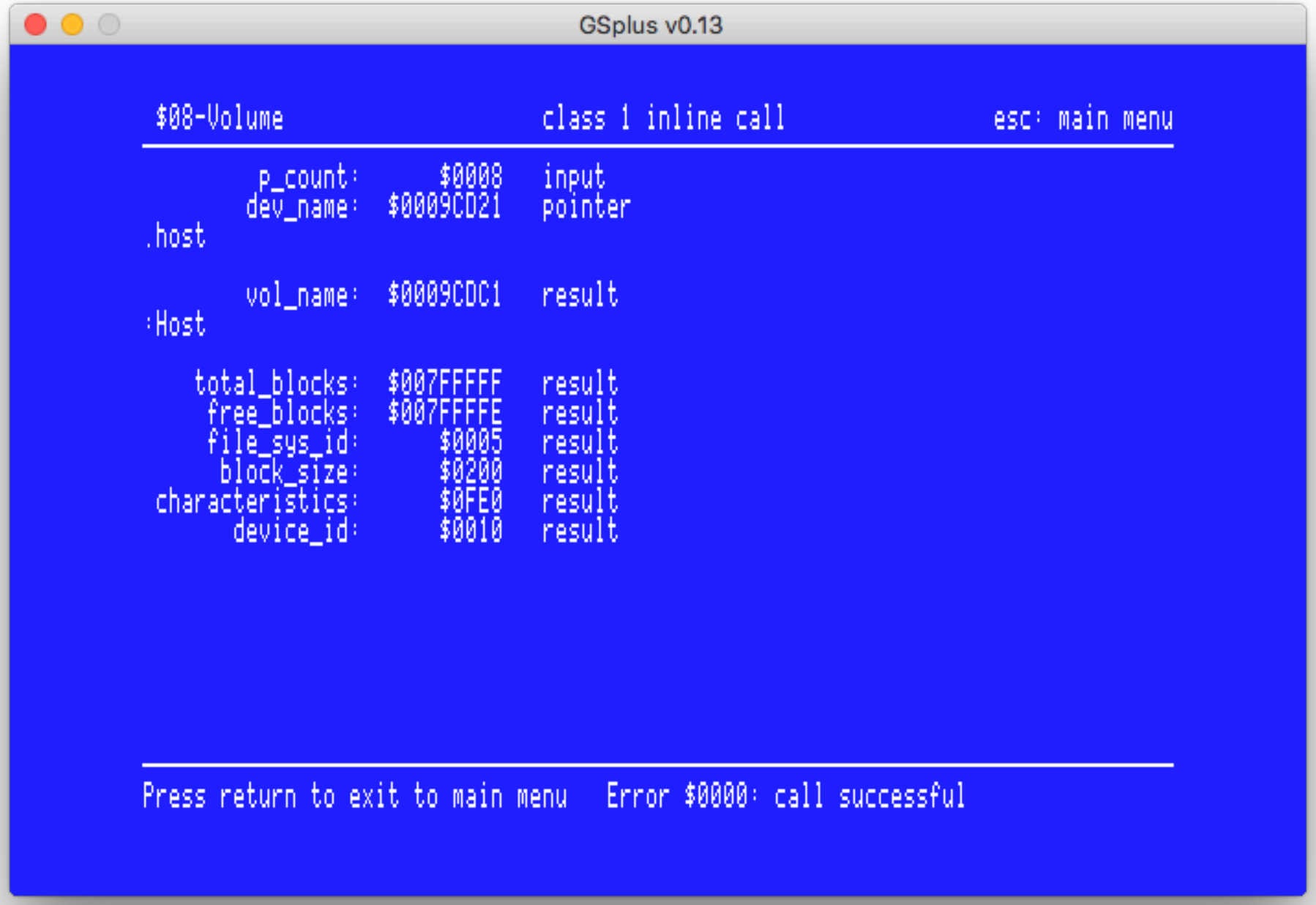
First, FSTs are not as modular as they could be. Some GS/OS level changes require changes to all of the FSTs to be implemented. These changes range in magnitude from internal system service call changes to adding new parameters to existing calls. GS/OS is not tolerant of FSTs that do not know about such changes. The FST structure is straightforward, but it is also complex enough that disassembly of existing FSTs does not cover all the bases.

Resources

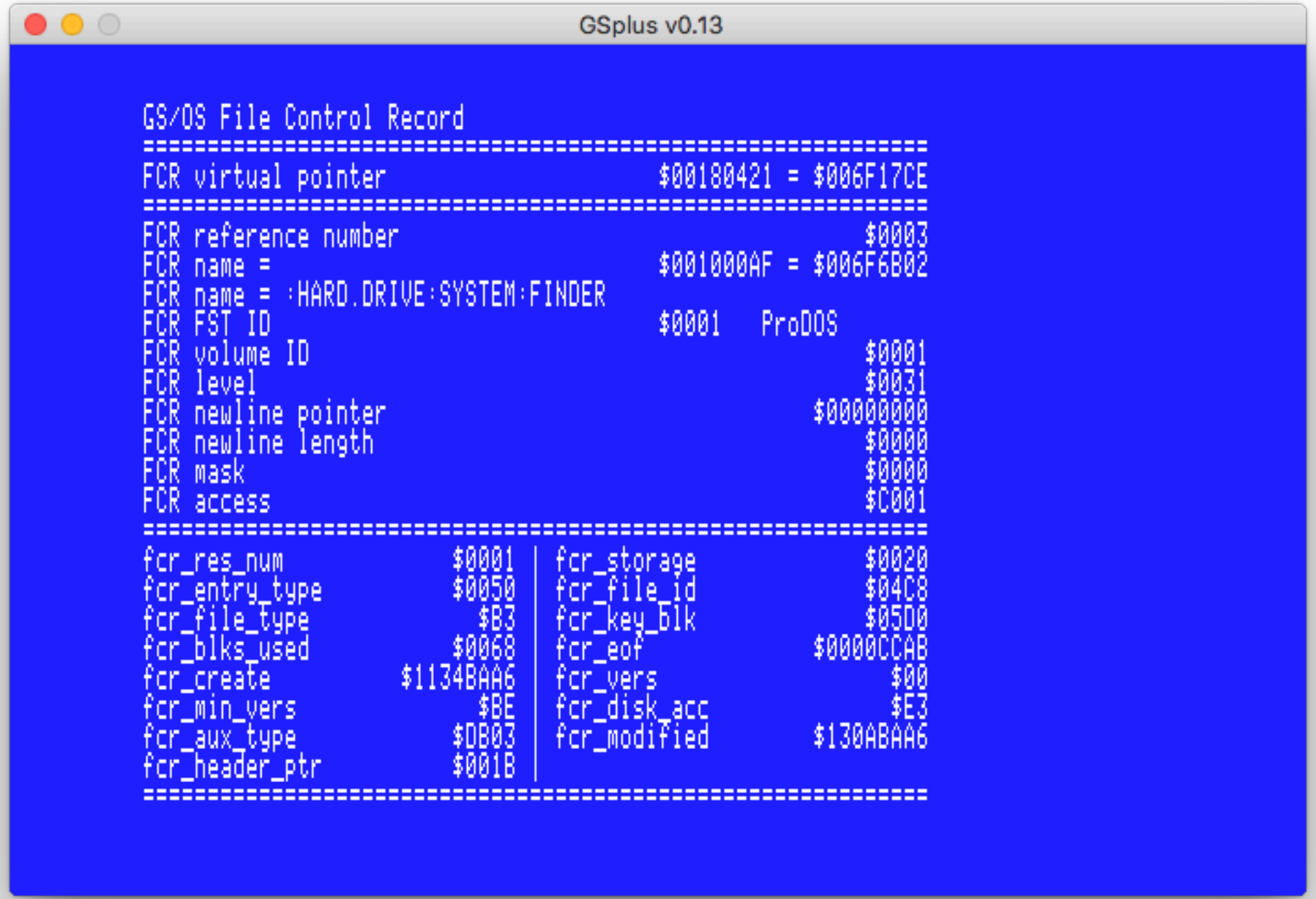
- **Apple IIgs GS/OS Reference**
- **Apple IIgs GS/OS Driver Reference**
- **Apple IIgs GS/OS Internals (Brutal Deluxe)**
- **System 6 source code**
- **GS/OS Internal Snooper CDA**
- **Exerciser DA**
- **https://github.com/ksherlock/minix.fst**
- **https://github.com/ksherlock/host-fst**

GSplus v0.13

```
GS/OS System Call Exerciser v6.0d8  09-Dec-91
Copyright 1984, 87-91  Apple Computer Inc.  All Rights Reserved

$01-Create          $10-Open            $1E-EndSession      $2D-DStatus
$02-Destroy         $11-NewLine         $1F-SessionStatus   $2E-DControl
$03-OSShutdown      $12-Read            $20-GetDevNumber    $2F-DRead
$04-ChangePath      $13-Write           $21-GET_LAST_DEV    $30-DWrite
$05-SetFileInfo     $14-Close           $22-READ_BLOCK      $31-BindInt
$06-GetFileInfo     $15-Flush           $23-WRITE_BLOCK     $32-UnbindInt
$07-JudgeName       $16-SetMark         $24-Format          $33-FSTSpecific
$08-Volume          $17-GetMark         $25-EraseDisk       $34-AddNotifyProc
$09-SetPrefix       $18-SetEOF          $27-GetName         $35-DelNotifyProc
$0A-GetPrefix       $19-GetEOF          $28-GetBootVol      $36-DRename
$0B-ClearBackup     $1A-SetLevel        $29-Quit            $37-GetStdRefNum
$0C-SetSysPrefs     $1B-GetLevel        $2A-GetVersion      $38-GetRefNum
$0E-ExpandPath      $1C-GetDirEntry     $2B-GetFSTInfo      $39-GetRefInfo
$0F-GetSysPrefs     $1D-BeginSession    $2C-DInfo           $3A-SetStdRefNum

J - Make inline calls to GS/OS          K - Make class 1 calls to GS/OS
L - Catalog a directory                 M - Modify the contents of memory
N - Catalog $00 levels of a directory   P - Now using maximum p_count for calls
Q - Quit back to caller                 R - Visit the Monitor

Select command: $08
```


GSplus v0.13
$08-Volume class 1 inline call esc: main menu
p_count: $0008 input
dev_name: $0009CD21 pointer
.host
vol_name: $0009CDC1 result
:Host
total_blocks: $007FFFFF result
free_blocks: $007FFFFE result
file_sys_id: $0005 result
block_size: $0200 result
characteristics: $0FE0 result
device_id: $0010 result
Press return to exit to main menu Error $0000: call successful


GSplus v0.13
GS/OS Internal Snooper Version 6.0d2
Copyright (c) 1987-1992
Apple Computer, Inc.
All rights reserved
Written by: Ray Montagne, ext. 4-6255
Updated by: Greg Branche, ext. 4-5523
THIS CDA DISPLAYS INTERNAL OPERATING SYSTEM DATA STRUCTURES AND MUST NOT BE DISTRIBUTED OUTSIDE OF APPLE COMPUTER, INC. UNDER ANY CIRCUMSTANCES
Quick Keys: (? returns to this page)
1 = System Calls
2 = System Svc
3 = Error Codes
4 = User ID
5 = Direct Page
6 = DIB Dumper
7 = SIB Dumper
8 = VCR Dumper
9 = FCR Dumper
A = Cache Dumper
B = Traps
C = FST Numbers
D = FST Headers
E = Block Diagram
F = $E1 Vectors
G = ExpressLoad
H = File List
I = VBL Queue
J = Death Trap
K = Driver Calls
L = Device Char.
M = Sup. Char.
N = Mem. Attr.
O = Memory Map
P = GOTO Monitor
Q = User ID
R = SCSI Dir. Pg
S = SCSI I/O Rec.
T = SCSI Dev. Rec.
U = SCSI Internals
V = Device ID List
W = Device Status
X = ASCII Chart
Y =
Z =
ESC = Quit


GSplus v0.13
GS/OS File Control Record
FCR virtual pointer $00180421 = $006F17CE
FCR reference number $0003
FCR name = $001000AF = $006F6B02
FCR name = :HARD.DRIVE:SYSTEM:FINDER
FCR FST ID $0001 ProDOS
FCR volume ID $0001
FCR level $0031
FCR newline pointer $00000000
FCR newline length $0000
FCR mask $0000
FCR access $C001
fcr_res_num $0001
fcr_entry_type $0050
fcr_file_type $B3
fcr_blks_used $0068
fcr_create $1134BAA6
fcr_min_vers $BE
fcr_aux_type $DB03
fcr_header_ptr $001B
fcr_storage $0020
fcr_file_id $04C8
fcr_key_blk $05D0
fcr_eof $0000CCAB
fcr_vers $00
fcr_disk_acc $E3
fcr_modified $130ABAA6

FST Header

```
header  proc
        str     'FST '
        dc.l    app_entry
        dc.l    sys_entry
        dc.w    fst_id

        dc.w    fst_attr    ; attributes
        dc.w    $0100       ; version
        dc.w    $0200       ; block size
        dc.l    $007FFFFF   ; maximum volume size
        dc.l    1           ; min volume size
        dc.l    $FFFFFFFF   ; max file size
        dc.l    0           ; reserved
        str.b   'Host'      ; name
        str.b   'Host FST                v01.00' ; comment
        dc.w    0           ; reserved
        ; credits
        str.b   'Host FST written by Kelvin W Sherlock.'
        endp
```

System Entry

```
entry:
x = call * 2
1: sys_startup
2: sys_shutdown
3: remove_vcr
4: deferred_flush

exit:
rtl
carry = error
a = error number
```

SYS_STARTUP

- **Called while booting**
- **Only ROM tools available**
- **Return Carry flag to prevent FST from loading**
- **Carry + A to indicate fatal error**
- **“Sorry, system error $xxxx occurred while loading the FST file xxx”**

SYS_SHUTDOWN

- **Called when GS/OS shuts down**

REMOVE_VCR

- Called from `release_vcr`

DEFERRED_FLUSH

- Called from `EndSessionGS`
- Call `cache_flsh_def`

Application Entry

```
entry:
x = call * 2
y = class * 2 (0 = P16, 2 = GS/OS)

exit:
jml sys_exit
carry = error
a = error number
```

GS/OS Calls

- CreateGS
- DestroyGS
- OSShutdownGS
- ChangePathGS
- SetFileInfoGS
- GetFileInfoGS
- JudgeNameGS
- VolumeGS
- SetPrefixGS
- GetPrefixGS
- ClearBackupGS
- SetSysPrefsGS
- ExpandPathGS
- GetSysPrefsGS
- OpenGS
- NewLineGS
- ReadGS
- WriteGS
- CloseGS
- FlushGS
- SetMarkGS
- GetMarkGS
- SetEOFGS
- GetEOFGS
- SetLevelGS
- GetLevelGS
- GetDirEntryGS
- BeginSessionGS
- EndSessionGS
- SessionStatusGS
- GetDevNumGS
- GET_LAST_DEV
- READ_BLOCK
- WRITE_BLOCK
- FormatGS
- EraseDiskGS
- GetNameGS
- GetBootVolGS
- QuitGS
- GetVersionGS
- GetFSTInfoGS
- DInfoGS
- DStatusGS
- DControlGS
- DReadGS
- DWriteGS
- BindIntGS
- UnbindIntGS
- FSTSpecificGS
- AddNotifyProcGS
- DelNotifyProcGS
- DRenameGS
- GetStdRefNumGS
- GetRefNumGS
- GetRefInfoGS
- SetStdRefNumGS

FST Calls

- **CreateGS**
- **DestroyGS**
- OSShutdownGS
- **ChangePathGS**
- **SetFileInfoGS**
- **GetFileInfoGS**
- **JudgeNameGS**
- **VolumeGS**
- SetPrefixGS
- GetPrefixGS
- **ClearBackupGS**
- SetSysPrefsGS
- ExpandPathGS
- GetSysPrefsGS
- **OpenGS**
- NewLineGS
- **ReadGS**
- **WriteGS**
- **CloseGS**
- **FlushGS**
- **SetMarkGS**
- **GetMarkGS**
- **SetEOFGS**
- **GetEOFGS**
- SetLevelGS
- GetLevelGS
- **GetDirEntryGS**
- BeginSessionGS
- EndSessionGS
- SessionStatusGS
- **GetDevNumGS**
- GET_LAST_DEV
- READ_BLOCK
- WRITE_BLOCK
- **FormatGS**
- **EraseDiskGS**
- GetNameGS
- GetBootVolGS
- QuitGS
- GetVersionGS
- GetFSTInfoGS
- DInfoGS
- DStatusGS
- DControlGS
- DReadGS
- DWriteGS
- BindIntGS
- UnbindIntGS
- **FSTSpecificGS**
- AddNotifyProcGS
- DelNotifyProcGS
- DRenameGS
- GetStdRefNumGS
- GetRefNumGS
- GetRefInfoGS
- SetStdRefNumGS

Communication

- **Parameter Block**
- **Direct Page**
- **GS/OS Service Calls**

Parameter Blocks

```
struct FileInfoRecGS {
    Word pCount;
    GSString255Ptr pathname;
    Word access;
    Word fileType;
    LongWord auxType;
    Word storageType;
    TimeRec createDateTime;
    TimeRec modDateTime;
    ResultBuf255Ptr optionList;
    LongWord eof;
    LongWord blocksUsed;
    LongWord resourceEOF;
    LongWord resourceBlocks;
};
```

Parameter Blocks

```
struct FileRec {
    Ptr pathname;
    Word fAccess;
    Word fileType;
    Longint auxType;
    Word storageType;
    Word createDate;
    Word createTime;
    Word modDate;
    Word modTime;
    Longint blocksUsed;
};
```

ProDOS 16 vs GS/OS

Parameter Count
String Format
Date Format
Path Delimiter

Direct Page

```
$00:

dev_num                 word
dev_callnum             word
dev_dev_id
dev_buff                long
dev_req_cnt             long
dev_xfer_cnt            long
dev_blk_num             long
…
```

Direct Page

```
$30:

call_number      word
param_blk_ptr    long
dev1_num         word
dev2_num         word
path1_ptr
fcr_ptr          long
path2_ptr
vcr_ptr          long
path_flag        word
span1            word
span2            word
```

Direct Page

`$80-$d3` available for FST use

GS/OS Service Call Vectors

```
dev_dispatcher  equ     $01FC00

alloc_vcr       equ     $01FC24
release_vcr     equ     $01FC28
find_vcr        equ     $01FC48
rename_vcr      equ     $01FC58
get_vcr         equ     $01FC60

alloc_fcr       equ     $01FC2C
release_fcr     equ     $01FC30
find_fcr        equ     $01FC4C
rename_fcr      equ     $01FC5C
get_fcr         equ     $01FC64

deref           equ     $01FC38
get_sys_gbuf    equ     $01FC3C
```

FST Calls

- **CreateGS**
- **DestroyGS**
- OSShutdownGS
- **ChangePathGS**
- **SetFileInfoGS**
- **GetFileInfoGS**
- **JudgeNameGS**
- **VolumeGS**
- SetPrefixGS
- GetPrefixGS
- **ClearBackupGS**
- SetSysPrefsGS
- ExpandPathGS
- GetSysPrefsGS
- **OpenGS**
- NewLineGS
- **ReadGS**
- **WriteGS**
- **CloseGS**
- **FlushGS**
- **SetMarkGS**
- **GetMarkGS**
- **SetEOFGS**
- **GetEOFGS**
- SetLevelGS
- GetLevelGS
- **GetDirEntryGS**
- BeginSessionGS
- EndSessionGS
- SessionStatusGS
- **GetDevNumGS**
- GET_LAST_DEV
- READ_BLOCK
- WRITE_BLOCK
- **FormatGS**
- **EraseDiskGS**
- GetNameGS
- GetBootVolGS
- QuitGS
- GetVersionGS
- GetFSTInfoGS
- DInfoGS
- DStatusGS
- DControlGS
- DReadGS
- DWriteGS
- BindIntGS
- UnbindIntGS
- **FSTSpecificGS**
- AddNotifyProcGS
- DelNotifyProcGS
- DRenameGS
- GetStdRefNumGS
- GetRefNumGS
- GetRefInfoGS
- SetStdRefNumGS

Path-Based FST Calls

- **CreateGS**
- **DestroyGS**
- OSShutdownGS
- **ChangePathGS**
- **SetFileInfoGS**
- **GetFileInfoGS**
- **JudgeNameGS**
- **VolumeGS**
- SetPrefixGS
- GetPrefixGS
- **ClearBackupGS**
- SetSysPrefsGS
- ExpandPathGS
- GetSysPrefsGS
- **OpenGS**
- NewLineGS
- ReadGS
- WriteGS
- CloseGS
- FlushGS
- SetMarkGS
- GetMarkGS
- SetEOFGS
- GetEOFGS
- SetLevelGS
- GetLevelGS
- GetDirEntryGS
- BeginSessionGS
- EndSessionGS
- SessionStatusGS
- **GetDevNumGS**
- GET_LAST_DEV
- READ_BLOCK
- WRITE_BLOCK
- **FormatGS**
- **EraseDiskGS**
- GetNameGS
- GetBootVolGS
- QuitGS
- GetVersionGS
- GetFSTInfoGS
- DInfoGS
- DStatusGS
- DControlGS
- DReadGS
- DWriteGS
- BindIntGS
- UnbindIntGS
- FSTSpecificGS
- AddNotifyProcGS
- DelNotifyProcGS
- DRenameGS
- GetStdRefNumGS
- GetRefNumGS
- GetRefInfoGS
- SetStdRefNumGS

Pathnames...

```
"/absolute/path"
"relative:path"
"31:path"
".dev1:path"
```

Canonical Format

- **Convert ProDOS 16 → GS/OS strings**
- **Expand Prefixes**
- **Convert / → :**
- **Device name → device number**
- **Uppercase (FST flag)**
- **Strip high bits (FST flag)**
- **NULL-terminate**
- **Calculate longest pathname component**

FST Call strategy

If volume name provided, find VCR, give FST first shot

Call each FST

`call_number`	`word`
`param_blk_ptr`	`long`
`dev1_num`	`word`
`dev2_num`	`word`
`path1_ptr`	`long`
`path2_ptr`	`long`
`path_flag`	`word`
`span1`	`word`
`span2`	`word`

Ref-Num FST Calls

- CreateGS
- DestroyGS
- OSShutdownGS
- ChangePathGS
- SetFileInfoGS
- GetFileInfoGS
- JudgeNameGS
- VolumeGS
- SetPrefixGS
- GetPrefixGS
- ClearBackupGS
- SetSysPrefsGS
- ExpandPathGS
- GetSysPrefsGS
- OpenGS
- NewLineGS
- **ReadGS**
- **WriteGS**
- **CloseGS**
- **FlushGS**
- **SetMarkGS**
- **GetMarkGS**
- **SetEOFGS**
- **GetEOFGS**
- SetLevelGS
- GetLevelGS
- **GetDirEntryGS**
- BeginSessionGS
- EndSessionGS
- SessionStatusGS
- GetDevNumGS
- GET_LAST_DEV
- READ_BLOCK
- WRITE_BLOCK
- FormatGS
- EraseDiskGS
- GetNameGS
- GetBootVolGS
- QuitGS
- GetVersionGS
- GetFSTInfoGS
- DInfoGS
- DStatusGS
- DControlGS
- DReadGS
- DWriteGS
- BindIntGS
- UnbindIntGS
- FSTSpecificGS
- AddNotifyProcGS
- DelNotifyProcGS
- DRenameGS
- GetStdRefNumGS
- GetRefNumGS
- GetRefInfoGS
- SetStdRefNumGS

Ref-Num

Look up FCR from refNum
Call the FST

```
call_number      word
param_blk_ptr    long
dev1_num         word
fcr_ptr          long
vcr_ptr          long
```

File Control Record

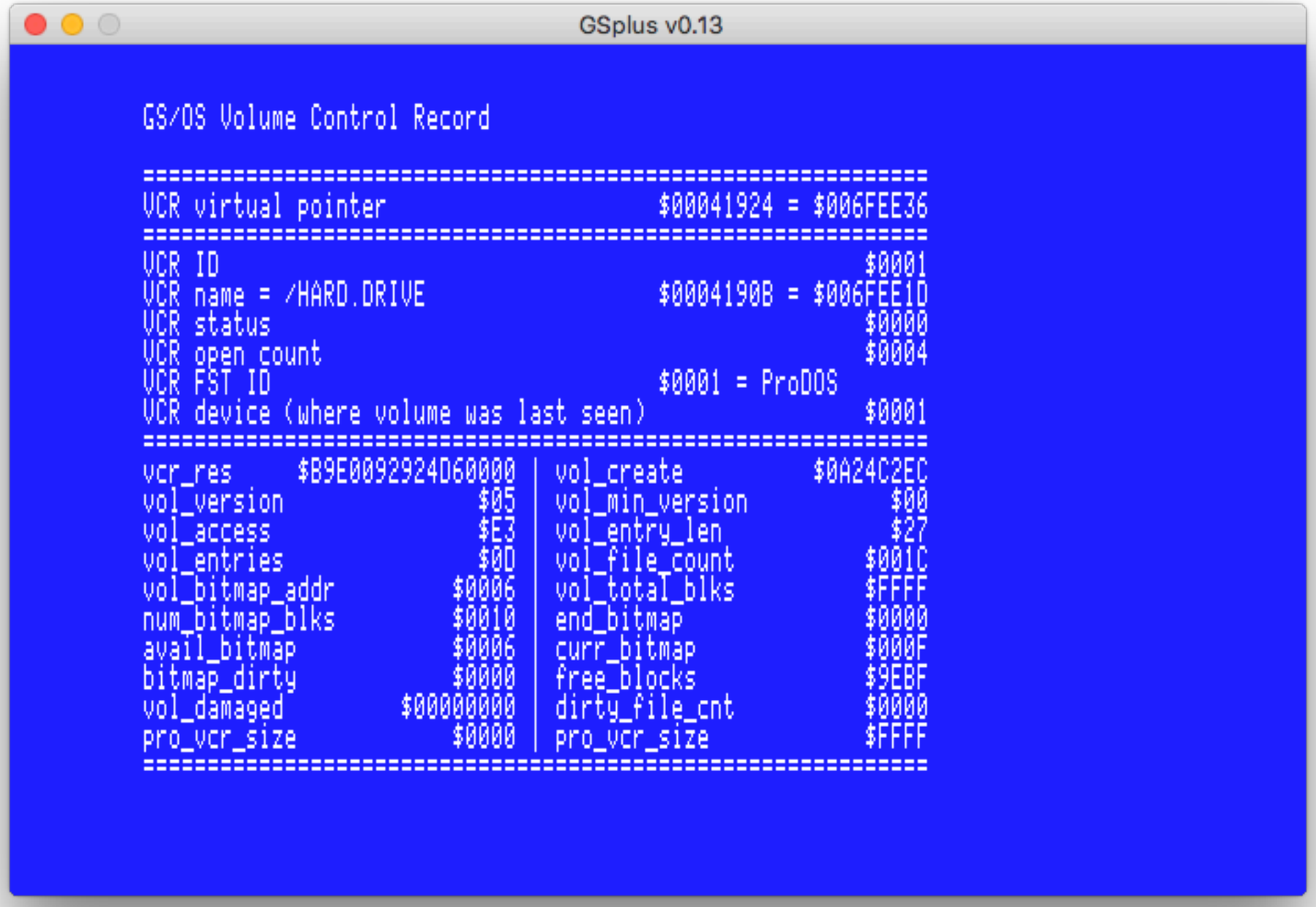
```
id                  word
path_name           long
fst_id              word
vcr_id              word
level               word
newline             long
newline_length      word
newline_mask        word
access              word
fst specific data…
```

GSplus v0.13

```
GS/OS File Control Record
============================================================
FCR virtual pointer                    $00180A80 = $006F1E2D
============================================================
FCR reference number                                   $0005
FCR name =                             $00100007 = $006F6B2A
FCR name = :HARD.DRIVE:SYSTEM:FINDER
FCR FST ID                             $0001   ProDOS
FCR volume ID                                          $0001
FCR level                                              $0070
FCR newline pointer                                $00000000
FCR newline length                                     $0000
FCR mask                                               $0000
FCR access                                             $8001
============================================================
fcr_res_num             $0000 | fcr_storage             $0030
fcr_entry_type          $0050 | fcr_file_id             $04C8
fcr_file_type             $B3 | fcr_key_blk             $05AF
fcr_blks_used           $0106 | fcr_eof             $00023DEC
fcr_create          $1134BAA6 | fcr_vers                  $00
fcr_min_vers              $BE | fcr_disk_acc              $E3
fcr_aux_type            $DB03 | fcr_modified        $130ABAA6
fcr_header_ptr          $001B |
============================================================
```

Volume Control Record

```
id                  word
name                long
status              word
open_count          word
fst_id              word
device              word
fst specific data…
```


GSplus v0.13
GS/OS Volume Control Record
VCR virtual pointer $00041924 = $006FEE36
VCR ID $0001
VCR name = /HARD.DRIVE $0004190B = $006FEE1D
VCR status $0000
VCR open count $0004
VCR FST ID $0001 = ProDOS
VCR device (where volume was last seen) $0001
vcr_res $B9E009292 4D60000
vol_create $0A24C2EC
vol_version $05
vol_min_version $00
vol_access $E3
vol_entry_len $27
vol_entries $0D
vol_file_count $001C
vol_bitmap_addr $0006
vol_total_blks $FFFF
num_bitmap_blks $0010
end_bitmap $0000
avail_bitmap $0006
curr_bitmap $000F
bitmap_dirty $0000
free_blocks $9EBF
vol_damaged $00000000
dirty_file_cnt $0000
pro_vcr_size $0000
pro_vcr_size $FFFF

Memory Management

- `get_sys_gbuf`
  - `$9a00-$9dff`
- `alloc_seg, alloc_vcr, alloc_fcr`
  - returns 32-bit virtual pointer
- `deref`
  - virtual pointer -> IIgs pointer
- `release_seg, release_vcr, release_fcr`
- Can also use Memory Manager

```
sys_entry       proc

                phk
                plb
                long    m,x

                cpx     #max_sys_call+1
                bge     rtl_no_error

                jmp     (@sys_table,x)

@sys_table
                dc.w    rtl_no_error
                dc.w    sys_startup
                dc.w    sys_shutdown
                dc.w    rtl_no_error    ; remove vcr
                dc.w    rtl_no_error    ; deferred flush
max_sys_call    equ     *-@sys_table-2

debugstr        str.b   'sys_entry'
                endp
```

```
sys_startup     proc

                stz dev_id

                ; sanity check that the global buffer location
                ; is where I expect it.
                jsl     get_sys_gbuf
                cpy     #$0000
                bne     no
                cpx     #$9a00
                bne     no

                ; check if host wdm active.
                lda     #0
                sec
                ldx     #$8001
                call_host
                ; wdm will clear carry if active.
                lda     #0
                rtl

no
                sec     ; unload me!
                lda #0
                rtl
                endp
```

```
app_entry       proc

                with dp
                with fst_parms

                phk
                plb
                long    m,x

                ; x = call number * 2
                ; y = call class * 2
                sty <call_class
                stx <tmp

                ; check the class 0 or 1 only.
                cpy     #2+1
                bge     @bad_system_call

                cpx     #max_app_call+1 ; 66+1
                bge     @bad_system_call

                ; class 1 -- check the pcount maximum.
                cpy     #2
                bne     @ok

                lda     table,x
                and     #$00ff
                cmp     [param_blk_ptr]
                ; gs/os already checks the minimum and verifies
                ; non-null names, etc.
                bcc     @invalid_pcount
```

```
@ok
                stz     my_fcr
                stz     my_fcr+2
                stz     my_vcr
                stz     my_vcr+2
                stz     cookie

@check_fcr
                ; check fcr bit, deref fcr and vcr, if provided

@check_path
                ; check path bit
                ; if path provided, verify volume (or device)
                ; is :HOST
```

```
@call
                ldx     <tmp

                ; fake an rtl address for sys_exit
                ; otherwise, would need to jml sys_exit from functions.
                pea     |(sys_exit-1)>>8
                phb
                lda     #<sys_exit-1
                sta     1,s

                ; call it...
                ldx     <tmp
                jmp     (app_table,x)
```

```
open            proc

                with    dp, fst_parms

                jsr     build_vcr
                bcs     exit

                lda     #invalid_fst_op
                ldx     call_number
                sec
                call_host
                bcs     exit
                stx     cookie
                sty     tmp     ; actual read/write access

                ; build the fcr.
                lda     #fcr.__sizeof
                ldx     #colon_Host
                ldy     #^colon_Host
                jsl     alloc_fcr
                bcs     close

                jsl     deref
                stx     my_fcr
                sty     my_fcr+2
```

```
                ldy     #vcr.open_count
                lda     [my_vcr],y
                inc     a
                sta     [my_vcr],y
                lda     tmp
                ldy     #fcr.access
                sta     [my_fcr],y
                lda     #fst_id
                ldy     #fcr.fst_id
                sta     [my_fcr],y
                lda     cookie
                ldy     #fcr.cookie
                sta     [my_fcr],y
                ldy     #vcr.id
                lda     [my_vcr],y
                ldy     #fcr.vcr_id
                sta     [my_fcr],y

                ; store the refnum for output.
                ; conveniently, call_class is the offset.
                lda     [my_fcr]
                ldy     call_class
                sta     [param_blk_ptr],y

exit_ok
                lda     #0
                clc
exit
                rtl
```

```
build_vcr       proc
                with dp

                ldx #host_name
                ldy #^host_name
                lda #0
                jsl find_vcr
                bcs create_vcr
                jsl deref

                stx my_vcr
                sty my_vcr+2
                ldy #vcr.fst_id
                lda [my_vcr],y
                cmp #fst_id
                bne dump_vcr

                ldy #vcr.status
                lda [my_vcr],y
                and #vcr_swapped
                beq @exit

                and #vcr_swapped_in
                sta [my_vcr],y

                ;lda device
                lda dev_id
                ldy #vcr.device
                sta [my_vcr],y
```

```
dump_vcr
                ; vcr exists for the filename but it's not mine.
                ; if inactive, kick it out.  otherwise, return dup error.

                ldy #vcr.open_count
                lda [my_vcr],y
                beq @dump

                lda #dup_volume
                sec
                rts

@dump
                ldy #vcr.id
                lda [my_vcr],y
                jsl release_vcr
                ; drop through.
```

```
create_vcr
                    lda #vcr.__sizeof
                    ldx #host_name
                    ldy #^host_name
                    jsl alloc_vcr
                    lda #out_of_mem
                    bcs exit

                    jsl deref
                    stx my_vcr
                    sty my_vcr+2

                    lda #0
                    ldy #vcr.status
                    sta [my_vcr],y
                    ldy #vcr.open_count
                    sta [my_vcr],y

                    lda #fst_id
                    ldy #vcr.fst_id
                    sta [my_vcr],y

                    lda dev_id
                    ldy #vcr.device
                    sta [my_vcr],y

                    lda #0
                    clc
exit
                    rts
```

```
; insight - vcr and fcr always go together.
vcr_used        equ     $8000
fcr_used        equ     $0000
path_used       equ     $4000

table           ; stores max pcount + 1
                dc.w    0
                dc.w    8+path_used                     ; ($01) Create
                dc.w    2+path_used                     ; ($02) Destroy
                dc.w    0                               ; ($03) OS Shutdown
                dc.w    4+path_used                     ; ($04) Change Path
                dc.w    13+path_used                    ; ($05) Set File Info
                dc.w    13+path_used                    ; ($06) Get File Info
                dc.w    7                               ; ($07) Judge Name
                dc.w    7                               ; ($08) Volume
                dc.w    0                               ; ($09) Set Prefix
                dc.w    0                               ; ($0A) Get Prefix
                dc.w    2+path_used                     ; ($0B) Clear Backup Bit
                dc.w    0                               ; ($0C) Set Sys Prefs
                dc.w    0                               ; ($0D) Null
                dc.w    0                               ; ($0E) Expand Path
                dc.w    0                               ; ($0F) Get Sys Prefs
                dc.w    16+path_used                    ; ($10) Open
                dc.w    0                               ; ($11) NewLine
                dc.w    6+vcr_used+fcr_used             ; ($12) Read
                dc.w    6+vcr_used+fcr_used             ; ($13) Write
                dc.w    2+vcr_used+fcr_used             ; ($14) Close
                dc.w    3+vcr_used+fcr_used             ; ($15) Flush
                dc.w    4+vcr_used+fcr_used             ; ($16) Set Mark
               ...
```

```
get_file_info   procname export

        with fst_parms
        with dev_parms
        with dp

        jsr path_to_inode
        bcs exit

        jsr load_inode
        bcs exit

        lda <call_class
        beq class0

class1
        lda [param_blk_ptr] ; pcount
        dec a
        asl a ; x 2
        asl a ; x 4
        tax
        dispatch file_info_dcb_1
        lda tool_error
        cmp #1
        rtl

class0
        ldx #file_info_dcb_0_size-4
        dispatch file_info_dcb_0
        lda tool_error
        cmp #1
        rtl
```

```
file_info_dcb_0
        with FileRec
        dc.w pathname, do_ignore ; pathname
        dc.w fAccess, do_access
        dc.w fileType, do_file_type
        dc.w auxType, do_aux_type
        dc.w storageType, do_storage_type
        dc.w createDate, do_create_date_time_0
        dc.w modDate, do_mod_date_time_0
        dc.w blocksUsed, do_blocks
        endwith
file_info_dcb_0_size equ *-file_info_dcb_0

file_info_dcb_1
        with FileInfoRecGS
        ;dc.w $00, do_ignore    ; pCount
        dc.w pathname, do_ignore ; pathname
        dc.w access, do_access
        dc.w fileType, do_file_type
        dc.w auxType, do_aux_type
        dc.w storageType, do_storage_type
        dc.w createDateTime, do_create_date_time
        dc.w modDateTime, do_mod_date_time
        dc.w optionList, do_option_list
        dc.w eof, do_eof
        dc.w blocksUsed, do_blocks
        dc.w resourceEOF, do_r_eof
        dc.w resourceBlocks, do_r_blocks
        endwith
        endp
```

```
        MACRO
&lab    dispatch &table

&lab
@loop
        ldy &table,x
        jsr (&table+2,x)
        dex
        dex
        dex
        dex
        bpl @loop

        MEND
```

```
do_blocks proc export
        with dp,fst_parms
        with v1

        ; minix supports sparse blocks.  Just guess based on size...

        ; size + 1023

        lda disk_inode.size
        clc
        adc #1023
        sta tmp

        lda disk_inode.size+2
        adc #0
        sta tmp+2

        ; divided by 1024
        ...

        lda tmp
        sta [param_blk_ptr],y
        iny
        iny
        lda tmp+2
        sta [param_blk_ptr],y

        rts
        endp
```

Bootable FSTs...

Dispatch Table

$2004	**read file**
$2006	**get boot name**
$2008	**get fst name**
$200a	**size of boot code**
$200c	**start.gs.os aux type**

Read File

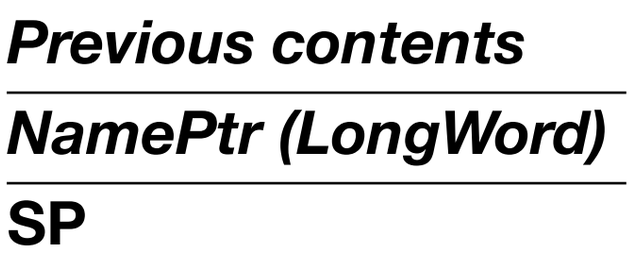
- **Given pathname and address**
- **reads file into memory at address**
- **returns file size, file type, aux type**

Previous contents
Space (8 bytes)
PathPtr (LongWord)
BufferPtr (LongWord)
SP

Previous contents
EOF (LongWord
Aux Type (Word)
File Type (Word)
SP

Get Boot Name

- **Returns the name of the boot volume**



Get FST Name

- **Returns the name of the FST**

Previous contents

NamePtr (LongWord)

SP

Previous contents

SP

Booting

Boot Loader

- **Set up Dispatch Table at $2004**
- **Load START.GS.OS at $6800**
- **JMP $6800**
- **LDA #bootflags / JMP $6803**

START.GS.OS

- **Aux type is last boot time (boot thermometer)**
- **Uses dispatch table to read other files**
- **(GS.OS, ERROR.MSG, GS.OS.DEV, Boot FST, BOOT.DRIVER)**
- **Switches over to Boot FST for I/O.**