Flash Enable
BIOS Reverse Engineering.

Luc Verhaegen <libv@skynet.be>

FOSDEM2010 Coreboot DevRoom.

2010-02-06

What to expect.

Legal?

New Board?

- Brand and model name of motherboard.
- Output of :
  - lspci -nnvvvxxx
  - superiotool -deV
  - flashrom -V
- (link to) Image of BIOS.

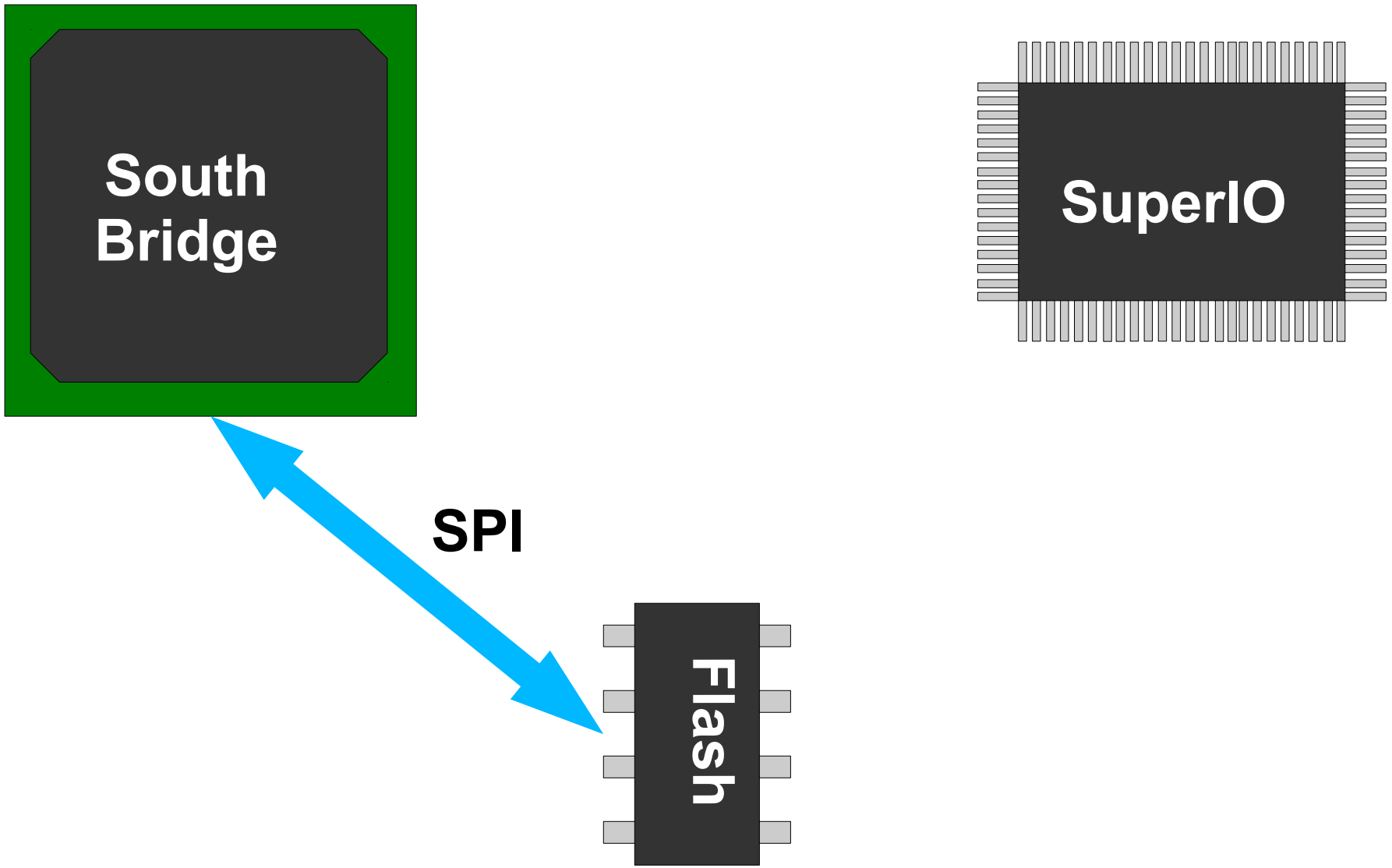
Tools

- A hex editor or hexdump.
- An x86 disassembler.
- flashrom source code.
- bios_extract:
  http://cgit.freedesktop.org/~libv/bios_extract
- A bit of a clue ;)

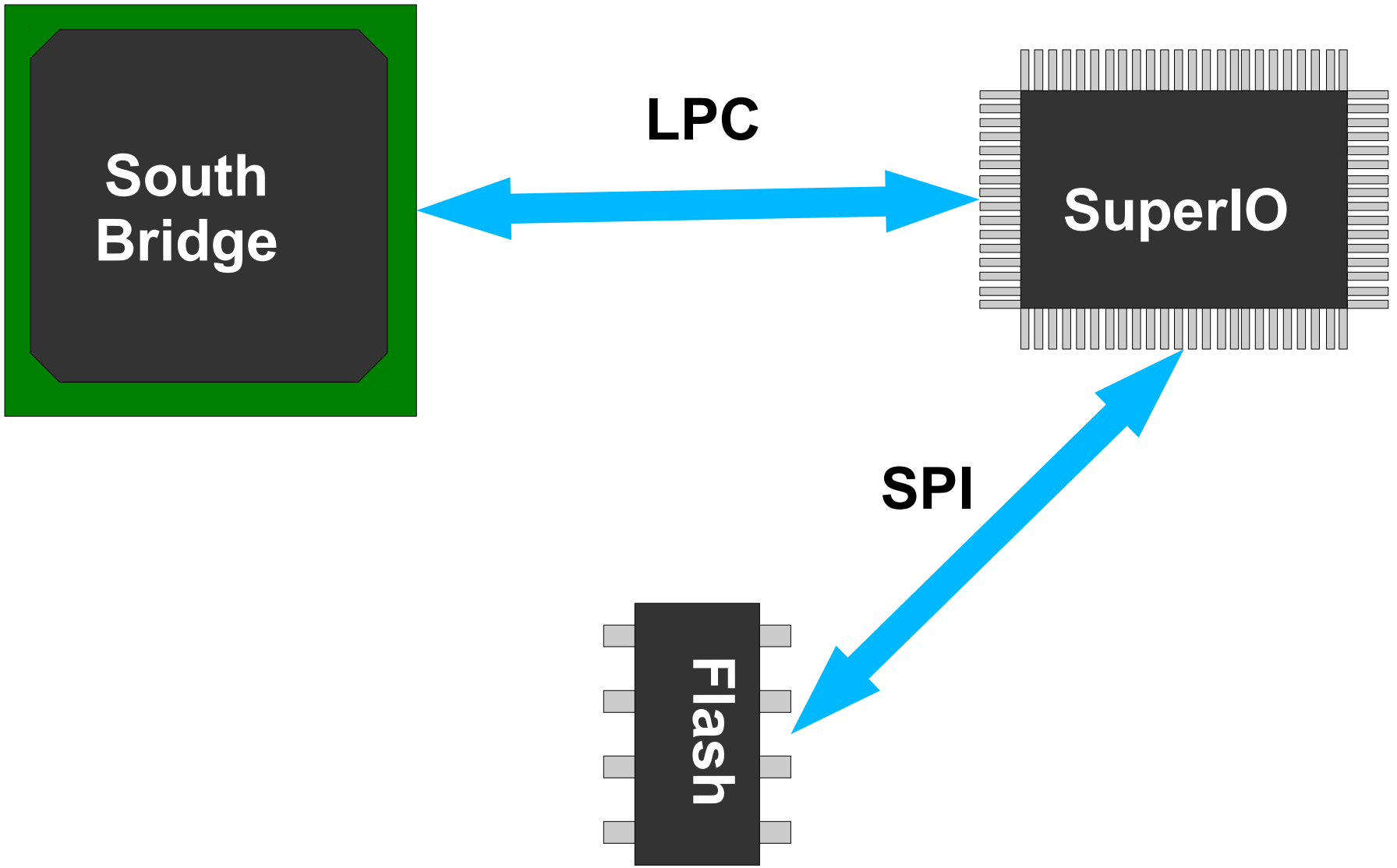
For our example

- grep, head.
- hexdump
- ndisasm (part of nasm)
- from http://people.freedesktop.org/~libv/
  - award_crafted.bin.bz2
  - flash_enable_bios_reverse_engineering.odp

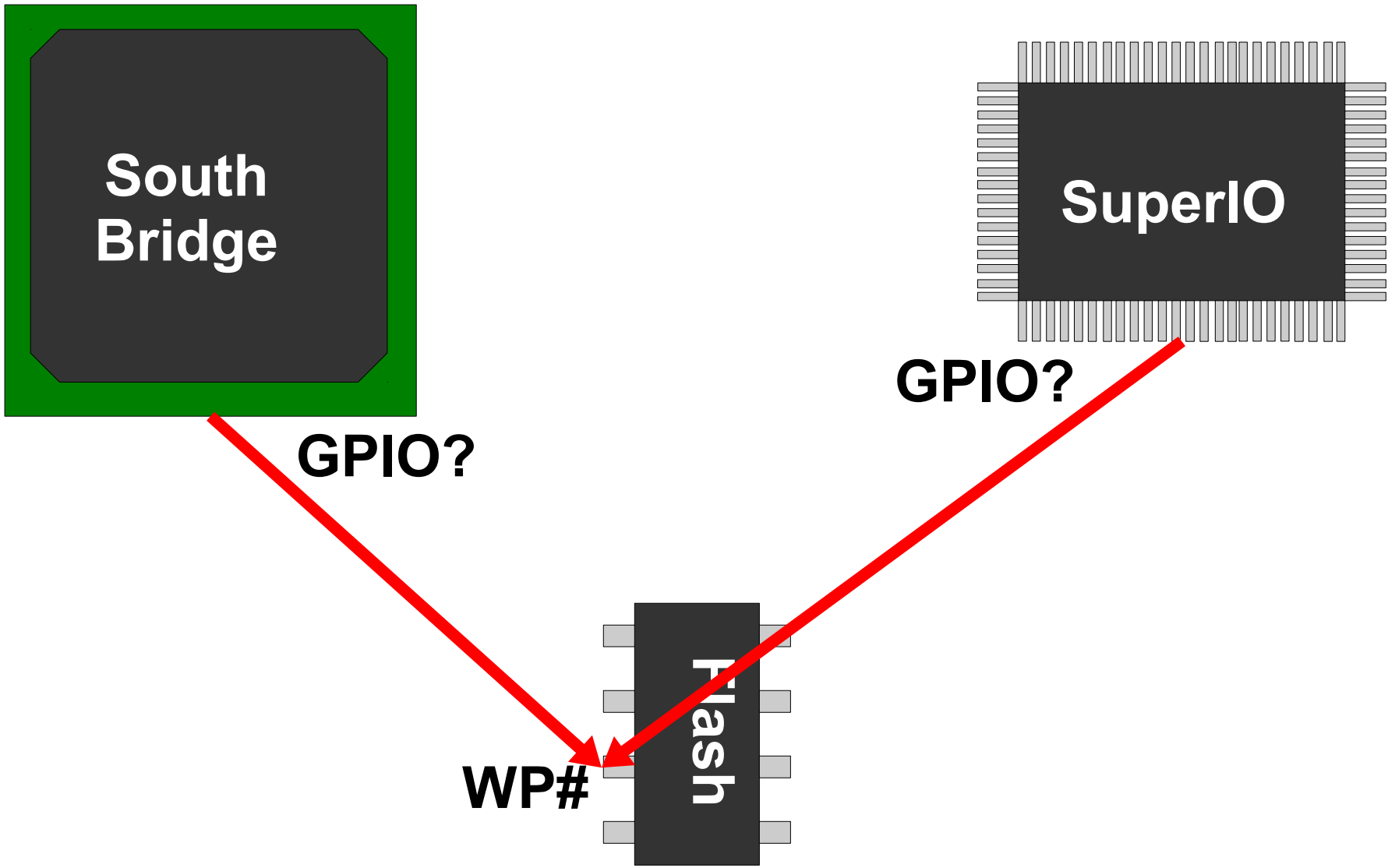
Common layout



Common layout



Write Protection



Generic Flash enable structure.

1) Chipset specific part.
2) SuperIO specific part.
3) GPIO line setting.

Award

- Keeps Flash Enable in F-Segment.
- Uses easy to spot structure (AWDFLASH) with function pointers (second is Flash Enable).
- Quick and easy.

Award Procedure

- bios_extract the vendor/complete image.

```
Found Award BIOS.
0x10000 ( 75384 bytes)    ->    BLOB.BIN          (131072 bytes)
0x2267A ( 39312 bytes)    ->    awardext.rom      ( 59040 bytes)
0x2C00B (  8417 bytes)    ->    ACPITBL.BIN       ( 25578 bytes)
0x2E0ED ( 26794 bytes)    ->    awardeyt.rom      ( 48256 bytes)
...
```

- hexdump -C BLOB.BIN | grep AWDFLASH

```
00014560  24 40 41 57 44 46 4c 41  53 48 8c 45 8f 45 9f 45  | $@AWDFLASH.E.E.E|
```

- Disassemble at 0x1458F.

ASUS

- Same as Award, just different string/structure.

ASUS Procedure

- bios_extract the vendor/complete image.

```
Found Award BIOS.
...
0x094B7 ( 20018 bytes)    ->    pci32.rom         ( 32768 bytes)
0x20000 ( 79091 bytes)    ->    blob.bin          (131072 bytes)
0x34000 (  4334 bytes)    ->    asus.bmp          ( 18062 bytes)
...
```

- hexdump -C blob.bin | grep -A1 ASUS_FLASH

```
00016b50  41 53 55 53 5f 46 4c 41  53 48 20 20 20 20 20 20  |ASUS_FLASH      |
00016b60  01 02 6e 6b 00 f0 29 00  7d 6b 00 f0 1a 00 52 51  |..nk..).}k....RQ|
```

- Disassemble at 0x16B7D.

AMI

- Flash Enable in F-Segment.
- Uses int16h AX=0xE007.
- int16h entrypoint is at 0xF000:0xE82E
- Not so quick and easy.

AMI Procedure

- bios_extract the vendor/complete image.

```
...
0x718D0 (  1436 bytes) -> amibody_19.rom        (  4868 bytes)  "ADM Font"
0x4719C (173853 bytes) -> amibody_1b.rom        (352724 bytes)  "SLAB"
0x44110 ( 12405 bytes) -> amilang_US.rom        ( 27734 bytes)  "Multilanguage"
...
```

- ami_slab amibody_1b.rom

```
Name              Tp LoadAddr     size initialized
RUN_CSEG          00 000f0000    65536   yes
POST_CSEG         01 00040000    49994   yes
...
```

- Disassemble at 0xE82E.

AMI Tips

1) Jumps... Jumps... Jumps...
2) Routine which checks AH, which usually does not handle AH=0xE0...
3) But before checking AH, there might be a call...
4) To a routine which checks AH=0xE0 and which calls the AL-th index in a function pointer table...
5) Our flash enable is the 7th!

Phoenix (New)

- “Phlash” blob appended to vendor bios image.
- Board enable easy to spot: right after the “ZFLPF” table.
- Even easier than award!

Phoenix (old)

- PLATFORM.BIN
- hexdump -C PLATFORM.BIN | grep -A1 ZQFC

```
00000100  5a 51 46 43 1c 0d 00 00  3b 0d 00 00 00 00 00 00  |ZQFC....;.......|
00000110  00 00 00 00 d2 0e 00 00  00 00 00 00 00 00 00 00  |................|
```

- Disassemble at 0x0ED2

Let's Get Cracking!

Fictitious board.

- VIA VT8237R SouthBridge:

  ISA Bridge at 00:11.0.

  PM IO Base Address at 0x400.
- Winbond WB83697HF SuperIO:

  IO access at 0x2E/0x2F.
- Award BIOS.

  E+F Segments: award_crafted.bin
- Flashrom fails to erase.

Retrieve the hook.

```
> hexdump -C award_crafted.bin | grep AWDFLASH
000155f0  24 40 41 57 44 46 4c 41  53 48 18 56 2c 56 1a 56  |$@AWDFLASH.V,V.V|
>_
```

Using ndisasm

```
ndisasm -k 0,0x1562C blob.bin
```

Top level (ndisasm)

```
> ndisasm -k 0,0x1562c award_crafted.bin | head
00000000  skipping 0x1562C bytes
0001562C  6660                pushad
0001562E  E82514              call word 0x6a56
00015631  E8627C              call word 0xd296
00015634  E84E63              call word 0xb985
00015637  6661                popad
00015639  CB                  retf
0001563A  FF                  db 0xFF
0001563B  FF                  db 0xFF
0001563C  FF                  db 0xFF
>_
```

Top level (clean)

```
pushad
call    word 0x6a56
call    word 0xd296
call    word 0xb985
popad
retf
```

PCI Accesses.

- 0xCF8: Address

  Bus:Device.function , config register Config:

  0x8000000 | ((((Bus << 4) | Device) << 3) | Function) << 8) | (Config)
- 0xCFC: Data

First routine (0x6A56)

```
mov     cx,0x8840          → 0:11.0  0x40
call    word 0xf73a        → PCIByteRead
or      al,0x80
call    word 0xf760        → PCIByteWrite
mov     cx,0x8859
call    word 0xf75a
and     al,0x7f
call    word 0xf760        → 0:11.0  0x59
ret                        → PCIByteRead

                           → PCIByteWrite
```

→ ROM Write Enable

→ All memory cycles to LPC

→ Chipset Enable!

Top level

```
pushad
call    ChipsetEnable
call    word 0xd296
call    word 0xb985
popad
retf
```

Second routine (0xD296)

```
call    word 0xd286
mov     cl,0x24
call    word 0xd26a
or      al,0x8
call    word 0xd277
call    word 0xd28e
ret
```

First sub-routine (0xD286)

```
mov     dx,0x2e
mov     al,0x87
out     dx,al
out     dx,al
ret
```

→ Enter WinBond extended mode

Second routine (0xD296)

```
call    WBExtModeEnter
mov     cl,0x24
call    word 0xd26a
or      al,0x8
call    word 0xd277
call    word 0xd28e
ret
```

Second sub-routine (0xD26A)

```
mov     al,cl
mov     dx,0x2e
out     dx,al
out     0xeb,al
inc     dx
in      al,dx
out     0xeb,al
ret
```

→ SuperIO Read Byte

Second routine (0xD296)

```
call    WBExtModeEnter
mov     cl,0x24
call    SIOReadByte
or      al,0x8
call    word 0xd277
call    word 0xd28e
ret
```

Third sub-routine (0xD277)

```
push    ax
mov     dx,0x2e
mov     al,cl
out     dx,al
out     0xeb,al
pop     ax
inc     dx
out     dx,al
out     0xeb,al
ret
```

→ SuperIO Write Byte

Second routine (0xD296)

```
call    WBExtModeEnter
mov     cl,0x24
call    SIOReadByte
or      al,0x8
call    SIOWriteByte
call    word 0xd28e
ret
```

Fourth sub-routine (0xD28E)

```
mov     dx,0x2e
mov     al,0xaa
out     dx,al
ret
```

→ Exit WinBond extended mode

Second routine (revisited)

```
call    WBExtModeEnter
mov     cl,0x24
call    SIOReadByte
or      al,0x8
call    SIOWriteByte
call    WBExtModeExit
ret
```

→ W83697HF MEMW# Enable.

→ SuperIO Enable!

Top level

```
pushad
call    ChipsetEnable
call    SuperIOEnable
call    word 0xb985
popad
retf
```

Third routine (0xB985)

```
mov     dx,0x44c  → VT8237R PM IO + GPIO offset.
in      al,dx
or      al,0x40   → GPIO6.
out     dx,al
ret
```

→ Raise VT8237R GPIO6!

Top level

```
pushad
call    ChipsetEnable
call    SuperIOEnable
call    VT823xGPIO6Raise
popad
retf
```

→ Automatic

→ Automatic soon.

→ Board Enable

Board Enable.

```
static int
fictitious_board(const char *name)
{
        return via_vt823x_gpio_set(0x06, 1);
}
```

Questions?