Enclosure Management in FreeBSD

Alexander Motin
mav@FreeBSD.org

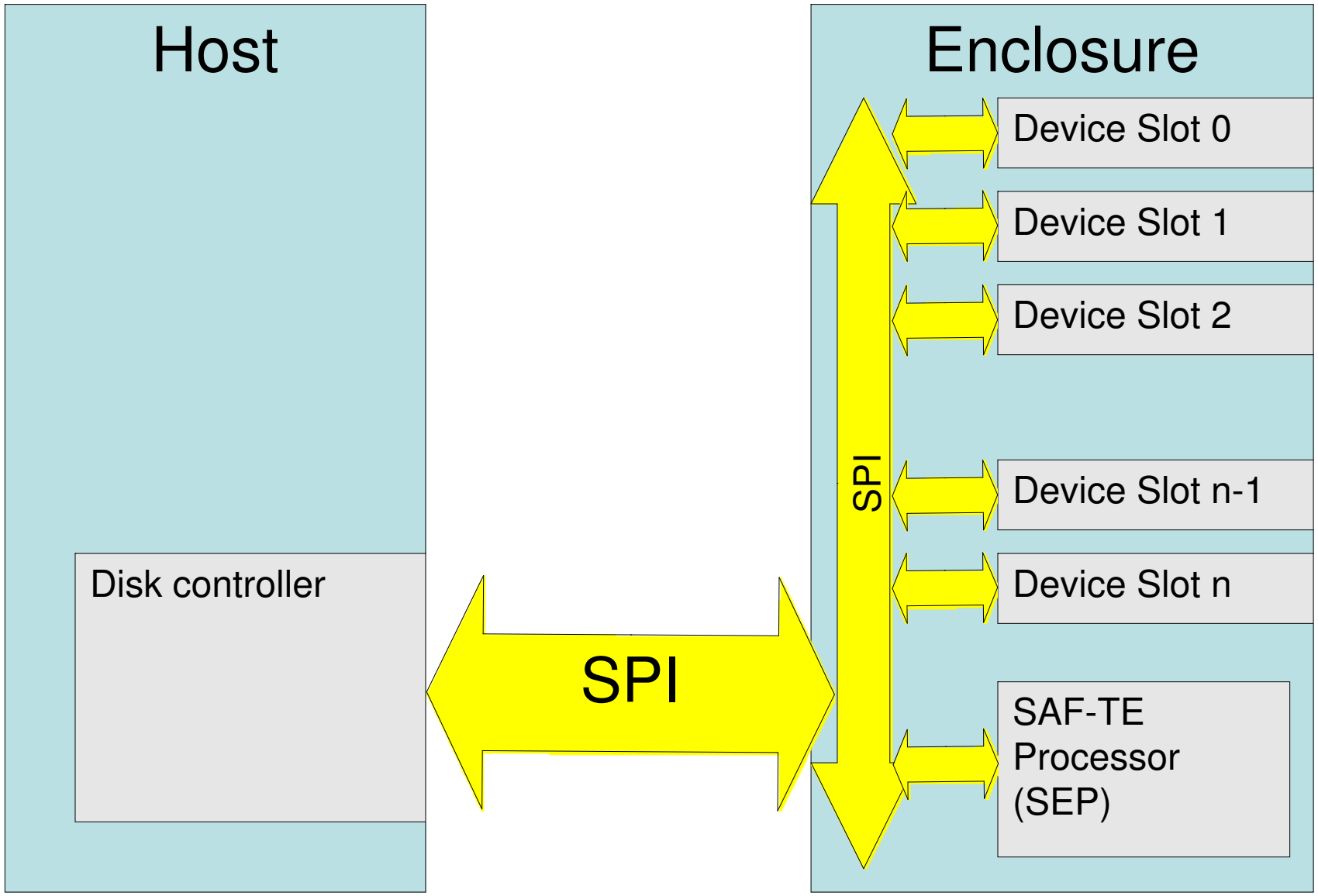
- Large storage systems usually assume some redundancy:
  - redundant disks (RAIDs),
  - redundant power supplies (PSUs),
  - redundant power sources (UPSes).
  - redundant cooling (fans, blowers, etc),
  - redundant connections (cables, switches, expanders, etc), ...
- To detect failure system should have number of sensors:
  - temperature,
  - power voltage/current,
  - UPS battery capacity,
  - fan speed, ...
- To support maintenance system should allow:
  - identify components,
  - disable failed component,
  - provide visual and audio status information.

- To provide that functionality several enclosure management interfaces are standardised now:
  - SCSI Accessed Fault-Tolerant Enclosure (SAF-TE),
  - SCSI Enclosure Services (SES),
  - Serial ATA Enclosure Management Bridge (SEMB),
  - SAF-TE/SES over I2C,
  - Serial General Purpose Input/Output (SGPIO).

- Lets look closer on that hardware:

- SAF-TE is an industry standard to interface an enclosure to a (parallel) SCSI subystem, developed in 1995-1997.
- SAF-TE reported as SCSI Processor type target device.
- It uses READ BUFFER/WRITE BUFFER SCSI commands to access enclosure configuration, status and control information.
- SAF-TE may report:
  - number of device slots and device presence,
  - presence and status of fans, power supplies, thermal sensors, speakers.
- SAF-TE may control:
  - devices power (for hot-plug),
  - device LEDs (locate, failure, RAID status: rebuild, check, fail, spare, etc),
  - devices SCSI IDs,
  - fan speeds,
  - power supplies on/off.

- SAF-TE on SPI:


Host
Disk controller
SPI
Enclosure
SPI
Device Slot 0
Device Slot 1
Device Slot 2
Device Slot n-1
Device Slot n
SAF-TE
Processor
(SEP)

- SES is a SAF-TE successor developed in 2002-2008. It has more flexible and unified design. It was improved to support SAS and FiberChannel protocols and wider list of elements.
- SES may exist as separate SCSI Enclosure type target device, or as logical unit of other device.
- It uses RECEIVE DIAGNOSTIC RESULTS/SEND DIAGNOSTIC SCSI commands to access several enclosure configuration, status and control diagnostic pages.
- Enclosure configuration defines how many elements of each type (Device Slots, Power Supplies, Fans, Temperature Sensors, etc) enclosure includes. Most of status reporting and control implemented in context of these elements.
- Control and status pages use 4 bytes to describe each element. First byte is unified, it includes general element status code and some common bits. Others 3 bytes are element type-specific.

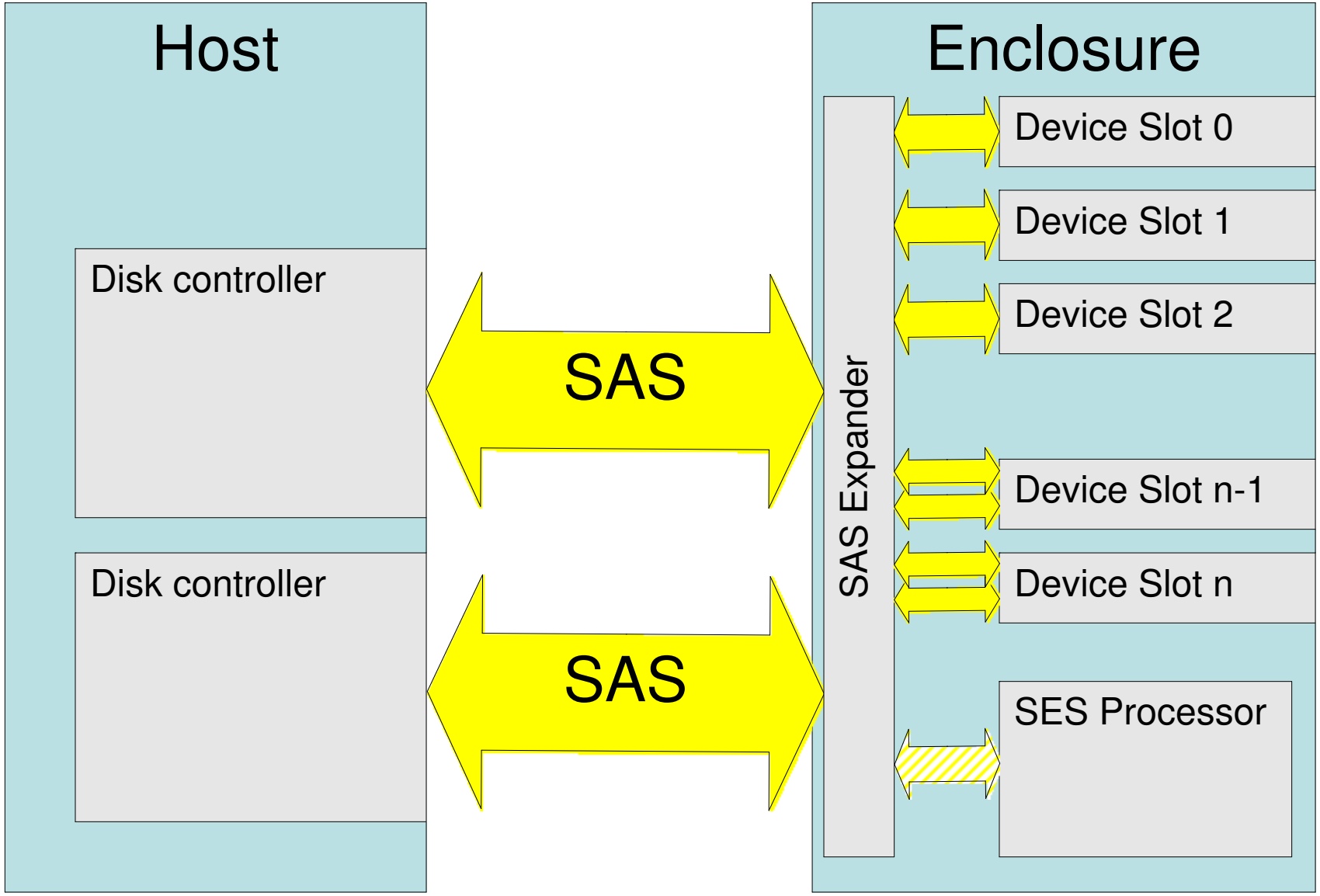
- SES element status:

Byte\Bit	7	6	5	4	3	2	1	0
0	COMMON STATUS							
	Reserved	PRDFAIL	DISABLED	SWAP	ELEMENT STATUS CODE			
1	Element type specific status information							
3								

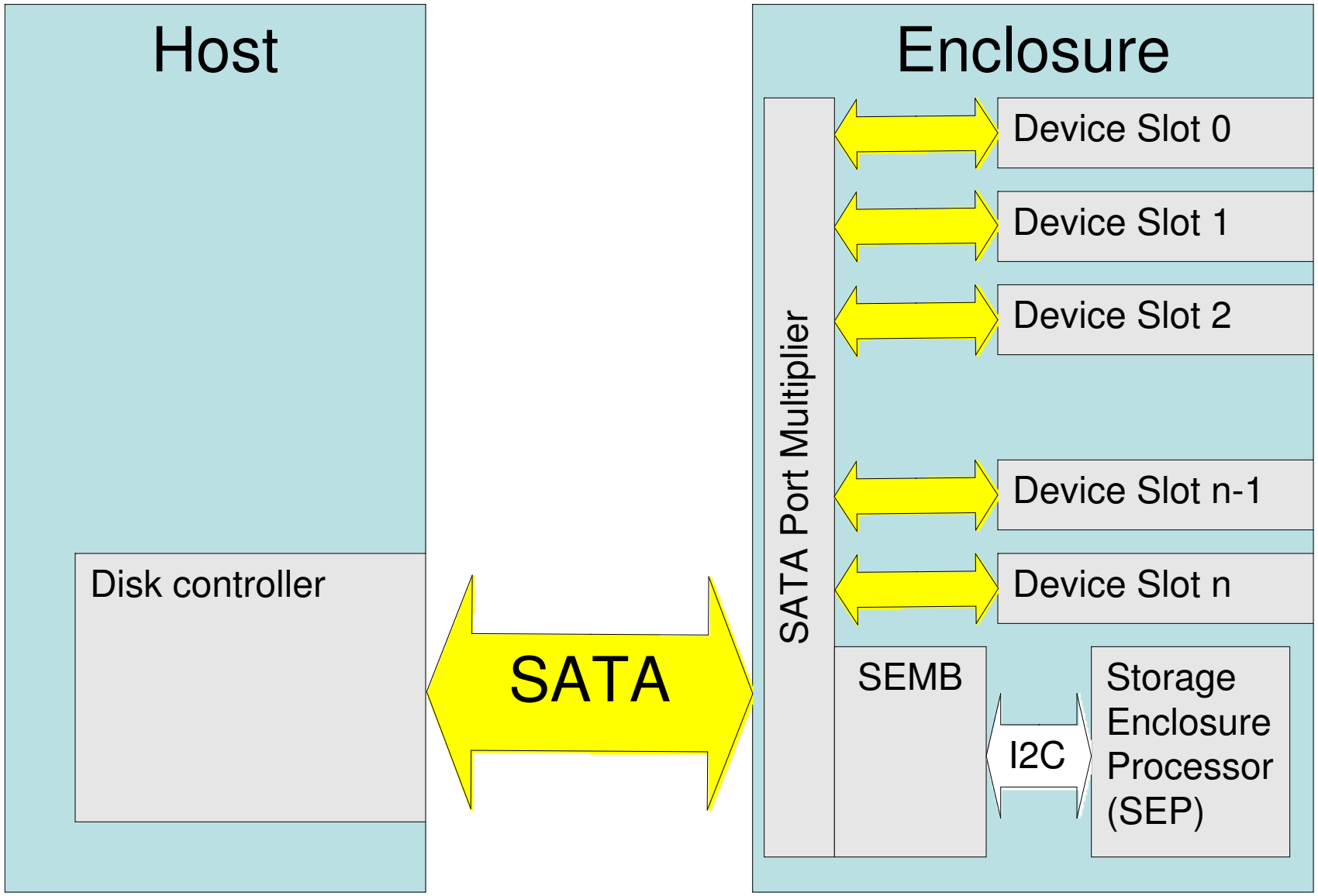
- SES Array Device Slot element status:

Byte\Bit	7	6	5	4	3	2	1	0
0	COMMON STATUS							
1	OK	RSVD DEVICE	HOT SPARE	CONS CHK	IN CRIT ARRAY	IN FAILED ARRAY	REBUILD/ REMAP	R/R ABORT
2	APP CLIENT BYPASSED A	DO NOT REMOVE	ENCLOSURE BYPASSED A	ENCLOSURE BYPASSED B	READY TO INSERT	RMV	IDENT	REPORT
3	APP CLIENT BYPASSED B	FAULT SENSED	FAULT REQSTD	DEVICE OFF	BYPASSED A	BYPASSED B	DEVICE BYPASSED A	DEVICE BYPASSED B

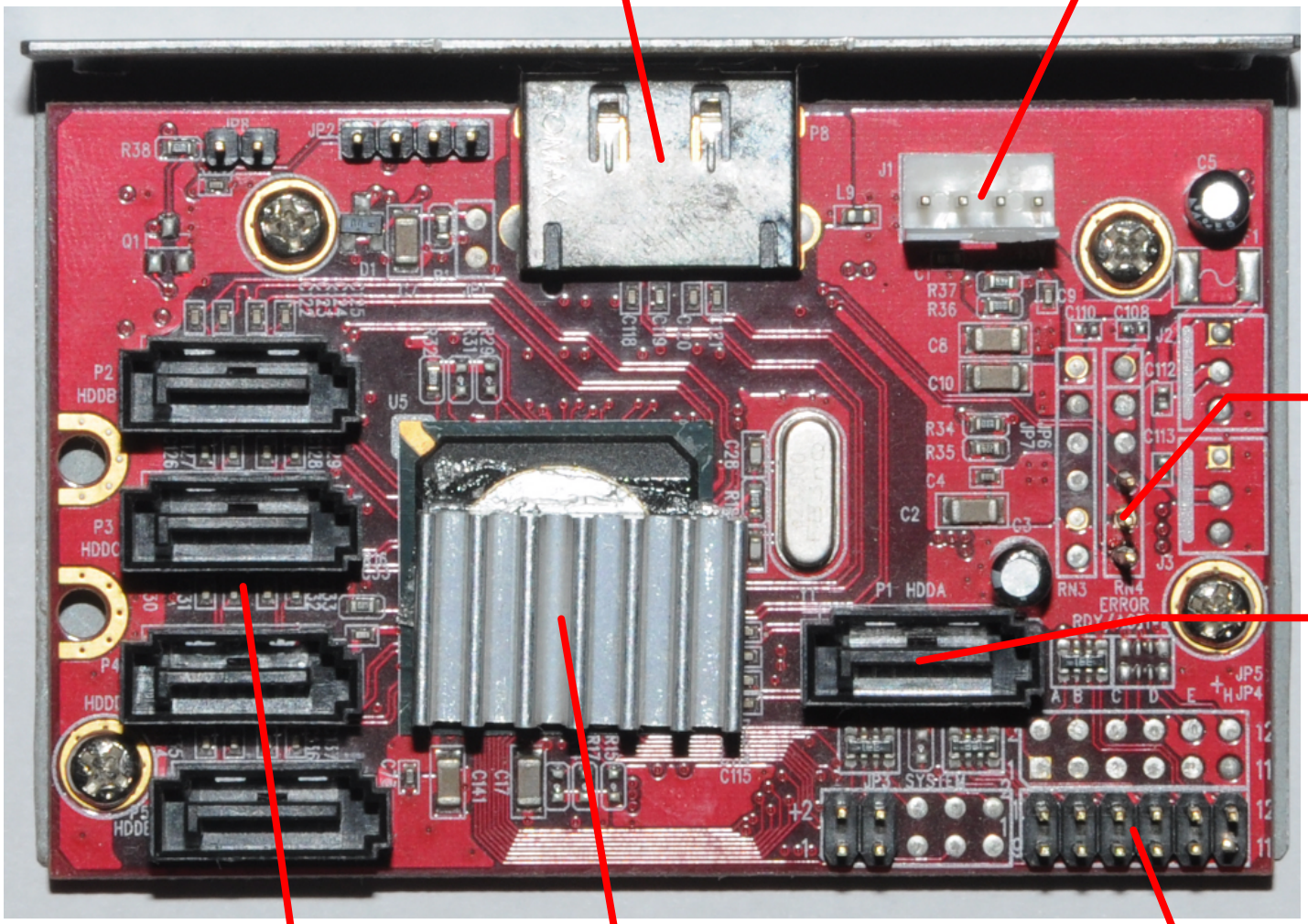
- SES on SAS expander:



- SES/SAF-TE with SEMB in SATA Port Multiplier:



- SiI3726 SATA Port Multiplier:



- I2C can be used to interface with SES/SAF-TE enclosures.
- I2C is a low-speed (10 kbit/s - 3.4 mbit/s) serial interface, using 4-pin connector with 3 wires: Clock (SCL), Data (SDA) and Groud.
- Supports up to 112 addressable devices, but in storages used as point-to-point with single target device at address C0h.
- Transfers commands and data blocks according to SES (RECEIVE DIAGNOSTIC RESULT/SEND DIAGNOSTIC) or SAF-TE (READ BUFFER/WRITE BUFFER) commands.

SEMB (master) to SEP (slave) transfer – transfer the command

S	SEP ADDRESS	R/W (0)	A	CMD_ TYPE	A	CHK SUM	A	SEMB ADDRESS	A	SEQ (0)	A	SEP_C MD	A	CHK SUM	A	P

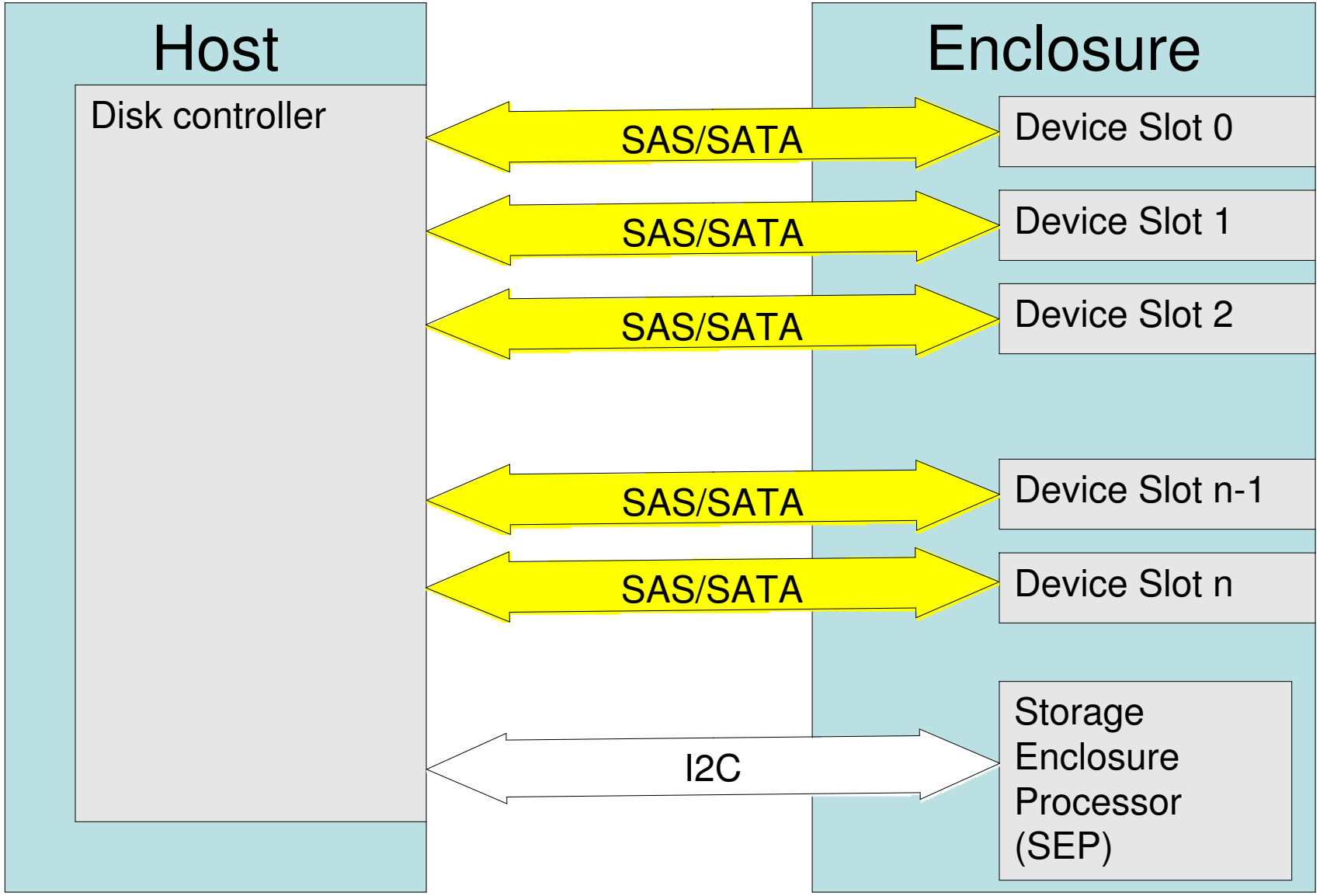
SEP (master) to SEMB (slave) transfer – transfers both the data and status

S	SEMB ADDRESS	R/W (0)	A	CMD_TYPE	A	CHK SUM	A	SEP ADDRESS	A	SEQ (0)	A	SEP_CMD	A	...

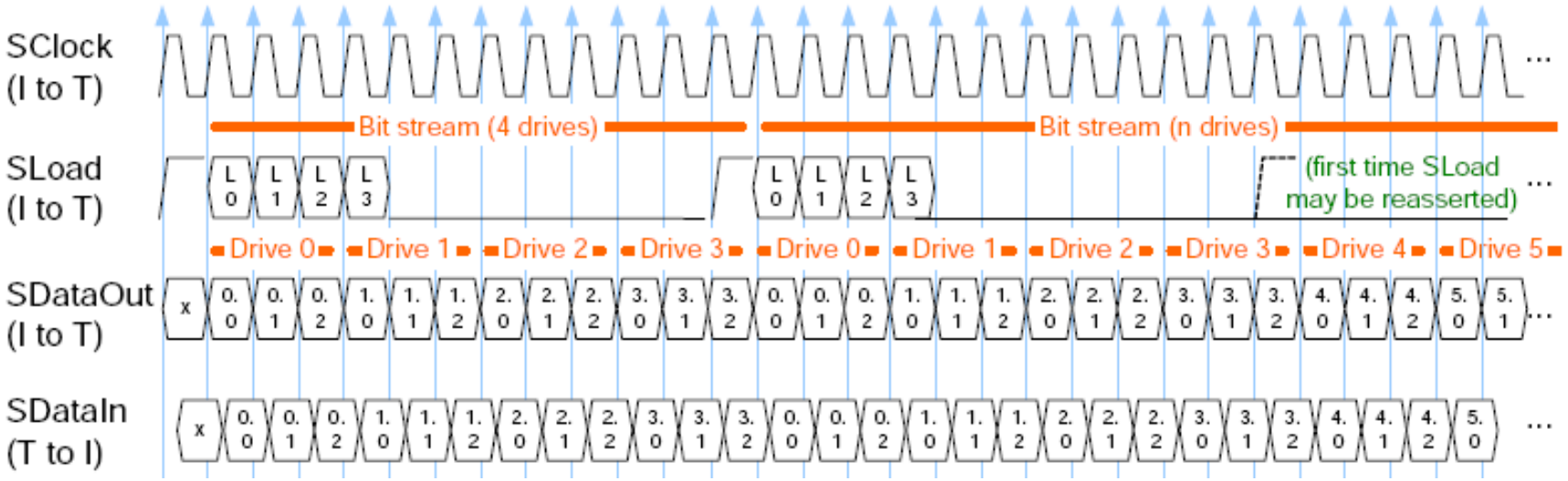
...	SEP STATUS	A	Data[0]	A	Data[1]	A	...	Data[LEN*4-1]	A	CHKSUM	A	P

From Master to Slave From Slave to Master

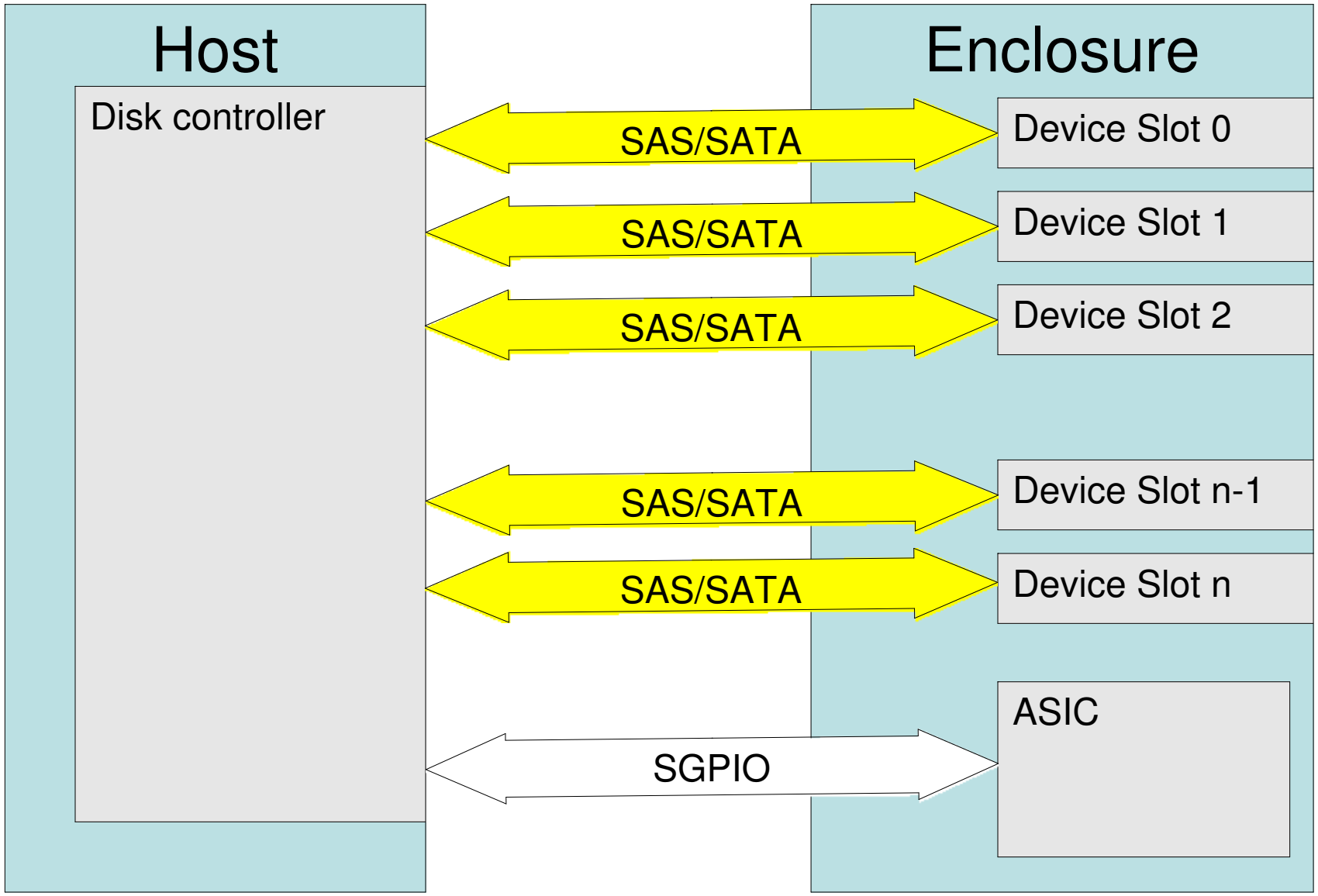
- SES/SAF-TE via I2C:



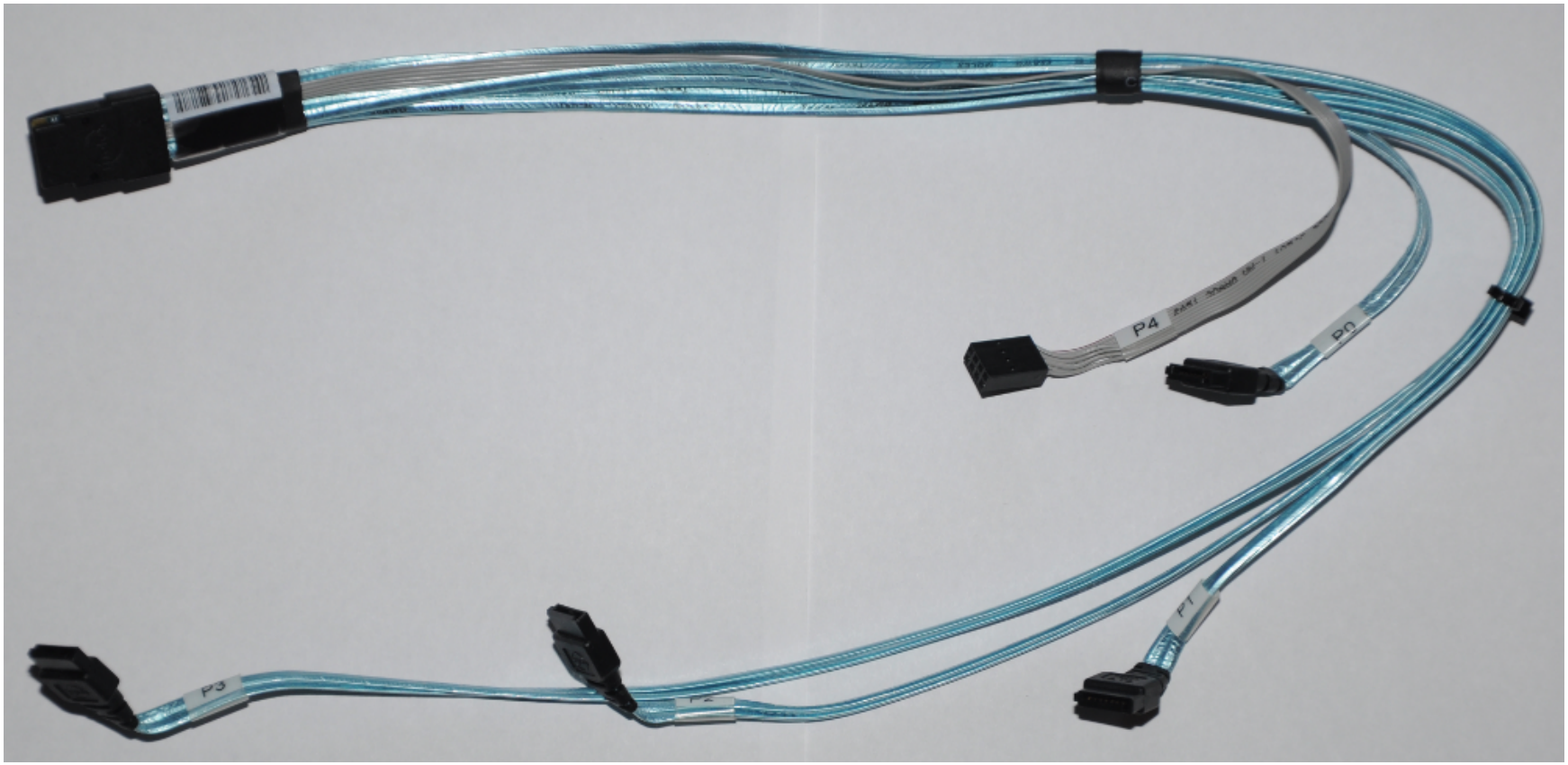
- In simple cases SGPIO interface may be used instead of I2C.
- SGPIO (SFF-8485) is a low-speed (32 bit/s to 100Kbit/s) serial interface. Uses 4 or 5 wires: Ground, SClock, SLoad, SDataOut and optionally SDataIn. Uses separate 6/8-pin cable/connector or dedicated wires in SFF-8087 internal mini-SAS connector.
- Transmits two bit streams with 3 bits allocated for each device. On output: activity, locate, fail. On input (when implemented): device presence.
- No enclosure monitoring, used mostly for internal backplanes.



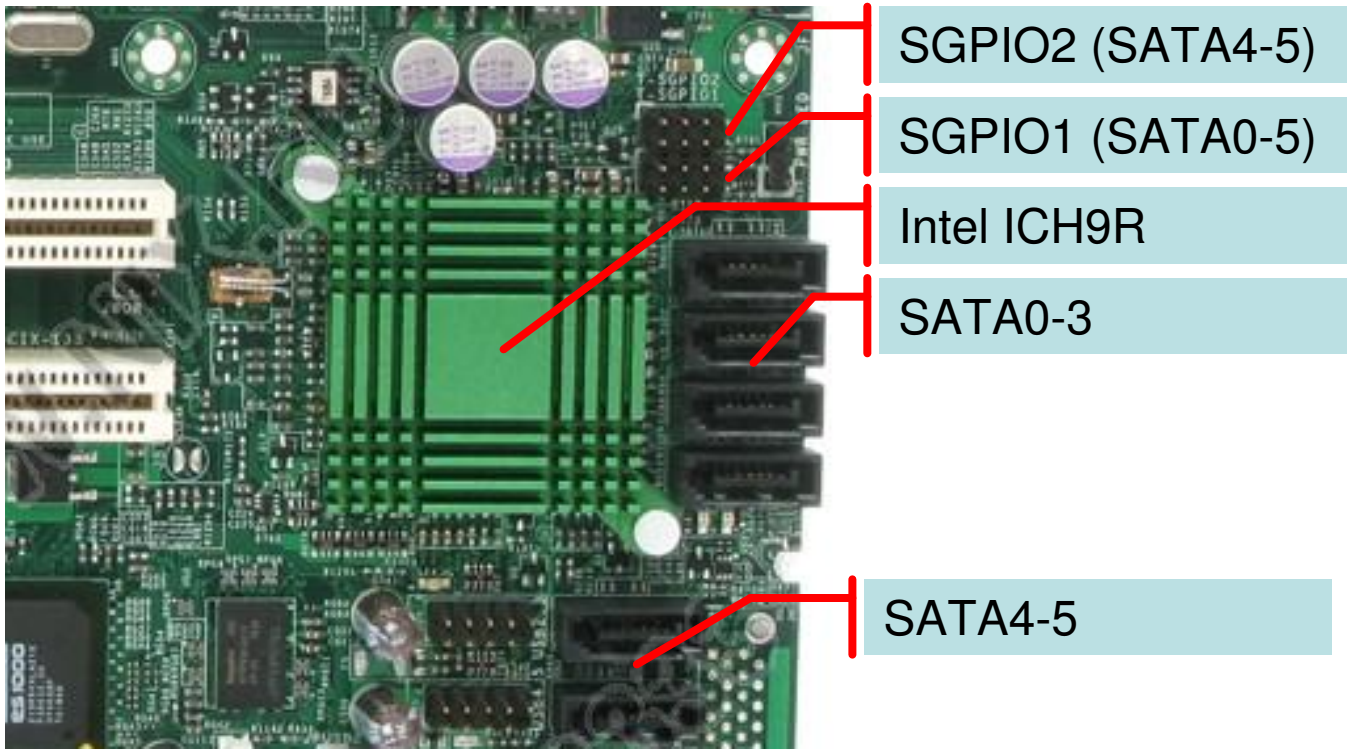
- SGPIO:


Host
Disk controller
Enclosure
SAS/SATA
SAS/SATA
SAS/SATA
SAS/SATA
SAS/SATA
SGPIO
Device Slot 0
Device Slot 1
Device Slot 2
Device Slot n-1
Device Slot n
ASIC

- SFF-8087 to 4xSATA cable with 8-pin SGPIO connector:



- In cases when enclosure connected via I2C or SGPIO to the controller, it's usage is controller dependent: RAID controllers may use it internally, while HBAs may pass it to the driver.
- AHCI specification defines API to communicate to SGPIO, SES and SAF-TE backplanes. Intel ICH8R and later chips support SGPIO interface.
  - SGPIO connectors on Supermicro X7SBE motherboard:



- And now to the FreeBSD:
- SES and SAF-TE SCSI devices supported by the «ses» driver of CAM. Driver API mostly based on SES protocol as more functional. «ses» driver emulates SES for SAF-TE devices.
- New «enc» CAM driver is WIP refactoring of «ses».

- SAF-TE enclosure with 5 slots and three drives on SPI:
  - SAF-TE enclosure in dmesg output:

```
%dmesg |grep enc1
enc1 at ahc0 bus 0 scbus5 target 6 lun 0
enc1: <ESG-SHV SCA HSBP M14 0.03> Fixed Processor SCSI-2 device
enc1: 3.300MB/s transfers
enc1: SAF-TE Compliant Device
enc1: Nfans 0 Npwr 0 Nslots 5 Lck 0 Ntherm 0 Nspkrs 0 Ntstats 0
```

  - camcontrol devlist output:

```
%camcontrol devlist
<IBM DDYS-T36950M S96H>            at scbus5 target 0 lun 0 (da0,pass6)
<IBM DDYS-T36950M S96H>            at scbus5 target 1 lun 0 (da1,pass7)
<IBM DDYS-T36950M S96H>            at scbus5 target 2 lun 0 (da2,pass8)
<ESG-SHV SCA HSBP M14 0.03>        at scbus5 target 6 lun 0 (enc1,pass4)
```

– getencstat tool allows to check enclosure status:

```
%getencstat -v /dev/encl
/dev/enc1: Enclosure Status <OK>
Element 0x0: Temperature Sensors, status: Not Available (0x07 0x00 0x00 0x00)
Element 0x1: Array Device Slot, status: OK (0x01 0x00 0x00 0x00)
Element 0x2: Array Device Slot, status: OK (0x01 0x00 0x00 0x00)
Element 0x3: Array Device Slot, status: OK (0x01 0x00 0x00 0x00)
Element 0x4: Array Device Slot, status: Not Installed (0x05 0x00 0x08 0x10)
Element 0x5: Array Device Slot, status: Not Installed (0x05 0x00 0x08 0x10)
```

- setobjstat tool allows to update control page for the specified element. For example, to turn on identify LED:
  - before:

```
%getencstat -v /dev/enc1
/dev/enc1: Enclosure Status <OK>
Element 0x0: Temperature Sensors, status: Not Available (0x07 0x00 0x00 0x00)
Element 0x1: Array Device Slot, status: OK (0x01 0x00 0x00 0x00)
Element 0x2: Array Device Slot, status: OK (0x01 0x00 0x00 0x00)
Element 0x3: Array Device Slot, status: OK (0x01 0x00 0x00 0x00)
Element 0x4: Array Device Slot, status: Not Installed (0x05 0x00 0x08 0x10)
Element 0x5: Array Device Slot, status: Not Installed (0x05 0x00 0x08 0x10)
```

  - after:

```
%setobjstat /dev/enc1 0x1 0x80 0x00 0x02 0x00
%getencstat -v /dev/enc1
/dev/enc1: Enclosure Status <OK>
Element 0x0: Temperature Sensors, status: Not Available (0x07 0x00 0x00 0x00)
Element 0x1: Array Device Slot, status: OK (0x01 0x80 0x02 0x00)
Element 0x2: Array Device Slot, status: OK (0x01 0x00 0x00 0x00)
Element 0x3: Array Device Slot, status: OK (0x01 0x00 0x00 0x00)
Element 0x4: Array Device Slot, status: Not Installed (0x05 0x00 0x08 0x10)
Element 0x5: Array Device Slot, status: Not Installed (0x05 0x00 0x08 0x10)
```

- One of the problem with enclosures is to associate devices with enclosure slots. New «enc» driver compares SAS device identifiers read from the device and from the SES enclosure:

```
/dev/enc0: Enclosure Status <INFO>
Element 0x0: Array Device Slot, status: Unsupported (0x00 0x00 0x00 0x00), descriptor: 'Drive Slots'
Element 0x1: Array Device Slot, status: OK (0x01 0x00 0x00 0x00), descriptor: 'Slot 01', dev: 'da0,pass0'
Element 0x2: Array Device Slot, status: OK (0x01 0x00 0x00 0x00), descriptor: 'Slot 02', dev: 'da1,pass1'
Element 0x3: Array Device Slot, status: OK (0x01 0x00 0x00 0x00), descriptor: 'Slot 03', dev: 'da2,pass2'
Element 0x4: Array Device Slot, status: Not Installed (0x05 0x00 0x00 0x00), descriptor: 'Slot 04'
Element 0x5: Array Device Slot, status: Not Installed (0x05 0x00 0x00 0x00), descriptor: 'Slot 05'
Element 0x6: Array Device Slot, status: Not Installed (0x05 0x00 0x00 0x00), descriptor: 'Slot 06'
Element 0x7: Array Device Slot, status: Not Installed (0x05 0x00 0x00 0x00), descriptor: 'Slot 07'
Element 0x8: Array Device Slot, status: Not Installed (0x05 0x00 0x00 0x00), descriptor: 'Slot 08'
Element 0x9: Array Device Slot, status: Not Installed (0x05 0x00 0x00 0x00), descriptor: 'Slot 09'
Element 0xa: Array Device Slot, status: Not Installed (0x05 0x00 0x00 0x00), descriptor: 'Slot 10'
Element 0xb: Array Device Slot, status: Not Installed (0x05 0x00 0x00 0x00), descriptor: 'Slot 11'
Element 0xc: Array Device Slot, status: Not Installed (0x05 0x00 0x00 0x00), descriptor: 'Slot 12'
Element 0xd: Temperature Sensors, status: Unsupported (0x00 0x00 0x00 0x00), descriptor: 'Temperature Sen
Element 0xe: Temperature Sensors, status: OK (0x01 0x00 0x31 0x00), descriptor: 'Temperature'
Element 0xf: Cooling, status: Unsupported (0x00 0x00 0x00 0x00), descriptor: 'Fans'
Element 0x10: Cooling, status: Not Installed (0x05 0x00 0x00 0x50), descriptor: 'Fan1'
Element 0x11: Cooling, status: Not Installed (0x05 0x00 0x00 0x50), descriptor: 'Fan2'
Element 0x12: Cooling, status: Not Installed (0x05 0x00 0x00 0x50), descriptor: 'Fan3'
Element 0x13: Audible alarm, status: Unsupported (0x00 0x00 0x00 0x00), descriptor: 'Buzzers'
Element 0x14: Audible alarm, status: OK (0x01 0x00 0x00 0x0f), descriptor: 'Buzzer'
Element 0x15: Voltage Sensor, status: Unsupported (0x00 0x00 0x00 0x00), descriptor: 'Voltage Sensors'
Element 0x16: Voltage Sensor, status: OK (0x01 0x00 0x01 0xf8), descriptor: '5V'
Element 0x17: Voltage Sensor, status: OK (0x01 0x00 0x04 0xa7), descriptor: '12V'
Element 0x18: Current Sensor, status: Unsupported (0x00 0x00 0x00 0x00), descriptor: 'Current Sensors'
Element 0x19: Current Sensor, status: OK (0x01 0x00 0x00 0xa8), descriptor: '5V Line Current Sensor'
Element 0x1a: Current Sensor, status: OK (0x01 0x00 0x10 0x42), descriptor: '12V Line Current Sensor'
Element 0x1b: Power Supply, status: Unsupported (0x00 0x00 0x00 0x00), descriptor: 'Power Supplies'
Element 0x1c: Power Supply, status: Not Installed (0x05 0x00 0x00 0x00), descriptor: 'Power Supply 1'
Element 0x1d: Power Supply, status: Not Installed (0x05 0x00 0x00 0x00), descriptor: 'Power Supply 2'
Element 0x1e: Enclosure, status: Unsupported (0x00 0x00 0x00 0x00), descriptor: 'Enclosure'
Element 0x1f: Enclosure, status: OK (0x01 0x00 0x00 0x00), descriptor: 'Enclosure'
```

- CAM-based ATA allowed to add SEMB support to the new «enc» driver (not committed yet):
- SES SATA backplane connected via I2C to SEMB in PMP:
  - SiI3726 Port Multiplier in dmesg output:

```
pmp0 at siisch3 bus 0 scbus3 target 15 lun 0
pmp0: <Port Multiplier 37261095 1706> ATA-0 device
pmp0: 300.000MB/s transfers (SATA 2.x, NONE, PIO 8192bytes)
pmp0: 6 fan-out ports
pmp0: port 0 status: 00000123
pmp0: port 1 status: 00000123
pmp0: port 2 status: 00000123
pmp0: port 3 status: 00000000
pmp0: port 4 status: 00000000
pmp0: port 5 status: 00000113
```

Disks

SEMB

  - SEMB SES enclosure in dmesg output:

```
%dmesg |grep enc0
enc0 at siisch3 bus 0 scbus3 target 5 lun 0
enc0: <AMI MG9071 1.00 0011> SEMB S-E-S 2.00 device
enc0: 150.000MB/s transfers (SATA 1.x, NONE, PIO 8192bytes)
enc0: SEMB SES Device
enc0: Generation Code 0x0 has 1 SubEnclosures
enc0:  SubEnclosure ID 0, 4 Types With this ID, Descriptor Length 36, offset 8
enc0: WWN: 3530303330343831
enc0:  Type Desc[0]: Type 0x17, MaxElt 4, In Subenc 0, Text Length 0:
enc0:  Type Desc[1]: Type 0x4, MaxElt 1, In Subenc 0, Text Length 0:
enc0:  Type Desc[2]: Type 0xe, MaxElt 1, In Subenc 0, Text Length 0:
enc0:  Type Desc[3]: Type 0x6, MaxElt 1, In Subenc 0, Text Length 0:
```

– camcontrol devlist output:

```
<ST3500418AS CC46>                 at scbus3 target 0 lun 0 (pass2,ada1)
<ST3500418AS CC46>                 at scbus3 target 1 lun 0 (pass3,ada2)
<ST3320418AS CC37>                 at scbus3 target 2 lun 0 (pass1,ada0)
<AMI MG9071 1.00 0011>             at scbus3 target 5 lun 0 (pass4,enc0)
<Port Multiplier 37261095 1706>    at scbus3 target 15 lun 0 (pmp0,pass0)
```

– getencstat output:

```
/dev/enc0: Enclosure Status <OK>
Element 0x0: Array Device Slot, status: Unsupported (0x00 0x00 0x00 0x00)
Element 0x1: Array Device Slot, status: OK (0x01 0x00 0x00 0x00), descriptor: 'SLOT 000'
Element 0x2: Array Device Slot, status: OK (0x01 0x00 0x00 0x00), descriptor: 'SLOT 001'
Element 0x3: Array Device Slot, status: OK (0x01 0x00 0x00 0x00), descriptor: 'SLOT 002'
Element 0x4: Array Device Slot, status: Not Installed (0x05 0x00 0x00 0x00), descriptor:
Element 0x5: Temperature Sensors, status: Unsupported (0x00 0x00 0x00 0x00)
Element 0x6: Temperature Sensors, status: OK (0x01 0x00 0x31 0x00)
Element 0x7: Enclosure, status: Unsupported (0x00 0x00 0x00 0x00)
Element 0x8: Enclosure, status: OK (0x01 0x00 0x00 0x00), descriptor: 'BOX 001 '
Element 0x9: Audible alarm, status: Unsupported (0x00 0x00 0x00 0x00)
Element 0xa: Audible alarm, status: OK (0x01 0x00 0x00 0x00), descriptor: 'BUZZER 0'
```

- ahci(4) driver supports AHCI SGPIO (LED) interface:

```
ahci0: <Intel ICH8 AHCI SATA controller> port 0xf0b0-0xf0b7,0xf0a0-0xf0a3,0xf090
-0xf097,0xf080-0xf083,0xf060-0xf07f mem 0xfe605000-0xfe6057ff irq 19 at device 3
1.2 on pci0
ahci0: attempting to allocate 1 MSI vectors (1 supported)
ahci0: using IRQ 257 for MSI
ahci0: AHCI v1.30 with 6 6Gbps ports, Port Multiplier not supported
ahci0: Caps: 64bit NCQ SNTF ALP AL CLO 6Gbps PMD SSC PSC 32cmd EM 6ports
ahci0: Caps2: APST
ahci0: EM Caps: ALHD XMT SMB LED
```

Now ahci(4) exports SGPIO LEDs via led(4):

```
# ls /dev/led
ahcich0.fault   ahcich1.locate  ahcich3.fault   ahcich4.locate
ahcich0.locate  ahcich2.fault   ahcich3.locate  ahcich5.fault
ahcich1.fault   ahcich2.locate  ahcich4.fault   ahcich5.locate
```

SES emulation for SGPIO is planned.
I2C SES and SAF-TE support is not implemented in any AHCI hardware now.

- Kernel now includes some support for enclosure LEDs:
  - GEOM RAID class reports disks statuses (ACTIVE, RESYNC, REBUILD, FAILED) to the underlying providers via GEOM::setstate attribute:

```
s = G_STATE_ACTIVE;
len = sizeof(s);
g_io_getattr(«GEOM::setstate», disk->d_consumer, &len, &s);
```

  - GEOM SLICE (GEOM PART, etc) passes all unknown attributes down.
  - GEOM DISK can be configured using kern.geom.disk.X.led sysctls to forward received GEOM::setstate attribute value to LEDs via led(4).

- TODO:
  - add support for SGPIO/I2C interfaces in more drivers,
  - associate devices with enclosure slots for non-SAS enclosures;
  - implement in-kernel interface for the «enc» CAM driver;
  - refactor GEOM::setstate API to handle full set of SES Array Device Slot flags;
  - explore possibility to make ZFS report disks statuses to GEOM using that API.

- Special thanks to:
  - iXsystems Inc. for provided hardware and supporting my work;

- Questions?