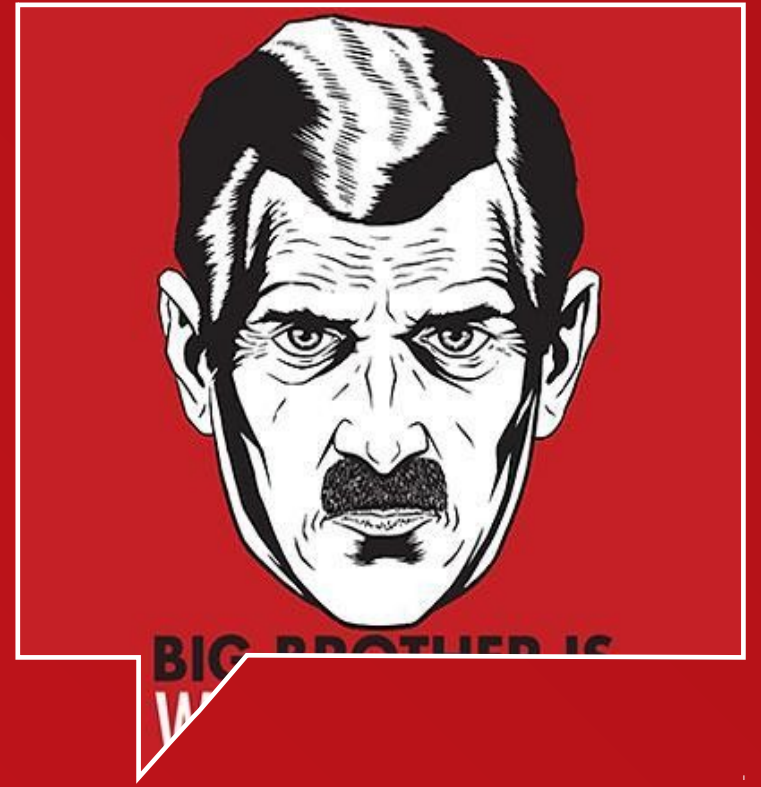


AMT

vPro

ME



# How to Become the Sole Owner of Your PC

POSITIVE TECHNOLOGIES

ptsecurity.com

**Maxim Goryachy**  
**Mark Ermolov**

# **AMT disable techniques**

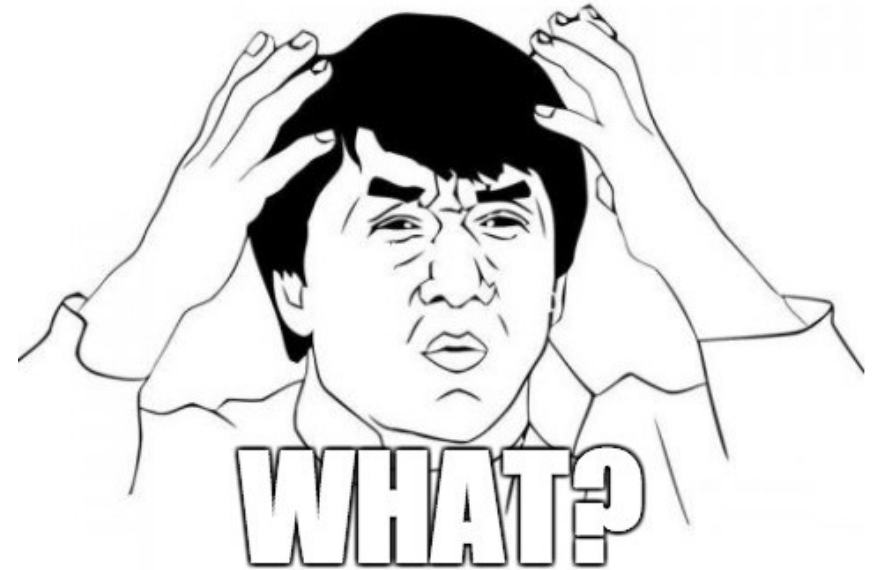
---

Positive Research Center

# What is it?

---

- **Second «hidden» processor** in your PC
- Built into every modern Intel-based PC
- Never sleeps (connected to mains? ME is active.)



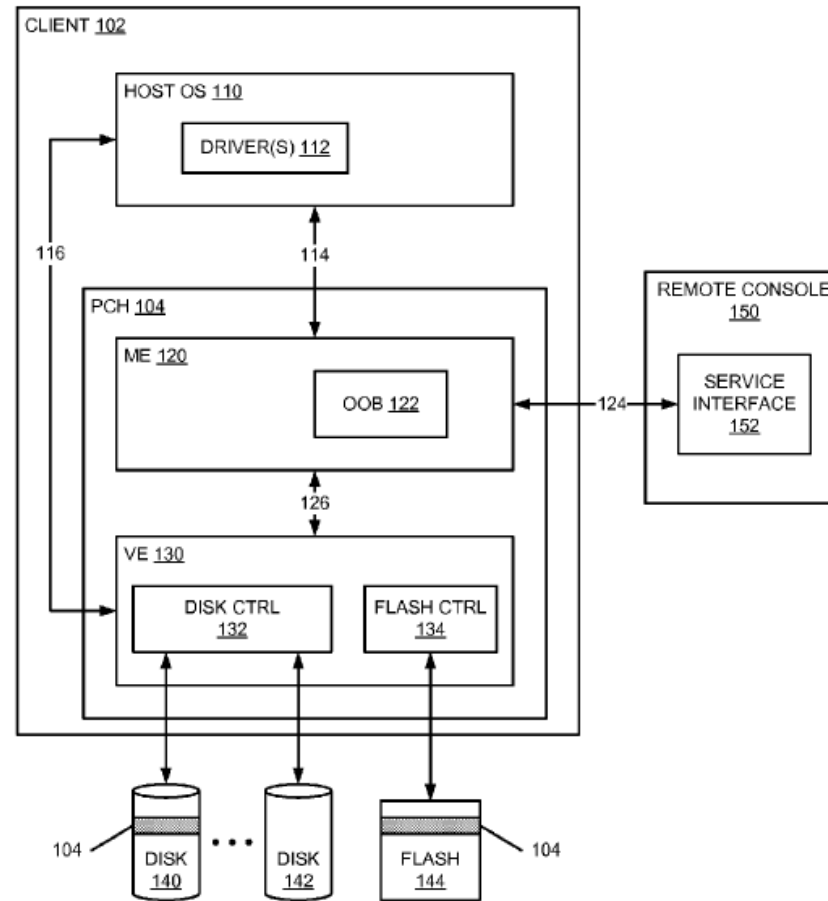
# Why you might want to disable it?

---

- A complicated hardware and firmware combination exposed to vulnerabilities and attacks (e.g. Alexander Tereshkin and Rafal Wojtczuk, “Introducing Ring -3 Rootkits”, Black Hat USA, July 29, 2009, Las Vegas, NV)
- Many potentially dangerous functions (remote control, NFC, hidden service partition)
- Undocumented interfaces and closed implementation (MEI, MDES, etc.)
- The platform vendor is the sole owner of the configuration policy

# Hidden Service Partition

Pat #: US 8,949,565 B2 VIRTUAL AND HIDDEN SERVICE PARTITION AND DYNAMIC ENHANCED THIRD PARTY DATA STORE



# What does “it” include

---

- **Out-of-band remote management solution** for personal computers in order to monitor, maintain, update, repair and otherwise control them (Web interface, WSMan based management API, IDE redirection, Serial-Over-Lan, KVM)
- **System defense component** including lowest-level network packet filter with customizable rules
- **Protected Audio/Video Pathway** for playback of DRM-protected media
- **Anti-Theft** to automatically lock the PC and erase encryption keys from TPM, either when a remote server signals the PC or upon delivery of “poison pill”

---

## • **Integrated Clock Control Service**

POSITIVE TECHNOLOGIES

- Some other system features (**ASF, QST**)

1. Failure of DRAM Init Done (DID)
2. Via ME flash region update mechanisms
  - HDA\_SDO pin-strap
  - HMR FPO Enable command
3. Soft temporary disable
4. ME runtime disable
5. Disruption of ME access path to UMA
6. Corruption of ME flash region

**Intel Management Engine (ME)** - an environment consisting of dedicated hardware and firmware components

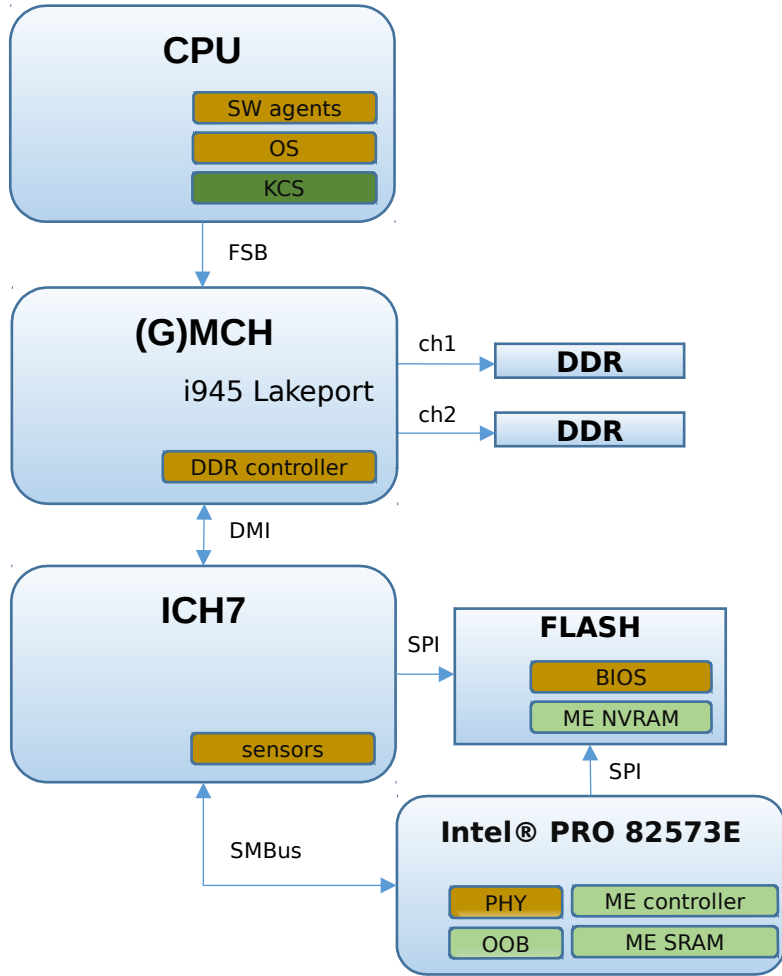
**Intel Active Management Technology (AMT)** - a firmware application running on the management engine

**Intel vPro** - a marketing name that covers a wide range of security and management features that are built in Intel processors and chipsets\*

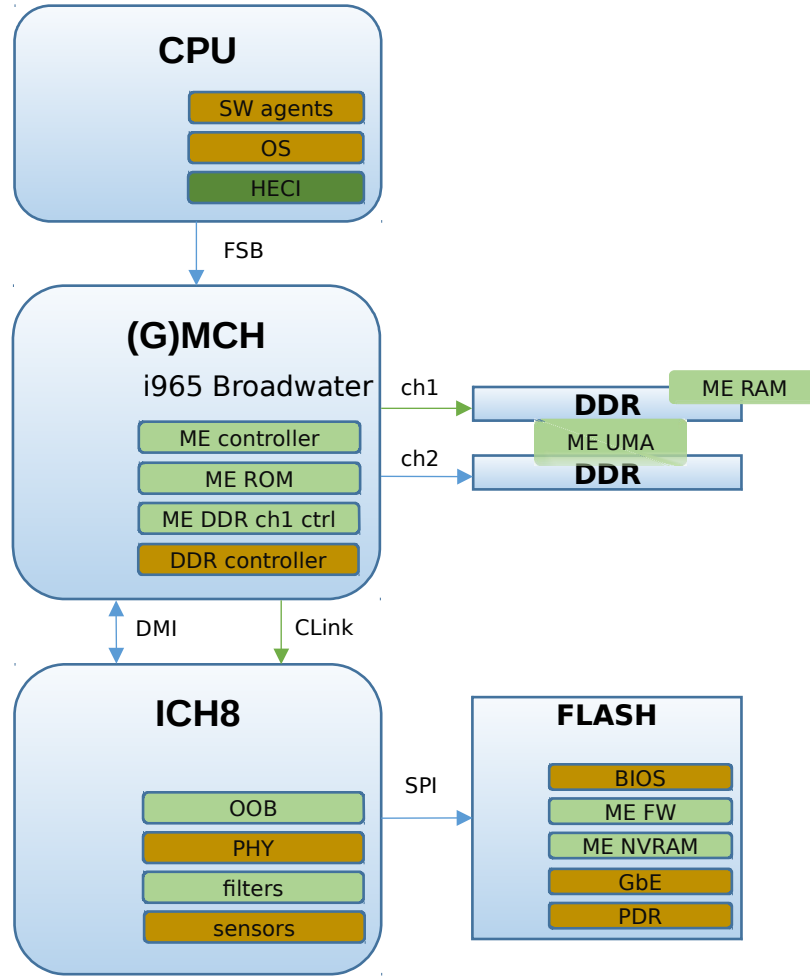


# AMT block scheme & evolution

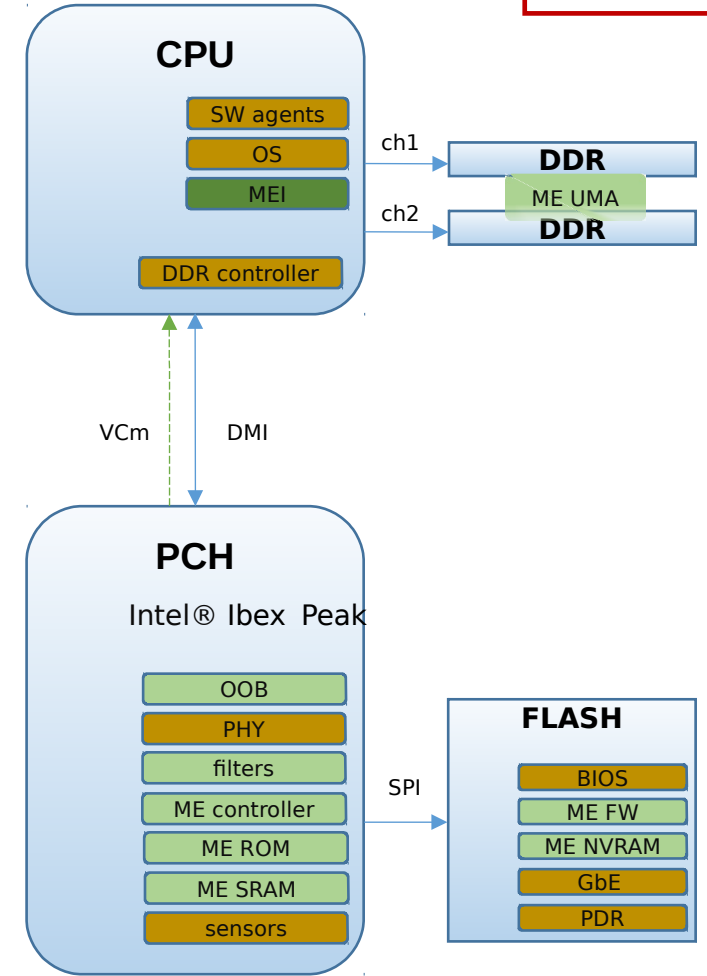
AMT 1.0 (Q1'05)



AMT 2.0 (Q2'06)



AMT 6.0 (Q3'09)



## **1. Failure of DRAM Init Done (DID)**

2. Via ME flash region update mechanisms

- HDA\_SDO pin-strap
- HMR FPO Enable command

3. Soft temporary disable

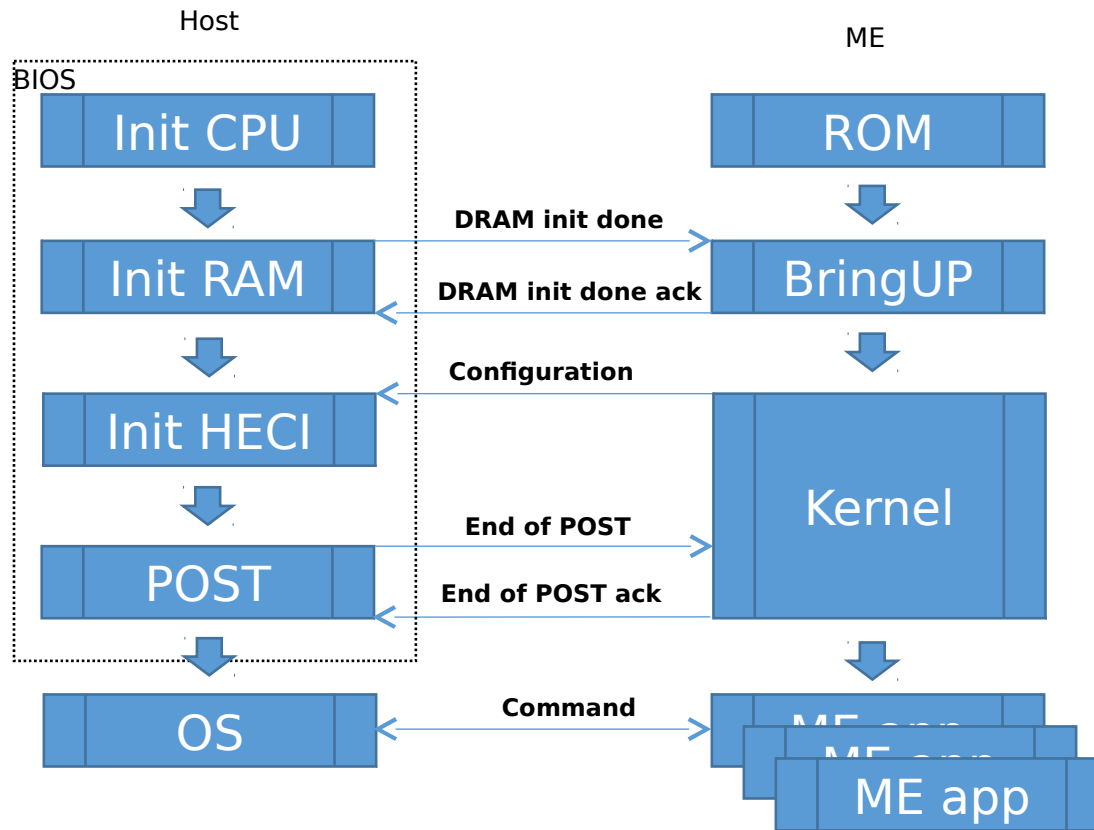
4. ME runtime disable

5. Disruption of ME access path to UMA

6. Corruption of ME flash region

- Host physical address space **stolen memory**
- Used as swap for ME SRAM
  - Code pages integrity checked by private CRC algorithm
  - Data pages are encrypted
- ME access UMA via PCI-E virtual channel (**VCm**)

# Power-on initialization scheme



# DRAM Init Done (DID)

## 23.1.1.16 H\_GS—Host General Status Register (Intel® MEI 1—D22:F0)

Address Offset: 4Ch–4Fh  
Default Value: 00000000h

Attribute: R/W  
Size: 32 bits

Bit	Description
31:0	Host General Status(H_GS)—R/W. General Status of Host, this field <u>is not used by Hardware</u>

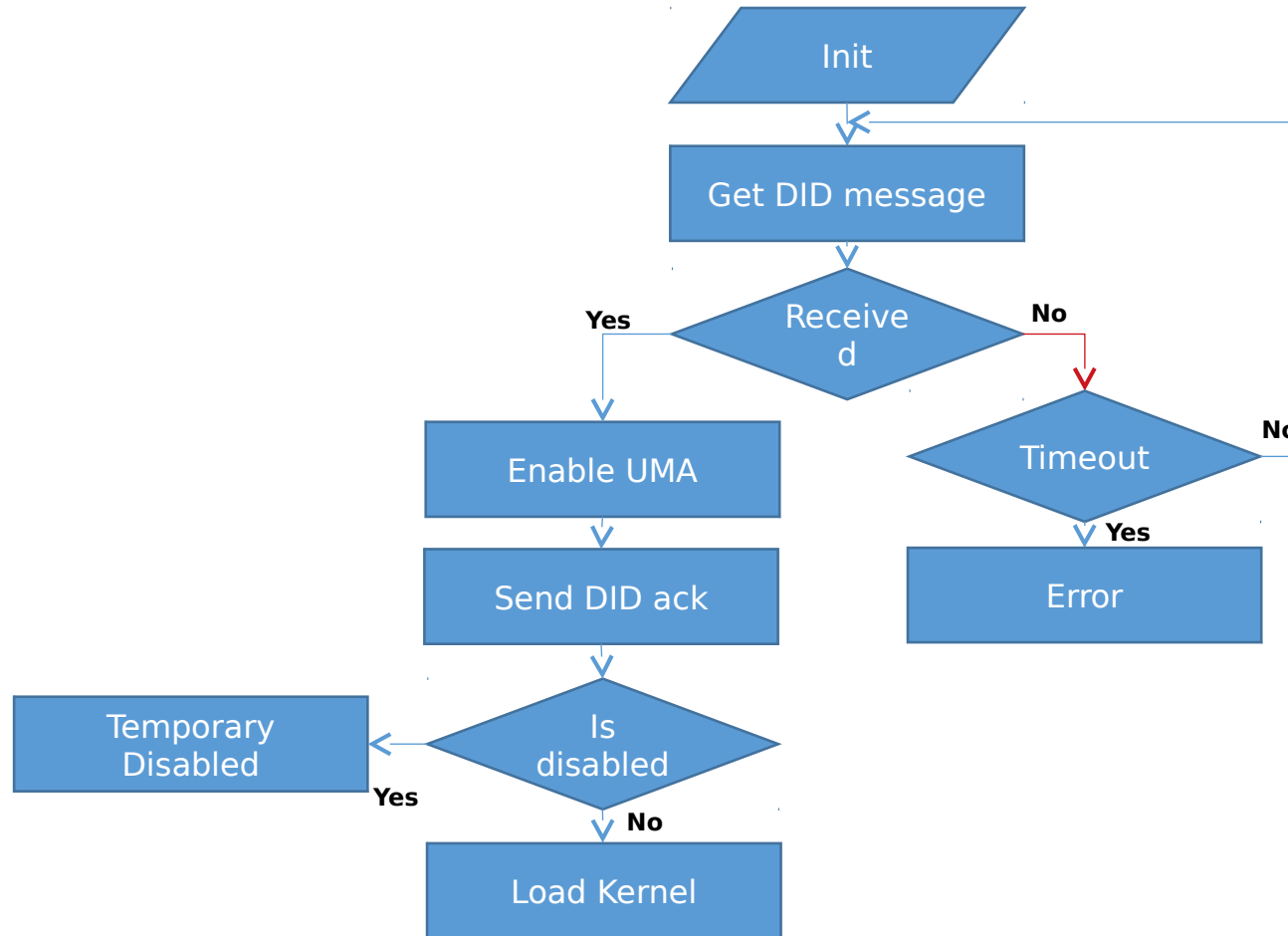
around the **definition of the DID message**, which should be written in H\_GS, **in core boot**

```
#define PCI_ME_H_GS      0x4c
#define ME_INIT_DONE      1
#define ME_INIT_STATUS_SUCCESS 0
#define ME_INIT_STATUS_NOMEM  1
#define ME_INIT_STATUS_ERROR  2
#define ME_INIT_STATUS_SUCCESS_OTHER 3 /* SEE ME9 BWG */

struct me_did {
    u32 uma_base: 16;
    u32 reserved: 7;
    u32 rapid_start: 1;
    u32 status: 4;
    u32 init_done: 4;
} __attribute__((packed));
```

# ME BringUP phase

14



1. Failure of DRAM Init Done (DID)

## **2. Via ME flash region update mechanisms**

- **HDA\_SDO pin-strap**
- **HMR FPO Enable command**

3. Soft temporary disable

4. ME runtime disable

5. Disruption of ME access path to UMA

6. Corruption of ME flash region

Intel® ME SPI Operations can be stopped in the following ways:

- Assert `HDA_SDO` (known as GPIO 33 or Flash descriptor override/Intel® ME manufacturing jumper) to high while powering on the system. This is not a valid method if the parameters are configured to ignore this jumper.
- Send the HMRFPO ENABLE Intel® MEI command to Intel® ME (for more information see the PCH Intel® ME BIOS writer's guide).\*

\* Intel ME System Tools user guide



# HDA\_SDO jumper

HDA_SDO	0	<p><b>Intel High Definition Audio Serial Data Out:</b> <u>Serial TDM data output to the codec(s).</u> This serial output is double-pumped for a bit rate of 48 Mb/s for Intel High Definition Audio.</p> <p><b>Note:</b> <u>This signal is sampled as a functional strap.</u> See Section 2.18 for more details. There is a weak integrated pull-down resistor on this pin.</p>
---------	---	---

Table 9-1. Functional Strap Definitions (Sheet 3 of 3)

Signal	Usage	When Sampled	Comment
HDA_SDO	Flash Descriptor Security Override	Rising edge of PCH_PWROK	<p>This signal has a weak internal pull-down.</p> <p>0 = <b>Enable</b> security measures defined in the Flash Descriptor. (Default)</p> <p>1 = <b>Disable Flash Descriptor Security (override).</b> <u>This strap should only be asserted high using external pull-up in manufacturing/debug environments ONLY.</u></p> <p><b>Notes:</b></p> <ol style="list-style-type: none"><li>1. The internal pull-down is disabled after PLTRST# de-asserts.</li><li>2. Asserting HDA_SDO high on the rising edge of PCH_PWROK will <u>also halt Intel Management Engine after Chipset bring up and disable runtime Intel ME features.</u> This is a debug mode and must not be asserted after manufacturing/debug.</li><li>3. This signal is in the primary well.</li></ol>

- Formerly called HECI (host-embedded communication interface)
- From host's view it is **internal PCI device** with BDF 0:22:0(1)
- Communication performed using **Ring Buffers** accessed by **MMIO registers of MEI**
- ME applications communicate with host applications through MEI using unique **client IDs hardcoded in firmware**
- Each client ID defines the structure of messages passing through MEI

# ME Kernel Host Interface (MKHI)

19

- MKHI functionality accessed using MEI client ID 0

```
#define MEI_ADDRESS_HBM      0x00
#define MEI_ADDRESS_CORE_WD 0x01
#define MEI_ADDRESS_AMT     0x02
#define MEI_ADDRESS_RESERVED 0x03
#define MEI_ADDRESS_WDT     0x04
#define MEI_ADDRESS_POLICY  0x05
#define MEI_ADDRESS_PASSWORD 0x06
#define MEI_ADDRESS_MKHI    0x07
#define MEI_ADDRESS_ICC     0x08
#define MEI_ADDRESS_THERMAL 0x09
#define MEI_ADDRESS_SPI     0x0a
```

```
struct mkhi_header {
    u32 group_id: 8;
    u32 command: 7;
    u32 is_response: 1;
    u32 reserved: 8;
    u32 result: 8;
} __attribute__((packed));
```

```
#define MKHI_GROUP_ID_CBM      0x00
#define MKHI_GROUP_ID_PM      0x01
#define MKHI_GROUP_ID_PWD     0x02
#define MKHI_GROUP_ID_FWCAPS  0x03
#define MKHI_GROUP_ID_APP     0x04
#define MKHI_GROUP_ID_HMRFPO  0x05
#define MKHI_GROUP_ID_MDES    0x08
#define MKHI_GROUP_ID_MAX     0x09
#define MKHI_GROUP_ID_GEN     0xff
```

- All MKHI messages have following header:

- Some MKHI command groups we've found in core boot

# HMR FPO Enable MKHI command

20

- HMR FPO - **Host ME Region Flash Protection Override**
- It has MKHI **command ID 0x01**, from the group MKHI\_GROUP\_ID\_HMRFPO (0x05)
- The binary sequence sent to MEI is: 0x800c0007 0x00000105  
0x00000000 0x00000000
- It can be sent only if another **MKHI HMR FPO Lock** command has not been sent yet
- It takes effect after next reboot and works only before subsequent reboot
- If the command is in effect, **ME region on SPI flash can be written** from host ignoring flash descriptor master access settings
- Some BIOS Setup have “ME FW Image Re-Flash” option that sends HMR FPO Enable

1. Failure of DRAM Init Done (DID)
2. Via ME flash region update mechanisms
  - HDA\_SDO pin-strap
  - HMR FPO Enable command

### **3. Soft temporary disable**

4. ME runtime disable
5. Disruption of ME access path to UMA
6. Corruption of ME flash region

# Soft temporary disable

22

- Performed also by MKHI command from MKHI\_GROUP\_ID\_FWCAPS (0x03, group
- The command has ID 0x03, core boot defines it as MKHI\_FWCAPS\_SET\_RULE, Rule ID for soft temporary disable is 0x06
- Binary sequence is: 0x800a0007 0x00000303 0x00000006 0x00000001
- **Can be send only before End of Post**
- Takes effect **after next reboot**, is stored in ME NVRAM and affects all subsequent reboots (and power offs)
- To bring out ME from the disabled state host writes dword value 0x20000000 to H\_GS MEI register
- In the soft temporary disabled state, the ME FW bring-up module **doesn't load the kernel** and freezes up while reading H\_GS
- ~~In some BIOS Setup, there is the option "Disable ME" that performs~~

1. Failure of DRAM Init Done (DID)
2. Via ME flash region update mechanisms
  - HDA\_SDO pin-strap
  - HMR FPO Enable command
3. Soft temporary disable
- 4. ME runtime disable**
5. Disruption of ME access path to UMA
6. Corruption of ME flash region

- Performed also by MKHI command from MKHI\_GROUP\_ID\_GEN (0xff) group
- The command ID is 0x10, **core boot doesn't define the command**
- Binary sequence is: 0x80040007 0x000010ff
- Can be completed successfully only if ME FW is in **Manufacture Mode**
- Can be sent to ME at any time, after End of Post and HMR FPO Lock
- Disable ME right away, doesn't need restart
- When the command is completed, ME doesn't detect CPU reset to receive DID or perform any communications via MEI
- ME recovers only after power off/power on cycle



# ME FW Manufacture mode

- A **special initial mode** of ME Firmware designed for platform testing by vendors \*
- Blocks HMR FPO Lock MEI command, so HMR FPO Enable can be sent at any moment to reflash ME FW
- Supports ME runtime disable MEI command
- Indicated by bit #4 of HFS MEI register (0x40 MEI config space offset)
- Do not set End of Manufacturing bit when BIOS Master access regions are set to Intel Recommended setting.

ng images with FW in

Note: Once an image is built with End of Manufacturing bit set, it cannot be unset or changed by decomposing an existing image in the tool. It is necessary to build a new image in order to have this bit unset.

\* See Intel FPT can set Manufacture Done bit in FW, so it switches itself to normal mode after restart Firmware Bring Up guide from Intel ME system tools

# ME Manufacture mode in the wild

---

26

- Gigabyte GA-Q87M-D2H motherboard
- Asus rampage iv extreme motherboard
- Apple Mac mini A1347 desktop computer
- Apple Macbook Pro 2015, mid 2015, 11.4, MJLQ2 notebook
- Lenovo Yoga 20CD thinkpad

1. Failure of DRAM Init Done (DID)
2. Via ME flash region update mechanisms
  - HDA\_SDO pin-strap
  - HMR FPO Enable command
3. Soft temporary disable
4. ME runtime disable
- 5. Disruption of ME access path to UMA**
6. Corruption of ME flash region

# Breaking ME access path to UMA

- ME accesses UMA by means of PCI-E Virtual Channel mechanism
- VCm virtual channel used by ME **can be disabled** in PCI-E Host bridge DMI BAR

## 4.1.15 DMIVCMRCTL—DMI VCm Resource Control

B/D/F/Type: 0/0/0/MEM			Access: RO; RW	
Size: 32	Default Value: 07000080h		Address Offset: 38h	
Bit Range	Acronym	Description	Default	Access
31	VCMEN	Virtual Channel enable: 0: Virtual Channel is disabled. 1: Virtual Channel is enabled.	0h	<u>RW</u>

Good news: after the channel is disabled, **ME freezes up completely**

Bad news: after ~40 sec **platform is powered off**

1. Failure of DRAM Init Done (DID)
2. Via ME flash region update mechanisms
  - HDA\_SDO pin-strap
  - HMR FPO Enable command
3. Soft temporary disable
4. ME runtime disable
5. Disruption of ME access path to UMA

## **6. Corruption of ME flash region**

# Corruption of ME flash region

30

ME flash region **protected by checksum and digital signatures**

Corruption leads to **ME Recovery State** initiated by ROM

In this state, **no FW module is loaded from flash** (AMT isn't functioning)

If you're lucky, this corruption might **burn** your CPU

After ~40 min in this state, ME performs **platform shutdown**



# ME is not working, really?

---

31

ME works in **two memory configurations**: SRAM only and SRAM+UMA

After DRAM Init Done, ME **always** switches to UMA mode

Statement: If ME is not working, **it doesn't access UMA being in UMA mode**

**DEMO**





- In modern platforms, ME can't be disabled by removing DDR modules from slots of channel 1
- ME can't be disabled by any PCH or CPU straps (as it was done in old platforms via ICH and MCH straps)
- Corruption of the Flash Descriptor signature (0x0FF0A55A, offset 16) doesn't allow SPI flash controller to work in the non-descriptor mode, thus effectively disabling ME. In all modern platform, this prevents PCH from starting up CPU, thus making the platform a complete brick

- There is no “silver bullet” to deactivate ME
- All disabling methods rely on the ME own mechanisms designed for platform vendors
- The methods described guarantee a DoS attack on the AMT technology in the area of remote management

# Thank you!

# Questions?

[www.ptsecurity.com](http://www.ptsecurity.com)

[blog.ptsecurity.com](http://blog.ptsecurity.com)

[@ptsecurity.com](mailto:@ptsecurity.com)

[github.com/ptresearch](https://github.com/ptresearch)

---

**POSITIVE TECHNOLOGIES**

[ptsecurity.com](http://ptsecurity.com)