Indholdsfortegnelse

UEFI – fra meget gamle til relativt nye standarder.

Der har været en tendens til at industri-PCer holder fast i de gamle standarder 5-10 år efter, de er forsvundet fra standard-PC markedet. Det var derfor meget praktisk, at 32-bit versionen af Gopa havde været under udrulning siden 2006 da det viste sig, at selv industri-PCer ved nullernes udgang ikke længere altid tillod dos-installationer med emm386.exe. Emm386.exe's evne til at finde ekstra ram inden for første Mbyte til brug for vores egen kode, havde været en afgørende forudsætning for 16-bit versionen, der havde været i drift siden 1996.

En af fordelene ved 32-bit versionen var - udover ikke at være afhængig af emm386.exe - , at man som alternativ til den gamle VGA 640x480 opløsning kunne bruge VESA 800x600. En vigtig forudsætning for den udbygning og effektivisering af fabrikkerne, der skete fra 2010 og fremadrettet, hvor vi fik kontrol over køreveje m.m. for bedre synkronisering af vægte med andet maskineri. Her var den lidt højere opløsning en forudsætning for rimelige operatørbilleder (SCADA).

Men der gik kun et år – til 2007 - før vi første gang mødte en standard-PC, der på bundkorts-grafikken ikke længere supporterede 800x600 16-farvers mode – men sjovt nok stadig den gamle 640x480.

Der gik ikke engang 5 år, før det samme begyndte at ske hos industri-PCerne. Det blev løst ved rutinemæssigt at installere et eksternt grafikkort, hvor 800x600 i 16 farver fortsat virkede, i et PCI-classic slot. Dette blev fast praksis allerede fra 2010 og fremefter – så kunne det være ligemeget om grafikken på CPU-kortet supporterede denne mode eller ej.

Det fungerede fint med disse industri-PCer, der kunne boote en gammeldaws DOS. I stedet for Windows-licensen, der fik lov at blive liggende på harddisken for at sikre noget genkendeligt til forhandleren, hvis PC-en skulle til garanti-reparation, installerede vi MS-DOS på en ny SSD disk. På den måde kunne vi overholde reglen om, at kun een Microsoft licens var i brug på samme tid. Jeg er dog ikke bekendt med, at det nogensinde skete. Industri-PCerne ventede mig bekendt pænt længe efter garantien var udløbet med at gå i stykker.

Grundet kundeønske om ethernet interface fik vi i 2015 support af netchippen RTL8139 (PCI-classic) fra Realtek. Fra sidst i 2015 kom der – sideløbende med kundeønsker om forskellige interfaces til feltbusser – nærmest konstant fokus på at få indhentet efterslæbet overfor kompatibilitet med nyere PC typer..

Vi vidste, at mange af de gamle standarder med rødder tilbage til IBMs PC-AT fra 1984, snart ville forsvinde. Vi ville til at begynde med trække på oplysninger fra Linux-miljøet, hvor vi opdagede, at man (via grub) havde tilpasset sig de nye vilkår for den indledende opstart af PC-er, nemlig UEFI.

UEFI var tilsyneladende noget, alle kunne blive enige om inden for PC-verdenen – både Linux og Windows. Det UEFI-projekt, eksempelvis Intel selv deltager i, hedder ”Tianocore”, og det dækker udover Intel og AMD flere processortyper med helt anden arkitektur, f.ex. ARM der kendes fra tablets og smartphones. Det blev også Tianocore, vi selv endte med at bruge.

Vi var begyndt at sætte os ind i disse ting fra starten af 2016, så vi ikke var helt uforberedte på, hvad der kunne ske fremadrettet – men blev dog alligevel overraskede, første gang vi stødte på ”SMI-emulering af ikke-eksisterende hardware”:

Fremtiden var begyndt i 2013, men vi opdagede det først i 2017.

I det tidlige efterår 2017 kom vi ud for noget mystisk. En PC-type, vi havde brugt siden 2013 og troede at kende ud og ind, viste en problemtype, vi ikke havde set siden 90-erne: At DMA operationer tilsyneladende forstyrrer interrupt-response tiden, så seriel-komm bliver forstyrret ved relativt lave baudrater. Den var god nok – kørsel med harddisken forstyrrede interruptet. Selv timerticks gik tabt.

Det var helt almindeligt i gamle dage f.ex. midt i 90-erne, at man i opsætning af netkort skulle fortælle, hvor høj en baudrate man gerne ville kunne bruge til f.ex. modem via serielport. Jo højere baudrate man ønskede, jo mindre blokstørrelse ville netkortet bruge til sine DMA transaktioner, og jo mindre effektiv ville net-kommunikationen blive. Til gengæld blev interrupt forsinket mindre, så man kunne køre hurtigere på serielporten uden risiko for tab af bytes. CPU-en og dermed interruptet kunne nemlig ikke komme til fadet, så længe en DMA-blok var i færd med at blive overført. I gamle dage ville cpuen altid sove tornerosesøvn, mens en DMA blok-transaktion var i gang.

Både CPU og BIOS hos den fejlramte PC var det samme som de mange tidligere eksemplarer af denne type PC. Noget der var testet i mange sammenhænge gennem 4 år.

Forskellen viste sig at ligge i BIOS setuppen. Ved de andre PC-er havde harddisk interface tilfældigvis stået til ”IDE” - ved den fejlramte PC stod det til ”AHCI”, der (modsat IDE) slet ikke kan kontaktes fra hverken Realmode eller Virtual 8086 mode (medmindre der bruges paging).

Det viste sig, at noget snavs – SMI kode - emulerede det gamle IDE I/O-baserede harddisk interface ved at overtage cpuen, hver gang programmet troede, at det tog fat i I/O-portene - midlertidigt switche den over i en anden mode (SMM mode) - køre noget kode, der får ny hardware (AHCI) til at spille rollen som gammel hardware - og tilbage igen.

Det konkrete problem kunne løses blot ved at skifte fra ”AHCI” til ”IDE” i BIOS setuppen, men det blev klart, at vores UEFI-løsning ikke kun skulle være et spørgsmål om stadig at kunne boote, når den gamle boot-sektor (MBR) i fremtiden ikke længere blev supporteret. Vi skulle også have direkte harddisk adgang gennem AHCI controlleren.

Hvis man uden at vide det henvender sig til ikke-eksisterende hardware, kan man risikere, at SMI-kode (System Management Interrupt) på samme måde som en virtuel PC emulerer (efterligner) funktionen ved hjælp af anden hardware. En virtuel PC vil også have svært ved at garantere reaktionstider i al slags vejr.

Og så kan der pletvis opstå store tidsforsinkelser (klumper i strømmen) også inde i interrupt-handlere, fordi SMI-koden misforstår situationer, og kan igangsætte trafikpropper på systembussen – det er særlig caching og (hvis i brug) paging, der kan blive slået ud af kurs af SMI-kodens akrobatik.

Slut med MS-DOS i HMA.

Kort efter opdagelsen af problemet med harddisk interface i sept. 17 prøvede vi at anskaffe en relativt ny standard PC – en minitower Dell Optiplex 7040. Her kunne man ikke vælge IDE, og – modsat tidligere - var det ikke længere muligt at installere DOS i den form, vi kunne bruge, d.v.s. i HMA.

Dog kunne man med ”Legacy boot” boote en dos, der så bare fyldte op i det lave ram-område.

HIMEM.SYS brød sig nemlig ikke længere om bios-sen, og blev hængende uden at komme videre. Og uden den ville dos ikke anbringe sig i området mellem 1MB og 1MB plus 64K.

Det kunne der rådes bod på med en simpel DOS-erstatning kaldet DSKLOA, der godt selv kan finde ud af, at lægge sig i HMA. Den blev lavet i starten af 2010-erne for at undgå licensproblemer, hvis man som standardpakke til fjernstyring skulle udsende programpakker til brug for kørsel i Virtual PC2007 emulatoren, der i modsætning til DosBox kun emulerer til bios-niveau. Det er et simpelt MBR-bootbart program, der leverer kommandolinie og de ganske få dos-funktioner, vores egne programmer bruger, d.v.s. fil-kald og start af exe-filer.

Første step, som vi kaldte for UEFI-1, var at få prøvekørt DSKLOA på Dell Optiplex 7040 i det virkelige liv. Selv grundige hjemme-tests havde ikke fejlet, så den havde tilsyneladende ikke samme slags problem som industri-PCen havde haft.

Den klarede i efteråret 2017 flere måneders test hos en mellemstor styring uden problemer, så vi troede, at vi her havde en UEFI-1 løsning, der kunne sikre reserve-PCer, indtil vi kom videre.

Men da vi over et år senere afprøvede den på den mest krævende app af alle (KOLD2), viste det sig, at den kunne fejle, m.h.t. reaktionstiden. Hyppigheden var lav, og viste sig først ved max belastning – og selv da med en hyppighed omkring 1 gang i døgnet – men den kunne altså tabe en byte ved en baudrate så lav som 9600. Og så er man meget, meget langt fra Las Vegas, hvis det skal forestille at være ”realtime”.

Brian Richardsons anbefalinger.

Vi havde haft køreplanen for UEFI-faserne på plads siden januar 2018. Betegnelserne UEFI-0 til UEFI-3 lånte vi fra et let satirisk skrift af en Intel-guru ved navn Brian Richardson, der stammer fra 2017. Dokumentets titel er ”Brian_Richardson_Intel_Final.pdf” og er nemt at finde på nettet (f.ex. på uefi.org). Han sætter tingene på spidsen, men den overordnede pointe er rigtig. Man får ikke fuld sikkerhed for stabilitet og god udnyttelse af cpu-tid, før man fuldt ud giver afkald på CSM, d.v.s. Compatibility Support Module. Dette er først opnået, når man er kommet til UEFI-3.

Kort fortalt er UEFI-0 den gamle IBM BIOS standard. Ved UEFI-1 er man begyndt at bruge nye standarder. Ved UEFI-2 er det kun få ting, der afhænger af CSM, og ved UEFI-3 er man helt i mål, d.v.s. kører uden CSM.

Oversat til vore behov, skulle vi gennem disse faser.

DSKLOA = Egen bootloader med primitiv kommandolinie til erstatning af DOS.
AHCI = SATAinterface harddisk til erstatning af det I/O baserede IDE interface.
XHCI = USB interface til mus og tastatur - erstatter PS2.
NY GRAFIK = Nyere grafikmode der ikke er VGA-kompatibel.
HPET/xAPIC = Nyt interrupt-sys og timertick hvor HPET/xAPIC erstatter 8254/8259.
NETKORT = Nyere netkort I-210 og to varianter af I-219 erstatter PCI classic kortet RTL8139. I alt 3 nye chips.

	DSKLOA	***AHCI***	***XHCI***	***NY GRAFIK***	***HPET/ xAPIC***	***NYE NETKORT***
UEFI-1	X					
UEFI-2	X	X	X			X
UEFI-3	X	X	X	X	X	X

Første punkt DSKLOA er strengt taget ikke med i Richardsons satire, og sidste punkt NETKORT var mere et kundeønske udløst af, at vores indtil da eneste ethernet-kort RTL8139 fra Realtek var PCI-classic, mens alt det nye jo er PCI-express.

Mini-tower PC-en Dell Optiplex 7040 havde tilfældigvis en I-219 på bundkortet, og desuden et enkelt PCI-classic slot vi kunne montere Realtek 8139 i. Vi udnyttede dette til at lave en ”Imponator” installation, hvor man alene med parameterændringer kunne skifte mellem 3 forskellige netchips: RTL8139 i PCI-classic slot, I-210 i PCI-express slot og endelig I-219 på bundkort. Den eneste af de 4 net-chips, der ikke var med i demoen, var I-219 V7.

Konceptet for de nye ting er i store træk det samme:

UEFI's 64-bit Open Source drivere starter busser og centrale chips op, hvorefter vi skifter ned til 32-bit mode, og arbejder videre med tingene på samme måde, som driverne gør, når chipsne først er kørende.
En sådan genvej havde været et no-go, hvis en gopa-PC skulle kunne gå i sleep-mode, fordi operatøren så og så længe ikke havde rørt tastatur eller mus. Men det har vi alligevel ikke brug for, da fabrikken så ville gå i stå. PC-ens strømforbrug ved at holde sig kørende er forsvindende sammenlignet med energibesparelsen fra den ekstra effektivitet, det giver fabrikken, at både PCer og

operatører altid er vågne.

Sidst i Februar 2021 var alle 4 punkter i UEFI-2 rækken klaret.

Lavpraktiske forskelle på legacy og UEFI.

Før UEFI (det vi fremadrettet kalder legacy-mode) var der kun een udgave af 32-bit kernen SYSLOAD.SYS, og der var kun to andre ting at tage stilling til:

1) Hvis det var 640x480 opløsning, hed grafikmodulet GOPA2.EXE. Var det 800x600 hed det GOPA3.EXE.
2) Skulle man køre på fysisk PC, skulle MB1.EXE kopieres fra MB1.PS2. Skulle man køre i DosBox emulator (udbredt emulator til retro-spil) skulle MB1.EXE kopieres fra MB1.USB.

Under UEFI-2 er antallet af moduler steget.

32-bit kernen findes nu i 3 versioner. Den oprindelige SYSLOAD.SYS hedder nu SYSLOAD.UE0 (UEFI-0 var Richardsons betegnelse for den gamle BIOS – før UEFI), og der er kommet to supplerende moduler SYSLOAD.PS2 og SYSLOAD.USB.

På 16-bit siden er der kommet to ekstra grafikmoduler, GOPA4.EXE og GOPA5.EXE.

1) Ved legacy-kørsel kopieres SYSLOAD.SYS fra SYSLOAD.UE0 og MB1.EXE varianten kopieres fortsat fra MB1.PS2 hvis fysisk PC og MB1.USB hvis DosBox.
2) Ved UEFI med PS2 kopieres SYSLOAD.SYS fra SYSLOAD.PS2 og MB1.EXE varianten kopieres fra MB1.PS2.
3) Ved UEFI med USB kopieres SYSLOAD.SYS fra SYSLOAD.USB og MB1.EXE varianten kopieres fra MB1.USB.

Ved legacy-versionen skal SYSLOAD.SYS (kopieret fra SYSLOAD.UE0) ligge i projektdir, hvor det uploades ved programstart. Ved de to UEFI versioner skal SYSLOAD.SYS (kopieret fra enten SYSLOAD.PS2 eller SYSLOAD.USB) ligge i C:\EFI\GOPA.

Legacy-versionen videreføres sideløbende med UEFI-versionerne primært af hensyn til en emulator som DosBox, der er praktisk til både fjernstyring og værktøjsprogrammer hos Windows-PC (testet til og med Windows-11).

Hvad angår de to nye grafikmoduler til UEFI bruges GOPA4.EXE til 640x480 (svarer til GOPA2.EXE under legacy) og GOPA5.EXE bruges til 800x600 (svarer til GOPA3.EXE under legacy).

At boote i UEFI-mode i stedet for legacy-mode.

Når PC-en i BIOS er sat til at starte op i UEFI-mode, og det ikke er på en Windows-partition (der

desuden normalt kræver tilvalg af ”secure boot”), vil UEFI starte op i noget, der kan minde om en gammeldags DOS kommandolinie. I BIOS-setuppen kan det være nødvendigt, at fravælge ”Legacy” i mere end een undermenu (typisk om ROM og Netkort), for at sikre et UEFI-boot.

I princippet kan der bootes fra ethvert device med et genkendeligt filsystem – f.ex. en USB stick.

I stedet for boot-sektor, som refererer til noget, der ligger under filsystemet, er det selve placeringen og filnavnene hos et genkendeligt filsystem, UEFI forholder sig til.

En 64-bit bootning af UEFI kræver blot, at man under roden i filsystemet på enheden har flg.

\EFI\BOOT\BOOTx64.EFI

BOOTx64.EFI er kommandolinie programmet. En slags krydsning mellem COMMAND.COM fra DOS-tiden og en luxuriøs debugger (med masser af hjælpetekster), hvor man fra kommandolinie kan udsætte chips for næsten alt, hvad CPU-en kan udsætte dem for.

Filer, der ender på .EFI, svarer til .exe filer kendt fra DOS/Windows. Der er tale om en lettere tilpasning af coff filformatet kendt fra Windows. Vi tager udgangspunkt i 64-bit UEFI, fordi 32-bit versionen kan risikere ikke at være ordentlig testet fra fabrikken. Det skyldes, at 64-bit kode har domineret mainstream-markedet i over 10 år. Og så er det jo også tanken, at resten af gopa skal over i 64 bit på et tidspunkt.

Exekverbare programmer i UEFI hedder altid .EFI til efternavn. Uanset om det er forgrunds-applikationer (som f.ex. BOOTx64.EFI), almindelige device-drivere (kaldes internt for DXE drivere) eller pre-boot device-drivere (kaldes internt for PXE – f.ex. til netkort, der skal kunne bootes fra).

Hvor .EFI filer svarer til .EXE filer, svarer .NSH filer til .BAT filer.

Hvis der i roden på det drev, der er bootet fra, befinder sig en fil ved navn STARTUP.NSH, spiller den samme rolle som AUTOEXEC.BAT gjorde hos MS-DOS. Den startes automatisk ved bootning (står dog og tæller i ca. 5 sekunder, hvor man kan forhindre starten ved at trykke ESC – eller hvis man er utålmodig, og gerne vil have startet med det samme, kan man trykke enhver anden tast end ESC).

Vores udgave af STARTUP.NSH indeholder (i udgangspunktet) ikke andet end:

```
fs0:
cd \efi\gopa
```

Første linie (fs0:) er valg af det drev, der er bootet fra (svarer til at skrive C: <ENTER> - noget der ikke var nødvendigt under DOS). Anden linie virker helt som hvis det havde været Dos/Win. Aktuelt drev-dir bliver til fs0:\efi\gopa

Dette var UEFI-bootningen.

Og det er her i fs0:\EFI\GOPA, vi finder hovedparten af de .EFI og .NSH filer, vi bruger overfor UEFI.

Opstartsforløbet i korte træk.

Når UEFI-boot er slut, og vi står i:

fs0:\EFI\GOPA

vil vi klargøre systemet på UEFI-siden med forskellige .NSH og .EFI filer, herunder start af eventuelle USB-drivere - at loade SYSLOAD.SYS ind i hukommelsen, skifte til den rigtige videomode (indtil videre med CSM support), skifte fra 64 til 32-bit mode og hoppe til entrypunkt i SYSLOAD.SYS.

Man kan nærmest sige, at vi ”hopper ned til 32-bit ovenfra” nemlig fra 64-bit mode, hvor legacy-versionen ”klatrede op til 32-bit nedefra” med afsæt i 16-bit realmode.

SYSLOAD.SYS etablerer herefter vores egen BIOS-support, og slutter (i første omgang) med at finde og loade to bootsektorer ind i ram. Dels MBR (hvor vi kun udnytter partitions-info) dels den rigtige bootsektor (vi selv har lavet) til den aktive partition (C-drevet). Sidste step er at starte en Virtual 8086 task uden paging og hoppe til det sted i ram, hvor den rigtige bootsektor befinder sig.

Når denne efterfølgende har bootet DSKLOA, vil det dir der under UEFI hed:

fs0:\EFI\GOPA

komme til at hedde:

C:\EFI\GOPA

Den enhed, vi har bootet fra, vil være en FAT16 formatteret SATA-disk (typisk SSD disk) med en MBR sektor.

Vores brug af UEFI's kommandolinie.

Mange kommandoer er de samme, f.ex. DIR, COPY m.m. - men der er også forskelle. F.ex. er der ikke nogen RENAME. I virkeligheden stammer UEFI fra Intels egen EFI (Extensible Firmware Interface).

Trods protester fra Microsoft begyndte Intel at dyrke Linux fra slutningen af 90-erne og frem. Den oprindelige EFI-syntax er derfor Linux, hvor f.ex. LS svarer til DIR, CP svarer til COPY o.s.v. Man har så senere indført en alias-tabel (brugeren endda kan udvide), som oversætter DOS/Windows kommandoer til de tilsvarende Linux (Unix) kommandoer. Man kan derfor bruge begge syntaxer.

Hvis der er brug for tekst-editor, kan man bare skrive ”edit filnavn”. Så kommer der en text-editor, der eksempelvis kan bruges til redigering af .NSH filer.

Såvel denne text-editor som hele kommandolinie-niveauet er brugervenligt overfor gamle traditioner – f.ex. 8-bit ascii. Skønt hele det underliggende API til .EFI programmer bruger 16-bit textstrenge internt, påtager UEFI-kommandolinien sig nødvendige konverteringer. Så 8-bit ascii-filer indlæst fra anden PC virker fint.

Folk, der er vant til at arbejde med f.ex. .INI filer i Windows-miljø, burde have let ved at arbejde i UEFI-miljø. De skal bare være hardcore nok til ikke at besvime ved synet af hexadecimale tal.

Oversigt over EFI og NSH filer i C:\efi\gopa

Vi har 4 EFI-filer, d.v.s. exekverbare programmer (a.la. .EXE filer) og 6 NSH-filer, d.v.s.

kommandofiler (a.la. .BAT filer). Ud af de 10 filer er nummer 1 til 5 beregnet til at blive startet af NSH-filer, mens 6 til 10 er brugerens kommandoer, hvor det er nok at skrive fornavnet på filen og trykke ENTER. De sidste to bruger-kommandoer, d.v.s. 9 og 10 starter 32-bit kernen op i enten 640x480 eller 800x600 mode, mens de øvrige 3 er config-kommandoer, man typisk kun bruger ved klargøring af ny PC, eller hvis der tilføres eller fjernes kort.

1) SYSLOAD.EFI: Loader 32-bit programfilen SYSLOAD.SYS ind i ram, switcher fra 64 til 32-bit mode og hopper til entry-punktet i SYSLOAD.SYS. Under normale omstændigheder foretager 32-bit SYSLOAD.SYS sig ikke andet end at hejse et flag for at huske, at den har været kaldt for første gang. Derefter returneres tilbage til 64-bit mode via SYSLOAD.EFI, og derfra tilbage til UEFI kommandolinien. Faciliteten er brugt til debug-formål, fordi vi fra vores eget 32-bit miljø har kunnet udforske, hvordan UEFI-miljøet ser ud, før CSM-support første gang startes op, og sammenholde det med hvordan UEFI-miljøet ser ud, når vi bliver kaldt anden og sidste gang, EFTER at CSM supporten (som kan have sin egen SMM kode) er startet op.
2) SYSCALL.EFI: Laver næsten det samme som SYSLOAD.EFI. Eneste forskel er, at den ikke starter med at loade 32-bit modulet SYSLOAD.SYS ind i ram. Dette formodes at være på plads, inden der kaldes, da der ellers hoppes ud i den blå luft (crash). Når dette andet kald til 32-bit SYSLOAD.SYS finder sted, startes 32-bit kernen op, og vender herefter ikke tilbage. På samme måde som ved boot af et traditionelt operativsystem – det vender heller ikke tilbage til boot-bios, men tager sig selv af at lukke og slukke.
3) VGA12ON.EFI: Tænder for CSM-supporten og skifter til den gamle VGA-mode hex 12 (640x480 16 farver) ved hjælp af den gamle int 0x10 video bios standard. Kun få gopa-apps kører i denne gamle mode, men en del små service-programmer til test af dit og dat og config-programmer bruger den stadig. F.ex RT16CFG.EXE.
4) VGAON.EFI: Tænder for CSM-supporten og skifter til SVGA (VESA) 800x600 i 16 farver ved hjælp af den gamle int 0x10 vesa bios standard.. Langt de fleste gopa-apps kører i denne mode.
5) TJEKUSB.NSH: Kommandofil der udskifter 4 USB-drivere med vores kendte versioner fra Tianocore EDK2. Skal kun køres, når tastatur og mus er USB.
6) SETUSB.NSH: De filer der er vist indtil nu, er ikke beregnede til at skulle aktiveres direkte af brugeren. SETUSB.NSH er den første, som brugeren kan aktivere ved at skrive fornavnet SETUSB og trykke ENTER. Der er tale om en indstillings-kommando, der kun skal bruges, når man vil skifte fra PS2 til USB. Indstillingen huskes fra boot til boot, indtil man bruger den modsatte kommando ved at aktivere SETPS2.NSH. Ved brug af USB keyboard og mus må der ikke sidde andet i USB-portene end netop keyboard og mus, og det skal være USB 3.0 porte (blå farve), fordi vi ikke supporterer andre slags USB-host end XHCI controlleren.
7) SETPS2.NSH: Også en kommando beregnet til at skulle aktiveres af brugeren ved at skrive SETPS2 <ENTER>. Den virker modsat SETUSB, d.v.s. flytter indstillingen fra USB tilbage til PS2. Har man glemt at skifte indstilling efter rent fysisk at have skiftet fra USB til PS2 (SETPS2), vil opstart af 32-bit kerne gå i stå, mens der skrives ”Searching for USB Keyboard”. Hvis man har fejlet den modsatte vej, d.v.s. skiftet fra PS2 til USB uden at aktivere SETUSB, gennemføres opstart af kommandolinie fint, men hverken keyboard eller mus virker efter ankomst. Charmen ved begge fejlsituationer er, at det ikke nytter at trykke ctrl+alt+del. Kun pwr-off eller reset virker.
8) FINDXHC.NSH: Finder handle-nummeret på XHCI driveren, og skriver det på skærmen. Det er et hex-tal, der skal indtastes to steder i skabelonfilen TJEKUSB.TXT, første gang en PC

klargøres, og hvis der ændres på hardware, f.ex. isættes eller fjernes kort.

9) EXV.NSH: Den første af de to mulige start-kommandoer. Starter gopas kommandolinie op i den gamle Vga mode på 640x480. Denne mode bruges kun af få gopa-apps, men af en hel del små konsol-baserede testprogrammer.

10) EXS.NSH: Den anden af de to start-kommandoer. Starter gopas kommandolinie op i VESA 800x600 mode, som de fleste gopa-apps bruger.

Der findes også .EFI filer i et subdir ved navn USB (fs0:\efi\gopa\usb) – de beskrives nærmere under TJEKUSB.NSH i afsnittet ”Indhold af de to config filer SETUSB.NSH og SETPS2.NSH”.

Fuldautomatisk opstart.

Før i tiden var det praksis, at en DOS-PC fik lov at boote op uden at starte programmet. Den stillede sig takket være AUTOEXEC.BAT automatisk i projektdir, så man bare skulle skrive EX <ENTER> for at starte gopa-appen. Det var en batch-fil ved navn EX.BAT, der løste opgaven.

Den tilsvarende praksis under UEFI er at lade STARTUP.NSH slutte med at starte den relevante af de to start-kommandoer, d.v.s. enten EXV.NSH (640x480) eller EXS.NSH (800x600).

Skulle der så være opstået et problem, der gør det uhensigtsmæssigt at der autostartes, kan man udnytte muligheden for at trykke ESC inden for de første 5 sekunder ved boot. Man skal bare lige huske efterfølgende at vælge boot-drev som aktuelt drev, hvis man har forhindret STARTUPNSH i at starte. Man skal bare manuelt skrive fs0: <ENTER>.

Hvis det f.ex. er en 800x600 app, kan STARTUP.NSH se sådan ud:

```
fs0:
cd \efi\gopa
exs.nsh
```

Indhold af de to opstarts-filer EXV.NSH og EXS.NSH.

EXV.NSH ser sådan ud:

```
TJEKUSB.NSH
SYSLOAD.EFI
VGA12ON.EFI
SYSCALL.EFI
```

TJEKUSB.NSH vil - hvis filen findes - installere vore USB-drivere, der er lavet med Tianocore EDK2. Hvis filen ikke findes, går vi bare videre til næste linie.

SYSLOAD.EFI loader 32-bit programmet SYSLOAD.SYS ind på sin faste adresse i ram, og kalder

det første gang for at give mulighed for at undersøge UEFI-miljøet, før CSM startes op.

VGA12ON.EFI tænder CSM-supporten og skifter videomode til 640x480 ved hjælp af det gamle int10 videobios interface.

SYSCALL.EFI kalder 32-bit programmet SYSLOAD.SYS for anden og sidste gang. Kaldet vender ikke tilbage, men medfører opstart af gopas kommandolinie (DSKLOA), der spiller samme rolle som MS-DOS gjorde før i tiden.

EXS.NSH er mage til EXV.NSH bortset fra at VGAON.EFI kaldes i stedet for VGA12ON.EFI. Den eneste forskel er, at videomoden bliver 800x600 i stedet for 640x480.

Indhold af de to config filer SETUSB.NSH og SETPS2.NSH.

SETUSB.NSH:

COPY TJEKUSB.TXT TJEKUSB.NSH

COPY SYSLOAD.USB SYSLOAD.SYS

SETPS2.NSH:

DEL TJEKUSB.NSH

COPY SYSLOAD.PS2 SYSLOAD.SYS

Fidusen er her, at man dels enten opretter (ved at kopiere fra TJEKUSB.TXT) eller sletter TJEKUSB.NSH - dels kopierer den rigtige version af 32-bit kernen – USB eller PS2 – til det fælles filnavn SYSLOAD.SYS.

TJEKUSB.NSH kaldes nemlig ubetinget af begge start-filer EXV.NSH og EXS.NSH. Findes den ikke, sker der ikke noget ved det, og findes den, får vi installeret kendte UEFI drivere, der er lavet med EDK2 byggesættet.

Så det er egentlig TJEKUSB.TXT vi skal kigge nærmere på, fordi det er den TJEKUSB.NSH dannes ud fra.

TJEKUSB.TXT:

CD USB

LOAD *.EFI

DISCONNECT 16f

CONNECT 16f

CD ..

Der er tale om at vi går ned i et subdir (USB), der indeholder 4 EFI-filer som alle bliver loaded ind i ram (LOAD *.EFI). Der er tale om 4 USB-drivere fra EDK2:

1) XhciDxe.efi - XHCI chipdriver.
2) UsbBusDxe - USB bus.
3) UsbKbDxe - USB keyboard.
4) UsbMouseDxe - mus.

LOAD *.EFI henter driverne ind i ram, og stiller dem i venteposition.

DISCONNECT 16f frakobler både XHCI-driveren og indirekte de øvrige 3 drivere, fordi bus-driveren er registreret som afhængig af XHCI-driveren, og de to sidste er registreret som afhængige af bus-driveren.

CONNECT 16f genindkobler XHCI-driver, d.v.s. den nye version, der blev hentet ind med LOAD. Herefter genindkobles automatisk nye versioner af de øvrige 3 i takt med, at det bliver muligt (fordi de drivere, de selv er afhængige af, melder at de er indkoblede og klar til brug).

16f skal med 99.99% sikkerhed erstattes af et andet hexadecimalt tal.

Manuel klargøring af USB-support - find handle-nummer.

16f er et tilfældigt eksempel på ”handle” nummeret for XHCI. Det vil næsten med sikkerhed være noget andet, fordi handle-numre tildeles af UEFI ved opstart. De er normalt de samme fra boot til boot, men vil typisk ændre sig, hvis der isættes eller fjernes kort. Det er del af klargøring af PC at finde det aktuelle handle-nummer. Der findes en kommandofil, der hedder FINDXHC.NSH og som indeholder en UEFI-kommando, der prøver at finde handle-oplysninger på alle aktive XHCI-drivere og skrive dem på skærmen. Den skal helst kun finde een af slagsen, og handle-nummeret er det hexadecimale tal til venstre i linien.

Dette tal skal manuelt med edit indsættes to steder i TJEKUSB.TXT filen i stedet for 16f. En efterfølgende SETUSB kommando vil så kopiere videre til TJEKUSB.NSH.

Kommandoen i FINDXHC.NSH er ret kortfattet:

dh -b -p UsbHc2

Det er indtil videre en lidt klodset tilgang som - når der indløber mere praktisk erfaring - bliver automatiseret - formentlig af en EFI-fil.

UEFI-2 faserne.

Første punkt: AHCI harddisk support blev klarmeldt 5-11-19.

Næste klarmelding var NETKORT d. 20-02-20, hvor både PCI-express kortet (Gigabit ethernet) I-210 fra Intel plus I-219 (integreret på bundkort) kom til at fungere. Resten af 2020 var lidt underdrejet på grund af Corona, bl.a. fordi udrulning af fjernsupport kom i fokus i denne periode.

Sidst på året kom endnu en I-219 variant med (Ver 7) og sidste punkt XHCI, d.v.s. USB keyboard og mus i stedet for PS2, blev klarmeldt d. 26-02-21.

Første installation i det virkelige liv af en UEFI-2 løsning skete i Januar 21 på en industri-PC, som egentlig godt kunne have kørt legacy (for at have en hurtig retrætesti), men det blev ikke nødvendigt.

Det var med gammeldags PS2. Det nye var, at det var et UEFI-boot, og at vi selv kørte harddisken via AHCI og DSKLOA. Men tanken var jo også, at kunne hamstre billige mini-tower PCer og bruge dem som reserve-PCer.

Kontor-PCer i industri-miljø.

I sommeren 2021 fik vi lov at testkøre en klassisk minitower-PC på en foderfabrik, hvor formålet b.la. var at få støj-testet det nye ICP-CON interface til relæboxe og analogmoduler. Den klarede sig fint i ugevis uden at tabe et eneste telegram. Men da vi et halvt år senere skulle teste den rigtigt, tog den pludselig el-støj ind til den store guldmedalje (forstyrret seriel-komm). På et tidspunkt kom elektrikeren med en sjov udmelding: ”Jamen der er jo ingen jordforbindelse”. Ganske rigtigt – det 3-die ben i strøm-stikket havde ingen forbindelse til PC-ens spinkle chassis. En gul/grøn ledning med kabelsko spændt i en af chassis-skruerne løste problemet. Grunden til det var gået godt i flere uger var formentlig, at PC-en havde stået på et skrivebord 4-5 meter fra en stor gammel VLT. Da den kom tættere på dette støjmonster (1.5 – 2 meter) gik støjen i kablerne. Men kun indtil chassis blev jordforbundet.

Den digitale kanariefugl.

Kanariefugle blev taget med ned i kulminer for at sikre luftkvaliteten. Længe før luften blev for dårlig til mennesker, ville kanariefuglen falde ned fra pinden, og derved advare minearbejderne om, at det var på tide at komme op fra minen.

I vores tilfælde når en UEFI-2 PC skal undersøges ift. hvor store interrupt-forsinkelser, man kan komme ud for worst case (b.la grundet drillerier fra SMI-kode), består kanariefuglen af et sub-D 9-polet hunstik, hvor ben 2 og 3 er loddet sammen. Bytes der udsendes fra serielporten ekkoes tilbage, fordi loddeklatten forbinder TX med RX.

Et test-program skal så prøve – interruptstyret – at sende bytes og læse dem tilbage igen. Det er vigtigt ikke at enable hardware-fifo hos serielporten.

På den måde opnår vi, at hvis PC-en er længere tid om at reagere på rx-interruptet fra den gamle byte, end det tager at modtage næste byte, vil den nye byte kunne nå at overskrive den gamle, før den gamle er hentet.

Forskellige PC-typer blev testet i resten af tiden frem til 1-04-22, og konklusionen var for alle typer, at hvis den kan klare 57.6 KB uden at tabe en char, når der køres maximalt med disken (DMA belastning), ville det være en fuldgod reserve-PC.

Timerticket på 2324 Hz er gopas ”hovedmotor”, og det drives af en firkant-bølge, d.v.s. signalet er kun tændt halvdelen af periodetiden. Periodetiden ved 2324 hz er 430 us og halvdelen er 215 us.

Her er en bestået 57.6 KB test garanti for, at ingen interrupt-forsinkelse har været større end 173.6 us, og der er rimelig sikkerhedsmargin til timerens pulsbredde på 215 us.

Ved en bestået 38.4 KB test er der sikkerhed for, at ingen interrupt-forsinkelse har været større end

260 us, hvad der netop ikke rækker (ift. 215). Men en PC-type, der fejler 57.6 men består 38.4, har vist sig ikke at få PC-ens sw-ur til at gå forkert, så måske er timerpulsen ikke helt firkantet. Måske er den bredere i ON stillingen (den der udløser interrupt).

Denne PC-type kan formentlig bruges de fleste steder bortset fra de mest krævende.