Open Core Debug

Beitrag von „praekon“ vom 15. Oktober 2019, 19:01

Moin,

ich versuche mich gerade an Open Core und habe so meine liebe Mühe damit.
Was am hinderlichsten ist, ist das ich es nicht schaffe den Apfel Ladescreen los zu werden und zu sehen was da nicht stimmt.

Wenn ich es richtig verstehe brauche ich Open Core mit DEBUG anstatt Release. Ok, habe ich mit OC_EFI_Maker erstellt (0.52).
Dann sind Werte in der config.plist nötig, wenn ich die PDF richtig verstehe.
Unter Misc - Debug. Aber irgendwie haut das mit den Werten nicht hin. Kann ich Data als Typ verwenden oder muss ich Number nehmen?
Hat da jemand mal ein Beispiel wie das aussehen sollte, bitte?

Beitrag von „karacho“ vom 15. Oktober 2019, 19:15

Unter nvram add bei boot-args -v eintragen, für eine Log Datei brauchst du die Debug Version und angepasste Einstellungen unter misc debug. Lies dir dazu die PDF nochmals durch. Bin gerade am Handy und kann nicht nachschauen.

Beitrag von „praekon“ vom 15. Oktober 2019, 19:24

Danke!
Ich war mir mit -v nicht sicher, weil ich das bisher nur CLOVER kannte.
Im PDF steht dazu folgendes:

4. Target
   Type: `plist integer`
   Failsafe: 0
   Description: A bitmask (sum) of enabled logging targets. By default all the logging output is hidden, so this option is required to be set when debugging is necessary.

   The following logging targets are supported:

   - 0x01 (bit 0) — Enable logging, otherwise all log is discarded.
   - 0x02 (bit 1) — Enable basic console (onscreen) logging.
   - 0x04 (bit 2) — Enable logging to Data Hub.
   - 0x08 (bit 3) — Enable serial port logging.
   - 0x10 (bit 4) — Enable UEFI variable logging.
   - 0x20 (bit 5) — Enable non-volatile UEFI variable logging.
   - 0x40 (bit 6) — Enable logging to file.

   Console logging prints less than all the other variants. Depending on the build type (`RELEASE`, `DEBUG`, or `NOOPT`) different amount of logging may be read (from least to most).

   Data Hub log will not log kernel and kext patches. To obtain Data Hub log use the following command in macOS:

Was nach meinem Verständnis nach, bedeuten würde bei Target 0x02 würde ne Screenausgabe bekommen.

Scheint wohl nicht zu stimmen, oder ich bin Begriffsstutzig 

Beitrag von „mhaeuser“ vom 15. Oktober 2019, 19:26

praekon Lies dir doch Mal die Bedeutung des ersten Bits (Bit 0) durch

---

Beitrag von „praekon“ vom 15. Oktober 2019, 19:29

Hast Recht, stand auf der Leitung. 

Ist ungefähr so wie bei den Rechten auf *NIX Systemen, ich Depp.
Danke!

Beitrag von „karacho“ vom 15. Oktober 2019, 19:31

Der Wert bei Target muss 0x40 sein, damit Opencore ein Textfile auf der EFI Partition erstellt. 0x40 musst du aber umrechnen nach Integer. Nimm den plist Editor von xcode, trage dort 0x40 ein und ändere dann von Data auf Integer. Xcode rechnet das dann automatisch um. Oder den Taschenrechner von MacOS im programmiermodus. Da klickst du oben rechts auf die kleine 16, gibst dort 40 ein und klickst dann auf die kleine 10.

Beitrag von „praekon“ vom 15. Oktober 2019, 20:20

Danke, nochmal.

Ich nehme meist nen Konverter im Netz. Hex zu Binary / Decimal etc. Die kenne ich durch das Signaldecoding von 433mhz Steckdosen
Wenn jemand also Tipps haben möchte wie er/sie Baumarktfunksteckdosen per Siri steuern kann, kann ich vielleicht etwas zurückgeben.
Solange, müsst Ihr meine Deppenfragen ertragen