

Häufig gestellte Fragen zur SolidCard II und Linux

Wenn ich das System auf eine serielle Konsole umstelle, kommen die Zeichen bis zum Starten des Init-Prozesses flüssig. Danach immer 16 auf einmal, dann eine längere Pause, dann wieder 16 Zeichen. Warum?

Wenn eine serielle Standard-Schnittstelle für eine Konsole verwendet wird, darf deren Interrupt nicht von einer zweiten Schnittstelle verwendet werden. Wenn also ttyS0 und ttyS2 sich den IRQ4 teilen (Stichwort shared interrupt), kann keine serielle Konsole realisiert werden.

Die Interrupt-Kanäle werden vom HyperBoot-Lader fest vergeben. Sprechen Sie mit uns, wenn Sie eine spezielle Verteilung der Interrupt-Kanäle benötigen.

Meine seriellen Schnittstellen funktionieren nicht zufriedenstellend bzw. wie gewohnt.

Die vier seriellen Schnittstellen der SolidCard II teilen sich paarweise einen Interrupt-Kanal. ttyS0 und ttyS2 verwenden Interrupt-Kanal 4, ttyS1 und ttyS3 verwenden den Interrupt-Kanal 4. Da das gemeinsame Verwenden von Interrupt-Kanälen ursprünglich für ISA-Bus-Geräte nicht vorgesehen war, sich bei den seriellen Schnittstellen jedoch eingebürgert hat, müssen Sie diese Möglichkeit beim Erstellen des Kernels explizit angeben. Wählen Sie dazu unter dem Menüpunkt „Character devices“, „Extended dumb serial driver options“ die Auswahl „Support for sharing serial interrupts“ aus, auch wenn Sie nicht vorhaben alle 4 Schnittstellen zu verwenden. Beachten Sie auch den Punkt 4!

Ich benutze nur ttyS0 und ttyS1. Sämtliche Kommunikation erfolgt aber nur mit einer erheblichen Zeitverzögerung bzw. unter hohem Datenverlust. Warum?

Normalerweise ist bei einem Interrupt-Sharing auf dem ISA-Bus eine „Wired-AND“-Verbindung gegeben. D.h. zwei Ausgänge wirken beide auf die gleiche Leitung und bestimmen zusammen den Pegel auf eben dieser Leitung. Wird nur eine der beiden Quellen tatsächlich aktiviert, ist die zweite hochohmig, stört also die erste Quelle nicht. Auf dem SolidCard II-System liegt aber eine echte ver-Oderung der Interruptsignale der beiden Schnittstellen vor. Wird nun ttyS2 oder ttyS3 nicht aktiviert, sind ihre Interrupt-Ausgänge hochohmig. Nach der ISA-Norm wird die Interruptleitung in einem solchen Fall mittels PullUp-Widerstand auf den hohen Ruhepegel gezwungen. Durch die ver-Oderung des internen Interruptkanals der ttyS0 und ttyS1 mit dem jeweiligen externen Interruptkanälen blockieren nun die externen „1“-Pegel jeden Interrupt an den Interrupt-Controller. Dies ist aber nur der Fall, wenn Sie die internen seriellen Schnittstellen (ttyS0 und ttyS1) verwenden wollen, die externen (ttyS2 und ttyS3) jedoch nicht. Wenn Sie alle verwenden tritt dieses Problem

nicht auf, ebenfalls nicht, wenn Sie nur die beiden externen verwenden. Als weitere Abhilfe haben Sie noch die Initialisierung aller 4 Schnittstellen, als wollten Sie alle verwenden, werten aber nur die Daten der ersten beiden tatsächlich aus. In diesem Fall aktiviert der Treiber alle Interrupt-Ausgänge der seriellen Schnittstellen und stellt darüber sicher, daß alle eine Interruptanforderung an die CPU senden können.

Der letzte Parameter einer Kernel-Kommandozeile, die ich dem mImage-Programm zum erstellen eine HyperBoot-Abbildes übergebe, wird vom startenden Kernel nicht bzw. fehlerhaft ausgewertet. Warum?

Die übergebene Zeichenkette wird von mImage mit einer 0 abgeschlossen (wie in 'C' üblich). Dadurch erkennen aber die Auswerteroutinen das letzte Argument nicht mehr richtig. Lösung: Hängen sie an das letzte Argument ein Leerzeichen (in der Textdatei). Dann wird auch das letzte Argument richtig erkannt bzw. verarbeitet.

Ich bekomme RTAI auf der SolidCard II nicht zum Laufen. Die Prozessorgeschwindigkeit wird nicht richtig ermittelt und es bricht mit einer Fehlermeldung ab.

RTAI verwendet normalerweise den TSC (Time-Stamp-Counter) der CPU. Dieser ist erst ab der Pentium®-CPU implementiert. Die SolidCard II hat aber eine 486er CPU. Modifizieren Sie im Kernel-Verzeichnisbaum die Datei include/asm-i386/rtai.h:

Originalzeile	Modifizierte Zeile	Änderung
#define <code>FREQ_8254</code> 1193180	#define <code>FREQ_8254</code> 1189200	Frequenz geändert
##define <code>EMULATE_TSC</code>	#define <code>EMULATE_TSC</code>	Nicht mehr auskommentiert

Damit wird die Verwendung des TSC unterbunden. Leider geht damit ein Performanceverlust einher, der aus der Verwendung des Timer 2 des Standard-8254 statt des TSC resultiert. Lesen Sie dazu im Handbuch zu RTAI das Kapitel Appendix A, An overview of RTAI schedulers.

Ich verwende RTAI. Mitten in der laufenden Applikation führt das System plötzlich einen Neustart durch. Woran liegt das?

Dabei handelt es sich um einen Bug der RTAI-Version, die mit der 2.0 ElinOS CD-Version ausgeliefert wird. Laden Sie aus: <http://ftp.sysgo.de/pub/elinos/2.0/updates/> die bereinigte RTAI-Version.

Ich habe meinen Kernel selbst zusammengestellt. Wenn sich dieser Kernel versucht zu initialisieren, bleibt er dabei hängen oder führt einen Neustart

Häufig gestellte Fragen zur SolidCard II und Linux

durch. Warum?

Auf der SolidCard I und II kommt ein PS/2 Tastatur- und Mauscontroller zum Einsatz. Aus diesem Grund muß auch eine PS/2-Mausunterstützung im Kernel freigeschaltet werden. Danach sollte der Kernel normal starten.

Ich kann kein Diskettenlaufwerk in meinem SolidCard-System verwenden. Der Treiber meldet einen ungültigen Laufwerkstypen im CMOS-RAM.

Der HyperBoot-Lader verwendet das CMOS-RAM nicht. Es werden also keine Informationen über ein angeschlossenes Diskettenlaufwerk hinterlegt. Verwenden Sie stattdessen den Kernelparameter `floppy=`. Ergänzen Sie beispielsweise:

```
floppy=0,4,cmos floppy=1,16,cmos
```

Damit legen Sie den Typen des ersten Laufwerks auf ein 3,5 Zoll, 1,4MB Laufwerk fest und das zweite Laufwerk erklären Sie damit für ungültig. Weitere Parameter bzw. nähere Erläuterungen finden Sie beispielsweise im `kernel-paramHowTo.pdf` oder in der Datei `README.fd` im Linux-Dateibaum unter `linux/drivers/block`.

Wenn der IDE-Treiber beim Hochlaufen des Kernels startet, erzeugt dieser eine Fehlermeldung, daß die Werte im CMOS ungültig seien. Hat dieser Fehler Auswirkungen?

Der HyperBoot-Lader verwendet das CMOS-RAM nicht. Daher enthält es keine Daten, wie bei einem normalen PC-BIOS. Da der IDE-Treiber jedoch selbst die Festplatte nach ihrer Geometrie abfragt, hat das Fehlen dieser Daten keine Auswirkung auf deren Funktion.

Da ich nur ein IDE-Gerät angeschlossen habe, sucht der IDE-Treiber beim Hochfahren des Systems sehr lange nach weiteren angeschlossenen Laufwerken. Kann man das verkürzen?

Ja. Über Kernelparameter können Sie den Treiber darüber informieren, welche Laufwerke tatsächlich vorhanden sind.

Mit:

- `hdx=none` (mit `x = a` oder `b`) teilen Sie dem Treiber mit, daß dieses Laufwerk nicht vorhanden ist.
- `idey=noprobe` (mit `y` von 0..2) teilen Sie dem Treiber mit, daß dieser ganze IDE-Port nicht vorhanden ist (und damit auch keine weiteren Laufwerke).

Für das SolidCard-Kit mit nur einem angeschlossenen IDE-Master empfehlen sich folgende Parameter:

```
hdb=none ide1=noprobe ide2=noprobe
```

`ide1` und `ide2` sind physikalisch auf dem SolidCard-Kit nicht vorhanden.

Ich möchte ein laufendes System in den Wartungsmodus des HyperBoot-Laders bringen. Dazu habe ich den dafür vorgesehenen DIP-Switch umgeschaltet und das System neu starten lassen. Jedoch bootet der HyperBoot-Lader das System sofort wieder neu, ohne die neue Schalterstellung zu berücksichtigen. Warum?

Die Schalterstellungen werden nur bei einem „harten“ Reset in ein prozessor-internes Register übernommen. Sollen also veränderte Schalterstellungen erkannt werden, genügt es nicht einen „Soft-Reset“ durchzuführen, sondern halten Sie das System an und lösen anschließend einen „harten“ Reset aus.

Die vom Kernel geführte Uhrzeit geht auf meinem SolidCard II-System nach. Woran liegt das?

Im SolidCard II-System werden alle benötigten Takte für die verschiedenen Peripherie-Geräte über eine PLL aus einem 33MHz-Grundtakt erzeugt. Dabei kommt es zu einer Abweichung vom Standard-PC Takt des Timers 0. Statt der üblichen 1,19318 MHz liegen am Timer 0 nur 1,1892 MHz an. Er läuft somit langsamer und damit die intern geführte Software-Uhr ebenfalls. Abhilfe schafft hier die Modifikation eines Werts in der Datei `include/asm/timex.h` (im Kernel-Dateibaum). Ändern Sie dort den Standardwert

```
#define CLOCK_TICK_RATE 1193180
```

in den neuen Wert

```
#define CLOCK_TICK_RATE 1189200.
```

Übersetzen Sie den Kernel neu. Jetzt läuft die Software-Uhr wieder synchron zur echten Zeit.

Ich habe in meinem SolidCard II System keine Grafikkarte. Kann ich dann den Speicher zwischen `0xA0000` bis `0xFFFFF` als normalen Systemspeicher verwenden?

Findet der HyperBoot-Lader keinen Grafik-Chip oder keine Video-BIOS Extension, wird der Bereich zwischen `0xA0000` bis `0xFDFFF` als normaler System-speicher nutzbar. Ab `0xFE000` bis `0xFFFFF` liegt eine schreibgeschützte Region. Diese enthält den Boot-Code.

Normalerweise nimmt der startende 2.2.x-Kernel nur den Speicher bis `0x9FFFF` als nutzbar an. Wenn Sie aber den Kernelparameter `endbase=` angeben, können Sie die Grenze nach oben verlegen (oder auch nach unten). Mit `endbase=0xFDFFF` können Sie 376kB zusätzlichen Speicher in einem SolidCard II System ohne VGA-Grafik nutzbar machen. Beachten

Häufig gestellte Fragen zur SolidCard II und Linux

Sie aber, daß ein derartiger Parameter bei einem System mit benutzter VGA-Grafik unweigerlich zum Absturz führt. Ebenfalls kann es zu Problemen kommen, wenn Treiber stillschweigend davon ausgehen in dieser Region ihre Hardware vorzufinden.

Ob Ihre Konstellation mit diesem erweiterten Basisspeicher funktioniert, muß also im einzelnen getestet werden.

Ich möchte den HyperBoot-Lader meines SolidCard II Systems updaten auf die neueste Version. Wie kann ich das durchführen?

Da der HyperBoot-Lader nur HyperBoot-Abbilder laden kann, muß auch ein Update über ein solches Abbild erfolgen. Dazu wird es mit einer speziellen Kennung versehen und es gibt einen speziellen Befehl, der das eigentliche Update durchführt.

Ich habe das Update zu einem HyperBoot-Abbild aufbereitet. Wie bekomme ich es nun in die SolidCard II?

Schreiben Sie das Abbild auf eine Diskette. Mit dem Kommando `u` wird dieses Abbild von Diskette geladen und nach einer Sicherheitsabfrage erfolgt das Update. Wenn Ihr System nicht mehr über ein Diskettenlaufwerk verfügt, können Sie auch über die serielle Schnittstelle ein Update durchführen. Mittels des `objcopy`-Programms aus den sog. Binutils wandeln Sie das erstellte HyperBoot-Abbild um in eine Textdatei im Intel-Hex-Format. Mit dem Kommando `us` starten Sie den Ladevorgang. Beginnen Sie dann die erstellte Hex-Datei zu übertragen. Solange eine kreisende Linie angezeigt wird, läuft der Ladevorgang. Danach erfolgt ebenfalls die Sicherheitsabfrage und danach das Update.

Ich habe für ein System ohne Diskettenlaufwerk. Um das Update durchzuführen verwende ich die serielle Konsole und habe deshalb das Abbild in eine Hex-Textdatei umgewandelt. Nach dem Übertragen der Datei auf die SolidCard II erhalte ich aber nur Fehlermeldungen (beispielsweise: Diese Datei ist kein ROM-Image, sondern 'XFDGDFERRREFD!'). Was läuft da schief?

Wenn Sie die Umwandlung des binären HyperBoot-Abbilds in eine Hex-Textdatei nicht mit dem `binutils`-Programm `objcopy` durchführen, kann ggf. ein Formatfehler vorliegen. Einige Programme wandeln nur 64k große Dateien, weil sie nur 16bit Offsets verwenden. Achten Sie bei Ihrem Umsetz-Programm unbedingt darauf das es 32 bit Offsets bei der Umsetzung erzeugt.

Glossar

Basisspeicher Damit ist der nutzbare RAM-Speicher von Adresse 0x00000 bis 0x9FFFF gemeint. Danach

beginnt normalerweise ein Bereich mit Ein- Ausgabefunktion bzw. ROM-Erweiterungen (sog. Upper Memory Area).

Echtzeit Dieser Begriff wird höchst unterschiedlich ausgelegt. Prinzipiell ist damit gemeint, daß ein zuvor eingetroffenes Ereignis fertig bearbeitet sein muß, bevor das nächste eintrifft. Diese Definition ist zunächst unabhängig von jeglicher Zeitforderung. In der Realität werden aber immer bestimmte Forderungen an die Reaktionszeiten gestellt. Teils müssen nur Millisekunden erreicht werden, zum Teil aber auch Mikrosekunden garantiert sein. Linux selbst kann keine Garantie geben, wie groß die Zeitspanne zwischen einem Ereignis (in der Regel ein Interrupt) und dessen Bearbeitung ist. Dafür existieren Echtzeit-Erweiterungen, wie beispielsweise RTAI.

Echtzeit-Erweiterung Damit ist ein Aufsatz auf ein bestehendes Betriebssystem gemeint. Da beispielsweise Linux® keine Echtzeit-Funktionalität bietet, kann dies mittels einer Echtzeit-Erweiterung erreicht werden. Allerdings ist die Echtzeit-Fähigkeit nur auf diese Erweiterung selbst oder dafür explizit entwickelte Software anzuwenden. Der Grund liegt darin, daß Linux® von dieser Erweiterung „gar nichts mitbekommt“. Hat man also Probleme mit einem Linux®-Standardtreiber (beispielsweise Datenverlust wegen zu langsamer Reaktion des Treibers), muß man diesen Treiber unter der Echtzeit-Erweiterung neu schreiben.

Interrupt Bezeichnet ein Signal, welches ein externes Gerät an den Prozessor senden kann, um von diesem bedient zu werden. Dazu ist ein sog. Treiber notwendig, also ein Programm, welches nur für dieses externe Gerät geschrieben wurde, also dessen spezielle Funktion berücksichtigt. Beispielsweise meldet sich das externe Gerät „serielle Schnittstelle“ mittels einen Interrupt, wenn es Daten empfangen hat. Der dafür vorgesehene Treiber übernimmt diese Daten und leitet sie an das Betriebssystem weiter (siehe auch Stichwort Echtzeit).

PLL Diese Abkürzung steht für „Phase Locked Loop“. Sie bezeichnet einen elektronischen Regelkreis, der aus einem Eingangstakt mit einer festen Frequenz (sog. Referenz-Takt) jede beliebige andere Frequenz erzeugen kann. Diese erzeugte Frequenz kann sowohl höher, als auch niedriger als der Referenz-Takt sein. Beispielsweise wird beim SolidCard II-System aus einem 33MHz Referenz-Takt der 132 MHz Prozessortakt erzeugt.

Treiber Ein Programm (und Teil des Betriebssystems), welches explizit für eine Hardware (ein Peripherie-Gerät) entwickelt wurde. Nur dieses Programm kann die Hardware richtig steuern und bedienen, da es „dessen Eigenheiten kennt“.