Early Printk

Mini-HOWTO for using the earlyprintk=dbgp boot option with a USB2 Debug port key and a debug cable, on x86 systems.

You need two computers, the 'USB debug key' special gadget and two USB cables, connected like this:

```
[host/target] <-------> [USB debug key] <-------> [client/console]
```

Hardware requirements

a. Host/target system needs to have USB debug port capability.

   You can check this capability by looking at a 'Debug port' bit in the lspci -vvv output:

```
# lspci -vvv
...
00:1d.7 USB Controller: Intel Corporation 82801H (ICH8 Family) USB2 EHCI Controller #1 (rev 03) (prog-if 20 [EHCI])
        Subsystem: Lenovo ThinkPad T61
        Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR+ FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B+ ParErr- DEVSEL=medium >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Latency: 0
        Interrupt: pin D routed to IRQ 19
        Region 0: Memory at fe227000 (32-bit, non-prefetchable) [size=1K]
        Capabilities: [50] Power Management version 2
                Flags: PMEClk- DSI- D1- D2- AuxCurrent=375mA PME(D0+,D1-,D2-,D3hot+,D3cold+)
                Status: D0 PME-Enable- DSel=0 DScale=0 PME+
        Capabilities: [58] Debug port: BAR=1 offset=00a0
                             ^^^^^^^^^^^ <==================== [ HERE ]
        Kernel driver in use: ehci_hcd
        Kernel modules: ehci-hcd
...
```

   > **Note**
   >
   > If your system does not list a debug port capability then you probably won't be able to use the USB debug key.

b. You also need a NetChip USB debug cable/key:

   http://www.plxtech.com/products/NET2000/NET20DC/default.asp

   This is a small blue plastic connector with two USB connections; it draws power from its USB connections.

c. You need a second client/console system with a high speed USB 2.0 port.

d. The NetChip device must be plugged directly into the physical debug port on the "host/target" system. You cannot use a USB hub in between the physical debug port and the "host/target" system.

   The EHCI debug controller is bound to a specific physical USB port and the NetChip device will only work as an early printk device in this port. The EHCI host controllers are electrically wired such that the EHCI debug controller is hooked up to the first physical port and there is no way to change this via software. You can find the physical port through experimentation by trying each physical port on the system and rebooting. Or you can try and use lsusb or look at the kernel info messages emitted by the usb stack when you plug a usb device into various ports on the "host/target" system.

   Some hardware vendors do not expose the usb debug port with a physical connector and if you find such a device send a complaint to the hardware vendor, because there is no reason not to wire this port into one of the physically accessible ports.

e. It is also important to note, that many versions of the NetChip device require the "client/console" system to be plugged into the right hand side of the device (with the product logo facing up and readable left to right). The reason being is that the 5 volt power supply is taken from only one side of the device and it must be the side that does not get rebooted.

Software requirements

a. On the host/target system:

You need to enable the following kernel config option:

```
CONFIG_EARLY_PRINTK_DBGP=y
```

And you need to add the boot command line: "earlyprintk=dbgp".

> **Note**
>
> If you are using Grub, append it to the 'kernel' line in /etc/grub.conf. If you are using Grub2 on a BIOS firmware system, append it to the 'linux' line in /boot/grub2/grub.cfg. If you are using Grub2 on an EFI firmware system, append it to the 'linux' or 'linuxefi' line in /boot/grub2/grub.cfg or /boot/efi/EFI/<distro>/grub.cfg.

On systems with more than one EHCI debug controller you must specify the correct EHCI debug controller number. The ordering comes from the PCI bus enumeration of the EHCI controllers. The default with no number argument is "0" or the first EHCI debug controller. To use the second EHCI debug controller, you would use the command line: "earlyprintk=dbgp1"

> **Note**
>
> normally earlyprintk console gets turned off once the regular console is alive - use "earlyprintk=dbgp,keep" to keep this channel open beyond early bootup. This can be useful for debugging crashes under Xorg, etc.

b. On the client/console system:

You should enable the following kernel config option:

```
CONFIG_USB_SERIAL_DEBUG=y
```

On the next bootup with the modified kernel you should get a /dev/ttyUSBx device(s).

Now this channel of kernel messages is ready to be used: start your favorite terminal emulator (minicom, etc.) and set it up to use /dev/ttyUSB0 - or use a raw 'cat /dev/ttyUSBx' to see the raw output.

c. On Nvidia Southbridge based systems: the kernel will try to probe and find out which port has a debug device connected.

Testing

You can test the output by using earlyprintk=dbgp,keep and provoking kernel messages on the host/target system. You can provoke a harmless kernel message by for example doing:

```
echo h > /proc/sysrq-trigger
```

On the host/target system you should see this help line in "dmesg" output:

```
SysRq : HELP : loglevel(0-9) reBoot Crashdump terminate-all-tasks(E) memory-full-oom-kill(F) kill-all-tasks(I) saK show-backtrace-all-act
```

On the client/console system do:

```
cat /dev/ttyUSB0
```

And you should see the help line above displayed shortly after you've provoked it on the host system.

If it does not work then please ask about it on the linux-kernel@vger.kernel.org mailing list or contact the x86 maintainers.