//Original message from Woodruff, Richard at u-boot mailing list 2004-10-18 15:12
// Adapted and tested on SX1 by Vovan888. 2005-10-30
// Made on Gentoo Linux with kernel 2.6.10
// The best SX1 community is at oslik.ru
// “We make things other way. The way of Linux.”

Software configuration

=====================

Host side

On the PC host side, you need to configure the kernel to support USB serial devices. Assuming you are using "make menuconfig" or "make xconfig", go to "USB Support" section and enable the options as follows ("M" means "enable as a loadable module"):

If, and only if, "Support for USB is set to <N>, then:

```
        - Support for USB                              <Y>
```

Also enable the /proc filesystem for USB:

```
        - Preliminary USB device filesystem                 <Y>
```

Then choose a driver for your USB controller:

```
        - UHCI (Intel PIIX4, VIA, ...) support              <Y>
        or
        - OHCI (Compaq, iMacs, OPTi, SiS, ALi, ...) support   <Y>
```

Then go to "USB Serial Converter support" subsection and enable:

```
        - USB Serial Converter support                 <M>
        - USB Generic Serial Driver                    <Y>
```

Next, recompile the kernel and install kernel itself and modules as usual.

After this, make sure you have the following module installed:
  * usbserial.o (usbserial.ko)- host-side serial function driver

For example, if the kernel modules are placed in their default locations, type:

```
  find /lib/modules/`uname -r`/kernel/drivers/usb -type f
```

at the shell prompt. You should see something like this:

```
        /lib/modules/2.6.10/kernel/drivers/usb/serial/usbserial.ko
```

If this is true, host configuration is completed.

Kermit

If you don’t already have kermit installed on your linux host machine, do the following:

- download C-kermit sources from ftp://kermit.columbia.edu/kermit/archives/cku211.zip
- unpack src to suitable directory, go to that dir and run "make linux" and "make install".

Now, when you type "which kermit" you should see "/usr/local/bin/kermit". This is the program that minicom will run when doing uploads to your device. Copy the attached file ./host/.kermrc into your home directory. This file contains important setup commands to initialize kermit.

Minicom

If you don’t already have minicom installed on your linux host machine, do the following:

- download sources from http://www.netsonic.fi/~walker/minicom.html
- unpack src, go to that dir, than run “./configure; make; make install”

To communicate with the TTY client, you’ll need to create a minicom configuration which uses /dev/ttyUSB0. Run minicom, as root, with the -s option and do the following:

- Type <Control-A> O (letter o) to get into the configuration menu.
- Choose "Filenames and paths". Change the value of "E - Kermit program" to the location of kermit on your host system. "which kermit" will give you the correct path to use here. Press enter to go back to the previous screen.
- Choose "File transfer protocols". You should have 2 entries for kermit. Modify the first (has U in the U/D column) so that the "Program" column has the following value: "kermit -i -l %f -s".
- Choose "Serial Port Setup". Now change the value of "A - Serial Device" to be /dev/ttyUSB0.
- Now press enter to go back to the previous screen and select "Save setup as...". Save this setup as usb.
- Exit from minicom.

Siemens SX1 client side
=======================

This part of entire job is done in Windows, since we have no i-boot interface in Linux yet. Install attached patch from ./patch/uboot_on_A.sxp. It replaces disk A with new image containing U-Boot bootloader. In winswup check Z1 and ROLF.

Running
=======

Setting up TTY-over-USB connectivity

On the host side, you need to load kernel modules by typing:

```
modprobe usbserial vendor=0x11F5 product=0x0004
```

Then plug in the standard SX1 USB cable, run U-Boot on SX1 by pressing * + # and power on the phone. (Be sure that USB cable is plugged into SX1 and kernel module is loaded before starting U-Boot !). Then examine log files (usually somewhere in /var/log, in my system it is /var/log/messages; check your etc/syslog.conf) on the host system. You should see something like the following (date and host name fields are skipped):

```
drivers/usb/serial/usb-serial.c: USB Serial support registered for Generic
usbcore: registered new driver usbserial_generic
usbcore: registered new driver usbserial
drivers/usb/serial/usb-serial.c: USB Serial Driver core v2.0
ohci_hcd 0000:00:02.1: wakeup
usb 3-1: new full speed USB device using ohci_hcd and address 2
usbserial_generic 3-1:1.0: Generic converter detected
usb 3-1: Generic converter now attached to ttyUSB0
```

Then check that /dev/ttyUSB0 exists. Maybe you`ll need to create link from /dev/usb/tts/0 to /dev/ttyUSB0. Now you can run minicom with your usb configuration: "minicom usb". You should see the first few lines of your u-boot console:

```
SX1#
```

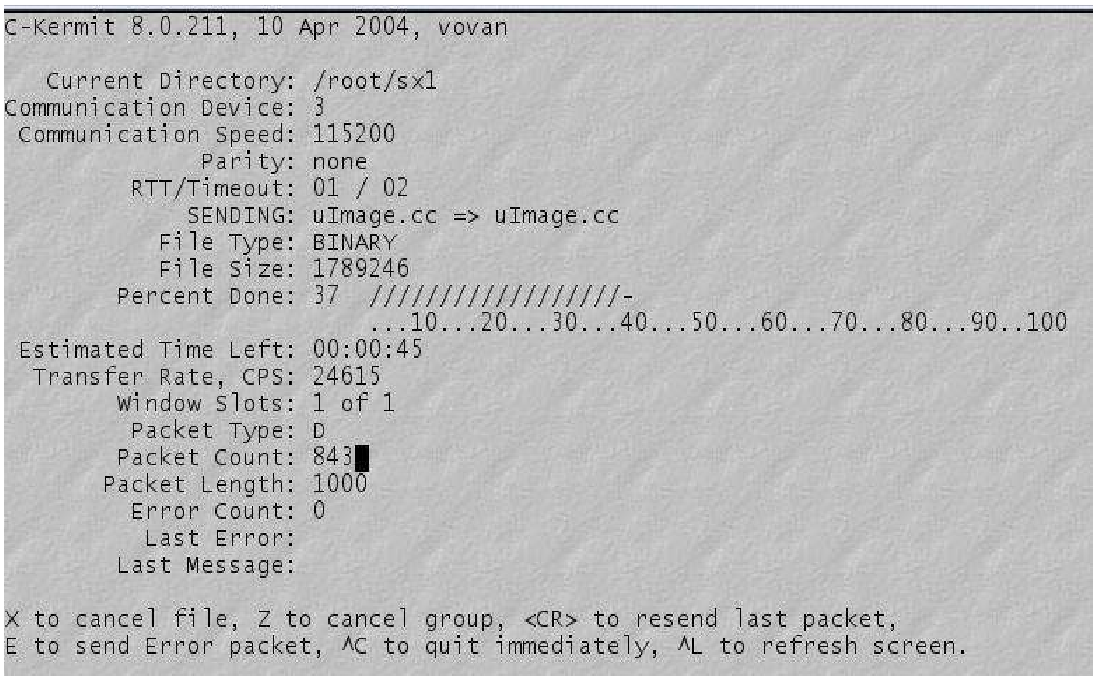
Making U-boot kernel image

First configure and compile kernel image patched for OMAP processors. Then take a look in attached script ./kernel/mkimage.kernel.sh and correct it with your settings. Run it to produce kernel image suitable for download to U-boot. Now you have U-boot kernel image in kernel source directory – uImage.bin or uImage.cc.

Downloading to the SX1

With the modules loaded on the host machine as explained above, set up u-boot for USB downloads using kermit.

- From SX1 console type "loadb". This will pause, waiting for data to be sent from the host.
- Now type <Control-A> S in minicom to bring up the "Upload" menu. Choose "kermit" as the upload protocol. Select the file you wish to transfer and then choose "Okay" (notice menu buttons at bottom of screen, you can use the arrow keys to move between these). You should see a kermit upload progress window. When the transfer has completed, u-boot should display a download summary showing the number of bytes transferred.
-

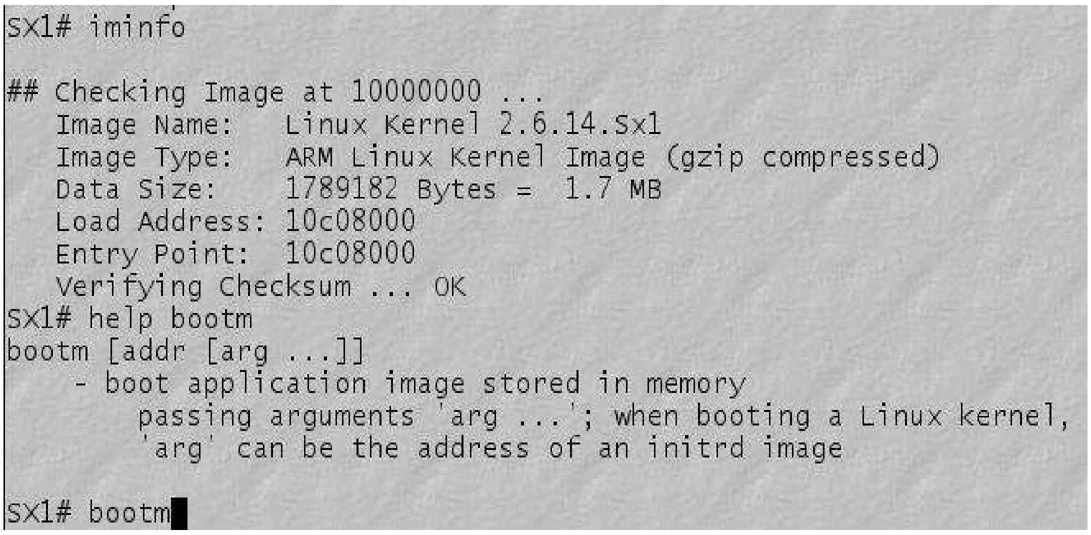


```
C-Kermit 8.0.211, 10 Apr 2004, vovan

   Current Directory: /root/sx1
Communication Device: 3
 Communication Speed: 115200
              Parity: none
         RTT/Timeout: 01 / 02
             SENDING: uImage.cc => uImage.cc
           File Type: BINARY
           File Size: 1789246
        Percent Done: 37  //////////////////-
                          ...10...20...30...40...50...60...70...80...90..100
 Estimated Time Left: 00:00:45
  Transfer Rate, CPS: 24615
        Window Slots: 1 of 1
         Packet Type: D
        Packet Count: 843
       Packet Length: 1000
         Error Count: 0
          Last Error:
        Last Message:

X to cancel file, Z to cancel group, <CR> to resend last packet,
E to send Error packet, ^C to quit immediately, ^L to refresh screen.
```

- After downloading has completed type from console: “iminfo”. You should see info about image downloaded into SDRAM
- Type “help” to get list of available commands.
- Never try flash-write relative commands unless you have some JTAG tool with TI OMAP support !!!
- Now you can type “bootm”, and get kernel image running on SX1.
- Repeat compiling and booting kernel until it works...



```
SX1# iminfo

## Checking Image at 10000000 ...
   Image Name:   Linux Kernel 2.6.14.Sx1
   Image Type:   ARM Linux Kernel Image (gzip compressed)
   Data Size:    1789182 Bytes =  1.7 MB
   Load Address: 10c08000
   Entry Point:  10c08000
   Verifying Checksum ... OK
SX1# help bootm
bootm [addr [arg ...]]
    - boot application image stored in memory
        passing arguments 'arg ...'; when booting a Linux kernel,
        'arg' can be the address of an initrd image

SX1# bootm
```

If console or SX1 hangs, correctly close Minicom (press Ctrl + A then X), remove and install battery on phone. Then start again.