SideARM Flash Image Guide

Revision History

Version	Date	Author	Description
0.1	07/12/04	Tim Meese	Initial Revision
0.2	09/29/04	Tim Meese	Added material on reflashing bootloader per requests

This document is intended to assist developers in reprogramming the on-board flash of the SideARM board. The instructions are specific to the version of firmware currently in use in the SideARM, but generally steps will apply no matter what firmware is employed.

We will start off with some information about where these flash images may be found, some caveats to keep in mind while doing this, and recommendations for maintaining flash images in CVS with a high degree of confidence in their integrity.

Image File Locations

In the current setup, the two image files you will need, **zImage.current** and **root.jffs2**, are stored for your convenience on the server **moonjelly.shore.mbari.org**. Optionally, if you need to reflash the bootloader itself, this file is callec **iboot.img**. When interacting with the bootloader currently utilized on the SideARM or many other bootloaders, file path information is often relative to the tftp server root. I have put the server root in braces below to denote this:

File	Description
`moonjelly:{/tftpboot/}sidearm4/zImage.current`	Linux kernel
`moonjelly:{/tftpboot/}sidearm4/root.jffs2`	JFFS2 root filesystem
`moonjelly:{/tftpboot/}sidearm4/iboot.img`	Bootloader image

Caveats

The following is a list of items to be aware of before you begin the reflashing process. These have been sources of issues in the past and awareness of them is helpful.

Tftp transfers

tftp as you may know is a very simple protocol and has been problematic in the past at times on the MBARI network due to the amount of traffic and the number of subnets involved. Generally, tftp is most robust when on a smaller network. This was the model for it's design (booting workstations on a LAN). I have had occasional problems with the tftp client in the bootloader in Building B and had to setup a private network with a hub and a tftp server running on a PC. I can recommend a good one if you want to do this.

Bootloader Address Parameters

The one trick about issuing the second 'flash' command to flash the root filesystem image is that you need to supply it with the length of the file that you downloaded, whereas the flash of the Linux kernel is always 1MB (0x100000) in length. For the CIMT root filesystem image pointed to by root.jffs2, the length is 0x980000 and you need to specify this number correctly, or else you may not be programming the entire jffs2 image.

Log your Terminal Session

Log your session as I have and go back to check and see if you have typed in all the parameters correctly. It is easy especially when you are just moving chunks of memory around without an MMU to point out that you input bogus values.

Image Archival and Verification Techniques

While image file corruption is not encountered frequently, there are a few steps you can take to ensure that the image files you check into CVS are intact and consistent when you use them to program a SideARM.

Image file verification

One tool that you can use to verify the interity of image files is **md5sum**, which computes a checksum value on the file. You can then check in the output of md5sum and use it again to verify your binary image before programming

For example, I calculated an md5sum on both standard image files for the June 2004 CIMT deployment checked them into the moosmc CVS repository under images/siam/cimt-june04. The md5sum for the root file system image (root.jffs2) is generated by using the 'md5sum' command on the SideARM. After running the clonesa.sh script, run the command like this example (./blah is an example file):

```
timm@seattle:~> md5sum blah
68b329da9893e34099c7d8ad5cb9c940  blah
timm@seattle:~> md5sum blah > blah.md5sum
```

You can then run md5sum in 'check' mode in your CVS tree or on the files on moonjelly to verify that they are the same as the original:

```
timm@seattle:~> md5sum -c blah.md5sum
blah: OK
```

I have done this in the case of the following two image files generated from the CIMT June 2004 deployment. The output of the md5sum command should be redirected into a file with an .md5sum extension (by convention):

```
timm@seattle:~> ls -al ~/moosmccvs/moosmc/images/siam/cimt-june04/
total 18046
drwxr-xr-x  3 timm users     248 2004-06-28 18:01 .
drwxr-xr-x  4 timm users     104 2004-06-28 18:01 ..
-rw-r--r--  1 timm users 9961472 2004-06-28 16:47 cimt-deployed.jffs2
-rw-r--r--  1 timm users      54 2004-06-28 16:47
```

```
cimt-deployed.jffs2.md5sum
-rw-r--r--  1 timm users 8490452 2004-06-28 16:47 cimt-deployed.tar.gz
-rw-r--r--  1 timm users      55 2004-06-28 16:47
cimt-deployed.tar.gz.md5sum
drwxr-xr-x  2 timm users     128 2004-06-28 18:01 CVS
```

I then use md5sum in check mode with the '-c' argument to check the integrity of the files:

```
timm@seattle:~> cd ~/moosmccvs/moosmc/images/siam/cimt-june04/
timm@seattle:~/moosmccvs/moosmc/images/siam/cimt-june04> md5sum -c
cimt-deployed.jffs2.md5sum
cimt-deployed.jffs2: OK
timm@seattle:~/moosmccvs/moosmc/images/siam/cimt-june04> md5sum -c
cimt-deployed.tar.gz.md5sum
cimt-deployed.tar.gz: OK
```

CVS Considerations

Also note that these files were checked in with the standard **-kb** flag todisable text string substitution by CVS when archiving binary files. Note that these are standard practices that are used in a lot of open source development (using CVS and md5sum to verify archived files).

SideARM Reflashing Procedure

Before you begin the reflashing process, ensure that you have a SideARM and a good network connection. Please be careful, patient, and doublecheck your text entry. Also, realize that there are some elements of this process that may not be as robust as desired due to the operating environment.

In the steps below, the central part of the step is in **bold** type. The interactions with the bootloader itself are rendered in a `monospace` font. Within these interfactions, the commands the user is expected to type are in a **`bold-monospace`** font.

1. Cold boot a SideARM, and **get the bootloader's attention** by typing [Ctrl]-C after the "`CS8900A found`" message.

2. Next, **erase the flash** allocated for the Linux kernel and JFFS2 root filesystem. Erase blocks in the Intel Strataflash part that we use on the SideARM are 128KBytes in length and equally sized (i.e. no boot block). The bootloader's 'eraseflash' command takes a starting and ending block number to erase. In our case the starting block number is**3**.

   ```
   IBoot> eraseflash 3
   Do you wish to erase flash blocks 3-127? [y/n] y
   ```

3. **Download the latest sidearm Linux image** from moonjelly. Moonjelly's IP address is 134.89.12.225 – prepend 'tftp:' to this, and use as the first argument. The file /tftpboot/sidearm4/zImage.current always points to the most current image and is the second argument. The third argument is always the hexidecimal constant 0xc0000000.

```
IBoot> download tftp:134.89.12.225 sidearm4/zImage.current
0xc0000000
.Downloading sidearm4/zImage.current: 000C01D0
Downloaded 786896 bytes at 384 kB/s
```

4. Now , **flash the Linux kernel** that has just been downloaded into memory. Enter the hexidecimal constants exactly as they are shown below:

```
IBoot> flashverify 0x60000 0xc0000000 0x100000
Flashing: 160000
```

5. **Download the latest sidearm root filesystem image** from moonjelly. Use the same specification of moonjelly's IP address that you did in step 3 above for the first argument, The file /tftpboot/sidearm4/root.jffs2 always points to the most current image, so use "sidearm4/root.jffs2" as the second argument. The third argument will always be the hexidecimal constant 0xc0000000.

```
IBoot> download tftp:134.89.12.225 sidearm4/root.jffs2 0xc0000000
Downloading sidearm4/root.jffs2: 00980000
Downloaded 9961472 bytes at 164 kB/s
```

6. **Take care to note the hexidecimal number** after the "Downloading sidearm4/root.jffs2:". This is the size of the jffs2 partition data and **you have to supply this number as the third parameter** to the flashverify command (prepending '0x' to it). Repeat, **you have to supply this number as the third parameter** to the flashverify command (prepending '0x' to it). Substitute this hex constant into the command where **[jffs2_size]** is below. Now, program the root filesystem image that has just been downloaded into memory. The first two parameters should be entered exactly as show below:

```
IBoot> flashverify 0x160000 0xc0000000 [jffs2_size]
Flashing: 00AE0000
```

7. Lastly, **issue the 'createfis' command to write the Redboot partition table** to the end of flash. This allows the kernel to determine which mtd partition to mount as the root filesystem. With the 2.6 kernel, this step won't be necessary.

```
IBoot> createfis 1 2 8 116
Flashing: 01000000
```

8. **Reboot the system by pressing the reset button or entering the "reboot" command.** When rebooted, the system will calculate some crypto keys needed for secure functions like ssh. This takes a few minutes on the SideARM, so grab your favorite beverage. The hostname should be reset to 'sidearm' -- you may change it to anything you like. When rebooting, the system console should look similar to the text in Appendix A.

Bootloader Reflashing Procedure

Note that more importantly, misentering address parameters during bootloader reflashing can render a board non-bootable. If this is the case (i.e. No messages are displayed on reset), the board must be programmed via JTAG.

The bootloader's

1. Cold boot a SideARM, and **get the bootloader's attention** by typing [Ctrl]-C after the "`CS8900A found`" message.

2. **Download the latest sidearm bootloader image** from moonjelly. Moonjelly's IP address is 134.89.12.225 – prepend 'tftp:' to this, and use as the first argument. The file /tftpboot/sidearm4/iboot.img always points to the most current image and is the second argument. The third argument is always the hexidecimal constant 0xc0000000.

   ```
   IBoot> download tftp:134.89.12.225 sidearm4/iboot.img
   0xc0000000
   .Downloading sidearm4/iboot.img: 0000D59C
   Downloaded 54684 bytes at 384 kB/s
   ```

3. Now , **flash the bootloader image** that has just been downloaded into memory. Enter the hexidecimal constants exactly as they are shown below:

   ```
   IBoot> flashloader 0 0xc0000000 0x20000
   Flashing: 20000
   ```

4. Lastly, reset the board using the rectangular pushbutton reset switch located in the middle of the board. You should observe the normal Linux boot messages documented in Appendix A.

Appendix A: Normal Linux boot messages

```
Restarting...
+
MBARI SideARM v4
IBoot version 1.7
  built on Feb 12 2004 at 10:46:27

Copyright 2001,2002 Intrinsyc Software Inc.
Copyright 2004 MBARI

CS8900A found at 0x08000300
#2#1#0
Relocating zImage from 00060000 to C0008000 (len=00100000)
Proper ARM zImage ID found. Booting...
Uncompressing
Linux..................................................... done,
booting the kernel.
Linux version 2.4.9-ac10-rmk2-np1-cerf2
(linuxdev@greenflash.shore.mbari.org) (gcc version 2.95.3 20010315
(release)) #10 Wed Apr 28 11:56:38 PDT 2004
Processor: Intel StrongARM-1110 revision 9
Architecture: MBARI SideARM v4
On node 0 totalpages: 16384
zone(0): 16384 pages.
zone(1): 0 pages.
zone(2): 0 pages.
Kernel command line: console=ttySA0 cpufreq=191700 root=1f04 rw
init=/linuxrc
Calibrating delay loop... 127.38 BogoMIPS
Memory: 64MB = 64MB total
Memory: 62428KB available (1615K code, 296K data, 64K init)
Dentry-cache hash table entries: 8192 (order: 4, 65536 bytes)
Inode-cache hash table entries: 4096 (order: 3, 32768 bytes)
Mount-cache hash table entries: 1024 (order: 1, 8192 bytes)
Buffer-cache hash table entries: 4096 (order: 2, 16384 bytes)
Page-cache hash table entries: 16384 (order: 4, 65536 bytes)
POSIX conformance testing by UNIFIX
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
Starting kswapd v1.8
Journalled Block Device driver loaded
JFFS2 version 2.1. (C) 2001 Red Hat, Inc., designed by Axis
Communications AB.
ttySA5 at MEM 0x80050000 (irq = 17) is a SA1100
ttySA6 at MEM 0x80030000 (irq = 16) is a SA1100
ttySA7 at MEM 0x80010000 (irq = 15) is a SA1100
pty: 256 Unix98 ptys configured
block: queued sectors max/low 41402kB/13800kB, 128 slots per queue
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
Uniform Multi-Platform E-IDE driver Revision: 6.31
ide: Assuming 50MHz system bus speed for PIO modes; override with
idebus=xx
loop: loaded (max 8 devices)
SA1100 flash: probing 16-bit flash bus
Using RedBoot partition definition
Creating 5 MTD partitions on "SA1100 flash":
0x00000000-0x00020000 : "I-Boot"
0x00000000-0x00020000 : "RedBoot"
0x00020000-0x00060000 : "Partition 2"
```

```
0x00060000-0x00160000 : "Partition 3"
0x00160000-0x00fe0000 : "Partition 4"
Linux Kernel Card Services 3.1.22
  options:  [pm]
SA-1100 PCMCIA (CS release 3.1.22)
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP, IGMP
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP: Hash tables configured (established 4096 bind 4096)
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
NetWinder Floating Point Emulator V0.95 (c) 1998-1999 Rebel.com
FAT: bogus logical sector size 408
UMSDOS: msdos_read_super failed, mount aborted.
FAT: bogus logical sector size 408
FAT: bogus logical sector size 408
Child dir "." (ino #1) of dir ino #1 appears to be a hard link
VFS: Mounted root (jffs2 filesystem).
Freeing init memory: 64K
Setting up RAMFS, please wait...
Executing /sbin/init...
INIT: version 2.78 booting
Mounting local filesystems...
mount: tmpfs already mounted on /mnt/ramfs
devpts on /dev/pts type devpts (rw,mode=0622)
mount: you didn't specify a filesystem type for /dev/hda1
       I will try all types mentioned in /etc/filesystems or
/proc/filesystems
Trying umsdos
mount: /dev/hda1: unknown device
mount: proc already mounted
Setting the System Clock using the Hardware Clock as reference...
Cannot access the Hardware Clock via any known method.
Use the --debug option to see the details of our search for an access
method.
Calculating module dependencies... depmod: *** Unresolved symbols in
/lib/modules/2.4.9-ac10-rmk2-np1-cerf2/kernel/drivers/net/loopback.o
depmod: *** Unresolved symbols in
/lib/modules/2.4.9-ac10-rmk2-np1-cerf2/kernel/drivers/net/net.o
depmod: *** Unresolved symbols in
/lib/modules/2.4.9-ac10-rmk2-np1-cerf2/kernel/drivers/net/net_init.o
done.
Loading modules: slhc CSLIP: code copyright 1989 Regents of the
University of California
ppp_generic PPP generic driver version 2.4.1
ppp_async ppp_deflate PPP Deflate Compression module registered
bsd_comp PPP BSD Compression module registered
cerf89x0 cerf89x0: DEBUG negating sleep pin
eth0: cs8900 rev J Base 0xF0000300<6>, IRQ 47, MAC 00:0C:6A:00:04:14
sa1100-rtc SA1100 Real Time Clock driver v1.00

INIT: Entering runlevel: 2
Configuring network interfaces: net_open: called
net_close: called
net_open: called
done.
Starting PCMCIA services: cardmgr.
Configuring sa1100_gpio gpio port: cardmgr[61]: starting, version is
3.1.22
cardmgr[61]: watching 2 sockets
SA1100 GPIO v1.3 device (10,224): stencil=0x000FFFF
cardmgr[61]: socket 1: ATA/IDE Fixed Disk
```

```
cardmgr[61]: executing: 'modprobe ide-cs'
done.
Configuring sa1100_lddio lddio port: SA1100 LDDIO v1.0 device (10,225):
stencil=0x0000FFF
Trying to free nonexistent resource <f7000000-f700000f>
hda: SanDisk SDCFH-128, ATA DISK drive
ide0 at 0xf7000000-0xf7000007,0xf700000e on irq 43
hda: 250880 sectors (128 MB) w/1KiB Cache, CHS=980/8/32
Partition check:
 hda:done.
Configuring sa1100_spi spi port:  hda1
ide_cs: hda: Vcc = 3.3, Vpp = 0.0
cardmgr[61]: executing: './ide start hda'
SA1100 SPI v1.3 device (10,223)
done.
Configuring Exar serial ports: xr788: version [1.3 (large flipbuf)]
base[10000000] irq[2] firstminor[0]
ttySX0 at MEM 0x10000000 (irq = 2) is a 16550
ttySX1 at MEM 0x10000020 (irq = 2) is a 16550
ttySX2 at MEM 0x10000040 (irq = 2) is a 16550
ttySX3 at MEM 0x10000060 (irq = 2) is a 16550
ttySX4 at MEM 0x10000080 (irq = 2) is a 16550
ttySX5 at MEM 0x100000a0 (irq = 2) is a 16550
ttySX6 at MEM 0x100000c0 (irq = 2) is a 16550
ttySX7 at MEM 0x100000e0 (irq = 2) is a 16550
xr788: version [1.3 (large flipbuf)] base[10000200] irq[3] firstminor[8]
ttySX8 at MEM 0x10000200 (irq = 3) is a 16550
ttySX9 at MEM 0x10000220 (irq = 3) is a 16550
ttySX10 at MEM 0x10000240 (irq = 3) is a 16550
ttySX11 at MEM 0x10000260 (irq = 3) is a 16550
ttySX12 at MEM 0x10000280 (irq = 3) is a 16550
ttySX13 at MEM 0x100002a0 (irq = 3) is a 16550
ttySX14 at MEM 0x100002c0 (irq = 3) is a 16550
ttySX15 at MEM 0x100002e0 (irq = 3) is a 16550
done.
Starting system logger:
Starting kernel logger:
Generating SSH DSA host key (this is slow, please be patient)...
Generating DSA parameter and key.
Your identification has been saved in /etc/ssh/ssh_host_dsa_key.
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
17:62:82:03:39:27:3e:fc:39:02:fe:a5:c4:c4:d8:55 root@sidearm
Done.
/etc/init.d/ssh: protocols: command not found
Starting OpenBSD Secure Shell server: sshd.
Starting Boa webserver: boa.
 hda: hda1
 hda: hda1
 hda: hda1
EXT2-fs warning: mounting unchecked fs, running e2fsck is recommended

Setting system date/time to 19/06/1922, 06:19:31.
10 Jun 21:22:16 ntpdate[107]: step time server 134.89.10.65 offset
441126165.155534 sec
Starting SIAM application:
/mnt/hda/siam/utils/startnode: /mnt/hda/siam/utils/siamEnv: No such file
or directory
10 Jun 21:22:16 ntpdate[114]: no servers can be used, exiting
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!             Warning SIAM_PORTAL not defined                  !
```

```
!             Starting node without portal                     !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
done.

************************************************
** Intrinsyc Linux (I-Linux)                   **
** Copyright 2001,2002 Intrinsyc Software Inc.  **
** Version: 4.4                                **
** Support: http://support.intrinsyc.com       **
************************************************
sidearm login: root
Password:
```