SideARM Low-level Software Documentation

Introduction

This section of the SideARM low-level software document will detail the installation of i-boot firmware and Linux operating system on the SideARM platform.

Installing bootloader and Linux Software

The process of loading Linux software onto a new SideARM board is accomplished in two phases. The first phase is to load the bootloader using a JTAG programming apparatus called the 'GregorAir' JTAG adapter. The second step is the installation of a Linux kernel and jffs2 root disk image.

Bootloader Installation and initial Configuration

To accomplish the installation of the bootloader, you will need the following:

- Windows NT/2000/XP laptop
- GregorAir Cerfboard JTAG adapter
- Null Modem serial cable
- Access to the MBARI network

1. Obtain the iboot.hex file from the MMC repository (moosmc) under the /images directory.
2. Attach the GregorAir JTAG adapter. It connects to the single row J6 connector on the SideARM. When attached, it should be positioned above the SideARM, not out to the side for correct connector orientation.
3. Run Hyperterm on Win2K or WinXP. Hyperterm must be used because minicom (Linux terminal application) does not observe Xon/Xoff protocol which is used by the GregorAir board for flow control. Configure the Hyperterm connection for 115200 baud, 8 bits, no parity, 1 stop bit, and Xon/Xoff (software) flow control.
4. Power on the board, you should see the following prompt:

```
Idcode, Read, Zap, Word write, Line write :
```

5. Enter 'I' two or three times, this tests the JTAG interface. You should see the following (verify that "C8086499" is displayed)

```
Idcode, Read, Zap, Word write, Line write :
i
C8086499
Idcode, Read, Zap, Word write, Line write :
i
C8086499
```

6. Use the Zap command to erase the flash. Enter "00" as the start block and "7F" as the end block. Enter "Y" to confirm...

```
Idcode, Read, Zap, Word write, Line write :
z
Start block (00..7F):00 00
End block (00..7F):7f 7F
OK? (Y/N)y80
80
```

The adapter will display "80" for each sector of flash that it has erased.

7. Enter 'L', and choose Transfer->Send Text File... from the Hyperterm menu. Select the location you placed iboot.hex in. The GregorAir adapter board should begin accepting data and echoing back "80" each time it has successfully programmed a sector of the flash.
8. Grab a cup of coffee while the GregorAir board accepts data and programs flash.
9. At some point, Hyperterm will finish sending data to the GregorAir board. At this point, you'll need to enter 'q' to finish the process.
10. Power down the system. You are now finished with the GregorAir adapter, so remove it carefully from the SideARM.
11. Close (Hangup) the current Hyperterm session and start a new session with the following parameters: 38400 baud, 8 data bits, 1 stop bit, no parity, no flow control.
12. Power the system on, you should see the following banner displayed if things have gone well:

```
+
************************************************
** Intrinsyc Bootloader (Iboot)                **
** Copyright 2001,2002 Intrinsyc Software Inc.  **
** Version: 1.7                                 **
** Support: http://www.intrinsyc.com            **
************************************************
CS8900A found at 0x08000300
0
Error: No OS image found.
```

13. At the iboot prompt, you will now set the MAC address of the SideARM board. The MAC address is a forty-eight bit identifier consisting of a 24-bit OUI and 24-bit device identifier. MBARI has obtained an OUI (Organization Unique Identifier) from IEEE of 00-0C-6A. These are the first three octets of the MAC address. The last three octets of the MAC address comprise the serial number of the board. If you were programming board number 11, the last three octets would be 00-00-11. Thus to perform an initialization of board number 11 we would issue the following commands:

```
IBoot> set mac 000c6a000011
CAUTION: Changing the MAC address can cause serious network problems.
Are you sure you want to continue? [y/n] y
CS8900A found at 0x08000300
MAC address: 000C6A000011
```

be sure to answer 'y' when prompted. Now issue the 'reboot' command.

Linux installation

After the bootloader has been installed, it is time to install the Linux kernel image and root filesystem image. This is accomplished via the SideARM's network adapter with a tftp (trivial file transfer protocol) server. Thus to perform this step, you need an accessible tftp server.

1. Download and flash the Linux kernel using the following commands. Note that the kernel image size is assumed to be 1MB or less.

```
IBoot> download tftp:134.89.12.225 sidearm/zImage 0xc0000000
```

```
Downloading sidearm/zImage: 0000000
Downloaded 762912 bytes at 57 kB/s
IBoot> flashverify 0x60000 0xc0000000 0x100000
Flashing:
00160000
```

1. Note that if you are using a different host to get your images from, the IP address and filename specified will be different.

Next, download and flash the initial jffs2 root filesystem image. Once again, if the tftp host and filename are different, then substitute the appropriate values. During this phase, the size returned by the download step should be noted as it will be used to specify the size of the image in memory to the 'flashverify' command. In our example here, it is 5373952 or 0x520000 bytes.

```
IBoot> download tftp:134.89.12.225 familiar.jffs2 0xc0000000
Downloading familiar.jffs2: 00520000
Downloaded 5373952 bytes at 87 kB/s
IBoot> flashverify 0x160000 0xc0000000 0x520000
Flashing: 00680000
```

2. Lastly, we need to issue the 'createfis' command to correctly specify the mtd partitions in the flash. We do this by issuing the 'createfis' command.

```
IBoot> createfis 1 2 8 116
Flashing:
0x00FE0000
```

3. As a final step, reboot the SideARM:

```
IBoot> reboot
Restarting...
```

As the system reboots, you should see system messages similar to the following:

```
+
************************************************
** Intrinsyc Bootloader (IBoot)                **
** Copyright 2001,2002 Intrinsyc Software Inc.  **
** Version: 1.7                                 **
** Support: http://www.intrinsyc.com            **
************************************************
CS8900A found at 0x08000300
Relocating zImage from 00060000 to C0008000 (len=00100000)
Proper ARM zImage ID found. Booting...
Uncompressing Linux..................................................... done, b
096 (order: 2, 16384 bytes)

Mounting local filesystems...
mount: tmpfs already mounted on /mnt/ramfs
devpts on /dev/pts type devpts (rw,mode=0622)
mount: proc already mounted
Calculating module dependencies... depmod: *** Unresolved symbols in /lib/modules/2.
depmod: *** Unresolved symbols in /lib/modules/2.4.9-ac10-rmk2-np1-cerf2/kernel/driv
depmod: *** Unresolved symbols in /lib/modules/2.4.9-ac10-rmk2-np1-cerf2/kernel/driv
```

```
done.
Loading modules:
Setting the System Clock using the Hardware Clock as reference...
Executing run once scripts: update-menus Done.

Configuring network interfaces: done.
Starting PCMCIA services: cardmgr.
cardmgr[70]: starting, version is 3.1.22
cardmgr[70]: watching 2 sockets
/etc/rc2.d/S50ssh: expr: command not found
/etc/init.d/ssh: expr: command not found
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:
af:9f:df:f0:f7:b0:43:f1:f1:0e:8e:07:6b:be:d6:08 root@familiar
Done.
/etc/init.d/ssh: protocols: command not found
Starting OpenBSD Secure Shell server: sshd.
**********************************************
** Intrinsyc Linux (I-Linux)                 **
** Copyright 2001,2002 Intrinsyc Software Inc.  **
** Version: 4.4                              **
** Support: http://support.intrinsyc.com     **
**********************************************
```

Linux configuration

After the system has booted, it should present a login prompt ('familiar login:'). We are using the familiar Linux distrubution (http://familiar.handhelds.org), and it includes a network package installation facility called ipkg. This allows us to download and install applications on the fly. This installation and initial configuration will be performed as the 'root' user. The root password on familiar distributions is 'rootme'.

1. Login as root. You should see the following:

   familiar login: root
   Password: [password is 'rootme']
   PAM_unix[87]: (login) session opened for user root by LOGIN(uid=0)

- Use the vi editor, or simply create a new /etc/hostname file that reflects the SideARM's anticipated hostname. Up until now, we have been using the form sidearmXX, where XX is the serial number of the sidearm board.
- Install the ntp client 'ntpdate' by issuing the following ipkg command:

  sh-2.03# ipkg install ntpdate

  You can safely ignore the messages that tar emits stating that the timestamp of the ntpdate files is in the future. We are installing ntpdate to resolve this issue. After installing ntpdate, issue the ntpdate command, specifying wave.mbari.org as the ntp server:

  sh-2.03# ntpdate wave.mbari.org

- Next, issue the following 'ipkg install' commands.

```
sh-2.03# ipkg install procps
sh-2.03# ipkg install ftp
sh-2.03# ipkg install libdb2
```

- Now, halt the system. When all of the LEDs on the top of the board are extinguished, it is safe to power the system down.

```
sh-2.03# halt
```

Exar 16L788 Driver (xr788.c)

The driver for the Exar 16L788 octal uart chip is called xr788.c and produces a kernel module called xr788.o. Because of the planar configuration of the Exar chips on the SideARM, the driver does not implement any autoconfiguration or device discovery (probe) code and relies on kernel parameters specified when the module is loaded for configuration of base address, interrupt number, and the first minor device number the driver should use. The driver was designed to be loaded once for each instance of the Exar 16L788 chip. This requires some extra parameters be provided to the Linux insmod command to avoid namespace collision caused by the default module naming scheme.

The three parameters that the Exar octal UART takes are called chipbase, chipirq, and firstminor. chipbase specifies the physical address at which the Exar chip in question is located at. This address is then mapped to a virtual address using the standard Linux ioremap mechanism. Chipirq specifies the interrupt number to be utilized. firstminor is used to allow the driver to be instantiated more than once with non-overlapping ranges

of minor device numbers. For the current SideARM implementation, these three parameters have the following values:

Exar chip	*chipbase*	*chipirq*	*firstminor*
0 (first)	0x10000000	2	0
1 (second)	0x10000200	3	8

The values of chipbase, chipirq, and firstminor default to 0x10000000, 2, and 0 respectively, which corresponds to the 'first' Exar 16L788 chip on the SideARM board. The values need to be specified manually for the second chip, which starts 0x200 bytes higher in the address space. A simple shell script to load both driver instances, specifying a unique module name for each to avoid namespace collision is the following:

```
#!/bin/sh
insmod -o xr788_0 xr788.o
insmod -o xr788_1 xr788.o chipbase=0x10000200 chipirq=3 firstminor=8
```

The driver assumes a major device number of 19. Although this number is reserved for use by the Cyclades multiport serial device, no such device is envisioned to be ever used on the SideARM, so this number is used as a temporary measure until devfs support can be added to the driver.

Exar Serial Port Allocation

As noted above, the driver is loaded once for each chip instance. The name for the serial ports controlled by the Exar driver(s) is /dev/ttySX*n* where *n* represents the minor device number. The port allocations to each chip as well as the Linux port names and descriptions are described in the table below:

Exar chip	*Exar channel*	*Linux device*	*Port description*
0	0	/dev/ttySX0	Backplane serial port
0	1	/dev/ttySX1	Backplane serial port
0	2	/dev/ttySX2	Backplane serial port
0	3	/dev/ttySX3	Backplane serial port
0	4	/dev/ttySX4	Backplane serial port
0	5	/dev/ttySX5	Backplane serial port
0	6	/dev/ttySX6	Backplane serial port
0	7	/dev/ttySX7	Backplane serial port
1	0	/dev/ttySX8	Backplane serial port
1	1	/dev/ttySX9	Backplane serial port
1	2	/dev/ttySX10	Backplane serial port
1	3	/dev/ttySX11	Backplane serial port
1	4	/dev/ttySX12	RF Modem/transceiver serial port 1
1	5	/dev/ttySX13	RF Modem/transceiver serial port 2
1	6	/dev/ttySX14	Uplink Port (RS485)
1	7	/dev/ttySX15	MSP430 (envp) communications

A short shell script is provided for the

Note that the two 'RF Modem/transceiver' ports are fully power switched. This power switching is accomplished using two bits in the SA1110