CrossCore XA

Software User Guide

crosscontrol

www.crosscontrol.com

Contents

Revision history

Rev	Date	Comments
1.0	2010-07-01	
1.1	2011-10-20	
1.2	2012-06-18	Updates to section 7.9, clarifying the Web interface
1.3	2012-10-25	Updates to 5.12, how to accuire GSM/GPRS modem power status

1. Introduction

This document is primary user guide for *CrossCore XA* and *CrossCore XA All-Integrated* devices.

1.1. Purpose

This user guide contains information of how to access the *CrossCore XA* products running the *Linux* operating system.

Information about configuration files and how to access different hardware modules are also described in this document.

A good prior understanding of Linux systems is needed to fully benefit from this documentation.

1.2. Conventions

Text formats used in this document.

Format	Use
Italics	Paths, filenames, Product names.
Bolded	Command names and important information

⚠ Is used to highlight important information.

1.3. References

[1] CrossCore XA – Programmers Guide

[2] CrossCore XA – Release Notes

2. Device Introduction

This chapter gives an overall description of the *CrossCore XA* device software.

2.1. Bootstrap Software

The boot loader of the *CrossCore XA* device is *at91bootstrap*. It is not interactive and its only responsibility is to load the Linux kernel into RAM memory from NAND flash, and start executing the Linux kernel.

2.2. Linux system

The device can be started into two different modes.

Normal system: this is the main operation mode. This mode includes all drivers to available hardware, system libraries and tools described in this and other documents.

Backup system: this is only for device updating and recovery use. It contains only basic maintenance tools. Updating the main system is done from backup side. Most of the hardware is not available while on backup system.

2.3. Start-up sequence

After power is connected to *CrossCore XA* device the normal system will start-up:

1. Bootstrap software executes. Status LED is static AMBER.
2. Linux is checking the system. Status LED is AMBER flashing at 15Hz.
   One time tasks, like key generations after factory reset is performed in this stage.
3. Linux is loading hardware drivers. Status LED is AMBER flashing at 2Hz.
   User mode execution possible, although all drivers and devices might not be ready.
4. System is up. Status LED is static GREEN.
   All systems go, although DHCP received Ethernet addresses might activate bit later.

2.4. Boot count

If main system fails to boot successfully for 5 times in a row, the device will automatically boot to backup system instead of main system. The reason for failed main system boot can then be examined and corrected in backup system.

As a control point, the boot is deemed successful if the system start-up process gets to level 60. This means that the script file */etc/rc3.d/S60system-up* executes successfully.

It is possible to start user applications before or after this control point. See chapter 3.4.6 for a description of start-up scripts.

2.5. Shutdown sequence

Once shutdown or reboot is initiated:

1. Shutting down, Status LED is AMBER flashing at 7Hz
2. Device is off, Status LED if OFF

2.6. Status led operation

Summary of status led modes:

LED state	Meaning	Comments
OFF	Device is OFF	No power applied or a broken device
GREEN ON	Device is ON in main system	Working with no errors detected
GREEN flashing 2 Hz	Device is ON in backup system	Working in backup system side
AMBER ON	Device in BOOT mode	Boot loader and kernel execution
AMBER flashing 2 Hz	Device in BOOT mode	User space execution, possibility that some hardware peripherals have not started working
AMBER flashing 7 Hz	Device is shutting down	Normal shutdown initiated
AMBER flashing 15 Hz	System Check	System checking file systems and similar actions
RED ON	Device is faulty	Internal fault or life-time of the device ending
RED Flashing 2 Hz	Firmware Update	Device is updating the firmware
RED Flashing 7 Hz	System repair and recovery	System running factory reset system and clearing out all user data

3. System Management and Configuration

3.1. User accounts

There are two user accounts created in the system at start-up, the root user and *cross_admin* user.

The password for the *cross_admin* user is **123cross345**. *Cross_admin* user is available for normal device use, for example running customer applications.

The password for the *root* user is **suseroot**. *Root* is standard Linux root-user which can do everything possible, including maintenance task.

Additional users cannot be created.

3.2. File system

The root file system is located in NAND flash memory. It is of type UBIFS and it is mounted as read-only for security reasons. It is mounted by the kernel boot-up sequence. The optional DFC/Compact Flash card is also mounted during boot-up. The Compact Flash card is formatted with an ext3 file system.

Any attached USB memory is automatically mounted once inserted. Supported formats for USB-memory include FAT which is the default format for USB-memories. USB-memory devices are never automatically formatted so if file system is unsupported, device is just not mounted.

Mount point	Mount status	Media
/	Read only	NAND flash memory
/usr/local	Read-write	NAND flash memory, secondary partition
/media/cf	Read-write	DFC/Compact Flash
/media/usb	Read-write	USB memory, if available
/media/usb2	Read-write	Second USB memory, if available
/tmp	Read-write	RAM memory, for storing temporary files
/var	Read-write	RAM memory, for storing logs etc. during runtime

The */usr/local/* partition of NAND flash contains some default configuration files, but most of its space is reserved for application programs. It is up to the user to decide which applications to run.

3.2.1. Wear leveling

As all flash memory devices have limited write cycles, wear leveling is an important issue. Effect is minimized by rotating the physical block where write access occurs. For */media/cf/* the Compact Flash handles this internally with its own controller. For */usr/local/* the selected file system is UBIFS, which does the same with NAND flash.

User can prolong the device age by keeping sure that the device has as much free space as possible so that the leveling function has enough free blocks to rotate writes with.

3.3. System clock

The system schedule clock is 1000Hz or 1ms.

3.4. User configuration files

The folder */usr/local/etc/* contains several configuration files, which are made available in a default version. These files can easily be replaced with specific customized versions if required. This chapter contains a brief description of the default configuration files and their usage.

3.4.1. Host name configuration

The device host name is supplied by user configurable file */usr/local/etc/hostname*. This file is created with a default host name at initial start-up. The user can then replace the file to set a specific host name.

If the file is removed, it will be restored to default state upon next system boot.

3.4.2. Time zone configuration

A time zone configuration normally points to a file containing time zone data. These time zone configuration files are located in */usr/share/zoneinfo/* or in subdirectories in that directory. To select a time zone, create a symbolic link file */usr/local/etc/localtime* to point to the appropriate time zone data file under */usr/share/zoneinfo/*.

If the */usr/local/etc/localtime*-file does not exist at start-up, a default version is created to use UTC time zone settings.

3.4.3. Network interfaces configuration

The network interfaces configuration file is located in */usr/local/etc/interfaces*. This file is configured to use one Ethernet interface with DHCP IP retrieval. It can be edited to change the default setup of the network interfaces on the target. For instance, it can be configured to use a static IP address on one interface and a DHCP address on the other interface, if two interfaces are required. The */etc/network/interfaces* file is a symbolical link to the */usr/local/etc/interfaces* file. For further information on how to write this file, see 'interfaces' man pages.

If the */usr/local/etc/interfaces*–file does not exist at start-up, a default version is created from template.

3.4.4. DHCP client

The template DHCP client configuration file, called *dhclient.conf.new*, located in */usr/local/etc* is used for creating the */usr/local/etc/dhclient.conf* configuration file. This file must contain the actual host name to be used. The template file *dhclient.conf.new* can be replaced with another configuration file, as long as the host name entry remains "*HMI_eth0*" in that file. This string is replaced with the actual host name in the dynamically created file, which is created on every start-up. For further information on how to write this file, see 'dhclient' man pages.

If the */usr/local/etc/dhclient.conf.new*–file does not exist at start-up, a default version is created from template. The default version looks like:

```
# Wait very long time, approx 3,5 days is 300000 seconds
timeout 300000;
retry 0;
reboot 0;
backoff-cutoff 4;
#select-timeout 5;
#initial-interval 2;
```

```
send host-name "HMI_eth0";
# This makes the dhcp-parameter-request-option
request subnet-mask, broadcast-address, host-name, routers,
    domain-name, domain-name-servers, ntp-servers,
    dhcp-lease-time, dhcp-message-type,
    vendor-encapsulated-options;
```

3.4.5. NTP

The NTP configuration file should be located as */usr/local/etc/ntp.conf*. This file should not contain any server sections since the DHCP request process will use this configuration file as a template to create the *ntp.conf.dhcp* file with NTP server settings from the DHCP reply. Note that the latter file exists only in temporary file system, and is regenerated on every DHCP request/reply cycle. Once this file exists, NTP daemon is restarted with the new configuration file. For more information about NTP and its configuration file, see 'ntpd' man pages.

If the */usr/local/etc/ntp.conf*–file does not exist at start-up, a default version is created from template.

3.4.6. Start-up scripts

The user has the possibility to start applications and scripts by modifying or adding start-up scripts. When the kernel is started, the start-up script *rc* located in */etc/init.d/* is executed. Normally, this script reads start-up scripts under */etc/rcX.d/,* depending on the actual run level X.

The default run level is 3, which is basically the only run level in this system. The *rc* script has been modified so that it parses start-up scripts found in either */etc/rcX.d/* and */usr/local/etc/rcX.d/*. The parsing is done in a temporary directory, so scripts from each source location are interleaved depending on their respective names as described below.

To start applications in run level 3, create a script located in */usr/local/etc/rc3.d/* that starts the desired applications. This is the default run level. Each script must be named *SXXname*, where XX is two digits and corresponds to the order when the script is run. Also note that these scripts are usually sourced, so no exit should be performed within these scripts.

Run level 6 is dedicated for shutdown. When device is shutting down the scripts from */etc/rc6.d/* and */usr/local/etc/rc6.d/* are executed to perform last clean-up actions. Required naming and execution order rules comply with start-up level 3.

3.4.7. SSH files

At initial start-up, the SSH keys are automatically generated and stored in folder */usr/local/etc/ssh/*. This generation process may cause the start-up to be delayed for approximately 10 seconds. Once generated, the key files are not regenerated again unless they are removed. These files should not normally be edited, but they may be replaced with specific key files, if desired. For more information about this, see http://matt.ucc.asn.au/dropbear/dropbear.html.

3.4.8. Logrotate configuration file

The *logrotate* configuration file */usr/local/etc/logrotate.conf* is not available by default. It is up to the user to create the file for appropriate logging, and to start the logrotate service. *Logrotate* is started at boot time only if the configuration file exists. *Logrotate* reads everything about the log files it should be handling from the configuration file. Each configuration file can set global options (local definitions override global ones, and later definitions override earlier ones) and specify log-files to rotate. For more information on how to write the *logrotate* configuration file, see '*logrotate*' man pages.

To start the *logrotate* service, do the following manually (or reboot the unit for same effect):

```
~# /etc/init.d/logrotate stop
~# /etc/init.d/logrotate start
```

Logrotate can be used for compressing and moving latest log files to */usr/local/* for safekeeping. Normally it is run at a 30 minute interval.

3.4.9. Watchdog configuration file

The device has a hardware watchdog that resets the device power. The watchdog can be controlled by the user. The watchdog timeout is 16 seconds by default, and can be changed using configuration file */usr/local/etc/watchdog_timeout.conf*. The file should contain only a single numeric value on first line – that value is the requested timeout in seconds.

Valid values are from 0 to 16. Using a value zero (0) disables the watchdog.

3.4.10. Bluetooth configuration file

Bluetooth is only available as an option in *CrossCore XA All-Integrated* devices.

The Bluetooth configuration files are located at */usr/local/etc/bluetooth/*. Two example configuration files *main.conf* and *rfcomm.conf* are also in that folder. Bluetooth is not activated at boot time. It is up to the user to setup Bluetooth connections.

To activate Bluetooth, do the following manually:

```
~# hciconfig hci0 up
```

3.4.11. GPS configuration file

GPS is only available as an option in *CrossCore XA All-Integrated* devices.

GPS has no default configuration files. See 5.11 for more information.

3.4.12. GSM/GPRS configuration file

GSM/GPRS is only available as an option in *CrossCore XA All-Integrated* devices.

GSM/GPRS has no default configuration files. See 5.12 for more information.

3.4.13. WLAN configuration file

WLAN is only available as an option in *CrossCore XA All-Integrated* devices.

WLAN uses no default configuration files, although one exists. See 5.13 for more information.

3.4.14. Accelerometer configuration file

Accelerometer is available as an option in *CrossCore XA All-Integrated* devices.

Accelerometer has no configuration files. See [1] for more information.

3.4.15. Serial Number Broadcast configuration

Serial Number Broadcast (SNB) is started by default at device boot-up and it will broadcast a specific identification message to the local network every fifth second. The message send frequency can be modified and the service can be completely disabled, using the configuration file */usr/local/etc/ccsnb.conf*. That file does not exist by default. Default values are used if any value is unset or the file does not exist.

```
# Serial Number Broadcast – configuration.
# Lines beginning with '#' are comments.  Unnecessary options can be omitted.

# Message send interval in seconds
INTERVAL=10

# Service disable switch (DISABLE|OFF|0)
#ACTIVE=DISABLE

# Advanced features only. Use with discretion.
#
# Firmware-field is auto discovered, but can be overwritten. String value
#FIRMWARE=1.0.0
#
# UnitType-field is TBD, if unset '0' is used. String value
#UNITTYPE=XA
#
```

3.4.16. Power monitor configuration

The power monitor is started by default at device boot-up. It monitors power state to ensure safe shutdown in case of a power failure.

It is possible to implement a customized power monitor. If a customized power monitor is to be used, the default power monitor needs to be disabled. This is done by creating file */usr/local/etc/disable-powermonitor*. Contents of the file is not important, it can be an empty file as well. Note that in a power failure, after user software has done its activities, the script */usr/bin/post-powerfailure* should be executed, in order to report power failure to the system.

3.5. User libraries

To install additional shared libraries, install the library files into */usr/local/lib/*. Then, update the used library cache file by executing the following command:

```
~# ldconfig -C /usr/local/etc/ld.so.cache
```

If additional library file locations are needed, the paths of these can be added to above command as parameters. The environment variable *$LD_LIBRARY_PATH* can also be used to find library files not in the cache.

Library cache file is never automatically updated. But if the file does not exist at system start-up, it is re-created with default version containing info only about the standard libraries.

3.6. User binaries

Additional binaries such as customer application software or additional open-source solutions are preferably installed to */usr/local/bin/ or /usr/local/sbin/*. These directories are available in the standard path, so adding binaries to these locations does not require an update to the *$PATH* environment variable.

If additional levels of binaries are required, the *$PATH* environment variable must be updated through a start-up script.

4. Accessing CrossCore

There are several ways to communicate with the *CrossCore XA*, either using local or remote access.

4.1. Remote access

The methods described in this chapter require an IP address being assigned to the *CrossCore XA* device. There are several ways of setting the IP address of a device. The default method is DHCP, but this can be changed by editing the file */usr/local/etc/interfaces*. For instance, it is possible to set a static IP address if desired.

4.1.1. SSH

To connect to the *CrossCore XA* device, issue the following command (and give password when asked):

```
# ssh root@X.X.X.X
```

User account *cross_admin* can also be used.

To connect to a host from the *CrossCore XA* device, issue the following command:

```
~# ssh Username@X.X.X.X
```

Above X.X.X.X is known as an SSH server IP address, with username Username. A password might be necessary.

4.1.2. SCP

To copy a file to target use the following command (and give password when asked):

```
# scp File1 root@X.X.X.X:/usr/local/File1
```

To copy a file from the *CrossCore XA* device to a host, use the following command:

```
~# scp File Username@X.X.X.X:File
```

Above X.X.X.X is known as an SSH server IP address, username Username. A password might be necessary.

4.1.3. Password-free login for SSH and SCP

Even though the *root* or *cross_admin* user has password, SSH-connections can be configured to connect without password, using identity files. This method is mainly useful for remotely executed scripts or alike.

On connecting host (not the target *CrossCore XA*), execute the command below and enter an empty passphrase when prompted.

```
~$ ssh-keygen -t rsa -f xa1_rsa
```

Copy (or append) the created *xa1_rsa.pub* file into target *CrossCore XA* as */usr/local/etc/ssh/authorized_keys* -file.

Move the xa1_rsa file to a usable location (e.g. *~/.ssh/*)

Either configure that id-file into use in your *ssh_config* or use it from location when executing **ssh** or **scp**.

```
~$ ssh -i ~/.ssh/xa1_rsa root@XA1
```

4.1.4. Remote command execution

After password-free login is enabled, any commands can be started remotely "without login".

```
~$ ssh root@XA1 "ls -al /usr/local/"
```

If starting services or "background" tasks, append "&" to command between quotes.

4.1.5. SFTP

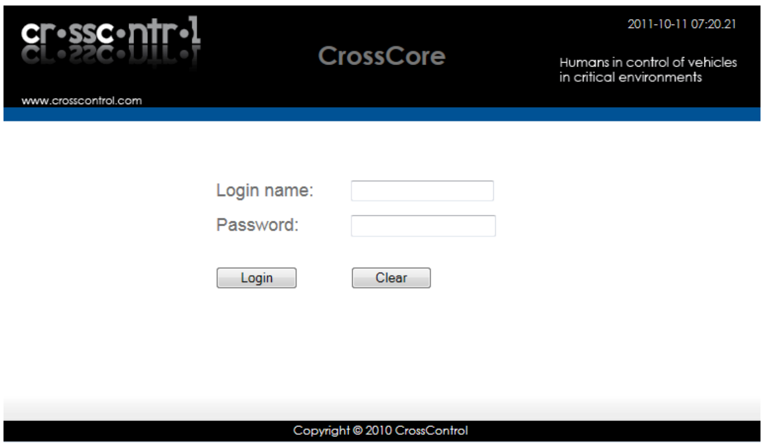
CrossCore XA has an in built FTP server to be used for remotely upload or download information from the unit.

The SFTP uses the same login information as SSH.

4.1.6. HTTP

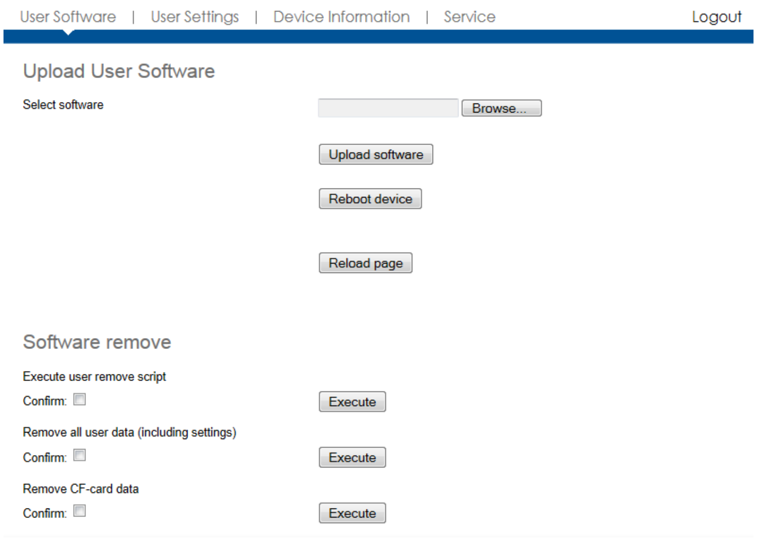
The device has a built in web server for software update and configuration features. It is accessed with a known host name (DNS resolved) or through a known IP address. A standard web browser can be used to access the web interface.

Screenshot below shows the login screen. User accounts root and *cross_admin* can be used.



Screenshot: Web GUI, login screen

User software is uploaded to device and clean-up actions performed using User Software page. More details in 7.9:


User Software | User Settings | Device Information | Service
Logout

Upload User Software

Select software [Browse...]

[Upload software]

[Reboot device]

[Reload page]

Software remove

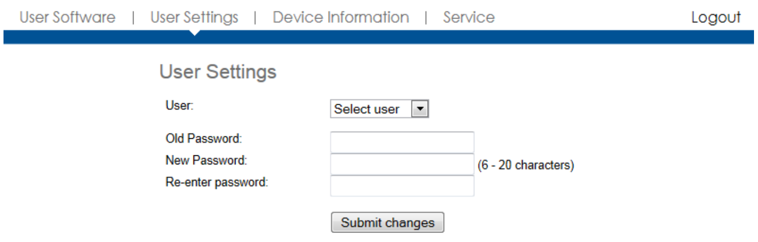
Execute user remove script
Confirm: ☐ [Execute]

Remove all user data (including settings)
Confirm: ☐ [Execute]

Remove CF-card data
Confirm: ☐ [Execute]

Screenshot: Web GUI, User Software page

Properties of both accounts can be edited in the User Settings page, shown below.


User Software | User Settings | Device Information | Service
Logout

User Settings

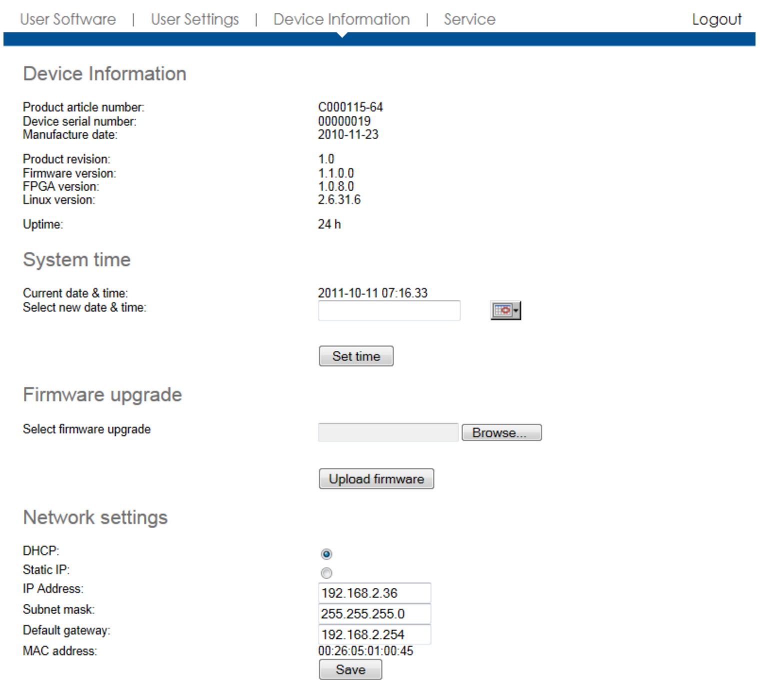
User: Select user

Old Password:
New Password: (6 - 20 characters)
Re-enter password:

[Submit changes]

Screenshot: Web GUI, User Settings page

Screenshot below contains the sample view of device information, with functionality to set time and date, upgrade system firmware, and set networking properties.


User Software | User Settings | Device Information | Service
Logout

Device Information

Product article number:	C000115-64
Device serial number:	00000019
Manufacture date:	2010-11-23
Product revision:	1.0
Firmware version:	1.1.0.0
FPGA version:	1.0.8.0
Linux version:	2.6.31.6
Uptime:	24 h

System time

Current date & time:	2011-10-11 07:16.33
Select new date & time:	

Firmware upgrade

Select firmware upgrade

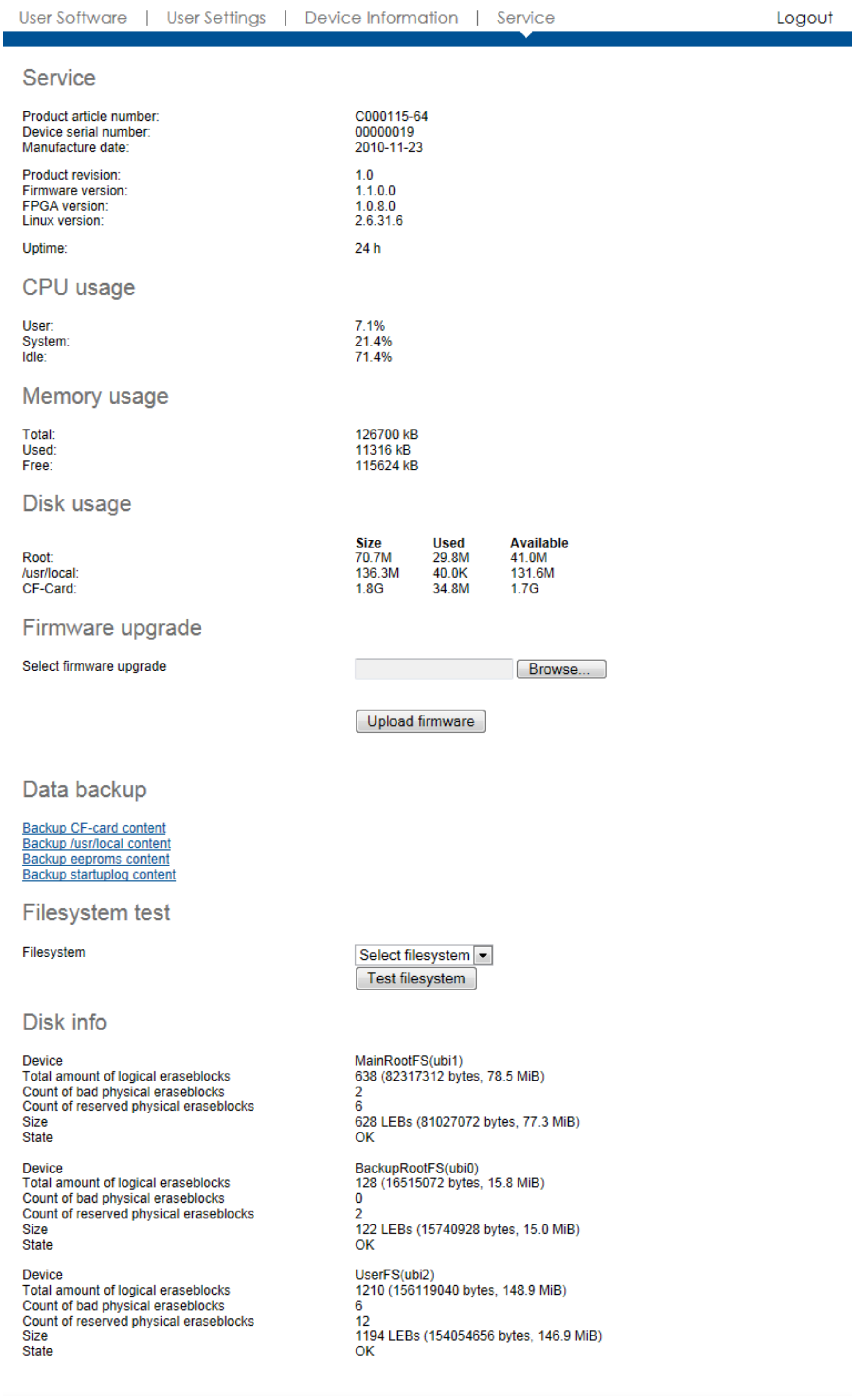
Browse...
Upload firmware

Network settings

DHCP:	●
Static IP:	○
IP Address:	192.168.2.36
Subnet mask:	255.255.255.0
Default gateway:	192.168.2.254
MAC address:	00:26:05:01:00:45

Screenshot: Web GUI, Device Information page

Screenshot below contains the sample view of Service page. It's mainly intended for service people. Backups from the device can be requested and an overall status of the device can be seen on this page.


User Software | User Settings | Device Information | Service
Logout

Service

Product article number:	C000115-64
Device serial number:	00000019
Manufacture date:	2010-11-23
Product revision:	1.0
Firmware version:	1.1.0.0
FPGA version:	1.0.8.0
Linux version:	2.6.31.6
Uptime:	24 h

CPU usage

User:	7.1%
System:	21.4%
Idle:	71.4%

Memory usage

Total:	126700 kB
Used:	11316 kB
Free:	115624 kB

Disk usage

	Size	Used	Available
Root:	70.7M	29.8M	41.0M
/usr/local:	136.3M	40.0K	131.6M
CF-Card:	1.8G	34.8M	1.7G

Firmware upgrade

Select firmware upgrade

Data backup

Backup CF-card content
Backup /usr/local content
Backup eeproms content
Backup startuplog content

Filesystem test

Filesystem

Disk info

Device	MainRootFS(ubi1)
Total amount of logical eraseblocks	638 (82317312 bytes, 78.5 MiB)
Count of bad physical eraseblocks	2
Count of reserved physical eraseblocks	6
Size	628 LEBs (81027072 bytes, 77.3 MiB)
State	OK
Device	BackupRootFS(ubi0)
Total amount of logical eraseblocks	128 (16515072 bytes, 15.8 MiB)
Count of bad physical eraseblocks	0
Count of reserved physical eraseblocks	2
Size	122 LEBs (15740928 bytes, 15.0 MiB)
State	OK
Device	UserFS(ubi2)
Total amount of logical eraseblocks	1210 (156119040 bytes, 148.9 MiB)
Count of bad physical eraseblocks	6
Count of reserved physical eraseblocks	12
Size	1194 LEBs (154054656 bytes, 146.9 MiB)
State	OK

Screenshot: Web GUI, Service page

4.2. Local access

Local access to device can be made using *SSH* as above. Local terminal connection to the device, described below, is not possible without additional options.

4.2.1. Debug card

Debug card is additional peripheral for *CrossCore XA* devices.

Debug card is an additionally available hardware peripheral. This external module contains various debug and development use connectors, such as JTAG and serial ports. It is connected to a connector under *CrossCore XA* device's bottom plate.

4.2.2. Debug serial access

This method requires that a serial cable is connected to the serial port connector on additional debug card. Normally, this is used only for service, development and major maintenance cases.

The serial port settings to be used when communicating with the target are *115200-8-N-1*. The serial interface is currently used as the console during the boot phase. When the system is up and running, it is also possible to use the console as a login facility, giving "physical" access possibilities with the **getty** program on device */dev/ttyS0*.

If PC does not have serial port available, dedicated USB-serial cable can be used together with the debug card to achieve same serial access.

5. Hardware Module Interfaces

The *CrossCore XA* and *CrossCore XA All-Integrated* devices can have different hardware configurations. Therefore, your module may not have all the hardware devices described in this chapter.

This chapter covers the software aspects of the interfaces. Consult the User Manual for more information regarding the actual hardware. Also see 5.6 for module specific information.

5.1. Ethernet

The Ethernet device appears as *eth0*. The Ethernet device is initialized with the **ifplugd** daemon which monitors the device for link status. Upon change of that status, it performs different actions. By default, it invokes the *ifup/ifdown* scripts, which in turn verifies which settings to use with the interfaces configuration file, i.e. to use DHCP or static IP, before actually setting up the interface and retrieving the IP address either statically or dynamically with DHCP-client **dhclient**.

5.2. CAN

The CAN driver is built as a loadable kernel module, which is loaded at system start-up, if the CAN hardware is available. The two CAN interfaces are accessible through network interfaces *can0* and *can1* respectively; they can be listed with the **ifconfig** command. The driver uses the Linux Network and Sockets Layer, providing the SocketCAN method of accessing the CAN buses. For more information about the CAN driver and its API, see the Programmers Guide.

5.3. USB host

The USB port supports USB memory sticks. Certain memory sticks may not work due to the restrictions of the internal USB Host Controller chip.

Up to two USB memories can be automatically mounted upon insertion. The mount points will be */media/usb/* for the first and */media/usb2/* for the one plugged while other is already mounted.

Care must be taken on memory device removal. **Umount** should be called or reasonable time waited after all writing actions before unplugging memory.

Most common USB memory file system formats are supported, including but not limited to FAT16/FAT32 and EXT2/EXT3.

5.4. USB device

The USB device port acts as an Ethernet over USB device. To use it with Windows, an inf – file, which instructs Windows to use the correct driver, is needed. Pre-existing RNDIS – driver is used to communicate with Windows. In Linux the device should work without any external files.

Configuration of the USB Ethernet device is done using the normal method as in 3.4.3.

Device can be configured to either static or dynamic IP. In static mode the device uses a static IP-address and starts up a DHCP server which provides an IP-address for the connecting PC. In dynamic mode the device requests IP-address from the DHCP at the PC.

5.5. RTC

There is a battery backed up RTC on the CrossCore XA device. **Date** command modifies the system time only. The **hwclock** command can be used to transfer system time to and from the RTC.

Normally, start and stop scripts are used for restoring and saving the RTC time in the unit. Note that if NTP is properly set up and available, it might override the RTC clock during start-up.

5.6. EEPROM

Module EEPROM is only available in *CrossCore XA All-Integrated* devices.

There are two EEPROMs in a *CrossCore XA* device and three EEPROMs on *CrossCore XA All-Integrated* device. The two common EEPROMs are accessed as */dev/serial-eeprom*, */dev/user-eeprom*. The additional CrossCore XA All-Integrated device is */dev/module-eeprom*. All available EEPROMs can be read, whilst only user EEPROM can be written by the user.

To read and write an EEPROM, use a standard tool or API for reading and writing files, i.e. **dd** program or **write()** and **read()** functions. The **lseek()** and **mmap()** functions are not supported. Up to 4095 bytes can be written or read.

The serial EEPROM and module EEPROM contains information about the device and are set at the factory. The device node appears in */dev/serial-eeprom*. It can be read as a normal file, but it cannot be written to. This is to protect the data from unintentional overwriting.

Note that the data in serial and module EEPROMs is DOS formatted (carriage return + new line character).

Example:

```
~# cat /dev/serial-eeprom
DATA:1
PCB_SERIAL:S00130503
UNIT_SERIAL:S00000003
DATE:2009-10-12
HWVERSION:PA1
ETH:1
MAC1:00-50-C2-18-98-D8
DIGIO:2
CAN:2
USBDEV:1
USBHOST:1
SERIAL:0
CF:4

~# cat /dev/module-eeprom
MOD_DATA:1
MOD_PCB_SERIAL:S00140500
MOD_DATE:2009-11-05
MOD_HWVERSION:PA1
WLAN:1
WLANMAC1:00-50-C2-18-A9-0C
GPS:0
GSM:1
BLUETOOTH:0
```

The user EEPROM is available for the user. The device node appears in */dev/user-eeprom*. To read and write an EEPROM, use a standard tool or API for reading and writing files, i.e. **dd** program or **write()** and **read()** functions. The **lseek()** and **mmap()** functions are not supported. Up to 4095 bytes can be written or read.

Example:

```
~# echo "mystring" > /dev/user-eeprom
~# cat /dev/user-eeprom
mystring
```

5.6.1. EEPROM data

Integral value fields in read-only serial and module EEPROMs:

Keyword	Value
PCB_SERIAL	Serial number of PCB board
UNIT_SERIAL	Serial number of device, same as written on device brand label.
DATE	Manufacturing date
PROD_REV	Product revision
HWVERSION	Version of HW
MAC1	MAC address of the Ethernet device
DIGIO	Number of digital I/O connectors (2/1/0)
CAN	Number of CAN channels (2/0)
USBDEV	Existence of USB Device port (1/0)
USBHOST	Existence of USB Host port (1/0)
SERIAL	Existence of Serial port connector (1/0)
CF	Size of Internal CF-card (Gigabytes, 0/4/8)
MOD_PCB_SERIAL	Serial number of CrossCore XA All-Integrated module PCB
MOD_HWVERSION	Version of CrossCore XA All-Integrated module HW
WLAN	Existence of WLAN device (1/0)
WLANMAC1	MAC address of the WLAN device
GPS	Existence of GPS device (1/0)
GSM	Existence of GSM/GPRS device (1/0)
BLUETOOTH	Existence of Bluetooth device (1/0)

5.7. Temperature sensor

The *CrossCore XA* device has a built-in temperature sensor. The resolution of the temperature sensor is one degree Celsius. The device node of the driver appears in */dev/tempsensor* which can only be read. Temperature value is updated at every read.

Example:

```
~# cat /dev/tempsensor
52 deg C
```

5.8. Front LED

The *CrossCore XA* device has a multi-color LED at the front of its enclosure, which can be set to; fixed red/green/amber light, off state or flashing. The device node of the driver appears in */dev/frontled*, it can only be written to. It is also available with **ioctl()** calls, see Programmers Guide for more information. Normally, the LED behaviour is already defined with aspect to indication from system software. The LED control from the system is not in any way prioritized, so the latest command always overrides the previous one permanently.

Example, set to off:

```
~# echo “0 0” > /dev/frontled
```

Example, set to fixed red:

```
~# echo “50 0” > /dev/frontled
```

Example, set to flashing green 10 Hz:

```
~# echo “10 1” > /dev/frontled
```

Example, set to flashing amber 5 Hz:

```
~# echo “5 2” > /dev/frontled
```

During normal execution of the system, the user can use the LED for status indication of choice.

Led colors:

Value	Color
0	RED
1	GREEN
2	AMBER

Hertz values:

Range	Led state
0	Off
1 – 50	Flashing at 1-50 Hz
> 50	On

5.9. Digital input and output

There are eight digital input signals that can be used for monitoring external units. Also two digital output signals are available. These signals are monitored through the device node */dev/digio* and can be either read with the **read()** operation, or written with the **write()** operation on a specific format. Signals are also available with **ioctl()** calls, see *Programmers Guide* for more information. The specific read format is illustrated in the example below.

Example:

```
~# cat /dev/digio
 1 = 0
 2 = 0
 3 = 0
 4 = 1
 5 = 0
 6 = 0
 7 = 0
 8 = 1
```

The following formats are allowed to set digital outputs:

Example, set first output to off:

```
~# echo “1 0” > /dev/digio
```

Example, set first output to on:

```
~# echo “1 1” > /dev/digio
```

Example, set second output to off:

```
~# echo “2 0” > /dev/digio
```

Example, set second output to on:

```
~# echo “2 1” > /dev/digio
```

5.10. DFC or Compact Flash

The Compact Flash card mounted in the unit is available on */dev/sda1*. Normally, the system formats this file system automatically once at first power-up. But it may be required to clean the system with new formatting later on. Note that the writeable file system needs to be unmounted before formatting can take place.

5.10.1. Create a file system with mke2fs

The following example illustrates the creation of a file system on the Compact Flash card.

```
~# umount /media/cf
~# /sbin/mke2fs -j /dev/sda1
mke2fs 1.38 (30-Jun-2005)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
31360 inodes, 125436 blocks
6271 blocks (5.00%) reserved for the super user
First data block=1
16 block groups
8192 blocks per group, 8192 fragments per group
1960 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 38 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
~# mount /media/cf
```

5.11. GPS

GPS is only available as an option in *CrossCore XA All-Integrated* devices.

The GPS device is connected on */dev/ttyUSB1*. The output of the GPS device can be verified with the following command:

```
~# stty -F /dev/ttyUSB1 ispeed 9600 && cat </dev/ttyUSB1
```

The command should display GPS data in NMEA format.

NMEA also allows the GPS device to be configured, for more details; locate *Fastrax iSuite*™ NMEA manual.

It is possible to add extra GPS software that can format this data and provide a more user-friendly approach to it. The functionality of the GPS receiver together with the open-source implementation *GPSD* has been verified. For more information, see http://gpsd.berlios.de/ .

5.12. GSM/GPRS

GSM/GPRS is only available as an option in *CrossCore XA All-Integrated* devices.

The GSM/GPRS modem can be accessed through a serial interface, */dev/ttyUSB0*. It supports the standard set of AT commands, so any software that supports the standard AT command specification using serial communication can be used.

The modem has separate power on signals that are required to be exercised to enable the modem.

To power on the modem use **gsmpower** command:

```
~# /usr/bin/gsmpower -on
```

To power off the modem use **gsmpower** command:

```
~# /usr/bin/gsmpower -off
```

Current power status is available via the AT command:

```
~# chat TIMEOUT 2 '' AT | microcom -t 5000 /dev/ttyUSB0
```

5.12.1. Pppd

Pppd allows a GSM data or a high speed GPRS connection to be used for accessing a TCP/IP network. There exists a simple default method to do that in the system. The configuration file */usr/local/etc/gsm-params.txt* contains a few modifiable settings, mainly SIM-card PIN number and authentication values for the network. You get these values from your mobile operator. After modifying the values use following command to create the connection.

```
/etc/init.d/ppp start
```

If the *ppp0* interface does not come up, check the log file from */var/log/pppd.log*.

One known connection failure is NO CARRIER return from ATD command. In this case, access */dev/ttyUSB0* with console and correct the default PDP context setting in your SIM card. If the return value of *"AT+CGDCONT?"*-command is not a valid line starting with *"+CGDCONT:"*, set it using *accesspointname* received from your operator. This configuration is needed only once.

```
~# microcom /dev/ttyUSB0
AT
OK
AT+CPIN?
+CPIN: READY
AT+CGDCONT?
OK
AT+CGDCONT=1,"IP","accesspointname","0.0.0.0",0,0
OK
```

If requiring more detailed configuration, see http://tldp.org/HOWTO/PPP-HOWTO/.

5.13. WLAN

WLAN is only available as an option in *CrossCore XA All-Integrated* devices.

The wireless LAN module is not loaded by default, so it needs to be loaded before the wireless network is setup. To power up WLAN adapter and load necessary modules use command:

```
~# /etc/init.d/wlan start
```

To unload modules and power down the WLAN use:

```
~# /etc/init.d/wlan stop
```

After powering up and enabling WLAN driver as above, the wireless tools that are installed on the device can be used for setting up the actual network. Such tools include the **iwconfig**, **iwlist**, **iwspy**, **iwpriv**, **wpa_supplicant** and **wpa_cli** commands. For details of these tools and instructions on how to set up and configure access to a wireless network, see: http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html.

5.13.1. Association and authentication

Unsecured networks (and simple WEP authentication) can be connected simply using **iwconfig** and “one liner” command. But WPA enabled access points require the use of **wpa_supplicant**-daemon, which uses its own configuration file. A default template of such configuration file is available at the device as */etc/wpa_supplicant.tmpl*. In addition to containing all of the configuration options of the **wpa_supplicant**, the file documents them all in great detail.

In most cases, the file can be stripped down to as simple as below example, which defines an access point with name *XA_WPA*, using *WPA/WPA2* encryption and passphrase “*connectme*”.

```
network={
      ssid="XA_WPA"
      psk="connectme"
      key_mgmt=WPA-PSK
}
```

Example on how to start **wpa_supplicant** and initiate *dhcp* on it:

```
# wpa_supplicant –Dwext –B –ieth1 –c /usr/local/etc/name_of_config
# dhclient –pf /var/run/dhclient.eth1.pid eth1
```

5.14. Bluetooth

Bluetooth is only available as an option in *CrossCore XA All-Integrated* devices.

The Bluetooth modules are not loaded by default, so they must be loaded before the Bluetooth network is setup. To load the Bluetooth modules and to activate the Bluetooth interface, use the following command:

```
~# /etc/init.d/bluetooth start
```

To deactivate the Bluetooth interface and to unload the modules use:

```
~# /etc/init.d/bluetooth stop
```

After this, the Bluetooth tools that are installed on the device can be used for testing the network. Such tools include the **hciconfig**, **hcitool** and **l2ping**. The actual Bluetooth stack is the *BlueZ* open-source Linux Bluetooth stack, see http://www.bluez.org/. *BlueZ* provides support to make connections to other Bluetooth devices, as well as transfer data between devices.

5.15. Accelerometer

Accelerometer is only available as an option in *CrossCore XA* devices.

The accelerometer can be accessed through the device node in */dev/accelerometer*. For its programming interface, see [1].

6. Linux Development Hosts

This section is dedicated to useful tips and hints about Linux development hosts for application development and debugging purposes.

6.1. Retrieval of tool chain

To get the *Code Sourcery G++* cross compiler tool chain, visit the *Code Sourcery* web site www.codesourcery.com. Register an account, and download a trial version to begin with. The current version at the time of writing this document is 4.4-93 for the **ARM GNU/Linux version**. The downloadable file is a binary file, which should have execution rights once downloaded to the development system.

Alternatively, if the Personal/Professional editions coupled with the license costs are not desired, the free binary-compatible *Lite Edition* can also be used. However, that edition doesn't include support the IDE integration from the vendor. If using the *Lite Edition*, for best compatibility it should be *Fall 2009 Release (2009q3),* which is based on GCC 4.4.1, same as the current Professional Edition.

For support on tool chain issues, please refer to *Code Sourcery*.

6.2. Installing tool chain

The installer binary requires the bash shell to be the default shell. If not enabled as default, make sure that */bin/sh* points to bash at least, or perform any other means of getting bash as the default shell. Make sure the installer binary has execution permission, and invoke it with root privileges:

```
# sudo sourceryg++-4.4-93-arm-none-linux-gnueabi.bin
```

The installer will start a Java-based installation GUI. The default settings should be applicable, but it is strongly advised to install the cross compiler to */opt/codesourcery/*, since it is basically a requirement from the build environment.

Once installed, it will try to start the IDE. Let it start Eclipse, and the license can be retrieved with the credentials from the registration at *Code Sourcery*.

6.2.1. Patching tool chain

Current version of the tool chain does not require additional patching.

6.3. Using tool chain

Once installed, the tool chain binaries should be found under */opt/codesourcery/bin/*, all with the **arm-none-linux-gnueabi-** prefix. The tool chain should be added to the path, i.e. add the directory mentioned to the *$PATH* environment variable.

```
user@host:~/$ arm-none-linux-gnueabi-gcc --version
arm-none-linux-gnueabi-gcc (Sourcery G++ 2010.09-103) 4.5.1
Copyright (C) 2010 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO warranty;
not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Now the cross compiler should be ready to use. To use the IDE, its binary can be found in the same directory, called **sourcerygxx-ide**:

```
user@host:~/$ sourcerygxx-ide --version
Sourcery G++ IDE Sourcery G++ 4.4-93
Report bugs to <URL:https://support.codesourcery.com/GNUToolchain/>
```

6.3.1. Using correct development headers

For the customer application to utilize the services on the device the cross-compiler has to find the correct development headers and libraries (see 7.2). Paths of those files must be added to the search path in project settings or the *Makefile*.

E.g.: if development package is extracted to */opt/XA1/*

```
# arm-none-linux-gnueabi-gcc [other_commands] -I/opt/XA1/include -
I/opt/XA1/usr/include -L/opt/XA1/lib -L/opt/XA1/usr/lib
```

6.4. Debugging remotely

6.4.1. gdbserver

To use GDB to debug an application running on the *CrossCore XA* Linux device, the application must have been compiled with the -g flag. Start **gdbserver** on the *CrossCore XA* –Linux device:

```
~# gdbserver :10000 testApplication
```

Then start the host GDB and connect to the server:

```
# arm-none-linux-gnueabi-gdb testApplication
# (gdb) target remote Y.Y.Y.Y:10000
```

Above Y.Y.Y.Y is the IP address of the *CrossCore XA* device. You can now debug the application normally, except that rather than to issue the run command one should use continue since the application is already running on the remote side.

Note that it is possible to fully debug the application but not the system calls made by the application. Such system calls include calls to the soft float library, like divide, add or multiply on floating point variables. It is therefore recommended to use next rather than step when such system calls are being made.

6.4.2. JTAG

Hardware has support for JTAG-debugger. Via JTAG user can debug the system more deeply than that it is possible via software only debuggers. JTAG has support for CPU, FPGA and Ethernet.

JTAG connectors are available on separate debug card, see 4.2.1.

7. Update of CrossCore XA software

The Linux system on a *CrossCore XA* can be updated by an administration user. The update process can also be used for resetting the device to the default state.

Warning: Errors during an update can set the module in an unrecoverable state. In such case, the module must be then shipped to factory for repair.

7.1. Released image files

New versions of *Linux* for *CrossCore XA* device are released as a set of binary format image files as well as additional configuration files and scripts required for update.

The complete system consists of five main partitions in NAND flash, main kernel, backup system kernel, backup system root file system, main root file system and user defined area. Additionally, there is bootstrap software in a data flash acting as the boot loader. Normally, software updates concern only main kernel and root file system, occasionally also the bootstrap.

New releases can also include new FPGA code. This can be updated to any device using debug card and external Xilinx programming tools. Depending on the existing FPGA version the update can also be made without external tools as described in chapter 7.10.

7.2. Released development files

The development package is released together with binary images. This package contains libraries available at the unit and all header files for utilizing them.

The development package is not installed to the *CrossCore XA* device, it shall be installed to the development host used for compiling customer software for *CrossCore XA*. The development package can be extracted to any location at the development host. The location is then specified to the compiler as described in chapter 6.3.1.

When contents of this package change (not necessary at every release), the new version completely replaces the old release. Therefore, it is recommended that the package is extracted to a dedicated directory, which can be erased and recreated without losing anything else.

E.g.: extracting to */opt/XA1/*

```
# mkdir /opt/XA1
# cd /opt/XA1
# tar xzf cross-xa1-devel-n.n.n.tgz
```

7.3. Updating with web interface

Warning: An update erases and replaces the old root file system completely! This should not affect contents of */usr/local/ or /media/cf/,* however backup of important files is advised.

CrossCore XA can also be updated using the maintenance web-interface from *Device Information* - page.

- Under Firmware upgrade, click "Browse..."
- Locate and select the upgrade binary package file (*cross-xa1-n.n.n.tgz*).

- Click button "Upload software".
  Once upload is complete, the page shows the upload complete and now waits for confirmation.
- Tick the checkbox and
- Click “Upgrade” to begin the upgrade procedure.

Do not power off the device, until it reboots itself!

7.4. Updating from console

Warning: An update erases and replaces the old root file system completely! This should not affect contents of */usr/local/ or /media/cf/*, however backup of important files is advised.

This method is possible when the device has working backup system. If backup system is not possible to enter, use external update (7.8).

7.4.1. Preparing update within Linux

If main system fails to boot 5 times in a row, backup system is automatically entered. From a working main system, entering the backup system is done using following reboot command:

```
~# reboot-rescue.sh
```

Once the module has entered backup system, copy in the update images to unit */tmp/* folder. **Scp**, USB-memory or NFS mount can all be used for copying.

Access the Linux console, either over SSH connection from another host, or using a serial console terminal access to Linux.

Warning: Avoid SSH method, if your Ethernet network is susceptible for connection breaks, as the image write can get interrupted.

7.4.2. Updating device within Linux

Prerequisite: New file images shall be located at the */tmp/* folder on the module. Update tool command **fullup.sh** is already present, or it can be copied to target explicitly.

Warning: All excess processes that might interfere or interrupt the update process should be terminated.

```
/tmp # ./fullup.sh
```

Example output:

```
==== CrossControl: Device Update ===== A8.2
Now running on   : BACKUP side
 Requested action: NORMAL system update
     Checking MD5: cross-xa1_kernel.bin: OK
     Checking MD5: cross-xa1_rootfs.bin: OK

Updating: kernel
Updating: root-fs      ALL FILES ON ROOT-FS WILL BE LOST
          Files on /usr/local/ and /media/cf unaffected

User verification: ARE YOU SURE ?
```

```
  - If yes, press Enter to continue.
  - IF NOT, press CTRL+C now to cancel update.
 This is you last change to do cancel!

** IF YOU CONTINUE, DO NOT INTERRUPT THE PROCESS UNTIL READY **
```

The process is pausing here and waiting for Enter or cancellation.

```
        Now doing: Requested reprogramming:

 * DO NOT BREAK PROCESS OR SHUTDOWN THE UNIT before update is complete. *

Erasing old kernel..
Erasing 128 Kibyte @ 3e0000 -- 96 % complete.
Copying new kernel image..
Writing data to block 0
Writing data to block 20000
--clip--
Writing data to block 140000
Erasing old root..
Erasing 128 Kibyte @ 1da0000 -- 37 % complete.
Skipping bad block at 0x01dc0000
Erasing 128 Kibyte @ 4fe0000 -- 99 % complete.
Copying new root-fs image..
Writing data to block 0
Writing data to block 20000
--clip--
Writing data to block fe0000
Completed. Rebooting..
```

7.5. Update automation

This example shows how to automate remote update so that user interaction is not required. Similar automation can also be saved to USB-memory; so that inserting the USB-memory into a working device will execute an update without interaction. This automation is possible on devices with complete and working *CrossCore XA* release 0.3.0 or later.

First setup the devices to enable password-less -login (see 4.1.3).

7.5.1. Updating automatically

Power-up the device. Copy the update images and *fullup.sh* and *update.sh* scripts to the device into */usr/local/fw_update/* -folder. **Scp**, USB-memory or NFS mount can all be used for copying. Then execute **reboot-rescue.sh**

For example:

```
scp -r fw_update -i xa1_rsa root@XA1:/usr/local/
ssh -i xa1_rsa root@XA1 "/usr/sbin/reboot-rescue.sh;/sbin/reboot"
```

Unit will reboot into secondary system, and then update the main system using the files copied above, and then reboot back into normal system.

7.6. USB Autorun

An autorun script can be invoked by connecting a USB memory to USB host interface. The USB memory must include an executable script file named *cc-auto.sh*. That file is then executed automatically at insertion. By placing in commands which copy in new applications, this feature can be used to auto-update user software. There are no specific limits on what user can do with *cc-auto.sh file*.

It is also possible to execute a script on the backup system side, with the following sequence:

1. *cc-auto.sh* creates folder */usr/local/fw_update/*.
2. *cc-auto.sh* copies all necessary files to that folder, including file called *update.sh*.
3. *cc-auto.sh* runs *reboot-rescue.sh*.

   At this point the usb-memory can be removed.
4. Device reboots to backup system.
5. Backup system executes */usr/local/fw_updata/update.sh*.
6. */usr/local/fw_updata/update.sh* does whatever it wants and then deletes the */usr/local/fw_updata/* folder to prevent these actions from executing twice.

7.7. Updating backup system

Note: Updating only backup system does not involve or affect main system.

When the backup system requires update, it is updated from the main system side. If the main side is non-operational and external update (described in chapter 7.8) is not possible to use, then update the main side first and then continue here.

7.7.1. Preparing update within Linux

Copy the backup system update images (cross-*xa1-bs_kernel.bin, cross-xa1-bs_rootfs.bin* and *cross-xa1.md5sum*) and script (*fullup.sh*) to unit */tmp/ folder*. Copy method choice is free; **scp**, USB-memory or NFS mount can all be used.

Access the Linux console as **root** user, either over SSH connection from other host or serial console terminal access to Linux.

7.7.2. Updating unit within Linux

Prerequisite: New file images are at the devices */tmp/* -folder. Update tool command **fullup.sh** is already present, or it can be copied to target explicitly.

Warning: All excess processes that might interfere or interrupt the update process should be terminated.

7.7.2.1. Upgrading whole secondary system

```
/tmp # ./fullup.sh -s
```

Note the -s flag, without it the normal side is updated!

If the -s flag is not recognized (first output line reads: "*Reflashing system with new images of kernel and root-fs.*") the version of *fullup.sh* is too old, locate a newer version and use it.

7.8. Updating unit using SAM-BA

Warning: Update using *SAM-BA* erases and replaces the old system, including contents of */usr/local/* completely! (Should not affect contents of */media/cf/,* however backup of important files is suggested)

7.8.1. Preparing SAM-BA external update

The processor used in the *CrossCore XA* has a built-in boot monitor program that can be used when performing an external update.

Required *SAM-BA* application can be installed from included *Install AT91-ISP v1.10.exe*. The installation add *CrossCore XA* specific files by extracting files from included lib.zip file to folder created by SAM-BA installation: *C:\Program Files\ATMEL Corporation\AT91-ISP v1.10\SAM-BA v2.6\lib\*. Installation may require reboot.

To enable update with USB, extract included *usb-drv.zip* and look instructions within.

7.8.2. Accessing device

Update can be done over serial connection or using USB cable.

Procedure for update via USB:

- disconnect device power
- press down the *disable boot-flash* button located on debug card, while connecting USB device cable to computer
- once the unit is up, release disable button and continue

Procedure for update via serial interface:

- disconnect device power
- press down the *disable boot-flash* button located on debug card, while reconnecting the power

In both cases, when the power is connected, the below text is output into serial console if one is connected. After that the *disable boot-flash* button can be released. However, monitoring the console output is not necessary, and if done so, the terminal connection must be closed at this point!

Output on console: unit ready for *SAM-BA* update.

```
RomBOOT
>
```

7.8.3. Upgrade entire system

Prerequisite: The *CrossCore XA* device is prepared as described in previous chapter.

Warning Updating the entire (main and backup) system erases also the */usr/local/* contents.

CF storage *(/media/cf/)* is unaffected and left untouched.

Execute suitable .bat file from folder containing released binary files:

• *cross-xa1_firmwareupdate-serial.bat*	to install over serial connection
• *cross-xa1_firmwareupdate.bat*	to install over USB connection

After execution, the log file is displayed. If log shows no errors, the unit is ready.

Disconnect all added cables, including USB and power. Then reconnect the necessary cables to continue with the new system.

7.9. Updating user software with web interface

User software, contents of */usr/local/*, can be also upgraded and cleared using the maintenance web-interface, see screenshot at 4.1.5 for sample view.

7.9.1. Upload user software

Warning: Uploading and extracting new user software package overrides any existing files with same filenames.

Click "Browse..." and select the archive file. Then, click button "Upload software" button. This will **prepare** the transfer the selected file to device, and once the “Reload Page” button is pressed, the transfer is completed and the archive file is automatically extracted (using **tar**) to */usr/local/*.

The “Reload Page” button press is required for correctly triggering internal functions of the device after the actual function has been requested.

7.9.2. Reboot actions

For certain software deployments and updates, a device reboot action is required. Press the “Reboot device” button and then wait around 10 seconds, before pressing the “Reload page” button. The device will then perform a reboot and reinitialize itself as intended.

The “Reload Page” button press is required for correctly triggering internal functions of the device after the actual function has been requested.

7.9.3. Clean-up actions

Warning: Clean-up actions can remove existing files.

- "Execute user remove script"
    - Runs the customer script stored at */usr/local/remove.sh*. *CrossCore XA* system does not include this script. It is up to the user to include it in user software, if wanted.
- "Remove all user data"
    - Deletes all content from */usr/local/*.
- "Remove CF-card data"
    - Clears the Compact Flash memory in */media/cf/*.

7.10. Updating FPGA code from console

Prerequisite: FPGA version 1.0.6.0 or later already installed.

Prerequisite: New FPGA code (mcs-file) shall be located at the */tmp/* folder on the module. Update tool command **fullup.sh** is already present, or it can be copied to target explicitly.

Warning: Failure during an update of FPGA code or updating with incompatible code can render the device unbootable. Broken FPGA needs to be reprogrammed using debug card and Xilinx programming tools.

```
/tmp # ./fullup.sh –I Voxnan_v1_0_7_0.mcs -m
```

Example output:

```
 ==== CrossControl: Device Update ===== A8.2
 Now running on   : normal side
  Requested action: FPGA update
   Used file image: Voxnan_v1_0_7_0.mcs
         Preparing: just a minute..

 Updating: FPGA

 User verification: ARE YOU SURE ?

   - If yes, press Enter to continue.
   - IF NOT, press CTRL+C now to cancel update.
  This is you last change to do cancel!

** IF YOU CONTINUE, DO NOT INTERRUPT THE PROCESS UNTIL READY **
```

The process is pausing here and waiting for Enter or cancellation.

```
        Now doing: Requested reprogramming:

 * DO NOT BREAK PROCESS OR SHUTDOWN THE UNIT before update is complete. *

Erasing and Flashing FPGA
Flashing /tmp/fpga.bin  ->  fpga
Verifying the flashed data
Files Voxnan_v1_0_7_0.mcs and readback.mcs are identical

Result: FPGA code flashed and verified successfully.
        The new code is used after next full power off! (not reboot)
```

If the displayed result is not a success, then the FPGA code is corrupt such way that device may not boot-up after power loss. Possible fixes include re-running the same command or trying with another *msc-file*.

8. Default Start-up Log

This is an example of output of device start-up, visible on debug serial port console.

```
RomBOOT
> *** AT91Bootstrap-2.13-rc4
NAND id:0x0000ECDA
> From: [0x00000000] To:       [0x20200000] Size:      [0x001E0000]
> SUCCESS
> Start:       [0x20200000] Uncompressing
Linux............................................................................... done, booting the
kernel.
Executing /etc/rc 3
Setting date from RTC
Starting Cross XA1...
wdt v1.0.1 (Date: 20091026-1246 Revision: 1438)
AT91SAM9263 DIG Driver v1.1.0 (Date: 20091111-1932 Revision: 1571)
AT91SAM9263 PWR Driver v1.0.1 (Date: 20091111-1932 Revision: 1571)
AT91SAM9263 USB Driver v1.0.0 (Date: 20091111-1932 Revision: 1571)
AT91SAM9263 Factory Reset Driver v0.2.0 (Date: 20091111-1932 Revision: 1571)
Setting up temp locations
Setting up local directories, if needed
Setting up DHCP configuration file
Using locally stored NTP configuration file
Using locally stored interfaces configuration file
Using UTC timezone configuration file
Setting up Filesystem in CF...
Starting network...
Setting MAC address to: 00:26:05:00:00:72
Starting dropbear sshd: Starting Network Interface Plugging Daemon:OK
System is up and running...


Welcome to the Cross XA1
CROSS-XA1 login:  eth0.
```

Technical Support

Contact your reseller or supplier for help with possible problems with your *CrossCore XA* device. In order to get the best help, you should have access to your *CrossCore XA* device and be prepared with the following information before you contact support:

- Part number and serial number of the unit, which you find on the brand label
- Date of purchase, which is found on the invoice
- The conditions and circumstances under which the problem arises
- LED indicator flash patterns
- *CrossCore XA* device log files (if possible)
- Version number of *Code Sourcery* –cross compiler used for own binaries
- Description of external equipment which is connected to the *CrossCore XA*

Trademark, etc.