

# CCilot XA and CCpilot XS

## Software guide



## Contents

<b>Revision history</b> .....	Error! Bookmark not defined.
<b>1. Introduction</b> .....	<b>4</b>
1.1. Conventions and defines .....	4
1.2. Identification .....	4
1.3. References .....	4
<b>2. Basic operation</b> .....	<b>5</b>
2.1. Login and passwords .....	5
2.2. Using the touch screen .....	5
2.3. Calibrating the touch screen .....	5
2.4. Keyboard .....	6
2.5. Software deployment .....	6
2.6. Recovery access – log in via serial port .....	6
<b>3. Device start-up behaviour</b> .....	<b>6</b>
3.1. Boot loader .....	6
3.2. Linux system start up specifics .....	6
3.3. Status LED indication .....	7
3.4. Default start-up applications .....	8
<b>4. Accessing and using the interfaces</b> .....	<b>9</b>
4.1. SD file storage .....	9
4.2. CAN .....	10
4.3. Ethernet .....	10
4.4. USB .....	10
4.5. RS232 Serial access .....	10
4.6. Video in .....	10
4.7. Audio Out .....	11
4.8. Digital Input and Output .....	11
4.9. Status LED .....	11
4.10. Backlight .....	11
4.11. Ambient light sensor .....	11
4.12. Buzzer .....	12
4.13. EEPROM .....	12
4.14. Hardware watchdog functionality .....	12
4.15. Temperature sensors .....	12
4.16. GPS .....	12
4.17. GPRS/GSM .....	13
4.18. WLAN .....	13
4.19. Bluetooth .....	14
<b>5. CCsettings</b> .....	<b>15</b>
5.1. Main menu .....	15
5.2. Version .....	15
5.3. Temperature .....	15
5.4. Display .....	16
5.5. Power .....	16

5.6. Heater .....	16
5.7. About .....	17
5.8. LED .....	17
5.9. Buzzer .....	17
5.10. CAN .....	17
5.11. Digital Input/Output .....	18
5.12. Touch screen .....	18
5.13. Light sensor .....	18
5.14. On/Off.....	18
5.15. Advanced.....	19
5.16. Telematics .....	19
<b>6. CCvideo .....</b>	<b>20</b>
6.1. CCvideo main window .....	20
6.2. ccvideostream main window .....	20
6.3. Selection menu .....	20
6.4. Input info .....	21
6.5. Video cropping.....	21
<b>7. Software configuration possibilities.....</b>	<b>21</b>
7.1. Installing new drivers, applications and system packages .....	21
7.2. Text editor.....	23
7.3. IP address configuration .....	23
7.4. Remote access .....	23
7.5. Frame buffer graphics .....	25
7.6. Default start-up application .....	25
7.7. Late load modules .....	27
7.8. Serial Number Broadcast configuration.....	27
7.9. USB memory installer .....	28
7.10. Media files playback .....	29
7.11. VNC .....	29
7.12. Removal of user data and files, i.e. factory reset .....	29
<b>8. Software update and recovery .....</b>	<b>30</b>
8.1. Restore firmware settings .....	30
8.2. Updating firmware components .....	30
8.3. Updating operating system .....	30
8.4. Released image files .....	31
8.5. Updating from console.....	31
8.6. Update automation .....	33
8.7. Updating backup system.....	33
<b>Technical support.....</b>	<b>35</b>
<b>Trade Mark, etc. ....</b>	<b>35</b>

# 1. Introduction

This software guide contains information on how to start using the *CC pilot XA*, *CCpilot XS* and CrossCore XS products and how to access Service functionality.

These devices are available with Linux as operating system. The reader should be familiar with the respective operating system and computer usage to fully benefit from this material.

## 1.1. Conventions and defines

CCpilot XA, CCpilot XS and CrossCore XS are in most cases identical in functionality and usage. The following definition is used to separate unit specific details. The observe symbol is also used to highlight such difference.

Defines	Use
CCpilot XA	Information that is specific for CCpilot XA
CCpilot XS	Information that is specific for CCpilot XS
CrossCore XS device	Information that is specific for CrossCore XS
	Information that applies to CCpilot XA, CCpilot XS and CrossCore XS



The observe symbol is used to highlight information in this document, such as differences between product models.



The A symbol is used to highlight information specific for All-Integrated models



The exclamation symbol is used to highlight important information.

Text formats used in this document.

Format	Use
<i>Italics</i>	Paths, filenames, definitions.
<b>Bold</b>	Command names and important information

## 1.2. Identification

On the side of the device there is a label with version and serial numbers which identify your unique computer. Take note of them. During service and other contact with the supplier it is important to be able to provide these numbers.

## 1.3. References

For further information on the device and the available APIs see the following references.

- [1] CCpilot XA and CCpilot XS – Programmers Guide
- [2] SocketCAN, <http://developer.berlios.de/projects/socketcan/>
- [3] CCAux API specification
- [4] CCpilot XA and CCpilot XS – Technical Manual

## 2. Basic operation

This section provides an overall description on basic usage of the device.

### 2.1. Login and passwords

By default auto login is enabled and no passwords are required to start using the device with the default graphic application. For other accesses, login is required.

The root/administrators passwords are as follows:

- Under Linux there are two users per default. The normal user is **ccs** and the password is **default**. This user has administrator rights by using the sudo approach. The **root** user also exists with password **suseroot** and has full administrator rights.
- Custom users can be added with the tools from respective operating system, but may require some specific prerequisites since the file system is normally write-protected.
  - The adduser program is configured to enable this type of operations in this Linux system.

### 2.2. Using the touch screen

Navigate the startup screen application using the touch screen with a stylus or finger.

- Tap the screen to perform the equality of a mouse click.
- Double click is performed similar to using an external pointing device. Tap the screen twice in the same place.
- Tap and hold the on the touch screen to perform the equivalent to a right click.

Besides the touch screen, keyboard and mouse connected via USB can also be used.



Touch screen functionality is only available in product versions equipped with touch screen display, such as CCpilot XA and CCpilot XS.

### 2.3. Calibrating the touch screen

To calibrate the touch screen, either start Touch Calibration from startup screen application, or open *CCsettings* from the startup screen application, select *Touch* and then *Calibrate Now*. Then follow the step by step sequence to calibrate the display. You can also start these programs from the command line, if required. The startup screen application is further described in 3.4 Default start-up application.

It is also possible to launch the calibration application explicitly from any custom application by calling the *TouchCalibrator* application. This application can also take additional arguments to configure the calibration method, such as number of calibration points (5, 9 or 13 points) and edge distance settings of the calibration points. The 5 point calibration is the default method. The 13 point calibration can be used if the results of the 5 point calibration are not satisfactory. The available command line options can be seen with the `--h` or `--help` option flags.



Touch screen functionality is only available in product versions equipped with touch screen display, Such as CCpilot XA and CCpilot XS.

## 2.4. Keyboard

There is no virtual keyboard available, so a user that requires keyboard access needs to connect a USB keyboard.

## 2.5. Software deployment

There are several methods to add your own software to the device. The standard methods to transfer software to the device are to either copy files using the network connection or to use USB storage devices, with manual or automatic copy functions. To install the software, follow the instructions for the respective software.

Additionally, software can be deployed with remote access functions, see the operating system specific parts in this document for more information.

## 2.6. Recovery access – log in via serial port

There are different ways to perform recovery of a device that has become corrupt in some way. One method is to log in via the external serial port interface. This login method uses the same credentials as other log in points, but the advantage by using the serial port login is that the log in via serial port is always enabled, unless specifically instructed to. The connection configuration is normally 115200-8-N-1 when using a serial port login.

See section 4.5 RS232 Serial access for additional details and how to disable the recovery login via external serial port method. Once logged in the user should have access to the device and should be able to perform necessary recovery actions.

# 3. Device start-up behaviour

At power on, the device has an internal microcontroller that monitors the power supply and performs start-up configurations and power settings, the System Supervisor, or SS. The SS then supplies the main processor and peripherals with power; from there the device execution begins, starting with the boot loader.

## 3.1. Boot loader

The boot loader is the first software block that executes in the main processor. It serves the purpose of setting up the main processor peripherals and timings, and will eventually load the operating system kernel (Linux kernel) into RAM memory. Once finished, the operating system execution takes over. This process takes very little time, in the order of hundreds of milliseconds only.

It is possible to connect to the boot loader through a debug serial port with special equipment attached to the device. The reason for connecting to the boot loader would be for debug purposes only, where specific settings are needed. Detailed information about the boot loader access can be requested upon need.

## 3.2. Linux system start up specifics

The Linux operating system can be started in two different modes:

- Normal or Main system:

- This is the main operation mode which includes all drivers to available hardware, system libraries, graphical applications and tools described in this and other documents.
- Rescue or Backup system
  - This is only for device updates and recovery use. It contains only basic maintenance tools. Updating the main system should be done from backup system. Most of the hardware is not available while on backup system.

Each of these modes is a completely separate a Linux operating system, each of which is a custom built i.MX53 ARM Linux version consisting of a kernel and a root file system with system binaries and configuration files. Such a system is started with the Linux kernel execution, which turns over the execution process to the init system in the root file system. That in turn loads drivers and programs according to the configuration files, and eventually loads the user applications. The startup time of the system is normally defined as the time from power on until the user application can begin to execute.

Depending on the level of functionality needed by the application, it can be started at different startup levels. A very fast startup level means that some of the hardware might not yet have been initialized properly, and thus the application needs to handle that properly. On the other hand, a slower startup level guarantees that the required functionality is available upon application initialization.

### 3.2.1. Boot count

If the main system fails to boot successfully for 5 times in a row, the device can automatically boot to rescue system instead of normal system. The reason for failed main system boot can then be examined and corrected in the 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 Start-up scripts (7.1.4) for a description of start-up scripts.

## 3.3. Status LED indication

This section describes the basic default status LED behavior. The LED is located in different locations depending on the device type, see *Technical Manual* for details. Note that the status LED behavior can be disabled totally and/or overridden by user software through the CCAux API.

### 3.3.1. Start-up sequence

During start-up, the LED indications are as follows:

1. Flashing YELLOW at 1 Hz. Pre-heating is activated and in effect, start-up is delayed while the device is heated.
2. Flashing YELLOW at 2 Hz. Device start-up phase begins.
3. Operating system is then started, and specific service software begins to execute.
4. Static GREEN. System is operational.

### 3.3.2. Shutdown sequence

Once shutdown or reboot is initiated:

1. Static YELLOW during shut down sequence. If it's a reboot, the startup sequence immediately follows the shutdown sequence.
2. LED OFF. Device is off

### 3.3.3. Suspend/resume sequence

Once suspend is initiated:

1. Flashing YELLOW at 0.2 Hz. In suspend mode
2. Flashing YELLOW at 2 Hz. Resuming operation
3. Static GREEN. System is operational.

### 3.3.4. Error indication

There are two different types of error indications available from the status LED:

- Flashing RED – An occasional three RED blinks during startup indicates an internal error that is recoverable, i.e. the system can continue its execution and normal operation. The error code can be retrieved through the CCAux API, since the status LED will follow its normal status indication scheme.
- Flashing BLUE – The device becomes non-operational, and seems to be dead, despite powered on. A number of BLUE LED blinks should be displayed. Record the number of blinks, as this can be useful in an eventual support contact.

Reason for BLUE LED blinking could be voltage levels out of range, temperature related problems or internal hardware errors. First steps should be to let the device cool off, and verify it has correct power supply attached, before starting the device again.

## 3.4. Default start-up applications

On the CCpilot versions, a default graphical application is started when the device is started. This application supports the user with a GUI for simple tasks and information until the startup application can be replaced with the proper user defined application. The replacement of this application is described in section 7.6.

It's also possible to connect to the device via a VNC Viewer application by default, it will however require a known IP address of the device. By default, it's only the CCSettings application that's accessible via the VNC protocol, but it can be replaced with any desired Qt application. Further information about that is found in section 7.11.

## 4. Accessing and using the interfaces

This section covers basic usage and access of the device hardware. Most of the hardware is accessed using the default Linux interfaces but some device specific interfaces may require additional software and/or interfaces to be accessed. See the *CCpilot XA and CCpilot XS – Programmers guide* documentation for general information regarding software development using the devices interfaces.

Depending on product model, all interfaces may not be present on your specific model. There may also be additional methods to access the device than the ones described herein, depending on operating system and additional installed software.

### 4.1. SD file storage

The device uses Secure Digital flash card based storage, i.e. SD card. The storage is identified to the system as a block disk device.



The Secure Digital flash card is industrial grade classified and has both static and dynamic wear levelling to prevent a premature aging and to ensure the longest lifetime. Still Secure Digital flash cards have a limited number of write cycles. It is recommended that the amounts of writing to storage are limited within the application. Rather keep information in RAM memory and write larger blocks at one time instead of frequently writing smaller pieces.

There's though a trade-off that an application needs to make here, if the data to be saved is mission critical or not. An application shouldn't cache files on the SD card, since in case of a sudden power loss, the SD card's writable partition is made write-protected to protect the files from being corrupted due to the power loss. An application needs to be careful when writing large files, as it can cause pro-longed write-protect sequences, which is a potential hazard to the file system and SD card.

Additional details about SD card usage recommendations can be found in *CCpilot XA and CCpilot XS – Programmers guide*.

#### 4.1.1. File system layout

In Linux, the Secure Digital flash card is partitioned into two root file system, which are write protected, and one user file system area, which is write enabled by default. The latter area is the preferred location for user software installations. See chapter 7.1 Installing new drivers, applications and system packages.

The two initial partitions hold each operating system mode's root file systems, i.e. first the rescue system root file system and second the main system root file system. The root file system is located in the SD flash card. It is of type EXT4 and it is mounted as read-only for security reasons. It is mounted by the kernel boot-up sequence.

Any attached USB memory is automatically mounted once inserted. Supported formats for USB-memory include FAT types 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	SD card, second partition
/usr/local	Read-write	SD card, third partition
/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/`, or `/opt`, partition of SD card 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.

## 4.2. CAN

The device can have different number of CAN interfaces with user configurable baud rate and frame type accessible from the `CCsettings` application and/or standard file system settings. Normally the device have two or four CAN interfaces

The CAN interfaces can also be accessed with the Linux operating system standard API *SocketCAN*. More information can be found in *CCpilot XA and CCpilot XS – Programmers Guide*.

## 4.3. Ethernet

Up to two Ethernet ports are available on the device, thus enabling simultaneously connection with two physical network infrastructures. The device is per default set up to use DHCP for IP address retrieval. The network connection settings can be changed within the operating system settings, i.e. by using the network interfaces file as described in 7.3.



Be aware that connecting the device to a network environment can impose a security threat if not taking the required security measures.

## 4.4. USB

Via the USB port, a multitude of peripherals can be connected to the device. For some peripherals, drivers compatible with the operating system must be installed in order for the peripherals to function. For such installations, please contact CrossControl for support in adding a suitable driver, or follow the instruction from the device.

## 4.5. RS232 Serial access

In Linux the external serial port is accessed with `/dev/ttymx1` and can allow connection of peripheral accessories. The serial port can be accessed with standard software methods. The Serial port in the device follows RS232 signalling levels, but with a limited set of signals. For more information see the Technical manual.

By default, serial port console login is enabled when connecting with 115200-8-N-1 settings. This allows for device access with simple methods if the graphical interface fails or is missing. The default login behaviour can be reconfigured to be disabled by creating file `/usr/local/etc/disable-getty` and restarting the device.

## 4.6. Video in

The video in signals can be viewed by the `CCvideo` or `CCvideostream` applications, further described in chapter 6 `CCvideo`. The video in signals can also be accessed and controlled using the `CCAux` API.

-  Video input channels 1 and 2 can be displayed without any significant CPU performance penalty, in one video window instance. Video channels 3 and 4 can be displayed as the second video window instance, but will have some CPU performance penalty due to hardware constraints.
-  Video input display is only valid for CCpilot product instances.

## 4.7. Audio Out

Audio out is controlled using the normal operating system functionality from the ALSA libraries. The audio device is identified as Freescale SGTL5000.

## 4.8. Digital Input and Output

The digital input and output signals are available for software developers using the CCAux API. The digital input and output status can be viewed or set within *CCsettings* for test purposes.

Each I/O signal is an open-drain output with internal pull-up. For example, each I/O can be asserted (pulled to GND) by either setting it to zero from the API (output), or physically connecting it to external GND (input).

The momentary state of each external I/O-pin is sensed and can be read back. This way the signals are not configured as either input or output, they are always both outputs and inputs.

## 4.9. Status LED

The status LED in the device is used by the computer itself and by the operating system to indicate different states or to present warnings, as described in 3.3 Status LED indication. The status LED can also be controlled by the applications running on the device using the CCAux API. The blinking frequency and colour of the LED can be controlled. The latest status LED instruction is always used, but is reset at startup, if the LED isn't completely disabled during startup.

The status LED can also be disabled which means that only hardware error codes and user application behaviour is enabled. Yellow blinking at start-up and static green in operational mode are turned off. The status LED also follows backlight dimming scheme if it's not disabled.

## 4.10. Backlight

The device has an adjustable backlight intensity level. In addition to the backlight control buttons on the device front, the backlight functionality can also be controlled from *CCsettings* and via software using the CCAux API. The most recent setting is always used, it is saved between restarts.

Together with the ambient light sensor it is possible to make a custom, fully automatic, backlight control. Such an automatic backlight control function is included in the device, but it is not enabled by default. It can be set up in *CCsettings* or through the CCAux API.

-  For CrossCore XS, the backlight is not available.

## 4.11. Ambient light sensor

The ambient light sensor measures light levels in front of the device; for example it's used for automatic backlight control. The ambient light sensor is accessed through the CCAux API. It can also be accessed for diagnostic through *CCsettings*.

-  For CrossCore XS, the ambient light sensor is not available.

## 4.12. Buzzer

In addition to the audio In/Out signals, the device is also equipped with a buzzer that can play tones in various frequency and intensity levels. The buzzer is accessed through the CCAux API. It can also be accessed for diagnostic through *CCsettings*.

 For CrossCore XS, the buzzer is not available.

## 4.13. EEPROM

A user accessible EEPROM is available in the device, if such storage is required. Normally a device of this type stores persistent settings in ordinary files instead. The EEPROM is accessed via the CCAux API or via standard read/write calls in Linux, to */dev/user-eprom*.

## 4.14. Hardware watchdog functionality

There is a hardware watchdog available in the device making it possible for the hardware to monitor an application so it does not stall the system. 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 127 Using a value zero (0) disables the watchdog.

By default the watchdog is taken care of by specific watchdog kicker software called *wdt*. If user software wants to take over the watchdog handling, then *wdt* task should be killed first (for example **killall wdt**). If the user software does not want to handle the watchdog anymore, then the *wdt* software should be restarted. *wdt* is started at start-up every time if the watchdog timeout is something other than zero.

## 4.15. Temperature sensors

There are several temperature sensors placed internally in the device. It is possible for an application to retrieve temperature information from the temperatures sensors through the CCAux API.

## 4.16. GPS

 The internal GPS receiver in the XA device follows NMEA-0183 standards. To access the GPS receiver data, you should open up a serial connection to */dev/ttyUSB1* (Linux), with 8-N-1-9600 settings. Note that the GPS receiver needs to be powered on to be able to operate. This is done with the *CCSettings* program or through the CC AUX API.

In Linux, the output of the GPS device can be verified with the following command:

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

Once initiated with an initial read operation, the GPS receiver will start to output NMEA-0183 formatted messages on the serial connection, containing GPS information. 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://www.catb.org/gpsd/>.

NMEA also allows the GPS device to be configured, for more details; locate *Fastrax iSuite™* NMEA manual. It is also possible to use CCSettings for initial verification of GPS receiver functionality, see more in section 5.16.

## 4.17. GPRS/GSM



The All-Integrated models have a GPRS modem for data connections. To access the GPRS/GSM modem, you should open up a serial connection to `/dev/ttyUSB0` (Linux), with 8-N-1-57600 settings. Note that the GPRS/GSM modem needs to be powered on to be able to operate. This is done with the CCSettings program or through the CC AUX API.

The GPRS modem supports the standard set of AT commands, so any software that supports the standard AT command specification using serial communication can be used. For instance, the current power status is available via the AT command (on Linux):

```
# chat TIMEOUT 2 " AT | microcom -t 5000 /dev/ttyUSB0
```

It is also possible to use CCSettings for initial verification of modem functionality, see more in section 5.16.

### 4.17.1. PPP connection

It is possible to set up a data connection over GPRS/GSM using the PPP protocol. This requires that settings relevant for the PPP peers are written in the standard format for this information and stored under `/opt/etc/ppp/peers`. Once this is properly configured, the PPP connection is initialized with the following command:

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

It is possible to stop the connection with the analogue stop command in the above script.

## 4.18. WLAN



The All-Integrated models support WLAN, interoperable with the standard 802.11 b/g networks. Note that the WLAN module needs to be powered on to be able to operate. This is done with the CCSettings program or through the CC AUX API.

In Linux, 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. Additionally, the network interfaces file can be used for configuring WLAN settings.

Additionally, the device needs to load the WLAN drivers for the WLAN to become operational. This is done within a script, which also enables the power to the WLAN circuit, if the API method isn't desired. The script command to use for initializing the WLAN is:

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

Of course, the analogue way for stopping the WLAN is also usable.

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](http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html).

### 4.18.1. 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 the **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
```

## 4.19. Bluetooth



Devices with All-Integrated options have HCI Bluetooth support. HCI stands for Host Controller Interface and provides a uniform interface method for accessing Bluetooth hardware capabilities. The Bluetooth module complies with the Bluetooth 2.1 standard. Note that the Bluetooth module needs to be powered on to be able to operate. This is done with the `CCSettings` program or through the `CC AUX API`.

In Linux, the official Linux Bluetooth protocol stack Bluez is supported, and can be used for communication with Bluetooth devices. Please see the Bluez documentation for more specific Bluetooth usage.

Additionally, the device needs to load the Bluetooth drivers for Bluetooth to become operational. This is done within a script, which also enables the power to the Bluetooth circuit, if the API method isn't desired. The script command to use for initializing Bluetooth is:

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

Of course, the analogue way for stopping Bluetooth is also usable.

## 5. CCsettings

The images here are for illustration purposes and may not represent the real device user interface

CCsettings is used for viewing and adjusting device specific features and information. Through the CCsettings main window, the different settings pages are reached, each containing different areas of settings and information.

All functionality and settings accessed using CCsettings is also available through the CCAux API to enable integration of settings and functionality within any additional user application, or from the command line with the tool `ccsettingsconsole`. For details about this, please see the API documentation or `ccsettingsconsole --help`.

Start CCsettings from its menu item or optional desktop shortcut. In Windows, CCsettings can also be started from the Control Panel.

### 5.1. Main menu

The main screen of CCsettings show the available settings pages.

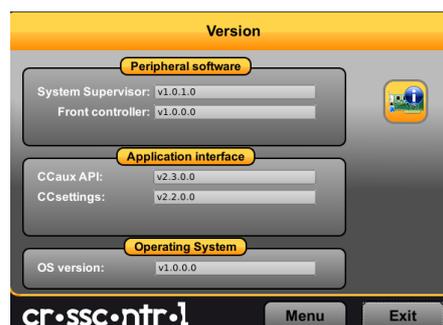
Click on the respective icons to enter its settings page.

It is always possible to revert to the main menu by pressing the *Menu* button. If a keyboard is in use, the Escape key can also be used to revert to the main menu.



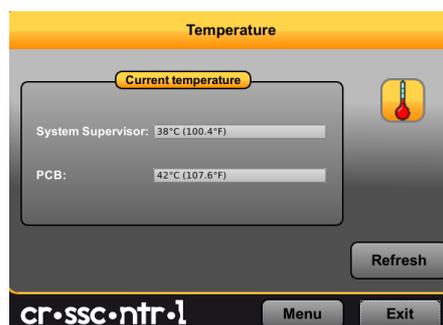
### 5.2. Version

The version page displays the device's internal software versions, as well as application interface version and the version of CCsettings itself.



### 5.3. Temperature

The device internal temperature sensors can be viewed from the temperature page.



## 5.4. Display

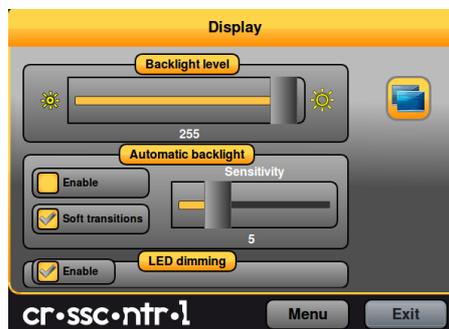
Use the display page to alter the device backlight functionality.

Adjust the backlight of the display by dragging the *Backlight level* slider.

Enable automatic backlight using the *Enable* checkbox. The display brightness is then adjusted using the device light sensor. *Soft transitions* enable smoother adjustment of the backlight.

The *Sensitivity* slider adjusts the level and phase of backlight adjustment depending on the surrounding light.

The Status LED can be automatically dimmed according to the backlight setting if LED dimming is enabled. Note that this may not work if a custom color is used or if a blink sequence is set up by user applications.



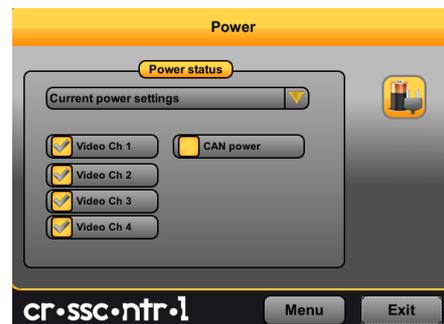
For CrossCore XS, the Display option is not available in CCSettings or ccsettingsconsole.

## 5.5. Power

The power page is used to enable or disable the built in power controls in the device.

Enable power by checking the respective check box.

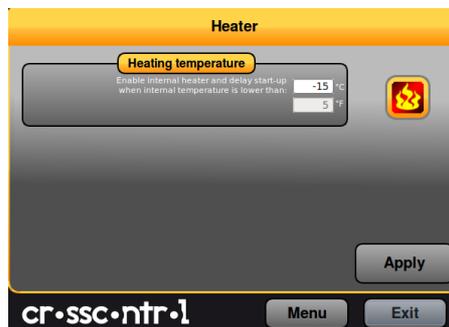
Using the *Power status* drop down box it is possible to adjust the current setup or the default setup during system start-up.



## 5.6. Heater

The heater page is used to adjust the starting temperature of the internal heater.

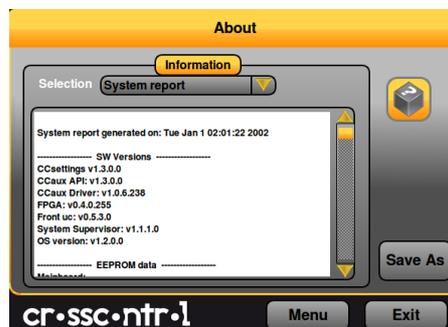
Adjust the temperature and activate the setting by pressing the *Apply* button



## 5.7. About

The about page displays device specific information and status information. Select what type of information to display using the *Selection* drop down box.

Select *System report* drop down to generate a full device information report, which can be saved to file. This report can be very useful for service and support questions.



## 5.8. LED

The LED page is used to view the possible LED indication types and to adjust the LED behavior during startup.

Enable or disable the status LED indication during startup of the device using the *Enable* checkbox.

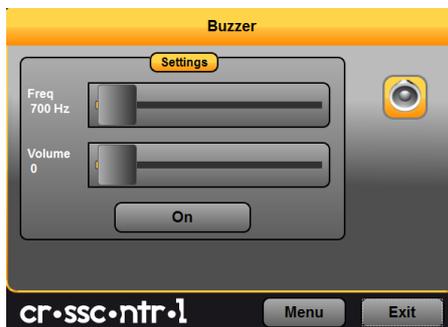


## 5.9. Buzzer

The Buzzer page is used to analyze the possible audio output from the Buzzer.

Adjust the frequency and volume of the built in buzzer using the sliders.

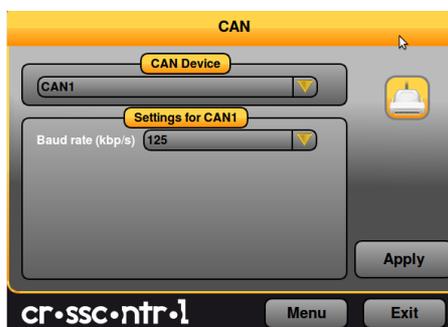
Test the buzzer sound by pressing the *On* button. Press it again to turn off the buzzer.



## 5.10. CAN

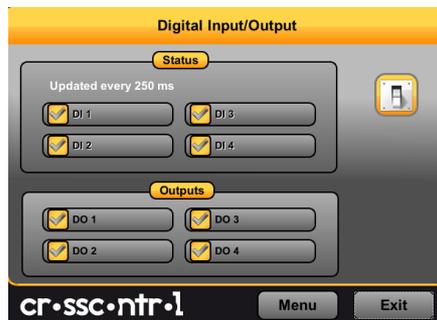
CAN is normally used and configured directly from the controlling software. But from *CCsettings* it is possible to adjust the default *Baud rate* and *CAN Frame type*.

The *Device* drop down box is used to select the CAN device to adjust.



## 5.11. Digital Input/Output

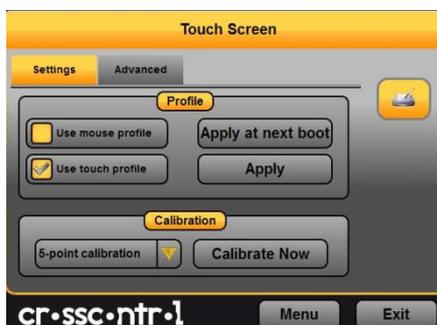
The digital input/output page displays the status of the digital input and output ports on the device.



## 5.12. Touch screen

The touch screen page is used for touch screen related settings. If the touch screen pointing precision seems low it is possible to make a calibration. Start the calibration sequence by selecting *Calibrate Now* and follow the step by step sequence. Different calibration methods can be chosen to perform more precise calibration.

In the *Advanced* tab, additional touch and calibration settings can be made. For instance, by entering a time at *Right click time* and pressing *Apply* it is possible to adjust the time needed until pressing on the touch screen will result in a right click action. See the CCAux API documentation for further information on the advanced settings.



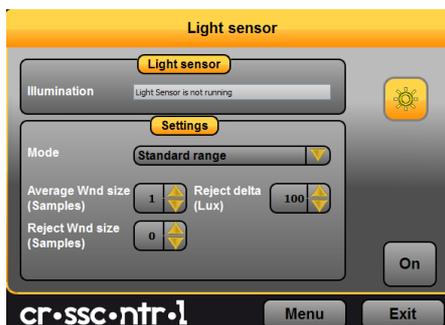
For CrossCore XS, the Touch Screen option is not available in CCSettings or ccsettingsconsole.

## 5.13. Light sensor

The light sensor page is used for Light sensor evaluation. This access is for evaluation, no settings are stored when leaving the light sensor page.



For CrossCore XS, the Light Sensor option is not available in CCSettings or ccsettingsconsole.

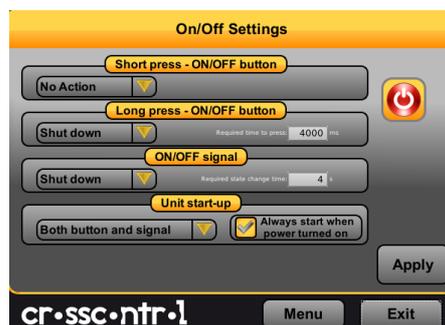


## 5.14. On/Off

The On/Off page is used to adjust the startup and shut down settings of the device.

Any altered setting must be activated using the *Apply* button.

*Short press - ON/OFF button* - adjust the unit behavior when making a quick press and release of the on/off button.



*Long press - ON/OFF button* - adjust the unit behavior when pressing and holding the On/Off button.

*ON/OFF signal* - adjust the behavior of the On/Off signal through the power connector. The time setting adjust how long the signal must stay low before the device reacts to the signal.

*Unit start-up* makes it possible to select if the unit shall respond to start-up from both the on/off button and/or the on/off signal in the power connector, or if it shall be configured to start up directly when supply voltage is applied.

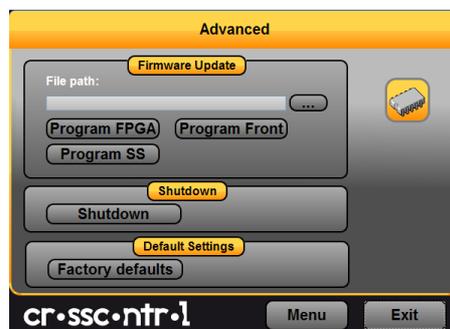
*TBD/Suspend mode* is used for activating automatic transition from suspend mode to shut down of the computer. The time defines the amount of minutes the device shall remain in suspend mode before shutting down. Setting the time value to 0 disables the shut-down feature; the device will stay in suspended mode indefinitely, until started or until power is removed.

## 5.15. Advanced

The Advanced page is used for loading new firmware into the device and to restore settings to factory defaults.

To update the respective firmware, make sure the firmware file is located on the device. Browse for the appropriate file, then press the corresponding upgrade button and wait for the firmware update to complete.

For detailed information about how to upgrade the firmware, please see additional documentation specifically for firmware update.

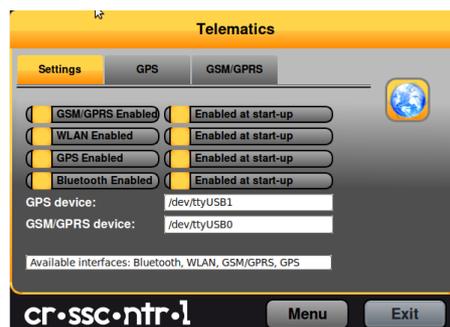


## 5.16. Telematics

**A** The Telematics page is used for control of All Integrated functionality such as the GPRS modem, the GPS receiver, the Bluetooth module and the WLAN chip.

The controllability is about applying power and enabling the specific functionality, either as one time power cycling or as permanent startup setting.

Additionally, the GPS receiver and the GSM/GPRS modem have additional status information and command possibilities in their specific tabs.



## 6. CCvideo

*CCvideo* is a pre-loaded application for viewing the device video in signals. The *CCVideo* application can show video from channels one and two using the CC AUX API, while the CC AUX API can assist other applications with showing video channels 3 and 4. There is also a second video application called *ccvideostream* for displaying video on channels 3 and 4. Two video signals can be simultaneously displayed, from 1,2 and 3,4 video channels respectively.

All functionality and settings accessed using *CCvideo* and *ccvideostream* is also available through the CCAux API to enable integration of settings and functionality within any additional user application. The API is further described in [1].



Video functionality viewing is not applicable for the CrossCore XS.

### 6.1. CCvideo main window

When starting *CCvideo* the main window will be shown as illustrated. This window displays the currently selected video signal from channels 1 or 2.



### 6.2. ccvideostream main window

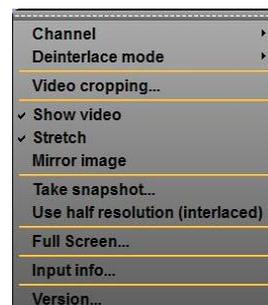
When starting *ccvideostream* the main window will be shown as illustrated. This window displays the currently selected video signal from channels 3 or 4.



### 6.3. Selection menu

The selection menu is activated by right clicking within the *CCvideo/ccvideostream* window. For *ccvideostream*, not all functions are supported.

- ---- - Lock the menu visible.
- *Channel* - Select the video in signal to display.
- *Deinterlace mode* - Select deinterlace mode for video generation.
- *Video cropping* - Opens the Video cropping dialog.
- *Show video* - Enables/disables the video in feed to be displayed.
- *Stretch* - Enables/disables stretching of the video image to fit the size of the *CCvideo* application window.

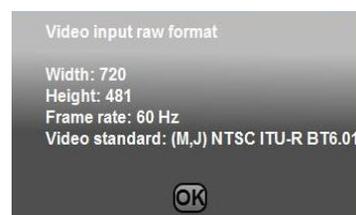


- *Mirror image* - Enables/disables video image mirroring.
- *Full Screen* – View video in full screen mode.
- *Input info* - Displays input information for the currently viewed video in feed.
- *Version* – Display the software version of *CCvideo*.

## 6.4. Input info

The input info dialog displays information for the currently used video in feed.

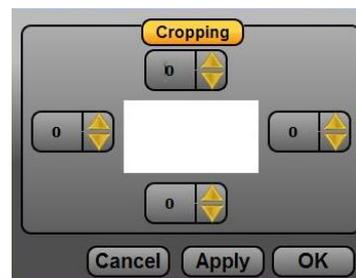
- *Width* - Native width of the signal in pixels.
- *Height* - Native height of the signal in pixels.
- *Frame rate* - Current frame rate of the video signal.
- *Video standard* – The detected video standard of the video signal.



## 6.5. Video cropping

Use the video format settings to adjust the video being shown.

- *Cropping* - Removes black or unwanted parts of the video signal. The video feed is cropped by the number of lines set for each part of the picture.



## 7. Software configuration possibilities

This section describes specific details for the configurability of the software components in the system, such as default configuration files, startup scripts and networking settings.

### 7.1. Installing new drivers, applications and system packages

All additional software or user files should normally be installed and stored under the user partition */opt/* (or */usr/local/*), which is mounted read-write. Under */opt/*, a limited set of standard directories are found which are always writable, those directories are */opt/etc/*, */opt/bin/*, */opt/lib/*, and */opt/sbin/*.

#### 7.1.1. Remounting file system in read-write mode

In very rare cases, editing write-protected files may be required. It is possible to temporarily mount the file system writeable, to allow edit of protected files, using the following commands.

```
# sudo mount -o remount,rw /
```



Important, remember to use the following command to remount the file system as write protected again, before shutting down or restarting the device. Note that any changes to the write-protected file system will be overwritten when performing an operating system upgrade.

```
# sudo mount -o remount,ro /
```

### 7.1.2. User libraries

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

```
~# sudo ldconfig -C /opt/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 for finding 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 information only about the standard libraries.

### 7.1.3. User binaries

Additional binaries such as customer application software or additional open-source solutions are preferably installed to `/opt/bin/` or `/opt/sbin/`. These directories are available in the standard path, 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.

### 7.1.4. 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, so applications should at least have startup scripts for this run level. The `rc` script has been modified to parse additionally start-up scripts found in `/opt/etc/rcX.d/` as well as standard system scripts. 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 `/opt/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 of the script execution.



Note that these scripts are usually sourced, so no exit should be performed within these scripts, nor should any application lock the scripts by not performing proper spawning or forking.

The scripts should follow the correct format for the start-stop system to work correctly. For more information on how the scripts should be created, see standard reference documentation for `rc` scripting.

Run level 6 is dedicated for shutdown. When the device is shutting down the scripts from `/etc/rc6.d/` and `/opt/etc/rc6.d/` are executed to perform last clean-up actions. Required naming and execution order rules comply with start-up level 3, but kill scripts are normally named `KXXname` instead.

Additionally, there's a specific shutdown script available for user editing. It is called `/opt/shutdown`, and is by default an empty script. Any user can add specific shutdown related tasks in that script, if desired.

## 7.2. Text editor

The console text editor *nano* is available per default for text editing, as well as the *vi* editor.

## 7.3. IP address configuration

There are several ways of setting the IP address of a device. The default method is DHCP, but a static IP address can also be used. This can be done through the network interfaces configuration file.

### 7.3.1. File method for IP address configuration

The network interfaces file is located in writable storage, but the edit process has been made transparent to that fact. This method requires knowledge about the interfaces file format, but a sample is given below.

```
# sudo nano /etc/network/interfaces
```

Sample of interfaces file setting same static address as the graphical method above.

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.2.185
netmask 255.255.255.0
gateway 192.168.2.254
```

Once the file has been edited, it is recommended to either reboot the device, or to bring the network interfaces down and up again, for the IP address configuration to take effect.

```
# sudo ifdown eth0
# sudo ifup eth0
```

## 7.4. Remote access

The methods described in this chapter require an IP address being assigned to the device.

### 7.4.1. SSH

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

```
# ssh ccs@X.X.X.X
```

To connect to a host from the 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.

### 7.4.2. SCP

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

```
# scp File1 ccs@X.X.X.X:/opt/File
```

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

```
# scp ccs@X.X.X.X:/opt/File File
```

To copy a file from a host while on a device to a host, use the following command:

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

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

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

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

### 7.4.3. Password-free login for SSH and SCP

Even though the *ccs* 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 device), execute the command below and enter an empty passphrase when prompted.

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

Copy (or append) the created *xa\_rsa.pub* file into target device as a */etc/ssh/authorized\_keys* -file. Note, this method needs to be done with root file system mounted as read-write.

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

Either configure the id-file into use in *ssh\_config* or assign it when executing **ssh** or **scp**.

```
~$ ssh -i ~/.ssh/xa_rsa ccs@X.X.X.X
```

### 7.4.4. Remote command execution

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

```
~$ ssh ccs@X.X.X.X "ls -al /opt/"
```

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

### 7.4.5. VNC

The device Linux version VNC server availability is currently unsupported. The idea is to use it for remote access via GUI, for instance when using CrossCore XS headless devices.

To access the device via VNC, you will need to use a VCN viewer application and enter the IP address of the device, along with the password when prompted for.

## 7.5. Frame buffer graphics

The graphics framework uses the Linux frame buffer for graphics operations. It's fast and efficient, and has vast standard support in the Linux user space world. For instance, Qt has a plugin that enables the Qt libraries to be built with direct frame buffer drawing possibilities. For Qt applications the impact for that means that it simply needs to be started with a specific flag, `-qws`, and built with the correct development libraries.

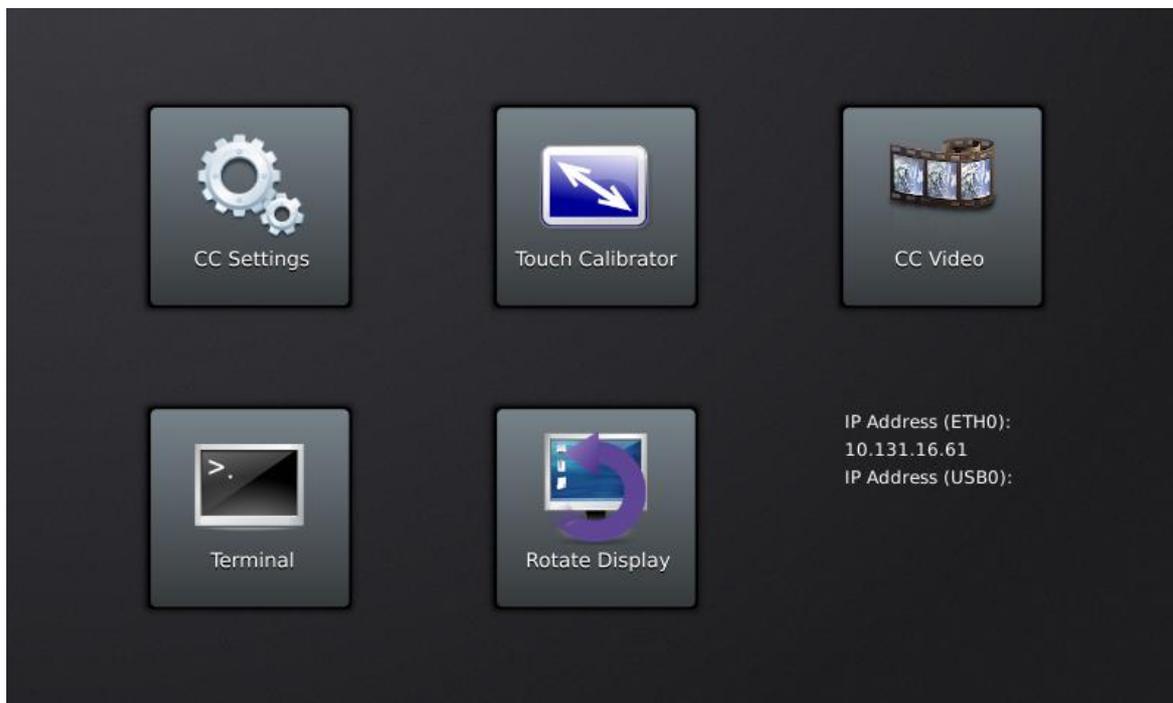
Other application software that uses the frame buffer directly can also be used. However, there is no default windowing system on the frame buffer, so careful consideration there needs to be taken when using this kind of applications.



Multiple instances of Qt applications work fine, and can be seen as a windowing system as well, since that's a characteristic of the Qt QWS system. However, only one Qt application should act as the GUI server, i.e. use the `-qws` flag when starting. Make sure that the graphical applications adhere to that requirement for proper usage of Qt applications on the frame buffer.

## 7.6. Default start-up application

As mentioned in 3.4 Default start-up application, CCpilot versions have a GUI application that is started by default. CrossCore XS versions still has the actual GUI application installed and executable, but it's not started since the system can detect that the display is missing. The default application looks like:



A user can and should replace or simply edit the default startup application script, i.e. the init script found at `/opt/etc/init.d/StartupGui`. This init script is automatically run when entering run level 3 (during boot) or run level 6 (during shutdown/reboot).



Note: This script is part of the rc script solution mention in section 7.1.4. Make sure the correct script behavior is met; otherwise it can break the XA device startup procedure.

An example of the default startup script is given here for illustration purposes:

```
#!/bin/sh
. /etc/init.d/cross-common

#
# END USER: Replace this with your own application.
# Take care using correct Qt flags.
#
APPLICATION_PATH="/usr/bin"
APPLICATION="StartupLauncherGui"

#
# END USER: To enable VNC, change following value from 0 to 1
#
# There will be performance penalty when VNC is enabled even if
# VNC connection is not open.
#
USE_VNC=0
####

RESOLUTION=$(get_display_resolution)
DEFAULT_RESOLUTION="1024x768"

case "$1" in
start)
    if [ -z $RESOLUTION ] ; then
        RESOLUTION=$DEFAULT_RESOLUTION
    fi

    if check_feature DISPLAY || check_feature EXTDISPLAY; then
        if [ $USE_VNC -ne 0 ]; then
            ( QWS_KEYBOARD="LinuxInput:/dev/kb0 LinuxInput:/dev/kb1 \
LinuxInput:/dev/kb2" \
QWS_MOUSE_PROTO="LinuxInput:/dev/touch0 \
MouseMan:/dev/mouse0" \
$APPLICATION_PATH/$APPLICATION -qws -display "Multi: \
transformed: VNC::size=${RESOLUTION}:depth=32:0" &> \
/dev/null & )
        Else
            ( QWS_KEYBOARD="LinuxInput:/dev/kb0 LinuxInput:/dev/kb1 \
LinuxInput:/dev/kb2" \
QWS_MOUSE_PROTO="LinuxInput:/dev/touch0 \
MouseMan:/dev/mouse0" \
$APPLICATION_PATH/$APPLICATION -qws -display transformed \ &>
/dev/null & )
        fi
    else
        # Display less unit supports only VNC
        if [ $USE_VNC -ne 0 ]; then
            ( QWS_KEYBOARD="LinuxInput:/dev/kb0 LinuxInput:/dev/kb1 \
LinuxInput:/dev/kb2" \
QWS_MOUSE_PROTO="LinuxInput:/dev/touch0 \
```

```
    MouseMan:/dev/mouse0" \  
    $APPLICATION_PATH/$APPLICATION -qws -display \  
    "VNC::size=${RESOLUTION}:depth=32:0" &> /dev/null & )  
    fi  
    fi  
    ;;  
stop)  
    killall $APPLICATION  
    ;;  
*)  
    echo $"Usage: $0 {start|stop}"  
    exit 1  
esac  
  
exit $?
```

If the user wants no startup application to be run during boot, an empty file must be stored in place of the script. If removed completely, the missing script will be replaced by the default on next boot. Note that it is still possible to start applications in parallel or with other run level scripts, as mentioned in 7.1.4 Start-up scripts.

## 7.7. Late load modules

To improve the startup time of the device, a few device drivers are being loaded after the application startup process has begun, which means that the graphical application is started early. This is because some hardware specific modules will take up a large proportion of the time it takes for the device to boot.

This can be changed, if critical functions in drivers are needed earlier, at the expense of slower startup of applications. The user can edit a configuration file to change the behavior. The file is */opt/etc/late-loaded-modules*, and changes are done by commenting out or removing corresponding modules from the file. By default it contains modules for devices like FPGA CAN controllers, video, audio, and USB host controller.

## 7.8. Serial Number Broadcast configuration

The device Linux version can identify itself over the IP network by sending out its serial number as a broadcast IP packet. The Serial Number Broadcast (SNB) service is started by default at the device boot-up and 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 */opt/etc/ccsnb.conf*. 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
```

## 7.9. USB memory installer

If a USB memory is inserted before start up, it is possible to activate a run time hook to enable automatic software execution. For instance, this function is suitable for production time installers, or automated SW updates.

If you add a script (executable file) that's called **cc-auto.sh** (only that) in the root of a USB memory, it will be executed during device startup. The USB memory can be a FAT-formatted one or with other suitable file system, so there is no specific requirement there, but remember that some Windows based editors will leave Windows EOL characters in edited files, including scripts. Such characters may or may not effect the execution of this type of script.

The *cc-auto.sh* script is then executed automatically at insertion of USB memory or during start up. By placing commands which copy new applications and perform updates and installations in that script, this feature can be used for creating auto-update of user and system software. There are no specific limits on what user can do with *cc-auto.sh file*, but it's recommended that all desired commands are applied within the *cc-auto.sh* script process.

### 7.9.1. Example of automatic software installation with USB memory

This is an example that enables operating system update to be automated, but it can be used as an illustration on what can be done with the installer script as such. This script solution is to be used for the backup system solution only, that's not needed in the normal user software update case.

1. *cc-auto.sh* creates folder */usr/local/my\_application/*.
2. *cc-auto.sh* copies all necessary files to that folder or to other folders from USB memory.
3. *cc-auto.sh* performs additional application setup actions.
4. *cc-auto.sh* may run *reboot* but it is not necessary. If *reboot* is written, remember to remove memory when restarting, otherwise the device will enter a reboot loop since the script is also run at startup.

At this point the USB memory can and should be removed, and device should reboot itself and start the installed applications. A small script example is given below:

```
#!/bin/sh
# Automatic installation script example

mkdir -p /opt/my_application
tar -C /opt/my_application -xzf /media/usb/my_application.tar.gz

ldconfig -C /opt/etc/ld.so.cache

sleep 10
```

```
reboot
```

## 7.10. Media files playback

Video and audio playback is supported by gstreamer multimedia framework, which supports multiple video (MPEG2, H.264/AVC, DivX, Xvid) and audio codecs (wav, MP3, AAC) and container formats (MPEG, AVI, mkv). Note that the CCPilot XA currently supports only codecs for WAW audio playback. Each supported kind of video can be played with the command:

```
# gplay <videofile> or  
# aplay <audiofile>
```

This command launches a player with interactive ASCII interface, but plays the file automatically so it can also be launched from a script. For custom applications, gstreamer-0.10 C API can be used (see documentation at <http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/index.html>).

There is also a Qt frontend system available for gstreamer as well, which is called Phonon.

## 7.11. VNC

The XA device also has support for a partial VNC server solution so that a GUI application can be accessed remotely, for instance when using CrossCore XS headless devices, or when a graphical remote environment is desired. The basics for the VNC functionality comes from support in the Qt libraries, and thus it is required that the VNC Server is actually a Qt application designed for that use case.

To access the device via VNC, you will need to use a VNC viewer application and enter the IP address of the XA device. By default, there is no application made available for access through VNC, but it can be enabled to use a default one, or custom made by the end customer.

On the device, the Qt VNC application requires the QWS flag as well as the VNC flag to operate properly as a VNC server. In the default version of the StartupGui startup script (below), a separate Qt VNC session can be started by setting a specific USE\_VNC flag, see the script for details.

Specifically starting a Qt VNC application is done as follows:

```
$/path/to/myapp -qws -display VNC::size=800x480:depth=32:0" &
```



Any application that should be run as the VNC server can be started in the same way in a startup script, but pay attention to the design guides for scripting, not locking the startup sequence.

## 7.12. Removal of user data and files, i.e. factory reset

The XA device normally stores all user data, applications and settings under /usr/local, or /opt which points to the same storage place. It is possible to remove all of the settings and files under that location, and have the XA device generate the default contents back upon a restart of the device.



Note: This can potentially also remove some applications and files that are part of the factory installation by CrossControl and delivered to you as such. If that is the case, please avoid this method of user data and file removal unless specific knowledge about this has been gathered.

The restoration is done via the command

```
# reboot-rescue.sh clear
```

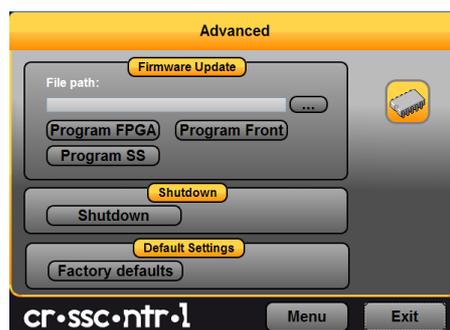
which will reboot the device twice and reformat the entire user partition, as well as restoring the default files that actually are there.

## 8. Software update and recovery

*The images here are for illustration purposes and may not represent the real device user interface*

### 8.1. Restore firmware settings

*CCsettings* can be used to reset the firmware settings to the factory default settings, if needed. Start *CCsettings* and press the *Factory defaults* button from the *Advanced* page.



### 8.2. Updating firmware components

The device can have up to three different firmware components; The System Supervisor is a microcontroller that controls the power configuration within the device. The Front microcontroller controls the functionality available in the front, including LED, Buzzer and touch screen handling. In some device instances, there's an FPGA that implements different communication controller interfaces internally in the device, such as secondary CAN controllers. The FPGA can also be updated programmatically with this tool.

Each of these firmware components can only be updated individually. The preferred method to perform the update is by using the *CCSettings* program, as briefly mention in section 5.15. It is also possible to use the command line based application *ccsettingsconsole*, or from your own program using the CC AUX API functions.

**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, or have internal parts replaced through service interfaces. Make sure that you carefully choose the correct files for updating and follow the instructions from the tools, including powering off the device when prompted.

For specific details on how to perform firmware component updates, please see additional documentation for firmware updates.

### 8.3. Updating operating system

The Linux system on an device 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, or have internal parts replaced through service interfaces.

## 8.4. Released image files

New versions of the operating system for the 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 parts in the SD flash card: main kernel, backup system kernel, backup system root file system, main root file system and user defined area. Additionally, there is boot loader software in the first part of the SD card. Normally, software updates concern only main kernel and root file system, occasionally also the other parts.

## 8.5. Updating from console



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

This method is possible when the device has working backup system.

### 8.5.1. 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 `/usr/local/` 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.

### 8.5.2. Updating device within Linux

**Prerequisite:** New file images shall be located at the `/usr/local` folder on the module, or in other appropriate directory. The files that can be present are normally a subset of these files below, depending on which parts require the update:

- `ccpilot-xa_full.bin`
- `ccpilot-xa_kernel.bin`
- `ccpilot-xa_rootfs.bin`
- `ccpilot-xa_u-boot.bin`
- `ccpilot-xa_mbr.bin`
- `ccpilot-xa.md5sum`
- `fullup.sh`

Update tool command **fullup.sh** should be copied to target explicitly as well.



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

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

**Example output:**

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

## 8.6. 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 method was introduced in 7.9 USB memory installer.

### 8.6.1. Updating automatically

Power-up the device, make sure the login less SSH access is set up as according to 7.4.3. Copy the update image files, *fullup.sh* and *update.sh* scripts to the device into */usr/local/fw\_update/* folder. USB memory or NFS mount can all be used for copying. Then execute **reboot-rescue.sh**

For example:

```
scp -r fw_update -i xa_rsa root@X.X.X.X:/usr/local/  
ssh -i xa_rsa root@X.X.X.X "/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.

### 8.6.2. Example of automatic system software update with USB memory

This is an example that enables operating system update to be automated, but it can be used as an illustration on what can be done with the installer script as such. This script solution is to be used for the backup system solution only, that's not needed in the normal user software update case.

5. *cc-auto.sh* creates folder */usr/local/fw\_update/*.
6. *cc-auto.sh* copies all necessary files to that folder, including file called *update.sh*.
7. *cc-auto.sh* runs *reboot-rescue.sh*.

At this point the USB memory can and should be removed.

8. Device reboots to backup system.
9. Backup system executes */usr/local/fw\_updata/update.sh*.
10. */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.
11. The last thing the script does is to reboot the device again, into normal operating mode.

## 8.7. 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, then update the main side first and then continue here.

### 8.7.1. Preparing update within Linux

Copy the backup system update images as listed:

- *ccpilot-xa-bs\_kernel.bin*,
- *ccpilot-xa-bs\_rootfs.bin*

- `ccpilot-xa-bs.md5sum) a`
- `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.

### 8.7.2. 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.

#### 8.7.2.1. 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.

## Technical support

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

- The part number and serial number of the device, 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.
- Prepare a system report on the device, from within *CCsettings* (if possible).
- Detailed description of all external equipment connected to the unit (when relevant to the problem).

## Trade Mark, etc.

© 2013 CrossControl

All trademarks sighted in this document are the property of their respective owners.

CCpilot is a trademark which is the property of CrossControl.

Intel is a registered trademark which is the property of Intel Corporation in the USA and/or other countries. Linux is a registered trademark of Linus Torvalds. Microsoft and Windows are registered trademarks which belong to Microsoft Corporation in the USA and/or other countries.

CrossControl is not responsible for editing errors, technical errors or for material which has been omitted in this document. CrossControl is not responsible for unintentional damage or for damage which occurs as a result of supplying, handling or using of this material including the devices and software referred to herein. The information in this handbook is supplied without any guarantees and can change without prior notification.

For CrossControl licensed software, CrossControl grants you a license, to under CrossControls intellectual property rights, to use, reproduce, distribute, market and sell the software, only as a part of or integrated within, the devices for which this documentation concerns. Any other usage, such as, but not limited to, reproduction, distribution, marketing, sales and reverse engineer of this documentation, licensed software source code or any other affiliated material may not be performed without written consent of CrossControl.

CrossControl respects the intellectual property of others, and we ask our users to do the same. Where software based on CrossControl software or products is distributed, the software may only be distributed in accordance with the terms and conditions provided by the reproduced licensors.

For end-user license agreements (EULAs), copyright notices, conditions, and disclaimers, regarding certain third-party components used in the device, refer to the copyright notices documentation.