Revision: 1.4.0
2018-11-19

# CC Linux

Software Guide

crosscontrol
www.crosscontrol.com

# Contents

**Revision history** ........ **3**
**1. Introduction** ........ **4**
1.1. Scope ........ 4
1.2. References ........ 4
**2. Basic operation** ........ **4**
2.1. Login and passwords ........ 4
2.2. Using the touch screen ........ 5
2.3. Keyboard ........ 5
2.4. Print screen ........ 5
2.5. Software deployment ........ 5
2.6. Boot loader ........ 6
2.7. Linux system start-up specifics ........ 6
2.8. Status indication (LED and Buzzer) ........ 6
**3. Accessing and using the interfaces** ........ **7**
3.1. Storage and file system ........ 7
3.2. CAN ........ 8
3.3. Ethernet ........ 8
3.4. USB ........ 8
3.5. Video in ........ 8
3.6. Configurable inputs ........ 9
3.7. Analog inputs ........ 9
3.8. Digital Outputs ........ 9
3.9. PWM Outputs ........ 9
3.10. Backlight ........ 9
3.11. Ambient light sensor ........ 9
3.12. Buzzer ........ 9
3.13. Temperature sensors ........ 10
3.14. Buttons ........ 10
3.15. RS232 external serial port ........ 10
3.16. RS485 external serial port ........ 10
**4. Pre-installed applications** ........ **11**
4.1. CCLauncher ........ 11
4.2. CCSettingsConsole ........ 11
4.3. CCSettings ........ 12
4.4. CCvideo ........ 13
4.5. CCMultitouchDemo ........ 13
4.6. CCVNCServer ........ 14
4.7. CCAux daemon ........ 14
**5. Software configuration possibilities** ........ **15**
5.1. Installing new drivers, applications and system packages ........ 15
5.2. Text editor ........ 16
5.3. Terminal ........ 16

5.4. IP address configuration ..... 17
5.5. Remote access ..... 17
5.6. Weston Graphics ..... 18
5.7. Default startup application ..... 19
5.8. Boot splash ..... 20
5.9. Serial Number Broadcast configuration ..... 20
5.10. USB memory installer ..... 20
5.11. Video files playback ..... 21
5.12. Adding user defined fonts to system ..... 21
**6. Software update and recovery ..... 21**
6.1. Restore firmware settings ..... 21
6.2. Updating SS firmware components ..... 21
6.3. Factory reset of operating system ..... 22
6.4. Updating the operating system ..... 22
6.5. Update automation ..... 25
**7. CCpilot VS 12" - features and behavior ..... 26**
7.1. Status indication (LED and Buzzer) ..... 27
**8. CCpilot VI 2nd generation - features and behavior ..... 27**
8.1. Status indication (Backlit soft keys and Buzzer) ..... 27
**Technical support ..... 29**
**Trademarks and terms of use ..... 29**

## Revision history

| Rev | Date | Comments |
|---|---|---|
| 1.4.0 | 2018-11-19 | Released.<br>Document compatible with devices running CC Linux 1.4.x.x |

# 1. Introduction

The devices contained within the CC Linux platform are powerful display computers, communication devices and controllers with a rich set of integrated functions. Together with the CC Linux operating system, they form an open platform that facilitates easy implementation of reliable controls.

## 1.1. Scope

This document is intended for anyone handling a device running the CC Linux platform or developing software for such a device. This document is not intended as a complete reference documentation of the CC Linux devices, software, or software development tools. It is intended to introduce the development engineer to the hardware and software, by providing a top-down overview and summary of the device and a description of the intended use. It is also intended to introduce the reader to the software on the target and the host software development tools.

All devices included in the CC Linux platform (at the time of writing: CCpilot VS 12" and 2nd generation CCpilot VI) are covered within this document. Depending on which CC Linux device model you are using, some features mentioned in this document might be unsupported. If so, it will be clearly stated using a greyed out icon as shown in **Figure 1**. If no icons are used, the feature is supported by all device models.

VS
VI

**1.1. RS232 external serial port**

There is one external RS232 port which can be accessed via /dev/ttyExt0.

***Figure 1:*** *Example of feature supported by CCpilot VS and unsupported by CCpilot VI*

There may be slight differences of supported features depending on your device model. If so, there will be device specific chapters covering the details at the end of this document. Any such occurrence will be clearly stated in the text.

## 1.2. References

[1] CC Linux– Programmer's Guide
[2] CCpilot VS 12" – Technical Manual
[3] CCpilot VI – Technical Manual
[4] CCAux API reference documentation
[5] Gstreamer manual:
http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/index.html

# 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 credentials as stated in Table 1 are required. Additional users can be added using the adduser Linux program.

*Table 1: Default login credentials*

| Username | Password | Access |
|---|---|---|
| root | suseroot | Full administrator |
| ccs | default | Full administrator using sudo |

**These passwords are publicly accessible and should be immediately changed on first boot in order to avoid security breaches. Issue the following commands using either SSH or Weston-terminal from CCLauncher.**

Change password of root user:

```
$ sudo mount -o remount,rw /
$ sudo passwd
```

Change password of ccs user:

```
$ sudo mount -o remount,rw /
$ sudo passwd ccs
```

## 2.2. Using the touch screen

Navigate the start-up screen application using the touch screen with a stylus or finger.

- Tap the screen to perform an action equivalent to a left mouse click.
- Two simultaneous touches can be recognized to implement pan, zoom and pinch, see chapter 4.5 for an example.

## 2.3. Keyboard

The graphical window manager Weston includes a virtual keyboard that can be used by customer applications. This requires special software consideration when developing the applications. None of the pre-installed application utilise this feature, so an external keyboard is needed to interact with them when necessary.

## 2.4. Print screen

It is possible to take screenshots using the weston-screenshooter included in the window manager. To take a screenshot, connect an external keyboard to the device and press Super+S (on Windows keyboards, the Super key is the Windows key). Note that the / directory must be remounted as read-write since the resulting image will end up in the / directory, see chapter 5.1.2.

## 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 a 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.

### 2.5.1. Device start-up behavior

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 (SS). The SS then supplies the main processor and peripherals with power; from there the device execution begins, starting with the boot loader.

## 2.6. 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 then load the Linux kernel into RAM memory. Once finished, the operating system execution takes over.

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 from CrossControl if needed.

## 2.7. Linux system start-up specifics

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

- Main system (normal, default):

  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 system (backup):

  This is only for device updates and recovery use. It contains only basic maintenance tools. Updating the main system should be done from rescue system. Most of the hardware is not accessible in the rescue system.

Both of these modes are a completely separate Linux operating system, each of which is a custom built 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.

## 2.8. Status indication (LED and Buzzer)

The buzzer and/or status LEDs (or backlit soft keys, depending on your device model) will be used to indicate the different running states of the device. For a detailed description, refer to the device specific chapters 7.1 (VS) and 8.1 (VI).

For error indication, all device models behave the same, as described in 2.8.1

### 2.8.1. Error indication

If an error occurs, the device will indicate the type of error by blinking the status LEDs (or backlit soft keys) and beeping the buzzer in different patterns. The device may be restarted by a button or ignition signal and, depending whether or not the error is severe, the device may start normally or go back to error indication.

The reason for fatal error situations 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.

See the *Technical Manual* of your device model for a more complete description of the error indication and a list of possible error codes.

# 3. 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 *CC Linux – Programmer's Guide* documentation for general information regarding software development using the device interfaces.

There may also be additional methods to access the device interfaces than the ones described herein, depending on additional installed software or connected hardware.

## 3.1. Storage and file system

The device uses an eMMC based storage, which is partitioned into protected operating system parts and writable user parts. The file system abstracting the eMMC is ext4.

The eMMC is industrial grade classified and has both static and dynamic wear levelling to prevent a premature aging and to ensure the longest lifetime. Still, eMMC has a limited number of write cycles. It is recommended that the amount of data written to storage is limited within the application. Rather keep information in RAM memory and write larger blocks at one time instead of frequently writing smaller pieces.

There is however 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 in the eMMC file system, since in case of a sudden power loss, the eMMC's writable partition is made write-protected to protect the files from being corrupted. 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 eMMC.

The eMMC is partitioned into two root file systems, 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 5.1. Table 2 and Table 3 show the file system layouts for main and rescue systems respectively.

In both main and rescue mode, 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, the device will remain unmounted.

*Table 2: File system layout for main system*

| Mount point | Mount status | Media |
|---|---|---|
| / | Read only | eMMC |

| | | |
|---|---|---|
| /usr/local or /opt | Read-write | eMMC |
| /media/usbsda1 | Read-write | First USB memory, if available |
| /media/usbsdb1 | 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, which points to the same storage place), partition of the eMMC 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.

*Table 3: File system layout for rescue system*

| Mount point | Mount status | Media |
|---|---|---|
| / | Read only | eMMC |
| /media/usbsda1 | Read-write | First USB memory, if available |
| /media/usbsdb1 | 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 |

## 3.2. CAN

The device has up to four CAN interfaces with user configurable baud rate and frame type accessible from the CCSettingsConsole application. The CAN interfaces can also be accessed with the Linux operating system standard API SocketCAN. More information can be found in the *CC Linux – Programmer's Guide.*

## 3.3. Ethernet

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 chapter 5.4.1.

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

## 3.4. USB

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

## 3.5. Video in

The video-in signal can be accessed and controlled using the CCAux API, in Qt using the QMultimedia framework, or directly using gstreamer. The application CCvideo has been developed to provide an example of how to use the video-in signal with QMultimedia in QML, see chapter 4.3.

Video input can be displayed without any significant CPU performance penalty, in one video window instance.

## 3.6. Configurable inputs

The configurable input channels are available for software developers using the CCAux API. Parts of the functionality available can be viewed or set within CCsettingsConsole for test purposes.

For additional technical details, please see the *Technical Manual* of your device model and the *CCAux API Reference Documentation*.

## 3.7. Analog inputs

The analog input channels are available for software developers using the CCAux API. The voltage reading can be viewed within CCsettingsConsole for test purposes.

For additional technical details, please see the *Technical Manual* of your device model and the *CCAux API Reference Documentation*.

## 3.8. Digital Outputs

The user-settable digital output signals are available for software developers using the CCAux API. The output status can be viewed or set within CCSettingsConsole for test purposes.

For additional technical details, please see the *Technical Manual* of your device model and the *CCAux API Reference Documentation*.

## 3.9. PWM Outputs

The user-settable PWM output signals are available for software developers using the CCAux API. The output settings can be viewed or set within CCSettingsConsole for test purposes.

For additional technical details, please see the *Technical Manual* of your device model and the *CCAux API Reference Documentation*.

## 3.10. Backlight

The device has an adjustable screen backlight intensity level. The backlight functionality can be controlled from CCSettingsConsole and via software using the CCAux API. The most recent setting is always used, it is saved between restarts.

## 3.11. Ambient light sensor

The ambient light sensor measures light levels in front of the device. The ambient light sensor is accessed through the CCAux API. It can also be accessed for diagnostic through CCSettingsConsole.

It is possible to use the ambient light sensor 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 CCSettingsConsole or through the CCAux API.

## 3.12. Buzzer

The device is 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 CCSettingsConsole.

**On some device models, the volume can be considered very loud when standing close to the device. Increment the volume gradually and use appropriate ear protection.**

## 3.13. Temperature sensors

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

## 3.14. Buttons

Depending on your device model there are up to 4 buttons with configurable functionality. The configuration of each button can be set using either CCAux API or CCSettingsConsole. Table 4 lists all available configurations for each device model. For additional details, see your device model's *Technical Manual*.

*Table 4: Available button configurations. Configuration can be set individually for each button.*

| Configuration | Description | Supported on device | |
|---|---|---|---|
| | | VS | VI |
| Only MP action | Button is not handled in SS. The button can be used in user software via the input event API.<br><br>Button events can be read from the device node /dev/input/event2 | No | Yes |
| Start-up trigger | Button is used by SS to trigger a start-up action; powers up the system or wakes up from suspend state (if supported). | Yes | Yes |
| Action trigger | Button is used as an action trigger (available actions listed in Table 5.) It is possible to choose one action for short presses and a different action for long presses. | Yes | Yes |
| Start-up & Action trigger | Button is used as both Start-up and Action trigger. | Yes | Yes |
| BL decrease | Button decreases display backlight intensity. | No | Yes |
| BL increase | Button increases display backlight intensity. | No | Yes |
| BL decrease & start up | Button decreases display backlight intensity and acts as a start-up trigger. | No | Yes |
| BL increase & start up | Button increases display backlight intensity and acts as a start-up trigger. | No | Yes |

*Table 5: Available actions for short/long presses on a button configured as an 'action trigger'. Actions are set globally for all buttons.*

| Action | Supported on device | |
|---|---|---|
| | VS | VI |
| No action | Yes | Yes |
| Shutdown | Yes | Yes |
| Suspend | No | Yes |

## 3.15. RS232 external serial port

There is one external RS232 port which can be accessed via /dev/ttyExt0.

## 3.16. RS485 external serial port

There is one external RS4875 port which can be accessed via /dev/ttyExt1.

# 4. Pre-installed applications

In addition to the standard Linux utilities the device comes pre-installed with a few CrossControl developed applications. The purpose of the pre-installed applications is to provide a quick and easy overview of the device's function and enable easy device control.

## 4.1. CCLauncher

By default, the graphical application CCLauncher is launched when the device is started. This application supports the user with a GUI for simple tasks and information until it can be replaced with the proper user defined application. If you access the terminal from the CClauncher on a VS device the user is granted SU access without the need to login.

The replacement of CCLauncher is described in chapter 5.7.

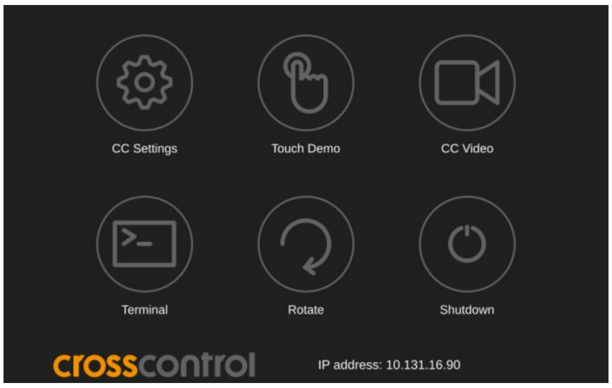

***Figure 2:*** *CCLauncher running on VS*

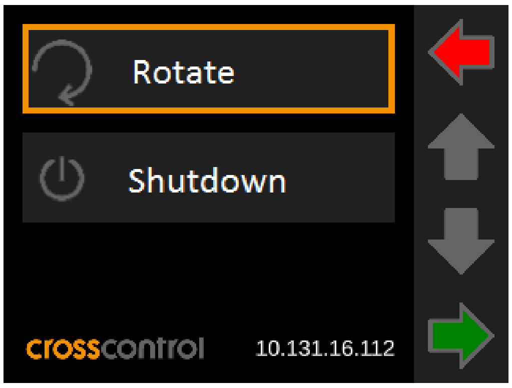

***Figure 3:*** *CCLauncher running on VI*

## 4.2. CCSettingsConsole

CCSettingsConsole is a console application for controlling the device settings (LED and buzzer settings, on/off behavior, etc.) using the CCAux API.

**CCSettingsConsole can be used to alter the way the device starts up or shuts down, update firmware and change important settings. Only change these settings if you know what you are doing.**

CCSettingsConsole can be run either remotely over ssh or locally by connecting a keyboard to the device and opening a terminal from CCLauncher.

To get a description of how to use the application, run the following command:

```
$ sudo ccsettingsconsole --help
```

Settings are grouped into categories like LED, buzzer and Diagnostics, to name a few. The following command is helpful to get a complete list of available categories their possible settings:

```
$ sudo ccsettingsconsole --list
```

For *each category* it is possible to:

- read all current settings at once:

  ```
  $ sudo ccsettingsconsole --category
  ```

- read one setting:

```
$ sudo ccsettingsconsole --category --setting
```

- set one setting:

```
$ sudo ccsettingsconsole –category --setting=value
```

### 4.2.1. Buzzer settings example

As an example, run the following command to get the current buzzer settings:

```
$ sudo ccsettingsconsole --buzzer
Buzzer Frequency: 2600
Buzzer Volume: 400
Buzzer status:  Disabled.
```

And the following to change the buzzer settings to 1000 Hz frequency, volume 200, enabled:

```
$ ccsettingsconsole --buzzer --frequency=1000
Buzzer frequency set to: 1000 Hz.
$ ccsettingsconsole --buzzer --volume=200
Buzzer volume set to: 200
$ ccsettingsconsole --buzzer --status=Enable
Buzzer status set to enable.
```

## 4.3. CCSettings

VS
VI

As a compliment to CCSettingsConsole, there is a graphical version, CCSettings. Features unsupported by VS are greyed out.

!

**CCSettings can be used to alter the way the device starts up or shuts down, update firmware and change important settings. Only change these settings if you know what you are doing.**

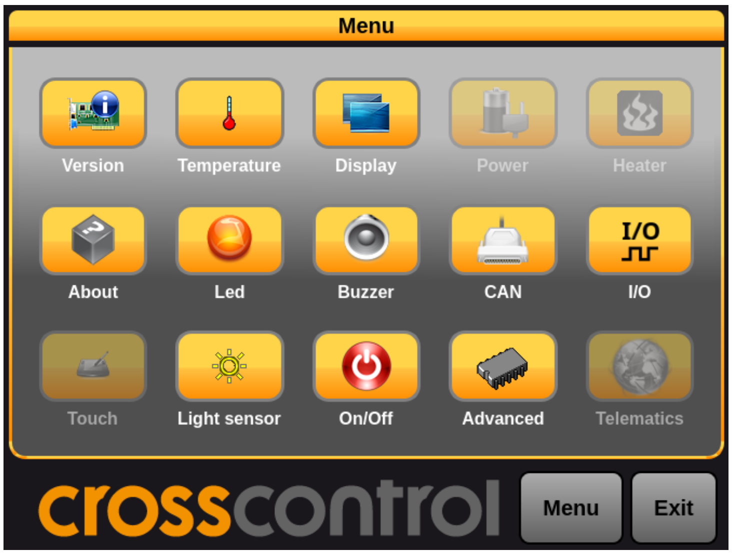

***Figure 4:*** *Main menu of CCsettings application running on VS.*

## 4.4. CCvideo

CCvideo is a pre-installed application for viewing the device video-in signal. The CCvideo application is written in QML using the QMultimedia framework in Qt. Any additional user application can implement video with ease using the same approach.

Additionally, the video stream and settings are available through CCAux API, to be used with non Qt applications or QWidget applications.

### 4.4.1. Using CCVideo

When starting CCVideo the main window will be shown as illustrated. This window displays the video signal, control menu and a video overlay in the form of the CrossControl logo.

The application allows for camera on/off, still image capture, gallery, video rotation and video scaling.

The captured image is stored in /opt/ccvideocaption.jpg and is overwritten upon each new caption in order to save user space.

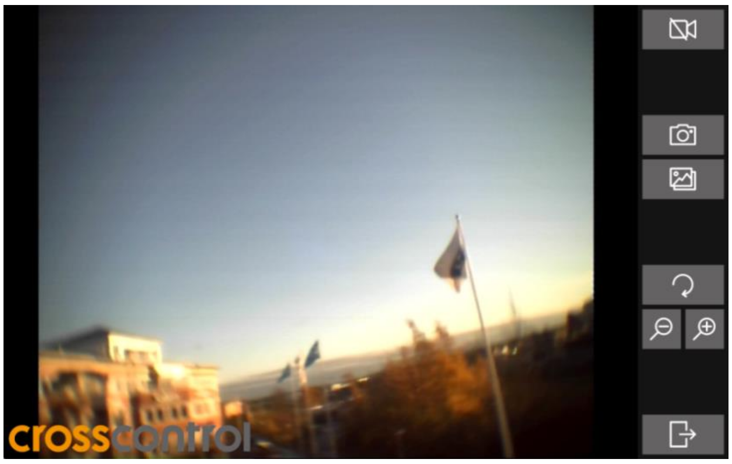

***Figure 5:*** *CCvideo application running on VS*

## 4.5. CCMultitouchDemo

CCMultitouchDemo is an example application showcasing the two-point multitouch capabilities of the device.

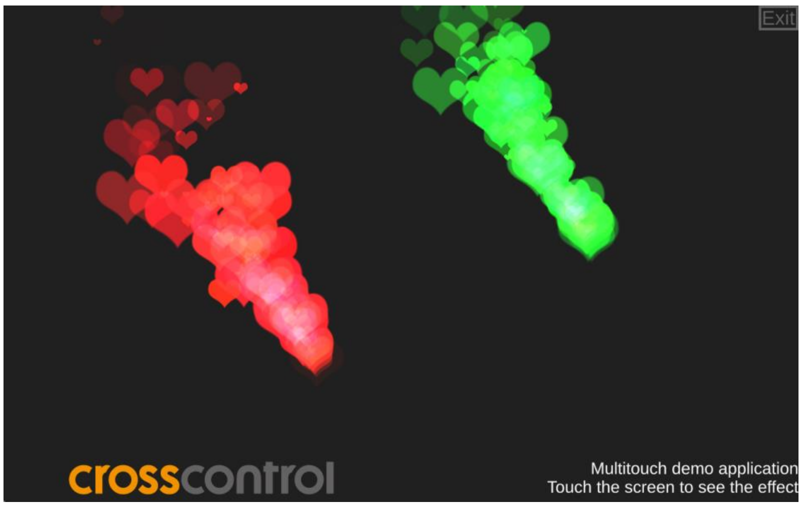
Exit
crosscontrol
Multitouch demo application
Touch the screen to see the effect

***Figure 6:*** *CCMultitouchDemo application running on VS*

## 4.6. CCVNCServer

The image includes the libVNC library for both server and client side. Additionally, a vnc server application is preinstalled on the device, ccvncserver. The application currently supports mouse input only. To use ccvncserver, follow the instructions below.

On the device:

```
$ ccvncserver
```

On the client PC, use vncviewer (or any other preferred VNC client software):

```
$ vncviewer <IP of device>
```

The framebuffer-vncserver can be passed the following (optional) parameters:

| | |
|---|---|
| --passwd= | Set password, default none. |
| --port= | Set port, default 5900. |
| --viewonly | Disable transfer of mouse events from the client to the server. |
| --verbose | Enable more verbose messages |

## 4.7. CCAux daemon

The CCAux daemon (ccauxd) is a background service designed to:

- Read status and event requests from the SS and initiate system restart and shutdown sequences, for example events triggered by button presses.
- Set the button status LED (or backlit soft keys) according to a predefined set of rules.
- Manage automatic backlight levels.
- Implement the PowerMgr feature set, which is configurable from the CCAux API.

The daemon can be removed if necessary functionality is implemented in user software.

# 5. 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.

## 5.1. 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.

### 5.1.1. Mounting user partition in rescue system

Normally, the eMMC user partition is not mounted in the rescue system. If access to the user partition is needed from the rescue system it can be mounted to /opt manually:

```
$ sudo mount /dev/mmcblk0p4 /opt
```

When changes are done, unmount using:

```
$ sudo umount /opt
```

### 5.1.2. 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 /
```

### 5.1.3. 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.

### 5.1.4. 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.

### 5.1.5. 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. See Table 6 for a short description of Linux run levels.

*Table 6: Linux run levels*

| Run level | Mode | Description |
|---|---|---|
| 0 | Halt | Shutdown |
| 1 | Single-user | Administrative tasks |
| 2 | Multi-user | Does not configure network interfaces nor export network services |
| 3 | Multi-user with networking | Start the system normally |
| 4 | Undefined | Not used/User-definable |
| 5 | GUI | Same as runlevel 3 + display manager |
| 6 | Reboot | Reboots the system |

The default run level is 5, 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 5, create a script located in /opt/etc/rc5.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 run command 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 5.

## 5.2. Text editor

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

## 5.3. Terminal

It is possible to access a local device by connecting an external keyboard to the device and opening an on screen terminal by pressing down ctrl+alt+F4 keys. The login credentials are stated in chapter 2.1.

## 5.4. 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.

### 5.4.1. File method for IP address configuration

The network interfaces file is located in read-only storage but is linked to a file in the writable /opt partition so that it can be edited. 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 static IP address:

```
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
```

## 5.5. Remote access

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

### 5.5.1. SSH

To connect to the device from a host, 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.

**Please note that root access over SSH is disabled.**

Access can be re-enabled using the SUDO command. The default login and password can be found in chapter 2.

### 5.5.2. SCP

To copy a file to the device (while on the host) use the following command (and give password when asked):

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

To copy a file from the device (while on the host) use the following command (and give password when asked):

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

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

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

To copy a file from the host (while on the device), 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.

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

Even though the ccs user is password protected, SSH-connections can be configured to connect without password, using identity files. This method is mainly useful for remotely executed scripts or similar.

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

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

Copy (or append) the created *vs_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 *vs_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/vs_rsa ccs@X.X.X.X
```

### 5.5.4. 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.

## 5.6. Weston Graphics

The graphics framework uses the Wayland protocol reference implementation Weston for graphics operations. Wayland is fast and efficient, and is used by all major Linux desktop systems, giving it vast standard support in the Linux user space world.

For instance, Qt has a plugin that enables the Qt libraries to be built for Weston. For Qt applications the impact for that means that it simply needs to be started with a specific flag, -platform wayland-egl, and built with the correct development libraries. In *CC Linux*, this flag has been set to the default graphics framework, hence there's no need to pass the flag.

Weston includes a windowing system, enabling several applications to be run overlapped.

### 5.6.1. Graphical application launch

In order to launch a graphical application on Weston, the XDG_RUNTIME_DIR environment variable needs to be set. This can easily be done by either adding the following line to the application's startup script, or running it from the command line before launching the application:

```
export XDG_RUNTIME_DIR=/run/user/root
```

### 5.6.2. Fixed window position

By default, Weston will place any opened window at a random position. In order to overrun this feature, Weston has been patched to read window coordinates from the file /tmp/weston_fixed_coordinates. If it doesn't exist, or if coordinates provided are negative, it will revert to using the default random positioning. The file needs to be overwritten with new coordinates for each window to be opened at a new position. A typical use case for this feature is showing two windows split screen. Following is an example script which opens two application windows; a pdf-reader at position (0,0) and a text editor at position (640,0):

```
#!/bin/sh

echo "0 0" > /tmp/weston_fixed_coordinates
/usr/bin/qpdfview &

sleep 5

echo "640 0" > /tmp/weston_fixed_coordinates
/usr/bin/weston-editor &
```

## 5.7. Default startup application

As mentioned in chapter 4.1, the device has a GUI application that is started by default. A user can (and should) disable or simply edit the default startup application script, i.e. the init script found at /opt/etc/rc5.d/S10cclauncher. This init script is automatically run when entering run level 5 (during boot) or run level 6 (during shutdown/reboot).

Note: This script is part of the rc script solution mentioned in chapter 5.1.5. Make sure the correct script behavior is met; otherwise it can break the device startup procedure.

To disable the default startup application init script, do the following:

```
Remove the script:
$ sudo rm -f /opt/etc/rc5.d/S10cclauncher

Replace it with an empty script
$ sudo touch /opt/etc/S10cclauncher
```

Simply removing the script will not work as the system automatically creates the script if it does not already exist.

## 5.8. Boot splash

*CC Linux* comes with a default boot splash screen containing a black background with a white progress bar. The boot splash used is called psplash; third party software released under the GPL2 license. If desired, it is possible to add a picture/logo and change the colors of the boot splash. In order to do so, the psplash source code needs to be modified and the binaries recompiled. Please refer to the document *CC Linux - change boot splash appearance* which can be found at the Knowledge base on the CrossControl support site.

## 5.9. Serial Number Broadcast configuration

The device 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 frequency of the broadcast 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=5

# 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, if unset '0' is used. String value
UNITTYPE=VS
```

## 5.10. USB memory installer

If a USB memory is inserted after 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, see chapter 6.5.

To activate the automatic execution, add a script named cc-auto.sh to the root of a FAT16/32 formatted USB memory. CC Linux is configured to automatically run any script with this name at insertion of USB memory. 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 is recommended that all desired commands are applied within the cc-auto.sh script process.

Note: some Windows based editors will leave Windows EOL characters in edited files, including scripts. Such characters may or may not affect the execution of this type of script.

The cc-auto.sh script automatically gets an argument from the OS containing the path to where the USB memory is mounted. Never use absolute paths to the USB memory from within cc-auto.sh as the path may vary.

### 5.11. Video files playback

Video file playback is supported by the gstreamer multimedia framework, which supports multiple video (MPEG2, H.264/AVC, DivX, Xvid) and container formats (MPEG, AVI, mkv). Each supported kind of video can be played with the command:

```
$ gst-launch-1.0 playbin uri=file://path-to-file
```

This command can also be used to play video files over a network and plays the file automatically so it can be launched from a script. For custom applications, gstreamer-1.0 C API can also be used, see documentation at [5].

http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/index.html.

### 5.12. Adding user defined fonts to system

CC Linux uses fontconfig to handle the installed fonts. Pre-installed fonts reside in /usr/share/fonts/ttf. New fonts can be added, preferably in a /opt/fonts directory. The new fonts need to be added to fontconfig using:

```
$ fc-cache </opt/fonts/>
```

It is possible to list all the installed fonts by running:

```
$ fc-list
```

# 6. Software update and recovery

## 6.1. Restore firmware settings

CCSettingsConsole can be used to reset the firmware settings to the factory default settings, if needed. Use the following command:

```
$ sudo ccsettingsconsole --advanced --factory
```

## 6.2. Updating SS firmware components

The *advanced* category of CCSettingsConsole is also used for loading new SS firmware into the device. To update the SS firmware, copy the firmware file to a USB stick and connect it to the device. Reboot the device to rescue system using the following command:

```
$ sudo reboot-rescue.sh
```

Login and issue the following command to update SS:

```
$ sudo ccsettingsconsole --advanced --update=SS --filepath=<full-path-to-file>
```

If only firmware verification is wanted, use the following command instead:

```
$ sudo ccsettingsconsole --advanced --verify=SS --filepath=<full-path-to-file>
```

If the file matches the current SS firmware, it is reported. The firmware is not written during verify. After each program or verify action, the device must be shut down. Press the shutdown button or other alternatives to shut down the OS after a firmware update.

The preferred method to perform the update is by using the CCSettingsConsole application. It is also possible to use the API directly from your own program using the CCAux API functions.

**Warning:** Errors during an update can set the module in an unrecoverable state. In such case, the module must be 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.

## 6.3. Factory reset of operating system

The device normally stores all user data, applications and settings under /usr/local, (or /opt which points to the same storage location). It is possible to remove all of the settings and files under that location, and have the 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

```
$ sudo reboot-rescue.sh clear
```

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

## 6.4. Updating the operating system

The CC Linux system on the 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 our factory for repair, or have internal parts replaced through service partners.

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 eMMC: main kernel, rescue system kernel, main root file system, rescue system root file system, and user defined area. Additionally, there is boot loader software in the first part of the eMMC. Normally, software updates concern only main kernel and root file system, though they may occasionally concern the other partitions.

Note: Always update the **main system first**, and **the rescue system second**. Alternatively, in order to update both systems at once, CrossControl recommends using the MfgTool2 approach described in *CC Linux – update using MfgTool2*.

File names will differ depending on which parts are being updated. Additionally, file names will vary depending on your device model. In the below sections, CCpilot VS 12” will be used as an example.

### 6.4.1. Updating main system

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

This method requires the device to have a working rescue system.

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

Follow these steps to update the main system:

1. Copy the main system update images to /usr/local/fw_update. **The folder name is important, otherwise, the update will not start**. Copy method choice is free; scp, USB memory or NFS mount can all be used.
    - ccpilot-vs_kernel.bin
    - ccpilot-vs_rootfs.bin
    - ccpilot-vs_u-boot.bin
    - ccpilot-vs.md5sum
    - fullup.sh
2. Reboot the device to rescue system:

```
$ sudo reboot-rescue.sh
```

3. Wait for the update to automatically start and reboot to main system when complete.

Example output on the device on-screen terminal:

```
Please wait: booting…

Recovery system vs/dev/tty1
vs login: ===== CrossControl: Device Update ===== AF
    System Hardware: ccpilot-vs
 Now running on      : BACKUP system
   Requested action: NORMAL system update
        Checking MD5: ccpilot-vs_kernel.bin: ccpilot_kernel.bin= OK
OK
        Checking MD5: ccpilot-vs_rootfs.bin: ccpilot_rootfs.bin= OK
OK

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

             Verification: Skipped, continuing..

       === Now doing: Requested actions ===
 * DO NOT BREAK PROCESS OR SHUTDOWN THE UNIT before update is complete.
*

Copying new kernel image..
Copying new root-fs image..
```

### 6.4.2. Updating rescue system

**Note:** Updating only the rescue system does not affect the main system. When the rescue system requires an update, it is updated from the main system side. If the main side is non-operational; update the main side first and continue to this step.

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

Follow these steps to update the rescue system:

1. Copy the backup system update images to /usr/local/folder (folder name has no importance when updating rescue system). Copy method choice is free; scp, USB memory or NFS mount can all be used.
    - ccpilot-vs_rescue_kernel.bin
    - ccpilot-vs_rescue_rootfs.bin
    - ccpilot-vs_rescue.md5sum
    - fullup.sh
2. Access the Linux console, either over SSH connection from another host, or using a serial console terminal access to Linux.

   **Warning:** Avoid using the SSH method, if your Ethernet network is susceptible to connection breaks, as the image write can get interrupted.
3. Update the rescue system from within the update folder:

   ```
   /usr/local/folder$ sudo ./fullup.sh -s
   ```

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

Example output on terminal:

```
===== CrossControl: Device Update ===== AF
   System Hardware: ccpilot-vs
 Now running on   : normal side
  Requested action: BACKUP system update
      Checking MD5: ccpilot-vs_rescue_kernel.bin: ccpilot-vs_rescue_kernel.bin:
OK
OK
      Checking MD5: ccpilot-vs_rescue_rootfs.bin: ccpilot-vs_rescue_rootfs.bin:
OK
OK


       === Listing: Actions: ===
          Updating: kernel
```

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

   === Verification: ARE YOU SURE ? ===

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

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

The process is pausing here, waiting for 'Enter' or cancellation.

```
    === Now doing: Requested actions ===

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

Copying new kernel image..
Copying new root-fs image..
131072+0 records in
131072+0 records out
67108864 bytes (67 MB, 64 MiB) copied, 14.0712 s, 4.8 MB/s
Completed. Rebooting..

Broadcast message from root@vs (pts/0) (Thu Nov  9 15:02:02 2017):

The system is going down for reboot NOW!

End of all actions, H o l d   o n   t i g h t
```

## 6.5. Update automation

This chapter shows how to use the auto started script cc-auto.sh to automate the update. In these examples the update process is automated to start upon insertion of a USB memory stick.

These examples can be modified in order to use the cc-auto.sh script to update the system remotely over SSH.

### 6.5.1. Automated update of main system using USB memory

The directory name fw_update must be kept for the update to work properly.

1. Create a directory fw_update in the root directory of the USB memory and copy the update images and fullup.sh script there.
2. Add a script cc-auto.sh to the root directory of the USB memory. Note that the script name must be correct in order for the update to start**.** An example of the script contents:

```
echo "Starting release update"

#
# To prevent accidental updates, use this stamp file for it.
# Remove file so if user inserts USB - stick second time the
# update will be started regardless.
#
if [ -e $1/update-done ] ; then
   rm $1/update-done
   exit 0
fi

touch $1/update-done

cd $1 && cp -a fw_update /opt

reboot-rescue.sh
```

3. Insert the USB stick to the device and the update script will start automatically, without requiring any user interaction to complete.

### 6.5.2. Automated update of rescue system using USB memory

For the rescue system update, the directory name can be chosen freely, but must match the directory in the created script.

1. Create a directory foldername in the root directory of the USB memory and copy the update images and fullup.sh script there
2. Add a script cc-auto.sh to the root directory of the USB memory. Note that the script name must be correct in order for the update to start. An example of the script contents:

```
echo "Starting rescue update"
#
# To prevent accidental updates, use this stamp file for it.
# Remove file so if user inserts USB - stick second time the
# update will be started regardless.
#
if [ -e $1/update-done ] ; then
   rm $1/update-done
   exit 0
fi

touch $1/update-done

cd $1/foldername && ./fullup.sh -f -s
```

3. Insert the USB stick to the device and the update script will start automatically, without requiring any further user interaction to complete.

# 7. CCpilot VS 12” - features and behavior

This chapter covers features and behavior unique to the CCpilot VS 12”.

## 7.1. Status indication (LED and Buzzer)

This section describes the basic default status indication behavior. Note that the status indication behavior can be disabled totally and/or configured by user software through the CCAux API.

### 7.1.1. Startup sequence - main system

During startup to the main system, the status LED (embedded in the power button) and buzzer indications are as follows:

- Status LED is blinking yellow at 2 Hz and buzzer makes short beep: Device start-up phase begins.
- Operating system is then started, and specific service software begins to execute.
- Status LED is constant green: Device is operational.

### 7.1.2. Startup sequence - rescue system

During startup to the rescue system, the status LED and buzzer indications are as follows:

- Status LED is constant orange and buzzer makes short beep: Device start-up phase begins.
- Operating system is then started, and specific service software begins to execute.
- Status LED is blinking green at 2 Hz: Device is operational.

### 7.1.3. Shutdown sequence

Once shutdown (power button is pressed, or ignition signal released) is initiated:

- Status LED is blinking blue 2 Hz and buzzer makes short beep: shutdown sequence has started.
- Status LED is off: Device is off

# 8. CCpilot VI 2nd generation - features and behavior

This chapter covers features and behavior unique to the CCpilot VI 2nd generation.

## 8.1. Status indication (Backlit soft keys and Buzzer)

This section describes the basic default status indication behavior. Note that the backlit soft keys status indication behavior can be disabled totally and/or configured by user software through the CCAux API. Buzzer indication is currently non-configurable.

### 8.1.1. Startup sequence - main system

During startup to the main system, the status backlit soft keys and buzzer indications are as follows:

- All 4 backlit soft keys are set to 13 % intensity, blinking at 2 Hz and buzzer makes short beep: Device start-up phase begins.
- Operating system is then started, and specific service software begins to execute.
- All 4 backlit soft keys are constant on, 13 % intensity: Device is operational.

### 8.1.2. Startup sequence - rescue system

During startup to the rescue system, the backlit soft keys and buzzer indications are as follows:

- The 2 middle backlit soft keys are set to 50 % intensity, blinking at 0.5 Hz and buzzer makes short beep: Device start-up phase begins.
- Operating system is then started, and specific service software begins to execute.
- All 4 backlit soft keys are set to 100 % intensity, blinking at 2 Hz: Device is operational.

### 8.1.3. Suspend

When device is in suspend mode, the backlit soft keys and buzzer indications are as follows:

- All 4 backlit soft keys are set to 13 % intensity, blinking at 0.2 Hz: System is suspended
- Resume from suspended state by pressing a configured button or the external on/off signal. All 4 backlit soft keys are set to 100 % intensity, blinking at 2 Hz: Device is operational.

### 8.1.4. Shutdown sequence

Once shutdown (power button is pressed, or ignition signal released) is initiated:

- All 4 backlit soft keys are set to 13 % intensity, blinking at 2 Hz and buzzer makes short beep: shutdown sequence has started.
- Backlit soft keys are off: Device is off

## 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 can be found on the brand label.
- Date of purchase, which can be found on the invoice.
- The conditions and circumstances under which the problem arises.
- Status indicator patterns (i.e. LED or backlit soft key blink pattern).
- Prepare a system report on the device, using CCSettingsConsole (if possible).
- Detailed description of all external equipment connected to the unit (when relevant to the problem).

## Trademarks and terms of use