Document revision: B
2018-11-02

CCpilot VS 12”

Quick Start Guide

crosscontrol

www.crosscontrol.com

Contents

Revision history

Rev	Date	Author	Comments
A	2017-11-03		Released
B	2018-11-02		Fix typo in command on how to boot to rescue system.

1. Introduction

The CCpilot VS 12" is a powerful Arm®-based on-board display computer and controller with a rich set of integrated functions. With its powerful Freescale iMX6 Quad CPU and Linux operating system, it is an open platform that facilitates easy implementation of reliable controls.

1.1. Scope

This document is intended for anyone handling the CCpilot VS device or developing software for the CCpilot VS. This document is not intended as a complete reference documentation of the CCpilot VS 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 CCpilot VS 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.

1.2. List of abbreviations

BSP	Board Support Package
DT	Deutsche
MP	i.MX6Q Main Processor running Linux
SDK	Software Development Kit
SS	STM32 System supervisor co-processor
VS	CCpilot VS

1.3. References

[1] CCpilot VS 12"– Programmer's Guide

[2] CCpilot VS 12" – Software Guide

[3] CCpilot VS 12" – Technical Manual

2. About the CCpilot VS 12”

CCpilot VS is a display computer with a 12” wide high-resolution TFT and PCAP touch screen. The strong LED backlight in combination with the optically bonded PCAP, results in excellent sunlight readability properties.

The powerful Arm® based main CPU (MP) and Linux® operating system constitutes an open platform that facilitates the implementation of premium user-machine interaction, reliable controls and integrated fleet management. In addition, there is a STM32 co-processor (SS) responsible for hardware control and supervision.

2.1. Device specifications

The following features are available in the CCpilot VS:

- i.MX6 Quad processor 1 GHz
- 2 GB DDR3 SDRAM, 800 MHz
- 8 GB on-board, industrial grade eMMC flash memory
- Programmable STM32 coprocessor for system supervision and control purposes
- Configurable RGB status LED
- Integrated high-performance buzzer
- Backup power capacitors for emergency power-loss handling
- Four CAN (multipin to DE-9) with hardware watchdog functionality
- Analog video input
- Two RS232 UARTs, RS485 UART
- 12.1” LCD display with 24-bit LVDS, adjustable backlight and integrated PCAP touch panel
- Two USB 2.0 (M12), one USB 2.0 (type A) and one USB OTG (type A) connectors
- Gigabit Ethernet connector (M12)
- CCLinux reference image pre-loaded on eMMC
- CCAux API and drivers for board-specific functions
- JTAG and serial port access via update board (not included)

2.2. Display layout and connectors

The unit connectors are shown in Figure 1 and Figure 2, with indicators displaying the location of important board features. Refer to these figures to locate connectors and components referred to by later chapters.

Figure 1: Left side and front components

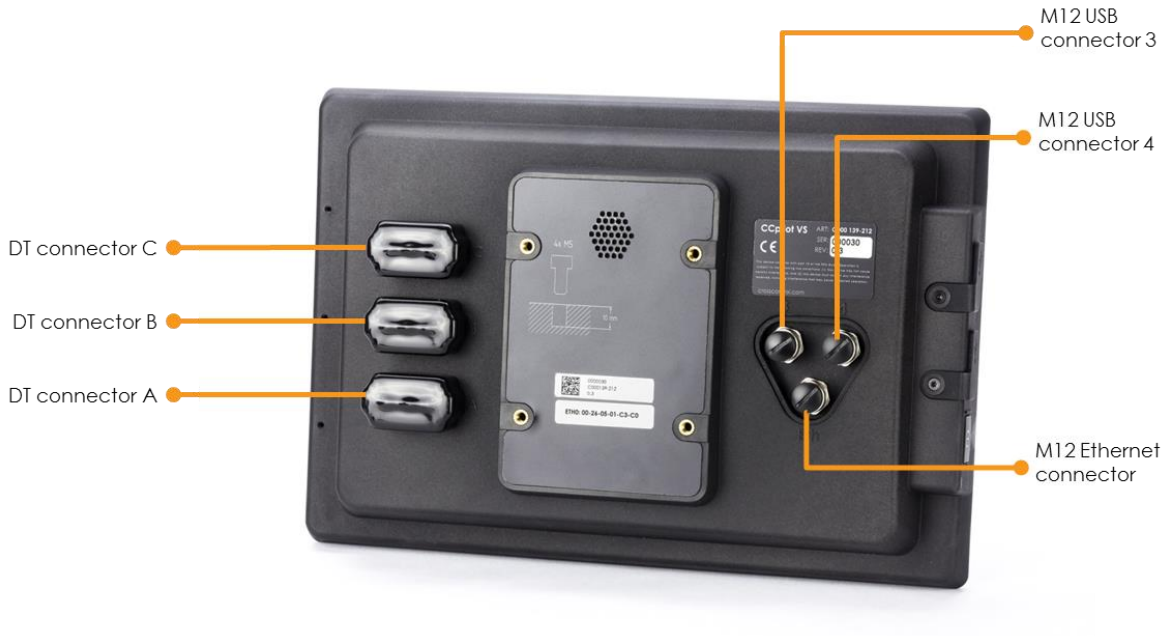


Figure 2: Rear connectors

2.3. Cables and peripherals

The CCpilot VS is available with all necessary cables required to operate the device and the interfaces. This section gives a brief summary of cable and connector functionality. For a detailed description of the pinout of each cable, please refer to *CCpilot VS – Technical Manual*.

2.3.1. DT multipin connector cables

There are three DT 12-pin connectors on the device. The cables are named A, B, and C, color coded according to Table 1. For easy installation, all connectors on the cables are marked with their respective signal names. Figure 3 shows DT cable B as an example.

Figure 3: DT cable B

Table 1: DT connector summary

DT A	Main power input, ground and shield, high-side outputs 1 and 2, CAN 1 and 2 (DE-9).
DT B	CAN 3 and 4 (DE-9). A number of pins for configurable inputs.
DT C	RS232 and RS485 (DE-9), video input, a number of pins for high sideoutputs

M12 connector cables

There are three M12 connectors on the rear side of the device, of which two are USB 2.0 and one is gigabit Ethernet, shown in Figure 4.

Figure 4: M12 gigabit Ethernet cable

2.3.2. Side button and interfaces

The left side of the device holds a software-configurable RGB color LED, a software-configurable power button and two USB type A ports (of which one is OTG, see Figure 1).

2.3.3. Light sensor

In the bottom left corner on the front of the device there is a light sensor for automatic display backlight control.

2.3.4. Touch screen

The device comes with 12.1" TFT display and PCAP touch panel. The display is powered through the backlight connector, while the display signal is transmitted through the LVDS connector. Touch panel data is sent via a serial port connected to UART5 (/dev/ttymxc4). The touch driver is included in the CCpilot VS BSP.

2.3.5. Update board and serial adapters

An optional update board provides serial port access to the MP, and JTAG and serial port connection to the SS for development and service purposes. The board and its connectors in use for the CCpilot VS board are displayed in Figure 5 and Figure 6.

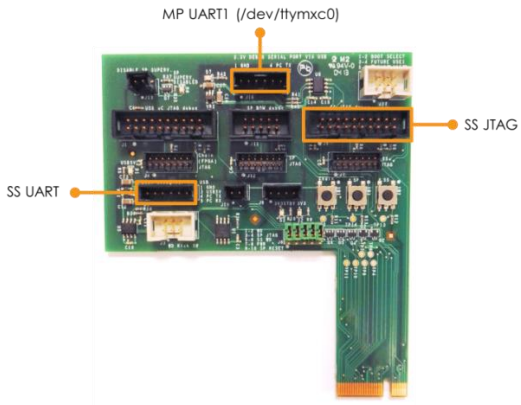


Figure 5: Optional update board with JTAG and UART connectors highlighted.

Figure 6: Optional TTL 232R cable for UART serial port access.

JTAG programming is performed using industry-standard 20-pin flat JTAG cables directly to the card. UART access is provided through six pin mini connectors which require a USB-UART converter shown in Figure 6. The USB converters require device drivers for the host computer that can be downloaded from http://www.ftdichip.com/.

2.4. Linux reference image

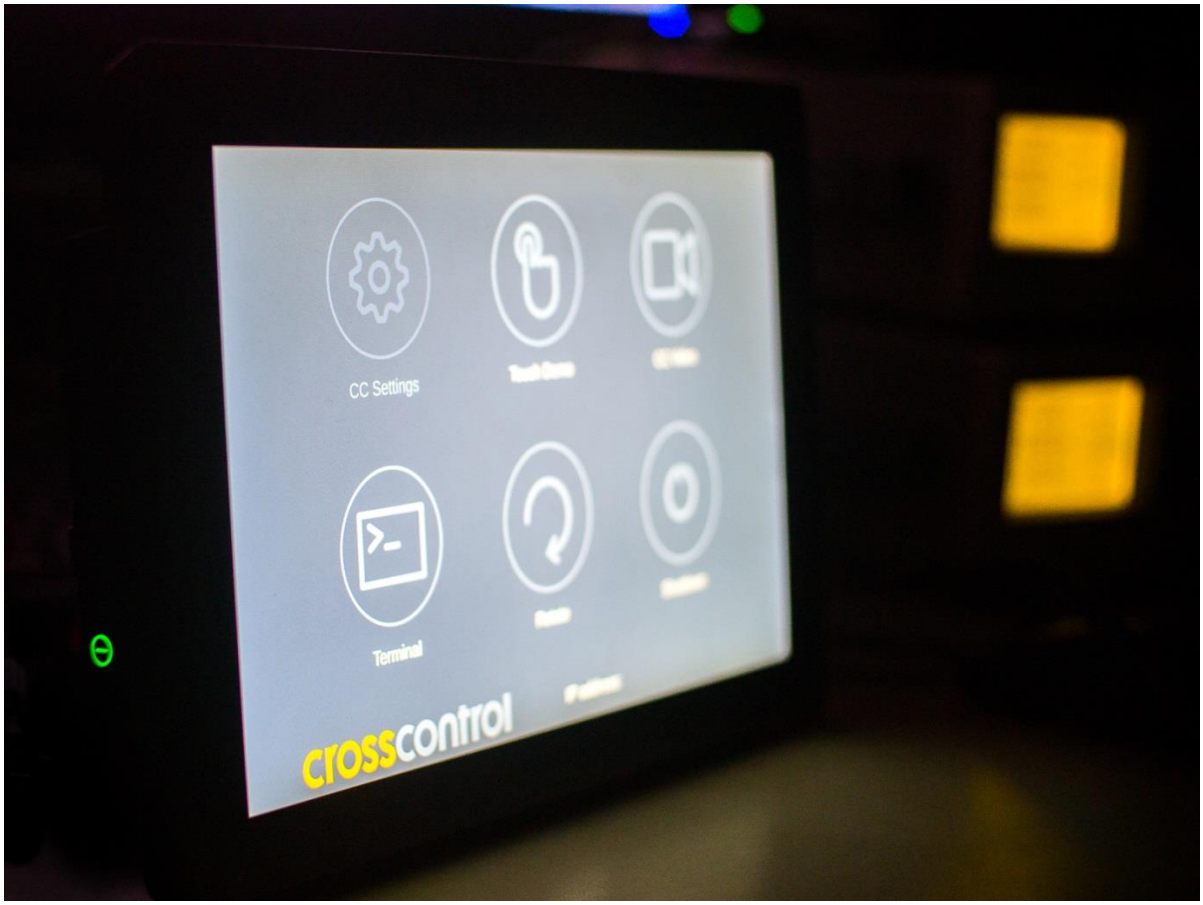
The CCpilot VS comes pre-programmed with a reference implementation of the CCLinux operating system platform (*image*). Please see the *CCpilot VS – Software Guide* for a detailed description of its contents and usage.

CCLinux contains the main operating system with drivers for all peripherals, graphics, and space for user applications. Additionally, the image contains a backup system called the *rescue system*. It is a separate Linux system with a separated file system, used for system recovery and update. The rescue system does not have any graphics and the amount of pre-installed software is limited. In normal operation, only the *main system* will be running.

In addition to the standard Linux utilities, CCLinux contains some CrossControl developed applications mentioned in the following sections.

2.4.1. CCLauncher

By default, CCLauncher is started when the device is started. This application provides the user with a GUI for opening the CrossControl developed applications TouchDemo and CCVideo. It also allows the user to perform simple tasks such as opening a terminal, rotating the screen and shutting down the device.



2.4.2. CCSettingsConsole

CCSettingsConsole is a console application for controlling the device settings (LED color, buzzer settings, on/off behavior, etc.) using the CCAux API. See the *CCpilot VS – Software Guide* for a more detailed list of all possible settings.

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.

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

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

The following command is helpful to get a complete list of possible settings:

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

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

```
$ sudo ccsettingsconsole --buzzer
Buzzer Frequency: 2600
Buzzer Volume: 100
Buzzer status:  Disabled.
```

And the following to change the buzzer settings to 800 Hz frequency, volume 100, enabled:

```
$ sudo ccsettingsconsole --buzzer --frequency=800
$ sudo ccsettingsconsole –buzzer –volume=100
$ sudo ccsettingsconsole --buzzer --status=Enable
```

Volume setting goes up 2000. However, at 1000 the volume can be considered very loud when standing close to the device. Increment the volume gradually and use appropriate ear protection.

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

3. Setting up the device

This chapter gives an overview on how to operate and configure the device, given that new software is available for programming or is pre-loaded on the boards.

3.1. Connecting power

To power the board, connect the following signals on the DT A connector to a stable power supply:

- “+12V In”, “Key Switch State (Ignition)” to +12V
- “Ground” to ground

Note that the board is designed to operate on +12 V, but also works with +24 V.

It is recommended to use a battery or a power supply which can deliver up to 10 A during short periods. If the power supply is too weak, the unit will fail to start and signal an error code with blue LED blinks.

The key switch signal acts as an on/off signal, which the System Supervisor uses in order to generate power-on or power-off events. It is configurable through the CCAux API.

3.2. Communicating with board components

This section describes how to interact with MP and SS.

3.2.1. Main processor serial interfaces

After boot, login via SSH is enabled over Ethernet and TCP/IP. To log into the Linux reference image (normal and rescue), specify login credentials from Table 2.

Table 2: Default login credentials for SSH, rescue system's on screen terminal, and main system's RS232 interface.

Username	Password	SSH access
root	suseroot	No
ccs	default	Yes

These passwords are publicly accessible and should be immediately changed on first boot in order to avoid security breaches. Issue the following commands:

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
```

3.2.1.1. Main system

In the main system, the main processor (MP) serial interface UART1 is available on the RS232 interface using baudrate 115200, 8 data bits, 1 stop bit, no bit parity and XON/XOFF flow control enabled. See Table 2 for login credentials.

The RS232 interface is accessible only on the optional update board.

3.2.1.2. Rescue system

In the rescue system, it is possible to login to the devices on screen terminal by connecting a keyboard. See Table 2 for login credentials.

3.2.2. System Supervisor serial interface

A serial connection is available to the System Supervisor (SS) via the optional update board as described in section 2.3.5. The application is preloaded and has a serial console on a RS232 interface using baudrate 19200, 8 data bits, 1 stop bit, no bit parity and XON/XOFF flow control enabled.

Under normal circumstances, access to the SS console should not be a necessity, but could provide useful information about system and board diagnostics in cases of hardware troubleshooting. To log into the SS console, type password "123" and follow the instructions. Note that the SS application will stop performing most of its normal functions while in console mode and may not respond to I/O calls on the MP-SS SPI link. Exit console mode by typing 'x'.

4. Software and development tools

This section briefly covers how to build the CCLinux reference image from the BSP, how to unpack and use the SDK and how to program the device. For more details, see *CCpilot VS – Programmer's Guide*.

4.1. Yocto board support package

In order to create software for the CCpilot VS, a board support package (BSP) has been created. The BSP is a Yocto-project build system that produces complete Linux images for the CCpilot VS device. It also includes necessary application and driver code, as well as example or template code that may serve as a basis for further application and driver development.

The open-source Yocto system has built-in package support with many thousands of maintained packages available, while also providing a set of standard tools and build guidelines. The BSP adds necessary drivers and applications required for the CCpilot VS board images. For more information regarding the Yocto project, please refer to the Yocto Project reference manual at: http://www.yoctoproject.org/docs/2.2/dev-manual/dev-manual.html

VS is the internal CrossControl code name for the CCpilot VS architecture/platform; therefore the BSP contains references to the VS platform such as "PLATFORM_VS".

In the same way, CrossControl specific components are named "cc", such as "meta-cc" and "recipes-cc".

4.1.1. Building the reference Linux image

The BSP package must be installed to a Linux host machine/build server. The BSP was created on an x86-64 machine running Ubuntu 14.04 LTS. Much disk space (at least several hundred gigabytes) and memory is required for the Yocto system, as the folder structure tends to grow quickly in size.

Out of the box, the VS BSP produces a Linux image containing a reference implementation, as described in section 2.4. The reference implementation is intended to be used as a basis for creating custom images.

The recipes for the Linux reference image are located at:

meta-cc/recipes-core/images/

The main system and rescue system have separate recipes; ccpilot-vs-image.bb and ccpilot-vs-rescue.bb respectively. See section 4.3.1 for a description on how to build the main and rescue systems using the recipes.

The first time the image is built; the bitbake system will download and cache all necessary packages, tools and source code. The resulting image files are located at platform/vs/.

4.2. SDK

The SDK must be installed to a Linux host machine/build server. It was created on an x86-64 machine running Ubuntu 14.04 LTS and comes in the format of a self-extracting shell script.

To extract the script to a specific directory, run:

```
$ sh scriptname.sh -d <directory>
```

Omitting the -d flag will install the SDK to the default directory, /opt/poky/2.2.1.

4.3. Programming the board components

This section describes how to program the MP and SS when new software has been constructed or is otherwise available.

4.3.1. Programming a Linux image to on-board eMMC

By default, the bootloader (u-boot) in the CCLinux reference image runs and boots from the on-board eMMC image. There are two ways of programming the Linux image, described in the following sections.

The main system and rescue system are programmed in separate steps, meaning that one can be updated while the other is kept the same.

To build update files for the main system, type:

```
$ make vs-update-image
```

To build update files for the rescue image, type:

```
$ make vs-rescue-image
```

4.3.1.1. Manual update of main system

To update the main system, do the following:

1. Unpack the update tarball for the main system to the /opt directory of the device. (Tarball should be on the format ccpilot-vs-linux-update-v1.0.0.0.tar.gz)
2. Reboot the device to rescue system:

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

3. The device will start the update automatically and reboot when completed.

4.3.1.2. Manual update of rescue system

To update the rescue system, do the following:

1. Unpack the update tarball for the rescue system to the /opt directory of the device. (Tarball should be on the format ccpilot-vs-linux-rescue-update-v1.0.0.0.tar.gz)
2. Move to the extracted folder (/opt/rescue_update) and run the update script:

```
$ sudo ./fullup.sh -s
```

3. The device will start the update automatically and reboot when completed.

4.3.1.3. Automated update with USB stick

This method updates the device automatically and works the same way when updating main system and rescue system.

1. Unzip the update files to the root of USB stick (file should be on the format ccpilot-vs-linux-update-v1.0.0.0-usb-stick.zip for main system or ccpilot-vs-linux-rescue-update-v1.0.0.0-usb-stick.zip for the rescue system)
2. Insert the USB stick to the device and wait for the automatic update to complete.

4.3.2. Programming the System Supervisor

The preloaded SS application can be updated through an internal serial port by using the CCAux API. The application *ccsettingsconsole* implements such a solution; to update the SS use the following command:

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

The file can be put on either USB stick connected to device or under /opt.

In order to update SS via the internal serial port, the device must be running the rescue system.

In case an invalid binary is written to the SS the system will not power up. This is due to the SS is being involved in the board bringup procedure. If this happens, the SS needs to be reprogrammed via the JTAG connection.

5. Trademarks

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

- Arm® is a registered trademark of ARM Limited (or its subsidiaries) in the US and/or elsewhere.
- Linux® is a registered trademark of Linus Torvalds in the U.S. and other countries.
- CrossControl, CCpilot and CCLinux are trademarks of CrossControl AB.