Automotive Box-PC 100 / 120
Automotive Box-PC 300

Software Development Guide

DELTA COMPONENTS GmbH · Auweg 27 · D-79761 Waldshut-Tiengen · Tel.: +49/7751/8399-0 · Fax: +49/7751/8399-99
www.delta-components.de E-mail: info@delta-components.de

Release Notes

Version	Release Date	Notes
1.0	October 2016	The 1st release

Disclaimer

This documentation is provided for use with DELTA COMPONENTS products. No license to DELTA COMPONENTS property right is granted. DELTA COMPONENTS assumes no liability, provides no warranty either expressed or implied relating to the usage, or intellectual property right infringement that may result from its use.

DELTA COMPONENTS provides this document without any warranty of any kind, expressed or implied, including, but not limited to, its particular purpose. IDELTA COMPONENTS may make changes to this document without notice.

Table of Contents

1 Precautions

1.1 Safety Precautions

In order to use this product safely, please take special note of the following precautions.

- Read all product manuals and related documentation before using this product. Use this product correctly and safely. Follow all warnings.
- If operating or extending this product in a manner not described in this manual, please do so at your own risk. Be sure to fully read this manual and other technical information on our website and proceed safely and responsibly.
- Do not install this product in a place with a lot of water, moisture, dust or soot. This could cause product failure, fire, or an electric shock.
- Some parts of this product generate heat and can reach high temperatures. This may cause burns if it is improperly handled. Do not touch the electronic components or surrounding area while powered on or immediately after being turned off.
- Carry out any design and development only after you have thoroughly read and understood this manual and any other related technical materials on the website or in the data sheets. Test your product thoroughly for reliability and safety.
- This product is not intended for applications that require extremely high reliability, safety, functionality and accuracy: including but not limited to medical equipment, traffic control systems, combustion control systems, and safety equipment. This company is not liable for death or injury if used in such systems.
- This product uses semiconductor components designed for generic electronics equipment such as office automation, communications, measurement equipment and machine tools. Foreign noise or a power surge may cause this product to malfunction or fail.
- To ensure there is no risk of bodily harm or property damage, be sure to take all electrical safety precautions such as protection circuits, limit switches, fuse breakers, or redundant systems. Only use the device after sufficient reliability and safety measures are in place.

1.2 Write Prohibited Regions

Data stored by the EEPROM, i.MX6 electrical fuse (e-Fuse) is used by the software contained in this product. Do not write to these regions as this may cause the product stop working correctly. Purposely writing to these regions voids the product warranty.

1.3 Warranty

As described in the Product Warranty Policy provided with this product, the main board is covered by a one year replacement warranty starting from the time of purchase. Please note that the other included goods and software are not covered under this warranty. Some knowledge used by DELTA COMPONENTS is provided by third parties, and DELTA COMPONENTS makes no representation or warranty as to the accuracy of such information.

2 Setup Software Development Environment

This document is to guide you how to :

- Setup software development environment (refer to Section 2-1, 2-2, 2-3 and 2-4)
- Install developed application software to Automotive Box-PC 100/120 (refer to Section 2-5)
- Build a new Automotive Box-PC 100/120 OS image (refer to Section 2-1, 2-2, 2-3, 2-6)
- Burn a new OS image to Automotive Box-PC 100/120 flash memory (Section 2-7)

2-1 Setup Host Computer

Setup a Linux host computer with the following minimum software and hardware requirements:

- Ubuntu version 12.04 or 14.04 only
- 120GB hard drive free space (for X11 backend) at least
- 4GB RAM size at least, more ram size more better
- CPU Intel core 2 daul (2 GHz) or later , AMD K8, K9 or later

2-2 Install Software Packages

Install the following packages on Ubuntu Host Computer:

- Essential Yocto Project host packages:

  $ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
  build-essential chrpath socat

- i.MX layers host packages for a Ubuntu 12.04 or 14.04 host setup:

  $ sudo apt-get install libsdl1.2-dev xterm sed cvs subversion coreutils texi2html \
  docbook-utils python-pysqlite2 help2 man make gcc g++ desktop-file-utils \
  libgl1-mesa-dev libglu1-mesa-dev mercurial autoconf automake groff curl lzop
  asciidoc

- i.MX layers host packages for a Ubuntu 12.04 host setup only:

  $ sudo apt-get install uboot-mkimage

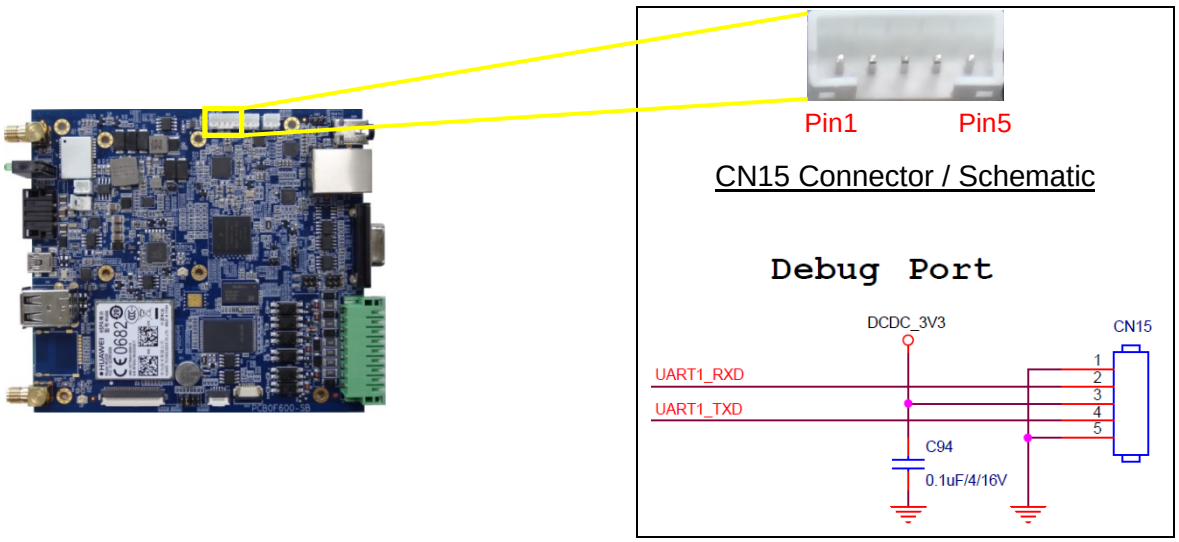
- i.MX layers host packages for a Ubuntu 14.04 host setup only:

  $ sudo apt-get install u-boot-tools

2-3 Setup Automotive Box-PC 100/120 Console Port

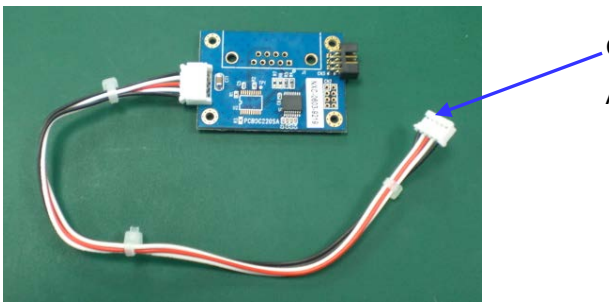
The console port (or debug port) is located inside Automotive Box-PC 100/120 (CN15 connector). Follow steps below to setup console port:

- Find CN15 console port connector on Automotive Box-PC 100/120 PCB.



- Make sure you have C220 debug board and cable. Note that C220 board is not part of the Automotive Box-PC 100/120 product. It has to be purchased separately.

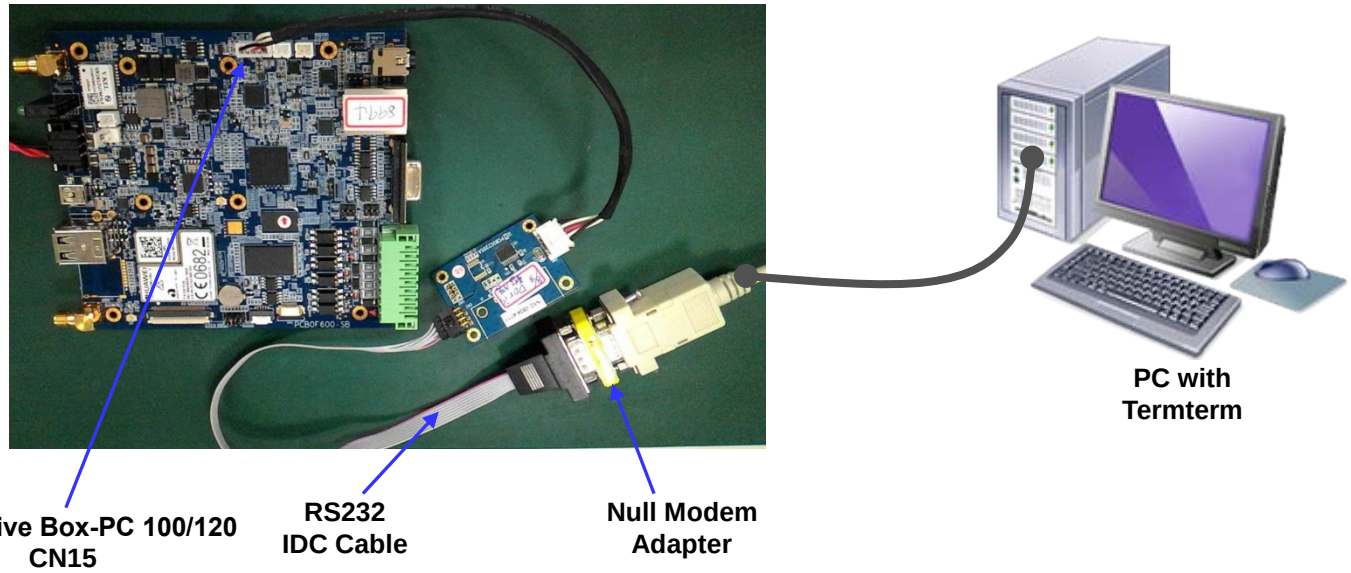
- Connect C220 debug board cable to Automotive Box-PC 100/120 CN15.

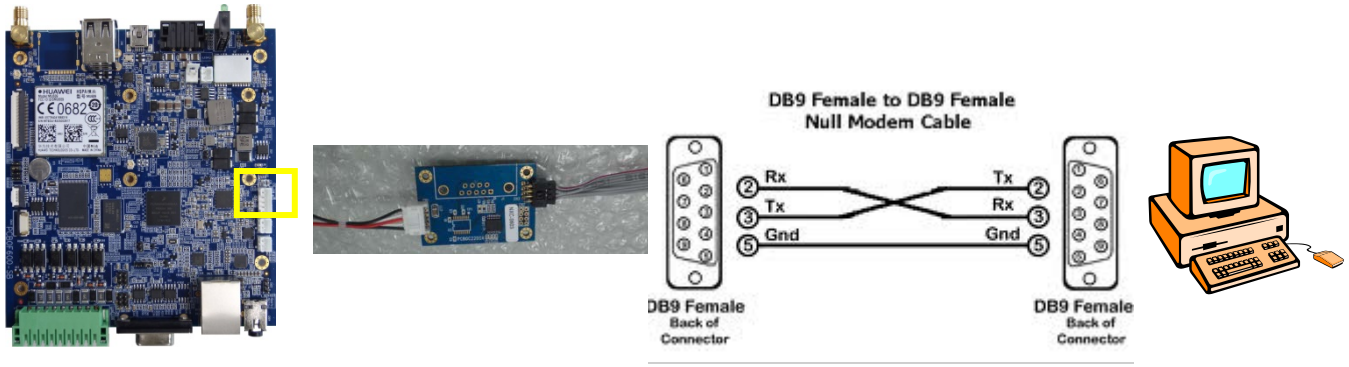


- Connect the RS232 IDC cable to C220 debug board 2x4 box header (black connector):

- A DB9 **null modem cable** (or adapter) is required when you want to connect console port to a PC with terminal emulation software such as TeraTerm.

- Turn on PC, run the terminal emulation program (e.g. ***TeraTerm***), and open TeraTerm COM port.



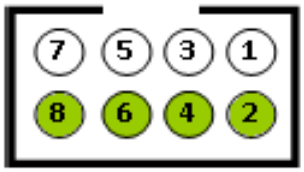


- Set TeraTerm COM port at **Baud Rate 115200, 8 data bits, no parity, 1 stop bit and no flow control**.

- After the above connection/setting, you will see Linux console prompt "$" in the PC TeraTerm.

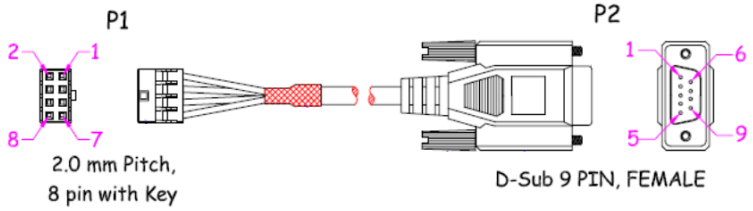
Pin Assignment: RS232 IDC Cable and Null Modem Adapter

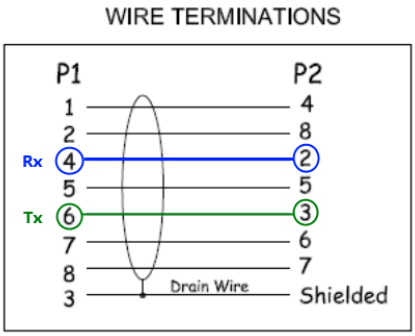
☐ Console box header (CN3 on C220 board) pin assignment



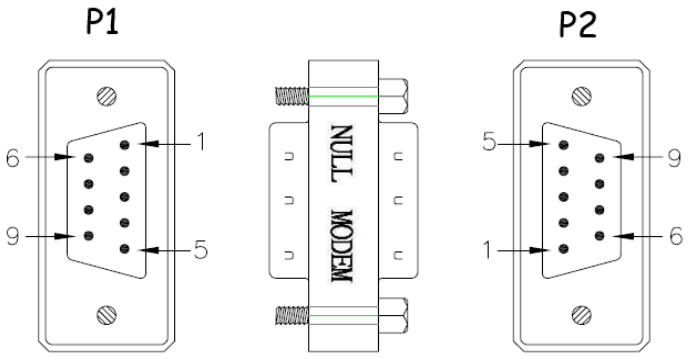
2 : CTS 6 : Tx
4 : Rx 8 : RTS

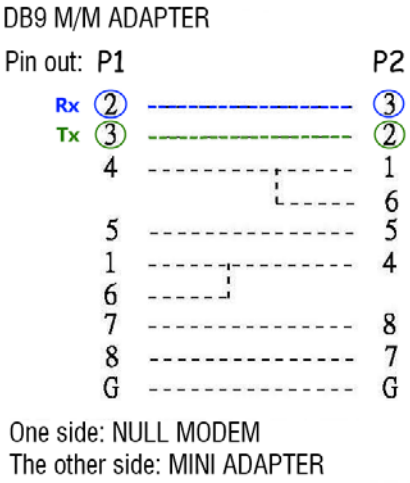
- RS232 IDC cable: pin header for DB9





- Null modem adapter: male-to-male for DB9





2-4 Setup Toolchain for Application Development

Before installing toolchain, please make sure the setups in Section 2-1, Section 2-2 and Section 2-3 are all successfully completed.

In Automotive Box-PC 100/120 software CD, find below file:

poky-eglibc-x86_64-meta-toolchain-qt5-cortexa9hf-vfp-neon-toolchain-1.6.2.sh

Follow steps below to install toolchain:

- Install SDK：

  $./poky-eglibc-x86_64-meta-toolchain-qt5-cortexa9hf-vfp-neon-toolchain-1.6.2.sh
- run QT creator：

  $source /opt/poky/1.6.2/environment-setup-cortexa9hf-vfp-neon-poky-linux-gnueabi qtcreator.sh

2-5 Install Application to Automotive Box-PC 100/120

Follow steps below to install your application software to Automotive Box-PC 100/120 (Yocto):

- In Automotive Box-PC 100/120 console port, type the following command:
  # mkdir /mnt/udisk

- Copy your application software from Ubuntu PC to a USB disk. Insert USB disk to Automotive Box-PC 100/120 USB port. In Automotive Box-PC 100/120 console port, you will see a new device created/dev/sdaX (if the USB disk is with only 1 partition, the newly created device is /dev/sda1). Use below command to mount device:

  #mount /dev/sda1 /mnt/udisk

- The mount command makes USB disk storage as the /mnt/udisk device in Automotive Box-PC 100/120. Find your application software in /mnt/udisk. You can run it now, or copy it to desired directory in Automotive Box-PC 100/120.

2-6 How to Build a New OS Image

Before building a new OS image, please make sure the setups in Section 2-1, Section 2-2 and Section 2-3 are all successfully completed.

Follow all the steps below to build a new OS image:

2-6-1 Setup the Repo Utility

- Create a bin folder in the home directory.

  $ mkdir ~/bin(this step may not be needed if the bin folder already exists)

  $ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo

  $ chmoda+x ~/bin/repo

- Add the following line to the .bashrc file to ensure that the ~/bin folder is in your PATH variable.

  export PATH=~/bin:$PATH

2-6-2 Yocto Project Setup

- Download the Freescale Yocto Project Community BSP recipe layers

  $ mkdir fsl-release-bsp

  $ cd fsl-release-bsp

  $ git config --global user.name "Your Name"

  $ git config --global user.email "Your Email"

  $ git config --list

  $ repo init -u git://git.freescale.com/imx/fsl-arm-yocto-bsp.git -b imx-3.10.53-1.1.0_ga

  $ repo sync

2-6-3 Build Image

- Choose a Freescale Yocto Project Image.

  Freescale Yocto Project Images list：

Image name	Target	Provided by layer
core-image-minimal	A small image that only allows a device to boot.	poky
core-image-base	A console-only image that fully supports	poky
core-image-sato	An image with Sato, a mobile environment and visual style for mobile devices. The image supports	poky

	X11 with a Sato theme, Pimlico applications. It contains a terminal, an editor and a file manager.	
fsl-image-machine-test	An FSL Community i.MX core image with console environment - no GUI interface	poky
fsl-image-gui	Builds a Freescale image with a GUI without any QT content. This image recipe works on all backends for X11, DirectFB, Frame Buffer and Wayland	meta-fsl-bsp-release/imx/meta-fslde mos
fsl-image-qt5	Builds a QT5 image for X11, Frame Buffer and Wayland backends	meta-fsl-bsp-release/imx/meta-fslde mos

The Machine Configurations , Bitbake Options , U-Boot Configuration, Building an Image, Build Scenarios :

Create yocto's project & Configurations , command like below:

$ MACHINE=<machine name> source fsl-setup-release.sh -b <build dir> -e <backend>

The example list :

X11 Image on i.MX 6Quad Sabre-SD

$ MACHINE=imx6qsabresd source fsl-setup-release.sh -b build-x11 -e x11

FB Image on i.MX 6Quad Sabre-AI

$ MACHINE=imx6qsabreauto source fsl-setup-release.sh -b build-fb –e fb

DFB Image on i.MX 6SoloX Sabre-SD

$ MACHINE=imx6sxsabresd source fsl-setup-release.sh -b build-dfb -e dfb

Example for build a QT5 image for X11, FrameBuffer and Wayland backend, like below :

$ MACHINE=imx6qsabresd source fsl-setup-release.sh -b build-x11 -e x11

$ cd build-x11

$ bitbake fsl-image-qt5

Restarting a Build Environment (When you logout, rebuild image or old project at next time) :

$ source setup-environment build-x11

$ bitbake fsl-image-qt5

2-6-4 Build SDK+QT

- In build folder：

  $ bitbake meta-toolchain-qt5

- Result：

  tmp/deploy/sdk/poky-eglibc-x86_64-meta-toolchain-qt5-cortexa9hf-vfp-neon-toolchain-1.6.2.sh

- Change GStreamer library：

  vifsl-release-bsp/sources/meta-qt5/recipes-qt/qt5/qtwebkit.inc

  For gstremaer 1.0

  PACKAGECONFIG ??= "gstreamer qtlocation qtmultimedia qtsensors"

  For gstreamer 0.10

  PACKAGECONFIG ??= "gstreamer010 qtlocation qtmultimedia qtsensors"

2-6-5 Setup QT creater with Yocto Build

- Edit local.conf in the conf folder：

  EXTRA_IMAGE_FEATURES = "debug-tweaks ssh-server-openssh"

  IMAGE_INSTALL_append = "openssh-sftp-server curl modemmanager minicom ppp lighttpd dhcp-sever iptables dnsmasq"

  * curl for add curl software package
  * modemmanager for add modem manager software (3G dialing software)
  * minicom for add minicom software package
  * ppp for add pppd software package (3G dialing software)
  * lighttpd for add lighttpd web server software package
  * dhcp-server for add dhcp & DNS server software package
  * iptables for add iptables (route , NAT , firewall)software package
  * dnsmasq for add dhcp & DNS server software package

- Add “ssh-server-openssh” to EXTRA_IMAGE_FEATURES to enable Yocto SSH server.
- Add “openssh-sftp-server” to IMAGE_INSTALL_append to enable Yoctosftp server.

2-6-6 Configure Yocto to Support Modem

- Edit local.conf in the conf folder：

  IMAGE_INSTALL_append = "**ppp modemmanager**"

- Edit **sources/poky/meta/recipes-connectivity/connman/connman/connman** to **remove do_start & do_stop**

- Add

  sources/poky/meta/recipes-connectivity/ppp/ppp/wcdma-chat-connect

  sources/poky/meta/recipes-connectivity/ppp/ppp/wcdma-chat-disconnect

- Modify

  sources/poky/meta/recipes-connectivity/ppp/ppp_2.4.6.bb

```
    file://wcdma-chat-connect \
    file://wcdma-chat-disconnect \
    ..
    ..
mkdir -p ${D}${sysconfdir}/ppp/peers
    install -m 0755 ${WORKDIR}/pap ${D}${sysconfdir}/chatscripts
    install -m 0755 ${WORKDIR}/wcdma-chat-connect ${D}${sysconfdir}/chatscripts
```

2-6-7 Lighttpd

- Add lighttpd to yocto build conf：

  $ vi conf/local.conf

```
PACKAGE_CLASSES ?= "package_rpm"
EXTRA_IMAGE_FEATURES = "debug-tweaks ssh-server-openssh"
IMAGE_INSTALL_append= "openssh-sftp-server modemmanager minicom curl
lighttpd ppp dhcp-sever iptables dnsmasq"
USER_CLASSES ?= "buildstats image-mklibs image-prelink"
```

- Add CGI to lighttpd and config lighttp：

  $ cd sources/poky/meta/recipes-extended/lighttpd

  $ vi lighttpd_1.4.33.bb

```
RDEPENDS_${PN} += " \
lighttpd-module-access \
lighttpd-module-accesslog \
lighttpd-module-cgi \
lighttpd-module-indexfile \
lighttpd-module-dirlisting \
lighttpd-module-staticfile \
```

  $ cd sources/poky/meta/recipes-extended/lighttpd/lighttpd

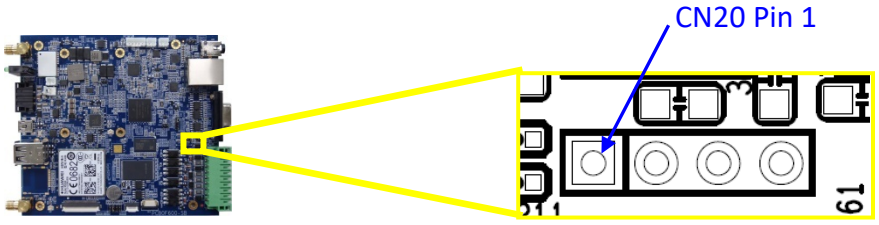
  $ vi lighttpd.conf

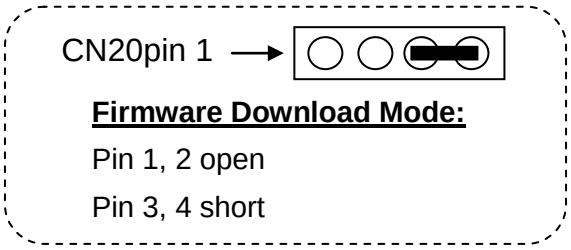
```
server.modules                  = (
??/span>
??/span>
"mod_cgi",
??/span>
??/span>

#### CGI module
$HTTP["url"] =~ "/cgi-bin/" {
cgi.assign = ( "" => "" )
}

cgi.assign          = (
        ".cgi"    => ""
)
```

2-7 How to Burn OS Image to Flash or MicroSD

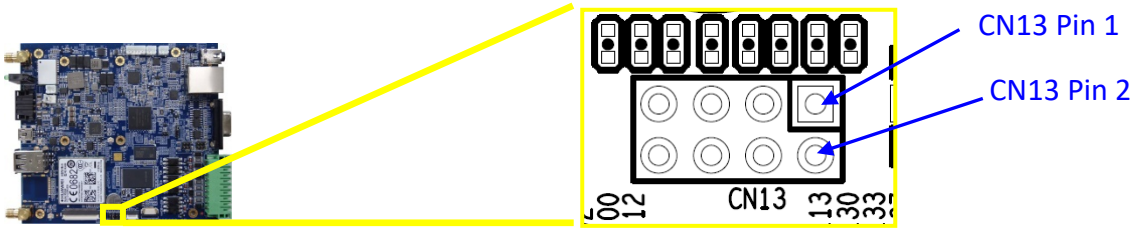
Please follow steps below to burn OS image to the Automotive Box-PC 100/120 flash or microSD:

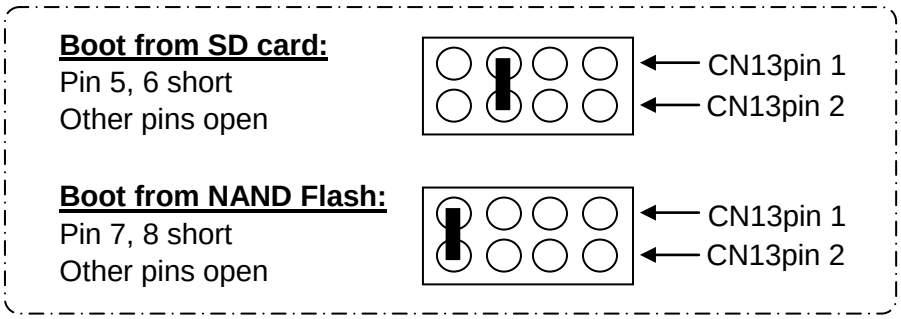
(I). Use CN20 pin header to select the Automotive Box-PC 100/120 operation mode: Normal Operation
mode or Firmware Download mode. Set CN20 at “Firmware Download Mode”.





(I). Use CN13 pin header to select boot device. Set CN13 to boot Automotive Box-PC 100/120 either from NAND flash or MicroSD.



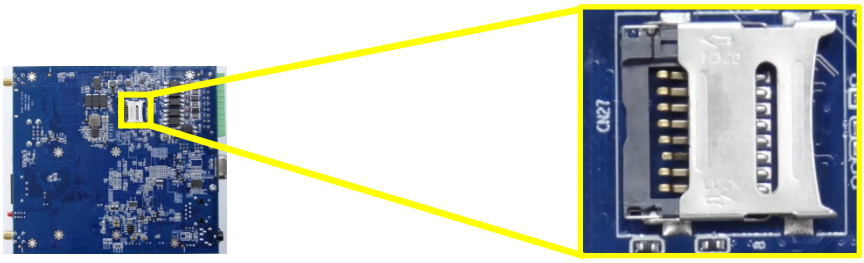


(II). Install iMX6UL mfgtools (mfgtools-14_52) on a Windows PC (contact us to get the tools). In the Windows PC with iMX6UL mfgtools, go into mfgtools-14_52 folder to yocto sub-folder：

(III). If CN13 is set to boot from SD card (pin 5&6 short, others open), go to step (IV).
If CN13 is set to boot from flash (pin 7&8 short, others open) go to step (V).

(IV). Boot Device is SD card:

- Go into <sdboot> folder and then to <256M> folder. Copy the files boot.scr and imx6ul.imx to the following folder：
  mfgtools-14_52\Profiles\Linux\OS Firmware\files\yocto
- Insert your sd card to the card holder (in the back side of the Automotive Box-PC 100/120 PCB):

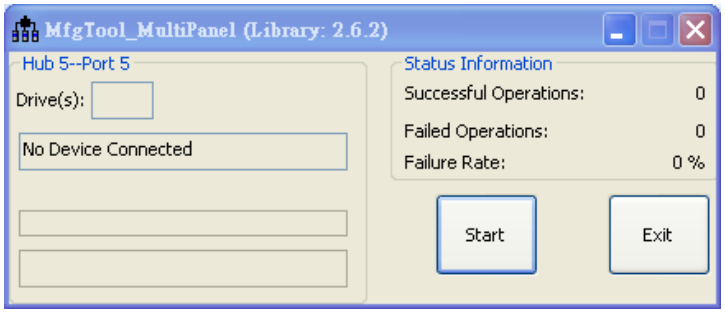


- Go back to mfgtools-14_52\ folder, double-click f600-sdcard-256M.vbs to run Mfgtool. Go to step (VI).

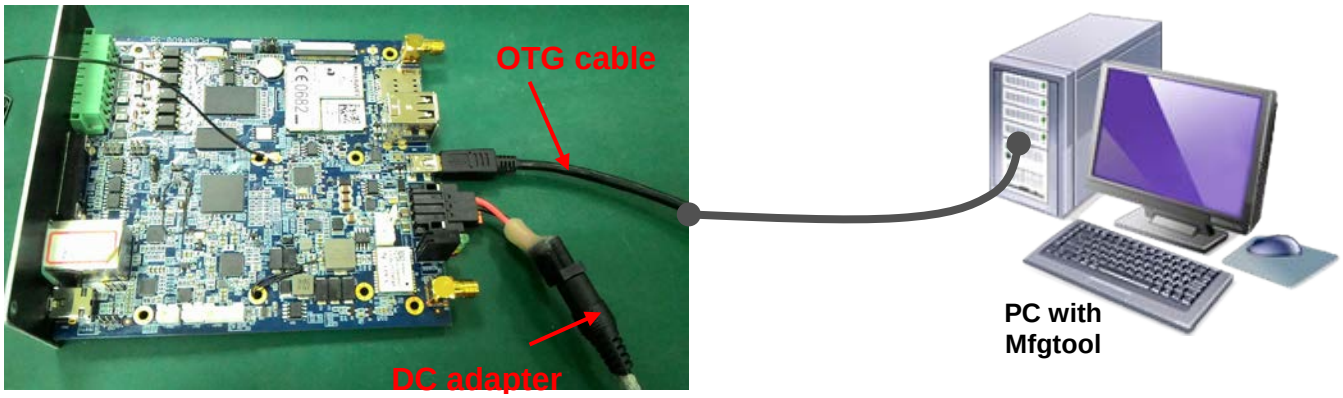
(V). Boot device is NAND flash:

- Go into <nand> folder and then to <R256M_N512M> folder. Copy the files boot.scr and imx6ul.imx to the following folder：
  mfgtools-14_52\Profiles\Linux\OS Firmware\files\yocto
- Go back to mfgtools-14_52\ folder, double-click f600-nand-R256M-N512M.vbs to run Mfgtool. Go to step (VI).

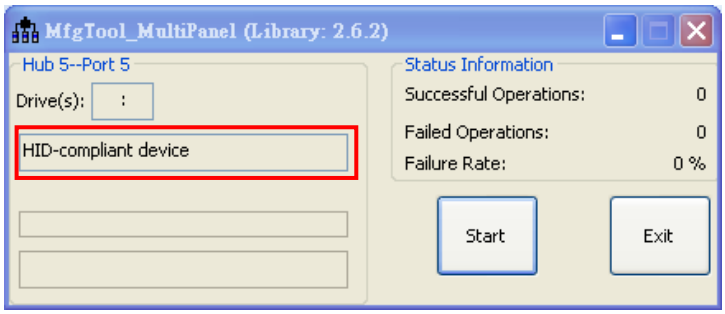
(VI). The MfgTool will display the following dialog box：



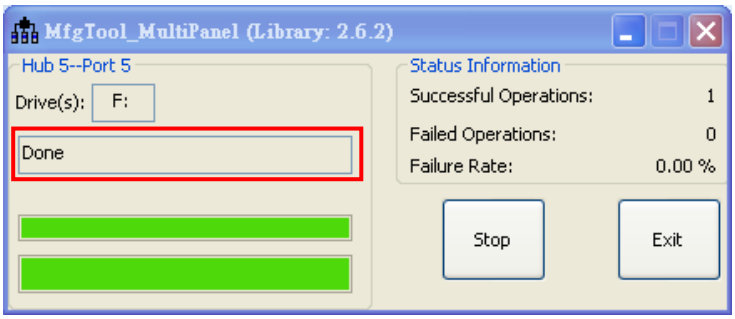
(VII). Connect OTG cable between Automotive Box-PC 100/120 and Windows PC. Connect power cable and power on the Automotive Box-PC 100/120.



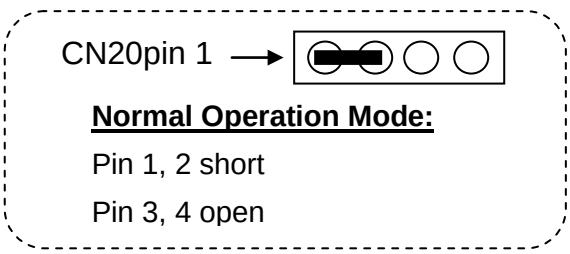
(VIII). The Automotive Box-PC 100/120 will be recognized by Mfgtool as a “HID-compliant device” (shown below).



(IX). Click Start to start writing OS to flash (or SD card). The Mfgtool will display a “Done” message, when OS writing is completed successfully. Click Stop and then Exit to exit Mfgtool.



(X). After OS image is successfully burned into the Automotive Box-PC 100/120 flash ROM. Set CN20 back to “Normal Operation Mode”.



(XI). Power off the Automotive Box-PC 100/120 device, and power it on again. The Automotive Box-PC 100/120 a new OS now running with a new OS.