INTEGRATING THE CP210X VIRTUAL COM PORT DRIVER INTO THE ANDROID PLATFORM

1. Introduction

This document describes how to build an Android kernel and the steps needed to integrate the CP210x virtual COM port (VCP) driver in to the build.

Android is based off of the Linux kernel, so there is already support for the CP210x device built in to the kernel. However, CP210x support is not included in a default kernel build configuration for most Android devices.

There are many devices that can run the Android operating system (OS), as well as several different versions of an Android operating system. Because Android is open source, it can be changed based on the needs of device manufacturers, wireless carriers and other developers. This fact makes it difficult to provide the exact steps for rebuilding, as each developer may be using different hardware and a different build of Android specific to that hardware.

This document was written based on the experience of building an Android Jelly Bean (4.3.1) kernel for a PandaBoard using a TI OMAP4430 processor. The Android port used originates from the 13.10 release from Linaro. Build materials and information on this specific release can be found at http://releases.linaro.org/13.10/android/panda/. If you are using another platform, you should consult the manufacturer's website to get the Android distribution support for your device. Although the steps below are targeted to Jelly Bean on the PandaBoard, developers can follow similar steps to download and rebuild the kernel for other platforms. For any other needs specific to Android, visit the Android Developer website at http://developer.android.com.

2. Materials

To build a custom Android kernel and image, you will need:

- CP210x USB to UART Bridge Evaluation Board
- Target Android Device (PandaBoard)
  - SD card to hold OS
  - Any cables for connectivity (HDMI video cable, network cable, USB power cable, USB to RS232 adapter)
- Source Distribution for Target Android Version (Linaro 13.10 - JellyBean 4.3.1 Android Release for PandaBoard)
- Toolchain to build the Source Distribution (Android NDK, Revision 9C)
- Development System Running Linux (Ubuntu 12.04 LTS)

3. Overview

The basic steps to integrate support for the CP210x driver are:

- Create bootable image on an SD Card from the Android source distribution
- Download the Android build tools for the target Android source and platform
- Download the Android source tree for the target platform
- Modify the configuration of the kernel to include the CP210x and other required drivers
- Build and install the new image to the target platform media

Copyright © 2014 by Silicon Laboratories

4. Create Bootable Image on an SD Card

In order to replace the kernel on a device's media, you will need to create a bootable device. Typically, a source distribution will contain prebuilt binaries that can be loaded directly on to an SD Card. Loading a complete prebuilt image enables you to make sure that the version works on your platform and allows you to replace only the kernel instead of rebuilding the entire Android distribution from scratch.

Linaro provides these prebuilt binaries for its supported devices. The following instructions describe how to use your development system to download the prebuilt images to a PandaBoard device and test that the PandaBoard can boot and run Android without any issues.

1. Use Android 4.3.1 (Jelly Bean) for the PandaBoard from Linaro's 13.10 release version. The specific version can be found under the Linaro Engineering Builds on http://releases.linaro.org/13.10/android/panda/. Download the following files to the development system to a known and accessible location, such as your home directory:
   - boot.tar.bz2
   - system.tar.bz2
   - userdata.tar.bz2

2. Next, on the development system, install and update for the Linaro Image Tools:

```
$ sudo add-apt-repository ppa:linaro-maintainers/tools

$ sudo apt-get update

$ sudo apt-get install linaro-image-tools
```

3. Disable the automount so you can properly image your SD card:

```
$ dconf write /org/gnome/desktop/media-handling/automount false

$ dconf write /org/gnome/desktop/media-handling/automount-open false
```

4. To image your SD card, first find out what the device name is. Start by inserting your SD card and running:

```
$ dmesg
```

5. You should see something similar to the following at the end of your log:

```
sdb: sdb1 sdb2 sdb3 sdb4 < sdb5 sdb6 >
```

6. Navigate to the directory where the boot/system/userdata.tar.bz2 files were downloaded. Ensure that the device above is indeed associated with your SD card (otherwise you can erase your hard drive) and create the media using the Linaro imaging tools:

```
$ linaro-android-media-create --mmc /dev/sdc --dev panda --boot
boot.tar.bz2
--system system.tar.bz2 --userdata userdata.tar.bz2
```

7. Next, install the graphics libraries to the device (do this immediately after creating the media, before removing the SD card):

```
$ wget http://people.linaro.org/~vishalbhoj/install-binaries-4.0.4.sh

$ chmod a+x install-binaries-4.0.4.sh

$ ./install-binaries-4.0.4.sh
```

8. Finally, restore the automount:

```
$ dconf write /org/gnome/desktop/media-handling/automount true

$ dconf write /org/gnome/desktop/media-handling/automount-open true
```

9. The SD card is now ready to boot. Insert it into the SD card slot on the PandaBoard and hook up a monitor to the HDMI output and a keyboard and mouse to the USB ports. Connect the RS232 connection up to your development system. This is the Android debug port, and will output logging information from the PandaBoard at a baud rate of 115200, 8N1. This also serves as a root terminal in to the device. Once everything is connected, power the board to start the boot sequence. If this works, then you can move on to the next steps of replacing the kernel image on this device.

5. Download the Android Build Tools

To build the source from an Android distribution, you will need the toolchain that can build the source for your platform. This can be different between devices, so check with the manufacturer to find out what toolchain to use for your device.

1. For Android Jelly Bean (4.3.1) on the PandaBoard, use the arm-linux-androideabi 4.6 version from the Android NDK, Revision 9C. This can be downloaded directly from the Android Developer website at: http://developer.android.com/tools/sdk/ndk/index.html. Here is a link to the specific version:

   Android NDK, Revision 9C: http://dl.google.com/android/ndk/android-ndk-r9c-linux-x86_64.tar.bz2

   After downloading these tools, extract them into a known and accessible location, such as your home directory.
2. Next, you will need to set the PATH variable to include the specific path to the correct version of the prebuilt toolchain. In this example, we have extracted the Android NDK to the home directory and used the following path to target the 4.6 version of the tools (replace <username> with your username):

```
$ export PATH=$PATH:/home/<username>/android-ndk-r9c/toolchains/arm-
linux-androideabi-4.6/prebuilt/linux-x86_64/bin
```

3. Additionally, a few other tools will need to be installed on your development system. Use the following commands to install them:

```
$ sudo apt-get install git

$ sudo apt-get install curl

$ sudo apt-get install ncurses-dev

$ sudo apt-get install uboot-mkimage
```

6. Download the Android Source Tree

Next, you will need to download the Android source. The manufacturer for your device should have its own repository for the download, or should be able to point to a specific revision to download.

In this example, you will clone the Linaro PandaBoard source and checkout a version specific to the 13.10 build. This will allow you to rebuild the kernel and replace it on the media that was also created from 13.10.

Note: The following steps were taken from Linaro's build script "linaro_kernel_build_cmds.sh", and is how to obtain the version to check out and links to rebuild. For reference, this file can be found at http://releases.linaro.org/13.10/android/panda/linaro_kernel_build_cmds.sh.

1. Download the source for your development board:

```
$ git clone git://android.git.linaro.org/kernel/panda
```

2. Navigate to the source directory and checkout the source tree for version 13.10:

```
$ cd linaro-kernel

$ git checkout ca4b45c8f598951b828ca968f5953b8d5e85e34c
```

7. Modify the Configuration of the Kernel

Once the Android kernel source has been downloaded, you will need to create a configuration for the kernel. Linaro already has a default configuration so it is easiest to start with this and add in your CP210x support. The following steps describe how to do this.

1. Download the kernel configuration from Linaro to a ".config" file to be used by the build:

```
$ curl -q http://snapshots.linaro.org/android/~linaro-android-member-ti/
panda-linaro-13.10-release/3/kernel_config > .config
```

2. Use the menuconfig to update the configuration for CP210x support:

```
$ make ARCH=arm menuconfig
```

In the menuconfig UI, navigate to the Device Drivers section:

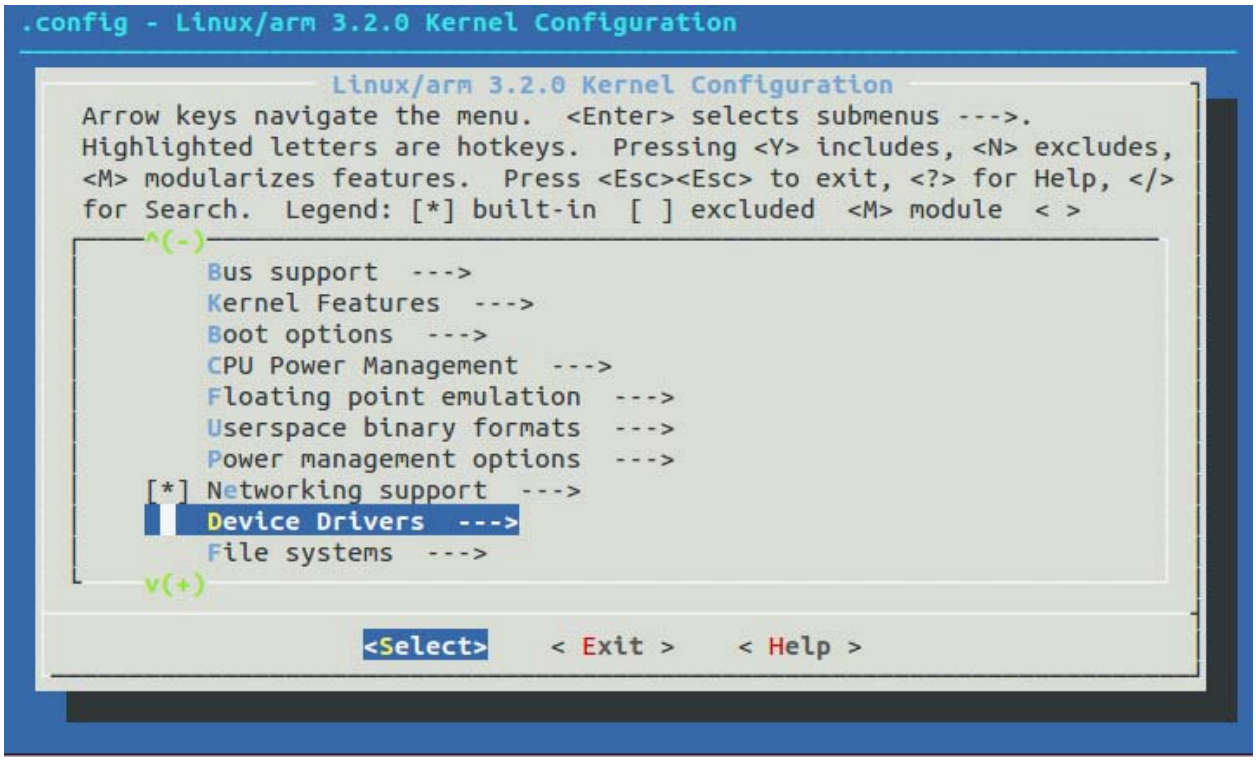

Figure 1. menuconfig: Device Drivers

Then navigate to the USB Support section (this should already be marked with a *).

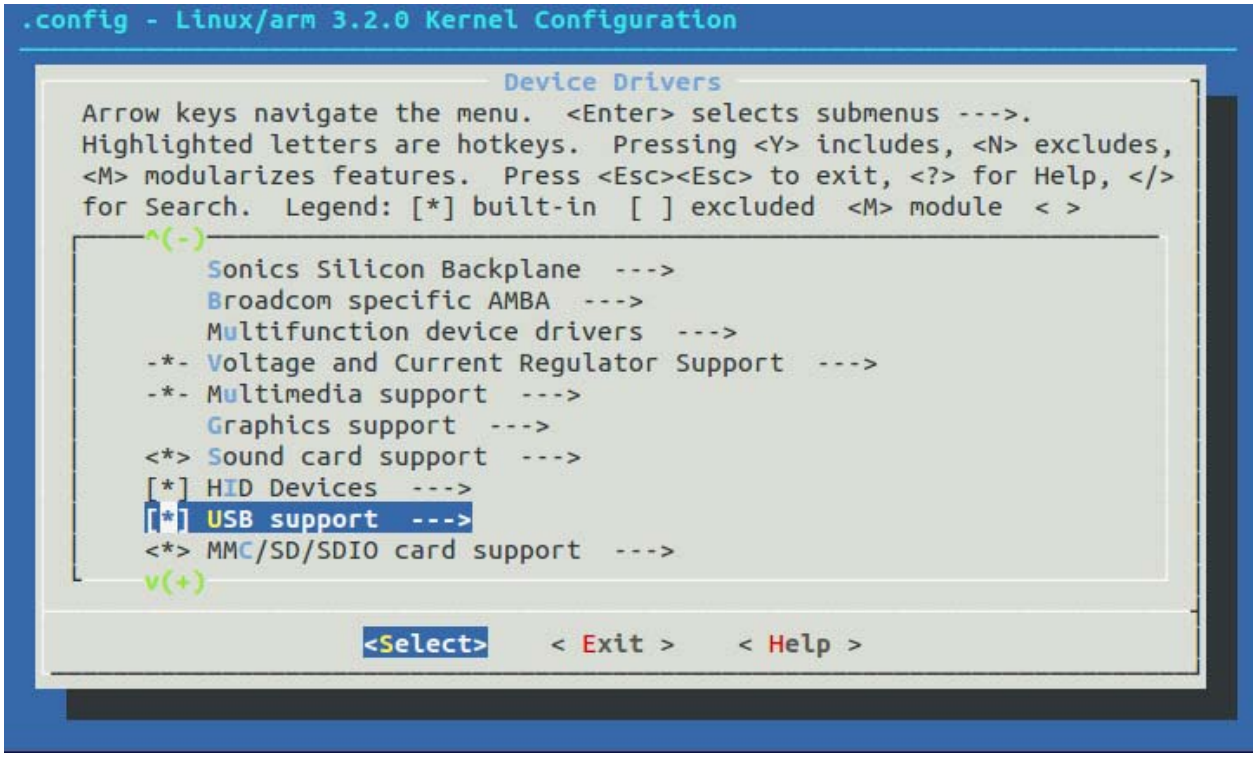

Figure 2. menuconfig: USB Support

Then navigate to the USB Serial Converter Support section. Press the space bar until this item shows up as a *, then enter the section to edit the USB converter support:

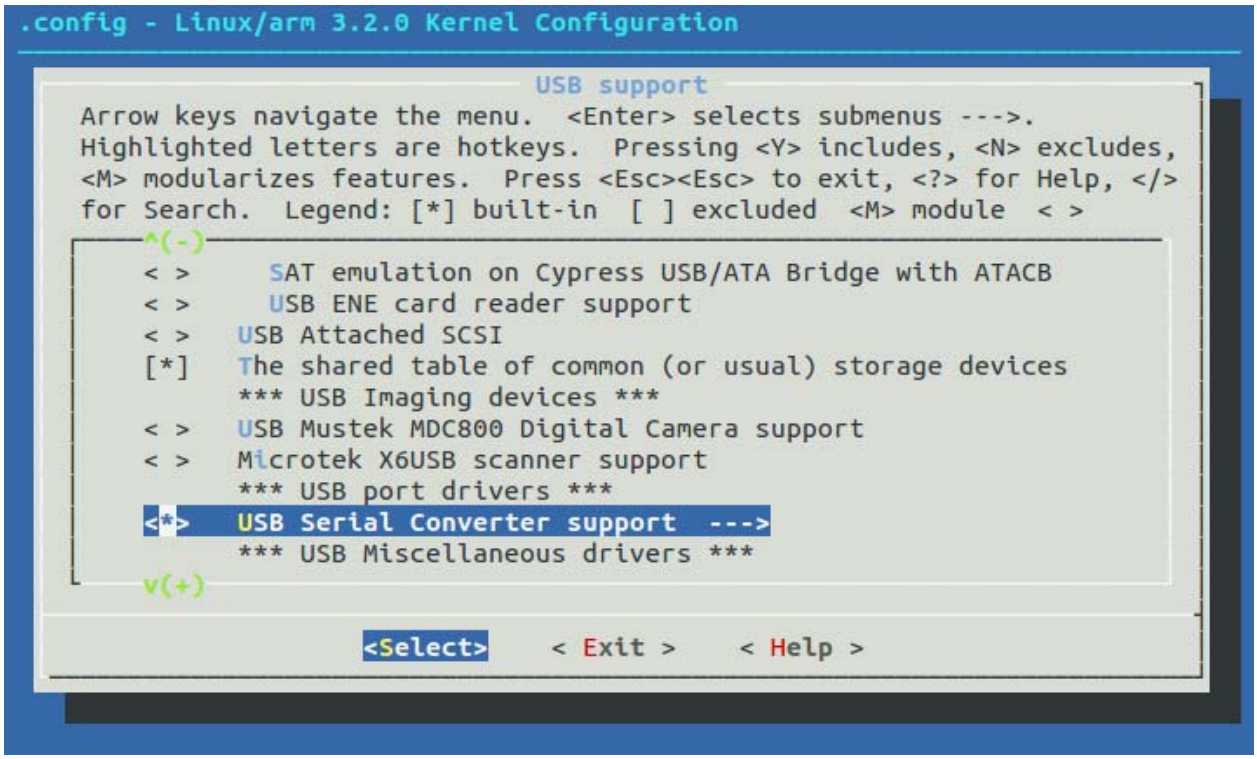

Figure 3. menuconfig: USB Serial Converter Support

Highlight USB Generic Serial Driver and press the space bar until this item shows up as a *. Then navigate down to the USB CP210x family of UART Bridge Controllers and press the space bar until this item shows up as a *. Finally, exit it out of each section:

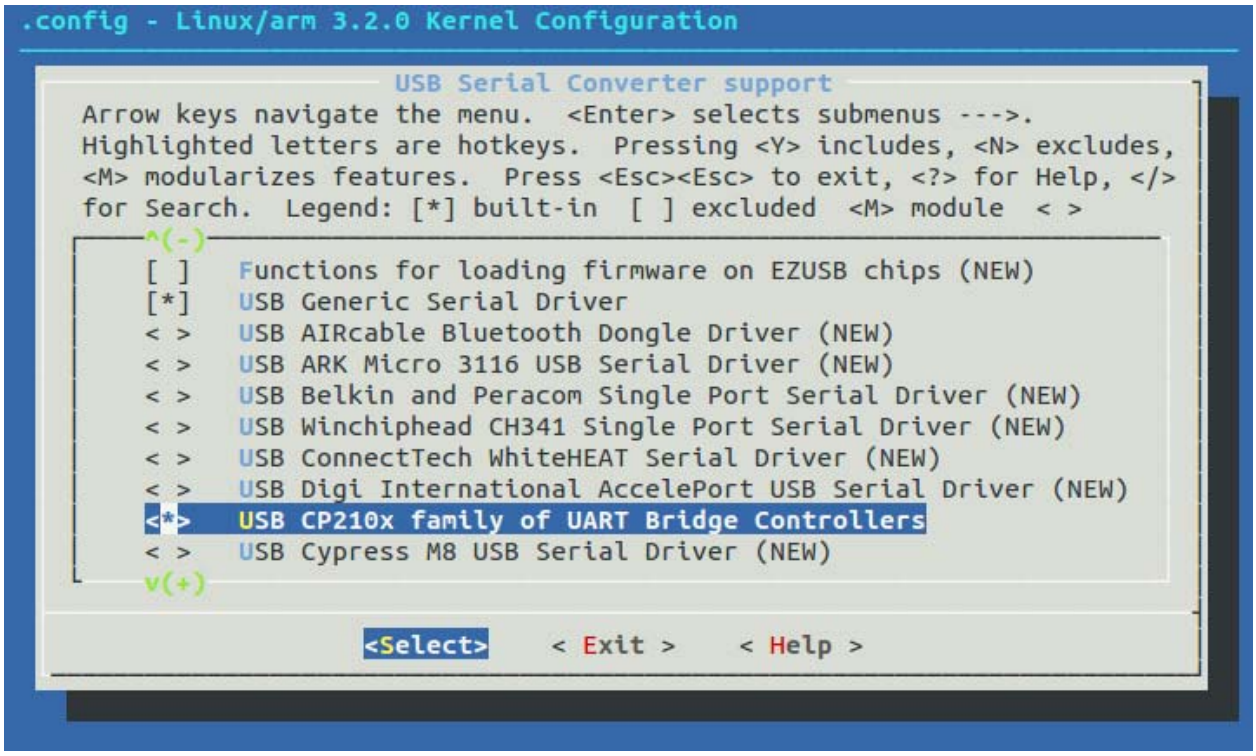

Figure 4. menuconfig: USB Serial Converter Features

On the final page, select Yes to save the configuration in to the ".config" file:

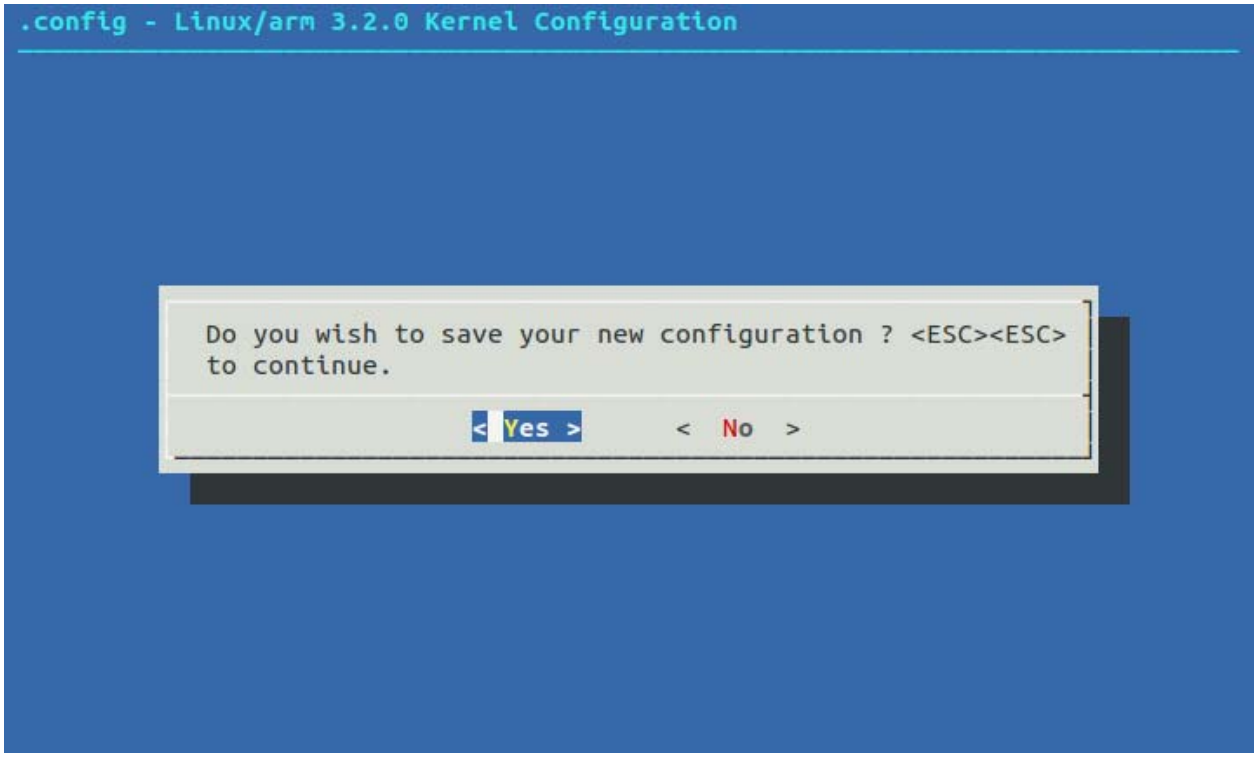

Figure 5. menuconfig: Save Configuration

At this point everything is setup to build the CP210x in to the kernel.

Note: The Android Jelly Bean (4.3.1) OS is based on a 2.6 version of the Linux kernel. Basic support for CP2101/2/3 devices are included in the kernel by default, but the driver is not up to date for newer devices such as CP2104/5/8/9. To get support for these, you will need to replace the cp210x.c driver file in the source before you build your kernel image. Perform the following steps to replace the file:

1. Download the Linux 2.6 driver from the Silicon Labs website: http://www.silabs.com/Support%20Documents/Software/Linux_2.6.x_VCP_Driver_Source.zip
2. Extract the cp210x.c file in to the kernel source tree. The original cp210x.c file is located in the tree under <base kernel directory>/drivers/usb/serial.
3. Rebuild the kernel using the steps below.

8. Build and Install the New Image

Once you confirm your device works and boots off of the known media, you have downloaded the necessary toolchain and the Android kernel source tree can build your kernel image.

1. To build the source run the following command:

```
$ make ARCH=arm CROSS_COMPILE=arm-linux=andorideabi- uImage
```

2. When your build completes successfully, you can find your ‘uImage’ file under <base kernel directory>/arch/arm/boot.
3. The final step is to replace the “uImage” file on the boot device with the new one. Reinsert the original SD card that contains the prebuilt bootable image. You should see multiple partitions come up on this device. Navigate to the “boot” partition and delete the “uImage” file on the device, then copy over the new one from your build.
4. The SD card is now ready to boot. Insert it into the SD card slot on the PandaBoard, hook up a monitor to the HDMI output and a keyboard and mouse to the USB ports. Connect the RS232 connection up to your development system using minicom or some other terminal program (115200, 8N1).

9. Testing the CP210x Driver in Android

When your PandaBoard boots up, you can plug in a CP210x device to the USB host port on the PandaBoard. On the host terminal connected to the Android debug port you should see something similar to the following output:

```
[  104.627380] usb 1-1.3: new full-speed USB device number 8 using ehci-
omap

[  104.764923] usb 1-1.3: New USB device found, idVendor=10c4,
idProduct=ea60

[  104.772338] usb 1-1.3: New USB device strings: Mfr=1, Product=2,
SerialNumber=3

[  104.786437] usb 1-1.3: Product: CP2104 USB to UART Bridge Controller

[  104.803405] usb 1-1.3: Manufacturer: Silicon Labs

[  104.810241] usb 1-1.3: SerialNumber: 0001

[  104.836822] cp210x 1-1.3:1.0: cp210x converter detected

[  104.845794] usb 1-1.3: cp210x converter now attached to ttyUSB0
```

The last line will specify what tty device the CP210x will be accessible through, in this case it is “ttyUSB0”.

To do a quick test to see that data is going through the device type the following commands in the Android debug port terminal:

```
$ stty -F /dev/ttyUSB0 115200

$ stty -F /dev/ttyUSB0 -a

speed 115200 baud;stty: /dev/ttyUSB0

line = 0;

intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;

eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt
= ^R;

werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;

-parenb -parodd cs8 hupcl -cstopb cread clocal -crtscts

-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -
ixoff

-iuclc -ixany -imaxbel -iutf8

opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0
vt0 ff0

isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -
echoprt

echoctl echoke
```

The first command will set the baud rate, then the second command will show the baud rate has been set (underlined and in green above). Hook your CP210x device up to another terminal set to that baud rate, 115200. Then we can see that data goes across by typing the following command in to our Android debug port terminal:

```
$ ls /dev > /dev/ttyUSB0
```

This should display the directory listing for /dev on your Android device in the other terminal and confirm that it is transmitting data across the CP210x as expected.

Note: Some CP210x devices flush buffers on close, so the directory listing might not be complete since the open, data transmission, and close happen quicker than data can exit the device. Certain devices can be configured to avoid this behavior by using the steps outlined in application note, “AN721: CP21xx Device Customization Guide.” Under normal operation where an application is developed, the port should be kept open for the duration of the time needed for transmission or configured to not flush buffers on closed if supported and desired.

10. Conclusion

This application note explained how to build the CP210x VCP driver into an Android kernel. Upon completion of these steps, a developer can then utilize the CP210x device as a data transmission or data acquisition device for Android. The device can be used in applications developed for the Android platform as services or end user applications developed with the Android SDK.

Contact Information

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032

Please visit the Silicon Labs Technical Support web page:
https://www.silabs.com/support/pages/contacttechnicalsupport.aspx
and register to submit a technical support request.

Patent Notice

Silicon Labs invests in research and development to help our customers differentiate in the market with innovative low-power, small size, analog-intensive mixed-signal solutions. Silicon Labs' extensive patent portfolio is a testament to our unique approach and world-class engineering team.

The information in this document is believed to be accurate in all respects at the time of publication but is subject to change without notice. Silicon Laboratories assumes no responsibility for errors and omissions, and disclaims responsibility for any consequences resulting from the use of information included herein. Additionally, Silicon Laboratories assumes no responsibility for the functioning of undescribed features or parameters. Silicon Laboratories reserves the right to make changes without further notice. Silicon Laboratories makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Silicon Laboratories assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Silicon Laboratories products are not designed, intended, or authorized for use in applications intended to support or sustain life, or for any other application in which the failure of the Silicon Laboratories product could create a situation where personal injury or death may occur. Should Buyer purchase or use Silicon Laboratories products for any such unintended or unauthorized application, Buyer shall indemnify and hold Silicon Laboratories harmless against all claims and damages.

Silicon Laboratories and Silicon Labs are trademarks of Silicon Laboratories Inc.
Other products or brandnames mentioned herein are trademarks or registered trademarks of their respective holders.