

Gateworks Venice Programming

Setup

To be able to get to the system console on first powering up of the board, the JTAG debugger will be needed. It will allow the system console to be presented to the PC as a serial port. The standard

Connectivity

Connect the USB/Serial/JTAG debugger cable for the system console and let the board boot normally the first time. Login with root.

If the ethernet interface doesn't come up and get an ip address.

```
root@noble-venice:~# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:d0:12:07:fa:02 brd ff:ff:ff:ff:ff:ff
    altname end0
    inet 172.20.10.71/24 brd 172.20.10.255 scope global dynamic noprefixroute eth0
        valid_lft 601249sec preferred_lft 525744sec
    inet6 fe80::d6dd:4eca:da70:827d/64 scope link
        valid_lft forever preferred_lft forever
```

The issue the following command

```
dhcpcd eth0
```

The internet connection is needed to add certain packages to the Ubuntu OS to be able to implement the solution.

Note the IP address that is assigned it will be useful later.

Additional packages

The following packages need to be added to the filesystem on the Gatework module.

Be sure there is a functioning network interface before attempting to complete the following commands.

```
apt update
apt-get install alsa-utils python3-serial microcom sox pulseaudio
```

Push Pool phone scripts

To be able to push the pool phone scripts to the Gateworks board a user needs to be created

```
adduser kuser
```

And create the user with a password. REMEMBER the password.

This user on the target needs to have some additional permissions granted. These are for accessing the modem and gpios and provides some additional security since the application is not being run as root.

```
usermod -aG dialout,audio,plugdev kuser
```

To facilitate the copying of files,

```
ssh-copy-id kuser@172.20.10.71
```

On the Gateworks target create the "/mnt/data/" directory.

```
mkdir /mnt/data
mkdir -p /mnt/data/sounds
mkdir -p /mnt/data/pulse
chmod ugo+w /mnt/data /mnt/data/sounds /mnt/data/pulse
```

Then push the scripts.

```
just push
```

Note: For this to work the HOST machine needs to have the tool "just" installed.

Configuring the system to perform the Pool phone function.

```
cp imx8mm-venice-gw7xxx-0x-gpio.dtbo /boot/.
reboot
```

Hit the "Enter" key several times while the UBoot screen is present to suspend the boot process, a boot script variable needs to be modified to add the overlays needed.

```
setenv fdt_overlays "imx8mm-venice-gw7xxx-0x-gpio.dtbo imx8mm-venice-gw72xx-0x-gw16157.dtbo"
```

```
saveenv
boot
```

System Setup

Once the target has completed the boot process, the following script can be run to set up the service for the pool phone application.

The following needs to be added to the /etc/modules-list.d/modules.config file.

```
snd_soc_fsl_sai
snd_soc_sgtl5000
snd_soc_simple_card
```

The following commands are in the file /mnt/data/config_sys.sh. This script can be executed on the terminal of the target.

```
cp /mnt/data/99-ignore-modemmanager.rules /etc/udev/rules.d/99-ignore-
modemmanager.rules
cp /mnt/data/switch_mon.service /etc/systemd/system/switch_mon.service

cp /mnt/data/pulse/daemon.conf /etc/pulse/daemon.conf

mkdir -p /home/kuser/.config/systemd/user
cp /mnt/data/pulse/pulseaudio.service
/home/kuser/.config/systemd/user/pulseaudio.service
chown -R kuser:kuser /home/kuser/.config

touch /mnt/data/calls.log
chmod ugo+w /mnt/data/calls.log

systemctl daemon-reload
systemctl enable switch_mon.service

# This must be done a kuser.
#systemctl --user enable pulseaudio.service
```

NOTE: The last part needs to be executed by the "kuser"

DTS Overlay

The following needs to be compiled to create a dts overlay (a .dtbo file)

```
/*
 * GW72xx GPIO:
 * - repurpose ecspi2 CLK/MISO/MOSI/CS0 as GPIO
 * - repurpose i2c3 CLK/DATA as GPIO
```

```

*/

#include <dt-bindings/gpio/gpio.h>

#include "imx8mm-pinfunc.h"

/dts-v1/;
/plugin/;

&uart3 {
    status = "disabled";
};

&gpio5 {
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_gpio5_hog>;
    gpio-line-names = "", "", "", "", "", "", "", "", "", "",
                      "", "", "", "", "", "", "", "", "", "",
                      "btn_red_led", "extra-io";
};

&iomuxc {
    /* pinmux: refer to IMX8M Reference Manuals for the bit configuration
     * - the first definition comes from imx8mm-pinfunc.h and sets the
     *   pins NUX_CTL register and input select
     * - the second definition sets the PAD_CTL register that configures
     *   things like drive strength, slew rate, pull-up, pull-down, and
     *   hysteresis.
     *   If you want to be able to read back the pin state you need to set
     *   the SION
     *   bit by setting bit 30 (as is done below)
     */
    pinctrl_gpio5_hog: gpio5hoggrp {
        fsl,pins = <
            MX8MM_IOMUXC_UART3_RXD_GPIO5_I026        0x140
            MX8MM_IOMUXC_UART3_TXD_GPIO5_I027        0x140
        >;
    };
};

```

To process this file. (Assuming the instructions [in the device tree usage doc from Gateworks](#)) have been followed.

```

cat gw-venice-gpio-overlay.dts > arch/arm64/boot/dts/freescale/imx8mm-venice-
gw7xxx-0x-gpio.dts

cpp -nostdinc -I include -I arch -undef -x assembler-with-cpp \
    arch/arm64/boot/dts/freescale/imx8mm-venice-gw7xxx-0x-gpio.dts \

```

```
imx8mm-venice-gw7xxx-0x-gpio.dts.tmp  
  
dtc -@ -i include/ -I dts -O dtb -o imx8mm-venice-gw7xxx-0x-gpio.dtbo \  
imx8mm-venice-gw7xxx-0x-gpio.dts.tmp
```

Sound file conversions

The following needs to be performed on any sound files that are needed from the original Kings III recordings to be able to use them with the GW16157 audio interface.

```
sox input.wav output.wav rate -v 48000  
done
```