antsdr_doc_en

Release v0.1

microphase

Feb 17, 2023

CONTENTS:

CHAPTER

ONE

DEVICE AND USAGE MANUAL

1.1 E200 Getting Started Guides

1.1.1 E200 Overview

The main CPU of the E200 is a Xilinx Zynq SoC XC7Z020. It is both a dual-core ARM Cortex A9 CPU and Artix-7 FPGA on a single die.

The programmable logic (PL, or FPGA) section of the SoC is responsible for handling all sampling data, DMA connections, and any other high-speed utility such as DDC/DUC. The processing system (PS, or CPU) is running a custom-build buildroot Linux operating system. The OS is responsible for all the device and peripheral management.

- Hardware Capabilities:
  - Xilinx Zynq SoC with dual-core ARM Cortex A9 and Artix-7 FPGA (XC7Z020)
  - Analog Devices AD9361 RFIC direct-conversion transceiver
  - 1 Gigabit Ethernet port
  - External PPS/10M reference input
  - 2 transmitters and 2 receivers, half or full duplex
  - Flexible rate, 12-bit ADC and DAC
  - Integrated RF frontend (AD9361: 70 MHz - 6 GHz, AD9363: 325MHz - 3.8GHz)
  - Variable analog bandwidth (AD9361: 200 kHz - 56 MHz, AD9363: 200KHz - 20MHz)
- Software Capabilities:
  - Full Linux system running on the ARM core
  - Communication between host computer and E200 is through libiio or uhd interface.

1.1.2 Getting Started

This will run you through the first steps relevant to getting your E200 up and running. Note: This guide was creating on an Ubuntu 20.04 machine, and other distributions or OS's may have different names/methods.

- Required equipment
  - one SD card
  - one ethernet cable
  - one usb Type-C cable

IIO based firmware

Download and update firmware

You can download the latest firmware form our antsdr-fw-patch github release.

Unzip the released file, and copy the boot image into a sd card (the sd card need fat32 partition). And then, insert the SD card into the device. Power on the board.

Install drivers on host

- Install dependencies

```
sudo apt install libxml2 libxml2-dev bison flex cmake git libaio-dev libboost-all-
↪dev \
libusb-1.0-0-dev libavahi-common-dev libavahi-client-dev doxygen bison flex cmake␣
↪git \
libgmp-dev liborc-dev swig
```

- Download and build libiio

```
git clone https://github.com/analogdevicesinc/libiio.git
cd libiio
mkdir build
cd build
cmake .. -DPYTHON_BINDINGS=ON
make
sudo make install
sudo ldconfig
cd ../..
```

- Download and build libad9361-iio

```
git clone https://github.com/analogdevicesinc/libad9361-iio.git
cd libad9361-iio
mkdir build
cd build
cmake .. -DPYTHON_BINDINGS=ON
make
sudo make install
sudo ldconfig
cd ../..
```

Verifying the Connection

Plug an Ethernet cable into the Ethernet port of E200. Connect the other end of the Ethernet cable to the Ethernet port on your computer.

The devices has a static ip address of 192.168.1.10, you should set your host ethernet port ip address to the same domain, such as 192.168.1.100.

Open the Network and Sharing Center. Click on Change adapter settings. Right-click on the Ethernet adapter and select Properties. Select Internet Protocol Version 4 (TCP/IPv4) and click the Properties button. Select the Use the following IP address option Enter the IP address as 192.168.1.100. Enter the Subnet mask as 255.255.255.0. Enter the Default gateway as 192.168.1.1. Click OK to save the settings.

You can `ping 192.168.1.10` to check the network connection, then you can use `iio_info -u "ip:192.168.1.10"` to get the device's information.

UHD based firmware

Download and update firmware

You can download the latest firmware form our antsdr-uhd github release. Unzip the released file, and copy the boot image into a sd card (the sd card need fat32 partition). And then, insert the SD card into the device. Power on the board.

Install drivers on host

- Install dependencies

```
sudo apt-get install autoconf automake build-essential ccache cmake cpufrequtils␣
↪doxygen ethtool \
g++ git inetutils-tools libboost-all-dev libncurses5 libncurses5-dev libusb-1.0-0␣
↪libusb-1.0-0-dev \
libusb-dev python3-dev python3-mako python3-numpy python3-requests python3-scipy␣
↪python3-setuptools \
python3-ruamel.yaml
```

- Getting the source code

```
git clone https://github.com/MicroPhase/antsdr_uhd
```

  Our source code repository contains of the following branch or tags: master: This is the main development branch, the uhd version is 4.1.0.0 for this branch now. uhd-3.15-release is for uhd 3.15-LTS.

Install drivers on host

```
cd antsdr_uhd/host
mkdir build
cd build
```

The default install path for uhd is **/usr/local**, if you want a custom install prefix, configuration variables can be passed into CMake via the command line.

For a custom install prefix: **-DCMAKE_INSTALL_PREFIX=**, for example:

```
cmake -DCMAKE_INSTALL_PREFIX=/usr ../
```

In this guide, we use the default path.

```
cmake ../
make -j$(nproc)
sudo make install
sudo ldconfig
```

Verifying the Connection

You can set your computer's ip address to `192.168.1.x`. Then you can use `uhd_find_devices` and `uhd_usrp_probe` to get the information of the board.

CHAPTER

TWO

APP NOTES

2.1 How to set ip or mac address

If you want to use multiple ANTSDR-E200 devices when using plutosdr compatible firmware, you will need this guide to set the mac and ip address.

2.1.1 QSPI Flash boot mode

Set mac address

1. You can enter the linux system of E200 through ssh or serial port tool. The default ip is **192.168.1.10** and the baud rate of the serial port is **115200**. when using the network port, please make sure you can ping the board.

   The **default username and password**:

   username:root

   password:analog

2. After enter the linux system, Set the mac address **using the command fw_setenv**, for example.

```
fw_setenv ethaddr 00:0a:35:00:08:30
```

3. Then enter the command reboot to restart the device or re-power.

```
reboot
```

4. If nothing else, Enter system and you can use the ifconfig command to check that the mac address has been successfully modified.

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:08:30
          inet addr:192.168.1.10  Bcast:0.0.0.0  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1065 (1.0 KiB)  TX bytes:2904 (2.8 KiB)
          Interrupt:29 Base address:0xb000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
```

(continues on next page)

(continued from previous page)

```
        UP LOOPBACK RUNNING  MTU:65536  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

usb0      Link encap:Ethernet  HWaddr 00:05:F7:19:E7:93
        inet addr:192.168.2.1  Bcast:0.0.0.0  Mask:255.255.255.0
        UP BROADCAST MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

The mac address of the network card eth0 has been successfully modified.

Set ip

The default ip is 192.168.1.10, if you need to modify the ip, you also need to enter the system.

1. If you just need to temporarily modify the ip address, you can use the **ifconfig** command like modifying the ip on the ubuntu system is enough.

   However, this method will restore the default ip address(192.168.1.10) after restarting the device.

2. If you want to permanently modify the ip address, you must use the **fw_setenv** command such as this.

   ```
   fw_setenv ipaddr_eth 192.168.2.1
   ```

3. Then restart the device.

2.1.2 SD card boot mode

Set mac address

1. If it is in the sd card boot mode, then you need to modify the sd card uEnv.txt file. Find `ethaddr=00:0a:35:00:01:22` and change it `ethaddr=xx:xx:xx:xx:xx:xx`

   Then start the device.

Set ip

The method of changing ip by starting sd card is the same as starting and changing ip by qspi.

CHAPTER

THREE

ABOUT MICROPHASE

MicroPhase develops FPGA development Platforms, SoMs and Software Define Radio Platforms for engineering teams, scientists and system integrators for algorithm development, prototyping and developing wireless technologies across a wide variety of applications.