YunSDR(PCIE 加速卡) C API

开发指南

Rev. 1.3

威视锐旗下品牌：

Sora Station　V3SDR　RED CYCLONE　ZING Zynq SoC Expert

修订记录

版本	修订日期	修订内容
1.0	2018 年 9 月 18 日	初始版本
1.1	2019 年 1 月 15 日	
1.2	2020 年 6 月 10 日	增加更多的函数说明及加速卡设备测试描述
1.3	2021 年 12 月 20 日	删除 SFP+和旧的 PCIE 接口

关于威视锐科技

北京威视锐科技有限公司是专注于软件定义无线通信（SDR）系统仿真、验证和测试平台的研发与生产，同时也提供通用高性能信号处理板卡，应用于机器视觉、生命科学和高能物理等科学计算领域。

威视锐与微软研究院联合开发的 Sora 软件无线电平台已经成为世界上知名大学和科研结构开展无线通信研究的首选平台，也是学术研究领域全球唯一的 100%基于 x86 的宽带实时软件无线电平台，目前已经有超过 20 多个国家的 300 多个科研用户。作为全球最大的可编程器件（FPGA）公司 XILINX 的全球合作伙伴，威视锐科技提供基于 XILINX FPGA/SoC 的全方位解决方案。特别是 ZYNQ 7000 系列 SoC 产品，威视锐携手 XILINX 发布了全球第一款基于 ZYNQ SoC 的低成本开源 SoC 模块 SNOWLeo，性价比远高于国外同类产品，大大降低了 SoC 系统的开发门槛。此外，威视锐旗下的红色飓风开发板自从 2004 年发售以来已成为国内销售时间最长、知名度最高以及用户量最多的开发板产品。

多年以来，威视锐科技坚持“Innovation for Research”的发展理念，与国内众多知名高校建立合作关系，帮助专家、学者和研发工程师将创新的理念变成现实。对于产业界客户，威视锐提供严格验证的核心模块、智能便携的测量仪器以及定制化的设计服务来加快产品研发周期。

目录

本指南适用于 Y5x0、Y7x0、IQX7x00 等系列设备，Y 系列设备通过加速卡与 PC 机连接，IQX 系列板设备为 PCIE 接口设备，直接插在 PC 机的 PCIE 卡槽上。

1. libyunsdr windows 下使用说明

软件需求：Windows10　Cmake 3.16.0　VS2019

1.1 代码及工具下载

下载 libyunsdr-master、cmake-3.16.0-win64-x64（自行下载即可），如下图所示：

https://github.com/v3tech/libyunsdr

libyunsdr的最新代码一般会包含在产品光盘中，优先使用光盘中的代码。

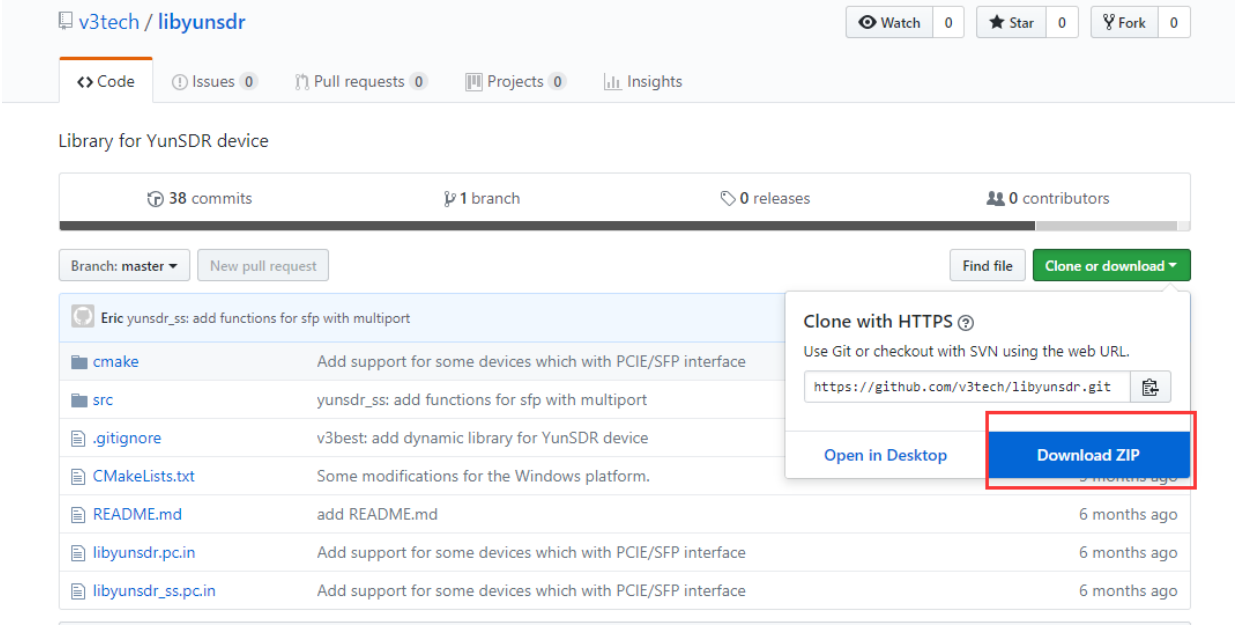


图 1 下载 libyunsdr-master

1.2 代码配置

在libyunsdr文件夹中建立build文件夹，打开cmake-gui，填写代码路径及build编译路径，如下图所示：

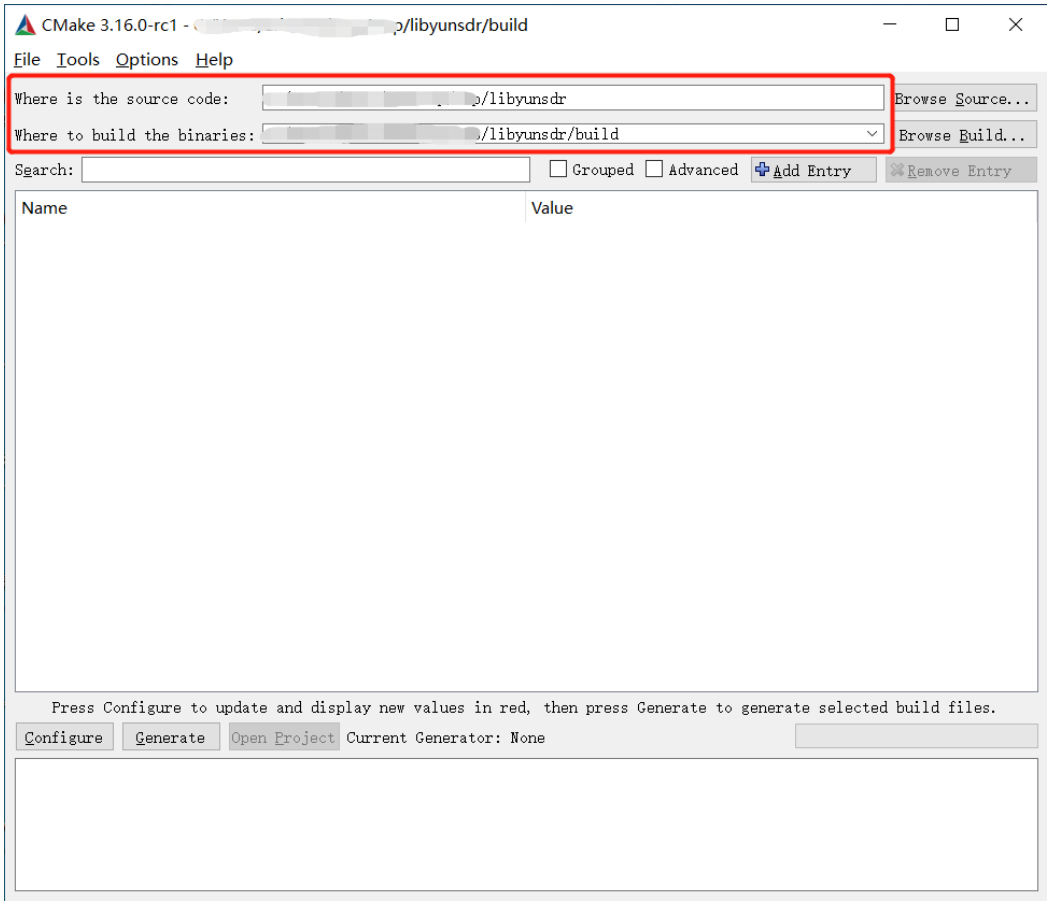


图 2 CMake 初始配置

然后点击Configure按钮，选择目标编译工具为VS2019，CMake默认生成64bit的工程配置，然后点击Finish，等待配置结束；

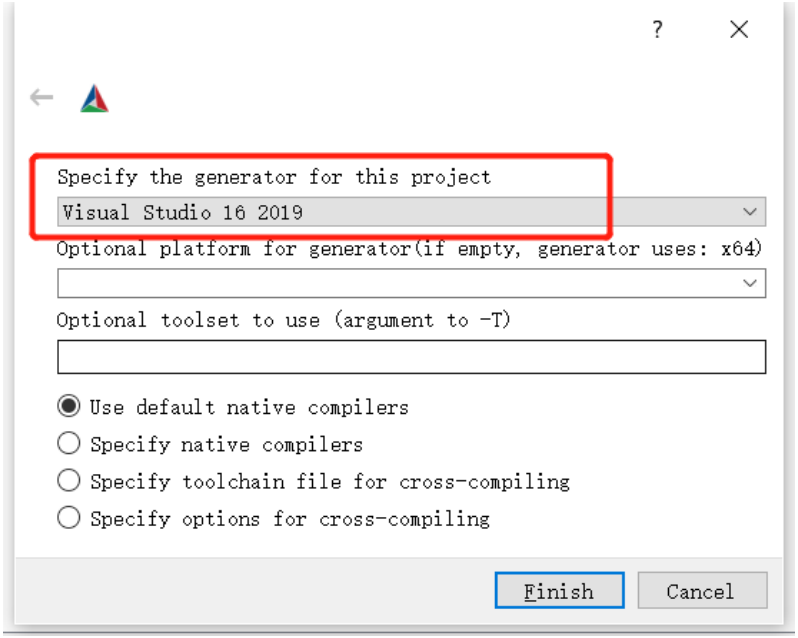


图 3 CMake 编译工具选择

配置结束后更新两个路径：

CMAKE_INSTALL_PREFIX：编译文件的安装路径;

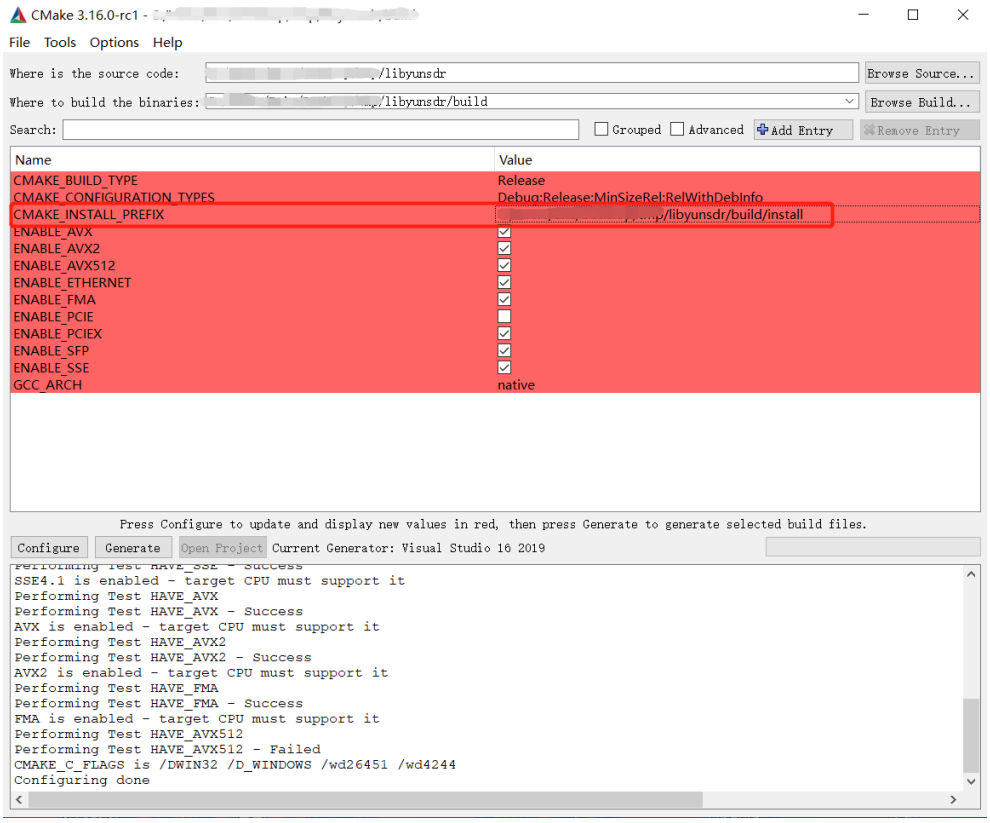
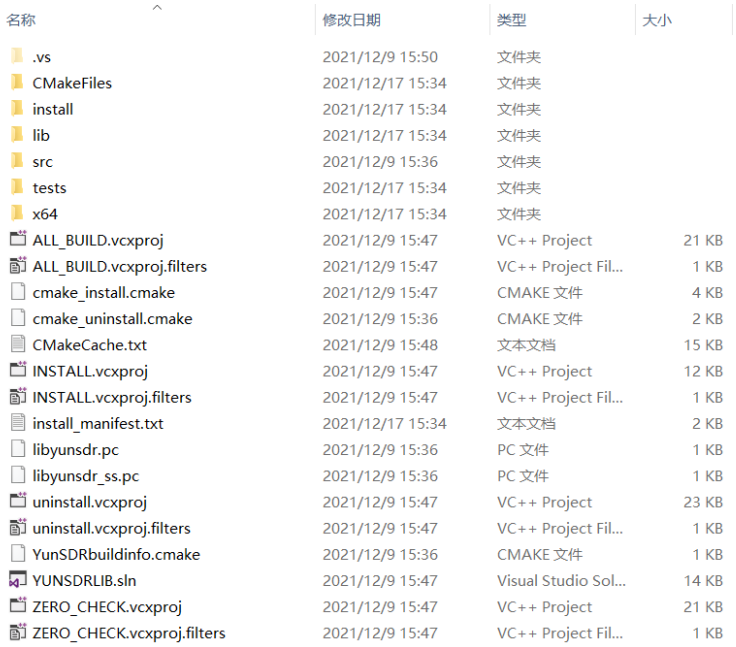


图 4 CMake 路径更新

路径更新完成后再次点击Configure，完成后点击Generate；



名称	修改日期	类型	大小
.vs	2021/12/9 15:50	文件夹	
CMakeFiles	2021/12/17 15:34	文件夹	
install	2021/12/17 15:34	文件夹	
lib	2021/12/17 15:34	文件夹	
src	2021/12/9 15:36	文件夹	
tests	2021/12/17 15:34	文件夹	
x64	2021/12/17 15:34	文件夹	
ALL_BUILD.vcxproj	2021/12/9 15:47	VC++ Project	21 KB
ALL_BUILD.vcxproj.filters	2021/12/9 15:47	VC++ Project Fil...	1 KB
cmake_install.cmake	2021/12/9 15:47	CMAKE 文件	4 KB
cmake_uninstall.cmake	2021/12/9 15:36	CMAKE 文件	2 KB
CMakeCache.txt	2021/12/9 15:48	文本文档	15 KB
INSTALL.vcxproj	2021/12/9 15:47	VC++ Project	12 KB
INSTALL.vcxproj.filters	2021/12/9 15:47	VC++ Project Fil...	1 KB
install_manifest.txt	2021/12/17 15:34	文本文档	2 KB
libyunsdr.pc	2021/12/9 15:36	PC 文件	1 KB
libyunsdr_ss.pc	2021/12/9 15:36	PC 文件	1 KB
uninstall.vcxproj	2021/12/9 15:47	VC++ Project	23 KB
uninstall.vcxproj.filters	2021/12/9 15:47	VC++ Project Fil...	1 KB
YunSDRbuildinfo.cmake	2021/12/9 15:36	CMAKE 文件	1 KB
YUNSDRLIB.sln	2021/12/9 15:47	Visual Studio Sol...	14 KB
ZERO_CHECK.vcxproj	2021/12/9 15:47	VC++ Project	21 KB
ZERO_CHECK.vcxproj.filters	2021/12/9 15:47	VC++ Project Fil...	1 KB

图 5 CMake 生成的 vs2019 工程

libyunsdr/build目录下即生成的VS2019工程。

1.3 代码编译

Cmake配置工程成功后build目录下会生成相应vs工程项目文件，用VS2019打开YUNSDRLIB.sln，选择Release模式，如下图所示：

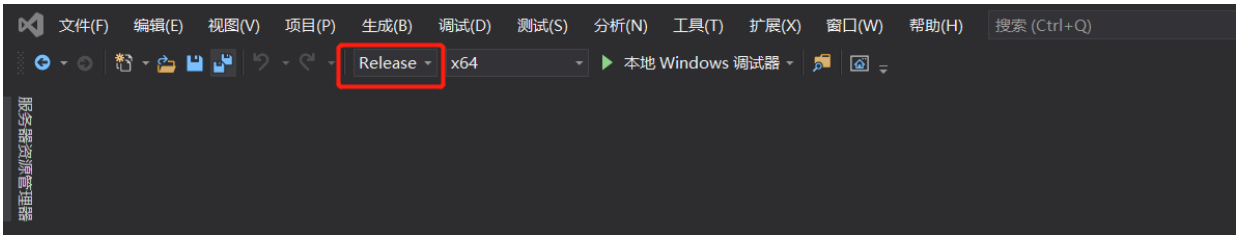


图 6 选择release模式

选择INSTALL点击右键，选择”重新生成”，如下图所示：

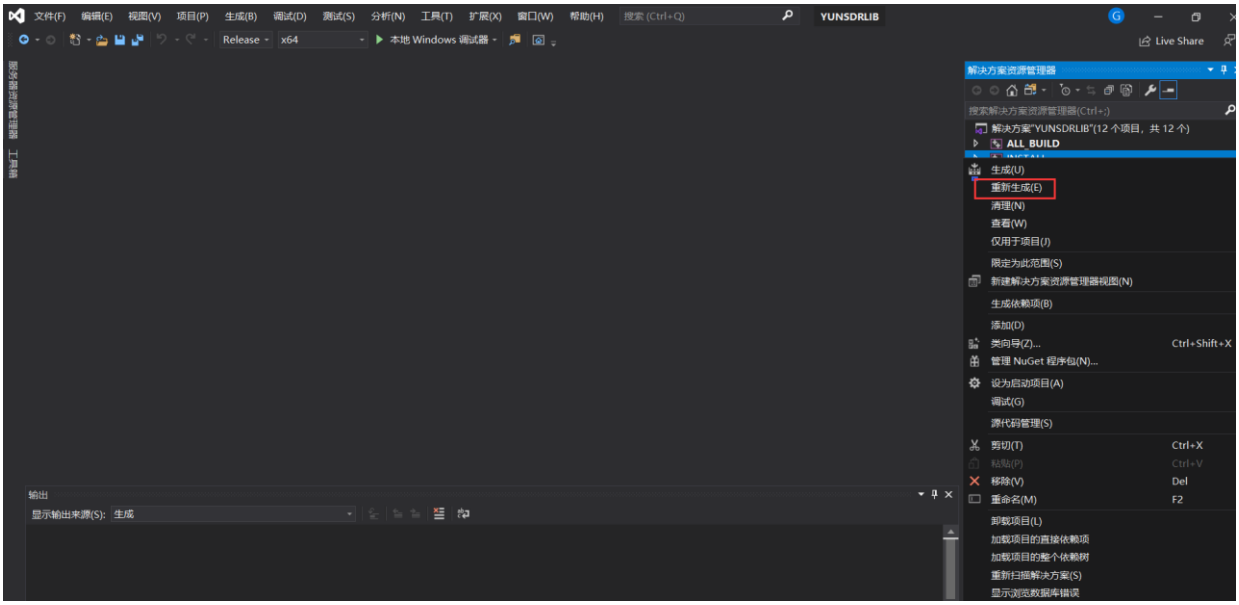


图 7 生成可执行文件和库文件

1.4 代码测试

把build\install\tests\ yunsdr_ss\yunsdr_ss_txrx_multiport.exe

build\install\lib\libyunsdr_ss.dll

拷贝到同一目录下。

powershell输入：.\yunsdr_ss_txrx_multiport.exe -a "pciex:0" -g 5 -G 70 -c 0x3 -C 0x3 -N 30720 -s 307200000

如下图所示：

-a "pciex:0" :

"pciex: "表示为加速卡连接方式的设备；"0"表示第一个加速卡设备。

-g表示RX增益参数，按需设置；一般范围为0~60dB

-G表示TX增益参数；一般范围为0~89dB

-c表示接收通道掩码，0x3指使用RX的前两个通道

-C表示发送通道掩码，0x3指使用TX的前两个通道

-N表示单帧数据采样点数，一般在TRX测试中设置为采样率/1000;

-s表示采样率，按需设置；

其他更多选项使用以下参数获取：

.\yunsdr_ss_txrx_multiport.exe -h

```
PS C:\Users\lichen\Desktop\pciex_test> .\yunsdr_ss_txrx_multiport.exe -a 'pciex:0' -g 5 -G 70 -c 0x3 -C 0x3 -N 3
0720 -s 30720000
Opening RF device...
Using pcie xdma interface.
Using args: 0
sys_info.dwPageSize: 4096
Device base path: \\?\pci#ven_10ee&dev_7028&subsys_000710ee&rev_00#4&22817009&0&0008#{74c7e4a9-6d5d-4a70-bc0d-20
691dff9e9d}
samples stream format is s16
device type: Y7000
-------------- GPS Information Start -------------
CST:                    0-0-0 08:00:00
receiver_mode:          0
disciplining_mode:      0
minor_alarms:           0
gnss_decoding_status:   0
disciplining_activity:  0
pps_indication:         0
pps_reference:          0
longtitude:             0.000000
latitude:               0.000000
altitude:               0.000000
---------------- GPS Information End --------------
-------------- RF Configuration Range -------------
Sampling frequency range: [122880000Hz, 960000Hz]
TX lo frequency range:    [5900000000Hz, 75000000Hz]
RX lo frequency range:    [5900000000Hz, 75000000Hz]
TX Gain range:            [60dB, 0dB]
RX Gain range:            [60dB, 0dB]
-------------------------------------------------
actual tx_lo:       2400.00 MHz
actual sample rate: 30.72 MHz
Tx1 attenuation:    10000 mdB
-------------- RF Current Configuration -------------
Subframe len:   30720 samples
Time advance:   0.000000 us
Set TX/RX rate: 30.72 MHz
Set RX gain:    5.0 dB
Set TX gain:    70.0 dB
Set TX/RX freq: 2400.00 MHz
-------------------------------------------------
Rx subframe 0
pciex_stream_recv3: Rx channel 0x0 change to 0x3
Rx subframe ts00 + flen30720 != tstamp59118450
Rx subframe 1
Rx subframe 2
Rx subframe 3
Rx subframe 4
Rx subframe 5
Rx subframe 6
Rx subframe 7
Rx subframe 8
Transmitting Signal
Rx subframe 9
Rx subframe 10
Rx subframe 11
Rx subframe 12
Rx subframe 13
Rx subframe time: 59517810
Rx subframe 14
Rx subframe 15
Rx subframe 16
Rx subframe 17
Rx subframe 18
Rx subframe 19
-------------- TRX Channel Event -------------
TX0 Channel timeout:     0
TX0 Channel underflow:   0
RX0 Channel overflow:    0
TX1 Channel timeout:     0
TX1 Channel underflow:   0
RX1 Channel overflow:    0
-------------------------------------------------
Done
PS C:\Users\lichen\Desktop\pciex_test>
```

图 8 加速卡设备测试运行结果

程序运行完成后将自动按照通道使能设置生成对应通道的采样数据文件，如下如所示：

名称	修改日期	类型	大小
rx_iq_samples_ch0.dat	2020/6/9 11:07	DAT 文件	120 KB
rx_iq_samples_ch1.dat	2020/6/9 11:07	DAT 文件	120 KB

图 9 加速卡设备数据文件生成

在执行文件yunsdr_ss_txrx_ multiport.exe相同目录下会生成文件rx_iq_samples_ch[x].dat（[x]代表通道号），从下图所示路径找到check.m和read_short_binary.m文件，拷贝这三个文件到同一目录下，然后修改check.m中的数据文件路径，最后在matlab下运行check.m即可，如下图所示：

libyunsdr › src › yunsdr_ss › tests　　搜索"tests"

名称	修改日期	类型	大小
check.m	2020/6/3 15:53	M 文件	1 KB
CMakeLists.txt	2020/6/3 15:53	文本文档	3 KB
read_short_binary.m	2020/6/3 15:53	M 文件	2 KB
wingetopt.c	2020/6/3 15:53	C Source	2 KB
wingetopt.h	2020/6/3 15:53	C/C++ Header	1 KB
yunsdr_ss_rx.c	2020/6/4 16:16	C Source	13 KB
yunsdr_ss_tx.c	2020/6/4 16:16	C Source	12 KB
yunsdr_ss_txrx.c	2020/6/4 18:02	C Source	9 KB
yunsdr_ss_txrx_multiport.c	2020/6/4 16:42	C Source	15 KB

图 10 check.m和read_short_binary.m位置

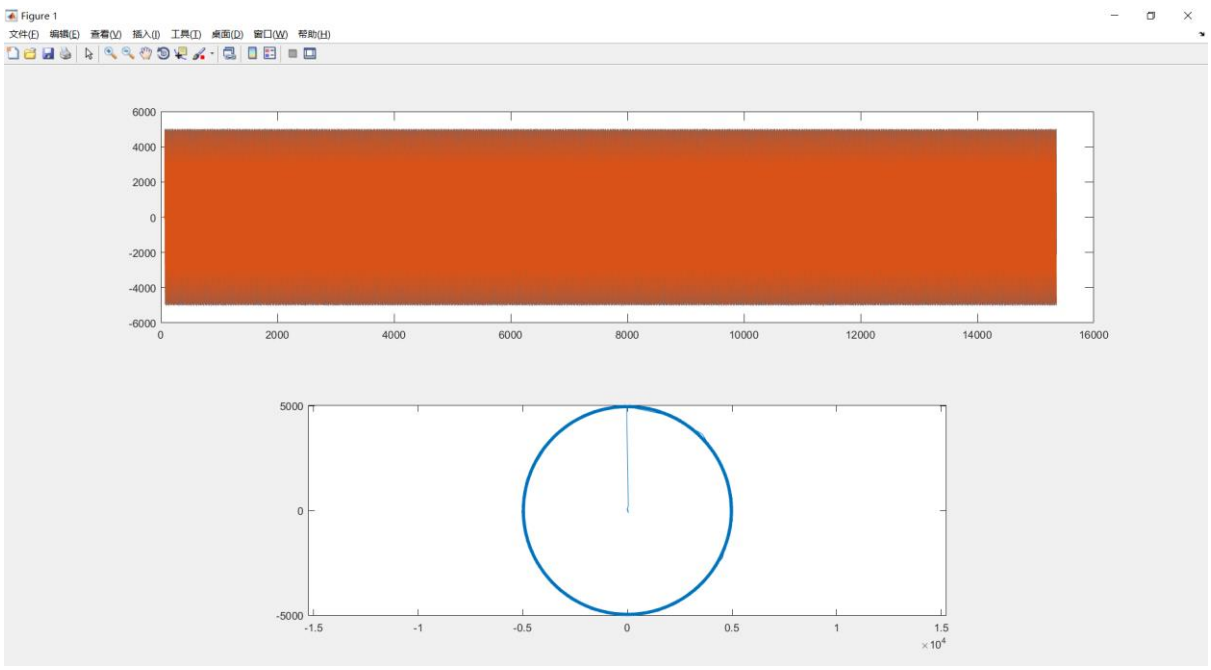


图 11 MATLAB 运行结果

2. libyunsdr 在 Linux 下编译

软件需求：Ubuntu18.04.6 以上　Cmake 3.16 以上

2.1 编译加载驱动(加速卡设备)

$ git clone https://github.com/Xilinx/dma_ip_drivers

$ cd dma_ip_drivers

$ git checkout 2019.2

$ cd XDMA/linux-kernel/xdma

$ make clean && make

$ cd ../tests

首先编辑 load_driver.sh

```
10  lsmod | grep xdma
11  if [ $? -eq 0 ]; then
12     rmmod xdma
13  fi
14  echo -n "Loading xdma driver..."
15  # Use the following command to Load the driver in the default
16  # or interrupt drive mode. This will allow the driver to
17  # interrupts to signal when DMA transfers are completed.
18  insmod ../xdma/xdma.ko
19  # Use the following command to Load the driver in Polling
20  # mode rather than than interrupt mode. This will allow the
21  # driver to use polling to determ when DMA transfers are
22  # completed.
23  #insmod ../xdma/xdma.ko poll_mode=1
24
25  if [ ! $? == 0 ]; then
26    echo "Error: Kernel module did not load properly."
27    echo " FAILED"
```

注释掉这行

这一行取消注释

然后再执行 sudo ./load_driver.sh

附：注意如果 Linux 的内核版本较新，此驱动未经签名是无法加载成功的，可以去 BIOS 设置里关闭安全启动即可。

2.2 编译 libyunsdr

$ git clone https://github.com/v3tech/libyunsdr (或者直接使用光盘目录中的代码)

$ cd libyunsdr

$ mkdir build

$ cd build

$ cmake ../

$ make && sudo make install && sudo ldconfig

2.3 代码测试

与 win10 下测试方法相同：

$ cd libyunsdr/build

$ ./test/yunsdr_ss_txrx_ multiport.exe -a "pciex:0" -g 5 -G 70 -c 0x3 -C 0x3 -N 30720 -s 307200000

3. YunSDR C/C++ API

3.1 代码路径

API 相关代码所在目录：libyunsdr/src/yunsdr_ss/include/yunsdr_api_ss.h

3.2 代码说明

3.2.1 数据结构定义

PPSModeEnum

```
typedef enum pps_enable {
    PPS_INTERNAL_EN,
    PPS_GPS_EN,
    PPS_EXTERNAL_EN,
}PPSModeEnum;
```

描述：

PPS_INTERNAL_EN：使能内部 PPS（由内部 FPGA 逻辑提供）

PPS_GPS_EN：使能 GPS 提供的 PPS （由板载 GPS 提供，如果设备包含 GPS 模块）

PPS_EXTERNAL_EN：使能外部 PPS（由外部其他设备提供）

DEV_CFG

```
typedef enum device_configuration {
    SingleSubDevSingleRF    = 0x1010,
    SingleSubDevDualRF      = 0x1020,
    SingleSubDevQuadRF      = 0x1040,
    DualSubDevDualRF        = 0x2011,
    DualSubDevTriple1RF     = 0x2012,
    DualSubDevTriple2RF     = 0x2021,
    DualSubDevQuadRF        = 0x2022,
    DualSubDevOctoRF        = 0x2044,
    NULLSubDevNULLRF    = 0,
}DEV_CFG;
```

描述：

SingleSubDevSingleRF：

系统有 1 个 YunSDR 子设备，每个设备包含 1 个射频芯片

SingleSubDevDualRF：

系统有 1 个 YunSDR 子设备，每个设备包含 2 个射频芯片

SingleSubDevQuadRF：

系统有 1 个 YunSDR 子设备，每个设备包含 4 个射频芯片

DualSubDevDualRF：

系统有 2 个 YunSDR 子设备，每个设备包含 1 个射频芯片

DualSubDevTriple1RF：

系统有 2 个 YunSDR 子设备，第一个子设备包含 2 个射频芯片，第二个子设备包含 1 个射频芯片

DualSubDevTriple2RF：

系统有 2 个 YunSDR 子设备，第一个子设备包含 1 个射频芯片，第二个子设备包含 2 个射频芯片

DualSubDevQuadRF：

系统有 2 个 YunSDR 子设备，每个设备包含 2 个射频芯片

DualSubDevOctoRF：

系统有 2 个 YunSDR 子设备，每个设备包含 4 个射频芯片

NULLSubDevNULLRF：

系统没有检测到 YunSDR 子设备

GPS_STATUS

```
typedef struct {
    uint8_t     receiver_mode;
    uint8_t     disciplining_mode;
    uint16_t    minor_alarms;
    uint8_t     gnss_decoding_status;
    uint8_t     disciplining_activity;
    uint8_t     pps_indication;
    uint8_t     pps_reference;
} GPS_STATUS;
```

描述：

GPS 信息，详见产品资料里包含的 GPS 模块手册。

TIME_24H

```
typedef struct {
    uint8_t second;
    uint8_t minute;
    uint8_t hour;
    uint8_t day;
    uint8_t month;
    uint32_t year;
} TIME_24H;
```

描述：

GPS 时间信息。

3.2.2 时间量转换操作

yunsdr_ticksToTimeNs ( )

uint64_t **yunsdr_ticksToTimeNs** (
const uint64_t ticks,
const double rate)

描述：

采样点数转换为时间量，单通道每个采样点为32bit，双通道每个采样点为64bit

输入参数：

ticks:采样点数

rate:当前射频前端采样率

返回值：

纳秒数

yunsdr_timeNsToTicks ( )

uint64_t **yunsdr_timeNsToTicks** (
const uint64_t timeNs,
const double rate)

描述：

时间量转换为采样点数，单通道每个采样点为32bit，双通道每个采样点为64bit

输入参数：

timeNs:时间量，纳秒

rate:当前射频前端采样率

返回值：

采样点数

3.2.3 设备级操作

设备打开、关闭、枚举。

yunsdr_open_device ( )

YUNSDR_DESCRIPTOR ***yunsdr_open_device** (
const char *uri)

描述：

打开yunsdr设备

输入参数：

uri: 加速卡设备，从0开始代表第一个设备，如“pciex:0”

附带参数：“nsamples_recv_frame”用于指定接收数据MTU长度，例如：“pciex:0, nsamples_recv_frame:7680”表示打开第一个PCIE设备并将接收数据MTU指定为7680个采样点(八通道设备中多通道接收数据推荐使用此附带参数)。

返回值：

yunsdr设备描述符，失败返回NULL

yunsdr_close_device ( )

int32_t **yunsdr_close_device** (
YUNSDR_DESCRIPTOR *yunsdr)

描述：

关闭yunsdr设备

输入参数：

yunsdr:将要关闭的设备描述符

返回值：

成功返回0，失败返回非零值

yunsdr_get_device_configuration ( )

int32_t **yunsdr_get_device_configuration** (
YUNSDR_DESCRIPTOR *yunsdr,
DEV_CFG *cfg);

描述：

获取设备组合方式

输入参数：

yunsdr:设备描述符

输出参数：

cfg：设备组合方式，详情见DEV_CFG定义。

返回值：

成功返回0，失败返回非零值

yunsdr_get_firmware_version ( )

int32_t **yunsdr_get_firmware_version** (
YUNSDR_DESCRIPTOR *yunsdr,
uint32_t *version);

描述：

读取固件版本号

输入参数：

yunsdr:设备描述符

输出参数：

version:版本号，低16位是FPGA固件版本号，高16位是软件固件版本号。

返回值：

成功返回0，失败返回非零值

yunsdr_get_model _version ( )

int32_t **yunsdr_get_model _version** (

YUNSDR_DESCRIPTOR *yunsdr,

uint32_t *version);

描述：

读取设备型号

输入参数：

yunsdr:设备描述符

输出参数：

version:设备型号，550、590、750、780、7100、7400等

返回值：

成功返回0，失败返回非零值

3.2.4GPS 操作

GPS 状态、时间、经纬度获取。

yunsdr_get_gps_status ( )

int32_t **yunsdr_get_gps_status** (
YUNSDR_DESCRIPTOR *yunsdr,
uint8_t dev_id,
GPS_STATUS *g_status);

描述：

获取GPS状态信息

输入参数：

yunsdr:设备描述符

dev_id:子设备ID

输出参数：

g_status：GPS状态信息，详情见GPS_STATUS定义。

返回值：

成功返回0，失败返回非零值

yunsdr_get_utc ( )

int32_t **yunsdr_get_utc** (
YUNSDR_DESCRIPTOR *yunsdr,
uint8_t dev_id,
TIME_24H *time);

描述：

获取GPS时间信息

输入参数：

yunsdr:设备描述符

rf_id:子设备ID

输出参数：

time：GPS时间信息，详情见TIME_24H定义。

返回值：

成功返回0，失败返回非零值

yunsdr_get_xyz ( )

int32_t **yunsdr_get_xyz** (
YUNSDR_DESCRIPTOR *yunsdr,
uint8_t dev_id,
double *longtitude,
double *latitude,
double *altitude);

描述：

获取GPS经纬度信息

输入参数：

yunsdr:设备描述符

dev_id:子设备ID

输出参数：

longtitude:GPS经度信息

latitude: GPS纬度信息

altitude: GPS高度信息

返回值：

成功返回0，失败返回非零值

3.2.5 射频参数配置范围获取

yunsdr_get_sampling_freq_range ( )

int32_t **yunsdr_get_sampling_freq_range** (
YUNSDR_DESCRIPTOR *yunsdr,
uint8_t dev_id,
uint32_t *sampling_freq_hz_max,
uint32_t *sampling_freq_hz_min)

描述：

获取射频采样率设置范围

输入参数：

yunsdr:设备描述符

dev_id:子设备ID

输出参数：

sampling_freq_hz_max:最大采样率，单位Hz

sampling_freq_hz_min:最小采样率，单位Hz

返回值：

成功返回0，失败返回非零值

yunsdr_get_rx_gain_range ( )

int32_t **yunsdr_get_rx_gain_range** (
YUNSDR_DESCRIPTOR *yunsdr,
uint8_t dev_id,
uint32_t *gain_db_max,
uint32_t *gain_db_min);

描述：

获取接收端增益设置范围

输入参数：

yunsdr:设备描述符

rf_id:子设备ID

输出参数：

gain_db_max:最大增益，单位dB

gain_db_min:最小增益，单位dB

返回值：

成功返回0，失败返回非零值

yunsdr_get_tx_gain_range ( )

int32_t **yunsdr_get_tx_gain_range** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t dev_id,

uint32_t *gain_db_max,

uint32_t *gain_db_min);

描述：

获取发送端增益设置范围

输入参数：

yunsdr:设备描述符

dev_id:子设备ID

输出参数：

gain_db_max:最大增益，单位dB

gain_db_min:最小增益，单位dB

返回值：

成功返回0，失败返回非零值

yunsdr_get_rx_freq_range ( )

int32_t **yunsdr_get_rx_freq_range** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t dev_id,

uint64_t *lo_freq_hz_max,

uint64_t *lo_freq_hz_min);

描述：

获取接收端中心频率设置范围

输入参数：

yunsdr:设备描述符

dev_id:子设备ID

输出参数：

lo_freq_hz_max:最大频率，单位Hz

lo_freq_hz_min:最小频率，单位Hz

返回值：

成功返回0，失败返回非零值

yunsdr_get_tx_freq_range ( )

int32_t **yunsdr_get_tx_freq_range** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t dev_id,

uint64_t *lo_freq_hz_max,

uint64_t *lo_freq_hz_min);

描述：

获取发送端中心频率设置范围

输入参数：

yunsdr:设备描述符

dev_id:子设备ID

输出参数：

lo_freq_hz_max:最大频率，单位Hz

lo_freq_hz_min:最小频率，单位Hz

返回值：

成功返回0，失败返回非零值

3.2.6 射频配置参数读取

yunsdr_get_tx1_attenuation ( )

int32_t **yunsdr_get_tx1_attenuation** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

uint32_t * attenuation_mdb);

描述：

获取tx1发送衰减

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

输出参数：

attenuation_mdb：读回的当前衰减值

返回值：

成功返回0，失败返回非零值

yunsdr_get_tx2_attenuation ( )

int32_t **yunsdr_get_tx2_attenuation** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

uint32_t *attenuation_mdb);

描述：

获取tx2发送衰减

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

输出参数：

attenuation_mdb:读回的当前衰减值

返回值：

成功返回0，失败返回非零值

yunsdr_get_tx_rf_bandwidth ( )

int32_t **yunsdr_get_tx_rf_bandwidth** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

uint32_t* bandwidth_hz);

描述：

获取发送带宽

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

输出参数：

bandwidth_hz:读回的当前带宽

返回值：

成功返回0，失败返回非零值

yunsdr_get_tx_sampling_freq ( )

int32_t **yunsdr_get_tx_sampling_freq** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

uint32_t *sampling_freq_hz);

描述：

获取发送采样频率

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

输出参数：

sampling_freq_hz:读回的当前采样率

返回值：

成功返回0，失败返回非零值

yunsdr_get_tx_lo_freq ( )

int32_t **yunsdr_get_tx_lo_freq** (
YUNSDR_DESCRIPTOR *yunsdr,
uint8_t rf_id,
uint64_t *lo_freq_hz);

描述：

获取发送中心频率

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

输出参数：

lo_freq_hz:读回的当前中心频率

返回值：

成功返回0，失败返回非零值

yunsdr_get_rx1_rf_gain ( )

int32_t **yunsdr_get_rx1_rf_gain** (
YUNSDR_DESCRIPTOR *yunsdr,
uint8_t rf_id,
int32_t *gain_db);

描述：

获取rx1接收增益

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

输出参数：

gain_db:读回的当前增益

返回值：

成功返回0，失败返回非零值

yunsdr_get_rx2_rf_gain ( )

int32_t **yunsdr_get_rx2_rf_gain** (
YUNSDR_DESCRIPTOR *yunsdr,
uint8_t rf_id,
int32_t *gain_db);

描述：

获取rx2接收增益

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

输出参数：

gain_db:读回的当前增益值

返回值：

成功返回0，失败返回非零值

yunsdr_get_rx_rf_bandwidth ( )

int32_t **yunsdr_get_rx_rf_bandwidth** (
YUNSDR_DESCRIPTOR *yunsdr,
uint8_t rf_id,
uint32_t *bandwidth_hz);

描述：

获取接收带宽

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

输出参数：

bandwidth_hz:读回的当前带宽

返回值：

成功返回0，失败返回非零值

yunsdr_get_rx_lo_freq ( )

int32_t **yunsdr_get_rx_lo_freq** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

uint64_t *lo_freq_hz);

描述：

获取接收中心频率

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

输出参数：

lo_freq_hz:读回的当前中心频率

返回值：

成功返回0，失败返回非零值

yunsdr_get_rx1_gain_control_mode ( )

int32_t **yunsdr_get_rx1_gain_control_mode** (
YUNSDR_DESCRIPTOR *yunsdr,
uint8_t rf_id,
RF_GAIN_CTRL_MODE *gc_mode);

描述：

获取rx1接收增益控制模式

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

输出参数：

gc_mode:读回的当前增益控制模式

0-- RF_GAIN_MGC

1-- RF_GAIN_FASTATTACK_AGC

2-- RF_GAIN_SLOWATTACK_AGC

返回值：

成功返回0，失败返回非零值

yunsdr_get_rx2_gain_control_mode ( )

int32_t **yunsdr_get_rx2_gain_control_mode** (
YUNSDR_DESCRIPTOR *yunsdr,
uint8_t rf_id,
RF_GAIN_CTRL_MODE *gc_mode);

描述：

获取rx2接收增益控制模式

参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

输出参数：

gc_mode:读回的当前增益控制模式

0-- RF_GAIN_MGC

1-- RF_GAIN_FASTATTACK_AGC

2-- RF_GAIN_SLOWATTACK_AGC

返回值：

成功返回0，失败返回非零值

3.2.7 射频参数配置

yunsdr_set_rx1_rf_gain ( )

int32_t **yunsdr_set_rx1_rf_gain** (

YUNSDR_DESCRIPTOR*yunsdr,

uint8_t rf_id,

int32_t gain_db);

描述：

设置rx1接收增益

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

gain_db:需要设置的增益值

返回值：

成功返回0，失败返回非零值

yunsdr_set_rx2_rf_gain ( )

int32_t **yunsdr_set_rx2_rf_gain** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

int32_t gain_db);

描述：

设置rx2接收增益

参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

gain_db:需要设置的增益值

返回值：

成功返回0，失败返回非零值

yunsdr_set_rx_rf_bandwidth ( )

int32_t **yunsdr_set_rx_rf_bandwidth** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

uint32_t bandwidth_hz);

描述：

设置接收带宽

参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

bandwidth_hz:需要设置的射频带宽

返回值：

成功返回0，失败返回非零值

yunsdr_set_rx_sampling_freq ( )

int32_t **yunsdr_set_rx_sampling_freq** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

uint32_t sampling_freq_hz);

描述：

设置接收采样频率

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

sampling_freq_hz:需要设置的射频采样频率

返回值：

成功返回0，失败返回非零值

yunsdr_set_rx_lo_freq ( )

int32_t **yunsdr_set_rx_lo_freq** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

uint64_t lo_freq_hz);

描述：

设置接收中心频率

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

lo_freq_hz:需要设置的射频中心频率

返回值：

成功返回0，失败返回非零值

yunsdr_set_rx1_gain_control_mode ( )

int32_t **yunsdr_set_rx1_gain_control_mode** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

RF_GAIN_CTRL_MODE gc_mode);

描述：

设置rx1接收增益控制模式

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

gc_mode:设置的当前增益控制模式

0-- RF_GAIN_MGC

1-- RF_GAIN_FASTATTACK_AGC

2-- RF_GAIN_SLOWATTACK_AGC

返回值：

成功返回0，失败返回非零值

yunsdr_set_rx2_gain_control_mode ( )

int32_t **yunsdr_set_rx2_gain_control_mode** (

YUNSDR_DESCRIPTOR *yunsdr,
uint8_t rf_id,
RF_GAIN_CTRL_MODE gc_mode);

描述：

设置rx2接收增益控制模式

参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

gc_mode:设置的当前增益控制模式

0-- RF_GAIN_MGC

1-- RF_GAIN_FASTATTACK_AGC

2-- RF_GAIN_SLOWATTACK_AGC

返回值：

成功返回0，失败返回非零值

yunsdr_set_rx_fir_en_dis ( )

int32_t **yunsdr_set_rx_fir_en_dis** (
YUNSDR_DESCRIPTOR *yunsdr,
uint8_t rf_id,
uint8_t enable);

描述：

设置接收滤波器使能

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

enable: 0关闭，1使能

返回值：

成功返回0，失败返回非零值

yunsdr_set_tx1_attenuation ( )

int32_t **yunsdr_set_tx1_attenuation** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

uint32_t attenuation_mdb);

描述：

设置tx1发送衰减

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

attenuation_mdb:需要设置的衰减值

返回值：

成功返回0，失败返回非零值

yunsdr_set_tx2_attenuation ( )

int32_t **yunsdr_set_tx2_attenuation** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

uint32_t attenuation_mdb);

描述：

设置tx2发送衰减

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

attenuation_mdb:需要设置的衰减值

返回值：

成功返回0，失败返回非零值

yunsdr_set_tx_rf_bandwidth ( )

int32_t **yunsdr_set_tx_rf_bandwidth** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

uint32_t bandwidth_hz);

描述：

设置发送带宽

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

bandwidth_hz:需要设置的带宽值

返回值：

成功返回0，失败返回非零值

yunsdr_set_tx_sampling_freq ( )

int32_t **yunsdr_set_tx_sampling_freq** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

uint32_t sampling_freq_hz);

描述：

设置发送采频样率

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

sampling_freq_hz:需要设置的采样频率

返回值：

成功返回0，失败返回非零

yunsdr_set_tx_lo_freq ( )

int32_t **yunsdr_set_tx_lo_freq** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

uint64_t lo_freq_hz);

描述：

设置发送中心频率

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

lo_freq_hz:需要设置的中心频率

返回值：

成功返回0，失败返回非零值

yunsdr_set_tx_fir_en_dis ( )

int32_t **yunsdr_set_tx_fir_en_dis** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,
uint8_t enable);

描述：

设置发送滤波器使能

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

enable: 0关闭，1使能

返回值：

成功返回0，失败返回非零值

3.2.8 设备级参数配置

yunsdr_set_ref_clock ( )

int32_t **yunsdr_set_ref_clock** (
YUNSDR_DESCRIPTOR *yunsdr,
uint8_t dev_id,
REF_SELECT select);

描述：

参考时钟选择

输入参数：

yunsdr:设备描述符

dev_id:子设备ID

select:参考时钟选择

INTERNAL_REFERENCE（内部时钟）

EXTERNAL_REFERENCE（外部时钟）

返回值：

成功返回0，失败返回非零值

yunsdr_set_vco_select ( )

int32_t **yunsdr_set_vco_select** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t dev_id,

VCO_CAL_SELECT select);

描述：

设置VCO类型

输入参数：

yunsdr:设备描述符

dev_id:子设备ID

select:VCO选择类型

ADF4001

AUXDAC1

返回值：

成功返回0，失败返回非零值

yunsdr_set_auxdac1 ( )

int32_t **yunsdr_set_auxdac1** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t dev_id,

uint32_t val);

描述：

设置AUXDAC1校准参数(当VCO选择AUXDAC1时)

输入参数：

yunsdr:设备描述符

dev_id:子设备ID

val:AUXDAC1校准参数mv (0-3000)

返回值：

成功返回0，失败返回非零值

yunsdr_set_pps_select ( )

int32_t **yunsdr_set_pps_select** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t dev_id,

PPSModeEnum pps);

描述：

设置PPS输入源

参数：

yunsdr:设备描述符

dev_id:子设备ID

pps:PPS输入源

PPS_INTERNAL_EN:使用内部PPS

PPS_GPS_EN:使用GPS提供的PPS

PPS_EXTERNAL_EN:使用外部PPS

返回值：

成功返回0，失败返回非零值

yunsdr_set_duplex_select ( )

int32_t **yunsdr_set_duplex_select** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t dev_id,

FDD_TDD_SEL select);

描述：

设置射频双工模式，部分产品型号不支持FDD模式，在不支持FDD模式的产品上配置此功能将不会生效。

参数：

yunsdr:设备描述符

dev_id:子设备ID

select:双工模式

TDD

FDD

返回值：

成功返回0，失败返回非零值

yunsdr_set_rx_ant_enable ( )

int32_t **yunsdr_set_rx_ant_enable** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t rf_id,

uint8_t enable);

描述：

RX端口使能，配合TDD模式和TRX端口的选择；

当设备支持TDD模式，并通过**yunsdr_set_duplex_select()**函数设置为TDD模式时：

✓ 收发通过复用TRX端口，TRX端口工作在发送模式还是接收模式取决于**yunsdr_set_rx_ant_enable**的enable参数是0还是1；

当设备支持FDD模式，并通过**yunsdr_set_duplex_select()**函数设置为FDD模式时：

✓ 应当分别使用TRX和RX端口，应使能RX端口，**yunsdr_set_rx_ant_enable**的enable参数应是1（默认情况）

输入参数：

yunsdr:设备描述符

dev_id:子设备ID

enable: 0 - RX，1 - TX

返回值：

成功返回0，失败返回非零值

yunsdr_tx_cyclic_enable ( )

int32_t **yunsdr_tx_cyclic_enable** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t dev_id,

uint8_t enable);

描述：

循环发送模式使能，调用本函数(enable为1)预使能后再调用yunsdr_write_samples_multiport将数据写入FPGA，然后再调用一次本函数(enable为3)使能FPGA逻辑将循环发送这一帧数据

输入参数：

yunsdr:设备描述符

rf_id:射频前端chip ID

enable: 0 - 不使能，1 - 预使能，3 - 使能

返回值：

成功返回0，失败返回非零值

注：此功能以实际设备固件是否支持为准。

yunsdr_enable_timestamp ( )

int32_t **yunsdr_enable_timestamp** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t dev_id,

uint8_t enable);

描述：

时戳使能，收发数据前必须调用此函数使能时戳。

输入参数：

yunsdr:设备描述符

dev_id:子设备ID

enable: 0 - 关闭，1 - 使能

返回值：

成功返回0，失败返回非零值

yunsdr_read_timestamp ( )

int32_t **yunsdr_read_timestamp** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t dev_id,

uint64_t *timestamp);

描述：

读取当前时戳计数

输入参数：

yunsdr:设备描述符

dev_id: 子设备ID

输出参数：

timestamp:返回当前时戳

返回值：

成功返回0，失败返回非零值

yunsdr_set_rxchannel_coef ( )

int32_t **yunsdr_set_rxchannel_coef** (
YUNSDR_DESCRIPTOR *yunsdr,
uint8_t dev_id,
RF_RX_CHANNEL channel,
int16_t coef1,
int16_t coef2);

描述：

设置接收端相位校准系数

输入参数：

yunsdr:设备描述符

dev_id:子设备ID

channel:接收通道

RX1_CHANNEL（RFCHIP0的RX1）

RX2_CHANNEL（RFCHIP0的RX2）

RX3_CHANNEL（RFCHIP1的RX1）

RX4_CHANNEL（RFCHIP1的RX2）

coef1:16bit校准系数

coef2: 16bit校准系数

返回值：

成功返回0，失败返回非零值

yunsdr_enable_rxchannel_corr ( )

int32_t **yunsdr_enable_rxchannel_corr** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t dev_id,

RF_RX_CHANNEL channel,

uint8_t enable);

描述：

使能接收端相位校准功能

参数：

yunsdr:设备描述符

dev_id: 子设备ID

channel:接收通道

RX1_CHANNEL（RFCHIP0的RX1）

RX2_CHANNEL（RFCHIP0的RX2）

RX3_CHANNEL（RFCHIP1的RX1）

RX4_CHANNEL（RFCHIP1的RX2）

enable:0禁用， 1使能

返回值：

成功返回0，失败返回非零值

yunsdr_set_txchannel_coef ( )

int32_t **yunsdr_set_txchannel_coef** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t dev_id,

RF_TX_CHANNEL channel,

int16_t coef1,

int16_t coef2);

描述：

设置发送端相位校准系数

参数：

yunsdr:设备描述符

dev_id: 子设备ID

channel:接收通道

TX1_CHANNEL（RFCHIP0的TX1）

TX2_CHANNEL（RFCHIP0的TX2）

TX3_CHANNEL（RFCHIP1的TX1）

TX4_CHANNEL（RFCHIP1的TX2）

coef1:16bit校准系数

coef2: 16bit校准系数

返回值：

成功返回0，失败返回非零值

yunsdr_enable_txchannel_corr ( )

int32_t **yunsdr_enable_txchannel_corr** (

YUNSDR_DESCRIPTOR *yunsdr,

uint8_t dev_id,

RF_TX_CHANNEL channel,

uint8_t enable);

描述：

使能发送端相位校准功能

参数：

yunsdr:设备描述符

dev_id: 射频前端chip ID

channel:接收通道

TX1_CHANNEL（RFCHIP0的TX1）

TX2_CHANNEL（RFCHIP0的TX2）

TX3_CHANNEL（RFCHIP1的TX1）

TX4_CHANNEL（RFCHIP1的TX2）

enable:0 - 禁用， 1 - 使能

返回值：

成功返回0，失败返回非零值

3.2.9 数据收发操作

yunsdr_read_samples_multiport ( )

int32_t **yunsdr_read_samples_multiport** (

YUNSDR_DESCRIPTOR *yunsdr,

void **buffer,

uint32_t count,

uint16_t channel_mask,

uint64_t *timestamp);

描述：

多通道采样帧接收

输入参数：

yunsdr:设备描述符

buffer:数据缓冲区数组，用于存储接收到的各通道采样数据

count:采样点数，每个采样点为4byte计数

channel_mask:接收通道掩码8bit，为1的位表示使能该通道接收。

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

RFChip3的RX1/RX2	RFChip2的RX1/RX2	RFChip1的RX1/RX2	RFChip0的RX1/RX2

注：16通道向高八位扩展。

输出参数：

buffer:数据缓冲区数组，存储接收到的各通道采样数据

timestamp:存储的是当前接收到数据的第一个采样点对应的时戳计数

返回值：

成功返回接收数据的采样点数，失败返回负值

yunsdr_read_samples_multiport_Matlab ( )

int32_t **yunsdr_read_samples_multiport_Matlab** (

YUNSDR_DESCRIPTOR *yunsdr,

void *buffer,

uint32_t count,

uint16_t channel_mask,

uint64_t *timestamp);

描述：

多通道采样帧接收(用于Matlab调用)

输入参数：

yunsdr:设备描述符

buffer:数据缓冲区，对应Matlab中的一维矩阵，大小必须为(32*count)Bytes，用于存储接收到的各通道采样数据；当此参数为NULL时，函数内部自动分配缓冲区，并将接收的数据按通道存储成IQ(int16)类型的二进制数据文件。

count:采样点数，每个采样点以4byte计数

channel_mask:接收通道掩码8bit，为1的位表示使能该通道接收。

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RFChip3的RX1/RX2		RFChip2的RX1/RX2		RFChip1的RX1/RX2		RFChip0的RX1/RX2	

注：16通道向高八位扩展。

输出参数：

timestamp:输出参数，存储的是当前接收到数据的第一个采样点对应的时戳计数

返回值：

成功返回接收数据的采样点数，失败返回负值

yunsdr_write_samples_multiport ( )

int32_t **yunsdr_write_samples_multiport** (

YUNSDR_DESCRIPTOR *yunsdr,

const void **buffer,

uint32_t count,

uint16_t channel_mask,

uint64_t timestamp,

uint32_t flags);

描述：

多通道采样帧发送函数

输入参数：

yunsdr:设备描述符

buffer:数据缓冲区数组，用于存储将要发送的各通道采样数据

count:采样点数，每个采样点以4byte计数

channel_mask:发送通道掩码8bit，为1的位表示使能该通道发送。

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RFChip3的TX1/TX2		RFChip2的TX1/TX2		RFChip1的TX1/TX2		RFChip0的TX1/TX2	

注：16通道向高八位扩展。

timestamp:为零立即发送，不为0时在timestamp的指定时刻发送

flags:TDD切换标记，默认为0

返回值：

成功返回发送数据的采样点数，失败返回负值

yunsdr_write_samples_multiport_Matlab ( )

int32_t **yunsdr_write_samples_multiport_Matlab** (

YUNSDR_DESCRIPTOR *yunsdr,

const void *buffer,

uint32_t count,

uint16_t channel_mask,

uint64_t timestamp,

uint32_t flags);

描述：

多通道采样帧发送(用于Matlab调用)

输入参数：

yunsdr:设备描述符

buffer:数据缓冲区，对应Matlab中的一维矩阵，大小必须为(32*count)Bytes，用于存储将要发送的各通道采样数据；

count:采样点数，每个采样点以4byte计数

channel_mask:发送通道掩码8bit，为1的位表示使能该通道发送。

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RFChip3的TX1/TX2		RFChip2的TX1/TX2		RFChip1的TX1/TX2		RFChip0的TX1/TX2	

注：16通道向高八位扩展。

timestamp:为零立即发送，不为0时在timestamp的指定时刻发送

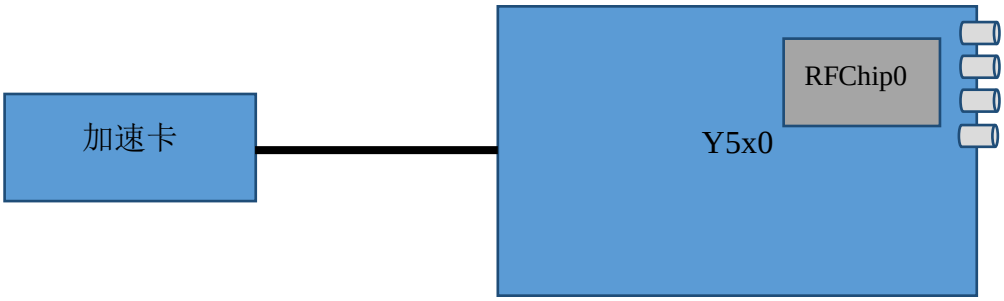
返回值：

成功返回发送数据的采样点数，失败返回负值

3.3 调用参考

3.3.1 单板单射频(2T2R)

拓扑结构

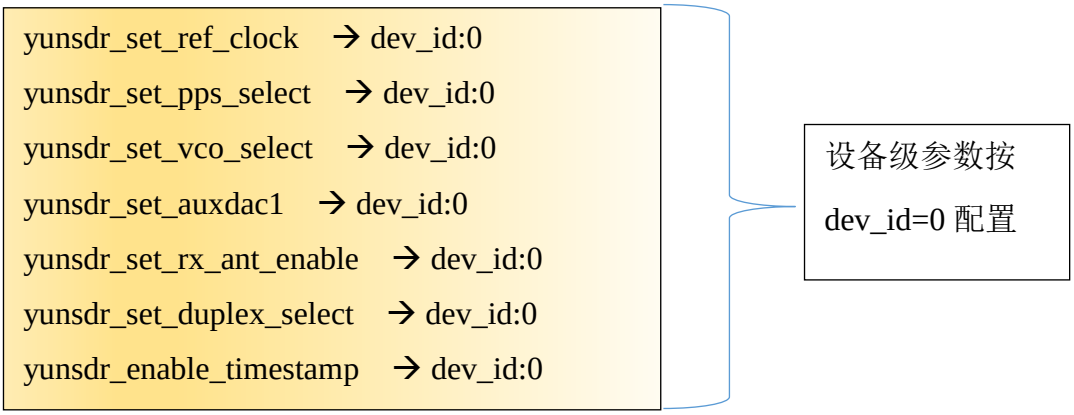


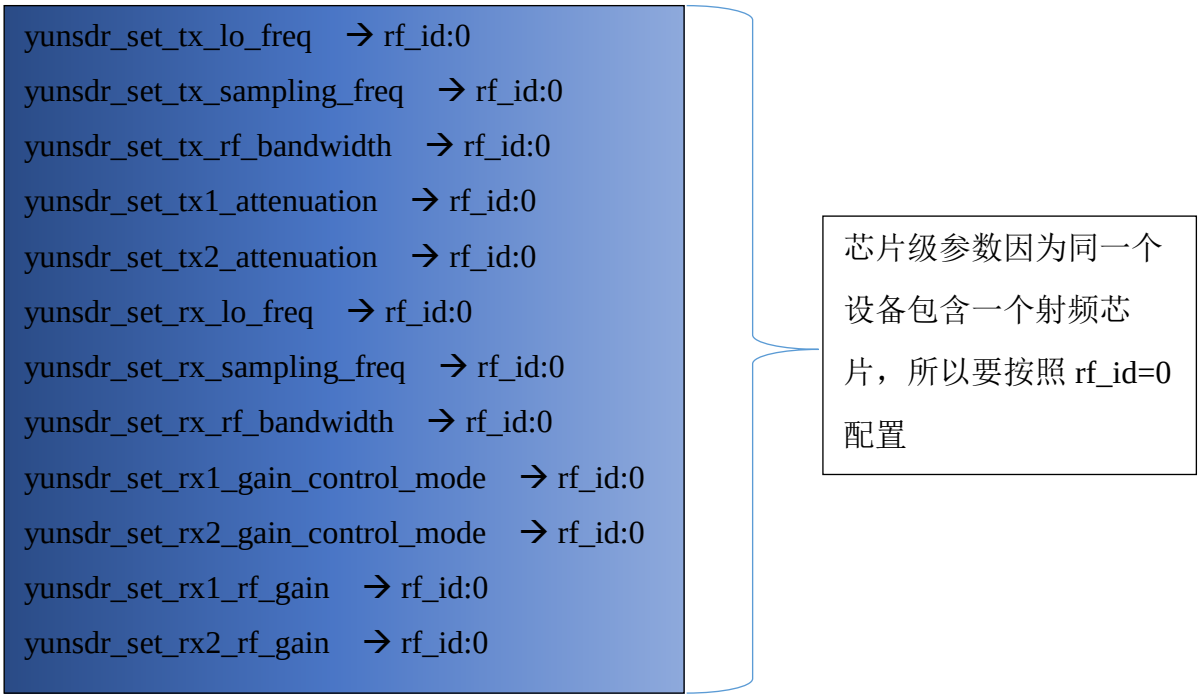
函数调用

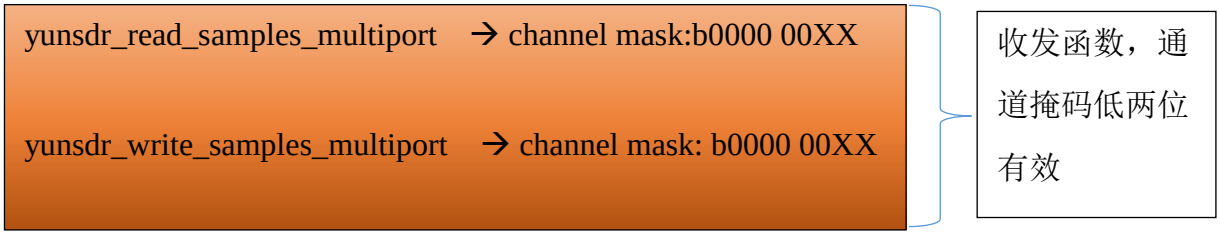
yunsdr_open_device → “pciex:0”

yunsdr_get_device_configuration →SingleSubDevSingleRF = 0x1010

一个设备，包含一个射频芯片(2 收 2 发)

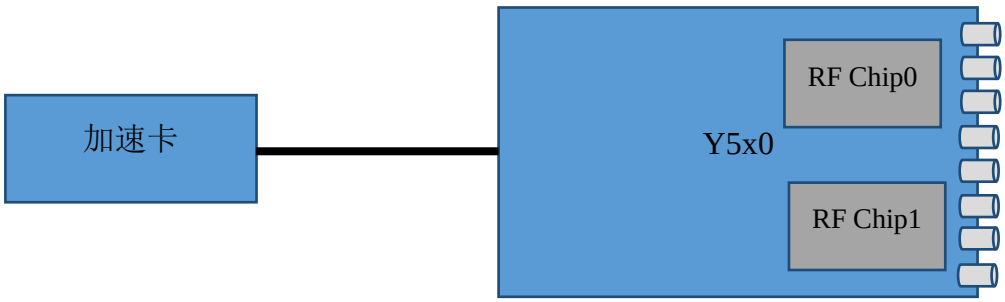



yunsdr_set_tx_lo_freq → rf_id:0
yunsdr_set_tx_sampling_freq → rf_id:0
yunsdr_set_tx_rf_bandwidth → rf_id:0
yunsdr_set_tx1_attenuation → rf_id:0
yunsdr_set_tx2_attenuation → rf_id:0
yunsdr_set_rx_lo_freq → rf_id:0
yunsdr_set_rx_sampling_freq → rf_id:0
yunsdr_set_rx_rf_bandwidth → rf_id:0
yunsdr_set_rx1_gain_control_mode → rf_id:0
yunsdr_set_rx2_gain_control_mode → rf_id:0
yunsdr_set_rx1_rf_gain → rf_id:0
yunsdr_set_rx2_rf_gain → rf_id:0
芯片级参数因为同一个设备包含一个射频芯片，所以要按照 rf_id=0 配置


yunsdr_read_samples_multiport → channel mask:b0000 00XX
yunsdr_write_samples_multiport → channel mask: b0000 00XX
收发函数，通道掩码低两位有效

3.3.2 单板双射频(4T4R)

拓扑结构



函数调用

yunsdr_open_device → “pciex:0”

yunsdr_get_device_configuration → SingleSubDevDualRF = 0x1020

一个设备两个射频芯片(4 收 4 发)

yunsdr_set_ref_clock → dev_id:0

yunsdr_set_pps_select → dev_id:0

yunsdr_set_vco_select → dev_id:0

yunsdr_set_auxdac1 → dev_id:0

yunsdr_set_trxsw_fpga_enable → dev_id:0

yunsdr_set_rx_ant_enable → dev_id:0

yunsdr_set_duplex_select → dev_id:0

yunsdr_enable_timestamp → dev_id:0

设备级参数，按 dev_id=0 配置

yunsdr_set_tx_lo_freq → rf_id:0，1

yunsdr_set_tx_sampling_freq → rf_id:0，1

yunsdr_set_tx_rf_bandwidth → rf_id:0，1

yunsdr_set_tx1_attenuation → rf_id:0，1

yunsdr_set_tx2_attenuation → rf_id:0，1

yunsdr_set_rx_lo_freq → rf_id:0，1

yunsdr_set_rx_sampling_freq → rf_id:0，1

yunsdr_set_rx_rf_bandwidth → rf_id:0，1

yunsdr_set_rx1_gain_control_mode → rf_id:0，1

yunsdr_set_rx2_gain_control_mode → rf_id:0，1

yunsdr_set_rx1_rf_gain → rf_id:0，1

yunsdr_set_rx2_rf_gain → rf_id:0，1

芯片级参数，因为同一个设备包含两个射频芯片，所以要按照 rf_id=0 和 rf_id=1 分开配置

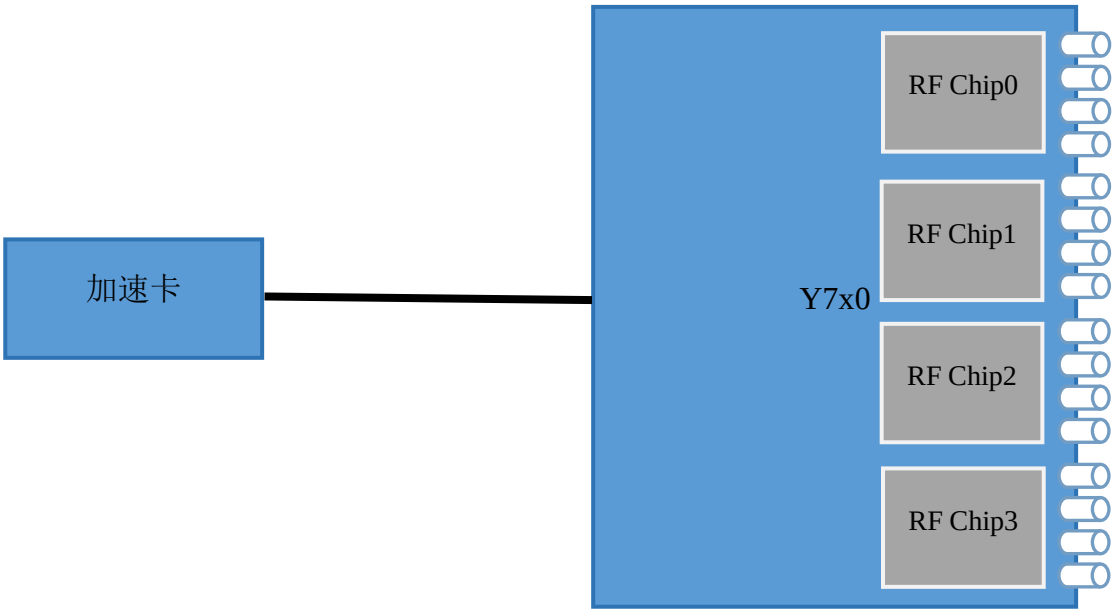
yunsdr_read_samples_multiport → channel mask: b0000 XXXX

yunsdr_write_samples_multiport → channel mask: b0000 XXXX

收发函数，通道掩码低四位有效

3.3.3 单板四射频(8T8R)

拓扑结构



函数调用

yunsdr_open_device → “pciex:0”

yunsdr_get_device_configuration → SingleSubDevQuadRF = 0x1040

一个设备四个射频芯片(8 收 8 发)

yunsdr_set_ref_clock → dev_id:0

yunsdr_set_pps_select → dev_id:0

yunsdr_set_vco_select → dev_id:0

yunsdr_set_auxdac1 → dev_id:0

yunsdr_set_rx_ant_enable → dev_id:0

yunsdr_set_duplex_select → dev_id:0

yunsdr_enable_timestamp → dev_id:0

设备级参数，按 dev_id=0 配置

yunsdr_set_tx_lo_freq → rf_id:0，1，2，3

yunsdr_set_tx_sampling_freq → rf_id:0，1，2，3

yunsdr_set_tx_rf_bandwidth → rf_id:0，1，2，3

yunsdr_set_tx1_attenuation → rf_id:0，1，2，3

yunsdr_set_tx2_attenuation → rf_id:0，1，2，3

yunsdr_set_rx_lo_freq → rf_id:0，1，2，3

yunsdr_set_rx_sampling_freq → rf_id:0，1，2，3

yunsdr_set_rx_rf_bandwidth → rf_id:0，1，2，3

yunsdr_set_rx1_gain_control_mode → rf_id:0，1，2，3

yunsdr_set_rx2_gain_control_mode → rf_id:0，1，2，3

yunsdr_set_rx1_rf_gain → rf_id:0，1，2，3

yunsdr_set_rx2_rf_gain → rf_id:0，1，2，3

芯片级参数，因为同一个设备包含两个射频芯片，所以要按照 rf_id=0、rf_id=1、rf_id=2 和 rf_id=3 分开配置

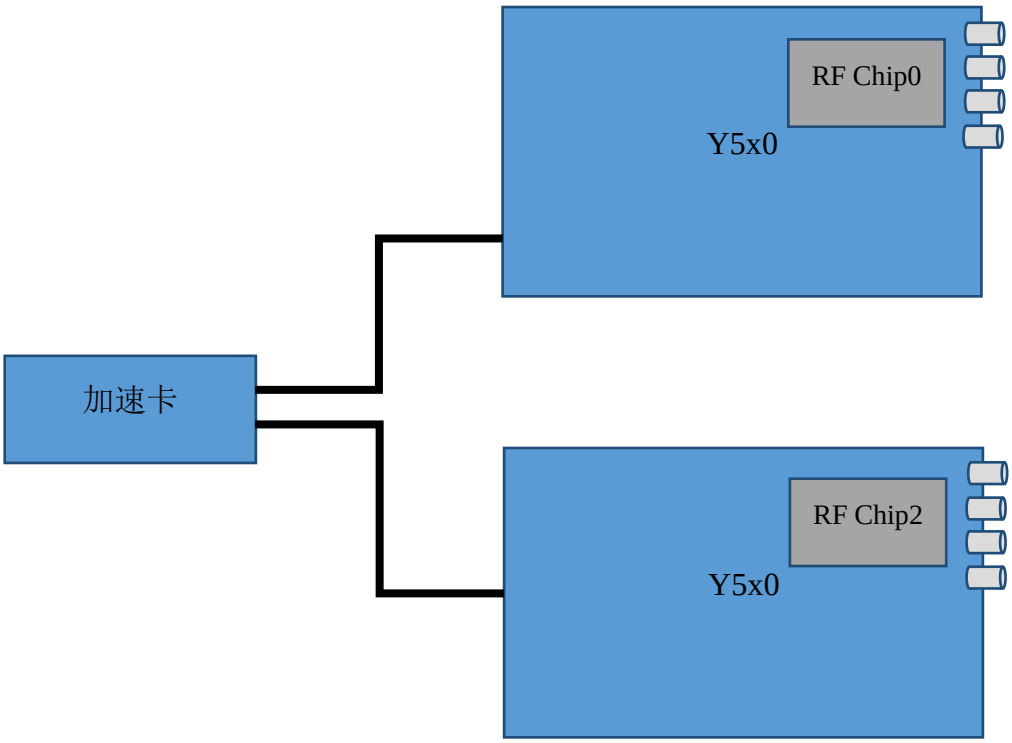
yunsdr_read_samples_multiport → channel mask: bXXXX XXXX

yunsdr_write_samples_multiport → channel mask: bXXXX XXXX

收发函数，通道掩码低 8 位有效

3.3.4 双板卡双射频(4T4R)

拓扑结构



函数调用

yunsdr_open_device → “pciex:0”

yunsdr_get_device_configuration → DualSubDevDualRF = 0x2011

两个光口接两个设备，每个设备各包含一个射频芯片，共两个射频芯片(4 收 4 发)

yunsdr_set_ref_clock → dev_id:0，2

yunsdr_set_pps_select → dev_id:0，2

yunsdr_set_vco_select → dev_id:0，2

yunsdr_set_auxdac1 → dev_id:0，2

yunsdr_set_rx_ant_enable → dev_id:0，2

yunsdr_set_duplex_select → dev_id:0，2

yunsdr_enable_timestamp → dev_id:0，2

设备级参数，因为有两个设备，所以要按照 dev_id=0 和 dev_id=2 分开配置

h

yunsdr_set_tx_lo_freq → rf_id:0，2

yunsdr_set_tx_sampling_freq → rf_id:0，2

yunsdr_set_tx_rf_bandwidth → rf_id:0，2

yunsdr_set_tx1_attenuation → rf_id:0，2

yunsdr_set_tx2_attenuation → rf_id:0，2

yunsdr_set_rx_lo_freq → rf_id:0，2

yunsdr_set_rx_sampling_freq → rf_id:0，2

yunsdr_set_rx_rf_bandwidth → rf_id:0，2

yunsdr_set_rx1_gain_control_mode → rf_id:0，2

yunsdr_set_rx2_gain_control_mode → rf_id:0，2

yunsdr_set_rx1_rf_gain → rf_id:0，2

yunsdr_set_rx2_rf_gain → rf_id:0，2

芯片级参数，因为两个设备分别包含一个射频芯片，所以要按照 rf_id=0 和 rf_id=2 分开配置

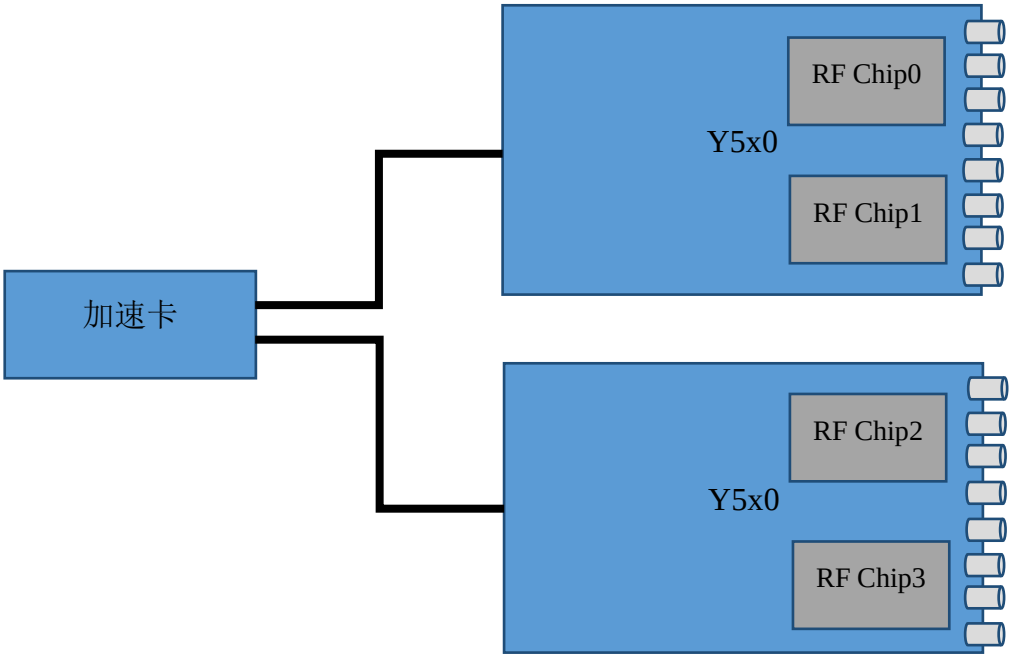
yunsdr_read_samples_multiport → channel mask:b00XX 00XX

yunsdr_write_samples_multiport → channel mask: b00XX 00XX

收发函数，通道掩码[0:1]位有效和[4:5]位有效

3.3.5 双板四射频(8T8R)

拓扑结构



函数调用

yunsdr_open_device → “pciex:0”

yunsdr_get_device_configuration → DualSubDevQuadRF = 0x2022

两个光口接两个设备，每个设备各包含两个射频芯片，共四个射频芯片(8 收 8 发)

yunsdr_set_ref_clock → dev_id:0，2

yunsdr_set_pps_select → dev_id:0，2

yunsdr_set_vco_select → dev_id:0，2

yunsdr_set_auxdac1 → dev_id:0，2

yunsdr_set_rx_ant_enable → dev_id:0，2

yunsdr_set_duplex_select → dev_id:0，2

yunsdr_enable_timestamp → dev_id:0，2

设备级参数，因为有两个设备，所以要按照 dev_id=0 和 dev_id=2 分开配置

yunsdr_set_tx_lo_freq → rf_id:0，1，2，3

yunsdr_set_tx_sampling_freq → rf_id: 0，1，2，3

yunsdr_set_tx_rf_bandwidth → rf_id: 0，1，2，3

yunsdr_set_tx1_attenuation → rf_id: 0，1，2，3

yunsdr_set_tx2_attenuation → rf_id: 0，1，2，3

yunsdr_set_rx_lo_freq → rf_id: 0，1，2，3

yunsdr_set_rx_sampling_freq → rf_id: 0，1，2，3

yunsdr_set_rx_rf_bandwidth → rf_id: 0，1，2，3

yunsdr_set_rx1_gain_control_mode → rf_id: 0，1，2，3

yunsdr_set_rx2_gain_control_mode → rf_id: 0，1，2，3

yunsdr_set_rx1_rf_gain → rf_id: 0，1，2，3

yunsdr_set_rx2_rf_gain → rf_id: 0，1，2，3

芯片级参数，因为两个设备分别包含两个个射频芯片，所以要按照 rf_id=0、rf_id=1、rf_id=2、rf_id=3 分开配置

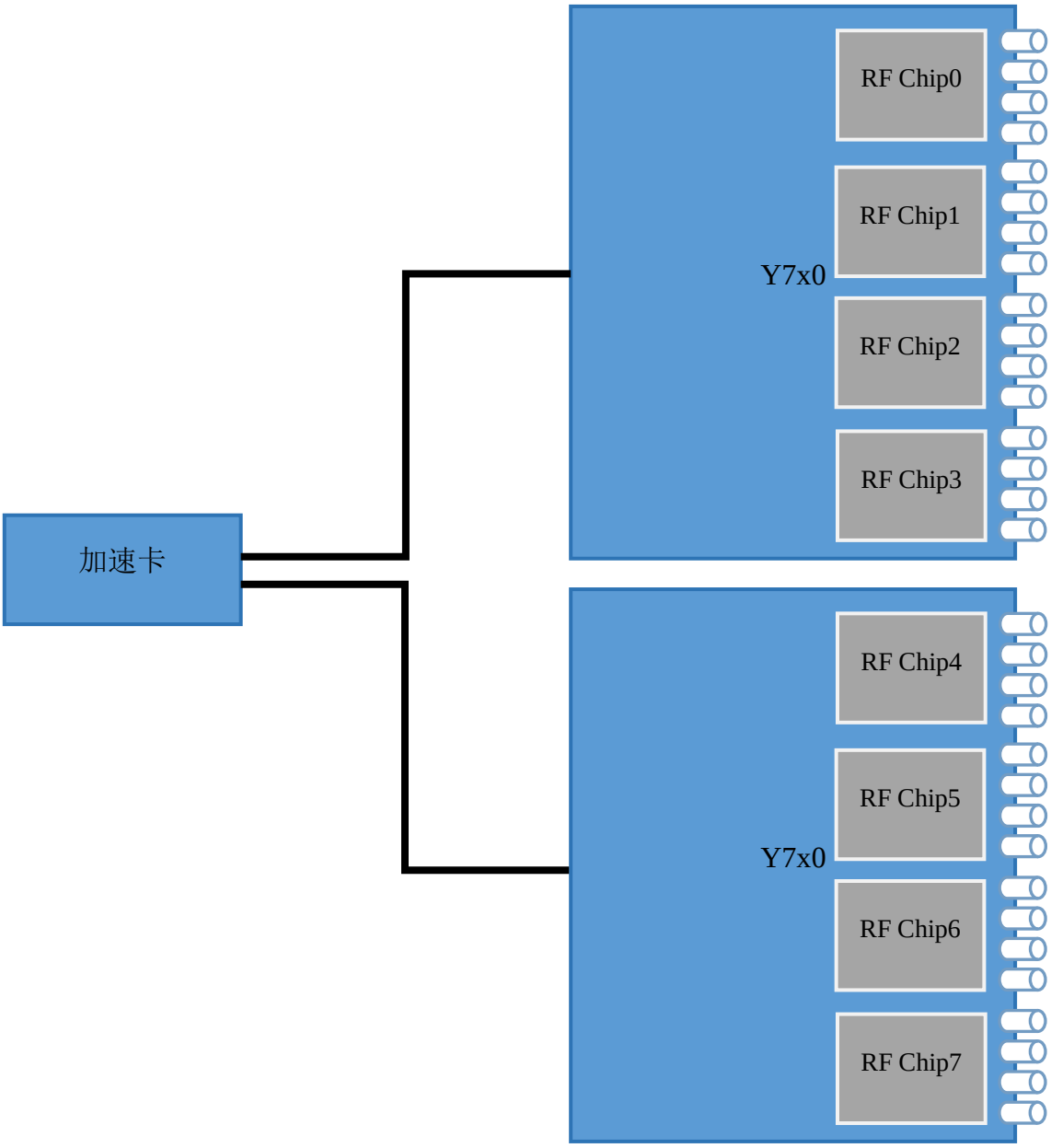
yunsdr_read_samples_multiport → channel mask:bXXXX XXXX

yunsdr_write_samples_multiport → channel mask: bXXXX XXXX

收发函数，通道掩码[0:7]位有效

3.3.6 双板八射频(16T16R)

拓扑结构



函数调用

yunsdr_open_device → “pciex:0”

yunsdr_get_device_configuration → DualSubDevOctoRF = 0x2044

两个设备八个射频芯片(16 收 16 发)

yunsdr_set_ref_clock → dev_id:0，4

yunsdr_set_pps_select → dev_id:0，4

yunsdr_set_vco_select → dev_id:0，4

yunsdr_set_auxdac1 → dev_id:0，4

yunsdr_set_rx_ant_enable → dev_id:0，4

yunsdr_set_duplex_select → dev_id:0，4

yunsdr_enable_timestamp → dev_id:0，4

设备级参数，按 dev_id=0 和 dev_id=4 配置

yunsdr_set_tx_lo_freq → rf_id:0，1，2，3，4，5，6，7

yunsdr_set_tx_sampling_freq → rf_id:0，1，2，3，4，5，6，7

yunsdr_set_tx_rf_bandwidth → rf_id:0，1，2，3，4，5，6，7

yunsdr_set_tx1_attenuation → rf_id:0，1，2，3，4，5，6，7

yunsdr_set_tx2_attenuation → rf_id:0，1，2，3，4，5，6，7

yunsdr_set_rx_lo_freq → rf_id:0，1，2，3，4，5，6，7

yunsdr_set_rx_sampling_freq → rf_id:0，1，2，3，4，5，6，7

yunsdr_set_rx_rf_bandwidth → rf_id:0，1，2，3，4，5，6，7

yunsdr_set_rx1_gain_control_mode → rf_id:0，1，2，3，4，5，6，7

yunsdr_set_rx2_gain_control_mode → rf_id:0，1，2，3，4，5，6，7

yunsdr_set_rx1_rf_gain → rf_id:0，1，2，3，4，5，6，7

yunsdr_set_rx2_rf_gain → rf_id:0，1，2，3，4，5，6，7

芯片级参数，因为同一个设备包含四个射频芯片，所以要按照 rf_id=0、rf_id=1、rf_id=2、rf_id=3、rf_id=4、rf_id=5、rf_id=6 和 rf_id=7 分开配置

yunsdr_read_samples_multiport → channel mask: bXXXX XXXX XXXX XXXX

yunsdr_write_samples_multiport → channel mask: bXXXX XXXX XXXX XXXX

收发函数，通道掩码低 16 位有效