YunSDR-Y 230(USB3.0)

Matlab 开发指南

Rev. 1.0

北京威视锐科技有限公司

威视锐®
V3 Technology,.Ltd.

修订记录

版本	修订日期	修订内容
1.0	2021 年 01 月 12 日	初始版本

关于威视锐科技

北京威视锐科技有限公司专注于软件定义（SDx）系列的研发与生产，面向无线通信、视频视觉和测试测量领域提供完整的解决方案，可应用于科研教学与产品研发。威视锐与微软研究院联合开发的 SoraSDR 软件无线电平台、YunSDR 软件无线电平台已经成为世界上知名大学和科研机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基于 x86 和 Windows 系统的宽带软件无线电平台，目前已经有超过 20 多个国家的 300 多个用户在 Sora 平台上开发无线局域网、移动通信、大规模 MIMO 等相关领域的应用。作为全球最大的可编程器件公司 Xilinx 的全球认证合作伙伴、授权培训合作伙伴和大学计划合作伙伴，威视锐科技提供基于 Xilinx FPGA/SoC 全方位解决方案。威视锐同时也是全球领先的高性能模拟器件厂家 ANALOG DEVICES 公司的第三方和大学计划合作伙伴，提供基于 ADI 的高性能射频收发器，转换器和传感器开发套件。特别是无线通信、物联网、视觉图像处理和数字信号处理的创新型实验室建设，威视锐可以提供完整的解决方案和技术支持服务。

多年以来，威视锐坚持“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、学者和研发工程师创新的理念变成现实和产品。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

一、文档概述

本文档适用于使用 USB3.0 的 YunSDR 硬件平台。使用 Matlab 软件进 IQ 数据收发。

二、软件环境配置

在使用相应设备之前，需要对 PC 的环境进行配置。为保证顺利使用，请按照下列要求进行配置。

- **PC 操作系统：**

  Windows7 x64

  Windows10 x64

  Windows Server x64

  Linux Ubuntu x64

- **Matlab 软件：**2016a 及以上；

  C 编译库：TDM-GCC，或其他 matlab 所支持的 C 编译器

  详细的编译库支持请参见 https://ww2.mathworks.cn/support/compilers.html

  环境配置完毕之后，打开 Matlab 的 Command Window，输入 mex -setup 命令，显示如下：

```
>> mex -setup
MEX configured to use 'MinGW64 Compiler (C)' for C language compilation.
Warning: The MATLAB C and Fortran API has changed to support MATLAB
     variables with more than 2^32-1 elements. In the near future
     you will be required to update your code to utilize the
     new API. You can find more information about this at:
     http://www.mathworks.com/help/matlab/matlab_external/upgrading-mex-files-to-use-64-b

To choose a different C compiler, select one from the following:
MinGW64 Compiler (C)  mex -setup:C:\Users\hans\AppData\Roaming\MathWorks\MATLAB\R2016a\me
Microsoft Visual C++ 2013 Professional (C)  mex -setup:'C:\Program Files\MATLAB\R2016a\bi

To choose a different language, select one from the following:
 mex -setup C++
 mex -setup FORTRAN
```

图 1　Matlab 下的 C++编译器安装

表明 Matlab 可以找到 C++编译器(MinGW64)，配置成功（若提示找不到编译器，请重启电脑或检查相关软件的安装路径），然后单击蓝色字体：mex -setup C++，进行编译器切换。详细安装流程参考 2.1 TDM-GCC 编译器安装与设置。

（一）TDM-GCC 编译器安装与设置

不同的 Matlab 推荐使用的 TDM-GCC 版本不同，用户可以根据安装的 Matlab 版本参考上面链接选取合适的 GCC 版本。例如 Matlab 2016a 配套的 GCC 编译器版本是 tdm64-gcc-4.9.2.exe,（实际测试 4.9.2 可以支持到 2019 版本）安装好后需要在环境变量中做如下设置：

找到“我的电脑”图标，右击点击“属性”，找到“高级系统设置”：

图 2 环境变量 1

点击“高级系统设置”，出现如下对话框，找到“环境变量(N)”选项，点击新建：

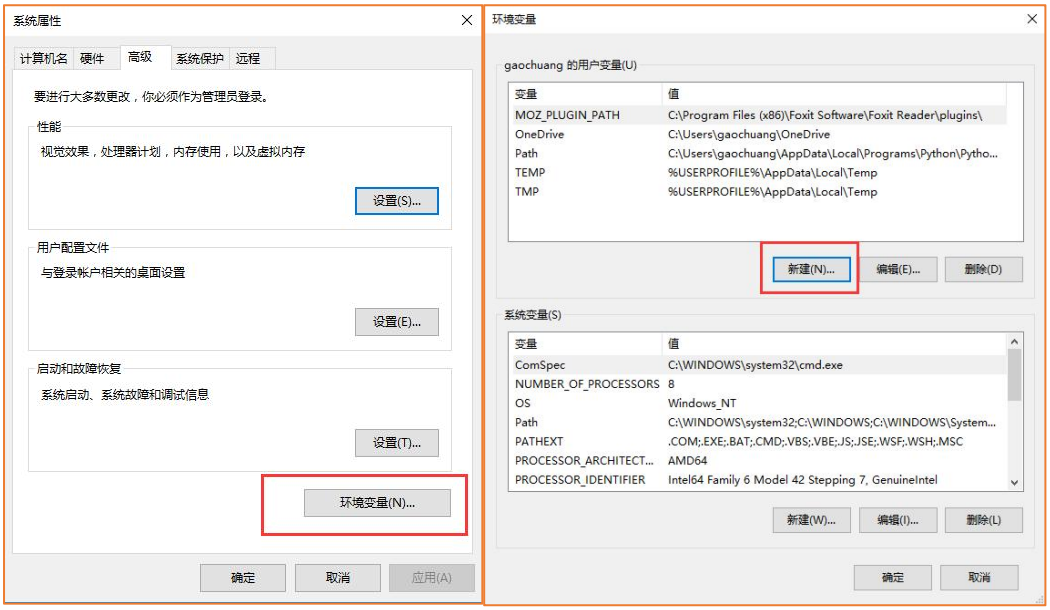

图 3 新建环境变量 1

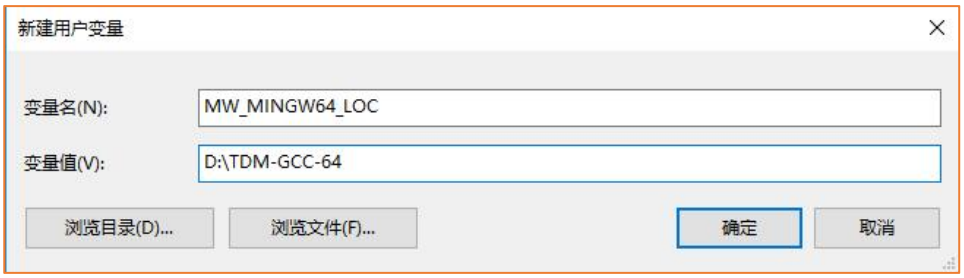

图 4 新建环境变量 2

注意：变量值填写 TDM-GCC 编译器的安装路径。

打开 MATLAB2016，输入以下命令：mex –setup，出现如下信息说明环境配置成功。

```
>> mex -setup
MEX configured to use 'MinGW64 Compiler (C)' for C language compilation.
Warning: The MATLAB C and Fortran API has changed to support MATLAB
     variables with more than 2^32-1 elements. In the near future
     you will be required to update your code to utilize the
     new API. You can find more information about this at:
     http://www.mathworks.com/help/matlab/matlab_external/upgrading-mex-files-to-use-64-b

To choose a different C compiler, select one from the following:
MinGW64 Compiler (C)  mex -setup:C:\Users\hans\AppData\Roaming\MathWorks\MATLAB\R2016a\me
Microsoft Visual C++ 2013 Professional (C)  mex -setup:'C:\Program Files\MATLAB\R2016a\bi

To choose a different language, select one from the following:
 mex -setup C++
 mex -setup FORTRAN
```

图 5　选择 mex

Matlab2020 及以后版本，除了更新 mingw 版本及以上步骤之外，还需要在 matlab 命令行窗口输入命令指定 mingw 位置，例如 setenv('MW_MINGW64_LOC','C:\mingw64')

（二）驱动环境准备

按照“01 setup”安装好 libusb 驱动后，第一次运行 matlab 程序，自动加载 FX3 固件。USB 设备更新 ID，之后需要再次安装 libusb 驱动。WINDOWS 系统可以用 zadig 软件安装。

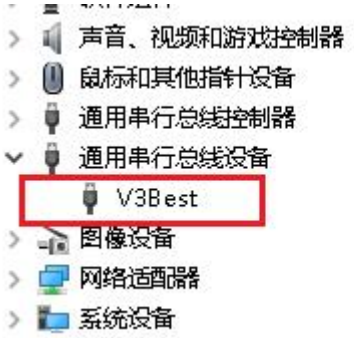

图 6 加载 FX3 后安装 libusb 驱动

三、Matlab 开发流程及配套文件说明

本文档介绍的 Matlab 例程，通过 Matlab 的 API 控制 YunSDR 的射频工作参数、控制 YunSDR 发送和接收 IQ 数据。PC 通过 USB3.0 与 YunSDR 相连，YunSDR 设备接收到 PC 发送到的一帧数据后，将数据缓存在 DDR 中，并可配置 YunSDR 的发送工作模式进入 txcyclic 循环模式（类似信号源的功能），TX 端口会反复发送这帧 IQ 数据；接收端收到数据后，YunSDR

将接收的数据搬移至 DDR，然后再通过通信总线发送至 PC，至此组成了一个无线模拟仿真系统。

由于 PC 和设备的接口通过通信总线实现，所以可在 PC 端通过 Matlab 等工具将数据按照用户自身需求进行调制，最终将调制后的数据通过通信总线下发至 YunSDR 设备；而接收端同样可以用 Matlab 将所接收的数据按照需求进行解调，基本流程参考下图。

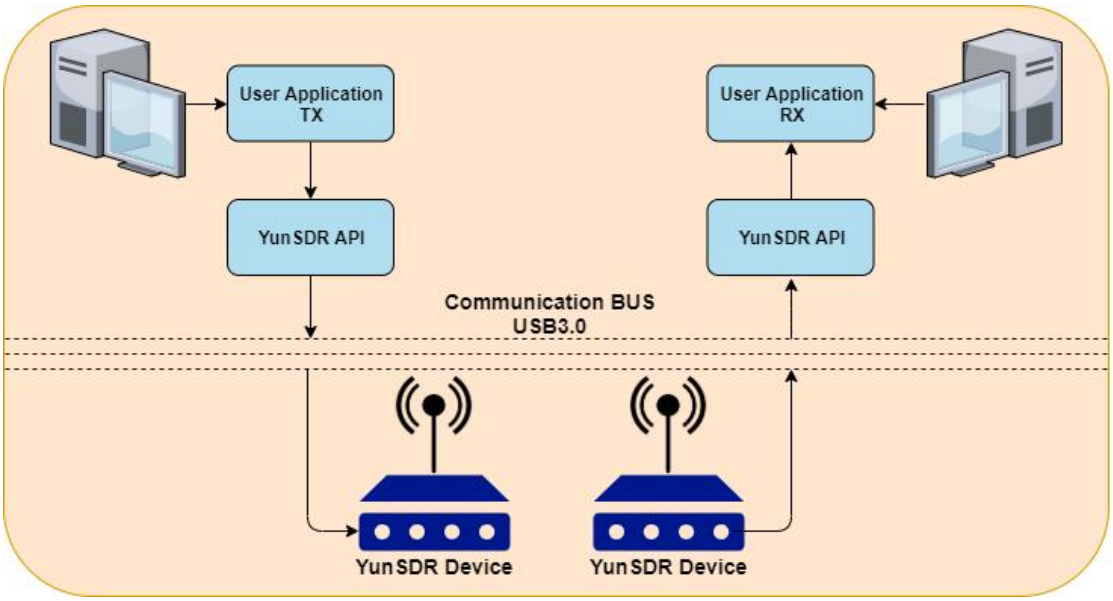

图 7　Matlab 软件功能流程

- 支持通过 Matlab 配置射频参数
- 发送端支持发送 Matlab 产生的 IQ 数据
- 通过 Matlab 可获取射频前端采集到的 IQ 数据
- 用户可定制自己的收发仿真系统

本文档所介绍的产品配套的 Matlab 程序文件包含用户交互文件及底层驱动文件，如下表所示：

表 1　文件列表

文件	类型	说明
yunsdr_tx.m	顶层文件	将 IQ 数据送给设备循环发送（txcyclic 模式）
yunsdr_rx.m	顶层文件	接收指定长度和通道数量的 IQ 数据
yunsdr.m	顶层文件	发送接收回环测试程序
gen_rxbuf.m	底层函数	RX 接收数据驱动
gen_txbuf.m	底层函数	TX 发送数据驱动

tone_plot.m	底层函数	画图
rf_init.m	常用函数	一些常用的射频配置函数
yunsdr_api_ss.h	函数列表	
libusb-1.0.dll	DLL 驱动文件	
libyunsdr_ss.dll	DLL 驱动文件	

四、参考例程说明

本文档提供基于 Matlab 环境的单音信号收发测试，在此基础上用户可以根据单音信号的数据格式，二次开发行程自定义的仿真系统。

（一）发送模式

yunsdr_tx.m，用于验证设备发送。

1. 测试类型选择

通过以下命令设置测试模式类型，分别为 tone 单音、LTE FDD、wifi，source 变量定义测试类型，即要发送的 IQ 数据类型。

```
source='tone';% tone lte_new ieee802_11a ieee802_11n
```

Tone 信号单音周期 32 个采样点，单帧数据长度 3200 个样点。最后将单音信号复制 2 个通道数量列，txch 表示发送数据的通道号掩码，1 通道，2 通道或者两个通道同时使能

LTE 信号，20MHz 带宽，发送数据长度 10ms。采样率 30.72MHz。

ieee802_11a，wifi 信号，四倍插值，按照设定采样率 30.72MHz，则信号带宽 15.36MHz，支持码速率 6 9 12 18 24 36 48 54Mbps 模式

ieee802_11n，wifi 信号，四倍插值，按照设定采样率 30.72MHz，则信号带宽 15.36MHz，支持 mimo，mcs0~7 单天线，8~15 两天线。

2. 加载动态链接库

通过以下命令载入动态库文件。

```
if not(libisloaded('libyunsdr_ss'))
    [notfound,warnings] =
loadlibrary('libyunsdr_ss','yunsdr_api_ss.h');
end
dptr = libpointer('yunsdr_device_descriptor');
devstring = libpointer('cstring');
devstring.Value = 'usb3:0,nsamples_recv_frame:7680';
dptr = calllib('libyunsdr_ss', 'yunsdr_open_device', devstring);
if isNull(dptr)
    disp 'open yunsdr failed!';
    return;
end
```

3. 设置射频参数

可以根据需要设定射频的频点采样率等参数，详见 rf_init.m，可以选择需要配置的参数粘贴到 yunsdr_tx.m 的%% add rf config 下方

```
%% rf config
%% tx
ret=calllib('libyunsdr_ss','yunsdr_set_tx_lo_freq',dptr,0,uint64(1500e6));
ret=calllib('libyunsdr_ss','yunsdr_set_tx1_attenuation',dptr,0,uint32(30e3));
%0~89e3
ret=calllib('libyunsdr_ss','yunsdr_set_tx2_attenuation',dptr,0,uint32(30e3));
ret=calllib('libyunsdr_ss','yunsdr_set_tx_rf_bandwidth',dptr,0,56e6);%
200e3~56e6
%% rx
ret=calllib('libyunsdr_ss','yunsdr_set_rx_lo_freq',dptr,0,uint64(1500e6));
ret=calllib('libyunsdr_ss','yunsdr_set_rx1_rf_gain',dptr,0,uint32(5));%1~71
ret=calllib('libyunsdr_ss','yunsdr_set_rx2_rf_gain',dptr,0,uint32(5));
ret=calllib('libyunsdr_ss','yunsdr_set_rx_rf_bandwidth',dptr,0,56e6);
%% ref clock
ret=calllib('libyunsdr_ss','yunsdr_set_tx_sampling_freq',dptr,0,uint32(40e6))
;
ret=calllib('libyunsdr_ss','yunsdr_set_auxdac1',dptr,0,1600);%0~3300mv
%% rx buffer
ret=calllib('libyunsdr_ss','yunsdr_set_hwbuf_depth',dptr,0,uint32(120*1024*10
24));
```

AD936x 发送和接收频点可以配置。两个通道共用一个频点。支持设置接收增益，1~71dB，1 最小，71dB 增益最大。支持设置发送衰减，0~89dB，0 是不衰减，89dB 最大衰减，单位

mdB，动态范围 89dB。AUXDAC 用于校准设备中的温补压控晶振的频率值，按照设备标签的 AUXDAC 进行设置，用户也可以微调。

Y230 的接收缓存容量可以支持软件配置，最大 120MB，用户可以配置。

4. 获取采样率复位时间戳

AD936x 支持的采样率范围是 521KHz~61.44MHz。时间戳是设备的定时机制，采用 64 位计数器以采样率为频率进行计数，接收和发送都是定位在这个时间戳的基础上。只有时间戳启动后设备才开始工作。

```
%% read tx_lo_freq
ret=calllib('libyunsdr_ss','yunsdr_get_tx_lo_freq',dptr,0,value64);
tx_lo_freq=double(value64.Value);
%% get samplerate
ret=calllib('libyunsdr_ss','yunsdr_get_tx_sampling_freq',dptr,0,value32);
samplerate=double(value32.Value);
%% set timpstamp start
ret=calllib('libyunsdr_ss','yunsdr_tx_cyclic_enable',dptr,0,0);% reset
txcyclic
pause(0.5);
ret=calllib('libyunsdr_ss','yunsdr_enable_timestamp',dptr,0,0);
ret=calllib('libyunsdr_ss','yunsdr_enable_timestamp',dptr,0,1);
```

5. 生成发送缓冲区

gen_txbuf.m 函数中对 txdata 数据进行量化，传输时将数据量化到 16bit 方便缓存与处理。量化后将 n 个通道的矩阵，按照 1 2.....的通道顺序排成一列，放入到生成的 int16ptr 缓冲区中。

```
function [tx_buf,length_tx,channel]=gen_txbuf(txdata)
length_tx=size(txdata,1);
channel=2^size(txdata,2)-1;
for i=1:size(txdata,2)
  c1=max(max([abs(real(txdata(:,i))),abs(imag(txdata(:,i)))]));
  if(c1>0)
    index=2000/c1;
  else
    index=0;
  end
  txdata1(:,i)=round(txdata(:,i).*index)*16;
end
txdata_s=txdata1(:);
```

```
txdatai=real(txdata_s);
txdataq=imag(txdata_s);
txdatam=zeros(length(txdatai)*2,1);
txdatam(1:2:end)=txdatai;
txdatam(2:2:end)=txdataq;
txdatamu=txdatam+(txdatam<0)*65536;
tx_buf = libpointer('voidPtrPtr');
tx_buf.Value = libpointer('int16Ptr', txdatamu);
end
```

6. 开始发送

将待发送的 IQ 数据缓存到自身的 DDR 缓存中，循环发送，模拟作信号源的功能。基于时戳定时的原理是：

- 首先获取设备的时间戳
- 将所有通道开始发送 IQ 数据的时刻 ts
- 指定发送时刻 ts+1s(采样率)
- 调用发送函数将 IQ 数据送给设备
- 设备收到 IQ 数据由于还没有到发送时刻，所以将 IQ 数据缓存
- 等待到达发送时刻所有通道一起启动 TX
- 判断上位机配置了 tx'cyclic 指令，设备循环启动 TX 发送，循环播出 TX 序列，可以在频谱以上观测发送的信号。

```
%% send data in txcyclic mode
ret=calllib('libyunsdr_ss','yunsdr_tx_cyclic_enable',dptr,0,0);% reset txcyclic
ret=calllib('libyunsdr_ss','yunsdr_tx_cyclic_enable',dptr,0,1);
ret=calllib('libyunsdr_ss','yunsdr_read_timestamp',dptr,0,value64);
ts3=value64.Value+samplerate;
nwrite = calllib('libyunsdr_ss', 'yunsdr_write_samples_multiport_Matlab',  ...
  dptr, tx_buf, length_tx, channel, ts3, 0);
ret=calllib('libyunsdr_ss','yunsdr_tx_cyclic_enable',dptr,0,3);% start txcyclic
```

7. 结束循环发送

如果需要结束循环发送，需要复位发送函数即可

```
ret=calllib('libyunsdr_ss','yunsdr_tx_cyclic_enable',dptr,0,0);% reset txcyclic
```

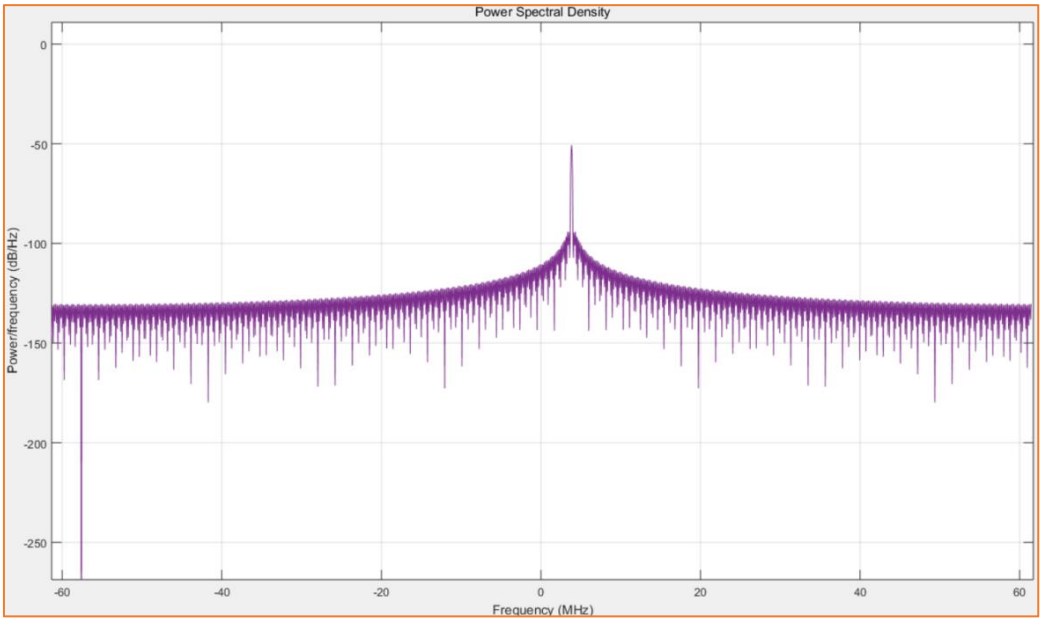

图 8　单音信号发送

（二）接收模式

yunsdr_rx.m，用于验证设备接收。接收与发送的流程基本一致，将采集到数据送给matlab。

首先使能加载动态库；

其次可以根据需要设定射频参数，和发送节一样。

之后获取采样率复位时间戳，接收端和发送端共用采样率。接收端和发送端以同一个时间戳为基准。

1.　设定接收长度和接收通道

可以选择使能接收通道的编号，以 0~3 通道掩码表示 2 个通道，最低 bit 代表通道 0，最高 bit 代表通道 1。所有通道均使能是 3。

接收长度可以设置 7680 的整数倍。

```
rxch=hex2dec('3');% mask of 4 channel of each bit,f is 4 chan
rxlength=614400;
```

2.　启动接收函数

调用接收函数后，会自动在当前目录存储符合通道数据量的数据文件，如果使能 2 个通道会存储 2 个 IQ 数据文件。接收函数也会返回 n 列的 IQ 复数序列。

```
[rxdata,rx_time]=gen_rxbuf(dptr,rxlength,rxch,0);
```

3. 数据处理

接收端运行效果如下图所示：

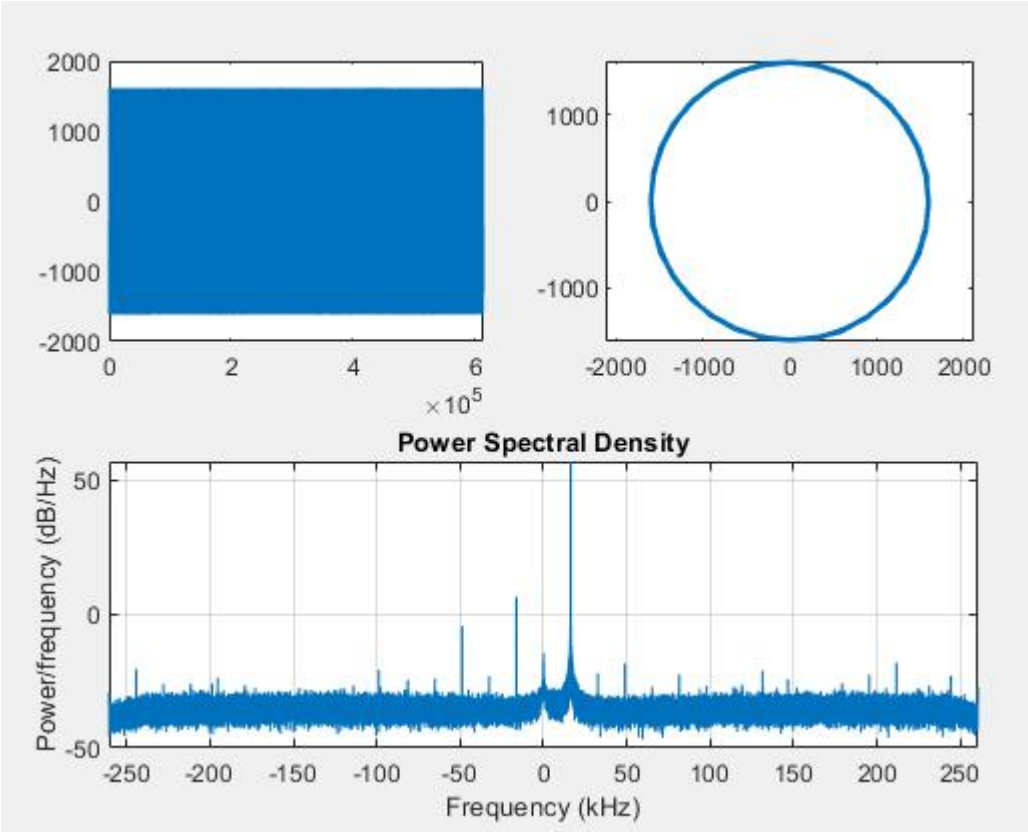

图 9 单音信号自收发运行效果

（三）收发回环

收发回环运行 yunsdr.m 文件，它集成了接收和发送的功能。可以在单独设备上验收接收和发送回环。

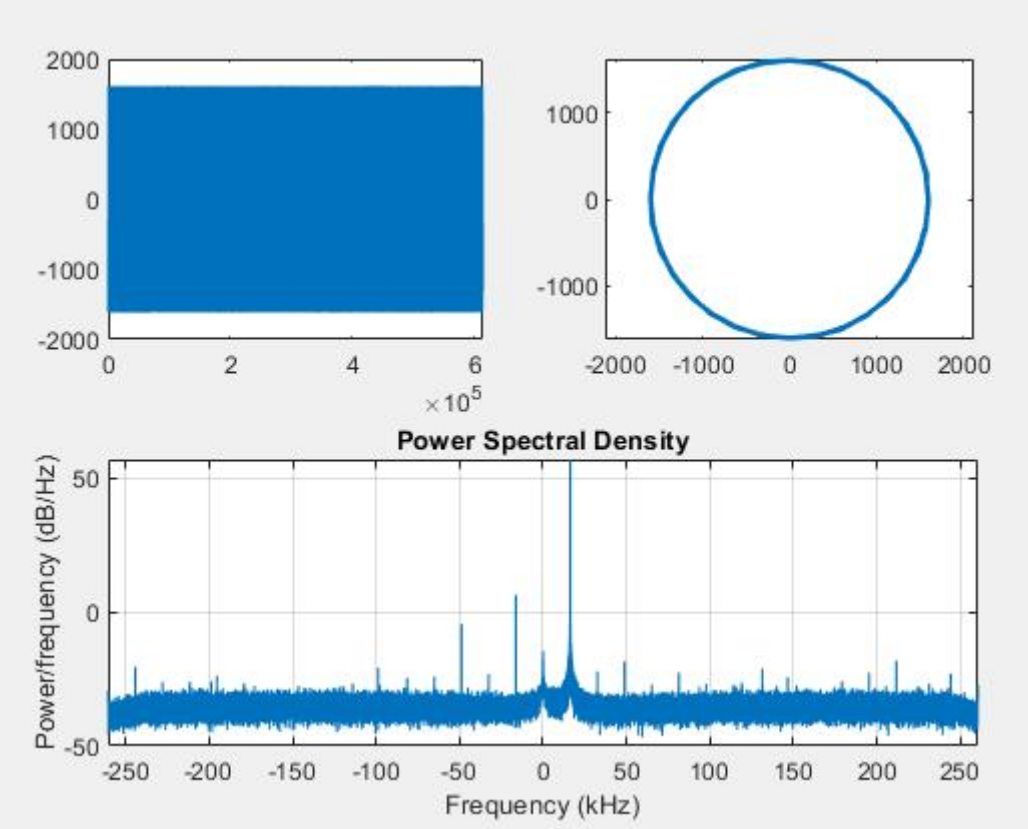

图 10 单音回环

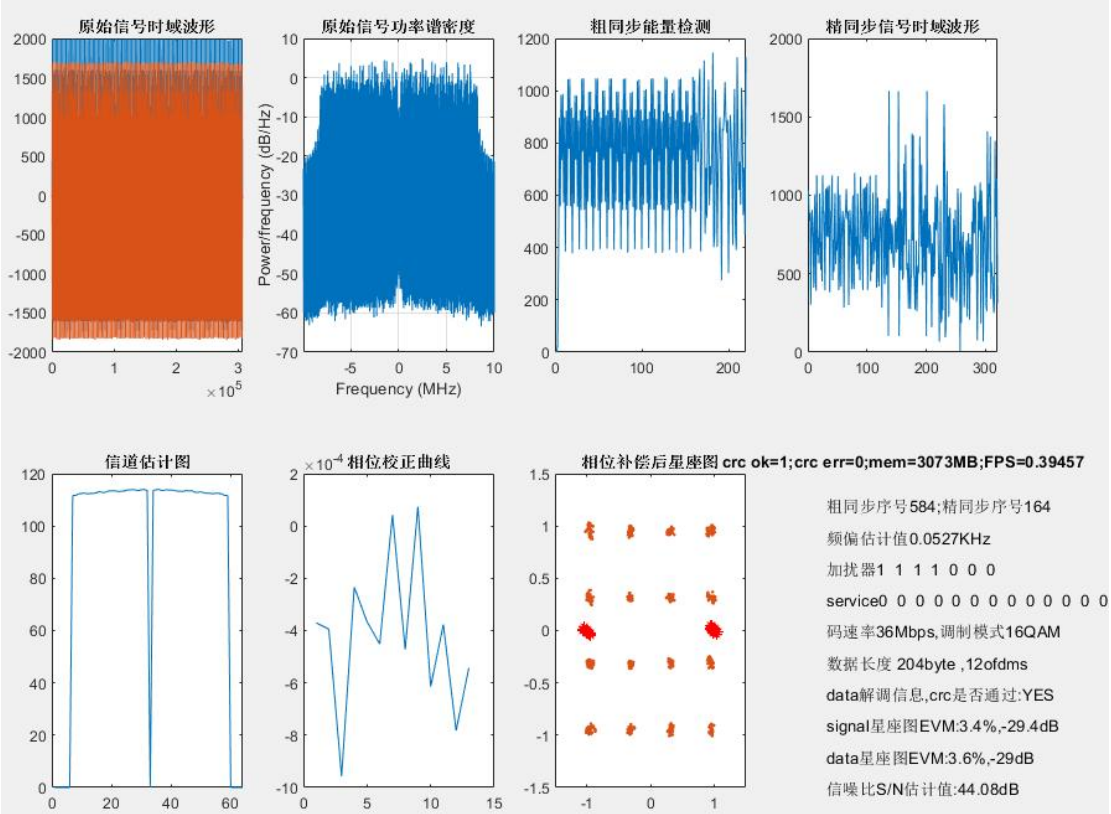

图 11 ieee802.11a 回环

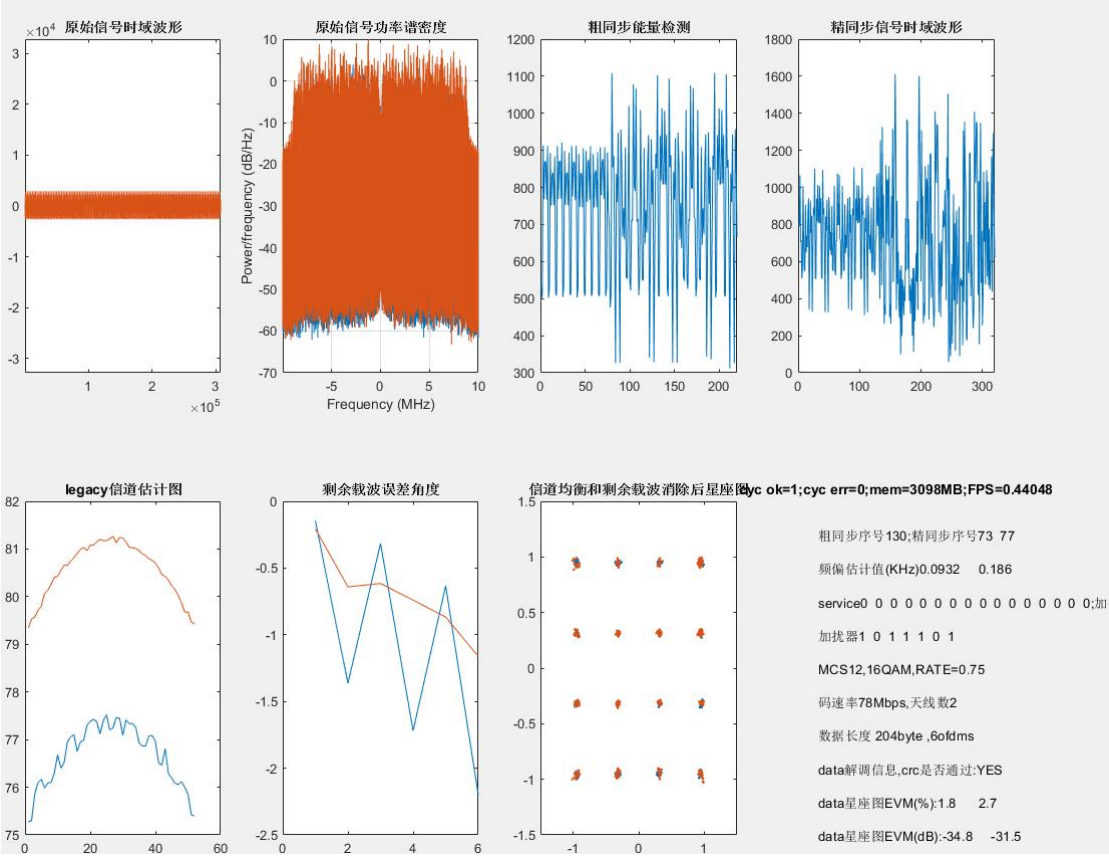

图 12 ieee802.11n 回环

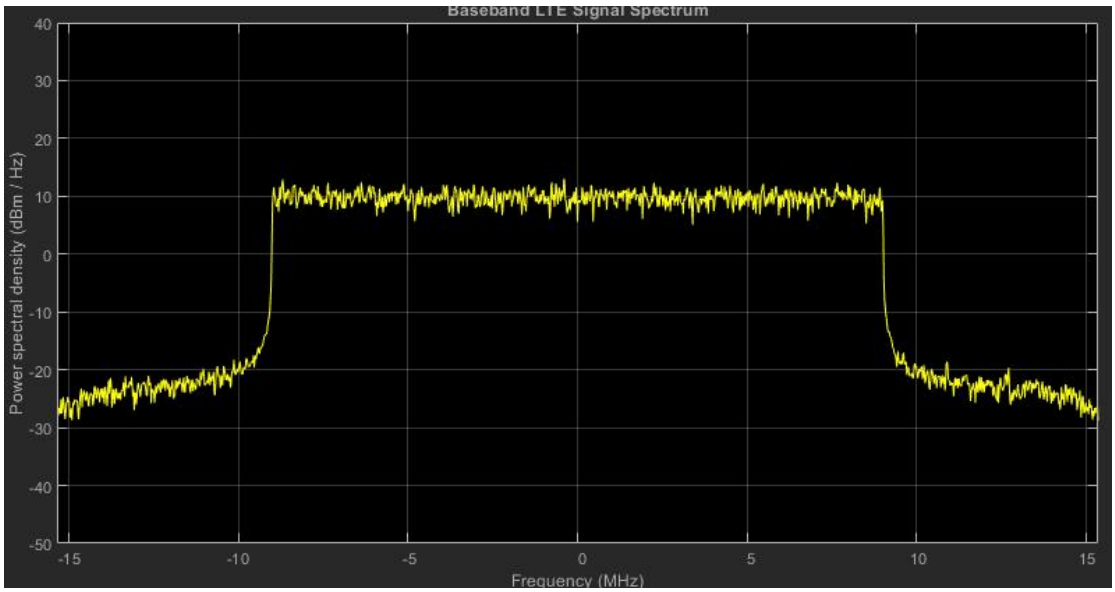

图 13 LTE 接收频谱

```
SDR hardware sampling rate configured to capture 100 LTE RBs.
Corrected a frequency offset of -0.335598 Hz.
Detected a cell identity of 17.
```

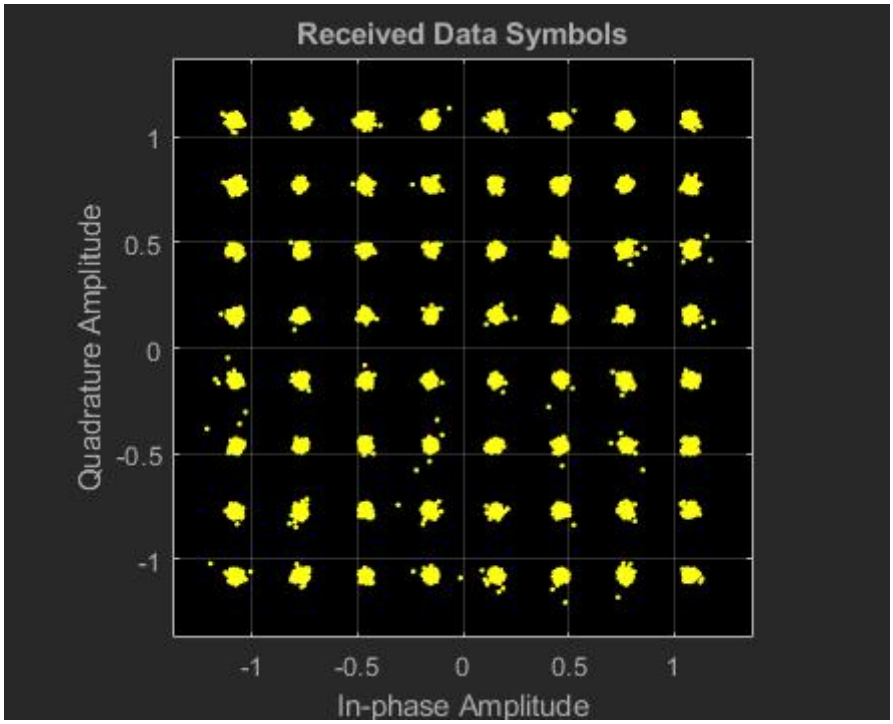

图 14 LTE 接收解调星座图

```
Low edge EVM, subframe 0: 1.921%
High edge EVM, subframe 0: 1.934%
Low edge EVM, subframe 1: 1.977%
High edge EVM, subframe 1: 1.960%
Low edge EVM, subframe 2: 1.968%
High edge EVM, subframe 2: 1.960%
Low edge EVM, subframe 3: 1.977%
High edge EVM, subframe 3: 1.984%
Low edge EVM, subframe 4: 1.955%
High edge EVM, subframe 4: 1.960%
Low edge EVM, subframe 6: 1.982%
High edge EVM, subframe 6: 1.978%
Low edge EVM, subframe 7: 1.998%
High edge EVM, subframe 7: 2.005%
Low edge EVM, subframe 8: 1.988%
High edge EVM, subframe 8: 1.987%
Low edge EVM, subframe 9: 2.074%
High edge EVM, subframe 9: 2.049%
Averaged low edge EVM, frame 0: 1.978%
Averaged high edge EVM, frame 0: 1.976%
Averaged EVM frame 0: 1.978%
Averaged overall EVM: 1.978%
```

附录：动态库函数列表

```
DLLEXPORT YUNSDR_DESCRIPTOR *yunsdr_open_device(const char *url);
```

打开设备

```
DLLEXPORT int32_t yunsdr_close_device(YUNSDR_DESCRIPTOR *yunsdr);
```

关闭设备

```
DLLEXPORT int32_t yunsdr_get_sampling_freq_range(YUNSDR_DESCRIPTOR
*yunsdr, uint8_t rf_id,uint32_t *sampling_freq_hz_max, uint32_t
*sampling_freq_hz_min);
```

获取支持的采样率范围，Y230 默认 61.44e6Hz，可配置范围 521e3Hz~61.44e6Hz。

```
rfid=0
```

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

获取支持的接收增益设置范围，Y230 接收增益的调整范围 1~71dB。rfid=0

DLLEXPORT int32_t yunsdr_get_rx_freq_range(YUNSDR_DESCRIPTOR *yunsdr, uint8_t rf_id,uint64_t *lo_freq_hz_max, uint64_t *lo_freq_hz_min);

获取支持的接收频点设置范围，70e6Hz ~ 6000e6Hz。rfid=0

DLLEXPORT int32_t yunsdr_get_tx_freq_range(YUNSDR_DESCRIPTOR *yunsdr, uint8_t rf_id,uint64_t *lo_freq_hz_max, uint64_t *lo_freq_hz_min);

获取支持的发送频点设置范围，70e6Hz ~ 6000e6Hz。rfid=0

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

获取当前发送频点，rfid0 通道的频点

DLLEXPORT int32_t yunsdr_get_tx_sampling_freq (YUNSDR_DESCRIPTOR *yunsdr, uint8_t rf_id,uint32_t *sampling_freq_hz);

获取当前发送采样率，rfid=0。

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

获取发送带宽 200e3~56e6Hz

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

获取当前接收频点，rfid=0

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

获取接收带宽

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

获取接收增益,rfid=0，rx1 表示第一路

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

获取接收增益,rfid=0，rx2 表第二路

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

配置接收频点，rfid=0。范围 70e6Hz ~ 6000e6Hz

DLLEXPORT int32_t yunsdr_set_rx1_rf_gain (YUNSDR_DESCRIPTOR *yunsdr, uint8_t rf_id,int32_t gain_db);

配置接收增益,设置范围 1~71。Rfid=0，rx1 表示每个射频通道的第一路

DLLEXPORT int32_t yunsdr_set_rx2_rf_gain (YUNSDR_DESCRIPTOR *yunsdr, uint8_t rf_id,int32_t gain_db);

配置接收增益,设置范围 1~71。Rfid=0，rx2 表示每个射频通道的第二路

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

配置发送频点，rfid=0。范围 70e6Hz ~ 6000e6Hz

DLLEXPORT int32_t yunsdr_set_tx_sampling_freq (YUNSDR_DESCRIPTOR *yunsdr, uint8_t rf_id,uint32_t sampling_freq_hz);

配置采样率，rfid=0 同时作用收发采样率，可配置范围 521e3~61.44e6Hz。rfid=0

DLLEXPORT int32_t yunsdr_set_auxdac1 (YUNSDR_DESCRIPTOR *yunsdr, uint8_t rf_id, uint32_t vol_mV);

配置晶振频偏调节电压，rfid=0。根据设备标签设定范围 0~3300

DLLEXPORT int32_t yunsdr_tx_cyclic_enable(YUNSDR_DESCRIPTOR *yunsdr, uint8_t rf_id,uint8_t enable);

配置循环发送模式，rf_id=0

Enable=0，关闭

Enable=1，准备

Enable=3，开始输出

DLLEXPORT int32_t yunsdr_set_hwbuf_depth(YUNSDR_DESCRIPTOR *yunsdr, uint8_t rf_id, uint32_t depth);

配置接收内存深度，rfid=0 单位 byte，默认是 120*1024*1024，最小 30000

DLLEXPORT int32_t yunsdr_get_hwbuf_depth(YUNSDR_DESCRIPTOR *yunsdr, uint8_t rf_id, uint32_t *depth);

获取接收内存深度 rfid=0

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

获取版本号 rfid=0

DLLEXPORT int32_t yunsdr_get_model_version(YUNSDR_DESCRIPTOR *yunsdr, uint32_t *version);

获取硬件设备号，rfid=0

DLLEXPORT int32_t yunsdr_set_pps_select (YUNSDR_DESCRIPTOR *yunsdr, uint8_t rf_id, PPSModeEnum pps);

配置时戳模式，rfid=0 默认是 0

0：内部产生

DLLEXPORT int32_t yunsdr_enable_timestamp(YUNSDR_DESCRIPTOR *yunsdr, uint8_t rf_id,uint8_t enable);

使能时间戳

DLLEXPORT int32_t yunsdr_read_timestamp(YUNSDR_DESCRIPTOR *yunsdr, uint8_t rf_id, uint64_t *timestamp);

获取时间戳

DLLEXPORT int32_t yunsdr_read_samples_multiport_Matlab (YUNSDR_DESCRIPTOR *yunsdr,void *buffer, uint32_t count, uint8_t channel_mask, uint64_t *timestamp);

RX 端收数据，具体用法请见 gen_rxbuf 函数

DLLEXPORT int32_t yunsdr_write_samples_multiport_Matlab (YUNSDR_DESCRIPTOR *yunsdr,const void *buffer, uint32_t count, uint8_t channel_mask, uint64_t timestamp, uint32_t flags);

TX 端发数据，具体用法请见 gen_txbuf 函数