SPI to USB Data Streaming on Apollo3Blue (AMU2S)

This page introduces the SPI to USB data streaming interface from Apollo3Blue to PC.
This is a high speed interface using IOM module on Apollo3Blue device to stream data out to PC for debug purpose.
The SPI slave to USB converter is FT4222H from FTDI.
One can use FT4222H-EV-D evaluation board from FTDI for this purpose.

Transfer speed can reach 200KBytes/sec (on blue-wired connection).
User can send generic data and audio data to PC over AMU2S interface at same time.
Audio data is supported in raw PCM (16bit/16Khz), SPP, OPUS format.

1. **Driver in the firmware**
   In the firmware, include the amu2s.c and amu2s.h source files to enable the feature.
   In this page, we are taking IOM0 as example, the IOM instance to be used and pins can be defined as needed.

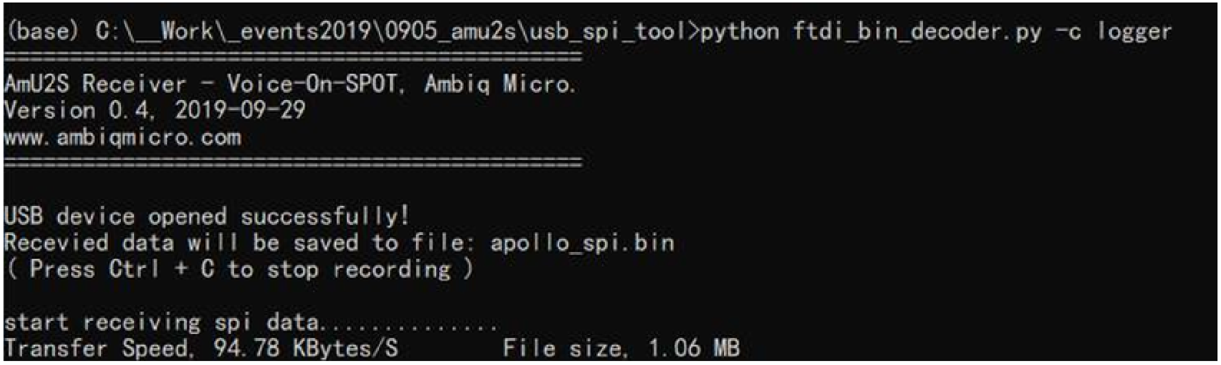
   In the application, include the amu2s.h in the source file where you need to send data.
   Call "amu2s_send()" function to send the data over SPI.

   The IOM is configured as non-blocking transfer.
   The feature uses around 16KBytes of RAM as data buffer.
   This can be reduced by changing the settings in the amu2s.h file.

   For more details, please check the amu2s.h header file.

2. **Amu2s PC tools**
   In theamu2s_tool_1010.7z, there are tools to capture and convert SPI data stream.
   a. ftdi_bin_decoder.py is the main script
      It starts the data collection and parse received data upon finish
   b. rtt_to_wav.py is the binary to wave file converter.
      Once you have all the data received and parsed as desired, you can use this script to convert them into .wav files.

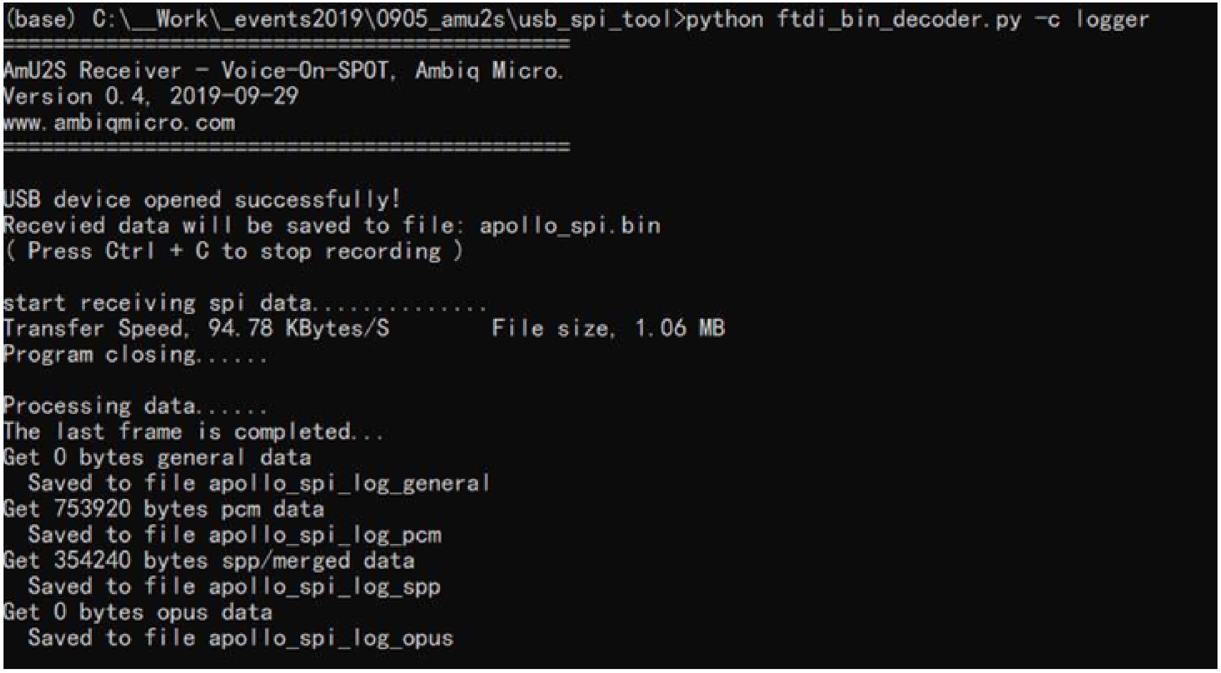
3. **How to use**
   a. Connect hardware:
      By default, we are using the following signals from Apollo3Blue as SPI signal lines and to the corresponding signals of FT4222H-EV, as shown below:
      Apollo3: **FT4222H-EV**
      GPIO5: SCK (IOM0)------------------SCK (pin9 of JP5)
      GPIO7: MOSI (IOM0)----------------MOSI (pin7 of JP5)
      GPIO42: CS (IOM0)------------------SS (pin11 of JP4)
      GND (any)------------------------------GND (any)
   b. Build and download the binary into Apollo3Blue EVB and run.
   c. Run ftdi_bin_decoder.py on PC:
      Example command option is: *python ftdi_bin_decoder.py -c logger*
      You should be able to see some output of the console as below:



```
(base) C:\__Work\_events2019\0905_amu2s\usb_spi_tool>python ftdi_bin_decoder.py -c logger
=============================================
AmU2S Receiver - Voice-On-SPOT, Ambiq Micro.
Version 0.4, 2019-09-29
www.ambiqmicro.com
=============================================

USB device opened successfully!
Recevied data will be saved to file: apollo_spi.bin
( Press Ctrl + C to stop recording )

start receiving spi data..............
Transfer Speed, 94.78 KBytes/S        File size, 1.06 MB
```

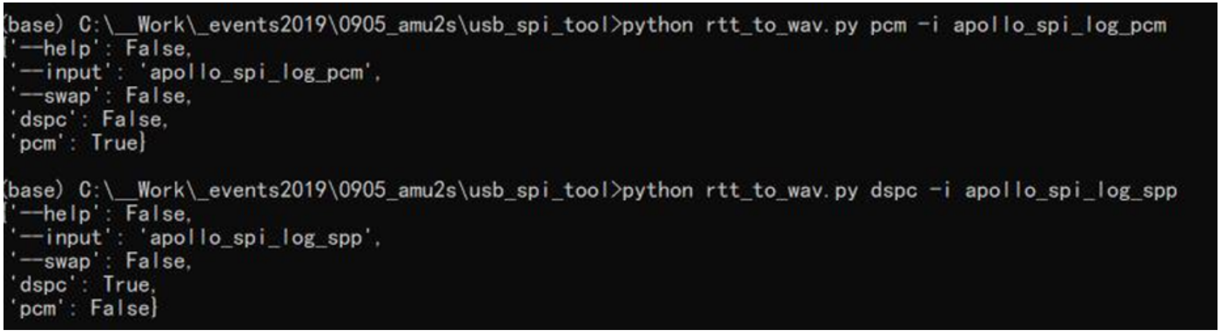
   d. Keep recording until you think you have collected enough data.
      You can still monitor SWO log for output in parallel.
   e. Use “Ctrl+C” to stop recording.
      After you stop the recording, the script will save a master data file: Apollo_spi.bin

And 4 child files: Apollo_spi_log_general, Apollo_spi_log_pcm, Apollo_spi_log_spp, Apollo_spi_log_opus.
(we currently implement only the pcm and spp, so other files should be empty.)



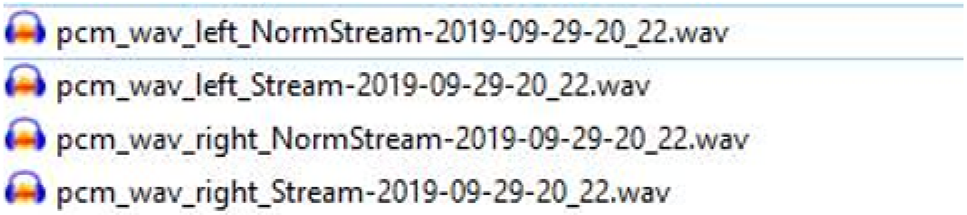
```
(base) C:\__Work\_events2019\0905_amu2s\usb_spi_tool>python ftdi_bin_decoder.py -c logger
============================================
AmU2S Receiver - Voice-On-SPOT, Ambiq Micro.
Version 0.4, 2019-09-29
www.ambiqmicro.com
============================================

USB device opened successfully!
Recevied data will be saved to file: apollo_spi.bin
( Press Ctrl + C to stop recording )

start receiving spi data..............
Transfer Speed, 94.78 KBytes/S        File size, 1.06 MB
Program closing......

Processing data......
The last frame is completed...
Get 0 bytes general data
  Saved to file apollo_spi_log_general
Get 753920 bytes pcm data
  Saved to file apollo_spi_log_pcm
Get 354240 bytes spp/merged data
  Saved to file apollo_spi_log_spp
Get 0 bytes opus data
  Saved to file apollo_spi_log_opus
```

f. Run rtt_to_wav.py script to convert raw binary into .wav files
Example command options are:
python rtt_to_wav.py pcm -I apollo_spi_log_pcm
python rtt_to_wav.py dspc -I apollo_spi_log_spp



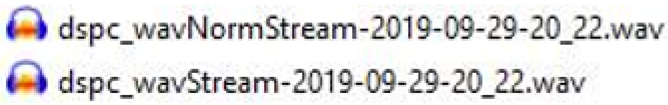
```
(base) C:\__Work\_events2019\0905_amu2s\usb_spi_tool>python rtt_to_wav.py pcm -i apollo_spi_log_pcm
{'--help': False,
 '--input': 'apollo_spi_log_pcm',
 '--swap': False,
 'dspc': False,
 'pcm': True}

(base) C:\__Work\_events2019\0905_amu2s\usb_spi_tool>python rtt_to_wav.py dspc -i apollo_spi_log_spp
{'--help': False,
 '--input': 'apollo_spi_log_spp',
 '--swap': False,
 'dspc': True,
 'pcm': False}
```

There will be .wav files generated in the same folder as the tool scripts:
Files with names starting from pcm_wav_xxxxx are left and right separated PCM raw audio (original and normalized).



pcm_wav_left_NormStream-2019-09-29-20_22.wav
pcm_wav_left_Stream-2019-09-29-20_22.wav
pcm_wav_right_NormStream-2019-09-29-20_22.wav
pcm_wav_right_Stream-2019-09-29-20_22.wav

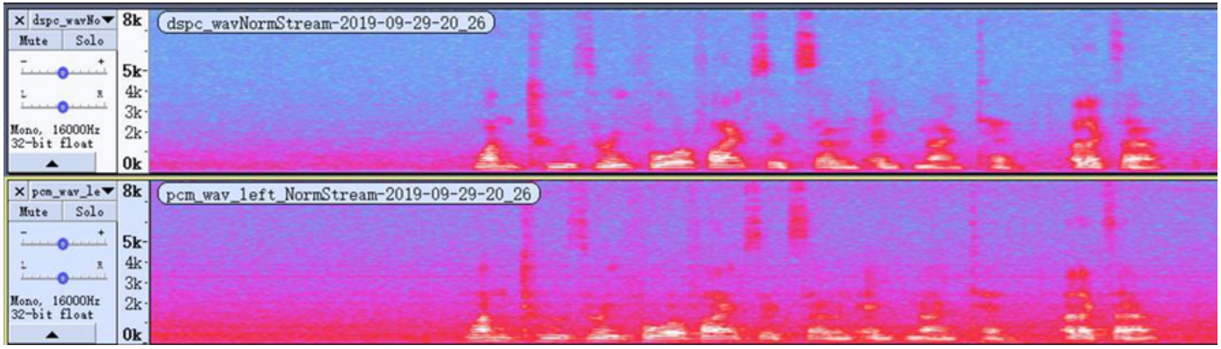
Files with names starting from dspc_wav_xxxx are mono audio after SPP (original and normalized).



dspc_wavNormStream-2019-09-29-20_22.wav

dspc_wavStream-2019-09-29-20_22.wav

You can play them or check the spectrogram.


dspc_wavNo
Mute
Solo
Mono, 16000Hz
32-bit float
8k
5k
4k
3k
2k
0k
dspc_wavNormStream-2019-09-29-20_26
pcm_wav_le
Mute
Solo
Mono, 16000Hz
32-bit float
8k
5k
4k
3k
2k
0k
pcm_wav_left_NormStream-2019-09-29-20_26