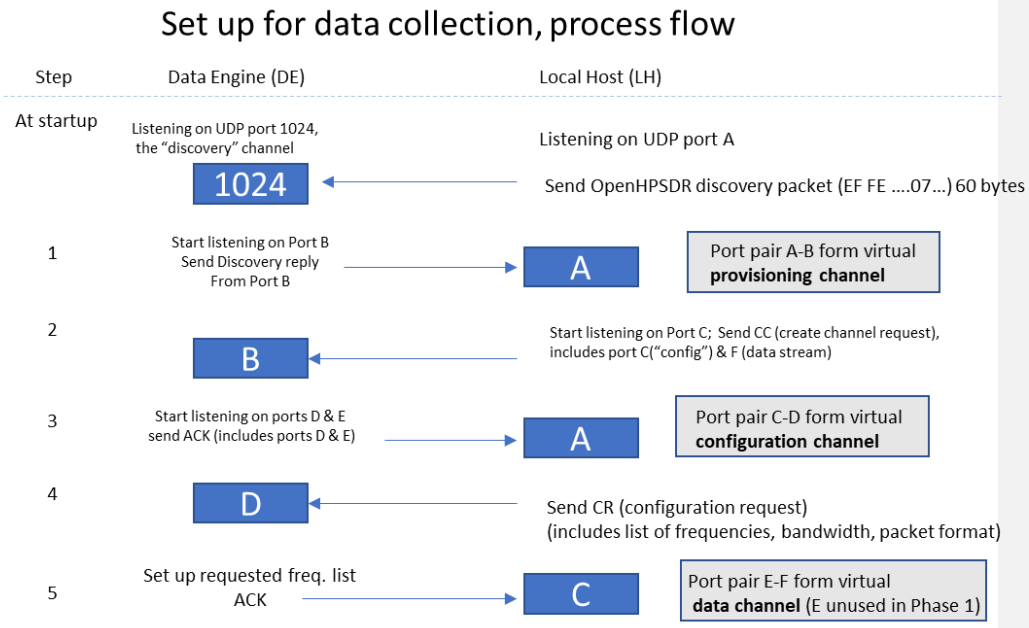

Set up for data collection, process flow
Step
Data Engine (DE)
Local Host (LH)
At startup
Listening on UDP port 1024, the "discovery" channel
Listening on UDP port A
1024
Send OpenHPSDR discovery packet (EF FE ....07...) 60 bytes
1
Start listening on Port B
Send Discovery reply
From Port B
A
Port pair A-B form virtual provisioning channel
2
B
Start listening on Port C; Send CC (create channel request), includes port C("config") & F (data stream)
3
Start listening on ports D & E
send ACK (includes ports D & E)
A
Port pair C-D form virtual configuration channel
4
D
Send CR (configuration request)
(includes list of frequencies, bandwidth, packet format)
5
Set up requested freq. list
ACK
C
Port pair E-F form virtual data channel (E unused in Phase 1)
At this point, the DE is ready to be triggered to start collecting data (continues next page)

Figure 1.

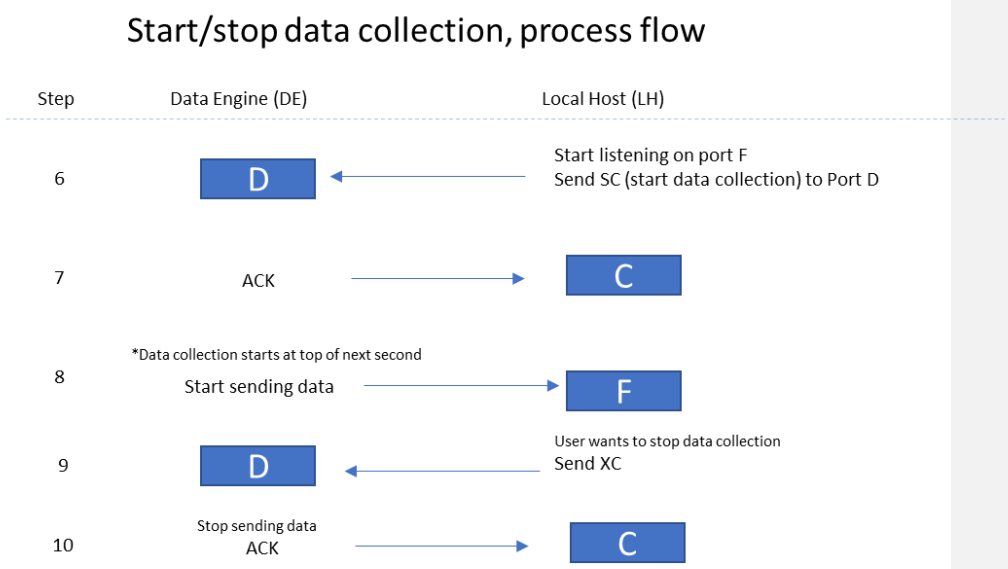

Start/stop data collection, process flow
Step
Data Engine (DE)
Local Host (LH)
6
D
Start listening on port F
Send SC (start data collection) to Port D
7
ACK
C
8
*Data collection starts at top of next second
Start sending data
F
9
D
User wants to stop data collection
Send XC
10
Stop sending data
ACK
C
(Future Phase) – prep to send data e.g., microphone stream (continues next page)
* For FT8 , data collection starts at top of next minute

Figure 2.

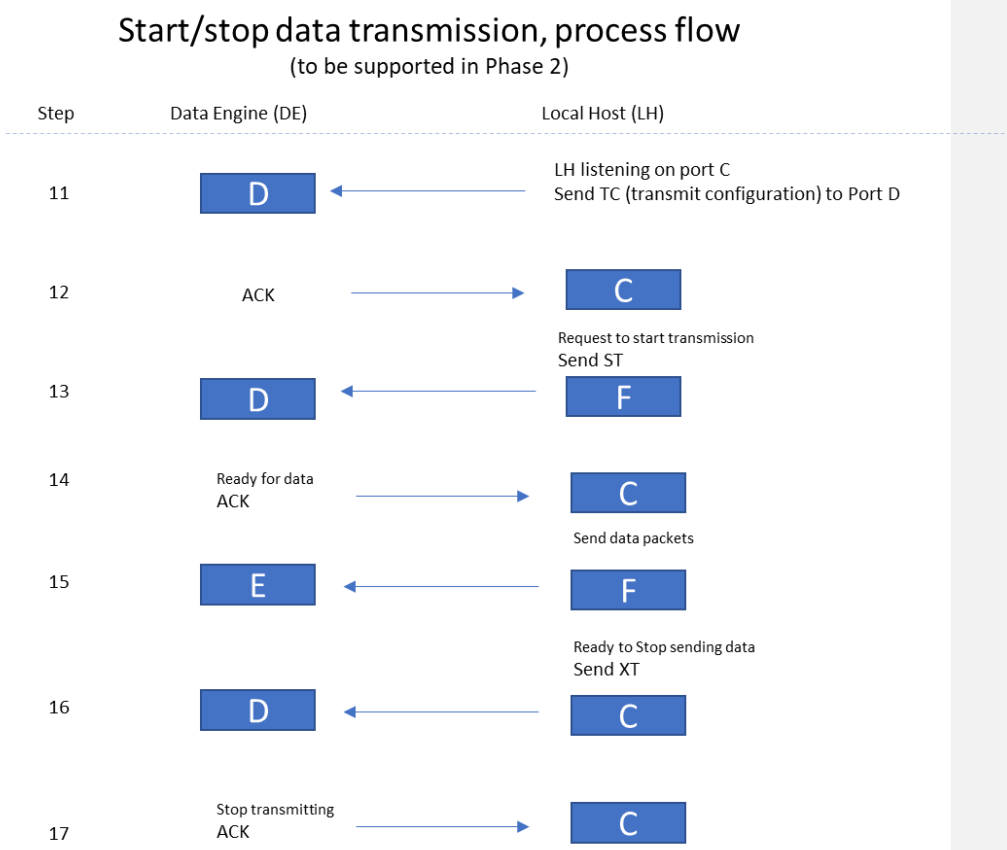


Figure 3.

Refer to narrative on following page.

Data Engine (DE) ← → Local Host Communication Protocol (LH)

Setting up: see Figure 1.

At startup, the DE listens for UDP on port 1024. Steps for standard data collection (Ring Buffer) as follows.

1. LH broadcasts a Discovery Packet (OpenHPSDR format starting with hex 0xEF 0xFE, with 0x07 in byte 7 identifying device as TangerineSDR). The DE is able to read the (randomly chosen) LH port from which this was sent, which we define as "Port A". The DE randomly selects a port to listen on ("Port B") for commands and sends a Discovery Reply to Port A. The LH reads Port B from the reply. This Port A – Port B pair forms the **Provisioning Channel.**
2. The LH selects another port ("Port C", which is in LH config.ini) and starts listening on that port. When ready to create the configuration channel pair, the LH sends the CC command (configuration channel create) to DE port B.
3. The DE selects 2 more ports (Ports D and E). For Phase 1, only Port D will be used. The DE starts listening on Port D. Assuming this setup works, the DE sends an ACK packet (which includes Port D and E values) to LH Port A. Ports C and D now form the **Configuration Channel.**
4. The LH sends a CR (configuration request to DE Port D). The CR packet includes a list of center frequencies and bandwidths (up to 16) and the desired packet format.
5. The DE attempts to set up the requested frequency channels. If the DE is successful, it sends ACK to LH Port C. At this point, the port pair E-F form the **Data Channel.** If DE can't satisfy request, it send NAK to LH Port C.

Starting/Stopping Data Collection: see Figure 2.

6. When ready to start collecting data, the LH sends the SC command to DE Port D.
7. The DE sends ACK to LH Port C.
8. The DE waits until the exact top of the next second (for FT8, waits until the exact top of the next minute), then starts collecting/sending data to DE Port F. These UDP data packets will be VITA-49 compliant when DE is collecting spectrum data for a single channel; when the DE is in multichannel mode, the packets will be VITA-49 compliant *except* that IQ data samples for the different subchannels will be interleaved in the data payload (for more details, see Section 4, below). In brief:

    a.—
    b. No ACKs are sent for data stream packets.
    c. If the DE is collecting data of other types besides spectrum from its radio (for example, magnetometer data), these data will be sent to the LH in a separate stream.
9. When LH wants to stop data acquisition, it sends the XC command to DE Port D.
10. DE sends ACK back to LH port C to acknowledge the stop command.

Starting/Stopping Data Transmission (Phase 2 and beyond): See Figure 3.

11. LH sends a TC ("transmission configuration") to DE Port D including the center frequency, packet format type, and data rate. The sets up a mic (or other) data stream.
12. DE ACKs the stream setup command to LH port C.
13. When ready to start transmitting, the LH sends ST command to DE port D.
14. DE sends ACK to LH port C.
15. LH streams UDP data to DE port E. (No ACKs sent for this). A watchdog timer will automatically stop transmission if time of transmission exceeds 5 minutes; in this case, a NAK is sent to LH.
16. When ready to stop transmission, LH sends XT to DE Port D
17. DE stops transmitting and sends ACK to LH port C

Error codes sent along with NAK

x0

1. attempted to start receive (or transmit) of a mode without a valid configuration
2. unsupported frequency requested
3. unsupported mode requested
4. unsupported data rate requested
5. sum of requested data rates exceeds device capacity
6. watchdog timeout: attempt to transmit for more than 5 minutes in a single transmission

Data Rate List

1. LH sends 2-byte command: R?
2. DE responds with "DR" followed by a table of supported data rates (starting with 1)
3. 32-bit integer rate number 32-bit integer rate (samples per second) for example:
    a. 1 4000
    b. 2 8000
    c. 3 12000
    d. 4 24000
    e. 5 48000
    f. 6 96000
    g. 7 128000
    h. 8 256000
4. Channel configuration commands specify a data rate by selecting one of the rate numbers, e.g., to specify a channel data rate of 8000 samples per second, the rate number of 2 is included in the channel configuration request. Note that when multiple subchannels will be running, the same data rate will be used for all subchannels.
5. The total data rate the DE is running at a given time is the number of running subchannels times the (single) data rate.

6. If the user tries to specify a number of subchannels and data rate that multiplies out to a higher total data rate than the DE can support, the DE sends a NAK (NK) in response to the configuration request.

Data Formats

Commands and inquiries going LH → DE always start with a 2-byte command type, defined as follows:

```
#define STATUS_INQUIRY      "S?"  // asks DE to send "OK" or "AK"
#define DATARATE_INQUIRY    "R?"  // asks DE to send a table of supported data rates
#define LED1_ON             "Y1"  // asks DE to turn on LED1
#define LED1_OFF            "N1"  // asks DE to turn off LED1
#define TIME_INQUIRY        "T?"  // asks DE to send the time from GPSDO
#define CREATE_CHANNEL      "CC"  // asks DE to create set of data channels
#define CONFIG_CHANNELS     "CH"  // gives DE channel configurations
#define UNDEFINE_CHANNEL    "UC"  // asks DE to drop its set of data channels
#define FIREHOSE_SERVER     "FH"  // puts DE into firehose mode
#define START_DATA_COLL     "SC"  // asks DE to start collecting data in ringbuffer mode
#define STOP_DATA_COLL      "XC"  // asks DE to stop collecting data in ringbuffer mode
#define DEFINE_FT8_CHAN     "FT"  // gives DE configuration for one FT8 channel
#define START_FT8_COLL      "SF"  // asks DE to start collecting FT8 data on all FT8 channels
#define STOP_FT8_COLL       "XF"  // asks DE to stop collecting FT8 data
#define LED_SET             "SB"  // in case we need to send a binary LED set byte
#define UNLINK              "UL"  // asks DE to disconnect from this LH
#define HALT_DE             "XX"  // asks DE to halt
#define RESTART_DE          "XR"  // asks DE to do a cold start
```

Responses sent from DE → LH also always start with a 2-byte command type, defined as follows:

```
#define DATARATE_RESPONSE   "DR"  // response: start of data rate table
#define TIME_STAMP          "TS"  // response: time of day
#define FT_DATA_BUFFER      "FT"  // this is an FT8 data packet
#define RG_DATA_BUFFER      "RG"   // this is a ringbuffer data packet
#define STATUS_OK           "OK"  or "AK"  // DE is alive / last command was accepted
```

Detailed content of the commands and buffers follows. In several cases, the C “union” indicates that the following fields occupy the same storage area, with the memory space required by the larger of the data types is reserved.

Here are the layouts of those data packets exchanged between DE and LH where the packet

1. CREATE CHANNEL.

```
typedef struct configChannelRequest
   {
    char cmd[2];  // content is "CC"
   uint16_t configPort;  // Port C
   uint16_t dataPort;    // Port F
   } CONFIGBUF;
```

Tells DE that LH wants to create a channel pair (this is Step 2 in the process flow), and tells the DE on what ports the LH will listen for channel configuration response ("Port C") and acquired data ("Port F"). The port numbers are unsigned 16-bit integers.

2. CONFIG (configure) CHANNELS

```
   struct channelBlock
        {
       int    channelNo;
       int    antennaPort;
       double channelFreq;
       };
   typedef struct channelBuf
        {
       char   chCommand[2];
       int    activeChannels;
       int    channelDatarate;
       struct channelBlock channelDef[16];
       } CHANNELBUF;
```

Tells DE how many subchannels to activate and the data rate, followed by a list of one to 16 channel blocks. Each channel block describes a subchannel with a channel number, antenna port, and channel center frequency.

Notes:

a. The number of active subchannels and channel data rate are 4-byte integers.
b. In TangerineSDR version 1, all subchannels run at the same data rate.
c. The channel number and antenna port values are 4-byte integers. The channel number is primarily for reference and debugging and its value is not critical.
d. TangerineSDR version 1 supports two antenna ports, numbered zero and one.
e. The channel frequency is the center frequency of the corresponding subchannel. It is an 8-byte integer specifying the frequency in Hz.

3. DATA RATE TABLE

```
   struct datarateEntry
       {
       int rateNumber;
```

```
    int rateValue;
    };

typedef struct datarateBuf
     {
     char buftype[2];
     struct datarateEntry dataRate[20];
    } DATARATEBUF;

typedef struct comboBuf
    {
    union
     {
      DATARATEBUF dbuf;
      char dbufc[175];
     }  ;
    } COMBOBUF;
```

This is the table sent from DE → LH in response to a DATARATE_INQUIRY, listing the data acquisition rates the DE supports. There may be up to 20 different rates. Each rate has a 32-bit integer data rate number and a 32-bit data rate value in samples per second.

Formatted: List Paragraph, Indent: Left: 0.31"

4. BUFFER DATA

NOTE: This section is completely different from the previous version of this specification, and reflects that we have decided to make the spectrum data buffer largely VITA-49 compliant. A specification can be found here:

https://www.redrapids.com/images/whitepapers/TWP-000-001-R00.pdf

TangerineSDR uses a version of this standard, as follows:

The TangerineSDR DE can be set up for spectrum output in two possible ways:

- Full VITA-49 compliance (each channel in a separate stream)
- VITA-49-T (all channels in a single stream, with data interleaved)

The second mode ("VITA-49-T") is necessary to support output of spectrum data in Digital RF.[1] The TangerineSDR implements the modes as follows:

4.1 VITA-49

4.1.1 Header

The data packet header starts with a 32-bit word:

[1] Technically, the VITA-49 standard could support multi-subchannel, interleaved IQ data in a single stream, by including a Channel Tag on each Data Item. This is not used in Tangerine Version 1 due to the processing and space requirements this would put on a microprocessor environment; this may change in future versions.

bits
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
0 0 0 X C T R R TSI TSF Packet Count Packet Size
Packet Type

Settings here are for the TangerineSDR DE in VITA-49 compliance mode:

- The packet type is set to 0001, indicating that a Stream Identifier is present.
- Bit "C" is set to 0, indicating that no Class Identifier is present in the data stream.
- Bit "T" is set to 0, indicating that there is no trailer included at end of packet.
- TSI is set to 1 (binary 01) indicating the time stamp is in UTC.[2]
- TSF is set to 1 (binary 01) indicating that that a Sample Count is included.
- The packet count is a 4-bit counter that counts from hex 0 to hex F and then repeats.
- The packet size is a 16-bit integer indicating packet size (in 32-bit words) including this header.

An example of a header: (hex) 10 53 08 0F
Breaking this down according to the above header layout, this header tells us we have a packet type of 0001 (stream identifier present); no Class Identifier or packet trailer; the time stamp will be UTC; the 3 indicates that this is packet# 3; the total packet size is 80F 4-byte words, which is decimal 2,063, or 8,252 bytes. Note that this includes the 4-byte UDP header.

4.1.2 Stream Identifier

The Stream Identifier is a 32-bit integer indicating the channel number. It is assumed that the system receiving the data knows the center frequency of each IQ channel. IMPORTANT: in the VITA-49 compliant mode, if multiple channels are running, each channel is sent in a separate stream.

4.1.3 Integer Seconds Timestamp

The timestamp is a 32-bit integer of UTC UNIX time, accurate to the second (footnote 2, above).

4.1.4 64-bit Sample Count

The sample count indicates the number of IQ samples sent so far in this data collection session, *not including the number of samples in the following payload*. This means that when a new data collection session is started, this value must be zero, and then incremented by the number of IQ samples sent. This allows the receiving system to detect missing packets.[3]

4.1.5 Data Payload

The data payload for the TangerineSDR consists of 1,024 IQ samples, each consisting of a pair of 32-bit floating point numbers.

4.2 VITA-49-T

4.2.1 Header

[2] In the Tangerine, the UTC time is a 32- bit UNIX time, i.e., seconds after Jan. 1, 1970.
[3] This is handled slightly differently in the VITA-49-T case; see following section.

When running in VITA-49-T mode, the header is the same as VITA-49 mode *except* that the first bit of the header is set to 1.

bits
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
1 0 0 X C T R R TSI TSF Packet Count Packet Size
Packet Type

This is to distinguish it from a packet that is standard VITA-49, and indicates that multiple channels may be interleaved in the Data Payload.

Other fields of the packet will be the same except for the following:

4.2.2 Stream Identifier
The stream identifier contains a 2-byte ASCII code (left-justified in a 4-byte fields), indicating the data collection mode for the stream:

RG – ring buffer or snapshotter data
MG – magnetometer data
FT – FT8 data
WS – WSPR data

Only one data stream of RG collection mode may run at a time; for FT8, the rightmost 2 bytes of the Stream Identifier is the channel number (ujo to 8 channels, “00” through “07”).
TODO: Steam ID details for WSPR need to be defined.
Note: RG, FT, and WS are IQ data, just with different sample rates and characteristics. Magnetometer data is stored as seven (7) 32-bit floating point values (Section 4.2.4).

4.2.3 Packet Count
The packet count is the number of IQ samples of each channel so far sent (essentially, it the count of IQ samples of Channel 1 that have been sent). So, if 2 channels are running, and 1,024 samples have been sent of each channel, the sample count is 1,024 (not 2,048). This simplifies the sample counting for missed packet handling in Digital RF.

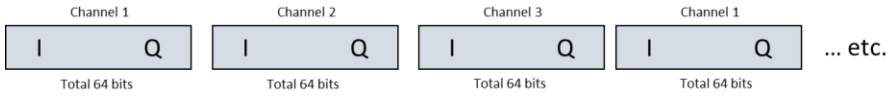
4.2.4 Data Payload
This section does not comply with the VITA-49 standard. When running in VITA-49-T mode, the Data Payload (for spectrum data) consists of the following number of IQ samples:

Int (1024 / # channels running) X 1024 (where result of the division is truncated)

For example, if the DE is running 9 channels, the number of samples in each buffer will be:

Int(1024/9)X1024 = 113 X 1024 = 1,017

The IQ samples from the multiple channels are interleaved as follows:



Each IQ sample is a pair of 32-bit floating point values. It is assumed in this case that the receiving system knows (a) the number of channels running, and (b) the center frequency of each channel.

For the magnetometer (when connected to the DE), will produce data (in Data Payload) consisting of seven (7) samples, each 32-bit floating point:

```
lt   x   y   z   rx   ry   rz
```

4.3 C structures for Data Layout

To clarify the specification, here are C language structures for working with the data:

```
struct IQdataSample
  {
  float I_val;
  float Q_val;
  };

typedef struct tangerineDataBuf
 {
 char VITA_hdr1[2];  // rightmost 4 bits is a packet counter
 int16_t  VITA_packetsize;
 char stream_ID[4];  // layout is data type dependent
 uint32_t time_stamp;
 uint64_t sample_counter;
 struct IQdataSample tangerineDatSample[1024];
 } TANGERINEBUF;
```

Note that, as defined in section 4.2.4, not all elements in the array `tangerineDataSample` may be used in the case where the number of channels running does not divide evenly into 1024.

4.4 FT8 Data

Monitoring FT8 works differently than collecting ringbuffer data. When doing data acquisition for ringbuffer, all subchannels are interleaved and sent in a single buffer; on the other hand, for FT8, each different band being monitored is sent in a separate buffer, and the buffers are interleaved. These differences are to make for interoperability with existing software (ringbuffer data is optimized for use with the Digital RF data storage package; FT8 is optimized for use with the FT8 decoding package). Up to 8 FT8 channels can run concurrently.

Each buffer of FT8 data (being sent from DE → LH) is identified by "FT" in the bufType and a channel number corresponding to the channel number in the FT8 Setup Panel. Since buffers are interleaved in FT8, all buffers are sent to the same UDP port.

W. Engelke, AB4EJ, 07 JUL 2020