PERSONAL SPACE WEATHER SYSTEM

TANGERINE SDR

LOCAL HOST TO DATA ENGINE PROTOCOL & USE CASES

Version Number: 1.2~~1~~

Version Date: Jun 8, 2021~~Nov. 23, 2020~~

W. Engelke AB4EJ

The University of Alabama

VERSION HISTORY

Version Number	Implemented By	Revision Date	Approved By	Approval Date	Description of Change
0.1, 1.0	W. Engelke				Initial version + cleanup.
1.1	W. Engelke	11/23/2020			Added ability to switch out of Firehose-L mode. Added LED1 turn-off;
1.2	W. Engelke	6/8/2021			Local Host to set all ports and communicate these to Data Engine; added Appendix 2 Error table; added section on Memory write/read/response

CONTENTS

1 INTRODUCTION

This Communication Protocol describes communication in the Tangerine SDR between the Local Host (Single Board Computer, or SBC) and the Data Engine (DE). It is organized into two parts:

- **Use Cases**, which explain the content of communication by logical process
- **Technical Details**, which describe the underlying technology approaches used to implement the logical actions.

Figure 1 shows an overview of the system architecture. This document is concerned with the communication (solely within the Tangerine) between the Local Host (LH) and the DE.

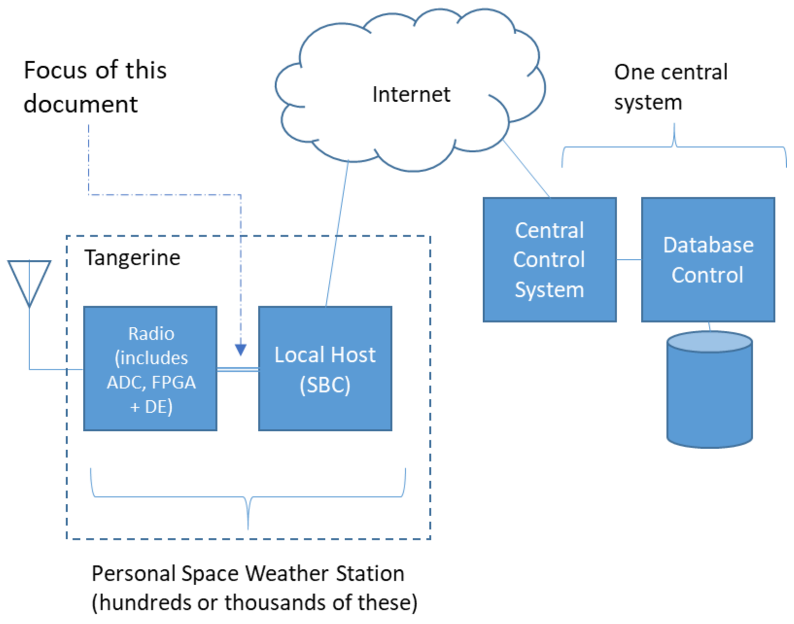


Figure 1. Conceptual Overview.

Note that the physical connection between the LH and DE goes through a gigabit Ethernet switch, which is not shown here.

2 USE CASES: OVERVIEW

2.1. ABOUT THIS SECTION

The Use Cases section progresses through the logical processes of operating the DE, starting with initial connection and progressing through setting up the DE for one or more channels, configuring them with data rates and center frequencies, starting/stopping data collection, and then includes the incidental actions that can be used at any time.

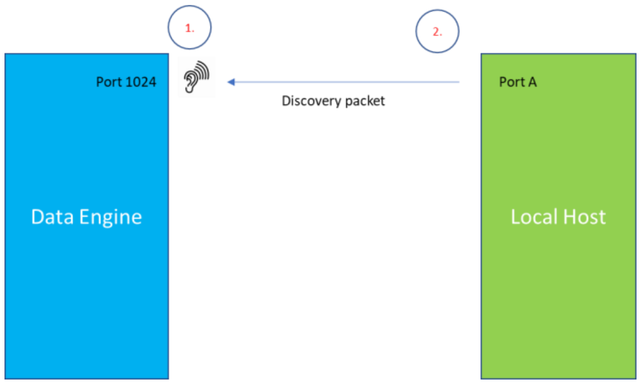
This document is based on an earlier document “DE_LH_protocol_V8” which has been previously circulated for review and comment within the Tangerine SDR development community. The protocol described here has already been implemented and tested between the LH prototype and the DE simulator.

The design philosophy of the protocol is to emphasize simplicity while implementing a number of capabilities not included in previous similar TAPR SDR-to-PC protocols such as HPSDR V1 and V2. One basic idea as part of this simplicity of operation is that a user should be able to control the DE with no more software than a terminal emulator, and commands/responses (except for spectrum data) should be human readable.

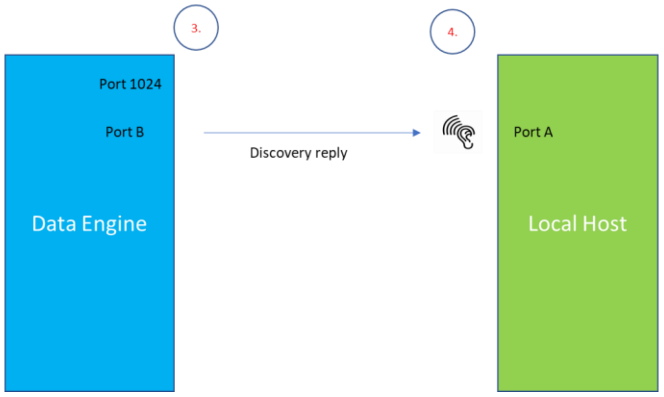
2.2. INTRODUCTION: PROCESS FLOW OVERVIEW

The overall process flow is designed to allow multiple Data Engines and multiple Local Hosts to be on the same Local Area Network, each (potentially) supporting multiple channels (each potentially with multiple subchannels), and maintain connectivity as needed without confusion. For this, we have the concept of channel pairs. For purposes of explanation, we designate the involved ports with letter designations A through F. Refer to Figure 1. (The narratives in the Use Cases explain the process in more detail).

First, we use a discovery process to connect the DE and LH. See Figure 1, below.

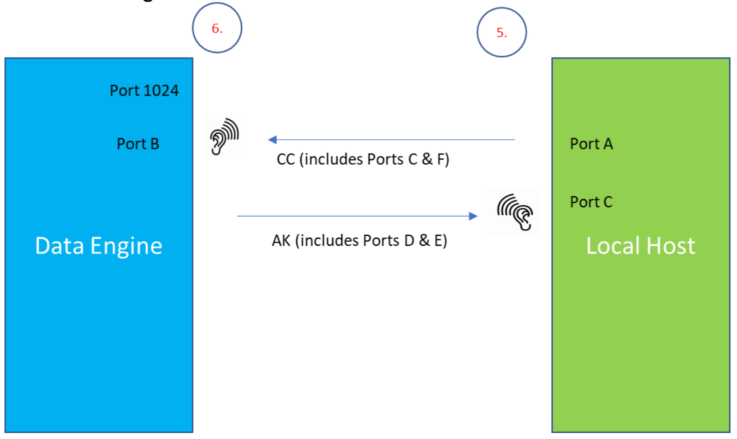


- At startup the DE listens on its port 1024. (1, 2)
- The LH sends a discovery packet from its provisioning port (“Port A”).; Figure 1, above. The DE determines the Port A port# from the Source Port field in the UDP header. (LH knows outbound port# via a getsockname call).

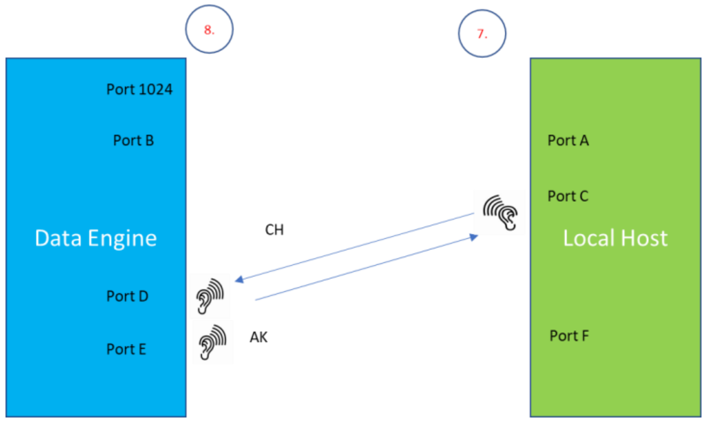


- The DE decides what port to use as Port B, and from this Port sends a discovery reply to Port A~~responds~~; from this response, the LH learns the DE’s IP address and provisioning port ("Port B"). Port B is in the Source Port# of the reply's UDP header. (3, 4)

- The DE now starts listening on its Port B. Port A – Port B form the **Provisioning Pair.** See Figure 3, below.

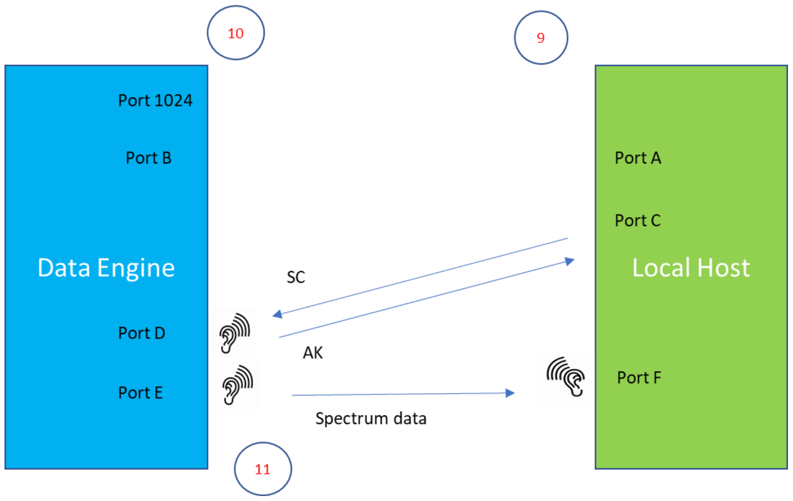


- The LH now starts listening on its Port C, and sends a Create Channel ("CC") request to DE Port B, which also tells the DE values for LH Port C and Port F. (5)
- The DE begins listening on its two ports Port D and Port E: these are the DE's configuration and data channels, respectively).
- The DE sends an AK to the LH which includes values for Port D and Port E. (6)
- The port pair Port C – Port D form the **Configuration Pair**; the Port pair Port E and Port F form the **Data Pair**. See Figure 4, below.

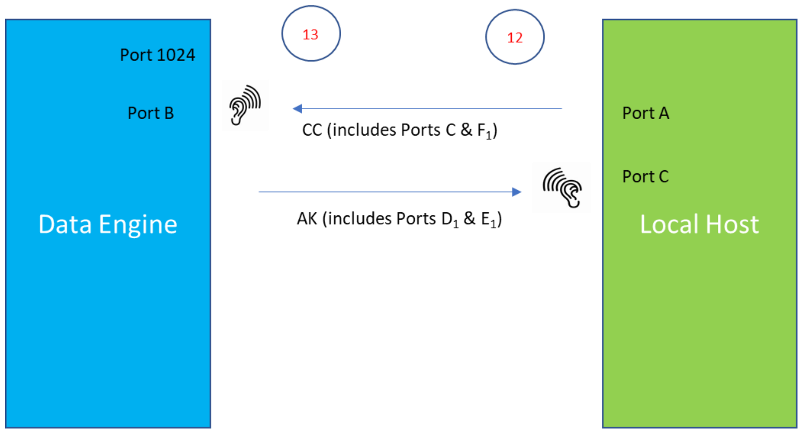


- To configure a channel, the LH sends a Configure Channel Request ("CH") to DE Port D for the desired channel. This request includes the data acquisition rate and for each subchannel its center frequency and radio port (0 or 1). The DE sends an AK if it can accommodate the request. (7,8)

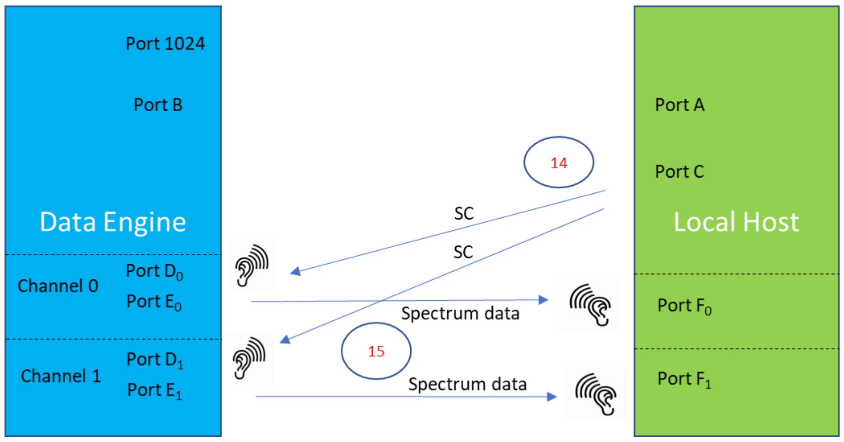
- When the LH wants to start data collection, it sends the SC command to the DE Port D for this channel. The DE acknowledges the request with AK to the LH Port C and starts sending spectrum data to LH Port F (9, 10, 11); Figure 5, below.



- Additional channels can be added and configured up to the capacity of the DE. To add a second channel, for example, the LH sends a CC to the DE Port B, specifying channel 1 (the first channel created was Channel 0); Figure 6, below.


13
12
Port 1024
Port B
Data Engine
Port A
Port C
Local Host
CC (includes Ports C & F$_1$)
AK (includes Ports D$_1$ & E$_1$)

- Assuming the DE is able to fulfil the request, it adds a second channel pair with AK, as shown above. (12, 13) Now, we have two channels that are independently controllable; Figure 7, below.



Each channel is numbered and can be started and stopped; when the LH wishes to delete a channel, it can issue the UC (Undefine Channel) command.

This has been tested using the LH code and simulator up to 3 simultaneously running channels, and performance is adequate to capture all data on the Raspberry Pi 4 B+ (4 GB).

The use cases below provide more detail as to operation for the major activities of:

Discovery
Create Channel
Configure Channel
Start Channel
Stop Channel
Request List of Data Rates
Request Telemetry
Restart
Delete channel
Status Inquiry
Firehose Server Connect
LED1 on/off
Memory write/read

Additional possible use cases yet to be defined include:
Load Firmware; Unlink

Unlink

USE CASES

USE CASES

All use cases are described as REQUEST – REPLY. Communication between the DE and LH is synchronous, with the DE responding with AK (acknowledged) or NK (negative acknowledgement) to each command. When the DE is in data collection mode, it sends continuous spectrum data according to the channel configuration(s) and no AK or NK are used.

3.1 DISCOVERY

Two different discovery methods are planned. The first conforms to HPSDR protocol 1; the second is a new method designed to allow manual operation of the Data Engine with commands that can be issued from a terminal emulator such as linux nc.

3.1.1 UDP Discovery using HPSDR Protocol 1

REQUEST. Upon startup, the DE listens for packets on port 1024. From HPSDR protocol:

> The Discovery packet is a UDP/IP frame sent to MAC address FF:FF:FF:FF:FF:FF and port 1024 of the broadcast address of the local subnets1 with the following payload:
>
> `<0xEFFE><0x02><60 bytes of 0x00>`

REPLY. The LH sends a Discovery Packet from its Port A. When a Discovery packet is received from the LH, the DE sends a reply packet from its Port B. (Ports A and B may be randomly selected or hard coded, but if hard coded, care must be taken to ensure that multiple devices on the same network use different ports.)

From HPSDR (version 1) protocol:

> Upon receipt of this broadcast a Metis board will reply with a UDP/IP frame sent to the 'from port' on the IP Address and MAC Address of the PC originating the Discovery broadcast.
>
> The payload of the UDP/IP reply frame is as follows:

```
<0xEFFE><Status><Metis MAC Address><Code
Version><Board_ID><49 bytes of 0x00>
```

> where

- Status = 1 byte, 0x02 if Metis **(/DE)** is not sending data and 0x03 if it is
- Metis MAC Address = 6 bytes (MAC Address of the Metis responding to the Discovery Broadcast) **(new: DE MAC Address)**
- Code Version = 1 byte, version number of code currently loaded into Metis
- Board_ID = 1 byte, 0x00 = Metis, 0x01 = Hermes, 0x02 = Griffin, 0x04 = Angelia,
- 0x05 = Orion, 0x6 = Hermes_Lite, **(new: 0x7: Tangerine)**

Note1: The IP address of the Metis board(s) responding can be obtained from the IP header.

The combination of the LH "from" port and the DE "from" port in this exchange is termed the Port A – Port B **Provisioning Channel**.

As will be seen in the following sections, commands and responses between the LH and DE are intended to use human-readable **space-delimited ASCII text** (a string), with the idea that it should be possible to command the DE using the Linux utility `nc`. Each string is terminated with 0x00 so it can be used as a string in C.[1] Hereinafter, a space is denoted by <~~b~~>, i.e., 0x20.

3.1.2 Manual Discovery

1. Initial setup: the user must determine the IP address of the DE.[2] The mac address should be provided with the DE hardware. To find the assigned IP address (with DE powered on), use a computer on the same LAN as the DE, and using terminal (Linux) or cmd (Windows) enter: arp -a . A list will be shown; find the DE's mac address in the list, and its IP address appears on the same line.
2. REQUEST. Using nc (or other terminal emulator[3]), connect to port 1024 of the DE, for example:

```
nc -u 192.168.1.85 1024
```

Once the terminal is connected, the user enters: `TA`

This command acts as a discovery packet to the DE.

3. REPLY. The DE responds by sending AK followed by whichever port the DE wishes to use as Port B for manual operation, sending this to the requester in human readable form, for example:

```
AK 25001
```

Here, the DE will start listening on its port 25001 for the Create Channel request (next section).

3.2 CREATE CHANNEL REQUEST

1. REQUEST. The LH selects a port ("Port C") for configuration. From Port C, LH sends the CC (Create Channel) packet to DE Port B. The CC packet is as follows (<~~b~~> indicates a space, i.e., 0x20):

[1] This leaves open the question of whether check digits or other error-proofing methods need to be employed.
[2] The TangerineSDR system assumes that the DE and LH are connected to a network such that they can be issued IP addresses.
[3] There is a comparable utility (ncat) for Windows.

```
     CC<ƀ>channel No.<ƀ>config port<ƀ>data port 0x00
```

All data here is handled as strings. The config port (“Port C”) and data port (“Port F”) are those ports on which the LH will listen for configuration information and data, respectively.

Example:

```
CC <ƀ>  0 <ƀ>  40001 <ƀ>  40002 0x00
```

This sets up channel 0 (zero) and indicates that the LH will listen for configuration info on it port 40001 and (spectrum) data on its port 40002 for that channel.

2. REPLY. Assuming the DE can satisfy this Create Channel request, the DE selects two ports for its own config port (“Port D”) and data port (“Port E”), and send an acknowledgement packet to LH Port A that also includes these two ports:

```
AK<ƀ>config port<ƀ>data port 0x00
```

The port pair C – D form the configuration channel; the port pair E-F for the data channel. Note that Port E is unused in phase 1, as it is for microphone data to be sent to a future DE that includes a transmitter.

Example:

```
AK <ƀ>  50002 <ƀ>   50003
```

Here, the DE accepting the CC it just received and indicating that it will listen for configuration info on its port 50002. (In the future, where they Tangerine supports a transmitter, it will listen also on port 50003 for microphone data).

If the DE cannot accommodate the channel request, it sends NK followed by an error code (see Error Code Table in DE_LH_protocol_V8.docx, p. 5).

3.3 CONFIGURE CHANNEL REQUEST

1. REQUEST. The LH uses its configuration channel (“Port C”) to send a CH (Configuration Request) to the DE at the DE port for this channel, containing:

```
CH <ƀ> channel No.<ƀ>data-standard<ƀ>subchannel-count<ƀ> data-rate<ƀ> 1 to
16 subchannel-definition-blocks
```

Contents are defined as:

a. Data-standard: governs layout of data streams
    1. V4: VITA-49 compliance (separate streams for subchannels)
    2. VT: VITA-T, which uses interleaved IQ data for subchannels[4]

[4] This is a version of VITA-49 intended only for use with the Tangerine, for interoperability with Digital RF. Users needing VITA-49 compliance should specify V4 here.

b. Subchannel count: number of subchannels to be sent for this channel. If using V4, this is the number of streams; if using VT, this is the number of subchannels interleaved in each packet within a single stream.

c. Data-rate: the data acquisition rate for this channel in samples per second. Note that all subchannels within a single channel must have the same data rate. If multiple channels are being used, each channel may have its own data rate.

d. Subchannel-definition-block: consists of:
`Subchannel No<ƀ> antenna port<ƀ> center frequency in MHz`
The antenna port may be 0 or 1.

Example:

```
CH  1  V4  5  4000 0 0 3.573 1 0 7.074 2 1 14.074 3 1 21.074 4 1 28.074 0x00
```

(The <ƀ> space indicators are omitted here for readability). Here, the LH is requesting to set up Channel# 1 using VITA-49 format, with 5 subchannels (which means 5 streams in the case of VITA-49), all of which will run at 4,000 samples per second. The subchannels then are:

0 – antenna port 0, center frequency 3.573 MHz

1- Antenna port 0, center frequency 7.074 MHz
2- Antenna port 1, center frequency 14.074 MHz
3- Antenna port 1, center frequency 21.074 MHz
4- Antenna port 1, center frequency 28.074

This would be a typical channel layout for monitoring five FT8 bands.[5]

2. REPLY. If the DE can accommodate the request, it sends AK . If the DE cannot accommodate the channel configuration request, it sends NK followed by an error code (see Error Code Table in DE_LH_protocol_V8.docx, p. 5).

3.4 START CHANNEL

1. REQUEST. The LH uses its configuration channel ("Port C") to send SC (Start Collection) to the DE at the DE port for this channel, containing:

```
SC <ƀ> channelNo 0x00
```

2. REPLY. If the DE can accommodate the request, it sends AK, and starts spectrum transmission to the LH **at the exact top of the next second.** If the DE cannot accommodate the Start Collection request, it sends NK followed by an error code (see Error Code Table in DE_LH_protocol_V8.docx, p. 5). The DE also starts collecting and sending data to the Port F for this channel, following the configured data format (V4 or VT), data rate, and subchannel

[5] There may be a need to account for the difference between dial frequency and center frequency, as typically in FT8 and WSPR the dial frequency is lower than the center frequency by half the mode's bandwidth.

layout for the requested channel. The format of spectrum data is described in the Technical Details section.

3.5 STOP CHANNEL

1. REQUEST. The LH uses its configuration channel ("Port C") to send XC (Stop Collection) to the DE at the DE port for this channel, containing:

```
XC <ƀ> channelNo 0x00
```

Commented [WE1]:

2. REPLY. If the DE can accommodate the request, it sends AK . If the DE cannot accommodate the Stop Collection request, it sends NK followed by an error code (see Error Code Table in DE_LH_protocol_V8.docx, p. 5).

3.6 REQUEST LIST OF DATA RATES

1. REQUEST. The LH uses its configuration channel ("Port C") to send R? (Data Rate List Request) to the DE at the DE port for this channel, containing:

```
R?  0x00
```

2. REPLY. If the DE can accommodate the request, it sends a data rate list as follows:
   `DR <ƀ> 1 to 10 data rate blocks`
   … where a data rate block is an increasing integer, a space, and an integer samples-per-second. The data rate blocks are separated by spaces.

   Example:

```
DR<ƀ>1<ƀ>375<ƀ>2<ƀ>4000<ƀ>3<ƀ>8000<ƀ>4<ƀ>12000<ƀ>5<ƀ>24000<ƀ>6<ƀ>48000 0x00
```

   In the above example, the DE is reporting that it can handle 7 data rates, at speeds of 375, 4000, 8000, 12000, 24000, and 48000 samples per second. Data rates provided in this list will be presented to the user as the options to select from when configuring a channel.

   If the DE cannot send a data rate list, it sends NK followed by an error code (see Error Code Table in DE_LH_protocol_V8.docx, p. 5).

3.7 REQUEST TELEMETRY

1. REQUEST. The LH uses its configuration channel ("Port C") to send T? (Telemetry Request) to the DE at the DE port for this channel, containing:

```
T? 0x00
```

2. REPLY. If the DE can accommodate the request, it sends a telemetry list as follows:
   ```
   DT <ƀ> 1 to n data blocks
   ```
   … where a data block is a 2- character ASCII code indicating what type of data follows, a space, and telemetry value. Each telemetry value is a string value expressing the data value and containing no spaces. The data blocks are separated by spaces.

   Example:

```
DT<ƀ>TP<ƀ>54.5<ƀ>SN<ƀ>637483<ƀ>GP<ƀ>1<ƀ>DT<ƀ>20211015T1503Z<ƀ>VL<ƀ> 5.1 0x00
```

   Here, the DE telemetry indicates its temperature (TP) is 54.5 deg.,C, its serial number (SN) is 637483; it has a GPSDO connected (GP of 1), the date-time is Oct. 15, 2021 at 15:03 Z, and it is receiving a power supply voltage (VL) of 5.1 VDC. Note that this is not a definitive list of telemetry items, but just examples of possible items.

   If the DE cannot send a list of telemetry data, it sends NK followed by an error code (see Error Code Table in DE_LH_protocol_V8.docx, p. 5).

3.8 RESTART

1. REQUEST. The LH uses its provisioning channel ("Port A") to send the XR command to the DE provisioning channel ("Port B").
2. REPLY. The DE cold starts and returns to waiting for UDPdiscovery. All configurations and settings are returned to manufacturer original values.

3.9 UNDEFINE (DELETE) CHANNEL

1. REQUEST. The LH uses its provisioning channel ("Port A") to send the UC command with channel number to the DE provisioning channel ("Port B"):
   ```
   UC 1 0x00
   ```
2. REPLY. The DE halts any data collection occurring on that channel, deletes subchannels defined for this channel, and acknowledges: AK

   If the channel does not exist, the DE sends NK.

3.10 STATUS INQUIRY

1. REQUEST. The LH uses its provisioning channel ("Port A") to send the S? command with channel number to the DE provisioning channel ("Port B"):
   ```
   S?
   ```
2. REPLY. The DE replies with AK

   If the DE is in a hard error state, the DE sends NK.

3.11 FIREHOSE SERVER CONNECT/DISCONNECT

This pertains to data collection mode Firehose-L (“Firehose Local”), which is a special case of data collection where spectrum data is to be sent directly from the DE to a local high-capacity server (instead of being sent to the LH). This is for cases where very wide bandwidth and/or very high sample rates are to be used where the combination would be beyond the capacity of the SBC used as the LH.

This command and its mode make the following assumptions:

- The channel to be used has already been created (CC) and configured (CH), but data collection (SC) has not yet started.
- The user knows the IP address and Port of the high-capacity server and has configured these values into the appropriate configuration values in the LH.
- The high capacity server is network-reachable from the DE.
- The high capacity server is set up and listening for UDP traffic on the given port.
- Control of the DE will remain with the LH.[6]
- The network has sufficient bandwidth to handle the planned traffic along the entire route from DE to high capacity server (including sufficiently fast switches and correct cable types[7]).

1. REQUEST. The LH sends (to “DE Port D”) the Firehose Server (“FH”) command which puts the specified channel into Firehose-L mode and defines the destination for the DE output data, as follows:

```
FH<ƀ>channelNo<ƀ>server-IP<ƀ>server-port 0x00
```

Example:

```
FH<ƀ>  1<ƀ>  168.192.1.125 <ƀ> 5000 0x00
```

Here, the DE (once it receives the SC command for this channel) will send UDP spectrum data (according to the current configuration for this channel) to IP address 168.192.1.125 port 5000.

2. REPLY. Assuming the DE can accommodate this request, it sends AK to the matching Port C of the LH; it then awaits the SC command for this channel. The DE sends NK with error status code if it cannot meet the request.

3. REQUEST. The DE remains “connected” to the Firehose-L high speed server, and starting/stopping data collection on Channel 0 controls this data feed. If the user wants to discontinue sending data to the Firehose-L server and resume data collection using Ringbuffer, Firehose-R or Snapshotter, the user

[6] It may be possible to install the Tangerine LH software suite on the high capacity server itself

[7] Gigabit or faster switches and CAT-6 or higher cabling are required.

deactivates Firehose-L mode in the user interface. In this case, the LH sends the Stop Firehose-L mode command:

```
XF<b>channelNo
```

4. REPLY. Assuming the DE can accommodate this request, it sends AK to the matching Port C of the LH. (Note: if the DE is actively sending data to the Firehose-L server and receives this command, it stops data collection on the channel.) The DE switches out of Firehose-L mode. The DE sends NK with error status code if it cannot meet the request.

3.12 LED1 ON/OFF

1. REQUEST. The LH uses its provisioning channel ("Port A") to send the Y1 command with channel number to the DE provisioning channel ("Port B"):

   ```
   Y1
   ```

2. REPLY. The DE turns on the requested LED on the board and replies with AK.

   If the DE is in a hard error state, the DE sends NK.

3. REQUEST. The LH uses its provisioning channel ("Port A") to send the N1 command with channel number to the DE provisioning channel ("Port B"):

   ```
   N1
   ```

4. REPLY. The DE turns off the requested LED on the board and replies with AK.

   If the DE is in a hard error state, the DE sends NK.

Formatted: Indent: Left: 0"

Note: additional LED controls may be defined in future revisions.

3.13 DEVICE READ/WRITE/RESPONSE

The Local Host will have the ability to read and write *a single byte* to locations on the Data Engine defined by a specific slot#, interface, and address. The <b> symbol indicates a blank (0x20). Errors (resulting in a NAK) are listed in the Error Table.

Below, the format for all inputs is 16 bits, expressed as 0xNNNN (e.g., 0x01F5). Note that some devices may only be able to accept a value with fewer bits (e.g., I2C uses 7 bit addresses); in such cases, the least significant bits (rightmost) will be used and higher order bits ignored; also, data being written is always a single byte (rightmost within 0xZZ**ZZ**). For simplicity, in Phase 1, both hexadecimal bytes must be supplied, with the "0x" prefix included.

Formatted: Font: Bold, Underline

a. Memory write ("MW")

1. LH sends a packet as follows:

```
MW <b> 0xssss <b> 0xiiii <b> 0xnnnn <b> 0xzzzz
```

ssss = slot#
iiii = interface
nnnn = address
zz = data, single hexadecimal byte (e.g., 0x03 is hex 03)

2. If the write is successful, the DE returns: AK
3. If there is an error, the DE returns NAK (error code)

b. Memory read (“MR”)

1. LH sends a packet as follows:

```
MR <b> 0xssss <b> 0xiiii <b> 0xnnnn
```

ssss = slot#
iiii = interface
nnnn = address

2. If the read is successful, the DE returns a packet as follows:

```
RR <b> s <b> iiiii <b> nnn <b> len <b> 0xZZ
```

If there is an error, the DE returns NAK (error code; see Error Table)

3.13.1 Example Exchange

```
DE                                    LH
    ← MW 0x0001 0x0005 0x0AC2 0x0001
                 AK →
    ← MR 0x0001 0x0005 0x0AC2
      RR  MW 0x0001 0x0005 0x0AC2 0x0001 →
```

In the above exchange, the Local Host writes a single byte containing 1 to slot 1, interface 5, address AC2. The DE acknowledges. The LH then reads the byte and the DE responds with the content of that memory location, which is now set to 1. Reading locations that have not been set produces an undefined result.

4. TECHNICAL DETAILS

4.1 SPECTRUM DATA FORMAT

Spectrum data is sent from the DE to the LH in two possible ways:

- VITA-49 compliant
- VITA-T: compliant with VITA-49 except for how subchannel data are handled

This second form, "VITA-T" was developed to optimize for handling high data rates with Digital RF as the end format, using a single-board computer (Raspberry Pi or similar).

VITA-49 has a variety of options, which are selected by the setting of bits in the record header. You can find a complete exposition of VITA-49 in the Red Rapids Technical White Paper REF-000-001-R00:

(https://www.redrapids.com/images/whitepapers/TWP-000-001-R00.pdf). The remainder of this section describes how the standard is used in the Tangerine SDR.

4.2 VITA-49

If you want to operate the TangerineSDR DE in VITA-49 mode, you specify this in the third field of the Configure Channel (CH) command. The LH puts the DE into this mode for collecting FT8 and WSPR signals.

A VITA packet consists of a header and a data payload, as follows.

4.2.1 Header

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

```
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:

- Bits 31 – 28: The packet type is set to 0001, indicating that a Stream Identifier is present.
- Bits 27 – 24: 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.
- Bits 23 – 22: TSI is set to 1 (binary 01) indicating the time stamp is in UTC.[8]
- Bits 21 - 20: TSF is set to 1 (binary 01) indicating that that a Sample Count is included.
- Bits 15 to 0: The packet count is a 4-bit counter that counts from hex 0 to hex F and then repeats (for recognizing if a packet has been lost).

[8] In the Tangerine, the UTC time is a 32- bit UNIX time, i.e., seconds after Jan. 1, 1970.

- The packet size is a 16-bit integer indicating packet size (in 32-bit words) including this header. (You must multiply by 4 to get the # of bytes!)

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 40-byte UDP header.

4.1.2 Stream Identifier

The Stream Identifier is a 32-bit integer indicating the subchannel 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 subchannels are running, each channel is sent in a separate stream. Subchannels are multiplexed as alternating packets in the stream, all going to the same IP port.

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.[9]

A C structure able to handle this data format is:

```
typedef struct VITAdataBuf
 {
 char VITA_hdr1[2];
 int16_t  VITA_packetsize;
 char stream_ID[4];
 uint32_t time_stamp;
 uint64_t sample_count;
 struct dataSample theDataSample[1024];
 } VITABUF;
```

[9] This is handled slightly differently in the VITA-49-T case; see following section.

4.1.5 Building the VITA-49 packet

- Assign the first header byte as fixed 0x10.
- Maintain a 1-byte counter which cycles from zero to fifteen, then repeats.
- Fix the first nybble of the second header byte to 5, and put the 1- byte counter into the second nybble; this value then ranges from 0x50 to 0x5F.
- Put the channel number (in binary) into the stream_ID.
- Put the Unix time (32 bits, correct to the second) into the time_stamp.
- Put a monotonically increasing integer into the sample_count. This must start from zero each time the DE executes the SC command for the channel.
- Fill the data area with 1,024 IQ samples:

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

Iterating over such a buffer of IQ samples can be handled by iterating i from 0 to 1023, where the values then are

```
this_I_value = theDataSample[i].I_val;
this_Q_value = theDataSample[i].Q_val;
```

4.2 VITA-T FORMAT

Recommendation: for VITA-T the highest-order bit in the VITA header (fixed to 0 in VITA-49) should be set to 1 to indicate that this packet is not VITA-49 compliant. Potentially, this causes an error if (non Tangerine) software attempts to use the VITA-T packet without the awareness that the subchannel data is different.

Recall that in the case of VITA-49, above, streaming multiple subchannels is handled by multiplexing the channels in packets, e.g., the data of channel 0 would be sent in packet#s 0, 2, 4, 6, etc., and the data of channel 1 would be sent in packet#s 1, 3, 5, etc.

In VITA-T, however, the subchannel IQ data samples are interleaved inside the packet; i.e., we have a single stream that includes all running subchannels for this channel. In this case, we would iterate over a buffer of data samples containing 2 subchannels by starting one counter at zero and a second counter at 1, and iterating by the number of subchannels (e.g.,, 2). The reason for this customization of VITA is to allow a buffer to be directly written into Digital RF without buffer reformatting, as DRF handles interleaved samples. The number of sample groups in a buffer is calculated as:

#samples = int (1024 / # subchannels)   (discard remainder)

While this results in different buffer sizes depending on the number of running subchannels, it avoids the issue of keeping track of sample groups that break across buffers. Again, this is optimized to be able to handle high speeds with a very small processor.

Note that the practice of assigning a VITA format to a channel allows us to concurrently run both VITA-49 and VITA-T channels, and this is indeed used when simultaneously running FT8 and Ringbuffer data, for example.

Building the VITA-T packet is the same as building the VITA-49 packet except that if the recommendation at the beginning of Section 4.2 is followed, the first byte of the packet will be 0x90 (i.e., the first bit of the packet is 1).

5. APPENDIX A: REFERENCES

The following table summarizes the documents referenced in this document.

Document Name	Description	Location
Tangerine SDR Requirements V0.3.pdf	*System requirements*	*https://tangerinesdr.com/TangerineSDR_documents/*
Local Host Functional Specification v1.0.2	*LH capabilities*	
LocalHost_Detailed_DesignSpec 0.2.docx	*LH detailed design*	
DE-LH protocol, V8	*Technical document*	
Red Rapids Technical White Paper REF-000-001-R00	*VITA-49 specificaion*	https://www.redrapids.com/images/whitepapers/TWP-000-001-R00.pdf

6. APPENDIX B: ERROR TABLE

When the Data Engine cannot carry out a requested command, it sends a negative acknowledgement ("NAK") followed by a code as follows:

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
7. Invalid slot# was passed in MW or MR command
8. Invalid interface was passed in MW or MR command
9. Invalid address was passed in MW or MR command
10. Invalid data provided in MW; does not conform to 0x0000 through 0xFFFF
11. Error when trying to write the data sent in MW command (may indicate sub-device malfunction)

APPENDIX C: COMMAND SUMMARY

7.

Mnemonics for commands and responses between LH and DE are as follows; for the latest version, refer to https://github.com/AB4EJ-1/TangerineSDR-pi/blob/master/includes/de_signals.h .

```
/*
de_signals.h
Maps program mnemonics to 2-byte commands to be passed to DE
*/
#define STATUS_INQUIRY       "S?"  // asks DE to send "AK"
#define DATARATE_INQUIRY     "R?"  // asks DE to send a table of supported data rates
#define DATARATE_RESPONSE    "DR"  // response: start of data rate table
#define LED1_ON              "Y1"  // asks DE to turn on LED1
#define LED1_OFF             "N1"  // asks DE to turn off LED1
#define REQUEST_TELEMETRY    "T?"  // asks DE to send telemtry packet
#define TELEMETRY_DATA       "TD"  // response: telemetry packet
#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-L mode (for Channel 0)
#define STOP_FIREHOSE        "XF"  // takes DR out of firehose-L 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 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
#define FT_DATA_BUFFER       "FT"  // this is an FT8 data packet
#define RG_DATA_BUFFER       "RG"  // this is a ringbuffer (or firehose) data packet
#define WS_DATA_BUFFER       "WS"  // this is a WSPR data buffer
#define STATUS_OK            "AK"  // last command was accepted
#define MEM_WRITE            "MW"  // write to sub-device memory
#define MEM_READ             "MR"  // read from sub-device memory
#define READ_RESPONSE        "RR"  // result of read of sub-device memory
```