250 Crosskeys Office Park
Suite 285
Fairport, New York 14450
Voice: (585) 419-9555 FAX: (585) 248-0537
http://www.vanteon.com

vImpulse

User Interface

Version 1.4b

October 17, 2017

Table of Contents

Approvals

Jeff Miller, Vanteon Account Manager — Date

Date

Date

Revision History

Revision	Date	Author	Description
1.0	9/13/2016	Gerry Callahan	Release
1.1	9/28/2016	Gerry Callahan	Release 1.1
1.1a	10/19/16	Joseph Ott	Release 1.1a
1.2	11/11/2016	Gerry Callahan	Additions, corrections
1.3	11/15/2016	Joseph Ott	Additions, updates
1.3a	3/14/2017	Gerry Callahan	Updates for FCC test
1.3b	6/1/2017	Gerry Callahan	Agency notices
1.3c	6/12/2017	Gerry Callahan	Added sync query
1.4	6/13/2017	Gerry Callahan	Add SyncWord command. Filled in some missing commands.
1.4a	9/19/2017	Gerry Callahan	Reformatted Antenna commands
1.4b	10/13/2017	Gerry Callahan	Added antenna list, and notification for Canada.

Authorship

Vanteon Corporation
250 CrossKeys Office Park, Suite 285
Fairport, New York 14450
Voice: (585) 419-9555
FAX: (585) 248-0537
http://www.vanteon.com

1. Agency Notices

1.1 FCC

Compliance Statement

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

1. *This device may not cause harmful interference, and*
2. *This device must accept any interference received, including interference that may cause undesired operation.*

Warning

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

FCC Interference Statement

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one of the following measures:

- *Reorient or relocate the receiving antenna.*
- *Increase the separation between the equipment and receiver.*
- *Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.*
- *Consult the dealer or an experienced radio/TV technician for help.*

Exposure

This portable transmitter with its antenna complies with FCC/IC RF exposure limits for general population / uncontrolled exposure.

1.2 Industry Canada

1.2.1 English

This Device complies with Industry Canada License-exempt RSS standard(s). Operation is subject to the following two conditions: 1) this device may not cause interference, and 2) this device must accept any interference, including interference that may cause undesired operation of the device.

Under Industry Canada regulations, this radio transmitter may only operate using an antenna of a type and maximum (or lesser) gain approved for the transmitter by Industry Canada. To reduce potential radio interference to other users, the antenna type and its gain should be so chosen that the equivalent isotropically radiated power (e.i.r.p.) is not more than that necessary for successful communication.

This radio transmitter (22785-IM1701B) has been approved by Industry Canada to operate with the antenna types listed below with the maximum permissible gain indicated. Antenna types not included in this list, having a gain greater than the maximum gain indicated for that type, are strictly prohibited for use with this device.

1.2.2 French

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes: 1) l'appareil ne doit pas produire de brouillage; 2) l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

Le présent émetteur radio (22785-IM1701B) a été approuvé par Industrie Canada pour fonctionner avec les types d'antenne énumérés ci-dessous et ayant un gain admissible maximal. Les types d'antenne non inclus dans cette liste, et dont le gain est supérieur au gain maximal indiqué, sont strictement interdits pour l'exploitation de l'émetteur.

1.3 Antenna

The vImpulse Radio must be used with the Linx ANT-916-MHW-RPS-S antenna, customized to replace the SMA connector with an MMCX male plug. The peak gain is 5.4 dBi.

2. Document Overview

2.1 Scope

This document describes the user interface for Vanteon's vImpulse Endpoint Radio. The console (UART) input and output, as well as LED indications, are defined.

2.2 Intended Audience

This document is of a technical nature. It is intended as a reference for customers using the vImpulse Endpoint Radio.

2.3 Definitions, Acronyms, and Abbreviations

- ASCII — American Standard Code for Information Interchange
- CTS — Clear To Send
- FCC — Federal Communications Commission
- IC — Industry Canada
- LED — Light Emitting Diode
- PA — Power Amplifier
- RF — Radio Frequency
- RSSI — Radio Signal Strength Indication
- RX — Receive
- TDMA — Time Division Multiple Access
- TX — Transmit
- UART — Universal Asynchronous Receiver/Transmitter

2.4 Conventions

- `Monospace font` indicates input to and output from the console. It is usually literal text (but see <> below).
- Square brackets ([text]) indicate optional text.
- Square brackets ([lower-upper]), where "lower" and "upper" are numbers, indicate an inclusive range (the limits are included within the range).
- Angle brackets (<name>) surrounding text indicate a named field, to be replaced by text as described in the definition.
- Braces and vertical bar ({ text | text }) indicate a choice between the text on the left and right of the bar.

3. Host Console Interface

3.1 UART

Connect the UART cable TXD, RXD, power, and ground, to the labeled pins. The serial port parameters are:

115.2 kbps, 8 bits, no parity, 1 stop bit, no handshaking

3.2 Command Summary

Commands are entered in ASCII format terminating with the CR (hex 0x0D). All commands are case insensitive.

When complete, commands respond with either "SUCCESS" or "FAILED". The most common cause of the "FAILED" response is a mistyped command. The response can also happen if a parameter is out of range, or if the command is not appropriate to the current configuration (for example, the `sendto` command will not work on Slave units).

3.3 Configuration

Show or change various configuration parameters. Some will not take effect until a restart. These commands are not persistent across a power cycle, unless the "`config save`" command is used. Multiple parameters may be used in one command, by simply adding parameters separated by spaces (for example, `config role=master save`). Do not use any spaces before or after an = sign. Parameters in italics require the advanced mode to be set (see `config advanced`).

Syntax: `config [show | save | member=# | family=# | role=<role> | SyncWord=<word> | SyncLength=# | txpower=<setting> | *txlow=<setting>* | *txmed=<setting>* | *txhigh=<setting>* | packet_size=# | poll=# | echo=# | antenna=<port> | powerpolicy=<setting>] | advanced=<setting> | admin=<setting>`

`config [show]`

List the configuration parameters. This is the default action if no parameters are specified.

Example:

```
*config
   member            = 1
   family            = 3
   role              = master
   Sync Word         = 00004356
   Sync Length       = 4
   packet_size       = 64
   poll              = 0
   echo              = 1
   TxPower           = 20 (High)
   Antenna           = J6
   PowerPolicy       = Normal

SUCCESS
```

NOTE: Configuration options displayed may differ based on configured role and version.

`config save`

Write the configuration parameters to flash storage.

`config member=[0-254]`

Set the member ID of the device. Must be unique for the family it is in. The default is 1.

`config family=[0-254]`

Set the Family ID of the device. Devices must have the same Family ID to communicate. The default is 0.

config role={master | slave | inactive}

Set the role of the device. Each family must have one and only one master. A family can have multiple slaves. Inactive allows the device to boot but it will not communicate as a radio (the console continues to function as normal). The factory default is Inactive. Changing to Master or Slave requires restart before taking full effect.

config SyncID=<word>

Set the Sync ID, 1 to 8 hexadecimal digits. The Sync ID (along with SyncLength and Family ID) is used to generate a Sync Word, which is transmitted and received at the beginning of every packet. The Sync ID is also used for determining the hop table. Devices must have the same Sync ID to communicate. The Sync Word is transmitted starting with the LSB, so if the Sync Word length (see next command) is set shorter than the Sync ID, then only the right-most bytes will be used. The default is 0x00000000.

config SyncLength=[1-4]

Set the Sync Word length, which is the number of bytes of the Sync Word to be transmitted or received at the beginning of every packet. Devices must have the same Sync Word length to communicate. The default is 4.

config TxPower={off | low | med | high | <level>}

Set the transmit power. <level > is a number between 0 and 127, where higher numbers indicate more transmit power. The default is Low.

config {txlow|txmed|txhigh}=[0-127]

Set the value that will be used for the `config txpower=low, med,` and `high` commands. The new value will be in effect the next time `TxPower` is set to the respective level.

config packet_size={32|64}

Sets the packet size which determines the payload size. A packet contains a 4-byte header plus the user payload. The supported packet sizes are 32 and 64 giving a respective payload size of 28 and 60. Changing the packet size requires restart before taking full effect.

config poll={0|1}

Set whether the host will poll for received packets. 0 indicates that packets will be output to the serial interface upon arrival. 1 indicates that the packets will remain in the receive buffer until read. If the buffer is full and another packet is received, the new packet is discarded. The default is 1.

config echo={0|1}

Set whether characters are echoed to the serial console. 0 disables echo; 1 enables it. The default is 1.

config antenna={1|2|J6|J7}

Set which antenna port is used. J6 and J7 refer to the hardware connectors on the module. Antenna 1 is on port J6, and antenna 2 is on J7.

config PowerPolicy={full | normal | medium | low}

Set the power policy, which determines how power consumption is managed. Full power will keep all components powered on continuously. Normal will save some power by turning off some components when they are not needed, while maintaining full functionality. Medium and Low will save even more power, although some functionality (such as transmit power) will be limited. The default is Normal.

config RssiThres=[1-255]

RSSI threshold for Clear Channel Assessment. Before transmitting, a slave will measure RSSI to see if the channel is clear. The measured RSSI must be lower than this threshold for the slave to determine that the channel is clear so it can transmit. If RSSI is greater than this threshold, it will wait to try transmitting. The default is 64.

config advanced={enable | disable}

Enable or disable certain configure commands (shown in italics above). The default is disable. This setting is not persistent.

config admin={admin | user}

Enable or disable showing more commands in the `help` display. The default is user. This setting is not persistent.

3.4 Help

Print list of commands

Syntax: `help` or `?`

3.5 Hopping

This command displays the hop table in use. The command may be shortened to `hop`.

Syntax: `hoptable`

`hoptable`

Display the hop table (ID and channel numbers) currently in use. Both are printed in hexadecimal.

3.6 Restart

`restart`

Reboot the device.

Syntax: `restart`

3.7 Receiving Packets

`payloadsize`

Return the number of bytes in the user payload.

Syntax: `payloadsize`

`buffersize`

Return the usable size of the TX and RX buffers.

Syntax: `buffersize`

`rxnumpackets`

Return the number of packets in the receive buffer. Packets can then be read using the `rxpackets` command. This is useful when configured for polling mode.

Syntax: `rxnumpackets`

`rxpackets`

Return oldest packet in the receive buffer and then remove that packet from the receive buffer. If the Receive buffer is empty, a null packet is printed. A null packet is simply a packet of 0 length and no payload. If <npackets> is supplied, that many payloads are returned, separated with a carriage return, and are removed from the receive buffer. If <npackets> is more than the actual number of packets in the buffer, or if the parameter is “all”, then all packets will be printed, and a null packet is printed at the end of the list.

The source and destination are printed in hexadecimal, with a comma after each. The payload is returned prepended with its length in decimal. A comma separates the length and payload bytes, which are in hexadecimal. A null packet is printed as “00,”.

The receive buffer can contain up to 31 packets. Use the `buffersize` command (above) to find the actual capacity based on the current configuration.

Syntax: `rxpackets [<npackets> | all]`

Example:

```
*rxpackets 2
69,08,09,0000000200002f0df4
69,08,09,0000000300003101f4
|  |  |  |_______________ Payload
|  |  |__________________ Payload length
|  |_____________________ Destination ID
|________________________ Source ID (0x69 = 105 decimal)
```

`rxpayload`

Return oldest payload in the receive buffer and then remove that packet from the receive buffer. If the Receive buffer is empty, a null payload is printed. A null payload is simply a packet of 0 length and no payload. If <npackets> is supplied, that many payloads are returned, separated with a carriage return, and are removed from the receive buffer. If <npackets> is more than the actual number of packets in the buffer, or if the parameter is "all", then all packets are printed and removed, and a null packet is printed at the end of the list.

The payload is returned prepended with its length in decimal. A comma separates the length and payload bytes, which are in hexadecimal. A null packet is printed as "00,".

Syntax: `rxpayload [<npackets> | all]`

Example:

```
*rxpayload 2
09,0000000200002f0df4
09,0000000300003101f4
|  |_____________________ Payload
|________________________ Payload length
```

3.8 Sending Packets

`cts`

Clear To Send. Return 1 if there is room in the Tx buffer for at least one packet; 0 if the buffer is full.

Syntax: `cts`

`ping`

Transmit a predefined payload, with an option to send a number of payloads at a desired interval. If the ping command is entered with no arguments when no previous ping command is running, a single packet is transmitted. If the ping command is entered with no arguments when a previous ping task is running, the status of the running ping command is printed. To automatically send more than one packet, set the first argument to the number of packets desired, or "continuous" to run forever (or until manually stopped). To end a running ping command, enter the `stop` command. If a number or "continuous" is supplied for the first argument, then an interval (in milliseconds) may be added to specify the average time between ping transmissions; if omitted, the default is 105 ms (one per slot).

Syntax: `ping [continuous|<count> [<interval>]]`

Examples:
Send 100 packets, at a rate of 20 packets per second:

```
*ping 100 50
      |   |___________ 50 milliseconds (1 second divided by 20)
      |_______________ 100 packets
```

send

Transmit the specified payload. If the device is a Master the payload is broadcast to all slaves sharing the same family ID. If the device is a Slave the payload is sent to its associated Master sharing the same family ID. The payload is specified in hexadecimal. The payload is specified as a single string containing 2 characters per payload byte without spaces. If the payload consists of an odd number of characters, the last character will be ignored. To check how many bytes can be in the payload, use the `payloadsize` command (section 3.7).

If the unit role is Inactive, or Slave and the unit is not sync'd, then the payload is added to the Tx buffer, to be transmitted when the unit gains sync. If the buffer is full, then this command will return "FAILED", and the buffer is unchanged. To check the buffer size, use the `buffersize` command (section 3.7).

Syntax: `send [x1x2…]`

Examples:
Send 3 bytes

```
*send A0B1C2
      | | |___________________ Byte 2 (0xC2)
      | |_____________________ Byte 1 (0xB1)
      |_______________________ Byte 0 (0xA0)
```

NOTE: The syntax for this command changed for version 1.3. The syntax for earlier versions is:

Syntax: `send [x1 [x2 […]]]`

sendto

Transmit the specified payload to the specified destination ID. This can only be used by a Master to send a payload to a specified slave ID sharing the same family ID. A destination ID of 255 broadcasts to all devices sharing the same family ID. The ID is specified in decimal, and the payload is hexadecimal. The payload is specified as a single string containing 2 characters per payload byte without spaces. If the payload consists of an odd number of characters, the last character will be ignored. To check how many bytes can be in the payload, use the `payloadsize` command (section 3.7).

Syntax: `sendto id [x1x2…]`

Examples:
Send 4 bytes to id 123

```
*sendto 123 A0B1C2D3
            | | | |___________________ Byte 3 (0xD3)
            | | |_____________________ Byte 2 (0xC2)
            | |_______________________ Byte 1 (0xB1)
            |_________________________ Byte 0 (0xA0)
```

sendwhite

Transmit "white" (random) data continuously. Normal data packets will be filled with random data, and transmitted as fast as possible under the current configuration. To end the continuous transmission, enter the `stop` command.

stop

Discontinue any continuous transmission, such as from the `sendwhite` command.

3.9 Status

status

Return status counters for packets received and transmitted, error conditions, RSSI, etc.

Syntax: `status`

Example:

```
*status
System Status
Elapsed Time: 0:08:26.505

Temperature          37.10
RX packets rcvd      2
TX packets sent      1

RSSI Current        35
RSSI last preamble  0
RSSI last valid pkt 0
RSSI Alternate ant  0

System Status       0x00000000
```

Explanation:

Elapsed Time shows the time elapsed, same as the `time` command.

Temperature is the temperature of the transceiver chip, in degrees Celsius.

RX packets rcvd and TX packets sent show the number of packets received and transmitted, respectively, by this unit. Note: the number of Tx packets may include some packets in the transmit buffer that have not yet been actually transmitted.

RSSI Current is the most recent Receive Signal Strength Indicator measurement. "RSSI last preamble" is the RSSI measurement on the most recent preamble detected, whether the preamble was for a packet addressed to this unit or not. "RSSI last valid pkt" is the RSSI measurement on the most recent preamble for a packet that was addressed to this unit. "RSSI Alternate ant" is the RSSI on the alternate antenna and will only appear if diversity is active.

System Status is a bit field of various indicators. Not all bits are used. The following bits are defined:

0x00000001	TX packet buffer overrun
0x00000002	RX packet buffer overrun
0x00000080	Slave in search mode
0x00010000	RX packet buffer not empty
0x00040000	Antenna connected to J7 in use (if cleared J6 is in use)

sync?

Return a string indicating the Slave unit synchronization state, either “Synchronized” or “Searching”. This command is not valid for Master or Inactive units.

Syntax: `sync?`

Example:

```
*sync?
Synchronized
```

3.10 System Time

Display the time elapsed since the unit was powered on. The time is formatted as Hours:Minutes:Seconds with Seconds including 3 digits after the decimal.

Syntax: `status`

Example:

```
*time
0:06:13.748
```

3.11 Version

Get firmware revision, hardware revision, etc.

Syntax: `version`

3.12 Comment

The unit will ignore the text. This is useful when the input is a script, or the session is being logged.

Syntax: `#[text]`

where **#** is literally the “hash” or “pound” character.

4. vImpulse LEDs

4.1 LED Usage

The vImpulse board contains two tri-color LEDs, labeled as LED1 and LED2. The following table describes the meaning of each LED in the various states and conditions.

Role:	Master		Slave	
State/Condition	LED1	LED2	LED1	LED2
Searching	N/A	N/A	Red (Long blink)	
Synchronized	Green (quick)		Green (quick)	
Programming	Red		Red	
Transmitting		Blue		Blue
Tx Buffer Full		Red		Red
Rx Buffer not empty		Green		Green

Blank cells indicate that the LED is off in that state, or that it is controlled by another condition. The Master is always assumed to be synchronized, so LED1 will always blink green. If a radio is set to the Inactive role, then LED1 will slowly blink red.

If a radio has packets in the receive buffer, LED2 will be green. For radios with polling enabled, LED2 will stay on until the last packet is read out of the buffer. When polling is disabled, packets are automatically sent to the host, thereby removing them from the buffer, so the LED blinks.