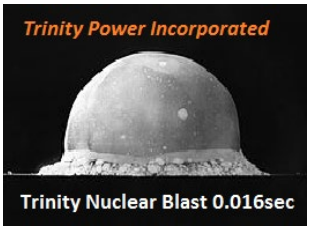

Trinity Power, Inc.

(512) 992-8657

Application Note AN-2 (rev 1.18)

for the

TPI Signal Generator

covering

User Command Structure

for TPI-1001, TPI-1002, & TPI-1005

unit firmware version 1.062

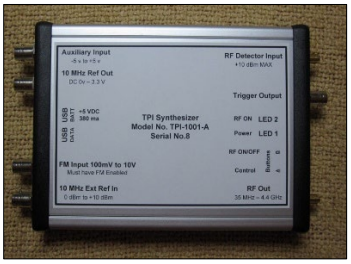

Table of Contents

1 Introduction

This application note is a supplement to the TPI-1001, TPI-1002, and TPI-1005 signal generator user manual and describes the user accessible command structure. With this information, users may create their own user interface to communicate with the signal generator instead of using the supplied graphical user interface (GUI) TPI-Link. Note that some features of the signal generator are only available via TPI-Link. Unit firmware version 1.017 is the first version of firmware that supports most of these commands.

1.1 USB Interface

Most units have two USB ports. One of the ports on each unit is only used for supplemental power from a second power source such as a battery. The other USB port is used for both powering the unit and communication via the USB port of a computer. The TPI-1001 communication port is labeled "USB Data", and the TPI-1002 communication port is labeled "USB".

Internally the communication USB port is connected to an FTDI chip that acts as a USB to UART translator. Therefore, the user's program accesses the unit as if it was a COM port. The driver installed in the computer maps the computer's USB port to a virtual COM port. TPI-Link can be used to identify the COM port number in use. This can also be seen using the Windows Device Manager.

The user's program must set the COM port parameters to the following…

- Baud rate: 3,000,000 (3 Mbit/sec)
- Data bits: 8
- Stop bits: 1
- Parity: None
- Handshake: Request to send

1.2 Packet Structure

Throughout this document byte values will be represented in hexadecimal notation with "0x" preceding the value unless otherwise noted as a decimal value. For example, the decimal value 123 is equivalent to 0x7B.

All communications to or from the unit are in the form of a defined packet with a header, a body, and a checksum. The header consists of four bytes…

- 1st byte: 0xAA (decimal 170) — First qualifier byte
- 2nd byte: 0x55 (decimal 85) — Second qualifier byte
- 3rd byte: High order byte of 16 bit integer representing number of bytes in body
- 4th byte: Low order byte of 16 bit integer representing number of bytes in body

The body of the packet consists of a minimum of two bytes. See the next section for a detailed list of the various commands available.

The body is followed by a one-byte checksum that is the ones' complement of the 8-bit sum of all preceding bytes in the packet excluding the two qualifier bytes. In other words, it is 0xFF minus the sum of all bytes beginning with the 3rd byte of the header through the last byte of the body. For example, if the body of a packet sent to the unit consisted of the two bytes 0x07, 0x02, the entire packet would be (header, body, checksum)…

0xAA, 0x55, 0x00, 0x02, 0x07, 0x02, 0xF4

1.3 Packet Framing

The two qualifier bytes (0xAA, 0x55) provide a means of identifying the beginning of a packet. In the event that the user's program receives an incorrect checksum or somehow gets out of sync with the expected number of bytes in the body of a packet, the program should look for the two-byte qualifier sequence to identify the beginning of the next packet. Note that it is possible that the same two-byte sequence may be contained within the body of a packet. Therefore, once the qualifier sequence has been received, the program must not interpret any subsequent identical two-byte sequences as packet qualifiers until the expected number of bytes has been received.

1.4 Traffic Control

When the unit receives a packet, it will interpret the packet and execute the command as quickly as possible. The unit will then reply with the data requested or a simple acknowledgement. If the user sends another packet before a reply is received from the prior packet, the unit will buffer the new packet for processing after the prior one is finished. The receive buffer is several hundred bytes, which can permit many packets to be sent before the first one is fully processed. However, it is recommended that the user's program does not send another packet until a reply has been received from the previous packet. This will keep buffers from overflowing and makes program debugging much easier.

2 Commands

This section describes the various command packets. Only the body of each packet is shown. It must be understood that each of the following is preceded by the four-byte header and followed by the one-byte checksum described in section 1.2.

The first byte of the body indicates whether the command is a read or write command. A first byte of 0x07 indicates a read command, and 0x08 indicates a write command.

2.1 User Control

User control must first be enabled. Send the following once.

0x08, 0x01

The full packet including the header and checksum is…
0xAA, 0x55, 0x00, 0x02, 0x08, 0x01, 0xF4
This checksum was calculated as follows…
0x00 + 0x02 + 0x08 + 0x01 = 0x0B
The ones' complement of 0x0B is 0xF4

The unit will respond with the same packet…

0x08, 0x01

User control is now enabled. The unit is ready to respond to any of the following commands. The user control state may be read at any time by sending the following bytes…

0x07, 0x01

The unit will respond with three bytes…

0x07, 0x01, n

…where n is 0x01 if user control is enabled and 0x00 if disabled.

2.2 Read Model Number

To read the model number of the unit, send…

0x07, 0x02

The unit will respond with 18 bytes…

0x07, 0x02, $n_0 \ldots n_{15}$

…where $n_0 \ldots n_{15}$ is a 16-byte ASCII string (not null terminated) representing the model number of the unit.

2.3 Read Serial Number

To read the serial number of the unit, send…

0x07, 0x03

The unit will respond with 18 bytes…

0x07, 0x03, $n_0 \ldots n_{15}$

…where $n_0 \ldots n_{15}$ is a 16-byte ASCII string (not null terminated) representing the serial number of the unit.

2.4 Read Hardware Version

To read the hardware version of the unit, send…

0x07, 0x04

The unit will respond with 18 bytes…

0x07, 0x04, $n_0 \ldots n_{15}$

…where $n_0 \ldots n_{15}$ is a 16-byte ASCII string (not null terminated) representing the hardware version of the unit.

2.5 Read Unit Firmware Version

To read the firmware version of the unit, send…

0x07, 0x05

The unit will respond with 18 bytes…

0x07, 0x05, $n_0 \ldots n_{15}$

…where $n_0 \ldots n_{15}$ is a 16-byte ASCII string (not null terminated) representing the firmware version of the unit.

2.6 Read Minimum TPI-Link Version Required

To read the minimum version of TPI-Link that is compatible with the firmware in the unit, send…

0x07, 0x06

The unit will respond with 10 bytes…

0x07, 0x06, n_0…n_7

…where…

n_0…n_1 is a 16-bit integer (LSB first) = major
n_2…n_3 is a 16-bit integer (LSB first) = minor
n_4…n_5 is a 16-bit integer (LSB first) = build
n_6…n_7 is a 16-bit integer (LSB first) = revision

The entire required GUI version is represented by "major.minor.build.revision". TPI-Link indicates its version by displaying only "major.minor" with minor represented as a three digit value. Therefore, a major value of "1" and a minor value of "23" will be displayed in TPI-Link as version 1.023.

Note: TPI-Link reads this information and compares it to the running version of TPI-Link to determine if the user should be warned about TPI-Link being incompatible with the current unit firmware. A custom user interface using the commands in this document should not be concerned about this command. Of greater concern to a custom user interface is if the user command structure has changed in any way. Any revisions to the user command structure in future versions of the unit firmware will be noted in revisions to this document.

2.7 Read Supply Voltages

To read the various supply voltages in the unit, send…

0x07, 0x07

The unit will respond with 26 bytes…

0x07, 0x07, n_0…n_{23}

…where…

n_0…n_3 is a 32-bit floating point value (IEEE-754, LSB first) = RF supply
n_4…n_7 is a 32-bit floating point value (IEEE-754, LSB first) = VCO supply
n_8…n_{11} is a 32-bit floating point value (IEEE-754, LSB first) = MCU supply
n_{12}…n_{15} is a 32-bit floating point value (IEEE-754, LSB first) = OSC supply
n_{16}…n_{19} is a 32-bit floating point value (IEEE-754, LSB first) = PA supply
n_{20}…n_{23} is a 32-bit floating point value (IEEE-754, LSB first) = DET supply*

*DET supply (TPI-1001); USB supply (TPI-1002)

2.8 Current Unit State

To read the current operational state of the unit, send…

0x07, 0x08

The unit will respond with 4 bytes…

0x07, 0x08, n_0, n_1

…where…

n_0	Unit State	n_1
0x00	Generator	Ignore value of n_1
0x01	Modulation: square wave running	n_1>0 means RF output auto on/off
0x02	Modulation: beacon running	
0x03	Modulation: script running	
0x04	Scanning	Read n_1: 1=scanning; 2=paused Write n_1: 0=stop; 1=scan; 2=pause
0x05	Control script running	Read n_1: 1=script waiting Write n_1: 0=stop; 1=start; 2=continue

To set the current operational state of the unit, send…

0x08, 0x08, n_0, n_1

The unit will respond with 2 bytes…

0x08, 0x08

It is your responsibility to ensure that any state other than generator is terminated before initiating a new state. For example, if the unit is scanning (n_0 = 4), you must first stop the scanning by sending 0x08, 0x08, 0x04, 0x00 before starting another state, such as modulation or control script. If not, unpredicted behavior will result. TPI-Link enforces this requirement.

2.9 Frequency

To read the current frequency, send…

0x07, 0x09

The unit will respond with 6 bytes…

0x07, 0x09, n_0…n_3

…where n_0…n_3 is a 32-bit integer (LSB first) representing the frequency in kHz.

To set the frequency, send…

0x08, 0x09, n_0…n_3

…where n_0…n_3 is a 32-bit integer (LSB first) representing the frequency in kHz. The value must be between 35000 and 4400000 inclusive.

The unit will respond with 2 bytes…

0x08, 0x09

2.10 RF Output Level

To read the current output level, send…

0x07, 0x0A

The unit will respond with 3 bytes…

0x07, 0x0A, n

…where n is a signed byte representing the output level in dBm.

To set the level, send…

0x08, 0x0A, n

…where n is a signed byte representing the desired output level in dBm. If the requested output level is outside the available range, the output will be set to the maximum or minimum as appropriate.

The unit will respond with 3 bytes…

0x08, 0x0A, n

…where n is a signed byte representing the new output level in dBm.

2.11 RF Output On/Off

To read the current state of the output, send…

0x07, 0x0B

The unit will respond with 3 bytes…

0x07, 0x0B, n

…where n = 0 when the output is off, and n = 1 when the output is on.

To control the output, send…

0x08, 0x0B, n

…where n = 0 to turn the output off, and n = 1 to turn the output on.

The unit will respond with 2 bytes…

0x08, 0x0B

2.12 Read Detector Level (TPI-1001 & TPI-1005 only)

To read the detector level, send…

0x07, 0x0C

The unit will respond with 7 bytes…

0x07, 0x0C, n_0…n_4

…where…

n_0…n_3 is a 32-bit floating point value (IEEE-754, LSB first) = input level in dBm [Note 2]

n_4 = 0 means input level is within detection range

n_4 = 1 means input level is >= maximum detection level

n_4 = 2 means input level is <= minimum detection level

Notes:

[1] If this command is sent to a TPI-1002, an error code will be triggered because the TPI-1002 does not have a detector input.

[2] If the detector is not turned on, the returned value must be disregarded.

[3] The detector is a broadband detector and so indicates the total RF power present across the spectrum. The detector's output has been calibrated at various frequencies and levels. Therefore, for highest accuracy, the unit needs to know the frequency of the signal presented to the RF detector input. It assumes that the frequency is the same as the frequency generated for the RF output. This feature accommodates the analyzer in TPI-Link. Therefore, if you are using the detector input as a generic RF power meter, it is important to set the RF output frequency to that of the frequency being measured for highest accuracy of the measurement.

2.13 Detector On/Off (TPI-1001 & TPI-1005 only)

To read the current state of the detector, send…

0x07, 0x0D

The unit will respond with 3 bytes…

0x07, 0x0D, n

…where n = 0 when the detector is off, and n = 1 when the detector is on.

To control the detector, send…

0x08, 0x0D, n

…where n = 0 to turn the detector off, and n = 1 to turn the detector on.

The unit will respond with 2 bytes…

0x08, 0x0D

2.14 Reference Source

To read the current reference source, send…

0x07, 0x0E

The unit will respond with 3 bytes…

0x07, 0x0E, n

…where n = 0 when the reference source is internal, and n = 1 when the reference source is external.

To select the reference source, send…

0x08, 0x0E, n

…where n = 0 to select the internal reference, and n = 1 to select the external reference.

The unit will respond with 2 bytes…

0x08, 0x0E

2.15 Reference Output On/Off

To read the current state of the reference output, send…

0x07, 0x0F

The unit will respond with 3 bytes…

0x07, 0x0F, n

…where n = 0 when the reference output is off, and n = 1 when the reference output is on.

To control the reference output, send…

0x08, 0x0F, n

…where n = 0 to turn the reference output off, and n = 1 to turn the reference output on.

The unit will respond with 2 bytes…

0x08, 0x0F

2.16 Read Auxiliary Input Voltage (TPI-1001 & TPI-1005 only)

To read the auxiliary input voltage, send…

0x07, 0x10

The unit will respond with 6 bytes…

0x07, 0x10, n_0…n_3

…where…

n_0…n_3 is a 32-bit floating point value (IEEE-754, LSB first) = auxiliary input in volts [Notes 2, 3]

Notes:

[1] If this command is sent to a TPI-1002, an error code will be triggered because the TPI-1002 does not have an auxiliary input.

[2] The auxiliary input is internally biased to approximately 0.5 volts. Therefore, if the input is not connected to anything, you will read approximately 0.5 volts with this command.

[3] The range of the auxiliary input is −5 volts to +5 volts.

2.17 Auxiliary Input Threshold (TPI-1001 & TPI-1005 only)

To read the auxiliary input threshold, send…

0x07, 0x11

The unit will respond with 6 bytes…

0x07, 0x11, n_0…n_3

…where…

n_0…n_3 is a 32-bit floating point value (IEEE-754, LSB first) = the auxiliary input threshold in volts
Note 2

To set the auxiliary input threshold, send…

0x08, 0x11, n_0…n_3

…where…

n_0…n_3 is a 32-bit floating point value (IEEE-754, LSB first) = the auxiliary input threshold in volts
Notes 1, 2

The unit will respond with 2 bytes…

0x08, 0x11

Notes:

[1] If this command is sent to a TPI-1002, an error code will be triggered because the TPI-1002 does not have an auxiliary input.

[2] The auxiliary input threshold has a range of −4 volts to +4 volts.

2.18 Auxiliary Input Greater/Less Than (TPI-1001 & TPI-1005 only)

To read the auxiliary input greater/less than, send…

0x07, 0x12

The unit will respond with 3 bytes…

0x07, 0x12, n

…where n = 0 means the auxiliary input action will occur when the input falls below the threshold, and n = 1 means the auxiliary input action will occur when the input rises above the threshold.

To set the auxiliary input greater/less than, send…

0x08, 0x12, n

…where n = 0 means you wish the auxiliary input action to occur when the input falls below the threshold, and n = 1 means you wish the auxiliary input action to occur when the input rises above the threshold.

The unit will respond with 2 bytes…

0x08, 0x12

2.19 Auxiliary Input Action (TPI-1001 & TPI-1005 only)

To read the current auxiliary input action, send…

0x07, 0x13

The unit will respond with 3 bytes…

0x07, 0x13, n

…where…

n	Auxiliary Input Action
0x00	None
0x01	RF out on while active
0x02	Turn RF out on
0x03	Turn RF out off
0x04	Increment frequency
0x05	Decrement frequency
0x06	Increment level
0x07	Decrement level
0x08	Mod: Square wave source
0x09	Mod: Square wave active
0x0A	Mod: Square wave start
0x0B	Mod: Square wave stop
0x0C	Mod: Beacon active
0x0D	Mod: Beacon start
0x0E	Mod: Beacon stop
0x0F	Mod: Script active
0x10	Mod: Script start
0x11	Mod: Script stop
0x12	Scan while active
0x13	Start scan
0x14	Stop scan
0x15	Pause scan
0x16	Script run active
0x17	Script start
0x18	Script stop
0x19	Script continue
0x1A	Auxiliary input controls level
0x1B	Auxiliary input controls freq
0x1C	Start analyzer
0x1D	Next frequency in list

To set the auxiliary input action, send…

0x08, 0x13, n

…where n equals the desired action from the list. The unit will respond with 2 bytes…

0x08, 0x13

2.20 Trigger Out Source (TPI-1001 & TPI-1005 only)

To read the current trigger out source, send…

0x07, 0x14

The unit will respond with 3 bytes…

0x07, 0x14, n

…where…

n	Trigger Out Source
0x00	None
0x01	Synth Lock Signal (MUXOUT)
0x02	RF Output State
0x03	Aux In > Threshold
0x04	Frequency = Match
0x05	Frequency > Match
0x06	Active while scanning (not paused or inter-scan)
0x07	Active while square wave "on"
0x08	Active while beacon "on"
0x09	Active while modulation script "on"
0x0A	Active during modulation script prefix
0x0B	Active during modulation script data
0x0C	Active during modulation script suffix
0x0D	Script controls trigger
0x0E	Detector level > level threshold
0x0F	Pulse at scan frequency step
0x10	Pulse at scan level step

See the user manual for a description of each source.

To set the trigger out source, send…

0x08, 0x14, n

…where n equals the desired source from the list.

The unit will respond with 2 bytes…

0x08, 0x14

2.21 Invert Trigger Out (TPI-1001 & TPI-1005 only)

To read the invert trigger out state, send…

0x07, 0x15

The unit will respond with 3 bytes…

0x07, 0x15, n

…where n = 0 means the trigger out state is not inverted, and n = 1 means the trigger out state is inverted.

To set the invert trigger out state, send…

0x08, 0x15, n

…where n = 0 means you do not wish to invert the trigger out state, and n = 1 means you wish to invert the trigger out state.

The unit will respond with 2 bytes…

0x08, 0x15

2.22 Frequency Presets

To read the twelve preset frequencies, send…

0x07, 0x16

The unit will respond with 50 bytes…

0x07, 0x16, n_0…n_{47}

…where…

n_0…n_3 is a 32-bit integer (LSB first) representing preset 1 frequency in kHz
n_4…n_7 is a 32-bit integer (LSB first) representing preset 2 frequency in kHz
n_8…n_{11} is a 32-bit integer (LSB first) representing preset 3 frequency in kHz
n_{12}…n_{15} is a 32-bit integer (LSB first) representing preset 4 frequency in kHz
n_{16}…n_{19} is a 32-bit integer (LSB first) representing preset 5 frequency in kHz
n_{20}…n_{23} is a 32-bit integer (LSB first) representing preset 6 frequency in kHz
n_{24}…n_{27} is a 32-bit integer (LSB first) representing preset 7 frequency in kHz
n_{28}…n_{31} is a 32-bit integer (LSB first) representing preset 8 frequency in kHz
n_{32}…n_{35} is a 32-bit integer (LSB first) representing preset 9 frequency in kHz
n_{36}…n_{39} is a 32-bit integer (LSB first) representing preset 10 frequency in kHz
n_{40}…n_{43} is a 32-bit integer (LSB first) representing preset 11 frequency in kHz
n_{44}…n_{47} is a 32-bit integer (LSB first) representing preset 12 frequency in kHz

To set the twelve preset frequencies, send…

0x08, 0x16, n_0…n_{47}

…where n_0…n_{47} are the twelve preset frequencies as outlined above.

The unit will respond with 2 bytes…

0x08, 0x16

2.23 Level Presets

To read the twelve preset levels, send…

0x07, 0x17

The unit will respond with 14 bytes…

0x07, 0x17, n_0…n_{11}

…where…

n_0 is a signed byte representing preset 1 level in dBm
n_1 is a signed byte representing preset 2 level in dBm
n_2 is a signed byte representing preset 3 level in dBm
n_3 is a signed byte representing preset 4 level in dBm
n_4 is a signed byte representing preset 5 level in dBm
n_5 is a signed byte representing preset 6 level in dBm
n_6 is a signed byte representing preset 7 level in dBm
n_7 is a signed byte representing preset 8 level in dBm
n_8 is a signed byte representing preset 9 level in dBm
n_9 is a signed byte representing preset 10 level in dBm
n_{10} is a signed byte representing preset 11 level in dBm
n_{11} is a signed byte representing preset 12 level in dBm

To set the twelve preset levels, send…

0x08, 0x17, n_0…n_{11}

…where n_0…n_{11} are the twelve preset levels as outlined above.

The unit will respond with 2 bytes…

0x08, 0x17

2.24 Beep

If a beep command is executed while a command script is running, the unit will send…

0x07, 0x18

This command is never requested by the user. When TPI-Link receives this command, it produces a short beep on the computer's audio output.

2.25 Control Button Function

To read the current control button function, send…

0x07, 0x19

The unit will respond with 3 bytes…

0x07, 0x19, n

…where…

n	Control Button Function
0x00	None
0x01	Square wave modulation
0x02	Beacon modulation
0x03	Modulation script
0x04	Scan
0x05	Control Script

See the user manual for a description of each function.

To set the control button function, send…

0x08, 0x19, n

…where n equals the desired function from the list.

The unit will respond with 2 bytes…

0x08, 0x19

2.26 Mute Till Lock Detect

To read the mute till lock detect (MTLD) state, send…

0x07, 0x1A

The unit will respond with 3 bytes…

0x07, 0x1A, n

…where n = 0 means MTLD is disabled, and n = 1 means MTLD is enabled.

To set the MTLD state, send…

0x08, 0x1A, n

…where n = 0 means you wish disable MTLD, and n = 1 means you wish enable MTLD.

The unit will respond with 2 bytes…

0x08, 0x1A

Note: MTLD should be disabled when using the FM input on the unit.

2.27 Frequency Increment

To read the current frequency increment, send…

0x07, 0x1B

The unit will respond with 6 bytes…

0x07, 0x1B, n_0…n_3

…where n_0…n_3 is a 32-bit integer (LSB first) representing the frequency increment in kHz.

To set the frequency increment, send…

0x08, 0x1B, n_0…n_3

…where n_0…n_3 is a 32-bit integer (LSB first) representing the frequency increment in kHz. The value must be between 1 and 4365000 inclusive.

The unit will respond with 2 bytes…

0x08, 0x1B

2.28 Level Increment

To read the current level increment, send…

0x07, 0x1C

The unit will respond with 3 bytes…

0x07, 0x1C, n

…where n is an unsigned byte representing the level increment in dBm.

To set the level increment, send…

0x08, 0x1C, n

…where n is an unsigned byte representing the desired level increment in dBm.

The unit will respond with 3 bytes…

0x08, 0x1C, n

Note: The level increment is always a positive value from 1 to 120 inclusive.

2.29 Match Frequency

To read the current match frequency, send…

0x07, 0x1D

The unit will respond with 6 bytes…

0x07, 0x1D, n_0…n_3

…where n_0…n_3 is a 32-bit integer (LSB first) representing the match frequency in kHz.

To set the match frequency, send…

0x08, 0x1D, n_0…n_3

…where n_0…n_3 is a 32-bit integer (LSB first) representing the match frequency in kHz. The value must be between 35000 and 4400000 inclusive.

The unit will respond with 2 bytes…

0x08, 0x1D

2.30 Select Frequency Preset

To select a frequency preset, send…

0x08, 0x1E, n

…where n is a value from 1 to 12 inclusive representing the desired preset.

The unit will respond with 6 bytes…

0x08, 0x1E, n_0…n_3

…where n_0…n_3 is a 32-bit integer (LSB first) representing the new frequency in kHz. If the value of n sent to the unit is not in the range of 1…12, the returned value of n_0…n_3 will be zero.

2.31 Select Level Preset

To select a level preset, send…

0x08, 0x1F, n

…where n is a value from 1 to 12 inclusive representing the desired preset.

The unit will respond with 3 bytes…

0x08, 0x1F, n

…where n is a signed byte representing the output level in dBm. If the value of n sent to the unit is not in the range of 1…12, the returned value of n will be 100 (decimal).

2.32 Save Current Sate as Default

To save the current state of the unit as the default, send…

0x08, 0x20

All the current parameter values will be saved in nonvolatile memory for use on power up. The current state of the unit will be saved, e.g., if the unit was scanning, it will begin scanning on power up.

The unit will respond with 3 bytes…

0x08, 0x20, n

…where n is zero if everything was saved with no errors, and one if there were any errors writing to nonvolatile memory.

2.33 Increment Frequency

To increment the frequency, send…

0x08, 0x21, n

…where n = 0 to increase the frequency by the frequency increment, and n = 1 to decrease the frequency by the frequency increment.

The unit will respond with 6 bytes…

0x08, 0x21, n_0…n_3

…where n_0…n_3 is a 32-bit integer (LSB first) representing the new frequency in kHz. If the value of n sent to the unit is not in the range of 0…1, the frequency will not be changed.

2.34 Increment Level

To increment the level, send…

0x08, 0x22, n

…where n = 0 to increase the level by the level increment, and n = 1 to decrease the level by the level increment.

The unit will respond with 3 bytes…

0x08, 0x22, n

… where n is a signed byte representing the new output level in dBm. If the value of n sent to the unit is not in the range of 0…1, the level will not be changed.

2.35 PLL Automatic Reporting

To read the current state of the phase locked loop (PLL) automatic reporting, send…

0x07, 0x23

The unit will respond with 3 bytes…

0x07, 0x23, n

…where…

n	PLL Automatic Reporting
0x00	PLL lock will not be automatically reported
0x01	PLL lock will be reported whenever it changes
0x02	PLL lock will be reported when it changes, but not more frequently than every 0.25 seconds.

To set the automatic reporting, send…

0x08, 0x23, n

…where n equals the desired function from the list.

The unit will respond with 2 bytes…

0x08, 0x23

Note that the unit will always default to zero or "No automatic reporting" on power up. When automatic reporting is enabled (n=1 or n=2), the unit will automatically send…

0x07, 0x24, n

…as indicated in section 2.36 whenever the PLL lock changes.

2.36 Read PLL Lock

To read the current state of the PLL, send…

0x07, 0x24

The unit will respond with 3 bytes…

0x07, 0x24, n

…where n = 0 when the PLL is unlocked, and n = 1 when the PLL is locked.

2.37 Read Running Time

To read the time that the unit has been powered, send…

0x07, 0x25

The unit will respond with 7 bytes…

0x07, 0x25, n_0…n_4

…where…

n_0…n_1 is a 16-bit integer (LSB first) representing the number of days (0–65535 decimal)
n_2 is a byte representing the number of hours (0–23 decimal)
n_3 is a byte representing the number of minutes (0–59 decimal)
n_4 is a byte representing the number of seconds (0–59 decimal)

2.38 Square Wave Modulation Parameters

To read the square wave modulation parameters, send…

0x07, 0x26

The unit will respond with 35 bytes…

0x07, 0x26, n_0…n_{32}

…where…

$n_{0...1}$ is a 16-bit integer (LSB first) = number of microseconds ON (10-16383)
$n_{2...3}$ is a 16-bit integer (LSB first) = number of microseconds OFF (10-16383)
n_4 is level mode (0 = no level mod; 1 = OOK; 2 = decrease output when "Low")
n_5 is number of dB to decrease output (+1 to +100) if $n_4 = 2$
n_6 is cycles mode (0 = continuous; 1 = modulate for "number of cycles")
$n_{7...8}$ is a 16-bit integer (LSB first) = number of cycles (1-60,000) if $n_6 = 1$
n_9 is square wave source (0 = internal; 1 = use aux input as mod source—aux input action must also be set to "Mod: Square wave source)
n_{10} is frequency mode (0 = no freq mod; 1 = + offset; 2 = − offset)
$n_{11...14}$ is a 32-bit integer (LSB first) representing the offset frequency in kHz if $n_{10} > 0$ (1-4,365,000 but current frequency + offset cannot exceed range of 35 MHz to 4.4 GHz)
n_{15} is frequency scan mode (0 = no frequency scan; 1 = scan frequency) If set to 1, n_{10} must be set to 0, and n_6 must be set to 0
$n_{16...19}$ is a 32-bit integer (LSB first) representing the start scan frequency in kHz if $n_{15} = 1$ (35,000 to 4,400,000)
$n_{20...23}$ is a 32-bit integer (LSB first) representing the stop scan frequency in kHz if $n_{15} = 1$ (35,000 to 4,400,000)
$n_{24...27}$ is a 32-bit integer (LSB first) representing the frequency step in kHz if $n_{15} = 1$ (1 to 1,000,000)
$n_{28...29}$ is a 16-bit integer (LSB first) = number of microseconds to dwell at each frequency (2-60,000)
n_{30} is frequency scan repetition (0 = continuous scan; 1 = one scan; 2 = number of scans)
$n_{31...32}$ is a 16-bit integer (LSB first) = number of frequency scans when $n_{30} = 2$ (2-60,000)

To set the square wave modulation parameters, send…

0x08, 0x26, n_0…n_{32}

…where n_0…n_{32} are the modulation parameters as outlined above. Note that you must include all data bytes (n_0…n_{32}) even if not using all features. Also, all parameters MUST be legal values as indicated above.

The unit will respond with 2 bytes…

0x08, 0x26

2.39 Scan Parameters

To read the scan parameters, send…

0x07, 0x27

The unit will respond with 43 bytes…

0x07, 0x27, n_0…n_{40}

…where…

n_0 is the frequency function (0=none; 1=scan up; 2=scan down; 3=up/down; 4=down/up)
$n_{1...4}$ is a 32-bit unsigned integer (LSB first) representing the high limit scan frequency in kHz
$n_{5...8}$ is a 32-bit unsigned integer (LSB first) representing the low limit scan frequency in kHz
n_9 is the level function (0=none; 1=scan up; 2=scan down; 3=up/down; 4=down/up)
$n_{10...11}$ is a 16-bit signed integer (LSB first) representing the high level limit in dBm
$n_{12...13}$ is a 16-bit signed integer (LSB first) representing the low level limit in dBm
n_{14} is the scan type (1=increment frequency; 2=increment level; 3=fit frequency; 4=fit level)
$n_{15...18}$ is a 32-bit unsigned integer (LSB first) representing the frequency increment in kHz
n_{19} is an unsigned byte representing the level increment limit in dBm
$n_{20...23}$ is a 32-bit unsigned integer (LSB first) representing the frequency time increment in milliseconds
$n_{24...27}$ is a 32-bit unsigned integer (LSB first) representing the level time increment in milliseconds
$n_{28...31}$ is a 32-bit unsigned integer (LSB first) representing the total scan time in milliseconds
n_{32} is the RF output auto on/off (0=don't automatically control output; 1=auto RF output on/off)
$n_{33...34}$ is a 16-bit unsigned integer (LSB first) representing the number of scans (0=continuous; 60,000 max)
$n_{35...36}$ is a 16-bit unsigned integer (LSB first) representing the inter-scan interval in ms (60,000 max)
n_{37} controls inter-scan RF output muting (0=don't mute; 1=mute)
n_{38} controls inter-scan frequency and level (0=jump to beginning-of-scan values; 1=hold end-of-scan values)
n_{39} controls step level after each scan (0=don't step level; 1= step level after each scan)
n_{40} controls repeat step level sequence (0=don't repeat; 1= repeat)

To set the scan parameters, send…

0x08, 0x27, n_0…n_{40}

…where n_0…n_{40} are the scan parameters as outlined above. Note that you must include all data bytes (n_0…n_{40}) even if not using all features. Also, all parameters MUST be legal values as indicated above.

The unit will respond with 2 bytes…

0x08, 0x27

For insight into the meaning of some of the more unusual parameters listed above, review the "Scan tab" section of the user manual.

2.40 Control Script

The unit can store up to 201 command steps in its control script. The end-of-script marker occupies a step, so a maximum of 200 executable commands maybe used. Each type of command is assigned a code number. Depending on the type of command, a parameter value may also be required. The parameter value must be set to zero for all commands that don't require an accompanying parameter. Following is the list of command codes and the range of acceptable values for each parameter.

Command Code	Description	Parameter Value	Parameter Units
0	End of script marker	0	
1	Turn RF output ON	0	
2	Turn RF output OFF	0	
3	Wait for command to continue	0	
4	Send beep command to host	0	
5	Set frequency	35.000 to 4400.000	MHz
6	Set power output level	-90 to +10	dBm
7	Pause	0.001 to 60.000	seconds
8	Trigger output ON	0	
9	Trigger output OFF	0	
10	Increment frequency	±0.001 to ±4365.000	MHz
11	Increment power	±1 to ±99	dB
12	Preset frequency[1]	1 to 12	
13	Preset power level[1]	1 to 12	
14	Script frequency[2]	1 to 10	
15	Script power level[2]	1 to 10	
16	Multiply frequency	0.100001 to 10	

[1] These refer to the presets numbered 1 to 12 for frequency and power level. They recall the value in the indicated preset and set the current frequency or power level to that value.

[2] These refer to the ten volatile frequency and ten volatile power level memories in the unit. They recall the value in the indicated memory and set the current frequency or power level to that value. See sections 2.40.10 and 2.40.12 to set these memories.

Each step in the script is assigned a step number, which is zero based, so the first step is step 0, and the highest possible step is step 200. The individual commands are sent to the unit, or read from the unit, one at a time. When a step is written, its step number must be included. Therefore, the steps don't necessarily need to be written in sequential order. The highest step number used contains the "End of script marker" or command code 0, and all lesser step numbers must be written with legitimate command codes and parameter values.

2.40.1 Read End of Script Step Number

To read the step number that contains the end-of-script marker, send…

0x07, 0x2A

The unit will respond with 3 bytes…

0x07, 0x2A, n

…where n equals the step number that was last written with the end-of-script marker. Note that this is also equal to the number of executable steps in the script. It is important to also note that attempting to read this value before the end-of-script marker has been written for a new script will result in reading the end-of-script step number from a previously written script. In other words, it is good practice to write all the steps of a new script followed by writing the end-of-script marker before attempting to read the step containing the end-of-script marker.

2.40.2 Write a Script Step

To write a script step to the unit, send…

0x08, 0x28, n_0…n_5

…where…

n_0 is the step number to write (0-200)
n_1 is the command code from the table to write (0-11)
$n_{2...5}$ is a 32-bit floating point value (IEEE-754, LSB first) = parameter value to write

The unit will respond with 2 bytes…

0x08, 0x28

2.40.3 Read a Script Step

To read a script step from the unit, send…

0x07, 0x28, n

…where n is the step number to read (0-200).

The unit will respond with 8 bytes…

0x07, 0x28, n_0…n_5

…where…

n_0 is the step number read (0-200)
n_1 is the command code for that step (0-11)
$n_{2...5}$ is a 32-bit floating point value (IEEE-754, LSB first) = parameter value for that step

The data returned from attempting to read a step number higher than the end-of-script step number is meaningless and should be ignored.

2.40.4 Write the Control Script Name

The control script may have a name attached to it. This name is displayed in TPI-Link when the script is read from the unit. The name is optional and may consist of up to 64 characters. To write a script name, send…

0x08, 0x29, n_0…n_{64}

…where…

n_0 is the number of characters in the scrip name (0-64)
$n_{1...64}$ is an ASCII string (not null terminated) = the script name

The unit will respond with 2 bytes…

0x08, 0x29

The number of bytes in the packet is dependent upon the length of the script name. To erase the name send…

0x08, 0x29, 0x00

2.40.5 Read the Control Script Name

To read the control script name, send…

0x07, 0x29

The unit will respond with 3 to 67 bytes…

0x07, 0x29, n_0…n_{64}

…where…

n_0 is the number of characters in the scrip name (0-64)

$n_{1…64}$ is an ASCII string (not null terminated) = the script name

2.40.6 Set the Control Script Run Control

To set the control script run control, send…

0x08, 0x2B, n_0, n_1

…where…

n_0	Unit State	n_1
0x00	Repeat script continuously	1-100 (ignored)
0x01	Run script once	1-100 (ignored)
0x02	Run script a number of times	1-100*

*Number of times to run script

The unit will respond with 2 bytes…

0x08, 0x2B

This command does not actually start the script running. It only specifies how many times the script will run when it is started. To start or stop the script running see section 2.8.

2.40.7 Read the Control Script Run Control

To read the control script run control, send…

0x07, 0x2B

The unit will respond with 4 bytes…

0x07, 0x2B, n_0, n_1

…where…

n_0	Unit State	n_1
0x00	Repeat script continuously	Ignore value
0x01	Run script once	Ignore value
0x02	Run script a number of times	1-100*

*Number of times to run script

2.40.8 Current Control Script Command

When each step of the command script is executed, the unit will send…

0x07, 0x2C, n_0…n_5

…where…

n_0 is the step number (0-200)
n_1 is the command code for that step (0-11)
$n_{2...5}$ is a 32-bit floating point value (IEEE-754, LSB first) = parameter value for that step

This is never requested by the user. It is automatically sent during script execution so that the user will know the current position of the executing script. Note that it is only sent during the first run of the script. If the run control is set for continuous running or to run a specific number of times, this packet will only be sent for each step of the first run of the script. The exception to this is that during all subsequent runs of the script, it will also be sent for each "wait" command step and each "beep" command step.

2.40.9 Read Control Script Frequency

The unit contains ten control script frequency memories that are volatile. They will all revert to 35 MHz each time the unit powers up. These memories may be written and read with the following two commands. Control script command code 14 is used within the script to set the current RF output frequency to the value in one of the ten memories.

To read the control script frequencies, send…

0x07, 0x2D, n

…where n is the number of the memory (1 through 10) that you wish to read.

The unit will respond with 7 bytes…

0x07, 0x2D, n_0…n_4

…where…

n_0 is the number of the memory (1-10)

$n_{1...4}$ is a 32-bit integer (LSB first) representing the frequency in kHz.

2.40.10 Set Control Script Frequency

To set the control script frequencies, send…

0x08, 0x2D, n_0…n_4

…where…

n_0 is the number of the memory (1-10)

n_1…n_3 is a 32-bit integer (LSB first) representing the frequency in kHz. The value must be between 35000 and 4400000 inclusive.

The unit will respond with 2 bytes…

0x08, 0x2D

2.40.11 Read Control Script Power Level

The unit contains ten control script power level memories that are volatile. They will all revert to 0 dBm each time the unit powers up. These memories may be written and read with the following two commands. Control script command code 15 is used within the script to set the current RF output power level to the value in one of the ten memories.

To read the control script power levels, send…

0x07, 0x2E, n

…where n is the number of the memory (1 through 10) that you wish to read.

The unit will respond with 4 bytes…

0x07, 0x2E, n_0…n_1

…where…

n_0 is the number of the memory (1-10)

n_1 is a signed byte representing the output level in dBm.

2.40.12 Set Control Script Power Level

To set the control script power levels, send…

0x08, 0x2E, n_0…n_1

…where…

n_0 is the number of the memory (1-10)

n_1 is a signed byte representing the output level in dBm.

The unit will respond with 2 bytes…

0x08, 0x2E

2.41 Read ADF4351 Registers

To read the current state of the ADF4351 registers, send…

0x07, 0x2F

The unit will respond with 26 bytes…

0x07, 0x2F, n_0…n_{23}

…where…

n_0…n_3 is a 32-bit integer (low byte first) representing the contents of register 0
n_4…n_7 is a 32-bit integer (low byte first) representing the contents of register 1
n_8…n_{11} is a 32-bit integer (low byte first) representing the contents of register 2
n_{12}…n_{15} is a 32-bit integer (low byte first) representing the contents of register 3
n_{16}…n_{19} is a 32-bit integer (low byte first) representing the contents of register 4
n_{20}…n_{23} is a 32-bit integer (low byte first) representing the contents of register 5

2.42 Set ADF4351 Registers

To set the contents of the ADF4351 registers, send…

0x08, 0x2F, n_0…n_{24}

The individual bits of n_0 determine what registers to write.

n_0	Unit State
bits 7:6	Reserved (always write 0)
bit 5	1 = write register 5
bit 4	1 = write register 4
bit 3	1 = write register 3
bit 2	1 = write register 2
bit 1	1 = write register 1
bit 0	1 = write register 0

Bytes n_1…n_{24} contain the data to write:

n_1…n_4 is a 32-bit integer (low byte first) representing the contents of register 0
n_5…n_8 is a 32-bit integer (low byte first) representing the contents of register 1
n_9…n_{12} is a 32-bit integer (low byte first) representing the contents of register 2
n_{13}…n_{16} is a 32-bit integer (low byte first) representing the contents of register 3
n_{17}…n_{20} is a 32-bit integer (low byte first) representing the contents of register 4
n_{21}…n_{24} is a 32-bit integer (low byte first) representing the contents of register 5

All 25 bytes of data must always be sent, even if you are not changing a register. For registers not changing, i.e., their corresponding n_0 bit is zero, the data will be ignored, so it may be any value. The registers to be written will be written in reverse order, i.e., register 5 will be written first, and register 0 will be written last.

The unit will respond with 2 bytes…

0x08, 0x2F

2.43 VCO Control

To read the VCO control state, send…

0x07, 0x30

The unit will respond with 3 bytes…

0x07, 0x30, n

…where n = 0 means the VCO is always powered, and n = 1 means the VCO is powered down when the RF output is muted.

To set the VCO control state, send…

0x08, 0x30, n

…where n = 0 means you wish the VCO to remain powered, and n = 1 means you wish to power down the VCO when the RF output is muted.

The unit will respond with 2 bytes…

0x08, 0x30

2.44 Auxiliary Level Control (TPI-1001 & TPI-1005 only)

To read the auxiliary input level control parameters, send…

0x07, 0x31

The unit will respond with 10 bytes…

0x07, 0x31, n_0…n_7

…where…

n_0…n_3 is a 32-bit floating point value (IEEE-754, LSB first) = voltage that equals 0 dBm
n_4…n_7 is a 32-bit floating point value (IEEE-754, LSB first) = dB/volt [Note 2]

To set the auxiliary input level control parameters, send…

0x08, 0x31, n_0…n_7

…where…

n_0…n_3 is a 32-bit floating point value (IEEE-754, LSB first) = voltage that equals 0 dBm
n_4…n_7 is a 32-bit floating point value (IEEE-754, LSB first) = dB/volt [Note 2]

The unit will respond with 2 bytes…

0x08, 0x31

Notes:

[1] If this command is sent to a TPI-1002, an error code will be triggered because the TPI-1002 does not have an auxiliary input.

[2] A negative dB/volt value will decrease level as the voltage increases.

2.45 Auxiliary Frequency Control (TPI-1001 & TPI-1005 only)

To read the auxiliary input frequency control parameters, send…

0x07, 0x32

The unit will respond with 11 bytes…

0x07, 0x32, n_0…n_8

…where…

n_0…n_3 is a 32-bit floating point value (IEEE-754, LSB first) = frequency in MHz at zero volts
n_4…n_7 is a 32-bit floating point value (IEEE-754, LSB first) = MHz/volt [Note 2]
n_8 is an unsigned integer = milliseconds between frequency changes (range is 2-100)

To set the auxiliary input frequency control parameters, send…

0x08, 0x32, n_0…n_8

…where…

n_0…n_3 is a 32-bit floating point value (IEEE-754, LSB first) = frequency in MHz at zero volts
n_4…n_7 is a 32-bit floating point value (IEEE-754, LSB first) = MHz/volt [Note 2]
n_8 is an unsigned integer = milliseconds between frequency changes (range is 2-100)

The unit will respond with 2 bytes…

0x08, 0x32

Notes:

[1] If this command is sent to a TPI-1002, an error code will be triggered because the TPI-1002 does not have an auxiliary input.

[2] A negative MHz/volt value will decrease frequency as the voltage increases.

2.46 Frequency List (TPI-1001 & TPI-1005 only)

There is a new feature in the TPI-1001 & TPI-1005 called Frequency List. You can program up to 1000 random frequencies in non-volatile memory. You then can step through the list of frequencies by applying a signal to the auxiliary input. The list of frequencies may contain from 1 to 1000 frequencies although a list of less than two frequencies is not useful. After the last frequency in the list you need to program an end-of-list marker in the next memory position so that the unit knows when to stop advancing to the next frequency. If memory position 1000 is used, you do not need to (actually cannot) program an end-of-list marker because there is no memory position 1001.

To set a frequency in the list, send…

0x08, 0x35, n_0…n_5

…where…

n_0…n_1 is a 16-bit unsigned integer (LSB first) = memory position (range is 1-1000)
n_2…n_5 is a 32-bit unsigned integer (LSB first) = frequency in kHz (range is 35,000 to 4,400,000)
If zero is written for the frequency, this indicates an end-of-list marker.

The unit will respond with 2 bytes…

0x08, 0x35

To read a frequency from the list, send…

0x07, 0x35, n_0…n_1

…where…

n_0…n_1 is a 16-bit unsigned integer (LSB first) = memory position (range is 1-1000)

The unit will respond with 6 bytes…

0x07, 0x35, n_0…n_3

…where…

n_0…n_3 is a 32-bit unsigned integer (LSB first) = frequency in kHz (range is 35,000 to 4,400,000)
If zero is returned, this memory position contains an end-of-list marker.

Before you begin stepping through the frequency list, you need to set a starting point in the list. In other words, you need to set the current frequency list position.

To set the position in the list, send…

0x08, 0x36, n_0…n_1

…where…

n_0…n_1 is a 16-bit unsigned integer (LSB first) = memory position (range is 1-1000)

The unit will respond with 2 bytes…

0x08, 0x36

To read the current position in the list, send…

0x07, 0x36

The unit will respond with 4 bytes…

0x07, 0x36, n_0…n_1

…where…

n_0…n_1 is a 16-bit unsigned integer (LSB first) = current memory position (range is 1-1000)

Note that it is possible to have several lists of frequencies stored one after the other with an End Marker following each list. Set the position to the first frequency in the list you wish to execute. The execution will stop when an end marker is encountered.

For the auxiliary input to function as a signal to advance to the next frequency in the list, you must set the auxiliary input action to “Next frequency in list”. See section 2.19.

2.47 Error Notification

If an error is encountered, the unit will send…

0x07, 0xFF, n

…where n is the error number (decimal) according to the following table…

Error #	Description
1	Checksum error
2	Undefined command type (first body byte)
3	Undefined command (second body byte)
4	Data out of range
5	Improper register ID
6	Illegal beacon message character
7	Requested RF level < -90 dBm
8	Internal beacon message length error
9	Unknown script command
10	No detector available
11	No auxiliary input available
12	No trigger output available
27	Communication watchdog timeout
88	Failed to write EEPROM
89	Failed to read EEPROM

3 C# Code Examples

This section will give some C# sample code snippets that can be referenced by a programmer to develop a program that communicates with the unit.

3.1 Establish the COM Port

Establish the COM port…

```
using System.IO;
using System.IO.Ports;

public partial class Form1 : Form {
       private SerialPort sp = new SerialPort();
       }

public Form1() {
       InitializeComponent();
       sp.DataReceived += new SerialDataReceivedEventHandler(sp_DataReceived);
       }
```

Create a function similar to this to open the port. This function will return “true” if the port was successfully opened.

```
private Boolean Open_Port() {

       try {
              sp.PortName = “COM4”;  // Substitute your COM port number for “COM4”
              sp.Open();
              if (sp.IsOpen) {
                     sp.BaudRate = 3000000;
                     sp.DataBits = 8;
                     sp.StopBits = StopBits.One;
                     sp.Parity = Parity.None;
                     sp.Handshake = Handshake.RequestToSend;
                     sp.Encoding = System.Text.Encoding.GetEncoding(1252);
                     return true;
                     }
              else {
                     return false;
                     }
              }
       catch {
              return false;
              }
    }
```

3.2 Send a Packet

As an example, send this packet to enable user control. The “7” is the number of bytes to send.

```
    sp.Write(new byte[] { 0xAA, 0x55, 0x00, 0x02, 0x08, 0x01, 0xF4 }, 0, 7);
```

3.3 Read Bytes From the COM Port

The following code is an example of how to read bytes from the COM port.

```
private void sp_DataReceived(object sender, SerialDataReceivedEventArgs e) {

	byte z;

	while (sp.IsOpen && sp.BytesToRead > 0) {

		z = (byte) sp.ReadByte();  // Get next byte from serial port
		// Do something here with the bytes received
		// This code will run for each byte received
		}
	}
```

The programmer needs to devise a method of parsing the bytes received to establish the beginning of a packet, its length, and if the checksum is correct. The data from the packet may then be examined to understand the meaning of the packet.

4 Revisions

Rev 1.0	September 23, 2015	Original version supporting unit firmware version 1.017
Rev 1.1	October 17, 2015	Added section 3 with programming examples
Rev 1.2	November 28, 2015	Noted section 2.13 as “TPI-1001 only” Fixed typo in section 3.1
Rev 1.3	February 23, 2016	Added sections 2.35 and 2.36 about PLL lock Corrected hex equivalent in section 1.2
Rev 1.4	April 4, 2016	Added section 2.37 about running time
Rev 1.5	April 21, 2016	Added section 0about square wave modulation parameters
Rev1.6	July 22, 2016	Added section 0about scan parameters
Rev 1.7	August 22, 2016	Added section 2.40 about control scripts
Rev 1.8	October 20, 2017	Section 2.40—added command codes 12 through 15 Added section 2.40.9—read control script frequency Added section 2.40.10—set control script frequency Added section 2.40.11—read control script power level Added section 2.40.12—set control script power level
Rev 1.9	March 4, 2018	Section 2.40—added command code 16
Rev 1.10	July 4, 2018	Added section headers to pdf bookmarks
Rev 1.11	August 19, 2018	Added sections 2.41 and 2.42 about ADF4351 registers
Rev 1.12	August 22, 2018	Fixed typo in section 0
Rev 1.13	September 18, 2018	Section 2.20—added trigger out sources
Rev 1.14	January 27, 2019	Section 2.38—added frequency scan
Rev 1.15	March 26, 2019	Sections 2.22 & 2.30—added 6 more frequency presets Sections 2.23 & 2.31—added 6 more level presets Section 2.38—reduced minimum on/off times
Rev 1.16	April 10, 2021	Added section 2.43 about VCO control
Rev 1.17	July 10, 2021	Section 2.19—added 3 auxiliary input actions Added section 2.44 about auxiliary level control Added section 2.45 about auxiliary frequency control
Rev 1.18	January 25, 2023	Section 2.19—added “Next frequency in list Added section 2.46 “Frequency List”