InPlay NanoBeacon™
IN100 Config Tool
Application Notes

NanoBeacon™

About Documentation

Document Name	IN100C0 Application Notes	
Part number		
Control Number	IN1IDOC-SW-IN100-EN-V1_04	External User
Revision	V1.04	

Product Status	Document Content	Data Status
In Development	Target specification / MRD	Initial release
Engineering Document	Main functions and features description	Preliminary version
Official Release Document	Description of all functional features	Subject to revision and updates

This document applies to the following products:

Document number	Applicable Products	Document Status
	IN100-D1-R-RC0I	Engineering sample
	IN100-Q1-R-RC0I	In development
	IN100-D1-R-RC0F	Engineering sample
	IN100-Q1-R-RC0F	In development

Contents

List of Figures

List of Tables

1. Introduction

NanoBeacon Config Tool is a Windows platform software program for testing and writing configurations for the NanoBeacon DK (Development Kit). When testing a configuration, the configuration is run from RAM and will execute as long as the tag is being powered. After finishing with the testing, the desired configuration may be written to the device's one-time programmable (OTP) memory. NanoBeacon Config Tool software is used to configure the Bluetooth beacon payload and related options. Figure 1 shows the overall setup.
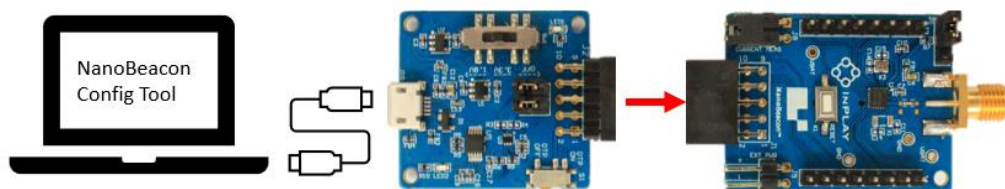
Figure 1 Overall view of NanoBeacon device programming setup

The NanoBeacon Config tool connects to the OTP memory programmer board via USB, which has a USB to UART adapter. The NanoBeacon tag connects to the programmer through a 8-pin connector. The GUI of the NanoBeacon Config program is shown in Figure 2.
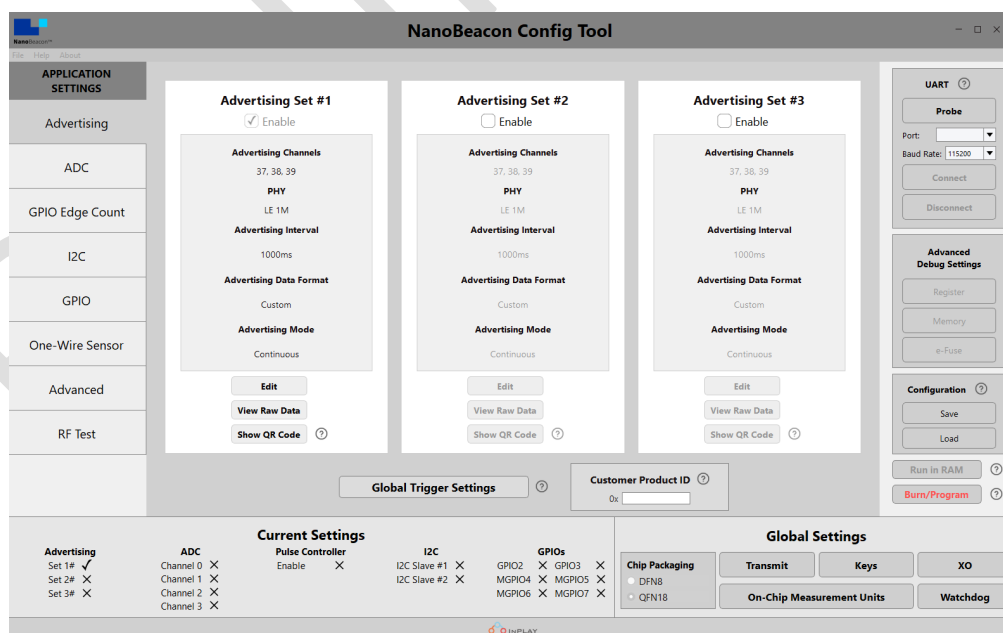
Figure 2 NanoBeacon config PC GUI tool

A closer look at the NanoBeacon DK is shown as in Figure 3.
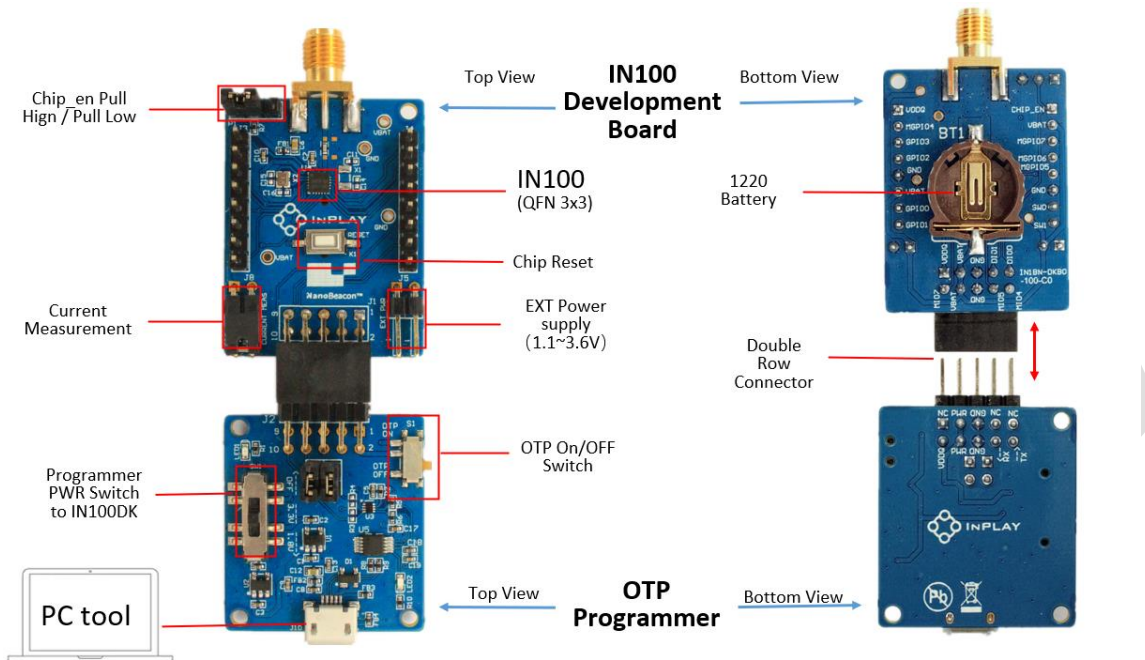
Figure 3 NanoBeacon development kit (front and back view)

Things You Will Need

Hardware
- IN100 NanoBeacon Development Kit - includes programmer and tag(s)
- 3.5mm male SMA RF antenna
- USB-A to Micro-USB cable
- Windows PC for running NanoBeacon Config software tool
- Device capable of scanning for Bluetooth packets (phone, tablet, etc)
- [optional] CR1220 battery

Software (freely available from InPlay website at https://inplay-tech.com/in100)
- NanoBeacon Config Tool software (Windows/PC) – for configuring the tag
- NanoBeacon Scan App (Android) or other Bluetooth Scan app (several freely available on Google Play Store and Apple App Store) – for scanning the tag once it starts transmitting

Tools
- [optional] Soldering iron + solder
- [optional] Dupont wires and connectors
- [optional] Ammeter for current measurement

How to Configure and Use NanoBeacon

Step 1: Insert IN100 NanoBeacon DK board into the programmer board (make sure to follow the orientation as shown in Figure 4).

At this point, it is not necessary to install battery in tag (DK board).

Step 2: Connect OTP memory programmer to PC with a USB cable. Green and red LEDs on programmer should light. Run NanoBeacon Config Tool on PC.
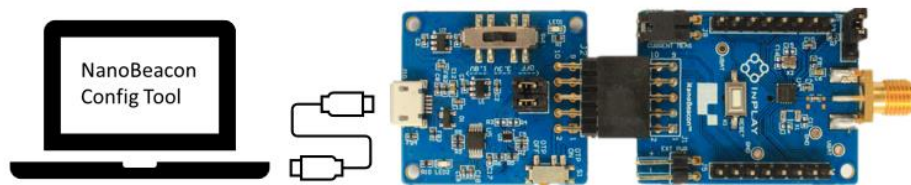
Figure 4 Connect NanoBeacon DK board to programmer board

Step 3: Uncompress the 'NanoBeaconConfigTool.7z' file archive, locate the 'NanoBeaconConfigTool' application and execute it.

Name	Date modified	Type	Size
NanoBeaconConfigTool	3/22/2022 8:09 AM	Application	5,556 KB

To control the programmer, a serial communication link must be established first by detecting the ports, selecting the right COM port and the correct Baud rate. After selecting the right port and baud rate, click the connect button to connect the PC with the tag.

Step 4: Obtain the configuration based on the user's needs, and verify the configuration using the "Run in RAM" mode, by clicking the 'Run in RAM' button. After clicking 'Run in RAM', use a scanning device like an Android phone or tablet to see if the beacons are received. Please refer to the 'InPlay IN100 NanoBeacon Android Scanner User Guide' about installing and using InPlay's custom NanoBeacon Android application. The Run in RAM loads the configuration into RAM and executes it while the device is powered. To quit from the "Run in RAM" mode, 'Close' the UART connection, and disconnect power from the tag by unplugging the programmer from the USB. Another way to

quit from the "Run in RAM" mode is simply to reset the device by pressing the chip reset button in the tag board.

After testing and verifying the configuration in "Run in RAM" mode, customers may burn the configuration into the device's OTP memory (Step 5). Please be noted: Configurations involve I2C operations are not supported in the "Run in RAM" mode.

Step 5: 'Burn' the desired configuration into the tag's OTP memory. The tag must be powered while burning. A status bar shows the progress of the burning operation. If the operation is successful, there will be a successful message popped up. If the operation fails, there will be an error message. After burning the configuration, the power to the development board must be removed and re-applied to start the tag executing according to its programmed behavior. A development board's OTP memory can only be written ONCE.

Step 6: Exit the NanoBeacon Config Tool program. Disconnect the development board from the programmer and insert a CR1220 battery in the board to start its operation.
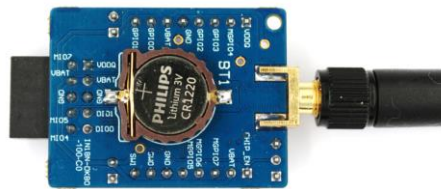
2. NanoBeacon config tool introduction

InPlay NanoBeacon Configuration Tool is a PC GUI (Graphical User Interface) tool for chip firmware configuration. The PC application file is NanoBeacon Config.exe. Developers should have the flexibility to configure the peripherals, advertising data and advertising parameters of the NanoBeacon device (IN100) according to user's application requirements.
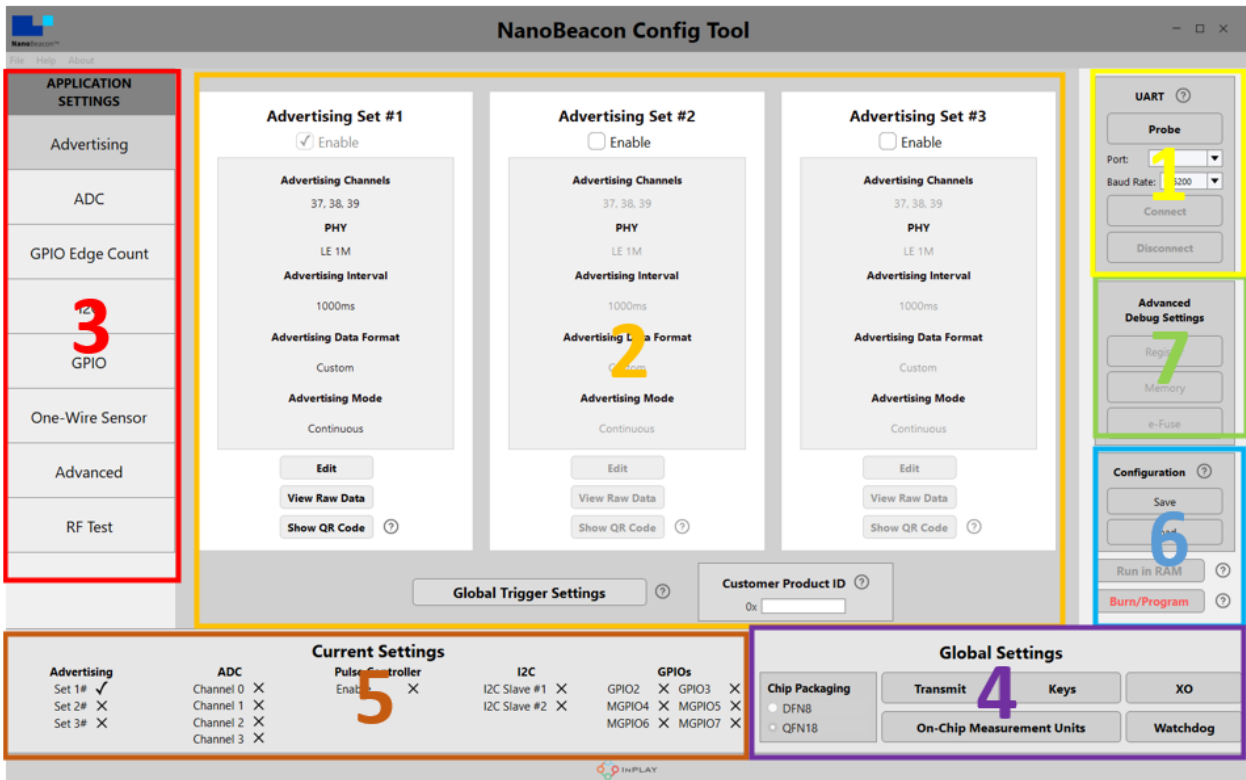
Figure 5 Overview of NanoBeacon configuration tool

Figure 5 shows the general look of the GUI software tools, which consists of :

1) **UART**, which is used for setting UART port and baud rate on the PC to communicate with the development kit. The default baud rate is 115200.

2) **Main window**, which provides a main configuration window for user to configure the device.

3) **Application Settings**, which includes all configurable option tabs including Advertising Settings, ADC, Plus Count Controller, I2C , GPIO and GPIO Edge Count. In addition, there is an advanced mode setting for using special device behavior settings not covered in the regular option settings. The RF Test tab provides an easy-to-use interface for users to run various RF tests, including BLE DTM mode test and carrier test.

4) **Global settings,** which provides settings to the device that applies to all advertising sets.

5) **Current Settings,** which provides an overview of device settings and configurations.

6) **Configuration,** which provides the user with a way to save the configuration to a file or load it from an existing configuration file. Once the configuration file is successfully loaded, the user has two options to try and test the configuration on the device. One is

"Run in RAM", which provides a way to test and verify the user's configuration before it is permanently burned to the device. Whenever there are any changes to the configuration, the user should always reset the device before operating "Run in RAM". The other is "Burn/Program" which will permanently burn the configuration file to the device, which is irreversible and should be performed as the last step.

7) **Advanced Debug Settings,** which provides a way to read/write device's register/memory/eFuse and this is used for debugging purposes only.

2.1   Advertising sets configuration

2.1.1   Overview

The IN100 device supports up to three advertising sets: Advertising set #1, #2, and #3 which can be individually configured by using the NanoBeacon configuration tool through **Advertising set #1**, **Advertising set #2** and **Advertising set #3**, respectively.

The overall brief configuration settings tab page for each advertising set is shown in Figure 6.
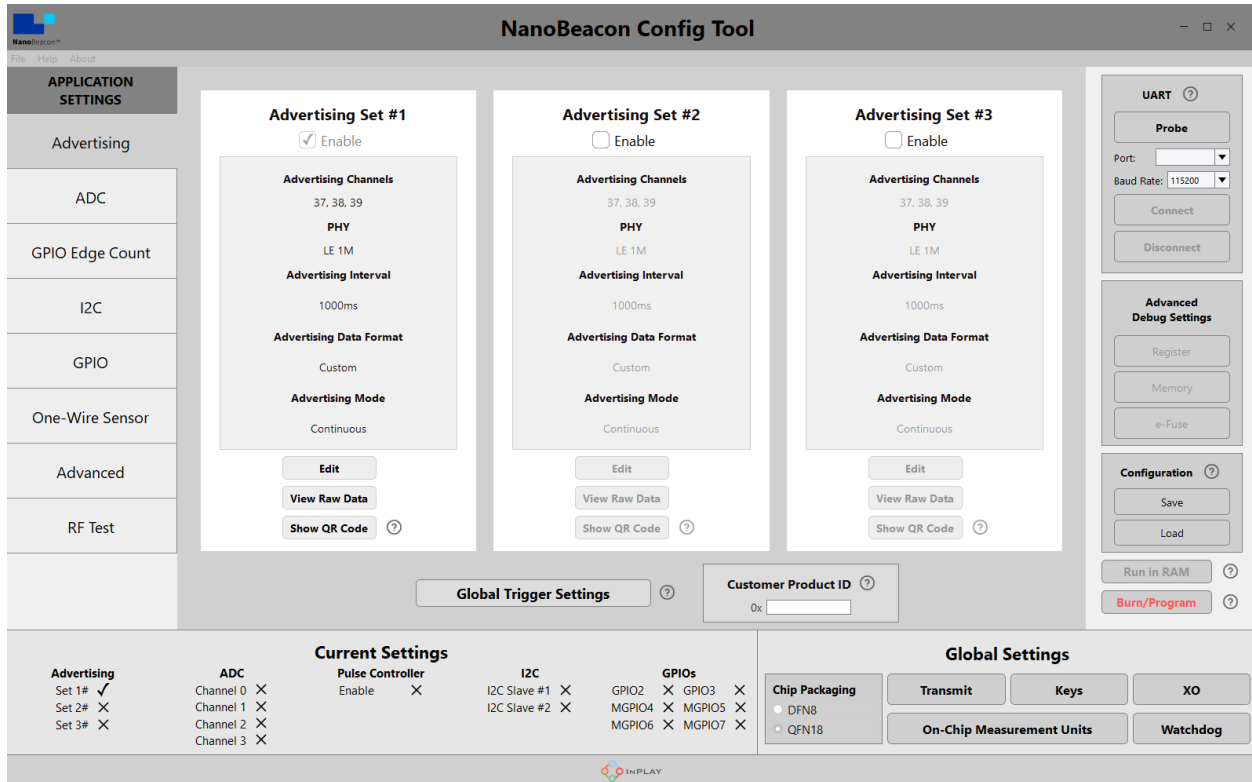
Figure 6 Advertising sets configuration overview

User can click the "Edit" button to access the configuration page to enter the configuration and parameters related to the advertising.

The tool also provides the option to "View Raw Data" for the user to review the full payload data of the configuration, including encryption status, endianness, and notes about the specific data fields. An example of such is shown as in Figure **7**.
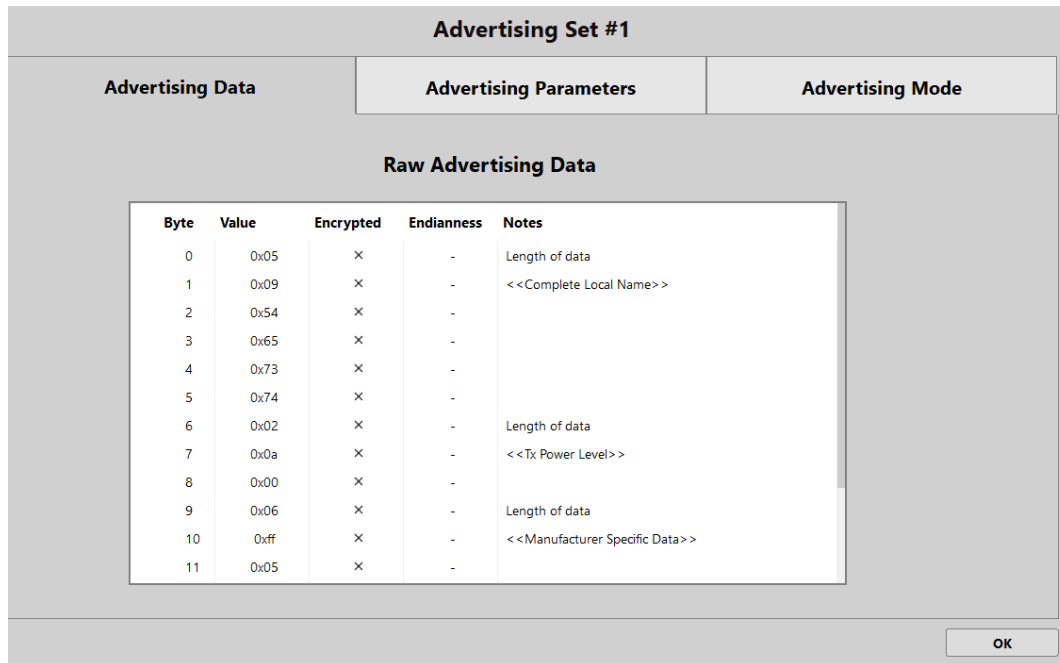
Figure 7 Raw advertising data review window

The "Show QR Code" button will provide a QR code which is the advertising address information for the user's mobile application to easily identify and parse data from advertisers.

The "Global Trigger Settings" allows users to set trigger conditions that apply to all advertising sets.

The "Customer Product ID" is a unique ID that can be assigned to each device. It can be used like a serial number, for example, to assign a unique number to each device during the manufacturing process. In our payload data field information, it is referred to as "CustID".

2.1.2 Advertising set configuration

The main window provides users to access to all possible configuration parameters for their application requirements including Advertising Data, Advertising Parameters and Advertising Mode, as shown in Figure **8**.
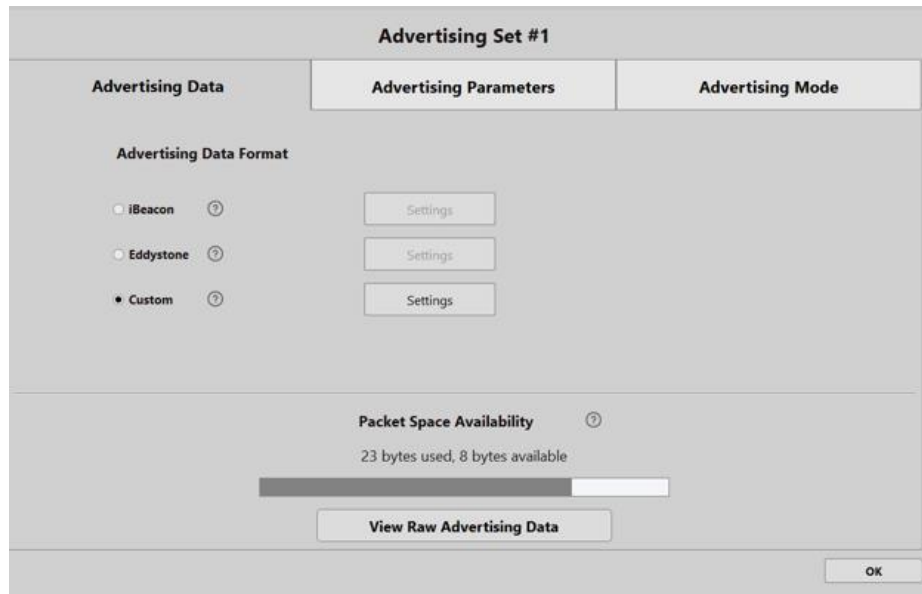
Figure 8 Advertising set configuration window

2.1.2.1   *Advertising data format*

The NanoBeacon config tool supports both standard BLE packet data configuration through AD structures and user defined configuration completely controlled by user.

Figure 9 shows the advertising data format. It consists of multiple BLE standard AD structures and/or multiple user defined data sections. It should be pointed out that the advertising data can just consist of as simple as one AD structure or one user defined data section. The standard BLE AD structure consists of a Length field of one byte, which contains the Length value, and a Data field of Length bytes. The first byte of the Data field is the AD type field. The content of the remaining Length – 1 bytes is called the AD data.
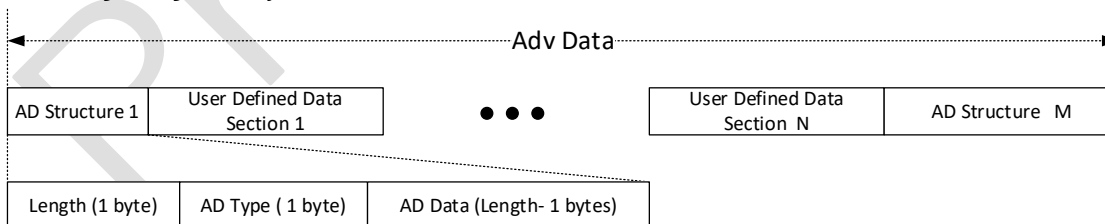
Figure 9 Adv data format

The AD data or the user defined data section consists of multiple data segments. There are two categories for the data source for each data segment: predefined and extended data.

Figure 10 shows an example of the data format for the AD data or the user defined data section.
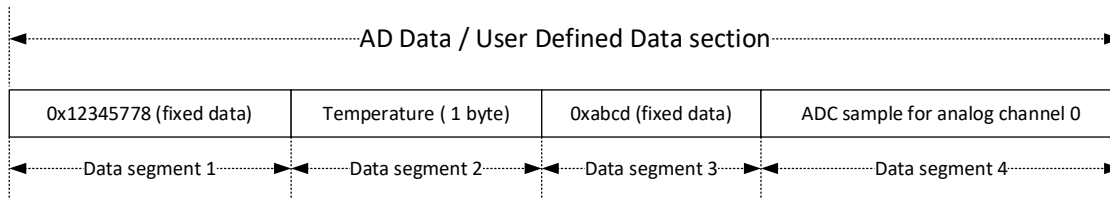
Figure 10 Data format for AD data/user defined data section

The tool supports three commonly used Advertising data formats.

- iBeacon, Apple proprietary Beacon advertising frame format based on Bluetooth standard advertising.
- Eddystone, a Google proprietary Beacon advertising frame format based on the service data of the Bluetooth standard advertising.
- Custom, an advertising data in a custom user-defined format.

1) In case of the "iBeacon Advertising Settings", the user needs to fill in the followings:

   o **UUID**: Application developer should define a UUID specific to their app and development use case. The default value is E2C56DB5-DFFB-48D2-B060-D0F5A71096E0.
   o **Major**: Further specifies a specific iBeacon and use case. For example, this could define a sub-region within a larger region defined by the UUID.
   o **Minor**: Allows further subdivision of region or use case, specified by the application developer.
   o **Measured Tx Power**: Apple recommends measuring the RSSI of the advertising beacon at a constant distance of 1 meter at multiple positions, then applying the average measured RSSI value to the beacons transmit power.

2) In case of the "Eddystone Advertising Settings", the user needs to follow the Eddystone protocol and configure the device:

   o For details, please refer to Google Eddystone protocol specification at https://github.com/google/eddystone/blob/master/protocol-specification.md.

3) In case of the "Custom Advertising Settings", the user can configure the below based on their application needs:

   o **Device Name:** This is a string that the user chooses to label the device in its advertising packets.
   o **Tx Power Level:** The Tx Power level data type indicates the transmitted power level of the advertising packet.

- o **Manufacturer Specific Data:** Used for manufacturer specific data. The first two data octets shall contain a company identifier from the Assigned Numbers (https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers/) reference. The default manufacture ID is 0x0505 which is the Assigned Number for InPlay Inc by Bluetooth SIG. The interpretation of any other octets within the data shall be defined by the manufacturer specified by the company identifier.
- o **User Defined Data:** This data is up to the user to define in their desired format.
- i. There is dynamic data source can be selected from the list provided and click "Append to Data" will add this selected data to the advertising packet.
- ii. Extended data source: Includes data samples from ADC and digital sensors, timers, GPIO status, registers and random numbers, etc. Most of the data is dynamic, such as the temperature sensor data. The list of predefined data sources is as followings:
  - a) VCC (8bit): The measured 8-bit chip's supply voltage. One LSB corresponds to 1/32 volt.
  - b) Internal Temperature (default 2 bytes): The measured 16-bit temperature using the internal temperature sensor inside the chip. It is 2's complement format with unit 0.01 degree Celsius. If 1 byte is entered for the temperature value representation, it is 2's complement format with unit 1 degree Celsius (TBA).
  - c) 1-Wire Sensor: Pulse count collected by the 1-wire digital pulse controller, 2 bytes.
  - d) GPIO Status: GPIO input status, 1 byte.
  - e) ADC CH0 – Digital sample of the external analog input to the mixed signal GPIO pin 4 (MGPIO4), 2 bytes.
  - f) ADC CH1 – Digital sample of the external analog input to the mixed signal GPIO pin 5 (MGPIO5), 2 bytes.
  - g) ADC CH2 – Digital sample of the external analog input to the mixed signal GPIO pin 6 (MGPIO6), 2 bytes.
  - h) ADC CH3 – Digital sample of the external analog input to the mixed signal GPIO pin 7 (MGPIO7), 2 bytes.
  - i) I2C Slave #1 Read Data: The data of a I2C Slave #1 data read out with a given I2C Slave #1 read address (default address is 0x2300). Read address **Offset** and read length (**Bytes**) can by entered in bytes. The read data can be with endianness and encryption options. Only the selected read data will be advertised as the data payload. An example of such is as shown in Figure 11.The offset is 4 and Bytes is 3. The device will only advertise the 3 bytes of I2C read data starting from the address of 0x2304.
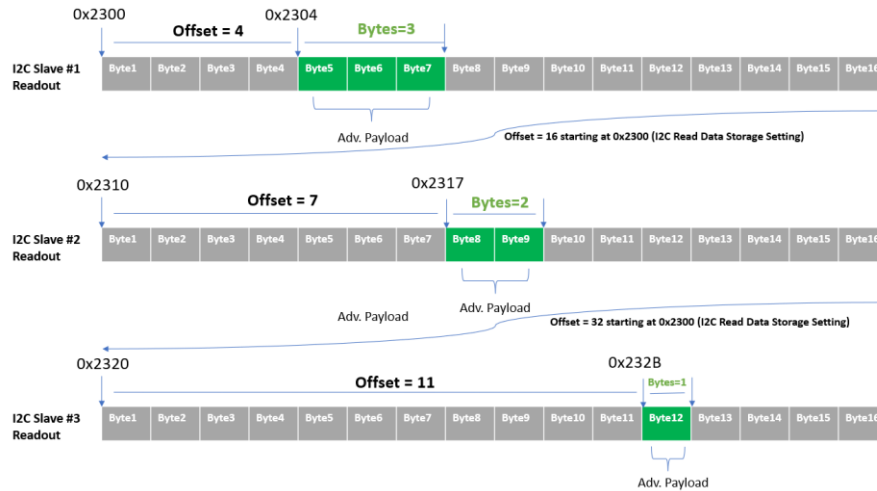
Figure 11 I2C read data as adv. payload

j) **I2C Slave #2 Read Data:** The data of a I2C data #2 data read out with a given I2C Slave #2 read address (default address is 0x2310). Read address **Offset** and read length (**Bytes**) can by entered in bytes. The read data can be with endianness and encryption options. Only the selected read data will be advertised as the data payload as shown in Figure 11.

k) **I2C Slave #3 Read Data:** The data of a I2C data #3 data read out with a given I2C Slave #3 read address (default address is 0x2320). Read address **Offset** and read length (**Bytes**) can by entered in bytes. The read data can be with endianness and encryption options. Only the selected read data will be advertised as the data payload as shown in Figure 11.

l) Time Stamp 0: Time stamp in 100 milliseconds as unit, up to 4 bytes.

m) Time Stamp 1: Timer stamp in 1 second as unit, up to 4 bytes.

n) ADV Count: The transmit advertising event count, up to 4 bytes.

o) Register Read Data: The data of a read register with a given address, up to 4 bytes.

p) Random Number -- Random data, up to 4 bytes.

q) Encrypted Raw: the pre-defined data to be encrypted.

r) EAX Salt: Salt for encryption and authentication

s) Auth Tag: The tag for authentication

t) Customer Product ID: a unique ID that can be assigned to each device.

u) Bluetooth Device Address: Bluetooth device address that is defined in Advertising parameters settings.

o **Data Encryption Settings:** User can encrypt the payload data with the Encryption key.

  i. **Encryption Key**: User can generate the cryptographic key for encryption through clicking "Keys" button in Global Settings. Up to three keys can be generated or entered manually. Each key is 16 bytes. They are mainly used for advertising data encryption or private resolvable advertising address encryption.

  ii. **Nonce Salt (2 bytes):** User can use randomly generated data or predefined 2 bytes of data as salt.

  iii. **Nonce Counter (4 bytes):** User can choose one of the three options as Nonce Counter: Time Stamp 1, Advertisement Count or Predefined value (4 bytes).

Note: Please refer to 2.1.2.3 for details on data encryption and authentication.

2.1.2.2   *Advertising data configuration*

The Advertising data consists of multiple user defined data sections.

To add a user defined data section to the advertising data, select data format from one of the three data formats below.

- iBeacon.
- Eddystone.
- Custom.

1) Enter one data segment:
    o If it is predefined data, user can directly type in the data entry box. **Note**: Only hexadecimal data is supported where 2 characters represent one byte.
    o If it is extended data, we can select the data from the drop-down list from Dynamic Data box and then press button "Append to Data". The added extended data will be shown in the data entry inside "<>", as shown in Figure 12. For most of the extended data, the user can configure the number of bytes to be advertised. For example, the "Adv Count" (advertising count) internally has 4 bytes. A user can configure the device to advertise only 2 LSB bytes only by changing the number of bytes to 2. Depending on the data source selected, the endianness, encryption option and Trigger Snapshot option can be selected collectively or respectively.
2) Repeat the 1st step as necessary.
3) After finishing configuration of the data segment, click the "OK" button in the "Advertising Data" tab to add the data to the Advertising Data . Note that each standard AD type can only be added once while user defined data type can be added multiple times. **Note**: For the standard BLE AD structure, the length field and AD type are automatically added to the advertising data once we press the "OK" button.
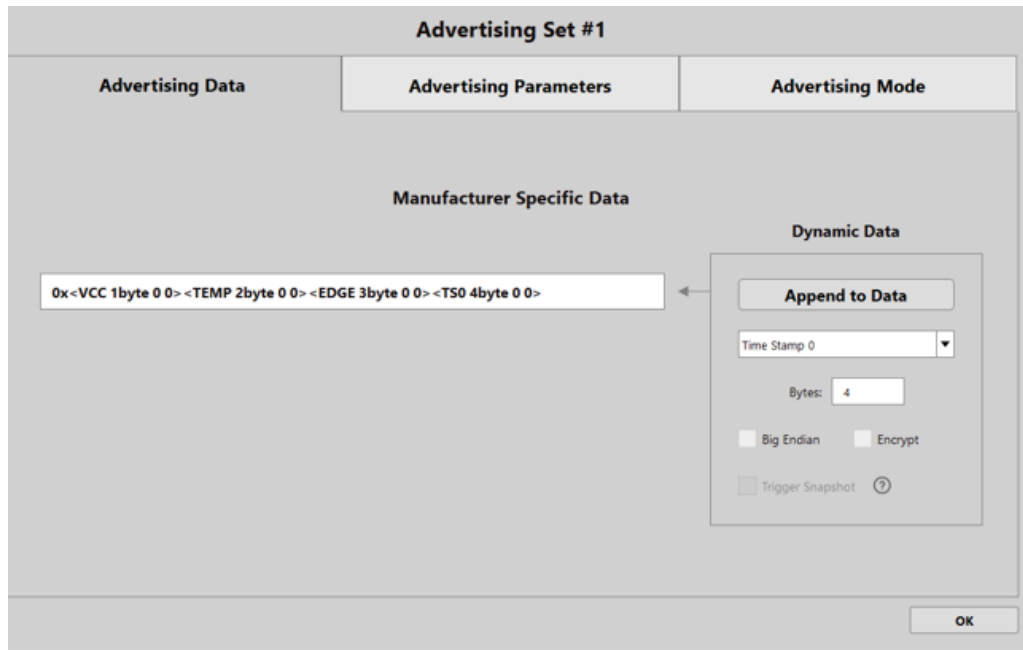
Figure 12 Selecting dynamic data

2.1.2.3   Advertising data encryption and authentication

The NanoBeacon device supports simultaneous encryption and authentication to the advertising data, based on the AES-EAX encryption algorithm (M. Bellare, P. Rogaway and D. Wagner. "EAX: A Conventional Authenticated-Encryption Mode", 2003). Each of three advertising data sets can be individually and independently configured regarding whether user needs to enable encryption and/or authentication.

The AES-EAX engine

To enable encryption and authentication, in the device, the user needs to specify the following inputs to the AES-EAX engine for each advertising set.

* A single 16-byte key used for both encryption and authentication.
* The selected data portion (plain text) in the advertising data to be encrypted.
* A 6-byte nonce.
* The number of bytes for the message authentication code (MAC, usually called as a tag).

The outputs of the AES-EAX engine are:

* The encrypted data (ciphertext), which has the same length as the plain text.
* The tag.

Figure 13 illustrates the advertising data without and with encryption/authentication. We assume the data portion 2 needs to be encrypted and authenticated. The salt in the figure is a part of 6-byte nonce.

The salt and the tag are optional, and they are not required to be present in the advertising data.

- If the intended receivers know what the salt is used, then customers can select not to present the salt in the advertising data through the NanoBeacon Config Tool.
- If the user does not need authentication, then the user can select not to present the tag in the advertising data through the NanoBeacon Config Tool.
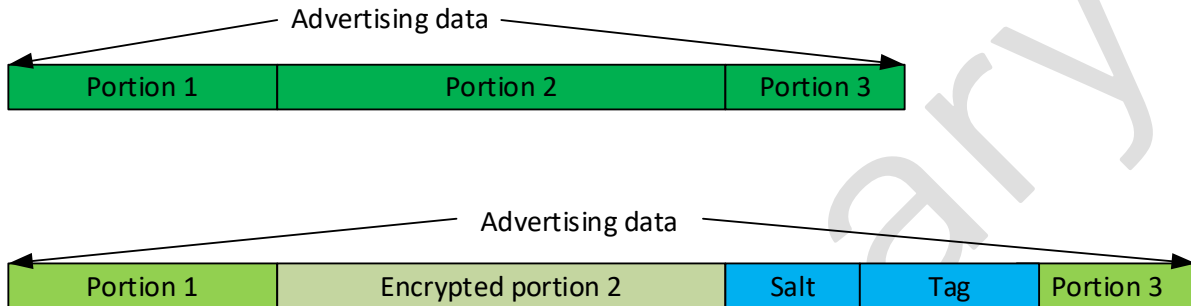

Figure 13 Advertising data encryption and authentication

To enable encryption and authentication for the specific data field in each advertising set, the user needs to check the "Encrypt" box as shown in Figure 14.
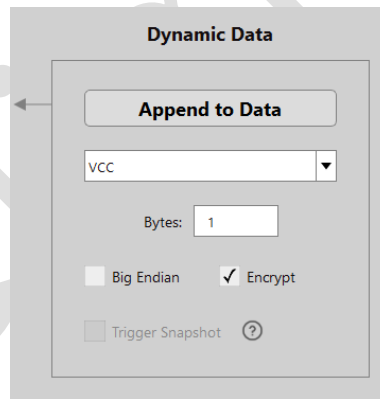

Figure 14 Encryption enabling

Keys

The AES-EAX needs a 16-byte key for the encryption and authentication for each advertising set. There are three keys (key0, key1 and key2) to be chosen from, as shown in Figure 15.
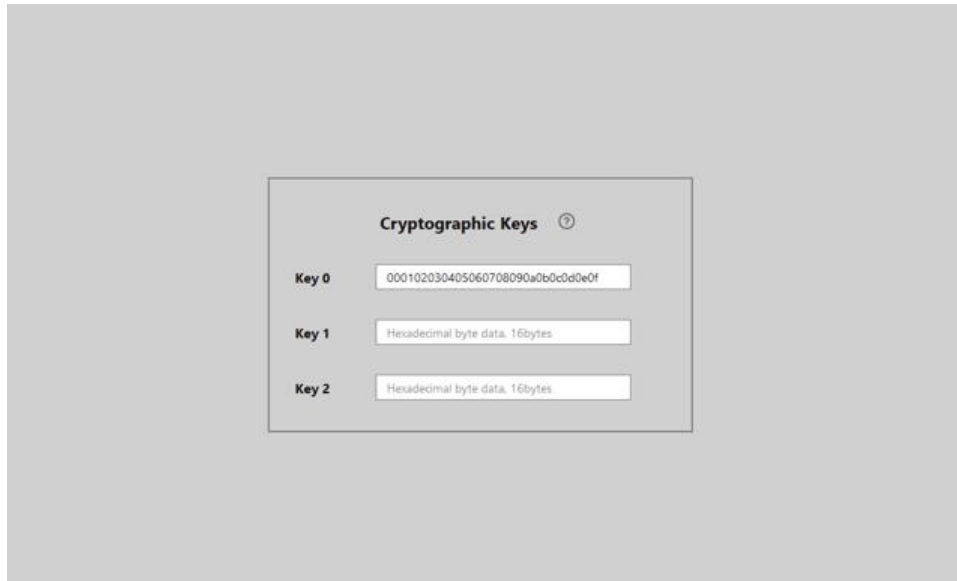
Figure 15 Selection of key and key for AES-EAX

The configuration of the three key is in the "Global Settings as shown in Figure 15. The input box of the corresponding key needs to be filled with hexadecimal data.

Nonce

The 6-byte nonce consists of a 2-byte salt and a 4-byte counter (we refer this counter as nonce counter). The device offers a variety of selections for the salt and the counter.

1) Nonce Salt

There are two selections available for the salt as shown in Figure 16:
- Random number: The salt will change every advertising.
- Predefined fixed number.

If the receivers know the salt, it does not need to be advertised. If the salt is a random number, we need to include the salt in the advertising data as a plain text, which is shown in Figure 16.
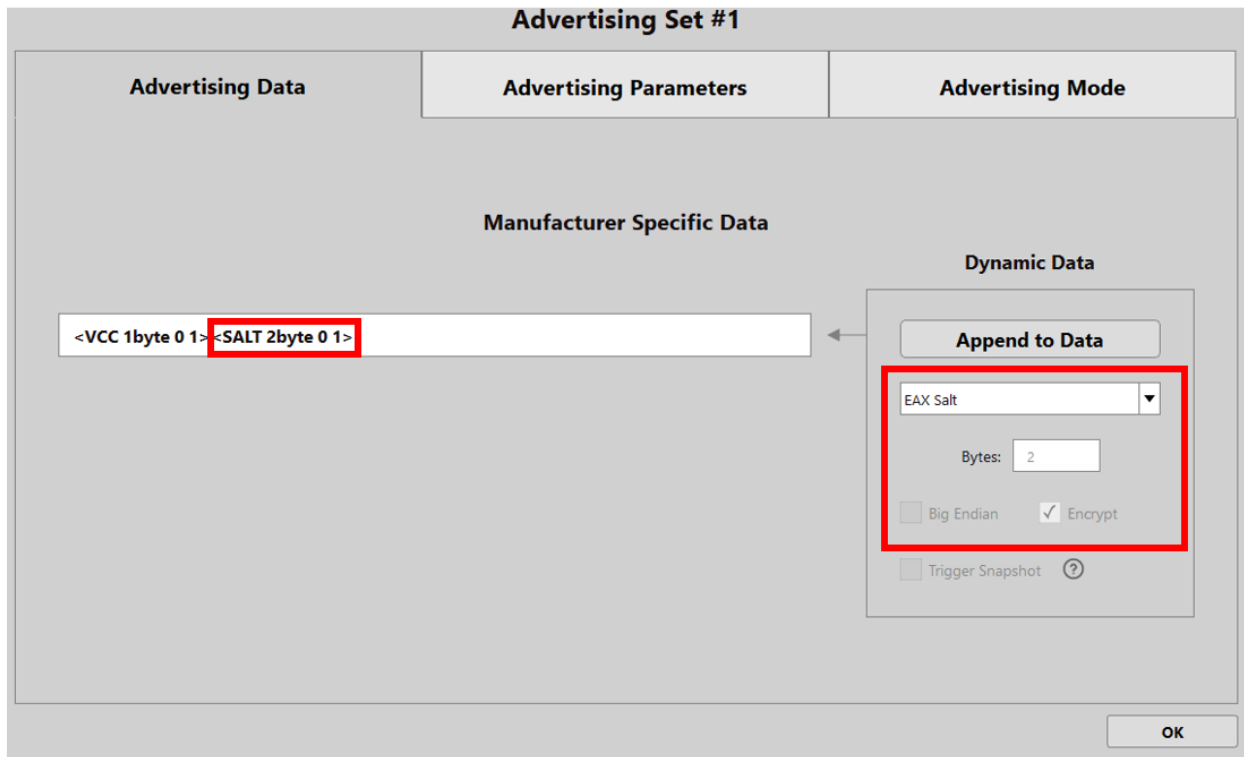
Figure 16 Including the AES-EAX salt in the advertising data

2) Nonce counter

There are multiple selections available for the nonce counter as shown in Figure 16:

- Time Stamp 1: The device has a 32-bit second counter. It starts counting from 0 upon power cycle.
- ADV Count (Advertising counter): The device provides an advertising counter for each advertising data set. When an advertising set is advertised over the air, the corresponding counter increases by 1. Upon power-cycle, the counters are initialized to 0.
- Predefined fixed number.

To achieve maximum security and privacy, it is recommended to use real-time varying numbers (like second counter, 100-ms counter, advertising counter and random number) for the nonce counter instead of the fixed numbers (like static random number and predefined fixed number).

Normally, the nonce counter is not advertised in the advertising data. If the selected nonce counter is the Time Stamp 1 (second counter), the receivers may resolve the nonce counter using the algorithm discussed by A. Hassidim, Y. Matias, M. Yung, and A. Ziv, "Ephemeral Identifiers: Mitigating Tracking & Spoofing Threats to BLE Beacons", 2016. To that end, the device supports Advertising of ephemeral identifier (EID).

The resolution of the nonce counter is complicated and requires intensive computation. To bypass these issues, the nonce counter can be part of the advertised data in plain text. Figure 17 and Figure

18 show how to include the nonce counter in the advertising data where the nonce counter is the second counter and the advertising counter, respectively. Please be noted, the "Big Endian" box must be checked for the corresponding item selected as the nonce counter. If the receivers know the nonce counter, then it does not need to be included in the advertising data. For example, if we select to use a predefined fixed number.
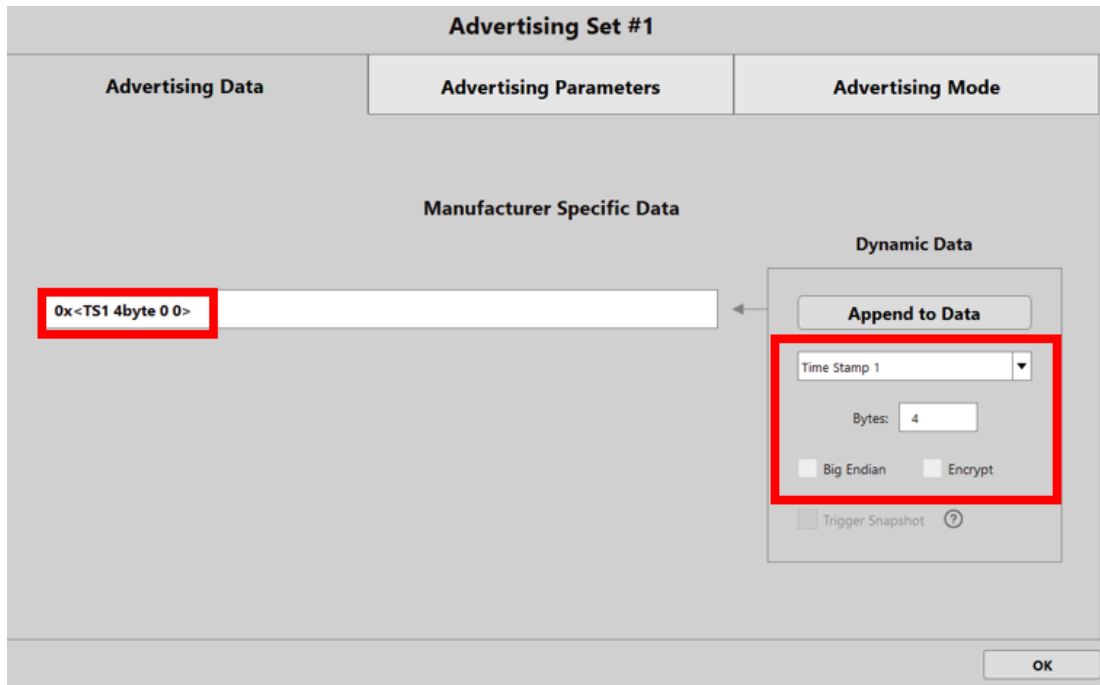

Figure 17 Including the nonce counter in the advertising data: Time Stamp 1
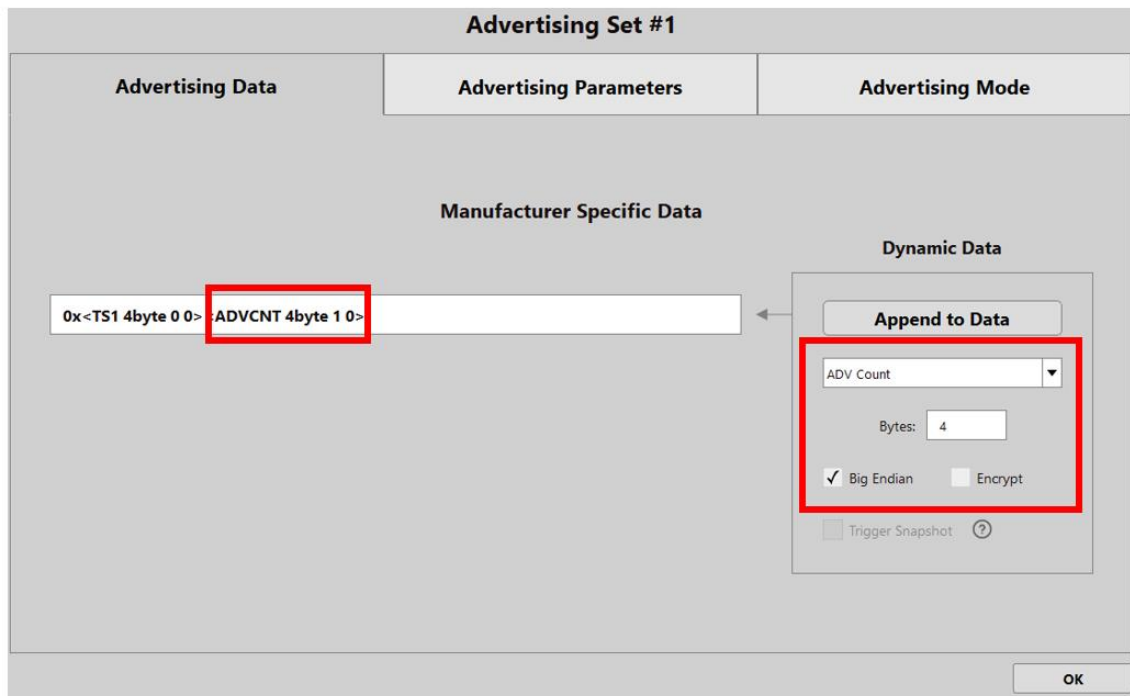
Figure 18 Including the nonce counter in the advertising data: advertising counter

Selection of advertising data to be encrypted

Part of the advertising data can be encrypted and authenticated. In this section, we describe how to configure the data to be included in the encryption and authentication. Please be aware that the device only supports the encryption of a continuous data block. For example, the device does not support the case where Data 1 and Data 3 are encrypted, and Data 2 are not encrypted as shown in Figure 19. The device only supports the encryption of a continuous data block as illustrated in Figure 20.

Data 1 (encrypted)	Data 2 (plain text)	Data 3 (encrypted)

Figure 19 Encryption of a non-continuous data block

Data 1 (encrypted)	Data 2 (encrypted)	Data 3 (encrypted)

Figure 20 Encryption of a continuous data block

Encryption of the predefined data

To encrypt the predefined data, the first step is to select "Encrypt Raw" in the "Extend Data Item" pull-down list, and then enter plaintext (hexadecimal byte data with two characters representing one byte) in the "Data" entry box, as shown in Figure 21.
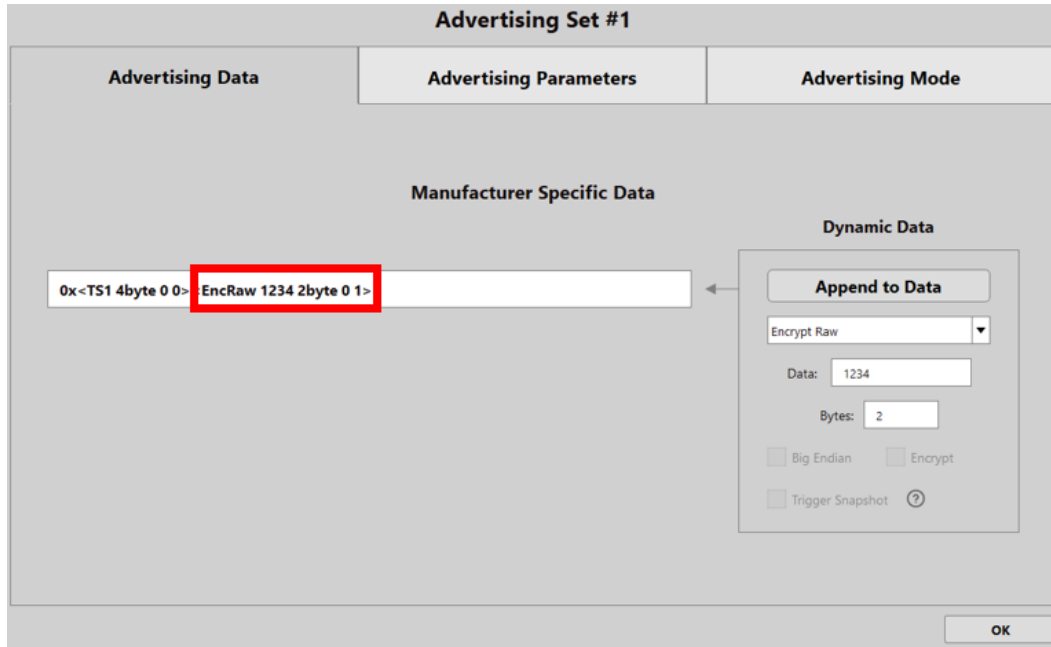
Figure 21 Encrypt the predefined data

Encryption of extended data item

To encrypt an extended data item , check "Encrypt" box when appending that item to the advertising data, as shown in Figure 22.
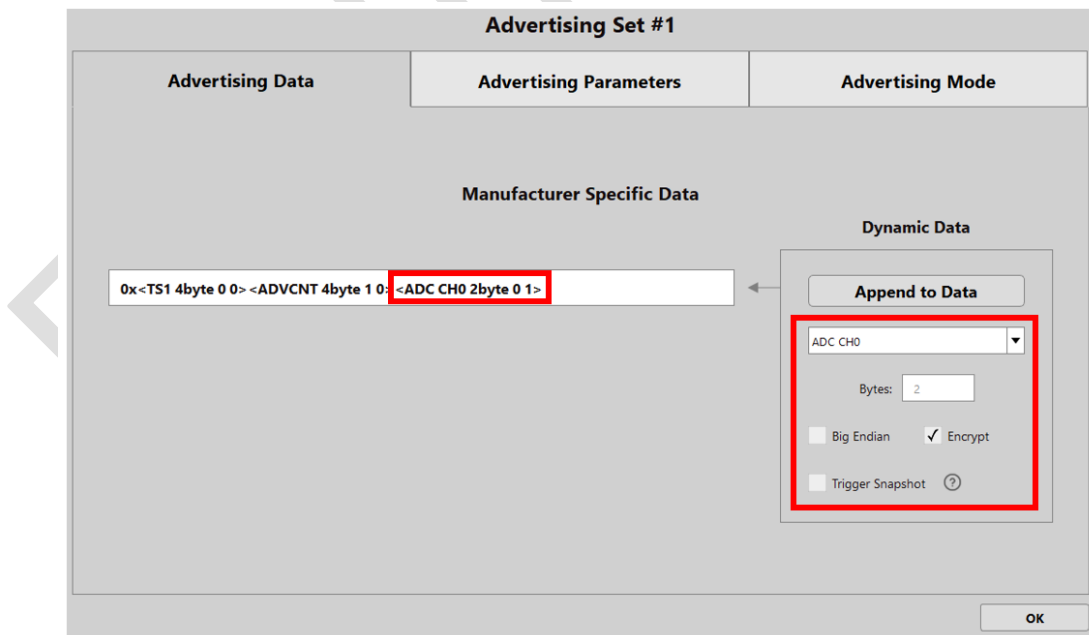

Figure 22 Encrypt an extend data item

Auth Tag

To include the tag in the advertising data, we need to select the "Auth Tag" in the "Extend Data Item" pull-down list, and then append it to the advertising data, as shown in Figure 23. The tag length is configurable from 1 byte up to 8 bytes. If only encryption is needed, and no authentication is needed, then we do not need to add the tag in the advertising data.
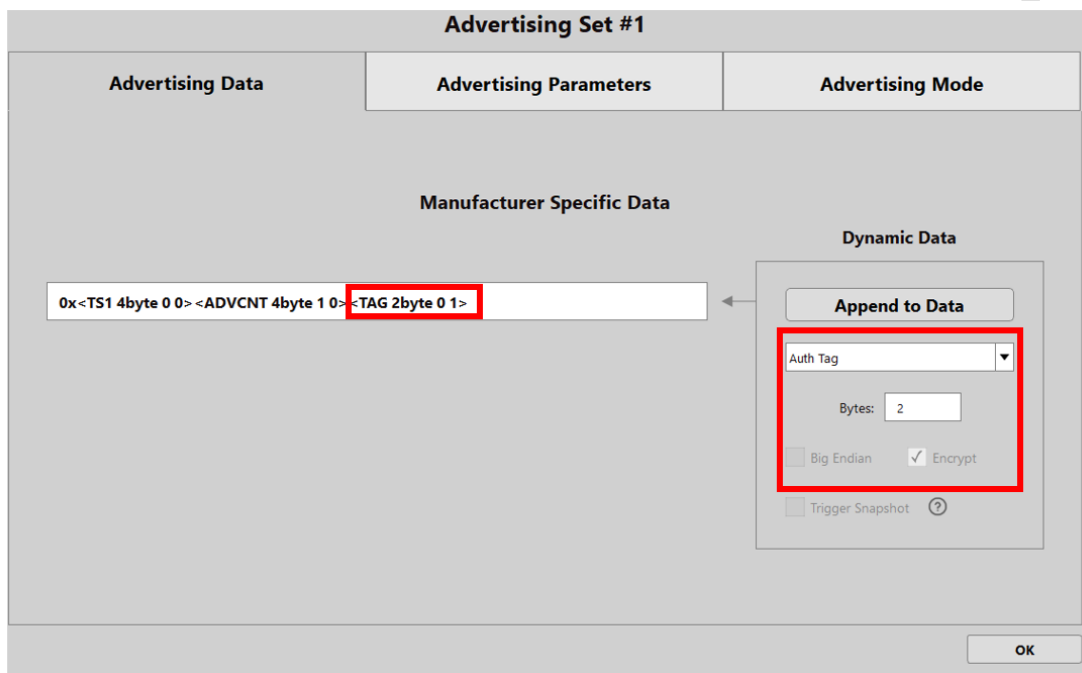

Figure 23 Including a tag in the advertising data

2.1.2.4   Advertising parameters configuration

The user can configure advertising parameters through "Advertising Parameters" tab window including:

- **Advertising Interval**: Used to define the rate at which the Advertising Set will send out advertising packets in milliseconds.
- **PHY Selection**: Defines the PHY used for sending out the advertising data packets. LE 1M is the most used and supported by Bluetooth devices. LE Coded PHY is typically used for long-range communication but is not widely supported by other Bluetooth devices.
- **CTE**: Once checked, the device will automatically add a Constant Tone Extension to the advertising packet. The CTE duration unit defined by Bluetooth Core specification is 8us, and user can choose the CTE duration range from 2 to 20 which equivalent to from 16us to 200us.
- **Advertising Channels**: Determines the RF channels used to send advertising data on. User needs to select at least one channel.

- **Bluetooth Device Address**: Defines what type of Bluetooth Device Address to use in the advertising packets.

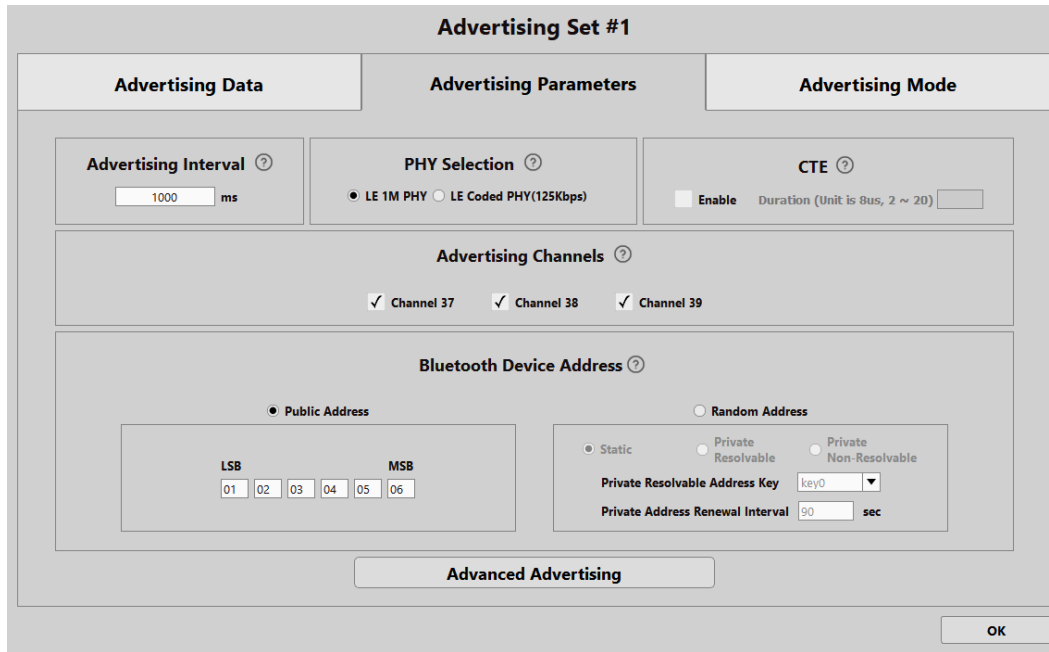An example configuration of the above is as shown in Figure 24.

Figure 24 Advertising parameters configuration

Advanced Advertising

A typical advertising packet's link layer packet format is shown in**Error! Reference source not found.**Figure 25. It consists of a preamble, access address, two header bytes, payload, CRC and CTE (constant tone extension). The header LSB (least significant byte) is configurable through the advanced advertising configuration tab. The header MSB is reserved to indicate the payload length. The payload typically consists of a 6-byte advertising address and a configurable advertising data covered in the other sections in this guideline. The configuration tool provides users with the flexibility to go beyond using standard Bluetooth link layer advertising packet format for their specific application needs.
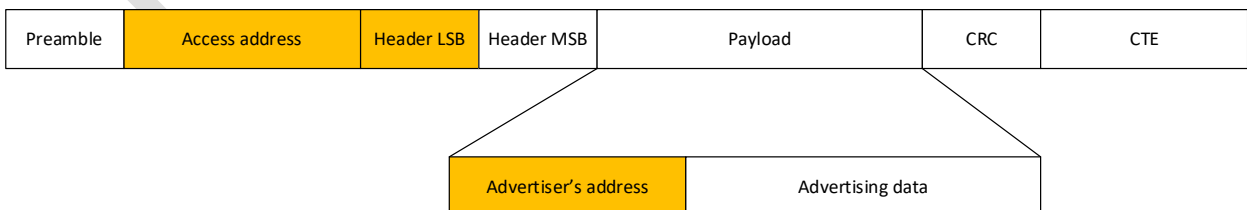

Figure 25 Link layer advertising packet format

In advanced advertising parameters setting tab, as shown in Figure 26, user can define and input their specific packet format.

- **Exclude Bluetooth Address from Advertising**: Once checked the box, the device will remove the Bluetooth address from the advertising packet.
- **Enable Custom Access Address**: User can input self-defined 4 bytes of access address here.
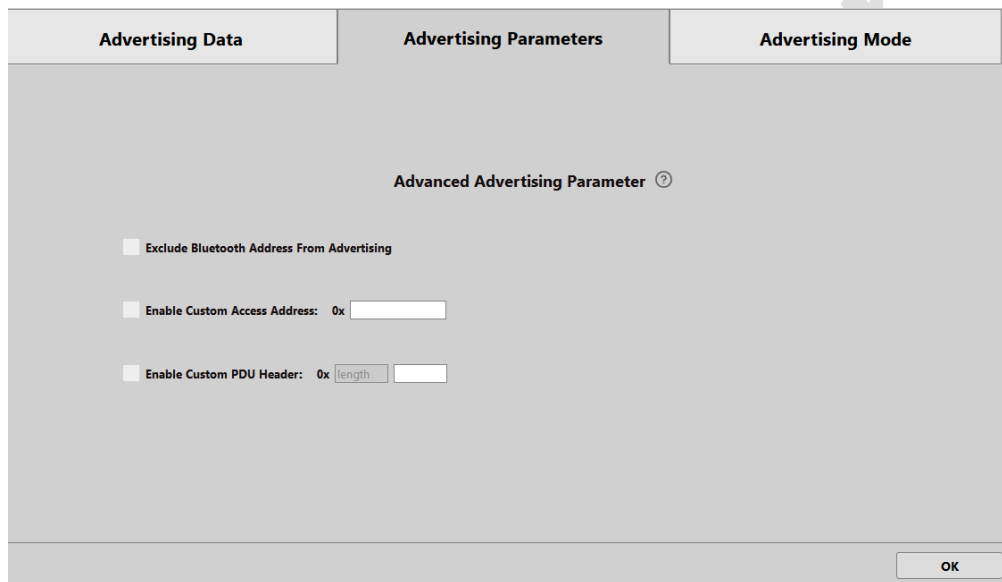- **Enable Custom PDU Header**: 1 byte of header LSB can be defined by user as shown in Figure 25.

Figure 26 Advanced Advertising Parameter Settings

2.1.2.5   *Advertising mode configuration*

NanoBeacon™ device supports two advertising modes: Continuous mode and event trigger mode which can be configured using the GUI according to Figure 27.
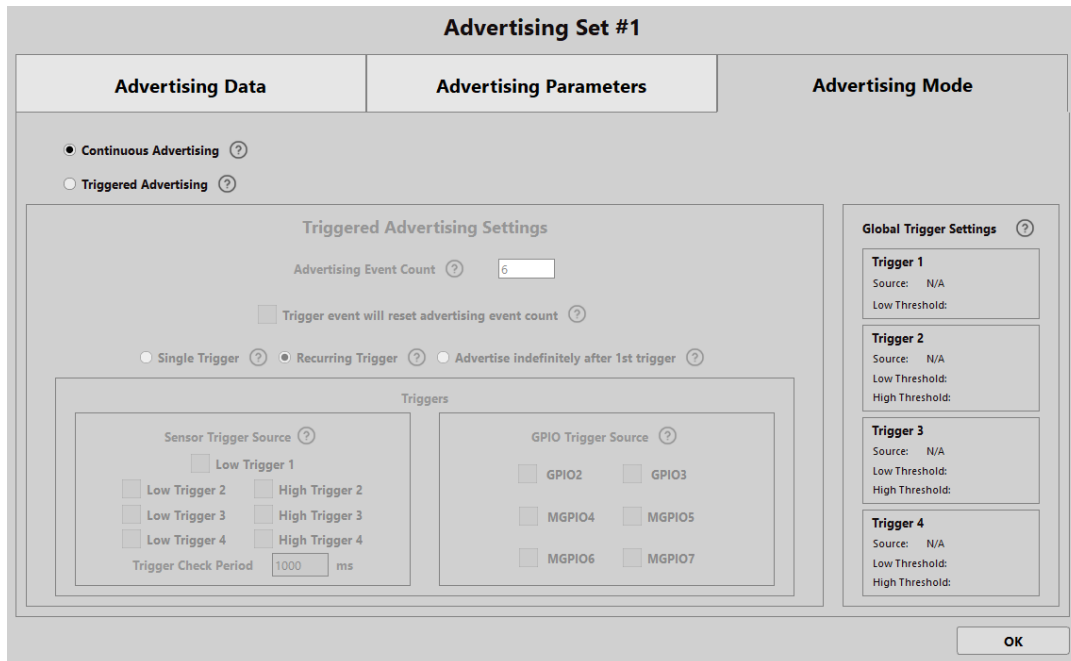
Figure 27 Advertising mode selection

Continuous advertising mode

In the continuous mode, NanoBeacon™ advertises periodically according to the interval set by advertising interval.

Triggered advertising mode

In the event trigger mode, NanoBeacon™ device advertises once an event is triggered according to the trigger parameters. The user can configure the number of advertising events sent out due to the occurrence of a trigger event through entering the "Advertising Event Count". "Trigger will reset advertising event count" enables resetting of the advertising event count whenever a Trigger condition is met at each enabled advertising event.

The device can have multiple sources used to trigger advertising. It also supports different advertising behaviors (modes) after an event is triggered as listed below.

- **Single Trigger**: Configures the device to advertise based on the occurrence of a single trigger event (caused by any trigger condition being met). Subsequent trigger events will not cause the device to advertise unless the device experiences a power reset.
- **Recurring Trigger**: Configures the device to advertise based on the occurrence of a trigger event (caused by any trigger condition being met). The device will initiate the advertising again based on any subsequent trigger events.
- **Advertise Indefinitely after 1st trigger**: Enables the device to continue advertising indefinitely at each advertising interval after any trigger condition is met.

Event trigger source

The event trigger source can be sensor inputs or GPIO states.

i) Sensor inputs

The device totally supports up to 4 sensor inputs as trigger sources (Trigger 1, Trigger 2, Trigger 3 and Trigger 4). User can go to "Global Trigger Settings" to configure them. If the digital samples of selected sensors are over or below the pre-specified thresholds, advertising will start.

Each of senor input trigger source (triggers 1, 2, 3 and 4) can be selected from the followings:

- VCC
  - The VCC is the supply to the device's VCC pin.
  - Low Threshold setting only. The input value unit can be determined through "On-Chip Measurement Units" settings as shown in Figure 28 and Figure 29.
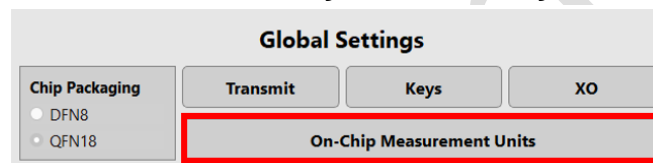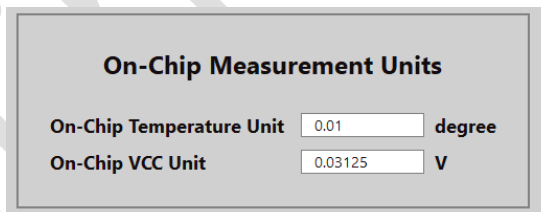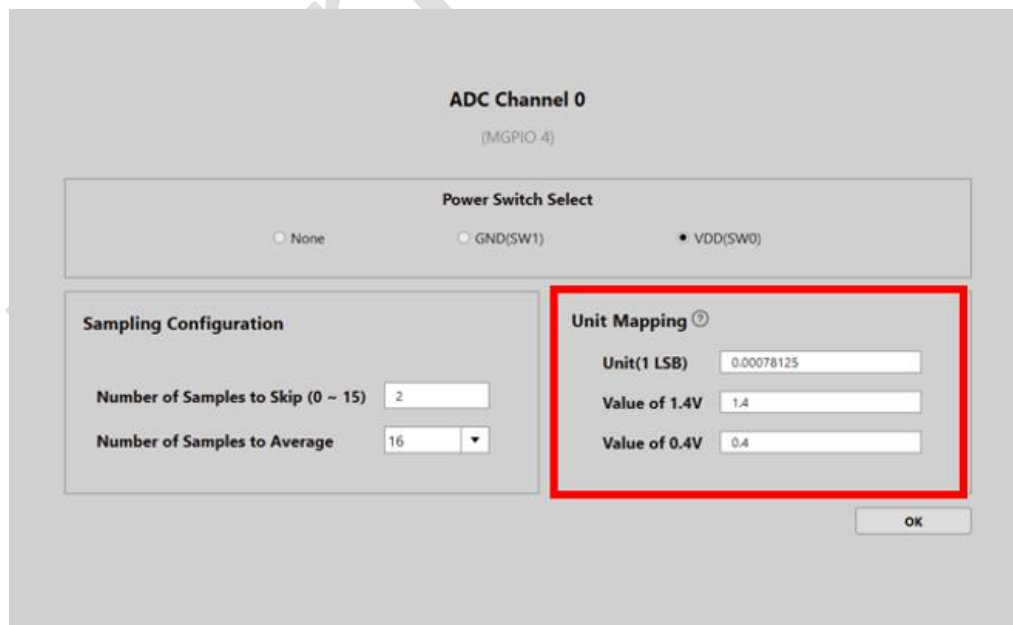

Figure 28 On-Chip measurement units

Figure 29 On-chip temperature and VCC unit setting

  - Threshold value = Desired threshold value / On-chip VCC unit. Integer input only. For example, if user wants the device to start advertising when the VCC is below 2.5V, then the threshold value should be 2.5/0.03125=80.
- ADC CH0
  - This is for the analog channel # 0 connected to the MGPIO-4 pin in the device.
  - The ADC inside the device converts the analog signal (Vin) presented on a mixed signal pin to an ADC code with range from 0 to 2047 according to

$$Raw\ ADC\ code = Vin/Vref * 1024$$

where Vref is the internal reference voltage, and it is equal to 0.8V. The analog input signal's range is from 0 to 2*Vref. **Note:** *The raw ADC code readout will deviate from the desired result due to DC errors such as offset, ambient temperature, differential non-linearity (DNL) and integral non-linearity (INL).*

- By mapping default, the customers can convert the thresholds in voltage to the corresponding ADC code values and enter the ADC values as thresholds in the configuration tool. Assume a customer wish to let the device start advertising when the analog signal on the MGPIO-4 pin is below 0.4V or is above 1.2V. The low threshold would be ADC code = 0.4/0.8*1024 = 512, and the high threshold would be ADC code =1.2/0.8*1024=1536.

- *The device supports linear conversion between two different units of measurements. The ADC of the device measures the voltage from a sensor output. If the customers want to have the advertising data to be broadcasted in a different unit instead of raw ADC code (raw ADC code = Vin*1024/Vref where Vref =0.8V), they can achieve that by appropriately configuring the unit mapping parameters. For example, a temperature sensor outputs 1.4V when the temperature is 100 Celsius degree, and outputs 0.4V when -10 Celsius degree. In addition, they want 1 LSB (least significant bit) in the advertising data to represent 0.1 Celsius degree. In that case, the unit should be 0.1, the value of 1.4V should be 100, and the value of 0.4V should be -10 as shown in* Figure 30. *Please be noted, only linear conversion is supported. When non-default mapping is used, the thresholds to trig advertising should be calculated according to the new mapping. Taking the above temperature mapping as an example, if a customer wants the device to advertise when the temperature is below -8 Celsius degree or is above 90 degree, the low threshold should be -8/0.1=-80, and the high threshold should be 90/0.1=900 where 0.1 is the unit that the customer wants.*
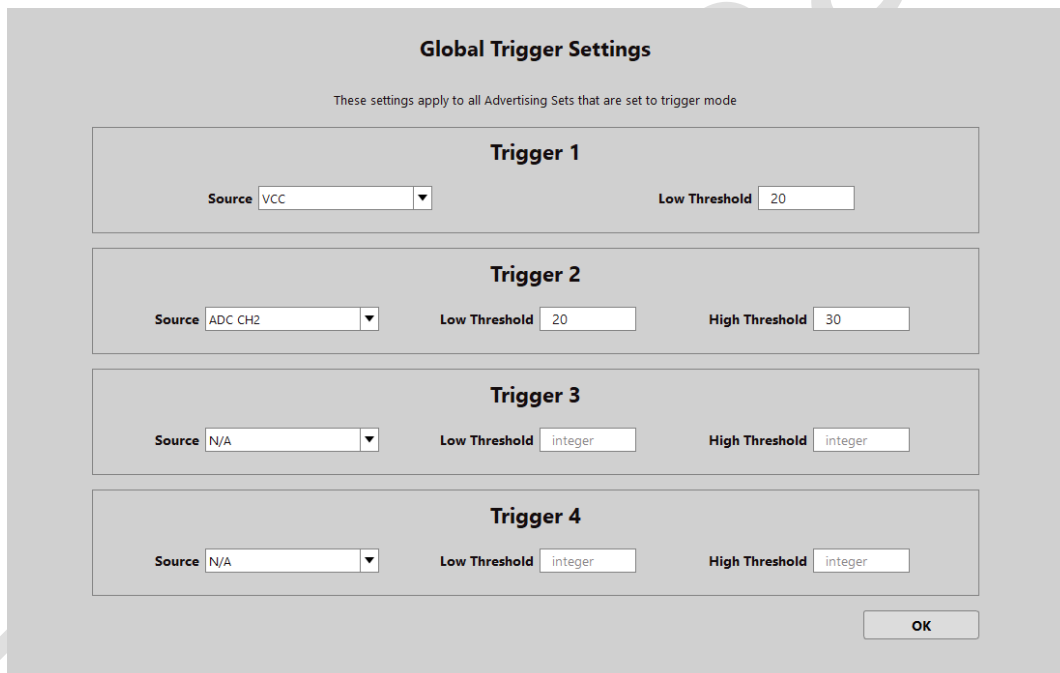

Figure 30 Unit mapping setting

- ADC CH1, ADC CH2 and ADC CH3

- These are the analog channels connected to the MGPIO-5, MGPIO-6, and MGPIO-7, respectively.
- The configurations for these channels are similar to the configuration for ADC CH0.
- 1-Wire Sensor
  - The number of pulses counted received by the device can be entered as threshold value. Integer input only.
- Internal chip temperature.
  - Device integrated an internal temperature sensor and user can set the threshold based on the desired temperature value. The input value unit can be determined through "On-Chip Measurement Units" settings.
  - Threshold value = Desired threshold value / On-chip temperature unit. Integer input only.
- I2C Slave #1 and I2C Slave #2 : Support only 16bit 2's complement format of data.
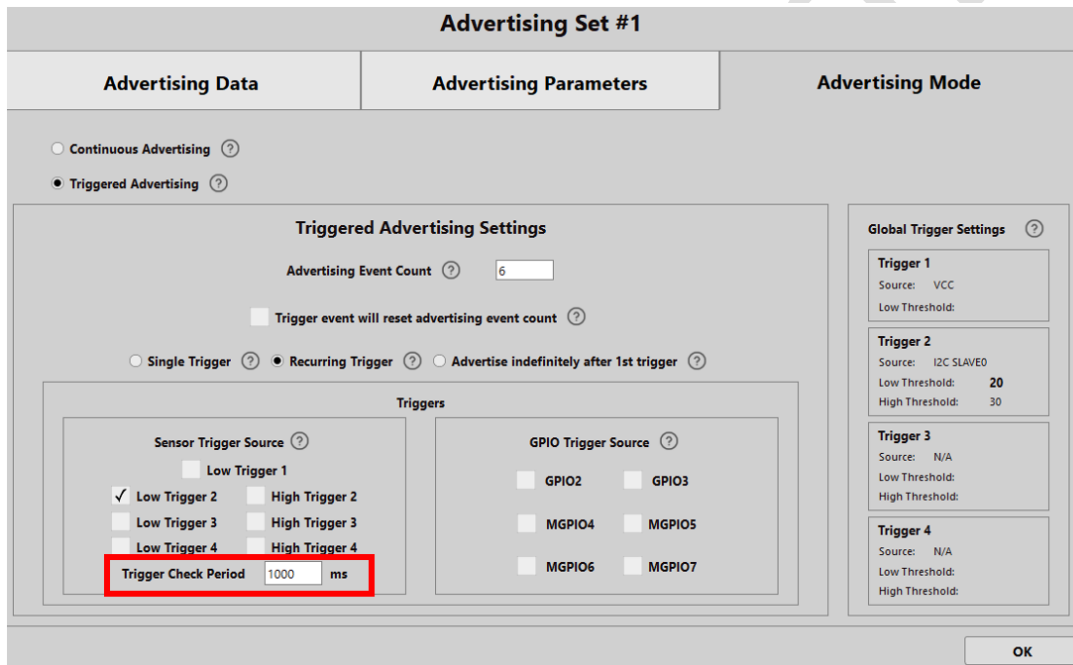

Figure 31 Sensor trigger source

The digital sample of each of the triggers can be configured to compare with certain thresholds and can be configured as shown in Figure 31.

- **Trigger 1 Low Threshold:** Trigger 1 triggers an advertising when the sample is below its set threshold, and Trigger 1 only has a low threshold comparator.
- **Trigger 2 High Threshold:** Trigger 2 triggers an advertising when the sample is above its set threshold.
- **Trigger 2 Low Threshold:** Trigger 2 triggers an advertising when the sample is below its set threshold.

- **Trigger–3 High Threshold:** Trigger 3 triggers an advertising when the sample is above its set threshold.
- **Trigger 3 Low Threshold:** Trigger 3 triggers an advertising when the sample is below its set threshold.
- **Trigger 4 High Threshold:** Trigger 4 triggers an advertising when the sample is above its set threshold.
- **Trigger 4 Low Threshold:** Trigger 4 triggers an advertising when the sample is below its set threshold.

To enable sensor input-based event trigger, we need to set the event monitoring period which is set by the "Trigger Check Period" item of the GUI Tool Trigger Advertising Settings as shown in Figure **32**.

Figure 32 Trigger check period setting

ii) GPIO states

The device supports GPIO level (high or low) or edge (rising or falling) trigger. To enable the GPIO trigger, check the corresponding GPIO as shown in Figure 33. The supported trigger modes are low level trigger, high level trigger, rising edge trigger and falling edge trigger which can be configured in the "GPIO" settings tab. The GPIOs selected for event trigger must be configured as input.
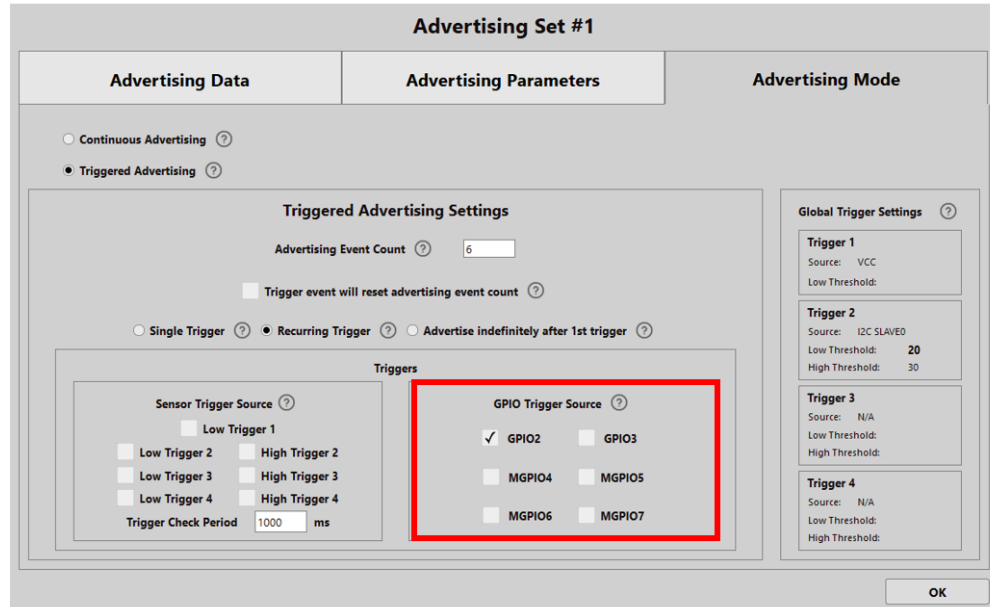
Figure 33 Enable of GPIO trigger

2.2 Analog Channels and ADC configuration

There are 4 analog input channels available in the device as shown in Table 1. The chip's internal ADC has 11 bits, referenced by internal 0.8v bandgap reference. The ADC can be used to digitize the analog inputs to the mixed signal GPIOs.

Table 1 MGPIO mapping to analog input channels

Channel Index	Pin
ADC CH0	MGPIO4
ADC CH1	MGPIO5
ADC CH2	MGPIO6
ADC CH3	MGPIO7

The device provides two load switch pads for external analog device power control and automatic power supply when ADC data acquisition is required, as shown in Figure 34. The switch can be used to shut off the power supply completely to the external sensors when they are not used.
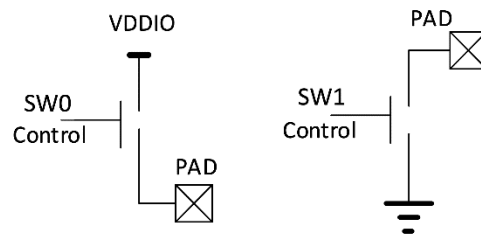
Figure 34 Load switch control

This GUI tool provides a flexible and simple ADC configuration tool. As shown in Figure 35.
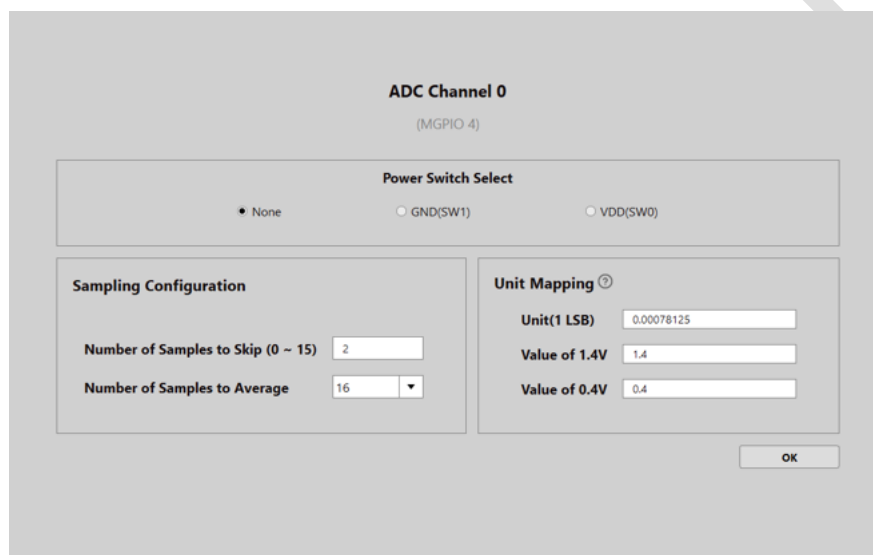

Figure 35 ADC channel configuration

Unit Mapping Calibration

If the data of the external analog sensor is linear with the voltage, the device can directly convert the data into the unit of the analog sensor. If it is not a linear relationship, please keep the data unchanged, and the advertising data (Raw ADC code as shown below) is hexadecimal data in mV.

Raw ADC code = Vin/Vref * 1024, where Vref = 800mV (internal reference)

The default unit mapping is as shown in Figure 35 with unit of 0.78125mV per LSB. For instance, if the advertising data for ADC read out is 0x400 (1024 in Decimal), the Vin is equal to:

Vin = 1024 (ADC code) * 0.78125mV = 800mV.

The user can also configure the Unit Mapping for the sensors which has the linear relationship with ADC read out. And in such case, a linear conversion can be automatically executed on chip for the result to be presented in the advertising data.

For example, a temperature sensor outputs 1.4V when the temperature is 100 Celsius degree, and outputs 0.4V when -10 Celsius degree. And 1 LSB (least significant bit) is used in the advertising data to represent 0.1 Celsius degree. In that case, the unit should be 0.1, the value of 1.4V should be 100, and the value of 0.4V should be -10.
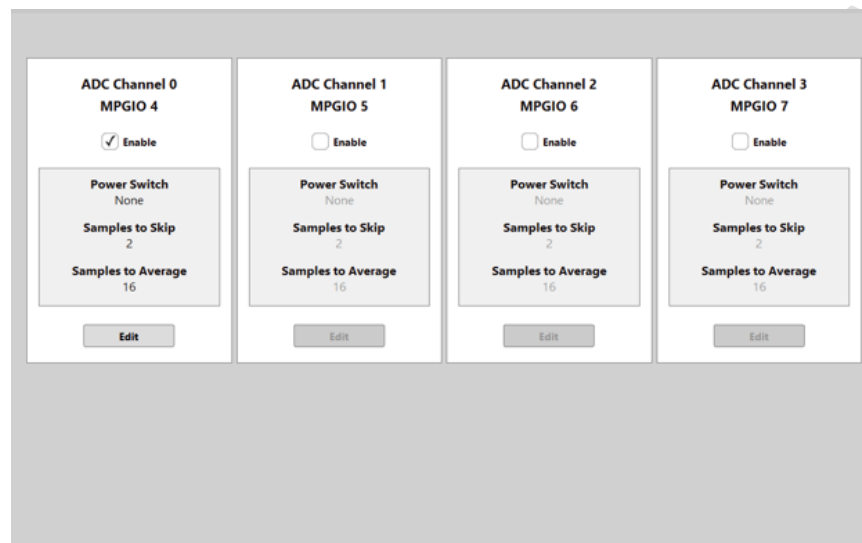

Figure 36 ADC configuration tab

To enable any ADC channel - Check "Enable" box on the corresponding ADC channel(s) as shown in Figure 36.

If the external analog device requires dynamic power control, then click "Edit" button and check the corresponding box according to the circuit design.

Power Switch VDD Enable **(SW0)** or Power Switch GND Enable **(SW1)**

Sampling Configuration: ADC sampling parameters, generally use the default parameters. In special cases, the parameters such as "Number of Samples to Skip" and "Number of Samples to Average" are adjusted according to the external analog device.

When "Enable" is checked for certain ADC channel, the digital GPIO function corresponding to the multiplexed pin needs to be turned off, and pull-up and pull-down need to be turned off. See the *GPIO Configuration* chapter for details.

2.3   GPIO edge count

The device supports counting of rising edges on a selected GPIO, as shown in Figure 37. The rising edge counting can be performed even when the device is in deep-sleep. The count length supported is up to 3 bytes, and customers can have the count shown in the advertising payload. Both the minimum time of tH and tL (shown in Figure 37, the durations of the input signal on high level and low level, respectively) must be larger than 1 ms.
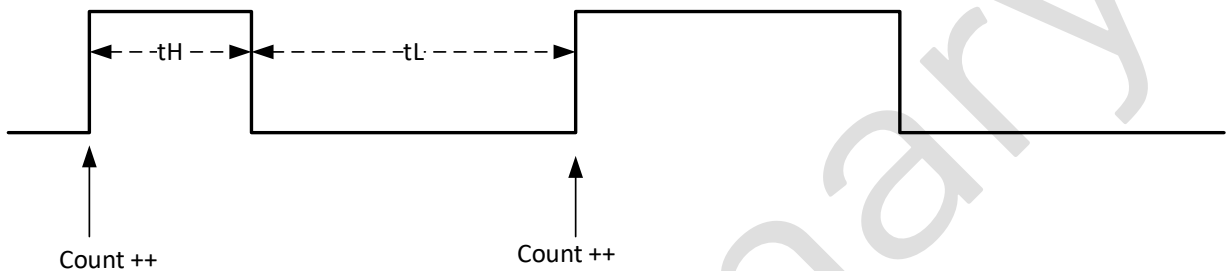

Figure 37 GPIO edge counting

User can configure the input GPIO source through GPIO Edge Count configuration tab as shown in Figure 38. Such count can be used as dynamic data to be advertised through configuring of the Advertising Data settings.
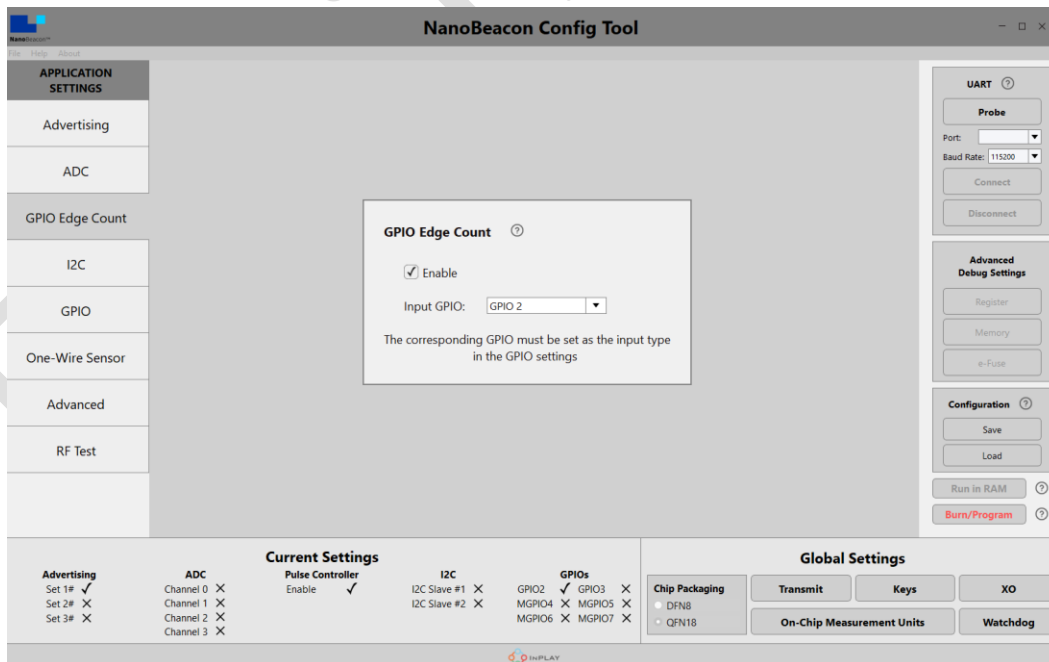

Figure 38 GPIO Edge Count Configuration

The input GPIO that the count event to be monitored at can be selected from the list provided. Please note that the selected GPIO need to be set as "Input" from GPIO configuration tab.

2.4   One-wire sensor interface configuration

The device integrates a one-wire (single-wire) pulse count controller and interface which can be used to interface a one-wire pulse sensor. There are many sensors which convert the measured physical quantity into number of pulses. The one-wire controller can provide power management and 16-bit pulse counting at the same time. Figure 39 shows the timing of power management and pulse counting sequence. Typical conversion time is from 30 to 60 ms, and pulse output time is from 30 to 60 ms (please check out the one-wire sensor that is being used). In the figure, tp is the period of a pulse, typically from a few us to a few tens us. The total time and the maximum period of a pulse can be configured by the configure tool. The device has two power switches which can be used for the power management of the sensor.
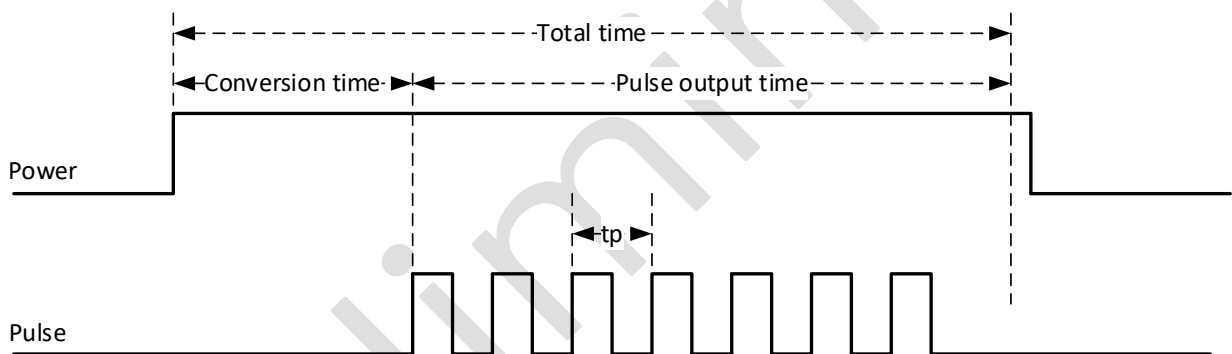

Figure 39 Power management and pulse counting

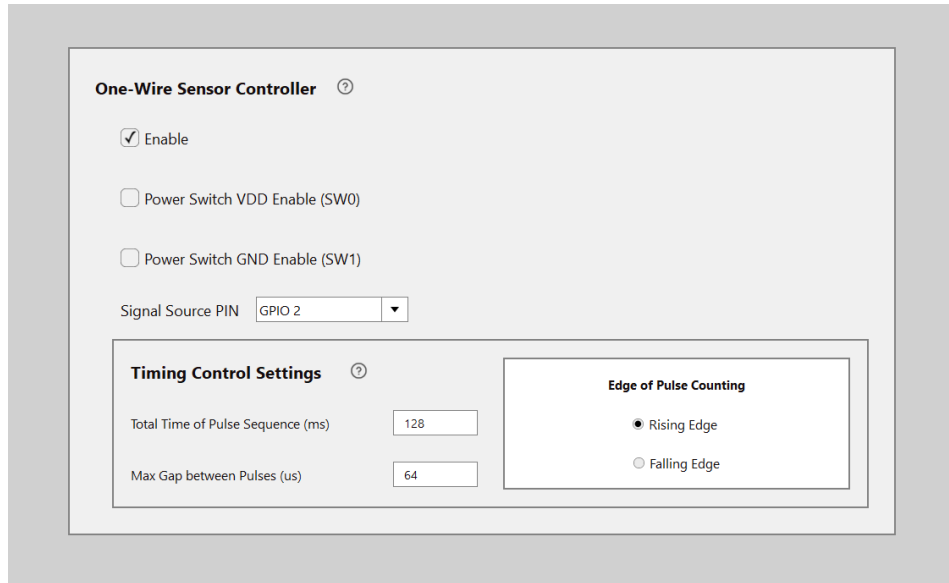The one-wire pulse count related settings are shown below.

Figure 40 One-wire count configuration tab

To enable: check "Enable" box
If the external sensor needs dynamic power control, then check the corresponding box according to the application circuit design.

Power Switch VDD Enable **(SW0)** or Power Switch GND Enable **(SW1)**

Signal Source PIN selection, select the corresponding GPIO according to the hardware circuit
Timing Control Settings parameters are adjusted according to the external sensor parameters.

2.5 I2C configuration

NanoBeacon Config Tool provides the user an easy way to configure the external I2C slave devices through I2C configuration tab window as shown in Figure 41.
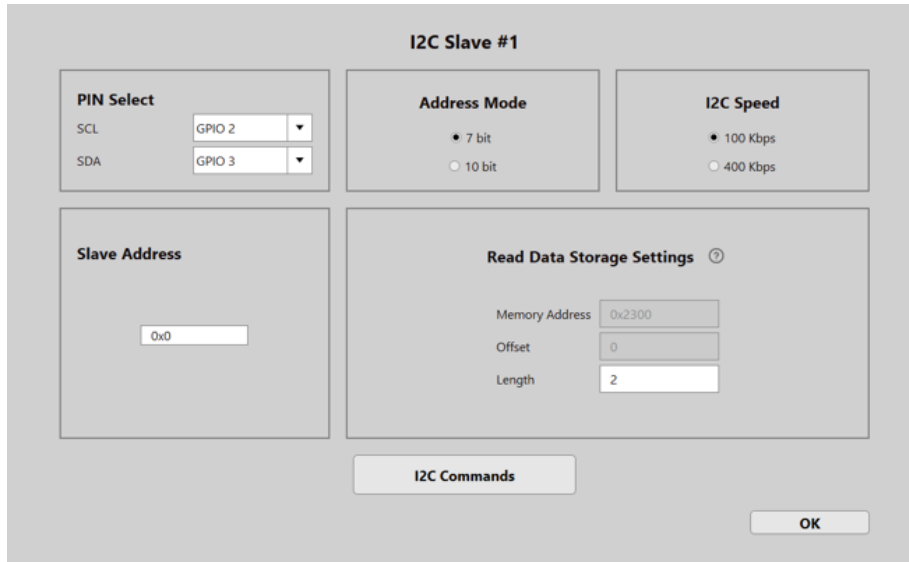
Figure 41 I2C configuration tab

The user can configure the following items related to I2C salve device.

- PIN Select: Select the pins corresponding to SCL and SDA of I2C devices according to the actual circuit. The Device's GPIO2/3/4/5/7 can be used as SCL and SDA of I2C interface.
- Address Mode: Supports 7bit and 10bit mode.
- I2C Speed: The device supports 100KHz and 400KHz clock rate.
- Slave Address: The user can enter the I2C slave device address in the box.
- Read Data Storage Settings: I2C read will store the data into the device's internal memory with the starting address at 0x2300; Offset is the offset of the address, the default setting is 0 for I2C Slave #1, 16 for I2C Slave#2 and 32 for I2C Slave #3 respectively; Length is the length of the I2C data read out memory length, and the unit is byte; I2C read data will be automatically wound up when the length exceeds the defined "Length".

I2C Commands

To drive external I2C slave device, an I2C communication protocol shall be defined and configured for both devices to establish the I2C communication with each other. The NanoBeacon Config Tool provides the "I2C Commands" input tool for such purpose.

Figure 43 shows the main configuration window for I2C Commands input. There are 4 commands available:
- I2C read: For this command, we need to specify the number of readings (r_len) from the slave in the GUI. The maximum r_len is 5. If more than 5 bytes read needed, the user needs to configure I2C reads multiple times separately.

- I2C write: For this command, we need to specify the byte sequence written to the slave in the GUI. If more than 5 bytes need to be written, then the user needs to configure I2C writes multiple times separately.
- I2C write_stop_read: For this command, the user need to specify the byte sequence to be written to the I2C slave device as well as the number of readings (r_len) from the slave in the GUI.
- Delay command: the users can use this command to insert delay or wait time between I2C commands, or delay/wait time before an I2C command.

The execute condition for the above commands can be configured individually using the check box "Execute I2C command when cold boot" (the cold boot only box) or "Execute I2C command when warm boot"(the warm boot box).

In typical application, the device switches its state between wakeup/active and sleep, as shown in Figure 42 . The first time going into active upon power-on is called cold boot, and the wakeups after sleep are called warm-boot.
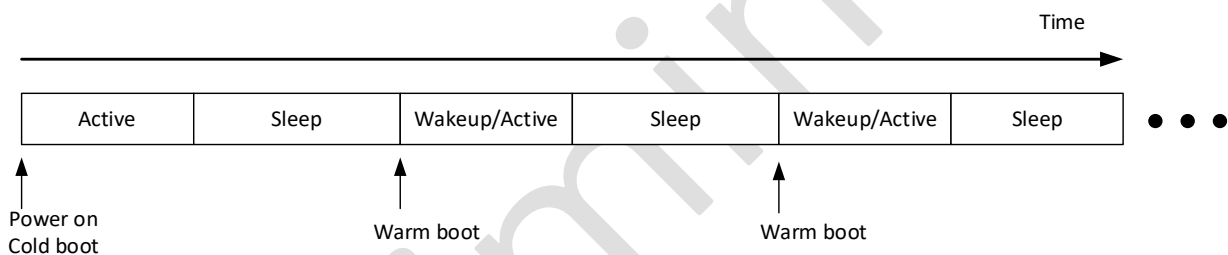

Figure 42 IN100 states

Execute I2C command when cold boot: Execute I2C read/write command when cold boot. If only this execution condition is checked, then the corresponding I2C command will be executed only once when the chip is cold booted, which is often used for the initial configuration of the I2C device.

Execute I2C command when warm boot: Indicates that the chip executes the I2C read/write command described above when it goes from sleep to wake up. When the device does not advertise data, it tends to go into sleep state, and when it needs to advertise, it first wakeup and then advertise. The loop flow is wakeup -> advertise -> sleep. So, it can be simply understood that if this option box is checked, then the corresponding I2C commands will be executed once for each wakeup event.

If the commands are only needed to be executed during cold boot, then we need to check the cold boot only box.
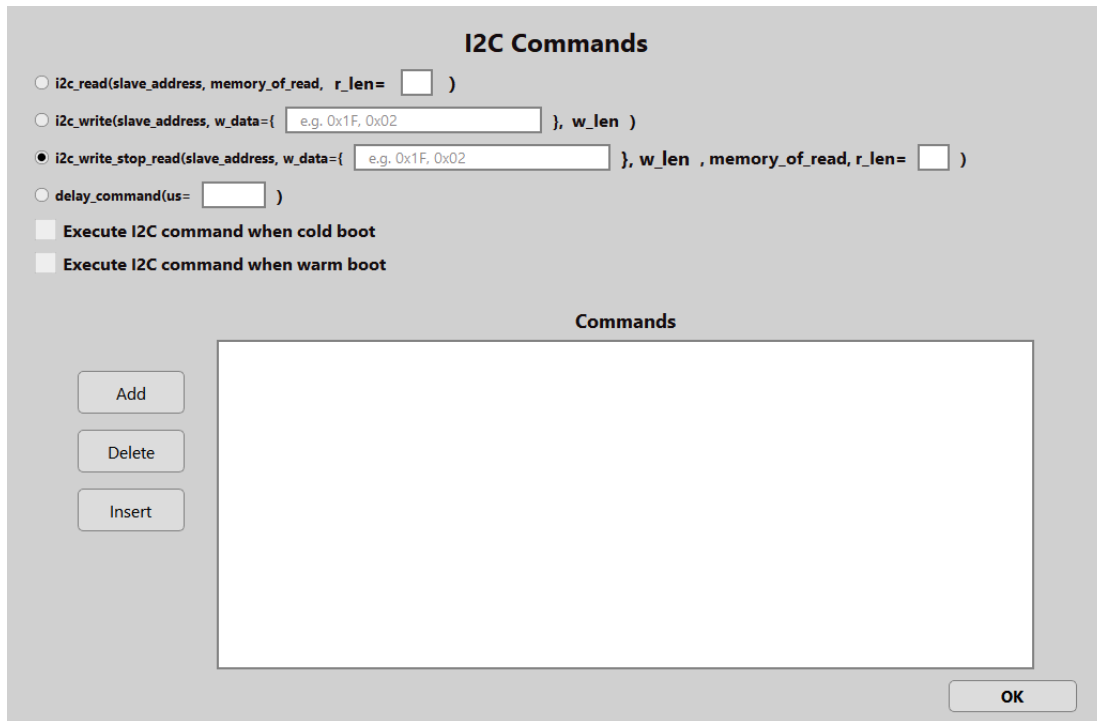
Figure 43 I2C commands configuration window

The steps to configure I2C are listed below:
- Select a command, and enter the details for that command
- Select the execution condition (by checking/unchecking the cold boot only box)
- Click the "add" button to add the command to the configuration.
- Repeat the above process, adding another command as needed.

The data read from the external sensor via the I2C Rx command will be stored in the memory at the location offset specified in the "Read Data Storage Settings" parameter definition section of the corresponding I2C slave device.

2.6 GPIO configuration

The GPIO configuration tab window is as shown in Figure 44.

Figure 44 GPIO configuration tab

The GPIO options are input, output, and disable. If the GPIO pin is used as an analog input to the ADC, you must select disable to turn off the digital function. If the GPIO pin is used as an input source for the pulse controller, then input must be selected, and the default is input.

Pull-up options are pull up, pull down and disable, which correspond to pull up, pull down and neither pull up nor pull down.

Disable must be selected if the GPIO pin is used as an analog input to the ADC, otherwise select as needed. The default is pull-up.

Adv. Trigger options are disable, high level, low level, raising edge and falling edge, which correspond to off trigger, high level, low level, rising edge and falling edge trigger advertising. Use with the GPIO Trig Source on the Advertising Data Settings configuration window. To enable GPIO ADV trigger, the corresponding GPIO needs to be set to input.

Wakeup has disable, high level and low level. disable is not to enable GPIO wakeup, high level is to wake up the chip when GPIO is high level, low level is to wake up the chip when low level.

Latch is used to keep a GPIO state when the device going sleep and out of sleep. By default, when the device is in sleep, the GPIO's default state is in "high-Z" state, and upon wakeup, its default

state is input with pull-up. For most of application cases, the ConfigTool handles the latch automatically, and users do not need to do anything. Some typical cases are:

- If a mixed GPIO is configurated as an analog channel input pin, that GPIO will be automatically disabled, and latched.
- If a GPIO is configured as an input wakeup pin, that GPIO will be automatically latched at input mode.

If a GPIO is configured as an output low or output high, and users want that GPIO to keep the state during sleep, users should enable latch for that GPIO.

2.7 Advanced Settings

Advanced mode settings are available for users when the regular GUI tool configuration options do not meet their specific application needs. A special support channel is available through the advanced register settings provided by InPlay support engineers. Always consult the InPlay support team before making any changes in this mode. Changes made here may override settings in other parts of the tool.
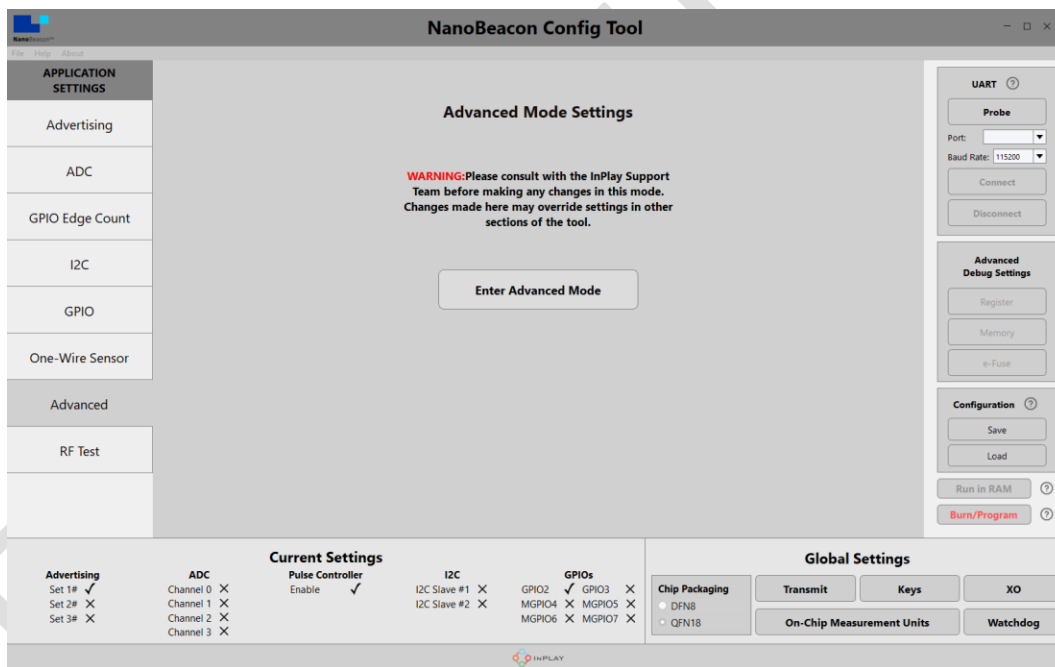

Figure 45 Advanced mode settings

As shown in Figure 45, users can follow the instructions of InPlay engineers to enter the registration command and then add it to the settings box. Then click the OK button to exit.

2.8 RF Test

The GUI tool provides an RF test configuration window that allows users to directly access the device's Direct Test Mode (DTM) and RF carrier test mode as shown in Figure 46.
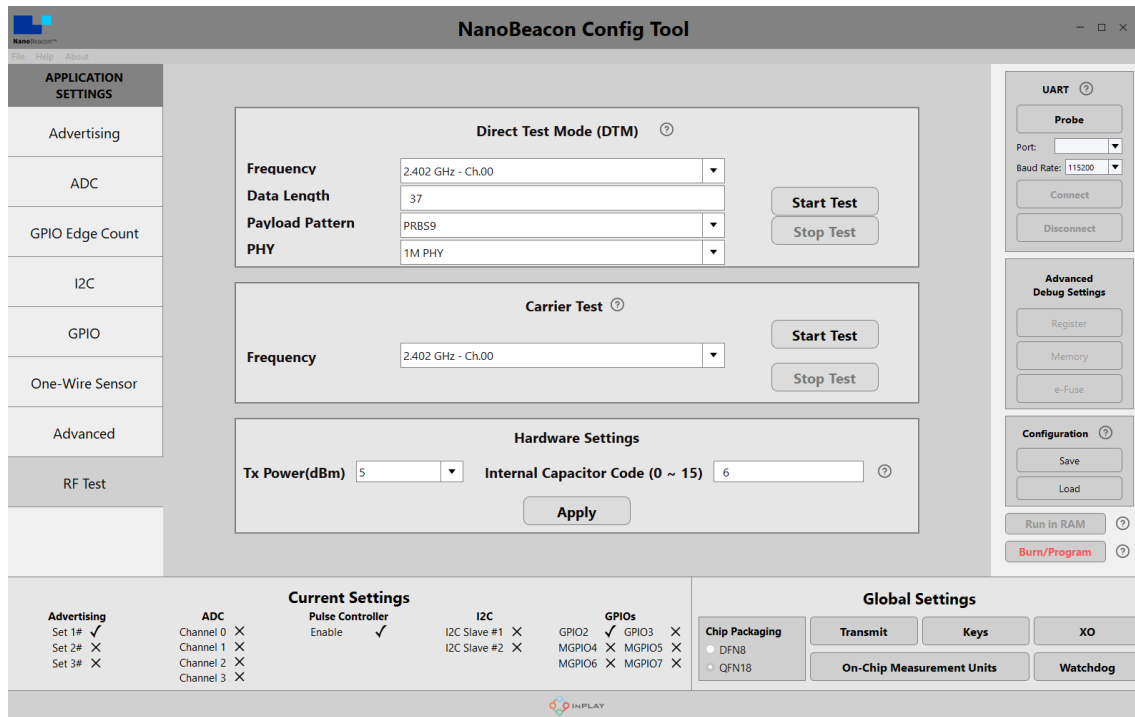

Figure 46 RF test configuration

Direct Test Mode (DTM)

The RF validation of Bluetooth device uses a protocol called Direct Test Mode as described in the Bluetooth Core Specification version 4.x and 5.0, Volume 6, Part F. The purpose is to test the radio at the physical layer of the DUT (Device Under Test) at the specified transmit power for PHY and protocol compliance testing.

User can use the Config Tool to choose the Frequency, Payload Pattern and PHY from the list provided with Data Length entered.

Carrier Test

The RF carrier test measures the RF carrier leakage relative to the modulated output signal. To perform the test, the user can measure the channel center frequency level, relative to a predetermined maximum output level in the DUT setup.

Hardware Settings

The user can enter the desired Tx output power in the Tx Power (dBm) box. User can also enter the XO Load Capacitance value to minimize the transmit channel frequency offset.

3. OTP memory (eFuse) programming and debugging

OTP Memory (eFuse) Programming

IN100 OTP memory programming: VDDQ pin (PIN6 for DFN8 and PIN13 for QFN18) is required to be powered by 3.3VDC, otherwise they will not be programmed correctly.
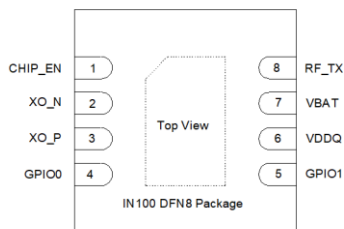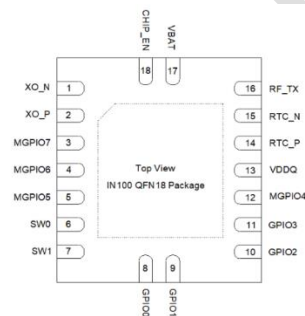

Figure 47 DFN8 package

Figure 48 QFN18 package

Below is a programing example with NanoBeacon™ Development Board (aka IN100 Eval Board).

NanoBeacon™ Config Tool connects to the OTP memory programmer board via USB. The NanoBeacon™ development board is connected to the OTP memory programmer board via a 10-pin connector. The connection is shown in the figure below.
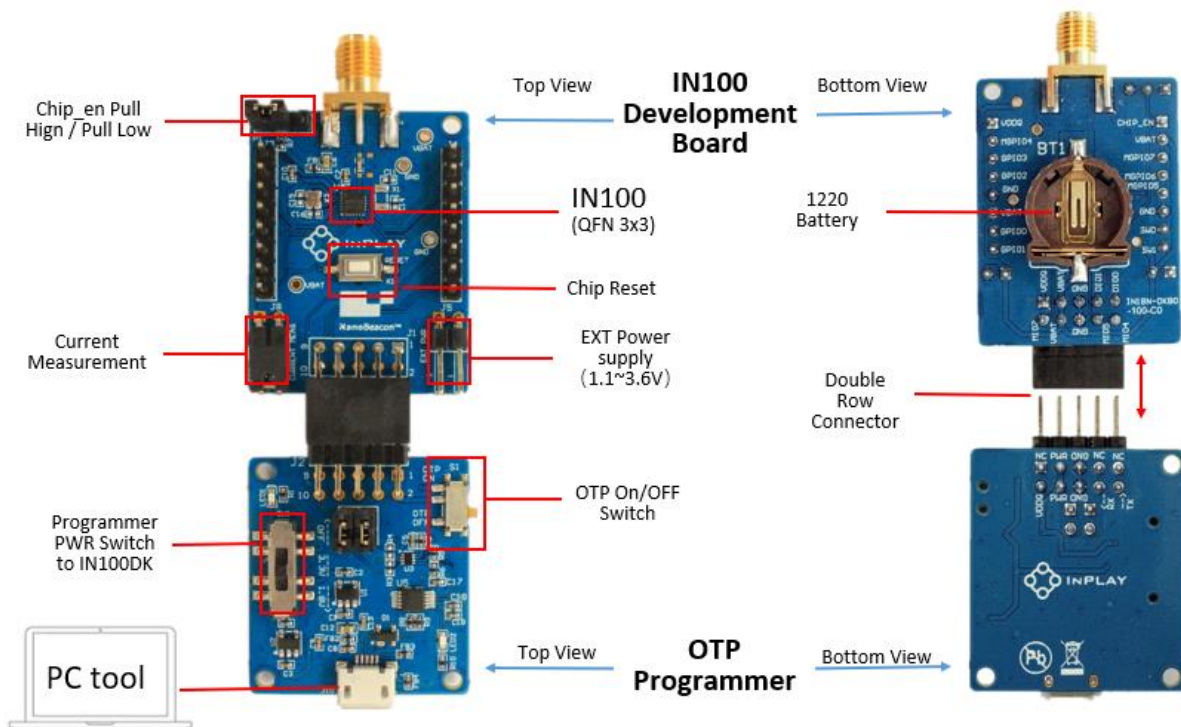
Figure 49 Programming with NanoBeacon™ config tool

Once the board is connected to the computer, open NanoBeacon™ Config PC Tool and click the Probe button in the UART area on the right side of the interface to get a list of available serial ports. Then select the corresponding port number in the Port drop-down box. Baud rate Baud uses the default 115200 . Then click the Connect button to establish a connection to the device. As shown in the figure below, whether the connection is successful or failed, there will be a pop-up box.
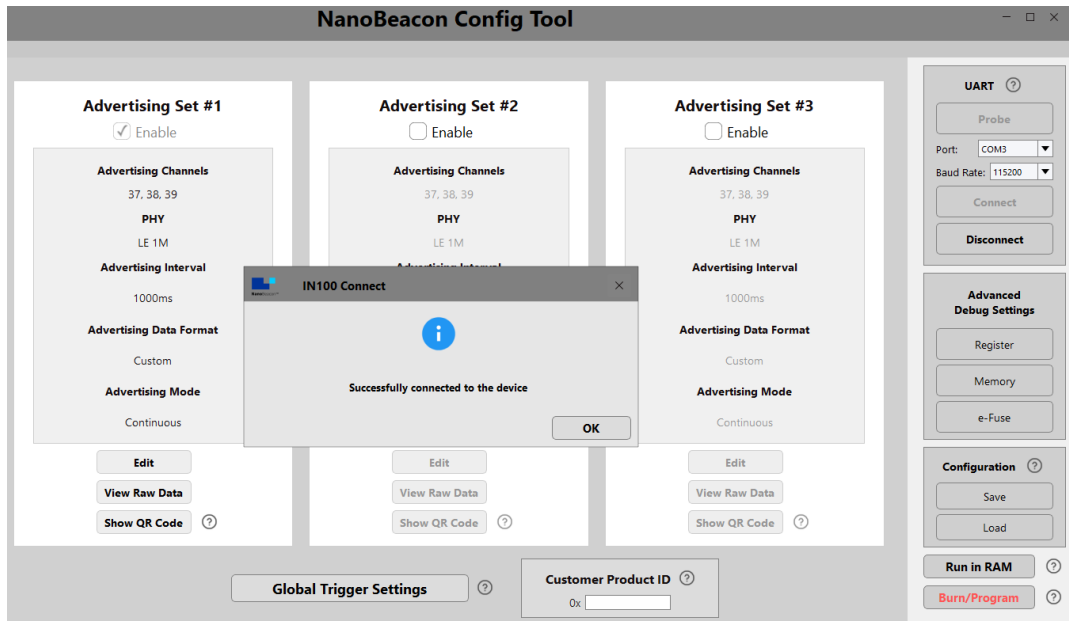
Figure 50 UART communication status report dialog

Debugging

Support writing data to RAM and Run in RAM mode debugging. Note that I2C does not support RAM debugging.

Once the advertising data and peripherals are configured and the connection with the device is established, user can first load the data into RAM for testing by clicking the "Run in RAM" button on the right side of NanoBeacon™ Config tool to write the configuration data into RAM. Then use the Bluetooth advertising scanning software to scan the data advertised by the NanoBeacon™ to confirm whether the advertising data is correct or not.
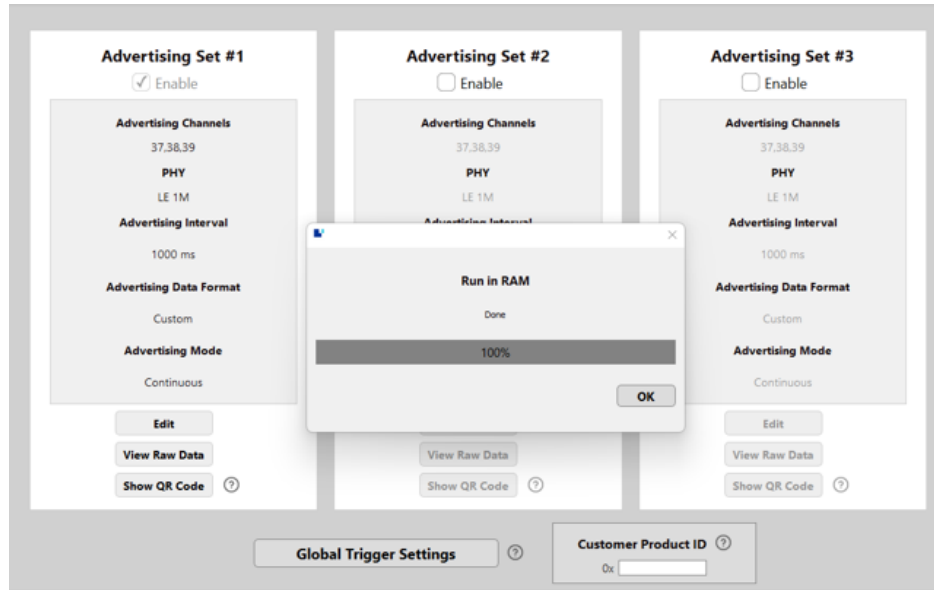
Figure 51 RAM run mode status report

OTP memory burn-in

After debugging and verification, user can program the configuration to the device with confidence.
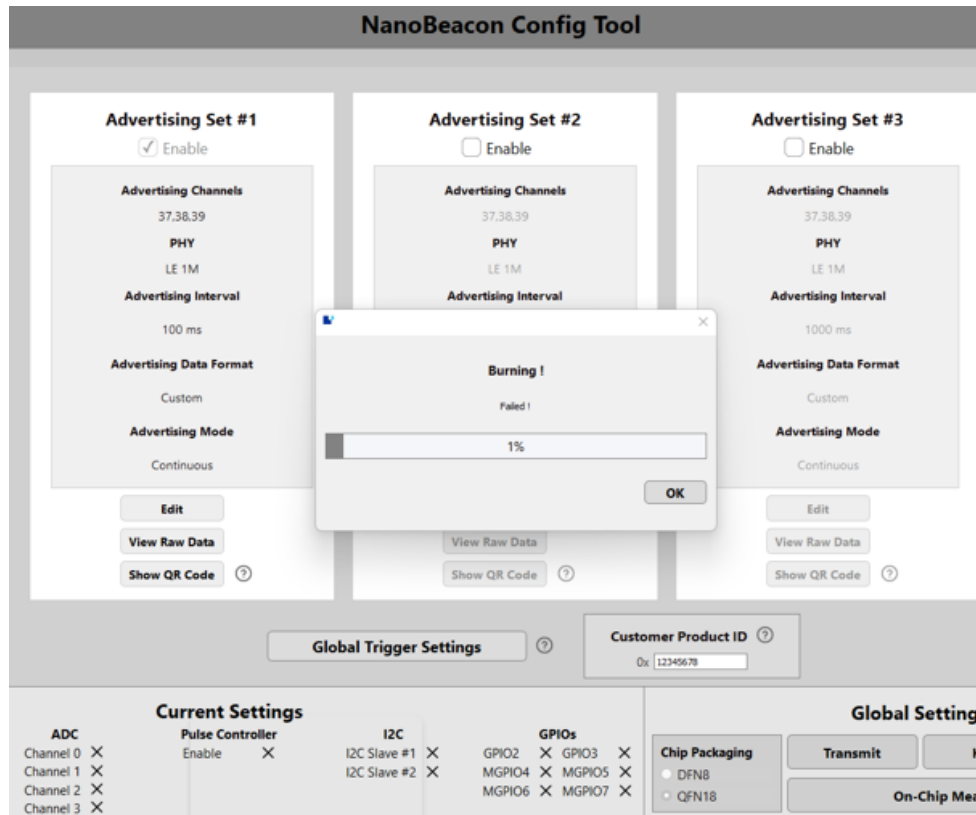
Figure 52 Burning in progress

Power up the NanoBeacon development board again and click the "Burn/Program" button on the right side of NanoBeacon™ Config tool to perform OTP memory burn. There will be a progress bar during the burning process, as shown in Figure 52, and there will be a pop-up box for successful burning report or a pop-up box for failed burning report just in case.
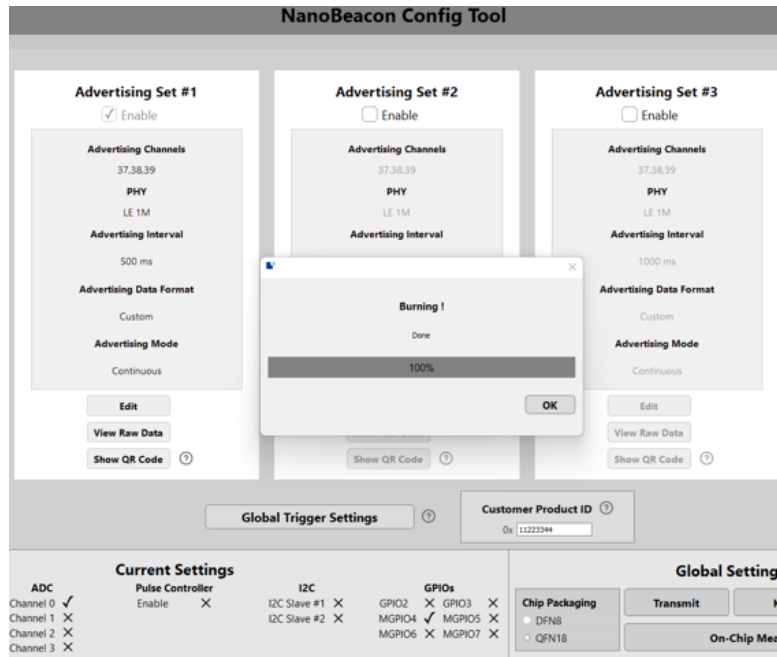
Figure 53 Programming progress complete

After burning, remove the OTP memory programmer board, install the battery onto the NanoBeacon development board and the board shall operate normally.

Note: Once burned, the device cannot be erased and burned again.

4. Revision History

Revision	Description	Prepared By	Date
Ver 0.9	Preliminary version	Y. Gu	2022-03-22
Ver 1.02	Typo and graphic fixes	J. Wu	2022-05-05
Ver 1.03	Updated I2C and GPIO configuration	Y. Gu	2022-05-31
Ver 1.04	Added CTE configuration.	J. Wu	2022-06-22

5. Disclaimer

InPlay has made every attempt to ensure the accuracy and reliability of the information provided on this document. However, the information is provided "as is" without warranty of any kind. The content of the document will subject to change without prior notice. InPlay does not accept any

responsibility or liability for the accuracy, content, completeness, legally, or reliability of the information contained on this document. We shall not be liable for any loss or damage of whatever nature (direct, indirect, consequential or other) whether arising in contract or otherwise, which may arise as a result of your use of (or inability to use) this document, or from your use of (or failure to use) the information on this document. InPlay Inc and its company logo are registered trademarks of InPlay Inc with its registered office at 1 Technology Drive, STE J728, Irvine, CA, USA 92618.

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

[SparkFun Electronics]():
  [WRL-21327]() [WRL-21293]()