InPlay NanoBeacon™ IN100 Config Tool User Guide

NanoBeacon™

Specifications and information are subject to change without notice.

About Documentation

Document Name	IN100 Config Tool User Guide	
Part number	IN100	
Control Number	IN1IDOC-SW-IN100-EN-V1_06	External User
Revision	V1.08	

This document applies to the following products:

Document number	Applicable Products	Document Status
IN100	IN100-D1-R-YC1I	Mass production
	IN100-Q1-R-YC1I	Mass production
	IN100-Q1-R-YC1F	Mass production
	IN100-W10-R-SC1I	Mass production

Contents

About Documentation 1

1. Introduction 6

1.1 Things you will need 7
1.2 How to configure and use the NanoBeacon 8

2. NanoBeacon Config Tool 9

2.1 Advertising sets configuration 10
2.1.1 Overview 10
2.1.2 Advertising data format 12
2.1.3 Advertising data configuration 14
2.1.4 Advertising data encryption and authentication 17
2.1.5 Advertising raw data review 24
2.1.6 Advertising parameter configuration 25
2.1.7 Advanced advertising parameter configuration 26
2.1.8 Advertising mode configuration 27
2.1.9 Trigger source configuration for triggered advertising 29
2.2 Analog channels and ADC configuration 32
2.2.1 Overview 32
2.2.2 Configuration 33
2.3 GPIO edge count 36
2.4 I2C configuration 37
2.5 GPIO configuration 41
2.6 One-wire sensor interface configuration 43
2.7 Square wave settings 44
2.8 Advanced Settings 47
2.9 XO Settings 48
2.10 RF Test 50

3. Programming OTP Memory 52

3.1 Before you start 52
3.2 Verification 54
3.3 OTP memory programming 55

4. Revision History 57

5. Reference 57

6. Disclaimer 57

List of Figures

Figure 1 : A typical devlopment envionment for IN100 devices 6
Figure 2 : NanoBeacon development kit (front and back view) 7
Figure 3 : Tag board powered by a coin-cell battery 9
Figure 4 : Overview of NanoBeacon config tool 9
Figure 5 : Advertising sets configuration overview 11
Figure 6 : Advertising set configuration window 12
Figure 7 : Advertising data format 12
Figure 8 : Customer-defined advertising data format 13
Figure 9 : Data segments for AD data/user defined data section 14
Figure 10 : On-chip temperature and VCC unit setting 16
Figure 11 : Selecting extended data 17
Figure 12 : Advertising data encryption and authentication 18
Figure 13 : Encryption enabling 18
Figure 14 : Selection of key and nonce for AES-EAX for each advertising set 19
Figure 15 : Configuration of encryption keys 19
Figure 16 : Including the AES-EAX salt in the advertising data 20
Figure 17 : Including the nonce counter in the advertising data: time stamp 1 21
Figure 18 : Including the nonce counter in the advertising data: advertising counter 22
Figure 19 : Encryption of a non-continuous data block 22
Figure 20 : Encryption of a continuous data block 22
Figure 21 : Encrypt the predefined data 23
Figure 22 : Encrypt an extend data item 23
Figure 23 : Including a tag in the advertising data 24
Figure 24 : Raw advertising data review window 25
Figure 25 : Advertising parameters configuration 26
Figure 26 : Link layer advertising packet format 26
Figure 27 : Advanced advertising parameter settings 27
Figure 28 : Advertising mode selection 28
Figure 29 : Sensor trigger source and trigger check period setting 30
Figure 30 : Sensor trigger source and thresholds 31
Figure 31 : Enable of GPIO input states as trigger 32
Figure 32 : Power switch control 33
Figure 33 : ADC configuration tab 34
Figure 34 : ADC channel configuration 35
Figure 35 : Unit mapping setting 36

Figure 36 : GPIO edge counting 37
Figure 37 : GPIO edge count configuration 37
Figure 38 : I2C configuration tab 38
Figure 39 : IN100 states 39
Figure 40 : I2C commands configuration window 40
Figure 41 : I2C read data as adv. payload 41
Figure 42 : GPIO configuration tab 42
Figure 43 : Power management and pulse counting 43
Figure 44 : One-wire count configuration tab 44
Figure 45: Square wave settings 45
Figure 46: Square wave settings with square wave output type 46
Figure 47: Square wave settings triggered by advertising set 47
Figure 48 : Advanced mode settings 48
Figure 49 : XO Circuit 49
Figure 50 : XO tab and XO setting configuration 49
Figure 51: XO startup 50
Figure 52 : RF test interface 51
Figure 53 : DFN8 package 52
Figure 54 : QFN18 package 52
Figure 55: WLCSP10 package 52
Figure 56 : Programming with NanoBeacon™ config tool 53
Figure 57 : UART communication status report dialog 53
Figure 58 : RAM run mode status report 54
Figure 59 : Burning in progress 55
Figure 60 : Programming progress complete 56

List of Tables

Table 1 : MGPIO mapping to analog input channels 33

1. Introduction

The IN100 device is a member of InPlay's NanoBeacon™ SoC product family, which supports Bluetooth low energy broadcasting. This device features an efficient and configurable state machine, non-volatile memory for user pre-defined data payloads, and data SRAM for dynamic data storage. The device is designed for maximum ease of use so there is no need to do any software programming. Instead, the IN100 is configured through a PC GUI tool, the InPlay NanoBeacon Config Tool.

The NanoBeacon Config Tool provided by InPlay allows the user to easily configure and test various advertising modes and payloads for the IN100. For testing the device, the configuration can be run from RAM and will execute its settings as long as the IN100 device is being powered. After testing, the desired configuration can be written to the device's one-time programmable (OTP) memory.

Figure 1 shows a typical development environment for IN100 devices using the NanoBeacon development kit. The development kit is connected to a PC via a USB cable. The NanoBeacon Config Tool GUI (Graphical User Interface) can be installed on a PC to test and configure the IN100 development kit.

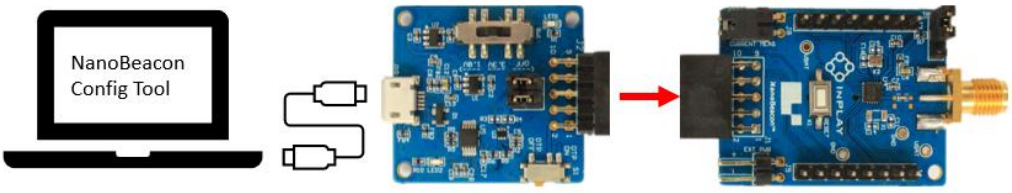

Figure 1 : A typical devlopment envionment for IN100 devices

The NanoBeacon development kit (DK) consists of two boards: an OTP programmer board, and an IN100 development board (also referred as a tag or tag board in this document) as shown in Figure 2. The programmer board has a USB to UART adapter to connect to the tag board through an 8-pin connector.

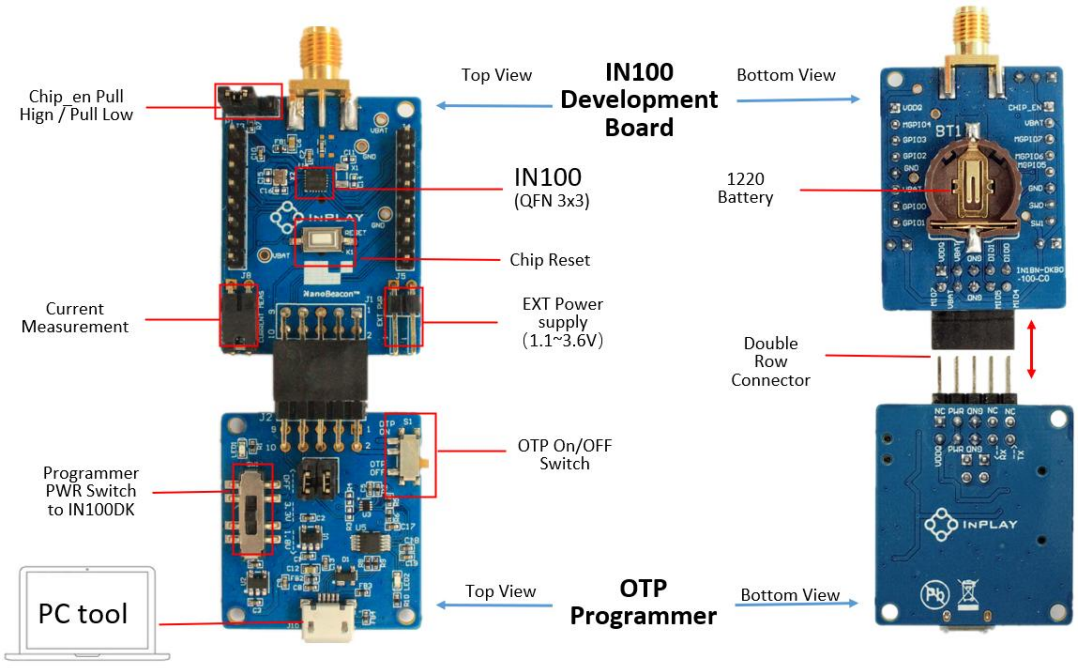

Figure 2 : NanoBeacon development kit (front and back view)

1.1 Things you will need

Hardware

IN100 NanoBeacon Development Kit - includes the programmer board and a tag board

- 3.5mm male SMA RF antenna
- USB-A to Micro-USB cable
- PC for running NanoBeacon config 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/Linux/Mac OS) – for configuring and testing the tag
- NanoBeacon BLE Scanner App (Android and iOS) or other Bluetooth Scan app (several freely available on Google Play Store and Apple App Store) – for scanning the tag once it starts advertising

Tools

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

1.2 How to configure and use the NanoBeacon

Below are the steps to use and configure the IN100 using the development kit.

1) Download the NanoBeacon config tool from https://inplay-tech.com/nanobeacon-config-tool and uncompress/install the downloaded file to a proper folder on your PC.
2) Connect the programmer board and development board as shown in Figure 2. Connect the programmer board to the PC with a USB cable (as shown in Figure 1). Green and red LEDs on programmer board should light up.
3) Run the PC application "NanoBeaconConfigTool". To control the tag board, a serial communication link must be established. This is done by first detecting the ports, selecting the correct COM port, and then selecting the correct baud rate (default is 115200 bps). After selecting the correct port and baud rate, click the connect button to connect to the PC with the tag board.
4) Change configuration settings based on the user's needs, and verify the configuration using the "Run in RAM" mode. Note: Configurations involving I2C operations are not supported in the "Run in RAM" mode.
    a. After configuring the tag board, the user may verify the configuration using the "Run in RAM" mode by clicking the 'Run in RAM' button. After clicking "Run in RAM", a scanning device like a phone or tablet can be used to see if the beacons are received. Please refer to the "InPlay IN100 NanoBeacon Scanner User Guide" about installing and using InPlay's custom NanoBeacon BLE Scanner application for Android and iOS. The "Run in RAM" mode 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 supply from the tag board by unplugging the programmer board 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 on the tag board.
    b. After testing and verifying the configuration in "Run in RAM" mode, customers may burn the configuration into the device's OTP memory.
5) "Burn" the desired configuration into the IN100's OTP memory.
    a. The tag board must be powered while burning the OTP memory. A status bar shows the progress of the burning operation. If the operation is successful, there will be a success message pop up. If the operation fails, there will be an error message. After burning the configuration, the user needs to power-cycle the tag board to let the IN100 start advertising according to the configuration stored inside the OTP memory. Note: the IN100 device's OTP memory can be only programmed once, and once it is programmed, it is irreversible.
6) Exit the NanoBeacon config tool program. If the user wants to test using a coin-cell battery, disconnect the tag board from the programmer board and install a CR1220 battery in the tag board.

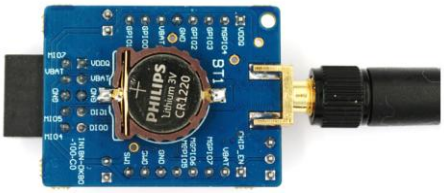

Figure 3 : Tag board powered by a coin-cell battery

2. NanoBeacon Config Tool

InPlay NanoBeacon config tool is a PC GUI (Graphical User Interface) tool for configuring and testing the IN100. The PC tool allows for flexible configuration of the IN100's peripherals, advertising data payloads, and advertising parameters according to the use case requirements.

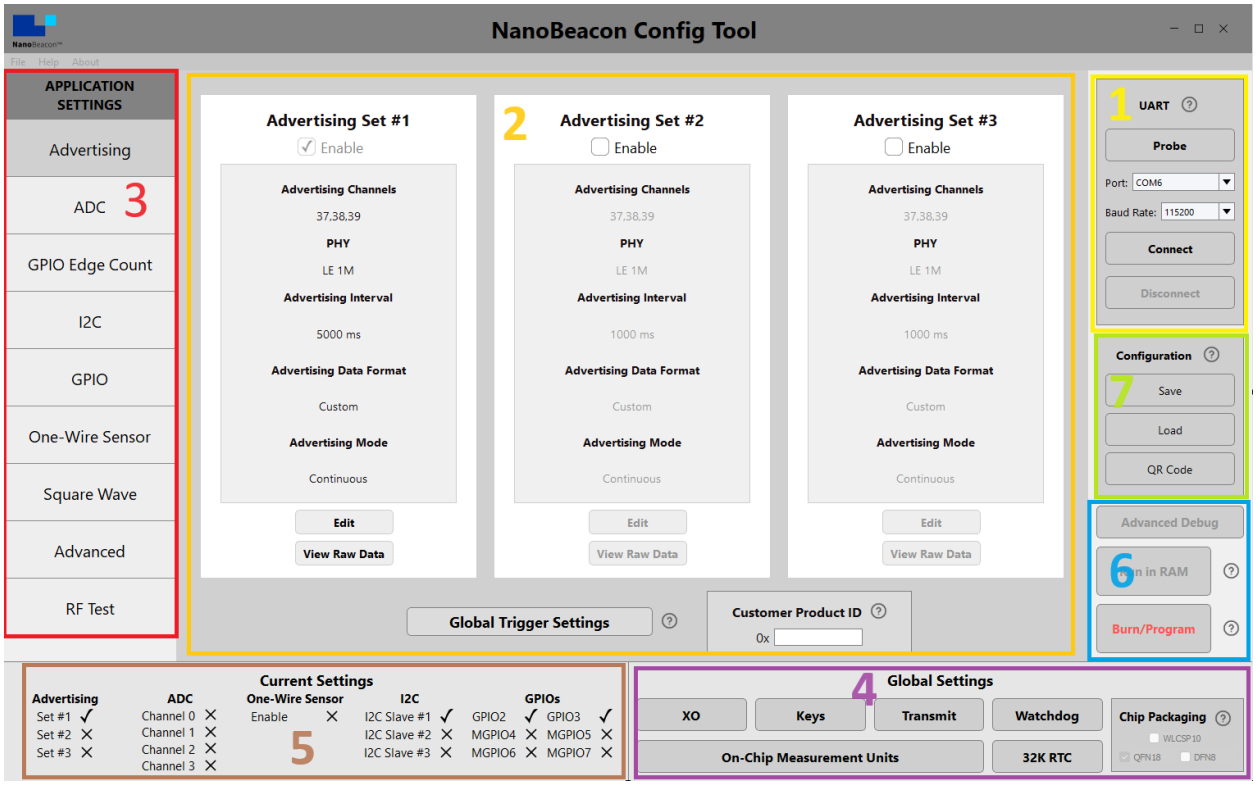

Figure 4 : Overview of NanoBeacon config tool

Figure 4 displays the config tool's default interface. This interface is used for navigating the IN100 configuration settings: :

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

2) **Advertising Settings:** Provides a window for configuring advertising data payloads, intervals, advertising modes (continuous advertising vs triggered advertising) , etc. for up to three advertising sets.
3) **Application Settings:** Includes all configurable option tabs including advertising settings, ADC, GPIO edge count, I2C, GPIO, one-wire sensor, square wave. In addition, there is an advanced mode setting to enable 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 CW (continuous waveform/carrier) test.
4) **Global settings:** Provides access to dvice settings that apply to all advertising sets.
5) **Current Settings:** Provides an overview of device settings and configurations that have been selected.
6) **Configuration:** Provides the user with a way to save the configuration to a file or load 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" mode, which provides a way to test and verify the user's configuration before it is permanently burned to the device. Whenever there are changes to the configuration, the user should always reset the device before operating "Run in RAM" mode. The other option is "Burn/Program" which will permanently burn the configuration file to OTP memory inside the device, which is irreversible and should be performed as the last step.
7) **Advanced Debug:** 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 config tool through **Advertising set #1**, **Advertising set #2** and **Advertising set #3**, respectively, as shown in Figure 5. For each advertising set, the user can click the corresponding "Edit" button to access the configuration settings for that advertising set. These settings include the advertising parameters. The user can use the "View Raw Data" button to review the advertising raw data that the user has configured.

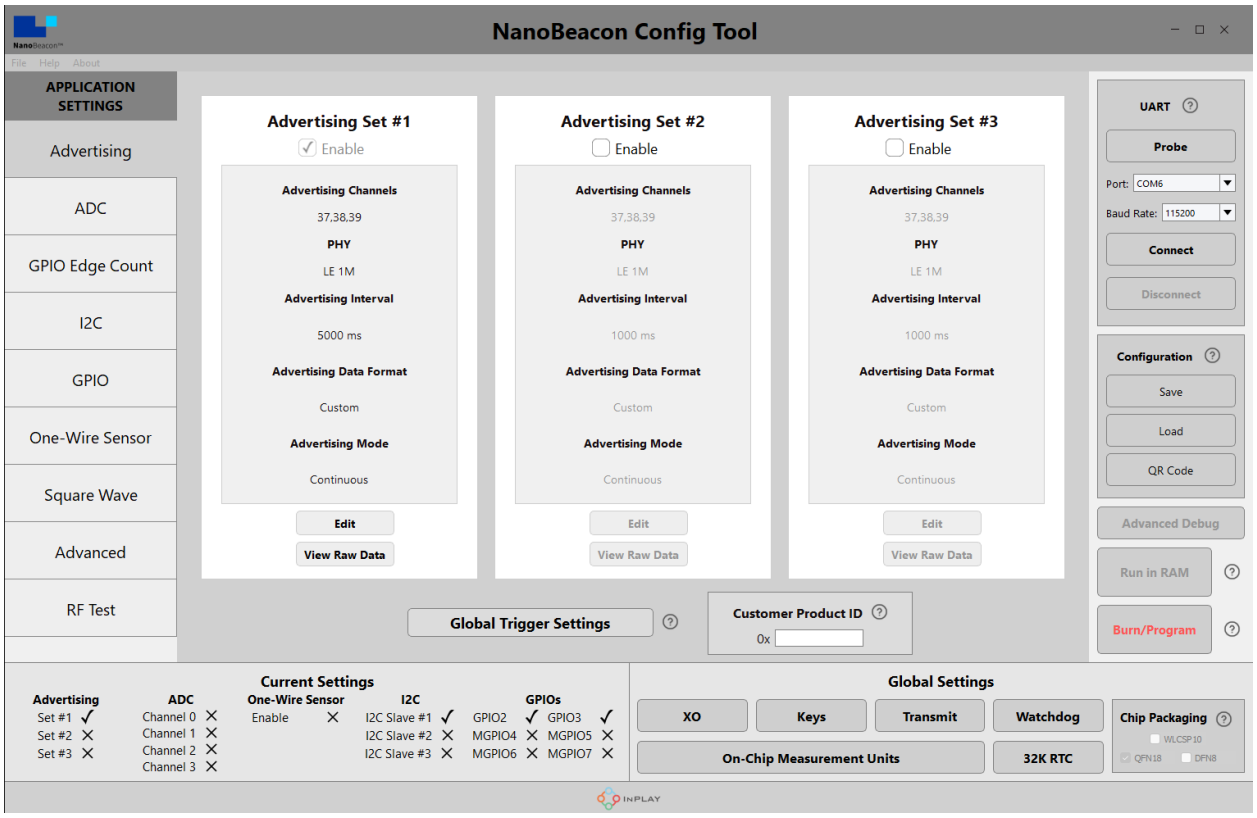

Figure 5 : Advertising sets configuration overview

The "QR Code" button will provide a QR code which contains the advertising address information for the user's mobile application to easily identify and parse data from advertisers. The InPlay Nanobeacon BLE Scanner App can scan this QR code to filter for the beacons that match the user's settings.

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 the payload data field information, it is referred to as "CustID".

The main window when editing an advertising set provides access to all possible configuration parameters for application requirements including Advertising Data, Advertising Parameters, and Advertising Mode, as shown in Figure 6.

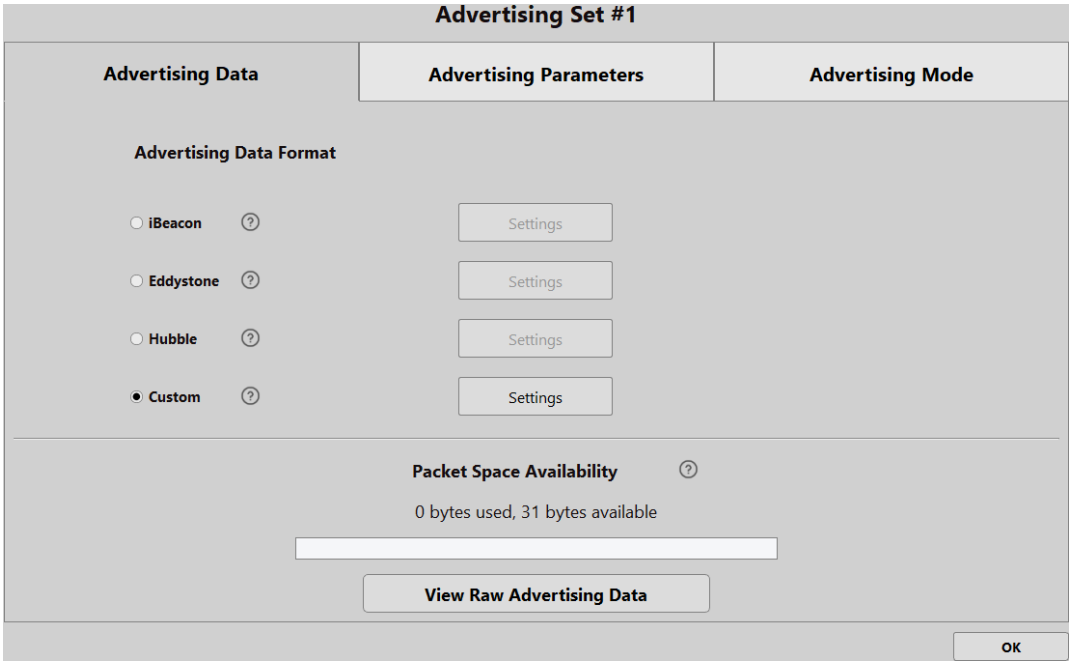

Figure 6 : Advertising set configuration window

2.1.2 *Advertising data format*

The NanoBeacon Config Tool supports both standard BLE packet data configuration through AD (Advertising Data) structures and user defined configuration completely controlled by the user.

Figure 7 shows the advertising data format, specified by Bluetooth SIG [1]. It consists of one or multiple AD structures. 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. For more details, please read reference [1].

The NanoBeacon Config Tool can be easily used to configure a Bluetooth SIG compliant advertising packet with one or multiple AD structures or non-compliant advertising packet shown in Figure 8 where the users can configure their own format in the "User Defined Data" section.

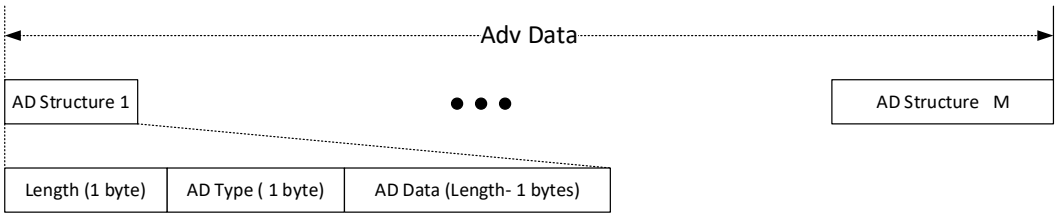

Figure 7 : Advertising data format

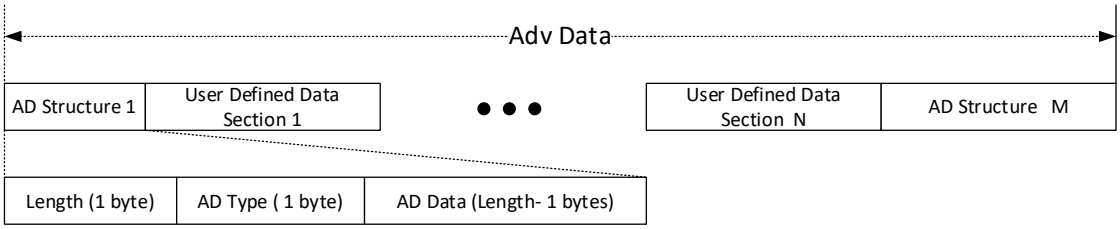

Figure 8 : Customer-defined advertising data format

The tool supports two commonly used advertising data formats as well a user defined format (custom format):

- 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, which users can use to define advertising packets in their own format other than iBeacon and Eddystone packets.

2.1.2.1 iBeacon

For the "iBeacon Advertising Settings", the user needs to fill in the following:

- **UUID:** Application developer should define a UUID specific to their app and development use case. The default value is E2C56DB5-DFFB-48D2-B060-D0F5A71096E0.
- **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.
- **Minor:** Allows further subdivision of region or use case, specified by the application developer.
- **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.

2.1.2.2 Eddystone

For the "Eddystone Advertising Settings", the user needs to follow the Eddystone protocol and configure the device accordingly. For details, please refer to the Google Eddystone protocol specification at https://github.com/google/eddystone/blob/master/protocol-specification.md.

2.1.2.3 Hubble

For the "Hubble Advertising Settings", the user can configure the encrypted 8-byte payload, the rotation exponent, and the encryption key being used. The Hubble format follows Hubble Network's payload format. For details, please refer to the example config file at https://github.com/HubbleNetwork/hubble-inplay-cfg/blob/main/samples/hubble-in100-1s-advinterval.cfg and the advertising packet details at https://hubble.com/docs/device-sdk/terrestrial/advertising-packet. If Hubble is used, its advertising set must be in continuous mode, and no other sets can be used.

- **Custom Encrypted Data:** Up to 8 bytes of encrypted data in the custom payload part of the Hubble advertising packet. This field cannot be left blank so one placeholder byte is there by default but may be replaced.

- **Rotation Exponent**: The rotation period exponent for the EID. When this value is X, the device will generate a new EID every 2^X seconds. The default value should be 15. For any other value, reach out to Hubble support.
- **Encryption Key**: The encryption key used for encrypting the payload.

2.1.2.4 Custom format

For the "Custom Advertising Settings", the user can configure the advertising payload based on their application needs:

- **Complete Local Name:** This is an AD type defined by Bluetooth SIG [2]. The user can edit this field to enter a string of characters to label the name of the device in its advertising packets. For more details, refer to reference [2].
- **Tx Power Level:** This is an AD type defined by Bluetooth SIG [2]. The Tx Power level data type indicates the transmitted power level of the advertising packet. For more details, refer to reference [2].
- **Manufacturer Specific Data:** This is an AD type defined by Bluetooth SIG [2]. The first two data octets shall contain a company identifier from the Assigned Numbers [3]. The default manufacture ID is 0x0505 which is the Assigned Number for InPlay Inc by Bluetooth SIG. For the AD data configuration, please refer to Section 2.1.3.
- **User Defined Data:** The user can edit this field to define their desired format or other AD types (other than those already listed in the config tool like "Complete Local Name", "Tx Power Level" and "Manufacturer Specific Data"). For the data configuration, please refer to Section 2.1.3.

2.1.3 Advertising data configuration

The AD data or the user defined data section (in Figure 7 and Figure 8) consists of multiple data segments, as shown in Figure 9. The user can easily use the config tool to select or enter each data segment.

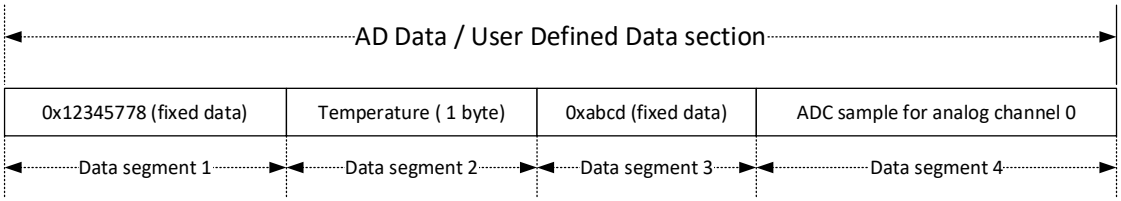

Figure 9 : Data segments for AD data/user defined data section

There are two categories for the data source for each data segment: predefined and extended data. The predefined data segment is fixed-pattern data which the user can directly type into the edit box. For the extended data, the user can select the source from the pull-down list and append it to the advertising packet.

2.1.3.1 Extended data

The config tool provides a number of extended data sources which the user can select and use in the advertising packet. Most extended data sources are dynamic, such as the internal temperature sensor data and the GPIO status. Below is the list of extended data sources:

- VCC: The measured 8-bit chip's supply voltage. One LSB corresponds to 0.03125 volts by default, unless the user changes the unit to a different number using the "On-Chip Measurement Units" tab shown in Figure 10.
- Internal Temperature (default 2 bytes): The measured 16-bit temperature using the internal temperature sensor inside the chip. It is in 2's complement format in units of 0.01 degree Celsius unless the user changes the unit to a different number using the "On-Chip Measurement Units" tab shown in Figure 10. If 1 byte is entered for the temperature value representation, the user should change the unit as shown in Figure 10 to avoid overflow due to short word-length.
- 1-Wire Count: Pulse count from the 1-wire sensor. 2 bytes.
- GPIO Status: GPIO input status. 1 byte (The most significant bit comes from MGPIO7, and the least from GPIO0).
- ADC CH0: Digital sample of the external analog input to the mixed signal GPIO pin 4 (MGPIO4). 2 bytes.
- ADC CH1: Digital sample of the external analog input to the mixed signal GPIO pin 5 (MGPIO5). 2 bytes.
- ADC CH2: Digital sample of the external analog input to the mixed signal GPIO pin 6 (MGPIO6). 2 bytes.
- ADC CH3: Digital sample of the external analog input to the mixed signal GPIO pin 7 (MGPIO7). 2 bytes.
- I2C Slave #1 Read Data: The read-out data from I2C Slave #1. For the offset configuration, please refer to Section 2.4.
- I2C Slave #2 Read Data: The read-out data from I2C Slave #2.
- I2C Slave #3 Read Data: The read-out data from I2C Slave #3.
- Time Stamp 0: Time stamp in units of 100 milliseconds. Up to 4 bytes.
- Time Stamp 1 (second counter): Time stamp in units of 1 second. Up to 4 bytes.
- ADV Count: The transmit advertising event count. Up to 4 bytes.
- Register Read Data: The read out from an internal register of the IN100 device with a given address. Up to 4 bytes.
- Random Number: Random number generated by the chip. Up to 4 bytes.
- Encrypted Raw: the pre-defined data to be encrypted. Please refer to Section 2.1.4.
- EAX Salt: Salt for encryption and authentication. Please refer to Section 2.1.4.
- Auth Tag: The message authentication code (also called authentication tag in this document). Please refer to Section 2.1.4.
- Customer Product ID: A unique ID that can be assigned to each device.
- Bluetooth Device Address: Bluetooth device address that is defined in advertising parameters settings (Only public address or static random address generated by PC tool or input by user is supported).
- Characters UTF-8: characters of 8-bit Unicode Transformation Format.

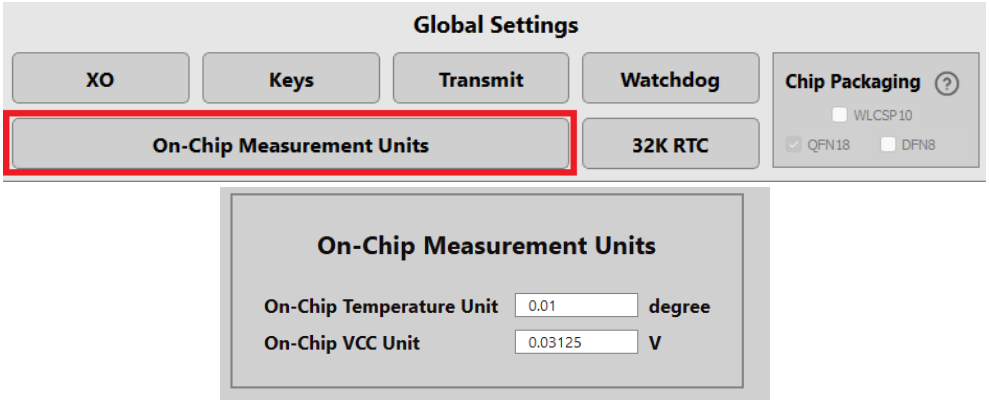

Figure 10 : On-chip temperature and VCC unit setting

2.1.3.2 Entry of advertising data

Below are the steps to entry the advertising data:

1) Enter one data segment:
   - 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.
   - If it is extended data, select the data from the pull-down list from Dynamic Data box and then press the button "Append to Data". The added extended data will be shown in the data entry inside "<>", as shown in Figure 11. 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 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 data configuration is complete, 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.

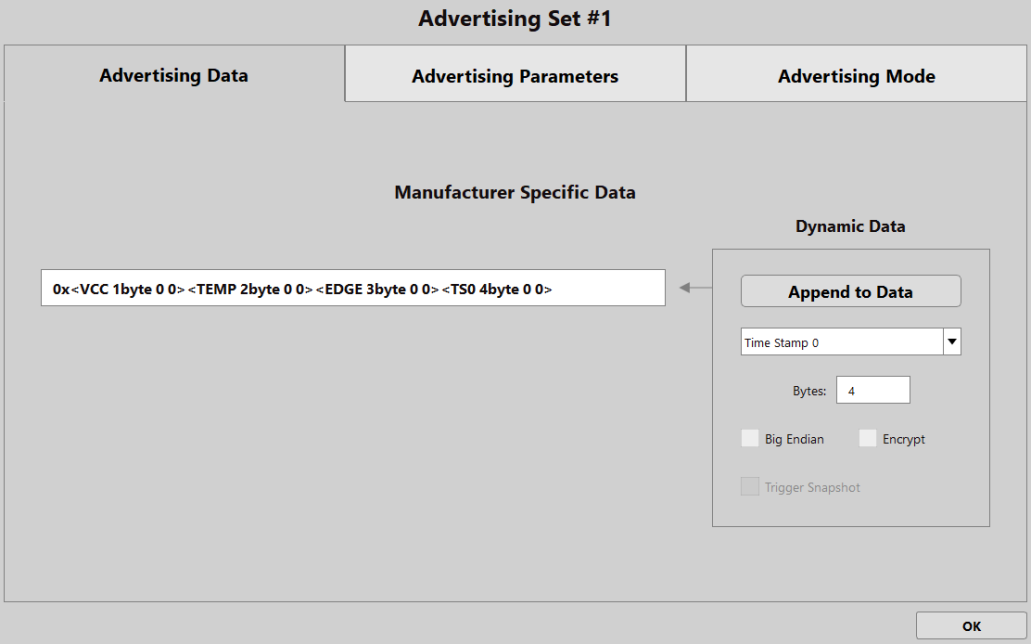

Figure 11 : Selecting extended data

2.1.4 Advertising data encryption and authentication

The NanoBeacon device supports simultaneous encryption and authentication for the advertising data based on the AES-EAX encryption algorithm [4]. Each of the three advertising data sets can be individually and independently configured regarding whether the user needs to enable encryption and/or authentication.

2.1.4.1 AES-EAX encryption

To enable encryption and authentication, in the device, the user needs to decide the following inputs to the AES-EAX engine for each advertising set in the config tool:

- 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 12 illustrates the advertising data without and with encryption/authentication. It is assumed that 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 salt is used, then customers can select not to present the salt in the advertising data.
- If the user does not need authentication, then the user can select not to present the tag in the advertising data.

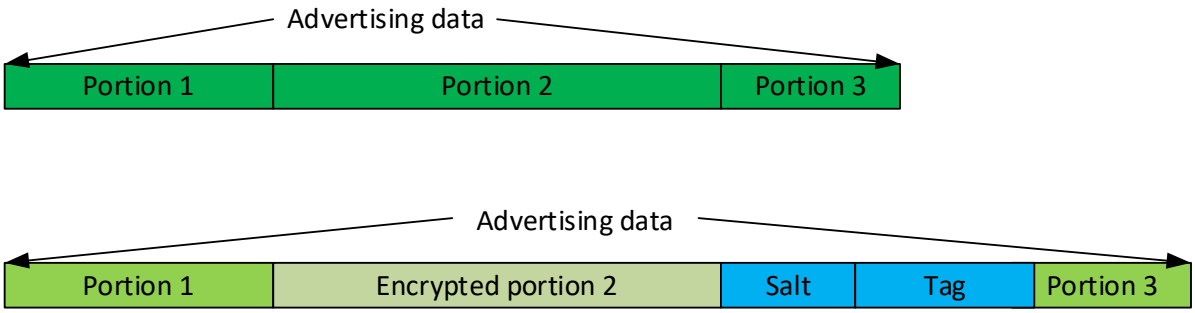

Figure 12 : Advertising data encryption and authentication

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 13.

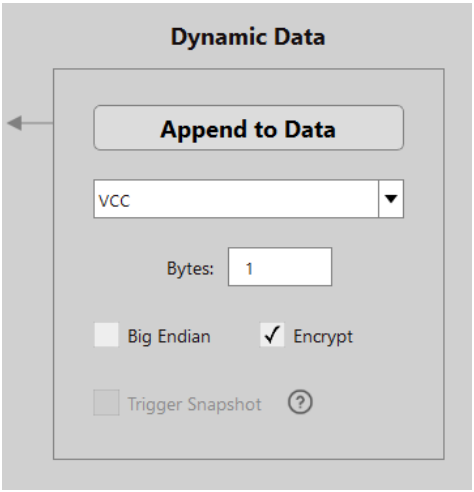

Figure 13 : Encryption enabling

2.1.4.2 Encryption key

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 available in the "Data Encryption Settings" tab, as shown in Figure 14.

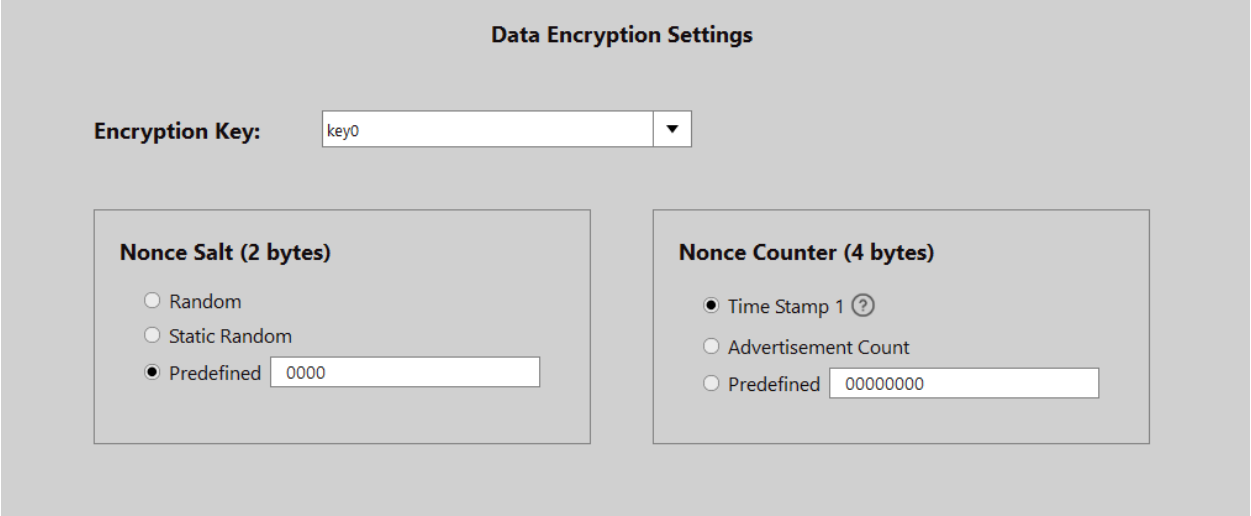

Figure 14 : Selection of key and nonce for AES-EAX for each advertising set

The configuration for the three encryption keys are under "Global Settings" as shown in Figure 15. The input box of the corresponding key needs to be filled with hexadecimal numbers.

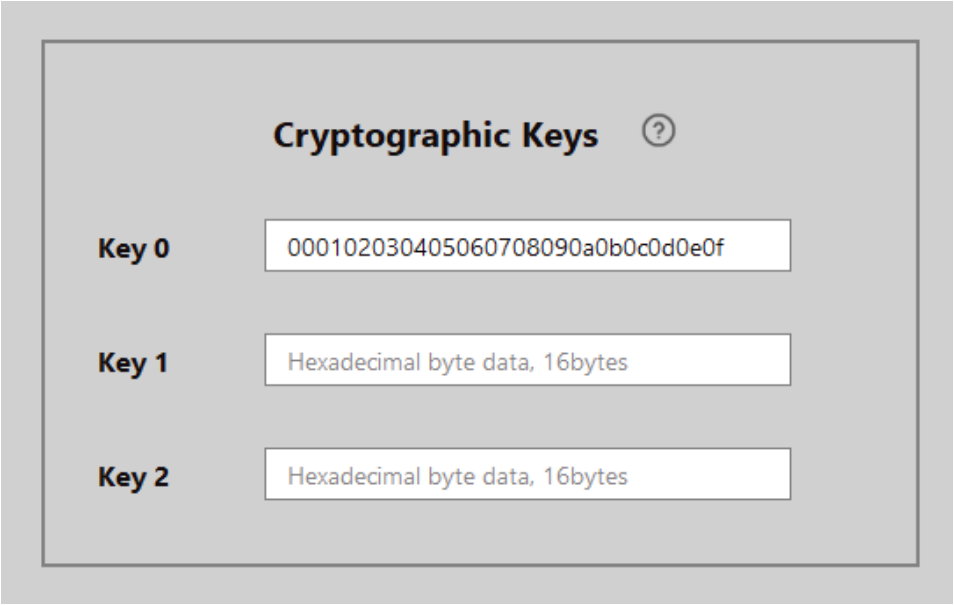

Figure 15 : Configuration of encryption keys

2.1.4.3 Nonce

The 6 byte nonce consists of a 2 byte salt and a 4 byte counter (this counter is referred to as the nonce counter). The device offers several sources for the salt and the counter.

The config tool offers three options for the salt, and the user can configure these options using the "Data Encryption Settings" tab, as shown in Figure 14.

- Static random number: The salt will change upon power cycle of the device, and remains the same after power-on.

- 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 or static random number, we need to include the salt in the advertising data as a plain text, which is shown in

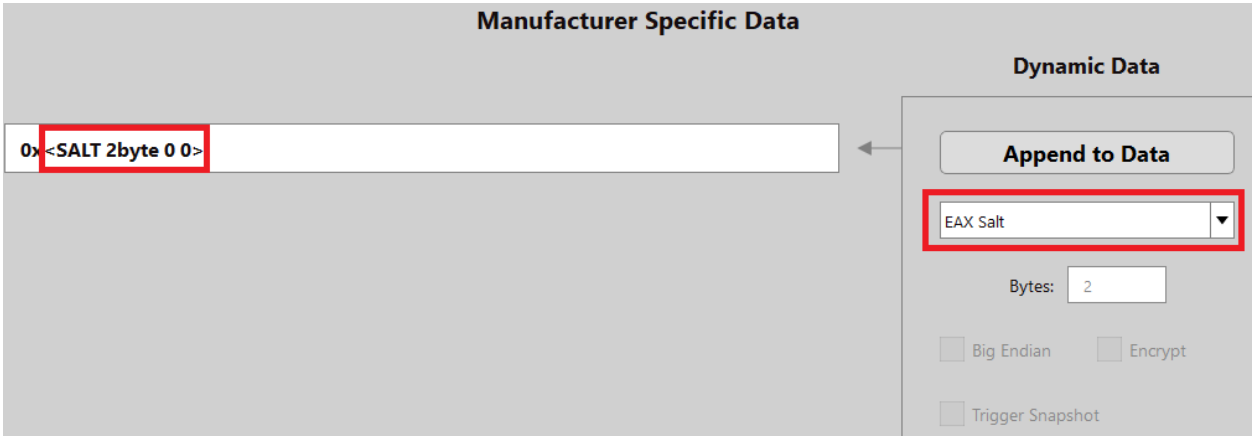

Figure 16 as an example.

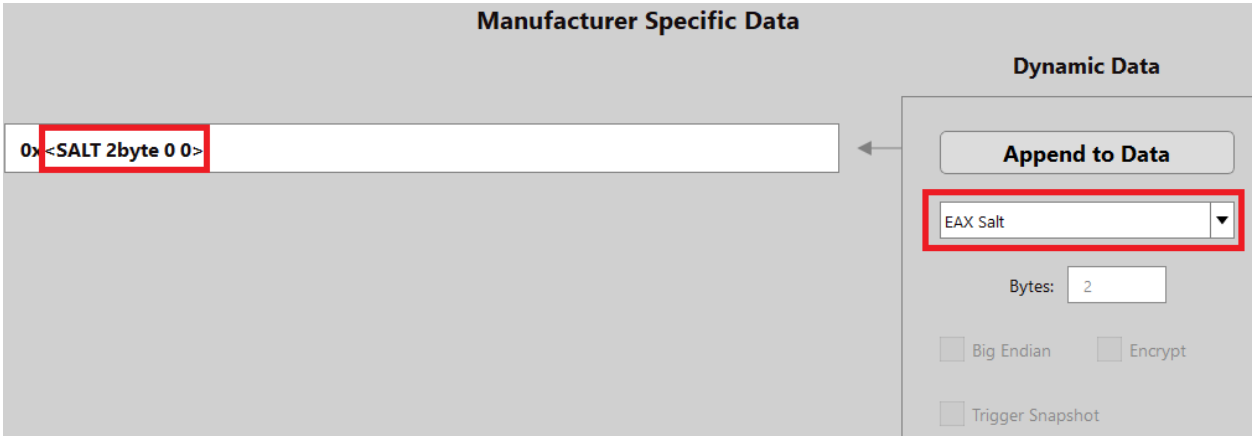

Figure 16 : Including the AES-EAX salt in the advertising data

There are multiple options available for the nonce counter as shown in Figure 14:

- 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 (Time-stamp 1 or the advertising counter) for the nonce counter instead of a fixed number.

Normally, the nonce counter is not advertised in the advertising data. If the selected nonce counter is Time Stamp 1(the 32-bit 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.

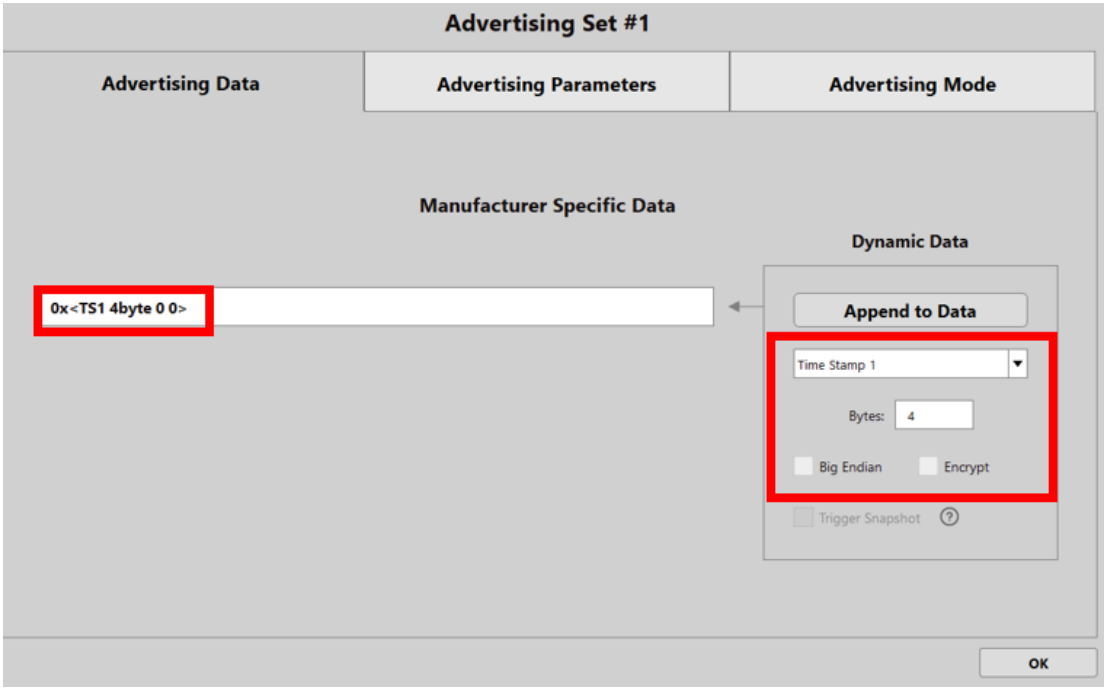

Figure 17 : Including the nonce counter in the advertising data: time stamp 1

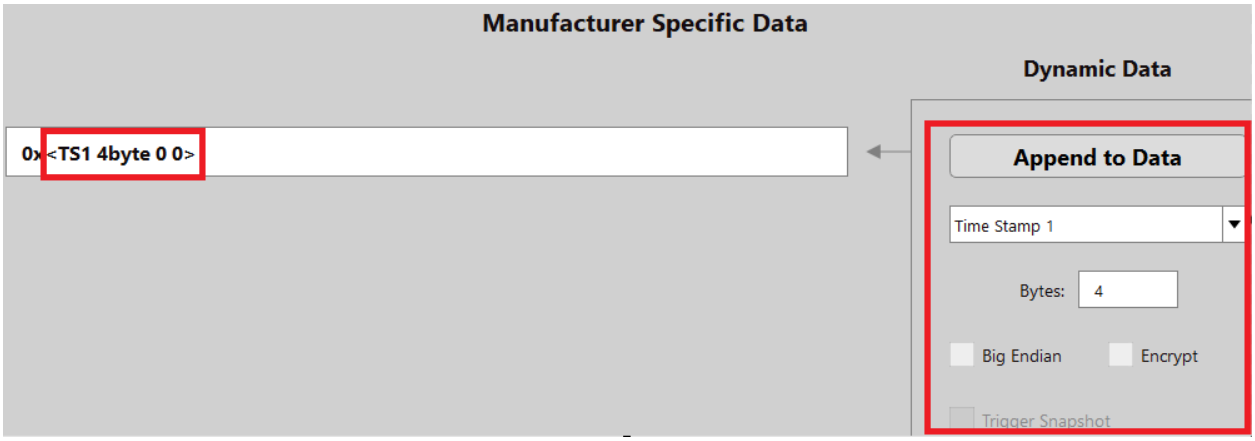

Figure 18 : Including the nonce counter in the advertising data: advertising counter

2.1.4.4 Encryption raw data configuration

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” from the “Extended Data” 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.

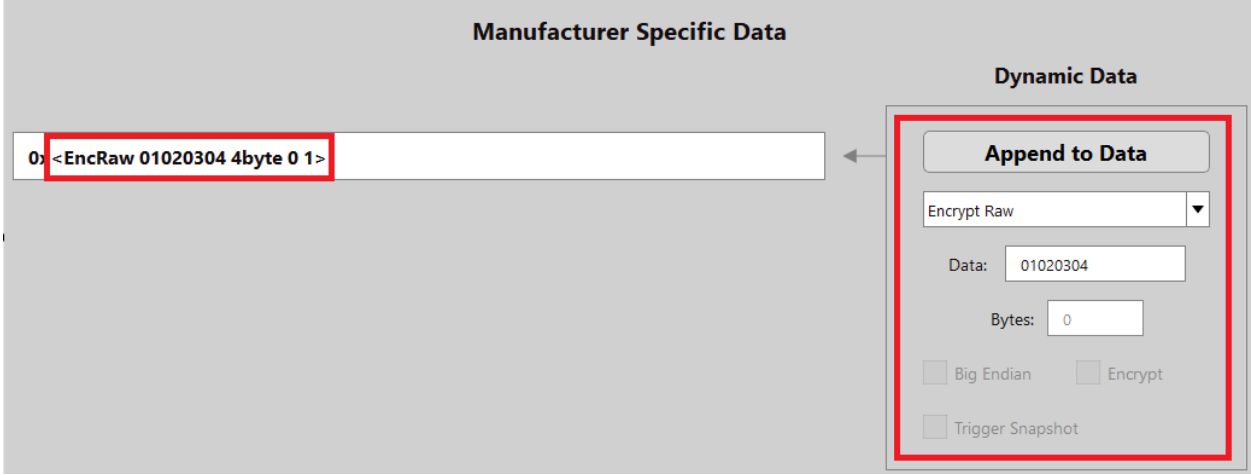

Figure 21 : Encrypt the predefined data

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.

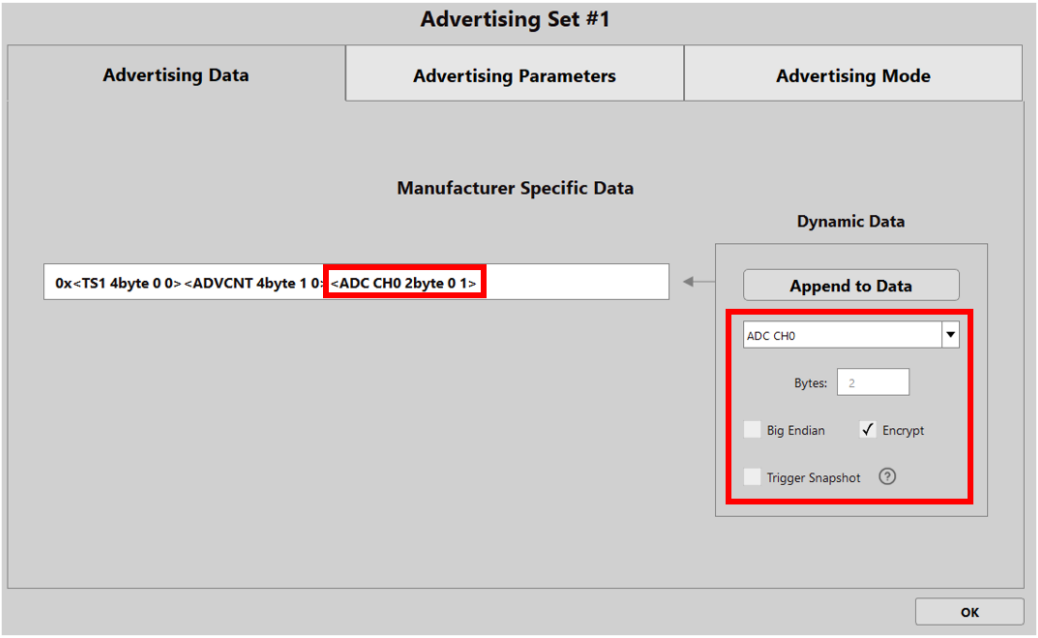

Figure 22 : Encrypt an extend data item

2.1.4.5 Authentication tag

To include the authentication tag in the advertising data, we need to select the "Auth Tag" in the "Extended Data" 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.

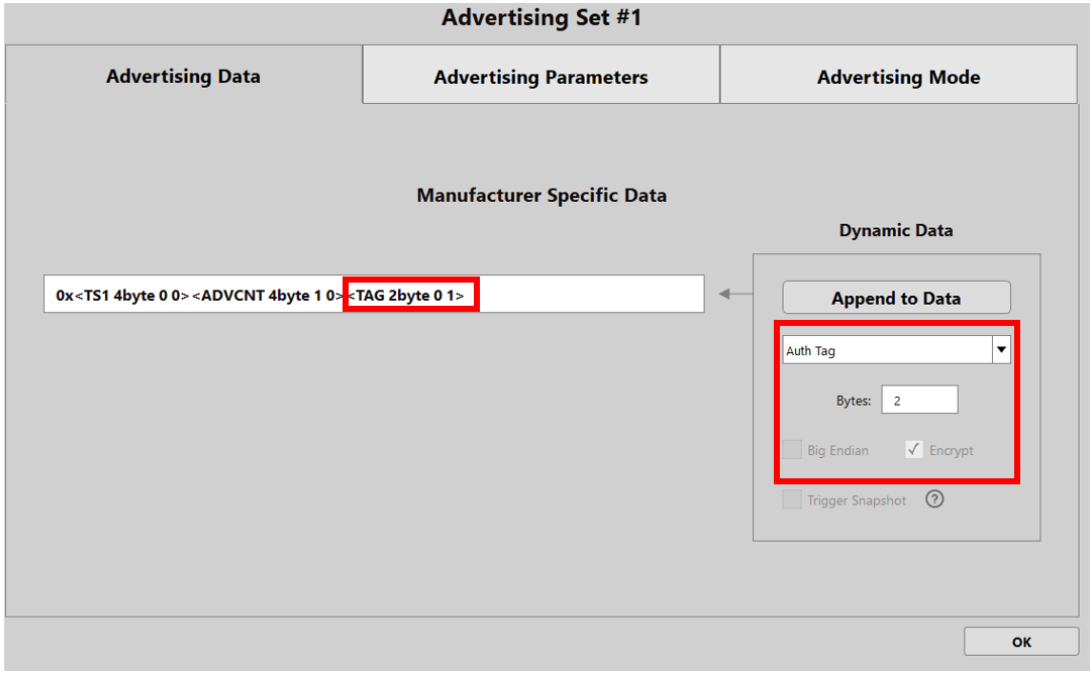

Figure 23 : Including a tag in the advertising data

2.1.5 *Advertising raw data review*

The tool also provides the option to "View Raw Data" for the user to review the payload data for each advertising set, including encryption status, endianness, and notes about the specific data fields. An example of such is shown as in Figure 24.

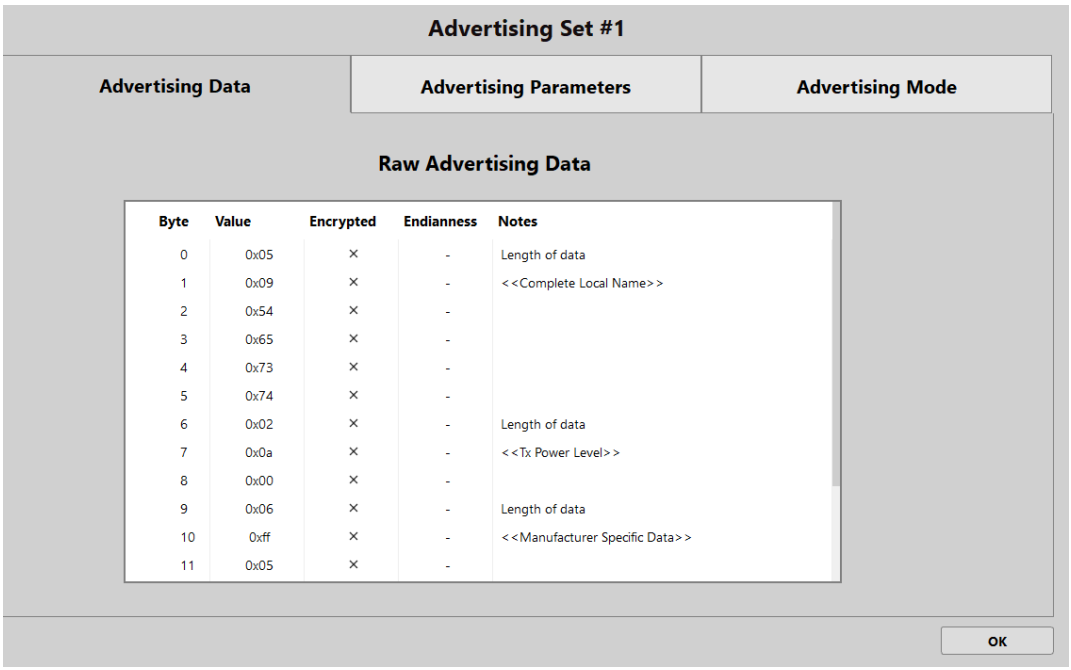

Figure 24 : Raw advertising data review window

2.1.6 Advertising parameter 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. The minimum value supported is 20 milliseconds and the maximum value supported is 10,485,759.375 milliseconds in increments of 0.625 milliseconds
- **PHY Selection**: Defines the PHY used for sending out the advertising data packets. The default rate is LE 1M which is the most used rate. The LE Coded PHY is typically used for long-range advertising. In the IN100 devices, the Coded PHY implementation uses the same channels as 1 M PHY, but changes the PHY rate.
- **CTE**: Once checked, the device will automatically add a Constant Tone Extension to the advertising packet. The CTE duration unit is 8us, and user can choose the CTE duration range from 2 to 20 which equivalent to from 16us to 200us.
- **Advertising Random Delay:** Introduces a random delay on top of the nominal advertising interval. The default value is between 0 to 10 ms but goes up to between 0 and 160 ms.
- **Advertising Channels**: Determines the RF channels used to send advertising data on. At least one channel must be selected.
- **Bluetooth Device Address**: Defines what type of Bluetooth Device Address to use in the advertising packets. The options include a static address either generated by the GUI or the by the chip on cold boot, or entered by the user; a public address entered by the user; or a private address which is either resolvable or non-resolvable.

An example configuration of the above is shown in Figure 25.

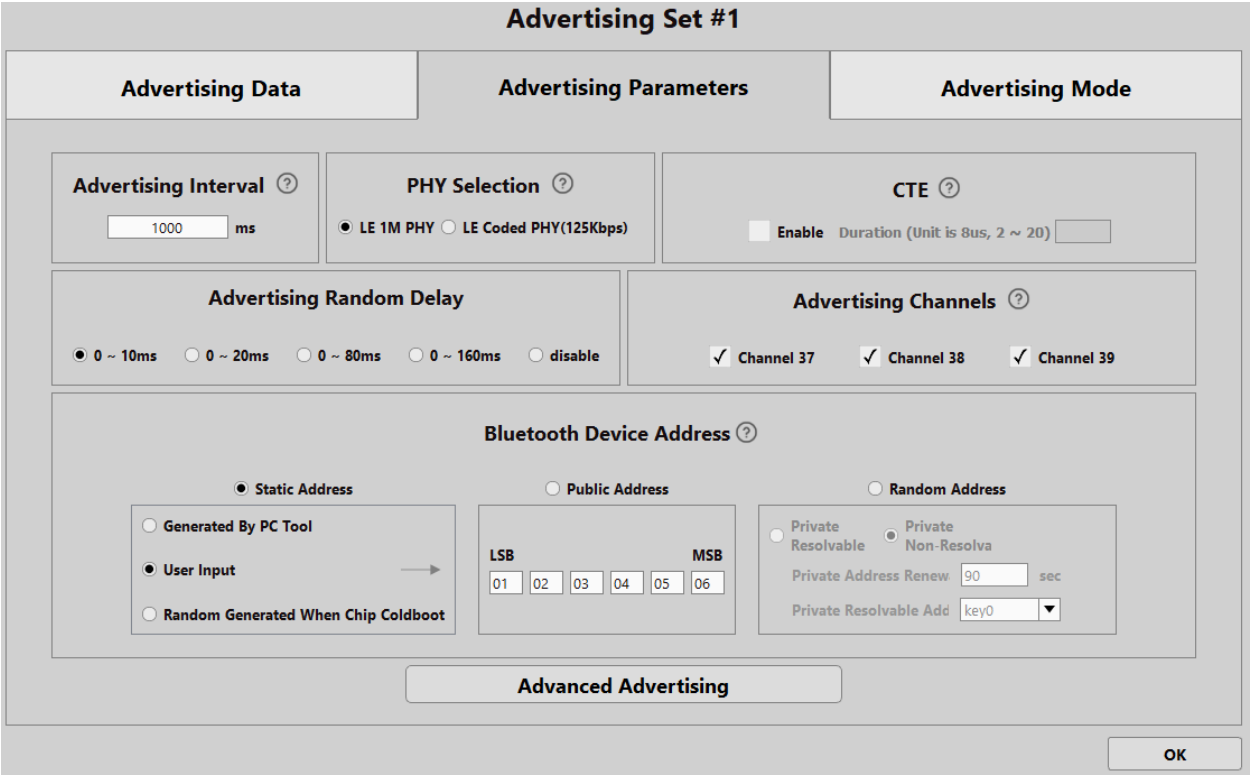

Figure 25 : Advertising parameters configuration

2.1.7 *Advanced advertising parameter configuration*

A typical advertising packet's link layer packet format is shown in Figure 26. 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 formats for their specific application needs.

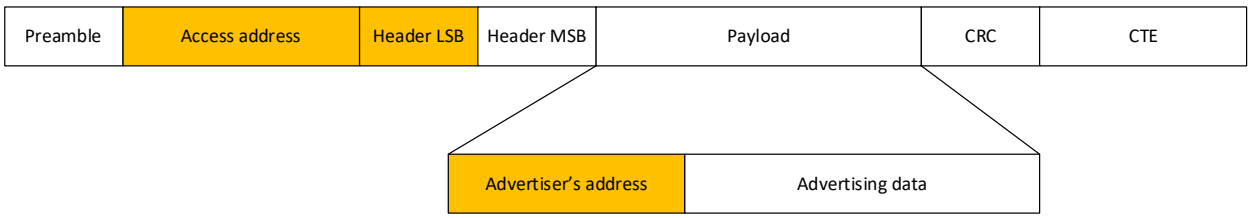

Figure 26 : Link layer advertising packet format

In the advanced advertising parameters setting tab, as shown in Figure 27, the user can define and input their specific packet format:

- **Exclude Bluetooth Address from Advertising:** Once this box is checked, 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:** The 1-byte header LSB (in Figure 26) can be defined by user as shown in in Figure 27. Otherwise, it is automatic generated the config tool.

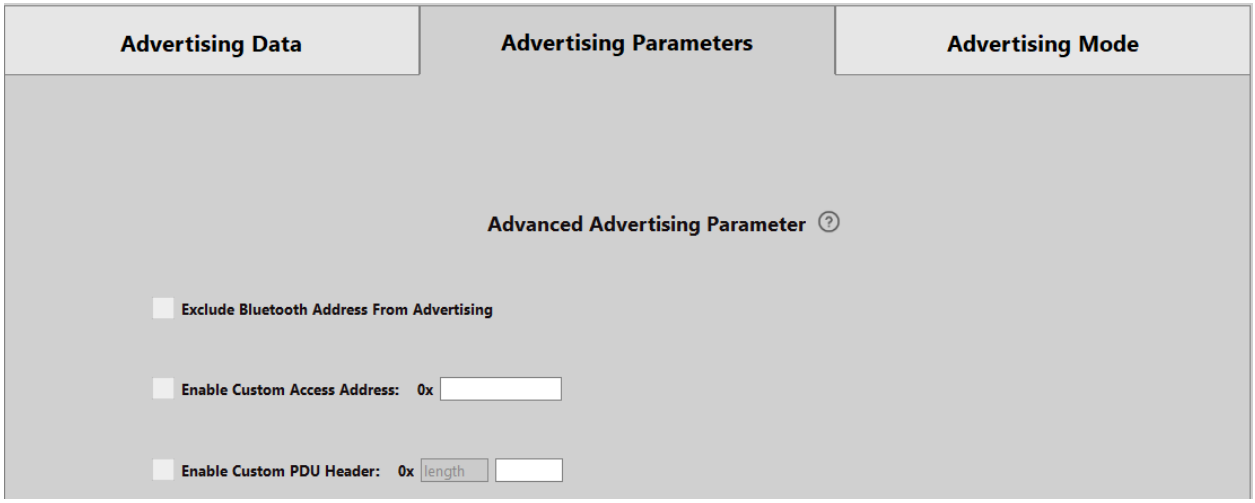

Figure 27 : Advanced advertising parameter settings

2.1.8 Advertising mode configuration

The NanoBeacon™ supports two advertising modes: continuous mode and event triggered mode which can be configured using the GUI as shown in Figure 28.

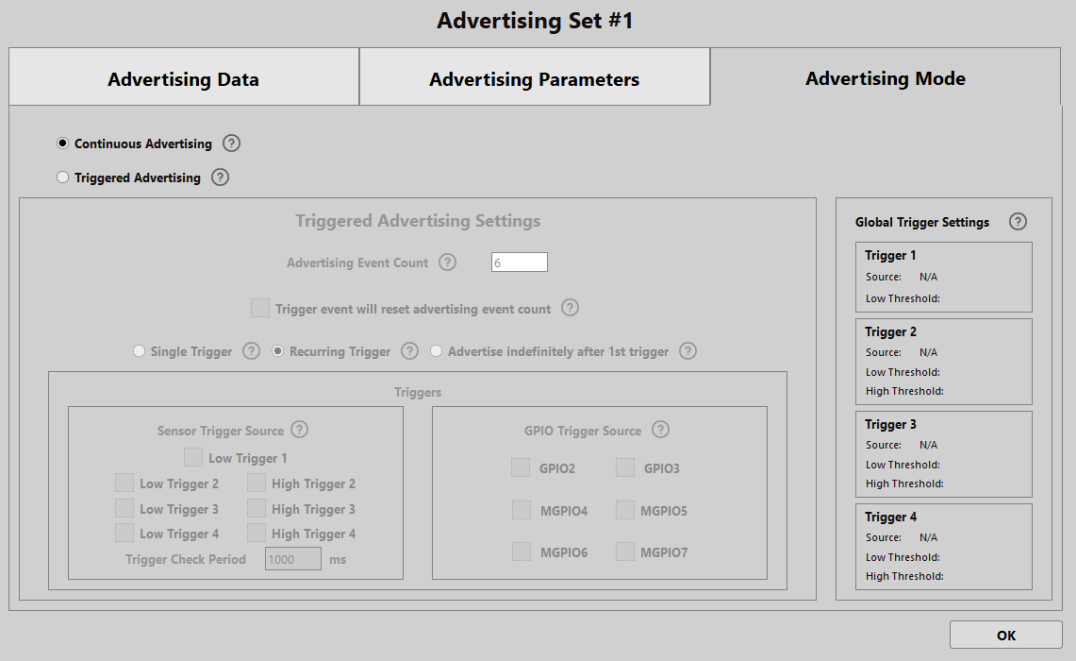

Figure 28 : Advertising mode selection

2.1.8.1 Continuous advertising mode

In continuous mode, the device advertises periodically according to the interval set by advertising interval powered on.

2.1.8.2 Triggered advertising mode

In the event trigger mode, the device advertises once an event is triggered according to the trigger parameters. The user can configure the number of advertising events per trigger event through the "Advertising Event Count" setting. "Trigger will reset advertising event count" enables resetting of the advertising event count whenever a trigger condition is met for each enabled advertising event.

The device can have multiple sources used to trigger advertising. For the trigger source configuration, please refer to Section 2.1.9. The device 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.

2.1.9 Trigger source configuration for triggered advertising

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

2.1.9.1 Sensor trigger source and threshold configuration

The IN100 device integrates an ADC which can digitize the analog signals on MGPIO4, MGPIO5, MGPIO6, and MGPIO7, as well as VCC (The chip's power supply) and internal temperature signal. In addition, the IN100 supports 1-wire sensor and three I2C slaves. The read outs from the ADC, 1-wire sensor, and I2C slaves can be used as triggered advertising sources. In this guideline, these options are referred to sensor inputs.

2.1.9.1.1 Source configuration

To enable one or more sensor inputs as a trigger source, the user needs to select the sensor trigger source as shown in Figure 29.

- **Trigger 1 Low Threshold:** Trigger 1 triggers an advertisement when the sample associated with Trigger 1 is below its set threshold. Trigger 1 only has a low threshold comparator.
- **Trigger 2 High Threshold:** Trigger 2 triggers an advertisement when the sample associated with Trigger 2 is above its set threshold.
- **Trigger 2 Low Threshold:** Trigger 2 triggers an advertisement when the sample associated with Trigger 2 is below its set threshold.
- **Trigger-3 High Threshold:** Trigger 3 triggers an advertisement when the sample associated with Trigger 3 is above its set threshold.
- **Trigger 3 Low Threshold:** Trigger 3 triggers an advertisement when the sample associated with Trigger 3 is below its set threshold.
- **Trigger 4 High Threshold:** Trigger 4 triggers an advertisement when the sample associated with Trigger 4 is above its set threshold.
- **Trigger 4 Low Threshold:** Trigger 4 triggers an advertisement when the sample associated with Trigger 4 is below its set threshold.

In addition to selecting a trigger source, the user needs to define the "Trigger Check Period" setting. This setting determines the interval at which the device will check the value of the sensor inputs. If the trigger condition is met, then the device will start advertising. Figure 29 shows the location of this parameter in the GUI tool.

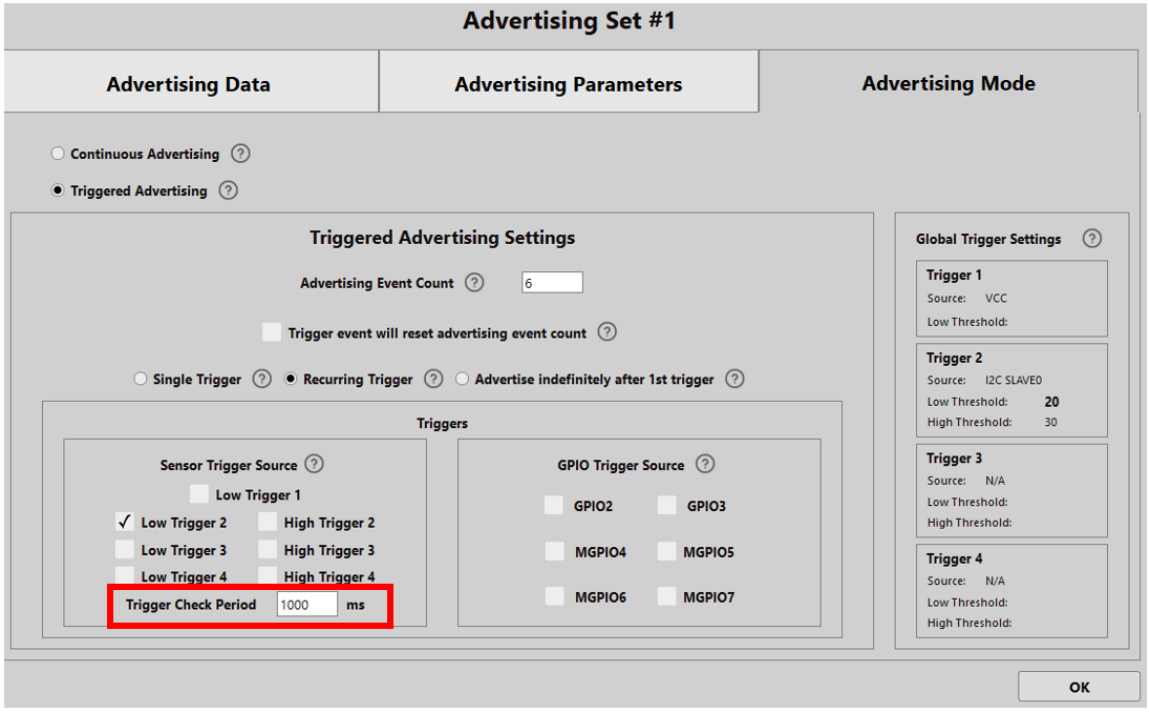

Figure 29 : Sensor trigger source and trigger check period setting

The user can use the "Global Trigger Settings" tab to map a senor input source to Triggers 1 through 4 as shown in Figure 30. In that tab, the user can also configure the thresholds.

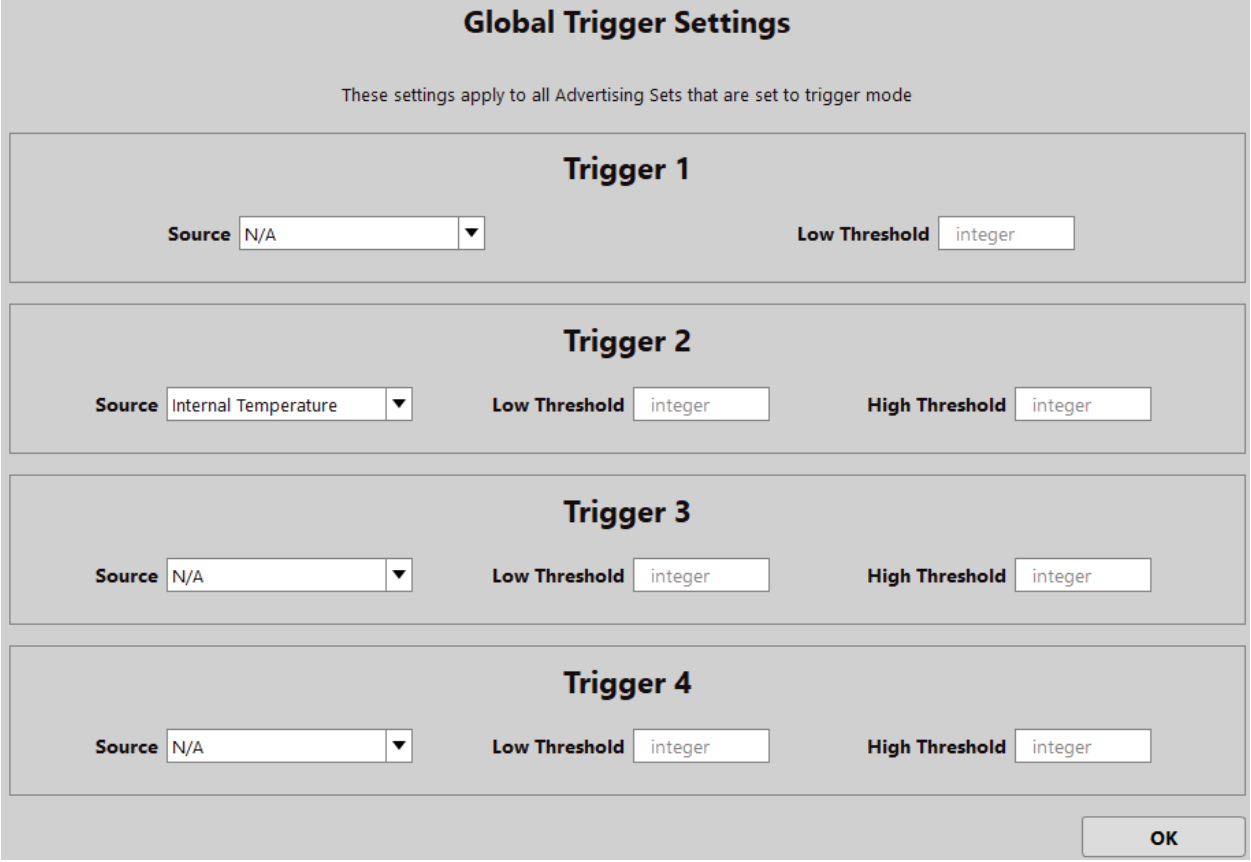

Figure 30 : Sensor trigger source and thresholds

2.1.9.1.2 Threshold configuration

For each trigger (trigger 1 through trigger 4), once the user selects its source, the user needs to configure the threshold for that source:

- **VCC:** The VCC is the power supply to the device's VCC pin. The default digitized VCC sample has a unit of 0.03125 V unless the user changes the unit through "On-Chip Measurement Units" tab shown in Figure 10. The number entered into the threshold box in Figure 30 should be equal to the desired threshold value (in volts) / On-chip VCC unit. For example, if the user wants the device to start advertising when the VCC is below 2.5V, then the low threshold value should be 2.5/0.03125=80.
- **ADC CH0**, **ADC CH1**, **ADC CH2** and **ADC CH3:** These are the analog channels connected to MGPIO4, MGPIO5, MGPIO6, and MGPIO7, respectively. The threshold configurations for these channels are similar to the configuration for ADC CH0. Please refer to Section 2.2 for more details.
- **One-Wire Sensor:** The device does not do any conversion for the received pulse count from the one-wire sensor. Thus, the user should enter a threshold value for the one-wire sensor as needed for the application. The device assumes the pulse count is an unsigned 16-bit number.
- **Internal chip temperature:** The device integrates an internal temperature sensor and the user can set the threshold based on the desired temperature value. The input value unit can be

determined through the "On-Chip Measurement Units" settings in Figure 10. The number entered to the threshold box in Figure 30 should be equal to the desired threshold value / On-chip temperature unit; integer inputs only. For example, if the user wants to have a high threshold of 80°C, and the unit is 0.1°C, then the threshold entered in Figure 30 should be 800. if the user wants to have a low threshold of -20°C, and the unit is 0.1°C, then the threshold entered in Figure 30 should be -200.

- **I2C Slave #1**, **#2**, and **#3:** The device does not do any conversion for the I2C readout. Thus, the user should enter a threshold value with the raw I2C readout in mind. During the comparison, the device assume the I2C read-out is in 16-bit 2's complement format.

2.1.9.2 GPIO input state as trigger source

The device supports GPIO level (high or low) or edge (rising or falling) as a trigger source. To enable the GPIO trigger, check the box corresponding with the GPIO as shown in Figure 31. The "GPIO" can be used to decide if the GPIO triggers on a level or edge. The GPIOs selected for event trigger must also be configured as an input.

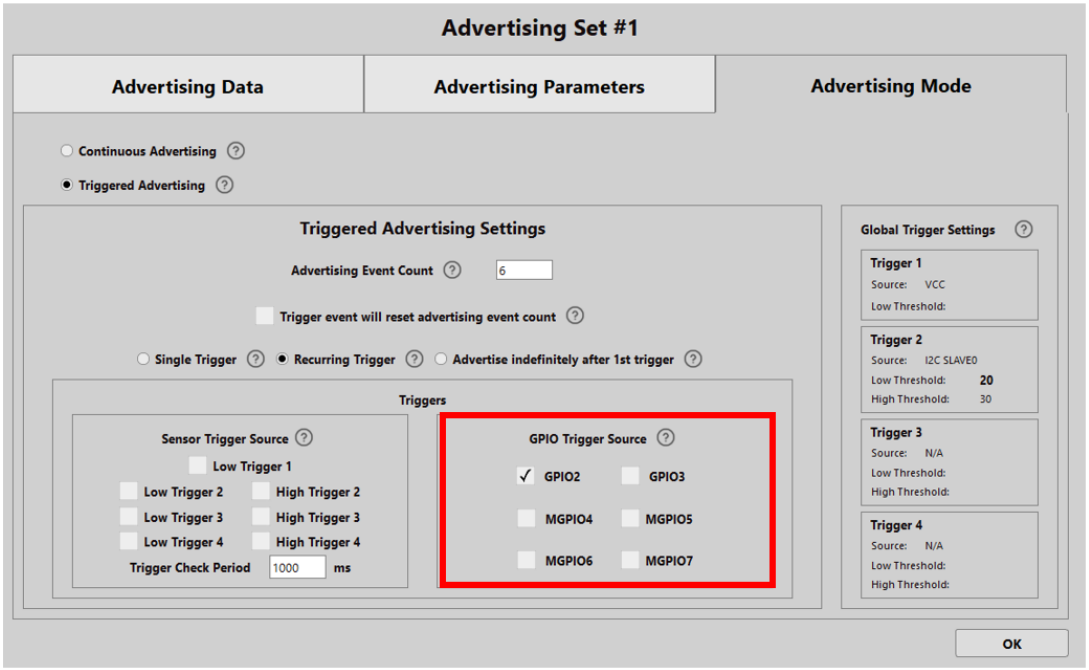

Figure 31 : Enable of GPIO input states as trigger

2.2 Analog channels and ADC configuration

2.2.1 Overview

There are up to 4 analog input channels (part number dependent) 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 power switch pads (part number dependent) for external analog device power control and automatic power supply when ADC data acquisition is required, as shown in Figure 32. The switch can be used to shut off the power supply completely to the external sensors when they are not used (such as when the device is in the sleep state).

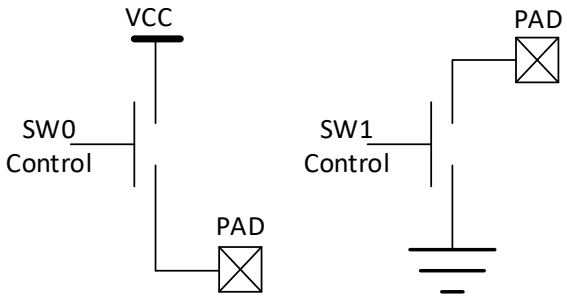

Figure 32 : Power switch control

2.2.2 Configuration

The GUI provides a flexible and simple ADC configuration tool. To enable any ADC channel – check the "Enable" box on the corresponding ADC channel(s) in the "ADC" tab as shown in Figure 33.

When "Enable" is checked for a 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. This can be done by setting the GPIO to analog (See section 2.5 for details).

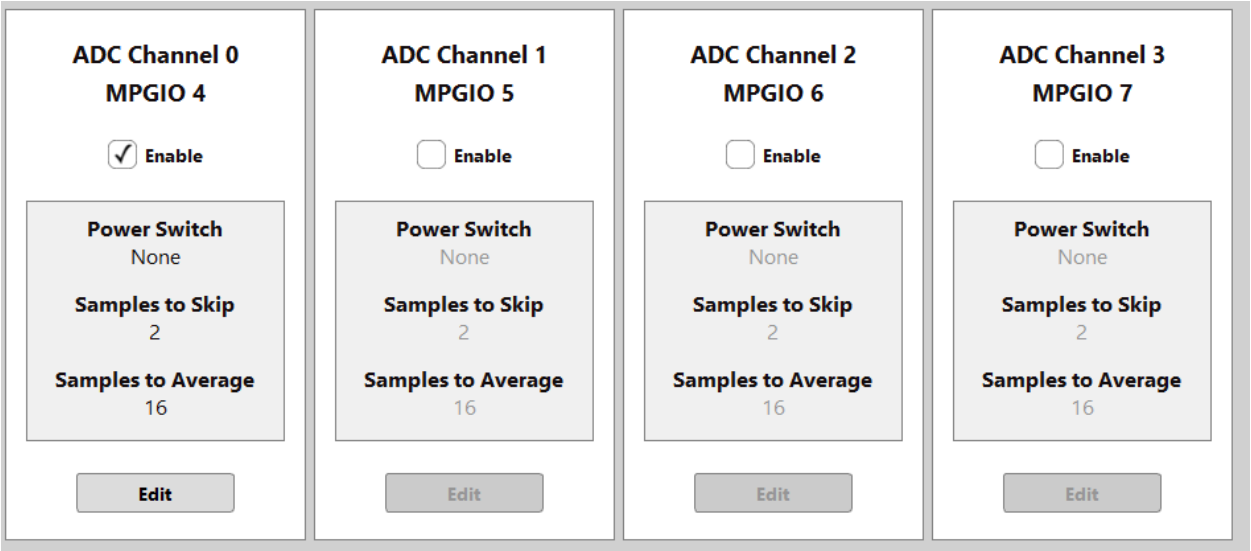

Figure 33 ：ADC configuration tab

For each ADC channel, the user can configure 1) Whether enable power switch; 2) Sampling configuration; 3) Unit mapping, as shown in Figure 34.

2.2.2.1 Power switch configuration

If the external analog device requires dynamic power control to shut the supply when it is not operating, then click the “Edit” button and check the corresponding box according to the circuit design.

2.2.2.2 Sampling configuration

For the ADC sampling parameters, it is recommended for the user to 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. By default, the ADC samples the input 18 times, and throws away the first 2 samples. The output is the average of last 16 samples.

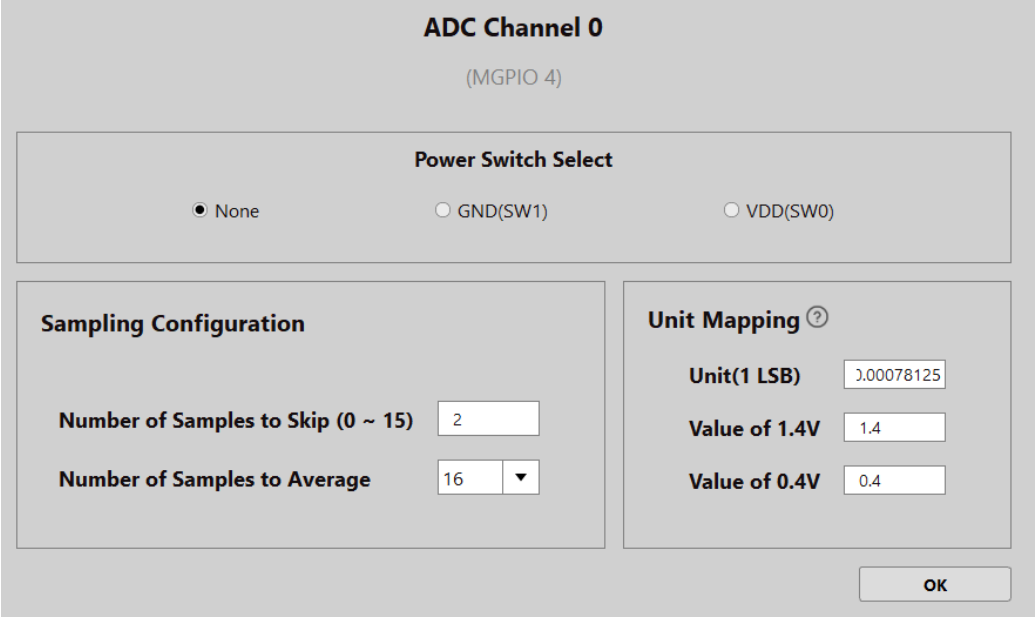

Figure 34 : ADC channel configuration

2.2.2.3 Unit mapping

After the ADC sampling, the resulting raw ADC code for an input, Vin, is:

$$\text{Raw ADC code} = \text{Vin}/\text{Vref} * 1024$$

where Vref is the internal reference voltage for the ADC and is equal to 800 mV. By default, the raw ADC code is used for advertising which means one LSB in the ADC code corresponds to 0.78125 mV. If the advertising data for ADC read out is 0x400 (1024 in decimal), then Vin is equal to:

$$\text{Vin} = \text{ADC code} * 0.78125 = 1024 * 0.78125 = 800\text{mV}.$$

The unit for the ADC data is configurable. For example, the user can change it to 0.5mV. In that case, if the ADC data is equal to raw ADC code *0.78125/0.5. If the ADC data in the advertising payload is 0x400 (1024 in decimal), then Vin is equal to:

$$\text{Vin} = 1024 * 0.5 = 512\ \text{mV}$$

If the measured physical variable is linear with the input voltage, Vin, then device can directly convert the measurement with the unit of the physical variable. The converted number is encoded in two's complement format. For example, a temperature sensor outputs 1.4V when the temperature is 100 Celsius degree, and outputs 0.4V when -10 Celsius degree. Additionally, 1 LSB (Least Significant Bit) is used in the advertising data to represent 0.1 degrees Celsius. 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. If the ADC data in the advertising payload is 0x3E8 (1000 in decimal), the corresponding temperature is equal to:

$$\text{Temperature} = 1000 * 0.1 = 100\ ^{\circ}\text{C}.$$

If the ADC data in the advertising payload is 0xFEC0 (65216 in decimal), the corresponding temperature is equal to:

$$\text{Temperature} = (65216 - 65536) * 0.1 = -32\ ^{\circ}\text{C}.$$

The ADC data is of 16-bit in 2's complement format. When the user is not using the default unit mapping in Figure 34, the user should take precautions to avoid over-flow.

For the triggered advertising, if the ADC CH0 is selected as the trigger source, the comparison uses the number after unit mapping instead of the raw ADC code. Take the above temperature sensor case as an example (the temperature sensor outputs 1.4V when the temperature is 100 Celsius degree, and outputs 0.4V when -10 Celsius degree, and the user wants to have a 0.1 degrees resolution in digitized sample). If the user wants to have a high threshold of 56°C, the user needs to enter 560 in the threshold edit box as shown in Figure 30. If the user wants to have a low threshold of -21°C, the user needs to enter -210 in the threshold edit box.

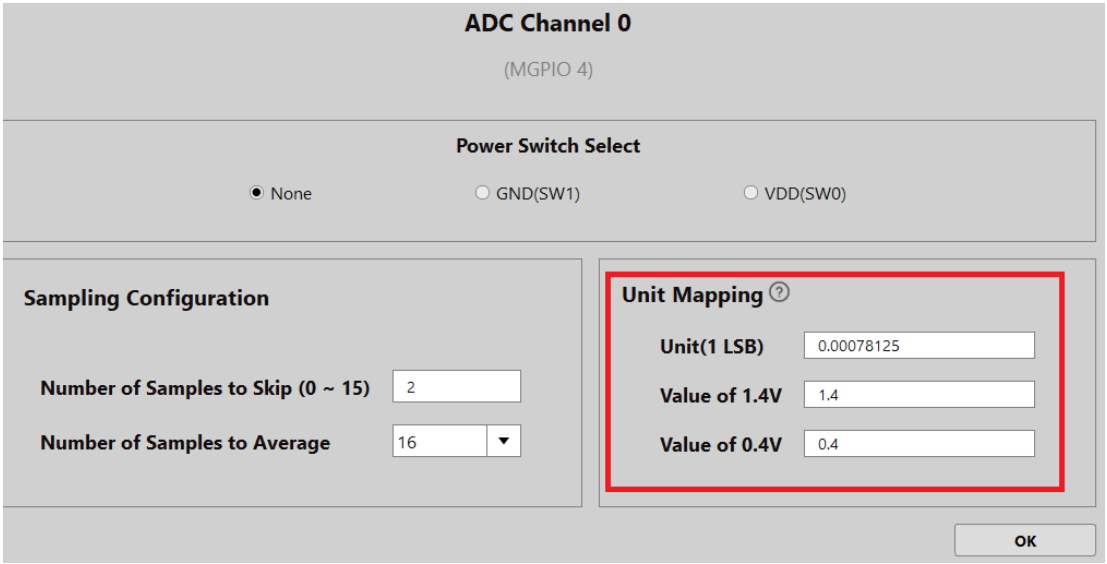

Figure 35 : Unit mapping setting

2.3 GPIO edge count

The device supports counting of rising edges on a selected GPIO, as shown in Figure 36. 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 users can have the count added to the advertising payload. Both the minimum time of tH and tL (shown in Figure 36, the durations of the input signal on high level and low level, respectively) must be larger than 1 ms.

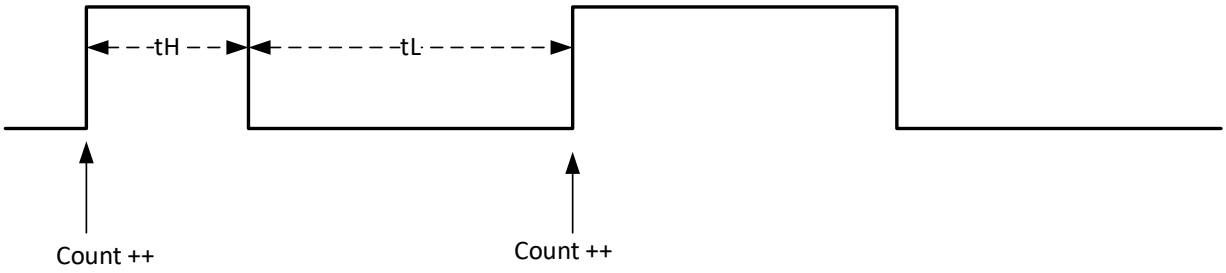

Figure 36 : GPIO edge counting

The user can configure the input GPIO source through GPIO Edge Count tab as shown in Figure 37.

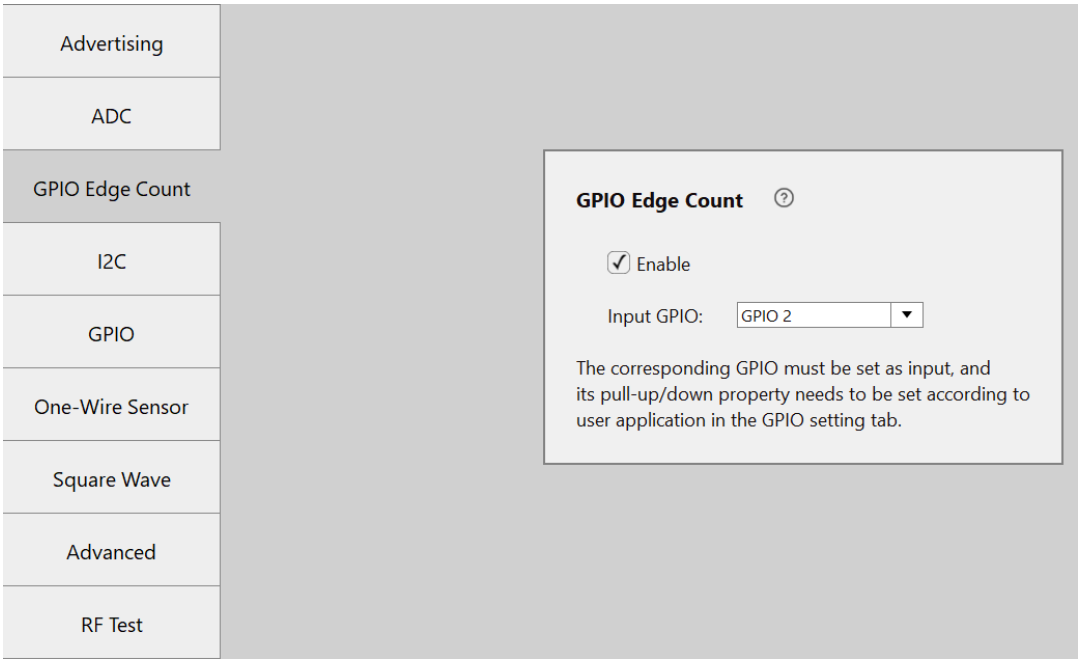

Figure 37 : GPIO edge count configuration

The input GPIO that will count the edges can be selected from the pull-down list in Figure 37. Please note that the selected GPIO need to be set as "Input" in the GPIO configuration tab.

2.4 I2C configuration

The NanoBeacon Config Tool provides the user with an easy way to configure the external I2C slave devices through I2C configuration tab window as shown in Figure 38.

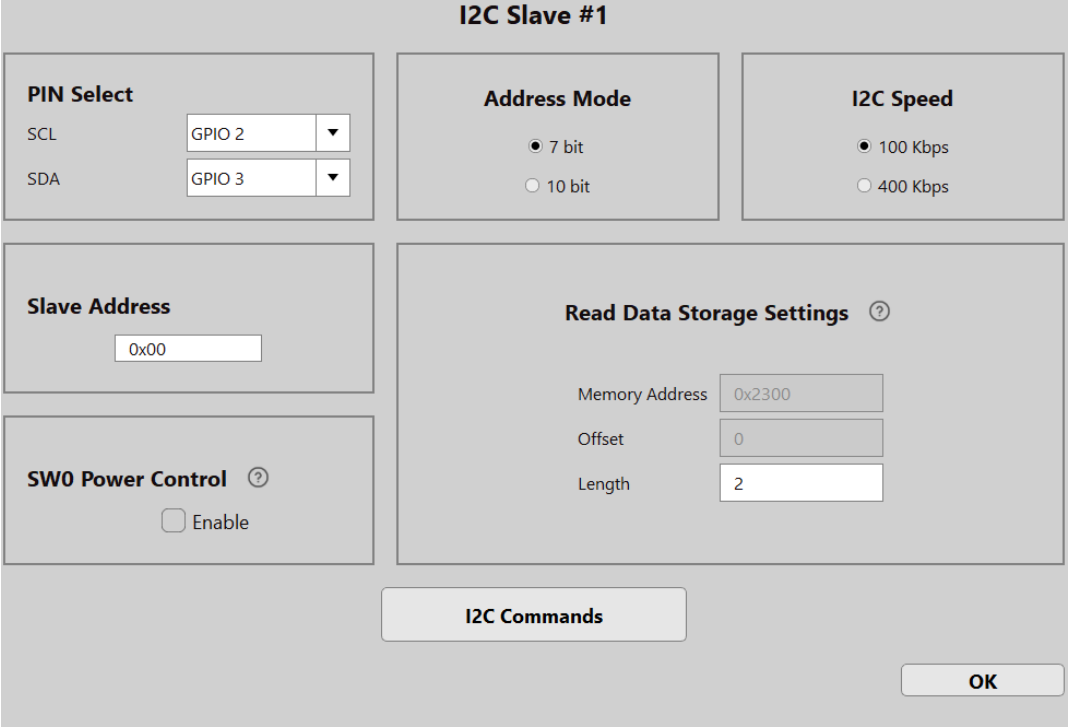

Figure 38 : I2C configuration tab

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

- PIN Select: Select the pins for the SDA and the SCL for the I2C devices. GPIOs 2/3/4/5/7 can be used as the SCL and SDA for the 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.
- SW0 Power Control: When enabled, the SW0 will go to ground when the IN100 is asleep, and provide power when the IN100 wakes up. This can be used to provide power to the external I2C sensor.
- Read Data Storage Settings: I2C read will store the data into the device's internal memory with the starting address at 0x2300. The Offset is the offset of the address with the default setting being 0 for I2C Slave #1, 16 for I2C Slave#2 and 32 for I2C Slave #3 respectively. The Length is the length of data read from the I2C each advertisement event in bytes. For example, if the user needs to read 6-bytes every time the IN100 wakes up, this value would be 6.

I2C Commands

To drive an external I2C slave device, an I2C communication protocol needs to be defined and configured accordingly. The NanoBeacon Config Tool provides the "I2C Commands" input section for this purpose.

Figure 40 shows the main configuration window for I2C Commands input. There are 4 commands available:

- I2C read: For this command, the user must specify the number of readings (r_len) in bytes that the IN100 will read from the slave. The maximum r_len is 5. If more than a 5 byte read is needed, the user needs to configure the I2C to read multiple times separately.
- I2C write: For this command, the user must specify the byte sequence written to the slave in the GUI.
- I2C write_stop_read: For this command, the user must 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 user can use this command to insert a delay or wait time between or before I2C commands.

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 a typical application, the device switches its state between wakeup/active and sleep, as shown in Figure 39 . The first time going active upon power-on is called a cold boot, and the wakeups after sleep are called warm-boot.

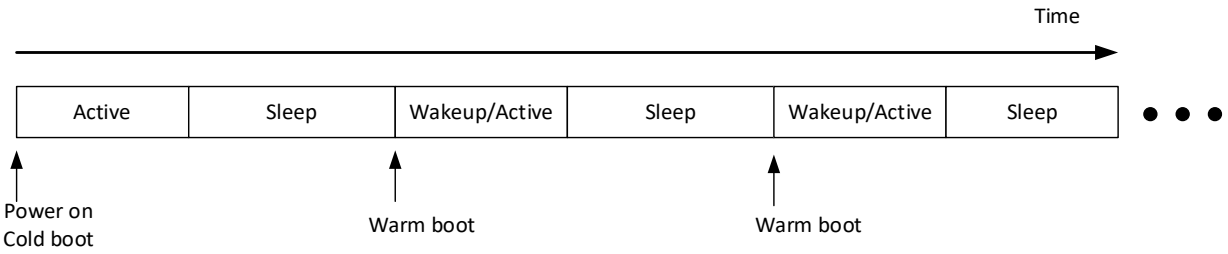

Figure 39 : IN100 states

Execute I2C command when cold boot: Execute I2C read/write command during a 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 during a warm boot. After advertising, the device will enter a sleep mode. When the device needs to advertise again, it will perform a warm boot and wakeup before advertising.

If the commands are only needed to be executed during cold boot, then only the cold boot box needs to be checked.

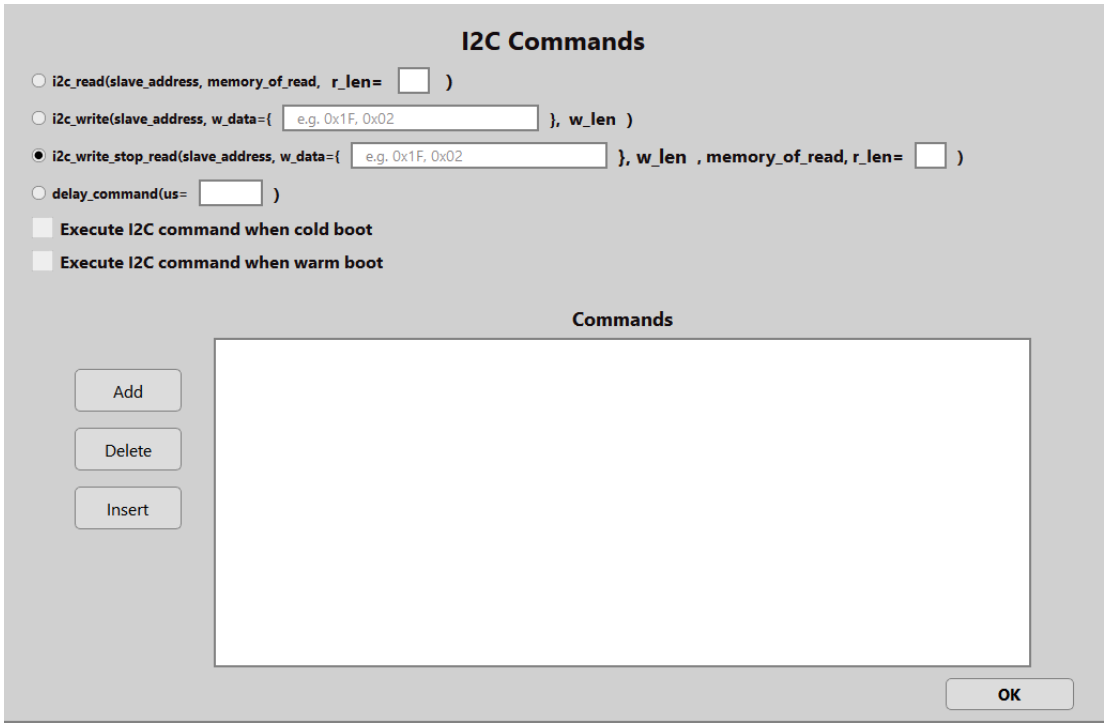

Figure 40 : I2C commands configuration window

The steps to configure I2C are as follows:

- Select a command, and enter the parameters for that command
- Select the execution condition (by checking/unchecking the cold boot or warm boot box)
- Click the "add" button to add the command to the configuration.
- Repeat the above process, adding additional commands 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" of the corresponding I2C slave device.

- I2C Slave #1 Read Data: the data of an 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 added to the payload with small or big endianness and encryption options. Only the selected read data will be advertised on the data payload. An example is shown in Figure 41. The offset is 4, and number of bytes is 3. Given these settings, the device will only advertise 3 bytes of I2C read data starting from the address of 0x2304.

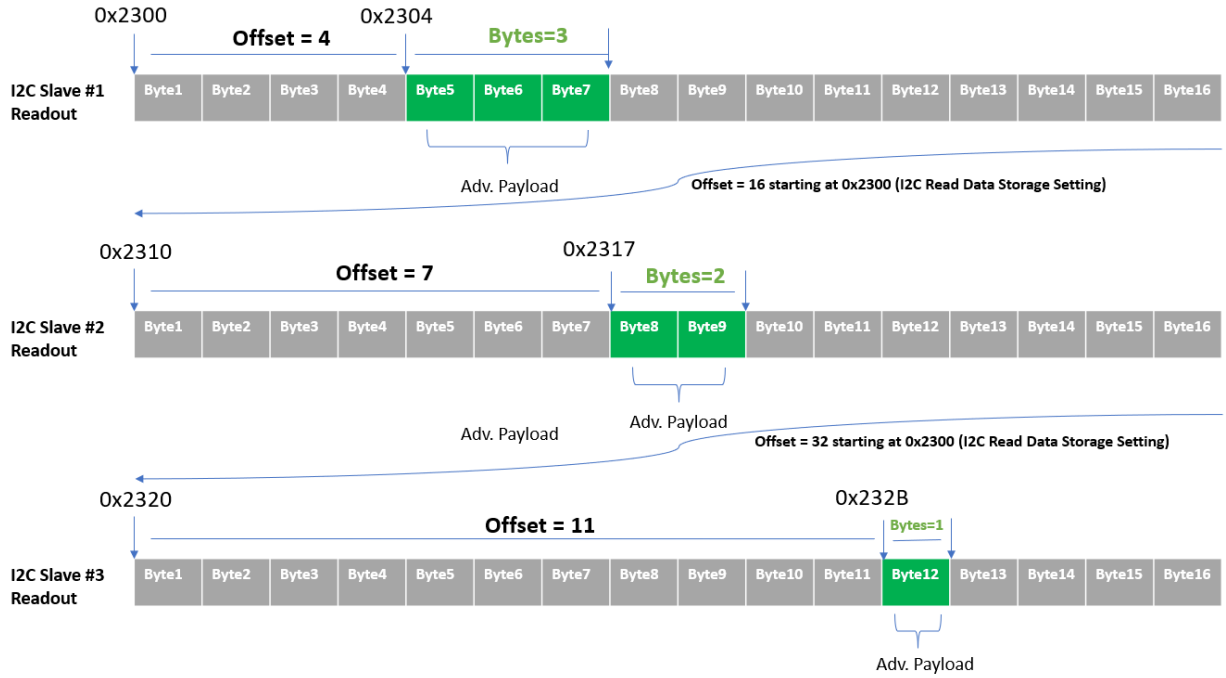

Figure 41 : I2C read data as adv. payload

- I2C Slave #2 Read Data the data of an I2C Slave #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 added to the payload with small or big endianness and encryption options. Only the selected read data will be advertised on the data payload. An example is shown in Figure 41.
- I2C Slave #3 Read Data: the data of an I2C Slave #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 added to the payload with small or big endianness and encryption options. Only the selected read data will be advertised on the data payload. An example is shown in Figure 41.

2.5 GPIO configuration

The GPIO configuration tab window is shown in Figure 42.

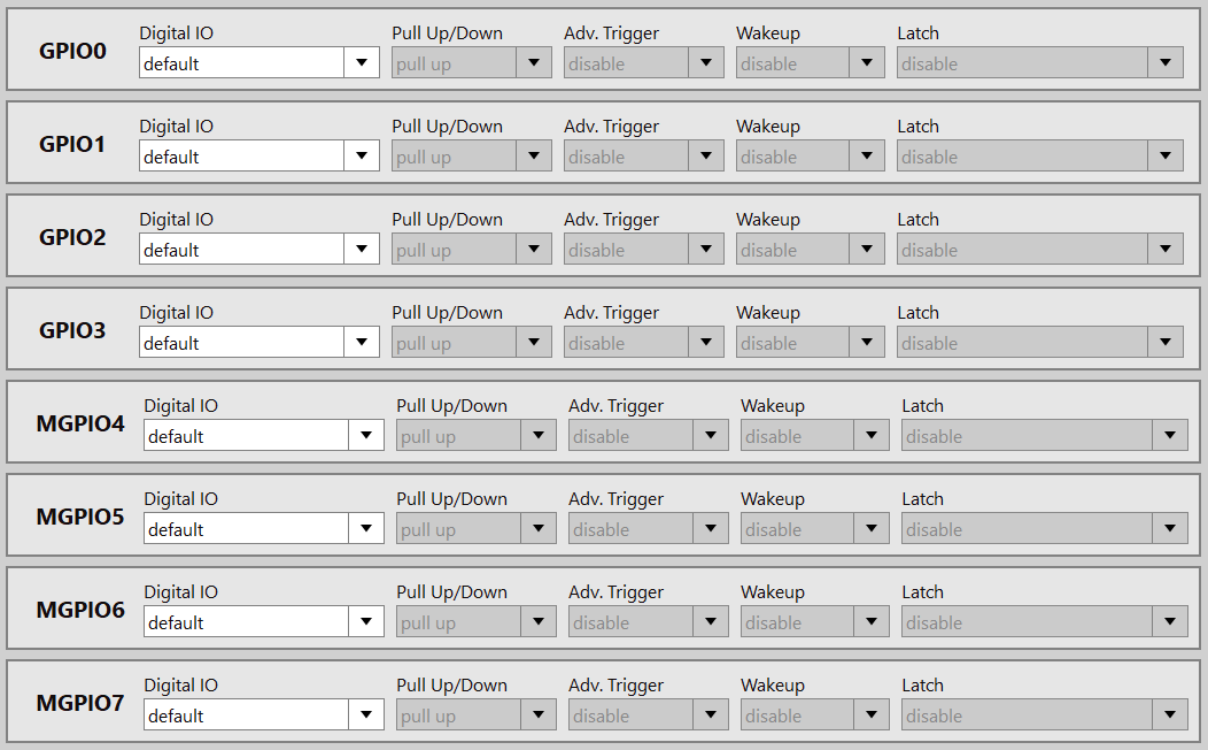

Figure 42 : GPIO configuration tab

The GPIO configuration options are input, output, analog, and default. For the output options, the GPIO can be configured as output low, output high, “wakeup high, sleep low”, or “wakeup low, sleep high.” Output low/high means the GPIO will act as an output and will either pull low or high. “Wakeup high, sleep low” will make the GPIO pull up when the device is awake and off when asleep. “Wakeup low, sleep high” is the reverse with the pin pulling low when the device is awake and high when it is asleep.

When configured as an input, the GPIO will be used either as an input for edge counting or as an input for an external trigger source as covered in section 2.1.9.2.

Pull-up/down options are pull up, pull down and disable. These options correspond to pull up, pull down, or neither pull up nor pull down behavior.

Analog 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, rising edge and falling edge, which correspond to no trigger, high level, low level, rising edge, or falling edge trigger advertising. Refer to section 2.1.9.2 for more information on configuring the GPIOs as inputs. To enable GPIO ADV trigger, the corresponding GPIO needs to be set to input.

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 application cases, the config tool 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 set as analog, 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.

2.6 One-wire sensor interface configuration

The device integrates a one-wire (single-wire) pulse count controller and interface for interfacing to a one-wire pulse sensor. There are many sensors which convert the measured physical quantity into a number of pulses. The one-wire controller can provide power management and 16-bit pulse counting at the same time. Figure 43 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 review the datasheet of one-wire sensor that is being used). In the figure, tp is the period of a pulse, typically from a few microseconds (us) to a few tens of microseconds. 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.

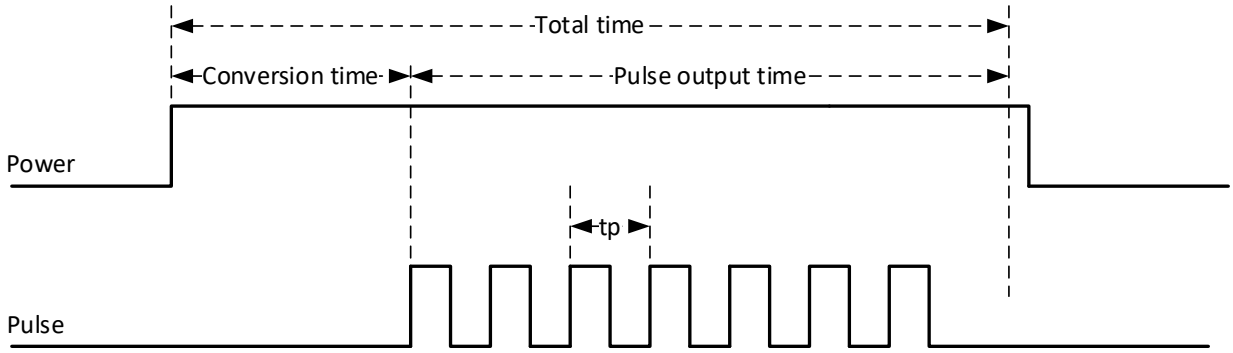

Figure 43 : Power management and pulse counting

The one-wire pulse count related settings are shown in Figure 44. To enable the controller: check "Enable" box. If the external sensor needs dynamic power control, then check the corresponding box according to the application circuit design. The user can also configure the GPIO connected to the one-wire sensor output. The user can adjust the timing parameters in the Timing Control Settings based on the one-wire sensor specification.

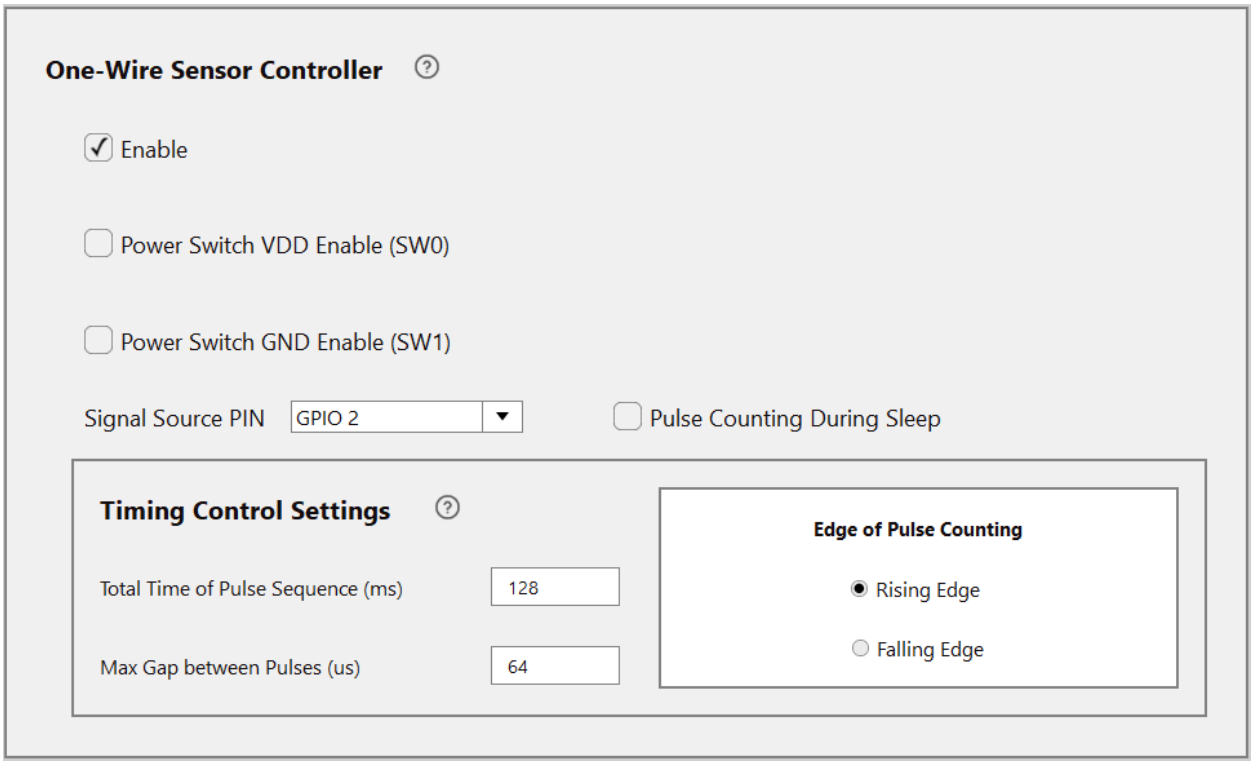

Figure 44 : One-wire count configuration tab

2.7 *Square wave settings*

The square wave tab appears as shown in Figure 45.

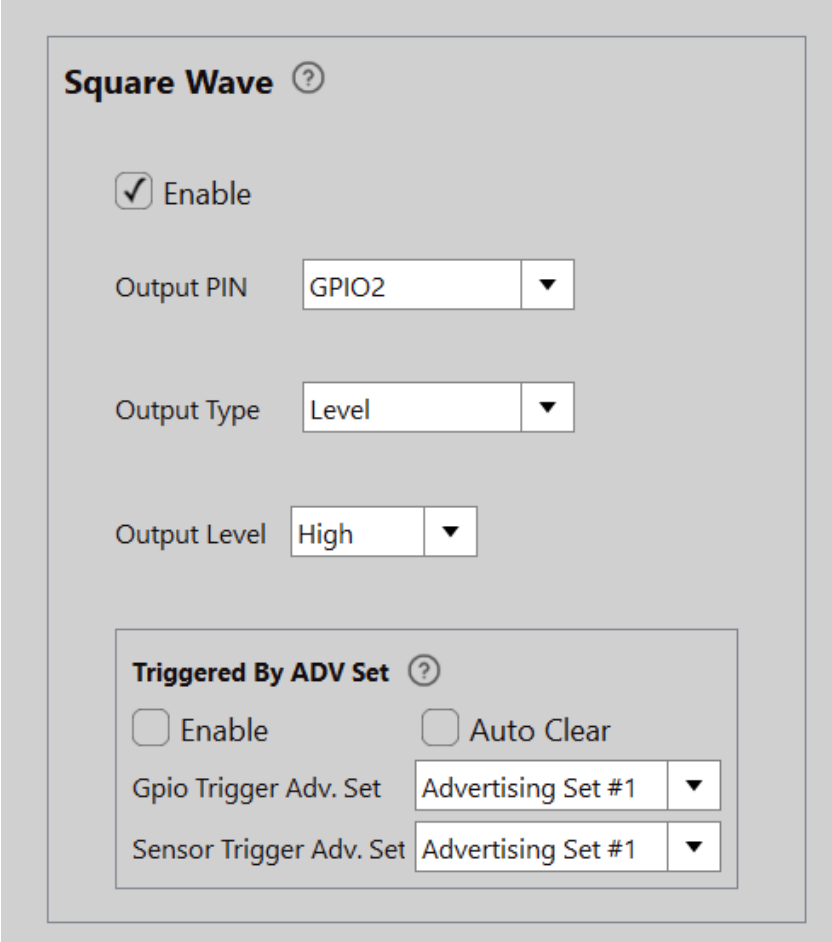

Figure 45: Square wave settings

The square wave will output on either GPIO 0, 2, or 3. Note that which GPIOs are free to use for this feature depend on the packaging of the IN100. For example, GPIO 0 is reserved for the UART on the QFN18 package, while the WLCSP package has GPIO 0 free. Which GPIOs are reserved is denoted in the IN100 data sheet.

The settings available for the square wave are as follows:

- Enable: Enables the square wave functionality when checked.
- Output PIN: Select which GPIO PIN outputs the square wave. The options are GPIO 0, 2, and 3.
- Output Type: Select between a level output of either high or low, or a square wave with a selectable frequency.
- Output Level: Only available when Output Type is set to Level. Choice is either a high or low output level.

When the output type of square wave is “Square Wave”, the user can configure the frequency of square wave as shown in Figure 46. The options range from 16 kHz at maximum and 62 Hz at minimum. The options decrease from 16 kHz by a factor of 2 each step, such as 16 kHz, then 8 kHz,

then 4 kHz, and so on down to 62 Hz. If RTC crystal is not installed for the IN100 device, the root clock of the square wave is from an RC oscillator. It can suffer a drift of +/-50%.

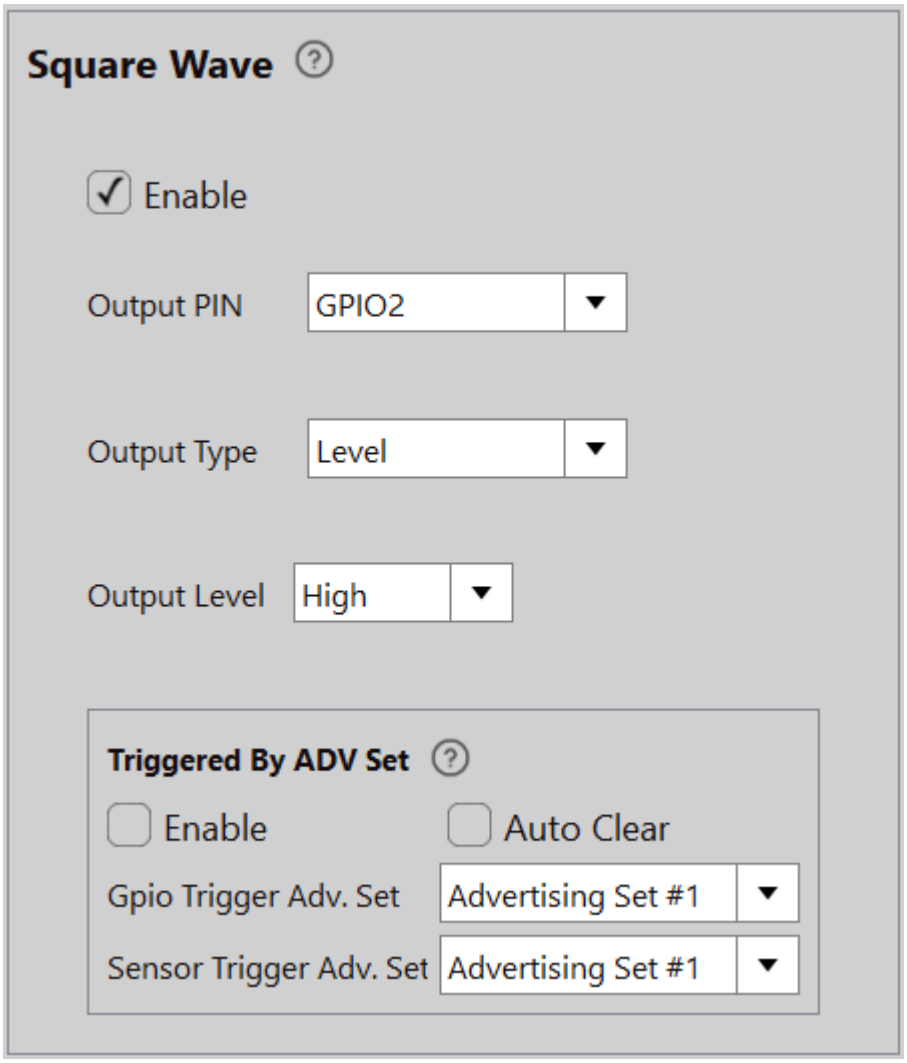

Figure 46: Square wave settings with square wave output type

Lastly, there is a functionality to trigger and disable the square wave output via a triggered advertising set, as shown in Figure 47. This is useful if the square wave output is needed to blink an LED in accordance with an advertising event or other similar uses.

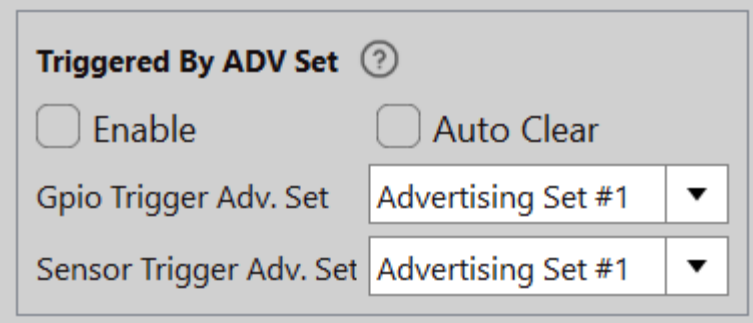

Figure 47: Square wave settings triggered by advertising set

The triggered by advertising set options are as follows:

- Enable: Enables the feature to trigger the square wave with a triggered advertising set when the box is checked.
- Auto Clear: When this box is checked, the square wave output will clear after the triggered advertising set is finished broadcasting.
- GPIO Trigger Adv. Set: Select which triggered advertising set using a GPIO as its trigger source will cause the square wave to output.
- Sensor Trigger Adv. Set: Similar to the GPIO option, select which triggered advertising set triggered by a sensor data source will cause the square wave to output.

2.8 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.

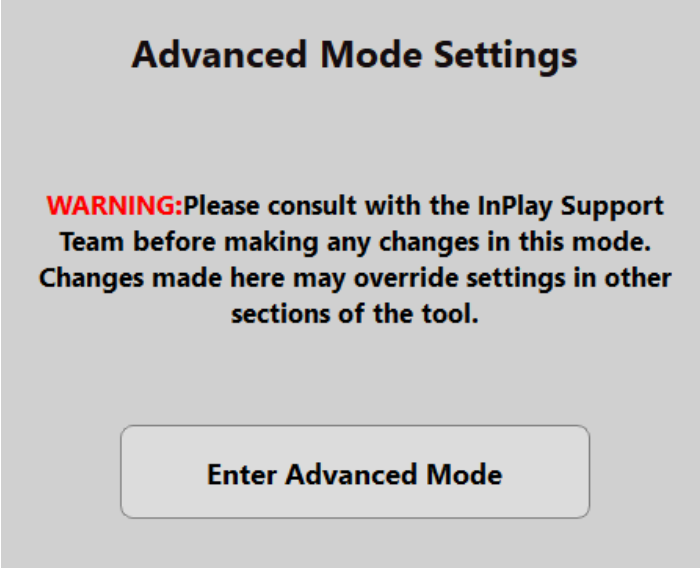

Figure 48 : Advanced mode settings

As shown in Figure 48, 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.

2.9 XO Settings

The IN100 device provide on-chip load capacitance for the external 26MHz XO crystal. The config tool provides a few configurable settings which include the Internal Capacitor Code, the Stable Time, and the Strength Code, as shown in Figure 50.

The Internal Capacitor Code refers to the configurable on-chip load capacitance for the external 26 MHz crystal. To minimize the frequency offset of the XO and ensure its reliability, it is important to configure the Internal Capacitor Code to meet crystal specification. Figure 49 illustrates the XO circuit. The external capacitors are denoted as Cp and Cn. Assume Cp is equal to Cn which is equal to Ce (equivalent capacitance), then the load capacitance of the crystal, CL, can be calculated as:

$$CL=(Ce+Ci)/2+Cstray$$

where Cstray is the parasite capacitance which is usually about one to two pF (dependent on the PCB and layout). The GUI settings can tune Ci to minimize the frequency offset. The internal Ci is programmable (in the XO tab, shown in Figure 50) as the Internal Capacitor Code from 0.5 pF (code 0) to 16 pF (code 15) with a step size of 0.5pF. In terms of load capacitance, the internal load range is from 0.5 pF to 8 pF.

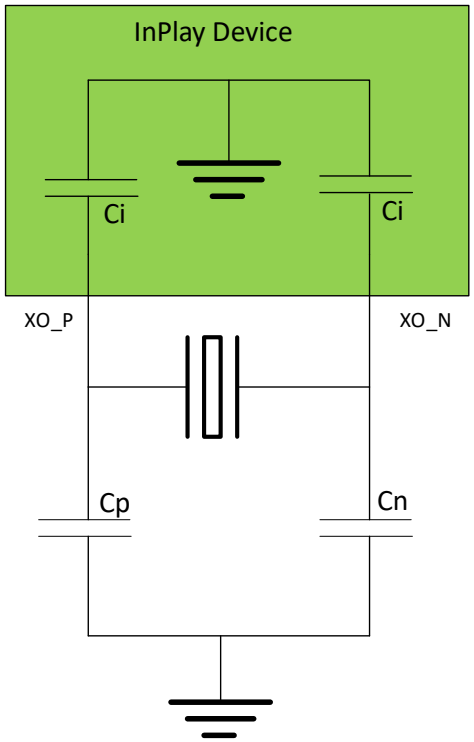

Figure 49 : XO Circuit

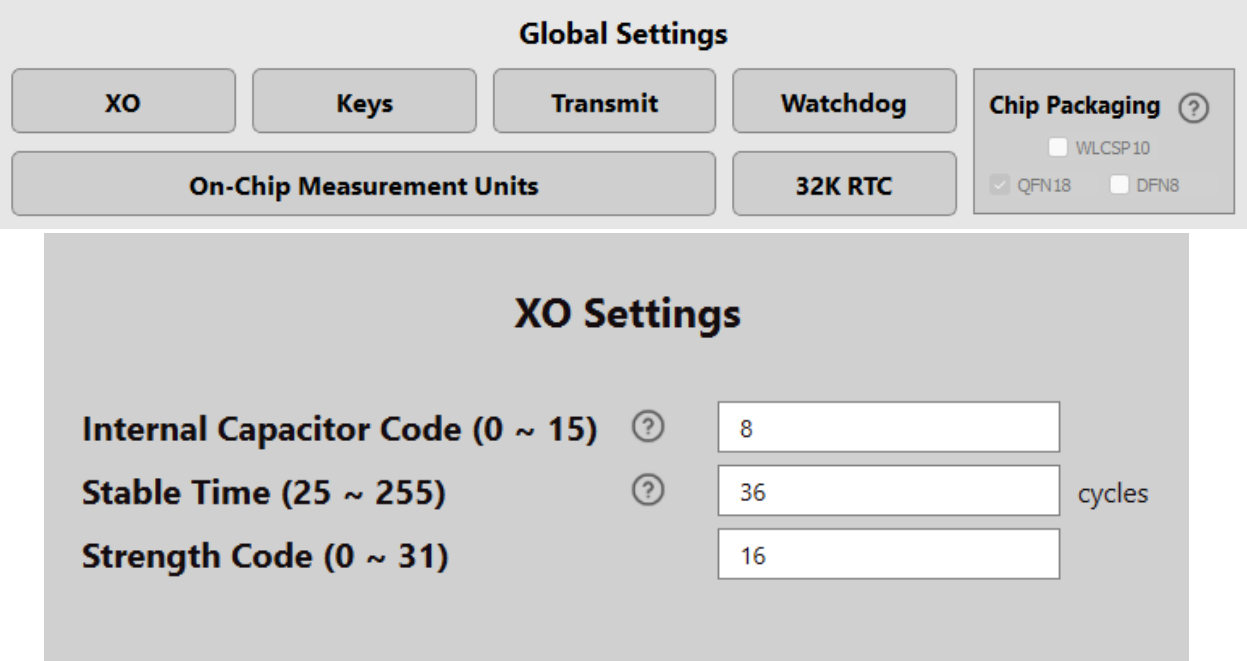

Figure 50 : XO tab and XO setting configuration

Upon enable, the XO takes time to oscillate as shown in Figure 51. The "Stable Time" shown in Figure 50, has a unit of 32 µs per cycle. The user needs to make sure the stable time entered in the configuration is larger than the actual measured XO start-up time. To measure the startup time, it is recommended to use a low load probe on the negative side of the XO pins (XO_N pin). Any additional parasitic capacitance can influence the results. Startup time is defined as the time from when the XO output first goes high, to when the waveform amplitude reaches 400 mV. Figure 51 shows a simplified diagram for the startup time similar to what would be observed on an oscilloscope.

The Strength Code value influences amplitude of the XO waveform. When the XO starts up, there is a brief period of time where the amplitude is higher, before it settles further and lowers in amplitude until the IN100 is finished advertising and goes to sleep. The amplitude during this awake time after startup and before ending must be at or above 500 mV in amplitude to ensure operation. Larger strength code results in larger amplitude.

XO Settings Disclaimer: In most cases, it is not recommended for the customer to evaluate the XO by themselves. Instead, InPlay provides a list of tested crystals that have been verified by the crystal manufacturer to work with the IN100. Although the XO settings are tunable through the GUI, it is difficult for anyone but the crystal manufacturer to do more specific tests such as an oscillation margin test or testing multiple samples over various temperatures and voltages. The list of approved crystals is found on the InPlay NanoBeacon GitHub page found here: https://github.com/NanoBeacon/config-files/tree/main

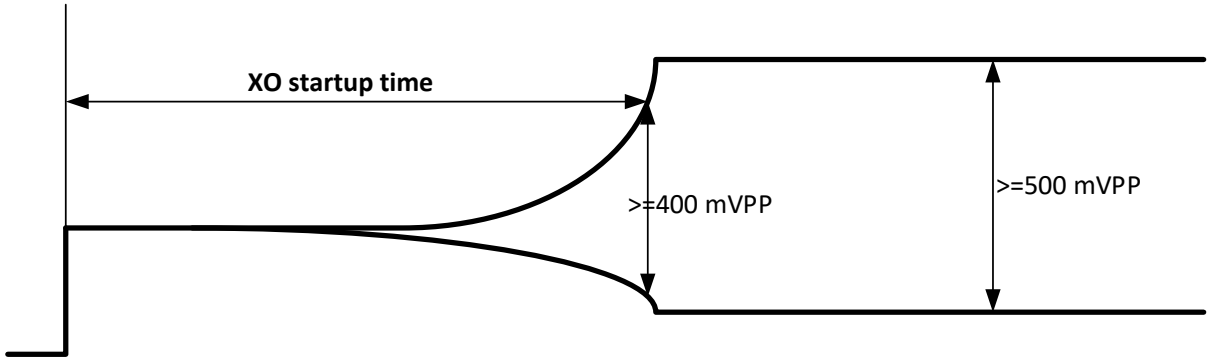

Figure 51: XO startup

2.10 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 52.

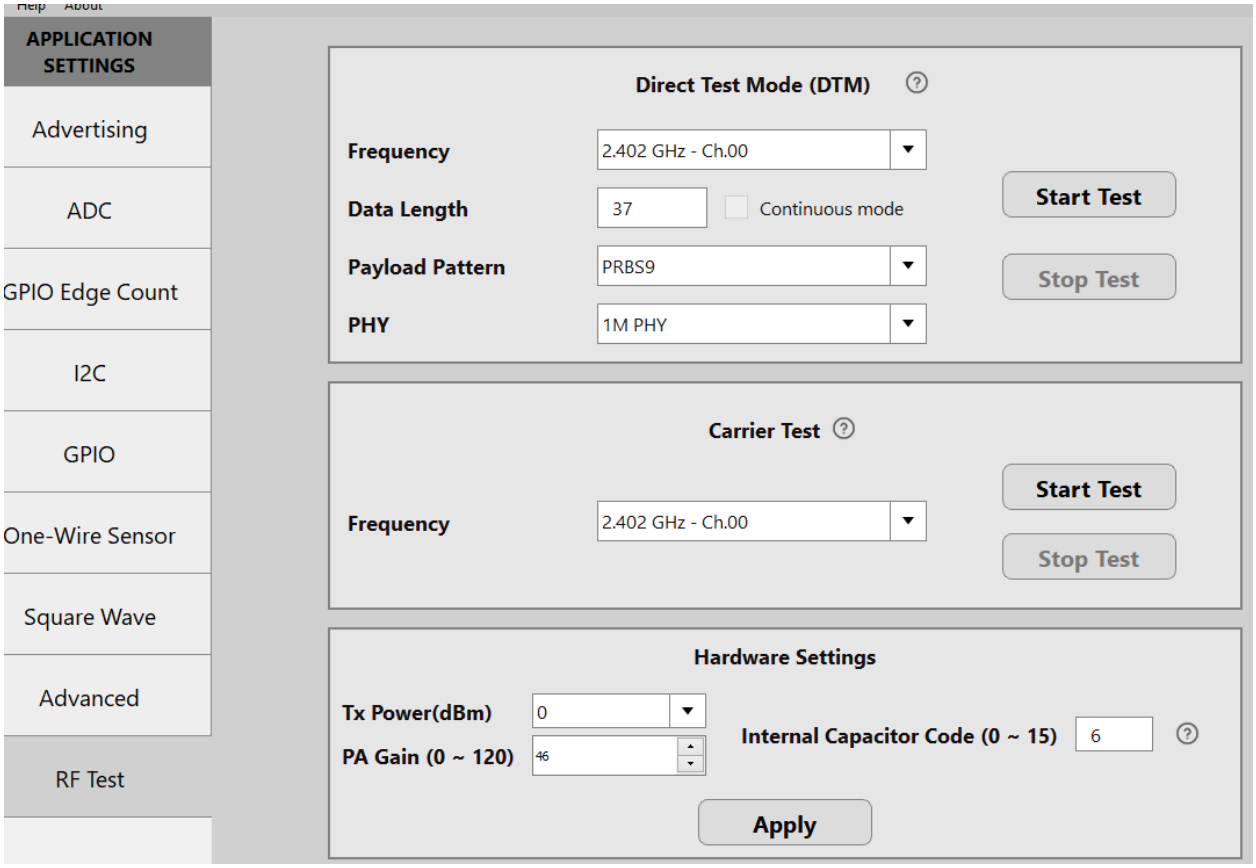

Figure 52 : RF test interface

Direct Test Mode (DTM)

The RF validation of Bluetooth device uses a protocol called Direct Test Mode as specified by the Bluetooth SIG [1]. 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. For continuous modulated testing, the user needs to check the "Continuous mode" box.

Carrier Test (Continuous Waveform Test)

The user needs to configure the frequency and TX power from pull-down lists.

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. Programming OTP Memory

3.1 Before you start

To program the IN100 OTP memory, the VDDQ pin (PIN6 for DFN8 in Figure 53, PIN13 for QFN18 in Figure 54 and PINA3 for WLCSP10 in Figure 54) is required to be powered by a stable 3.3V power supply, otherwise the OTP memory will not be programmed correctly.

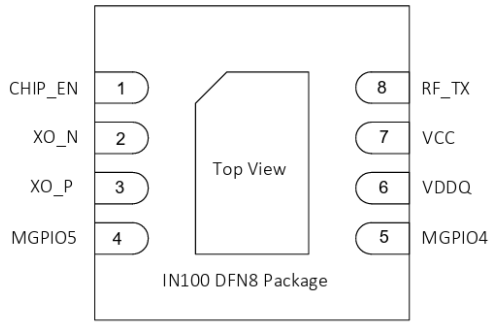

Figure 53 : DFN8 package

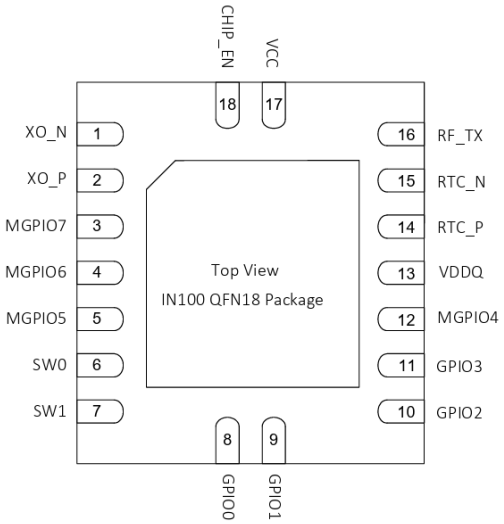

Figure 54 : QFN18 package

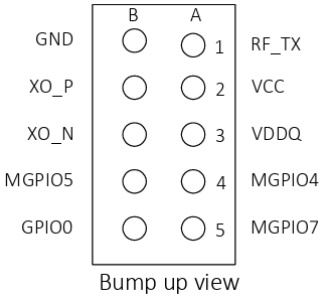

Figure 55: WLCSP10 package

Below are the steps of OTP memory programing example with NanoBeacon™ Development Board.

- Connect the development board and OTP programmer board as shown in Figure 56.
- Connect the OTP programmer board to the PC with a USB cable. Make sure that the OTP switch on the programmer board is on state.
- Open NanoBeacon™ config tool on the PC 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, as shown in Figure 57.

- Configure the advertising sets and peripherals.
- Verify if the configuration meets the application needs.
- Press the "Burn/Program" button to program the OTP memory.

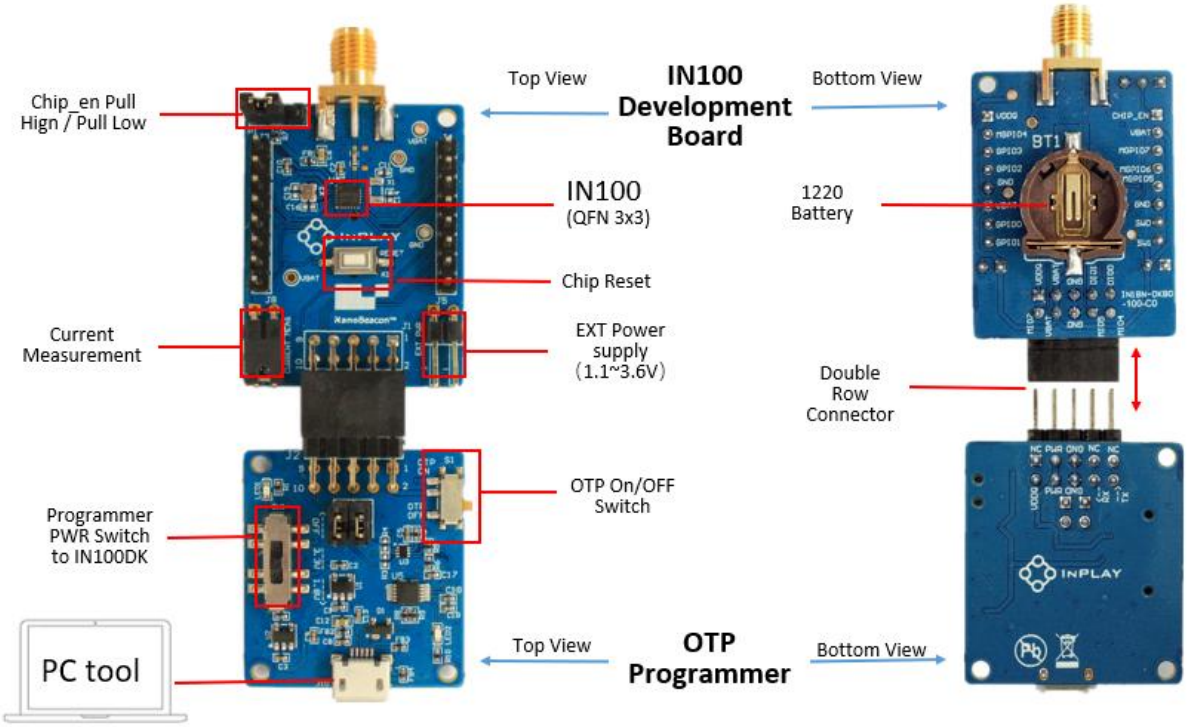

Figure 56 : Programming with NanoBeacon™ config tool

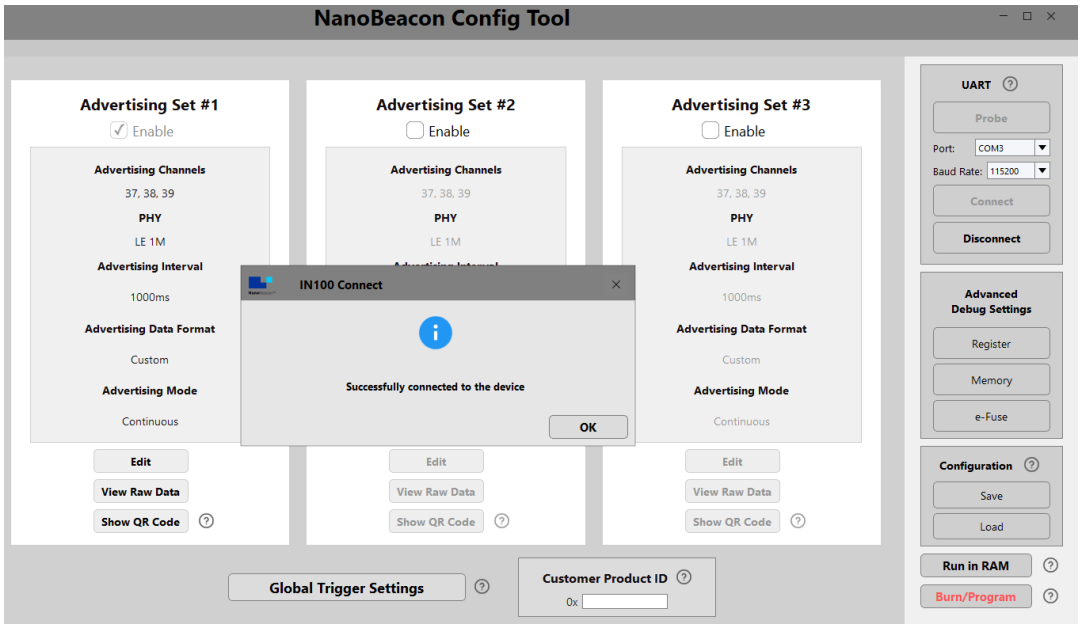

Figure 57 : UART communication status report dialog

3.2 Verification

After the user finishes the configuration, the user can use “Run in RAM mode” for verification and debug. Note that “Run in RAM mode”” does not I2C operation.

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.

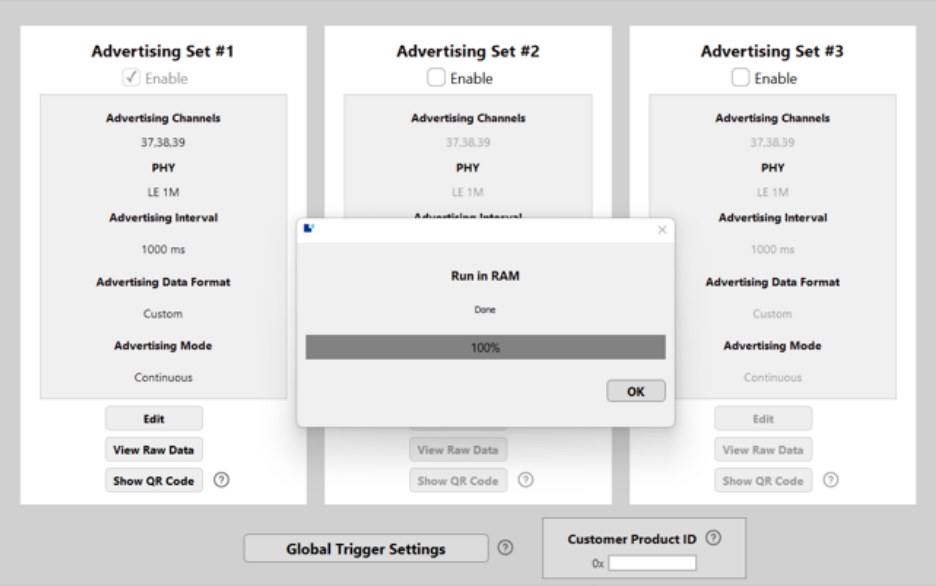

Figure 58 : RAM run mode status report

3.3 OTP memory programming

After debugging and verification, user can program the configuration to the device with confidence.

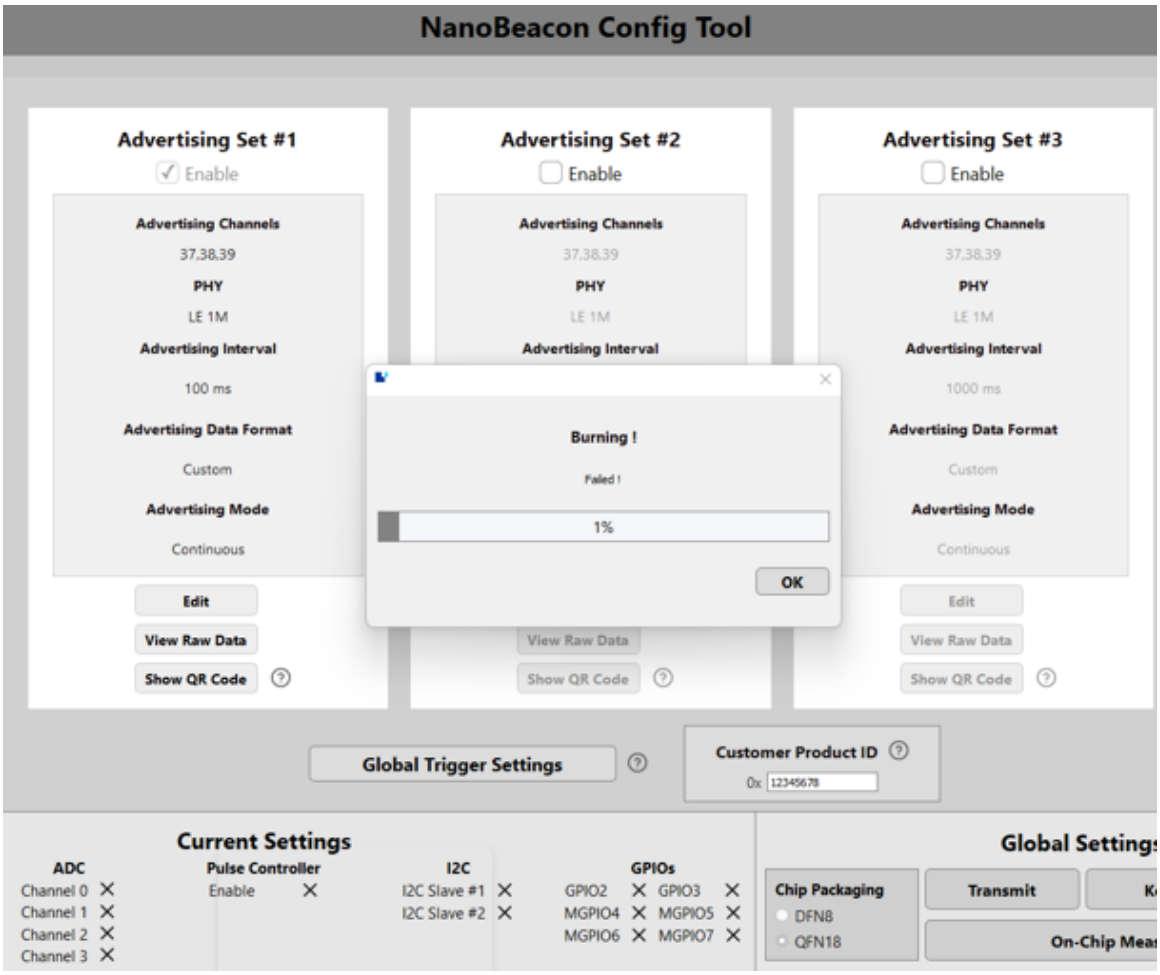

Figure 59 : Burning in progress

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 59, and there will be a pop-up box for successful burning report or a pop-up box for failed burning report just in case.

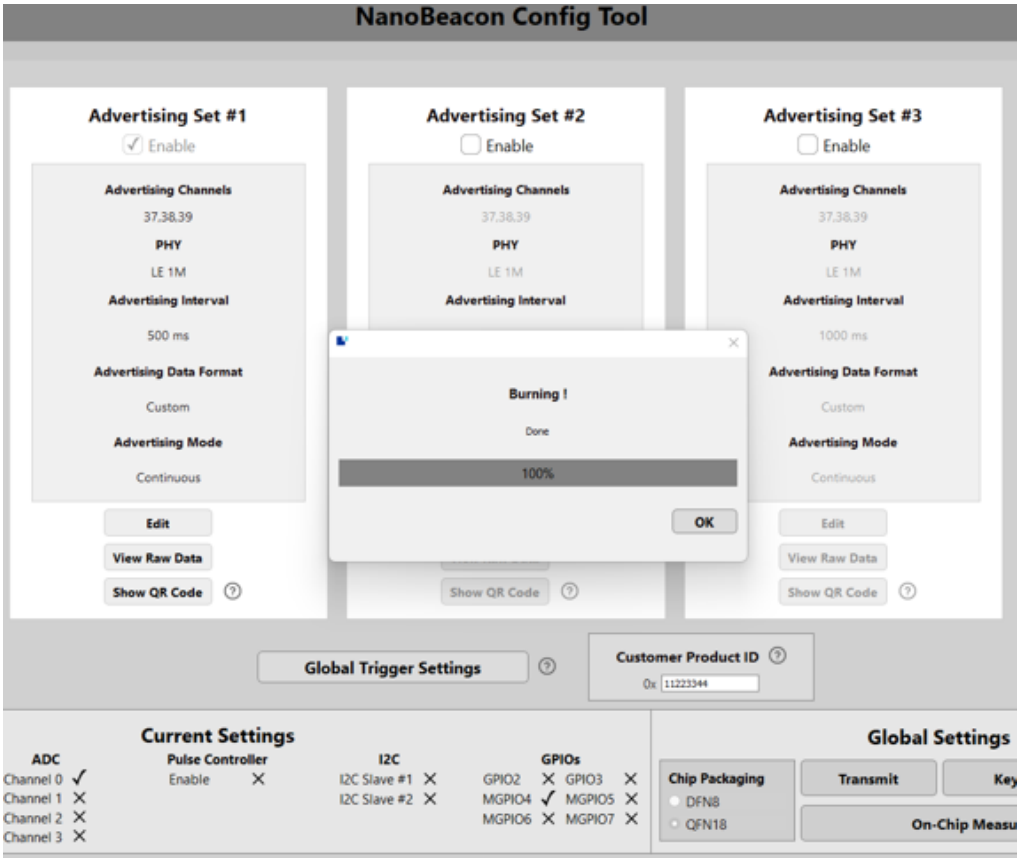

Figure 60 : Programming progress complete

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
Ver 1.05	Added XO setting.	J. Wu	2022-08-10
Ver 1.06	Added Square Wave Settings	A. Martens	2024-02-14
Ver 1.07	Fixed issue in I2C section with description of I2C write	A. Martens	2024-11-22
Ver 1.08	Added Hubble settings	A. Martens	2026-10-23

5. Reference

[1] "Bluetooth Core Specification V5.3", at https://www.bluetooth.com/.
[2] "Supplement to the Bluetooth Core Specification V11", at https://www.bluetooth.com/.
[3] "Assigned Numbers", at https://www.bluetooth.com/.
[4] "EAX: A Conventional Authenticated-Encryption Mode", M. Bellare, P. Rogaway and D. Wagner. 2003.

6. 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.