

# User Manual Document

## Fast Beam Switch Controller IP on NI FPGA for BBox™

---

V0.0.3

CT Fang ([ct\\_fang@tmytek.com](mailto:ct_fang@tmytek.com))

Tobias Huang ([tobias\\_huang@tmytek.com](mailto:tobias_huang@tmytek.com))

Alin Su ([alin\\_su@tmytek.com](mailto:alin_su@tmytek.com))

## Change history

Version	TLKCore API version	Sample code version	Author	Date	Description
V0.0.1			Tobias Huang	2021/11/25	Initial version.
V0.0.2			Tobias Huang	2022/02/17	Update pin define about TX_EN/RX_EN
V0.0.3	V3.3.16.3	V1.0.0.5	Alin Su	2022/09/23	Add customized beam - channel control

## System Overview

This is a system integration solution for reducing beam switch time down to hundreds of nanoseconds (ns). The demonstration contains a single BBox One/Lite and is executed by NI LabVIEW.


- Hardware Requirements :
- NI USRP29xx

TMYTEK BBox Series

## Getting Start

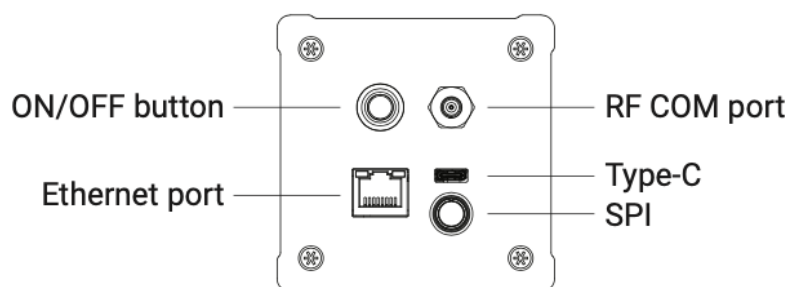
### A. Hardware Requirements and Configuration

1. Connect AUX IOs of USRP 29xx following the pin defined below to TMYTEK proprietary cable

AUX I/O Connector	Pin	NI-USRP Terminal Name	USRP RIO (LV FPGA) IO Node Terminal Name	TMYTEK SPI Definition
	1	+3.3 V	+3.3 V	Don't care
	2	GPIO 0	AUX I/O 0	GND( *Note : Please also connect SPI_SDI pin to this IO)
	3	GPIO 1	AUX I/O 1	SPI_LDB
	4	GPIO 2	AUX I/O 2	SPI_CSB
	5	GPIO 3	AUX I/O 3	SPI_CLK
	6	GPIO 4	AUX I/O 4	SPI_PDI
	7	GPIO 5	AUX I/O 5	SPI_SDO

AUX I/O Connector	Pin	NI-USRP Terminal Name	USRP RIO (LV FPGA) IO Node Terminal Name	TMYTEK SPI Definition
	8	GPIO 6	AUX I/O 6	TX_EN
	9	GPIO 7	AUX I/O 7	RX_EN
	10	GPIO 8	AUX I/O 8	Don't care
	11	GPIO 9	AUX I/O 9	Don't care
	12	GPIO 10	AUX I/O 10	Don't care
	13	GPIO 11	AUX I/O 11	Don't care
	14	0 V	0 V	Don't care
	15	0 V	0 V	Don't care

2. Connect the proprietary cable to BBox 5G device for external SPI control
3. Connect USRP and BBox 5G by ethernet cable



### B. Software Requirements and Configuration

<b>Operation System</b>	Win 10 or later
<b>LabVIEW</b>	2019 or later
<b>Python</b>	3.7.0 or later
<b>.NET Framework</b>	4.6.1 or later

### C. Remote Programming Setup and Interface

Remote programming and operation of the instrument is accomplished via the Ethernet. The following sections provide information about the interface connections, cable requirements, and remote operation setup.

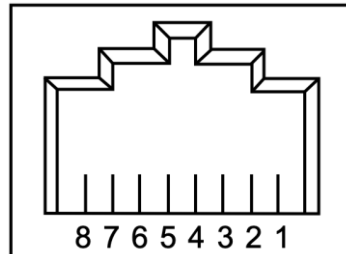
The BBox series device uses Ethernet to communicate remotely with a controller. Most instrument functions (except power on/off) can be controlled via an Ethernet connection to a PC connected directly (with an Ethernet cross-over cable) or through a network. The instrument software supports the TCP/IP network protocol.

#### Ethernet Interface Connection and Setup

Ethernet networking uses a bus or star topology in which all of the interfacing devices are connected to a central cable called the bus, or are connected to a hub.

The TCP/IP setup requires the following:

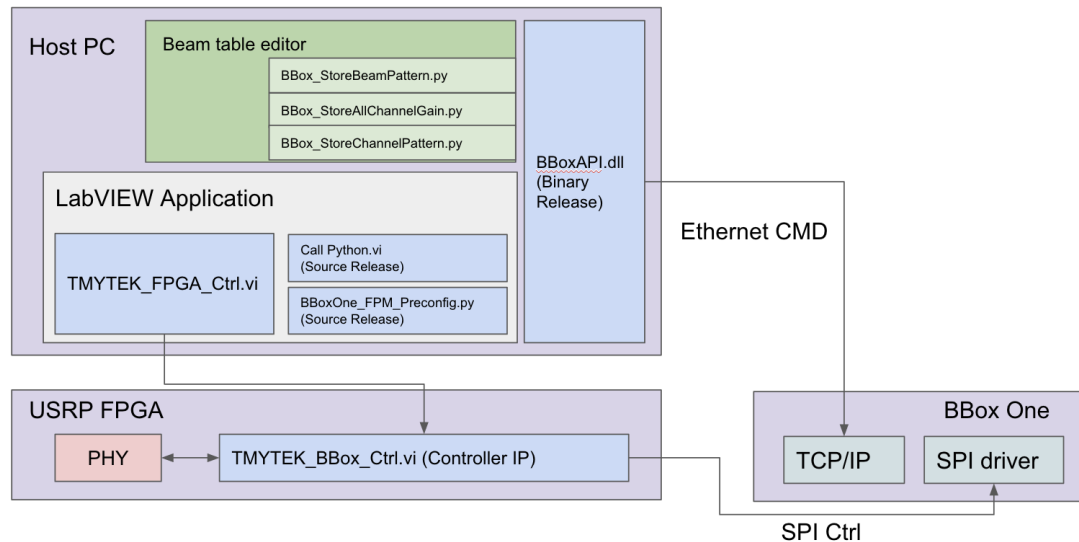
- **IP Address:** Every computer and electronic device in a TCP/IP network requires an IP address. An IP address has four numbers (each between 0 and 255) separated by periods. For example: 128.111.122.42 is a valid IP address.
- **Subnet Mask:** The subnet mask distinguishes the portion of the IP address that is the network ID from the portion that is the station ID. The subnet mask 255.255.0.0, when applied to the IP address given above, would identify the network ID as 128.111 and the station ID as 122.42. All stations in the same local area network should have the same network ID, but different station IDs.
- **Default Gateway:** A TCP/IP network can have a gateway to communicate beyond the LAN identified by the network ID. A gateway is a computer or electronic device that is connected to two different networks and can move TCP/IP data from one network to the other. A single LAN that is not connected to other LANs requires a default gateway setting of 0.0.0.0. If you have a gateway, then the default gateway would be set to the appropriate value of your gateway.
- **Ethernet Address:** An Ethernet address is a unique 48-bit value that identifies a network interface card to the rest of the network. Every network card has a unique Ethernet address (MAC address) permanently stored into its memory.

**Table 1-1.** 8-pin Ethernet RJ45 Connector Pinout Diagram

Pin	Name	Description	Wire Color
1	TX+	Transmit data ( $> +3$ volts)	White/Orange Orange
2	TX-	Transmit data ( $< -3$ volts)	Orange
3	RX+	Receive data ( $> +3$ volts)	White/Green
4	—	Not used (common mode termination)	Blue
5	—	Not used (common mode termination)	White/Blue
6	RX-	Receive data ( $< -3$ volts)	Green
7	—	Not used (common mode termination)	White/Brown
8	—	Not used (common mode termination)	Brown

### D. Application

#### Software Stack & environment



TMYTEK releases the parts in the blue blocks above.

- The developer needs to integrate **TMYTEK\_BBBox\_Ctrl** (a NI LabVIEW FPGA module) with his own FPGA source.
- LabVIEW Application makes a TCP Connection and Beam Patterns pre-configuration with BBox One via `BBoxOne_FPM_Preconfig.py` by `BBoxAPI.dll`.
- LabVIEW Application trigger **TMYTEK\_BBBox\_Ctrl** via **TMYTEK\_FPGA\_Ctrl** to achieve us level beam steering control

And the green part is for beam table editor, includes:

- `BBox_StoreBeamPattern.py`
- `BBox_StoreAllChannelGain.py`
- `BBox_StoreChannelPattern.py`

The folder contents are shown as below.

名稱	修改日期	類型	大小
__pycache__	2022/9/22 上午 10:46	檔案資料夾	
doc	2022/9/22 上午 10:44	檔案資料夾	
files	2022/9/22 上午 10:44	檔案資料夾	
FPGA	2022/9/22 上午 10:44	檔案資料夾	
FPGA Bitfiles	2022/9/22 上午 10:44	檔案資料夾	
.DS_Store	2022/9/15 上午 10:59	DS_STORE 檔案	7 KB
BBox_StoreAllChannelGain.py	2022/9/16 上午 11:01	PY 檔案	2 KB
BBox_StoreBeamPattern.py	2022/9/19 下午 03:54	PY 檔案	2 KB
BBox_StoreChannelPattern.py	2022/9/15 下午 05:50	PY 檔案	2 KB
BBoxAPI.dll	2022/9/21 下午 05:14	應用程式擴充	2,633 KB
BBoxOne_FPM_Preconfig.py	2022/9/15 下午 06:00	PY 檔案	6 KB
Common.py	2022/9/22 上午 10:45	PY 檔案	19 KB
NLog.config	2022/9/15 上午 10:59	CONFIG 檔案	2 KB
README.md	2022/9/16 上午 11:07	MD 檔案	3 KB
SPI.lvproj	2022/9/15 上午 10:59	LVPROJ 檔案	69 KB
TMYTEK_BBBox_Ctrl.vi	2022/9/15 上午 10:59	VI 檔案	435 KB

Please make sure the table file and the beam pattern list file into **files** folder

- {device\_sn}\_{operation\_frequency}GHz.csv
- {aakit\_name}.csv
- BeamTable/Beam\_Configuration\_{device\_sn}\_{operation\_frequency}GHz.json

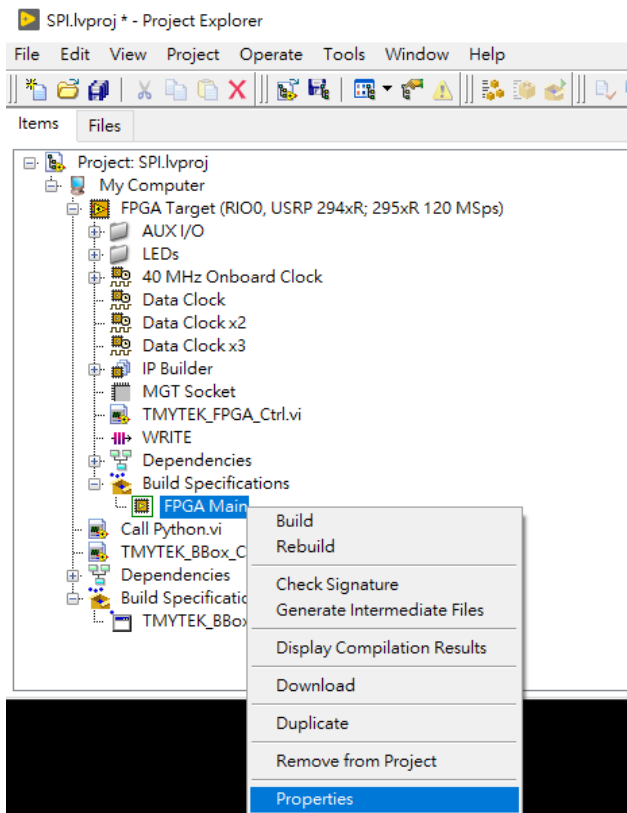
### Software Deliverables:

Module Name	Format	Source/Binary Release	Description
FPGA Bitfiles\Call Python.vi	*.vi	Source Release	This allows user applications to use the BBoxOne_FPM_Preconfig.py
BBoxOne_FPM_ Preconfig.py	*.py	Source Release	This allows user applications to use the BBoxAPI.dll to communicate with BBoxOne
FPGA\TMYTEK_ FPGA_Ctrl.vi	*.vi	Source Release	This allows user applications in LabVIEW to hand over control to NI USRP FPGA.
BBoxAPI.dll(v3.3. 16.3)	.NET Windows DLL	Binary Release	This controls BBox One via 100M Ethernet.
TMYTEK_BBox_ Ctrl	- *.vi - bit file (binary) - NI LabVIEW FPGA Module	Source Release	This allows software radio in NI USRP FPGA to send beam switch commands to BBox One w. Minimal latency.
BBox_StoreBea mPattern.py	*.py	Source Release	This edits for the beam into beam table with json format.
BBox_StoreAllCh annelGain.py	*.py	Source Release	This edits for one gain to all channels into beam table with json format.
BBox_StoreChan nelPattern.py	*.py	Source Release	This edits for specific channel into beam table with json format.

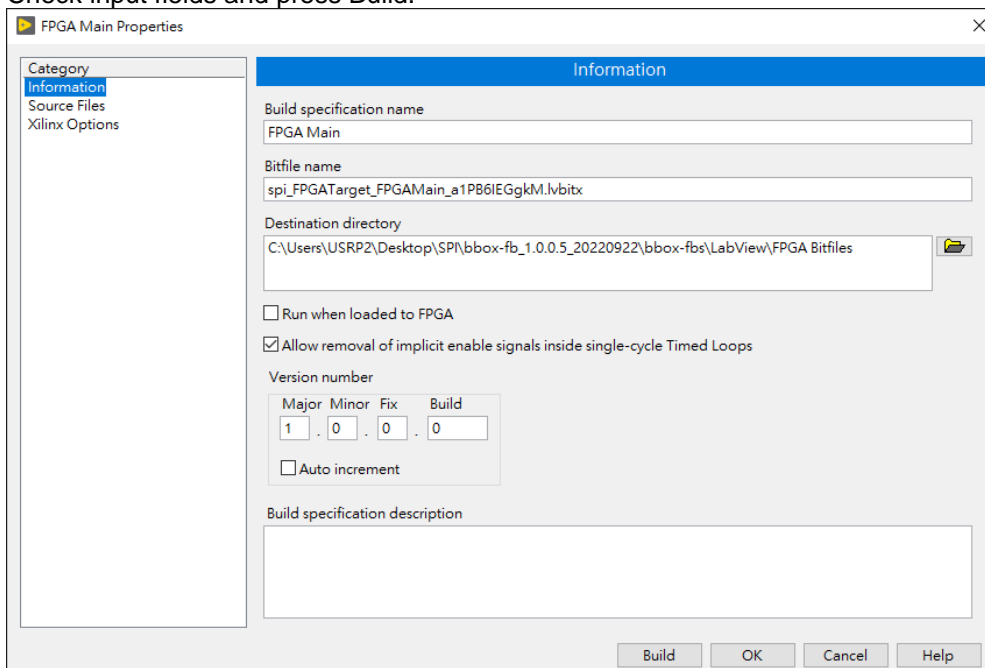
### E. Setup & running procedure

#### Setup USRP (Only once)

1. Launch TMYTEK\_FBS\_Example by double click the **SPI.lvproj** file.
2. Right click **FPGA Main** then click **Properties**.

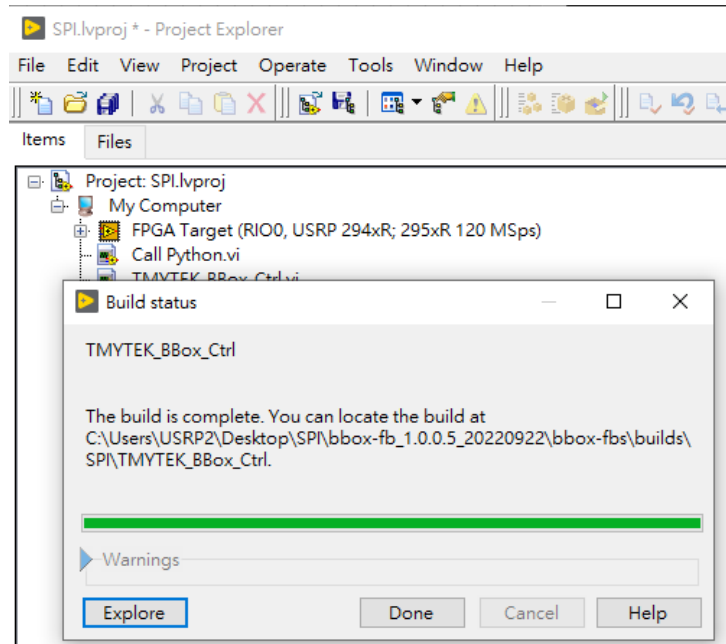


3. Check input fields and press Build.

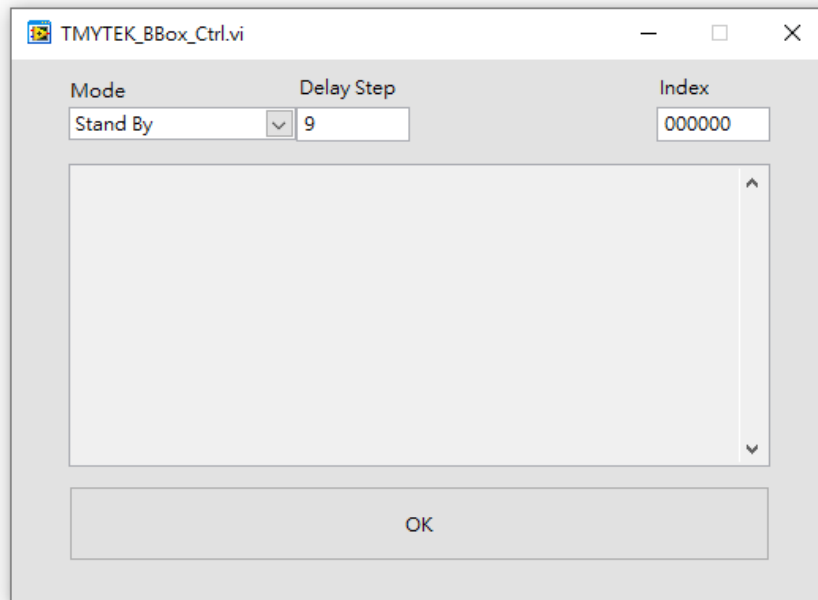
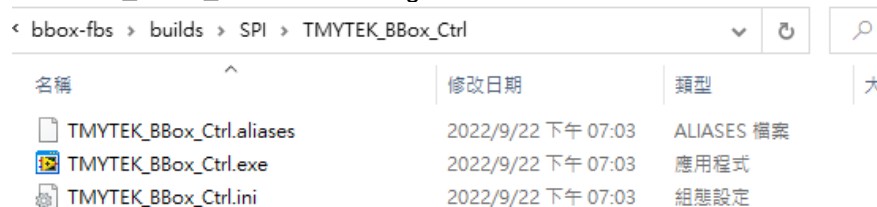


4. After built, **TMYTEK\_BBBox\_Ctrl.exe** generated at builds folder.





### 5. TMYTEK\_Bbox\_Ctrl.exe will be generated



### Predefine beams

This procedure is for user to customize their beams, there are 2 methods to customize user's beams, [NOTICE] we strongly suggest using **TMXLAB Kit** from TMYTEK(<https://www.tmytek.com/downloads>), and follow suggest beam/channel settings to apply to these 2 methods.

1. Using beam table editor scripts via Python to update beam table, please see the below table or README.md:

Module	Usage	Description
BBox_StoreBeamPattern.py	python BBox_StoreBeamPattern.py [-h] [--mode MODE] [--beamId BEAMID] [--db DB] [--theta THETA] [--phi PHI]	Store the whole <u>beam info</u> (db/theta/phi) with specific mode into assigned beamId.
BBox_StoreAllChannelGain.py	python BBox_StoreAllChannelGain.py [-h] [--mode MODE] [--beamId BEAMID] [--db DB]	Store the gain to all <u>channels</u> with specific mode into assigned beamId.
BBox_StoreChannelPattern.py	python BBox_StoreChannelPattern.py [-h] [--mode MODE] [--beamId BEAMID] [--ch CH] [--sw SW] [--db DB] [--deg DEG]	Store the <u>specific channel info</u> (sw/db/deg) with specific mode into assigned beamId.

The following is use cases (still suggest to follow suggest beam/channel settings on TMXLAB Kit):

- a. Assign TX beam (db:8, theta:25, phi:150) to beamId:1
    - i. `python BBox_StoreBeamPattern.py --mode 1 --beamId 1 --db 8 --theta 25 --phi 150`
  - b. Assign gain:-2 to all channels for RX mode as beam id: 2.(Usually we would this cmd while resetting to the same average gain)
    - i. `python BBox_StoreAllChannelGain.py --mode 2 --beamId 2 --db -2`
  - c. Assign customized TX beam with ch:9, sw:0, db:12, deg:100, and ch:10, sw:0, db:11, deg:200, and ch:11, sw:1, db:10, deg:0 as beam id: 3 (hint: write a batch file to integrate multiple settings is a good way)
    - i. `python BBox_StoreChannelPattern.py --mode 1 --beamId 3 --ch 9 --sw 0 --db 12 --deg 100`
    - ii. `python BBox_StoreChannelPattern.py --mode 1 --beamId 3 --ch 10 --sw 0 --db 11 --deg 200`
    - iii. `python BBox_StoreChannelPattern.py --mode 1 --beamId 3 --ch 11 --sw 1 --db 11 --deg 200`
2. Direct to edit the contents of Beam\_Configuration\_{device\_sn}\_{operation\_frequency}GHz.json in files\BeamTable.

There are 64 beam ids from 1 to 64 each mode (TX/RX) in this file.

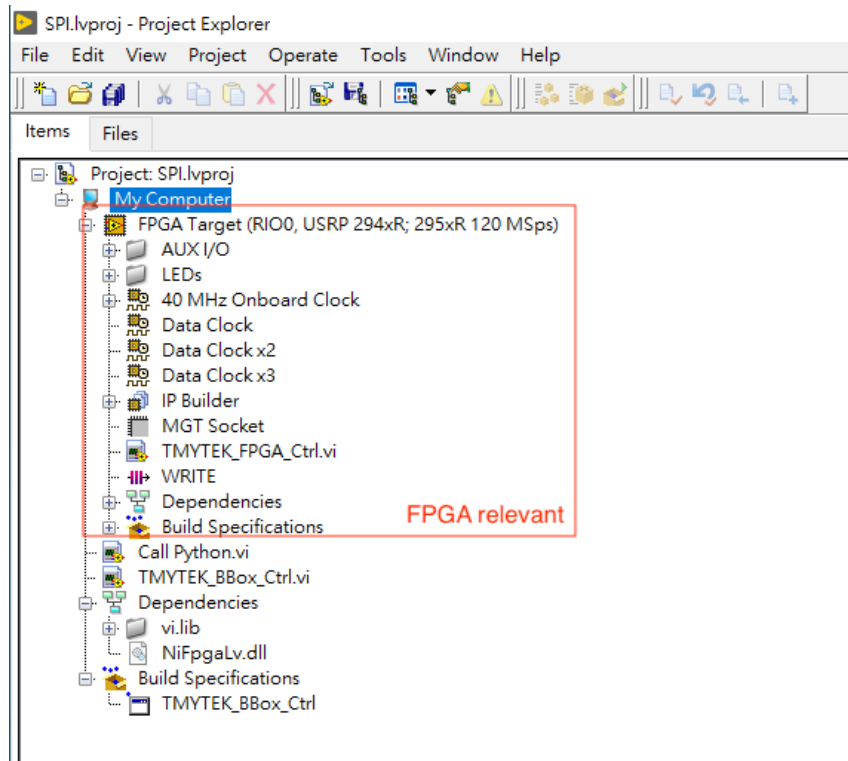
```

1  {
2      "version": "v1.1.0",
3      "SN": "D2133E012-28",
4      "TX_beam_list": [
5          {
6              "beam_type": 0,
7              "beamId": 1,
8              "mode": 1,
9              "beam_config": {
10                 "db": 9.0,
11                 "theta": 10,
12                 "phi": 10
13             },
14             "channel_config": [
15                 {
16                     "channel": 1,
17                     "sw": 0,
18                     "db": 11.0,
19                     "deg": 0
20                 },
21                 {
22                     "channel": 2,
23                     "sw": 0,
24                     "db": 11.0,
25                     "deg": 0
26                 },
27                 {
28                     "channel": 3,
29                     "sw": 0,
30                     "db": 11.0,
31                     "deg": 0
32                 },
33                 {
34                     "channel": 4,
35                     "sw": 0,
36                     "db": 11.0,
37                     "deg": 0
38                 },
39                 {
40                     "channel": 5,
41                     "sw": 0,
42                     "db": 11.0,
43                     "deg": 0
44                 },
45                 {
46                     "channel": 6,
47                     "sw": 0,
48                     "db": 11.0,
49                     "deg": 0
50                 },
51                 {
52                     "channel": 7,
53                     "sw": 0,
54                     "db": 11.0,
55                     "deg": 0
56                 },
57                 {
58                     "channel": 8,
59                     "sw": 0,
60                     "db": 11.0,
61                     "deg": 0
62                 },
63                 {
64                     "channel": 9,
65                     "sw": 0,
66                     "db": 11.0,
67                     "deg": 0
68                 },
69                 {
70                     "channel": 10,
71                     "sw": 0,
72                     "db": 11.0,
73                     "deg": 0
74                 },
75                 {
76                     "channel": 11,
77                     "sw": 0,
78                     "db": 11.0,
79                     "deg": 0
80                 },
81                 {
82                     "channel": 12,
83                     "sw": 0,
84                     "db": 11.0,
85                     "deg": 0
86                 },
87                 {
88                     "channel": 13,
89                     "sw": 0,
90                     "db": 11.0,
91                     "deg": 0
92                 },
93                 {
94                     "channel": 14,
95                     "sw": 0,
96                     "db": 11.0,
97                     "deg": 0
98                 },
99                 {
100                    "channel": 15,
101                    "sw": 0,
102                    "db": 11.0,
103                    "deg": 0
104                },
105                {
106                    "channel": 16,
107                    "sw": 0,
108                    "db": 11.0,
109                    "deg": 0
110                }
111            ],
112            "dev_type": 9
113        }
114    ]
115 }
    
```

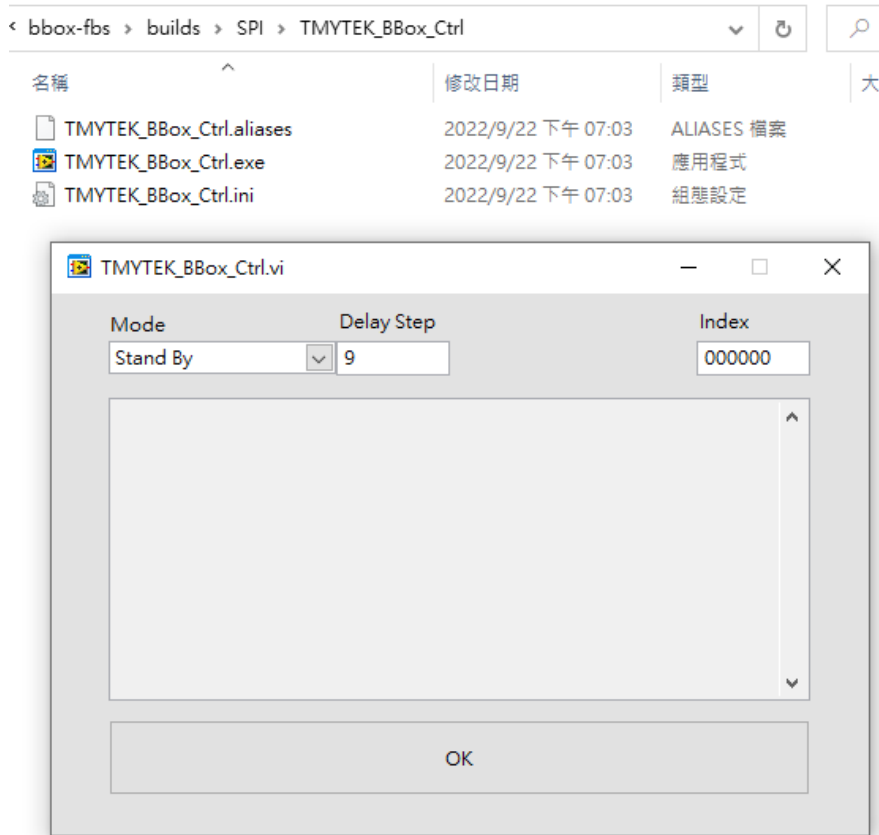
- Edit “SN” field to your device sn
- For each beamId in each mode(TX/RX):
  - Edit “beam\_type” field to select your beam type (0 means to using beam\_config and 1 means using channel\_config)
  - beam\_config
    - Edit “db” field to your target beam gain
    - Edit “theta” field to your target beam theta angle
    - Edit “phi” field to your target beam phi angle
  - channel\_config
    - Include channel list: 1~16, each channel (element in list) contains:
      - “sw” for decide channel is on:0 or off:1
      - “db” field to your target channel gain
      - “deg” field to your target channel phase degree

### Execute specific beam id

1. Launch TMYTEK\_FBS\_Example by double click the **SPI.lvproj** file



2. Launch **TMYTEK\_BBox\_Ctrl.exe**, application GUI as below



3. There are 4 selections of **Mode**.
  - a. Stand By
  - b. Tx Mode
  - c. Rx Mode

- d. Sleep Mode
- 4. **Delay Step** implies different SPI clock speed.
  - CLK speed =  $120/((\text{delay\_step}+1)*2)$  MHz
- 5. **Index** implies different Beam patterns defined by the user in the pre-configuration stage.
  - Index : 0 - 63 in binary format, and mapping to beamID from 1 to 64
- 6. Press OK button could trigger the FPGA sending control beam pattern for specific beamID.

