User Manual Document Fast Beam Switch Controller IP on NI FPGA for BBox™

V0.0.1

CT Fang (ct_fang@tmytek.com), Tobias Huang (tobias_huang@tmytek.com)

Version

Version	Author	Date	Description
V0.0.1	Tobias Huang	2021/11/25	Initial version.

System Overview

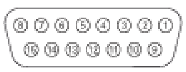
This is a system integration solution for reducing beam switch time down to hundreds of nanoseconds (ns). The demonstration contains a single BBox One/Lite and is executed by NI LabView.

- Hardware Requirements :
- NI USRP29xx
- TMYTEK BBox Series

Getting Start

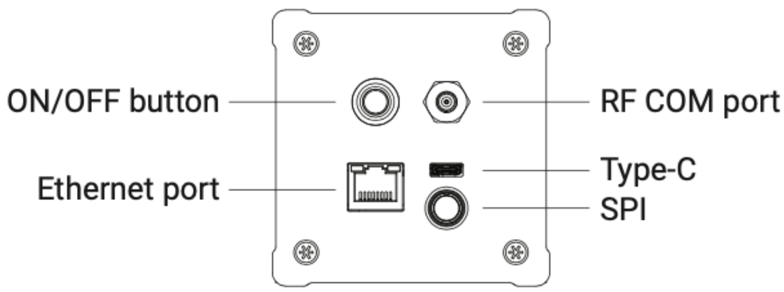
A. Hardware Requirements and Configuration

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_DPI
	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	Don't care
	9	GPIO 7	AUX I/O 7	Don't care
	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

Operation System	Win 10 or later
LabView	2019 or later
Python	3.7.0 or later
.Net Framework	4.5.2 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.

Ethernet Interface Connection and Setup

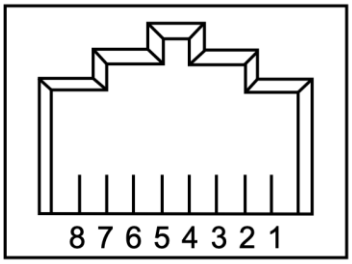
The BBox series device uses Ethernet to communicate remotely with a controller. Most instrument functions (except power on/off) can be controlled via an Ethernet connection to a PC connected directly (with an Ethernet cross-over cable) or through a network. The instrument software supports the TCP/IP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

Ethernet 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.

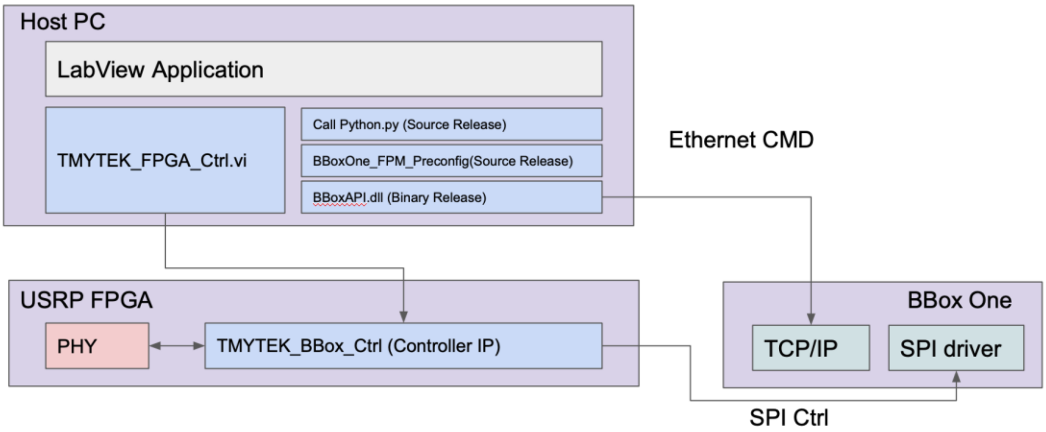
Table 1-1. 8-pin Ethernet RJ45 Connector Pinout Diagram



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 Stock



TMYTEK releases the parts in the blue blocks above.

- The developer needs to integrate **TMYTEK_BBox_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 by python script, BBoxOne_FPM_Preconfig.py , and BBoxAPI.dll
- LabView Application trigger **TMYTEK_BBox_Ctrl** via **TMYTEK_FPGA_Ctrl** to achieve us level beam steering control

1. The folder contents are shown as below.

名稱	修改日期	類型	大小
files	2021/11/30 下午 04:42	檔案資料夾	
FPGA	2021/11/30 下午 02:31	檔案資料夾	
FPGA Bitfiles	2021/11/30 下午 05:18	檔案資料夾	
logs	2021/11/30 下午 04:42	檔案資料夾	
TMYTEK_BBox_Ctrl	2021/11/30 下午 04:40	檔案資料夾	
BBoxAPI.dll	2021/11/4 下午 07:18	應用程式擴充	2,333 KB
NLog.config	2021/10/7 下午 08:00	LabVIEW NXG	2 KB
BBoxOne_FPM_preconfig.py	2021/11/5 上午 10:58	Python File	6 KB
SPI.aliases	2021/11/30 下午 05:22	ALIASES 檔案	1 KB
SPI.lvlps	2021/11/30 下午 05:22	LVLPS 檔案	1 KB
SPI.lvproj	2021/11/30 下午 05:22	LabVIEW Project	69 KB
TMYTEK_BBox_Ctrl.vi	2021/11/30 下午 05:22	LabVIEW Instrum...	432 KB

2. Please put the table file and the beam pattern list file into "files" folder
   a. {device_sn}_{operation_frequency}GHz.csv
   b. {aakit_name}.csv
   c. BeamTable/Beam_Configuration_{device_sn}_{operation_frequency}GHz.json

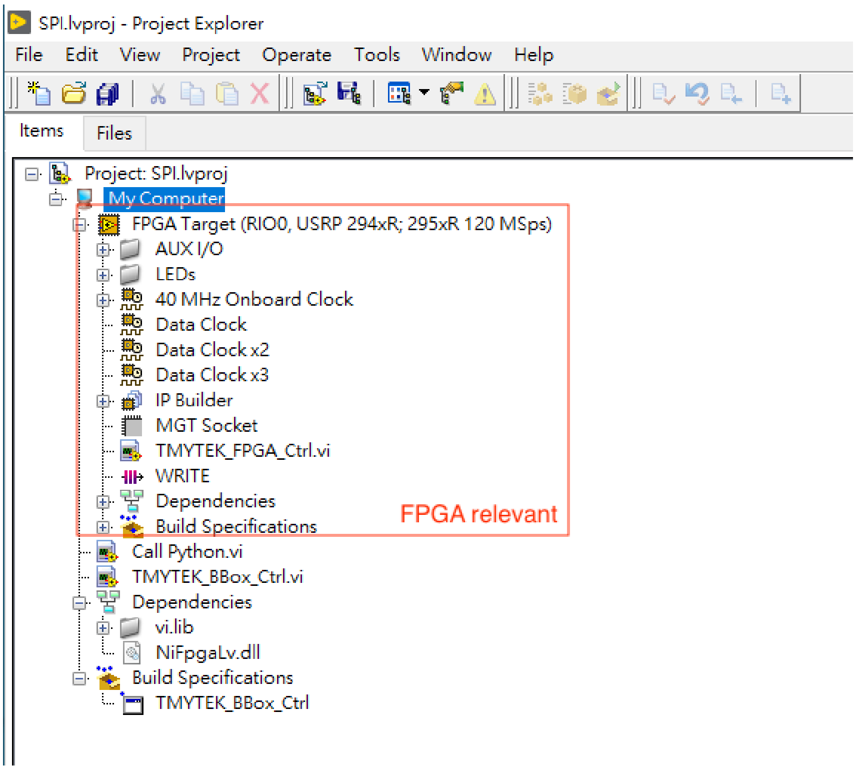
3. Edit the contents of Beam_Configuration_{device_sn}_{operation_frequency}GHz.json

There are 64 beam Id from 1 to 64 in this file.

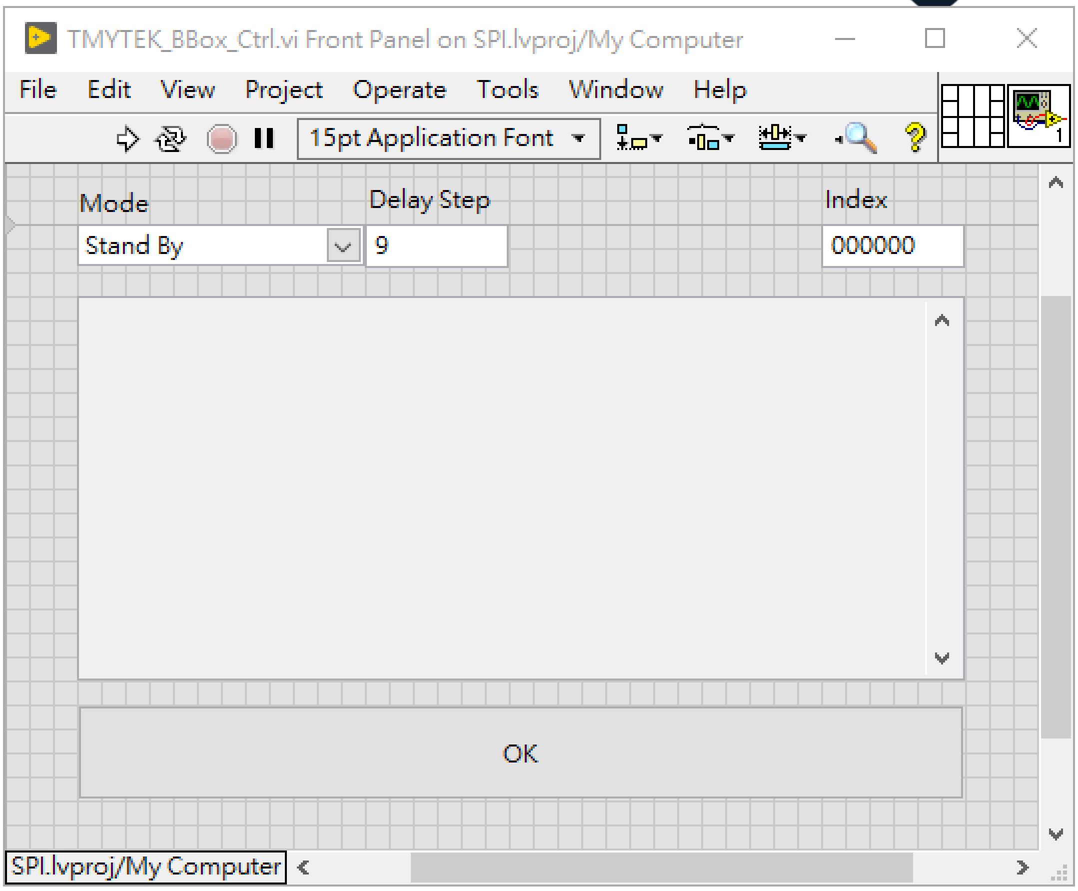
```
{
  "version": "v1.0.0",
  "SN": "D2144E017-28",
  "TX_beam_list": [
    {
      "beamId": 1,
      "mode": 1,
      "db": 11.0,
      "theta": 0,
      "phi": 0,
      "aakit_name": "TMYTEK_28ONE_4x4_C2104L020-28",
      "channel_switch": [
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0,
        0
      ],
      "dev_type": 9
    },
```

- Edit **"SN"** field to your device sn
- For each beamId :
  - edit **"mode"** field to your target mode, Tx : 1 and Rx : 2
  - 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

4. Launch TMYTEK_FBS_Example by double click the SPI.lvproj file



5. double click **TMYTEK_BBox_Ctrl.exe, application GUI as below**



6. There are 4 selections of **Mode**.
    a. Stand By
    b. Tx Mode
    c. Rx Mode
    d. Sleep Mode
7. **Delay Step** implies different SPI clock speed.
    a. CLK speed = 120/((delay_step+1)*2) MHz
8. **Index** implies different Beam patterns defined by the user in the pre-configuration stage.
    a. Index : 0 - 63 in binary format, and mapping to beamID from 1 to 64

9. press OK button could trigger the FPGA sending control beam pattern for specific beamID.

Software Deliverables:

Module Name	Format	Source/Binary Release	Description
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
TMYTEK_FPGA_Ctrl	*.vi	Source Release	This allows user applications in LabVIEW to hand over control to NI USRP FPGA.
BBoxAPI.dl(v3.3.6)l	.Net Windows DLL	Binary Release	This controls BBox One via 100M Ethernet.
TMYTEK_BBox_Ctrl	- 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.