Debug Scripts User Manual

Revision history

Date	Version	Decription
27/06/2020	1	File created.

TTIM/BEC port assignment

Each BEC has 48 ports. The port number is assigned from left-up to right-down, shown below.

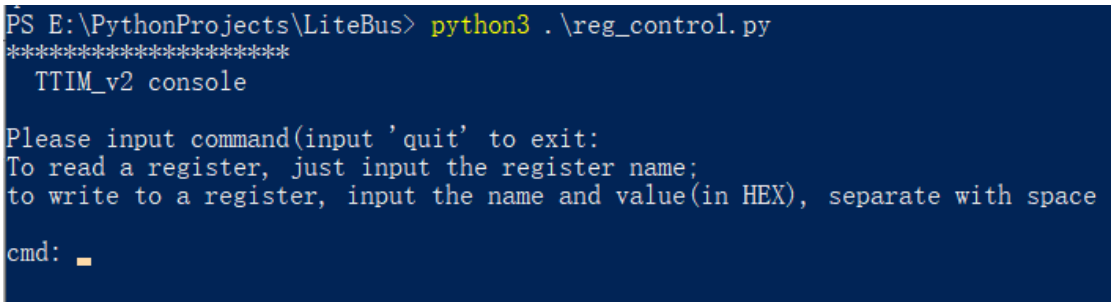
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41	43	45	47
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48

The essence of the scripts is manipulating the registers inside the firmware. For those registers controlling channel or port, such as test_mode, each bit (48 bits in total) controls 1 channel or port. The bit order corresponds to the channel/port number.

Take test_mode for example. If you want to set port 1 to test mode, you need to write 0x1 to test_mode; if you want to set port 48 to test mode, 0x800000000000 should be used.

reg_control.py

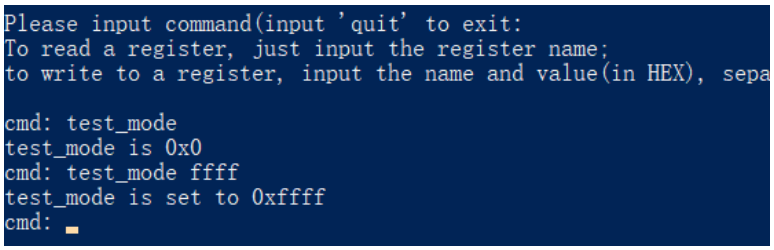
This is a general controller of all registers. The interface is shown below:



To read from or write to a register, find the register name first. The register name can be found in TTIM_v2_registers.dat and chapter 7 of TTIM_firmware3.docx.

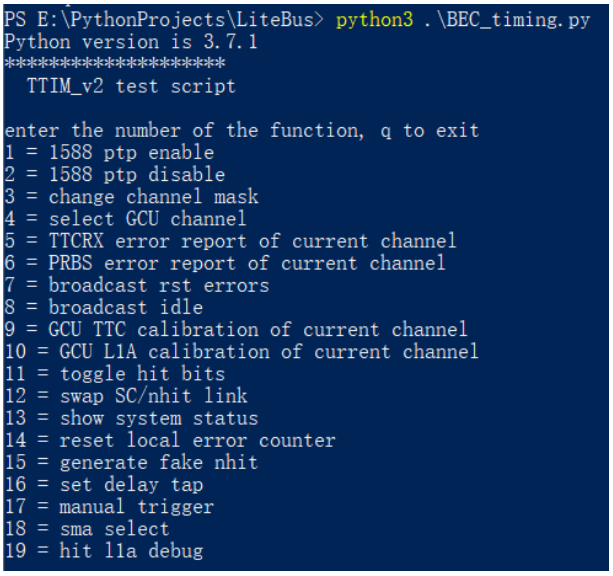
*Note: the register name in those two files may be different, use the one in TTIM_v2_registers.dat. The TTIM_firmware3.docx is used to describe the registers and will correct the register names in the future.

The following picture shows an example of reading and writing. Note that when writing to a register, the script treats the value as hexadecimal number. So, don't write 0xffff!



BEC_timing.py

This script contains some frequently used functions. The interface is shown below:



To use the function, just input the function number and press Enter.

prbs_eye_scan.py

This script is used to scan the sampling point of the RX channels. The script will first set the test_mode bit of selected channel.

```
PS E:\PythonProjects\LiteBus> python3 .\prbs_eye_scan.py
----------Eye scan - Demonstration script----------------
channel to run eye scan(1 - 48): 1
eye scan start...
tap_cnt    error
0    0
1    0
2    0
3    0
```

After the channel selected, the script will add the delay tap from 0 to 62 and monitor the error counter. When the scan is done, the script will set the delay tap to a proper value.

```
61    0
62    0
RX1 tap_cnt set to 31
```

```
62    24996860
RX2 tap_cnt set to 0
eye scan stop
```

*Note: there's a known bug inside the script, so the delay tap value is not always correct. When there are glitches inside the scan window, the script is not able to recognize the best eye and may set the tap value to a strange value. Always check the tap_cnt value when use this script. If the value is not correct, you need to set the tap_cnt manually.

To set the delay tap manually, use the BEC_timing.py. Choose function number 16, then input the channel you want to set and the tap value. Choose the RX pair at last.

```
9 = GCU TIC calibration of current channel
10 = GCU L1A calibration of current channel
11 = toggle hit bits
12 = swap SC/nhit link
13 = show system status
14 = reset local error counter
15 = generate fake nhit
16 = set delay tap
17 = manual trigger
18 = sma select
19 = hit l1a debug
16
input GCU channel(1 - 48, 0 to check current channel): 1
input tap value: 20
1 -> RX1, 2 -> RX22
```

Loop Test Instruction

When test the BEC in standalone setup, loop test is always necessary. This chapter shows instruction to do a loop test.

1. set test mode:

   In test mode, the TX source is PRBS-7 pattern and RX use a PRBS checker to check the incoming data stream. In loop test, we can set all channels to test mode. Use reg_control.py, set test_mode register to 0xffffffffffff.

```
PS E:\PythonProjects\LiteBus> python3 .\reg_control.py
**********************
  TTIM_v2 console

Please input command(input 'quit' to exit:
To read a register, just input the register name;
to write to a register, input the name and value(in HEX), separate w

cmd: test_mode ffffffffffff
test_mode is set to 0xffffffffffff
cmd:
```

2. set TX pair 1 source:

   In loop test mode, the TX pair 1 should send also PRBS-7 pattern instead of clock. Use reg_control.py, set tx1_sel register to 0xffffffffffff.

3. set sampling points:

   use prbs_eye_scan.py to scan the eye of selected channel. The RX pair 2 should always has an eye if hardwares are working properly but RX pair 1 may not. If there's always error during eye scan, you need to invert the corresponding channel TX. For example,

```
**********************
  TTIM_v2 console

Please input command(input 'quit' to exit:
To read a register, just input the register name;
to write to a register, input the name and value(in HE

cmd: test_mode ffffffffffff
test_mode is set to 0xffffffffffff
cmd: invert_tx1 800000000000
invert_tx1 is set to 0x800000000000
cmd:
```

   channel 1 and channel 48 are connected together as a loop. When scan the eye of channel 1, the RX1 has no eye, then you need to invert the TX1 of the channel 48. In this case, use reg_control.py, set invert_tx1 to 0x800000000000. After setting the invertion, re-run the eye scan.

4. monitor the error counters:

   use the BEC_timing.py, choose function number 6 to show current channel's error counter. To check error counter of other channels, use function numer 4 to change channel number.