# TTIM firmware specification

Version 2

Revision History

The following table shows the revision history for this document.

|  |  |  |
| --- | --- | --- |
| Data | Version | Revision |
| 15/10/2019 | 1 | Initial firmware release. |
| 21/11/2019 | 2 | SYNC link module function changed according to Filippo Marini’s firmware. Some control registers’ address added and changed. |
| 2/12/2019 | 2.1 | Delete timestamp generating inside clk time module. Add timestamp fetching form mini-WR. Timestamp format is 40 bits UTC + 28 bits 8ns counter. |
| 7/12/2019 | 2.1 | Data format from/to mini-WR changed due to firmware change of the mini-WR. |
| 17/6/2020 | 3 | transfer to TTIM\_v3. Add trigger window for local nhit trigger. Registers update. |

## Chapter 1 Firmware structure

This chapter introduces the firmware structure of the TTIM.

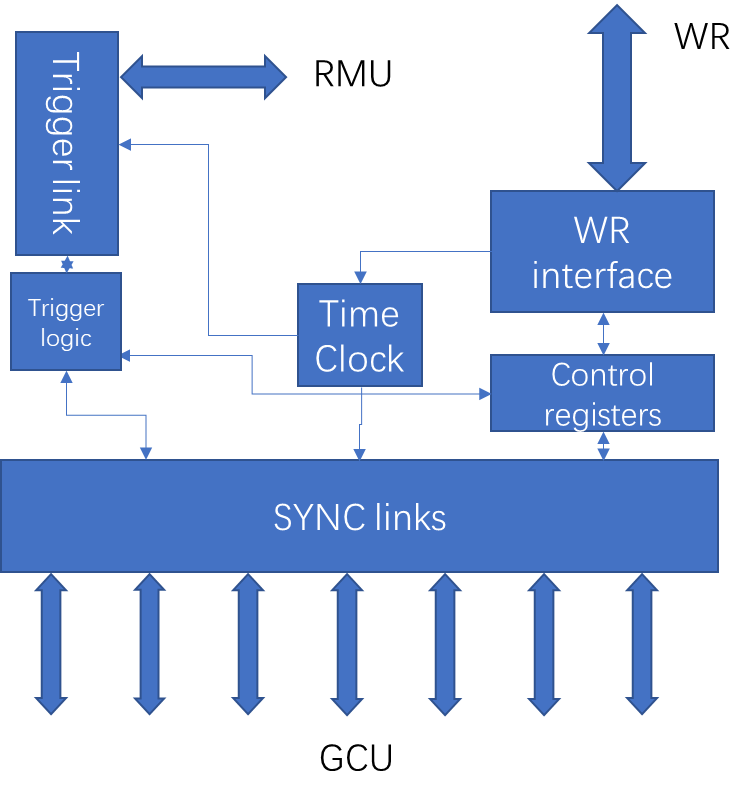
The firmware is divided into several module, shown in figure 1.1.

Figure 1.1 Firmware structure of the TTIM

* Trigger link: handle communication with the RMU. Line rate of this optical link is 1.25Gps. Detailed introduction in chapter 2.
* Trigger logic: receive trigger decision from the RMU and deliver to SYNC links. This module can also generate trigger itself. Detailed introduction in chapter 3.
* SYNC links: handle communication with the GCUs. Fanout trigger decision and receive hit information. Detailed introduction in chapter 4.
* Time Clock: generates clock and timestamps, align system clock to PPS from White-Rabbit (WR). Detailed introduction in chapter 5.
* WR interface: interface with the onboard mini-WR mezzanine. The slow control interface for TTIM is implemented inside this module. Detailed introduction in chapter 6.
* Control registers: configuration registers of the board. Detailed introduction in chapter 7.

## Chapter 2 Trigger link

Trigger link is implemented using the GTX, acting as bridge between the RMU and the BEC. The interface between this module and FPGA fabric logic is 16 bits data with 62.5MHz clock.

48 GCU hit information is summed up by the trigger logic module and generated as an 8 bits data. Constant x”BC” on the lower 8 bits with nhit on the higher 8 bits builds a transmitting frame. The x”BC” is encoded as comma every 65536 clock cycles (about 1 ms) for comma alignment. The transmitting frame format is shown in table 2.1.

Table 2.1 txdata frame of the Trigger link

|  |  |
| --- | --- |
| b15 b8 | b7 b0 |
| nhit | x”BC” |

The received data of the trigger link is the trigger information. The MSB of rxdata is trigger decision (‘1’ indicates a trigger accept), then follows 7 bits trigger type. The lowest 8 bits are x”BC”. The rxdata path uses rx\_slide to align the incoming data. After rxdata aligned, the rx\_aligned signal will be asserted high. Table 2.2 shows the rxdata frame.

Table 2.2 rxdata frame of the Trigger link

|  |  |  |
| --- | --- | --- |
| b15 | b14 b8 | b7 b0 |
| T | Trigger type | x”BC” |

The definition of the trigger type is,

* b[14:13]: reserved, always 0;
* b[12:10]: to be decided, candidates are random trigger, external trigger, manual trigger;
* b[9]: periodic trigger;
* b[8]: nhit trigger.

## Chapter 3 Trigger logic

The trigger logic module sums up all 48 channels GCU hit information and send to the trigger link module. After a trigger decision received, this module will transfer the trigger information to SYNC links module. This module can also generate all kinds of trigger signals itself.

This module will add up the nhit number of all 48 GCU channels in a configurable time window. Using the nhit sum result and threshold configured, this module can generate a physics trigger.

Other trigger sources are provided. Periodic trigger can be generated by configure the period counter; SMA connector can introduce an external trigger source; manual trigger can be generated by slow control.

All the trigger sources are or-ed together to generate trigger signal and each has a mask to be disabled (‘0’ to disable). The mask register, en\_trig\_src[4:0], has relationship with the source as follow:

* en\_trig\_src[0]: local physics trigger;
* en\_trig\_src[1]: local periodic trigger;
* en\_trig\_src[2]: local external trigger;
* en\_trig\_src[3]: local manual trigger;
* en\_trig\_src[4]: trigger from the RMU.

A fake nhit counter counts from 0 to x”FF”. When gen\_fake\_nhit is set to ‘1’, this counter will take place of the real nhit and sent to the trigger link module.

## Chapter 4 SYNC links

SYNC links module handles the communication with the GCUs.

Connection with the GCU uses 4 differential pairs. Definition of each pair is as below:

* Orange: 62.5MHz clock to GCUs;
* Green: slow control link from GCU to TTIM, 125Mbps line rate, data is BMC coded;
* Blue: slow control link from TTIM to GCU, 125Mbps line rate, data is BMC coded;
* Brown: nhit link from GCU to TTIM, 125Mbps line rate, data is scrambled.

The slow control links use TTC protocol. There are 2 kinds of commands: broadcast and long frame commands. Broadcast commands will be accepted by all GCUs while long frame commands are addressed commands that can be accepted by GCU with corresponding GCU ID. Table 4.1 and 4.2 shows the frame format of the commands.

Table 4.1 broadcast commands format

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SOF | Type | Data | Hamming | EOF |
| ‘0’ | ‘0’ | 8 bits | 5 bits | ‘1’ |

Table 4.2 long frame commands format

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SOF | Type | Address | E | Sub-addr | Data | Hamming | EOF |
| ‘0’ | ‘1’ | 14 bits | ‘1’ | 8 bits | 8 bits | 7 bits | ‘1’ |

Currently supported broadcast commands are shown in table 4.3. Only idle and reset error implemented. Other commands needed can be added in the future.

Table 4.3 supported broadcast commands

|  |  |
| --- | --- |
| Command | Data (binary) |
| idle | 11111100 |
| reset time | 11111101 |
| reset event | 11111110 |
| supernova | 11111000 |
| reset error | 11101100 |
| auto trigger | 11101000 |
| time request | 11110000 |
| enable acquire | 11100100 |

Currently implemented long frame commands are shown in table 4.4.

Table 4.4 supported long frame commands

|  |  |  |
| --- | --- | --- |
| Command | Sub-addr(hex) | Description |
| sync\_req | 09 – 0E | Synchronous request to the GCU |
| delay\_req | 19 | Delay request from the GCU |
| delay\_rsp | 11 – 16 | Delay response to the GCU |
| tap\_incr\_sc | 1A | increase the delay tap of slow control link by 1 to the GCU |
| tap\_incr\_hit | 1D | increase the delay tap of hit link by 1 to the GCU |
| tap\_rst\_sc | 1C | reset the delay tap of slow control link to the GCU |
| tap\_rst\_hit | 1F | reset the delay tap of hit link to the GCU |
| trig\_time | 01 - 06 | send trigger timestamp to the GCU |
| hit\_calib | 20 | control the GCU to send PRBS on hit link |

For the sample position finding, the TTIM controls the GCU to reset or increase the output delay tap and monitor local error counters to determine current position is inside the eye or not. 4 delay modules are used in the GCU to cover the whole eye. The calibration module firstly scans the eye, then put the sample position in the middle of the eye. For slow control link, the TTIM use comm\_error\_counter to check sample position is good or not, while PRBS error counter is used for the hit link.

Currently the calibration procedure is done manually. This could work for test purpose to calibrate a few channels but not for final system. An automatic method should be used to do this.

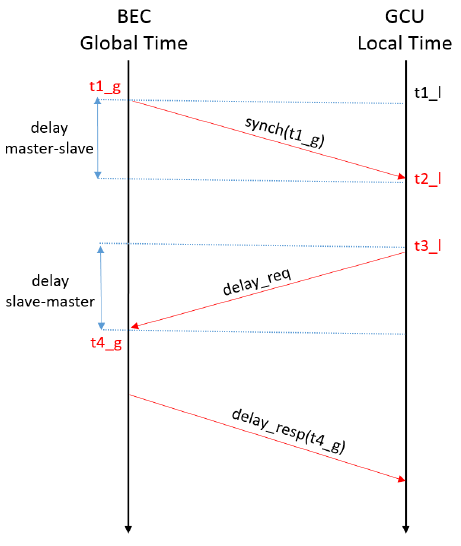
The synchronization procedure is handled by the bec\_1588\_ptp FSM, shown in figure 4.1. The synchronization is initiated by the TTIM (BEC) sending sync\_req command to the GCU; the GCU gets the t1\_g at local time t2\_l from this command and sends delay\_req to the BEC at t3\_l; the BEC receives delay\_req at t4\_g and send the timestamp back with delay\_rsp command. With all this information, the GCU can calculate the offset between local time and global time with equation (1), then correct local time counter.

Figure 4.1 synchronization procedure

(1)

Equation (1) assumes the delay master-slave is equal to delay slave-master. According to cable delay tests by Yanke Cai, this will cause inconformity between the GCUs. More accurate measurement should be considered to calibrate the cable asymmetry.

The sync link module support test mode. To change a channel to test mode, write the corresponding bit of test\_mode register to ‘1’. In test mode, the slow control link from TTIM to GCU will send PRBS-7 pattern and the 2 up link data will go to PRBS checker. The clock link can also send PRBS pattern in case loop test is need. The control registers, tx1\_sel and invert\_tx1 can change the source and polarity of the clock links. The slow control link output can be disabled using register tx2\_en.

In test mode, the link sampling position need to be find manually using tap\_cnt and load\_tap registers. First, use the channel\_sel to select the channel need to be operated, then write reasonable value to the tap\_cnt, write load\_tap ‘1’ to load the value into the delay module at last. The delay module uses 2 IDELAYE2 primitives, covers about 4.8 ns sample window.

## Chapter 5 Time Clock

This module generates system clock and other clock needed by other modules.

In TTIM\_v3, two clock sources (local oscillator and clock from mini-WR) go to the FPGA GTX reference clock directly.

The clock from mini-WR is used as the original system clock and goes into an MMCM. The MMCM generates 2 62.5MHz system clocks (180° phase difference), 125MHz clock for SYNC links and mini-WR interface, 200MHz for delay module.

The local clock also uses an MMCM to generate local 62.5MHz and 125MHz to use as none-lost clock source.

The original clock is 125MHz. When divided, the system clock will have 180° uncertainty. The PPS signal is used to choose system clock. The firmware uses one of the two 62.5MHz clock and the 125MHz clock to latch in the PPS signal separately as pps\_r1 and pps\_b, then uses the same 62.5MHz clock to find the PPS rising edge. At the rising edge of PPS, use 62.5MHz clock to sample the pps\_b as pps\_r. The pps\_r will be ‘1’ or ‘0’ once the phase is fixed. If pps\_r is ‘1’, choose the sampling 62.5MHz clock, otherwise choose the other one.

When using the above method, some delay values may have two peaks due to the routing path of the two 62.5MHz clock. The reset method is considered to solve this issue and will be implemented in the future.

## Chapter 6 WR interface

The WR interface instantiates the parallel interface between the mini-WR and the firmware to get packets from Ethernet. This module also buffers in the PPS signal.

The timing diagram of the parallel interface is introduced in mini-WR user manual. The packets from mini-WR is payload data after UDP unpacking (Attention: currently mini-WR only listen and send to port 2000!).

For the receiver interface, “01” on the control lines (rx\_ctrl[1:0]) indicates start of a frame and “10”indicates end of frame. “11” on rx\_ctrl means this frame has ended incorrectly. 8 bits data on the rx\_data lines is latched in 1 clock cycle.

For the transmitter interface, ‘1’ on tx\_cts means the interface is free to use. By asserting the tx\_vld signal, data to send is driven on the tx\_data lines. The frame will be packed by mini-WR when de-asserting the tx\_vld. A tx frame should not exceed 1400 bytes.

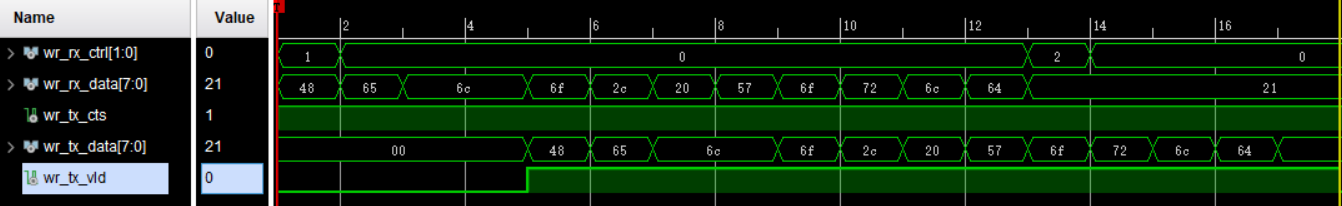
Figure 6.1 shows an example by receiving and sending “Hello world!”.

Figure 6.1 Example timing sequence of WR interface

A slow control protocol, LiteBus, is developed to control, configure and monitor the board. LiteBus is an IPbus like protocol, uses address to locate certain registers to perform reading and writing.

The LiteBus frame format is shown in table 6.1. The request and respond frame has same format, each contains 4 bits header, 3 bits transaction ID for reliability, 1 bit read/write indicator (‘1’ for write), 8 bits address (support 256 registers max) and 48 bits payload data.

Table 6.1 LiteBus frame format

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Header | Trans ID | R/W | Address | Payload data |
| “0100” | 0 ~ 7 | 1 to write | 8 bits | 48 bits |

Each request frame must have corresponding respond frame. The respond frame always set R/W bit ‘0’, and return the current value of the register in payload data.

Registers implemented is introduced in chapter 7.

Modification: the new mini-WR firmware changed the data transfer format. For the receiver interface, there will be 2 bytes header, the first byte is x”AB”, the second byte is x”CX”, X indicates the data packet source (0 from port 0, 1 from port 1). Similarly, when send data packets to the mini-WR, 2 bytes header should be added. Currently only port 0 is used, so the header should be x”AB00”.

Currently the UTC time is coded in PPS line. After a pulse on the pps, the UTC time code starts transferring. A local 28 bits counter is running and clears every pps. So, the timestamp is 40 bits UTC second counter plus 28 bits 8ns counter.

To generate the 48 bits timestamp for SYNC link, the lower 20 bits of UTC counter times 125M, then add the 8ns counter as equation below.

timer\_48b = timer\_utc\_u \* 125000000 + timer\_8ns\_u

## Chapter 7 Control registers

The control registers are used to configure, control and monitor the board.

The registers are slaves of the LiteBus. From PC software, the value of the register be read and write. To save address space, some internal signals share one register.

Table x lists the function of the registers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Register | Name | address(hex) | Write | Read |
| 0 | test\_mode | 41 | change corresponding channel to test mode. | show current value. |
| 1 | channel mask | 42 | masked channels will not be taken into count when calculating nhit. 1588PTP will also ignore them. | show current value. |
| 2 | tx2\_en | 43 | ‘1’ to disable the downlink to GCU | show current value. |
| 3 | tx1\_sel | 44 | ‘1’ to change the clock link to send prbs-7. Used in test mode. | show current value. |
| 4 | invert\_tx1 | 45 | ‘1’ to invert the clock link. Used in test mode. | show current value. |
| 5 | tap\_cnt | 46 | tap value to be loaded in to IDELAY. Used in test mode. | show current value. |
| 6 | channel\_sel | 47 | select channel to operate. | show current value. |
| 7 | load\_tap | 48 | load tap value into IDELAY. Bit0 for RX1, bit1 for RX2. | show current value. |
| 8 | inject\_reset | 49 | Bit2 ptp\_enable; Bit1 inject error to prbs generator; Bit0 reset error counters. | show current value. |
| 9 | pair\_swap | 4A | swap function (nhit/SC) of the 2 RX pairs | show current value. |
| 10 | en\_trig\_src | 4B | enable trigger sources. | show current value. |
| 11 | system\_status | 4C | Bit0 chb\_req; Bit1 ttc\_idle; Bit2 ttc\_rst\_error | show system status |
| 12 | channel\_rdy | 4D | Bit1 tap\_incr; Bit0 tap\_rst; | show ready GCU channels |
| 13 | error\_time1 | 4E | Bit0 l1a\_go\_prbs | show RX1 first error time. Used in test mode. |
| 14 | error\_time2 | 4F | Bit0 calib\_enable; Bit1 l1a\_calib\_enable | show RX2 first error time. Used in test mode. |
| 15 | error\_cnt1 | 50 | Bit0 manual trigger | show RX1 error counter. Used in test mode. |
| 16 | error\_cnt2 | 51 | Null | show RX2 error counter. Used in test mode. |
| 17 | hit\_toggle | 52 | swap hit bits | show current value |
| 18 | sbit\_error\_cnt | 53 | Bit0 generate fake nhit | show 1bit error counter of current channel |
| 19 | dbit\_error\_cnt | 54 | Null | show 2bit error counter of current channel |
| 20 | comm\_error\_cnt | 55 | Null | show common error counter of current channel |
| 21 | eye | 56 | Null | show eye result of current channel |
| 22 | l1a\_eye | 57 | Null | show l1a eye result of current channel |
| 23 | tap\_error\_cnt | 58 | Null | show tap error counter when scan eye |
| 24 | threshold | 59 | set trigger threshold | show current value |
| 25 | trig\_period | 5A | set trigger period | show current value |
| 26 | temp | 5B | Null | bit[26:18]: temperature register 1;  bit[17:9]: temperature register 2;  bit[8:0]: temperature register 3; |
| 27 | voltage | 5C | Null | bit[35:24]: sense resistor voltage;  bit[23:12]: BEC voltage;  bit[11:0]: TTIM voltage; |
| 28 | hit\_debug | 5D | Null | nhit value in the past 6 clock cycles |
| 29 | l1a\_debug | 5E | Null | trigger accept in the past 48 clock cycles |
| 30 | trig\_window | 5F | set trigger window | show current value |