ePIC Data Acquisition Working Group

Description and Technical Information for Global Timing Unit (GTU)

Initial release: Mar. 11, 2025

Updated on May 23, 2025

# Table of Contents

# Section Title Page

|  |  |  |
| --- | --- | --- |
| 1 | Introduction | 3 |
| 2 | Purpose of GTU Module | 4 |
| 3 | Function Description | 5 |
| 4 | Specifications | 9 |
| 5 | GTU operation | 15 |
| 6 | Programming Requirements | 16 |
| 7 | Front Panel and Back panel IO ports | 27 |
| Appendix A | Citations | 28 |
| Appendix B | Revision History | 29 |
| Appendix C | Test plans and schedules | 30 |

# 1 Introduction

The Global Timing Unit (GTU) is being designed for the Electron Iron Collider (EIC) [1] experiment ePIC [2] data acquisition system [3]. The GTU should be compatible with the second experimental detector at EIC. This module is the interface between the EIC machine control, DAQ run control user interface, and the data acquisition system of the experiment. The GTU acts as the central control point for data acquisition. It gets the beam synchronized clock and beam structure from EIC common platform [4], and distribute the synchronous clocks to the front end electronics (FEB) (ASIC for example) via Data Aggregation Module (DAM) [5] modules and the Optical Interface of Detector Readout (RDO) [6] modules.

Together with the DAM, RDO, and optic fibers, the clock and sync signals are distributed to the frontend electronics, and the status (BUSY for example) signals from the frontend electronics are monitored. Figure 1 shows the placement of the GTU, DAM, RDO and FEB in experiment setup.



Figure ePIC DAQ system

There is one GTU module in the system. The GTU will be located in the counting room in the middle of the online computer (ECHLON 0, with DAM modules) racks. The GTU connects to each DAM with an MTP fibers with a length of about ten meters. The DAM (in the quantity of about one hundred) connects (up) to 48 RDOs in the experimental hall near the detector via duplex fibers

# 2 Purpose of the GTU module

The GTU is the top level control module for Data acquisition system. It receives clock and beam information from the EIC control, user run control command through the Ethernet. It generates synchronized run control commands to the DAQ system, which includes streaming data time intervals.

When the EIC clock is not available (pre-experiment test for example), the GTU can generate system clock with the proper EIC beam structure.

In addition to the streaming readout, the GTU can also initiate trigger for legacy DAQ compatibility.

The GTU should support subsystem partitions, so that the different detector sub-system(s) can run in parallel (simultaneously and independently). This feature is especially useful in detector commissioning stage. To the users (detector commissioners for example), each has full control of the run-control system.

# 3 Functional Descriptions

3.1 General description

Figure 2 shows the block diagram of the GTU module, indicating the major components used in the design. Three AMD Ultrascale+ FPGA (one xczu19P, .two xcku15p) will be used instead of one bigger FPGA considering the printed circuit board design.



Figure 2 GTU functional diagram

3.2: Clock generation:

One of the GTU’s major functions is the EIC beam crossing synchronized clock distribution. As the ePIC is using (and adapting) ASICs developed for LHC experiments, some frontend electronics require a clock at about 40 MHz (±1 MHz).

The EIC beam crossing clock is 98.5 MHz, not a multiple or division of 40 MHz. The closest (not too complicated) clock frequency is: (98.525 MHz \* 13 / 32 =) 40.026 MHz, and the simplest is (98.525 MHz \* 2 / 5 =) 39.41 MHz, which is within the requirement. Another limiting factor is the beam orbit, 1260, which is 4 \* 5 \* 7 \* 9. We have chosen the simple one, 19.7 MHz (98.525/5) as the clock distributed from GTU to DAM. The DAM will recover the 39.4 MHz clock, and/or the 98.5 MHz clock phase aligned with each other.

The GTU divides the EIC beam crossing clock to get a lower frequency clock as ePIC system clock. The DAM multiplies the system clock to get the required clock frequencies. As the divider is synchronous with the beam orbit start (first beam crossing in the orbit), all the recovered clocks are phase aligned at the orbit start.



Figure 3 ePIC clock and the EIC beam crossing

3.3: Clock stability control:

The high precision Time of Flight (TOF) detectors are using the clock edge as the START of the particles flight time. The dRICH detector will use the clock edge for shutter control to reduce the background. These require a precise phase between the EIC beam crossing and the clock. Our plan is to measure the electron beam phase relative to a clock (near the ePIC detector), and feed the measurement back to the GTU to adjust the distributed clock phase, so that the phase between the electron beam and the clock in the hall stays constant (fixed).

If the clock from EIC common platform CANNOT guarantee the clock (to GTU) is phase aligned with the beam (crossing), without any drift (<5 ps), we will need a (electron) beam pick up signal, and measure the phases between the beam pickup and a clock recovered in the hall. The measurement result is fed back to the GTU, and adjust the distributed clock phases, so that the phase between the hall recovered clock and the beam pickup is fixed. Figure 4 shows the phase control plan.



Figure 4 Phase alignment of the ePIC clock with EIC electron beam.

If the clock from EIC common platform CAN guarantee the clock (to GTU) is phase aligned with the beam (crossing), without any drift (<5 ps), the TClinks technique (developed in CERN) can be implemented (between DAM and RDO) to keep the RDO clock phase aligned (and fixed) with the beam crossing.

To achieve this, we need to collaborate with other group to get the electron beam pick up signals. Either way, it is nice (and safer) for the GTU to be able to monitor the beam crossing phases and the (RDO recovered) clock phases (also see section 3.7).

3.4 Fiber links

The 40 Gb QSFP optic transceivers (FINNISAR FTL410QD4C for example) will be used on the GTU. The QSFP is chosen because they are industry standardized, widely used, and economical (less expensive). The MTP fibers are chosen because they are less expensive. This enables the direct connection between the GTU and the DAM (without any patch panels). The simpler system means more reliable system. Figure 5 shows the QSFP/MTP connection between the GTU and the DAM.



Figure 5 QSFP/MTP connection between the GTU and the DAM

On the GTU, three of the four transmitters of the QSFP are used, and two of the four receivers are used. On the DAM, two of the four transmitters are used, and three of the four receivers are used. Figure 6 is an example of the downlink and uplink data. The data format will be changing as the GTU development progresses.



Figure 6 An example of possible data format for the links between the GTU and DAM modules. The downlink is an example of 19.7 MHz system clock, the uplink is an example of 9.85 MHz system clock.

For the downlink, DAM will re-align (to correct for the uncertainty of the MGT transceivers) the packets with the system clock to get the precise timing of the run control commands, and repack them in its clock domain (985 MHz, or 39.4 MHz) and sends to the RDO/FEB.

3.5: GTU partitions (sub-systems grouping)

To facilitate the detector commissioning and sub-system tests, the GTU can be partitioned, so that many users (DAQ Run control interfaces) can run in parallel (simultaneously and independently). Because the connections between the GTU and each DAM are independent (though they are common in clocks), the minimum unit of the partition is a DAM (one MGT transceiver, and one FPGA IO transceiver).



Figure 7 An example of GTU partition implementation

Each MGT/DAM is assigned an identification number, and each Run\_Control\_User is assigned an identification number. The GTU accepts the Run\_control\_user’s instructions, and routes to the proper MGT/DAM groups (sub-system). As of today, there are 24 subsystems. There should be no problem to support 24 partitions. The actual implementation will depend on the GTU development. The implementation will be the most practical option (simplest, and most reliable) for the firmware and software.

3.6: DAQ streaming support

The GTU will send a synchronous time-window START signal to the DAM. The size of the time window is settable on the GTU. The nominal time window width is about 600 µs, with the step of (about) 50 ns (five beam crossing clock periods). The GTU can also send a time window STOP signal, which instruct all the data slice assembler to dump the data after this END signal. This will cause ePIC global system busy, and dead time. We are designing the DAQ to have enough buffer space so this will never happen during the run. The START signal will be sent periodically, so that the DAQ can end the current time slice, and start a new time slice. The DAM can also implement a timer to check the synchronization with this START signal.

****

Figure 8 Streaming data time slices

3.7: Clock monitoring

With spare fibers laid for the ePIC daq system, a GTU plugin card is planned for the fiber loopback (passive loopback by a patch fiber, or active loopback with QSFP, SFP or VTRx+/LpGBT electrical loopback). The plugin card will measure the clock phases to monitor the clock distribution phase shift over the fibers versus the environment (temperature), and send the measurement results back to the GTU base board. Figure 9 is a functional diagram of the plugin card.



Figure 9 Functional diagram of the GTU distributed clock phase monitoring plugin card.

# 4. The hardware design and specification

4.1 Mechanical

The GTU is a 4U high, 19 inch rack mounted box, and requires another (at least) 1U gap above for warm air outlet. The depth of GTU has not been determined yet, but it will definitely fit in a rack over 18 inches deep.



Figure 10 The GTU, a 4U high rack (19” wide) mounted ‘computer’ box

The back panel (the left side in figure 10) has 144 QSFP ports connecting to (up) to 144 DAM modules through MTP fibers in the vicinity computer racks. The MTP fibers will be (about) 10 meter in length, 850 nm multimode 8 or 12 stranded OEM4 optic fibers. There may be a tray in the front to supply cable tension relief.

The front panel (the right side in figure 10) has four cooling fans (one built in the power supply AC-DC converter) draw air from the front panel, and push the cool air through the box, and out of the top vent (back half). The front panel implements the necessary IO ports for the efficient operation and debug of the GTU.

The detailed IO ports of the front panel and the back panel are documented in section 7, Front Panel and Back panel IO ports.

4.2: GTU PCB design:

The GTU box has a similar structure as a rack mounted workstation computer, where the GTU base board as the computer motherboard, and the optic transceiver daughter cards as the PCIe plug in cards. An ATX AC-DC converter is used to power the GTU.

 

Figure 11 The GTU optic transceiver daughter card (left) and the GTU base board (right)

### **4.2.1 GTU base board**

To speed up the GTU development and reduce the GTU cost, three SOM modules are used on the GTU base board, with the programmable resources of one AMD xczu19eg ZYNQ UltraScale plus MPSoC, two AMD xcku15p Kintex UltraScale plus FPGAs. There are 5 GB DDR4 memory for the quad core ARM® processor and 4 GB DDR4 memory for each FPGA (or the programmable logic of the ZYNQ MPSoC). The width of the GTU base board (~425 mm) is limited by the 19” rack width standard. There is no hard limit on the depth of the PCB except the production cost.

### **4.2.2 GTU plug in modules**

There will be (at least) three kinds of GTU plug in modules, but all of them will be using the PICexpress connector compatible design (power, ground, and signal pin usage).

4.2.2.1 The optic transceiver daughter card for DAM

There will be (up to) eighteen this kind of cards per GTU. Each of this card (as shown in figure 11 left side) will have eight QSFP transceivers connecting to eight DAM modules. The height of the GTU optic transceiver daughter card (~170 mm) is limited by the 4U high standard. These cards will be plugged in to the lower left 18 slots of the GTU base board.

4.2.2.2 The optic transceiver daughter card for RDO clock distribution

There will be (up to) three this kind of cards per GTU. Each of this card will have eight (maybe nine) MTP-24 connectors. With the MTP🡪LC harness, each card can distribute a dedicated clock to 192 (maybe 216) RDOs in the hall. These cards will be plugged in to the three upper left slots of the GTU base board.

4.2.2.3 The clock monitoring daughter card

There will be one card per GTU. On this card, the same clock buffer chips and the optic transceivers will be used. An FPGA SOM will be implemented to measure the loopback clock phases (relative to the initial clock from EIC common platform). The measurement results will be transmitted to the GTU base board. This card will be plugged in to the top right slot of the GTU base board.

4.3: Clock generation and fanout circuits:

One of the GTU’s major functions is the EIC beam crossing synchronized clock distribution. The EIC beam crossing clock, and the on-board oscillator are received by a high precision 4x4 cross-point switch (SY58040U). The clock (98.5 MHz nominal frequency) is frequency divided by five (phase aligned with the beam orbit) using a clock synthesizer (AD9510) to get the 19.7 MHz system clock (if divided by ten, the system clock will be 9.85 MHz). The divided clock is fed to another clock synthesizer and jitter cleaner chip (Si5395A) to generate the high quality system clock (19.7 MHz), and the beam crossing clock (98.5 MHz). The system clock (19.7 MHz) is further fanned out using ultra low jitter clock chips (Si53302) to all the DAM boards. The recovered beam crossing clock is further fanned out for FPGA’s Multiple Gigabit Transceivers (MGTs) reference clocks. Figure 11 shows the details of the clock distribution on the GTU.



Figure 12 detailed GTU clock distribution

All the Si5395A chips (including the chip used on DAM boards) are set to Zero Delay Mode, so that all the clocks in ePIC are phase aligned (known and fixed phase at the first beam crossing of every EIC beam orbit). Figure 4 shows the relations between the EIC beam crossing and the clocks in the system. The GTU distribute system clock in either 19.7 MHz or 9.85 MHz. (The clock divider AD9510 prefers 9.85 MHz for a symmetric output clock, but the optic transceivers prefer 19.7 MHz for a higher frequency. The clock frequency will be decided after the real GTU tests)

4.4: The GTU to DAM circuit:

The large number of one-to-one connection between the GTU’s MGT and the DAM requires a large number of MGTs on the GTU base board. A single FPGA option (AMD versal® xcvp2902, or Vrrtex UltraScale plus® xcuv19p) would cost too much and post higher engineering risks. Instead, three SOM are used. The minor disadvantage of the three SOM design is the interconnections among them.



Figure 13 Detailed MGT allocation on the GTU

Each FPGA will connect to 52 DAM modules, and the Zynq will connect to 40 DAM via the optic transceiver daughter cards. There are four DAM connections on the two Kintex FPGAs (not via the Optic Transceivers Daughter Card).

There are two MGT, and 24 general IO between the Zynq MPSoC and each Kintex FPGA.

The default user interface to the GTU is the 1000BASET Ethernet port. A 40 GbE (or 100 GbE) fiber port can be developed if required. The hardware is implemented, but need firmware and software support.

A PCIexpress Gen2x4 root port can be developed for further GTU capability expansion.

4.5: The GTU slow control circuit (I2C and SPI):

The Zynq MPSoC can supply two I2C masters for slow control. The first I2C is used to monitor and set parameters for the clock related components. The second I2C is used to monitor the QSFP optic transceiver modules, like temperature, voltage, receiver optic power level, transmitter bias currents etc. Several layer of I2C hubs (TCA9548A) are used on the GTU base board and the daughter cards.



Figure 14 GTU I2C slow control

The SPI\_0 from the MPSoC connects to the AD9510 directly.

4.6: The GTU FPGA programming circuit:

### 4.6.1 JTAG

A JTAG connector (10-pin, dual row, 100 mil pitch) is implemented on the GTU baseboard front panel. With the 3-position switches, any one of the three AMD devices can be selected, any two of the three devices can be selected, and all the three can be selected in a daisy chain connection. The three AMD devices and their connected SPI flash memories (Micron MT25QU512 for example) can be programmed via the JTAG programmer.

### 4.6.2 SD card

The Zynq MPSoC can also be programmed by a SD card, which is accessible via the GTU front panel.

### 4.6.3 remote firmware update

There are several ways to update the GTU firmware remotely.

The first way is to update the SD card with the right firmware. The SD card can be updated via another computer, or the MPSoC itself.

The second way is to update the connected SPI flash memory directly provided the FPGA is loaded with the proper firmware (like I did for the PCIexpress TI).

4.7 Front panel:

Here is a picture (not available yet) of the front panel. The three 12V, 120mm cooling fans are mounted on the upper right with finger protection grills. The ATX AC-DC converter is mounted on the upper left with built in cooling fan and the AC input connector.

The lower part of the front panel is for the connectors directly mounted on the GTU base board. They are:

* One 1000baseT Ethernet port;
* One SD card slot;
* two SMA connectors for EIC clock input;
* SFP connector for EIC clock input;
* Four QSFP connectors for DAM connection
* One QSFP for 40 GBE (or 100 GBE)
* One USB port (type B or mini USB);
* GTU power on/off switch
* Two ten-pin 100 mil dual row connectors, connected to the Zynq PL IO
* One ten-pin 100mil dual row connector for JTAG
* Two outputs and two inputs low speed optic (ST) ports (support DC signal)

4.8 Back panel:

The back panel looks simple, but the full 4Ux19” panel are used. It looks like a 21 slots crates. The right side is eighteen slots with eight QSFP transceivers (MTP fiber), with a total of 144 ports. These MTP fibers will connect (up) to 144 DAM modules. The left side is three slots for dedicated clock output to the RDOs, each slot will have eight (or nine) 24-strand MTP fibers. This MTP fibers will be split into single fiber cables (most likely LC style) connecting (up) to 576 (or 648 depending on the spacing between the MTP fiber connectors) RDOs in the experimental hall.

4.9 Programming:

The main interface between the users and the GTU is the 1000baseT Ethernet. A built-in multicore ARM® processor runs a Linux OS. It configures, controls, and monitors the GTU via a block of memory-mapped registers (detailed in section 6, register space allocation).

4.10 Power requirements (to put in the numbers after the GTU test):

AC wall connector. The built in DC power converter is ATX standard, 110~240 Volt, 50~60 Hz at 10 A maximum.

The fully loaded GTU box (GTU base board with all the 21 optic transceiver daughter cards) draws 5 amperes on the 110 volt AC supply, and the DC power usage are:

* +12 Volt: 20 Amperes;
* +5 Volt: 10 Amperes;
* +3.3 Volt: 20 Amperes;

4.11 Environment:

Forced air cooling: GTU built in cooling fan,

Commercial grade components (0-75 Celsius or better).

# 5 GTU operation procedure:

The GTU needs be properly configured on power up, though the power up default will enable the GTU’s essential functionality.

5.1 GTU Power supply:

The GTU uses an ATX standard AC-DC converter. The 24-pin power connector is used. The dual supply improves its reliability.

5.2 Hardware setting (Switch etc.):

A four-bit slide switch, SM1, is used to set the Zynq Ultrascale+ MPSoC boot up/configuration mode. When ON, the mode is shorted to ground and hence low (or ‘0’), when OFF, the mode is pulled up high (or ‘1’). Here is the four mode pins MODE(4:1):

* When “0000”: JTAG configure;
* When “1000”, SD card;
* When “0010”, SPI Flash memory

Four 3-position (pull-up, pull-down, no-pull) switches are used to set the power up default of the clock distributions. These switches are:

* SC1: default Si5395 input select\_0;
* SC2: default Si5395 input select\_1;
* SC3:
* SC4:

Four 3-position switches are used to set the default JTAG chain for FPGA programming/debug. These switches do not matter in normal GTU operations. These switches are:

* SJ1: bypass the Zynq Ultrascale+ MPSoC;
* SJ2: bypass the top Kintex Ultrascale+ FPGA;
* SJ3: bypass the lower Kintex UltraScale+ FPGA;
* SJ4: ?

5.3 Software setting:

After the GTU is powered up, and configured (Zynq UltraScale+ MPSoC by the SD card, Kintex Ultrascale+ FPGAs by their SPI flash memories), some software setting needs be applied for the GTU to work correctly. The GTU software setup sequence:

* GTU clock and orbit signal selection: The ePIC clock source, and the beam structure;
* GTU clock configuration: system clock generation, phase tuning and monitoring;
* MGT ID setting, if it is different from the power up default.

# 6. Memory-Mapped Registers Programming Requirements

There are four register spaces from the ZYNQ processor. Three of them (16 KB each) are equal in size for the zynq programmable logic, the top Kintex FPGA, and the bottom Kintex FPGA respectively. The fourth one is for ZYNQ only. The interfaces are standard axi\_lite4.

6.1 The three equal spaces, Configuration Registers for the programmable logic:

The base address is determined by the ZYNQ memory map. The register offset (in the 16 KB area) are accessible in 32-bit words only (address[1:0] is always “00”).

* Address offset: 0x00000: Board ID:

Bit 7-0 (R/W): Crate ID; Reset default 0x00;

Bit 15-8 (R): Firmware version numbers;

Bit 31-16 (R): FPGA ID code.

* Address offset: 0x00068 : single word readout of the data buffer

Bit 31-0 (R): FIFO data

* Address offset: 0x00078: Forced Synchronous code

Bit 7-0: Run control code,

* 0x11: VME clock DCM reset, and full reset;
* 0x22: CLK250 resync (AD9510, DCM resync and MGT reset);
* 0x33: AD9510 re-sync (slower clock phase adjustment), part of 0x22 function;
* 0x44: Reset the MGT status\_B registers;
* 0x55: Trigger link enable (serial link started), FIFO read counter reset;
* 0x77: Trigger link disable, trigger FIFO write counter reset;
* 0x88: PCIexpress core reset. (be careful when using it).
* 0xAA: reset the TI\_trigger\_enabled registers on TD.
* 0xBB: Event number reset, and trigger input scalar reset;
* 0xDD: (SyncReset), FPGA logic and counter reset, this reset all goes to SD, CTP/GTP;
* 0x99: Force SyncReset high if this feature is enabled (by offset 0x24, bit 7);
* 0xCC: set the SyncReset low if it is forced high by code 0x99.
* 0xEE: set the syncReset low for ~4 us.
* 0x66: to be assigned;
* 0x00, 0xff: reserved, not to be assigned

Bit 15-8: Repeat the run control code of Bit 7-0, if not matching, no code will be sent to DAM.

* Address offset: 0x000E4 (R/W): Register to Flash memory SPI engine. The Read and the Write has different meanings. This is for the first QSPI flash memory only.

Bit 9 (W): ‘1’, Last command/write bit, the DO will be tri-stated at the end of this clock;

Bit 8 (W): ‘0’, last transfer, the Chip\_Select\_B will be disabled (high) at the end of this clock;

Bit 7-4 (W): ‘1’: Tri-state the DO;

Bit 3-0 (W): DO(3:0);

Bit 23-20 (R): EOS, PREQ, FCSBO, FCSBTS of the flash interface;

Bit 19-16 (R): DI(3:0), Flash outputs. DI(1) for SPIx1, DI(1:0) for SPIx2, DI(3:0) for SPIx4;

Bit 15-0 (R): Data(15:0) of the 0xE4 write immediately before this read.

* Address offset: 0x00110 (R/W): First MGT of the first MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00114 (R/W): Second MGT of the first MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00118 (R/W): Third MGT of the first MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x0011C (R/W): Fourth MGT of the first MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00120 (R/W): First MGT of the second MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00124 (R/W): Second MGT of the second MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00128 (R/W): Third MGT of the second MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x0012C (R/W): Fourth MGT of the second MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00130 (R/W): First MGT of the third MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00134 (R/W): Second MGT of the third MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00138 (R/W): Third MGT of the third MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x0013C (R/W): Fourth MGT of the third MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00140 (R/W): First MGT of the fourth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00144 (R/W): Second MGT of the fourth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00148 (R/W): Third MGT of the fourth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x0014C (R/W): Fourth MGT of the fourth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00150 (R/W): First MGT of the fifth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00154 (R/W): Second MGT of the fifth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00158 (R/W): Third MGT of the fifth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x0015C (R/W): Fourth MGT of the fifth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00160 (R/W): First MGT of the sixth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00164 (R/W): Second MGT of the sixth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00168 (R/W): Third MGT of the sixth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x0016C (R/W): Fourth MGT of the sixth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00170 (R/W): First MGT of the seventh MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00174 (R/W): Second MGT of the seventh MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00178 (R/W): Third MGT of the seventh MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x0017C (R/W): Fourth MGT of the seven MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00180 (R/W): First MGT of the eighth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00184 (R/W): Second MGT of the eighth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00188 (R/W): Third MGT of the eighth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x0018C (R/W): Fourth MGT of the eighth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00190 (R/W): First MGT of the ninth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00194 (R/W): Second MGT of the ninth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x00198 (R/W): Third MGT of the ninth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x0019C (R/W): Fourth MGT of the ninth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001A0 (R/W): First MGT of the tenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001A4 (R/W): Second MGT of the tenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001A8 (R/W): Third MGT of the tenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001AC (R/W): Fourth MGT of the tenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001B0 (R/W): First MGT of the eleventh MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001B4 (R/W): Second MGT of the eleventh MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001B8 (R/W): Third MGT of the eleventh MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001BC (R/W): Fourth MGT of the eleventh MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001C0 (R/W): First MGT of the twelfth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001C4 (R/W): Second MGT of the twelfth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001C8 (R/W): Third MGT of the twelfth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001CC (R/W): Fourth MGT of the twelfth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001D0 (R/W): First MGT of the thirteenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001D4 (R/W): Second MGT of the thirteenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001D8 (R/W): Third MGT of the thirteenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001DC (R/W): Fourth MGT of the thirteenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001E0 (R/W): First MGT of the fourteenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001E4 (R/W): Second MGT of the fourteenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001E8 (R/W): Third MGT of the fourteenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001EC (R/W): Fourth MGT of the fourteenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001F0 (R/W): First MGT of the fifteenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001F4 (R/W): Second MGT of the fifteenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001F8 (R/W): Third MGT of the fifteenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

* Address offset: 0x001FC (R/W): Fourth MGT of the fifteenth MGTQUAD ID setting

Bit 0: ‘1’: MGT active; ‘0’: MGT off;

Bit 24:1: The MGT ID, only one-bit to be set as ‘1’;

Bit 29:25: The decimal code of the run control user ID (user interface ID);

Bit 31:30: MGT status.

6.1.2 Address offset 0’b0000,1xxx,xxxx,xx00 (Address Bit(12:11) = 01): Register to I2C (master) engines:

The middle 2K bytes are used for PCI express to I2C engines. Address Bit(10) is used to select the I2C engines. The lower eight bits of the 32-bit data (Address Bit(1:0)=”00”) are used for I2C read or write.

Address Bit(10): ‘0’: for optic transceiver QSFP#A I2C; ‘1’: for transceiver QSFP#B I2C.

Address Bit(9:2): eight bits of the I2C byte address.

Address Bit(1:0): to be set as “00”.

The device address is generated as “0xA0” automatically as required by the QSFP I2C interface standard. (This means that the engine, firmware design, will not work as generic I2C engine, which require 7 bits of device address, or 32K bytes memory space minimum).

6.1.3 Address offset 0’b0001,xxxx,xxxx,xx00 (Address Bit(12) = ‘1’): register to JTAG engine:

The upper 4K bytes are used for PCI express to JTAG engines. Address Bit(12) = 1, and Address Bit(11) for the two JTAG engines. There is no specific use of the JTAG on the GTU yet. For data transfer less than 32 bits, the lower N bits (specified by the address) are used.

Address Bit(10:6): Number of bits to shift (Instruction register or Data register), n=Addr(10:6)+1, that is the minimum number of bits to shift is 1, and maximum number of bits to shift is 32, the lower order of bits in the 32-bit data are used.

Address Bit(5:4): Specific JTAG command. “01”: JTAG data register shift; “10”: JTAG instruction register shift; “11”: JTAG reset, (set the JTAG to RESET\_IDLE state).

Address Bit(3): JTAG TRAILER\_ENABLE. This will enable several extra JTAG clocks (after register shift) to move the JTAG state machine from register shift (either data register or instruction register) to RESET\_IDLE.

Address Bit(2): JTAG HEADER\_ENABLE. This will enable several extra JTAG clocks (before register shift) to move the JTAG state machine from RESET\_IDLE to register shift (either data register or instruction register depending on the shift type A(5:4))

Address Bit(1:0): “00” only.

The register read to the device will return the data currently stored in the TDO shift register (32-bit). The read address is Address Bit(11). Bits(10:0) are DONOT care bits. 32-bit data read only.

6.2 The ZYNQ (only) register space:

The base address is determined by the ZYNQ memory map. The register offset is accessible in 32-bit words only (address[1:0] is always “00”).

# 7 Front panel and Backplane pin out tables:

Figure 11 shows GTU front panel, and the connector names.

Figure 12 shows GTU back panel , and table 1 shows the corresponding DAM connection

Table 1 GTU to DAM MTP fiber connections

|  |  |  |
| --- | --- | --- |
| QSFPport  Column\_Row | GTU MGT port#  FPGA\_BANK\_GT | DAM device  Computer\_Detector\_ID |
| C1\_R1 | KB\_224\_0 | C01\_dRICH\_1 |
| C1\_R2 | KB\_224\_1 | C02\_dRICH\_2 |
| C1\_R3 | KB\_224\_2 | C03\_dRICH\_3 |
| C1\_R4 | KB\_224\_3 | C04\_bTOF\_1 |
|  |  |  |
| C7\_R6 | ZYNQ\_132\_1 | Not assigned |
|  |  |  |
| C8\_R6 | ZYNQ\_128\_1 | C80\_muon\_3 |
|  |  |  |
| C18\_R8 | KA\_232\_3 | C144\_eTOF\_9 |

Table 2 GTU to RDO dedicated clock distribution MTP fiber connections

|  |  |  |
| --- | --- | --- |
| MTP  Column\_Row | GTU Clock port#  Clock\_FiberTx | RDO device  DAM\_Detector\_ID |
| C19\_R1 | CA\_Samtec#1\_2 | D05\_dRICH\_1, 2, 3, …, 24 |
|  |  |  |
| C21\_R9 | CC\_Samtec#17\_18 | D12\_bTOF\_25, 26, …, 48 |

Appendix A: Citations:

1. EIC
2. ePIC
3. ePIC DAQ
4. EIC common platform
5. DAM
6. RDO

Appendix B: Revision history:

Mar. 10, 2025: First version, Initial release

Mar. 11, 2025: Added the GTU test plans

Mar. 20, 2025: Added the mini-DAM in GTU test plan.

May 23, 2025: Added section 3.7 for distributed clock phase monitoring (over the fibers), and further elaborated the GTU PCB design.

Appendix C: Test plans and Schedule:

Make an FMC\_HPC adaptor with QSFP\_GTU for GTU connection, and QSFP\_RDO for four RDO connections. The Si5395 to be implemented the similar way as that on FLX155 (ePIC-DAM).



ALINX’s AXAU15P with custom designed FMC adaptor as mini-DAM.