ProtoDUNE-SP Timing System: Interfaces and Protocol

D. Newbold, D. Cussans, J. Brooke

v4 – 27th October 2016

Contents

Summary

This document acts as the repository for:

- The definition of interfaces at hardware, firmware and software level between the timing system and the other components of ProtoDUNE-SP
- The definition of the internal protocol and assignments of command encodings, etc.

It is intended to be updated periodically as the specification is finalised and amended.

1 Overview

The timing system is required to: provide a stable and phase-aligned master clock to all DAQ components; receive external signals (including triggers) into the ProtoDUNE clock domain and time-stamp them; distribute synchronization, trigger and calibration commands to the DAQ system; and conduct continuous checks of its own function. In addition, the timing system acts as a data source, providing a record of timing signals received, distributed, or throttled. The system is designed to meet the full eventual requirements of the DUNE experiment, but will need only a subset of that functionality for ProtoDUNE.

An FPGA-based master unit receives a high-quality clock (provided in ProtoDUNE by a GPS-disciplined oscillator) and external signals from the trigger system and SPS accelerator. It interfaces to the ProtoDUNE control and DAQ via a gigabit Ethernet interface. The master unit multiplexes synchronization and trigger commands, along with arbitrary command sequences generated by software, into a single encoded data stream, which is broadcast to all timing endpoints, and decoded into separate clock and data signals. A uniform phase-aligned cycle counter, updating at the ProtoDUNE system frequency of 50MHz, is maintained at all endpoints, allowing commands to take effect simultaneously at all endpoints regardless of cable lengths or other phase delays.

The timing signal is broadcast via multi-mode optical fiber, with conversion to LVDS signals over short twisted pair cable for backwards compatibility with existing timing interfaces (e.g. on SSPs). The system uses duplex links, allowing all endpoints to be regularly interrogated during system operation to verify correct operation and reception of timing commands. Along with the possibility to select any given endpoint via a unique network address, this mechanism also allows automatic phase-adjustment of all local clocks at system start-up, through precise measurement of the returned clock phase at the master unit. The definition of timing groups allows the system to be partitioned into a number of independent timing zones.

Endpoints decode the timing signal into separate clock and data signals using a commercial clock-data-recovery device, which in turn can feeds a low-bandwidth PLL for applications requiring a very low jitter local clock. The data stream employs 8b/10b encoding, ensuring sufficient transitions in the timing signal for clock recovery and correct operation of optical links, and uses a random idle pattern to minimise EMI from copper segments. A common firmware block is used to decode the timing protocol, which is incorporated into the overall firmware design for the receiving FPGA in each DAQ component. This block provides a cycle counter, several independent trigger, calibration and synchronisation signals, and a general-purpose packet data output to each endpoint. The cycle counter may be used further to generate low-frequency timing signals for further propagation, e.g. the 2MHz sampling signal for the cold ADCs.

2 Timing Master Interfaces

2.1 Timing Master Functions

The timing system operates in a master-slave configuration. All timing signals are broadcast by the master, and interpreted by slaves. Responses from slaves are only generated when requested

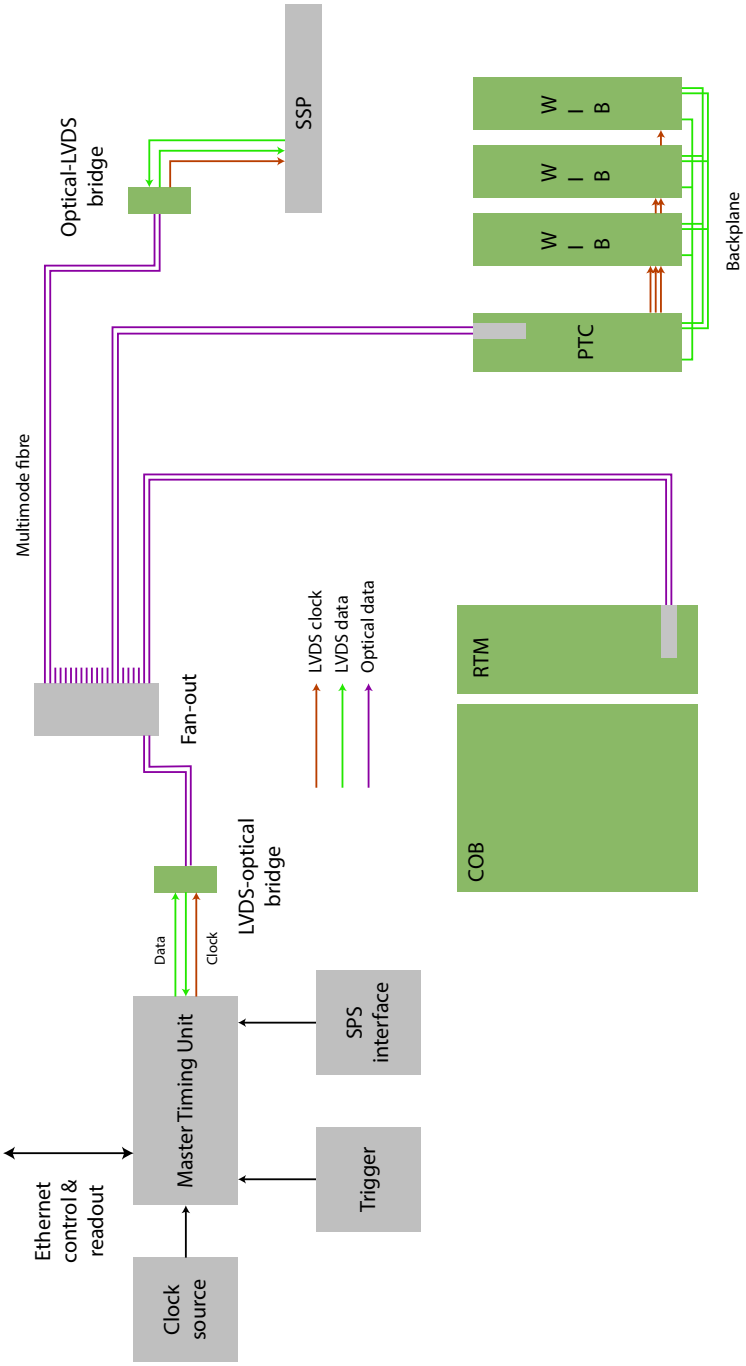

Figure 1: Simplified system overview

by the master.

The timing master has the following functions:

- Reception of master clock signal and serialised time-of-day information (e.g. from an external GPS receiver)
- Reception of external timing and trigger signals
- Logging and time-stamping of signals received, distributed or throttled, and transmission to DAQ
- Serialisation of timing commands and transmission to the timing network
- Phase measurement of incoming timing signals from slaves, allowing phase adjustment under software control
- Transmission of arbitrary commands and control data under software control

2.2 Timing Master Implementation

The master unit is physically implemented as a pair of custom FMC cards, hosted by an FPGA-based carrier module. The interface FMC (based on the AIDA mini-TLU [1]) receives external signals from the trigger, SPS accelerator interface, and other external sources, and provides a number of spare general-purpose IO interfaces. The timing FMC, which is also used elsewhere in the timing system, provides clock-data recovery and jitter reduction functions and the serial optical interface to the timing system.

2.3 Master Clock Interface

The master unit receives a clock frequency reference from an external source (e.g. GPS-disciplined oscillator or White Rabbit clock distribution system). In the absence of an external source, including under laboratory conditions, the system clock may be produced internally from a programme quartz oscillator source. For ProtoDUNE-SP, the clock frequency is fixed at 50.0MHz, dictated by the requirements of the TPC front end electronics.

In addition, the master unit can receive a synchronized serial data stream from an external source for external timestamping of events. The master unit is able to output the system clock or a divided version of it for monitoring or other purposes.

2.4 Trigger Inputs

The master unit receives trigger information via an serial link, using the same protocol as for general timing information. Since the timing master and trigger processor are expected to be adjacent, this link is implemented using LVDS signals.

The master unit receives up to six external trigger inputs via single-ended coaxial connectors (LEMO 00). These inputs feed discriminators, and may be used as edge or level sensitive inputs for the trigger distribution logic. On arrival, signals are registered into the ProtoDUNE-SP timing

domain, and the receipt of signals is recorded and available to DAQ. Since external signals not aligned with the system clock must be re-registered, it is not guaranteed that signals that are synchronous in the driving clock domain will be registered on the same 50MHz cycle.

2.5 Other External Inputs

Non-trigger external inputs may be received synchronized into the ProtoDUNE-SP timing domain, and used either as stimulus for automatic synchronous commands, or gates or vetos for any trigger signal. An example of such a signal is the SPS spill warning signal.

2.6 Run Control and DAQ Interface

The master unit interfaces to the ProtoDUNE-SP computing system for run control and DAQ, via a gigabit Ethernet interface. The system uses the IPbus protocol for all communication. Functions available to software include:

- Timing system reset and configuration
- Measurement of endpoint clock phases
- Enabling and disabling of automatic synchronous commands (i.e. those issued in response to trigger inputs)
- Configuration of periodic synchronous commands (e.g. for calibration)
- Issuing of asynchronous commands
- Readout of buffered trigger information

FUTURE_UPDATE Full specification of registers, etc, should be documented here.

3 Timing Protocol and Transport

3.1 Timing Signal Functions

The 50MHz clock and timing commands are distributed to all timing endpoints as a broadcast signal. However, synchronous commands may be addressed to specific timing groups, and asynchronous commands may be addressed to individual endpoints or timing groups.

3.2 Timing Signal Implementation

Conversion between the transport methods defined below will be handled by an conversion module, implemented by the timing FMC card. Conversion takes place transparently at protocol level. The FMC card also acts as a test platform for protocol development, and as a reference design for the timing interface of ProtoDUNE-SP DAQ modules. A top-level diagram of the FMC signal paths is shown in Figure 2.

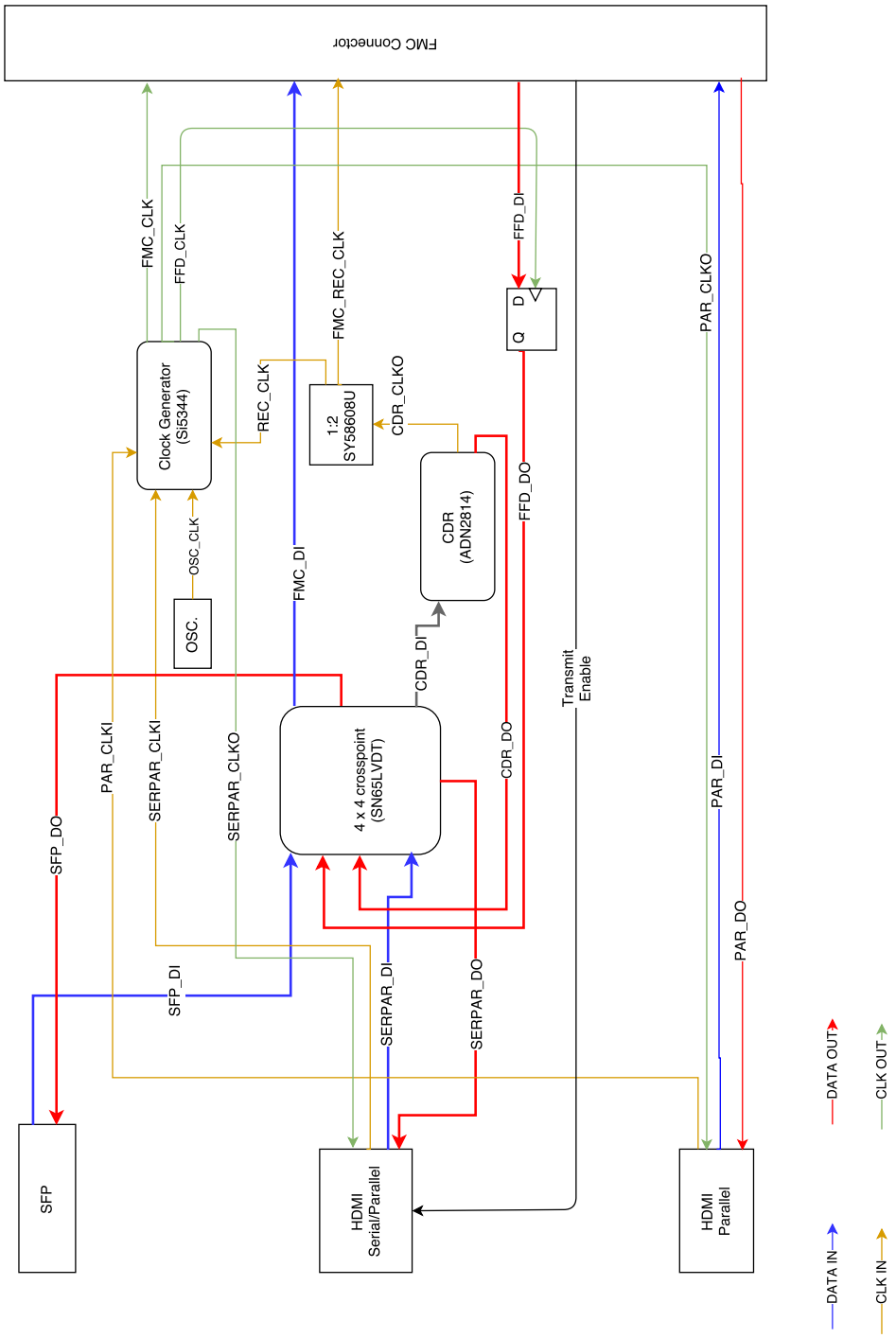

Figure 2: Timing FMC signal paths

3.3 Layer 0: Physical Signalling

Clock and data are typically distributed as a single signal, using a CDR device at endpoints to recover separate clock and data signals. However, some endpoints may need to be supplied with a separate clock signal, and this is supported via the timing FMC.

The data stream is encoded as a 100Mb/s DC-balanced signal suitable for distribution via optical or electrical interfaces. This low data rate allows the use of inexpensive components designed for 100Mb/s ethernet: either 100base-SX fibre optics, or Cat5/6 twisted pair cable with RJ45 connectors.

There are three defined transport methods for the timing signal:

1. Encoded data stream transported via optical fibre. The optical interface is implemented via commercial plug-in SFP modules, allowing flexibility in implementation. It is anticipated for ProtoDUNE-SP that 100base-SX components will be used, i.e. using simplex multimode fibre at 850nm wavelength for up to 500m transmission distance. Division and recombination of signals will be via passive commercial splitter modules (for lab use), with inactive endpoints disabling their optical transmitter, or via an active fan out (for use at the experiment site with large numbers of endpoints).

2. Encoded data stream transported via LVDS over twisted pair. Two pairs will carry broadcast and return data respectively. Division of signals will be via an active fan-out unit if required (e.g. as implemented on the WIB crate backplane), recombination of signals will be via an active logical 'or' or Bus LVDS wire-or, with inactive endpoints transmitting continuous '0' level. This method is expected to be used on crate backplanes only.

3. Encoded data stream along with a separate phase-aligned clock signal, transported via LVDS over twisted pair. A third pair will be used to carry the clock signal. Returned data is transmitted on the same clock. This method will be used only for point-to-point signalling, for compatibility with existing interfaces.

Conversion between these three methods is transparent at protocol level, with the same encoded data stream used regardless of whether an explicit clock signal is transmitted.

3.4 Layer 1: Data Link

The data stream uses 8b/10b encoding to transmit one byte per five 50MHz clock cycles. The standard comma mechanism is used for word alignment, with K28.5 used as the comma character. The timing data stream consists of a sequence of synchronous packets, asynchronous packets, and idle packets, in that order of priority.

- Synchronous packets begin with a K28.1 character (/S/), and are of fixed two-character length. Synchronous packets may interrupt other traffic at any time, and carry synchronous commands that are guaranteed to be issued at endpoints in the correct order, and with a fixed timing relationship to their request at the master unit.

- Asynchronous packets are of arbitrary length, begin immediately on the first non-comma character encountered in the data stream, and end with a K28.5 character (/C/). Asynchronous packets carry general control information to endpoints, that is not guaranteed to arrive with a fixed latency.
- Idle packets are a particular type of asynchronous packet, transmitted when no other packet information is queued. They consist of a pseudorandom byte sequence in order to minimise EMI due to repetitive patterns during link idle. Idle packets are of short length to ensure sufficient comma density for rapid link alignment.

Return data uses the same protocol, with the exception that endpoints never send synchronous commands. Figure 3 shows an simplified example of the structure of the timing data stream. Note that, in practice, a minimum period of 1ms of idle packets is required after an endpoint is requested to send data, in order for the optical link amplifiers to stabilise and master unit CDR chip to lock.

3.5 Layer 2: Addressing and Protocol

The synchronous and asynchronous packet formats are shown in Tables 2 and 1.

Byte	Content
0	/S/ (K28.1)
1	Group / Command

Table 1: Synchronous packet format

Byte	Content
0	Address(7:0)
1	Address(15:8)
2	Command
3	Data byte 0
$3+n$	Data byte n
$3+n+1$	Checksum(7:0)
$3+n+2$	Checksum(15:8)
$3+n+3$	/C/ (K28.5)

Table 2: Asynchronous packet format

Synchronous packets contain a single four-bit timing group specifier, and a single four-bit command. The command field encodes sixteen separate synchronous commands, corresponding directly to the sync outputs of the decoder block. Every timing endpoint may be configured with a timing group mask, such that it accept synchronous commands addressed only to selected timing groups. This allows flexible partitioning of the system.

The timing protocol allows transmission of synchronous commands at a rate of no more than once per 10 clock cycles, which is adequate for the low rate of synchronous commands expected in both ProtoDUNE-SP and DUNE. In the event of clashing synchronous commands, a priority encoding scheme is used, such that lower-numbered commands pre-empt high-numbered ones, and lower-numbered timing groups pre-empt higher numbered ones. Under rare or pathological conditions this may result in a synchronous command being rejected; a record is kept of all propagated and rejected signals.

Asynchronous packets contain a single one-byte command followed by an arbitrary number of data bytes. The address is a unique experiment-wide 16b endpoint identifier, or has the special value of 0xfff in the top 12 bits for a broadcast packet, with the timing group identifier in the bottom four bits. The checksum is a standard CRC-16-CCITT, calculated on all packet bytes including the address. Return data packets lack the address field, but are otherwise identical. Idle packets are identified by the all-zeroes address, and contain random data which is not interpreted further by the endpoint, except to validate the checksum.

FUTURE_UPDATE Document the ProtoDUNE-SP address assignment.

3.6 Layer 3: Command assignment

Currently defined synchronous and asynchronous commands are shown in Tables 3, 4 and 5. In the case of synchronous command queue conflict, priority for transmission is assigned to lower-numbered commands (i.e. counter sync command has priority over all over traffic). Asynchronous commands meant to be interpreted internally by the timing system endpoint decoder use are distinguished by a 0 in the topmost bit of the command byte.

Cmd bit	Name	Notes
0	S_Sync	Used to align and check system cycle counter; issued at 1ms intervals
14	S_Spill	SPS spill warning toggle
15	S_Trig	Beam trigger signal

Table 3: Synchronous commands

Cmd	Name	Data bytes	Notes
0x0	A_Reset	0	Resets state of endpoint
0x1	A_Sync	8	Stores cycle counter value to be loaded at next S_Sync
0x2	A_Enable	0	Enables endpoint transmission
0x3	A_Status	1	Request endpoint status packet transmission
0x4	A_Adjust	2	Adjusts packet delay and clock phase of endpoint

Table 4: Asynchronous commands

FUTURE_UPDATE Define further synchronous and asynchronous commands.

Cmd	Name	Data bytes	Notes
0x0	R_Status	TBD	Endpoint status report

Table 5: Asynchronous return commands

4 Timing Endpoint Interfaces

4.1 Timing Endpoint Functions

A timing endpoint is characterised by the following properties:

- A single system clock phase, adjustable under timing system control
- A unique address
- A timing group mask

A conceptual diagram of an endpoint is shown in Figure 4. It consists of three major components:

- A clock and data recovery IC (e.g. ADN2814). This component is not required for endpoints with a clock + data interface.
- A PLL for jitter reduction and phase adjustment, with the latter function controlled from the endpoint FPGA. The PLL may be implemented as an internal clock controller (e.g. MMCM) within the FPGA, or may be an external component. This component is not required for endpoints with a clock + data interface, since phase may be controlled by the optical-LVDS converter module if necessary.
- A firmware protocol decoder, implemented inside the FPGA as part of the overall system firmware. This block receives and interprets the timing data stream, and provides decoded commands to other blocks inside the FPGA. It also monitors and counts errors observed in the data stream.

Error conditions monitored by the decoder include:

- 8b/10b decoder errors
- Packet checksum errors
- Counter-out-of-sync error
- CDR / PLL lock timeout

The status of the decoder may be monitored locally via the status flags (see below), or for soft errors, remotely via the R_Status command. An endpoint that is in error can be reset locally or via the timing system, and then brought back into synchronisation with the rest of the system at the next S_Sync command.

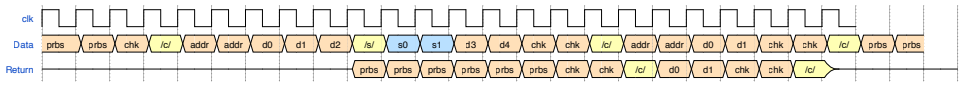

Figure 3: Example timing data stream

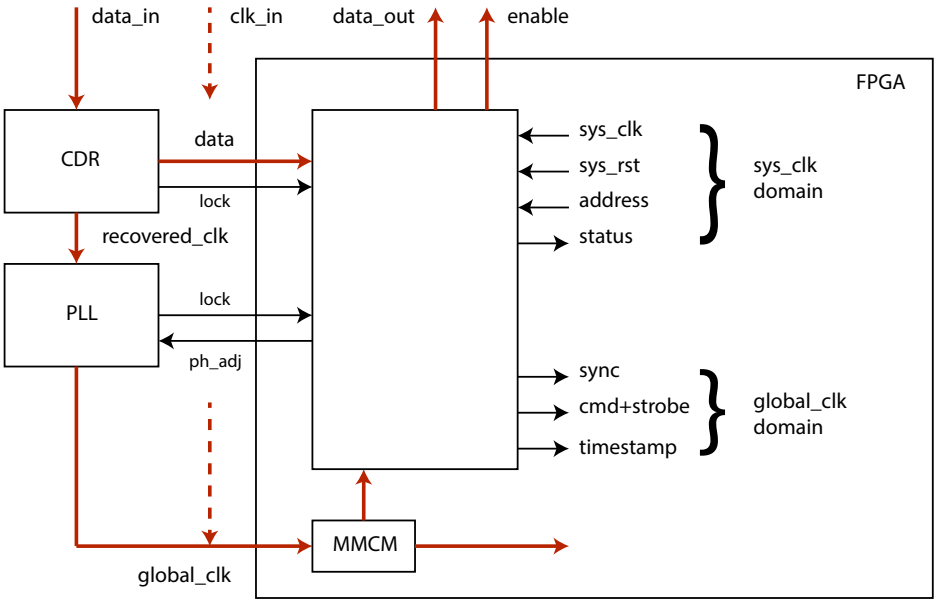

Figure 4: Endpoint signal paths

4.2 Endpoint Firmware Interface

The decoder block exposes three sets of signals:

4.2.1 sys_clk domain signals

- **sys_clk** (o) is the free-running local clock provided by the host FPGA
- **sys_rst** (i) is a startup synchronous reset signal
- **address** (i) is a 32b word specifying: the address identifying this endpoint (see above for addressing scheme); and the timing group mask. It should not change once sys_rst is released.
- **status** (o, 4b) is a set of output signals indicating the internal state of the decoder, including: whether the core is in contact with the timing master; is in one of several possible setup or error conditions; is providing a stable global clock. The signal may be used to drive a reset of all logic governed by the global clock.

FUTURE_UPDATE Write state transition diagram for the decoder.

4.2.2 Global_clk domain signals

- **global_clk** (i) is the phase-adjusted 50MHz system clock from the PLL
- **sync** (o, 16b) is a set of bits corresponding to decoded synchronous command content. These bits go high for one clock cycle on receipt of the corresponding synchronous command.
- **cmd** (o, 8b) is a byte-wide output carrying a copy of received packet data addressed to this endpoint.
- **strobe** (o) is a signal qualifying the validity of the cmd output. It stays high for an entire packet of data, then goes low at the end of the packet.
- **timestamp** (o, 64b) is the 64b system cycle counter, cycling at global clock frequency.

4.2.3 Other signals

- **data** (i) is the recovered or received serial timing data stream. This signal is automatically registered on the rising or falling edge of global_clk in order to cope with arbitrary phase alignment.
- **cdr_lock** (i) is the lock status signal of the CDR chip
- **pll_lock** (i) is the lock status of the PLL
- **pll_adj** (i, 8b) is a word indicating the desired phase adjustment for global_clock. The physical interface to the PLL is implemented by the host FPGA, since this may require communication via a range of means to the PLL depending on implementation.

5 System Level Operations

The steps required to accomplish basic system level operations are listed below in simplified form. These steps are driven by software.

FUTURE_UPDATE Add calibration sequences as required.

5.1 System Startup

This procedure is carried out at system startup.

1. Reset and initialise timing master unit
2. Perform master self-test via timing path loopback
3. Enable command transmission

5.2 Phase Adjustment

This procedure is carried out after endpoints are powered and reset, but before final endpoint initialisation. It is assumed that a 'running' status from the timing decoder will be a necessary step in individual endpoint initialisation sequences. For each endpoint in turn, phase adjustment is carried out as follows:

1. Enable endpoint transmission
2. Wait for CDR lock and measure clock phase
3. Request status packet
4. Measure whole-cycle offset to establish path delay to / from endpoint
5. Issue delay / phase adjustment command
6. Verify correct alignment
7. Disable endpoint transmission

This process is estimated to require around 10ms per endpoint (dominated by the time required for CDR lock and clock phase measurement), allowing ProtoDUNE-SP to be brought up in a few seconds.

FUTURE_UPDATE Document measured phase adjustment time as measured in hardware.

5.3 Operations Startup

This procedure is carried out after all endpoints are aligned, and after time-of-day is established via GPS.

1. Reset master cycle counter
2. Enable command and input logging
3. Enable regular S_Sync commands
4. Enable trigger commands

5.4 System Monitoring

System monitoring is continually carried out by continuous repetition of the endpoint phase check sequence, and logging of any error counts reported by endpoints. This activity has lower priority than other system activities.

5.5 Operations Shutdown

This procedure is carried out after all DAQ queues have drained, as one of the last end-of-run steps.

1. Disable trigger commands
2. Wait until next S_Sync command
3. Disable regular S_Sync commands
4. Issue S_Reset
5. Disable command and input logging

6 ProtoDUNE-SP Interfaces

6.1 Clock source interface

The master unit will receive a 10MHz clock signal and IRIG time-of-day data from a local White Rabbit timing distribution unit, which is responsible for forwarding signals from the SPS control room. This approach ensures synchronization with the ProtoDUNE beam instrumentation, which has a separate timing system, but based on the same clock.

6.2 Trigger interface

The trigger processor will be a client of the timing system, in that it will receive clock and command data via the (LVDS version of) the standard timing interface. Unlike other endpoints, however, the trigger processor is permitted to send synchronous commands on the return path, which carry trigger information. This mechanism allows the transmission of up to 14 different trigger types.

6.3 WIB interface

The WIB crate PTC will receive an optical signal from the timing system, convert this to a serial electrical signal, and distribute to the WIBs themselves via the backplane. At protocol level, each WIB constitutes a timing system endpoint. The WIB will contain a stand-alone jitter-reducing PLL for clock forwarding to the TPC front end electronics.

6.4 COB interface

The COB will receive fibre from the timing system, via the RTM. The RTM will distribute timing signals to the DTM, which will constitute the timing endpoint. Internal clock and timing signals will then be distributed to the RCEs. The COB will implement a PLL internally to the DTM FPGA.

6.5 FELIX interface

There will be no hardware interface to FELIX. The timing system control software will send trigger messages to FELIX over the network, including timestamp information. It will also be necessary to send any signal that results in a reset of timestamp counters.

6.6 SSP interface

The SSP will receive the three-signal timing interface (clk, din, dout) via an optical-LVDS converter module.

6.7 CRT interface

The muon veto readout system will receive a a clock and simple synchronisation pulse from the timing system, generated by a timing FMC acting as a self-contained endpoint.

References

[1] V. Boudry, F. Magniette, and D. Cussans. “Publication of Specification Documents for the Common DAQ”. In: (2014). URL: https://cds.cern.ch/record/1666866.