GBT PROJECT

GBTX MANUAL

V0.18
DRAFT

20/07/2021

GBT project homepage: http://cern.ch/proj-gbt
Technical contact: GBTX-support@cern.ch

Edited by: P. Moreira, J. Christiansen, K. Wyllie

Document History

Version 0.18

Note on the Reliability and radiation hardness of the E-Fuses (section 15.3)

Note added on the I2C read command 5.2

IC channel read and write-read sequence updated on section 5.1

Version 0.17

Section 3.9.3.1, from Table 36 to Table 42, fixed wrong register addresses for phase-aligners of G4 and G5.

Version 0.16

Section 10.2 reseting the ASIC.

Section 16.7.1 added contining "Must do information," about the JTAG configuration pins.

Version 0.15

Section 3.4 added

Version 0.14

The following sections were reviwed: 5.2.1, 7.2.5 and 9.3.1.1

Version 0.13

Clarification on the RXLOCKMODE of the CDR in Chapter 9.3.1. refPLL mode should not be used.

Version 0.12

Chapter 3 on eLink operation expanded.

Version 0.11

Chapter 21 on power consumption added.

Version 0.10

Value of *REFCLKSELECT* corrected to be 1'b1 in page 68

Version 0.9

Details added on configuring the EC channel ePort (Chapter 8).

PS (TTC) PLL-enable configuration bits clarified in Section 12.3.

Version 0.8

Details added on watchdog operation in SEU environment (Sections 10.4, 10.7)

Version 0.7

Clarified internal pull-ups/downs on I2C-address and JTAG inputs.

Section 10.1 added 'Recommendations for powering sequence'

Version 0.6:

Section 13.4 added

Version 0.5:

Section 3.7 added

Version 0.3:

Sections 9.4 and 9.5 and chapters 11 and 12 expanded

Version 0.1 DRAFT: first draft.

GBTX specifications reviewing procedure

A GBTX specification is defined and discussed within the internal "GBTX specifications group". The proposals are then discussed within the "optical link specifications group" with LHC experiment representatives (The electronics coordinators plus possible specific "clients").

The GBTX specification are now frozen and the chip has entered the production stage.

GBTX specifications

Paulo Moreira

GBTX specifications group

Jorgen Christiansen
Alessandro Marchioro
Paulo Moreira
Jan Troska
Francois Vasey

Optical link specifications group

Jorgen Christiansen
Philippe Farthouat (ATLAS)
Magnus Hansen (CMS)
Alessandro Marchioro
Paulo Moreira
Jan Troska
Francois Vasey
Ken Wyllie (LHCb)

GBTX design team

Sandro Bonacini
Rui Oliveira Francisco
Ping Gui
Kostas Kloukinas
Alessandro Marchioro
Paulo Moreira
Filip Tavernier
Ken Wyllie

GBT-FPGA development team

Sophie Baron
Sebastian Feger
Pedro Leitao
Manoel Marin

GBTX test team

Sophie Baron
Sebastian Feger
Tullio Grassi
Pedro Leitão

David Porret
José da Silva

Note:

This is a working document and is therefore neither final nor complete. It is made available to potential GBT link users to provide early information and to allow them to provide feedback to the GBT design and development teams.

Table Of contents

1. INTRODUCTION

The GBTX is a radiation tolerant chip that can be used to implement multipurpose high speed (3.2-4.48 Gb/s user bandwidth) bidirectional optical links for high-energy physics experiments.

Logically the link provides three "distinct" data paths for Timing and Trigger Control (TTC), Data Acquisition (DAQ) and Slow Control (SC) information. In practice, the three logical paths do not need to be physically separated and are merged on a single optical link as indicated in Figure 1. The aim of such architecture is to allow a single bidirectional link to be used simultaneously for data readout, trigger data, timing control distribution, and experiment slow control and monitoring. This link establishes a point-to-point, optical, bidirectional (two fibres), constant latency connection that can function with very high reliability in the harsh radiation environment typical of high energy physics experiments at LHC.

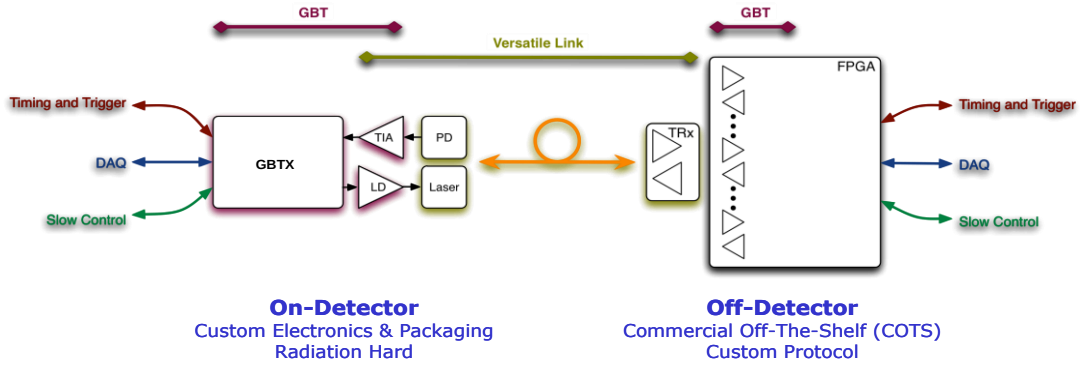


Figure 1: Link architecture with the GBT chip set and the Versatile Link opto-components.

The development of the proposed link is conceptually divided into two distinct but complementary parts: the GBT link chips and the Versatile link opto components. The versatile link selects and qualifies appropriate fibres and opto-electronic components for use in radiation. The GBT develops and qualifies the required radiation hard ASICs.

The link is implemented by a combination of custom developed and Commercial-Off-The-Shelf (COTS) components. In the counting room the receiver and transmitters are implemented using COTS components and FPGA's. Embedded in the experiments, the receivers and transmitters are implemented by the GBT chipset and the Versatile Link optoelectronic components. This architecture clearly distinguishes between the counting room and front-end electronics because of the very different radiation environments. The on-detector front-end electronics works in a hostile radiation environment requiring custom made components. The counting room components operate in a radiation free environment and can be implemented by COTS components. The use of COTS components in the counting house allows this part of the link to take full advantage of the latest commercial technologies and components (e.g. FPGA's with many link interfaces [4], [5], [6], [7]) enabling efficient data concentration and data processing from many front-end sources to be implemented in very compact and cost efficient trigger and DAQ interface systems.

The GBTX ASIC is part of the GBT chipset composed of the following chips: a Trans-Impedance Amplifier for the optical receiver (GBTIA) [1], a Laser Driver (GBLD) [2], a Slow Control Adapter ASIC (GBT-SCA) [3] and the GBTX link ASIC that implements all the needed functions of the data and timing transceiver.

The GBTX is a highly flexible link interface chip with a large number of programmable options to enable its efficient use in a large variety of front-end applications:

- Can be configured to be a bidirectional transceiver, a unidirectional transmitter or a unidirectional receiver.
- Different front-end interface modes and options.
- Extensive features for precise timing control.
- Extensive Control and monitoring features.
- Very high level of error correction from SEU's and transmission errors.

1.1 Radiation environment

Due to the very high beam luminosity planned for the LHC machine upgrade, the radiation levels for the innermost layers of vertex detectors in the LHC experiments may exceed 100 Mrad, 10^{16} n/cm^2 and $10^{?}$ High-energy hadrons/cm^2 over the ~10 year lifetime of the experiments. These extremely high levels of radiation pose significant challenges for the electronics and optoelectronics components installed in the detectors, due to total dose, Non Ionizing Energy Loss (NIEL) and Single-Event Upsets (SEU). Total dose and NIEL effects are mitigated in the GBT chipset using an extensively radiation qualified commercial 130 nm CMOS technology following special layout techniques. SEU are a major impairment to error free data transmission in HEP applications. The GBTX uses a particular robust line coding and error correction scheme, capable of correction single bit and bursts errors caused by SEU's and transmission errors. In its standard mode of operation ~30% of the optical link bandwidth is assigned to the transmission of a Forward Error Correction (FEC) code. The GBT chips also use dedicated design methodologies to resolve SEUs in internal logic and registers.

1.2 GBTX Architecture

The general architecture of the GBTX ASIC and its main external connections are displayed in Figure 2. In its generic configuration the GBTX connects to the GBLD laser driver ASIC and to the GBTIA trans-impedance amplifier ASIC.

The Clock and Data Recovery (CDR) circuit receives high speed serial data from the GBTIA. It recovers and generates an appropriate high speed clock to correctly sample the incoming data stream. The serial data is then de-serialized (that is converted in parallel form) and then DECoded, with appropriate error corrections, and finally DeSCRambled (DSCR). This will sometimes be referred to as the 'downlink'.

In the transmitter part the data to be transmitted is SCRambled (SCR), to obtain DC balance, and then encoded with a Forward Error Correction (FEC) code before being serialized and sent to the GBLD laser driver. This will sometimes be referred to as the 'uplink'. The configuration of the GBLD can be performed via a simplified "I2C-Light" connection from the GBTX.

A clock manager circuit takes care of generating and manage the different high speed and low speed clocks needed in the different parts of the GBTX. A programmable Phase Shifter is available to generate 8 external user clocks with programmable frequency and phase. An external clock or an on-package crystal oscillator is used during start-up as a locking aid for the CDR circuit and as a clock reference for the ASIC watchdog circuit.

General control and monitoring logic takes care of controlling the different parts of the chip according to the operation mode selected and the ASIC configuration information. Initial configuration information is taken from the on chip e-Fuses that can then be modified via the optical link itself or via an I2C slave interface. A JTAG interface is available for boundary scan.

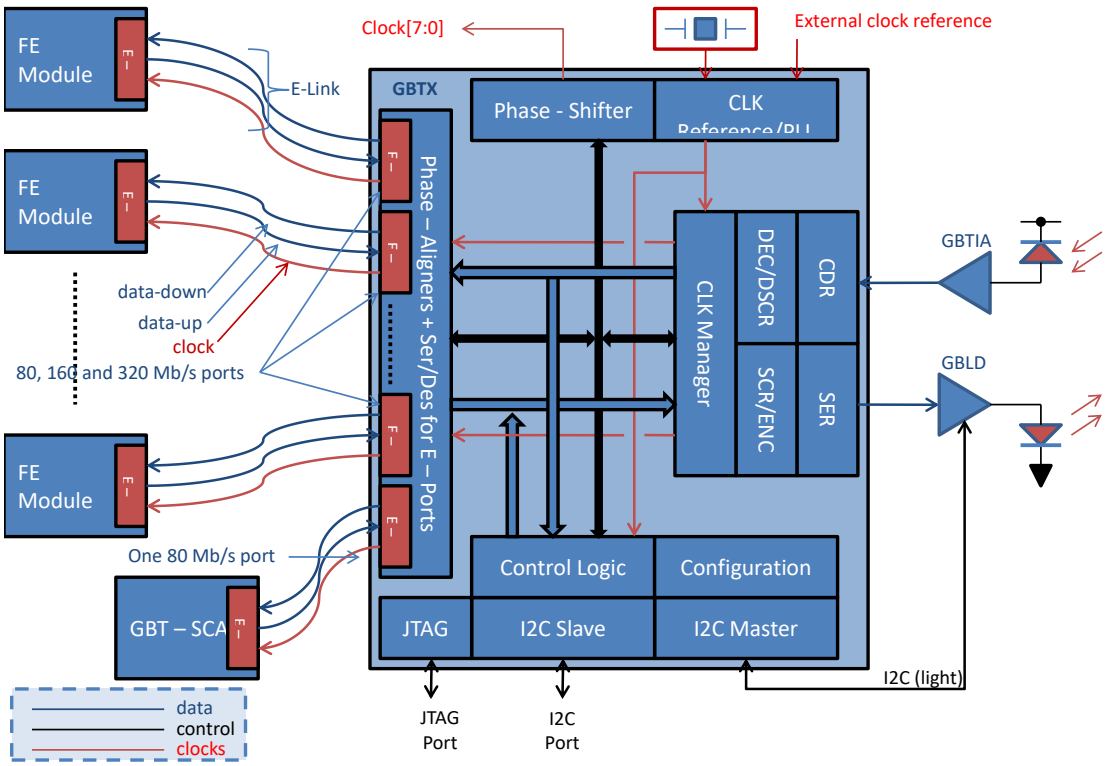


Figure 2 GBTX architecture and interfaces.

Connections to the front-end modules or ASICs are made through sets of local Electrical Links (E-Links). Depending on the data rate and transmission media used, E-Links allow connections that can extend up to a few meters. E-Links use Low-Voltage Differential Signalling, with signal amplitudes that are programmable to suit different requirements in terms of transmission distances, bit rate and power consumption (see chapter 13 for further details). The E-Links are driven by a series of ePorts on the GBTX and are associated with E-link ports in the front-end modules. The number of active E-Links and their data rate are programmable (see chapter 3 for further details). Parallel front-end interfaces with different bit widths are valid sub-sets of the flexible E-Links.

1.3 Transceiver modes

The GBTX supports both bidirectional and unidirectional data transmission. This imposes particular constraints on how the link can be configured and initialized at start-up. In all cases the GBTX will be capable of establishing a working link connection by "itself". To assure that this can be accomplished the basic transceiver modes are selected via dedicated configuration pins that must be hardwired in the user application according to the use of the GBTX. Further default configuration of the GBTX is loaded from its internal E-Fuse Bank (see chapter 15) at power-up. The final configuration can, after basic link initialization, be modified either through the GBT link itself (if the ASIC is configured as a bidirectional link) or through the I2C configuration interface. In all modes the GBTX needs a local clock reference (from the on-package quartz crystal (XTAL) or from an external reference) to correctly initialize itself and the optical link.

1.3.1 Simplex transmitter

In this mode the GBTX works as a simple link transmitter receiving the data to be transmitted from the front-end modules through the E-links. The system and link clock reference must be driven to the GBTX from an external clock source and the front-end modules must transmit data to the GBTX synchronously with this clock

reference. This can be achieved by either clocking the front-ends by one of the GBTX programmable phase clock outputs or by the E-link clock (derived from the GBTX system clock). Detailed configuration of the GBTX must be done via the I2C configuration interface. The link receiver part is fully powered down and E-link lines not used can be powered down via the configuration to minimize power consumption.

When operating as a simplex transmitter the E-Link ports operate as receivers only. In this case the E-Link clocks are derived from the transmitter section of the GBTX.

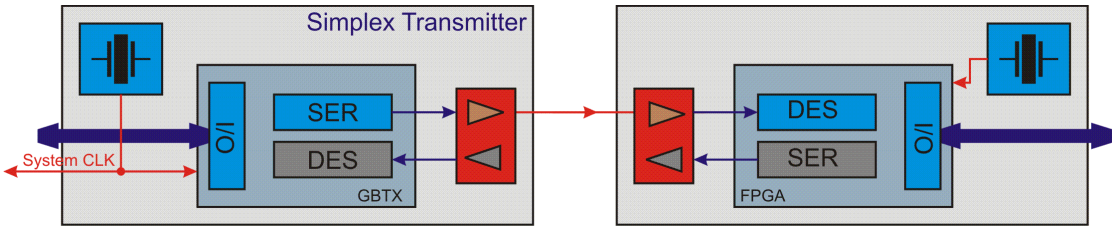


Figure 3 GBTX simplex transmitter

1.3.2 Simplex receiver

In this mode the GBTX works as a simple link receiver, receiving data and the clock reference from the counting room through the optical fibre channel. The received data and clock are fed to the front-end modules through the E-links. Detailed configuration of the GBTX must be done at the start-up from the E-Fuses and can later be modified via the I2C configuration interface. The link transmitter part is fully powered down and E-link lines not used can be powered down via the configuration to minimize power consumption.

In the simplex receiver mode, the E-Link ports operate as transmitters only. In this case the link clocks are derived from the receiver section of the GBTX.

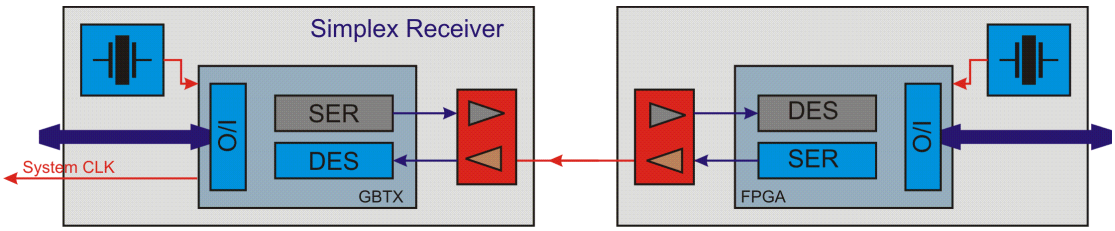


Figure 4 GBTX simplex receiver.

1.3.3 Transceiver

In this mode the GBTX works as a full link transceiver with bidirectional data communication with the front-ends and the counting room. The GBTX delivers the global system clock reference, coming from the counting room, to all front-ends. In this mode the detailed configuration (and monitoring) can be performed via the IC field (see paragraph 2.2.2.1) of the optical link itself or via the I2C configuration interface.

In this mode, the ePorts operate as transceivers with the ePort receivers feeding data to the serializer and the ePort transmitters receiving data from the CDR circuit. In this case the ePort clocks are derived from the receiver section of the GBTX.

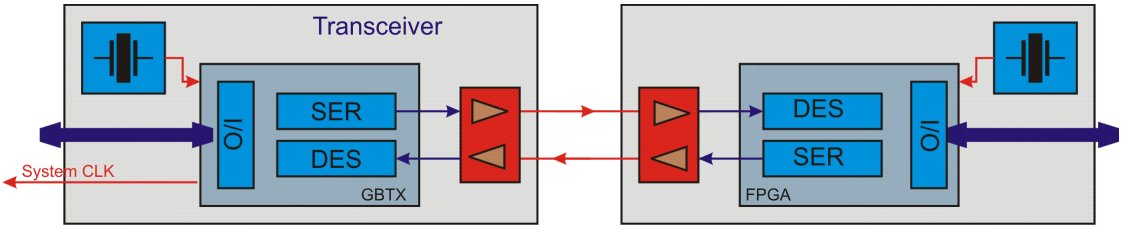


Figure 5 GBTX bidirectional transceiver.

1.4 Setting up the GBTX – Important note

As is discussed through the text the GBT configuration is setup through a set of pins and a relatively large number of registers (see chapter 17 for a complete list of the GBTX registers). To achieve robustness against Single Event Upsets (SEU) many of the GBTX circuits use Triple Modular Redundancy (TMR). This applies also to a large fraction of the configuration registers. As such, please note that for correct operation of the TMR logic the three instances of the same register ("registerNameA", "registerNameB" and "registerNameC") must be programmed with the same data.

2. LINK FRAME AND ENCODING

The optical link will simultaneously carry readout data, trigger data, timing information, trigger and control signals and experiment-control data that must be transferred with very high reliability.

Single Event Upset (SEU) rates will be a major impairment to error free data transmission at high data rates in harsh radiation environments. The GBT adopts a robust line coding and correction scheme that can correct bursts of bit errors caused by SEU's.

2.1 Error correction

A typical approach to overcome SEU's in logic circuits is to use triple-modular redundancy [8] or Hamming encoding [9] to obtain some degree of error correction capability. In addition to errors generated in the internal logic of the ASIC, particles in HEP experiments can deposit sufficient energy in the photodiode receiver thereby corrupting data bits and generate phase errors. The use of triple-redundancy techniques would impose a significant speed penalty on the SERializer and DESerializer (SERDES) circuits not compatible with the required data rate. The approach chosen in the GBT link is to transmit data with a Forward Error Correction code (FEC), generated before serialization and decoded/corrected after de-serialization. Any transmission errors or single event upsets occurring in the Serializer (SER), GBLD, PIN-diode, GBTIA, Clock and Data Recovery (CDR) and De-serializer (DES) will be corrected in the decoding operation. The forward error correction coding has been particular chosen to provide a high level of error protection that can also deal with bursts of errors ([10], [11]). A double interleaved Reed-Solomon two-errors correcting code was chosen. The code is built by interleaving two Reed-Solomon RS(15,11) [12] encoded words with 4-bit symbols, each capable of correcting a double symbol error. This in practice means that a sequence of up to 16 consecutive incorrectly-received bits can be corrected. For the data payload offered by the GBTX (88-bits) an extra FEC field of 32 bits is required in the frame, resulting in a code efficiency of 70%. This must be considered very efficient for a code that obtains this high level of immunity to single bit and long bursts of errors. The chosen FEC in combination with a data scrambler also provides the required DC-balancing of the serial data stream ([13], [14]). A standard 8B/10B line code, providing no error correction functionality, has in comparison a coding "efficiency" of 80%.

The GBTX implements three different frame formats for data transmission. The default (and highly recommended) "GBT frame format" uses the forward error correction scheme as described above. An alternative frame format is available for applications where the absolute maximum data rate is required on the uplink and where error correction is not needed in this direction (the sensitive optical receiver in this case is located outside radiation areas). A frame format with 8B/10B encoding is also available for the uplink as requested by particular users.

2.2 GBT frame format

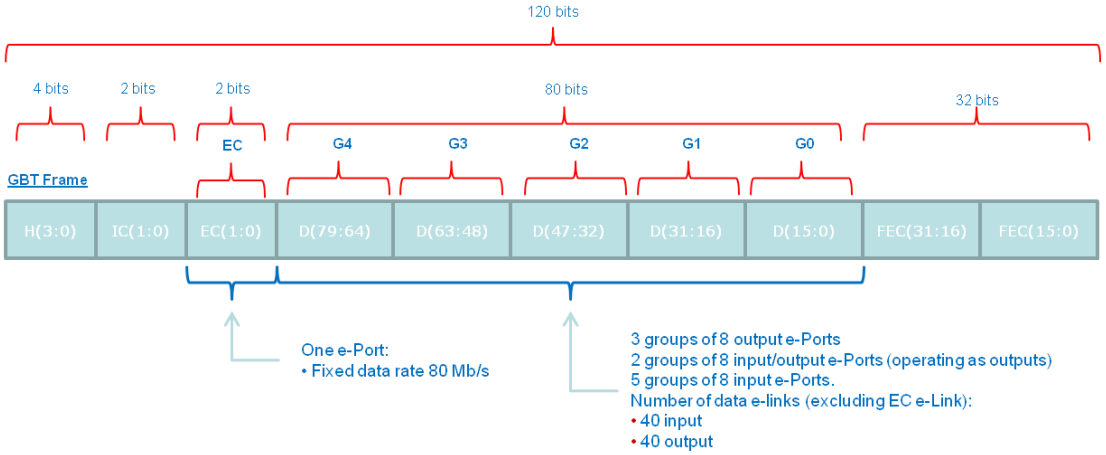


Figure 6 GBT frame structure

The 120-bit "GBT frame format", sketched in Figure 6, is transmitted during a single LHC bunch crossing interval (25 ns), resulting in a line rate of 4.8 Gb/s. Four bits are used for the frame Header (H) and 32 are used for Forward Error Correction (FEC). This leaves a total of 84 bits for data transmission corresponding to a user bandwidth of 3.36 Gb/s. Of the 84-bits, 4 are always reserved for Slow Control information (Internal Control (IC) and External Control (EC) fields), leaving 80-bits for user Data (D) transmission. The 'D' and EC fields use is not pre-assigned and can be used indistinguishably for Data Acquisition (DAQ), Timing Trigger & Control (TTC) and Experiment Control (EC) applications.

DC-balance of the data being transmitted over the optical fibre is ensured by scrambling the data contained in the SC and D fields. For forward error correction the scrambled data and the header are Reed-Solomon encoded before serialization. The 4-bit frame header is chosen to be DC balanced. The line encoding/decoding process is represented in Figure 7.

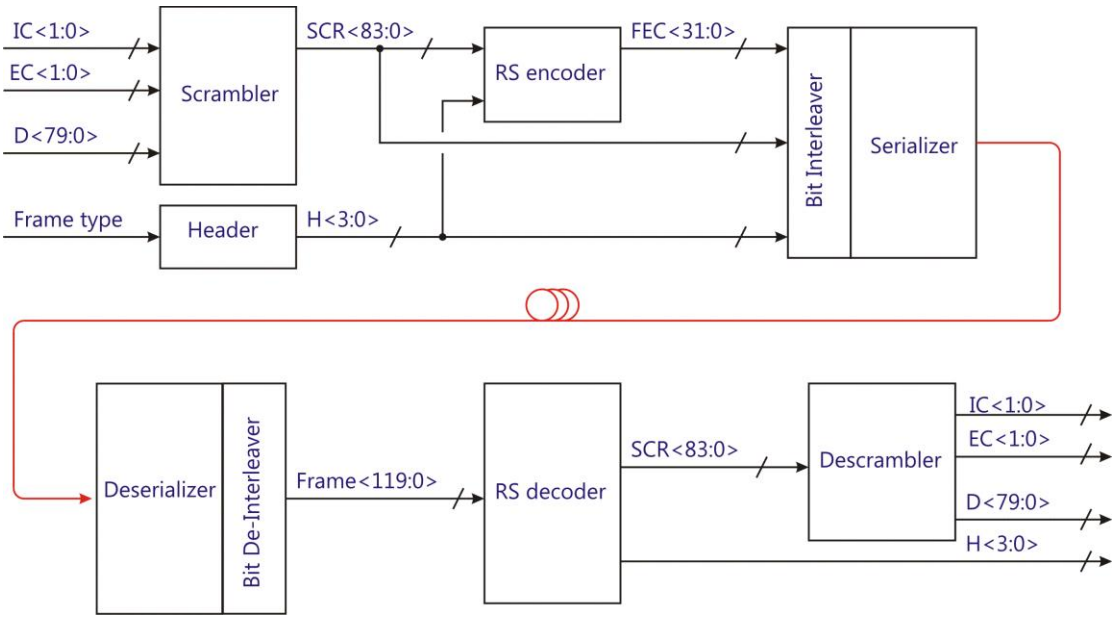


Figure 7 GBT encoding/decoding block diagram.

As shown in the figure above, 84-bits (80 bits data, 2 bits IC and 2 bits EC) are first processed by the scrambler, the header is then added (4 bits), the Reed-

Solomon (RS) encoding and interleaving takes place and finally the data is serialized. While the scrambler maintains the word size, the RS encoder adds the 32-bit Forward Error Correction (FEC) field adding up to a total frame length of 120 bits. This leads to an overall line code efficiency of 84/120 = 70%. At the receiver end the inverse operations are repeated in the reverse order. The fact that RS encoding and decoding are the first and the last operations to be done respectively at the transmitter and receiver (before transmission and after reception) ensures that transmission errors do not get multiplied by the scrambler operation ([13], [14]).

2.2.1 Header field (H)

A 4-bit header field is transmitted at the beginning of each frame. The header field is required to synchronize the data stream at the frame level. Repeated recognition of a valid header indicates correct frame-locking. Repeated non-valid header recognition indicates that frame synchronization has been lost and that a frame synchronization cycle has to be initiated. To avoid transmission errors due to false identification of the header, the header is also protected by the forward error correction. To enable straight and reliable header recognition it is transmitted unscrambled and valid header codes are chosen to be DC-balanced as shown in Table 1.

Header	Code
Idle	**0110**
Data	**0101**

Table 1 Header codes.

The use of Idle versus Data frames is fully transparent in the GBTX. The GBTX does not use this idle/data information for its internal function (neither for its initial link synchronization). It can be freely used at the system level (e.g. for system synchronization). The header type of a received frame is indicated by a data-valid output (rxDataValid) signal, which is high if the header is 'data' (0101). In a transmitted frame, the type of header is selected by a data-valid input signal (txDataValid).

2.2.2 Slow Control field (SC)

A 4-bit slow control field is dedicated to the execution of routine and control operations that do not necessarily require precise timing. Two bits are reserved for the Internal Control (IC) of the GBTX itself while the other two implement a fixed bandwidth port for an external slow control (EC) port (e.g. for the GBT-SCA chip). The SC field is fully protected by the forward error correction.

2.2.2.1 Internal Control field (IC)

The 2-bit IC control field is used to control and monitor the GBTX operation. It implements an 80 Mb/s communication link with the GBTX ASIC. Its use is strictly reserved for the GBTX control.

2.2.2.2 External Control field (EC)

The 2-bit External Control (EC) field has an associated bandwidth of 80 Mb/s and it is part of the slow control channel. Although intended to implement a slow control channel (e.g. for the GBT-SCA), its use is not restricted to this application and can be used for generic data transmission applications.

As indicated in Figure 6 the EC field interfaces with frontend electronics through a dedicated ePort that operates at 80 Mb/s (see chapter 3 for more information).

2.2.3 Data field (Data)

The 80bit data field is used for generic transmission of data, having an associated bandwidth of 3.2 Gb/s. The data field is fully available to the user via the flexible E-links and is fully protected by the FEC. Data transmitted has fixed latency in both directions enabling its efficient use for trigger information and timing control.

Figure 6 shows how the data field of the GBT frame is associated with the 5 ePort bi-directional (input/output) groups (see chapter 3 for more information).

2.2.4 Forward Error Correction field (FEC)

The 32-bit FEC field is used to protect all the other fields in the frame against transmission errors due to link noise and single event upsets. The forward error correction algorithm used is a 2 times interleaved Reed-Salomon RS(15,11) code with 4-bit symbols as described in paragraph 2.1. The code is built by interleaving two Reed-Solomon [12] encoded words with 4-bit symbols, each capable of correcting a double symbol error. This in practice means that a sequence of up to 16 consecutive corrupted bits can be corrected. This correction technique requires a FEC field of 32 bits in the frame of 120-bits.

In the 130 nm CMOS technology used for the GBTX both the encoding and decoding operations can be done within a single machine clock cycle ([25 ns) thus having a minimal impact on the transmission latency.

A detailed account on the Reed-Salomon coding is given in chapter 0.

2.3 Scrambling

DC-balancing of the transmitted data is achieved by using a self-synchronizing scrambler that "randomizes" the data pattern and guarantees a proper distribution of 0's and 1's in the data stream [13], [14]. The scrambling function is implemented before the Reed-Solomon encoding (and the de-scrambling after the decoding).

For details on the GBT scrambling principle please refer to chapter 0.

2.4 8B/10B frame mode

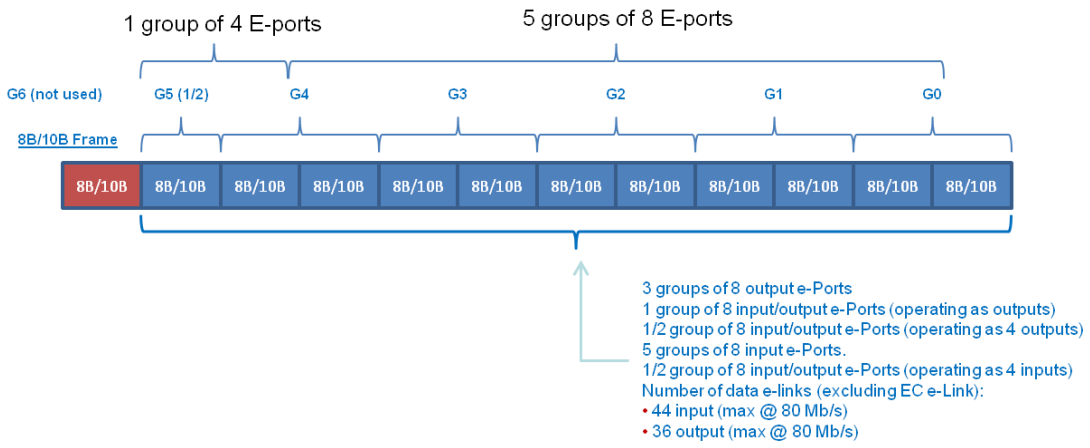


Figure 8 8B/10B transmitter frame format

An 8B/10B encoding option is available to readout data from the front-end modules to the counting room. This mode is therefore only available in the transmitter part of the GBTX. The 8B/10B frame format is shown in Figure 8. No error correction and only very limited error detection capability is possible with this format. As the 8B/10B coding is DC balanced no data scrambling is used in this mode. The 120-bit link frame consists of 12 8B/10B words of which the first is always a comma

character required for the receiver synchronization. The effective number of user bits in this mode are 11 x 8 = 88 giving a coding "efficiency" of 88/120 = 73% which is only marginally "better" than the standard GBT frame, at the cost of no error correction and limited error detection capability. The main justification to use this mode is to reduce resources needed in the counting room FPGA's to implement the normal GBT forward error correction.

The 80 LSB bits of the 88 bits uplink data are available on the E-link as the normal user dataFieldIn[79:0]. The additional 8 bits dataFieldIn[87:80] are made available on pins normally used for E-link group 5 outputs, implying that in this mode fewer output data bits will be available on the E-links. As Figure 8 shows, in this mode 5 ½ groups are used as inputs and 4 ½ groups are used as outputs. This results in a maximum (at 80 Mb/s) of 44 input E-Links and 36 output E-links.

2.5 Wide frame mode

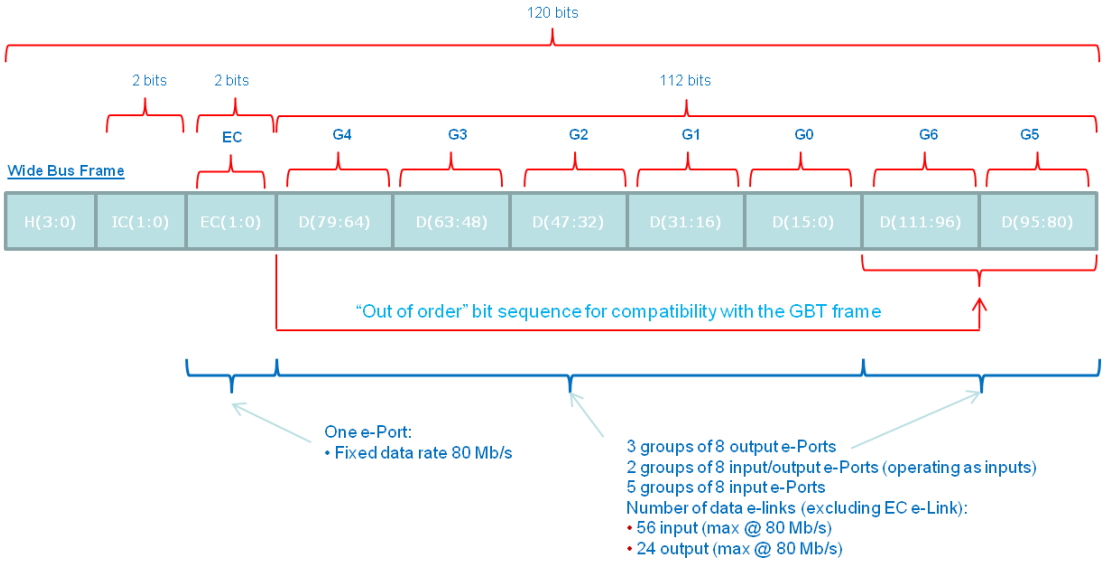


Figure 9 Wide frame format on uplink.

A "wide frame" mode format with only scrambling is available for the transmitter direction. It provides an alternative for data transmission where the forward error correction functionality is traded off for bandwidth. That is, in the wide frame mode the FEC field is not present (see Figure 9) and the space taken by the FEC code in the GBT frame is used to transmit data. As a consequence, the data field increases to 112 bits resulting in a total user bandwidth of 4.48 Gb/s, representing an increase of (112-80)/80 = 40% of available bandwidth when compared with the GBT frame format. However this is done at the cost of having no SEU error protection on the transmitted data. Frame bits D[111:80] are scrambled separately to maintain DC balance of transmitted data (see chapter 3 for more information on data scrambling).

When the wide frame format is used in transceiver mode the down link maintains its use of the standard GBT frame format, with extensive error detection and correction. The same applies for the 8B/10B mode.

Both for the down and uplinks the IC and EC fields are maintained, enabling slow control functions to be used. However in the case of the uplink no error detection or correction is implemented.

The 80 LSB bits of the 112-bit user data are taken from the E-Links from the normal data inputs dIn[79:0]. The additional 32 bits dIn[111:80] are made available on pins normally used as outputs in groups 3 and 4, implying that in this mode when

operating the E-Links at 80 Mb/s fewer output data bits will be available on the downlink E-links (See chapter 3 for further details).

2.6 Frame detection

A frame header is used to delimit the frame boundaries (frame synchronization) and to indicate the type of frame being received. In the presence of transmission errors, due to noise or single event upsets, a robust algorithm must be used by the receiver in order to maintain reliable communication between the GBTX and counting-room receiver. To assure robust frame synchronization two operation modes are used: frame-lock acquisition and frame-tracking. During frame lock the receiver should acquire quickly frame lock in as short time as possible to minimize the dead time in case of a loss of lock during normal operation (although the process is intrinsically relatively slow). However, during the tracking phase the receiver must avoid restarting a locking cycle unless multiple frame errors are observed in a relatively short period (please see section 2.6 for information on the GBTX frame locking and tracking mechanism).

Since in the wide bus and 8B/10B modes the uplink header is not FEC protected it is possible that the frame type (idle/data) is badly interpreted. It is thus not recommended to rely on the header type information in this mode. That is, the header should only be used for frame synchronization.

2.6.1 Frame-lock acquisition (down link)

At power on or after a loss of synchronization (frame or bit synchronization) the GBTX receiver will start a frame-lock acquisition cycle to find the frame boundaries and acquire frame synchronization.

The frame-lock acquisition mode operates as follows: For each received frame the four bits in the header position are checked for header validity (both data and idle frame headers are considered valid). If N_{HF} consecutive frames contain a valid header the frame is considered locked. Otherwise, the frame is shifted by one bit and the valid header checking procedure is repeated. After frame-lock is achieved, the receiver switches to the frame-tracking mode.

Although the header is FEC protected, the frame-locking state machine only uses the header bits information. In what concerns the frame header, the FEC information is only used (after the frame lock has been achieved) to protect the data-frame/idle-frame information contained in the header.

2.6.2 Frame-tracking

The aim of the frame-tracking mode is to maintain frame synchronization even in the presence of headers corrupted by noise or single event upsets. The phase tracking mode must thus be tolerant to a low rate of detection of invalid headers. Provided that frame synchronization is maintained, a corrupted header will not introduce a transmission error since the header field is protected by forward error correction and the robust frame locking and frame-tracking algorithm described below.

The frame-tracking mode operates as follows: After a successful frame-lock acquisition cycle has been executed the receiver enters the frame-tracking mode. In this mode the receiver strives to maintain frame synchronization. It checks the validity of the headers and counts the number of invalid headers received in consecutive frames after the first invalid header has been detected. If the number of invalid headers received in consecutive frames is bigger than N_{MIH} then the receiver re-enters the frame-lock acquisition mode.

Since Errors due to single event upsets on the header bits will occur sparsely, it is necessary to avoid that the receiver will enter the frame-lock acquisition mode unnecessarily by accumulation of bit errors on the header bits due to sporadic SEUs

that will accumulate over time. If, after detection of one or more invalid headers (that don't exceed N_{MIH}), a specified minimum number of consecutive valid headers (N_{MVH}) are detected, then the count of invalid errors is reset to zero. The numbers N_{HF}, N_{MIH} and N_{MVH} are programmable and their function is summarized in Table 2.

Table 2 Frame-Lock Parameters

N_{HF}[7:0]	Header Found count	Register t.b.d	Number of consecutive times a header needs to be detected before the frame is considered aligned (during frame-lock acquisition).
N_{MIH}[7:0]	Maximum number of Invalid Headers	Register t.b.d	After frame alignment, this is the maximum number of invalid headers that can be detected before the frame is considered unlock (during frame-tracking).
N_{MVH}[7:0]	Minimum number of Valid Header	Register t.b.d	After frame alignment, if one or more invalid headers are detected, this is the minimum number of consecutive valid headers that must be detected for the frame to be considered locked (during frame-tracking).

3. E-LINKS

The GBTX can be electrically interfaced with the on detector electronics using different topologies. The simplest one consists of interconnecting the GBTX and a front-end device through a parallel lane while the most sophisticated allows the GBTX to interface simultaneously with up to 40 front-end devices via duplex local Electrical serial links (E-links). The use of one single parallel lane, the use of multiple parallel lanes or individual serial connections are valid subsets of the E-link programmable features.

Each E-link normally consists of three signal lines (differential pairs):

- Differential Clock line (dClk+/dClk-): Clock driven by GBTX to front-end module.
- Differential Downlink data output (dOut+/dOut-): Data line from GBTX to the front-end module.
- Differential Uplink data input (dIn+/dIn-): Data line from front-end module to GBTX

In particular cases not all three E-Link signals are used (e.g. if unidirectional link or link clock encoded in serial data)

Figure 10 represents the general interconnection topology between the GBTX chip and the Front End electronics using E-links.

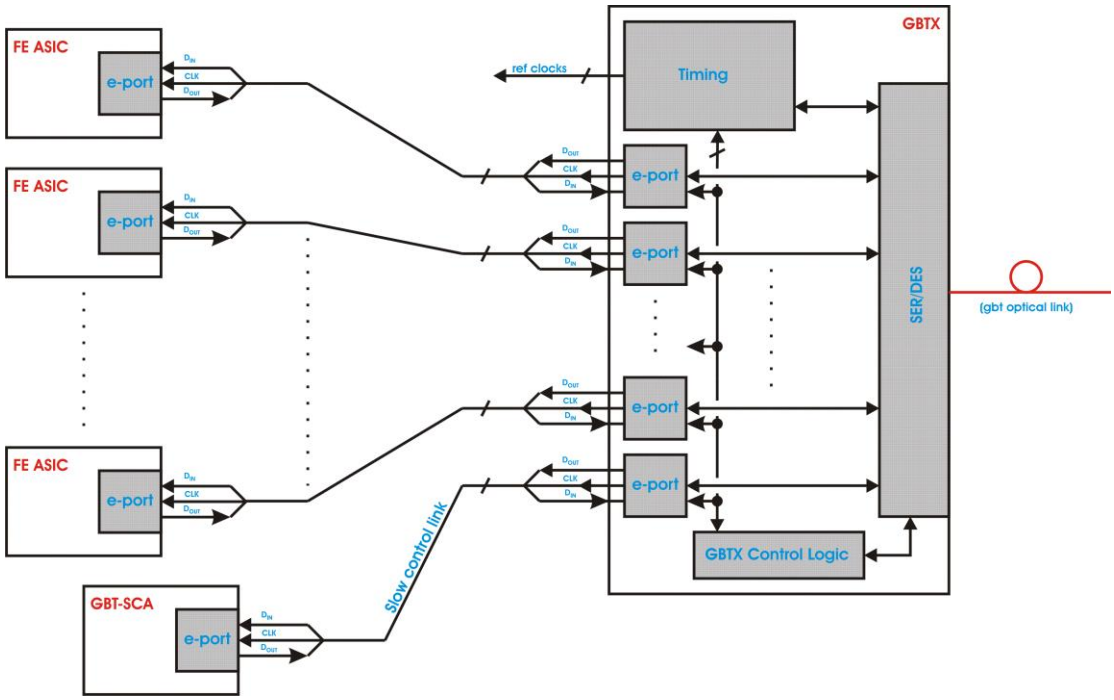


Figure 10 E-link connection topology.

The minimum date rate in the E-Links is 80 Mb/s resulting in a maximum of 8 E-Links per group. Each E-Link has one differential clock line (dClk+/dClk-), a differential downlink output (dOut+/dOut-) and a differential uplink input (dIn+/dIn-) The maximum number of differential E-link signals are thus 8 x 3 = 24 per group corresponding to 48 signal pins per group. Overall in the GBTX 24 x5 = 120 pins are dedicated to E-Links (plus 6 dedicated to the EC link).

The E-Link data rates are programmable in a per group basis and are given in Table 3. Within a group, the input and output lanes can be set to work at different data

rates and an unused group can be turned off to reduce the power consumption in the corresponding internal logic.
The bit shift in/out order for the E-Link data inputs and outputs is MSB first.

Table 3 E-Link data rates

Mode	Type	Data Rate	Notes
OFF	Power off	-	Whole group powered off
2 ×	Serial/parallel	80 Mb/s	8 E-links per group
4 ×	serial	160 Mb/s	4 E-links per group
8 ×	serial	320 Mb/s	2 E-links per group

E-Links are implemented with differential signals following, for nominal settings, the JEDEC standard JESD8-13: "Scalable Low-Voltage Signalling for 400 mV (SLVS-400)" [16]. The E-Link drivers have programmable drive currents allowing the signal amplitude to be adapted either for low power consumption or high drive capability. The physical E-link connections are assumed to be differential transmission lines (or twisted pair cables) with a differential impedance of 100 Ω with appropriate line termination at receiver end to assure the best possible signal quality and transmission reliability. E-Link receivers have built-in 100 Ω terminations that can be disabled if required. E-Link signal drivers and receivers not actively used can be powered down to reduce power consumption (I/O power is a significant part of the total GBTX power consumption). See chapter 13 for more information on E-Link drivers and receivers.

3.1 E-links data rate

E-links can be configured to run at different data rates, use different clocking modes and can finally also be used with multiple data lines in parallel. These are discussed later in this chapter.

3.1.1 Programming the output E-Links data rates

The data rate of each output ePort group (please see next section for the definition of ePort groups) is programmed by the contents of the set of registers given in Table 4 (using the values defined in Table 5) and Table 6. Note that the EC E-Link is limited to 80 Mb/s.

Table 4 Output ePorts data rate programing registers

Register	**Register Name**	**Function (see** Table 5**)**
254 [1:0]	outEportCtr0[1:0]	Mode Group 0 A
257 [1:0]	outEportCtr3[1:0]	Mode Group 1 A
260 [1:0]	outEportCtr6[1:0]	Mode Group 2 A
263 [1:0]	outEportCtr9[1:0]	Mode Group 3 A
266 [1:0]	outEportCtr12[1:0]	Mode Group 4 A
332 [1:0]	outEportCtr20[1:0]	Mode Group 0 B
335 [1:0]	outEportCtr23[1:0]	Mode Group 1 B
338 [1:0]	outEportCtr26[1:0]	Mode Group 2 B

341 [1:0]	outEportCtr29[1:0]	Mode Group 3 B
344 [1:0]	outEportCtr32[1:0]	Mode Group 4 B
347 [1:0]	outEportCtr35[1:0]	Mode Group 0 C
350 [1:0]	outEportCtr38[1:0]	Mode Group 1 C
353 [1:0]	outEportCtr41[1:0]	Mode Group 2 C
356 [1:0]	outEportCtr44[1:0]	Mode Group 3 C
359 [1:0]	outEportCtr47[1:0]	Mode Group 4 C

Table 5 Output ePorts data rates

modeYX **with Y = A, B, C** **X = 0, 1, 2, 3, 4**	**Data rate**
2'b00	disabled
2'b01	80 Mb/s
2'b10	160 Mb/s
2'b11	320 Mb/s

Table 6 Output EC channel data rate

Register 254[6:4] = outEportCtr0[6:4] **=** **{modeEcC, modeEcB, modeEcA}**	**Data rate**
3'b000	Disabled (data & clock pins disabled)
3'b111	80 Mb/s (data & clock pins enabled)
Other combinations not allowed	

3.1.2 Programming the input E-Links data rates

The data rate of the input ePort groups (see next section) is set by the registers given in Table 7 using the values defined in Table 8. The input eports use the phase aligner circuit described in section 3.7.2 to guarantee that the incoming data stream is correctly sampled by the GBTX.

Table 7 Input ePorts data rate programing registers

Register	**Register Name**	**Function (see** Table 8**)**
63 [1:0]	inEportCtr1[1:0]	Phase aligner data rate Group 0 A
63 [3:2]	inEportCtr1[3:2]	Phase aligner data rate Group 0 B
63 [5:4]	inEportCtr1[5:4]	Phase aligner data rate Group 0 C
87 [1:0]	inEportCtr25[1:0]	Phase aligner data rate Group 1 A
87 [3:2]	inEportCtr25[3:2]	Phase aligner data rate Group 1 B
87 [5:4]	inEportCtr25[5:4]	Phase aligner data rate Group 1 C

111 [1:0]	inEportCtr49[1:0]	Phase aligner data rate Group 2 A
111 [3:2]	inEportCtr49[3:2]	Phase aligner data rate Group 2 B
111 [5:4]	inEportCtr49[5:4]	Phase aligner data rate Group 2 C
135 [1:0]	inEportCtr73[1:0]	Phase aligner data rate Group 3 A
135 [3:2]	inEportCtr73[3:2]	Phase aligner data rate Group 3 B
135 [5:4]	inEportCtr73[5:4]	Phase aligner data rate Group 3 C
159 [1:0]	inEportCtr97[1:0]	Phase aligner data rate Group 4 A
159 [3:2]	inEportCtr97[3:2]	Phase aligner data rate Group 4 B
159 [5:4]	inEportCtr97[5:4]	Phase aligner data rate Group 4 C
183 [1:0]	inEportCtr121[1:0]	Phase aligner data rate Group 5 A
183 [3:2]	inEportCtr121[3:2]	Phase aligner data rate Group 5 B
183 [5:4]	inEportCtr121[5:4]	Phase aligner data rate Group 5 C
207 [1:0]	inEportCtr145[1:0]	Phase aligner data rate Group 6 A
207 [3:2]	inEportCtr145[3:2]	Phase aligner data rate Group 6 B
207 [5:4]	inEportCtr145[5:4]	Phase aligner data rate Group 6 C

Table 8 Input ePorts (phase-aligner) data rates.

Phase aligner data rate ABC	
2'b00	Disabled
2'b01	80 Mb/s
2'b10	160 Mb/s
2'b11	320 Mb/s

3.2 E-Link groups

The E-Links are associated in groups where each group is associated with up to 8 E-Links and correspond to 16 bits in the uplink and downlink frames. The mapping between the GBT frame, the groups and the I/O depends on the GBTX operation mode: GBT, 8B/10B and Wide Mode.

3.2.1 GBT Mode

In the GBTX mode when the GBTX is being used as a transceiver, the up and down links are symmetrical with the GBT frame used as described in section 2.2. Table 4 and Table 5 give the mapping between the I/O signals dIn[39:0] and dOut[39:0] and the GBT frames for the up and down links respectively (FRMUP[119:0] and FRMDWN[119:0]). The correspondence between the frame bits and the GBT frame fields can be found in Table 6. Table 7 summarizes the maximum number of useful links in this mode.

Table 9 Uplink I/O to frame mapping in the GBT mode

I/O Data Rate			**Group**	**Frame**
x2	x4	x8		
dIn[7:0]			0	FRMUP[47:32]

	dIn[6,4,2,0]			
		dIn[4,0]		
dIn[15:8]			1	FRMUP[63:48]
	dIn[14,12,10,8]			
		dIn[12,8]		
dIn[23:16]			2	FRMUP[79:64]
	dIn[22,20,18,16]			
		dIn[20,16]		
dIn[31:24]			3	FRMUP[95:80]
	dIn[30,28,26,24]			
		dIn[28,24]		
dIn[39:32]			4	FRMUP[111:96]
	dIn[38,36,34,32]			
		dIn[36,32]		

Table 10 Downlink frame to I/O mapping in the GBT mode

Frame	**Group**	**I/O Data Rate**		
		x2	x4	x8
FRMDWN[47:32]	0	dIO[7:0]		
			dIO[6,4,2,0]	
				dIO[4,0]
FRMDWN[63:48]	1	dIO[15:8]		
			dIO[14,12,10,8]	
				dIO[12,8]
FRMDWN[79:64]	2	dOut[23:16]		
			dOut[22,20,18,16]	
				dOut[20,16]
FRMDWN[95:80]	3	dOut[31:24]		
			dOut[30,28,26,24]	
				dOut[28,24]
FRMDWN[111:96]	4	dOut[39:32]		
			dOut[38,36,34,32]	
				dOut[36,32]

Table 11 Correspondence between the frame bits and the GBT fields

FRMUP[119:116] / FRMDWN[119:116]	H(3:0) (header field)
FRMUP[115:114]./ FRMDWN[115:114]	IC[1:0] (internal control field)
FRMUP[113:112] / FRMDWN[113:112]	EC[1:0] (external control field)
FRMUP[111:32] / FRMDWN[111:32]	Data[79:0] (data field)
FRMUP[31:0] / FRMDWN[31:0]	FEC[31:0] (forward error correction field)

Table 12 Maximum number of Up/Down links in the GBT mode

E-Link Data Rate	80 Mb/s	160 Mb/s	320 Mb/s
Max # of up-links	40	20	10
Max # of down-links	40	20	10

3.2.2 Wide Bus Mode

As explained in section 2.5, in the wide bus mode no FEC encoding is used in the uplink and the bits that in the GBT mode are used to carry the FEC field are used to transmit user data. When using the wide bus mode, if the GBTX is used as a transceiver, the downlink still uses the GBT frame (see section 2.2) but the data payload for the down link is reduced when operating in the x2 mode (not in the x4 and x8 modes). The following two tables summarize the mapping between the I/O signals and the uplink and downlink frames. Restrictions to data transmission on the downlink can also be inferred by the information on Table 9. Notice that, depending on the data rate, E-Links signals "dOut[15:0]" may become input signals for the uplink in the Wide Bus Mode.

Frame bits FRMUP[119:116] are reserved for the frame header which is the same as the GBT frame header (see section 2.2.1).

Table 13 Uplink I/O to frame mapping in the Wide Bus mode

E-Link Data Rate: Up-Link			**Group**	**Frame**
80 Mb/s	160 Mb/s	320 Mb/s		
dIn[7:0]			0	FRMUP[47:32]
	dIn[6,4,2,0]			
		dIn[4,0]		
dIn[15:8]			1	FRMUP[63:48]
	dIn[14,12,10,8]			
		dIn[12,8]		
dIn[23:16]			2	FRMUP[79:64]
	dIn[22,20,18,16]			
		dIn[20,16]		
dIn[31:24]			3	FRMUP[95:80]
	dIn[30,28,26,24]			
		dIn[28,24]		
dIn[39:32]			4	FRMUP[111:96]
	dIn[38,36,34,32]			
		dIn[36,32]		
dIO[7:0]			5	FRMUP[15:0]
	dIO[7,5,3,1]			
		dIO[5,1]		
dIO[15:8]			6	FRMUP[31:16]
	dIO[15,13,11,9]			
		dIO[13,9]		

Table 14 Downlink frame to I/O mapping in the Wide Bus mode

Frame	Group	E-Link Data Rate: Down-Link		
		80 Mb/s	160 Mb/s	320 Mb/s
FRMDWN[47:32]	0	n.a.		
			dIO[6,4,2,0]	
				dIO[4,0]
FRMDWN[63:48]	1	n.a.		
			dIO[14,12,10,8]	
				dIO[12,8]
FRMDWN[79:64]	2	dOut[23:16]		
			dOut[22,20,18,16]	
				dOut[20,16]
FRMDWN[95:80]	3	dOut[31:24]		
			dOut[30,28,26,24]	
				dOut[28,24]
FRMDWN[111:96]	4	dOut[39:32]		
			dOut[38,36,34,32]	
				dOut[36,32]

Table 15 Maximum number of Up/Down Links in the Wide bus mode

E-Link Data Rate	80 Mb/s	160 Mb/s	320 Mb/s
Max # of up-links	56	28	14
Max # of down-links	24	20	10

3.2.3 8B/10B Mode

The uplink frame for the 8B/10B is described in section 2.4. In this mode the uplink uses 8B/10B encoding for data transmission and the uplink frame is composed of 12 8B/10B words. The first word transmitted is a comma character (for frame synchronization) and the remaining 11 words are used to transmit user data. As for the wide bus mode, in the 8B/10B mode the downlink still uses a GBT frame. In this case, depending on the data rate, the E-Link signals dOut[3:0] may become inputs for the uplink and they are thus unavailable for the down link. Table 11 and Table 12 summarize the mapping of the I/Os into the uplink and downlink, respectively.

Table 16 Uplink I/O to frame mapping in the 8B/10B mode

E-Link Data Rate: Up-Link			Group	Frame
80 Mb/s	160 Mb/s	320 Mb/s		
dIn[7:0]			0	FRMUP[19:0]
	dIn[6,4,2,0]			
		dIn[4,0]		

dIn[15:8]			1	FRMUP[39:20]
	dIn[14,12,10,8]			
		dIn[12,8]		
dIn[23:16]			2	FRMUP[59:40]
	dIn[22,20,18,16]			
		dIn[20,16]		
dIn[31:24]			3	FRMUP[79:60]
	dIn[30,28,26,24]			
		dIn[28,24]		
dIn[39:32]			4	FRMUP[99:80]
	dIn[38,36,34,32]			
		dIn[36,32]		
dIO[3:0]			5	FRMUP[109:100]
	dIO[3,1]			
		dIO[1]		

Table 17 Downlink frame to I/O mapping in the 8B/10B mode

Frame	**Group**	**E-Link Data Rate: Down-Link**		
		80 Mb/s	160 Mb/s	320 Mb/s
FRMDWN[39:32]	0	n.a		
			dIO[2,0]	
				dIO[0]
FRMDWN[47:40]	0	dIO[7:4]		
			dIO[6,4]	
				dIO[4]
FRMDWN[63:48]	1	dIO[15:8]		
			dIO[14,12,10,8]	
				dIO[12,8]
FRMDWN[79:64]	2	dOut[23:16]		
			dOut[22,20,18,16]	
				dOut[20,16]
FRMDWN[95:80]	3	dOut[31:24]		
			dOut[30,28,26,24]	
				dOut[28,24]
FRMDWN[111:96]	4	dOut[39:32]		
			dOut[38,36,34,32]	
				dOut[36,32]

Table 18 Maximum number of Up/Down Links in the 8B/10B mode

E-Link Data Rate	80 Mb/s	160 Mb/s	320 Mb/s
Max # of up-links	44	22	11
Max # of down-links	36	20	10

3.2.4 Enabling disabling individual channels

The number of input/output data channels that can be used is constrained by the GBT mode and the data rate as specified in Table 9 to Table 17. Within the allowed set, channels can be individually enable disabled as explained next.

3.2.4.1 Enabling the output E-Links (E-Ports)

For each GBT mode and a given E-Link data rate Table 10, Table 14 and Table 17 specify the active groups and the association between output ports and the frame field. For each group, the registers used to enable/disable a specific channel are given in Table 19. One should note that only channels allowed by a given GBT mode and E-Link data rate can be enabled.

Table 19 Output data ports enable registers

Register	**Register Name**	**Function**
256 [7:0]	outEportCtr2[7:0]	dataPortEnable Group 0 channels [7:0] A
259 [7:0]	outEportCtr5[7:0]	dataPortEnable Group 1 channels [7:0] A
262 [7:0]	outEportCtr8[7:0]	dataPortEnable Group 2 channels [7:0] A
265 [7:0]	outEportCtr11[7:0]	dataPortEnable Group 3 channels [7:0] A
268 [7:0]	outEportCtr14[7:0]	dataPortEnable Group 4 channels [7:0] A
334 [7:0]	outEportCtr22[7:0]	dataPortEnable Group 0 channels [7:0] B
337 [7:0]	outEportCtr25[7:0]	dataPortEnable Group 1 channels [7:0] B
340 [7:0]	outEportCtr28[7:0]	dataPortEnable Group 2 channels [7:0] B
343 [7:0]	outEportCtr31[7:0]	dataPortEnable Group 3 channels [7:0] B
346 [7:0]	outEportCtr34[7:0]	dataPortEnable Group 4 channels [7:0] B
349 [7:0]	outEportCtr37[7:0]	dataPortEnable Group 0 channels [7:0] C
352 [7:0]	outEportCtr40[7:0]	dataPortEnable Group 1 channels [7:0] C
355 [7:0]	outEportCtr43[7:0]	dataPortEnable Group 2 channels [7:0] C

358 [7:0]	outEportCtr46[7:0]	dataPortEnable Group 3 channels [7:0] C
361 [7:0]	outEportCtr49[7:0]	dataPortEnable Group 4 channels [7:0] C

3.2.4.2 Enabling the input E-Links (E-Ports)

As for the output E-Links, the set of input E-Links that can be used is restricted by the GBT mode and the selected E-Link data rate. Table 9, Table 13 and Table 16 specify the active groups and the association between input ports and the frame field. For each group, the registers used to enable/disable a specific channel are given in Table 20 and Table 21. One should note that only channels allowed by a given GBT mode and E-Link data rate can be enabled.

Table 20 Input E-Ports (phase aligner channels) enable registers

Register	Register Name	Function
81 [7:0]	inEportCtr19[7:0]	Phase aligner enable channels [7:0] Group 0 A
82 [7:0]	inEportCtr20[7:0]	Phase aligner enable channels [7:0] Group 0 B
83 [7:0]	inEportCtr21[7:0]	Phase aligner enable channels [7:0] Group 0 C
105 [7:0]	inEportCtr43[7:0]	Phase aligner enable channels [7:0] Group 1 A
106 [7:0]	inEportCtr44[7:0]	Phase aligner enable channels [7:0] Group 1 B
107 [7:0]	inEportCtr45[7:0]	Phase aligner enable channels [7:0] Group 1 C
129 [7:0]	inEportCtr67[7:0]	Phase aligner enable channels [7:0] Group 2 A
130 [7:0]	inEportCtr68[7:0]	Phase aligner enable channels [7:0] Group 2 B
131 [7:0]	inEportCtr69[7:0]	Phase aligner enable channels [7:0] Group 2 C
153 [7:0]	inEportCtr91[7:0]	Phase aligner enable channels [7:0] Group 3 A
154 [7:0]	inEportCtr92[7:0]	Phase aligner enable channels [7:0] Group 3 B
155 [7:0]	inEportCtr93[7:0]	Phase aligner enable channels [7:0] Group 3 C
177 [7:0]	inEportCtr115[7:0]	Phase aligner enable channels [7:0] Group 4 A
178 [7:0]	inEportCtr116[7:0]	Phase aligner enable channels [7:0] Group 4 B
179 [7:0]	inEportCtr117[7:0]	Phase aligner enable channels [7:0] Group 4 C
201 [7:0]	inEportCtr139[7:0]	Phase aligner enable channels [7:0] Group 5 A
202 [7:0]	inEportCtr140[7:0]	Phase aligner enable channels [7:0] Group 5 B
203 [7:0]	inEportCtr141[7:0]	Phase aligner enable channels [7:0] Group 5 C
225 [7:0]	inEportCtr163[7:0]	Phase aligner enable channels [7:0] Group 6 A
226 [7:0]	inEportCtr164[7:0]	Phase aligner enable channels [7:0] Group 6 B
227 [7:0]	inEportCtr165[7:0]	Phase aligner enable channels [7:0] Group 6 C

Table 21 Input EC - channel enable register

Register	Register Name	Function
248 [2:0]	inEportCtr186[2:0]	Phase aligner enable EC channel {C,B,A}

3.3 E-link clocks

For each group the E-Link clock signals (dClk+/dClk-) can be programmed independently to have any of the following frequencies: 40 MHz, 80 MHz, 160 MHz or 320 MHz. If, for example, the E-Link clocks are programmed to be 40 MHz for the 2x data rate (80 Mb/s) the links and its associated clocks basically run as a Double Date Rate (DDR) connection with the clock having rising and falling transitions occurring in the middle of the bit period (see Figure 11). When an e-Clock of 80, 160 or 320 MHz is used the reader is made aware that in the front-end module there will be a phase uncertainty with relation to the 25 ns basic clock/frame cycle as shown in Figure 11. It is fully up to the user to resolve this at the system level (e.g. sending specific synchronization data at initialization).

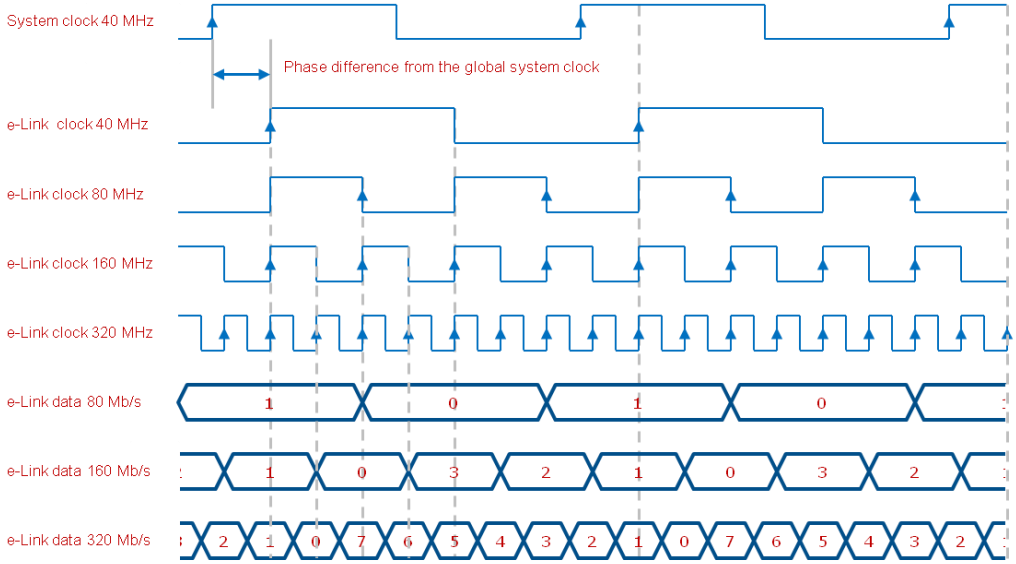


Figure 11 E-Link clocks and data outputs

It should also be noticed that the absolute phase of the e-clocks will have a certain phase shift in relation to the global system (from counting house) depending on delays in the link and the global system (as illustrated in figure above). The absolute phase of front-end systems has to be dealt with as part of general timing alignment approach used in each system.

3.4 Group allowed I/O and Clock ports combinations

Within a group the number of channels and clock ports that can be used are restricted by the ePort mode and data rate. The dependencies can be found Table 22 to Table 24

Table 22 ePort group transmitter channel dependencies

GBTX MODE		Transmitter (FEC or widebus)		
eportTX MODE	eportRX MODE	eportTX	eportRX	eportClk
IDLE	IDLE	NONE	NONE	CH[7:0]
	80 Mbps	NONE	CH[7:0]	CH[7:0]
	160 Mbps	NONE	CH[6,4,2,0]	CH[6,4,2,0]
	320 Mbps	NONE	CH[4,0]	CH[4,0]
80 Mbps	IDLE	NONE	NONE	CH[7:0]
	80 Mbps	NONE	CH[7:0]	CH[7:0]

	160 Mbps	NONE	CH[6,4,2,0]	CH[6,4,2,0]
	320 Mbps	NONE	CH[4,0]	CH[4,0]
160 Mbps	IDLE	NONE	NONE	CH[7:0]
	80 Mbps	NONE	CH[7:0]	CH[7:0]
	160 Mbps	NONE	CH[6,4,2,0]	CH[6,4,2,0]
	320 Mbps	NONE	CH[4,0]	CH[4,0]
320 Mbps	IDLE	NONE	NONE	CH[7:0]
	80 Mbps	NONE	CH[7:0]	CH[7:0]
	160 Mbps	NONE	CH[6,4,2,0]	CH[6,4,2,0]
	320 Mbps	NONE	CH[4,0]	CH[4,0]

Table 23 ePort group receiver channel dependencies

GBTX MODE		**Receiver**		
eportTX MODE	**eportRX MODE**	**eportTX**	**eportRX**	**eportClk**
IDLE	IDLE	NONE	NONE	CH[7:0]
	80 Mbps	NONE	NONE	CH[7:0]
	160 Mbps	NONE	NONE	CH[7:0]
	320 Mbps	NONE	NONE	CH[7:0]
80 Mbps	IDLE	CH[7:0]	NONE	CH[7:0]
	80 Mbps	CH[7:0]	NONE	CH[7:0]
	160 Mbps	CH[7:0]	NONE	CH[7:0]
	320 Mbps	CH[7:0]	NONE	CH[7:0]
160 Mbps	IDLE	CH[6,4,2,0]	NONE	CH[7:0]
	80 Mbps	CH[6,4,2,0]	NONE	CH[7:0]
	160 Mbps	CH[6,4,2,0]	NONE	CH[7:0]
	320 Mbps	CH[6,4,2,0]	NONE	CH[7:0]
320 Mbps	IDLE	CH[4,0]	NONE	CH[7:0]
	80 Mbps	CH[4,0]	NONE	CH[7:0]
	160 Mbps	CH[4,0]	NONE	CH[7:0]
	320 Mbps	CH[4,0]	NONE	CH[7:0]

Table 24 ePort clock channel dependencies

GBTX MODE		**Transceiver**		
eportTX MODE	**eportRX MODE**	**eportTX**	**eportRX**	**eportClk**
IDLE	IDLE	NONE	NONE	CH[7:0]
	80 Mbps	NONE	CH[7:0]	CH[7:0]
	160 Mbps	NONE	CH[6,4,2,0]	CH[6,4,2,0]
	320 Mbps	NONE	CH[4,0]	CH[4,0]
80 Mbps	IDLE	CH[7:0]	NONE	CH[7:0]

	80 Mbps	CH[7:0]	CH[7:0]	CH[7:0]
	160 Mbps	CH[7:0]	CH[6,4,2,0]	CH[6,4,2,0]
	320 Mbps	CH[7:0]	CH[4,0]	CH[4,0]
160 Mbps	IDLE	CH[6,4,2,0]	NONE	CH[7:0]
	80 Mbps	CH[6,4,2,0]	CH[7:0]	CH[7:0]
	160 Mbps	CH[6,4,2,0]	CH[6,4,2,0]	CH[6,4,2,0]
	320 Mbps	CH[6,4,2,0]	CH[4,0]	CH[4,0]
320 Mbps	IDLE	CH[4,0]	NONE	CH[7:0]
	80 Mbps	CH[4,0]	CH[7:0]	CH[7:0]
	160 Mbps	CH[4,0]	CH[6,4,2,0]	CH[6,4,2,0]
	320 Mbps	CH[4,0]	CH[4,0]	CH[4,0]

3.5 E-link Lanes

E-links can be grouped together as lanes to obtain higher effective data bandwidth to/from a single front-end module. In this case a single dClk+/dClk- can be used for several dOut+/dOut- and/or dIn+/dIn- serial data signals. The E-Link configuration for each serial data signal must be equivalent and it is up to the user to assure that an appropriate phase alignment is maintained (equal distance and loading of dClk+/dClk- and dOut+/dOut- lines).

The use of one large wide parallel lane per GBTX is effectively a subset of this with all E-Groups configured to run in the 2x serial mode and only use one common dClk+/dClk- of 40MHz.

It is noted that an E-Link with for instance a lane of 8 signals will not have the bits multiplexed at the byte level (as usually seen) but at the bit level as indicated below for a 8 line lane running at 8 x 80Mb/s = 640Mb/s. The GBT link interface in the counting house (FPGA's) can easily remap/swap this to fit the application in mind.

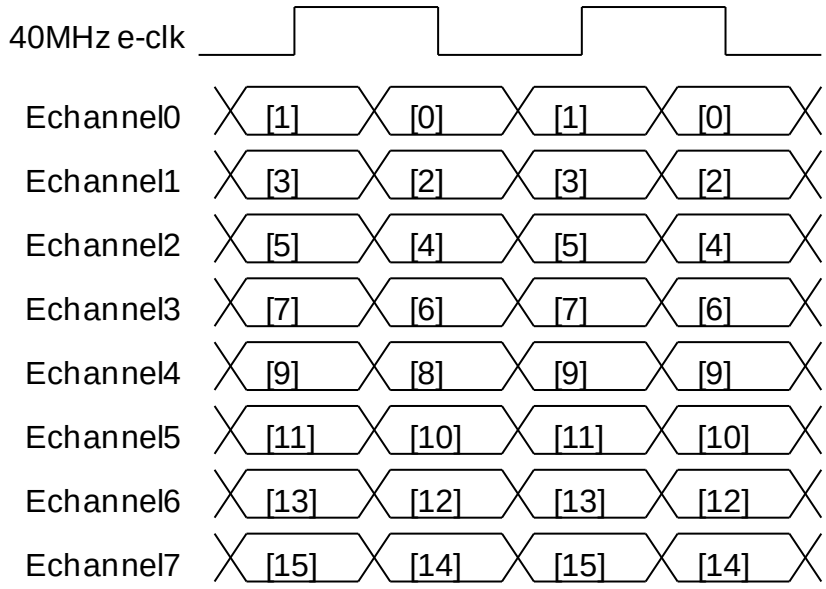


Figure 12. 8 channel lane with 80Mb/s per data line and 40MHz clock

3.6 E-link Port adaptor

To ensure compatibility and facilitate the development of the front-end electronics, the GBT project proposes the use of an E-Link Port Adaptor (EPA) "macro" that will

be available for integration in front-end ASICs (see [17]). This E-link port adaptor takes care of the physical E-link interface in front-end ASICs covering the multiple data rates of E-Links and its grouping into lanes. Such an ePort facilitates the interaction with the user application since it hides from it the transmission frame details and uses a standard Atlantic BUS [18] to interface with the user application in the ASIC.

3.7 Phase alignment

Phase delays between the GBTX and the front-end electronics will depend on the system configuration, local cable lengths and delays in the front-end circuits. It is thus necessary that the E-Link ports provide a means of adjusting the phases of the incoming data signals so that data is sampled reliably in the middle of the eye-opening. A phase adjustment/alignment mechanism is thus necessary in the GBTX data input, and in some cases also in the ePorts of the front-end ASICs.

3.7.1 Downlink phase alignment

Two possibilities are foreseen for the E-Links in the down direction from the GBTX to the ePort of the front-end ASICs. In one case both data and clock are simultaneously transmitted to the front-end ASIC and in the other only data is transmitted without a dedicated E-link clock. As the GBTX is unaware of the code/frame/data structure carried by the E-Links, it does not contribute to the data down phase alignment and synchronization mechanism.

When both dOut+/dOut- data and e-Clock lines are routed to the frontend ASIC (Figure 13, top E-Link) they must follow the same electrical route and have the same loading to assure that their relative phase is maintained at the arrival to the front-end ASIC.

In case no dClk+/dClk- is used and only the dOut+/dOut- data line is routed to the frontend ASIC (to save signal lines) the E-Link clock needs to be recovered locally in the frontend ASIC (see Figure 13, bottom E-Link) with a Clock and Data Recovery (CDR) circuit. To assure that such a local CDR circuit can reliably re-generate the link clock, the data must be appropriately encoded with sufficient data transitions (and normally also DC balanced to enable AC coupling of the data signals). The GBTX does not have built in dedicated logic for such a line encoding as it can be done in the counting house (and the GBTX is fully data transparent). Proposed line codes for this are scrambling, which incurs no bandwidth penalty, and 7B/8B encoding with a code efficiency of 87.5%. The 7B/8B code has built-in comma characters that can be used for frame delimiting and synchronization. Other codes are possible but they must ensure a sufficiently high density of transitions for clock recovery (see 3.6 for more details).

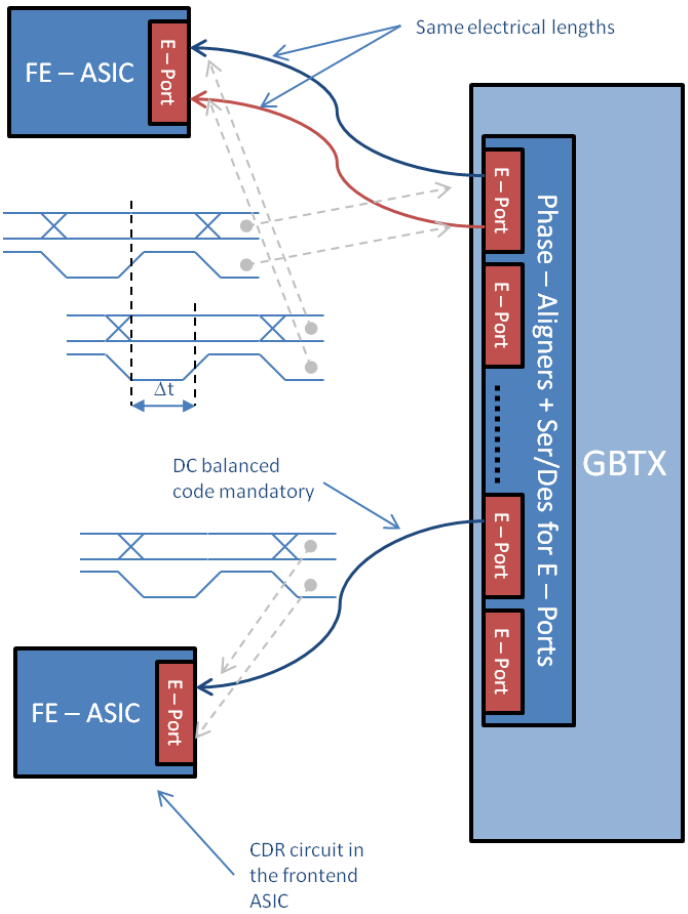


Figure 13 Downlink synchronization with and without an E-Link clock.

3.7.2 Up-Link phase alignment

For the return links (Figure 14) the phase of the incoming data to the GBTX is unknown. However, the data rate is known from the GBTX and frontend modules configuration. The GBTX clocks are synchronous with the frontend module clocks with a fixed and stable phase relationship. It is thus unnecessary to recover the clock from the data but it is necessary to phase align the incoming dIn+/dIn- data with the clock in the GTBX for each E-link. A dedicated phase-aligner circuit is responsible for this for each E-Link.

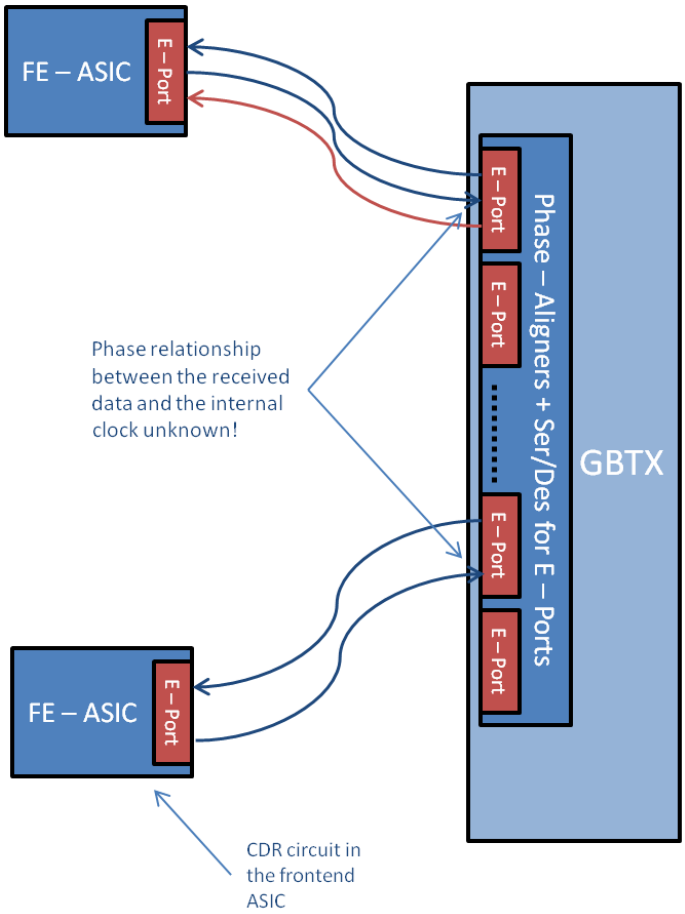


Figure 14 E-link connections and phase of dIn+/dIn- data signals.

The phase aligner circuit ensures that the E-link data received by the GBTX is sampled by the GBTX internal clock in the middle of the eye-diagram. The block diagram of the circuit is shown in Figure 15. One general phase aligner per E-Group is available with 8 phase adjustable channels as all E-Links in a group work at the same data rate, but may have different phases.

The phase aligner is composed of a master Delay Locked Loop (DLL) and eight replica delay-lines with programmable phase taps. Each replica delay-line can adjust the phase of the incoming data via the delay taps along the delay line. The phase aligner channels can operate in three different modes: static phase selection, initial training with learned static phase selection and automatic phase tracking.

In "static phase selection" mode the channel phase is set either at system initialization via the configuration stored in the E-Fuses or via a dedicated ePort command through the IC channel. This allows the E-link ports to accept data without any DC balance restrictions.

In the "initial training with learned static phase selection" mode, initially the user has to set the ePort in the phase learning mode and ensure that DC balanced data is sent through the ePort channel being "trained". After the phase-aligner has found the optimum phase settings for the specified port, the user can set the phase-aligner to hold the phase setting and then resume normal data transmission through the E-Link without concerns of DC balance on the data being transmitted. The phase settings found can be read through the IC channel. The main difference between this mode and the previous is that in this case the phase settings do not need to be pre-programmed but can be learned from a system initialization phase. This mode requires however the intervention of the user to set the learning phase, to set the frontend module to send balanced data, to set the hold phase-settings operation and then to program the frontend modules to resume normal data

transmission. The most likely use of this mode is during the initial-phases of system development or operation to determine the best phase settings which can then later be used regularly with the static phase selection mode.

In "automatic phase tracking" mode the actual phase of the received data is estimated from data bit transitions and used to dynamically adjust the phase alignment. For this to work reliable, data transitions are necessary. In this case a DC balanced code is desirable however it is strictly not necessary since the phase-aligner waits for 8 data transitions in a given ePort channel before it decides to adjust the phase setting. In an ePort group the phase alignment is made in a "circular" fashion skipping channels that are not being used. It is very important that the user will disable the channels that are unused in order to save power. The algorithm used, steps the phase up or down is based on an average of 8 samples and phase-changes are done incrementally in steps of ±T/8, where T is the bit period. The phase-changes are done in such a way that no data transmission errors are introduced when that phase is being changed on the fly.

There is a lock state machine that counts the number of transitions that fall in the expected region. If 64 transitions are detected, then the channel is declared as locked. Conversely, the channel is considered unlocked if 64 transitions fall outside the expected range.

In all modes, unused channels can be powered down to save power.

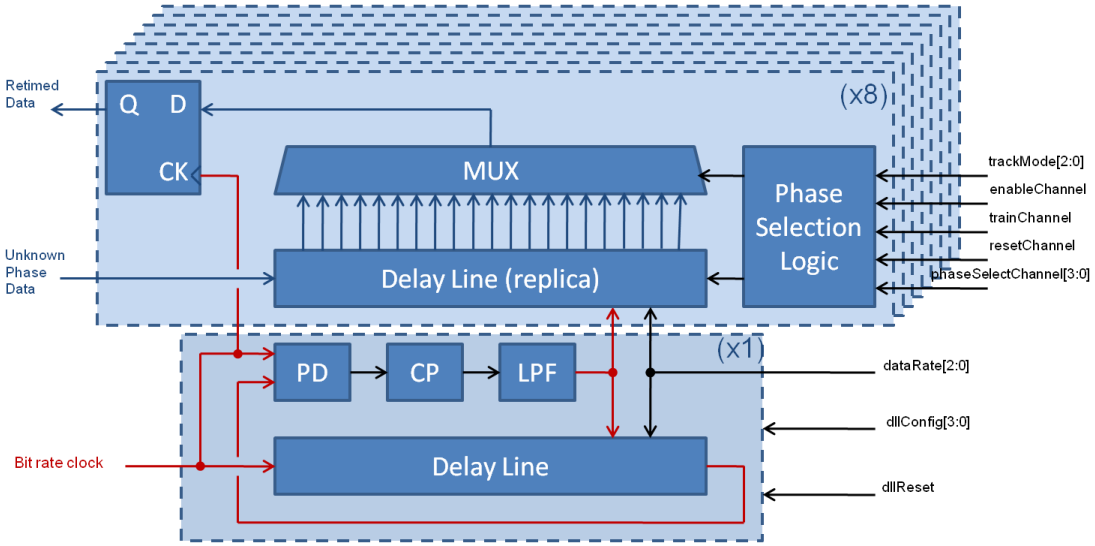


Figure 15 Phase – Aligner block diagram

3.7.2.1 Selecting the phase aligner tracking mode

The phase aligner tracking mode is selected by programming register 62 as follows:

Register 62[5:0] = inEportCtr0[5:0]
= {paModeC[1:0], paModeB[1:0], paModeA[1:0]}

Where paModeA/B/C[1:0] are given in Table 25.

Table 25 Phase tracking mode

Phase aligner tracking mode ABC	
2'b00	Static phase selection
2'b01	Training mode
2'b10	Automatic phase tracking
2'b11	Not used

3.8 DC balancing and data/clock encoding

For some applications it might be necessary to AC couple the E-Link connections (e.g. serial powering of front-ends). In this case a DC-Balanced code must be transmitted over each data line. If the E-Clock is not used on the E-links, Clock and Data Recovery (CDR) is required in the front-end ASIC. This encoding/decoding must take place at the optical link source in the counting room, where flexible FPGA based link interfaces are used, and in the front-end itself. The encoding overhead will depended on the type of encoding used. Suggested encoding schemes are listed in Table 26, with their relative merits. As the GBTX is fully transparent to the user data being transferred it is not directly involved in any line coding being used on the local E-links.

Table 26 Options for DC balanced local codes

	7B/8B	Scrambler	8B/10B	Manchester
"Odd" word size	Yes	No	Yes	No
User bandwidth	87.5%	100%	80%	50%
Frame synch. "built in"	Yes	No	Yes	No

3.9 Programming the E-Links

Programming the E-Links involves the following operations:

Clock ports:

- Set the clock bus frequency for each group (see 3.9.1);
  - For 160 and 320 MHz clocks, the E-PLL RX (when in transceiver and simplex-RX modes) or the E-PLL TX (when in simplex-TX mode) must be enabled and configured (see section 9.5).
- Enable individual clock ports on each group (see 3.9.1);
- Set the clock drivers strength per group (see 13.2).

Data output ports:

- Set the data rate for each group (see 3.1.1);
  - For 160 and 320 Mb/s the E-PLL RX needs to be enabled and configured (see section 9.5).
- Enable individual data ports on each group (see 3.2.4.1).
- Set the data drivers strength per group (see 13.2);

Data input ports:

- Set the phase-aligner mode (common to all the groups – see 3.7.2.1);

- Set the DLL parameters for each group (see 3.9.2).
- Set the data rate for each group (see 3.1.2);
    - For 160 and 320 Mb/s the E-PLL TX needs to be enabled and configured (see section 9.5).
- Enable individual data ports on each group (see 3.2.4.2).
- Depending on the mode: select the phase for each active channel or execute a train/hold sequence for each active channel or set the automatic phase align mode.

Note that the input and output data rates can be programmed independently from each other and the clock frequencies independently from the input/output data rates.

3.9.1 Programming the E-Link clocks

All the clocks in a group have the same frequency; however the group frequency can be set independently of the data rate programmed for the data inputs and outputs of that group.

The registers that program the clock bus frequency and enable/disable the clock ports are triplicated and consequently each Triple Modular Redundancy (TMR) register must contain the same data; the user must ensure that this is the case.

Table 27 to Table 29 innumerate the registers used to set the clock bus frequencies and the corresponding clock settings.

Table 27 E-Link clock frequency programing registers

Register	**Register Name**	**Function (see** Table 28**)**
254 [3:2]	outEportCtr0[3:2]	clockBusFrequencyA0[3:2]
257 [3:2]	outEportCtr3[3:2]	clockBusFrequencyA1[3:2]
260 [3:2]	outEportCtr6[3:2]	clockBusFrequencyA2[3:2]
263 [3:2]	outEportCtr9[3:2]	clockBusFrequencyA3[3:2]
266 [3:2]	outEportCtr12[3:2]	clockBusFrequencyA4[3:2]
332 [3:2]	outEportCtr20[3:2]	clockBusFrequencyB0[3:2]
335 [3:2]	outEportCtr23[3:2]	clockBusFrequencyB1[3:2]
338 [3:2]	outEportCtr26[3:2]	clockBusFrequencyB2[3:2]
341 [3:2]	outEportCtr29[3:2]	clockBusFrequencyB3[3:2]
344 [3:2]	outEportCtr32[3:2]	clockBusFrequencyB4[3:2]
347 [3:2]	outEportCtr35[3:2]	clockBusFrequencyC0[3:2]
350 [3:2]	outEportCtr38[3:2]	clockBusFrequencyC1[3:2]
353 [3:2]	outEportCtr41[3:2]	clockBusFrequencyC2[3:2]
356 [3:2]	outEportCtr44[3:2]	clockBusFrequencyC3[3:2]
359 [3:2]	outEportCtr47[3:2]	clockBusFrequencyC4[3:2]

Table 28 E-Link clock frequency settings

clockBusFrequencyYX[3:2]	**Clock frequency in MHz**

with Y = A, B and C X = 0, 1, 2, 3 and 4	
2'b00	40
2'b01	80
2'b10	160
2'b11	320

Table 29 EC port clock settings

Register 257[6:4] = outEportCtr3[3:2] = {clockBusFrequencyEcC, clockBusFrequencyEcB, clockBusFrequencyEcA}	**Clock frequency in MHz**
3'b000	40
3'b111	80
Other combinations not allowed	

Individual clock lines are enabled/disabled according to the data in Table 30. A zero in the corresponding channel bit will disable the channel while a one will enable it. Unused channels should be disabled to save power. The bit correspondence is bit 0 to channel 0, bit 1 to channel 1, etc.

Table 30 Clock port enable registers

Register	**Register Name**	**Function**
255 [7:0]	outEportCtr1[7:0]	clockPortEnableGroupA0[7:0]
258 [7:0]	outEportCtr4[7:0]	clockPortEnableGroupA1[7:0]
261 [7:0]	outEportCtr7[7:0]	clockPortEnableGroupA2[7:0]
264 [7:0]	outEportCtr10[7:0]	clockPortEnableGroupA3[7:0]
267 [7:0]	outEportCtr13[7:0]	clockPortEnableGroupA4[7:0]
333 [7:0]	outEportCtr21[7:0]	clockPortEnableGroupB0[7:0]
336 [7:0]	outEportCtr24[7:0]	clockPortEnableGroupB1[7:0]
339 [7:0]	outEportCtr27[7:0]	clockPortEnableGroupB2[7:0]
342 [7:0]	outEportCtr30[7:0]	clockPortEnableGroupB3[7:0]
345 [7:0]	outEportCtr33[7:0]	clockPortEnableGroupB4[7:0]
348 [7:0]	outEportCtr36[7:0]	clockPortEnableGroupC0[7:0]
351 [7:0]	outEportCtr39[7:0]	clockPortEnableGroupC1[7:0]
354 [7:0]	outEportCtr42[7:0]	clockPortEnableGroupC2[7:0]
357 [7:0]	outEportCtr45[7:0]	clockPortEnableGroupC3[7:0]
360 [7:0]	outEportCtr48[7:0]	clockPortEnableGroupC4[7:0]

3.9.2 Programming and initialization of the phase-aligner DLLs

Each ePort receiver group (ePort input group) is associated with phase-aligners that serve the 8 inputs associated with it. The phase-aligners are calibrated by

a DLL. Consequently, before an ePort group is ready for operation one has to choose the data rate and program all the DLL parameters.

3.9.2.1 DLL Charge-Pump current

There are 8 DLLs associated with the 8 ePort groups and one DLL associated with the EC channel group. These DLLs are configured by the set of registers given in Table 31 and Table 32, respectively. Each DLL requires its charge-pump current to be set: this is done by setting 3 words of for bits for each DLL as indicated in the tables. The recommended value is 4'b1011. As an example, to configure the DLL charge-pump current for group 2 the user has to set registers 112 and 113 as follows:

112 [3:0] = 4'b1011;
112 [7:4] = 4'b1011;
113 [3:0] = 4'b1011;

Table 31 ePort DLLs configuration registers

Register	Register Name	Function
64 [3:0]	inEportCtr2[3:0]	Phase aligner dll config Group 0 A
64 [7:4]	inEportCtr2[7:4]	Phase aligner dll config Group 0 B
65 [3:0]	inEportCtr3[3:0]	Phase aligner dll config Group 0 C
88 [3:0]	inEportCtr26[3:0]	Phase aligner dll config Group 1 A
88 [7:4]	inEportCtr26[7:4]	Phase aligner dll config Group 1 B
89 [3:0]	inEportCtr27[3:0]	Phase aligner dll config Group 1 C
112 [3:0]	inEportCtr50[3:0]	Phase aligner dll config Group 2 A
112 [7:4]	inEportCtr50[7:4]	Phase aligner dll config Group 2 B
113 [3:0]	inEportCtr51[3:0]	Phase aligner dll config Group 2 C
136 [3:0]	inEportCtr74[3:0]	Phase aligner dll config Group 3 A
136 [7:4]	inEportCtr74[7:4]	Phase aligner dll config Group 3 B
137 [3:0]	inEportCtr75[3:0]	Phase aligner dll config Group 3 C
160 [3:0]	inEportCtr98[3:0]	Phase aligner dll config Group 4 A
160 [7:4]	inEportCtr98[7:4]	Phase aligner dll config Group 4 B
161 [3:0]	inEportCtr99[3:0]	Phase aligner dll config Group 4 C
184 [3:0]	inEportCtr122[3:0]	Phase aligner dll config Group 5 A
184 [7:4]	inEportCtr122[7:4]	Phase aligner dll config Group 5 B
185 [3:0]	inEportCtr123[3:0]	Phase aligner dll config Group 5 C
208 [3:0]	inEportCtr146[3:0]	Phase aligner dll config Group 6 A
208 [7:4]	inEportCtr146[7:4]	Phase aligner dll config Group 6 B
209 [3:0]	inEportCtr147[3:0]	Phase aligner dll config Group 6 C

Table 32 EC channel DLL configuration register

Register	Register Name	Function
231 [3:0]	inEportCtr169[3:0]	Phase aligner dll config EC channel A
231 [7:4]	inEportCtr169[7:4]	Phase aligner dll config EC channel B
232 [3:0]	inEportCtr170[3:0]	Phase aligner dll config EC channel C

3.9.2.2 Phase-Aligners Lock Detection Mode

The DLLs in the phase-aligners have a lock detection mechanism to monitor their operation. There are two operating modes that are set in register 233 as indicated in Table 33. The lock detection mode is set for all DLLs identical. Practical experience has determined that the "coarse lock detection" is the most robust mechanism so it is recommended that the user sets:

233 [6:4] = 3'b111;

Table 33 Phase-Aligner Lock detection mode setting register

Register	Register Name	Function
233 [6:4]	inEportCtr171[6:4]	Phase aligner dll coarse lock detection {C,B,A} = 3'b000 fine lock detection = 3'b111 coarse lock detection

3.9.2.3 Phase-Aligners DLL reset

After the data rate, the DLL charge-pump current and the lock detection mode are set the DLL needs to be reset. This is done through the set of registers given Table 34 and Table 35. Resetting one group implies setting three bits in a register to reset the DLL and then clear these bits to allow the DLL to operate normally. So for example resetting the DLL in group 5 requires the following sequence:

1. Write:
   185 [6:4] = 3'b111;
2. Write:
   185 [6:4] = 3'b000;

If the chip has been previously configured by either I2C, IC – channel or the fuses have been fused, the initialization state machine will take care of doing the DLL reset cycle after the chip is reset (see chapter 10).

Table 34 ePort group DLL reset registers

Register	Register Name	Function
65 [6:4]	inEportCtr3[6:4]	Phase aligner dll reset Group 0 {C,B,A}
89 [6:4]	inEportCtr27[6:4]	Phase aligner dll reset Group 1 {C,B,A}
113 [6:4]	inEportCtr51[6:4]	Phase aligner dll reset Group 2 {C,B,A}
137 [6:4]	inEportCtr75[6:4]	Phase aligner dll reset Group 3 {C,B,A}
161 [6:4]	inEportCtr99[6:4]	Phase aligner dll reset Group 4 {C,B,A}
185 [6:4]	inEportCtr123[6:4]	Phase aligner dll reset Group 5 {C,B,A}
209 [6:4]	inEportCtr147[6:4]	Phase aligner dll reset Group 6 {C,B,A}

Table 35 EC channel DLL reset register

Register	Register Name	Function
232 [6:4]	inEportCtr170[6:4]	Phase aligner dll reset EC channel {C,B,A}

3.9.3 Input ePort Phase Selection

As discussed in 3.7 there are three possibilities to select the phase-alignment for the ePorts. The most basic consists of using the "static phase selection" where the user programs the phase for each ePort. A second possibility is the "training mode" where the user first allows the phase-aligner to find the optimum phase and after this initial "training" cycle the guessed phase is frozen for the remainder of the operation. The third option is the "automatic mode" where the circuit at all times tries to lock to the optimum phase. Although this last mode has potential advantages it was observed that in the presence of Single Event Upsets (SEU) the circuit can fail so its use is not recommended in environments where SEUs are likely (please see section 3.7.2.1 on how to choose the phase tracking mode).

3.9.3.1 Static phase selection registers

When in the static phase selection mode the user has to set, for each active input ePort, the required phase. Choosing the correct phase is critical for error free operation of the system where the GBTX is being used. A possible procedure to help choosing the phase of each channel is as follows:

1. Sweep the channel phase;
2. For each phase, record if the data transmission occurs error free or not;
3. For operation choose a phase value that is in the middle of an error free band.

When applying the procedure above the user must keep in mind that while sweeping the phase it is possible that bits can "jump" between two consecutive bunch crossings while being still correctly sampled. If not correctly accounted for, this might be interpreted as a transmission error. The user must thus be attentive to this issue when deciding which phase to choose.

For each ePort group Table 36 to Table 43 give the set of registers that should be used to program the phase of each channel. Notice that the registers are triplicated; programming the phase of given channel requires thus writing the "A", "B" and "C" registers.

It is strongly recommended that in environments where SEUs are a concern that the static phase selection mode is used. In those cases either the training or the automatic mode could be used during the development phases to choose the correct values to program when later using the static phase selection mode.

Table 36 Group 0 phase select registers

Register	Register Name	Function
66 [3:0]	inEportCtr4[3:0]	Phase aligner phase select Group 0 channel 6 A
66 [7:4]	inEportCtr4[7:4]	Phase aligner phase select Group 0 channel 7 A
67 [3:0]	inEportCtr5[3:0]	Phase aligner phase select Group 0 channel 4 A
67 [7:4]	inEportCtr5[7:4]	Phase aligner phase select Group 0 channel 5 A
68 [3:0]	inEportCtr6[3:0]	Phase aligner phase select Group 0 channel 2 A
68 [7:4]	inEportCtr6[7:4]	Phase aligner phase select Group 0 channel 3 A
69 [3:0]	inEportCtr7[3:0]	Phase aligner phase select Group 0 channel 0 A
69 [7:4]	inEportCtr7[7:4]	Phase aligner phase select Group 0 channel 1 A
70 [3:0]	inEportCtr8[3:0]	Phase aligner phase select Group 0 channel 6 B
70 [7:4]	inEportCtr8[7:4]	Phase aligner phase select Group 0 channel 7 B
71 [3:0]	inEportCtr9[3:0]	Phase aligner phase select Group 0 channel 4 B

71 [7:4]	inEportCtr9[7:4]	Phase aligner phase select Group 0 channel 5 B
72 [3:0]	inEportCtr10[3:0]	Phase aligner phase select Group 0 channel 2 B
72 [7:4]	inEportCtr10[7:4]	Phase aligner phase select Group 0 channel 3 B
73 [3:0]	inEportCtr11[3:0]	Phase aligner phase select Group 0 channel 0 B
73 [7:4]	inEportCtr11[7:4]	Phase aligner phase select Group 0 channel 1 B
74 [3:0]	inEportCtr12[3:0]	Phase aligner phase select Group 0 channel 6 C
74 [7:4]	inEportCtr12[7:4]	Phase aligner phase select Group 0 channel 7 C
75 [3:0]	inEportCtr13[3:0]	Phase aligner phase select Group 0 channel 4 C
75 [7:4]	inEportCtr13[7:4]	Phase aligner phase select Group 0 channel 5 C
76 [3:0]	inEportCtr14[3:0]	Phase aligner phase select Group 0 channel 2 C
76 [7:4]	inEportCtr14[7:4]	Phase aligner phase select Group 0 channel 3 C
77 [3:0]	inEportCtr15[3:0]	Phase aligner phase select Group 0 channel 0 C
77 [7:4]	inEportCtr15[7:4]	Phase aligner phase select Group 0 channel 1 C

Table 37 Group 1 phase select registers

Register	**Register Name**	**Function**
90 [3:0]	inEportCtr28[3:0]	Phase aligner phase select Group 1 channel 6 A
90 [7:4]	inEportCtr28[7:4]	Phase aligner phase select Group 1 channel 7 A
91 [3:0]	inEportCtr29[3:0]	Phase aligner phase select Group 1 channel 4 A
91 [7:4]	inEportCtr29[7:4]	Phase aligner phase select Group 1 channel 5 A
92 [3:0]	inEportCtr30[3:0]	Phase aligner phase select Group 1 channel 2 A
92 [7:4]	inEportCtr30[7:4]	Phase aligner phase select Group 1 channel 3 A
93 [3:0]	inEportCtr31[3:0]	Phase aligner phase select Group 1 channel 0 A
93 [7:4]	inEportCtr31[7:4]	Phase aligner phase select Group 1 channel 1 A
94 [3:0]	inEportCtr32[3:0]	Phase aligner phase select Group 1 channel 6 B
94 [7:4]	inEportCtr32[7:4]	Phase aligner phase select Group 1 channel 7 B
95 [3:0]	inEportCtr33[3:0]	Phase aligner phase select Group 1 channel 4 B
95 [7:4]	inEportCtr33[7:4]	Phase aligner phase select Group 1 channel 5 B
96 [3:0]	inEportCtr34[3:0]	Phase aligner phase select Group 1 channel 2 B
96 [7:4]	inEportCtr34[7:4]	Phase aligner phase select Group 1 channel 3 B
97 [3:0]	inEportCtr35[3:0]	Phase aligner phase select Group 1 channel 0 B
97 [7:4]	inEportCtr35[7:4]	Phase aligner phase select Group 1 channel 1 B
98 [3:0]	inEportCtr36[3:0]	Phase aligner phase select Group 1 channel 6 C
98 [7:4]	inEportCtr36[7:4]	Phase aligner phase select Group 1 channel 7 C
99 [3:0]	inEportCtr37[3:0]	Phase aligner phase select Group 1 channel 4 C
99 [7:4]	inEportCtr37[7:4]	Phase aligner phase select Group 1 channel 5 C
100 [3:0]	inEportCtr38[3:0]	Phase aligner phase select Group 1 channel 2 C
100 [7:4]	inEportCtr38[7:4]	Phase aligner phase select Group 1 channel 3 C
101 [3:0]	inEportCtr39[3:0]	Phase aligner phase select Group 1 channel 0 C
101 [7:4]	inEportCtr39[7:4]	Phase aligner phase select Group 1 channel 1 C

Table 38 Group 2 phase select registers

Register	Register Name	Function
114 [3:0]	inEportCtr52[3:0]	Phase aligner phase select Group 2 channel 6 A
114 [7:4]	inEportCtr52[7:4]	Phase aligner phase select Group 2 channel 7 A
115 [3:0]	inEportCtr53[3:0]	Phase aligner phase select Group 2 channel 4 A
115 [7:4]	inEportCtr53[7:4]	Phase aligner phase select Group 2 channel 5 A
116 [3:0]	inEportCtr54[3:0]	Phase aligner phase select Group 2 channel 2 A
116 [7:4]	inEportCtr54[7:4]	Phase aligner phase select Group 2 channel 3 A
117 [3:0]	inEportCtr55[3:0]	Phase aligner phase select Group 2 channel 0 A
117 [7:4]	inEportCtr55[7:4]	Phase aligner phase select Group 2 channel 1 A
118 [3:0]	inEportCtr56[3:0]	Phase aligner phase select Group 2 channel 6 B
118 [7:4]	inEportCtr56[7:4]	Phase aligner phase select Group 2 channel 7 B
119 [3:0]	inEportCtr57[3:0]	Phase aligner phase select Group 2 channel 4 B
119 [7:4]	inEportCtr57[7:4]	Phase aligner phase select Group 2 channel 5 B
120 [3:0]	inEportCtr58[3:0]	Phase aligner phase select Group 2 channel 2 B
120 [7:4]	inEportCtr58[7:4]	Phase aligner phase select Group 2 channel 3 B
121 [3:0]	inEportCtr59[3:0]	Phase aligner phase select Group 2 channel 0 B
121 [7:4]	inEportCtr59[7:4]	Phase aligner phase select Group 2 channel 1 B
122 [3:0]	inEportCtr60[3:0]	Phase aligner phase select Group 2 channel 6 C
122 [7:4]	inEportCtr60[7:4]	Phase aligner phase select Group 2 channel 7 C
123 [3:0]	inEportCtr61[3:0]	Phase aligner phase select Group 2 channel 4 C
123 [7:4]	inEportCtr61[7:4]	Phase aligner phase select Group 2 channel 5 C
124 [3:0]	inEportCtr62[3:0]	Phase aligner phase select Group 2 channel 2 C
124 [7:4]	inEportCtr62[7:4]	Phase aligner phase select Group 2 channel 3 C
125 [3:0]	inEportCtr63[3:0]	Phase aligner phase select Group 2 channel 0 C
125 [7:4]	inEportCtr63[7:4]	Phase aligner phase select Group 2 channel 1 C

Table 39 Group 3 phase select registers

Register	Register Name	Function
138 [3:0]	inEportCtr76[3:0]	Phase aligner phase select Group 3 channel 6 A
138 [7:4]	inEportCtr76[7:4]	Phase aligner phase select Group 3 channel 7 A
139 [3:0]	inEportCtr77[3:0]	Phase aligner phase select Group 3 channel 4 A
139 [7:4]	inEportCtr77[7:4]	Phase aligner phase select Group 3 channel 5 A
140 [3:0]	inEportCtr78[3:0]	Phase aligner phase select Group 3 channel 2 A
140 [7:4]	inEportCtr78[7:4]	Phase aligner phase select Group 3 channel 3 A
141 [3:0]	inEportCtr79[3:0]	Phase aligner phase select Group 3 channel 0 A
141 [7:4]	inEportCtr79[7:4]	Phase aligner phase select Group 3 channel 1 A
142 [3:0]	inEportCtr80[3:0]	Phase aligner phase select Group 3 channel 6 B
142 [7:4]	inEportCtr80[7:4]	Phase aligner phase select Group 3 channel 7 B
143 [3:0]	inEportCtr81[3:0]	Phase aligner phase select Group 3 channel 4 B

143 [7:4]	inEportCtr81[7:4]	Phase aligner phase select Group 3 channel 5 B
144 [3:0]	inEportCtr82[3:0]	Phase aligner phase select Group 3 channel 2 B
144 [7:4]	inEportCtr82[7:4]	Phase aligner phase select Group 3 channel 3 B
145 [3:0]	inEportCtr83[3:0]	Phase aligner phase select Group 3 channel 0 B
145 [7:4]	inEportCtr83[7:4]	Phase aligner phase select Group 3 channel 1 B
146 [3:0]	inEportCtr84[3:0]	Phase aligner phase select Group 3 channel 6 C
146 [7:4]	inEportCtr84[7:4]	Phase aligner phase select Group 3 channel 7 C
147 [3:0]	inEportCtr85[3:0]	Phase aligner phase select Group 3 channel 4 C
147 [7:4]	inEportCtr85[7:4]	Phase aligner phase select Group 3 channel 5 C
148 [3:0]	inEportCtr86[3:0]	Phase aligner phase select Group 3 channel 2 C
148 [7:4]	inEportCtr86[7:4]	Phase aligner phase select Group 3 channel 3 C
149 [3:0]	inEportCtr87[3:0]	Phase aligner phase select Group 3 channel 0 C
149 [7:4]	inEportCtr87[7:4]	Phase aligner phase select Group 3 channel 1 C

Table 40 Group 4 phase select registers

Register	**Register Name**	**Function**
162 [3:0]	inEportCtr100[3:0]	Phase aligner phase select Group 4 channel 6 A
162 [7:4]	inEportCtr100[7:4]	Phase aligner phase select Group 4 channel 7 A
163 [3:0]	inEportCtr101[3:0]	Phase aligner phase select Group 4 channel 4 A
163 [7:4]	inEportCtr101[7:4]	Phase aligner phase select Group 4 channel 5 A
164 [3:0]	inEportCtr102[3:0]	Phase aligner phase select Group 4 channel 2 A
164 [7:4]	inEportCtr102[7:4]	Phase aligner phase select Group 4 channel 3 A
165 [3:0]	inEportCtr103[3:0]	Phase aligner phase select Group 4 channel 0 A
165 [7:4]	inEportCtr103[7:4]	Phase aligner phase select Group 4 channel 1 A
166 [3:0]	inEportCtr104[3:0]	Phase aligner phase select Group 4 channel 6 B
166 [7:4]	inEportCtr104[7:4]	Phase aligner phase select Group 4 channel 7 B
167 [3:0]	inEportCtr105[3:0]	Phase aligner phase select Group 4 channel 4 B
167 [7:4]	inEportCtr105[7:4]	Phase aligner phase select Group 4 channel 5 B
168 [3:0]	inEportCtr106[3:0]	Phase aligner phase select Group 4 channel 2 B
168 [7:4]	inEportCtr106[7:4]	Phase aligner phase select Group 4 channel 3 B
169 [3:0]	inEportCtr107[3:0]	Phase aligner phase select Group 4 channel 0 B
169 [7:4]	inEportCtr107[7:4]	Phase aligner phase select Group 4 channel 1 B
170 [3:0]	inEportCtr108[3:0]	Phase aligner phase select Group 4 channel 6 C
170 [7:4]	inEportCtr108[7:4]	Phase aligner phase select Group 4 channel 7 C
171 [3:0]	inEportCtr109[3:0]	Phase aligner phase select Group 4 channel 4 C
171 [7:4]	inEportCtr109[7:4]	Phase aligner phase select Group 4 channel 5 C
172 [3:0]	inEportCtr110[3:0]	Phase aligner phase select Group 4 channel 2 C
172 [7:4]	inEportCtr110[7:4]	Phase aligner phase select Group 4 channel 3 C
173 [3:0]	inEportCtr111[3:0]	Phase aligner phase select Group 4 channel 0 C
173 [7:4]	inEportCtr111[7:4]	Phase aligner phase select Group 4 channel 1 C

Table 41 Group 5 phase select registers

Register	Register Name	Function
186 [3:0]	inEportCtr124[3:0]	Phase aligner phase select Group 5 channel 6 A
186 [7:4]	inEportCtr124[7:4]	Phase aligner phase select Group 5 channel 7 A
187 [3:0]	inEportCtr125[3:0]	Phase aligner phase select Group 5 channel 4 A
187 [7:4]	inEportCtr125[7:4]	Phase aligner phase select Group 5 channel 5 A
188 [3:0]	inEportCtr126[3:0]	Phase aligner phase select Group 5 channel 2 A
188 [7:4]	inEportCtr126[7:4]	Phase aligner phase select Group 5 channel 3 A
189 [3:0]	inEportCtr127[3:0]	Phase aligner phase select Group 5 channel 0 A
189 [7:4]	inEportCtr127[7:4]	Phase aligner phase select Group 5 channel 1 A
190 [3:0]	inEportCtr128[3:0]	Phase aligner phase select Group 5 channel 6 B
190 [7:4]	inEportCtr128[7:4]	Phase aligner phase select Group 5 channel 7 B
191 [3:0]	inEportCtr129[3:0]	Phase aligner phase select Group 5 channel 4 B
191 [7:4]	inEportCtr129[7:4]	Phase aligner phase select Group 5 channel 5 B
192 [3:0]	inEportCtr130[3:0]	Phase aligner phase select Group 5 channel 2 B
192 [7:4]	inEportCtr130[7:4]	Phase aligner phase select Group 5 channel 3 B
193 [3:0]	inEportCtr131[3:0]	Phase aligner phase select Group 5 channel 0 B
193 [7:4]	inEportCtr131[7:4]	Phase aligner phase select Group 5 channel 1 B
194 [3:0]	inEportCtr132[3:0]	Phase aligner phase select Group 5 channel 6 C
194 [7:4]	inEportCtr132[7:4]	Phase aligner phase select Group 5 channel 7 C
195 [3:0]	inEportCtr133[3:0]	Phase aligner phase select Group 5 channel 4 C
195 [7:4]	inEportCtr133[7:4]	Phase aligner phase select Group 5 channel 5 C
196 [3:0]	inEportCtr134[3:0]	Phase aligner phase select Group 5 channel 2 C
196 [7:4]	inEportCtr134[7:4]	Phase aligner phase select Group 5 channel 3 C
197 [3:0]	inEportCtr135[3:0]	Phase aligner phase select Group 5 channel 0 C
197 [7:4]	inEportCtr135[7:4]	Phase aligner phase select Group 5 channel 1 C

Table 42 Group 6 phase select registers

Register	Register Name	Function
210 [3:0]	inEportCtr148[3:0]	Phase aligner phase select Group 6 channel 6 A
210 [7:4]	inEportCtr148[7:4]	Phase aligner phase select Group 6 channel 7 A
211 [3:0]	inEportCtr149[3:0]	Phase aligner phase select Group 6 channel 4 A
211 [7:4]	inEportCtr149[7:4]	Phase aligner phase select Group 6 channel 5 A
212 [3:0]	inEportCtr150[3:0]	Phase aligner phase select Group 6 channel 2 A
212 [7:4]	inEportCtr150[7:4]	Phase aligner phase select Group 6 channel 3 A
213 [3:0]	inEportCtr151[3:0]	Phase aligner phase select Group 6 channel 0 A
213 [7:4]	inEportCtr151[7:4]	Phase aligner phase select Group 6 channel 1 A
214 [3:0]	inEportCtr152[3:0]	Phase aligner phase select Group 6 channel 6 B
214 [7:4]	inEportCtr152[7:4]	Phase aligner phase select Group 6 channel 7 B
215 [3:0]	inEportCtr153[3:0]	Phase aligner phase select Group 6 channel 4 B

215 [7:4]	inEportCtr153[7:4]	Phase aligner phase select Group 6 channel 5 B
216 [3:0]	inEportCtr154[3:0]	Phase aligner phase select Group 6 channel 2 B
216 [7:4]	inEportCtr154[7:4]	Phase aligner phase select Group 6 channel 3 B
217 [3:0]	inEportCtr155[3:0]	Phase aligner phase select Group 6 channel 0 B
217 [7:4]	inEportCtr155[7:4]	Phase aligner phase select Group 6 channel 1 B
218 [3:0]	inEportCtr156[3:0]	Phase aligner phase select Group 6 channel 6 C
218 [7:4]	inEportCtr156[7:4]	Phase aligner phase select Group 6 channel 7 C
219 [3:0]	inEportCtr157[3:0]	Phase aligner phase select Group 6 channel 4 C
219 [7:4]	inEportCtr157[7:4]	Phase aligner phase select Group 6 channel 5 C
220 [3:0]	inEportCtr158[3:0]	Phase aligner phase select Group 6 channel 2 C
220 [7:4]	inEportCtr158[7:4]	Phase aligner phase select Group 6 channel 3 C
221 [3:0]	inEportCtr159[3:0]	Phase aligner phase select Group 6 channel 0 C
221 [7:4]	inEportCtr159[7:4]	Phase aligner phase select Group 6 channel 1 C

Table 43 EC group phase select registers

Register	Register Name	Function
233 [3:0]	inEportCtr171[3:0]	Phase aligner phase select EC channel A
237 [3:0]	inEportCtr175[3:0]	Phase aligner phase select EC channel B
241 [3:0]	inEportCtr179[3:0]	Phase aligner phase select EC channel C

3.9.3.2 Inportant note on the phase selection in Wide Bus mode

Control of the groups 5 and 6 (only available in wide bus mode) depends on the data rate. For 80 Mb/s the channel number and the dIO ports match. For example the dIO[1] phase is controlled by "Phase aligner phase select Group 5 channel 1 A/B/C" (see Table 41). However, for 160 and 320 Mb/s dIO[1] the phase is controlled by "Phase aligner phase select Group 5 channel 0 A/B/C". This is clarified in the following table.

Table 44 Phase selection in the wide bus mode for groups 5 and 6 (Please refer to Table 41 and Table 42)

Mode	Group	elinkRX name	Controlled as
80 Mb/s	5	dIO[0]	Group 5 channel 0
		dIO[1]	Group 5 channel 1
		dIO[2]	Group 5 channel 2
		dIO[3]	Group 5 channel 3
		dIO[4]	Group 5 channel 4
		dIO[5]	Group 5 channel 5
		dIO[6]	Group 5 channel 7
		dIO[7]	Group 5 channel 7
	6	dIO[8]	Group 6 channel 0
		dIO[9]	Group 6 channel 1
		dIO[10]	Group 6 channel 2

		dIO[11]	Group 6 channel 3
		dIO[12]	Group 6 channel 4
		dIO[13]	Group 6 channel 5
		dIO[14]	Group 6 channel 7
		dIO[15]	Group 5 channel 7
160 Mb/s	5	dIO[0]	NONE
		dIO[1]	Group 6 channel 0
		dIO[2]	NONE
		dIO[3]	Group 5 channel 2
		dIO[3]	NONE
		dIO[5]	Group 5 channel 4
		dIO[6]	NONE
		dIO[7]	Group 5 channel 6
	6	dIO[8]	NONE
		dIO[9]	Group 6 channel 0
		dIO[10]	NONE
		dIO[11]	Group 6 channel 2
		dIO[12]	NONE
		dIO[13]	Group 6 channel 4
		dIO[14]	NONE
		dIO[15]	Group 6 channel 6
320 Mb/s	5	dIO[0]	NONE
		dIO[1]	Group 6 channel 0
		dIO[2]	NONE
		dIO[3]	NONE
		dIO[3]	NONE
		dIO[5]	Group 6 channel 4
		dIO[6]	NONE
		dIO[7]	NONE
	6	dIO[8]	NONE
		dIO[9]	Group 6 channel 0
		dIO[10]	NONE
		dIO[11]	NONE
		dIO[12]	NONE
		dIO[13]	Group 6 channel 4
		dIO[14]	NONE
		dIO[15]	NONE

3.9.3.3 Phase selection in the training mode

The training mode is used when the user wants to have the chip guessing the optimum sampling phase but also wants to freeze phase tracking for the remaining of the data transmission operation. The training mode consists thus of two phases: the training phase where the user sets the train bit to "1" for

each channel he wants to train, and the hold phase (freeze) where training of the channels is halted by setting the train bit to "0" for all channels. Note that it is mandatory to have data transitions on the input ePorts while the individual channels are being trained.

Table 45 and Table 46 list the registers used to set the training phase for each channel. Notice that registers are triplicated so to set a channel to be in the training modes (or clear it) three registers need to be written.

This mode is not recommended for environments were SEUs are a concern.

Table 45 Phase aligner train channels registers

Register	Register Name	Function
78 [7:0]	inEportCtr16[7:0]	Phase aligner train channels [7:0] Group 0 A
79 [7:0]	inEportCtr17[7:0]	Phase aligner train channels [7:0] Group 0 B
80 [7:0]	inEportCtr18[7:0]	Phase aligner train channels [7:0] Group 0 C
102 [7:0]	inEportCtr40[7:0]	Phase aligner train channels [7:0] Group 1 A
103 [7:0]	inEportCtr41[7:0]	Phase aligner train channels [7:0] Group 1 B
104 [7:0]	inEportCtr42[7:0]	Phase aligner train channels [7:0] Group 1 C
126 [7:0]	inEportCtr64[7:0]	Phase aligner train channels [7:0] Group 2 A
127 [7:0]	inEportCtr65[7:0]	Phase aligner train channels [7:0] Group 2 B
128 [7:0]	inEportCtr66[7:0]	Phase aligner train channels [7:0] Group 2 C
150 [7:0]	inEportCtr88[7:0]	Phase aligner train channels [7:0] Group 3 A
151 [7:0]	inEportCtr89[7:0]	Phase aligner train channels [7:0] Group 3 B
152 [7:0]	inEportCtr90[7:0]	Phase aligner train channels [7:0] Group 3 C
174 [7:0]	inEportCtr112[7:0]	Phase aligner train channels [7:0] Group 4 A
175 [7:0]	inEportCtr113[7:0]	Phase aligner train channels [7:0] Group 4 B
176 [7:0]	inEportCtr114[7:0]	Phase aligner train channels [7:0] Group 4 C
198 [7:0]	inEportCtr136[7:0]	Phase aligner train channels [7:0] Group 5 A
199 [7:0]	inEportCtr137[7:0]	Phase aligner train channels [7:0] Group 5 B
200 [7:0]	inEportCtr138[7:0]	Phase aligner train channels [7:0] Group 5 C
222 [7:0]	inEportCtr160[7:0]	Phase aligner train channels [7:0] Group 6 A
223 [7:0]	inEportCtr161[7:0]	Phase aligner train channels [7:0] Group 6 B
224 [7:0]	inEportCtr162[7:0]	Phase aligner train channels [7:0] Group 6 C

Table 46 Phase aligner train EC channel register

Register	Register Name	Function
245 [2:0]	inEportCtr183[2:0]	Phase aligner train EC channel {C,B,A}

3.9.3.4 Phase selection in the automatic phase tracking mode

In this mode the phase-aligner automatically selects the optimum phase for each channel and also automatically tracks any phase drift that might occur during system operation. However this phase tracking is limited to ±3/8 of the eLink bit period. Under normal circumstances this range should be enough to compensate most system drifts. However, if the phase drift exceeds these values the phase-aligner needs to be reset to select a new optimum sampling

value. The registers used to reset the channels are given in Table 47 and Table 48.

This mode is not recommended for environments were SEUs are a concern.

Table 47 Phase aligner reset channels registers

Register	Register Name	Function
84 [7:0]	inEportCtr22[7:0]	Phase aligner reset channels [7:0] Group 0 A
85 [7:0]	inEportCtr23[7:0]	Phase aligner reset channels [7:0] Group 0 B
86 [7:0]	inEportCtr24[7:0]	Phase aligner reset channels [7:0] Group 0 C
108 [7:0]	inEportCtr46[7:0]	Phase aligner reset channels [7:0] Group 1 A
109 [7:0]	inEportCtr47[7:0]	Phase aligner reset channels [7:0] Group 1 B
110 [7:0]	inEportCtr48[7:0]	Phase aligner reset channels [7:0] Group 1 C
132 [7:0]	inEportCtr70[7:0]	Phase aligner reset channels [7:0] Group 2 A
133 [7:0]	inEportCtr71[7:0]	Phase aligner reset channels [7:0] Group 2 B
134 [7:0]	inEportCtr72[7:0]	Phase aligner reset channels [7:0] Group 2 C
156 [7:0]	inEportCtr94[7:0]	Phase aligner reset channels [7:0] Group 3 A
157 [7:0]	inEportCtr95[7:0]	Phase aligner reset channels [7:0] Group 3 B
158 [7:0]	inEportCtr96[7:0]	Phase aligner reset channels [7:0] Group 3 C
180 [7:0]	inEportCtr118[7:0]	Phase aligner reset channels [7:0] Group 4 A
181 [7:0]	inEportCtr119[7:0]	Phase aligner reset channels [7:0] Group 4 B
182 [7:0]	inEportCtr120[7:0]	Phase aligner reset channels [7:0] Group 4 C
204 [7:0]	inEportCtr142[7:0]	Phase aligner reset channels [7:0] Group 5 A
205 [7:0]	inEportCtr143[7:0]	Phase aligner reset channels [7:0] Group 5 B
206 [7:0]	inEportCtr144[7:0]	Phase aligner reset channels [7:0] Group 5 C
228 [7:0]	inEportCtr166[7:0]	Phase aligner reset channels [7:0] Group 6 A
229 [7:0]	inEportCtr167[7:0]	Phase aligner reset channels [7:0] Group 6 B
230 [7:0]	inEportCtr168[7:0]	Phase aligner reset channels [7:0] Group 6 C

Table 48 Phase aligner reset EC channel

Register	Register Name	Function
251 [2:0]	inEportCtr189[2:0]	Phase aligner reset EC channel {C,B,A}

4. LATENCY AND PHASE STABILITY

The GTBX guarantees constant latency over time and from one power-up/initialization to the next. Constant latency enables the GBTX to be used in clock synchronous trigger systems and for precise TTC distribution to the many front-end destinations. It is brought to the attention of the reader that this is for the GBTX itself but does not necessarily apply to the link interface made in FPGA's for the off detector part of the link (see separate GBT documentation for this part in chapter 20).

There will inevitable be some phase variation of a complete link because of temperature/voltage variations in different parts of the link system. At present time, the sensitivity of the GBTX phase with temperature and voltage is still being evaluated.

More information on this topic will be added in this chapter...

5. GBTX REGISTER ACCESS

As discussed in section 2.2.2 four bits of the GBT frame are reserved for slow control applications. Two of these bits (IC[1:0]) are reserved for control and monitoring of the GBTX operation. The other two (EC[1:0]) are made available externally to allow the implementation of a slow control link to another chip, however their actual use is not restricted to that type of applications. The GBTX can also be controlled and monitored via an I2C interface as an alternative to the IC channel. The selection of IC or I2C is made using the pin configSelect. More details are given below and in Section 17.

When I2C mode is selected, the user should ensure that the bits IC[1:0] in the downlink GBTX frame are set to 2'b11.

5.1 IC control and monitoring channel

Data field bits IC[1:0] are reserved for control and monitoring of the GBTX operation. These 2 bits implement an 80 Mb/s serial channel that is used to read and write the GBTX internal registers. This channel is used at start-up to configure the GBTX and during normal operation to program and monitor the operation of the GBTX.

Programming of the GBTX through this channel can be done when the GBTX is operated in Transceiver mode. The registers in the GBLD can also be accessed by this route via a simplified I2C port between the GBTX and the GBLD (see chapter 6). The two bits IC[1:0] of the SC field in the received GBT frame contain the configuration data. These are de-multiplexed to form 8-bit words which follow a frame-based protocol. The protocol for data sent to the GBTX for a write-read operation is shown in Table 49 and for a read-only operation in Table 50.

When a write-read or read-only frame is received by the GBTX and the addresses matches the ASIC I2C address (see section 5.2.1), the GBTX will acknowledge receipt of the data by sending a similar frame back on the uplink. GBTXs that are not addressed will not return any data. A broadcast address (7'b0000000) can be used to write the same data to a number of GBTXs. In this case, the GBTX will not send the acknowledge frame back on the uplink. Note that the GBTX will not carry out any subsequent operations until the read sequence is complete. Also note that this operation is not possible if the GBTX is in simplex receiver mode.

Table 49 IC channel frame structure sent to GBTX for a write-read sequence

A	Frame delimiter 8'b 01111110	Not in parity check
B	Reserved (8 bits)	Not in parity check
C	GBTX i2c address (7 bits) + R/W bit = 0	Not in parity check
D	Command (8 bits)	In parity check
E	Number of data words n[7:0]	In parity check
	Number of data words n[15:8]	In parity check
F	Memory address [7:0]	In parity check
	Memory address [15:8]	In parity check
G	1st data (8 bits)	In parity check
	……	In parity check
	nth data (8 bits)	In parity check
H	Parity word (8 bits)	In parity check
A	Frame delimiter 8'b 01111110	Not in parity check

Table 50 IC channel frame structure sent to GBTX in a read-only sequence

A	Frame delimiter 8'b 01111110	Not in parity check
B	Reserved (8 bits)	Not in parity check
C	GBTX i2c address (7 bits) + R/W bit = 1	Not in parity check
D	Command (8 bits)	In parity check
E	Number of data words n[7:0]	In parity check
	Number of data words n[15:8]	In parity check
F	Memory address [7:0]	In parity check
	Memory address [15:8]	In parity check
G	Parity word (8 bits)	In parity check
A	Frame delimiter 8'b 01111110	Not in parity check

As shown in Table 49 and Table 50 the write-read and write-only operations follow the following structure:

A) The beginning and end of the frame are marked with the delimiter word (8'b 01111110). To ensure that a payload word is not misinterpreted as the delimiter, bit stuffing is used so that any sequence of five consecutive 1s in the payload is always followed by a 0. This bit-stuffing must be carried out by the corresponding transmitter and the de-stuffing by the receiver.

B) This is a reserved command, user can opt by either repeat the GBTX address or send 8'b 00000000.

C) An address word is then transmitted and contains the 7-bit address of that particular GBTX and a Read/Write (R/W) bit. Notice that the IC channel address is the same as the I2C (see section 5.2.1). If the address does not match, then the subsequent actions are not carried out and the GBTX will not send the acknowledge frame back on the uplink. If the R/W bit is 1, then the configuration registers are not modified but their contents are read back in the transmitted GBT frame. If the R/W bit is 0, then the registers are over-written with the values transmitted within this frame. The new values are read back in the transmitted GBT frame.

D) A Command word is then transmitted. In version 1 and 2 of the GBTX, the data in this word is ignored.

E) This is followed by two bytes to indicate the number of data words (n) in the packet, maximum 65k.

F) Then the internal address (2 bytes) of the first register to be accessed is transmitted.

G) The n data words then follow. This scheme allows access to a single register or a block of registers in consecutive memory addresses. In the frame for a read-only sequence, no data bytes are transmitted to the GBTX.

H) Finally, a parity word is transmitted where each bit is the final parity of that bit through all bytes of the frame, except the frame delimiters and the first byte "GBTX i2c address + R/W". The user should calculate a running parity and transmit it as this final word. The GBTX constructs the same parity sum from the received data and compares it to the last word of the data packet. The result of this comparison is stored in a status register and can be accessed by the user (logic 1 if the parity check was OK).

Return Frame

The structure of the frame returned by the GBTX is the same as in Table 49, with the exception of the Command word. Here, the word consists of seven 0s concatenated with an LSB which is the status bit of the previous parity check (logic 1 if the parity check was OK). Independently of the result of the parity

check, if this is a write-read operation, the data payload is always written to the respective registers and the data bytes in the returned frame are the new values that have just been written into the registers. The parity word returned is calculated based on these values.

The last parity bit check result can be read from bit zero of registers 371, 372 and 373, that is:

Register 371[0] = scStatusA[0] = parity bit A
Register 372[0] = scStatusB[0] = parity bit B
Register 373[0] = scStatusC[0] = parity bit C

Note that the result of parity check for a read-only command is not stored in the status register so that the status bit always reflects the result of the parity check for the last write-read command.

5.2 I2C slave interface

The I2C slave port allows the writing and reading of the GBT configuration registers. This can be used when the GBTX is operated in any of its modes. The GBTX is equipped with a standard I2C slave interface, and is accessed by an I2C master, for example the GBT-SCA, transmitting data with the correct address.

This configuration mode supports access to one individual register or a block of registers in consecutive addresses. To access registers, the I2C master must issue the correct slave-address, write the register address and then write/read the register data. The steps in the protocol are as follows:

Write to Register

1. Master transmits START command
2. Master transmits the 7-bit GBTX address followed by the 8th bit (R/W_) set to zero.
3. Master transmits bits [7:0] of the register address.
4. Master transmits bits [15:8] of the register address.
5. Master transmits 8-bit register data word (can be repeated).
6. Master transmits STOP command.

After step 5, the register address is automatically incremented. This feature allows a block of consecutive registers to be written to in one sequence. The address in steps 3/4 is the first register of the block and step 5 is repeated with the correct register data introduced each time.

Read from Register

1. Master transmits START command
2. Master transmits 7-bit GBTX address followed by the 8th bit (R/W_) set to zero.
3. Master transmits bits [7:0] of the register address.
4. Master transmits bits [15:8] of the register address.
5. Master transmits repeated START command.
6. Master transmits 7-bit GBTX address followed by the 8th bit (R/W_) set to one.
7. Slave transmits 8-bit register data word (can be repeated).
8. Master transmits STOP command.

After step 7, the register address is automatically incremented. This feature allows a block of configuration registers to be read in one sequence starting with the register addressed by steps 3/4.
The GBTX will also accept in step 5 above a STOP command (instead of repeated START). The multi-byte read can be continued later resuming steps 6 to 8.

5.2.1 GBTX I2C Address

The GBTX I2C and IC-channel addresses are identical and are set by four pins named "I2CADDRESS[3:0]" as follows:

GBTX I2C Address = {000, I2CADDRESS[3:0]}

The four pins have internal pull-up/pull-down resistors to set a default I2C-address. Bit[0] is pulled-up to VddIO and bits[3:1] are pulled-down to GND, thus giving a default address of {0000001}. The internal resistors are 4.7 kohm. To change the address, the user should the address pins to VDD or GND to overcome the internal pull-up/downs as required.

6. GBLD REGISTER ACCESS THROUGH THE GBTX

The GBTX contains a simplified I2C master that can be used to configure the GBLD. This master is designed specifically to write and read the 7 GBLD registers and cannot be used for programming other devices. Seven registers in the GBTX are reserved for storing the values to write to the GBLD. These can be accessed by a normal write to the GBTX by the IC interface (Figure 16 A) or by I2C (Figure 16 B). These values are then transmitted to the GBLD by accessing a reserved memory address within the GBTX, again either through I2C or the IC interface. Similarly, data can be read from the GBLD by accessing another reserved address in the GBTX. The seven values read from the GBLD are stored locally inside the GBTX and can be accessed by I2C or the IC interface.

The protocol between GBTX and GBLD uses the addressing scheme of I2C and the destination address (GBLD) is taken from a GBTX register. Thus a few GBLDs with different addresses can be connected to this bus and addressed individually using this feature.

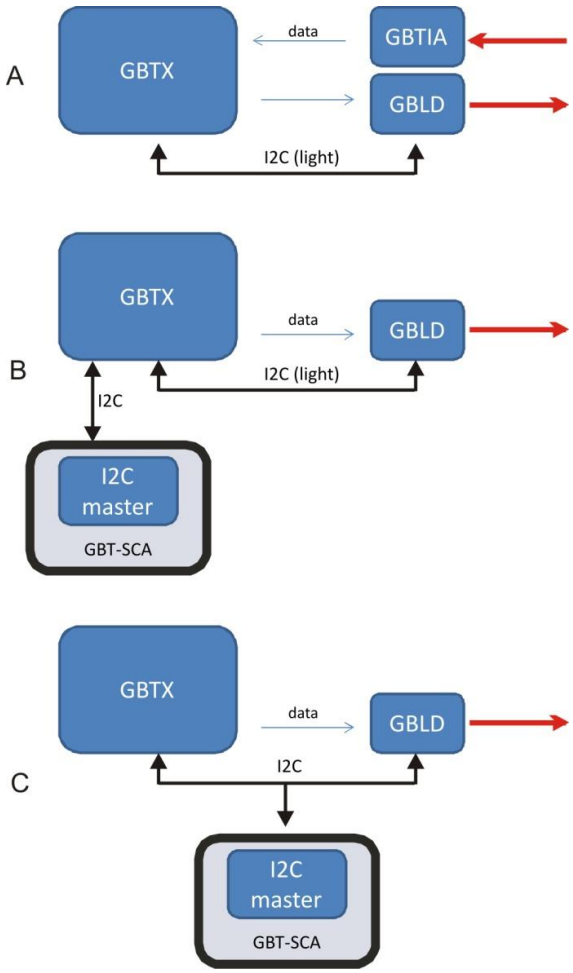


Figure 16 GBLD configuration options

When the GBLD access is launched through the GBTX IC interface, the GBTX will return a data frame on the uplink with the format of Table 49 but containing dummy data. The clock signal frequency (GBTX pin ldScl) is 40 kHz and the transaction between GBTX and GBLD lasts for 2.2 ms. During the transaction, a status bit is held low until the transaction is complete. This bit is triplicated and can be read from bits 7 of register addresses 371, 372, and 372 (decimal).

Alternatively, the GBTX and GBLD can be placed on the same I2C bus and accessed via their I2C interfaces (note that the GBLD I2C address is limited to 4-bits). This option is shown Figure 16 C.

6.1 GBLD write sequence

The following steps should be carried out to write data to the registers in the GBLD:

1. The seven 8-bit values to send to the GBLD should be written to the corresponding registers in the GBTX using the I2C or IC interfaces. The GBTX register addresses and their correspondence with the GBLD registers are listed in Table 51 (please refer to the GBLD manual for a detailed description of the GBLD registers). All seven values should be correct.
2. The I2C address of the GBLD in the optical package should be written to the corresponding GBTX register. This is GBTX register address 253 (dec).
3. GBTX register address 388 (dec) should be accessed by a write sequence of the I2C or IC interfaces. This sequence must be a single-byte operation, not a block transfer. A dummy 8-bit word should be written to the register. The transaction between GBTX and GBLD will then start and complete after 2.2 ms.

GBTX register address (dec)	GBLD register address (dec)	GBLD register name
55	0	Control
56	1	Modulation current
57	2	Bias current
58	3	Pre-emphasis
59	4	Modulation mask
60	5	Bias mask
61	6	Pre-driver

Table 51 Correspondence between GBTX and GBLD registers for GBLD writing

6.2 GBLD read sequence

The following steps should be carried out to read data from the registers in the GBLD. In version 1 of the GBTX, a GBLD read sequence can only be launched using the I2C interface of the GBTX. In version 2 of the GBTX, the GBLD read sequence can be launched using the I2C or IC interface of the GBTX.

1. The I2C address of the GBLD in the optical package should be written to the corresponding GBTX register. This is GBTX register address 253 (dec).
2. GBTX register address 389 (dec) should be accessed by a write sequence of the I2C interface. This sequence must be a single-byte operation, not a block transfer. A dummy 8-bit word should be written to the register. The transaction between GBTX and GBLD will then start and complete after 2.2 ms. The values read from the GBLD are stored in registers in the GBTX.
3. The values transferred from GBLD to GBTX can then be read by a read sequence through the I2C interface. The GBTX register addresses and their correspondence with the GBLD registers are listed in Table 52.

GBLD register address (dec)	GBLD register name	GBTX register address (dec)
0	Control	381
1	Modulation current	382
2	Bias current	383
3	Pre-emphasis	384

4	Modulation mask	385
5	Bias mask	386
6	Pre-driver	387

Table 52: Correspondence between GBLD and GBTX registers for GBLD reading

7. DATA PATH

This chapter describes the data path of the GBTX. This is shown in the block diagram of Figure 17. The top of the diagram shows the receiver logic blocks and the bottom the transmitter logic blocks.

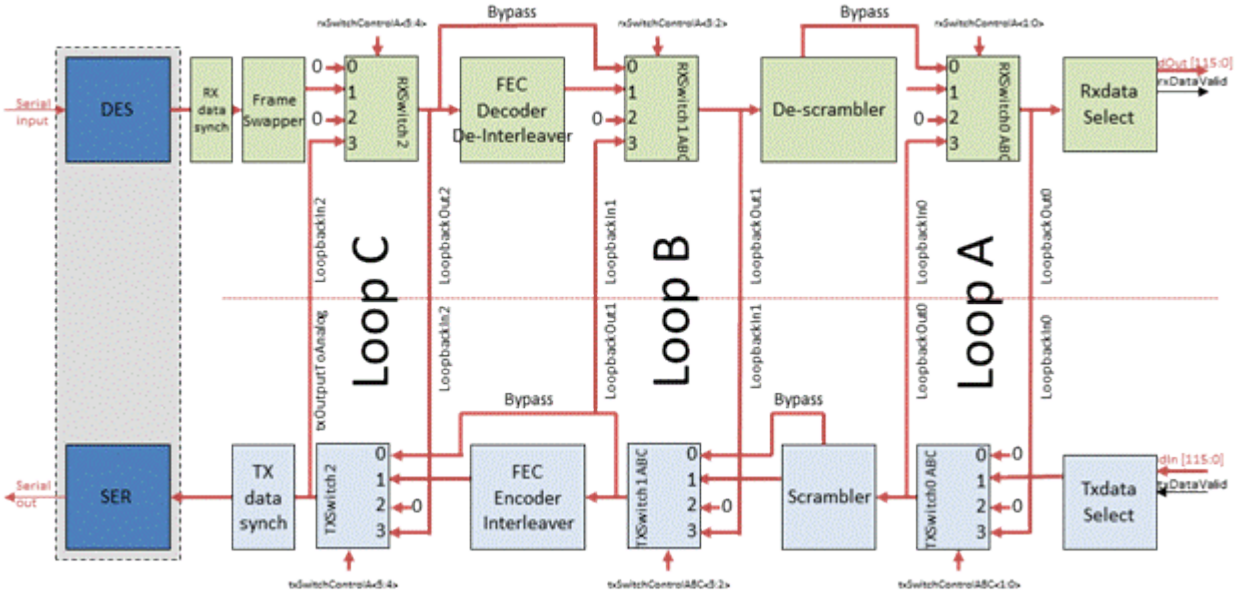


Figure 17 GBTX data path block diagram

7.1 Transmitter (TX) logic

The TX logic receives data from the ePortRX together with the external signal txDataValid. The signal txDataValid is a strobe to indicate if the frame header should contain the data-header bits (0101) or the idle-header bits (0110). If txDataValid is set by the user, the data-header is sent in the transmitted frames, otherwise the idle-header is sent. If the received frames contain the data-header, then rxDataValid is set by the GBTX.
The transmitter logic runs synchronously with the 40 MHz TX clock (txClock40). The logic is automatically reset by the power-up sequence. It can also be reset by setting bits in a configuration register via I2C:

i2cResetTx[C, B, A] = wdogCtr4[2:0] = Register 54 [2:0]

7.1.1 TXdataSelect

This block in the TX can be operated in one of four different modes. Each mode will push different types of data onto the internal 120-bit bus and into the TX logic. The mode is controlled by an internal signal called **txTestMode[1:0]** connected to a configuration register. In all modes, the header bits [119:116] are set according to the value of txDataValid. The remaining bits are mapped to:

1. Data received from the ePort and internal SC interface (normal operating mode).
2. A fixed pattern as defined below.
3. Copies of a 30-bit up-counter (counter[29:0]) synchronous with the 40MHz clock.

4. Copies of a 7-bit linear feedback shift register (prbs[6:0]). The shift register and feedback are shown in Figure 18. The value on reset is prbs[6:0] = 7'b1001000.

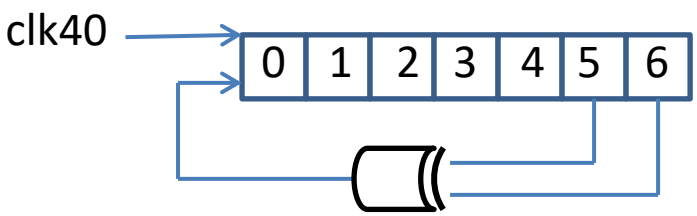


Figure 18: Linear feedback register for pseudo-random-bit-sequence

txTestMode[1:0] = txCtr0[1:0] = Register 28[1:0]

txTestMode[1:0]	120-bit bus (hex)
00	Header + IC [115:114] + ePort data[113:0]
01	Header + AAA_BBBB_AAAA_BBBB_AAAA_BBBB_AAAA_BB
10	Header + counter[25:0] + counter[29:0] + counter[29:0] + counter[29:0]
11	Header + 4'h0 + prbs[6:0] x 16

7.1.2 Scrambler

This block scrambles the 116 bits of payload data as described in chapter 0. This block does not require configuration.

7.1.3 FEC Encoder/Interleaver

This block calculates the Reed-Solomon code and interleaves the data, as described in chapter 0. Alternatively, it does the encoding in 8b/10b mode. In widebus mode, these functions are automatically bypassed.
The triple-voting of the FEC encoding can be disabled by asserting the following configuration register bit:

txDisableEncoderTMR = txCtr0[6] = Register 28 [6]

7.1.4 TXdataSynch

At the end of the TX data path before data is transferred to the serialiser, a circuit re-synchronises the data to the rising and falling edge of txClock40. A static multiplexer selects which of these to transmit to the serialiser. The multiplexer is controlled by the following configuration bits:

TXselectPosEdge[C, B, A] = inEportCtr182[5:3] = Register 244[5:3].

Setting these bits to 111 will select the data synchronised to the rising edge, while 000 selects the data synchronised to the falling edge.

7.1.5 TXSwitches

These can be used for bypassing blocks in the data path and for loopbacks, as described in section 16.2. Each switch has a 2-bit select value, and these values

are stored in the following configuration registers. The different selectable paths and their corresponding bits in the registers are shown in Figure 17 and are given in Table 53, Table 54 and Table 55.

txSwitchesControlA[7:0] = txCtr1[7:0] = Register 29 [7:0]
txSwitchesControlB[7:0] = txCtr2[7:0] = Register 30 [7:0]
txSwitchesControlC[7:0] = txCtr3[7:0] = Register 31 [7:0]

Table 53 TX switch 0 mapping

TX Switch 0	
txSwitchesControlA/B/C[1:0] = Register 29/30/31 [1:0]	*Switch input*
2'b00	120'h000000000000000000000000000000
2'b01	Header + IC [115:114] + Eport data[113:0]
2'b10	120'h000000000000000000000000000000
2'b11	RX Switch 0 output

Table 54 TX switch 1 mapping

TX Switch 1	
txSwitchesControlA/B/C[3:2] = Register 29/30/31 [3:2]	*Switch input*
2'b00	TX Switch 0 output followed by one stage pipeline delay (txClock40)
2'b01	Scrambler output
2'b10	120'h000000000000000000000000000000
2'b11	RX Switch 1 output

Table 55 TX switch 2 mapping

TX Switch 2	
txSwitchesControlA/B/C[5:4] = Register 29/30/31 [5:4]	*Switch input*
2'b00	TX Switch 1 output
2'b01	FEC encoder interleaver output
2'b10	120'h000000000000000000000000000000
2'b11	RX Switch 2 output

7.2 Receiver (RX) logic

The RX logic receives data from the deserialiser. If the received frame contains the data-header (0101), then rxDataValid is set high by the GBTX. If the frame contains the idle-header (0110), then rxDataValid is low.
The receiver core logic runs synchronously with the 40 MHz RX clock (rxClock40).
The logic is automatically reset by the power-up sequence. It can also be reset by setting bits in a configuration register via I2C:

i2cResetRx[C, B, A] = wdogCtr0[5:3] = Register 50 [5:3]

7.2.1 RXdataSynch

At the beginning of the RX data path, where data arrives from the deserialiser, this circuit synchronises the data to the rising and falling edge of rxClock40. A static multiplexer selects which of these to transmit to the RX data path. The multiplexer is controlled by the following configuration bits:

RXselectPosEdge[C, B, A] = inEportCtr182[2:0] = Register 244[2:0].

Setting these bits to 111 will select the data synchronised to the rising edge while 000 will select the data synchronised to the falling edge.

7.2.2 Frame Swapper

This selects the correct aligned data words. This block does not require configuration.

7.2.3 FEC Decoder/De-Interleaver

This carries out the error-correction using the Reed-Solomon code and reverses the data interleaving.
The triple-voting of the FEC decoding can be disabled by asserting the following configuration register bit:

rxDisableDecoderTMR = rxCtr14[1] = Register 49 [1]

7.2.4 De-scrambler

This de-scrambles the data words. This block does not require configuration.

7.2.5 RXdataSelect

This block can be operated in one of two different modes. Each mode will push different types of data into the EportTX. The mode is controlled by internal signals called **rxTestMode[C, B, A]** connected to a configuration register:

rxTestMode[C, B, A] = rxCtr14[7:5] = Register 49[7:5]

When **rxTestMode[C, B, A]** = 000, data from the RX core logic is sent to the EportTX (normal operating mode). When **rxTestMode[C, B, A]** = 111, fixed test patterns are sent to the EportTX. These patterns are chosen to allow diagnostics on the ePort and to allow correct alignment of the data sent on to a front-end chip. The type of pattern depends on the mode selected for each of the 5 groups of the ePort. Each ePort group handles 16 bits of data. The ePort modes are defined as:

Mode[1:0]	
00	Idle
01	x 2 = 80 Mb/s
10	x 4 = 160 Mb/s
11	x 8 = 320 Mb/s

Note that each ePort group can be independently configured in one of these modes. The 16-bit patterns transmitted to each ePort group are determined by the mode setting in the following manner:

Mode[1:0]	16-bit data to ePort Group
00 = idle	0101010101010101 = 16'h5555
01 = x 2	0101010101010101 = 16'h5555
10 = x 4	0111011101110111 = 16'h7777

11 = x 8	0111111101111111 = 16'h7F7F

A 16-bit error-counter is also included in this block. It will compare the data received from the RX logic (bits [115:32] of the frame) and compares them with the fixed pattern:
[115:32] = AAA_BBBB_AAAA_BBBB_AAAA_BB

The counter is enabled by the internal signals enableBERT_ABC connected to a configuration register:

enableBERT[C, B, A] = rxCtr14[4:2] = Register 49[4:2]

The counter is zeroed by turning on the enable bits. The 16-bit counter value can be read via the I2C or IC interfaces by reading registers 369 (counter[7:0]) and 370 (counter[15:8]). Notice that the counter will stop counting once the counter reaches the end count (0xFFFF).

7.2.6 RXSwitches

These can be used for bypassing blocks in the data path and for loopbacks, as described in section 16.2. Each switch has a 2-bit select value, and these values are stored in the following configuration registers. The different selectable paths and their corresponding bits in the registers are shown in Figure 17 and are given in Table 56, Table 57 and Table 58.

rxSwitchesControlA[7:0] = rxCtr11[7:0] = Register 46 [7:0]
rxSwitchesControlB[7:0] = rxCtr12[7:0] = Register 47 [7:0]
rxSwitchesControlC[7:0] = rxCtr13[7:0] = Register 48 [7:0]

Table 56 RX switch 0 mapping

RX Switch 0	
rxSwitchesControlA/B/C[1:0] = Register 46/47/48 [1:0]	*Switch input*
2'b00	RX switch 1 output followed by one stage pipeline delay (rxClock40)
2'b01	De-scrambler output
2'b10	120'h000000000000000000000000000000
2'b11	TX switch 1 output

Table 57 RX switch 1 mapping

RX Switch 1	
rxSwitchesControlA/B/C[3:2] = Register 46/47/48 [3:2]	*Switch input*
2'b00	RX switch 2 output
2'b01	FEC decoder and de-interleaver output
2'b10	120'h000000000000000000000000000000
2'b11	TX switch 1 output

Table 58 RX switch 2 mapping

RX Switch 2

rxSwitchesControlA/B/C[5:4] = Register 46/47/48 [5:4]	*Switch input*
2'b00	120'h000000000000000000000000000000
2'b01	Frame swapper output
2'b10	120'h000000000000000000000000000000
2'b11	TX switch 2 output

7.3 Summary of configuration inputs

The registers listed in Table 59 are used to configure the GBTX data path.

Table 59 Data path configuration registers

Register address	**Register name**	**Bits**
28	txCtr0	[1:0]
28	txCtr0	[6]
244	inEportCtr182	[5:3]
244	inEportCtr182	[2:0]
49	rxCtr14	[1]
49	rxCtr14	[7:5]
49	rxCtr14	[4:2]
29	txCtr1	[7:0]
30	txCtr2	[7:0]
31	txCtr3	[7:0]
46	rxCtr11	[7:0]
47	rxCtr12	[7:0]
48	rxCtr13	[7:0]
54	wdogCtr4	[2:0]
50	wdogCtr0	[5:3]

8. SLOW CONTROL CHANNEL (EC)

The two bit EC field of the GBT frame is available to implement a bidirectional serial channel with a bandwidth of 80 Mb/s (if bidirectional link). This operates an E-Link in the 2 x mode with an e-Clock of 40/80 MHz. The slow control path is primarily intended to communicate with a GBT-SCA chip but can be used for other purposes. Conversely, the GBT-SCA can operate with any other ePort provided that that port is operated as an E-Link in the 2 × mode.

Note that for communication with the GBT-SCA, the e-Clock must be 40 MHz to allow dual-data-rate communication with the 80 Mb/s serial stream.

The following settings should be used to configure the E-link for communicating with the GBT-SCA.

EC DLL configuration A[3:0] = inEportCtr169[3:0] = Register 231[3:0]
EC DLL configuration B[3:0] = inEportCtr169[7:4] = Register 231[7:4]
EC DLL configuration C[3:0] = inEportCtr170[3:0] = Register 232[3:0]
EC DLL reset ABC = inEportCtr170[6:4] = Register 232[6:4]
EC DLL coarse-lock-detection ABC = inEportCtr171[6:4] = Register 233[6:4]

EC enable ABC = inEportCtr186[2:0] = Register 248[2:0]
Phase aligner track mode A[1:0] = inEportCtr0[1:0] = Register 62[1:0]
Phase aligner track mode B[1:0] = inEportCtr0[3:2] = Register 62[3:2]
Phase aligner track mode C[1:0] = inEportCtr0[5:4] = Register 62[5:4]

EC channel reset ABC = inEportCtr189[2:0] = Register 251[2:0]
EC channel train ABC = inEportCtr183[2:0] = Register 245[2:0]

EC sample phase select A[3:0] = inEportCtr171[3:0] = Register 233[3:0]
EC sample phase select B[3:0] = inEportCtr175[3:0] = Register 237[3:0]
EC sample phase select C[3:0] = inEportCtr179[3:0] = Register 241[3:0]

EC data rate mode ABC = outEportCtr0[6:4] = Register 254[5:4]
Set to 3'b111 to select 80 Mb/s for the GBT-SCA
EC clock frequency ABC = outEportCtr3[6:4] = Register 257[6:4]
Set to 3'b000 to select 40 MHz for the GBT-SCA

EC enable receiver termination = ckCtr4[5] = Register 273[5]
EC driver current control [3:0] = ckCtr4[3:0] = Register 273[3:0]

9. ASIC OPERATION CONTROL

This chapter describes the control logic of the GBTX. In general, this carries out the initialisation of a number of sub-blocks and subsequently monitors their status. Most of these status signals are feedback to the power-up state machine and the watchdog (see chapter 10 for power-up and watchdog operation). The GBT configuration is loaded at power-on, or after a GBTX reset, from a bank of electrically programmable fuses (see chapter 15 for further details on e-Fuses). The fuses allow the GBTX to be optimized for specific applications already at the power-up stage. The e-Fuse configuration data is programmed after production testing of the ASICs targeting specific user requirements**.** The settings of the e-Fuses are copied into the GBTX configuration registers (see chapter 16 for a complete list of the GBTX registers) after which the GBTX will start its internal initialization procedure according to the selected link mode (receiver, transmitter or transceiver).

After appropriate initialization, the GBTX configuration can be modified via the optical link itself, that is the IC channel (if used in transceiver mode), or via the I2C configuration interface. Monitoring of the GBT/link and read-back of configuration data can also be performed via these two control interfaces.

All configuration registers in the GBTX are clock gated (for low power consumption) and fully protected against SEU's with triple redundant registers with auto error recovery.

9.1 Transceiver modes

External mode signals **MODE [3:0]** (see chapter 18 for pin numbers) define the most basic operation modes (see 1.3) and configuration of the GBT according to the table below.

Table 60 Transceiver mode configuration signals

Encoding/Bus mode	Transceiver mode	MODE [3:0]
FEC	Simplex TX	0000
FEC	Simplex RX	0001
FEC	Transceiver	0010
FEC	Test	0011
WideBus	Simplex TX	0100
WideBus	Simplex RX	0101
WideBus	Transceiver	0110

WideBus	Test	0111
8b/10b	Simplex TX	1000
8b/10b	Simplex RX	1001
8b/10b	Transceiver	1010
8b/10b	Test	1011
(reserved)	(reserved)	1100
(reserved)	(reserved)	1101
(reserved)	(reserved)	1110
(reserved)	(reserved)	1111

9.2 TX control

This is a state-machine which initialises and monitors the status of the serialiser. It monitors the locked status of the serialiser PLL, and if this is stable for longer than a specified time (see txPllLockTime in Table 61) then it signals the power-up state-machine/watchdog that the serialiser is locked and operational. It will signal that the serialiser has lost lock either immediately or only if lock is lost for longer than a specified time if soft-loss-of-lock is enabled (see txLossOfLockTime in Table 61). Soft-loss-of-lock is enabled by writing a 1'b1 in bit 6 of register 33:

Register 33[6] = txCtr5[6] = txEnableSoftLossOfLock

When lock is lost, the TX Loss of Lock Counter will increment. The contents of the counter can be read through I2C or SC-IC. The counter is only reset by the power-on-reset or external reset. The loss of lock count can be read from register 376:

Register 376[7:0] = txLossOfLockCount[7:0]

The output of the control block can be forced into the "locked" state via I2C or SC-IC by writing into bit 4 of register 32:

Register 32[4] = txForceLockState

The block is automatically reset at power-up and by the external reset. It can also be reset by writing into configuration register 33 via I2C.

Register 33[2:0] = txCtr5[2:0] = i2cTxReset[C, B, A]

A reset through the I2C requires bits *i2cTxReset[C, B, A]* to be set and then cleared.

The TX control configurable parameters are listed in Table 61.

Table 61 TX control parameters

Parameter	Register	Bits	Function
txPllLockTime	32 = txCtr4	[3:0]	Sets number of cycles that serialiser PLL must stay locked before TXcontrol asserts locked state. Number of cycles = 2 ^ txPllLockTime. Maximum value = 9 (dec).
txLossOfLockTime	32 = txCtr4	[7:5]	Sets number of cycles for which serialiser PLL should be unlock before TXcontrol de-asserts locked state. Only active if soft-loss-of-lock is enabled. Number of cycles = 2 ^ txLossOfLockTime. Maximum value = 4 (dec).
txEnableSoftLossOfLock	33 = txCtr5	[6]	Enables the soft-loss-lock.
txForceLockState	32 = txCtr4	[4]	Forces TX control to assert the locked state.
i2cTxReset[C, B, A]	33 = txCtr5	[2:0]	Resets TX control

9.3 RX control

The *RX Control State Machine* (RXCSM) controls the operation of the receiver section of the GBTX. The receiver is divided in two main circuits: the *Clock and Data Recovery* (CDR) circuit and the *Frame Aligner* (FA) circuit. The CDR function is further subdivided into *Frequency Locking* (FL) and *Phase Locking* (PL) functions. The block diagrams of the CDR and FA circuits are represented in Figure 19 and Figure 20 respectively.

9.3.1 Clock and Data Recovery

For the 4.8 Gb/s serial data stream to be correctly received by the GBTX both the frequency and phase of the de-serializer clock (CDR clock) need to match precisely the carrier frequency and phase of the incoming data. In the case of the GBTX a *Half-Rate* (HR) CDR architecture is used which results (in the CDR circuit) in a 180 degree phase ambiguity that is resolved later in the receiving chain with the help of the frame aligner circuit (see Frame Aligner).
CDR is achieved in the GBTX by a two stage process. In the first phase the frequency of the CDR PLL is pulled close to 2.4 GHz ("half data rate" clock) either by an auxiliary PLL, locked to a reference 40 MHz clock, or by a "calibration" DAC under the control of a frequency calibration state machine. In the second phase, after completion of frequency calibration, the CDR loop is enabled and the CDR PLL locks to the frequency and phase of the incoming data stream (see 9.3.2).
The locking procedure is fully controlled by the RX Control State Machine (RXCSM) without user intervention. However, it is also possible to fully control the lock procedure through the I2C interface. This option is however strictly reserved for test purposes only.

9.3.1.1 Lock mode selection

Important note: During production testing, it has been found that in some chips the reference PLL frequency calibration mode does not work (RXLOCKMODE[1:0]=2'b10). So for yield reasons, the users

should not use this mode and must use the DAC frequency calibration (RXLOCKMODE[1:0]=2'b01).

The different locking modes mentioned above, are selected by the pins *RXLOCKMODE [1:0]* according to Table 62 (see chapter 18 for pin numbers).

Table 62 Receiver lock mode configuration signals

RXLOCKMODE [1:0]	**Receiver lock mode**
2'b00	I2C frequency calibration
2'b01	Automatic DAC frequency calibration
2'b10	Automatic Reference PLL frequency calibration (**do not use this mode**)
2'b11	(reserved)

9.3.2 CDR Circuit Architecture and Operation

The architecture of the CDR circuit is shown in Figure 19. There it can be seen that the CDR PLL receives a coarse tuning voltage that centres the CDR PLL VCO frequency around 2.4 GHz. This voltage is provided either from the reference PLL or from the DAC. A few switches are responsible for the source selection:

1. The DAC switch allows the DAC to drive the coarse tuning voltage and is enabled / disabled by the signals *rxDacEnable[C, B, A]*. When these switches are enabled the DAC controls the coarse tuning voltage. Whenever the REF PLL is used to control the coarse voltage these switches must be disabled.
2. The coarse voltage can be driven by the reference PLL when either the Low-Pass Filter (LPF) is enabled *rxFilterEnable[C, B, A] = 3'b111* or when the filter is bypassed *rxFilterBipass[C, B, A] = 3'b111*. Whenever the DAC is used to control the coarse voltage both switches must be open.

The CDR switches are automatically controlled by the *Lock Control State Machine* (LCSM). However, for testing purposes, these switches can be controlled through the I2C interface. This requires the user to set the mode pins to *RXLOCKMODE[1:0]* = 2'b00 or the signals *i2cRxControlOverride[C, B, A]* = 3'b111 before the contents of registers 42 to 45 can have any effect on the GBTX operation (see Table 71 to Table 76 for details). This last option is given only for ASIC test purposes; the user is strongly discouraged from trying to control the CDR function through the I2C interface.

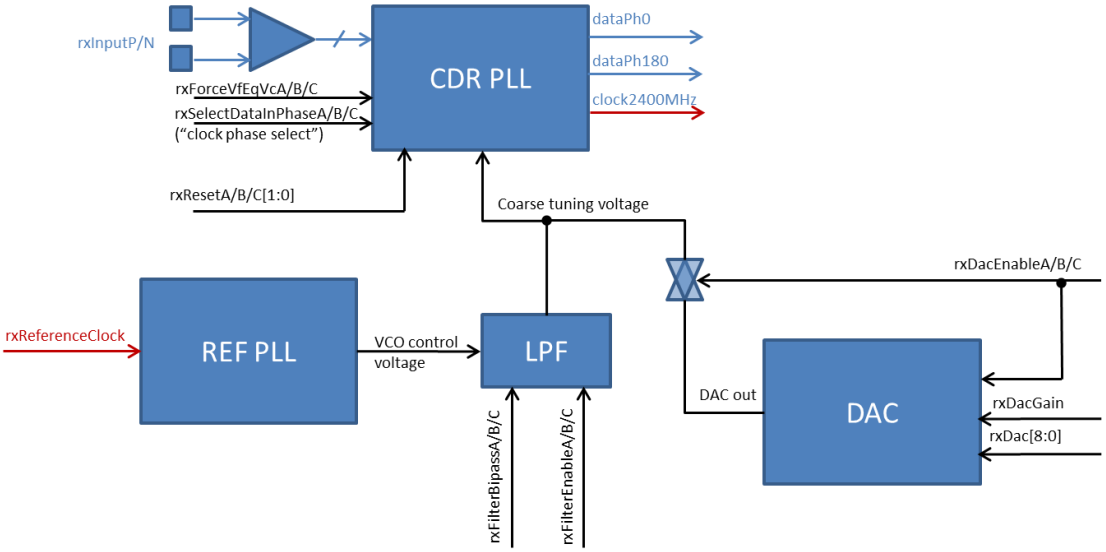


Figure 19 Clock and Data Recovery circuit block diagram

9.3.2.1 Rx Frequency Centring

Depending on the lock mode (*RXLOCKMODE [1:0]*) the lock control state machine will select either the REF PLL or the DAC for frequency centring. In both cases until the frequency centring process is concluded, the fine and coarse control voltages of the CDR VCO are shorted together and the CDR loop has no control over the VCO frequency. This is done by asserting signal *rxForceVfEqVc[C, B, A]* = 3'b111 and setting the phase and frequency detectors charge pumps currents to zero. All this is made under the supervision of the LCSM without user intervention. However, for testing purposes, this can be controlled through the I2C interface by the signals *i2cRxForceVfEqVc[C, B, A]*.
When the I2C control is enabled (*i2cRxControlOverride[C, B, A]* = 3'b111 or *RXLOCKMODE [1:0]* = 2'b00) the DAC can also be controlled through the I2C interface through the signals *rxDacGain* and *i2cRxDac [8:0]*. The former signal sets the DAC range (high/low) and the latter is the DAC count to be digital-to-analogue converted (see Table 65, Table 68 and Table 73). When enabled, the DAC drives the coarse tuning voltage and thus controls the CDR VCO centre frequency.
Notice that the signal *rxDacGain* is a configuration setting signal. It is always active in lock modes 2'b00 and 2'b01 (provided the DAC is enabled) independently of the state of signal *i2cRxControlOverride[C, B, A]*.

Frequency centring is complete when the REF PLL has acquired lock (*RXLOCKMODE [1:0]* = 2'b10) or the lock control state machine has concluded a binary search process to calibrate the CDR VCO frequency (*RXLOCKMODE [1:0]* = 2'b01). In the automatic DAC frequency calibration mode, the LCSM uses two 11-bit counters to compare the frequency of the CDR VCO (divided by 60) to the frequency of the reference clock. During a binary search cycle the two counters are simultaneously reset and then allowed to count up till one of them reaches the "end count". One of the counters runs with the reference clock (40 MHz) and is called *referenceFast* and the other, *vcoFast*, runs with the RX clock *rxClock40*. Whichever counter reaches the end count first gives an indication to the state machine if the CDR VCO clock frequency is high or low in relation to the reference clock. When running the locking process under the I2C control, there are a few registers that can be used to control the *referenceFast* and *vcoFast* counters and to verify the status of the frequency comparison, these are:

1. Signal *i2cStartRace[C, B, A] = Register 42 [2:0]* used to enable and clear the race counters: *referenceFast* and *vcoFast*. Setting this bits to "3'b000"

clears the race counters and setting them to "3'b111" enables the counters (see Table 71).

2. Signal *vcoFast = Register 377 [1]* reports the result of the frequency race counters (see Table 77). When bit *Register 377 [1]* is read, if the returned value is "1'b0" this indicates that the reference frequency is higher than the VCO frequency, if it is "1'b1" it means that the VCO frequency is higher than the reference frequency. Notice that there is no flag indicating that a "frequency race" is complete. Because of this, when controlling the "race" counters through I2C, enough time must be allowed between enabling the "race' counters and reading the status bit *Register 377 [1]*. The minimum waiting time is 110 ms.

9.3.2.2 Rx Phase Locking

When the frequency centring process is concluded, the CDR loop enters a phase-frequency locking procedure. This is enabled by setting *rxForceVfEqVc[C, B, A]* = 3'b000 and by enabling the CDR charge-pump currents. The CDR loop will be then pulled in frequency and phase until it locks to the incoming data. As mentioned before phase locking with an half rate CDR architecture has an intrinsic 180 degree phase ambiguity of the recovered clock. The control signal *rxSelectDataInPhase[C, B, A]* is used to resolve that ambiguity and guarantee that the GBTX clock and data can be recovered with no phase ambiguity and thus the ASIC will always display constant latency. The signals *rxSelectDataInPhase[C, B, A]* are under the control of the *RX Control State Machine* (RXCSM) as explained later when the frame aligner will be discussed (section 9.3.3).
When I2C control is enabled these signals can be controlled by signal *i2cRxSelectDataInPhase[C, B, A] = Register 36[5:3]* (see Table 69).
The loop parameters of the CDR and Reference PLL are controlled by the contents of registers 34, 35 and 41 as indicated in Table 66, Table 67 and Table 70. These registers are always active independently of the lock mode. However, during frequency centring the charge-pump currents of the CDR loop are overwritten (set to zero) by the lock control state machine.

Table 63 Reference PLL filter resistor selection

rxSelectR[1:0] = Register 34 [1:0]	**Reference PLL filter resistor**
2'b00	500 Ω
2'b01	700 Ω
2'b10	1000 Ω
2'b11	2000 Ω

Table 64 CDR PLL filter resistor selection

rxSelectR[3:2] = Register 34 [3:2]	**CDR PLL filter resistor**
2'b00	500 Ω
2'b01	700 Ω
2'b10	1000 Ω
2'b11	2000 Ω

Table 65 Signal rxDacGain[C, B, A]

rxDacGain[C, B, A] = Register 34 [6:4]	**DAC voltage range**
3'b000	570 mV to 640 mV
3'b111	590 mV to 710 mV

Other combinations are invalid	n.a.

Table 66 CDR frequency detector charge pump current selection

i2cRxSelectI2[3:0] = Register 35 [3:0]	**CDR frequency detector charge pump current [μA]**
4'bxxxx	I = 0.375 × i2cRxSelectI2[3:0]

Table 67 CDR phase detector charge pump current selection

i2cRxSelectI2[7:4] = Register 35 [7:4]	**CDR phase detector charge pump current [μA]**
4'bxxxx	I = 0.375 × i2cRxSelectI2[7:4]

Table 68 Signal i2cRxControlOverride[C, B, A] function

i2cRxControlOverride[C, B, A] = Register 36 [2:0]	**I2C control**
3'b000	Disabled
3'b111	Enabled
Other combinations are invalid	n.a.

Table 69 Signal i2cRxSelectDataInPhase (valid only if i2cRxControlOverride[C, B, A] = 3'b111 or RXLOCKMODE [1:0] = 2'b00)

i2cRxSelectDataInPhase[C, B, A] = Register 36 [5:3]	**CDR data phase selection**
3'b000	Select data in anti-phase
3'b111	Select In phase data
Other combinations are invalid	n.a.

Table 70 Reference PLL charge-pump current selection

i2cRxSelectI1[7:4] = Register 41 [7:4]	**Reference PLL charge pump current [μA]**
4'bxxxx	I = 6.67 × i2cRxSelectI1[7:4]

Table 71 Signal i2cStartRace[C, B, A] (valid only if i2cRxControlOverride[C, B, A] = 3'b111 or RXLOCKMODE [1:0] = 2'b00)

i2cStartRace[C, B, A] = Register 42 [2:0]	**Race counters action**
3'b000	Clear "race" counters
3'b111	Enable "race" counters
Other combinations are invalid	n.a.

Table 72 Signal i2cRxDacEnable[C, B, A] function (valid only if i2cRxControlOverride[C, B, A] = 3'b111 or RXLOCKMODE [1:0] = 2'b00)

i2cRxDacEnable[C, B, A] = Register 42 [5:3]	**DAC state**
3'b000	disabled
3'b111	Enabled

Other combinations are invalid	n.a.

Table 73 Signal i2cRxDac [8:0] (valid only if i2cRxControlOverride[C, B, A] = 3'b111 or RXLOCKMODE [1:0] = 2'b00)

i2cRxDac [8:0] = {Register 44 [0], Register 43 [7:0]}	**DAC output voltage**
If rxDacGain = 1'b0	570 mV to 640 mV
If rxDacGain = 1'b1	590 mV to 710 mV

Table 74 Signal i2cRxFilterBypass[C, B, A] function (valid only if i2cRxControlOverride[C, B, A] = 3'b111 or RXLOCKMODE [1:0] = 2'b00)

i2cRxFilterBypass[C, B, A] = Register 44 [3:1]	**Low-Pass Filter bypass**
3'b000	Disabled
3'b111	Enabled
Other combinations are invalid	n.a.

Table 75 Signal i2cRxFilterEnable[C, B, A] function (valid only if i2cRxControlOverride[C, B, A] = 3'b111 or RXLOCKMODE [1:0] = 2'b00)

i2cRxFilterEnable[C, B, A] = Register 45 [2:0]	**Low-Pass Filter**
3'b000	Disabled
3'b111	Enabled
Other combinations are invalid	n.a.

Table 76 Signal i2cRxForceVfEqVc[C, B, A] function (valid only if i2cRxControlOverride[C, B, A] = 3'b111 or RXLOCKMODE [1:0] = 2'b00)

i2cRxForceVfEqVc[C, B, A] = Register 45 [5:3]	**V_{fine}**
3'b000	Under the CDR loop control
3'b111	Shorted to V_{coarse}
Other combinations are invalid	n.a.

Table 77 Race status bit "vcoFast"

vcoFast = Register 377 [1]	**Race status**
1'b0	VCO frequency low
1'b1	VCO frequency high

9.3.3 Frame Aligner

The CDR circuit (see 9.3.2) recovers the clock and the serial data at half-rate, that is, the recovered clock frequency is 2.4 GHz (*clock2400MHz*) and two data serial streams at 2.4 Gb/s (*dataPh0* and *dataPh180*) are recovered from the original 4.8 Gb/s data stream. The CDR circuit is followed by the *Frame Aligner* (FA) circuit (see Figure 20 for a block diagram) whose function is to resolve the 180 degree phase ambiguity of the recovered clock and to further de-serialize the two 2.4 Gb/s data streams recovered by the CDR circuit and assemble them into a single 120-bit word (frame).

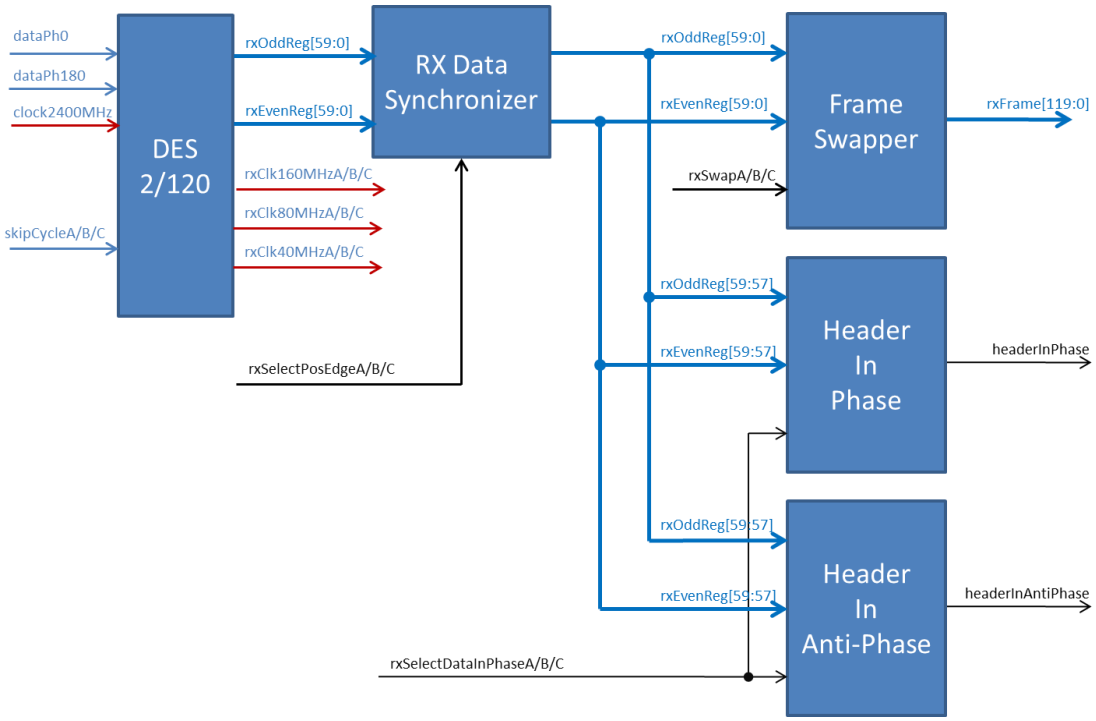


Figure 20 Frame Aligner circuit block diagram

As can be seen in Figure 20, FA circuit contains a *DESerializer* (DES 2/120) whose roles are:

1. To convert the two half rate serial streams into two 60-bit words.
2. To produce the lower clock frequencies needed to run the GBTX receiver (*rxClk160MHz[C, B, A]*, *rxClk80MHz[C, B, A]* and *rxClk40MHz[C, B, A]*).
3. To phase align the clock signals *rxClk40MHz[C, B, A]* with the phase of the header of the incoming GBT frame (see section 2.2 for information on the GBTX frame).

The phase of the *rxClk40MHz[C, B, A]* clocks can be rotated by pulsing the signals *skipCycle[C, B, A]*. This is done automatically under the control of the *Frame Aligner State Machine* (FASM). When the I2C control is enabled (*i2cRxControlOverride[C, B, A]* = 3'b111 or *RXLOCKMODE [1:0]* = 2'b00) the signals *skipCycle[C, B, A]* can be controlled through the signal i2cRxSkipCycle[C, B, A] = Register 40[2:0] see Table 78. Notice that each time these signal are pulsed high *i2cRxSkipCycle[C, B, A]* = 3'b111 a phase rotation step of 416.7 ps is done. To execute another phase rotation step the signals *i2cRxSkipCycle[C, B, A]* must be first set to 3'b000 and then to 3'b111.

Following the DES 2/120 circuit there is the *RX Data Synchronizer* that allows to choose the clock edge used to sample the data coming out of the DES 2/120 circuit. This circuit (implemented in standard cells) is on the boundary between the full custom circuits and standard cells circuits of the receiver. The sampling phase is controlled by the signals *RXselectPosEdge[C, B, A]* cording to Table 80 (see section 7.2.1 for more details). The signals *RXselectPosEdge[C, B, A]* are a configurations setting whose value will be pre-stored in the corresponding E-Fuses. For testing purposes these signals can be controlled through I2C. However, changing the default setting of these signals is not allowed for user operation.

Connected to the *RX data synchronizer* there are the *Frame Swapper* (FS) circuit, the *Header In Phase* (HIP) and the *Header In Anti-Phase (HIAP)* detector circuits. The *FS* circuit is controlled by the *Frame Aligner State Machine* and the *HIP* and *HIAP* circuits detect and inform the *FASM* of the phase of the recovered

clock in relation to the phase of the header in the received GBT frame. The *FS* circuit is controlled by the signals *rxSwap[C, B, A]* that are under the control of the *FASM*. When the I2C control is enabled (*i2cRxControlOverride[C, B, A]* = 3'b111 or *RXLOCKMODE [1:0]* = 2'b00) these signals are controlled through the I2C interface according to Table 79.

Table 78 Signal i2cRxSkipCycle[C, B, A] (valid only if i2cRxControlOverride[C, B, A] = 3'b111 or RXLOCKMODE [1:0] = 2'b00)

i2cRxSkipCycle[C, B, A] = Register 40 [2:0]	40 MHz RX clock
3'b000	Standby for the next phase shift
3'b111	Single step 416.7ps phase shift
Other combinations are invalid	n.a.

Table 79 Signal i2cRxSwap[C, B, A] (valid only if i2cRxControlOverride[C, B, A] = 3'b111 or RXLOCKMODE [1:0] = 2'b00)

i2cRxSwap[C, B, A] = Register 40 [5:3]	Frame assembly
3'b000	rxOddReg[59:0] mapped into odd bits of rxFrame[119:0] and rxEvenReg[59:0] mapped into even bits of rxFrame[119:0]
3'b111	rxOddReg[59:0] mapped into even bits of rxFrame[119:0] and rxEvenReg[59:0] mapped into odd bits of rxFrame[119:0]
Other combinations are invalid	n.a.

Table 80 *RXselectPosEdge[C, B, A]*

RXselectPosEdge[C, B, A] = Register 244 [2:0]	**Data sampling**
3'b000	On the falling edge
3'b111	On the rising edge
Other combinations are invalid	n.a.

9.3.3.1 Frame Aligner State Machine

The Frame Aligner State Machine (*FASM*) controls the locking process of GBTX receiver. Its state flow diagram is represented in Figure 21 and Figure 22. The states can be generically divided into search states and tracking states. The search states are entered at beginning of operation or when the FASM can't identify valid headers in the received frames. The tracking states keep track of frames with valid headers and, based on the locking phase, control the frame swapper for correct assembly of the frame. The FASM also controls the CDR clock phase to guarantee that the receiver displays constant latency every time the CDR circuit locks to the incoming serial data. The tracking states implement a "robust" algorithm to avoid that either transmission errors or single event upsets will unduly lead the FASM to enter the search states.

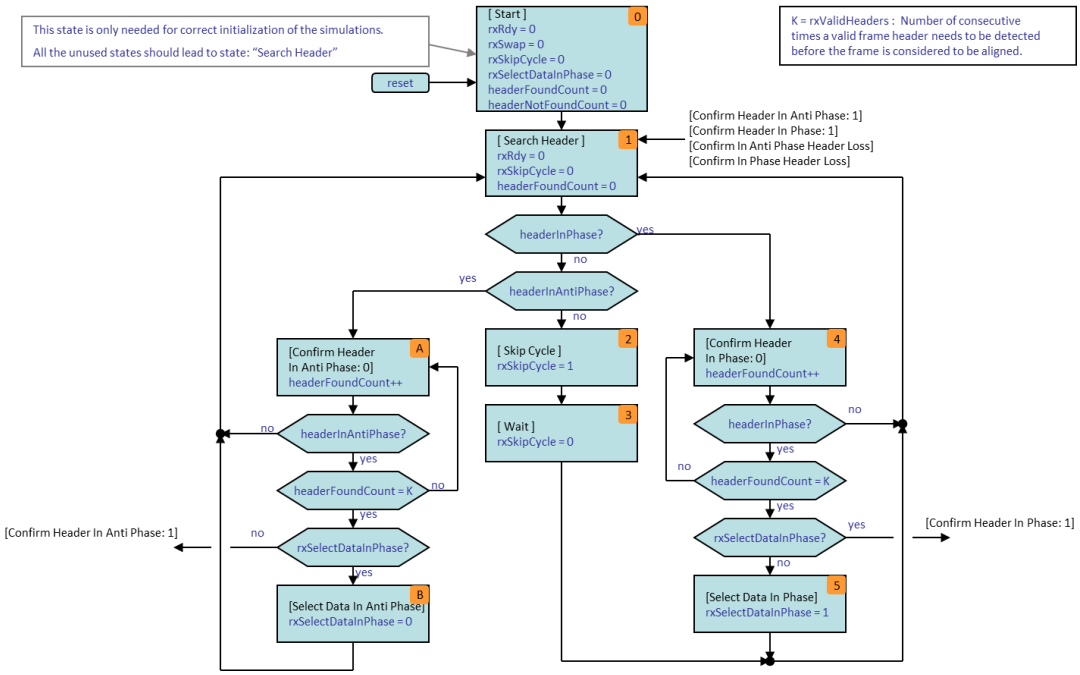


Figure 21 *Frame Aligner State Machine state flow diagram (1/2). See also* Figure 22*.*

When searching for the header, due to the random nature of the received data, it is likely that "valid frame headers" will be found even if the recovered clock phase (40 MHz) is not aligned with the frame header. To prevent false locks (frame lock), the FASM waits till it sees a certain number of consecutive frames detected with the same phase. Only if this condition is fulfilled does the FASM consider that frame lock has been achieved. The number of consecutive valid frame headers that have to be detected before frame lock is assumed is given by the contents of register:

Register 37[7:0]* = rxCtr2[7:2] = *rxValidHeaders[7:0]

When in the tracking states the FASM will "resist" being lead back to search states, that is, it will avoid that either transmission errors or SEUs will lead to the search states since the search operation is time consuming and will result in a relatively large dead time for the link. How quick the FASM will return to the search states is controlled by two registers:

Register 38[7:0]* = rxCtr3[7:2] = *rxMaxInvalidHeaders[7:0]
Register 39[7:0]* = rxCtr4[7:2] = *rxMinValidHeaders[7:0]

For the FASM, *rxMaxInvalidHeaders[7:0]* represents the maximum number of invalid headers (consecutive or not) that can be received before the frame is considered misaligned. If the number of detected invalid headers does not exceed the number stored in *rxMaxInvalidHeaders[7:0]* and if the number of consecutive detected valid headers exceeds the number stored in *rxMinValidHeaders[7:0]*, then the invalid frame count is reset. The default values for the three registers that control the FASM operation are given in tables: Table 81, Table 82 and Table 83.

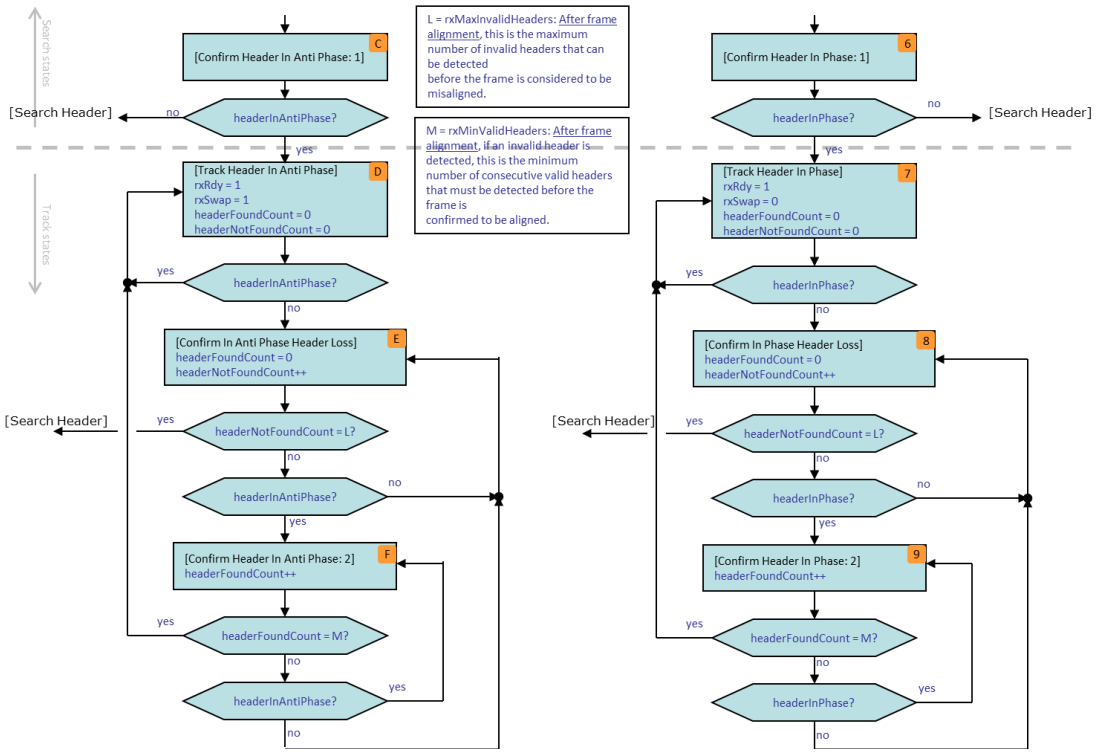


Figure 22 *Frame Aligner State Machine state flow diagram (2/2). See also* Figure 21.

Table 81 Default value for *Register 37[7:0] = rxValidHeaders[7:0]*

Register 37[7:0]* = rxCtr2[7:2] = *rxValidHeaders[7:0]	
8'hXXX...	Default value

Table 82 Default value for *Register 38[7:0] = rxMaxInvalidHeaders[7:0]*

Register 38[7:0]* = rxCtr3[7:2] = *rxMaxInvalidHeaders[7:0]	
8'hXXX...	Default value

Table 83 Default value for *Register 39[7:0] = rxMinValidHeaders[7:0]*

Register 39[7:0]* = rxCtr4[7:2] = *rxMinValidHeaders[7:0]	
8'hXXX...	Default value

9.3.4 Monitoring the Status of the GBTX Receiver

The GBTX receiver operation can be monitored by reading the contents of registers 378, 379, 380 and 432. Registers 378, 379 and 380 are TMR implementations of the same register. The meaning of the register bits is as follows:

- **Register 378[3:0] = Register 379[3:0] = Register 380[3:0] = fasmState[3:0]**. The number read back indicates the current FASM state number. In Figure 21 and Figure 22 the state number (hexadecimal) is indicated on the top right corner of each state "bubble".

- **Register 378[4] = Register 379[4] = Register 380[4] = rxSkipCycle**
- **Register 378[5] = Register 379[5] = Register 380[5] = rxSelectDataInPhase**
- **Register 378[6] = Register 379[6] = Register 380[6] = rxSwap**
- **Register 378[7] = Register 379[7] = Register 380[7] = rxReady**

9.3.5 Resetting the GBTX receiver

The GBTX receiver is reset at power-up or when the external reset is asserted. These are the two "standard" ways of resetting the receiver; the reset sequence follows the steps indicated section 10.1. For testing purposes it is also possible to reset the GBTX receiver under I2C control as detailed next.

9.3.5.1 Resetting the GBTX receiver through the I2C

The following registers can be used to reset the receiver through the I2C interface:

- **Register 41[2:0] = rxCtr6[2:0] = i2cRxControlReset[C, B, A]**
- **Register 50[5:3] = wdogCtr0[5:3] = i2cResetRx[C, B, A]**
- **Register 53[1:0] = wdogCtr3[1:0] = i2cRxResetA[1:0]**
- **Register 53[3:2] = wdogCtr3[3:2] = i2cRxResetB[1:0]**
- **Register 53[5:4] = wdogCtr3[5:4] = i2cRxResetC[1:0]**

Registers 41 and 50 (*i2cRxControlReset[C, B, A]* and *i2cResetRx[C, B, A]*) have virtually the same effect on the *RX Control State Machine* (RXCSM) however register 50 additionally controls the reset of the RX data path (see section 7.2). A reset action always requires a set and clear operation of a specific bit or group of bits in these registers. That is, when set to "1" the corresponding reset signal or signals will be maintained active until the bit or bits are cleared.

The reset signals originating on register 41 propagate directly to the RX control state machine while the ones originating in registers 50 and 53 propagate through the watchdog. The actions of the registers above depend on the state of the signals *i2cRxControlOverride[C, B, A]* and the state of the *Lock Control State Machine* (LCSM).

The signals *i2cRxControlReset[C, B, A]* always reset the LCSM and FASM state machines. The LCSM produces the CDR output reset signals *rxResetA/B/C[1:0]* (see Figure 19) based on the *Lock Mode* and the contents of Register 53.

9.3.5.1.1 i2cRxControlOverrideA/B/C = 1 (I2C takes control):

In this case, the reset signals of the CDR PLL and REF PLL *rxResetA/B/C[1:0]* are directly controlled by the signals i2cRxResetA/B/C[1:0]:

CDR PLL reset signal: *rxResetA/B/C[1] = i2cRxResetA/B/C[1]*;

REF PLL reset signal: *rxResetA/B/C[0] = i2cRxResetA/B/C[0]*.

The signals *i2cRxControlReset[C, B, A]* have no control on the CDR PLL and REF PLL reset signals but they still reset the LCSM and FASM state machines.

9.3.5.1.2 i2cRxControlOverrideA/B/C = 0 (RXCSM takes control)

The signals *i2cRxControlResetA/B/C* reset the LCSM and the FASM state machines and the CDR and REF PLLs according to the *Lock Mode*.

I2C frequency calibration *(RXLOCKMODE[1:0]* = 2'b00):

CDR PLL reset signal: *rxResetA/B/C[1] = i2cRxResetA/B/C[1]*;

REF PLL reset signal: *rxResetA/B/C[0] = i2cRxResetA/B/C[0]*.

Automatic DAC frequency calibration *(RXLOCKMODE[1:0]* = 2'b01):

In this case the signals *i2cRxResetA/B/C*[1:0] have no influence on the circuit operation:

The CDR PLL is always active, that is, it is never reset: *rxResetA/B/C*[1] = 1'b0 always.

The REF PLL is always disabled, that is, it is always reset: *rxResetA/B/C* [0] = 1'b1 always.

Automatic Reference PLL frequency calibration *(RXLOCKMODE[1:0]* = 2'b10):

During RESET:

The signals *i2cRxResetA/B/C*[1:0] have no influence;

The CDR PLL is reset: *rxResetA/B/C*[1] = 1'b1;

The REF PLL is reset: *rxResetA/B/C*[0] = 1'b1.

During frequency centring:

The signals *i2cRxResetA/B/C*[1:0] have no influence;

The CDR PLL is "passive", that is, it is under the control of the REF PLL and tracks the REF PLL: *rxResetA/B/C*[1] = 1'b0;

The REF PLL is active: *rxResetA/B/C*[0] = 1'b0.

During lock acquisition and lock:

The signals *i2cRxResetA/B/C*[1:0] have no influence;

The CDR PLL active: *rxResetA/B/C*[1] = 1'b0;

The REF PLL active: *rxResetA/B/C*[1] = 1'b0.

9.4 VCXO based PLL (xPLL)

The xPLL is an on chip *Phase-Locked Loop* (PLL) based on a *Voltage Controlled Quartz Crystal Oscillator* (VCXO). The main function of the xPLL is to provide a reference clock to the GBTX receiver, transmitter, start-up and watchdog logic. This clock can be generated by the xPLL by setting the xPLL in the Quartz *Crystal Oscillator* (XOSC) or the PLL mode.

When running the xPLL in the PLL mode an external 40.0789 MHz reference clock needs to be provided to the chip. This mode is advised when a reference clock is available in the system but its jitter characteristics are poor. In that case, the xPLL will lock to the reference clock and will produce a low jitter reference clock compatible with the requirements of the GBTX receiver and transmitter. If no system clock is available then the xPLL needs to be set in XOSC mode. In this mode the xPLL will generate the GBTX reference clock signal acting as a simple Quartz Crystal Oscillator (no external clock reference required). It is however important to note that in this case the clock provided by the VCXO is not synchronous with the system clock. In case a high performance reference clock is available in the system it is also possible to use it directly in the GBTX by disabling altogether the xPLL operation. This is done by setting the external signal *REFCLKSELECT* (see chapter 18 for pin number) according to the data in Table 84 and by selecting the ASIC reference clock according to Table 96.

Selection of the xPLL mode (PLL or XOSC) is done with bit 7 of registers 313, 314 and 315 as indicated in Table 85.

Table 84 Reference clock selection

REFCLKSELECT *(external signal)*	**GBTX reference clock source**
1'b0	XPLL (either XOSC or PLL mode)
1'b1	Reference clock input: REFCLKP/N

Table 85 XPLL mode selection

{Register 315[7], Register 314[7], Register 313[7] }= {xPllEnableC, xPllEnableB, xPllEnableA}	**XPLL Mode**
1'b0	XOSC
1'b1	PLL

The signals *REFCLKSELECT and xPllEnableA/B/C* are control inputs to the watchdog and power-up state machine and thus influence the start-up sequence and the watchdog operation. Please see chapter "GBTX Start-up and Watchdog" for details.

9.4.1 XPLL operation

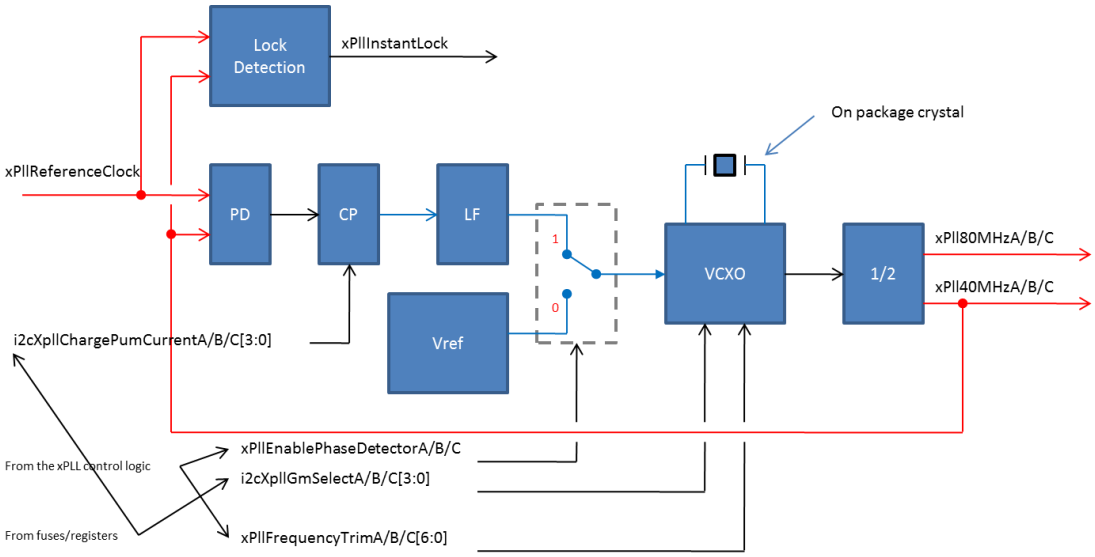


Figure 23 XPLL block diagram

The block diagram of the XPLL is shown in Figure 23. It is composed of the standard PLL building blocks: A bang-bang Phase Detector (PD), a Charge Pump (CP), a Loop Filter (LP), a Voltage Controlled Quartz Crystal Oscillator (VCXO) and a Clock Divider (1/2). The XPLL also contains a Lock detection circuit and a Voltage Reference (V_{ref}) circuit. When operating in the XOSC mode the voltage generated by V_{ref} drives the VCXO making it operate as a fixed frequency oscillator (XOSC). The voltage reference is also used during the calibration procedure when the XPLL is in the PLL mode.

9.4.2 Monitoring the XPLL operation

The *Lock Detection* (*LD*) circuit monitors the lock state of the XPLL and feeds this information to the *xPLL Control State Machine* (*XCSM*). The signal produced by the *LD* (*xPllInstantLock*) circuit gives a "coarse lock" indication that is further filtered by the *XCSM* to provide a reliable lock indication (signal *xPllLocked*),

eliminating locking glitches (during the lock process) and possible glitches (during normal operation) due to single event upsets. The lock state of the XPLL can be monitored by reading registers bits 4 and 7 of registers 427 and 428 respectively:

- **Register 428[7] = xPllInstantLock** ("raw" xPLL lock indication)
- **Register 427[4] = xPllLocked** (xPLL lock flag – processed by the *XCSM*)

9.4.2.1 Test Output

For testing purposes the signals *xPllInstantLock* and *xPllLocked* can be also monitored through the test output port "*TESTOUTPUT*" (see chapter 18 for pin number) according to Table 86).

Table 86 Setting the *TESTOUT* port to monitor *xPllInstantLock* and *xPllLocked*

Register 280[7:0] = testOutputSelect[7:0]	***TESTOUTPUT***
8'd24	*xPllInstantLock*
8'd29	*xPllLocked*
See section 16.3 for a complete list of signals available through the *TESTOUTPUT* port.	

9.4.3 Setting the XPLL operation modes

As described above the xPLL can be bypassed, act as a crystal oscillator or as a PLL. How to set the different modes is described in the following sections.

9.4.3.1 Bypassing the XPLL

Bypassing the xPLL means that an external clock is directly used as the reference clock for the receiver and transmitter sections of the GBTX. The external clock is also used as the master clock for the start-up and watchdog logic. To bypass the xPLL, the GBTX must be setup as follows:

- Provide a 40.0786 MHz clock to the differential clock reference input REFCLKP / REFCLKN. This input accepts LVDS and SLVS levels and it is internally terminated to 100 Ω (differential).
- Set the GBTX input *REFCLKSELECT* = 1'b1 (seeTable 84). This is a CMOS input with 1'b1 = 1.5V
- Set clock manager register 281 as:
  - Register 281[1:0] = cmReferenceClockSelectB[1:0] = 2'b00
  - Register 281[3:2] = cmReferenceClockSelectB[1:0] = 2'b00
  - Register 281[5:4] = cmReferenceClockSelectC[1:0] = 2'b00
- Set the following xPLL registers as:
  - Register 235[3:0] = xPllChargePumpCurrentA[3:0] = 4'b0000
  - Register 235[7:4] = xPllChargePumpCurrentB[3:0] = 4'b0000
  - Register 236[3:0] = xPllChargePumpCurrentC[3:0] = 4'b0000
  - Register 313[6:0] = xPllFrequencyTrimA[6:0] = 7'bxxxxxxx
  - Register 313[7] = xPllEnableA = 1'b0
  - Register 314[6:0] = xPllFrequencyTrimB[6:0] = 7'bxxxxxxx
  - Register 314[7] = xPllEnableB = 1'b0
  - Register 315[6:0] = xPllFrequencyTrimC[6:0] = 7'bxxxxxxx
  - Register 315[7] = xPllEnableC = 1'b0

- Register 316[3:0] = xPllGmSelectA[3:0] = 4'b0000
- Register 316[7:4] = xPllGmSelectB[3:0] = 4'b0000
- Register 317[3:0] = xPllGmSelectC[3:0] = 4'b0000
- Register 317[4] = xPllEnablePhaseDetectorA = 1'bx
- Register 317[5] = xPllEnablePhaseDetectorB = 1'bx
- Register 317[6] = xPllEnablePhaseDetectorC = 1'bx
- Register 318[0] = xPllControlOverrideA = 1'b0
- Register 318[1] = xPllControlOverrideB = 1'b0
- Register 318[2] = xPllControlOverrideC = 1'b0
- Register 318[3] = xPllEnableAutoRestartA = 1'bx
- Register 318[4] = xPllEnableAutoRestartB = 1'bx
- Register 318[5] = xPllEnableAutoRestartC = 1'bx

Please see 10.1 for details on the start-up sequence under these conditions. Note that in this mode the contents of registers 313[6:0], 314[6:0], 315[6:0], 317[6:4] and 318[5:3] are ignored. Registers 235, 236[7:4], 316 and 317[3:0] although not influencing the behaviour of the chip in this mode should be set as indicated to reduce the power consumption.

9.4.3.2 Quartz Crystal Oscillator (XOSC) Mode

In the *XOSC* mode the GBTX generates the reference clock internally and an external reference clock is thus not required. The internal clock is used as a reference for the receiver and transmitter and as the master clock for the watchdog and start-up state machines. There are two methods of setting up the XOSC mode. The first is slightly simpler to setup but doesn't allow trimming of the oscillator frequency (set to its lower oscillation frequency). The second requires a few more registers to be controlled but allows the user to trim the oscillator frequency.

9.4.3.2.1 Setting up the XOSC mode, method 1:

- Set the GBTX input *REFCLKSELECT* = 1'b0 (see Table 84). This is a CMOS input with 1'b0 = 0.0V
- Set the following registers as:
  - Register 235[3:0] = xPllChargePumpCurrentA[3:0] = 4'b0000
  - Register 235[7:4] = xPllChargePumpCurrentB[3:0] = 4'b0000
  - Register 236[3:0] = xPllChargePumpCurrentC[3:0] = 4'b0000
  - Register 313[6:0] = xPllFrequencyTrimA[6:0] = 7'b1011101 (default value)
  - Register 313[7] = xPllEnableA = 1'b0
  - Register 314[6:0] = xPllFrequencyTrimB[6:0] = 7'b1011101 (default value)
  - Register 314[7] = xPllEnableB = 1'b0
  - Register 315[6:0] = xPllFrequencyTrimC[6:0] = 7'b1011101 (default value)
  - Register 315[7] = xPllEnableC = 1'b0
  - Register 316[3:0] = xPllGmSelectA[3:0] = 4'b1010
  - Register 316[7:4] = xPllGmSelectB[3:0] = 4'b1010
  - Register 317[3:0] = xPllGmSelectC[3:0] = 4'b1010
  - Register 317[4] = xPllEnablePhaseDetectorA = 1'bx
  - Register 317[5] = xPllEnablePhaseDetectorB = 1'bx
  - Register 317[6] = xPllEnablePhaseDetectorC = 1'bx

- Register 318[0] = xPllControlOverrideA = 1'b0
- Register 318[1] = xPllControlOverrideB = 1'b0
- Register 318[2] = xPllControlOverrideC = 1'b0
- Register 318[3] = xPllEnableAutoRestartA = 1'bx
- Register 318[4] = xPllEnableAutoRestartB = 1'bx
- Register 318[5] = xPllEnableAutoRestartC = 1'bx

Note that in this mode the contents of registers 313, 314 and 315 will be ignored and the oscillator will always run at its lower frequency. If frequency trimming is required then the method that follows must be used.

9.4.3.2.2 Setting up the XOSC mode, method 2 (recommended):

- Set the GBTX input *REFCLKSELECT* = 1'b0 (see Table 84). This is a CMOS input with 1'b0 = 0.0V
- Set the following registers as:
  - Register 235[3:0] = xPllChargePumpCurrentA[3:0] = 4'b0000
  - Register 235[7:4] = xPllChargePumpCurrentB[3:0] = 4'b0000
  - Register 236[3:0] = xPllChargePumpCurrentC[3:0] = 4'b0000
  - Register 313[6:0] = xPllFrequencyTrimA[6:0] = 7'b1011101 (default value)
  - Register 313[7] = xPllEnableA = 1'b0
  - Register 314[6:0] = xPllFrequencyTrimB[6:0] = 7'b1011101 (default value)
  - Register 314[7] = xPllEnableB = 1'b0
  - Register 315[6:0] = xPllFrequencyTrimC[6:0] = 7'b1011101 (default value)
  - Register 315[7] = xPllEnableC = 1'b0
  - Register 316[3:0] = xPllGmSelectA[3:0] = 4'b1010
  - Register 316[7:4] = xPllGmSelectB[3:0] = 4'b1010
  - Register 317[3:0] = xPllGmSelectC[3:0] = 4'b1010
  - Register 317[4] = xPllEnablePhaseDetectorA = 1'b0
  - Register 317[5] = xPllEnablePhaseDetectorB = 1'b0
  - Register 317[6] = xPllEnablePhaseDetectorC = 1'b0
  - Register 318[0] = xPllControlOverrideA = 1'b1
  - Register 318[1] = xPllControlOverrideB = 1'b1
  - Register 318[2] = xPllControlOverrideC = 1'b1
  - Register 318[3] = xPllEnableAutoRestartA = 1'bx
  - Register 318[4] = xPllEnableAutoRestartB = 1'bx
  - Register 318[5] = xPllEnableAutoRestartC = 1'bx

The contents of registers 313[6:0], 314[6:0], 315[6:0], 316[7:0] and 317[3:0] are for a "typical" crystal. The e-fuses will be programed to contain a value tailored to the particular crystal mounted with the GBTX.

9.4.3.3 PLL modes

In the PLL mode the xPLL locks to an external reference clock 40.0789 MHz and produces the reference clocks for the transmitter and receiver as well as the master clocks for the start-up and watchdog logic. The interest of this mode is that it allows a relatively poor quality system clock to be used as a reference for the GBTX.

In the PLL mode the xPLL can be programed to do a frequency calibration cycle every time the locking procedure is started or only upon an external or watch-

dog reset. This feature is controlled by the signals *xPllEnableAutoRestartA/B/C*. In any of the PPL modes, the *xPll Control State Machine* (*XCSM*) automatically choses the values of the *xPllFrequencyTrimA/B/C[6:0]* signals that best centre the VCXO frequency range around the reference clock frequency.

Signals *xPllEnableAutoRestartA/B/C* control the operation of *XCSM enabling or not a frequency calibration cycle every time the PLL loses lock.*

To setup this mode:

- Provide a 40.0786 MHz clock to the differential clock reference input REFCLKP / REFCLKN. This input accepts LVDS and SLVS levels and it is internally terminated to 100 Ω (differential).
- Set the GBTX input *REFCLKSELECT* = 1'b1 (see Table 84). This is a CMOS input with 1'b1 = 1.5V
- Set the following registers as:
    - Register 235[3:0] = xPllChargePumpCurrentA[3:0] = 4'b????
    - Register 235[7:4] = xPllChargePumpCurrentB[3:0] = 4'b????
    - Register 236[3:0] = xPllChargePumpCurrentC[3:0] = 4'b????
    - xPllFrequencyTrimA[6:0] = 7'b1011101 (default value)
    - Register 313[7] = xPllEnableA = 1'b1
    - Register 314[6:0] = xPllFrequencyTrimB[6:0] = 7'b1011101 (default value)
    - Register 314[7] = xPllEnableB = 1'b1
    - Register 315[6:0] = xPllFrequencyTrimC[6:0] = 7'b1011101 (default value)
    - Register 313[7] = xPllEnableA = 1'b1
    - Register 314[6:0] = xPllFrequencyTrimB[6:0] = 7'b1011101 (default value)
    - Register 314[7] = xPllEnableB = 1'b1
    - Register 315[6:0] = xPllFrequencyTrimC[6:0] = 7'b1011101 (default value)
    - Register 315[7] = xPllEnableC = 1'b1
    - Register 316[3:0] = xPllGmSelectA[3:0] = 4'b1010
    - Register 316[7:4] = xPllGmSelectB[3:0] = 4'b1010
    - Register 317[3:0] = xPllGmSelectC[3:0] = 4'b1010
    - Register 317[4] = xPllEnablePhaseDetectorA = 1'bx
    - Register 317[5] = xPllEnablePhaseDetectorB = 1'bx
    - Register 317[6] = xPllEnablePhaseDetectorC = 1'bx
    - Register 318[0] = xPllControlOverrideA = 1'b0
    - Register 318[1] = xPllControlOverrideB = 1'b0
    - Register 318[2] = xPllControlOverrideC = 1'b0
    - For automatic frequency calibration on relock (recommended) set:
        - Register 318[3] = xPllEnableAutoRestartA = 1'b1
        - Register 318[4] = xPllEnableAutoRestartB = 1'b1
        - Register 318[5] = xPllEnableAutoRestartC = 1'b1
    - For frequency calibration on reset only set:
        - Register 318[3] = xPllEnableAutoRestartA = 1'b0
        - Register 318[4] = xPllEnableAutoRestartB = 1'b0
        - Register 318[5] = xPllEnableAutoRestartC = 1'b0

9.4.3.4 PLL settings

9.4.3.5 Controlling the xPLL through the I2C interface

9.4.3.6 Crystal specifications

The xPLL uses as a resonator a quartz crystal that is mounted on the ASIC package. The crystal is manufacture by Micro Crystal Switzerland and has dimensions 3.5 mm × 2.2 mm × 0.8 mm as shown in Figure 24

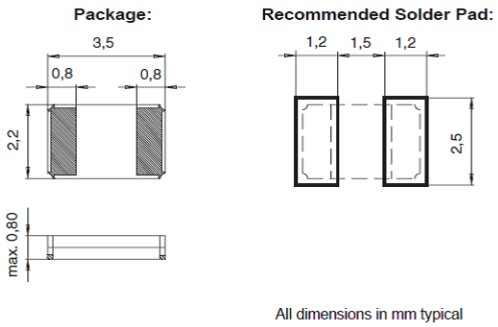


Figure 24 Quartz crystal package dimensions

The Quartz crystal characteristics are given in

Pos	Description	Symbol	Typ.	Min.	Max	Unit
1	Load Frequency[1]	FL	80.157372			MHz
2	Load Capacitance[2]	CL	9.2			pF
3	Frequency Tolerance at 25°C	ΔFL/FL		-18	18	ppm
4	Motional Capacitance	C1		8.0		fF
5	Static Capacitance	Co			2.8	pF
6	Drive Level	P	100		500	μW
7	**O**perating **T**emperature **R**ange	OTR		-20	60	°C
8	Series Resistance over OTR	Rs			30	Ω
9	Frequency Tolerance over OTR	ΔFL/FL		-10	10	ppm
10	Aging first year	ΔFL/FL			±3	ppm

Notes:

1 This spec needs a 100% frequency verification over temperature range (5 °C intervals)

2 Value to be confirmed

9.5 ePorts Phase-Locked Loop (ePLL)

The GBTX contains two 320 MHz Phase Lock Loops (ePLL) that are dedicated to generate the 160 and 320 MHz clock signals required by the ePorts when operating at data rates of 160 and 320 Mb/s, respectively. One ePLL is associated with the GBTX receiver while the other one with the transmitter.

9.5.1 e-PLL Operation

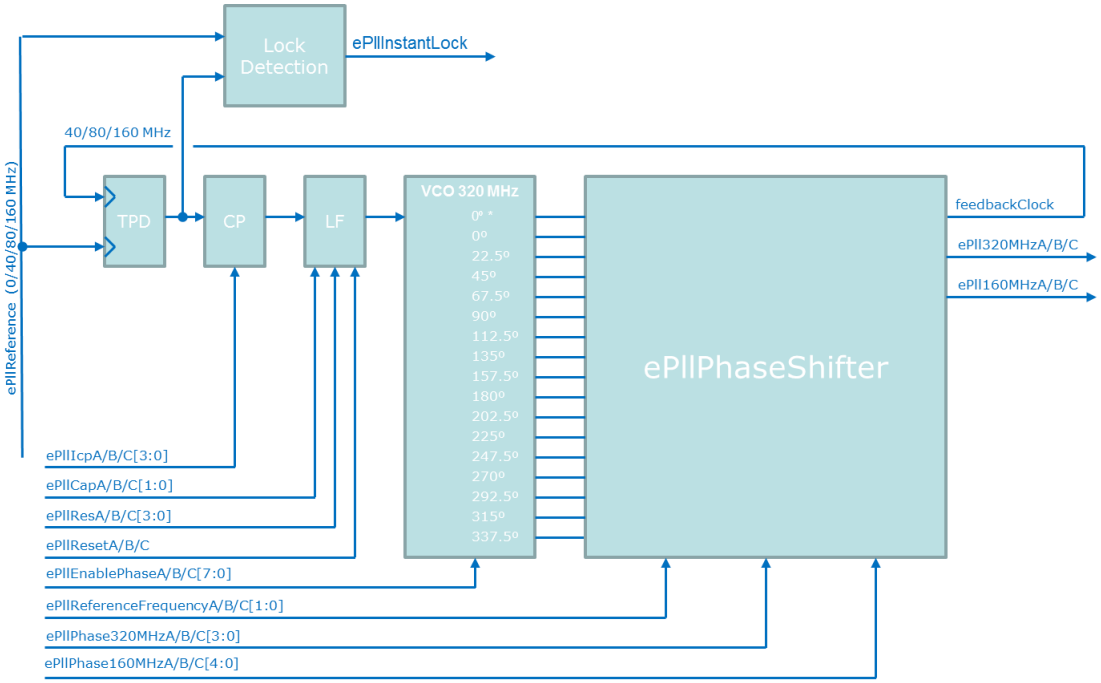


Figure 25 ePLL block diagram

The block diagram of the ePLLs is shown in Figure 25. The ePLL is composed of the usual functional elements: three-state phase-detector (TPD), charge-pump (CP), loop-filter (LF), and a voltage controlled oscillator (VCO). Besides those the ePLLs include also a phase-shifter and a lock detection circuit. The charge-pump and the loop-filter are programable in order to optimize the performance of the PLL.

9.5.1.1 ePLL reference clock

The reference clock of the ePLLs can be set to 40, 80 or 160 MHz or be disabled altogether (in case the 160 and 320 Mb/s data rates are not used). For normal operation the ePLLs use the 160 MHz clocks produced by the GBTX SER or CDR circuits for the transmitter and receiver ePLL, respectively. The other possible reference frequencies are used only for tests purposes.

How to select the ePLL reference clock is explained in detail in section 11.4

9.5.1.1.1 ePLL reference clock frequency

The feedback dividers of the ePLLs have to be set according to the reference clock frequency (see section 11.4 for information on the ePLL reference clock source selection). Setup of the feedback dividers ratio is made through registers 242 and 243 for the RX and TX ePLLs as indicated in Table 87 and Table 88.

Table 87 Rx ePLL reference clock frequency

Reg242 [1:0] = inEportCtr180 [1:0] = ePllRxReferenceFreqA[1:0] **Reg242 [3:2] = inEportCtr180 [3:2] = ePllRxReferenceFreqB[1:0]** **Reg242 [5:4] = inEportCtr180 [5:4] = ePllRxReferenceFreqC[1:0]**	***Rx ePLL reference clock frequency***

ePllRxReferenceFreq C/B/A[1:0] = 00	*0*
ePllRxReferenceFreq C/B/A[1:0] = 01	*40 MHz*
ePllRxReferenceFreq C/B/A[1:0] = 10	*80 MHz*
ePllRxReferenceFreq C/B/A[1:0] = 11	*160 MHz*

Table 88 Tx ePLL reference clock frequency

Reg243 [1:0] = inEportCtr181 [1:0] = ePllTxReferenceFreqA[1:0] **Reg243 [3:2] = inEportCtr181 [3:2] = ePllTxReferenceFreqB[1:0]** **Reg243 [5:4] = inEportCtr181 [5:4] = ePllTxReferenceFreqC[1:0]**	***Tx ePLL reference clock frequency***
ePllTxReferenceFreq C/B/A[1:0] = 00	*0*
ePllTxReferenceFreq C/B/A[1:0] = 01	*40 MHz*
ePllTxReferenceFreq C/B/A[1:0] = 10	*80 MHz*
ePllTxReferenceFreq C/B/A[1:0] = 11	*160 MHz*

9.5.1 160 and 320 MHz internal clock phases

The 160 and 320 MHz (internal) clock phases are adjusted in the transmitter and receiver ePLLs. This is made to ensure the correctness of the internal timing. **The correct values for all internal clock phases will be provided to the users and the users are not allowed to modify them at the risk of malfunctioning of the device.**

The phases of the 160 and 320 MHz clocks can be phase shifted up to 360 degrees in steps of 195.31 ps according to the phase control words given in the four tables below. To obtain the phase of the clock signal in question multiply the binary number written in the control register by 195.31 ps. Notice that for each clock signal the triplicated numbers (A/B/C) must all be written with the same value.

Table 89 Tx 320 MHz clock phase control registers

Register	**Name**	**Signal**
291 [3:0]	ckCtr22 [3:0]	ePllTxPhase320MHzA[3:0]
291 [7:4]	ckCtr22 [7:4]	ePllTxPhase320MHzB[3:0]
292 [3:0]	ckCtr23 [3:0]	ePllTxPhase320MHzC[3:0]

Table 90 Tx 160 MHz clock phase control registers

Register	**Name**	**Signal**
296 [4:0]	ckCtr27 [4:0]	ePllTxPhase160MHzA[4:0]
297 [4:0]	ckCtr28 [4:0]	ePllTxPhase160MHzB[4:0]
298 [4:0]	ckCtr29 [4:0]	ePllTxPhase160MHzC[4:0]

Table 91 Rx 320 MHz clock phase control registers

Register	Name	Signal
302 [3:0]	ckCtr33 [3:0]	ePllRxPhase320MHzA[3:0]
302 [7:4]	ckCtr33 [7:4]	ePllRxPhase320MHzB[3:0]
292 [7:4]	ckCtr23 [7:4]	ePllRxPhase320MHzC[3:0]

Table 92 Rx 160 MHz clock phase control registers

Register	Name	Signal
307 [4:0]	ckCtr38 [4:0]	ePllRxPhase160MHzA[4:0]
308 [4:0]	ckCtr39 [4:0]	ePllRxPhase160MHzB[4:0]
309 [4:0]	ckCtr40 [4:0]	ePllRxPhase160MHzC[4:0]

9.5.1.1 Controlling the loop dynamics

The PLL dynamic behaviour can be programed by controlling the value of the charge-pump

9.5.2 ePLL control state machine

This is a state-machine which monitors the status of the ePLL. There is one block for ePLL-RX and one for ePLL-TX. The locked status will be asserted only when the ePLL stays locked for more than 800 consecutive cycles of the reference clock. The locked status will be de-asserted only when the ePLL stays unlocked for more than 32 consecutive cycles of the reference clock.

The state machine is automatically reset at power-up and by the external reset. It can also be reset by setting bits in a configuration register via I2C as shown in Table 93. These are the same resets used for the ePLLs themselves.

Table 93 ePLL reset control

Parameter	Register	Bits	Function
ePllTxReset[C, B, A]	303 = ckCtr34	[2:0]	Resets ePLL-TX control
ePllRxReset[C, B, A]	303 = ckCtr34	[6:4]	Resets ePLL-RX control

The block contains a loss-of-lock counter that can be read through I2c or SC-IC. It will increment when the state-machine de-asserts the locked status. This counter is not resettable.

10. GBTX START-UP AND WATCHDOG

When the GBTX is powered or the external reset is asserted, the chip will run an automatic configuration sequence. This is controlled by a finite-state machine (FSM), which issues resets to various blocks and monitors the state of the blocks until the complete chip is ready for operation. According to the choice of mode, the FSM will ignore unused blocks. When the GBTX is ready for operation, the FSM will assert the txRdy and rxRdy outputs.

The FSM is synchronous with the reference clock of the GBTX, selected by the external signal *REFCLKSELECT*.

When the user chooses to configure the GBTX from its fuse values (see Chapter 15), the sampling of the fuse values is controlled by the FSM.

Watchdog monitoring can also be used in the GBTX. This will automatically reset a particular block if it malfunctions.

To allow debugging, the automatic changing of states can be disabled and the FSM stepped through its various states under control of the I2C interface.

10.1 Recommendations for powering sequence of the GBTX

The start-up of the GBTX relies on the power being stable when the sequence starts. This depends on the ramp-up time of the power supply, for example a DC-DC convertor or linear regulator. If the power is not stable when the sequence starts, this can cause incorrect sampling of the fuse values and hence the wrong configuration of the chip. The following steps should be taken to protect against this problem:

1. If a fail-safe mechanism of providing a reset to the GBTX is available, this should be used by connecting it to the RESETB input.

2. If the GBTX has no external clock and uses the XPLL to generate its reference clock (XOSC mode, as described in Chapter 9.4 with REFCLKSELECT=0), then the FSM will pass through a 300 us wait-state. In some cases, these 300 us will be sufficient for the power supply to stabilise before the fuse sampling starts. As an extra measure, the user should implement the passive circuit described below.

3. In all scenarios, the user should implement a passive resistor-capacitor network connected to the RESETB input of the GBTX. The resistor should be connected between RESETB and VDDIO, and the capacitor between RESETB and GND. The values of R and C depend on the ramp-up time of the supply. As an example, for the FEAST-MP DC-DC convertor typically used to power the GBTX, values of R = 5 kΩ and C = 100nF are found to be ideal.

4. If the FEAST-MP DC-DC convertor is used, the 'power-good' output of this device can be used as a power-on-reset signal to the GBTX directly connected to its RESETB input. The 'power-good' output of the FEAST-MP is an open-drain configuration that requires a pull-up resistor to the output voltage of the convertor. This has been tested with one FEAST-MP powering one GBTX, but has not been tested with multiple GBTXs powered from the same FEAST-MP. The user should verify in their system that this 'power-good' signal does not introduce spurious glitches that would reset the GBTX.

The 'power-good' signal can be used together with a passive resistor-capacitor network as described above.

5. If the GBTX is operated with an external clock (REFCLKSELECT=1), the user should enable this clock only after the power is switched on and has stabilised.

6. The timeOut function should be enabled.

10.2 Reseting the ASIC

It is very important for the reset signal (RESETB) to be only released after the supply voltage has reached atleast 90% of its final value. The minimum duration of the reset pulse (RESETB = 0 V) is 1 μs. If an RC network is to be used to time the RESETB signal, the time constant used must be such that when the supply reaches its final value the RESETB signal has only reached (at maximum) 10% of the supply value (and the RESET pulse duration should never be smaller than 1 μs).

10.3 Power-up state FSM

The FSM is shown in Figure 26. The states are colour-coded according to the sub-blocks they control or monitor. The functions of each state are explained in Table 94 together with the input signals that control the state transitions.

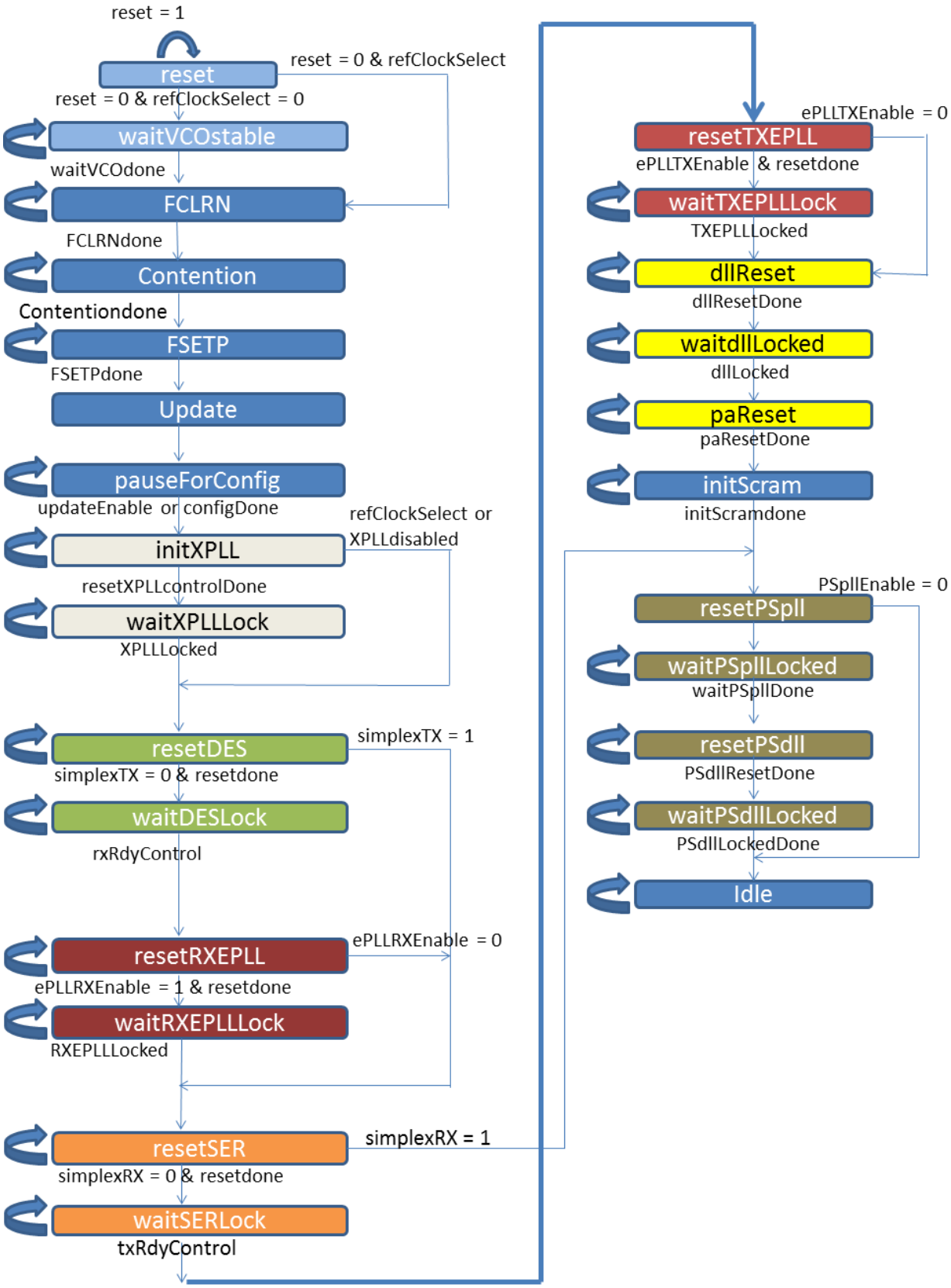


Figure 26 Power-up State machine flow diagram

Table 94 Function of the start-up STM state

State	State ID 5 bits	Function	Notes
reset	00000	FSM reset state.	Stays in state for duration of power-on-reset or if external reset is asserted.
waitVCOstable	11001	Waits for XPLL VCO to stabilise.	Fixed duration = 12,000 cycles (300us). Bypassed if **REFCLKSELECT** = 1.
FCLRN	00001	Clear the fuse sample latches.	Fixed duration = 2 cycles (50ns)
Contention	00010	Start fuse sampling	Fixed duration = 2 cycles (50ns)
FSETP	00011	Complete fuse sampling	Fixed duration = 2 cycles (50ns)
Update	00100	Transfer fuse values into registers	Transfer executed only if **updateEnable** is high.
pauseForConfig	00101	Pause state foreseen for I2C access of registers	Exits state immediately if **updateEnable** is high. Otherwise, waits until **configDone** is asserted.
initXPLL	00110	Reset XPLL control logic.	Fixed duration = 2 cycles (50ns)
waitXPLLLock	00111	Waits for XPLL to lock.	Exits state when XPLLLocked is asserted. State bypassed if **REFCLKSELECT** or **XPLLdisabled** is high.
resetDES	01000 01001	Resets deserialiser	Fixed duration = 5 cycles (125ns). Exits state immediately if **simplexTX** is asserted.
waitDESLock	01010	Waits for deserialiser to lock	Exits state when **rxRdyControl** is asserted. State bypassed if **simplexTX** is asserted.
resetRXEPLL	01011 01100	Resets ePLL-RX	Fixed duration = 4 cycles (100ns). Exits state immediately if **ePLLRXEnable** is low. State bypassed if **simplexTX** is asserted.
waitRXEPLLLock	01101	Waits for ePLL-RX to lock	Exits state when **RXEPLLLocked** is asserted. State bypassed if **simplexTX** is asserted or **ePLLRXEnable** is low.
resetSER	01110 01111	Resets serialiser	Fixed duration = 5 cycles (125ns). Exits state immediately if **simplexRX** is asserted.
waitSERLock	10000	Waits for the serialiser to lock	Exits state when **txRdyControl** is asserted. State bypassed if **simplexRX** is asserted.
resetTXEPLL	10001 10010	Resets ePLL-TX	Fixed duration = 4 cycles (100ns). Exits state immediately if **ePLLTXEnable** is low. State bypassed if **simplexRX** is asserted.
waitTXEPLLLock	10011	Waits for ePLL-TX to lock	Exits state when **TXEPLLLocked** is asserted. State bypassed if **simplexRX** is asserted or **ePLLTXEnable** is low.
dllReset	10100	Resets dlls in EportRX	Fixed duration = 4 cycles (100ns). State bypassed if **simplexRX** is asserted.
waitdllLocked	10101	Waits for dlls in EportRX to lock	Exits state when **dllLocked** is asserted. State bypassed if **simplexRX** is asserted.
paReset	10110	Resets phase-aligner control logic in EportRX	Fixed duration = 4 cycles (100ns). State bypassed if **simplexRX** is asserted.
initScram	10111	Initialises scrambler in TX logic	Fixed duration = 2 cycles (50ns). State bypassed if **simplexRX** is asserted.
resetPSpll	11010 11011	Resets PLL in phase shifter	Fixed duration = 1 cycle (25ns). Immediately goes to Idle state if **PSpllEnable** is low.
waitPSpllLocked	11100	Waits for PLL in phase shifter to lock	Fixed duration = 200 cycles (5us). State bypassed if **PSpllEnable** is low.
resetPSdll	11101	Resets DLL in phase shifter	Fixed duration = 240 cycles (6us). State bypassed if **PSpllEnable** is low.
waitPSdllLocked	11110	Waits for DLL in phase shifter to lock	Fixed duration = 2000 cycles (50us). State bypassed if **PSpllEnable** is low.
Idle	11000	GBTX is operational	txRdy and rxRdy are asserted (depending on mode)

Signal	User selectable?	Function and Source	Register to address
REFCLKSELECT	Yes	Selects reference clock for GBTX (0=XPLL, 1=external clock)	-
updateEnable	Yes	Enables automatic transfer of fuse values to registers. updateEnable is high if the fuses configFuse[2:0] have been burned (value = 111).	Efuse 366; bits[2:0]
configDone	Yes	Signals that the user has finished an I2C access and the FSM should continue. configDone is asserted by writing 8'hAA into the register.	Register 365; bits [7:0]
XPLLLocked	No	Indicates that the XPLL is locked. Generated by the XPLL control block.	-
XPLLdisabled	Yes	Indicates that the XPLL is not used as a PLL and is running in VCO mode. Asserted if xpllEnable[C, B, A]=0.	Register 313; bit[7] Register 314; bit[7] Register 315; bit[7]
simplexTX	Yes	Indicates that the GBTX is in simplex transmitter mode. Asserted if the external mode bits[1:0] = 00.	-
rxRdyControl	No	Indicates that the serialiser is operational. Generated by the RX control block.	-
ePLLRXEnable	Yes	Indicates that the ePLL-RX is in use. Asserted if any of the phases of the ePLL-RX are enabled.	Register 304; bits[7:0] Register 305; bits[7:0] Register 306; bits[7:0]
RXEPLLLocked	No	Indicates that the ePLL-RX is operational. Generated by the ePLL-RX control block.	-
simplexRX	Yes	Indicates that the GBTX is in simplex receiver mode. Asserted if the external mode bits[1:0] = 01.	-
txRdyControl	No	Indicates that the deserialiser is operational. Generated by the TX control block.	-
ePLLTXEnable	Yes	Indicates that the ePLL-TX is in use. Asserted if any of the phases of the ePLL-RX are enabled.	Register 293; bits[7:0] Register 294; bits[7:0] Register 295; bits[7:0]
TXEPLLLocked	No	Indicates that the ePLL-TX is operational. Generated by the ePLL-TX control block.	-
dllLocked	No	Asserted if the dlls in all enabled EportRX groups are locked	-
PSpllEnable	Yes	Indicates that the phase-shifter pll is in use. Asserted if PSpllEnable[C, B, A]=1	Register 52; bits[2:0]

10.4 Pausing the FSM

The pauseForConfig state is foreseen for initial testing of the GBTX when optimal registers settings are not yet known and the Efuses have not been burned. If updateEnable is not asserted, the FSM will wait in this state until configDone is asserted. While in this state, the user can use the I2C interface to write values to the registers, and when this is finished the configDone bits should be set by writing to the register described in Table 153. The FSM then will then continue the rest of the sequence.

Note that the configDone bits are reset to zero at power-on or when the external reset is asserted.

10.5 Watchdog operation

When enabled, this will monitor the state of each sub-block. If any sub-block stops operating correctly (for example a PLL loses lock), the watchdog will force the FSM to return to the reset state of that sub-block and the sequence will continue from that point until normal operating conditions are achieved.

For example, if the FSM is in the Idle state and the ePLL-TX loses lock (TXEPLLLocked goes low), the FSM will jump back to the resetTXEPLL state and re-start the sequence from there.

The intended use of the watchdog is to allow the chip to automatically recover from a functional interruption (such as a fibre disconnect) without the need of power-cycling or resetting. It does however have an impact on the number of data errors caused by a single-event-upset, as discussed below.\

State transitions triggered by the watchdog are shown in the figure below.

The watchdog is enabled by asserting the following bits:

Register 50 (wdogCtr0); bits[2:0] = 111.

10.6 AutoReset feature (GBTX version 2 only)

The user can turn on the autoReset feature to ensure that the FSM is correctly initialised before executing the full start-up procedure.

The autoReset feature is enabled by asserting the external signal autoReset/SLVS-Test-Inject.

10.7 TimeOut feature (GBTX version 2 only)

Some of the states of the FSM wait for a particular circuit to lock. In some cases, this locking may not occur (for example, if the downlink optical fibre is unplugged when the GBTX is powered-on then the deserialiser will not lock). To resolve problems like this, the 'wait-states' have time-outs (set at 2 seconds). If this time is exceeded the FSM moves back to the reset state and re-starts the full start-up procedure. In the example, the FSM will continue this time-out loop until the fibre is plugged in and the deserialiser can lock.

The timeOut is applied to the following states:

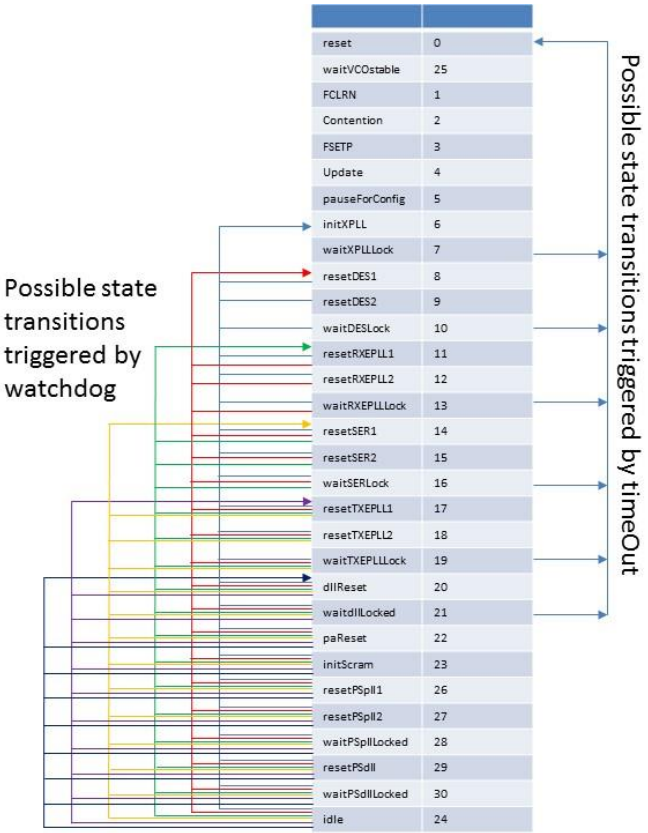
waitXPLLLock, waitDESLock, waitRXEPLLLock, waitSERLock, waitTXEPLLLock, waitdllLocked.

The timeOut is enabled by asserting the following bits:

Register 52 (wdogCtr2); bits[5:3] = 111.

When the GBTX is already operating (FSM is in the idle state) and a problem occurs that lasts a long time (for example, the downlink fibre is unplugged) the combination of the watchdog and timeOut features will ensure that the GBTX automatically recovers when the problem is resolved (the reconnection of the fibre).

State transitions triggered by the timeOut are shown in the figure below.



10.8 Notes on using the watchdog in an SEU environment

The GBTX has been extensively tested with heavy ions provoking single-event upsets in the chip. In particular, the sensitivity of the serialiser/deserialiser circuits to SEUs and the resulting data errors have been measured (see [19] for more details). With the watchdog disabled, it has been observed that the chip always recovers to its functional state after an SEU. A certain number of bit and frame errors occur while the chip is recovering. With the watchdog enabled, the chip again recovers back to its functional state but the recovery process is longer and hence more bit and frame errors are observed.

The user should therefore carefully assess the advantages and disadvantages of enabling the watchdog if they will operate in a significant radiation environment.

10.9 Disabling the power-up sequence

The FSM can be halted at any time by asserting the stateOverride external signal. The state is then selected by the value written to a configuration register:

Register 51 (wdogCtr1); bits[4:0].

The 5-bit state identifiers are listed in Table 94.

10.10 Summary of Configuration inputs

The registers listed in Table 95 are used to configure the power-up state machine.

Table 95 Power-up configuration registers

Register address	Register name	Bits

Efuse 366		bits[2:0]
Register 365	configDone	bits [7:0]
Register 313	ckCtr44	bit[7]
Register 314	ckCtr45	bit[7]
Register 315	ckCtr46	bit[7]
Register 304	ckCtr35	bits[7:0]
Register 305	ckCtr36	bits[7:0]
Register 306	ckCtr37	bits[7:0]
Register 293	ckCtr24	bits[7:0]
Register 294	ckCtr25	bits[7:0]
Register 295	ckCtr26	bits[7:0]
Register 52	wdogCtr2	bits[2:0]
Register 50	wdogCtr0	bits[2:0]
Register 51	wdogCtr1	bits[4:0]

11. CLOCK MANAGER

The correct operation of the GBTX requires the presence of a reference clock signal which can be externally provided to the chip or can be obtained from an on chip oscillator with an on-package crystal. This clock signal is used as a clock reference during the initial stages of clock and data recover (lock acquisition) and as a clock source for the internal monitoring functions (watchdog circuit) and I2C interface.

The reference clock will be used as a clock reference for the CDR circuit, in transceiver and simplex receiver modes, or as a reference for the serializer circuit, in the simplex transmitter mode.

In the case the external reference is jittery it can be filtered by the internal VCXO based PLL (xPLL) before it is used as the chip reference.

When an external reference is not provided to the GBTX, the xPLL can be set to operate as a quartz crystal oscillator implementing a local reference. It is however necessary to note that in this case the reference is not synchronous with the system clock.

When an external reference clock is provided to the GBTX, the device is capable of operating with a reference frequency between 20 and 40 MHz ($f_{LHC}/2$ to f_{LHC}) with all data rates and programmable delays scaling directly with the reference clock frequency. However, xPLL operation is restricted to the LHC bunch crossing rate: 40.0786 MHz, due to the tight frequency constraints for the quartz crystal. If the xPLL is required to work at frequencies other than the LHC frequency then a custom crystal has to be custom cut for the required frequency.

11.1 Clock manager operation

The clock manager function sets up the GBTX clocks according to the different transceiver modes (see section 9.1). This requires no user intervention since the clock manager is automatically setup according to the transceiver mode selected by the **mode [3:0]** input signals. The "test mode" however allows the clock manager to be fully controlled through the control registers. This is only intended for testing purposes and the user should not use the device in test mode.

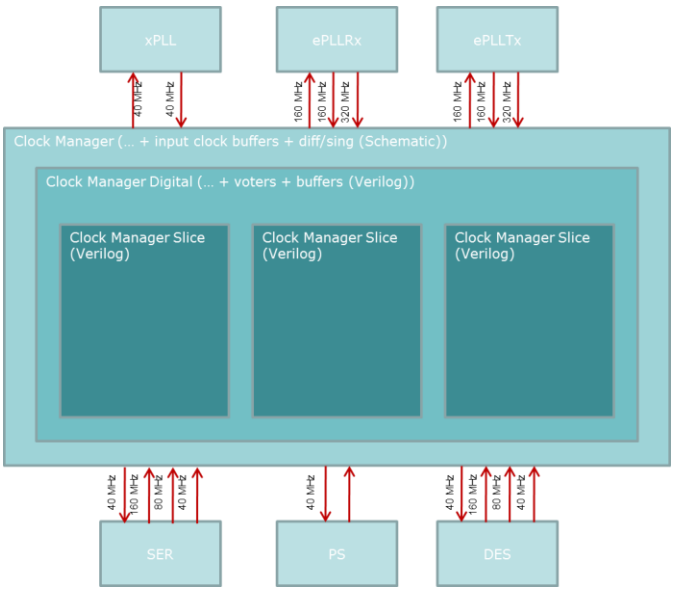


Figure 27 Clock manager block diagram

The block diagram of the clock manager is represented in Figure 27. It feeds the reference clocks to the Serializer, DESerializer, Phase-Shifter, XPLL, Rx – ePLL and Tx – ePLL. From these circuits the clock manager receives several clocks (at various frequencies) that are then routed to the data – path, control functions and ePorts. In the clock manager the triplication of the clock threes takes place.

11.2 ASIC reference clock selection and the XPLL

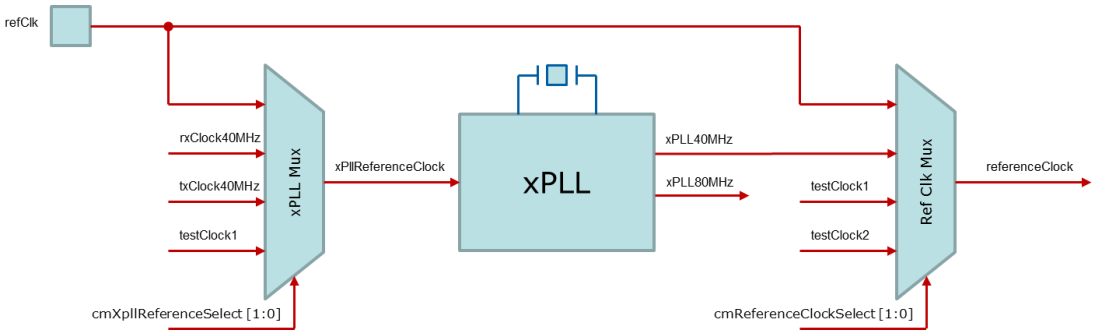


Figure 28 xPLL embedded in the clock manager

The ASIC reference clock, used for the Serializer, DESerializer etc., can be selected from two sources: the external reference clock (***refClk***) and the xPLL (see Figure 28). For test purposes the two external clock inputs (***testClock1*** and ***testClock2***) can also be used as the ASIC reference clock. The selection of the reference clock source is made by the "Reference Clock Mux" according to Table 96.

Table 96 ASIC clock reference selection

Register 281[1:0] = ckCtr12 [1:0] = cmReferenceClockSelectA[1:0] **Register 281[3:2] = ckCtr12 [3:2] = cmReferenceClockSelectB[1:0]** **Register 281[5:4] = ckCtr12 [5:4] = cmReferenceClockSelect C[1:0]**	***referenceClock***
cmReferenceClockSelect C/B/A[1:0] = 00	*refClk = REFCLKP - REFCLKN (external reference clock)*
cmReferenceClockSelect C/B/A[1:0] = 01	*xPLL40MHz (on package crystal oscillator)*
cmReferenceClockSelect C/B/A[1:0] = 10	*testClock1*
cmReferenceClockSelect C/B/A[1:0] = 11	*testClock2*

When the xPLL is used in the PLL mode a reference clock must be provided to the ASIC through the input ***refClk*** (see 9.4.1 for a detailed description on how to setup the xPLL). Settings for the xPLL Mux are given in Table 97 .Under most circumstances the setting "00" (***refClk*** selected) must be used. When the xPLL is in the VCXO mode the input signals ***REFCLKP*** *and* ***REFCLKN*** *must be hardwired so that* ***REFCLKP*** *= 1'b1 and* ***REFCLKN*** *= 1'b0.*

Table 97 xPLL reference clock selection

Register290 [1:0] = ckCtr21 [1:0] = cmXpllReferenceSelectA[1:0] **Register290 [3:2] = ckCtr21 [3:2] = cmXpllReferenceSelectB[1:0]** **Register290 [5:4] = ckCtr21 [5:4] = cmXpllReferenceSelectC[1:0]**	***xPllReferenceClock***
cmXpllReferenceSelect C/B/A[1:0] = 00	*refClk = REFCLKP - REFCLKN* *(external reference clock)*
cmXpllReferenceSelect C/B/A[1:0] = 01	*rxClk40MHz* *(for test purposes only)*
cmXpllReferenceSelect C/B/A[1:0] = 10	*txClk40MHz* *(for test purposes only)*
cmXpllReferenceSelect C/B/A[1:0] = 11	*testClock1*

11.3 Monitoring clock tree

The monitoring clock tree is used in the ASIC to clock all start-up and watchdog functions that should not depend on the clocks generated by the CDR and SER circuits. As illustrated in Figure 29 the monitoring clock tree can be driven by one of two clock sources: the external reference clock or the 40MHz generated by the xPLL. Selection of the clock source is done by the ASIC input signal ***REFCLKSELECT.***

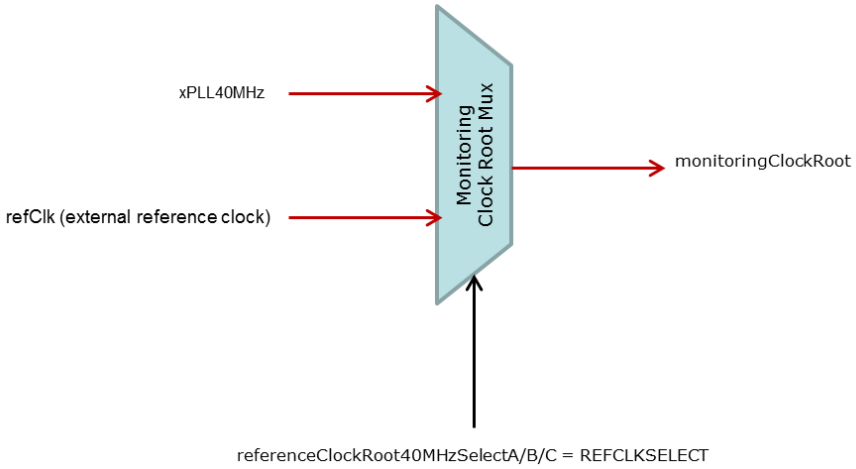


Figure 29 Monitoring clock tree multiplexer

11.4 ePLLRx and ePLLTx reference clocks

The GBTX contains two PLLs (ePLLs) to generate the 160, and 320 MHz clocks used in the ePorts (see section 9.5 for details on the ePLL operation). The selection of the reference clocks for the two ePLLs is given in, Table 98 to Table 101. Only in test mode is the user required to select the reference clock, in the other modes the reference clock signal is automatically selected depending on the ***mode [1:0]*** ASIC input signals. (Please notice that the names ePLLRx (receiver) and ePLLTx (transmitter) relate to the high speed (4.8 Gbps) receiver and transmitter and not the ePorts themselves. Following that logic the ePLLRx is used in the transmitting ePorts of the GBTX receiver and the ePLLTx is used in the receiving ePorts of the GBTX transmitter.)

Table 98 Receiver ePLL (ePLLRx) reference clock multiplexer

Mode [1:0]	***ePllRxMux (reference clock multiplexer)***
00 (simplex Tx)	0
01 (simplex Rx)	rxClk160MHz
10 (transceiver)	*rxClk160MHz*
11 (test mode)	*ePllRxTestMux*

Table 99 Receiver ePLL (ePLLRx) reference clock test multiplexer

Register0 [1:0] = ckCtr0 [1:0] = cmEpllRxReferenceSelectA[1:0] **Register0 [3:2] = ckCtr0 [3:2] = cmEpllRxReferenceSelectB[1:0]** **Register0 [5:4] = ckCtr0 [5:4] = cmEpllRxReferenceSelectC[1:0]**	***ePllRxTestMux (reference clock test multiplexer)***
cmEpllRxReferenceSelect C/B/A[1:0] = 00	rxClk40MHz
cmEpllRxReferenceSelect C/B/A[1:0] = 01	rxClk80MHz
cmEpllRxReferenceSelect C/B/A[1:0] = 10	*rxClk160MHz*
cmEpllRxReferenceSelect C/B/A[1:0] = 11	*testClock1*

Table 100 Transmitter ePLL (ePLLRx) reference clock multiplexer

Mode [1:0]	***ePllTxMux (reference clock multiplexer)***
00 (simplex Tx)	txClk160MHz
01 (simplex Rx)	0
10 (transceiver)	*txClk160MHz*
11 (test mode)	*ePllTxTestMux*

Table 101 Transmitter ePLL (ePLLTx) reference clock test multiplexer

Register1 [1:0] = ckCtr1 [1:0] = cmEpllTxReferenceSelectA[1:0] **Register1 [3:2] = ckCtr1 [3:2] = cmEpllTxReferenceSelectB[1:0]** **Register1 [5:4] = ckCtr1 [5:4] = cmEpllTxReferenceSelectC[1:0]**	***ePllTxTestMux (reference clock test multiplexer)***
cmEpllTxReferenceSelect C/B/A[1:0] = 00	txClk40MHz
cmEpllTxReferenceSelect C/B/A[1:0] = 01	txClk80MHz
cmEpllTxReferenceSelect C/B/A[1:0] = 10	*txClk160MHz*
cmEpllTxReferenceSelect C/B/A[1:0] = 11	*testClock2*

11.4.1 Selecting the reference clock frequency

The feedback dividers of the ePLLs must be programed so that the division ratio is set according to the reference frequency, for details section 9.5.1.1.1

11.5 Phase-Shifter reference clock

The phase-shifter reference clock is selected according to Table 102 and Table 103. The **mode [1:0]** ASIC input signals determine which clock is given as a reference clock to the phase-shifter (see Table 102). When in test mode the phase-shifter test mux has to be further setup (see Table 103) to select the phase-shifter reference clock.

Table 102 Phase-Shifter reference clock multiplexer

Mode [1:0]	***psMux (reference clock multiplexer)***
00 (simplex Tx)	txClk40MHz
01 (simplex Rx)	rxClk40MHz
10 (transceiver)	*rxClk40MHz*
11 (test mode)	*psTestMux*

Table 103 Phase-Shifter reference clock test multiplexer

Register282 [1:0] = ckCtr13 [1:0] = cmPsReferenceSelectA[1:0] **Register282 [3:2] = ckCtr13 [3:2] = cmPsReferenceSelectB[1:0]** **Register282 [5:4] = ckCtr13 [5:4] = cmPsReferenceSelectC[1:0]**	***psTestMux (reference clock test multiplexer)***
cmPsReferenceSelect C/B/A[1:0] = 00	*referenceClock (external or xPLL)*
cmPsReferenceSelect C/B/A[1:0] = 01	*txClk40MHz*
cmPsReferenceSelect C/B/A[1:0] = 10	*rxClk40MHz*
cmPsReferenceSelect C/B/A[1:0] = 11	*testClock1*

11.6 Rx (CDR) reference clock

The high speed receiver (CDR) reference clock is automatically set by the **mode [1:0]** ASIC input signals. However, when in test mode the user has the freedom to choose the receiver reference clock source by setting the test multiplexer. This is detailed in Table 104 and Table 105.

Table 104 Rx (CDR) reference clock multiplexer

Mode [1:0]	***rxReferenceClock (reference clock multiplexer)***
00 (simplex Tx)	*0*
01 (simplex Rx)	*referenceClock*

10 (transceiver)	*referenceClock*
11 (test mode)	*rxReferenceTestMux*

Table 105 Rx (CDR) reference clock test multiplexer

Register2 [1:0] = ckCtr2 [1:0] = cmRxReferenceTestMuxSelectA[1:0] **Register2 [3:2] = ckCtr2 [3:2] = cmRxReferenceTestMuxSelectB[1:0]** **Register2 [5:4] = ckCtr2 [5:4] = cmRxReferenceTestMuxSelectC[1:0]**	***rxReferenceTestMux (reference clock test multiplexer)***
cmRxReferenceTestMuxSelect C/B/A[1:0] = 00	*referenceClock*
cmRxReferenceTestMuxSelect C/B/A[1:0] = 01	*txClk40MHz*
cmRxReferenceTestMuxSelect C/B/A[1:0] = 10	*testClock1*
cmRxReferenceTestMuxSelect C/B/A[1:0] = 11	*testClock2*

11.1 Tx (SER) reference clock

The high speed transmitter (SER) reference clock is automatically set by the **mode [1:0]** ASIC input signals. However, when in test mode the user has the freedom to choose the transmitter reference clock source by setting the test multiplexer. This is detailed in Table 106 and Table 107.

Table 106 Tx (SER) reference clock multiplexer

Mode [1:0]	***txReferenceClock (reference clock multiplexer)***
00 (simplex Tx)	*referenceClock*
01 (simplex Rx)	*0*
10 (transceiver)	*rxClk40MHz*
11 (test mode)	*txReferenceTestMux*

Table 107 Tx (SER) reference clock test multiplexer

Register3 [1:0] = ckCtr3 [1:0] = cmTxReferenceTestMuxSelectA[1:0] **Register3 [3:2] = ckCtr3 [3:2] = cmTxReferenceTestMuxSelectB[1:0]** **Register3 [5:4] = ckCtr3 [5:4] = cmTxReferenceTestMuxSelectC[1:0]**	***txReferenceTestMux (reference clock test multiplexer)***
cmTxReferenceTestMuxSelect C/B/A[1:0] = 00	*referenceClock*
cmTxReferenceTestMuxSelect C/B/A[1:0] = 01	*rxClk40MHz*
cmTxReferenceTestMuxSelect C/B/A[1:0] = 10	*xPLL40MHz*

cmTxReferenceTestMuxSelect C/B/A[1:0] = 11	*testClock1*

11.2 Receiver (Rx) clocks

The GBTX receiver clocks are setup according to what is given in Table 108 to Table 115. All the receiver clocks are set automatically according to the **mode [1:0]** ASIC input signals. Only the test mode allows the user choose the clock settings.

Table 108 Rx 40 MHz clock mux

Mode [1:0]	***rxMux40MHz***
00 (simplex Tx)	*0*
01 (simplex Rx)	*rxClk40MHz*
10 (transceiver)	*rxClk40MHz*
11 (test mode)	*rxTestMux40MHz*

Table 109 Rx 40 MHz test clock mux

Register279 [1:0] = ckCtr10 [1:0] = cmRxTestMuxSelect40C[1:0] **Register278 [1:0] = ckCtr9 [1:0] = cmRxTestMuxSelect40B[1:0]** **Register277 [1:0] = ckCtr8 [1:0] = cmRxTestMuxSelect40A[1:0]**	***rxTestMux40MHz***
cmRxTestMuxSelect40 C/B/A[1:0] = 00	*rxClk40MHz*
cmRxTestMuxSelect40 C/B/A[1:0] = 01	*txClk40MHz*
cmRxTestMuxSelect40 C/B/A[1:0] = 10	*testClock1*
cmRxTestMuxSelect40 C/B/A[1:0] = 11	*testClock2*

Table 110 Rx 80 MHz clock mux

Mode [1:0]	***rxMux80MHz***
00 (simplex Tx)	*0*
01 (simplex Rx)	*rxClk80MHz*
10 (transceiver)	*rxClk80MHz*
11 (test mode)	*rxTestMux80MHz*

Table 111 Rx 80 MHz test clock mux

Register279 [3:2] = ckCtr10 [3:2] = cmRxTestMuxSelect80C[1:0]	***rxTestMux80MHz***

Register278 [3:2] = ckCtr9 [3:2] = cmRxTestMuxSelect80B[1:0] **Register277 [3:2] = ckCtr8 [3:2] = cmRxTestMuxSelect80A[1:0]**	
cmRxTestMuxSelect80 C/B/A[1:0] = 00	*rxClk80MHz*
cmRxTestMuxSelect80 C/B/A[1:0] = 01	*txClk80MHz*
cmRxTestMuxSelect80 C/B/A[1:0] = 10	*testClock1*
cmRxTestMuxSelect80 C/B/A[1:0] = 11	*testClock2*

Table 112 Rx 160 MHz clock mux

Mode [1:0]	***rxMux160MHz***
00 (simplex Tx)	*0*
01 (simplex Rx)	*ePllRx160MHx*
10 (transceiver)	*ePllRx160MHz*
11 (test mode)	*rxTestMux160MHz*

Table 113 Rx 160 MHz test clock mux

Register279 [5:4] = ckCtr10 [5:4] = cmRxTestMuxSelect160C[1:0] **Register278 [5:4] = ckCtr9 [5:4] = cmRxTestMuxSelect160B[1:0]** **Register277 [5:4] = ckCtr8 [5:4] = cmRxTestMuxSelect160A[1:0]**	***rxTestMux160MHz***
cmRxTestMuxSelect160 C/B/A[1:0] = 00	*ePllRx160MHz*
cmRxTestMuxSelect160 C/B/A[1:0] = 01	*ePllTx160MHz*
cmRxTestMuxSelect160 C/B/A[1:0] = 10	*rxClk160MHz*
cmRxTestMuxSelect160 C/B/A[1:0] = 11	*testClock2*

Table 114 Rx 320 MHz clock mux

Mode [1:0]	***rxMux320MHz***
00 (simplex Tx)	*0*
01 (simplex Rx)	*ePllRx320MHz*
10 (transceiver)	*ePllRx320MHz*
11 (test mode)	*rxTestMux320MHz*

Table 115 Rx 320 MHz test clock mux

Register279 [7:6] = ckCtr10 [7:6] = cmRxTestMuxSelect320C[1:0] **Register278 [7:6] = ckCtr9 [7:6] = cmRxTestMuxSelect320B[1:0]**	***rxTestMux320MHz***

Register277 [7:6] = ckCtr8 [7:6] = cmRxTestMuxSelect320A[1:0]	
cmRxTestMuxSelect320 C/B/A[1:0] = 00	*ePllRx320MHz*
cmRxTestMuxSelect320 C/B/A[1:0] = 01	*ePllTx320MHz*
cmRxTestMuxSelect320 C/B/A[1:0] = 10	*testClock1*
cmRxTestMuxSelect320 C/B/A[1:0] = 11	*testClock2*

11.3 Transmitter (Tx) clocks

The GBTX transmitter clocks are setup according to what is given in Table 116 to Table 123. All the transmitter clocks are set automatically according to the **mode [1:0]** ASIC input signals. Only the test mode allows the user choose the clock settings.

Table 116 Tx 40 MHz clock mux

Mode [1:0]	***txMux40MHz***
00 (simplex Tx)	*txClk40MHz*
01 (simplex Rx)	*0*
10 (transceiver)	*txClk40MHz*
11 (test mode)	*txTestMux40MHz*

Table 117 Tx 40 MHz test clock mux

Register289 [1:0] = ckCtr20 [1:0] = cmTxTestMuxSelect40C[1:0] **Register288 [1:0] = ckCtr19 [1:0] = cmTxTestMuxSelect40B[1:0]** **Register287 [1:0] = ckCtr18 [1:0] = cmTxTestMuxSelect40A[1:0]**	***txTestMux40MHz***
cmTxTestMuxSelect40 C/B/A[1:0] = 00	*rxClk40MHz*
cmTxTestMuxSelect40 C/B/A[1:0] = 01	*txClk40MHz*
cmTxTestMuxSelect40 C/B/A[1:0] = 10	*testClock1*
cmTxTestMuxSelect40 C/B/A[1:0] = 11	*testClock2*

Table 118 Tx 80 MHz clock mux

Mode [1:0]	***txMux80MHz***
00 (simplex Tx)	*txClk80MHz*
01 (simplex Rx)	*0*
10 (transceiver)	*txClk80MHz*
11 (test mode)	*txTestMux80MHz*

Table 119 Tx 80 MHz test clock mux

Register289 [3:2] = ckCtr20 [3:2] = cmTxTestMuxSelect80C[1:0] **Register288 [3:2] = ckCtr19 [3:2] = cmTxTestMuxSelect80B[1:0]** **Register287 [3:2] = ckCtr18 [3:2] = cmTxTestMuxSelect80A[1:0]**	***txTestMux80MHz***
cmTxTestMuxSelect80 C/B/A[1:0] = 00	*rxClk80MHz*
cmTxTestMuxSelect80 C/B/A[1:0] = 01	*txClk80MHz*
cmTxTestMuxSelect80 C/B/A[1:0] = 10	*testClock1*
cmTxTestMuxSelect80 C/B/A[1:0] = 11	*testClock2*

Table 120 Tx 160 MHz clock mux

Mode [1:0]	***txMux160MHz***
00 (simplex Tx)	*ePllTx160MHz*
01 (simplex Rx)	*0*
10 (transceiver)	*ePllTx160MHz*
11 (test mode)	*txTestMux160MHz*

Table 121 Tx 160 MHz test clock mux

Register289 [5:4] = ckCtr20 [5:4] = cmTxTestMuxSelect160C[1:0] **Register288 [5:4] = ckCtr19 [5:4] = cmTxTestMuxSelect160B[1:0]** **Register287 [5:4] = ckCtr18 [5:4] = cmTxTestMuxSelect160A[1:0]**	***txTestMux160MHz***
cmTxTestMuxSelect160 C/B/A[1:0] = 00	*ePllRx160MHz*
cmTxTestMuxSelect160 C/B/A[1:0] = 01	*ePllTx160MHz*
cmTxTestMuxSelect160 C/B/A[1:0] = 10	*txClk160MHz*
cmTxTestMuxSelect160 C/B/A[1:0] = 11	*testClock1*

Table 122 Tx 320 MHz clock mux

Mode [1:0]	***txMux320MHz***
00 (simplex Tx)	*ePllTx320MHz*
01 (simplex Rx)	*0*
10 (transceiver)	*ePllTx320MHz*
11 (test mode)	*txTestMux320MHz*

Table 123 Tx 320 MHz test clock mux

Register289 [7:6] = ckCtr20 [7:6] = cmTxTestMuxSelect320C[1:0] **Register288 [7:6] = ckCtr19 [7:6] = cmTxTestMuxSelect320B[1:0]** **Register287 [7:6] = ckCtr18 [7:6] = cmTxTestMuxSelect320A[1:0]**	***txTestMux320MHz***
cmTxTestMuxSelect320 C/B/A[1:0] = 00	*ePllRx320MHz*
cmTxTestMuxSelect320 C/B/A[1:0] = 01	*ePllTx320MHz*
cmTxTestMuxSelect320 C/B/A[1:0] = 10	*testClock1*
cmTxTestMuxSelect320 C/B/A[1:0] = 11	*testClock2*

11.4 Rx and Tx phase trimming

It is possible to trim the phases of the receiver clocks, (rx40MHz, rx80MHz, rx160MHz and rx320MHz) and the transmitter clocks (tx40MHz, tx80MHz, tx160MHz and tx320MHz). The trimming of the 40 and 80 MHz clock phases is done in the "clock manager" while trimming of the 160 and 320 MHz clocks is done in the e-PLLs. Please note that this phase trimming capability is only intending to guarantee correct timing among the GBTX clocks and not to be used as a system feature. **The correct values for all internal clock phases will be pre-programed in the e-Fuse bank and the users are not allowed to modify them at the risk of malfunctioning of the device.**

11.4.1 40 and 80 MHz internal clock phases

The phases of the 40 and 80 MHz clocks can be phase shifted up to a maximum of 4.5 ns in steps of 0.3 ns according to the phase control 4-bit words given in the four tables below. To obtain the phase of the clock signal in question multiply the 4-bit binary number written in the control register by 300 ps. Notice that for each clock signal the triplicated numbers (A/B/C) must all be written with the same value.

Table 124 Rx 40 MHz clock phase control registers

Register	**Name**	**Signal**
Register 274[3:0]	ckCtr 5[3:0]	cmRxPhase40MHzA[3:0]
Register 274[7:4]	ckCtr 5[7:4]	cmRxPhase40MHzB[3:0]
Register 275[3:0]	ckCtr 6[3:0]	cmRxPhase40MHzC[3:0]

Table 125 Rx 80 MHz clock phase control registers

Register	**Name**	**Signal**
Register 275[7:4]	ckCtr 6[7:4]	cmRxPhase80MHzA[3:0]
Register 276[3:0]	ckCtr 7[3:0]	cmRxPhase80MHzB[3:0]
Register 276[7:4]	ckCtr 7[7:4]	cmRxPhase80MHzC[3:0]

Table 126 Tx 40 MHz clock phase control registers

Register	Name	Signal
Register 284[3:0]	ckCtr 15[3:0]	cmTxPhase40MHzA[3:0]
Register 284[7:4]	ckCtr 15[7:4]	cmTxPhase40MHzB[3:0]
Register 285[3:0]	ckCtr 16[3:0]	cmTxPhase40MHzC[3:0]

Table 127 Tx 80 MHz clock phase control registers

Register	Name	Signal
Register 285[7:4]	ckCtr 16[7:4]	cmTxPhase80MHzA[3:0]
Register 286[3:0]	ckCtr 17[3:0]	cmTxPhase80MHzB[3:0]
Register 286[7:4]	ckCtr 17[7:4]	cmTxPhase80MHzC[3:0]

11.4.2 160 and 320 MHz internal clock phases

The 160 and 320 MHz phases are adjusted in the transmitter and receiver ePLLs. Please see section 9.5.1 for details on the e-PLL operation.

11.5 Clock root sampler

Please ignore this section a bug in the implementation prevents its use as it is described here.

It is possible to inspect the relative clock phases at the root of the Tx and Rx clock trees using the "clock root sampler". The schematic principle for the device when using the rxClockRoot40MHz as the sampling clock is illustrated in Figure 30. Similar circuits exit for the other clock signals, a complete list of the available signals is given in Table 128.

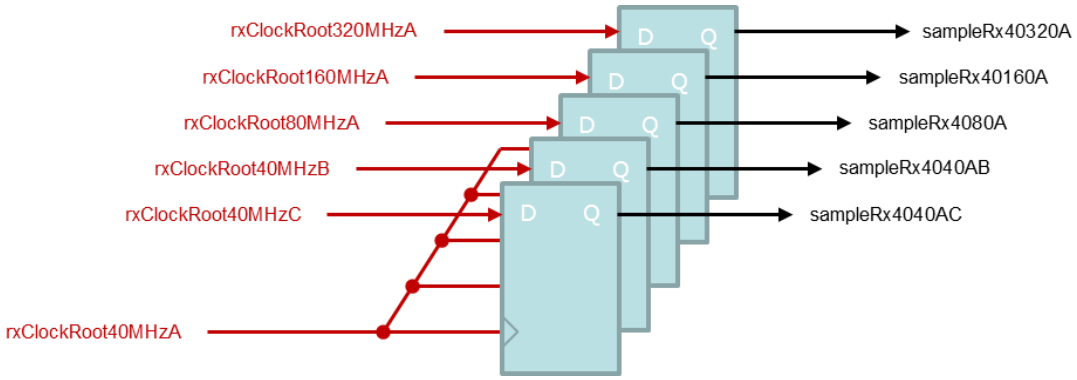


Figure 30 Clock root sampler for the rxClockRoot40MHzA sampling clock

The device provides an early/late indication on the relative phases between two clocks that is limited to the phase resolution of the clocks being sampled. When a high value is sampled this means that the sampled clock is early in relation to the sampling clock and, conversely, a low value indicates the sampled clock being late. To use the device to determine the relative phases between two clocks one should set the phase of the sampling clock and sweep the phase of the sampled clock. When the sampled value changes from low-to-high or high-to-low the "zero"-phase position has been found. The resolution is ±150 ps for the 40 and 80 MHz clocks (300 ps phase steps) and ±104 ps for the 160 and 320 MHz clocks (208 ps phase

steps). It is possible to use the device by fixing the phase of the sampled clocks and sweep the phase of the sampling clock. However in this case the resolution will be limited to the phase resolution of the sampling clock (±150 ps) in all cases. The sampled values can be observed through the **testClockOut**. Please see section 16.4.2 for details on how to use and program the **testClockOut** signal.

Table 128 Clock root sampler output multiplexer

Register 319[4:0]	Sampling clock	Sampled clock	Sample signal
0	n. a.	n. a.	n.a.
1	rxClockRoot40MHzA	rxClockRoot80MHzA	sampleRx4080A
2	rxClockRoot40MHzA	rxClockRoot160MHzA	sampleRx40160A
3	rxClockRoot40MHzA	rxClockRoot320MHzA	sampleRx40320A
4	rxClockRoot40MHzA	rxClockRoot40MHzB	sampleRx4040AB
5	rxClockRoot40MHzA	rxClockRoot40MHzC	sampleRx4040AC
6	rxClockRoot40MHzB	rxClockRoot80MHzB	sampleRx4080B
7	rxClockRoot40MHzB	rxClockRoot160MHzB	sampleRx40160B
8	rxClockRoot40MHzB	rxClockRoot320MHzB	sampleRx40320B
9	rxClockRoot40MHzB	rxClockRoot40MHzA	sampleRx4040BA
10	rxClockRoot40MHzB	rxClockRoot40MHzC	sampleRx4040BC
11	rxClockRoot40MHzC	rxClockRoot80MHzC	sampleRx4080C
12	rxClockRoot40MHzC	rxClockRoot160MHzC	sampleRx40160C
13	rxClockRoot40MHzC	rxClockRoot320MHzC	sampleRx40320C
14	rxClockRoot40MHzC	rxClockRoot40MHzA	sampleRx4040CA
15	rxClockRoot40MHzC	rxClockRoot40MHzB	sampleRx4040CB
16	txClockRoot40MHzA	txClockRoot80MHzA	sampleTx4080A
17	txClockRoot40MHzA	txClockRoot160MHzA	sampleTx40160A
18	txClockRoot40MHzA	txClockRoot320MHzA	sampleTx40320A
19	txClockRoot40MHzA	txClockRoot40MHzB	sampleTx4040AB
20	txClockRoot40MHzA	txClockRoot40MHzC	sampleTx4040AC
21	txClockRoot40MHzB	txClockRoot80MHzB	sampleTx4080B
22	txClockRoot40MHzB	txClockRoot160MHzB	sampleTx40160B
23	txClockRoot40MHzB	txClockRoot320MHzB	sampleTx40320B
24	txClockRoot40MHzB	txClockRoot40MHzA	sampleTx4040BA
25	txClockRoot40MHzB	txClockRoot40MHzC	sampleTx4040BC
26	txClockRoot40MHzC	txClockRoot80MHzC	sampleTx4080C
27	txClockRoot40MHzC	txClockRoot160MHzC	sampleTx40160C
28	txClockRoot40MHzC	txClockRoot320MHzC	sampleTx40320C
29	txClockRoot40MHzC	txClockRoot40MHzA	sampleTx4040CA
30	txClockRoot40MHzC	txClockRoot40MHzB	sampleTx4040CB
31	n. a.	n. a.	0

12. PHASE/FREQUENCY PROGRAMMABLE CLOCKS

The GBTX ASIC provides 8 SLVS clock outputs that can be used as local timing references for the front-end modules. These clocks are generated by the phase-shifter circuit (see Figure 31), and are fully synchronous with one of the GBTX 40 MHz clocks (depending on the transceiver mode) which in turn are synchronous with LHC bunch crossing reference and maintain with it a stable phase relationship. The 8 reference clocks are programmable both in phase, 50 ps resolution and frequency 40, 80 160 and 320 MHz. These clocks are associated with differential drivers with programmable driving strength (see chapter 13) and their frequency can be set independently from each other.

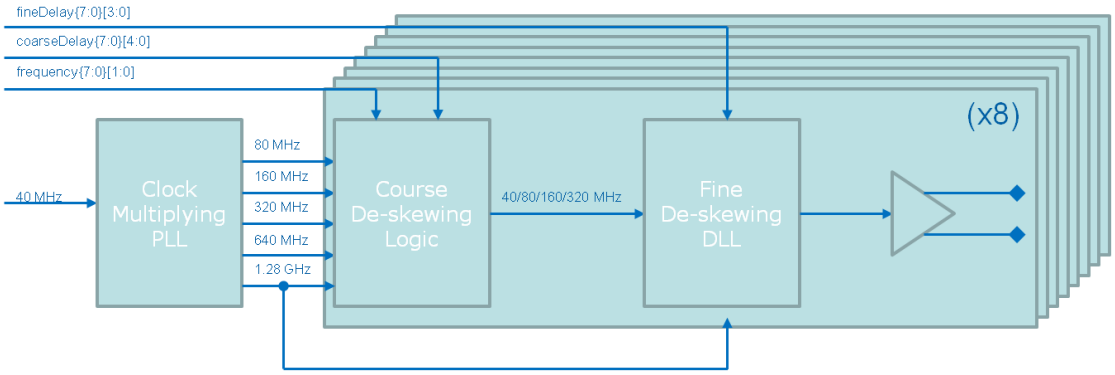


Figure 31. Architecture of the phase-shifter

12.1 Phase-shifter operation

As seen in Figure 31 the phase-shifter is composed of a PLL that multiplies the GBTX 40 MHz clock signal generating in-phase 80, 160, 320, 640 and 1280 MHz clocks. These clocks are fed to 8 independently programmable phase-shifter channels where a clock signal with selectable frequency (40, 80, 160 and 320 MHz) and phase is generated. The clock phase can be controlled with a resolution of 48.8 ps. Phase shifting is done in two stages: a coarse phase-shifting stage where the phase is controlled with a resolution of 781.25 ps within the a full clock period followed by a fine phase-shifting stage that further interpolates the clock phase with a resolution of 48.8 ps within the 781.25 ps. This results in a total phase span of 0 to 2π with a resolution of 48.8 ps. The fine phase-shifting stage is based on a DLL calibrated by the 1.28 GHz clock. There are thus 8 DLLs, one per phase-shifter channel.

12.2 Programing the phase-shifter

Programing the phase-shifter amounts to enable the desired number of active channels and for each channel to choose its frequency and phase. Circuit parameters for the PLL and DLLs must also be set.

12.2.1 Programing the phase-shifter channels' frequency

The channel frequency is set by programming registers 16 to 23 according to Table 129 and Table 130.

Table 129 Phase-Shifter frequency control registers

Register	Name	Channel	Signal
16 [5:4]	ttcCtr12[5:4]	0	FREQ0[1:0]

17 [5:4]	ttcCtr13[5:4]	1	FREQ1[1:0]
18 [5:4]	ttcCtr14[5:4]	2	FREQ2[1:0]
19 [5:4]	ttcCtr15[5:4]	3	FREQ3[1:0]
20 [5:4]	ttcCtr16[5:4]	4	FREQ4[1:0]
21 [5:4]	ttcCtr17[5:4]	5	FREQ5[1:0]
22 [5:4]	ttcCtr18[5:4]	6	FREQ6[1:0]
23 [5:4]	ttcCtr19[5:4]	7	FREQ7[1:0]

Table 130 Phase-Shifter frequency control settings

FREQX[1:0] with X = 0, 1, 2, 3, 4, 5, 6 & 7	**Frequency in MHz**
00	40
01	80
10	160
11	320

12.2.2 Programming the phase-shifter channels' phase

Programing a phase-shifter channel delay involves setting two parameters: the coarse delay parameters CDLX[4:0] (with X = 0, 1, 2, 3, 4, 5, 6 & 7) and the fine delay parameters FDLX[3:0] (with X = 0, 1, 2, 3, 4, 5, 6 & 7). The channel phase-shift (or delay) is given by:

Channel Delay = CDLX[4:0] × 781.25 ps+ FDLX[3:0] × 48.8 ps

A list of the coarse and fine delay registers can be found in tables Table 131 and Table 132 respectively.

Table 131 Coarse delay registers

Register	**Name**	**Channel**	**Signal**
8 [4:0]	ttcCtr4[4:0]	0	CDL0[4:0]
9 [4:0]	ttcCtr5[4:0]	1	CDL1[4:0]
10 [4:0]	ttcCtr6[4:0]	2	CDL2[4:0]
11 [4:0]	ttcCtr7[4:0]	3	CDL3[4:0]
12 [4:0]	ttcCtr8[4:0]	4	CDL4[4:0]
13 [4:0]	ttcCtr9[4:0]	5	CDL5[4:0]
14 [4:0]	ttcCtr10[4:0]	6	CDL6[4:0]
15 [4:0]	ttcCtr11[4:0]	7	CDL7[4:0]

Table 132 Fine delay registers

Register	**Name**	**Channel**	**Signal**
4 [3:0]	ttcCtr0[3:0]	0	FDL0[3:0]
4 [7:4]	ttcCtr0[7:4]	1	FDL1[3:0]
5 [3:0]	ttcCtr1[3:0]	2	FDL2[3:0]

5 [7:4]	ttcCtr1[7:4]	3	FDL3[3:0]
6 [3:0]	ttcCtr2[3:0]	4	FDL4[3:0]
6 [7:4]	ttcCtr2[7:4]	5	FDL5[3:0]
7 [3:0]	ttcCtr3[3:0]	6	FDL6[3:0]
7 [7:4]	ttcCtr3[7:4]	7	FDL7[3:0]

12.3 Phase-Shifter PLL and DLL settings

For correct operation of the phase-shifter, the PLL and DLL parameters need to set at start-up. Table 133 and Table 134 indicate the recommended settings. Notice that the PLL is common to all channels while each channel has its own DLL.

Table 133 PLL loop-filter control registers

Register	Name	Signal	Recommended Setting
26 [3:0]	ttcCtr22[3:0]	PLLcp[3:0]	4'b1111
26 [6:4]	ttcCtr22[6:4]	PLLres[6:4]	3'b111
52 [2:0]	wdogCtr2[2:0]	PLL enable	3'b111

Table 134 DLLs charge-pump control registers

Register	Name	Channel	Signal	Recommended Setting
16 [3:0]	ttcCtr12[3:0]	0	iSel0[3:0]	4'b0100
17 [3:0]	ttcCtr13[3:0]	1	iSel1[3:0]	4'b0100
18 [3:0]	ttcCtr14[3:0]	2	iSel2[3:0]	4'b0100
19 [3:0]	ttcCtr15[3:0]	3	iSel3[3:0]	4'b0100
20 [3:0]	ttcCtr16[3:0]	4	iSel4[3:0]	4'b0100
21 [3:0]	ttcCtr17[3:0]	5	iSel5[3:0]	4'b0100
22 [3:0]	ttcCtr18[3:0]	6	iSel6[3:0]	4'b0100
23 [3:0]	ttcCtr19[3:0]	7	iSel7[3:0]	4'b0100

12.4 Resetting the phase-shifter PLL and DLLs

The phase-shifter PLL and DLLs reset signals are cycled at start-up by the initialization state machine (see chapter 10). However these circuits can be reset by user intervention. The order to reset them is PLL first followed by the DLLs. The reset signals are active low so to reset these circuits the user should write a "0" and then a "1" to the corresponding register. After a PLL reset the user needs to wait 5 μs for the PLL to lock and them can issue DLLs resets (the DLLs will take approximately 50 μs to lock).

Table 135 Phase-Shifter reset registers

Register	Name	Channel	Signal	Recommended Setting
24 [0]	ttcCtr20[0]	0	resetBar0	1'b1
24 [1]	ttcCtr20[1]	1	resetBar1	1'b1
24 [2]	ttcCtr20[2]	2	resetBar2	1'b1
24 [3]	ttcCtr20[3]	3	resetBar3	1'b1
24 [4]	ttcCtr20[4]	4	resetBar4	1'b1
24 [5]	ttcCtr20[5]	5	resetBar5	1'b1
24 [6]	ttcCtr20[6]	6	resetBar6	1'b1
24 [7]	ttcCtr20[7]	7	resetBar7	1'b1
25 [0]	ttcCtr21[0]	n.a.	resetPLLBar	1'b1

12.5 Phase-shifter outputs

The phase-shifter has 8 outputs (see Table 136) ports with independently programmable strength (see Table 137 and chapter 13)

Table 136 Phase-shifter output signals

Channel Number	Pin Name	Pin number
0	CLOCKDES0N CLOCKDES0P	H1 G1
1	CLOCKDES1N CLOCKDES1P	J4 H4
2	CLOCKDES2N CLOCKDES2P	K5 K4
3	CLOCKDES3N CLOCKDES3P	L5 L4
4	CLOCKDES4N CLOCKDES4P	M5 M4
5	CLOCKDES5N CLOCKDES5P	N2 N1
6	CLOCKDES6N CLOCKDES6P	M7 M6
7	CLOCKDES7N CLOCKDES7P	P2 P1

Table 137 Phase-shifter outputs control registers

Register	Name	Channel	Signal	Setting for SLVS levels
269 [3:0]	ttcCtr23[3:0]	0	cset0[3:0]	2'b1001
269 [7:4]	ttcCtr23[7:4]	1	cset1[3:0]	2'b1001
270 [3:0]	ttcCtr24[3:0]	2	cset2[3:0]	2'b1001
270 [7:4]	ttcCtr24[7:4]	3	cset3[3:0]	2'b1001

271 [3:0]	ttcCtr25[3:0]	4	cset4[3:0]	2'b1001
271 [7:4]	ttcCtr25[7:4]	5	cset5[3:0]	2'b1001
272 [3:0]	ttcCtr26[3:0]	6	cset6[3:0]	2'b1001
272 [7:4]	ttcCtr26[7:4]	7	cset7[3:0]	2'b1001

12.6 Phase-shifter test features

For testing purposes some of the PLL and DLLs signals can be observed through the **TESTOUTPUT** port (see section 16.4.1) or through the read only registers (see section 17.3). In the latter case the read back value will be a sample of the state of the signal at readout time. Some of the DLLs signals can also be controlled (in a "static way") using the ASIC control registers.

12.6.1 PLL test signals

Some of the Phase-Shifter PLL signals can be observed on the **TESTOUTPUT** port, Table 138 gives the details (see also section 16.4.1).

Table 138 Phase-Shifter PLL test signals selection

Register 280[7:0]	**Selected signal**	**Function**
8'd0	ttcDivideOut	PLL feedback clock (40 MHz)
8'd1	ttcTestDown	PLL charge-pump down signal[1]
8'd2	ttcTestUp	PLL charge-pump up signal[1]
Note 1	The signals **ttcTestDown** and **ttcTestUp** need to be enabled by writing Register 25[1] = enableTestBar = 1'b0	

Sampled values of these signals at read time can be obtained by reading register 428 (bits 0, 1 and 2), as detailed in Table 154.

12.6.2 DLL test signals

The early and late signals originating from the phase-shifter DLLs control loop can be observed on the **TESTOUTPUT** port (see section 16.4.1) as indicated in Table 139.

Table 139 Phase-Shifter DLLs test signals selection

Register 280[7:0]	Selected signal	Function
8'd3	ttcEarly[0]	Channel 0 DLL early signal
8'd4	ttcEarly[1]	Channel 1 DLL early signal
8'd5	ttcEarly[2]	Channel 2 DLL early signal
8'd6	ttcEarly[3]	Channel 3 DLL early signal
8'd7	ttcEarly[4]	Channel 4 DLL early signal
8'd8	ttcEarly[5]	Channel 5 DLL early signal
8'd9	ttcEarly[6]	Channel 6 DLL early signal
8'd10	ttcEarly[7]	Channel 7 DLL early signal

8'd11	ttcLate[0]	Channel 0 DLL late signal
8'd12	ttcLate[1]	Channel 1 DLL late signal
8'd13	ttcLate[2]	Channel 2 DLL late signal
8'd14	ttcLate[3]	Channel 3 DLL late signal
8'd15	ttcLate[4]	Channel 4 DLL late signal
8'd16	ttcLate[5]	Channel 5 DLL late signal
8'd17	ttcLate[6]	Channel 6 DLL late signal
8'd18	ttcLate[7]	Channel 7 DLL late signal

For test purposes it is possible to control the phase-shifter DLLs' early and late signals through the configuration registers, the required settings are given in Table 140 and Table 141.

Table 140 Phase-shifter DLLs early signals control.

Register [bit]	Signal name	Channel	Value	Early signal driven by
8 [5]	extS0[0]	0	1'b0 1'b1	DLL control loop extEarly0 = Register8[7]
9 [5]	extS1[0]	1	1'b0 1'b1	DLL control loop extEarly1 = Register9[7]
10 [5]	extS2[0]	2	1'b0 1'b1	DLL control loop extEarly2 = Register10[7]
11 [5]	extS3[0]	3	1'b0 1'b1	DLL control loop extEarly3 = Register11[7]
12 [5]	extS4[0]	4	1'b0 1'b1	DLL control loop extEarly4 = Register12[7]
13 [5]	extS5[0]	5	1'b0 1'b1	DLL control loop extEarly5 = Register13[7]
14 [5]	extS6[0]	6	1'b0 1'b1	DLL control loop extEarly6 = Register14[7]
15 [5]	extS7[0]	7	1'b0 1'b1	DLL control loop extEarly7 = Register15[7]

Table 141 Phase-shifter DLLs late signals control.

Register [bit]	Signal name	Channel	Value	Late signal driven by
8 [6]	extS0[1]	0	1'b0 1'b1	DLL control loop extLate0 = Register16[6]
9 [6]	extS1[1]	1	1'b0 1'b1	DLL control loop extLate1 = Register17[6]

10 [6]	extS2[1]	2	1'b0 1'b1	DLL control loop extLate2 = Register18[6]
11 [6]	extS3[1]	3	1'b0 1'b1	DLL control loop extLate3 = Register19[6]
12 [6]	extS4[1]	4	1'b0 1'b1	DLL control loop extLate4 = Register20[6]
13 [6]	extS5[1]	5	1'b0 1'b1	DLL control loop extLate5 = Register21[6]
14 [6]	extS6[1]	6	1'b0 1'b1	DLL control loop extLate6 = Register22[6]
15 [6]	extS7[1]	7	1'b0 1'b1	DLL control loop extLate7 = Register23[6]

12.7 Phase-shifter performance

Typical vales for the performance of the phase-shifter are given in Table 142 and Table 143. Table 112 displays the phase-shifter integral and differential non-linearity for the four operating frequencies. In all cases the RMS values is always below one LSB (50 ps) and the maximum deviation is less than 1.03 LSB (51.6 ps).

Table 142 Phase-shifter integral and differential non-linearity

Non-Linearity		Integral Non-Linearity			Differential Non-Linearity		
Frequency	Δt [ps]	σ [ps]	Max [ps]	Min [ps]	σ [ps]	Max [ps]	Min [ps]
40 MHz	50	14.8	51.6	-30.1	16.9	28.5	-31.5
80 MHz	50	11.8	35.6	-23.2	8.2	20.3	-14.6
160 MHz	50	12.8	39.8	-22.9	8.7	18.5	-16.0
320 MHz	50	15.7	47.1	-30.7	17.0	29.7	-30.6

Table 143 displays typical values for the jitter measured on the phase-shifter outputs. The jitter is measured for the simplex transmitter and duplex modes and at the four operating frequencies. Measurements are made for minimum and maximum phase-shifter delay showing that the delay setting has little impact on the results.

Table 143 Phase-shifter jitter

Jitter		Min Delay (0 ns)		Max Delay (25 ns)	
Frequency	Mode	σ [ps]	P-P [ps]	σ [ps]	P-P [ps]
40 MHz	Simplex Tx	3.8	37.0	4.4	40.7
80 MHz	Simplex Tx	3.4	36.7	3.8	37.5
160 MHz	Simplex Tx	4.8	36.2	8.5	48.6
320 MHz	Simplex Tx	13.5	63.0	15.2	67.3
40 MHz	Duplex	3.9	40.9	4.4	45.0
80 MHz	Duplex	3.6	35.0	3.8	37.5
160 MHz	Duplex	5.4	42.3	8.9	49.5
320 MHz	Duplex	13.4	65.7	15.3	70.6

13. E-LINK DRIVERS AND RECEIVERS

The GBTX ePorts use differential signaling for both the transmitters and the receivers

13.1 ePort line receiver

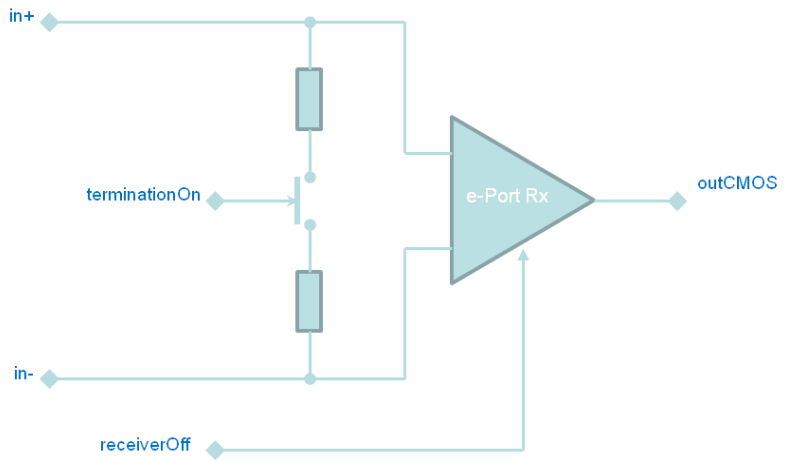


Figure 32 ePort line receiver block diagram

The ePort line receiver is capable of handling two signaling standards, LVDS and SLVS. The LVDS signals are differential with a differential voltage swing of ±400 mV centred on 1.2 V (see the ANSI/TIA/EIA-644-A standard published in 2001 for a full specification of the LVDS signalling). The SLVS standard is also differential but has a reduced voltage swing of ±200 mV centred on 0.2 V (see the JEDEC standard, JESD8-13 for a full specification of SLVS).

The line receiver incorporates a termination resistor that can be programmed to be either a high impedance or 100 Ω. The setting of the termination resistors is made in a per port basis as indicated in tables Table 144 and Table 145. The E-Link line receiver also incorporates a power-down function so that line receivers can be turned off individually when not in use. The receivers are automatically powered on or off every time a phase aligner channel is powered on or off respectively. Please see chapter 3 for details.

The typical power consumption of an ePort line receiver is 0.62 mW

Table 144 Data ports (ePort) Enable receiver termination registers

Register	Register Name	Function
320 [7:0]	inEportCtr192[7:0]	Phase aligner enable termination channels [7:0] Group 0
321 [7:0]	inEportCtr193[7:0]	Phase aligner enable termination channels [7:0] Group 1
322 [7:0]	inEportCtr194[7:0]	Phase aligner enable termination channels [7:0] Group 2
323 [7:0]	inEportCtr195[7:0]	Phase aligner enable termination channels [7:0] Group 3
324 [7:0]	inEportCtr196[7:0]	Phase aligner enable termination channels [7:0] Group 4
325 [7:0]	inEportCtr197[7:0]	Phase aligner enable termination channels [7:0] Group 5
326 [7:0]	inEportCtr198[7:0]	Phase aligner enable termination channels [7:0] Group 6

Table 145 EC channel (ePort) enable receiver termination registers

Register	Register Name	Function
273 [5]	ckCtr4[5]	Phase aligner enable termination EC channel {C,B,A}

13.2 ePort line driver

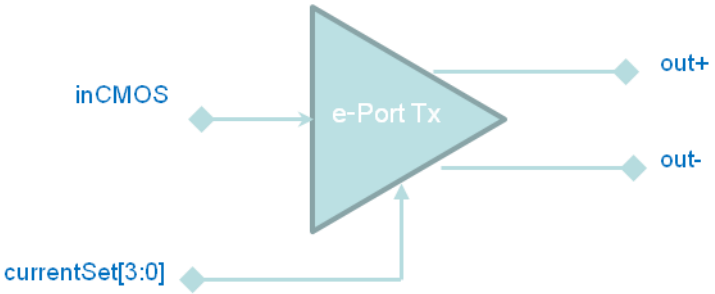


Figure 33 ePort line driver block diagram

The ePort line driver has programmable output current. This feature allows power saving in situations where the clock or data signals are transmitted over very short distances and the driver modulation current can thus be minimized. The modulation current is programmed by the "currentSet [3:0]" signals which also allows the user to power down the circuit when currentSet [3:0] are set to 4'b1111.

When the ePort line driver modulation current is set to 2 mA then the output signal complies with the SLVS standard. In this case typical power consumption is 3.8 mW. Figure 34 represents the measured differential output swing of the line driver as function of the programming settings.

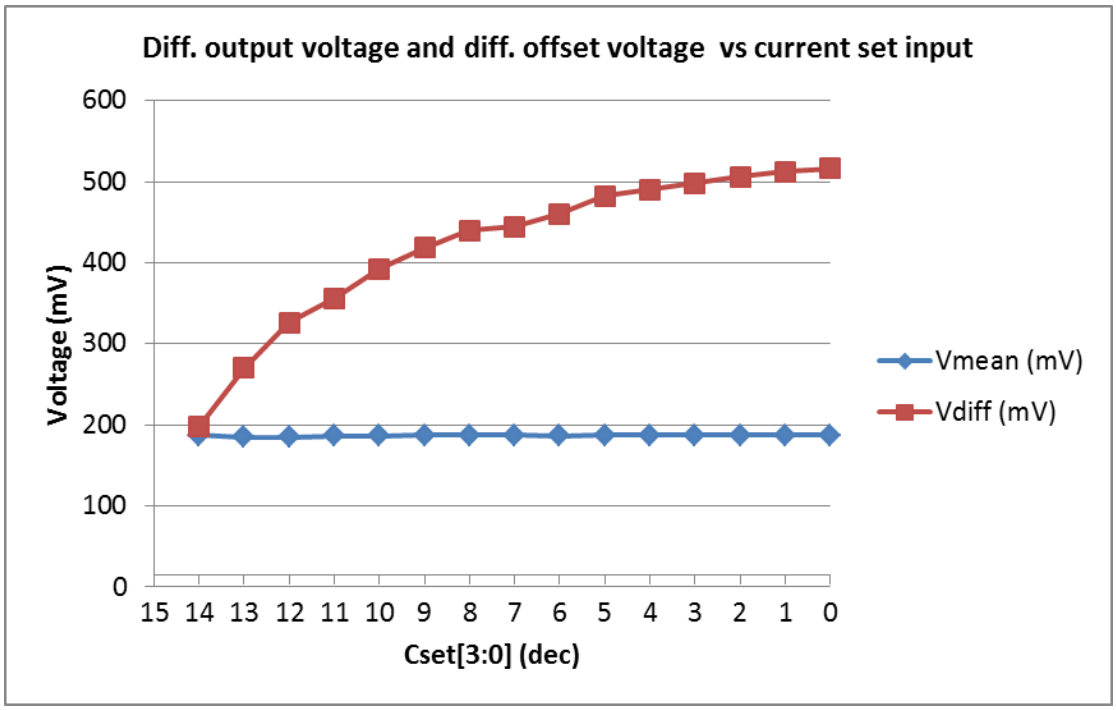


Figure 34 Line driver differential output swing as function of current Cset[3:0]

Figure 35 illustrates two selected cases from Figure 34 in terms of signals swing and common mode.

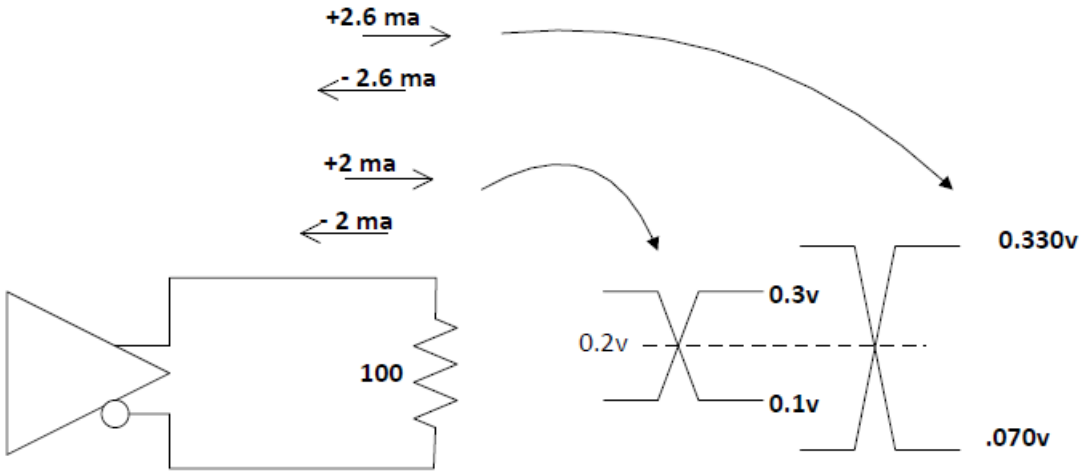


Figure 35 illustrates the differential swing and common mode for two different settings: SLVS levels, cset[3:0] = 10 and Maximum swing cset[3:0] = 00.

The control registers that control the strength of the clock a data drivers are given in Table 146 to Table 148.

Table 146 Clock driver strength

Register	Register Name	Function
329 [7:4]	outEportCtr17[7:4]	Clock driver driveStrength Group 0
330 [3:0]	outEportCtr18[3:0]	Clock driver driveStrength Group 1
330 [7:4]	outEportCtr18[7:4]	Clock driver driveStrength Group 2
331 [3:0]	outEportCtr19[3:0]	Clock driver driveStrength Group 3
331 [7:4]	outEportCtr19[7:4]	Clock driver driveStrength Group 4

Table 147 Data driver strength (one setting per group)

Register	Register Name	Function
327 [3:0]	outEportCtr15[3:0]	Data driver driveStrength Group 0
327 [7:4]	outEportCtr15[7:4]	Data driver driveStrength Group 1
328 [3:0]	outEportCtr16[3:0]	Data driver driveStrength Group 2
328 [7:4]	outEportCtr16[7:4]	Data driver driveStrength Group 3
329 [3:0]	outEportCtr17[3:0]	Data driver driveStrength Group 4

Table 148 Driver strength for EC channel data and clock outputs

Register	Register Name	Function
273 [3:0]	ckCtr4[3:0]	Driver driveStrength EC channel

13.3 ePort Line driver receiver

For the bidirectional ePorts, the line driver and receiver are used as illustrated in Figure 36. The connections between the driver and the receiver are handled inside the GBTX. When used as a driver or receiver (wideBus or 8B10B modes), the user can choose the high impedance or 100 Ω termination, as in 13.1.

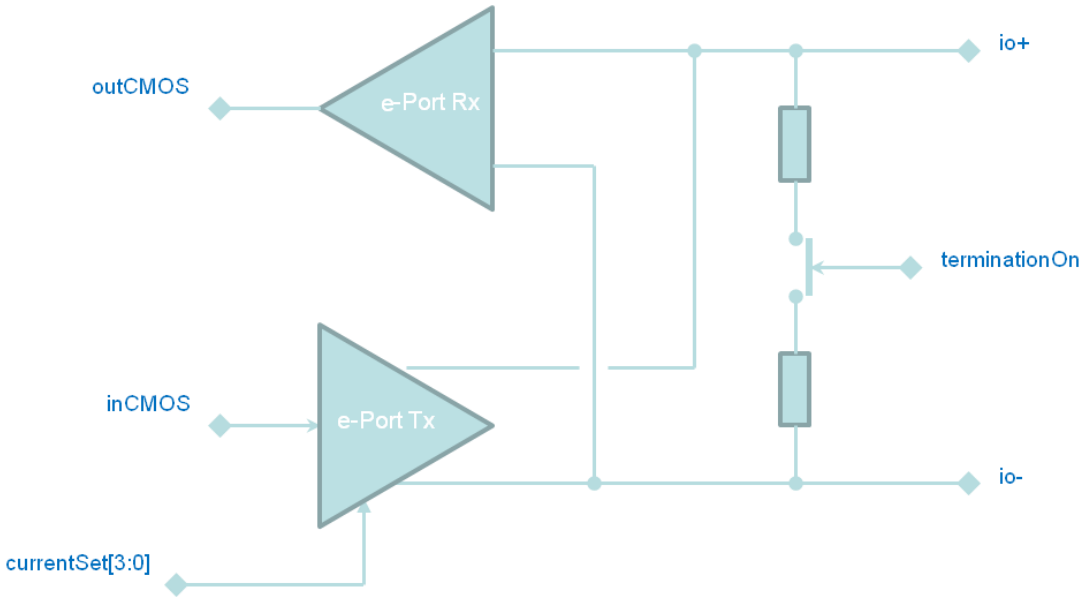


Figure 36 Bidirectional ePort line driver/receiver block diagram.

See Table 144 for information on the 100 Ω termination control registers.

13.4 Using ePorts in multi-drop configuration

Although the GBTX transceiver hasn't been designed with multi-drop capabilities in mind some guidelines are given here to help the user in setting up such systems. It should be noted that these systems depend essentially on the careful design of the interconnects (transmission lines) and thus need to be studied case-by-case by the users. In the case of multiple transmitters on a single line an arbitration system is required to avoid collisions.

13.4.1 Generic

When designing multi-drop systems the engineer must keep in mind that at each signal drop the driver/receiver presents a finite capacitance that will lower locally the equivalent impedance of the line. Multiple drops will thus introduce multiple reflections along the line. Receivers have lower capacitances than transmitters and they will be easier to handle than transmitters. It is possible to adapt the impedance of the line locally so that the lumped capacitance of the driver/transmitter plus the matching network still look like an 100Ω transmission line. At the frequencies the e-Link operates these matching networks require the use of relatively large value inductors that are unlikely to be used in the typical high energy physics environments due to material budget considerations.

13.4.2 Single clock or data driver with multiple receivers

A single GBTX clock or data driver is likely to be used with multiple receivers. In this case the natural configuration is for the driver to be at the beginning end of the transmission line with the receivers are distributed along the line and the

termination impedance at the far end (see Figure 37). In this case a single termination is required. For SLVDS levels the driver should be configure to cset[3:0] = 10.

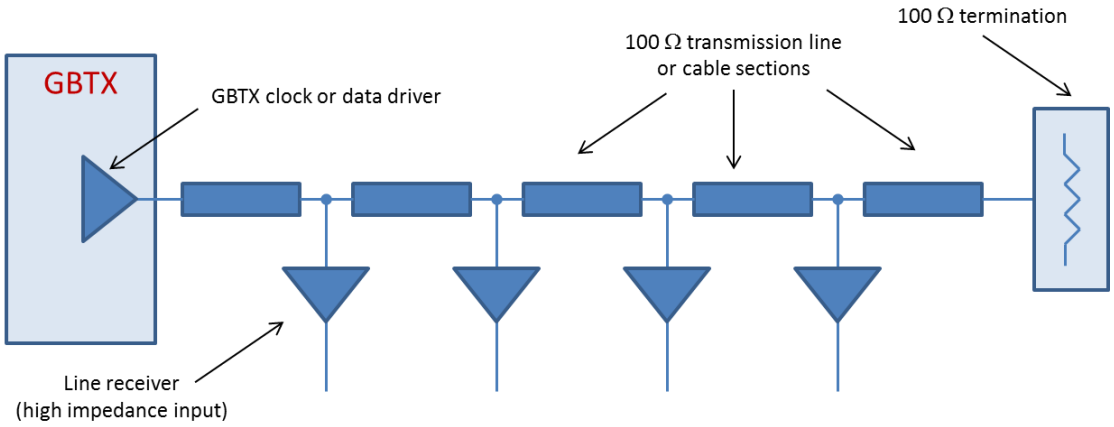


Figure 37 Multi-drop configuration: single driver multiple receiver with single termination.

For some applications it might be required to have the driver somewhere in the middle of the transmission line. In this case the transmission line should be terminated at both ends like what is shown in Figure 38. The driver will see half of the impedance (50Ω) and consequently the voltage swing will be reduced to half. In order to counteract this effect one can increase the buffer driving current. For example, in the single terminated case if Cset[3:0] =0 the voltage swing will be 520 mV while for the double termination case the swing will be 260 mV.

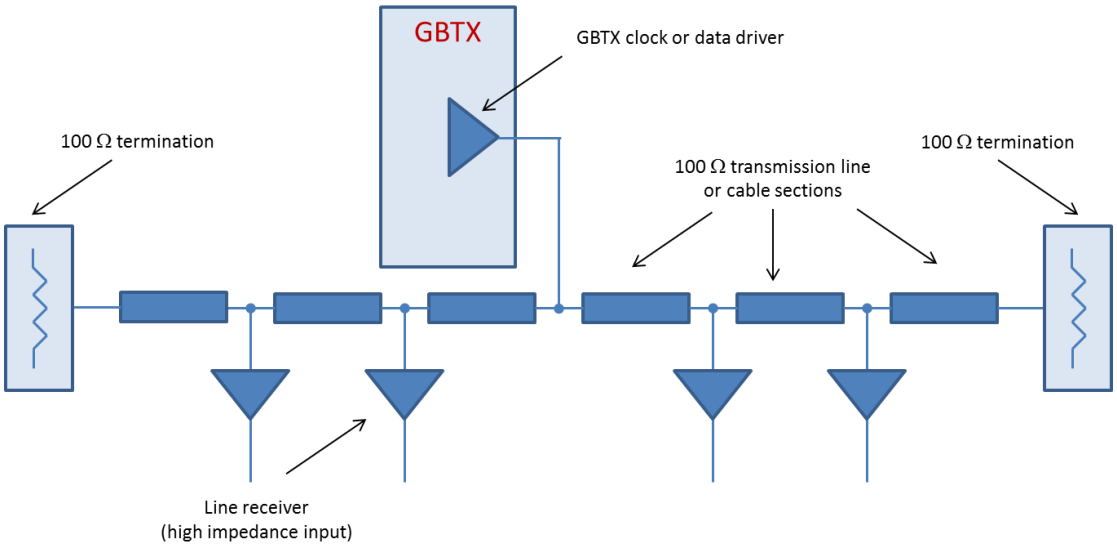


Figure 38 Multi-drop configuration: single driver multiple receiver with double termination.

13.4.3 Multiple driver systems

Multiple driver systems must be double terminated see Figure 39 (with the exception of systems where all the drivers share the driving end of the transmission line). These systems require each driver to have an high impedance state and an arbitration system that decides which one of the drivers

is allowed to drive the line while the others go into the high impedance state freeing the line. Although the GBTX receivers can be set in high impedance state, this cannot be done on an individual basis so this implementation is not practical with the GBTX ASIC.

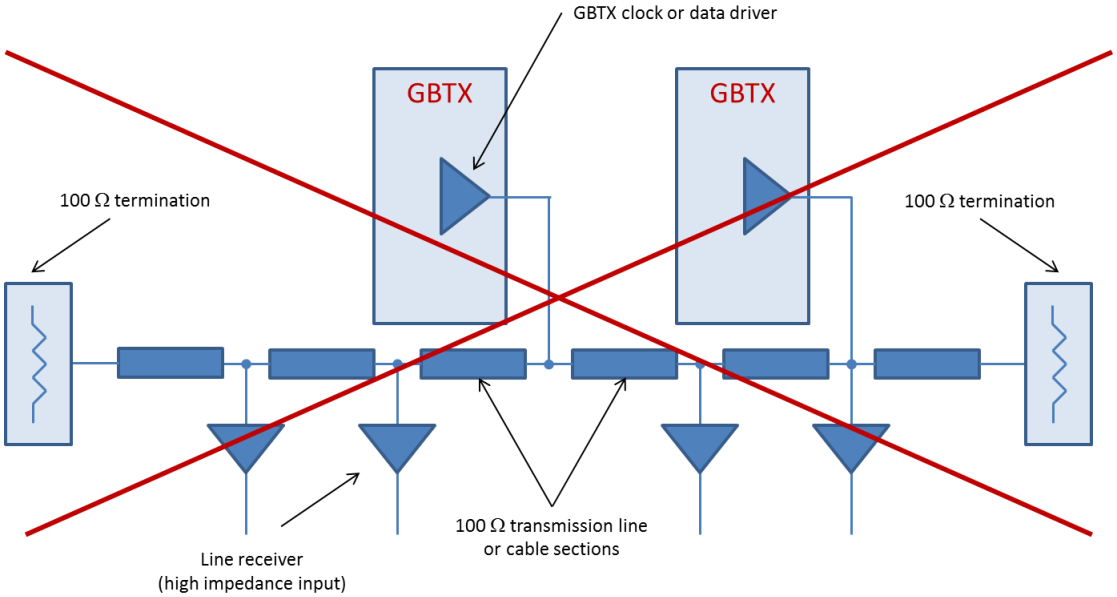


Figure 39 Multiple driver systems

14. REED-SOLOMON ENCODING, DATA SCRAMBLING AND SEU PROTECTION

Detailed information will be added to this chapter on the Forward, scrambling algorithms and on the detailed schemes used in the GBTX to protect against SEU data transmission corruption...

15. E-FUSES

The GBTX is equipped with a number of E-fuses. Each of the bits of the 366 write-able configuration registers has a corresponding E-fuse. The optimal configuration parameters of the GBTX can then be written into the fuse array. The transfer from fuses to configuration registers is done during the automatic power-up sequence. This then allows the GBTX to self-configure into an operational state after power-up.

Each E-fuse connects to a node whose logical state can be sampled and stored in the corresponding register bit. By default, an un-programmed E-fuse will pull this node down to logic 0. If the E-fuse is programmed, the node will be pulled up to logic 1.

The sampling of the fuse values is handled automatically by the power-up state machine. This is described in chapter 10.

15.1 E-fuse addressing

The E-fuses are grouped into 8 fuses to correspond to the 8 bits of each configuration register. Each group of fuses has a 16-bit address which is the same as the address of the corresponding register as listed in Table 153. All of the 8 fuses in one group are programmed together.

There are three special 8-bit fuses listed in Table 149 that are not connected to configuration registers. They can be programmed with the same procedure as the others, and their values can be read via I2C or SC-IC using their fuse addresses. One is called configFuse, and is used in the power-up sequence described in chapter 10, and the other two are test fuses to allow testing of the fuse programming procedure. Their values are not used in any of the GBTX blocks.

Table 149 Special fuses

	Fuse address (dec)	Fuse Address [15:8]	Fuse Address [7:0]
configFuse	366	00000001	01101110
TestFuse1	367	00000001	01101111
TestFuse2	368	00000001	01110000

15.2 E-fuse programming

Each 8-bit fuse is programmed using the I2C interface by first writing the 16-bit address (the same as the corresponding configuration register), loading the 8-bit data pattern and then burning the fuses. The sequence is:

1. Use I2C to load bits[7:0] of the fuse address into the following register: **fuseBlowAddressLSB[7:0] = Register 238[7:0]**.
2. Use I2C to load bits[15:8] of the fuse address into the following register: **fuseBlowAddressMSB[7:0] = Register 239[7:0]**.
3. Use I2C to load the 8-bit pattern to be programmed into the fuses into the following register. Logic 1 will burn the fuse, logic 0 will not burn the fuse. **fuseBlowData[7:0] = Register 240[7:0]**.
4. Switch on EfusePower = 3.3V.
5. Pulse EfuseProgramPulse for 200us.

After programming is complete, EfusePower should be connected to VDD.

On completion of these steps, the corresponding fuses will have been burned. However, the fuse values will only be transferred to the configuration registers if the three LSBs of the configFuse (address 366[2:0]) have been burned to the value 3'b111. When these three bits are set, the power-up state machine will automatically transfer all the fuse values into the corresponding configuration registers in the Update state, as described in Chapter 10. Note that ALL bits of the configuration registers will be loaded with the value of their corresponding fuse, logic 1 (burned) or logic 0 (not burned).

The fuse burning is not a reversible process.

15.3 E-Fuses reliability and radiation hardness

Experience has shown that, when booting the chip from the E-Fuses in approximately 2% of the power-up cycles some of the e-fuse values can be wrongly transferred to the configuration registers. Radiation tests (X-rays) show that, in average, the wrong start-up rate does not increase significantly with radiation. Three stages are observed during the GBTX radiation:

- First, stage, between 0.5 – 10 Mrad: the GBTX experiences the increase of the static leakage current, in the digital logic, with the consequent increase of the chip power consumption. During this stage, the wrong start-up rate drops to approximately 0%;
- Second stage, between 10 – 25 Mrad, while the leakage current effect decreases, the GBTX progressively starts to display bit flips on its fused configuration during the configuration readout. If having at least 1 bit flip is considered as a wrong cycle, start-up fail rate is between 0.5% - 4%;
- Third and final stage, from 25 MRad onwards: the loading of the GBTX configuration registers from the e-fuses behaves almost in the same way as without radiation. When a wrong start-up occurs, the number of bit flips can be high and the fused configuration can be totally corrupted. The GBTx has between 96% and 99.5% chances to have a good start-up, this is very similar to the non-irradiated fail rate.

Therefore, we still recommend to:

- Limit the number of GBTx on the same power supply to less than 10;
- Verify the boot operation by reading back the GBTx configuration after start-up;
- If possible, reset the ASIC instead of a power cycling it. This is a higher reliability process. We thus recommended performing resets instead of power cycles when trying to initialize several GBTX in the same power supply.

16. TESTABILITY

Built in link test features are essential for evaluation, production and in system testing. Moreover, they allow standardized link tests procedures at the system level. The GBTX offers a variety of link test features.

16.1 Data transmission testing

The GBTX transmitter can be programmed to transmit the following:

- A fixed pattern: 80'hAABB BBAA AABB BBAA AABB;
- A repeating and incrementing 16-bit count;
- A pseudo random bit sequence with 2^7-1 run length.

The GBTX receiver can be set to count errors on the received data. In that case, the receiver expects to receive the fixed pattern: 80'hAABB BBAA AABB BBAA AABB and each word received is checked for reception errors. The receiver can count up to 2^{16}-1 errors which can be read from the status registers. The error count is not allowed to overflow and it is cleared each time the error count feature is enabled.

It should be noticed that, due the presence of the scrambler, when a fixed pattern is used to test the data transmission in fact a pseudo random sequence is actually transmitted over the GBT frame. However, since the scrambler can be bypassed it is also possible to send the "raw" fixed pattern imbedded in the GBT frame. Since the fixed pattern transmitted is DC balanced the receiver will have no problem locking to the incoming data stream.

Full details to be added later to the chapter...

16.2 Loopback tests

When the GBTX operates as a full link transceiver it is able to implement extensive loopback tests. These tests can be subdivided in receiver loopback tests and transmitter loopback tests as shown in Figure 40. The former type is mainly for "in system testing", while the latter is mainly for evaluation and production testing of the GBTX itself.

Receiver loopbacks:

- RLC: Testing of the SER and DES;
- RLB: Testing of the receiver chain from the de-serializer input up to the decoder and the transmitter chain from the encoder up to the serializer output;
- RLA: Testing of the receiver chain from the de-serializer input up to the de-scrambler output and testing of the transmitter chain from the scrambler input up to serializer output.

Transmitter loopback:

- TLA: Testing of the transmitter and receiver I/O interface.
- TLB: Testing of the transmitter chain from the ePorts up to the scrambler output and the receiver chain from the de-scrambler input to the ePorts.
- TLC: Testing of the transmitter chain from the ePorts up to the encoder output and the receiver chain from the decoder input to the ePorts.

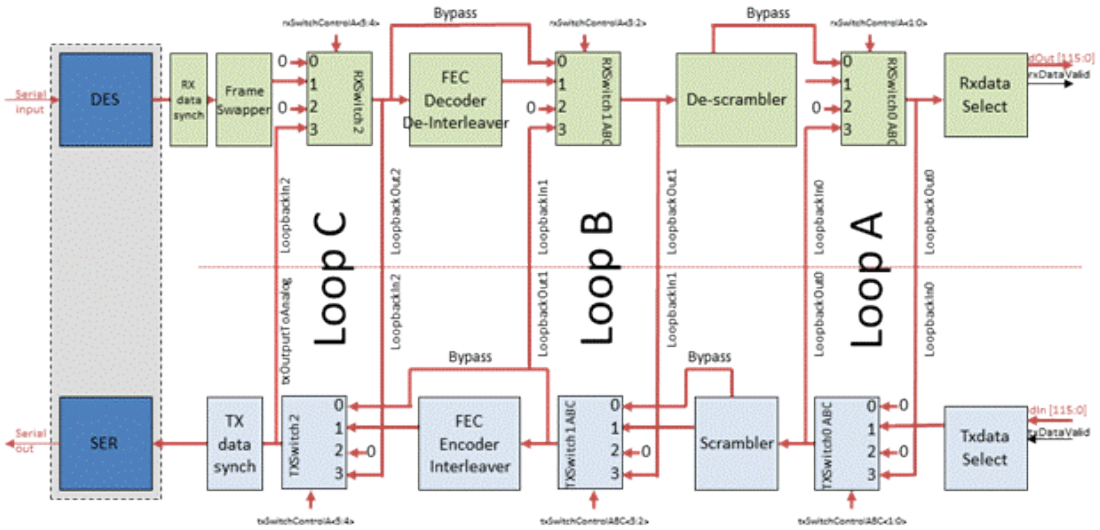


Figure 40 GBTX main loopback paths

16.3 ePort loopbacks and bypass

This section describes additional loopbacks that can be used between the ePorts and how to bypass the ePorts. The functionality is shown in Figure 41. The top of the diagram shows the receiver logic blocks and the bottom the transmitter logic blocks.

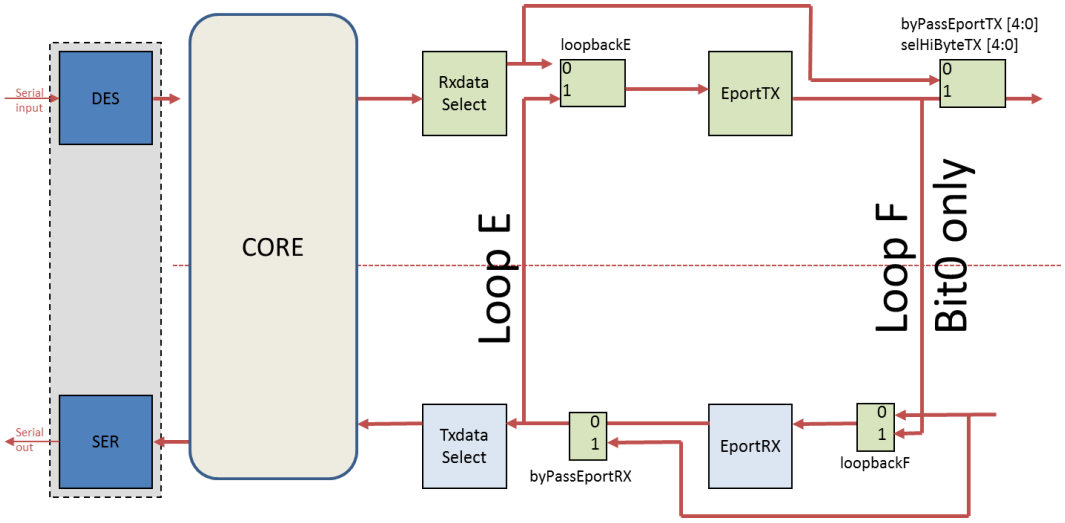


Figure 41 ePort loopbacks

16.3.1 Loop E

This loop is intended for testing the ePort functionality without relying on the rest of the GBTX core. It connects the data input to the TXdataSelect (nominally the output of the ePortRx) to the input of the ePortTx.

The loop is activated by setting the following bits:

loopbackE ABC = inEportCtr172[2:0] = Register 234 [2:0]

16.3.2 Loop F

This loop is intended for testing the serialiser/deserialiser functions of the ePorts. It allows the connection of ePortTx[0] to ePortRx[0]. Note that this affects only channel 0 of the ePorts.

The loop is activated by setting the following bits:

loopbackF ABC = inEportCtr172[5:3] = Register 234[5:3]

16.3.3 Bypass ePortRx

This allows a full bypass of the ePortRx block. The input data word received from the SLVS receivers (dataIn[39:0]) is connected directly to input of the GBTX core. In FEC mode, the GBTX core expects 80 bits of data, so dIn[39:0] is duplicated. When running in wideBus mode (112 bits of data), the extra data bits (dInWideBusMode[15:0]) are also duplicated.
The input data frames to the GBTX core (TXdata) are shown in the table below.

Table 150 Frame transmitted by the GBTX when the ePortRx circuit is bypassed

Mode	Data frame
FEC	TXdata[79:0] = {dIn[39:0], dIn[39:0]}
wideBus	TXdata[111:0] = {dIn[39:0], dIn[39:0], dInWideBusMode[15:0], dInWideBusMode[15:0]}

Note that if the ePortRx bypass feature is to be used, then the ePortRx must be configured in 80 Mbit-mode. This ensures that all the SLVS receivers are enabled (in 160 or 320 Mbit modes, some receivers are automatically disabled).

The ePortRX bypass is activated by setting the following bits:

bypassEportRX ABC = inEportCtr190[2:0] = Register 252 [2:0]

16.3.4 Bypass ePortTx

This allows bypassing the ePortTx and connects the output signals from the GBTX core directly to the SLVS transmitters. This can be done for each individual ePort group. For a group, the eight SLVS transmitters are connected to one byte of the sixteen bits allocated to that group, as described below.

Note that if the ePortTx bypass feature is to be used, then the ePortTx must be configured in 80 Mbit-mode. This ensures that all the SLVS transmitters are enabled (in 160 or 320 Mbit modes, some transmitters are automatically disabled).

The ePortTx bypass per group [4:0] is activated by setting the following bits:

bypassEportTX0 ABC = inEportCtr184[2:0] = Register 246 [2:0]
bypassEportTX1 ABC = inEportCtr184[5:3] = Register 246 [5:3]
bypassEportTX2 ABC = {inEportCtr185[0], inEportCtr184[7:6]}
= {Register 247 [0], Register 246 [7:6]}
bypassEportTX3 ABC = inEportCtr185[3:1] = Register 247 [3:1]
bypassEportTX4 ABC = inEportCtr185[6:4] = Register 247 [6:4]

16.3.4.1 Selecting the Lo or Hi Bytes

Each ePort group normally transmits sixteen data bits on eight (or four or two) SLVS transmitters. When the ePortTx is bypassed, the transmitters can be connected directly to only a single byte of the sixteen data bits. The user can select which byte (Hi or Lo) to connect according the following table.

Table 151 Lo or Hi bytes selection

Group	Configuration Signal	Configuration Register	Data bits transmitted	
			selHiByte=0	selHiByte=1
0	selHiByteTX0ABC	inEportCtr187[2:0] = Register 249[2:0]	[7:0]	[15:8]
1	selHiByteTX1ABC	inEportCtr187[5:3] = Register 249[5:3]	[23:16]	[31:24]
2	selHiByteTX2ABC	{inEportCtr188[0],inEportCtr187[7:6]} {Register 250[0], Register 249[7:6]}	[39:32]	[47:40]
3	selHiByteTX3ABC	inEportCtr188[3:1] = Register 250[3:1]	[55:48]	[63:56]
4	selHiByteTX4ABC	inEportCtr188[6:4] = Register 250[6:4]	[71:64]	[79:72]

16.4 Test input/outputs

The GBTX has the possibility to inject test inputs into certain blocks. It also has some test output signals that can be connected to internal signals. In addition, some of the SLVS input/output pads can be configured for test purposes.

16.4.1 Test Output

The output port **TESTOUTPUT** can be connected to one of a number of different internal signals. These are selected by the following configuration register:

testOutputSelect[7:0] = ckCtr11[7:0] = Register 280[7:0].

The signals are listed below, together with the corresponding value of register 280[7:0], in decimal.

Register 280 value		Register 280 value		Register 280 value	
0	ttcDivideOut	49	channelLockedGroup1[0]	98	powerUpFSMState[0]
1	ttcTestDown	50	channelLockedGroup1[1]	99	powerUpFSMState[1]
2	ttcTestUp	51	channelLockedGroup1[2]	100	powerUpFSMState[2]
3	ttcEarly[0]	52	channelLockedGroup1[3]	101	powerUpFSMState[3]
4	ttcEarly[1]	53	channelLockedGroup1[4]	102	powerUpFSMState[4]
5	ttcEarly[2]	54	channelLockedGroup1[5]	103	EportDataTXA[32]
6	ttcEarly[3]	55	channelLockedGroup1[6]	104	EportDataTXA[33]
7	ttcEarly[4]	56	channelLockedGroup1[7]	105	EportDataTXA[34]
8	ttcEarly[5]	57	channelLockedGroup2[0]	106	EportDataTXA[35]
9	ttcEarly[6]	58	channelLockedGroup2[1]	107	EportDataTXA[36]
10	ttcEarly[7]	59	channelLockedGroup2[2]	108	EportDataTXA[37]
11	ttcLate[0]	60	channelLockedGroup2[3]	109	EportDataTXA[38]
12	ttcLate[1]	61	channelLockedGroup2[4]	110	EportDataTXA[39]
13	ttcLate[2]	62	channelLockedGroup2[5]	111	EportDataTXA[48]
14	ttcLate[3]	63	channelLockedGroup2[6]	112	EportDataTXA[49]
15	ttcLate[4]	64	channelLockedGroup2[7]	113	EportDataTXA[50]
16	ttcLate[5]	65	channelLockedGroup3[0]	114	EportDataTXA[51]
17	ttcLate[6]	66	channelLockedGroup3[1]	115	EportDataTXA[52]
18	ttcLate[7]	67	channelLockedGroup3[2]	116	EportDataTXA[53]
19	txTestUp	68	channelLockedGroup3[3]	117	EportDataTXA[54]
20	txTestDown	69	channelLockedGroup3[4]	118	EportDataTXA[55]
21	SLVSinTest	70	channelLockedGroup3[5]	119	EportDataTXA[64]
22	ePLLRXInstantLock	71	channelLockedGroup3[6]	120	EportDataTXA[65]
23	ePLLTXInstantLock	72	channelLockedGroup3[7]	121	EportDataTXA[66]
24	xPllInstantLock	73	channelLockedGroup4[0]	122	EportDataTXA[67]
25	txInstantLockGated	74	channelLockedGroup4[1]	123	EportDataTXA[68]
26	rxInstantLockRefGated	75	channelLockedGroup4[2]	124	EportDataTXA[69]
27	RXEPLLLocked	76	channelLockedGroup4[3]	125	EportDataTXA[70]
28	TXEPLLLocked	77	channelLockedGroup4[4]	126	EportDataTXA[71]
29	XPLLLocked	78	channelLockedGroup4[5]	127	EportDataTXA[80]
30	txReady_control	79	channelLockedGroup4[6]	128	EportDataTXA[81]
31	rxReady_control	80	channelLockedGroup4[7]	129	EportDataTXA[82]
32	dllLockedGroup[0]	81	channelLockedGroup5[0]	130	EportDataTXA[83]
33	dllLockedGroup[1]	82	channelLockedGroup5[1]	131	EportDataTXA[84]
34	dllLockedGroup[2]	83	channelLockedGroup5[2]	132	EportDataTXA[85]
35	dllLockedGroup[3]	84	channelLockedGroup5[3]	133	EportDataTXA[86]
36	dllLockedGroup[4]	85	channelLockedGroup5[4]	134	EportDataTXA[87]
37	dllLockedGroup[5]	86	channelLockedGroup5[5]	135	EportDataTXA[96]
38	dllLockedGroup[6]:	87	channelLockedGroup5[6]	136	EportDataTXA[97]
39	1'b0	88	channelLockedGroup5[7]	137	EportDataTXA[98]
40	dllLockedEc	89	channelLockedGroup6[0]	138	EportDataTXA[99]
41	channelLockedGroup0[0]	90	channelLockedGroup6[1]	139	EportDataTXA[100]
42	channelLockedGroup0[1]	91	channelLockedGroup6[2]	140	EportDataTXA[101]
43	channelLockedGroup0[2]	92	channelLockedGroup6[3]	141	EportDataTXA[102]
44	channelLockedGroup0[3]	93	channelLockedGroup6[4]	142	EportDataTXA[103]
45	channelLockedGroup0[4]	94	channelLockedGroup6[5]	143	headerInPhase
46	channelLockedGroup0[5]	95	channelLockedGroup6[6]	144	headerInAntiPhase
47	channelLockedGroup0[6]	96	channelLockedGroup6[7]	145	rxSkipCycle
48	channelLockedGroup0[7]	97	channelLockedEc	146	rxSelectDataInPhase

16.4.2 Test Clock Output

The output pad **testClockOut** can be connected to one of a number of different signals from the clock manager. These are selected by the following configuration register:

cmTestMuxSelect[4:0] = ckCtr50[4:0] = Register 319[4:0]

The signals are listed below, together with the corresponding value of register 319[4:0], in decimal.

Register 319 [4:0]	Selected signal
0	iCmTestOut :
1	sampleRx4080A :
2	sampleRx40160A
3	sampleRx40320A
4	sampleRx4040AB
5	sampleRx4040AC
6	sampleRx4080B :
7	sampleRx40160B
8	sampleRx40320B
9	sampleRx4040BA
10	sampleRx4040BC
11	sampleRx4080C
12	sampleRx40160C
13	sampleRx40320C
14	sampleRx4040CA
15	sampleRx4040CB
16	sampleTx4080A
17	sampleTx40160A
18	sampleTx40320A
19	sampleTx4040AB
20	sampleTx4040AC
21	sampleTx4080B
22	sampleTx40160B
23	sampleTx40320B
24	sampleTx4040BA
25	sampleTx4040BC
26	sampleTx4080C
27	sampleTx40160C
28	sampleTx40320C
29	sampleTx4040CA
30	sampleTx4040CB

The signal selected by register 319[7:0] = 0 (iCmTestOut) can be connected to one of another selection of signals. The signal can be selected with the flowing configuration register:

cmTestOutSelect[6:0] = ckCtr14[6:0] = Register 283[6:0].

The signals are listed in the table below according to the setting of register 283, in decimal.

Register 283[6:0]		Register 283[6:0]	
0	zeroWire	29	txTestMux160MHz;
1	refClk	30	txMux320MHz;
2	referenceClock	31	txTestMux320MHz;
3	rxReferenceClock	32	xPll40MHz;
4	rxReferenceTestMux	33	xPll80MHz;
5	rxClk40MHz	34	xPllReferenceClock;
6	rxMux40MHz	35	ePllRx160MHz;
7	rxTestMux40MHz	36	ePllRx320MHz;
8	rxClk80MHz	37	ePllRxReference;
9	rxMux80MHz	38	ePllRxTestMux;
10	rxTestMux80MHz;	39	ePllTx160MHz;
11	rxClk160MHz;	40	ePllTx320MHz;
12	rxMux160MHz;	41	ePllTxReference;
13	rxTestMux160MHz;	42	ePllTxTestMux;
14	rxMux320MHz;	43	psTestMux;
15	rxTestMux320MHz;	44	psReference;
16	rxClkRefPll40MHz;	45	referenceClockRoot40MHz;
17	rxClkRefPll80MHz;	46	rxClockRoot40MHz;
18	rxClkRefPll160MHz;	47	rxClockRoot80MHz;
19	txReferenceClock;	48	rxClockRoot160MHz;
20	txReferenceTestMux;	49	rxClockRoot320MHz;
21	txClk40MHz;	50	txClockRoot40MHz;
22	txMux40MHz;	51	txClockRoot80MHz;
23	txTestMux40MHz;	52	txClockRoot160MHz;
24	txClk80MHz;	53	txClockRoot320MHz;
25	txMux80MHz;	54	cmTestClockIn1;
26	txTestMux80MHz;	55	cmTestClockIn2;
27	txClk160MHz;		
28	txMux160MHz;		

16.5 SLVS test circuit

Input/output pads can be configured for testing the SLVS circuitry. The SLVS output pair dOut[32] is used for this. It can be driven with different signals selected by the SLVSoutTestSel controls.

SLVSoutTestSel[1:0] = ckCtr0[7:6] = Register 0[7:6]

The selection of the signals is shown in Figure 42. dOut32 can be driven by the normal data output, by input dIn32, or by a signal injected into the external CMOS pad SLVSTestInject.

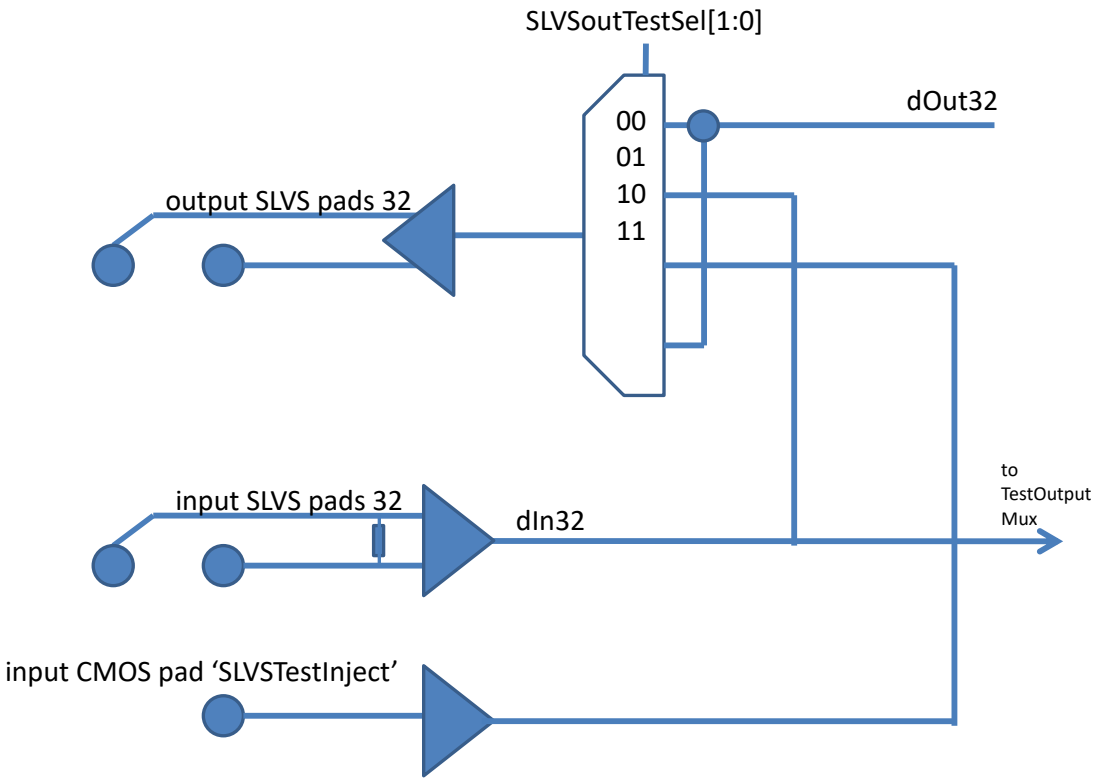


Figure 42. SLVS test circuit topology

16.6 Summary of configuration inputs

The registers listed in Table 152 are used to configure the test ports

Table 152 SLVS test port configuration registers

Register address	Register name	Bits
280	ckCtr11	[7:0]
319	ckCtr50	[4:0]
283	ckCtr14	[6:0]
0	ckCtr0	[7:6]

16.7 Boundary scan

The GBTX has boundary scan capability via its JTAG interface to verify the connections of its hundreds of SLVS and CMOS pins (high speed pins not included). The reader is made aware that the use of normal boundary scan does not allow a through verification of differential signal connections (e.g. one differential pin floating at mid-level will often not be detected with a static boundary scan test.).

16.7.1 Important note on JTAG

User should be aware that, although his/her use of the ASIC might not require the JTAG functionality, the JTAG control pins must be set correctly for normal operation (data transmission). **It is mandatory to set pins TMS and TRESETB to 0V (GND) for normal ASIC operation**.

16.8 Evaluation/production testing

To perform exhaustive production and evaluation tests the GBTX has special testing modes where internal PLL's are bypassed and internal logic can be single stepped (not possible with PLL's).The GBTX also implements extensive bypassing of the logic functions allowing those to be efficiently tested through the ePorts during production testing.

16.8.1 TMR testing

The GBTX logic uses Triple Modular Redundancy (TMR). When testing a TMR circuit there is always a risk that one of the triplicated sections is not working but the TMR logic makes it appear that all is performing well. It is only at the time when an SEU hits one of the functional circuits that the fault is revealed. To test successfully a TMR circuit requires that all the instantiations of the triplicated circuit be tested individually. To avoid this lengthy procedure, in the GBTX is possible to stop individually any of the triplicated clocks. Stopping one of these clocks is like injecting a fault in one of the triplicated sections of the circuit and will result in malfunction if any of the other two circuits (fed by the other two clocks) is defective. To cover fully the TMR logic it is thus just necessary to run the logic tests twice each time with a different triplicated section whose clock is stopped.

Detailed information will be added to this chapter on the GBTX test features and procedures...

17. GBTX REGISTERS

The GBTX contains a number of configuration registers. During the power-up sequence, these registers can be optionally loaded with the values of the e-Fuses. The registers can also be accessed (write or read) through the I2C interface or the SC-IC interface. There is also a number of read-only registers to allow monitoring of the status of certain logic blocks.

17.1 GBTX register access

Figure 43 illustrates the different options for accessing the GBTX registers.

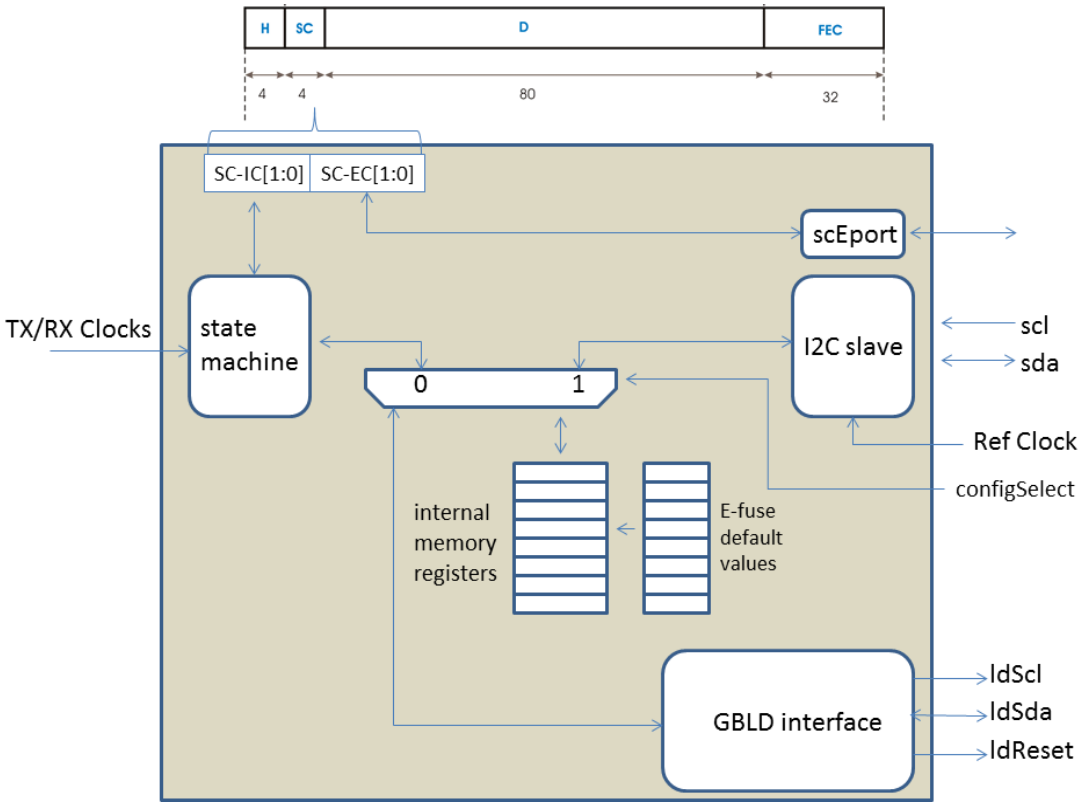


Figure 43 The different options for accessing the GBTX configuration registers.

1. Transfer from e-Fuses to registers. This is done only during the start-up sequence and if the three **update** fuses have been blown.
2. Through the SC-IC interface. The data is provided from the GBT frame and hence requires that the full duplex link is operational to allow write and read access. It uses the TX and RX clocks and hence requires that the serialiser and deserialiser are working correctly. This interface is described in detail in Section 5.1. This option is selected by setting the external pin **configSelect** to 0.
3. Through the I2C interface. This is an asynchronous interface and does not require that the GBTX link is working properly. It does require that the reference clock is present. This interface is described in detail in Section 5.2. This option is selected by setting the external pin **configSelect** to 1.

17.2 GBTX writeable registers

There are in total 366 8-bit registers whose values can be written using the interfaces described above. Each register has a unique 16-bit address. The registers are mostly grouped in address space according to the functional blocks they control.

Some registers have been judged critical to SEU and hence have been triplicated (indicated by labels A, B, C). In this case, the same value should be written in all three registers.
The tables below list the registers and their functions. For reference, each 8-bit register has a name (e.g. ckCtr0).

Table 153 GBTX writeable registers

Block	Manual reference	Memory address	Register Name	bits	Signal name	Function
Clock manager		0	ckCtr0	0	cmEpllRxReferenceSelectA[0]	Selects reference clock A for EPLL-RX
Clock manager				1	cmEpllRxReferenceSelectA[1]	Selects reference clock A for EPLL-RX
Clock manager				2	cmEpllRxReferenceSelectB[0]	Selects reference clock B for EPLL-RX
Clock manager				3	cmEpllRxReferenceSelectB[1]	Selects reference clock B for EPLL-RX
Clock manager				4	cmEpllRxReferenceSelectC[0]	Selects reference clock C for EPLL-RX
Clock manager				5	cmEpllRxReferenceSelectC[1]	Selects reference clock C for EPLL-RX
Test Outputs				6	SLVSoutTestSel[0]	Selects signal to drive SLVS output test (dOut32)
Test Outputs				7	SLVSoutTestSel[1]	Selects signal to drive SLVS output test (dOut32)
Clock manager		1	ckCtr1	0	cmEpllTxReferenceSelectA[0]	Selects reference clock A for EPLL-TX
Clock manager				1	cmEpllTxReferenceSelectA[1]	Selects reference clock A for EPLL-TX
Clock manager				2	cmEpllTxReferenceSelectB[0]	Selects reference clock B for EPLL-TX
Clock manager				3	cmEpllTxReferenceSelectB[1]	Selects reference clock B for EPLL-TX
Clock manager				4	cmEpllTxReferenceSelectC[0]	Selects reference clock C for EPLL-TX
Clock manager				5	cmEpllTxReferenceSelectC[1]	Selects reference clock C for EPLL-TX
				6		unused
				7		unused
Clock manager		2	ckCtr2	0	cmRxReferenceTestMuxSelectA[0]	Selects reference clock A for RX in Test mode
Clock manager				1	cmRxReferenceTestMuxSelectA[1]	Selects reference clock A for RX in Test mode
Clock manager				2	cmRxReferenceTestMuxSelectB[0]	Selects reference clock B for RX in Test mode
Clock manager				3	cmRxReferenceTestMuxSelectB[1]	Selects reference clock B for RX in Test mode
Clock manager				4	cmRxReferenceTestMuxSelectC[0]	Selects reference clock C for RX in Test mode
Clock manager				5	cmRxReferenceTestMuxSelectC[1]	Selects reference clock C for RX in Test mode
				6		unused
				7		unused
Clock manager		3	ckCtr3	0	cmTxReferenceTestMuxSelectA[0]	Selects reference clock A for TX in Test mode
Clock manager				1	cmTxReferenceTestMuxSelectA[1]	Selects reference clock A for TX in Test mode
Clock manager				2	cmTxReferenceTestMuxSelectB[0]	Selects reference clock B for TX in Test mode
Clock manager				3	cmTxReferenceTestMuxSelectB[1]	Selects reference clock B for TX in Test mode
Clock manager				4	cmTxReferenceTestMuxSelectC[0]	Selects reference clock C for TX in Test mode
Clock manager				5	cmTxReferenceTestMuxSelectC[1]	Selects reference clock C for TX in Test mode
				6		unused
				7		unused
IOs		273	ckCtr4	0	scCset[0]	Drive current control for SC-EC Eport
IOs				1	scCset[1]	Drive current control for SC-EC Eport
IOs				2	scCset[2]	Drive current control for SC-EC Eport
IOs				3	scCset[3]	Drive current control for SC-EC Eport
				4		unused
IOs				5	scEnableTermination	Enable termination for SC-EC Eport
				6		unused
				7		unused
Clock manager		274	ckCtr5	0	cmRxPhase40MHzA[0]	Fine phase delay of RX 40MHz clock A
Clock manager				1	cmRxPhase40MHzA[1]	Fine phase delay of RX 40MHz clock A
Clock manager				2	cmRxPhase40MHzA[2]	Fine phase delay of RX 40MHz clock A
Clock manager				3	cmRxPhase40MHzA[3]	Fine phase delay of RX 40MHz clock A
Clock manager				4	cmRxPhase40MHzB[0]	Fine phase delay of RX 40MHz clock B
Clock manager				5	cmRxPhase40MHzB[1]	Fine phase delay of RX 40MHz clock B
Clock manager				6	cmRxPhase40MHzB[2]	Fine phase delay of RX 40MHz clock B
Clock manager				7	cmRxPhase40MHzB[3]	Fine phase delay of RX 40MHz clock B
Clock manager		275	ckCtr6	0	cmRxPhase40MHzC[0]	Fine phase delay of RX 40MHz clock C

Clock manager				1	cmRxPhase40MHzC[1]	Fine phase delay of RX 40MHz clock C
Clock manager				2	cmRxPhase40MHzC[2]	Fine phase delay of RX 40MHz clock C
Clock manager				3	cmRxPhase40MHzC[3]	Fine phase delay of RX 40MHz clock C
Clock manager				4	cmRxPhase80MHzA[0]	Fine phase delay of RX 80MHz clock A
Clock manager				5	cmRxPhase80MHzA[1]	Fine phase delay of RX 80MHz clock A
Clock manager				6	cmRxPhase80MHzA[2]	Fine phase delay of RX 80MHz clock A
Clock manager				7	cmRxPhase80MHzA[3]	Fine phase delay of RX 80MHz clock A
Clock manager		276	ckCtr7	0	cmRxPhase80MHzB[0]	Fine phase delay of RX 80MHz clock B
Clock manager				1	cmRxPhase80MHzB[1]	Fine phase delay of RX 80MHz clock B
Clock manager				2	cmRxPhase80MHzB[2]	Fine phase delay of RX 80MHz clock B
Clock manager				3	cmRxPhase80MHzB[3]	Fine phase delay of RX 80MHz clock B
Clock manager				4	cmRxPhase80MHzC[0]	Fine phase delay of RX 80MHz clock C
Clock manager				5	cmRxPhase80MHzC[1]	Fine phase delay of RX 80MHz clock C
Clock manager				6	cmRxPhase80MHzC[2]	Fine phase delay of RX 80MHz clock C
Clock manager				7	cmRxPhase80MHzC[3]	Fine phase delay of RX 80MHz clock C
Clock manager		277	ckCtr8	0	cmRxTestMuxSelect40A[0]	Selects RX 40MHz clock A in Test mode
Clock manager				1	cmRxTestMuxSelect40A[1]	Selects RX 40MHz clock A in Test mode
Clock manager				2	cmRxTestMuxSelect80A[0]	Selects RX 80MHz clock A in Test mode
Clock manager				3	cmRxTestMuxSelect80A[1]	Selects RX 80MHz clock A in Test mode
Clock manager				4	cmRxTestMuxSelect160A[0]	Selects RX 160MHz clock A in Test mode
Clock manager				5	cmRxTestMuxSelect160A[1]	Selects RX 160MHz clock A in Test mode
Clock manager				6	cmRxTestMuxSelect320A[0]	Selects RX 320MHz clock A in Test mode
Clock manager				7	cmRxTestMuxSelect320A[1]	Selects RX 320MHz clock A in Test mode
Clock manager		278	ckCtr9	0	cmRxTestMuxSelect40B[0]	Selects RX 40MHz clock B in Test mode
Clock manager				1	cmRxTestMuxSelect40B[1]	Selects RX 40MHz clock B in Test mode
Clock manager				2	cmRxTestMuxSelect80B[0]	Selects RX 80MHz clock B in Test mode
Clock manager				3	cmRxTestMuxSelect80B[1]	Selects RX 80MHz clock B in Test mode
Clock manager				4	cmRxTestMuxSelect160B[0]	Selects RX 160MHz clock B in Test mode
Clock manager				5	cmRxTestMuxSelect160B[1]	Selects RX 160MHz clock B in Test mode
Clock manager				6	cmRxTestMuxSelect320B[0]	Selects RX 320MHz clock B in Test mode
Clock manager				7	cmRxTestMuxSelect320B[1]	Selects RX 320MHz clock B in Test mode
Clock manager		279	ckCtr10	0	cmRxTestMuxSelect40C[0]	Selects RX 40MHz clock C in Test mode
Clock manager				1	cmRxTestMuxSelect40C[1]	Selects RX 40MHz clock C in Test mode
Clock manager				2	cmRxTestMuxSelect80C[0]	Selects RX 80MHz clock C in Test mode
Clock manager				3	cmRxTestMuxSelect80C[1]	Selects RX 80MHz clock C in Test mode
Clock manager				4	cmRxTestMuxSelect160C[0]	Selects RX 160MHz clock C in Test mode
Clock manager				5	cmRxTestMuxSelect160C[1]	Selects RX 160MHz clock C in Test mode
Clock manager				6	cmRxTestMuxSelect320C[0]	Selects RX 320MHz clock C in Test mode
Clock manager				7	cmRxTestMuxSelect320C[1]	Selects RX 320MHz clock C in Test mode
Test outputs		280	ckCtr11	0	testOutputSelect[0]	Selects the signal to drive TestOutput
Test outputs				1	testOutputSelect[1]	Selects the signal to drive TestOutput
Test outputs				2	testOutputSelect[2]	Selects the signal to drive TestOutput
Test outputs				3	testOutputSelect[3]	Selects the signal to drive TestOutput
Test outputs				4	testOutputSelect[4]	Selects the signal to drive TestOutput
Test outputs				5	testOutputSelect[5]	Selects the signal to drive TestOutput
Test outputs				6	testOutputSelect[6]	Selects the signal to drive TestOutput
Test outputs				7	testOutputSelect[7]	Selects the signal to drive testOutput
Clock manager		281	ckCtr12	0	cmReferenceClockSelectA[0]	Selects global reference clock A
Clock manager				1	cmReferenceClockSelectA[1]	Selects global reference clock A
Clock manager				2	cmReferenceClockSelectB[0]	Selects global reference clock B
Clock manager				3	cmReferenceClockSelectB[1]	Selects global reference clock B
Clock manager				4	cmReferenceClockSelectC[0]	Selects global reference clock C
Clock manager				5	cmReferenceClockSelectC[1]	Selects global reference clock C
				6		unused
				7		unused
Clock manager		282	ckCtr13	0	cmPsReferenceSelectA[0]	Selects Phase-Shifter reference clock A
Clock manager				1	cmPsReferenceSelectA[1]	Selects Phase-Shifter reference clock A
Clock manager				2	cmPsReferenceSelectB[0]	Selects Phase-Shifter reference clock B
Clock manager				3	cmPsReferenceSelectB[1]	Selects Phase-Shifter reference clock B
Clock manager				4	cmPsReferenceSelectC[0]	Selects Phase-Shifter reference clock C

Clock manager				5	cmPsReferenceSelectC[1]	Selects Phase-Shifter reference clock C
				6		unused
				7		unused
Clock manager		283	ckCtr14	0	cmTestOutSelect[0]	Selects which signal to send to testClockOut from first subset
Clock manager				1	cmTestOutSelect[1]	Selects which signal to send to testClockOut from first subset
Clock manager				2	cmTestOutSelect[2]	Selects which signal to send to testClockOut from first subset
Clock manager				3	cmTestOutSelect[3]	Selects which signal to send to testClockOut from first subset
Clock manager				4	cmTestOutSelect[4]	Selects which signal to send to testClockOut from first subset
Clock manager				5	cmTestOutSelect[5]	Selects which signal to send to testClockOut from first subset
Clock manager				6	cmTestOutSelect[6]	Selects which signal to send to testClockOut from first subset
				7		unused
Clock manager		284	ckCtr15	0	cmTxPhase40MHzA[0]	Fine phase delay of TX 40MHz clock A
Clock manager				1	cmTxPhase40MHzA[1]	Fine phase delay of TX 40MHz clock A
Clock manager				2	cmTxPhase40MHzA[2]	Fine phase delay of TX 40MHz clock A
Clock manager				3	cmTxPhase40MHzA[3]	Fine phase delay of TX 40MHz clock A
Clock manager				4	cmTxPhase40MHzB[0]	Fine phase delay of TX 40MHz clock B
Clock manager				5	cmTxPhase40MHzB[1]	Fine phase delay of TX 40MHz clock B
Clock manager				6	cmTxPhase40MHzB[2]	Fine phase delay of TX 40MHz clock B
Clock manager				7	cmTxPhase40MHzB[3]	Fine phase delay of TX 40MHz clock B
Clock manager		285	ckCtr16	0	cmTxPhase40MHzC[0]	Fine phase delay of TX 40MHz clock C
Clock manager				1	cmTxPhase40MHzC[1]	Fine phase delay of TX 40MHz clock C
Clock manager				2	cmTxPhase40MHzC[2]	Fine phase delay of TX 40MHz clock C
Clock manager				3	cmTxPhase40MHzC[3]	Fine phase delay of TX 40MHz clock C
Clock manager				4	cmTxPhase80MHzA[0]	Fine phase delay of TX 80MHz clock A
Clock manager				5	cmTxPhase80MHzA[1]	Fine phase delay of TX 80MHz clock A
Clock manager				6	cmTxPhase80MHzA[2]	Fine phase delay of TX 80MHz clock A
Clock manager				7	cmTxPhase80MHzA[3]	Fine phase delay of TX 80MHz clock A
Clock manager		286	ckCtr17	0	cmTxPhase80MHzB[0]	Fine phase delay of TX 80MHz clock B
Clock manager				1	cmTxPhase80MHzB[1]	Fine phase delay of TX 80MHz clock B
Clock manager				2	cmTxPhase80MHzB[2]	Fine phase delay of TX 80MHz clock B
Clock manager				3	cmTxPhase80MHzB[3]	Fine phase delay of TX 80MHz clock B
Clock manager				4	cmTxPhase80MHzC[0]	Fine phase delay of TX 80MHz clock C
Clock manager				5	cmTxPhase80MHzC[1]	Fine phase delay of TX 80MHz clock C
Clock manager				6	cmTxPhase80MHzC[2]	Fine phase delay of TX 80MHz clock C
Clock manager				7	cmTxPhase80MHzC[3]	Fine phase delay of TX 80MHz clock C
Clock manager		287	ckCtr18	0	cmTxTestMuxSelect40A[0]	Selects TX 40MHz clock A in Test mode
Clock manager				1	cmTxTestMuxSelect40A[1]	Selects TX 40MHz clock A in Test mode
Clock manager				2	cmTxTestMuxSelect80A[0]	Selects TX 80MHz clock A in Test mode
Clock manager				3	cmTxTestMuxSelect80A[1]	Selects TX 80MHz clock A in Test mode
Clock manager				4	cmTxTestMuxSelect160A[0]	Selects TX 160MHz clock A in Test mode
Clock manager				5	cmTxTestMuxSelect160A[1]	Selects TX 160MHz clock A in Test mode
Clock manager				6	cmTxTestMuxSelect320A[0]	Selects TX 320MHz clock A in Test mode
Clock manager				7	cmTxTestMuxSelect320A[1]	Selects TX 320MHz clock A in Test mode
Clock manager		288	ckCtr19	0	cmTxTestMuxSelect40B[0]	Selects TX 40MHz clock B in Test mode
Clock manager				1	cmTxTestMuxSelect40B[1]	Selects TX 40MHz clock B in Test mode
Clock manager				2	cmTxTestMuxSelect80B[0]	Selects TX 80MHz clock B in Test mode
Clock manager				3	cmTxTestMuxSelect80B[1]	Selects TX 80MHz clock B in Test mode
Clock manager				4	cmTxTestMuxSelect160B[0]	Selects TX 160MHz clock B in Test mode
Clock manager				5	cmTxTestMuxSelect160B[1]	Selects TX 160MHz clock B in Test mode
Clock manager				6	cmTxTestMuxSelect320B[0]	Selects TX 320MHz clock B in Test mode
Clock manager				7	cmTxTestMuxSelect320B[1]	Selects TX 320MHz clock B in Test mode
Clock manager		289	ckCtr20	0	cmTxTestMuxSelect40C[0]	Selects TX 40MHz clock C in Test mode
Clock manager				1	cmTxTestMuxSelect40C[1]	Selects TX 40MHz clock C in Test mode
Clock manager				2	cmTxTestMuxSelect80C[0]	Selects TX 80MHz clock C in Test mode
Clock manager				3	cmTxTestMuxSelect80C[1]	Selects TX 80MHz clock C in Test mode
Clock manager				4	cmTxTestMuxSelect160C[0]	Selects TX 160MHz clock C in Test mode

Clock manager				5	cmTxTestMuxSelect160C[1]	Selects TX 160MHz clock C in Test mode
Clock manager				6	cmTxTestMuxSelect320C[0]	Selects TX 320MHz clock C in Test mode
Clock manager				7	cmTxTestMuxSelect320C[1]	Selects TX 320MHz clock C in Test mode
Clock manager		290	ckCtr21	0	cmXpllReferenceSelectA[0]	Selects Xpll reference clock A
Clock manager				1	cmXpllReferenceSelectA[1]	Selects Xpll reference clock A
Clock manager				2	cmXpllReferenceSelectB[0]	Selects Xpll reference clock B
Clock manager				3	cmXpllReferenceSelectB[1]	Selects Xpll reference clock B
Clock manager				4	cmXpllReferenceSelectC[0]	Selects Xpll reference clock C
Clock manager				5	cmXpllReferenceSelectC[1]	Selects Xpll reference clock C
				6		unused
				7		unused
EPLL-TX		291	ckCtr22	0	ePllTxPhase320MHzA[0]	Sets EPLL-TX 320MHz phase A
EPLL-TX				1	ePllTxPhase320MHzA[1]	Sets EPLL-TX 320MHz phase A
EPLL-TX				2	ePllTxPhase320MHzA[2]	Sets EPLL-TX 320MHz phase A
EPLL-TX				3	ePllTxPhase320MHzA[3]	Sets EPLL-TX 320MHz phase A
EPLL-TX				4	ePllTxPhase320MHzB[0]	Sets EPLL-TX 320MHz phase B
EPLL-TX				5	ePllTxPhase320MHzB[1]	Sets EPLL-TX 320MHz phase B
EPLL-TX				6	ePllTxPhase320MHzB[2]	Sets EPLL-TX 320MHz phase B
EPLL-TX				7	ePllTxPhase320MHzB[3]	Sets EPLL-TX 320MHz phase B
EPLL-TX		292	ckCtr23	0	ePllTxPhase320MHzC[0]	Sets EPLL-TX 320MHz phase C
EPLL-TX				1	ePllTxPhase320MHzC[1]	Sets EPLL-TX 320MHz phase C
EPLL-TX				2	ePllTxPhase320MHzC[2]	Sets EPLL-TX 320MHz phase C
EPLL-TX				3	ePllTxPhase320MHzC[3]	Sets EPLL-TX 320MHz phase C
EPLL-RX				4	ePllRxPhase320MHzC[0]	Sets EPLL-RX 320MHz phase C
EPLL-RX				5	ePllRxPhase320MHzC[1]	Sets EPLL-RX 320MHz phase C
EPLL-RX				6	ePllRxPhase320MHzC[2]	Sets EPLL-RX 320MHz phase C
EPLL-RX				7	ePllRxPhase320MHzC[3]	Sets EPLL-RX 320MHz phase C
EPLL-TX		293	ckCtr24	0	ePllTxEnablePhaseA[0]	Enable EPLL-TX phase A
EPLL-TX				1	ePllTxEnablePhaseA[1]	Enable EPLL-TX phase A
EPLL-TX				2	ePllTxEnablePhaseA[2]	Enable EPLL-TX phase A
EPLL-TX				3	ePllTxEnablePhaseA[3]	Enable EPLL-TX phase A
EPLL-TX				4	ePllTxEnablePhaseA[4]	Enable EPLL-TX phase A
EPLL-TX				5	ePllTxEnablePhaseA[5]	Enable EPLL-TX phase A
EPLL-TX				6	ePllTxEnablePhaseA[6]	Enable EPLL-TX phase A
EPLL-TX				7	ePllTxEnablePhaseA[7]	Enable EPLL-TX phase A
EPLL-TX		294	ckCtr25	0	ePllTxEnablePhaseB[0]	Enable EPLL-TX phase B
EPLL-TX				1	ePllTxEnablePhaseB[1]	Enable EPLL-TX phase B
EPLL-TX				2	ePllTxEnablePhaseB[2]	Enable EPLL-TX phase B
EPLL-TX				3	ePllTxEnablePhaseB[3]	Enable EPLL-TX phase B
EPLL-TX				4	ePllTxEnablePhaseB[4]	Enable EPLL-TX phase B
EPLL-TX				5	ePllTxEnablePhaseB[5]	Enable EPLL-TX phase B
EPLL-TX				6	ePllTxEnablePhaseB[6]	Enable EPLL-TX phase B
EPLL-TX				7	ePllTxEnablePhaseB[7]	Enable EPLL-TX phase B
EPLL-TX		295	ckCtr26	0	ePllTxEnablePhaseC[0]	Enable EPLL-TX phase C
EPLL-TX				1	ePllTxEnablePhaseC[1]	Enable EPLL-TX phase C
EPLL-TX				2	ePllTxEnablePhaseC[2]	Enable EPLL-TX phase C
EPLL-TX				3	ePllTxEnablePhaseC[3]	Enable EPLL-TX phase C
EPLL-TX				4	ePllTxEnablePhaseC[4]	Enable EPLL-TX phase C
EPLL-TX				5	ePllTxEnablePhaseC[5]	Enable EPLL-TX phase C
EPLL-TX				6	ePllTxEnablePhaseC[6]	Enable EPLL-TX phase C
EPLL-TX				7	ePllTxEnablePhaseC[7]	Enable EPLL-TX phase C
EPLL-TX		296	ckCtr27	0	ePllTxPhase160MHzA[0]	Sets EPLL-TX 160MHz phase A
EPLL-TX				1	ePllTxPhase160MHzA[1]	Sets EPLL-TX 160MHz phase A
EPLL-TX				2	ePllTxPhase160MHzA[2]	Sets EPLL-TX 160MHz phase A
EPLL-TX				3	ePllTxPhase160MHzA[3]	Sets EPLL-TX 160MHz phase A
EPLL-TX				4	ePllTxPhase160MHzA[4]	Sets EPLL-TX 160MHz phase A
EPLL-TX				5	ePllTxCapA[0]	Sets EPLL-TX Cap A in filter
EPLL-TX				6	ePllTxCapA[1]	Sets EPLL-TX Cap A in filter
				7		unused
EPLL-TX		297	ckCtr28	0	ePllTxPhase160MHzB[0]	Sets EPLL-TX 160MHz phase B

EPLL-TX				1	ePllTxPhase160MHzB[1]	Sets EPLL-TX 160MHz phase B
EPLL-TX				2	ePllTxPhase160MHzB[2]	Sets EPLL-TX 160MHz phase B
EPLL-TX				3	ePllTxPhase160MHzB[3]	Sets EPLL-TX 160MHz phase B
EPLL-TX				4	ePllTxPhase160MHzB[4]	Sets EPLL-TX 160MHz phase B
EPLL-TX				5	ePllTxCapB[0]	Sets EPLL-TX Cap B in filter
EPLL-TX				6	ePllTxCapB[1]	Sets EPLL-TX Cap B in filter
				7		unused
EPLL-TX		298	ckCtr29	0	ePllTxPhase160MHzC[0]	Sets EPLL-TX 160MHz phase C
EPLL-TX				1	ePllTxPhase160MHzC[1]	Sets EPLL-TX 160MHz phase C
EPLL-TX				2	ePllTxPhase160MHzC[2]	Sets EPLL-TX 160MHz phase C
EPLL-TX				3	ePllTxPhase160MHzC[3]	Sets EPLL-TX 160MHz phase C
EPLL-TX				4	ePllTxPhase160MHzC[4]	Sets EPLL-TX 160MHz phase C
EPLL-TX				5	ePllTxCapC[0]	Sets EPLL-TX Cap C in filter
EPLL-TX				6	ePllTxCapC[1]	Sets EPLL-TX Cap C in filter
				7		unused
EPLL-TX		299	ckCtr30	0	ePllTxIcpA[0]	Sets EPLL-TX charge pump current A
EPLL-TX				1	ePllTxIcpA[1]	Sets EPLL-TX charge pump current A
EPLL-TX				2	ePllTxIcpA[2]	Sets EPLL-TX charge pump current A
EPLL-TX				3	ePllTxIcpA[3]	Sets EPLL-TX charge pump current A
EPLL-TX				4	ePllTxResA[0]	Sets EPLL-TX Res A in filter
EPLL-TX				5	ePllTxResA[1]	Sets EPLL-TX Res A in filter
EPLL-TX				6	ePllTxResA[2]	Sets EPLL-TX Res A in filter
EPLL-TX				7	ePllTxResA[3]	Sets EPLL-TX Res A in filter
EPLL-TX		300	ckCtr31	0	ePllTxIcpB[0]	Sets EPLL-TX charge pump current B
EPLL-TX				1	ePllTxIcpB[1]	Sets EPLL-TX charge pump current B
EPLL-TX				2	ePllTxIcpB[2]	Sets EPLL-TX charge pump current B
EPLL-TX				3	ePllTxIcpB[3]	Sets EPLL-TX charge pump current B
EPLL-TX				4	ePllTxResB[0]	Sets EPLL-TX Res B in filter
EPLL-TX				5	ePllTxResB[1]	Sets EPLL-TX Res B in filter
EPLL-TX				6	ePllTxResB[2]	Sets EPLL-TX Res B in filter
EPLL-TX				7	ePllTxResB[3]	Sets EPLL-TX Res B in filter
EPLL-TX		301	ckCtr32	0	ePllTxIcpC[0]	Sets EPLL-TX charge pump current C
EPLL-TX				1	ePllTxIcpC[1]	Sets EPLL-TX charge pump current C
EPLL-TX				2	ePllTxIcpC[2]	Sets EPLL-TX charge pump current C
EPLL-TX				3	ePllTxIcpC[3]	Sets EPLL-TX charge pump current C
EPLL-TX				4	ePllTxResC[0]	Sets EPLL-TX Res C in filter
EPLL-TX				5	ePllTxResC[1]	Sets EPLL-TX Res C in filter
EPLL-TX				6	ePllTxResC[2]	Sets EPLL-TX Res C in filter
EPLL-TX				7	ePllTxResC[3]	Sets EPLL-TX Res C in filter
EPLL-RX		302	ckCtr33	0	ePllRxPhase320MHzA[0]	Sets EPLL-RX 320MHz phase A
EPLL-RX				1	ePllRxPhase320MHzA[1]	Sets EPLL-RX 320MHz phase A
EPLL-RX				2	ePllRxPhase320MHzA[2]	Sets EPLL-RX 320MHz phase A
EPLL-RX				3	ePllRxPhase320MHzA[3]	Sets EPLL-RX 320MHz phase A
EPLL-RX				4	ePllRxPhase320MHzB[0]	Sets EPLL-RX 320MHz phase B
EPLL-RX				5	ePllRxPhase320MHzB[1]	Sets EPLL-RX 320MHz phase B
EPLL-RX				6	ePllRxPhase320MHzB[2]	Sets EPLL-RX 320MHz phase B
EPLL-RX				7	ePllRxPhase320MHzB[3]	Sets EPLL-RX 320MHz phase B
EPLL-TX		303	ckCtr34	0	ePllTxResetA	Reset of EPLL-TX filter A
EPLL-TX				1	ePllTxResetB	Reset of EPLL-TX filter B
EPLL-TX				2	ePllTxResetC	Reset of EPLL-TX filter C
				3		unused
EPLL-RX				4	ePllRxResetA	Reset of EPLL-RX filter A
EPLL-RX				5	ePllRxResetB	Reset of EPLL-RX filter B
EPLL-RX				6	ePllRxResetC	Reset of EPLL-RX filter C
				7		unused
EPLL-RX		304	ckCtr35	0	ePllRxEnablePhaseA[0]	Enable EPLL-RX phase A
EPLL-RX				1	ePllRxEnablePhaseA[1]	Enable EPLL-RX phase A
EPLL-RX				2	ePllRxEnablePhaseA[2]	Enable EPLL-RX phase A
EPLL-RX				3	ePllRxEnablePhaseA[3]	Enable EPLL-RX phase A
EPLL-RX				4	ePllRxEnablePhaseA[4]	Enable EPLL-RX phase A

EPLL-RX				5	ePllRxEnablePhaseA[5]	Enable EPLL-RX phase A
EPLL-RX				6	ePllRxEnablePhaseA[6]	Enable EPLL-RX phase A
EPLL-RX				7	ePllRxEnablePhaseA[7]	Enable EPLL-RX phase A
EPLL-RX		305	ckCtr36	0	ePllRxEnablePhaseB[0]	Enable EPLL-RX phase B
EPLL-RX				1	ePllRxEnablePhaseB[1]	Enable EPLL-RX phase B
EPLL-RX				2	ePllRxEnablePhaseB[2]	Enable EPLL-RX phase B
EPLL-RX				3	ePllRxEnablePhaseB[3]	Enable EPLL-RX phase B
EPLL-RX				4	ePllRxEnablePhaseB[4]	Enable EPLL-RX phase B
EPLL-RX				5	ePllRxEnablePhaseB[5]	Enable EPLL-RX phase B
EPLL-RX				6	ePllRxEnablePhaseB[6]	Enable EPLL-RX phase B
EPLL-RX				7	ePllRxEnablePhaseB[7]	Enable EPLL-RX phase B
EPLL-RX		306	ckCtr37	0	ePllRxEnablePhaseC[0]	Enable EPLL-RX phase C
EPLL-RX				1	ePllRxEnablePhaseC[1]	Enable EPLL-RX phase C
EPLL-RX				2	ePllRxEnablePhaseC[2]	Enable EPLL-RX phase C
EPLL-RX				3	ePllRxEnablePhaseC[3]	Enable EPLL-RX phase C
EPLL-RX				4	ePllRxEnablePhaseC[4]	Enable EPLL-RX phase C
EPLL-RX				5	ePllRxEnablePhaseC[5]	Enable EPLL-RX phase C
EPLL-RX				6	ePllRxEnablePhaseC[6]	Enable EPLL-RX phase C
EPLL-RX				7	ePllRxEnablePhaseC[7]	Enable EPLL-RX phase C
EPLL-RX		307	ckCtr38	0	ePllRxPhase160MHzA[0]	Sets EPLL-RX 160MHz phase A
EPLL-RX				1	ePllRxPhase160MHzA[1]	Sets EPLL-RX 160MHz phase A
EPLL-RX				2	ePllRxPhase160MHzA[2]	Sets EPLL-RX 160MHz phase A
EPLL-RX				3	ePllRxPhase160MHzA[3]	Sets EPLL-RX 160MHz phase A
EPLL-RX				4	ePllRxPhase160MHzA[4]	Sets EPLL-RX 160MHz phase A
EPLL-RX				5	ePllRxCapA[0]	Sets EPLL-RX Cap A in filter
EPLL-RX				6	ePllRxCapA[1]	Sets EPLL-RX Cap A in filter
				7		unused
EPLL-RX		308	ckCtr39	0	ePllRxPhase160MHzB[0]	Sets EPLL-RX 160MHz phase B
EPLL-RX				1	ePllRxPhase160MHzB[1]	Sets EPLL-RX 160MHz phase B
EPLL-RX				2	ePllRxPhase160MHzB[2]	Sets EPLL-RX 160MHz phase B
EPLL-RX				3	ePllRxPhase160MHzB[3]	Sets EPLL-RX 160MHz phase B
EPLL-RX				4	ePllRxPhase160MHzB[4]	Sets EPLL-RX 160MHz phase B
EPLL-RX				5	ePllRxCapB[0]	Sets EPLL-RX Cap B in filter
EPLL-RX				6	ePllRxCapB[1]	Sets EPLL-RX Cap B in filter
				7		unused
EPLL-RX		309	ckCtr40	0	ePllRxPhase160MHzC[0]	Sets EPLL-RX 160MHz phase C
EPLL-RX				1	ePllRxPhase160MHzC[1]	Sets EPLL-RX 160MHz phase C
EPLL-RX				2	ePllRxPhase160MHzC[2]	Sets EPLL-RX 160MHz phase C
EPLL-RX				3	ePllRxPhase160MHzC[3]	Sets EPLL-RX 160MHz phase C
EPLL-RX				4	ePllRxPhase160MHzC[4]	Sets EPLL-RX 160MHz phase C
EPLL-RX				5	ePllRxCapC[0]	Sets EPLL-RX Cap C in filter
EPLL-RX				6	ePllRxCapC[1]	Sets EPLL-RX Cap C in filter
				7		unused
EPLL-RX		310	ckCtr41	0	ePllRxIcpA[0]	Sets EPLL-RX charge pump current A
EPLL-RX				1	ePllRxIcpA[1]	Sets EPLL-RX charge pump current A
EPLL-RX				2	ePllRxIcpA[2]	Sets EPLL-RX charge pump current A
EPLL-RX				3	ePllRxIcpA[3]	Sets EPLL-RX charge pump current A
EPLL-RX				4	ePllRxResA[0]	Sets EPLL-RX Res A in filter
EPLL-RX				5	ePllRxResA[1]	Sets EPLL-RX Res A in filter
EPLL-RX				6	ePllRxResA[2]	Sets EPLL-RX Res A in filter
EPLL-RX				7	ePllRxResA[3]	Sets EPLL-RX Res A in filter
EPLL-RX		311	ckCtr42	0	ePllRxIcpB[0]	Sets EPLL-RX charge pump current B
EPLL-RX				1	ePllRxIcpB[1]	Sets EPLL-RX charge pump current B
EPLL-RX				2	ePllRxIcpB[2]	Sets EPLL-RX charge pump current B
EPLL-RX				3	ePllRxIcpB[3]	Sets EPLL-RX charge pump current B
EPLL-RX				4	ePllRxResB[0]	Sets EPLL-RX Res B in filter
EPLL-RX				5	ePllRxResB[1]	Sets EPLL-RX Res B in filter
EPLL-RX				6	ePllRxResB[2]	Sets EPLL-RX Res B in filter
EPLL-RX				7	ePllRxResB[3]	Sets EPLL-RX Res B in filter
EPLL-RX		312	ckCtr43	0	ePllRxIcpC[0]	Sets EPLL-RX charge pump current C

EPLL-RX				1	ePllRxIcpC[1]	Sets EPLL-RX charge pump current C
EPLL-RX				2	ePllRxIcpC[2]	Sets EPLL-RX charge pump current C
EPLL-RX				3	ePllRxIcpC[3]	Sets EPLL-RX charge pump current C
EPLL-RX				4	ePllRxResC[0]	Sets EPLL-RX Res C in filter
EPLL-RX				5	ePllRxResC[1]	Sets EPLL-RX Res C in filter
EPLL-RX				6	ePllRxResC[2]	Sets EPLL-RX Res C in filter
EPLL-RX				7	ePllRxResC[3]	Sets EPLL-RX Res C in filter
XPLL		313	ckCtr44	0	xPllFrequencyTrimA[0]	Sets XPLL frequency trim A
XPLL				1	xPllFrequencyTrimA[1]	Sets XPLL frequency trim A
XPLL				2	xPllFrequencyTrimA[2]	Sets XPLL frequency trim A
XPLL				3	xPllFrequencyTrimA[3]	Sets XPLL frequency trim A
XPLL				4	xPllFrequencyTrimA[4]	Sets XPLL frequency trim A
XPLL				5	xPllFrequencyTrimA[5]	Sets XPLL frequency trim A
XPLL				6	xPllFrequencyTrimA[6]	Sets XPLL frequency trim A
XPLL				7	xPllEnableA	Enables XPLL A
XPLL		314	ckCtr45	0	xPllFrequencyTrimB[0]	Sets XPLL frequency trim B
XPLL				1	xPllFrequencyTrimB[1]	Sets XPLL frequency trim B
XPLL				2	xPllFrequencyTrimB[2]	Sets XPLL frequency trim B
XPLL				3	xPllFrequencyTrimB[3]	Sets XPLL frequency trim B
XPLL				4	xPllFrequencyTrimB[4]	Sets XPLL frequency trim B
XPLL				5	xPllFrequencyTrimB[5]	Sets XPLL frequency trim B
XPLL				6	xPllFrequencyTrimB[6]	Sets XPLL frequency trim B
XPLL				7	xPllEnableB	Enables XPLL B
XPLL		315	ckCtr46	0	xPllFrequencyTrimC[0]	Sets XPLL frequency trim C
XPLL				1	xPllFrequencyTrimC[1]	Sets XPLL frequency trim C
XPLL				2	xPllFrequencyTrimC[2]	Sets XPLL frequency trim C
XPLL				3	xPllFrequencyTrimC[3]	Sets XPLL frequency trim C
XPLL				4	xPllFrequencyTrimC[4]	Sets XPLL frequency trim C
XPLL				5	xPllFrequencyTrimC[5]	Sets XPLL frequency trim C
XPLL				6	xPllFrequencyTrimC[6]	Sets XPLL frequency trim C
XPLL				7	xPllEnableC	Enables XPLL C
XPLL		316	ckCtr47	0	xPllGmSelectA[0]	Select XPLL Gm A
XPLL				1	xPllGmSelectA[1]	Select XPLL Gm A
XPLL				2	xPllGmSelectA[2]	Select XPLL Gm A
XPLL				3	xPllGmSelectA[3]	Select XPLL Gm A
XPLL				4	xPllGmSelectB[0]	Select XPLL Gm B
XPLL				5	xPllGmSelectB[1]	Select XPLL Gm B
XPLL				6	xPllGmSelectB[2]	Select XPLL Gm B
XPLL				7	xPllGmSelectB[3]	Select XPLL Gm B
XPLL		317	ckCtr48	0	xPllGmSelectC[0]	Select XPLL Gm C
XPLL				1	xPllGmSelectC[1]	Select XPLL Gm C
XPLL				2	xPllGmSelectC[2]	Select XPLL Gm C
XPLL				3	xPllGmSelectC[3]	Select XPLL Gm C
XPLL				4	xPllEnablePhaseDetectorA	Enable XPLL Phase Detector A
XPLL				5	xPllEnablePhaseDetectorB	Enable XPLL Phase Detector B
XPLL				6	xPllEnablePhaseDetectorC	Enable XPLL Phase Detector C
				7		unused
XPLL		318	ckCtr49	0	xPllControlOverrideA	Override XPLL control A
XPLL				1	xPllControlOverrideB	Override XPLL control B
XPLL				2	xPllControlOverrideC	Override XPLL control C
XPLL				3	xPllEnableAutoRestartA	Enable XPLL Auto restart A
XPLL				4	xPllEnableAutoRestartB	Enable XPLL Auto restart B
XPLL				5	xPllEnableAutoRestartC	Enable XPLL Auto restart C
				6		unused
				7		unused
Clock manager		319	ckCtr50	0	cmTestMuxSelect[0]	Selects which signal to send to testClockOut from second subset
Clock manager				1	cmTestMuxSelect[1]	Selects which signal to send to testClockOut from second subset
Clock manager				2	cmTestMuxSelect[2]	Selects which signal to send to testClockOut from second subset

Clock manager				3	cmTestMuxSelect[3]	Selects which signal to send to testClockOut from second subset
Clock manager				4	cmTestMuxSelect[4]	Selects which signal to send to testClockOut from second subset
				5		unused
				6		unused
				7		unused

Block	Manual reference	Memory address	Register Name	bits	Signal name	Function
Phase shifter		4	ttcCtr0[7:0]	0	FDL0[0]	Channel 0 fine delay
Phase shifter				1	FDL0[1]	
Phase shifter				2	FDL0[2]	
Phase shifter				3	FDL0[3]	
Phase shifter				4	FDL1[0]	Channel 1 fine delay
Phase shifter				5	FDL1[1]	
Phase shifter				6	FDL1[2]	
Phase shifter				7	FDL1[3]	
Phase shifter		5	ttcCtr1[7:0]	0	FDL2[0]	Channel 2 fine delay
Phase shifter				1	FDL2[1]	
Phase shifter				2	FDL2[2]	
Phase shifter				3	FDL2[3]	
Phase shifter				4	FDL3[0]	Channel 3 fine delay
Phase shifter				5	FDL3[1]	
Phase shifter				6	FDL3[2]	
Phase shifter				7	FDL3[3]	
Phase shifter		6	ttcCtr2[7:0]	0	FDL4[0]	Channel 4 fine delay
Phase shifter				1	FDL4[1]	
Phase shifter				2	FDL4[2]	
Phase shifter				3	FDL4[3]	
Phase shifter				4	FDL5[0]	Channel 5 fine delay
Phase shifter				5	FDL5[1]	
Phase shifter				6	FDL5[2]	
Phase shifter				7	FDL5[3]	
Phase shifter		7	ttcCtr3[7:0]	0	FDL6[0]	Channel 6 fine delay
Phase shifter				1	FDL6[1]	
Phase shifter				2	FDL6[2]	
Phase shifter				3	FDL6[3]	
Phase shifter				4	FDL7[0]	Channel 7 fine delay
Phase shifter				5	FDL7[1]	
Phase shifter				6	FDL7[2]	
Phase shifter				7	FDL7[3]	
Phase shifter		8	ttcCtr4[7:0]	0	CDL0[0]	Channel 0 coarse delay
Phase shifter				1	CDL0[1]	
Phase shifter				2	CDL0[2]	
Phase shifter				3	CDL0[3]	
Phase shifter				4	CDL0[4]	
Phase shifter				5	extS0[0]	
Phase shifter				6	extS0[1]	
Phase shifter				7	extEarly0	
Phase shifter		9	ttcCtr5[7:0]	0	CDL1[0]	Channel 1 coarse delay
Phase shifter				1	CDL1[1]	
Phase shifter				2	CDL1[2]	
Phase shifter				3	CDL1[3]	
Phase shifter				4	CDL1[4]	
Phase shifter				5	extS1[0]	
Phase shifter				6	extS1[1]	
Phase shifter				7	extEarly1	
Phase shifter		10	ttcCtr6[7:0]	0	CDL2[0]	Channel 2 coarse delay
Phase shifter				1	CDL2[1]	
Phase shifter				2	CDL2[2]	
Phase shifter				3	CDL2[3]	
Phase shifter				4	CDL2[4]	
Phase shifter				5	extS2[0]	
Phase shifter				6	extS2[1]	
Phase shifter				7	extEarly2	
Phase shifter		11	ttcCtr7[7:0]	0	CDL3[0]	Channel 3 coarse delay
Phase shifter				1	CDL3[1]	

Phase shifter				2	CDL3[2]	
Phase shifter				3	CDL3[3]	
Phase shifter				4	CDL3[4]	
Phase shifter				5	extS3[0]	
Phase shifter				6	extS3[1]	
Phase shifter				7	extEarly3	
Phase shifter		12	ttcCtr8[7:0]	0	CDL4[0]	Channel 4 coarse delay
Phase shifter				1	CDL4[1]	
Phase shifter				2	CDL4[2]	
Phase shifter				3	CDL4[3]	
Phase shifter				4	CDL4[4]	
Phase shifter				5	extS4[0]	
Phase shifter				6	extS4[1]	
Phase shifter				7	extEarly4	
Phase shifter		13	ttcCtr9[7:0]	0	CDL5[0]	Channel 5 coarse delay
Phase shifter				1	CDL5[1]	
Phase shifter				2	CDL5[2]	
Phase shifter				3	CDL5[3]	
Phase shifter				4	CDL5[4]	
Phase shifter				5	extS5[0]	
Phase shifter				6	extS5[1]	
Phase shifter				7	extEarly5	
Phase shifter		14	ttcCtr10[7:0]	0	CDL6[0]	Channel 6 coarse delay
Phase shifter				1	CDL6[1]	
Phase shifter				2	CDL6[2]	
Phase shifter				3	CDL6[3]	
Phase shifter				4	CDL6[4]	
Phase shifter				5	extS6[0]	
Phase shifter				6	extS6[1]	
Phase shifter				7	extEarly6	
Phase shifter		15	ttcCtr11[7:0]	0	CDL7[0]	Channel 7 coarse delay
Phase shifter				1	CDL7[1]	
Phase shifter				2	CDL7[2]	
Phase shifter				3	CDL7[3]	
Phase shifter				4	CDL7[4]	
Phase shifter				5	extS7[0]	
Phase shifter				6	extS7[1]	
Phase shifter				7	extEarly7	
Phase shifter		16	ttcCtr12[7:0]	0	iSel0[0]	
Phase shifter				1	iSel0[1]	
Phase shifter				2	iSel0[2]	
Phase shifter				3	iSel0[3]	
Phase shifter				4	FREQ0[0]	Channel 0 frequency
Phase shifter				5	FREQ0[1]	
Phase shifter				6	extLate0	
				7		
Phase shifter		17	ttcCtr13[7:0]	0	iSel1[0]	
Phase shifter				1	iSel1[1]	
Phase shifter				2	iSel1[2]	
Phase shifter				3	iSel1[3]	
Phase shifter				4	FREQ1[0]	Channel 1 frequency
Phase shifter				5	FREQ1[1]	
Phase shifter				6	extLate1	
				7		
Phase shifter		18	ttcCtr14[7:0]	0	iSel2[0]	
Phase shifter				1	iSel2[1]	
Phase shifter				2	iSel2[2]	
Phase shifter				3	iSel2[3]	
Phase shifter				4	FREQ2[0]	Channel 2 frequency
Phase shifter				5	FREQ2[1]	

Phase shifter				6	extLate2	
				7		
Phase shifter		19	ttcCtr15[7:0]	0	iSel3[0]	
Phase shifter				1	iSel3[1]	
Phase shifter				2	iSel3[2]	
Phase shifter				3	iSel3[3]	
Phase shifter				4	FREQ3[0]	Channel 3 frequency
Phase shifter				5	FREQ3[1]	
Phase shifter				6	extLate3	
				7		
Phase shifter		20	ttcCtr16[7:0]	0	iSel4[0]	
Phase shifter				1	iSel4[1]	
Phase shifter				2	iSel4[2]	
Phase shifter				3	iSel4[3]	
Phase shifter				4	FREQ4[0]	Channel 4 frequency
Phase shifter				5	FREQ4[1]	
Phase shifter				6	extLate4	
				7		
Phase shifter		21	ttcCtr17[7:0]	0	iSel5[0]	
Phase shifter				1	iSel5[1]	
Phase shifter				2	iSel5[2]	
Phase shifter				3	iSel5[3]	
Phase shifter				4	FREQ5[0]	Channel 5 frequency
Phase shifter				5	FREQ5[1]	
Phase shifter				6	extLate5	
				7		
Phase shifter		22	ttcCtr18[7:0]	0	iSel6[0]	
Phase shifter				1	iSel6[1]	
Phase shifter				2	iSel6[2]	
Phase shifter				3	iSel6[3]	
Phase shifter				4	FREQ6[0]	Channel 6 frequency
Phase shifter				5	FREQ6[1]	
Phase shifter				6	extLate6	
				7		
Phase shifter		23	ttcCtr19[7:0]	0	iSel7[0]	
Phase shifter				1	iSel7[1]	
Phase shifter				2	iSel7[2]	
Phase shifter				3	iSel7[3]	
Phase shifter				4	FREQ7[0]	Channel 7 frequency
Phase shifter				5	FREQ7[1]	
Phase shifter				6	extLate7	
				7		
Phase shifter		24	ttcCtr20[7:0]	0	resetBar0	Channel 0 reset
Phase shifter				1	resetBar1	Channel 1 reset
Phase shifter				2	resetBar2	Channel 2 reset
Phase shifter				3	resetBar3	Channel 3 reset
Phase shifter				4	resetBar4	Channel 4 reset
Phase shifter				5	resetBar5	Channel 5 reset
Phase shifter				6	resetBar6	Channel 6 reset
Phase shifter				7	resetBar7	Channel 7 reset
Phase shifter		25	ttcCtr21[7:0]	0	resetPLLBar	PLL reset
Phase shifter				1	enableTestBar	
Phase shifter				2		
Phase shifter				3		
Phase shifter				4		
Phase shifter				5		
				6		
				7		
Phase shifter		26	ttcCtr22[7:0]	0	PLLcp[0]	PLL charge pump current
Phase shifter				1	PLLcp[1]	

Phase shifter				2	PLLcp[2]	
Phase shifter				3	PLLcp[3]	
Phase shifter				4	PLLres[0]	PLL filter resistor
Phase shifter				5	PLLres[1]	
Phase shifter				6	PLLres[2]	
				7		
Phase shifter		269	ttcCtr23[7:0]	0	cset0[0]	Channel 0 driving strength
Phase shifter				1	cset0[1]	
Phase shifter				2	cset0[2]	
Phase shifter				3	cset0[3]	
Phase shifter				4	cset1[0]	Channel 1 driving strength
Phase shifter				5	cset1[1]	
Phase shifter				6	cset1[2]	
Phase shifter				7	cset1[3]	
Phase shifter		270	ttcCtr24[7:0]	0	cset2[0]	Channel 2 driving strength
Phase shifter				1	cset2[1]	
Phase shifter				2	cset2[2]	
Phase shifter				3	cset2[3]	
Phase shifter				4	cset3[0]	Channel 3 driving strength
Phase shifter				5	cset3[1]	
Phase shifter				6	cset3[2]	
Phase shifter				7	cset3[3]	
Phase shifter		271	ttcCtr25[7:0]	0	cset4[0]	Channel 4 driving strength
Phase shifter				1	cset4[1]	
Phase shifter				2	cset4[2]	
Phase shifter				3	cset4[3]	
Phase shifter				4	cset5[0]	Channel 5 driving strength
Phase shifter				5	cset5[1]	
Phase shifter				6	cset5[2]	
Phase shifter				7	cset5[3]	
Phase shifter		272	ttcCtr26[7:0]	0	cset6[0]	Channel 6 driving strength
Phase shifter				1	cset6[1]	
Phase shifter				2	cset6[2]	
Phase shifter				3	cset6[3]	
Phase shifter				4	cset7[0]	Channel 7 driving strength
Phase shifter				5	cset7[1]	
Phase shifter				6	cset7[2]	
Phase shifter				7	cset7[3]	

Block	Manual reference	Memory address	Register Name	bits	Signal name	Function
Serialiser		27	serCtr0[7:0]	0	txSelectI[0]	
Serialiser				1	txSelectI[1]	
Serialiser				2	txSelectI[2]	
Serialiser				3	txSelectI[3]	
Serialiser				4	txSelectR[0]	
Serialiser				5	txSelectR[1]	
Serialiser				6	txEnableTest	
				7		unused

Block	Manual reference	Memory address	Register Name	bits	Signal name	Function
TX logic		28	txCtr0	0	txTestMode[0]	Select TX data source
TX logic				1	txTestMode[1]	Select TX data source
				2		unused
				3		unused
				4		unused
				5		unused
TX logic				6	txDisableEncoderTMR	Disable TMR in FEC encoder
				7		unused
TX logic		29	txCtr1	0	txSwitchesControlA[0]	Select data path through TX logic A
TX logic				1	txSwitchesControlA[1]	Select data path through TX logic A
TX logic				2	txSwitchesControlA[2]	Select data path through TX logic A
TX logic				3	txSwitchesControlA[3]	Select data path through TX logic A
TX logic				4	txSwitchesControlA[4]	Select data path through TX logic A
TX logic				5	txSwitchesControlA[5]	Select data path through TX logic A
TX logic				6		Unused
TX logic				7		Unused
TX logic		30	txCtr2	0	txSwitchesControlB[0]	Select data path through TX logic B
TX logic				1	txSwitchesControlB[1]	Select data path through TX logic B
TX logic				2	txSwitchesControlB[2]	Select data path through TX logic B
TX logic				3	txSwitchesControlB[3]	Select data path through TX logic B
TX logic				4	txSwitchesControlB[4]	Select data path through TX logic B
TX logic				5	txSwitchesControlB[5]	Select data path through TX logic B
TX logic				6		Unused
TX logic				7		Unused
TX logic		31	txCtr3	0	txSwitchesControlC[0]	Select data path through TX logic C
TX logic				1	txSwitchesControlC[1]	Select data path through TX logic C
TX logic				2	txSwitchesControlC[2]	Select data path through TX logic C
TX logic				3	txSwitchesControlC[3]	Select data path through TX logic C
TX logic				4	txSwitchesControlC[4]	Select data path through TX logic C
TX logic				5	txSwitchesControlC[5]	Select data path through TX logic C
TX logic				6		Unused
TX logic				7		Unused
TX control		32	txCtr4	0	txPLLLockTIme[0]	Set wait time for serialiser PLL to lock
TX control				1	txPLLLockTIme[1]	Set wait time for serialiser PLL to lock
TX control				2	txPLLLockTIme[2]	Set wait time for serialiser PLL to lock
TX control				3	txPLLLockTIme[3]	Set wait time for serialiser PLL to lock
TX control				4	txForceLockState	Force serialiser control to locked state
TX control				5	txLossOfLockTime[0]	Set time limit on unlock
TX control				6	txLossOfLockTime[1]	Set time limit on unlock
TX control				7	txLossOfLockTime[2]	Set time limit on unlock
TX control		33	txCtr5	0	i2cTxResetA	Reset TX control and serialiser A
TX control				1	i2cTxResetB	Reset TX control and serialiser B
TX control				2	i2cTxResetC	Reset TX control and serialiser C
				3		Unused
				4		Unused
				5		Unused
TX control				6	txEnableSoftLossOfLock	Enable soft loss of lock
				7		unused

Block	Manual reference	Memory address	Register Name	bits	Signal name	Function
Deserialiser		34	desCtr0	0	rxSelectR[0]	
Deserialiser				1	rxSelectR[1]	
Deserialiser				2	rxSelectR[2]	
Deserialiser				3	rxSelectR[3]	
Deserialiser				4	rxDacGainA	
Deserialiser				5	rxDacGainB	
Deserialiser				6	rxDacGainC	
				7		unused

Block	Manual reference	Memory address	Register Name	bits	Signal name	Function
RX control		35	rxCtr0	0	i2cRxSelectI2[0]: freq detector	
RX control				1	i2cRxSelectI2[1]: freq detector	
RX control				2	i2cRxSelectI2[2]: freq detector	
RX control				3	i2cRxSelectI2[3]: freq detector	
RX control				4	i2cRxSelectI2[4]: phase detector	
RX control				5	i2cRxSelectI2[5]: phase detector	
RX control				6	i2cRxSelectI2[6]: phase detector	
RX control				7	i2cRxSelectI2[7]: phase detector	
RX control		36	rxCtr1	0	i2cRxControlOverrideA	
RX control				1	i2cRxControlOverrideB	
RX control				2	i2cRxControlOverrideC	
RX control				3	i2cRxSelectDataInPhaseA	
RX control				4	i2cRxSelectDataInPhaseB	
RX control				5	i2cRxSelectDataInPhaseC	
				6		Unused
				7		Unused
RX control		37	rxCtr2	0	rxValidHeaders[0]	
RX control				1	rxValidHeaders[1]	
RX control				2	rxValidHeaders[2]	
RX control				3	rxValidHeaders[3]	
RX control				4	rxValidHeaders[4]	
RX control				5	rxValidHeaders[5]	
RX control				6	rxValidHeaders[6]	
RX control				7	rxValidHeaders[7]	
RX control		38	rxCtr3	0	rxMaxInvalidHeaders[0]	
RX control				1	rxMaxInvalidHeaders[1]	
RX control				2	rxMaxInvalidHeaders[2]	
RX control				3	rxMaxInvalidHeaders[3]	
RX control				4	rxMaxInvalidHeaders[4]	
RX control				5	rxMaxInvalidHeaders[5]	
RX control				6	rxMaxInvalidHeaders[6]	
RX control				7	rxMaxInvalidHeaders[7]	
RX control		39	rxCtr4	0	rxMinValidHeaders[0]	
RX control				1	rxMinValidHeaders[1]	
RX control				2	rxMinValidHeaders[2]	
RX control				3	rxMinValidHeaders[3]	
RX control				4	rxMinValidHeaders[4]	
RX control				5	rxMinValidHeaders[5]	
RX control				6	rxMinValidHeaders[6]	
RX control				7	rxMinValidHeaders[7]	
RX control		40	rxCtr5	0	i2cRxSkipCycleA	
RX control				1	i2cRxSkipCycleB	
RX control				2	i2cRxSkipCycleC	
RX control				3	i2cRxSwapA	
RX control				4	i2cRxSwapB	
RX control				5	i2cRxSwapC	
				6		
				7		
RX control		41	rxCtr6	0	i2cRxControlResetA	FASM and LCSM reset
RX control				1	i2cRxControlResetB	FASM and LCSM reset
RX control				2	i2cRxControlResetC	FASM and LCSM reset
				3		
RX control				4	i2cRxSelectI1[0]	
RX control				5	i2cRxSelectI1[1]	
RX control				6	i2cRxSelectI1[2]	
RX control				7	i2cRxSelectI1[3]	
RX control		42	rxCtr7	0	i2cStartRaceA	
RX control				1	i2cStartRaceB	

RX control				2	i2cStartRaceC	
RX control				3	i2cRxDacEnableA	
RX control				4	i2cRxDacEnableB	
RX control				5	i2cRxDacEnableC	
				6		
				7		
RX control		43	rxCtr8	0	i2cRxDac[0]	
RX control				1	i2cRxDac[1]	
RX control				2	i2cRxDac[2]	
RX control				3	i2cRxDac[3]	
RX control				4	i2cRxDac[4]	
RX control				5	i2cRxDac[5]	
RX control				6	i2cRxDac[6]	
RX control				7	i2cRxDac[7]	
RX control		44	rxCtr9	0	i2cRxDac[8]	
RX control				1	i2cRxFilterBypassA	
RX control				2	i2cRxFilterBypassB	
RX control				3	i2cRxFilterBypassC	
RX control				4		Unused
RX control				5		Unused
RX control				6		Unused
				7		
RX control		45	rxCtr10	0	i2cRxFilterEnableA	
RX control				1	i2cRxFilterEnableB	
RX control				2	i2cRxFilterEnableC	
RX control				3	i2cRxForceVfEqVcA	
RX control				4	i2cRxForceVfEqVcB	
RX control				5	i2cRxForceVfEqVcC	
				6		
				7		
RX logic		46	rxCtr11	0	rxSwitchesControlA[0]	Select data path through RX logic A
RX logic				1	rxSwitchesControlA[1]	Select data path through RX logic A
RX logic				2	rxSwitchesControlA[2]	Select data path through RX logic A
RX logic				3	rxSwitchesControlA[3]	Select data path through RX logic A
RX logic				4	rxSwitchesControlA[4]	Select data path through RX logic A
RX logic				5	rxSwitchesControlA[5]	Select data path through RX logic A
RX logic				6		Unused
RX logic				7		Unused
RX logic		47	rxCtr12	0	rxSwitchesControlB[0]	Select data path through RX logic B
RX logic				1	rxSwitchesControlB[1]	Select data path through RX logic B
RX logic				2	rxSwitchesControlB[2]	Select data path through RX logic B
RX logic				3	rxSwitchesControlB[3]	Select data path through RX logic B
RX logic				4	rxSwitchesControlB[4]	Select data path through RX logic B
RX logic				5	rxSwitchesControlB[5]	Select data path through RX logic B
RX logic				6		Unused
RX logic				7		Unused
RX logic		48	rxCtr13	0	rxSwitchesControlC[0]	Select data path through RX logic C
RX logic				1	rxSwitchesControlC[1]	Select data path through RX logic C
RX logic				2	rxSwitchesControlC[2]	Select data path through RX logic C
RX logic				3	rxSwitchesControlC[3]	Select data path through RX logic C
RX logic				4	rxSwitchesControlC[4]	Select data path through RX logic C
RX logic				5	rxSwitchesControlC[5]	Select data path through RX logic C
RX logic				6		Unused
RX logic				7		Unused
		49	rxCtr14	0		Unused
RX logic				1	rxDisableDecoderTMR	Disable TMR in FEC decoder
RX logic				2	enableBERTA	Enable BERT in RX logic A
RX logic				3	enableBERTB	Enable BERT in RX logic B
RX logic				4	enableBERTC	Enable BERT in RX logic C
RX logic				5	rxTestModeA	Select RX data source A

RX logic				6	rxTestModeB	Select RX data source B
RX logic				7	rxTestModeC	Select RX data source C

Block	Manual reference	Memory address	Register Name	bits	Signal name	Function
Power-up		50	wdogCtr0	0	enable watchdog FSM A	enable watchdog A
Power-up				1	enable watchdog FSM B	enable watchdog B
Power-up				2	enable watchdog FSM C	enable watchdog C
RX logic				3	i2cResetRxA	reset RX logic A
RX logic				4	i2cResetRxB	reset RX logic B
RX logic				5	i2cResetRxC	reset RX logic C
				6		unused
				7		unused
Power-up		51	wdogCtr1	0	stateForced[0]	select state of power-up sequence
Power-up				1	stateForced[1]	select state of power-up sequence
Power-up				2	stateForced[2]	select state of power-up sequence
Power-up				3	stateForced[3]	select state of power-up sequence
Power-up				4	stateForced[4]	select state of power-up sequence
				5		unused
				6		unused
				7		unused
Phase shifter		52	wdogCtr2	0	PSpllEnableA	enable phase-shifter PLL
Phase shifter				1	PSpllEnableB	enable phase-shifter PLL
Phase shifter				2	PSpllEnableC	enable phase-shifter PLL
				3	timeOutEnableA	enable timeOuts
				4	timeOutEnableB	enable timeOuts
				5	timeOutEnableC	enable timeOuts
				6		unused
				7		unused
RX control		53	wdogCtr3	0	i2cRxResetA[0]	(Conditional) reset of the RX Reference PLL
RX control				1	i2cRxResetA[1]	(Conditional) reset of the CDR PLL
RX control				2	i2cRxResetB[0]	(Conditional) reset of the RX Reference PLL
RX control				3	i2cRxResetB[1]	(Conditional) reset of the CDR PLL
RX control				4	i2cRxResetC[0]	(Conditional) reset of the RX Reference PLL
RX control				5	i2cRxResetC[1]	(Conditional) reset of the CDR PLL
				6		unused
				7		unused
TX logic		54	wdogCtr4	0	i2cResetTxA	reset TX logic A
TX logic				1	i2cResetTxB	reset TX logic B
TX logic				2	i2cResetTxC	reset TX logic C
				3	i2cldResetA	Controls state of ldReset pin
				4	i2cldResetB	Controls state of ldReset pin
				5	i2cldResetC	Controls state of ldReset pin
				6		unused
				7		unused
Power-up		365	configDone	0	configDone[0]	re-starts power-up sequence if set to AA hex
Power-up				1	configDone[1]	re-starts power-up sequence if set to AA hex
Power-up				2	configDone[2]	re-starts power-up sequence if set to AA hex
Power-up				3	configDone[3]	re-starts power-up sequence if set to AA hex
Power-up				4	configDone[4]	re-starts power-up sequence if set to AA hex
Power-up				5	configDone[5]	re-starts power-up sequence if set to AA hex
Power-up				6	configDone[6]	re-starts power-up sequence if set to AA hex
Power-up				7	configDone[7]	re-starts power-up sequence if set to AA hex

Block	Manual reference	Memory address	Register Name	bits	Signal name	Function
GBLD access		55	gbld_w0	0	gbld_w0[0]	Value to write to GBLD
GBLD access				1	gbld_w0[1]	Value to write to GBLD
GBLD access				2	gbld_w0[2]	Value to write to GBLD
GBLD access				3	gbld_w0[3]	Value to write to GBLD
GBLD access				4	gbld_w0[4]	Value to write to GBLD
GBLD access				5	gbld_w0[5]	Value to write to GBLD
GBLD access				6	gbld_w0[6]	Value to write to GBLD
GBLD access				7	gbld_w0[7]	Value to write to GBLD
GBLD access		56	gbld_w1	0	gbld_w1[0]	Value to write to GBLD
GBLD access				1	gbld_w1[1]	Value to write to GBLD
GBLD access				2	gbld_w1[2]	Value to write to GBLD
GBLD access				3	gbld_w1[3]	Value to write to GBLD
GBLD access				4	gbld_w1[4]	Value to write to GBLD
GBLD access				5	gbld_w1[5]	Value to write to GBLD
GBLD access				6	gbld_w1[6]	Value to write to GBLD
GBLD access				7	gbld_w1[7]	Value to write to GBLD
GBLD access		57	gbld_w2	0	gbld_w2[0]	Value to write to GBLD
GBLD access				1	gbld_w2[1]	Value to write to GBLD
GBLD access				2	gbld_w2[2]	Value to write to GBLD
GBLD access				3	gbld_w2[3]	Value to write to GBLD
GBLD access				4	gbld_w2[4]	Value to write to GBLD
GBLD access				5	gbld_w2[5]	Value to write to GBLD
GBLD access				6	gbld_w2[6]	Value to write to GBLD
GBLD access				7	gbld_w2[7]	Value to write to GBLD
GBLD access		58	gbld_w3	0	gbld_w3[0]	Value to write to GBLD
GBLD access				1	gbld_w3[1]	Value to write to GBLD
GBLD access				2	gbld_w3[2]	Value to write to GBLD
GBLD access				3	gbld_w3[3]	Value to write to GBLD
GBLD access				4	gbld_w3[4]	Value to write to GBLD
GBLD access				5	gbld_w3[5]	Value to write to GBLD
GBLD access				6	gbld_w3[6]	Value to write to GBLD
GBLD access				7	gbld_w3[7]	Value to write to GBLD
GBLD access		59	gbld_w4	0	gbld_w4[0]	Value to write to GBLD
GBLD access				1	gbld_w4[1]	Value to write to GBLD
GBLD access				2	gbld_w4[2]	Value to write to GBLD
GBLD access				3	gbld_w4[3]	Value to write to GBLD
GBLD access				4	gbld_w4[4]	Value to write to GBLD
GBLD access				5	gbld_w4[5]	Value to write to GBLD
GBLD access				6	gbld_w4[6]	Value to write to GBLD
GBLD access				7	gbld_w4[7]	Value to write to GBLD
GBLD access		60	gbld_w5	0	gbld_w5[0]	Value to write to GBLD
GBLD access				1	gbld_w5[1]	Value to write to GBLD
GBLD access				2	gbld_w5[2]	Value to write to GBLD
GBLD access				3	gbld_w5[3]	Value to write to GBLD
GBLD access				4	gbld_w5[4]	Value to write to GBLD
GBLD access				5	gbld_w5[5]	Value to write to GBLD
GBLD access				6	gbld_w5[6]	Value to write to GBLD
GBLD access				7	gbld_w5[7]	Value to write to GBLD
GBLD access		61	gbld_w6	0	gbld_w6[0]	Value to write to GBLD
GBLD access				1	gbld_w6[1]	Value to write to GBLD
GBLD access				2	gbld_w6[2]	Value to write to GBLD
GBLD access				3	gbld_w6[3]	Value to write to GBLD
GBLD access				4	gbld_w6[4]	Value to write to GBLD
GBLD access				5	gbld_w6[5]	Value to write to GBLD
GBLD access				6	gbld_w6[6]	Value to write to GBLD
GBLD access				7	gbld_w6[7]	Value to write to GBLD
GBLD access		253	gbld_ID	0	gbld_ID[0]	i2c address of GBLD
GBLD access				1	gbld_ID[1]	i2c address of GBLD

GBLD access				2	gbld_ID[2]	i2c address of GBLD
GBLD access				3	gbld_ID[3]	i2c address of GBLD
GBLD access				4	gbld_ID[4]	i2c address of GBLD
GBLD access				5	gbld_ID[5]	i2c address of GBLD
GBLD access				6	gbld_ID[6]	i2c address of GBLD
				7		unused
GBLD access		388		0	gbld write	address this register to launch GBLD write
GBLD access				1	gbld write	address this register to launch GBLD write
GBLD access				2	gbld write	address this register to launch GBLD write
GBLD access				3	gbld write	address this register to launch GBLD write
GBLD access				4	gbld write	address this register to launch GBLD write
GBLD access				5	gbld write	address this register to launch GBLD write
GBLD access				6	gbld write	address this register to launch GBLD write
GBLD access				7	gbld write	address this register to launch GBLD write
GBLD access		389		0	gbld read	address this register to launch GBLD read
GBLD access				1	gbld read	address this register to launch GBLD read
GBLD access				2	gbld read	address this register to launch GBLD read
GBLD access				3	gbld read	address this register to launch GBLD read
GBLD access				4	gbld read	address this register to launch GBLD read
GBLD access				5	gbld read	address this register to launch GBLD read
GBLD access				6	gbld read	address this register to launch GBLD read
GBLD access				7	gbld read	address this register to launch GBLD read

Block	Manual reference	Memory address	Register Name	bits	Signal name	Function
EportRX		62	inEportCtr0	0	paModeA[0]	Phase-aligner track mode A
EportRX				1	paModeA[1]	Phase-aligner track mode A
EportRX				2	paModeB[0]	Phase-aligner track mode B
EportRX				3	paModeB[1]	Phase-aligner track mode B
EportRX				4	paModeC[0]	Phase-aligner track mode C
EportRX				5	paModeC[1]	Phase-aligner track mode C
				6		unused
				7		unused
EportRX		63	inEportCtr1	0	paDataRateAGroup0[0]	Set Phase-aligner speed A
EportRX				1	paDataRateAGroup0[1]	Set Phase-aligner speed A
EportRX				2	paDataRateBGroup0[0]	Set Phase-aligner speed B
EportRX				3	paDataRateBGroup0[1]	Set Phase-aligner speed B
EportRX				4	paDataRateCGroup0[0]	Set Phase-aligner speed C
EportRX				5	paDataRateCGroup0[1]	Set Phase-aligner speed C
				6		unused
				7		unused
EportRX		64	inEportCtr2	0	paDllConfigAGroup0[0]	Set Phase-aligner dll A
EportRX				1	paDllConfigAGroup0[1]	Set Phase-aligner dll A
EportRX				2	paDllConfigAGroup0[2]	Set Phase-aligner dll A
EportRX				3	paDllConfigAGroup0[3]	Set Phase-aligner dll A
EportRX				4	paDllConfigBGroup0[0]	Set Phase-aligner dll B
EportRX				5	paDllConfigBGroup0[1]	Set Phase-aligner dll B
EportRX				6	paDllConfigBGroup0[2]	Set Phase-aligner dll B
EportRX				7	paDllConfigBGroup0[3]	Set Phase-aligner dll B
EportRX		65	inEportCtr3	0	paDllConfigCGroup0[0]	Set Phase-aligner dll C
EportRX				1	paDllConfigCGroup0[1]	Set Phase-aligner dll C
EportRX				2	paDllConfigCGroup0[2]	Set Phase-aligner dll C
EportRX				3	paDllConfigCGroup0[3]	Set Phase-aligner dll C
EportRX				4	paDllResetAGroup0	Reset Phase-aligner dll A
EportRX				5	paDllResetBGroup0	Reset Phase-aligner dll B
EportRX				6	paDllResetCGroup0	Reset Phase-aligner dll C
				7		unused
EportRX		66	inEportCtr4	0	paPhaseSelectAGroup0[0]	Select sample phase channel 6 A
EportRX				1	paPhaseSelectAGroup0[1]	Select sample phase channel 6 A
EportRX				2	paPhaseSelectAGroup0[2]	Select sample phase channel 6 A
EportRX				3	paPhaseSelectAGroup0[3]	Select sample phase channel 6 A
EportRX				4	paPhaseSelectAGroup0[4]	Select sample phase channel 7 A
EportRX				5	paPhaseSelectAGroup0[5]	Select sample phase channel 7 A
EportRX				6	paPhaseSelectAGroup0[6]	Select sample phase channel 7 A
EportRX				7	paPhaseSelectAGroup0[7]	Select sample phase channel 7 A
EportRX		67	inEportCtr5	0	paPhaseSelectAGroup0[8]	Select sample phase channel 4 A
EportRX				1	paPhaseSelectAGroup0[9]	Select sample phase channel 4 A
EportRX				2	paPhaseSelectAGroup0[10]	Select sample phase channel 4 A
EportRX				3	paPhaseSelectAGroup0[11]	Select sample phase channel 4 A
EportRX				4	paPhaseSelectAGroup0[12]	Select sample phase channel 5 A
EportRX				5	paPhaseSelectAGroup0[13]	Select sample phase channel 5 A
EportRX				6	paPhaseSelectAGroup0[14]	Select sample phase channel 5 A
EportRX				7	paPhaseSelectAGroup0[15]	Select sample phase channel 5 A
EportRX		68	inEportCtr6	0	paPhaseSelectAGroup0[16]	Select sample phase channel 2 A
EportRX				1	paPhaseSelectAGroup0[17]	Select sample phase channel 2 A
EportRX				2	paPhaseSelectAGroup0[18]	Select sample phase channel 2 A
EportRX				3	paPhaseSelectAGroup0[19]	Select sample phase channel 2 A
EportRX				4	paPhaseSelectAGroup0[20]	Select sample phase channel 3 A
EportRX				5	paPhaseSelectAGroup0[21]	Select sample phase channel 3 A
EportRX				6	paPhaseSelectAGroup0[22]	Select sample phase channel 3 A
EportRX				7	paPhaseSelectAGroup0[23]	Select sample phase channel 3 A
EportRX		69	inEportCtr7	0	paPhaseSelectAGroup0[24]	Select sample phase channel 0 A
EportRX				1	paPhaseSelectAGroup0[25]	Select sample phase channel 0 A

EportRX				2	paPhaseSelectAGroup0[26]	Select sample phase channel 0 A
EportRX				3	paPhaseSelectAGroup0[27]	Select sample phase channel 0 A
EportRX				4	paPhaseSelectAGroup0[28]	Select sample phase channel 1 A
EportRX				5	paPhaseSelectAGroup0[29]	Select sample phase channel 1 A
EportRX				6	paPhaseSelectAGroup0[30]	Select sample phase channel 1 A
EportRX				7	paPhaseSelectAGroup0[31]	Select sample phase channel 1 A
EportRX		70	inEportCtr8	0	paPhaseSelectBGroup0[0]	Select sample phase channel 6 B
EportRX				1	paPhaseSelectBGroup0[1]	Select sample phase channel 6 B
EportRX				2	paPhaseSelectBGroup0[2]	Select sample phase channel 6 B
EportRX				3	paPhaseSelectBGroup0[3]	Select sample phase channel 6 B
EportRX				4	paPhaseSelectBGroup0[4]	Select sample phase channel 7 B
EportRX				5	paPhaseSelectBGroup0[5]	Select sample phase channel 7 B
EportRX				6	paPhaseSelectBGroup0[6]	Select sample phase channel 7 B
EportRX				7	paPhaseSelectBGroup0[7]	Select sample phase channel 7 B
EportRX		71	inEportCtr9	0	paPhaseSelectBGroup0[8]	Select sample phase channel 4 B
EportRX				1	paPhaseSelectBGroup0[9]	Select sample phase channel 4 B
EportRX				2	paPhaseSelectBGroup0[10]	Select sample phase channel 4 B
EportRX				3	paPhaseSelectBGroup0[11]	Select sample phase channel 4 B
EportRX				4	paPhaseSelectBGroup0[12]	Select sample phase channel 5 B
EportRX				5	paPhaseSelectBGroup0[13]	Select sample phase channel 5 B
EportRX				6	paPhaseSelectBGroup0[14]	Select sample phase channel 5 B
EportRX				7	paPhaseSelectBGroup0[15]	Select sample phase channel 5 B
EportRX		72	inEportCtr10	0	paPhaseSelectBGroup0[16]	Select sample phase channel 2 B
EportRX				1	paPhaseSelectBGroup0[17]	Select sample phase channel 2 B
EportRX				2	paPhaseSelectBGroup0[18]	Select sample phase channel 2 B
EportRX				3	paPhaseSelectBGroup0[19]	Select sample phase channel 2 B
EportRX				4	paPhaseSelectBGroup0[20]	Select sample phase channel 3 B
EportRX				5	paPhaseSelectBGroup0[21]	Select sample phase channel 3 B
EportRX				6	paPhaseSelectBGroup0[22]	Select sample phase channel 3 B
EportRX				7	paPhaseSelectBGroup0[23]	Select sample phase channel 3 B
EportRX		73	inEportCtr11	0	paPhaseSelectBGroup0[24]	Select sample phase channel 0 B
EportRX				1	paPhaseSelectBGroup0[25]	Select sample phase channel 0 B
EportRX				2	paPhaseSelectBGroup0[26]	Select sample phase channel 0 B
EportRX				3	paPhaseSelectBGroup0[27]	Select sample phase channel 0 B
EportRX				4	paPhaseSelectBGroup0[28]	Select sample phase channel 1 B
EportRX				5	paPhaseSelectBGroup0[29]	Select sample phase channel 1 B
EportRX				6	paPhaseSelectBGroup0[30]	Select sample phase channel 1 B
EportRX				7	paPhaseSelectBGroup0[31]	Select sample phase channel 1 B
EportRX		74	inEportCtr12	0	paPhaseSelectCGroup0[0]	Select sample phase channel 6 C
EportRX				1	paPhaseSelectCGroup0[1]	Select sample phase channel 6 C
EportRX				2	paPhaseSelectCGroup0[2]	Select sample phase channel 6 C
EportRX				3	paPhaseSelectCGroup0[3]	Select sample phase channel 6 C
EportRX				4	paPhaseSelectCGroup0[4]	Select sample phase channel 7 C
EportRX				5	paPhaseSelectCGroup0[5]	Select sample phase channel 7 C
EportRX				6	paPhaseSelectCGroup0[6]	Select sample phase channel 7 C
EportRX				7	paPhaseSelectCGroup0[7]	Select sample phase channel 7 C
EportRX		75	inEportCtr13	0	paPhaseSelectCGroup0[8]	Select sample phase channel 4 C
EportRX				1	paPhaseSelectCGroup0[9]	Select sample phase channel 4 C
EportRX				2	paPhaseSelectCGroup0[10]	Select sample phase channel 4 C
EportRX				3	paPhaseSelectCGroup0[11]	Select sample phase channel 4 C
EportRX				4	paPhaseSelectCGroup0[12]	Select sample phase channel 5 C
EportRX				5	paPhaseSelectCGroup0[13]	Select sample phase channel 5 C
EportRX				6	paPhaseSelectCGroup0[14]	Select sample phase channel 5 C
EportRX				7	paPhaseSelectCGroup0[15]	Select sample phase channel 5 C
EportRX		76	inEportCtr14	0	paPhaseSelectCGroup0[16]	Select sample phase channel 2 C
EportRX				1	paPhaseSelectCGroup0[17]	Select sample phase channel 2 C
EportRX				2	paPhaseSelectCGroup0[18]	Select sample phase channel 2 C
EportRX				3	paPhaseSelectCGroup0[19]	Select sample phase channel 2 C
EportRX				4	paPhaseSelectCGroup0[20]	Select sample phase channel 3 C
EportRX				5	paPhaseSelectCGroup0[21]	Select sample phase channel 3 C

EportRX				6	paPhaseSelectCGroup0[22]	Select sample phase channel 3 C
EportRX				7	paPhaseSelectCGroup0[23]	Select sample phase channel 3 C
EportRX		77	inEportCtr15	0	paPhaseSelectCGroup0[24]	Select sample phase channel 0 C
EportRX				1	paPhaseSelectCGroup0[25]	Select sample phase channel 0 C
EportRX				2	paPhaseSelectCGroup0[26]	Select sample phase channel 0 C
EportRX				3	paPhaseSelectCGroup0[27]	Select sample phase channel 0 C
EportRX				4	paPhaseSelectCGroup0[28]	Select sample phase channel 1 C
EportRX				5	paPhaseSelectCGroup0[29]	Select sample phase channel 1 C
EportRX				6	paPhaseSelectCGroup0[30]	Select sample phase channel 1 C
EportRX				7	paPhaseSelectCGroup0[31]	Select sample phase channel 1 C
EportRX		78	inEportCtr16	0	paTrainAGroup0[0]	Train channel 0 A
EportRX				1	paTrainAGroup0[1]	Train channel 1 A
EportRX				2	paTrainAGroup0[2]	Train channel 2 A
EportRX				3	paTrainAGroup0[3]	Train channel 3 A
EportRX				4	paTrainAGroup0[4]	Train channel 4 A
EportRX				5	paTrainAGroup0[5]	Train channel 5 A
EportRX				6	paTrainAGroup0[6]	Train channel 6 A
EportRX				7	paTrainAGroup0[7]	Train channel 7 A
EportRX		79	inEportCtr17	0	paTrainBGroup0[0]	Train channel 0 B
EportRX				1	paTrainBGroup0[1]	Train channel 1 B
EportRX				2	paTrainBGroup0[2]	Train channel 2 B
EportRX				3	paTrainBGroup0[3]	Train channel 3 B
EportRX				4	paTrainBGroup0[4]	Train channel 4 B
EportRX				5	paTrainBGroup0[5]	Train channel 5 B
EportRX				6	paTrainBGroup0[6]	Train channel 6 B
EportRX				7	paTrainBGroup0[7]	Train channel 7 B
EportRX		80	inEportCtr18	0	paTrainCGroup0[0]	Train channel 0 C
EportRX				1	paTrainCGroup0[1]	Train channel 1 C
EportRX				2	paTrainCGroup0[2]	Train channel 2 C
EportRX				3	paTrainCGroup0[3]	Train channel 3 C
EportRX				4	paTrainCGroup0[4]	Train channel 4 C
EportRX				5	paTrainCGroup0[5]	Train channel 5 C
EportRX				6	paTrainCGroup0[6]	Train channel 6 C
EportRX				7	paTrainCGroup0[7]	Train channel 7 C
EportRX		81	inEportCtr19	0	paEnableAGroup0[0]	Enable channel 0 A
EportRX				1	paEnableAGroup0[1]	Enable channel 1 A
EportRX				2	paEnableAGroup0[2]	Enable channel 2 A
EportRX				3	paEnableAGroup0[3]	Enable channel 3 A
EportRX				4	paEnableAGroup0[4]	Enable channel 4 A
EportRX				5	paEnableAGroup0[5]	Enable channel 5 A
EportRX				6	paEnableAGroup0[6]	Enable channel 6 A
EportRX				7	paEnableAGroup0[7]	Enable channel 7 A
EportRX		82	inEportCtr20	0	paEnableBGroup0[0]	Enable channel 0 B
EportRX				1	paEnableBGroup0[1]	Enable channel 1 B
EportRX				2	paEnableBGroup0[2]	Enable channel 2 B
EportRX				3	paEnableBGroup0[3]	Enable channel 3 B
EportRX				4	paEnableBGroup0[4]	Enable channel 4 B
EportRX				5	paEnableBGroup0[5]	Enable channel 5 B
EportRX				6	paEnableBGroup0[6]	Enable channel 6 B
EportRX				7	paEnableBGroup0[7]	Enable channel 7 B
EportRX		83	inEportCtr21	0	paEnableCGroup0[0]	Enable channel 0 C
EportRX				1	paEnableCGroup0[1]	Enable channel 1 C
EportRX				2	paEnableCGroup0[2]	Enable channel 2 C
EportRX				3	paEnableCGroup0[3]	Enable channel 3 C
EportRX				4	paEnableCGroup0[4]	Enable channel 4 C
EportRX				5	paEnableCGroup0[5]	Enable channel 5 C
EportRX				6	paEnableCGroup0[6]	Enable channel 6 C
EportRX				7	paEnableCGroup0[7]	Enable channel 7 C
EportRX		84	inEportCtr22	0	paResetAGroup0[0]	Reset channel 0 A
EportRX				1	paResetAGroup0[1]	Reset channel 1 A

EportRX				2	paResetAGroup0[2]	Reset channel 2 A
EportRX				3	paResetAGroup0[3]	Reset channel 3 A
EportRX				4	paResetAGroup0[4]	Reset channel 4 A
EportRX				5	paResetAGroup0[5]	Reset channel 5 A
EportRX				6	paResetAGroup0[6]	Reset channel 6 A
EportRX				7	paResetAGroup0[7]	Reset channel 7 A
EportRX		85	inEportCtr23	0	paResetBGroup0[0]	Reset channel 0 B
EportRX				1	paResetBGroup0[1]	Reset channel 1 B
EportRX				2	paResetBGroup0[2]	Reset channel 2 B
EportRX				3	paResetBGroup0[3]	Reset channel 3 B
EportRX				4	paResetBGroup0[4]	Reset channel 4 B
EportRX				5	paResetBGroup0[5]	Reset channel 5 B
EportRX				6	paResetBGroup0[6]	Reset channel 6 B
EportRX				7	paResetBGroup0[7]	Reset channel 7 B
EportRX		86	inEportCtr24	0	paResetCGroup0[0]	Reset channel 0 C
EportRX				1	paResetCGroup0[1]	Reset channel 1 C
EportRX				2	paResetCGroup0[2]	Reset channel 2 C
EportRX				3	paResetCGroup0[3]	Reset channel 3 C
EportRX				4	paResetCGroup0[4]	Reset channel 4 C
EportRX				5	paResetCGroup0[5]	Reset channel 5 C
EportRX				6	paResetCGroup0[6]	Reset channel 6 C
EportRX				7	paResetCGroup0[7]	Reset channel 7 C
EportRX		87	inEportCtr25	0	paDataRateAGroup1[0]	Set Phase-aligner speed A
EportRX				1	paDataRateAGroup1[1]	Set Phase-aligner speed A
EportRX				2	paDataRateBGroup1[0]	Set Phase-aligner speed B
EportRX				3	paDataRateBGroup1[1]	Set Phase-aligner speed B
EportRX				4	paDataRateCGroup1[0]	Set Phase-aligner speed C
EportRX				5	paDataRateCGroup1[1]	Set Phase-aligner speed C
				6		unused
				7		unused
EportRX		88	inEportCtr26	0	paDllConfigAGroup1[0]	Set Phase-aligner dll A
EportRX				1	paDllConfigAGroup1[1]	Set Phase-aligner dll A
EportRX				2	paDllConfigAGroup1[2]	Set Phase-aligner dll A
EportRX				3	paDllConfigAGroup1[3]	Set Phase-aligner dll A
EportRX				4	paDllConfigBGroup1[0]	Set Phase-aligner dll B
EportRX				5	paDllConfigBGroup1[1]	Set Phase-aligner dll B
EportRX				6	paDllConfigBGroup1[2]	Set Phase-aligner dll B
EportRX				7	paDllConfigBGroup1[3]	Set Phase-aligner dll B
EportRX		89	inEportCtr27	0	paDllConfigCGroup1[0]	Set Phase-aligner dll C
EportRX				1	paDllConfigCGroup1[1]	Set Phase-aligner dll C
EportRX				2	paDllConfigCGroup1[2]	Set Phase-aligner dll C
EportRX				3	paDllConfigCGroup1[3]	Set Phase-aligner dll C
EportRX				4	paDllResetAGroup1	Reset Phase-aligner dll A
EportRX				5	paDllResetBGroup1	Reset Phase-aligner dll B
EportRX				6	paDllResetCGroup1	Reset Phase-aligner dll C
				7		unused
EportRX		90	inEportCtr28	0	paPhaseSelectAGroup1[0]	Select sample phase channel 6 A
EportRX				1	paPhaseSelectAGroup1[1]	Select sample phase channel 6 A
EportRX				2	paPhaseSelectAGroup1[2]	Select sample phase channel 6 A
EportRX				3	paPhaseSelectAGroup1[3]	Select sample phase channel 6 A
EportRX				4	paPhaseSelectAGroup1[4]	Select sample phase channel 7 A
EportRX				5	paPhaseSelectAGroup1[5]	Select sample phase channel 7 A
EportRX				6	paPhaseSelectAGroup1[6]	Select sample phase channel 7 A
EportRX				7	paPhaseSelectAGroup1[7]	Select sample phase channel 7 A
EportRX		91	inEportCtr29	0	paPhaseSelectAGroup1[8]	Select sample phase channel 4 A
EportRX				1	paPhaseSelectAGroup1[9]	Select sample phase channel 4 A
EportRX				2	paPhaseSelectAGroup1[10]	Select sample phase channel 4 A
EportRX				3	paPhaseSelectAGroup1[11]	Select sample phase channel 4 A
EportRX				4	paPhaseSelectAGroup1[12]	Select sample phase channel 5 A
EportRX				5	paPhaseSelectAGroup1[13]	Select sample phase channel 5 A

EportRX				6	paPhaseSelectAGroup1[14]	Select sample phase channel 5 A
EportRX				7	paPhaseSelectAGroup1[15]	Select sample phase channel 5 A
EportRX		92	inEportCtr30	0	paPhaseSelectAGroup1[16]	Select sample phase channel 2 A
EportRX				1	paPhaseSelectAGroup1[17]	Select sample phase channel 2 A
EportRX				2	paPhaseSelectAGroup1[18]	Select sample phase channel 2 A
EportRX				3	paPhaseSelectAGroup1[19]	Select sample phase channel 2 A
EportRX				4	paPhaseSelectAGroup1[20]	Select sample phase channel 3 A
EportRX				5	paPhaseSelectAGroup1[21]	Select sample phase channel 3 A
EportRX				6	paPhaseSelectAGroup1[22]	Select sample phase channel 3 A
EportRX				7	paPhaseSelectAGroup1[23]	Select sample phase channel 3 A
EportRX		93	inEportCtr31	0	paPhaseSelectAGroup1[24]	Select sample phase channel 0 A
EportRX				1	paPhaseSelectAGroup1[25]	Select sample phase channel 0 A
EportRX				2	paPhaseSelectAGroup1[26]	Select sample phase channel 0 A
EportRX				3	paPhaseSelectAGroup1[27]	Select sample phase channel 0 A
EportRX				4	paPhaseSelectAGroup1[28]	Select sample phase channel 1 A
EportRX				5	paPhaseSelectAGroup1[29]	Select sample phase channel 1 A
EportRX				6	paPhaseSelectAGroup1[30]	Select sample phase channel 1 A
EportRX				7	paPhaseSelectAGroup1[31]	Select sample phase channel 1 A
EportRX		94	inEportCtr32	0	paPhaseSelectBGroup1[0]	Select sample phase channel 6 B
EportRX				1	paPhaseSelectBGroup1[1]	Select sample phase channel 6 B
EportRX				2	paPhaseSelectBGroup1[2]	Select sample phase channel 6 B
EportRX				3	paPhaseSelectBGroup1[3]	Select sample phase channel 6 B
EportRX				4	paPhaseSelectBGroup1[4]	Select sample phase channel 7 B
EportRX				5	paPhaseSelectBGroup1[5]	Select sample phase channel 7 B
EportRX				6	paPhaseSelectBGroup1[6]	Select sample phase channel 7 B
EportRX				7	paPhaseSelectBGroup1[7]	Select sample phase channel 7 B
EportRX		95	inEportCtr33	0	paPhaseSelectBGroup1[8]	Select sample phase channel 4 B
EportRX				1	paPhaseSelectBGroup1[9]	Select sample phase channel 4 B
EportRX				2	paPhaseSelectBGroup1[10]	Select sample phase channel 4 B
EportRX				3	paPhaseSelectBGroup1[11]	Select sample phase channel 4 B
EportRX				4	paPhaseSelectBGroup1[12]	Select sample phase channel 5 B
EportRX				5	paPhaseSelectBGroup1[13]	Select sample phase channel 5 B
EportRX				6	paPhaseSelectBGroup1[14]	Select sample phase channel 5 B
EportRX				7	paPhaseSelectBGroup1[15]	Select sample phase channel 5 B
EportRX		96	inEportCtr34	0	paPhaseSelectBGroup1[16]	Select sample phase channel 2 B
EportRX				1	paPhaseSelectBGroup1[17]	Select sample phase channel 2 B
EportRX				2	paPhaseSelectBGroup1[18]	Select sample phase channel 2 B
EportRX				3	paPhaseSelectBGroup1[19]	Select sample phase channel 2 B
EportRX				4	paPhaseSelectBGroup1[20]	Select sample phase channel 3 B
EportRX				5	paPhaseSelectBGroup1[21]	Select sample phase channel 3 B
EportRX				6	paPhaseSelectBGroup1[22]	Select sample phase channel 3 B
EportRX				7	paPhaseSelectBGroup1[23]	Select sample phase channel 3 B
EportRX		97	inEportCtr35	0	paPhaseSelectBGroup1[24]	Select sample phase channel 0 B
EportRX				1	paPhaseSelectBGroup1[25]	Select sample phase channel 0 B
EportRX				2	paPhaseSelectBGroup1[26]	Select sample phase channel 0 B
EportRX				3	paPhaseSelectBGroup1[27]	Select sample phase channel 0 B
EportRX				4	paPhaseSelectBGroup1[28]	Select sample phase channel 1 B
EportRX				5	paPhaseSelectBGroup1[29]	Select sample phase channel 1 B
EportRX				6	paPhaseSelectBGroup1[30]	Select sample phase channel 1 B
EportRX				7	paPhaseSelectBGroup1[31]	Select sample phase channel 1 B
EportRX		98	inEportCtr36	0	paPhaseSelectCGroup1[0]	Select sample phase channel 6 C
EportRX				1	paPhaseSelectCGroup1[1]	Select sample phase channel 6 C
EportRX				2	paPhaseSelectCGroup1[2]	Select sample phase channel 6 C
EportRX				3	paPhaseSelectCGroup1[3]	Select sample phase channel 6 C
EportRX				4	paPhaseSelectCGroup1[4]	Select sample phase channel 7 C
EportRX				5	paPhaseSelectCGroup1[5]	Select sample phase channel 7 C
EportRX				6	paPhaseSelectCGroup1[6]	Select sample phase channel 7 C
EportRX				7	paPhaseSelectCGroup1[7]	Select sample phase channel 7 C
EportRX		99	inEportCtr37	0	paPhaseSelectCGroup1[8]	Select sample phase channel 4 C
EportRX				1	paPhaseSelectCGroup1[9]	Select sample phase channel 4 C

EportRX				2	paPhaseSelectCGroup1[10]	Select sample phase channel 4 C
EportRX				3	paPhaseSelectCGroup1[11]	Select sample phase channel 4 C
EportRX				4	paPhaseSelectCGroup1[12]	Select sample phase channel 5 C
EportRX				5	paPhaseSelectCGroup1[13]	Select sample phase channel 5 C
EportRX				6	paPhaseSelectCGroup1[14]	Select sample phase channel 5 C
EportRX				7	paPhaseSelectCGroup1[15]	Select sample phase channel 5 C
EportRX		100	inEportCtr38	0	paPhaseSelectCGroup1[16]	Select sample phase channel 2 C
EportRX				1	paPhaseSelectCGroup1[17]	Select sample phase channel 2 C
EportRX				2	paPhaseSelectCGroup1[18]	Select sample phase channel 2 C
EportRX				3	paPhaseSelectCGroup1[19]	Select sample phase channel 2 C
EportRX				4	paPhaseSelectCGroup1[20]	Select sample phase channel 3 C
EportRX				5	paPhaseSelectCGroup1[21]	Select sample phase channel 3 C
EportRX				6	paPhaseSelectCGroup1[22]	Select sample phase channel 3 C
EportRX				7	paPhaseSelectCGroup1[23]	Select sample phase channel 3 C
EportRX		101	inEportCtr39	0	paPhaseSelectCGroup1[24]	Select sample phase channel 0 C
EportRX				1	paPhaseSelectCGroup1[25]	Select sample phase channel 0 C
EportRX				2	paPhaseSelectCGroup1[26]	Select sample phase channel 0 C
EportRX				3	paPhaseSelectCGroup1[27]	Select sample phase channel 0 C
EportRX				4	paPhaseSelectCGroup1[28]	Select sample phase channel 1 C
EportRX				5	paPhaseSelectCGroup1[29]	Select sample phase channel 1 C
EportRX				6	paPhaseSelectCGroup1[30]	Select sample phase channel 1 C
EportRX				7	paPhaseSelectCGroup1[31]	Select sample phase channel 1 C
EportRX		102	inEportCtr40	0	paTrainAGroup1[0]	Train channel 0 A
EportRX				1	paTrainAGroup1[1]	Train channel 1 A
EportRX				2	paTrainAGroup1[2]	Train channel 2 A
EportRX				3	paTrainAGroup1[3]	Train channel 3 A
EportRX				4	paTrainAGroup1[4]	Train channel 4 A
EportRX				5	paTrainAGroup1[5]	Train channel 5 A
EportRX				6	paTrainAGroup1[6]	Train channel 6 A
EportRX				7	paTrainAGroup1[7]	Train channel 7 A
EportRX		103	inEportCtr41	0	paTrainBGroup1[0]	Train channel 0 B
EportRX				1	paTrainBGroup1[1]	Train channel 1 B
EportRX				2	paTrainBGroup1[2]	Train channel 2 B
EportRX				3	paTrainBGroup1[3]	Train channel 3 B
EportRX				4	paTrainBGroup1[4]	Train channel 4 B
EportRX				5	paTrainBGroup1[5]	Train channel 5 B
EportRX				6	paTrainBGroup1[6]	Train channel 6 B
EportRX				7	paTrainBGroup1[7]	Train channel 7 B
EportRX		104	inEportCtr42	0	paTrainCGroup1[0]	Train channel 0 C
EportRX				1	paTrainCGroup1[1]	Train channel 1 C
EportRX				2	paTrainCGroup1[2]	Train channel 2 C
EportRX				3	paTrainCGroup1[3]	Train channel 3 C
EportRX				4	paTrainCGroup1[4]	Train channel 4 C
EportRX				5	paTrainCGroup1[5]	Train channel 5 C
EportRX				6	paTrainCGroup1[6]	Train channel 6 C
EportRX				7	paTrainCGroup1[7]	Train channel 7 C
EportRX		105	inEportCtr43	0	paEnableAGroup1[0]	Enable channel 0 A
EportRX				1	paEnableAGroup1[1]	Enable channel 1 A
EportRX				2	paEnableAGroup1[2]	Enable channel 2 A
EportRX				3	paEnableAGroup1[3]	Enable channel 3 A
EportRX				4	paEnableAGroup1[4]	Enable channel 4 A
EportRX				5	paEnableAGroup1[5]	Enable channel 5 A
EportRX				6	paEnableAGroup1[6]	Enable channel 6 A
EportRX				7	paEnableAGroup1[7]	Enable channel 7 A
EportRX		106	inEportCtr44	0	paEnableBGroup1[0]	Enable channel 0 B
EportRX				1	paEnableBGroup1[1]	Enable channel 1 B
EportRX				2	paEnableBGroup1[2]	Enable channel 2 B
EportRX				3	paEnableBGroup1[3]	Enable channel 3 B
EportRX				4	paEnableBGroup1[4]	Enable channel 4 B
EportRX				5	paEnableBGroup1[5]	Enable channel 5 B

EportRX				6	paEnableBGroup1[6]	Enable channel 6 B
EportRX				7	paEnableBGroup1[7]	Enable channel 7 B
EportRX		107	inEportCtr45	0	paEnableCGroup1[0]	Enable channel 0 C
EportRX				1	paEnableCGroup1[1]	Enable channel 1 C
EportRX				2	paEnableCGroup1[2]	Enable channel 2 C
EportRX				3	paEnableCGroup1[3]	Enable channel 3 C
EportRX				4	paEnableCGroup1[4]	Enable channel 4 C
EportRX				5	paEnableCGroup1[5]	Enable channel 5 C
EportRX				6	paEnableCGroup1[6]	Enable channel 6 C
EportRX				7	paEnableCGroup1[7]	Enable channel 7 C
EportRX		108	inEportCtr46	0	paResetAGroup1[0]	Reset channel 0 A
EportRX				1	paResetAGroup1[1]	Reset channel 1 A
EportRX				2	paResetAGroup1[2]	Reset channel 2 A
EportRX				3	paResetAGroup1[3]	Reset channel 3 A
EportRX				4	paResetAGroup1[4]	Reset channel 4 A
EportRX				5	paResetAGroup1[5]	Reset channel 5 A
EportRX				6	paResetAGroup1[6]	Reset channel 6 A
EportRX				7	paResetAGroup1[7]	Reset channel 7 A
EportRX		109	inEportCtr47	0	paResetBGroup1[0]	Reset channel 0 B
EportRX				1	paResetBGroup1[1]	Reset channel 1 B
EportRX				2	paResetBGroup1[2]	Reset channel 2 B
EportRX				3	paResetBGroup1[3]	Reset channel 3 B
EportRX				4	paResetBGroup1[4]	Reset channel 4 B
EportRX				5	paResetBGroup1[5]	Reset channel 5 B
EportRX				6	paResetBGroup1[6]	Reset channel 6 B
EportRX				7	paResetBGroup1[7]	Reset channel 7 B
EportRX		110	inEportCtr48	0	paResetCGroup1[0]	Reset channel 0 C
EportRX				1	paResetCGroup1[1]	Reset channel 1 C
EportRX				2	paResetCGroup1[2]	Reset channel 2 C
EportRX				3	paResetCGroup1[3]	Reset channel 3 C
EportRX				4	paResetCGroup1[4]	Reset channel 4 C
EportRX				5	paResetCGroup1[5]	Reset channel 5 C
EportRX				6	paResetCGroup1[6]	Reset channel 6 C
EportRX				7	paResetCGroup1[7]	Reset channel 7 C
EportRX		111	inEportCtr49	0	paDataRateAGroup2[0]	Set Phase-aligner speed A
EportRX				1	paDataRateAGroup2[1]	Set Phase-aligner speed A
EportRX				2	paDataRateBGroup2[0]	Set Phase-aligner speed B
EportRX				3	paDataRateBGroup2[1]	Set Phase-aligner speed B
EportRX				4	paDataRateCGroup2[0]	Set Phase-aligner speed C
EportRX				5	paDataRateCGroup2[1]	Set Phase-aligner speed C
				6		unused
				7		unused
EportRX		112	inEportCtr50	0	paDllConfigAGroup2[0]	Set Phase-aligner dll A
EportRX				1	paDllConfigAGroup2[1]	Set Phase-aligner dll A
EportRX				2	paDllConfigAGroup2[2]	Set Phase-aligner dll A
EportRX				3	paDllConfigAGroup2[3]	Set Phase-aligner dll A
EportRX				4	paDllConfigBGroup2[0]	Set Phase-aligner dll B
EportRX				5	paDllConfigBGroup2[1]	Set Phase-aligner dll B
EportRX				6	paDllConfigBGroup2[2]	Set Phase-aligner dll B
EportRX				7	paDllConfigBGroup2[3]	Set Phase-aligner dll B
EportRX		113	inEportCtr51	0	paDllConfigCGroup2[0]	Set Phase-aligner dll C
EportRX				1	paDllConfigCGroup2[1]	Set Phase-aligner dll C
EportRX				2	paDllConfigCGroup2[2]	Set Phase-aligner dll C
EportRX				3	paDllConfigCGroup2[3]	Set Phase-aligner dll C
EportRX				4	paDllResetAGroup2	Reset Phase-aligner dll A
EportRX				5	paDllResetBGroup2	Reset Phase-aligner dll B
EportRX				6	paDllResetCGroup2	Reset Phase-aligner dll C
				7		unused
EportRX		114	inEportCtr52	0	paPhaseSelectAGroup2[0]	Select sample phase channel 6 A
EportRX				1	paPhaseSelectAGroup2[1]	Select sample phase channel 6 A

EportRX				2	paPhaseSelectAGroup2[2]	Select sample phase channel 6 A
EportRX				3	paPhaseSelectAGroup2[3]	Select sample phase channel 6 A
EportRX				4	paPhaseSelectAGroup2[4]	Select sample phase channel 7 A
EportRX				5	paPhaseSelectAGroup2[5]	Select sample phase channel 7 A
EportRX				6	paPhaseSelectAGroup2[6]	Select sample phase channel 7 A
EportRX				7	paPhaseSelectAGroup2[7]	Select sample phase channel 7 A
EportRX		115	inEportCtr53	0	paPhaseSelectAGroup2[8]	Select sample phase channel 4 A
EportRX				1	paPhaseSelectAGroup2[9]	Select sample phase channel 4 A
EportRX				2	paPhaseSelectAGroup2[10]	Select sample phase channel 4 A
EportRX				3	paPhaseSelectAGroup2[11]	Select sample phase channel 4 A
EportRX				4	paPhaseSelectAGroup2[12]	Select sample phase channel 5 A
EportRX				5	paPhaseSelectAGroup2[13]	Select sample phase channel 5 A
EportRX				6	paPhaseSelectAGroup2[14]	Select sample phase channel 5 A
EportRX				7	paPhaseSelectAGroup2[15]	Select sample phase channel 5 A
EportRX		116	inEportCtr54	0	paPhaseSelectAGroup2[16]	Select sample phase channel 2 A
EportRX				1	paPhaseSelectAGroup2[17]	Select sample phase channel 2 A
EportRX				2	paPhaseSelectAGroup2[18]	Select sample phase channel 2 A
EportRX				3	paPhaseSelectAGroup2[19]	Select sample phase channel 2 A
EportRX				4	paPhaseSelectAGroup2[20]	Select sample phase channel 3 A
EportRX				5	paPhaseSelectAGroup2[21]	Select sample phase channel 3 A
EportRX				6	paPhaseSelectAGroup2[22]	Select sample phase channel 3 A
EportRX				7	paPhaseSelectAGroup2[23]	Select sample phase channel 3 A
EportRX		117	inEportCtr55	0	paPhaseSelectAGroup2[24]	Select sample phase channel 0 A
EportRX				1	paPhaseSelectAGroup2[25]	Select sample phase channel 0 A
EportRX				2	paPhaseSelectAGroup2[26]	Select sample phase channel 0 A
EportRX				3	paPhaseSelectAGroup2[27]	Select sample phase channel 0 A
EportRX				4	paPhaseSelectAGroup2[28]	Select sample phase channel 1 A
EportRX				5	paPhaseSelectAGroup2[29]	Select sample phase channel 1 A
EportRX				6	paPhaseSelectAGroup2[30]	Select sample phase channel 1 A
EportRX				7	paPhaseSelectAGroup2[31]	Select sample phase channel 1 A
EportRX		118	inEportCtr56	0	paPhaseSelectBGroup2[0]	Select sample phase channel 6 B
EportRX				1	paPhaseSelectBGroup2[1]	Select sample phase channel 6 B
EportRX				2	paPhaseSelectBGroup2[2]	Select sample phase channel 6 B
EportRX				3	paPhaseSelectBGroup2[3]	Select sample phase channel 6 B
EportRX				4	paPhaseSelectBGroup2[4]	Select sample phase channel 7 B
EportRX				5	paPhaseSelectBGroup2[5]	Select sample phase channel 7 B
EportRX				6	paPhaseSelectBGroup2[6]	Select sample phase channel 7 B
EportRX				7	paPhaseSelectBGroup2[7]	Select sample phase channel 7 B
EportRX		119	inEportCtr57	0	paPhaseSelectBGroup2[8]	Select sample phase channel 4 B
EportRX				1	paPhaseSelectBGroup2[9]	Select sample phase channel 4 B
EportRX				2	paPhaseSelectBGroup2[10]	Select sample phase channel 4 B
EportRX				3	paPhaseSelectBGroup2[11]	Select sample phase channel 4 B
EportRX				4	paPhaseSelectBGroup2[12]	Select sample phase channel 5 B
EportRX				5	paPhaseSelectBGroup2[13]	Select sample phase channel 5 B
EportRX				6	paPhaseSelectBGroup2[14]	Select sample phase channel 5 B
EportRX				7	paPhaseSelectBGroup2[15]	Select sample phase channel 5 B
EportRX		120	inEportCtr58	0	paPhaseSelectBGroup2[16]	Select sample phase channel 2 B
EportRX				1	paPhaseSelectBGroup2[17]	Select sample phase channel 2 B
EportRX				2	paPhaseSelectBGroup2[18]	Select sample phase channel 2 B
EportRX				3	paPhaseSelectBGroup2[19]	Select sample phase channel 2 B
EportRX				4	paPhaseSelectBGroup2[20]	Select sample phase channel 3 B
EportRX				5	paPhaseSelectBGroup2[21]	Select sample phase channel 3 B
EportRX				6	paPhaseSelectBGroup2[22]	Select sample phase channel 3 B
EportRX				7	paPhaseSelectBGroup2[23]	Select sample phase channel 3 B
EportRX		121	inEportCtr59	0	paPhaseSelectBGroup2[24]	Select sample phase channel 0 B
EportRX				1	paPhaseSelectBGroup2[25]	Select sample phase channel 0 B
EportRX				2	paPhaseSelectBGroup2[26]	Select sample phase channel 0 B
EportRX				3	paPhaseSelectBGroup2[27]	Select sample phase channel 0 B
EportRX				4	paPhaseSelectBGroup2[28]	Select sample phase channel 1 B
EportRX				5	paPhaseSelectBGroup2[29]	Select sample phase channel 1 B

EportRX				6	paPhaseSelectBGroup2[30]	Select sample phase channel 1 B
EportRX				7	paPhaseSelectBGroup2[31]	Select sample phase channel 1 B
EportRX		122	inEportCtr60	0	paPhaseSelectCGroup2[0]	Select sample phase channel 6 C
EportRX				1	paPhaseSelectCGroup2[1]	Select sample phase channel 6 C
EportRX				2	paPhaseSelectCGroup2[2]	Select sample phase channel 6 C
EportRX				3	paPhaseSelectCGroup2[3]	Select sample phase channel 6 C
EportRX				4	paPhaseSelectCGroup2[4]	Select sample phase channel 7 C
EportRX				5	paPhaseSelectCGroup2[5]	Select sample phase channel 7 C
EportRX				6	paPhaseSelectCGroup2[6]	Select sample phase channel 7 C
EportRX				7	paPhaseSelectCGroup2[7]	Select sample phase channel 7 C
EportRX		123	inEportCtr61	0	paPhaseSelectCGroup2[8]	Select sample phase channel 4 C
EportRX				1	paPhaseSelectCGroup2[9]	Select sample phase channel 4 C
EportRX				2	paPhaseSelectCGroup2[10]	Select sample phase channel 4 C
EportRX				3	paPhaseSelectCGroup2[11]	Select sample phase channel 4 C
EportRX				4	paPhaseSelectCGroup2[12]	Select sample phase channel 5 C
EportRX				5	paPhaseSelectCGroup2[13]	Select sample phase channel 5 C
EportRX				6	paPhaseSelectCGroup2[14]	Select sample phase channel 5 C
EportRX				7	paPhaseSelectCGroup2[15]	Select sample phase channel 5 C
EportRX		124	inEportCtr62	0	paPhaseSelectCGroup2[16]	Select sample phase channel 2 C
EportRX				1	paPhaseSelectCGroup2[17]	Select sample phase channel 2 C
EportRX				2	paPhaseSelectCGroup2[18]	Select sample phase channel 2 C
EportRX				3	paPhaseSelectCGroup2[19]	Select sample phase channel 2 C
EportRX				4	paPhaseSelectCGroup2[20]	Select sample phase channel 3 C
EportRX				5	paPhaseSelectCGroup2[21]	Select sample phase channel 3 C
EportRX				6	paPhaseSelectCGroup2[22]	Select sample phase channel 3 C
EportRX				7	paPhaseSelectCGroup2[23]	Select sample phase channel 3 C
EportRX		125	inEportCtr63	0	paPhaseSelectCGroup2[24]	Select sample phase channel 0 C
EportRX				1	paPhaseSelectCGroup2[25]	Select sample phase channel 0 C
EportRX				2	paPhaseSelectCGroup2[26]	Select sample phase channel 0 C
EportRX				3	paPhaseSelectCGroup2[27]	Select sample phase channel 0 C
EportRX				4	paPhaseSelectCGroup2[28]	Select sample phase channel 1 C
EportRX				5	paPhaseSelectCGroup2[29]	Select sample phase channel 1 C
EportRX				6	paPhaseSelectCGroup2[30]	Select sample phase channel 1 C
EportRX				7	paPhaseSelectCGroup2[31]	Select sample phase channel 1 C
EportRX		126	inEportCtr64	0	paTrainAGroup2[0]	Train channel 0 A
EportRX				1	paTrainAGroup2[1]	Train channel 1 A
EportRX				2	paTrainAGroup2[2]	Train channel 2 A
EportRX				3	paTrainAGroup2[3]	Train channel 3 A
EportRX				4	paTrainAGroup2[4]	Train channel 4 A
EportRX				5	paTrainAGroup2[5]	Train channel 5 A
EportRX				6	paTrainAGroup2[6]	Train channel 6 A
EportRX				7	paTrainAGroup2[7]	Train channel 7 A
EportRX		127	inEportCtr65	0	paTrainBGroup2[0]	Train channel 0 B
EportRX				1	paTrainBGroup2[1]	Train channel 1 B
EportRX				2	paTrainBGroup2[2]	Train channel 2 B
EportRX				3	paTrainBGroup2[3]	Train channel 3 B
EportRX				4	paTrainBGroup2[4]	Train channel 4 B
EportRX				5	paTrainBGroup2[5]	Train channel 5 B
EportRX				6	paTrainBGroup2[6]	Train channel 6 B
EportRX				7	paTrainBGroup2[7]	Train channel 7 B
EportRX		128	inEportCtr66	0	paTrainCGroup2[0]	Train channel 0 C
EportRX				1	paTrainCGroup2[1]	Train channel 1 C
EportRX				2	paTrainCGroup2[2]	Train channel 2 C
EportRX				3	paTrainCGroup2[3]	Train channel 3 C
EportRX				4	paTrainCGroup2[4]	Train channel 4 C
EportRX				5	paTrainCGroup2[5]	Train channel 5 C
EportRX				6	paTrainCGroup2[6]	Train channel 6 C
EportRX				7	paTrainCGroup2[7]	Train channel 7 C
EportRX		129	inEportCtr67	0	paEnableAGroup2[0]	Enable channel 0 A
EportRX				1	paEnableAGroup2[1]	Enable channel 1 A

EportRX				2	paEnableAGroup2[2]	Enable channel 2 A
EportRX				3	paEnableAGroup2[3]	Enable channel 3 A
EportRX				4	paEnableAGroup2[4]	Enable channel 4 A
EportRX				5	paEnableAGroup2[5]	Enable channel 5 A
EportRX				6	paEnableAGroup2[6]	Enable channel 6 A
EportRX				7	paEnableAGroup2[7]	Enable channel 7 A
EportRX		130	inEportCtr68	0	paEnableBGroup2[0]	Enable channel 0 B
EportRX				1	paEnableBGroup2[1]	Enable channel 1 B
EportRX				2	paEnableBGroup2[2]	Enable channel 2 B
EportRX				3	paEnableBGroup2[3]	Enable channel 3 B
EportRX				4	paEnableBGroup2[4]	Enable channel 4 B
EportRX				5	paEnableBGroup2[5]	Enable channel 5 B
EportRX				6	paEnableBGroup2[6]	Enable channel 6 B
EportRX				7	paEnableBGroup2[7]	Enable channel 7 B
EportRX		131	inEportCtr69	0	paEnableCGroup2[0]	Enable channel 0 C
EportRX				1	paEnableCGroup2[1]	Enable channel 1 C
EportRX				2	paEnableCGroup2[2]	Enable channel 2 C
EportRX				3	paEnableCGroup2[3]	Enable channel 3 C
EportRX				4	paEnableCGroup2[4]	Enable channel 4 C
EportRX				5	paEnableCGroup2[5]	Enable channel 5 C
EportRX				6	paEnableCGroup2[6]	Enable channel 6 C
EportRX				7	paEnableCGroup2[7]	Enable channel 7 C
EportRX		132	inEportCtr70	0	paResetAGroup2[0]	Reset channel 0 A
EportRX				1	paResetAGroup2[1]	Reset channel 1 A
EportRX				2	paResetAGroup2[2]	Reset channel 2 A
EportRX				3	paResetAGroup2[3]	Reset channel 3 A
EportRX				4	paResetAGroup2[4]	Reset channel 4 A
EportRX				5	paResetAGroup2[5]	Reset channel 5 A
EportRX				6	paResetAGroup2[6]	Reset channel 6 A
EportRX				7	paResetAGroup2[7]	Reset channel 7 A
EportRX		133	inEportCtr71	0	paResetBGroup2[0]	Reset channel 0 B
EportRX				1	paResetBGroup2[1]	Reset channel 1 B
EportRX				2	paResetBGroup2[2]	Reset channel 2 B
EportRX				3	paResetBGroup2[3]	Reset channel 3 B
EportRX				4	paResetBGroup2[4]	Reset channel 4 B
EportRX				5	paResetBGroup2[5]	Reset channel 5 B
EportRX				6	paResetBGroup2[6]	Reset channel 6 B
EportRX				7	paResetBGroup2[7]	Reset channel 7 B
EportRX		134	inEportCtr72	0	paResetCGroup2[0]	Reset channel 0 C
EportRX				1	paResetCGroup2[1]	Reset channel 1 C
EportRX				2	paResetCGroup2[2]	Reset channel 2 C
EportRX				3	paResetCGroup2[3]	Reset channel 3 C
EportRX				4	paResetCGroup2[4]	Reset channel 4 C
EportRX				5	paResetCGroup2[5]	Reset channel 5 C
EportRX				6	paResetCGroup2[6]	Reset channel 6 C
EportRX				7	paResetCGroup2[7]	Reset channel 7 C
EportRX		135	inEportCtr73	0	paDataRateAGroup3[0]	Set Phase-aligner speed A
EportRX				1	paDataRateAGroup3[1]	Set Phase-aligner speed A
EportRX				2	paDataRateBGroup3[0]	Set Phase-aligner speed B
EportRX				3	paDataRateBGroup3[1]	Set Phase-aligner speed B
EportRX				4	paDataRateCGroup3[0]	Set Phase-aligner speed C
EportRX				5	paDataRateCGroup3[1]	Set Phase-aligner speed C
				6		unused
				7		unused
EportRX		136	inEportCtr74	0	paDllConfigAGroup3[0]	Set Phase-aligner dll A
EportRX				1	paDllConfigAGroup3[1]	Set Phase-aligner dll A
EportRX				2	paDllConfigAGroup3[2]	Set Phase-aligner dll A
EportRX				3	paDllConfigAGroup3[3]	Set Phase-aligner dll A
EportRX				4	paDllConfigBGroup3[0]	Set Phase-aligner dll B
EportRX				5	paDllConfigBGroup3[1]	Set Phase-aligner dll B

EportRX				6	paDllConfigBGroup3[2]	Set Phase-aligner dll B
EportRX				7	paDllConfigBGroup3[3]	Set Phase-aligner dll B
EportRX		137	inEportCtr75	0	paDllConfigCGroup3[0]	Set Phase-aligner dll C
EportRX				1	paDllConfigCGroup3[1]	Set Phase-aligner dll C
EportRX				2	paDllConfigCGroup3[2]	Set Phase-aligner dll C
EportRX				3	paDllConfigCGroup3[3]	Set Phase-aligner dll C
EportRX				4	paDllResetAGroup3	Reset Phase-aligner dll A
EportRX				5	paDllResetBGroup3	Reset Phase-aligner dll B
EportRX				6	paDllResetCGroup3	Reset Phase-aligner dll C
				7		unused
EportRX		138	inEportCtr76	0	paPhaseSelectAGroup3[0]	Select sample phase channel 6 A
EportRX				1	paPhaseSelectAGroup3[1]	Select sample phase channel 6 A
EportRX				2	paPhaseSelectAGroup3[2]	Select sample phase channel 6 A
EportRX				3	paPhaseSelectAGroup3[3]	Select sample phase channel 6 A
EportRX				4	paPhaseSelectAGroup3[4]	Select sample phase channel 7 A
EportRX				5	paPhaseSelectAGroup3[5]	Select sample phase channel 7 A
EportRX				6	paPhaseSelectAGroup3[6]	Select sample phase channel 7 A
EportRX				7	paPhaseSelectAGroup3[7]	Select sample phase channel 7 A
EportRX		139	inEportCtr77	0	paPhaseSelectAGroup3[8]	Select sample phase channel 4 A
EportRX				1	paPhaseSelectAGroup3[9]	Select sample phase channel 4 A
EportRX				2	paPhaseSelectAGroup3[10]	Select sample phase channel 4 A
EportRX				3	paPhaseSelectAGroup3[11]	Select sample phase channel 4 A
EportRX				4	paPhaseSelectAGroup3[12]	Select sample phase channel 5 A
EportRX				5	paPhaseSelectAGroup3[13]	Select sample phase channel 5 A
EportRX				6	paPhaseSelectAGroup3[14]	Select sample phase channel 5 A
EportRX				7	paPhaseSelectAGroup3[15]	Select sample phase channel 5 A
EportRX		140	inEportCtr78	0	paPhaseSelectAGroup3[16]	Select sample phase channel 2 A
EportRX				1	paPhaseSelectAGroup3[17]	Select sample phase channel 2 A
EportRX				2	paPhaseSelectAGroup3[18]	Select sample phase channel 2 A
EportRX				3	paPhaseSelectAGroup3[19]	Select sample phase channel 2 A
EportRX				4	paPhaseSelectAGroup3[20]	Select sample phase channel 3 A
EportRX				5	paPhaseSelectAGroup3[21]	Select sample phase channel 3 A
EportRX				6	paPhaseSelectAGroup3[22]	Select sample phase channel 3 A
EportRX				7	paPhaseSelectAGroup3[23]	Select sample phase channel 3 A
EportRX		141	inEportCtr79	0	paPhaseSelectAGroup3[24]	Select sample phase channel 0 A
EportRX				1	paPhaseSelectAGroup3[25]	Select sample phase channel 0 A
EportRX				2	paPhaseSelectAGroup3[26]	Select sample phase channel 0 A
EportRX				3	paPhaseSelectAGroup3[27]	Select sample phase channel 0 A
EportRX				4	paPhaseSelectAGroup3[28]	Select sample phase channel 1 A
EportRX				5	paPhaseSelectAGroup3[29]	Select sample phase channel 1 A
EportRX				6	paPhaseSelectAGroup3[30]	Select sample phase channel 1 A
EportRX				7	paPhaseSelectAGroup3[31]	Select sample phase channel 1 A
EportRX		142	inEportCtr80	0	paPhaseSelectBGroup3[0]	Select sample phase channel 6 B
EportRX				1	paPhaseSelectBGroup3[1]	Select sample phase channel 6 B
EportRX				2	paPhaseSelectBGroup3[2]	Select sample phase channel 6 B
EportRX				3	paPhaseSelectBGroup3[3]	Select sample phase channel 6 B
EportRX				4	paPhaseSelectBGroup3[4]	Select sample phase channel 7 B
EportRX				5	paPhaseSelectBGroup3[5]	Select sample phase channel 7 B
EportRX				6	paPhaseSelectBGroup3[6]	Select sample phase channel 7 B
EportRX				7	paPhaseSelectBGroup3[7]	Select sample phase channel 7 B
EportRX		143	inEportCtr81	0	paPhaseSelectBGroup3[8]	Select sample phase channel 4 B
EportRX				1	paPhaseSelectBGroup3[9]	Select sample phase channel 4 B
EportRX				2	paPhaseSelectBGroup3[10]	Select sample phase channel 4 B
EportRX				3	paPhaseSelectBGroup3[11]	Select sample phase channel 4 B
EportRX				4	paPhaseSelectBGroup3[12]	Select sample phase channel 5 B
EportRX				5	paPhaseSelectBGroup3[13]	Select sample phase channel 5 B
EportRX				6	paPhaseSelectBGroup3[14]	Select sample phase channel 5 B
EportRX				7	paPhaseSelectBGroup3[15]	Select sample phase channel 5 B
EportRX		144	inEportCtr82	0	paPhaseSelectBGroup3[16]	Select sample phase channel 2 B
EportRX				1	paPhaseSelectBGroup3[17]	Select sample phase channel 2 B

EportRX				2	paPhaseSelectBGroup3[18]	Select sample phase channel 2 B
EportRX				3	paPhaseSelectBGroup3[19]	Select sample phase channel 2 B
EportRX				4	paPhaseSelectBGroup3[20]	Select sample phase channel 3 B
EportRX				5	paPhaseSelectBGroup3[21]	Select sample phase channel 3 B
EportRX				6	paPhaseSelectBGroup3[22]	Select sample phase channel 3 B
EportRX				7	paPhaseSelectBGroup3[23]	Select sample phase channel 3 B
EportRX		145	inEportCtr83	0	paPhaseSelectBGroup3[24]	Select sample phase channel 0 B
EportRX				1	paPhaseSelectBGroup3[25]	Select sample phase channel 0 B
EportRX				2	paPhaseSelectBGroup3[26]	Select sample phase channel 0 B
EportRX				3	paPhaseSelectBGroup3[27]	Select sample phase channel 0 B
EportRX				4	paPhaseSelectBGroup3[28]	Select sample phase channel 1 B
EportRX				5	paPhaseSelectBGroup3[29]	Select sample phase channel 1 B
EportRX				6	paPhaseSelectBGroup3[30]	Select sample phase channel 1 B
EportRX				7	paPhaseSelectBGroup3[31]	Select sample phase channel 1 B
EportRX		146	inEportCtr84	0	paPhaseSelectCGroup3[0]	Select sample phase channel 6 C
EportRX				1	paPhaseSelectCGroup3[1]	Select sample phase channel 6 C
EportRX				2	paPhaseSelectCGroup3[2]	Select sample phase channel 6 C
EportRX				3	paPhaseSelectCGroup3[3]	Select sample phase channel 6 C
EportRX				4	paPhaseSelectCGroup3[4]	Select sample phase channel 7 C
EportRX				5	paPhaseSelectCGroup3[5]	Select sample phase channel 7 C
EportRX				6	paPhaseSelectCGroup3[6]	Select sample phase channel 7 C
EportRX				7	paPhaseSelectCGroup3[7]	Select sample phase channel 7 C
EportRX		147	inEportCtr85	0	paPhaseSelectCGroup3[8]	Select sample phase channel 4 C
EportRX				1	paPhaseSelectCGroup3[9]	Select sample phase channel 4 C
EportRX				2	paPhaseSelectCGroup3[10]	Select sample phase channel 4 C
EportRX				3	paPhaseSelectCGroup3[11]	Select sample phase channel 4 C
EportRX				4	paPhaseSelectCGroup3[12]	Select sample phase channel 5 C
EportRX				5	paPhaseSelectCGroup3[13]	Select sample phase channel 5 C
EportRX				6	paPhaseSelectCGroup3[14]	Select sample phase channel 5 C
EportRX				7	paPhaseSelectCGroup3[15]	Select sample phase channel 5 C
EportRX		148	inEportCtr86	0	paPhaseSelectCGroup3[16]	Select sample phase channel 2 C
EportRX				1	paPhaseSelectCGroup3[17]	Select sample phase channel 2 C
EportRX				2	paPhaseSelectCGroup3[18]	Select sample phase channel 2 C
EportRX				3	paPhaseSelectCGroup3[19]	Select sample phase channel 2 C
EportRX				4	paPhaseSelectCGroup3[20]	Select sample phase channel 3 C
EportRX				5	paPhaseSelectCGroup3[21]	Select sample phase channel 3 C
EportRX				6	paPhaseSelectCGroup3[22]	Select sample phase channel 3 C
EportRX				7	paPhaseSelectCGroup3[23]	Select sample phase channel 3 C
EportRX		149	inEportCtr87	0	paPhaseSelectCGroup3[24]	Select sample phase channel 0 C
EportRX				1	paPhaseSelectCGroup3[25]	Select sample phase channel 0 C
EportRX				2	paPhaseSelectCGroup3[26]	Select sample phase channel 0 C
EportRX				3	paPhaseSelectCGroup3[27]	Select sample phase channel 0 C
EportRX				4	paPhaseSelectCGroup3[28]	Select sample phase channel 1 C
EportRX				5	paPhaseSelectCGroup3[29]	Select sample phase channel 1 C
EportRX				6	paPhaseSelectCGroup3[30]	Select sample phase channel 1 C
EportRX				7	paPhaseSelectCGroup3[31]	Select sample phase channel 1 C
EportRX		150	inEportCtr88	0	paTrainAGroup3[0]	Train channel 0 A
EportRX				1	paTrainAGroup3[1]	Train channel 1 A
EportRX				2	paTrainAGroup3[2]	Train channel 2 A
EportRX				3	paTrainAGroup3[3]	Train channel 3 A
EportRX				4	paTrainAGroup3[4]	Train channel 4 A
EportRX				5	paTrainAGroup3[5]	Train channel 5 A
EportRX				6	paTrainAGroup3[6]	Train channel 6 A
EportRX				7	paTrainAGroup3[7]	Train channel 7 A
EportRX		151	inEportCtr89	0	paTrainBGroup3[0]	Train channel 0 B
EportRX				1	paTrainBGroup3[1]	Train channel 1 B
EportRX				2	paTrainBGroup3[2]	Train channel 2 B
EportRX				3	paTrainBGroup3[3]	Train channel 3 B
EportRX				4	paTrainBGroup3[4]	Train channel 4 B
EportRX				5	paTrainBGroup3[5]	Train channel 5 B

EportRX				6	paTrainBGroup3[6]	Train channel 6 B
EportRX				7	paTrainBGroup3[7]	Train channel 7 B
EportRX		152	inEportCtr90	0	paTrainCGroup3[0]	Train channel 0 C
EportRX				1	paTrainCGroup3[1]	Train channel 1 C
EportRX				2	paTrainCGroup3[2]	Train channel 2 C
EportRX				3	paTrainCGroup3[3]	Train channel 3 C
EportRX				4	paTrainCGroup3[4]	Train channel 4 C
EportRX				5	paTrainCGroup3[5]	Train channel 5 C
EportRX				6	paTrainCGroup3[6]	Train channel 6 C
EportRX				7	paTrainCGroup3[7]	Train channel 7 C
EportRX		153	inEportCtr91	0	paEnableAGroup3[0]	Enable channel 0 A
EportRX				1	paEnableAGroup3[1]	Enable channel 1 A
EportRX				2	paEnableAGroup3[2]	Enable channel 2 A
EportRX				3	paEnableAGroup3[3]	Enable channel 3 A
EportRX				4	paEnableAGroup3[4]	Enable channel 4 A
EportRX				5	paEnableAGroup3[5]	Enable channel 5 A
EportRX				6	paEnableAGroup3[6]	Enable channel 6 A
EportRX				7	paEnableAGroup3[7]	Enable channel 7 A
EportRX		154	inEportCtr92	0	paEnableBGroup3[0]	Enable channel 0 B
EportRX				1	paEnableBGroup3[1]	Enable channel 1 B
EportRX				2	paEnableBGroup3[2]	Enable channel 2 B
EportRX				3	paEnableBGroup3[3]	Enable channel 3 B
EportRX				4	paEnableBGroup3[4]	Enable channel 4 B
EportRX				5	paEnableBGroup3[5]	Enable channel 5 B
EportRX				6	paEnableBGroup3[6]	Enable channel 6 B
EportRX				7	paEnableBGroup3[7]	Enable channel 7 B
EportRX		155	inEportCtr93	0	paEnableCGroup3[0]	Enable channel 0 C
EportRX				1	paEnableCGroup3[1]	Enable channel 1 C
EportRX				2	paEnableCGroup3[2]	Enable channel 2 C
EportRX				3	paEnableCGroup3[3]	Enable channel 3 C
EportRX				4	paEnableCGroup3[4]	Enable channel 4 C
EportRX				5	paEnableCGroup3[5]	Enable channel 5 C
EportRX				6	paEnableCGroup3[6]	Enable channel 6 C
EportRX				7	paEnableCGroup3[7]	Enable channel 7 C
EportRX		156	inEportCtr94	0	paResetAGroup3[0]	Reset channel 0 A
EportRX				1	paResetAGroup3[1]	Reset channel 1 A
EportRX				2	paResetAGroup3[2]	Reset channel 2 A
EportRX				3	paResetAGroup3[3]	Reset channel 3 A
EportRX				4	paResetAGroup3[4]	Reset channel 4 A
EportRX				5	paResetAGroup3[5]	Reset channel 5 A
EportRX				6	paResetAGroup3[6]	Reset channel 6 A
EportRX				7	paResetAGroup3[7]	Reset channel 7 A
EportRX		157	inEportCtr95	0	paResetBGroup3[0]	Reset channel 0 B
EportRX				1	paResetBGroup3[1]	Reset channel 1 B
EportRX				2	paResetBGroup3[2]	Reset channel 2 B
EportRX				3	paResetBGroup3[3]	Reset channel 3 B
EportRX				4	paResetBGroup3[4]	Reset channel 4 B
EportRX				5	paResetBGroup3[5]	Reset channel 5 B
EportRX				6	paResetBGroup3[6]	Reset channel 6 B
EportRX				7	paResetBGroup3[7]	Reset channel 7 B
EportRX		158	inEportCtr96	0	paResetCGroup3[0]	Reset channel 0 C
EportRX				1	paResetCGroup3[1]	Reset channel 1 C
EportRX				2	paResetCGroup3[2]	Reset channel 2 C
EportRX				3	paResetCGroup3[3]	Reset channel 3 C
EportRX				4	paResetCGroup3[4]	Reset channel 4 C
EportRX				5	paResetCGroup3[5]	Reset channel 5 C
EportRX				6	paResetCGroup3[6]	Reset channel 6 C
EportRX				7	paResetCGroup3[7]	Reset channel 7 C
EportRX		159	inEportCtr97	0	paDataRateAGroup4[0]	Set Phase-aligner speed A
EportRX				1	paDataRateAGroup4[1]	Set Phase-aligner speed A

EportRX				2	paDataRateBGroup4[0]	Set Phase-aligner speed B
EportRX				3	paDataRateBGroup4[1]	Set Phase-aligner speed B
EportRX				4	paDataRateCGroup4[0]	Set Phase-aligner speed C
EportRX				5	paDataRateCGroup4[1]	Set Phase-aligner speed C
				6		unused
				7		unused
EportRX		160	inEportCtr98	0	paDllConfigAGroup4[0]	Set Phase-aligner dll A
EportRX				1	paDllConfigAGroup4[1]	Set Phase-aligner dll A
EportRX				2	paDllConfigAGroup4[2]	Set Phase-aligner dll A
EportRX				3	paDllConfigAGroup4[3]	Set Phase-aligner dll A
EportRX				4	paDllConfigBGroup4[0]	Set Phase-aligner dll B
EportRX				5	paDllConfigBGroup4[1]	Set Phase-aligner dll B
EportRX				6	paDllConfigBGroup4[2]	Set Phase-aligner dll B
EportRX				7	paDllConfigBGroup4[3]	Set Phase-aligner dll B
EportRX		161	inEportCtr99	0	paDllConfigCGroup4[0]	Set Phase-aligner dll C
EportRX				1	paDllConfigCGroup4[1]	Set Phase-aligner dll C
EportRX				2	paDllConfigCGroup4[2]	Set Phase-aligner dll C
EportRX				3	paDllConfigCGroup4[3]	Set Phase-aligner dll C
EportRX				4	paDllResetAGroup4	Reset Phase-aligner dll A
EportRX				5	paDllResetBGroup4	Reset Phase-aligner dll B
EportRX				6	paDllResetCGroup4	Reset Phase-aligner dll C
				7		unused
EportRX		162	inEportCtr100	0	paPhaseSelectAGroup4[0]	Select sample phase channel 6 A
EportRX				1	paPhaseSelectAGroup4[1]	Select sample phase channel 6 A
EportRX				2	paPhaseSelectAGroup4[2]	Select sample phase channel 6 A
EportRX				3	paPhaseSelectAGroup4[3]	Select sample phase channel 6 A
EportRX				4	paPhaseSelectAGroup4[4]	Select sample phase channel 7 A
EportRX				5	paPhaseSelectAGroup4[5]	Select sample phase channel 7 A
EportRX				6	paPhaseSelectAGroup4[6]	Select sample phase channel 7 A
EportRX				7	paPhaseSelectAGroup4[7]	Select sample phase channel 7 A
EportRX		163	inEportCtr101	0	paPhaseSelectAGroup4[8]	Select sample phase channel 4 A
EportRX				1	paPhaseSelectAGroup4[9]	Select sample phase channel 4 A
EportRX				2	paPhaseSelectAGroup4[10]	Select sample phase channel 4 A
EportRX				3	paPhaseSelectAGroup4[11]	Select sample phase channel 4 A
EportRX				4	paPhaseSelectAGroup4[12]	Select sample phase channel 5 A
EportRX				5	paPhaseSelectAGroup4[13]	Select sample phase channel 5 A
EportRX				6	paPhaseSelectAGroup4[14]	Select sample phase channel 5 A
EportRX				7	paPhaseSelectAGroup4[15]	Select sample phase channel 5 A
EportRX		164	inEportCtr102	0	paPhaseSelectAGroup4[16]	Select sample phase channel 2 A
EportRX				1	paPhaseSelectAGroup4[17]	Select sample phase channel 2 A
EportRX				2	paPhaseSelectAGroup4[18]	Select sample phase channel 2 A
EportRX				3	paPhaseSelectAGroup4[19]	Select sample phase channel 2 A
EportRX				4	paPhaseSelectAGroup4[20]	Select sample phase channel 3 A
EportRX				5	paPhaseSelectAGroup4[21]	Select sample phase channel 3 A
EportRX				6	paPhaseSelectAGroup4[22]	Select sample phase channel 3 A
EportRX				7	paPhaseSelectAGroup4[23]	Select sample phase channel 3 A
EportRX		165	inEportCtr103	0	paPhaseSelectAGroup4[24]	Select sample phase channel 0 A
EportRX				1	paPhaseSelectAGroup4[25]	Select sample phase channel 0 A
EportRX				2	paPhaseSelectAGroup4[26]	Select sample phase channel 0 A
EportRX				3	paPhaseSelectAGroup4[27]	Select sample phase channel 0 A
EportRX				4	paPhaseSelectAGroup4[28]	Select sample phase channel 1 A
EportRX				5	paPhaseSelectAGroup4[29]	Select sample phase channel 1 A
EportRX				6	paPhaseSelectAGroup4[30]	Select sample phase channel 1 A
EportRX				7	paPhaseSelectAGroup4[31]	Select sample phase channel 1 A
EportRX		166	inEportCtr104	0	paPhaseSelectBGroup4[0]	Select sample phase channel 6 B
EportRX				1	paPhaseSelectBGroup4[1]	Select sample phase channel 6 B
EportRX				2	paPhaseSelectBGroup4[2]	Select sample phase channel 6 B
EportRX				3	paPhaseSelectBGroup4[3]	Select sample phase channel 6 B
EportRX				4	paPhaseSelectBGroup4[4]	Select sample phase channel 7 B
EportRX				5	paPhaseSelectBGroup4[5]	Select sample phase channel 7 B

EportRX				6	paPhaseSelectBGroup4[6]	Select sample phase channel 7 B
EportRX				7	paPhaseSelectBGroup4[7]	Select sample phase channel 7 B
EportRX		167	inEportCtr105	0	paPhaseSelectBGroup4[8]	Select sample phase channel 4 B
EportRX				1	paPhaseSelectBGroup4[9]	Select sample phase channel 4 B
EportRX				2	paPhaseSelectBGroup4[10]	Select sample phase channel 4 B
EportRX				3	paPhaseSelectBGroup4[11]	Select sample phase channel 4 B
EportRX				4	paPhaseSelectBGroup4[12]	Select sample phase channel 5 B
EportRX				5	paPhaseSelectBGroup4[13]	Select sample phase channel 5 B
EportRX				6	paPhaseSelectBGroup4[14]	Select sample phase channel 5 B
EportRX				7	paPhaseSelectBGroup4[15]	Select sample phase channel 5 B
EportRX		168	inEportCtr106	0	paPhaseSelectBGroup4[16]	Select sample phase channel 2 B
EportRX				1	paPhaseSelectBGroup4[17]	Select sample phase channel 2 B
EportRX				2	paPhaseSelectBGroup4[18]	Select sample phase channel 2 B
EportRX				3	paPhaseSelectBGroup4[19]	Select sample phase channel 2 B
EportRX				4	paPhaseSelectBGroup4[20]	Select sample phase channel 3 B
EportRX				5	paPhaseSelectBGroup4[21]	Select sample phase channel 3 B
EportRX				6	paPhaseSelectBGroup4[22]	Select sample phase channel 3 B
EportRX				7	paPhaseSelectBGroup4[23]	Select sample phase channel 3 B
EportRX		169	inEportCtr107	0	paPhaseSelectBGroup4[24]	Select sample phase channel 0 B
EportRX				1	paPhaseSelectBGroup4[25]	Select sample phase channel 0 B
EportRX				2	paPhaseSelectBGroup4[26]	Select sample phase channel 0 B
EportRX				3	paPhaseSelectBGroup4[27]	Select sample phase channel 0 B
EportRX				4	paPhaseSelectBGroup4[28]	Select sample phase channel 1 B
EportRX				5	paPhaseSelectBGroup4[29]	Select sample phase channel 1 B
EportRX				6	paPhaseSelectBGroup4[30]	Select sample phase channel 1 B
EportRX				7	paPhaseSelectBGroup4[31]	Select sample phase channel 1 B
EportRX		170	inEportCtr108	0	paPhaseSelectCGroup4[0]	Select sample phase channel 6 C
EportRX				1	paPhaseSelectCGroup4[1]	Select sample phase channel 6 C
EportRX				2	paPhaseSelectCGroup4[2]	Select sample phase channel 6 C
EportRX				3	paPhaseSelectCGroup4[3]	Select sample phase channel 6 C
EportRX				4	paPhaseSelectCGroup4[4]	Select sample phase channel 7 C
EportRX				5	paPhaseSelectCGroup4[5]	Select sample phase channel 7 C
EportRX				6	paPhaseSelectCGroup4[6]	Select sample phase channel 7 C
EportRX				7	paPhaseSelectCGroup4[7]	Select sample phase channel 7 C
EportRX		171	inEportCtr109	0	paPhaseSelectCGroup4[8]	Select sample phase channel 4 C
EportRX				1	paPhaseSelectCGroup4[9]	Select sample phase channel 4 C
EportRX				2	paPhaseSelectCGroup4[10]	Select sample phase channel 4 C
EportRX				3	paPhaseSelectCGroup4[11]	Select sample phase channel 4 C
EportRX				4	paPhaseSelectCGroup4[12]	Select sample phase channel 5 C
EportRX				5	paPhaseSelectCGroup4[13]	Select sample phase channel 5 C
EportRX				6	paPhaseSelectCGroup4[14]	Select sample phase channel 5 C
EportRX				7	paPhaseSelectCGroup4[15]	Select sample phase channel 5 C
EportRX		172	inEportCtr110	0	paPhaseSelectCGroup4[16]	Select sample phase channel 2 C
EportRX				1	paPhaseSelectCGroup4[17]	Select sample phase channel 2 C
EportRX				2	paPhaseSelectCGroup4[18]	Select sample phase channel 2 C
EportRX				3	paPhaseSelectCGroup4[19]	Select sample phase channel 2 C
EportRX				4	paPhaseSelectCGroup4[20]	Select sample phase channel 3 C
EportRX				5	paPhaseSelectCGroup4[21]	Select sample phase channel 3 C
EportRX				6	paPhaseSelectCGroup4[22]	Select sample phase channel 3 C
EportRX				7	paPhaseSelectCGroup4[23]	Select sample phase channel 3 C
EportRX		173	inEportCtr111	0	paPhaseSelectCGroup4[24]	Select sample phase channel 0 C
EportRX				1	paPhaseSelectCGroup4[25]	Select sample phase channel 0 C
EportRX				2	paPhaseSelectCGroup4[26]	Select sample phase channel 0 C
EportRX				3	paPhaseSelectCGroup4[27]	Select sample phase channel 0 C
EportRX				4	paPhaseSelectCGroup4[28]	Select sample phase channel 1 C
EportRX				5	paPhaseSelectCGroup4[29]	Select sample phase channel 1 C
EportRX				6	paPhaseSelectCGroup4[30]	Select sample phase channel 1 C
EportRX				7	paPhaseSelectCGroup4[31]	Select sample phase channel 1 C
EportRX		174	inEportCtr112	0	paTrainAGroup4[0]	Train channel 0 A
EportRX				1	paTrainAGroup4[1]	Train channel 1 A

EportRX				2	paTrainAGroup4[2]	Train channel 2 A
EportRX				3	paTrainAGroup4[3]	Train channel 3 A
EportRX				4	paTrainAGroup4[4]	Train channel 4 A
EportRX				5	paTrainAGroup4[5]	Train channel 5 A
EportRX				6	paTrainAGroup4[6]	Train channel 6 A
EportRX				7	paTrainAGroup4[7]	Train channel 7 A
EportRX		175	inEportCtr113	0	paTrainBGroup4[0]	Train channel 0 B
EportRX				1	paTrainBGroup4[1]	Train channel 1 B
EportRX				2	paTrainBGroup4[2]	Train channel 2 B
EportRX				3	paTrainBGroup4[3]	Train channel 3 B
EportRX				4	paTrainBGroup4[4]	Train channel 4 B
EportRX				5	paTrainBGroup4[5]	Train channel 5 B
EportRX				6	paTrainBGroup4[6]	Train channel 6 B
EportRX				7	paTrainBGroup4[7]	Train channel 7 B
EportRX		176	inEportCtr114	0	paTrainCGroup4[0]	Train channel 0 C
EportRX				1	paTrainCGroup4[1]	Train channel 1 C
EportRX				2	paTrainCGroup4[2]	Train channel 2 C
EportRX				3	paTrainCGroup4[3]	Train channel 3 C
EportRX				4	paTrainCGroup4[4]	Train channel 4 C
EportRX				5	paTrainCGroup4[5]	Train channel 5 C
EportRX				6	paTrainCGroup4[6]	Train channel 6 C
EportRX				7	paTrainCGroup4[7]	Train channel 7 C
EportRX		177	inEportCtr115	0	paEnableAGroup4[0]	Enable channel 0 A
EportRX				1	paEnableAGroup4[1]	Enable channel 1 A
EportRX				2	paEnableAGroup4[2]	Enable channel 2 A
EportRX				3	paEnableAGroup4[3]	Enable channel 3 A
EportRX				4	paEnableAGroup4[4]	Enable channel 4 A
EportRX				5	paEnableAGroup4[5]	Enable channel 5 A
EportRX				6	paEnableAGroup4[6]	Enable channel 6 A
EportRX				7	paEnableAGroup4[7]	Enable channel 7 A
EportRX		178	inEportCtr116	0	paEnableBGroup4[0]	Enable channel 0 B
EportRX				1	paEnableBGroup4[1]	Enable channel 1 B
EportRX				2	paEnableBGroup4[2]	Enable channel 2 B
EportRX				3	paEnableBGroup4[3]	Enable channel 3 B
EportRX				4	paEnableBGroup4[4]	Enable channel 4 B
EportRX				5	paEnableBGroup4[5]	Enable channel 5 B
EportRX				6	paEnableBGroup4[6]	Enable channel 6 B
EportRX				7	paEnableBGroup4[7]	Enable channel 7 B
EportRX		179	inEportCtr117	0	paEnableCGroup4[0]	Enable channel 0 C
EportRX				1	paEnableCGroup4[1]	Enable channel 1 C
EportRX				2	paEnableCGroup4[2]	Enable channel 2 C
EportRX				3	paEnableCGroup4[3]	Enable channel 3 C
EportRX				4	paEnableCGroup4[4]	Enable channel 4 C
EportRX				5	paEnableCGroup4[5]	Enable channel 5 C
EportRX				6	paEnableCGroup4[6]	Enable channel 6 C
EportRX				7	paEnableCGroup4[7]	Enable channel 7 C
EportRX		180	inEportCtr118	0	paResetAGroup4[0]	Reset channel 0 A
EportRX				1	paResetAGroup4[1]	Reset channel 1 A
EportRX				2	paResetAGroup4[2]	Reset channel 2 A
EportRX				3	paResetAGroup4[3]	Reset channel 3 A
EportRX				4	paResetAGroup4[4]	Reset channel 4 A
EportRX				5	paResetAGroup4[5]	Reset channel 5 A
EportRX				6	paResetAGroup4[6]	Reset channel 6 A
EportRX				7	paResetAGroup4[7]	Reset channel 7 A
EportRX		181	inEportCtr119	0	paResetBGroup4[0]	Reset channel 0 B
EportRX				1	paResetBGroup4[1]	Reset channel 1 B
EportRX				2	paResetBGroup4[2]	Reset channel 2 B
EportRX				3	paResetBGroup4[3]	Reset channel 3 B
EportRX				4	paResetBGroup4[4]	Reset channel 4 B
EportRX				5	paResetBGroup4[5]	Reset channel 5 B

EportRX				6	paResetBGroup4[6]	Reset channel 6 B
EportRX				7	paResetBGroup4[7]	Reset channel 7 B
EportRX		182	inEportCtr120	0	paResetCGroup4[0]	Reset channel 0 C
EportRX				1	paResetCGroup4[1]	Reset channel 1 C
EportRX				2	paResetCGroup4[2]	Reset channel 2 C
EportRX				3	paResetCGroup4[3]	Reset channel 3 C
EportRX				4	paResetCGroup4[4]	Reset channel 4 C
EportRX				5	paResetCGroup4[5]	Reset channel 5 C
EportRX				6	paResetCGroup4[6]	Reset channel 6 C
EportRX				7	paResetCGroup4[7]	Reset channel 7 C
EportRX		183	inEportCtr121	0	paDataRateAGroup5[0]	Set Phase-aligner speed A
EportRX				1	paDataRateAGroup5[1]	Set Phase-aligner speed A
EportRX				2	paDataRateBGroup5[0]	Set Phase-aligner speed B
EportRX				3	paDataRateBGroup5[1]	Set Phase-aligner speed B
EportRX				4	paDataRateCGroup5[0]	Set Phase-aligner speed C
EportRX				5	paDataRateCGroup5[1]	Set Phase-aligner speed C
				6		unused
				7		unused
EportRX		184	inEportCtr122	0	paDllConfigAGroup5[0]	Set Phase-aligner dll A
EportRX				1	paDllConfigAGroup5[1]	Set Phase-aligner dll A
EportRX				2	paDllConfigAGroup5[2]	Set Phase-aligner dll A
EportRX				3	paDllConfigAGroup5[3]	Set Phase-aligner dll A
EportRX				4	paDllConfigBGroup5[0]	Set Phase-aligner dll B
EportRX				5	paDllConfigBGroup5[1]	Set Phase-aligner dll B
EportRX				6	paDllConfigBGroup5[2]	Set Phase-aligner dll B
EportRX				7	paDllConfigBGroup5[3]	Set Phase-aligner dll B
EportRX		185	inEportCtr123	0	paDllConfigCGroup5[0]	Set Phase-aligner dll C
EportRX				1	paDllConfigCGroup5[1]	Set Phase-aligner dll C
EportRX				2	paDllConfigCGroup5[2]	Set Phase-aligner dll C
EportRX				3	paDllConfigCGroup5[3]	Set Phase-aligner dll C
EportRX				4	paDllResetAGroup5	Reset Phase-aligner dll A
EportRX				5	paDllResetBGroup5	Reset Phase-aligner dll B
EportRX				6	paDllResetCGroup5	Reset Phase-aligner dll C
				7		unused
EportRX		186	inEportCtr124	0	paPhaseSelectAGroup5[0]	Select sample phase channel 6 A
EportRX				1	paPhaseSelectAGroup5[1]	Select sample phase channel 6 A
EportRX				2	paPhaseSelectAGroup5[2]	Select sample phase channel 6 A
EportRX				3	paPhaseSelectAGroup5[3]	Select sample phase channel 6 A
EportRX				4	paPhaseSelectAGroup5[4]	Select sample phase channel 7 A
EportRX				5	paPhaseSelectAGroup5[5]	Select sample phase channel 7 A
EportRX				6	paPhaseSelectAGroup5[6]	Select sample phase channel 7 A
EportRX				7	paPhaseSelectAGroup5[7]	Select sample phase channel 7 A
EportRX		187	inEportCtr125	0	paPhaseSelectAGroup5[8]	Select sample phase channel 4 A
EportRX				1	paPhaseSelectAGroup5[9]	Select sample phase channel 4 A
EportRX				2	paPhaseSelectAGroup5[10]	Select sample phase channel 4 A
EportRX				3	paPhaseSelectAGroup5[11]	Select sample phase channel 4 A
EportRX				4	paPhaseSelectAGroup5[12]	Select sample phase channel 5 A
EportRX				5	paPhaseSelectAGroup5[13]	Select sample phase channel 5 A
EportRX				6	paPhaseSelectAGroup5[14]	Select sample phase channel 5 A
EportRX				7	paPhaseSelectAGroup5[15]	Select sample phase channel 5 A
EportRX		188	inEportCtr126	0	paPhaseSelectAGroup5[16]	Select sample phase channel 2 A
EportRX				1	paPhaseSelectAGroup5[17]	Select sample phase channel 2 A
EportRX				2	paPhaseSelectAGroup5[18]	Select sample phase channel 2 A
EportRX				3	paPhaseSelectAGroup5[19]	Select sample phase channel 2 A
EportRX				4	paPhaseSelectAGroup5[20]	Select sample phase channel 3 A
EportRX				5	paPhaseSelectAGroup5[21]	Select sample phase channel 3 A
EportRX				6	paPhaseSelectAGroup5[22]	Select sample phase channel 3 A
EportRX				7	paPhaseSelectAGroup5[23]	Select sample phase channel 3 A
EportRX		189	inEportCtr127	0	paPhaseSelectAGroup5[24]	Select sample phase channel 0 A
EportRX				1	paPhaseSelectAGroup5[25]	Select sample phase channel 0 A

EportRX				2	paPhaseSelectAGroup5[26]	Select sample phase channel 0 A
EportRX				3	paPhaseSelectAGroup5[27]	Select sample phase channel 0 A
EportRX				4	paPhaseSelectAGroup5[28]	Select sample phase channel 1 A
EportRX				5	paPhaseSelectAGroup5[29]	Select sample phase channel 1 A
EportRX				6	paPhaseSelectAGroup5[30]	Select sample phase channel 1 A
EportRX				7	paPhaseSelectAGroup5[31]	Select sample phase channel 1 A
EportRX		190	inEportCtr128	0	paPhaseSelectBGroup5[0]	Select sample phase channel 6 B
EportRX				1	paPhaseSelectBGroup5[1]	Select sample phase channel 6 B
EportRX				2	paPhaseSelectBGroup5[2]	Select sample phase channel 6 B
EportRX				3	paPhaseSelectBGroup5[3]	Select sample phase channel 6 B
EportRX				4	paPhaseSelectBGroup5[4]	Select sample phase channel 7 B
EportRX				5	paPhaseSelectBGroup5[5]	Select sample phase channel 7 B
EportRX				6	paPhaseSelectBGroup5[6]	Select sample phase channel 7 B
EportRX				7	paPhaseSelectBGroup5[7]	Select sample phase channel 7 B
EportRX		191	inEportCtr129	0	paPhaseSelectBGroup5[8]	Select sample phase channel 4 B
EportRX				1	paPhaseSelectBGroup5[9]	Select sample phase channel 4 B
EportRX				2	paPhaseSelectBGroup5[10]	Select sample phase channel 4 B
EportRX				3	paPhaseSelectBGroup5[11]	Select sample phase channel 4 B
EportRX				4	paPhaseSelectBGroup5[12]	Select sample phase channel 5 B
EportRX				5	paPhaseSelectBGroup5[13]	Select sample phase channel 5 B
EportRX				6	paPhaseSelectBGroup5[14]	Select sample phase channel 5 B
EportRX				7	paPhaseSelectBGroup5[15]	Select sample phase channel 5 B
EportRX		192	inEportCtr130	0	paPhaseSelectBGroup5[16]	Select sample phase channel 2 B
EportRX				1	paPhaseSelectBGroup5[17]	Select sample phase channel 2 B
EportRX				2	paPhaseSelectBGroup5[18]	Select sample phase channel 2 B
EportRX				3	paPhaseSelectBGroup5[19]	Select sample phase channel 2 B
EportRX				4	paPhaseSelectBGroup5[20]	Select sample phase channel 3 B
EportRX				5	paPhaseSelectBGroup5[21]	Select sample phase channel 3 B
EportRX				6	paPhaseSelectBGroup5[22]	Select sample phase channel 3 B
EportRX				7	paPhaseSelectBGroup5[23]	Select sample phase channel 3 B
EportRX		193	inEportCtr131	0	paPhaseSelectBGroup5[24]	Select sample phase channel 0 B
EportRX				1	paPhaseSelectBGroup5[25]	Select sample phase channel 0 B
EportRX				2	paPhaseSelectBGroup5[26]	Select sample phase channel 0 B
EportRX				3	paPhaseSelectBGroup5[27]	Select sample phase channel 0 B
EportRX				4	paPhaseSelectBGroup5[28]	Select sample phase channel 1 B
EportRX				5	paPhaseSelectBGroup5[29]	Select sample phase channel 1 B
EportRX				6	paPhaseSelectBGroup5[30]	Select sample phase channel 1 B
EportRX				7	paPhaseSelectBGroup5[31]	Select sample phase channel 1 B
EportRX		194	inEportCtr132	0	paPhaseSelectCGroup5[0]	Select sample phase channel 6 C
EportRX				1	paPhaseSelectCGroup5[1]	Select sample phase channel 6 C
EportRX				2	paPhaseSelectCGroup5[2]	Select sample phase channel 6 C
EportRX				3	paPhaseSelectCGroup5[3]	Select sample phase channel 6 C
EportRX				4	paPhaseSelectCGroup5[4]	Select sample phase channel 7 C
EportRX				5	paPhaseSelectCGroup5[5]	Select sample phase channel 7 C
EportRX				6	paPhaseSelectCGroup5[6]	Select sample phase channel 7 C
EportRX				7	paPhaseSelectCGroup5[7]	Select sample phase channel 7 C
EportRX		195	inEportCtr133	0	paPhaseSelectCGroup5[8]	Select sample phase channel 4 C
EportRX				1	paPhaseSelectCGroup5[9]	Select sample phase channel 4 C
EportRX				2	paPhaseSelectCGroup5[10]	Select sample phase channel 4 C
EportRX				3	paPhaseSelectCGroup5[11]	Select sample phase channel 4 C
EportRX				4	paPhaseSelectCGroup5[12]	Select sample phase channel 5 C
EportRX				5	paPhaseSelectCGroup5[13]	Select sample phase channel 5 C
EportRX				6	paPhaseSelectCGroup5[14]	Select sample phase channel 5 C
EportRX				7	paPhaseSelectCGroup5[15]	Select sample phase channel 5 C
EportRX		196	inEportCtr134	0	paPhaseSelectCGroup5[16]	Select sample phase channel 2 C
EportRX				1	paPhaseSelectCGroup5[17]	Select sample phase channel 2 C
EportRX				2	paPhaseSelectCGroup5[18]	Select sample phase channel 2 C
EportRX				3	paPhaseSelectCGroup5[19]	Select sample phase channel 2 C
EportRX				4	paPhaseSelectCGroup5[20]	Select sample phase channel 3 C
EportRX				5	paPhaseSelectCGroup5[21]	Select sample phase channel 3 C

EportRX				6	paPhaseSelectCGroup5[22]	Select sample phase channel 3 C
EportRX				7	paPhaseSelectCGroup5[23]	Select sample phase channel 3 C
EportRX		197	inEportCtr135	0	paPhaseSelectCGroup5[24]	Select sample phase channel 0 C
EportRX				1	paPhaseSelectCGroup5[25]	Select sample phase channel 0 C
EportRX				2	paPhaseSelectCGroup5[26]	Select sample phase channel 0 C
EportRX				3	paPhaseSelectCGroup5[27]	Select sample phase channel 0 C
EportRX				4	paPhaseSelectCGroup5[28]	Select sample phase channel 1 C
EportRX				5	paPhaseSelectCGroup5[29]	Select sample phase channel 1 C
EportRX				6	paPhaseSelectCGroup5[30]	Select sample phase channel 1 C
EportRX				7	paPhaseSelectCGroup5[31]	Select sample phase channel 1 C
EportRX		198	inEportCtr136	0	paTrainAGroup5[0]	Train channel 0 A
EportRX				1	paTrainAGroup5[1]	Train channel 1 A
EportRX				2	paTrainAGroup5[2]	Train channel 2 A
EportRX				3	paTrainAGroup5[3]	Train channel 3 A
EportRX				4	paTrainAGroup5[4]	Train channel 4 A
EportRX				5	paTrainAGroup5[5]	Train channel 5 A
EportRX				6	paTrainAGroup5[6]	Train channel 6 A
EportRX				7	paTrainAGroup5[7]	Train channel 7 A
EportRX		199	inEportCtr137	0	paTrainBGroup5[0]	Train channel 0 B
EportRX				1	paTrainBGroup5[1]	Train channel 1 B
EportRX				2	paTrainBGroup5[2]	Train channel 2 B
EportRX				3	paTrainBGroup5[3]	Train channel 3 B
EportRX				4	paTrainBGroup5[4]	Train channel 4 B
EportRX				5	paTrainBGroup5[5]	Train channel 5 B
EportRX				6	paTrainBGroup5[6]	Train channel 6 B
EportRX				7	paTrainBGroup5[7]	Train channel 7 B
EportRX		200	inEportCtr138	0	paTrainCGroup5[0]	Train channel 0 C
EportRX				1	paTrainCGroup5[1]	Train channel 1 C
EportRX				2	paTrainCGroup5[2]	Train channel 2 C
EportRX				3	paTrainCGroup5[3]	Train channel 3 C
EportRX				4	paTrainCGroup5[4]	Train channel 4 C
EportRX				5	paTrainCGroup5[5]	Train channel 5 C
EportRX				6	paTrainCGroup5[6]	Train channel 6 C
EportRX				7	paTrainCGroup5[7]	Train channel 7 C
EportRX		201	inEportCtr139	0	paEnableAGroup5[0]	Enable channel 0 A
EportRX				1	paEnableAGroup5[1]	Enable channel 1 A
EportRX				2	paEnableAGroup5[2]	Enable channel 2 A
EportRX				3	paEnableAGroup5[3]	Enable channel 3 A
EportRX				4	paEnableAGroup5[4]	Enable channel 4 A
EportRX				5	paEnableAGroup5[5]	Enable channel 5 A
EportRX				6	paEnableAGroup5[6]	Enable channel 6 A
EportRX				7	paEnableAGroup5[7]	Enable channel 7 A
EportRX		202	inEportCtr140	0	paEnableBGroup5[0]	Enable channel 0 B
EportRX				1	paEnableBGroup5[1]	Enable channel 1 B
EportRX				2	paEnableBGroup5[2]	Enable channel 2 B
EportRX				3	paEnableBGroup5[3]	Enable channel 3 B
EportRX				4	paEnableBGroup5[4]	Enable channel 4 B
EportRX				5	paEnableBGroup5[5]	Enable channel 5 B
EportRX				6	paEnableBGroup5[6]	Enable channel 6 B
EportRX				7	paEnableBGroup5[7]	Enable channel 7 B
EportRX		203	inEportCtr141	0	paEnableCGroup5[0]	Enable channel 0 C
EportRX				1	paEnableCGroup5[1]	Enable channel 1 C
EportRX				2	paEnableCGroup5[2]	Enable channel 2 C
EportRX				3	paEnableCGroup5[3]	Enable channel 3 C
EportRX				4	paEnableCGroup5[4]	Enable channel 4 C
EportRX				5	paEnableCGroup5[5]	Enable channel 5 C
EportRX				6	paEnableCGroup5[6]	Enable channel 6 C
EportRX				7	paEnableCGroup5[7]	Enable channel 7 C
EportRX		204	inEportCtr142	0	paResetAGroup5[0]	Reset channel 0 A
EportRX				1	paResetAGroup5[1]	Reset channel 1 A

EportRX				2	paResetAGroup5[2]	Reset channel 2 A
EportRX				3	paResetAGroup5[3]	Reset channel 3 A
EportRX				4	paResetAGroup5[4]	Reset channel 4 A
EportRX				5	paResetAGroup5[5]	Reset channel 5 A
EportRX				6	paResetAGroup5[6]	Reset channel 6 A
EportRX				7	paResetAGroup5[7]	Reset channel 7 A
EportRX		205	inEportCtr143	0	paResetBGroup5[0]	Reset channel 0 B
EportRX				1	paResetBGroup5[1]	Reset channel 1 B
EportRX				2	paResetBGroup5[2]	Reset channel 2 B
EportRX				3	paResetBGroup5[3]	Reset channel 3 B
EportRX				4	paResetBGroup5[4]	Reset channel 4 B
EportRX				5	paResetBGroup5[5]	Reset channel 5 B
EportRX				6	paResetBGroup5[6]	Reset channel 6 B
EportRX				7	paResetBGroup5[7]	Reset channel 7 B
EportRX		206	inEportCtr144	0	paResetCGroup5[0]	Reset channel 0 C
EportRX				1	paResetCGroup5[1]	Reset channel 1 C
EportRX				2	paResetCGroup5[2]	Reset channel 2 C
EportRX				3	paResetCGroup5[3]	Reset channel 3 C
EportRX				4	paResetCGroup5[4]	Reset channel 4 C
EportRX				5	paResetCGroup5[5]	Reset channel 5 C
EportRX				6	paResetCGroup5[6]	Reset channel 6 C
EportRX				7	paResetCGroup5[7]	Reset channel 7 C
EportRX		207	inEportCtr145	0	paDataRateAGroup6[0]	Set Phase-aligner speed A
EportRX				1	paDataRateAGroup6[1]	Set Phase-aligner speed A
EportRX				2	paDataRateBGroup6[0]	Set Phase-aligner speed B
EportRX				3	paDataRateBGroup6[1]	Set Phase-aligner speed B
EportRX				4	paDataRateCGroup6[0]	Set Phase-aligner speed C
EportRX				5	paDataRateCGroup6[1]	Set Phase-aligner speed C
				6		unused
				7		unused
EportRX		208	inEportCtr146	0	paDllConfigAGroup6[0]	Set Phase-aligner dll A
EportRX				1	paDllConfigAGroup6[1]	Set Phase-aligner dll A
EportRX				2	paDllConfigAGroup6[2]	Set Phase-aligner dll A
EportRX				3	paDllConfigAGroup6[3]	Set Phase-aligner dll A
EportRX				4	paDllConfigBGroup6[0]	Set Phase-aligner dll B
EportRX				5	paDllConfigBGroup6[1]	Set Phase-aligner dll B
EportRX				6	paDllConfigBGroup6[2]	Set Phase-aligner dll B
EportRX				7	paDllConfigBGroup6[3]	Set Phase-aligner dll B
EportRX		209	inEportCtr147	0	paDllConfigCGroup6[0]	Set Phase-aligner dll C
EportRX				1	paDllConfigCGroup6[1]	Set Phase-aligner dll C
EportRX				2	paDllConfigCGroup6[2]	Set Phase-aligner dll C
EportRX				3	paDllConfigCGroup6[3]	Set Phase-aligner dll C
EportRX				4	paDllResetAGroup6	Reset Phase-aligner dll A
EportRX				5	paDllResetBGroup6	Reset Phase-aligner dll B
EportRX				6	paDllResetCGroup6	Reset Phase-aligner dll C
				7		unused
EportRX		210	inEportCtr148	0	paPhaseSelectAGroup6[0]	Select sample phase channel 6 A
EportRX				1	paPhaseSelectAGroup6[1]	Select sample phase channel 6 A
EportRX				2	paPhaseSelectAGroup6[2]	Select sample phase channel 6 A
EportRX				3	paPhaseSelectAGroup6[3]	Select sample phase channel 6 A
EportRX				4	paPhaseSelectAGroup6[4]	Select sample phase channel 7 A
EportRX				5	paPhaseSelectAGroup6[5]	Select sample phase channel 7 A
EportRX				6	paPhaseSelectAGroup6[6]	Select sample phase channel 7 A
EportRX				7	paPhaseSelectAGroup6[7]	Select sample phase channel 7 A
EportRX		211	inEportCtr149	0	paPhaseSelectAGroup6[8]	Select sample phase channel 4 A
EportRX				1	paPhaseSelectAGroup6[9]	Select sample phase channel 4 A
EportRX				2	paPhaseSelectAGroup6[10]	Select sample phase channel 4 A
EportRX				3	paPhaseSelectAGroup6[11]	Select sample phase channel 4 A
EportRX				4	paPhaseSelectAGroup6[12]	Select sample phase channel 5 A
EportRX				5	paPhaseSelectAGroup6[13]	Select sample phase channel 5 A

EportRX				6	paPhaseSelectAGroup6[14]	Select sample phase channel 5 A
EportRX				7	paPhaseSelectAGroup6[15]	Select sample phase channel 5 A
EportRX		212	inEportCtr150	0	paPhaseSelectAGroup6[16]	Select sample phase channel 2 A
EportRX				1	paPhaseSelectAGroup6[17]	Select sample phase channel 2 A
EportRX				2	paPhaseSelectAGroup6[18]	Select sample phase channel 2 A
EportRX				3	paPhaseSelectAGroup6[19]	Select sample phase channel 2 A
EportRX				4	paPhaseSelectAGroup6[20]	Select sample phase channel 3 A
EportRX				5	paPhaseSelectAGroup6[21]	Select sample phase channel 3 A
EportRX				6	paPhaseSelectAGroup6[22]	Select sample phase channel 3 A
EportRX				7	paPhaseSelectAGroup6[23]	Select sample phase channel 3 A
EportRX		213	inEportCtr151	0	paPhaseSelectAGroup6[24]	Select sample phase channel 0 A
EportRX				1	paPhaseSelectAGroup6[25]	Select sample phase channel 0 A
EportRX				2	paPhaseSelectAGroup6[26]	Select sample phase channel 0 A
EportRX				3	paPhaseSelectAGroup6[27]	Select sample phase channel 0 A
EportRX				4	paPhaseSelectAGroup6[28]	Select sample phase channel 1 A
EportRX				5	paPhaseSelectAGroup6[29]	Select sample phase channel 1 A
EportRX				6	paPhaseSelectAGroup6[30]	Select sample phase channel 1 A
EportRX				7	paPhaseSelectAGroup6[31]	Select sample phase channel 1 A
EportRX		214	inEportCtr152	0	paPhaseSelectBGroup6[0]	Select sample phase channel 6 B
EportRX				1	paPhaseSelectBGroup6[1]	Select sample phase channel 6 B
EportRX				2	paPhaseSelectBGroup6[2]	Select sample phase channel 6 B
EportRX				3	paPhaseSelectBGroup6[3]	Select sample phase channel 6 B
EportRX				4	paPhaseSelectBGroup6[4]	Select sample phase channel 7 B
EportRX				5	paPhaseSelectBGroup6[5]	Select sample phase channel 7 B
EportRX				6	paPhaseSelectBGroup6[6]	Select sample phase channel 7 B
EportRX				7	paPhaseSelectBGroup6[7]	Select sample phase channel 7 B
EportRX		215	inEportCtr153	0	paPhaseSelectBGroup6[8]	Select sample phase channel 4 B
EportRX				1	paPhaseSelectBGroup6[9]	Select sample phase channel 4 B
EportRX				2	paPhaseSelectBGroup6[10]	Select sample phase channel 4 B
EportRX				3	paPhaseSelectBGroup6[11]	Select sample phase channel 4 B
EportRX				4	paPhaseSelectBGroup6[12]	Select sample phase channel 5 B
EportRX				5	paPhaseSelectBGroup6[13]	Select sample phase channel 5 B
EportRX				6	paPhaseSelectBGroup6[14]	Select sample phase channel 5 B
EportRX				7	paPhaseSelectBGroup6[15]	Select sample phase channel 5 B
EportRX		216	inEportCtr154	0	paPhaseSelectBGroup6[16]	Select sample phase channel 2 B
EportRX				1	paPhaseSelectBGroup6[17]	Select sample phase channel 2 B
EportRX				2	paPhaseSelectBGroup6[18]	Select sample phase channel 2 B
EportRX				3	paPhaseSelectBGroup6[19]	Select sample phase channel 2 B
EportRX				4	paPhaseSelectBGroup6[20]	Select sample phase channel 3 B
EportRX				5	paPhaseSelectBGroup6[21]	Select sample phase channel 3 B
EportRX				6	paPhaseSelectBGroup6[22]	Select sample phase channel 3 B
EportRX				7	paPhaseSelectBGroup6[23]	Select sample phase channel 3 B
EportRX		217	inEportCtr155	0	paPhaseSelectBGroup6[24]	Select sample phase channel 0 B
EportRX				1	paPhaseSelectBGroup6[25]	Select sample phase channel 0 B
EportRX				2	paPhaseSelectBGroup6[26]	Select sample phase channel 0 B
EportRX				3	paPhaseSelectBGroup6[27]	Select sample phase channel 0 B
EportRX				4	paPhaseSelectBGroup6[28]	Select sample phase channel 1 B
EportRX				5	paPhaseSelectBGroup6[29]	Select sample phase channel 1 B
EportRX				6	paPhaseSelectBGroup6[30]	Select sample phase channel 1 B
EportRX				7	paPhaseSelectBGroup6[31]	Select sample phase channel 1 B
EportRX		218	inEportCtr156	0	paPhaseSelectCGroup6[0]	Select sample phase channel 6 C
EportRX				1	paPhaseSelectCGroup6[1]	Select sample phase channel 6 C
EportRX				2	paPhaseSelectCGroup6[2]	Select sample phase channel 6 C
EportRX				3	paPhaseSelectCGroup6[3]	Select sample phase channel 6 C
EportRX				4	paPhaseSelectCGroup6[4]	Select sample phase channel 7 C
EportRX				5	paPhaseSelectCGroup6[5]	Select sample phase channel 7 C
EportRX				6	paPhaseSelectCGroup6[6]	Select sample phase channel 7 C
EportRX				7	paPhaseSelectCGroup6[7]	Select sample phase channel 7 C
EportRX		219	inEportCtr157	0	paPhaseSelectCGroup6[8]	Select sample phase channel 4 C
EportRX				1	paPhaseSelectCGroup6[9]	Select sample phase channel 4 C

EportRX				2	paPhaseSelectCGroup6[10]	Select sample phase channel 4 C
EportRX				3	paPhaseSelectCGroup6[11]	Select sample phase channel 4 C
EportRX				4	paPhaseSelectCGroup6[12]	Select sample phase channel 5 C
EportRX				5	paPhaseSelectCGroup6[13]	Select sample phase channel 5 C
EportRX				6	paPhaseSelectCGroup6[14]	Select sample phase channel 5 C
EportRX				7	paPhaseSelectCGroup6[15]	Select sample phase channel 5 C
EportRX		220	inEportCtr158	0	paPhaseSelectCGroup6[16]	Select sample phase channel 2 C
EportRX				1	paPhaseSelectCGroup6[17]	Select sample phase channel 2 C
EportRX				2	paPhaseSelectCGroup6[18]	Select sample phase channel 2 C
EportRX				3	paPhaseSelectCGroup6[19]	Select sample phase channel 2 C
EportRX				4	paPhaseSelectCGroup6[20]	Select sample phase channel 3 C
EportRX				5	paPhaseSelectCGroup6[21]	Select sample phase channel 3 C
EportRX				6	paPhaseSelectCGroup6[22]	Select sample phase channel 3 C
EportRX				7	paPhaseSelectCGroup6[23]	Select sample phase channel 3 C
EportRX		221	inEportCtr159	0	paPhaseSelectCGroup6[24]	Select sample phase channel 0 C
EportRX				1	paPhaseSelectCGroup6[25]	Select sample phase channel 0 C
EportRX				2	paPhaseSelectCGroup6[26]	Select sample phase channel 0 C
EportRX				3	paPhaseSelectCGroup6[27]	Select sample phase channel 0 C
EportRX				4	paPhaseSelectCGroup6[28]	Select sample phase channel 1 C
EportRX				5	paPhaseSelectCGroup6[29]	Select sample phase channel 1 C
EportRX				6	paPhaseSelectCGroup6[30]	Select sample phase channel 1 C
EportRX				7	paPhaseSelectCGroup6[31]	Select sample phase channel 1 C
EportRX		222	inEportCtr160	0	paTrainAGroup6[0]	Train channel 0 A
EportRX				1	paTrainAGroup6[1]	Train channel 1 A
EportRX				2	paTrainAGroup6[2]	Train channel 2 A
EportRX				3	paTrainAGroup6[3]	Train channel 3 A
EportRX				4	paTrainAGroup6[4]	Train channel 4 A
EportRX				5	paTrainAGroup6[5]	Train channel 5 A
EportRX				6	paTrainAGroup6[6]	Train channel 6 A
EportRX				7	paTrainAGroup6[7]	Train channel 7 A
EportRX		223	inEportCtr161	0	paTrainBGroup6[0]	Train channel 0 B
EportRX				1	paTrainBGroup6[1]	Train channel 1 B
EportRX				2	paTrainBGroup6[2]	Train channel 2 B
EportRX				3	paTrainBGroup6[3]	Train channel 3 B
EportRX				4	paTrainBGroup6[4]	Train channel 4 B
EportRX				5	paTrainBGroup6[5]	Train channel 5 B
EportRX				6	paTrainBGroup6[6]	Train channel 6 B
EportRX				7	paTrainBGroup6[7]	Train channel 7 B
EportRX		224	inEportCtr162	0	paTrainCGroup6[0]	Train channel 0 C
EportRX				1	paTrainCGroup6[1]	Train channel 1 C
EportRX				2	paTrainCGroup6[2]	Train channel 2 C
EportRX				3	paTrainCGroup6[3]	Train channel 3 C
EportRX				4	paTrainCGroup6[4]	Train channel 4 C
EportRX				5	paTrainCGroup6[5]	Train channel 5 C
EportRX				6	paTrainCGroup6[6]	Train channel 6 C
EportRX				7	paTrainCGroup6[7]	Train channel 7 C
EportRX		225	inEportCtr163	0	paEnableAGroup6[0]	Enable channel 0 A
EportRX				1	paEnableAGroup6[1]	Enable channel 1 A
EportRX				2	paEnableAGroup6[2]	Enable channel 2 A
EportRX				3	paEnableAGroup6[3]	Enable channel 3 A
EportRX				4	paEnableAGroup6[4]	Enable channel 4 A
EportRX				5	paEnableAGroup6[5]	Enable channel 5 A
EportRX				6	paEnableAGroup6[6]	Enable channel 6 A
EportRX				7	paEnableAGroup6[7]	Enable channel 7 A
EportRX		226	inEportCtr164	0	paEnableBGroup6[0]	Enable channel 0 B
EportRX				1	paEnableBGroup6[1]	Enable channel 1 B
EportRX				2	paEnableBGroup6[2]	Enable channel 2 B
EportRX				3	paEnableBGroup6[3]	Enable channel 3 B
EportRX				4	paEnableBGroup6[4]	Enable channel 4 B
EportRX				5	paEnableBGroup6[5]	Enable channel 5 B

EportRX				6	paEnableBGroup6[6]	Enable channel 6 B
EportRX				7	paEnableBGroup6[7]	Enable channel 7 B
EportRX		227	inEportCtr165	0	paEnableCGroup6[0]	Enable channel 0 C
EportRX				1	paEnableCGroup6[1]	Enable channel 1 C
EportRX				2	paEnableCGroup6[2]	Enable channel 2 C
EportRX				3	paEnableCGroup6[3]	Enable channel 3 C
EportRX				4	paEnableCGroup6[4]	Enable channel 4 C
EportRX				5	paEnableCGroup6[5]	Enable channel 5 C
EportRX				6	paEnableCGroup6[6]	Enable channel 6 C
EportRX				7	paEnableCGroup6[7]	Enable channel 7 C
EportRX		228	inEportCtr166	0	paResetAGroup6[0]	Reset channel 0 A
EportRX				1	paResetAGroup6[1]	Reset channel 1 A
EportRX				2	paResetAGroup6[2]	Reset channel 2 A
EportRX				3	paResetAGroup6[3]	Reset channel 3 A
EportRX				4	paResetAGroup6[4]	Reset channel 4 A
EportRX				5	paResetAGroup6[5]	Reset channel 5 A
EportRX				6	paResetAGroup6[6]	Reset channel 6 A
EportRX				7	paResetAGroup6[7]	Reset channel 7 A
EportRX		229	inEportCtr167	0	paResetBGroup6[0]	Reset channel 0 B
EportRX				1	paResetBGroup6[1]	Reset channel 1 B
EportRX				2	paResetBGroup6[2]	Reset channel 2 B
EportRX				3	paResetBGroup6[3]	Reset channel 3 B
EportRX				4	paResetBGroup6[4]	Reset channel 4 B
EportRX				5	paResetBGroup6[5]	Reset channel 5 B
EportRX				6	paResetBGroup6[6]	Reset channel 6 B
EportRX				7	paResetBGroup6[7]	Reset channel 7 B
EportRX		230	inEportCtr168	0	paResetCGroup6[0]	Reset channel 0 C
EportRX				1	paResetCGroup6[1]	Reset channel 1 C
EportRX				2	paResetCGroup6[2]	Reset channel 2 C
EportRX				3	paResetCGroup6[3]	Reset channel 3 C
EportRX				4	paResetCGroup6[4]	Reset channel 4 C
EportRX				5	paResetCGroup6[5]	Reset channel 5 C
EportRX				6	paResetCGroup6[6]	Reset channel 6 C
EportRX				7	paResetCGroup6[7]	Reset channel 7 C
Eport-EC		231	inEportCtr169	0	paDllConfigAEC[0]	Set EC Phase-aligner dll A
Eport-EC				1	paDllConfigAEC[1]	Set EC Phase-aligner dll A
Eport-EC				2	paDllConfigAEC[2]	Set EC Phase-aligner dll A
Eport-EC				3	paDllConfigAEC[3]	Set EC Phase-aligner dll A
Eport-EC				4	paDllConfigBEC[0]	Set EC Phase-aligner dll B
Eport-EC				5	paDllConfigBEC[1]	Set EC Phase-aligner dll B
Eport-EC				6	paDllConfigBEC[2]	Set EC Phase-aligner dll B
Eport-EC				7	paDllConfigBEC[3]	Set EC Phase-aligner dll B
Eport-EC		232	inEportCtr170	0	paDllConfigCEC[0]	Set EC Phase-aligner dll C
Eport-EC				1	paDllConfigCEC[1]	Set EC Phase-aligner dll C
Eport-EC				2	paDllConfigCEC[2]	Set EC Phase-aligner dll C
Eport-EC				3	paDllConfigCEC[3]	Set EC Phase-aligner dll C
Eport-EC				4	paDllResetAEC	Reset EC Phase-aligner dll A
Eport-EC				5	paDllResetBEC	Reset EC Phase-aligner dll B
Eport-EC				6	paDllResetCEC	Reset EC Phase-aligner dll C
				7		unused
Eport-EC		233	inEportCtr171	0	paPhaseSelectAEC[0]	Select sample phase EC channel A
Eport-EC				1	paPhaseSelectAEC[1]	Select sample phase EC channel A
Eport-EC				2	paPhaseSelectAEC[2]	Select sample phase EC channel A
Eport-EC				3	paPhaseSelectAEC[3]	Select sample phase EC channel A
EportRX				4	dllCoarseLockDetectionA	dll coarse lock detection A
EportRX				5	dllCoarseLockDetectionB	dll coarse lock detection B
EportRX				6	dllCoarseLockDetectionC	dll coarse lock detection C
				7		unused
EportRX-TX RX, TX logic		234	inEportCtr172	0	loopbackEA	enable loopbackE A

EportRX-TX RX, TX logic				1	loopbackEB	enable loopbackE B
EportRX-TX RX, TX logic				2	loopbackEC	enable loopbackE C
EportRX-TX				3	loopbackFA	enable loopbackF A
EportRX-TX				4	loopbackFB	enable loopbackF B
EportRX-TX				5	loopbackFC	enable loopbackF C
				6		unused
				7		unused
XPLL		235	inEportCtr173	0	xPllChargePumpCurrentA[0]	xPll Charge Pump Current A
XPLL				1	xPllChargePumpCurrentA[1]	xPll Charge Pump Current A
XPLL				2	xPllChargePumpCurrentA[2]	xPll Charge Pump Current A
XPLL				3	xPllChargePumpCurrentA[3]	xPll Charge Pump Current A
XPLL				4	xPllChargePumpCurrentB[0]	xPll Charge Pump Current B
XPLL				5	xPllChargePumpCurrentB[1]	xPll Charge Pump Current B
XPLL				6	xPllChargePumpCurrentB[2]	xPll Charge Pump Current B
XPLL				7	xPllChargePumpCurrentB[3]	xPll Charge Pump Current B
XPLL		236	inEportCtr174	0	xPllChargePumpCurrentC[0]	xPll Charge Pump Current C
XPLL				1	xPllChargePumpCurrentC[1]	xPll Charge Pump Current C
XPLL				2	xPllChargePumpCurrentC[2]	xPll Charge Pump Current C
XPLL				3	xPllChargePumpCurrentC[3]	xPll Charge Pump Current C
				4		unused
				5		unused
				6		unused
				7		unused
Eport-EC		237	inEportCtr175	0	paPhaseSelectBEC[0]	Select sample phase EC channel B
Eport-EC				1	paPhaseSelectBEC[1]	Select sample phase EC channel B
Eport-EC				2	paPhaseSelectBEC[2]	Select sample phase EC channel B
Eport-EC				3	paPhaseSelectBEC[3]	Select sample phase EC channel B
				4		unused
				5		unused
				6		unused
				7		unused
Efuses		238	fuseBlowAddressLSB	0	fuseBlowAddress[0]	address of fuse to blow
Efuses				1	fuseBlowAddress[1]	address of fuse to blow
Efuses				2	fuseBlowAddress[2]	address of fuse to blow
Efuses				3	fuseBlowAddress[3]	address of fuse to blow
Efuses				4	fuseBlowAddress[4]	address of fuse to blow
Efuses				5	fuseBlowAddress[5]	address of fuse to blow
Efuses				6	fuseBlowAddress[6]	address of fuse to blow
Efuses				7	fuseBlowAddress[7]	address of fuse to blow
Efuses		239	fuseBlowAddressMSB	0	fuseBlowAddress[8]	address of fuse to blow
Efuses				1	fuseBlowAddress[9]	address of fuse to blow
Efuses				2	fuseBlowAddress[10]	address of fuse to blow
Efuses				3	fuseBlowAddress[11]	address of fuse to blow
Efuses				4	fuseBlowAddress[12]	address of fuse to blow
Efuses				5	fuseBlowAddress[13]	address of fuse to blow

Efuses				6	fuseBlowAddress[14]	address of fuse to blow
Efuses				7	fuseBlowAddress[15]	address of fuse to blow
Efuses		240	fuseBlowData	0	fuseBlowData[0]	data to blow fuse
Efuses				1	fuseBlowData[1]	data to blow fuse
Efuses				2	fuseBlowData[2]	data to blow fuse
Efuses				3	fuseBlowData[3]	data to blow fuse
Efuses				4	fuseBlowData[4]	data to blow fuse
Efuses				5	fuseBlowData[5]	data to blow fuse
Efuses				6	fuseBlowData[6]	data to blow fuse
Efuses				7	fuseBlowData[7]	data to blow fuse
Eport-EC		241	inEportCtr179	0	paPhaseSelectCEC[0]	Select sample phase EC channel C
Eport-EC				1	paPhaseSelectCEC[1]	Select sample phase EC channel C
Eport-EC				2	paPhaseSelectCEC[2]	Select sample phase EC channel C
Eport-EC				3	paPhaseSelectCEC[3]	Select sample phase EC channel C
				4		unused
				5		unused
				6		unused
				7		unused
EPLL-RX		242	inEportCtr180	0	ePllRxReferenceFreqA[0]	Select EPLL-RX reference frequency A
EPLL-RX				1	ePllRxReferenceFreqA[1]	Select EPLL-RX reference frequency A
EPLL-RX				2	ePllRxReferenceFreqB[0]	Select EPLL-RX reference frequency B
EPLL-RX				3	ePllRxReferenceFreqB[1]	Select EPLL-RX reference frequency B
EPLL-RX				4	ePllRxReferenceFreqC[0]	Select EPLL-RX reference frequency C
EPLL-RX				5	ePllRxReferenceFreqC[1]	Select EPLL-RX reference frequency C
				6		unused
				7		unused
EPLL-TX		243	inEportCtr181	0	ePllTxReferenceFreqA[0]	Select EPLL-TX reference frequency A
EPLL-TX				1	ePllTxReferenceFreqA[1]	Select EPLL-TX reference frequency A
EPLL-TX				2	ePllTxReferenceFreqB[0]	Select EPLL-TX reference frequency B
EPLL-TX				3	ePllTxReferenceFreqB[1]	Select EPLL-TX reference frequency B
EPLL-TX				4	ePllTxReferenceFreqC[0]	Select EPLL-TX reference frequency C
EPLL-TX				5	ePllTxReferenceFreqC[1]	Select EPLL-TX reference frequency C
				6		unused
				7		unused
RX logic		244	inEportCtr182	0	RXselectPosEdgeA	select +ve clock edge in RX synchroniser A
RX logic				1	RXselectPosEdgeB	select +ve clock edge in RX synchroniser B
RX logic				2	RXselectPosEdgeC	select +ve clock edge in RX synchroniser C
TX logic				3	TXselectPosEdgeA	select +ve clock edge in TX synchroniser A
TX logic				4	TXselectPosEdgeB	select +ve clock edge in TX synchroniser B
TX logic				5	TXselectPosEdgeC	select +ve clock edge in TX synchroniser C
				6		unused
				7		unused
Eport-EC		245	inEportCtr183	0	paTrainAEC	Train EC channel A
Eport-EC				1	paTrainBEC	Train EC channel B
Eport-EC				2	paTrainCEC	Train EC channel C
				3		unused
				4		unused
				5		unused
				6		unused
				7		unused
EportTX		246	inEportCtr184	0	bypassEportTX0A	bypass Eport TX group0 A
EportTX				1	bypassEportTX0B	bypass Eport TX group0 B
EportTX				2	bypassEportTX0C	bypass Eport TX group0 C
EportTX				3	bypassEportTX1A	bypass Eport TX group1 A

EportTX				4	bypassEportTX1B	bypass Eport TX group1 B
EportTX				5	bypassEportTX1C	bypass Eport TX group1 C
EportTX				6	bypassEportTX2A	bypass Eport TX group2 A
EportTX				7	bypassEportTX2B	bypass Eport TX group2 B
EportTX		247	inEportCtr185	0	bypassEportTX2C	bypass Eport TX group2 C
EportTX				1	bypassEportTX3A	bypass Eport TX group3 A
EportTX				2	bypassEportTX3B	bypass Eport TX group3 B
EportTX				3	bypassEportTX3C	bypass Eport TX group3 C
EportTX				4	bypassEportTX4A	bypass Eport TX group4 A
EportTX				5	bypassEportTX4B	bypass Eport TX group4 B
EportTX				6	bypassEportTX4C	bypass Eport TX group4 C
EportTX				7		unused
Eport-EC		248	inEportCtr186	0	paEnableAEC	Enable EC channel A
Eport-EC				1	paEnableBEC	Enable EC channel B
Eport-EC				2	paEnableCEC	Enable EC channel C
				3		unused
				4		unused
				5		unused
				6		unused
				7		unused
EportTX		249	inEportCtr187	0	selHiByteTX0A	select HiByte in bypassed Eport TX group0 A
EportTX				1	selHiByteTX0B	select HiByte in bypassed Eport TX group0 B
EportTX				2	selHiByteTX0C	select HiByte in bypassed Eport TX group0 C
EportTX				3	selHiByteTX1A	select HiByte in bypassed Eport TX group1 A
EportTX				4	selHiByteTX1B	select HiByte in bypassed Eport TX group1 B
EportTX				5	selHiByteTX1C	select HiByte in bypassed Eport TX group1 C
EportTX				6	selHiByteTX2A	select HiByte in bypassed Eport TX group2 A
EportTX				7	selHiByteTX2B	select HiByte in bypassed Eport TX group2 B
EportTX		250	inEportCtr188	0	selHiByteTX2C	select HiByte in bypassed Eport TX group2 C
EportTX				1	selHiByteTX3A	select HiByte in bypassed Eport TX group3 A
EportTX				2	selHiByteTX3B	select HiByte in bypassed Eport TX group3 B
EportTX				3	selHiByteTX3C	select HiByte in bypassed Eport TX group3 C
EportTX				4	selHiByteTX4A	select HiByte in bypassed Eport TX group4 A
EportTX				5	selHiByteTX4B	select HiByte in bypassed Eport TX group4 B
EportTX				6	selHiByteTX4C	select HiByte in bypassed Eport TX group4 C
EportTX				7		unused
Eport-EC		251	inEportCtr189	0	paResetAEC	Reset EC channel A
Eport-EC				1	paResetBEC	Reset EC channel B
Eport-EC				2	paResetCEC	Reset EC channel C
				3		unused
				4		unused
				5		unused
				6		unused
				7		unused
EportRX		252	inEportCtr190	0	bypass EportRXA	bypass EportRX A
EportRX				1	bypass EportRXB	bypass EportRX B

EportRX				2	bypass EportRXC	bypass EportRX C
				3		unused
				4		unused
				5		unused
				6		unused
				7		unused
EportRX		320	inEportCtr192	0	enableTermination0[0]	enable Termination group0 channel 0
EportRX				1	enableTermination0[1]	enable Termination group0 channel 1
EportRX				2	enableTermination0[2]	enable Termination group0 channel 2
EportRX				3	enableTermination0[3]	enable Termination group0 channel 3
EportRX				4	enableTermination0[4]	enable Termination group0 channel 4
EportRX				5	enableTermination0[5]	enable Termination group0 channel 5
EportRX				6	enableTermination0[6]	enable Termination group0 channel 6
EportRX				7	enableTermination0[7]	enable Termination group0 channel 7
EportRX		321	inEportCtr193	0	enableTermination1[0]	enable Termination group1 channel 0
EportRX				1	enableTermination1[1]	enable Termination group1 channel 1
EportRX				2	enableTermination1[2]	enable Termination group1 channel 2
EportRX				3	enableTermination1[3]	enable Termination group1 channel 3
EportRX				4	enableTermination1[4]	enable Termination group1 channel 4
EportRX				5	enableTermination1[5]	enable Termination group1 channel 5
EportRX				6	enableTermination1[6]	enable Termination group1 channel 6
EportRX				7	enableTermination1[7]	enable Termination group1 channel 7
EportRX		322	inEportCtr194	0	enableTermination2[0]	enable Termination group2 channel 0
EportRX				1	enableTermination2[1]	enable Termination group2 channel 1
EportRX				2	enableTermination2[2]	enable Termination group2 channel 2
EportRX				3	enableTermination2[3]	enable Termination group2 channel 3
EportRX				4	enableTermination2[4]	enable Termination group2 channel 4
EportRX				5	enableTermination2[5]	enable Termination group2 channel 5
EportRX				6	enableTermination2[6]	enable Termination group2 channel 6
EportRX				7	enableTermination2[7]	enable Termination group2 channel 7
EportRX		323	inEportCtr195	0	enableTermination3[0]	enable Termination group3 channel 0
EportRX				1	enableTermination3[1]	enable Termination group3 channel 1
EportRX				2	enableTermination3[2]	enable Termination group3 channel 2
EportRX				3	enableTermination3[3]	enable Termination group3 channel 3
EportRX				4	enableTermination3[4]	enable Termination group3 channel 4
EportRX				5	enableTermination3[5]	enable Termination group3 channel 5
EportRX				6	enableTermination3[6]	enable Termination group3 channel 6
EportRX				7	enableTermination3[7]	enable Termination group3 channel 7
EportRX		324	inEportCtr196	0	enableTermination4[0]	enable Termination group4 channel 0
EportRX				1	enableTermination4[1]	enable Termination group4 channel 1
EportRX				2	enableTermination4[2]	enable Termination group4 channel 2
EportRX				3	enableTermination4[3]	enable Termination group4 channel 3
EportRX				4	enableTermination4[4]	enable Termination group4 channel 4
EportRX				5	enableTermination4[5]	enable Termination group4 channel 5
EportRX				6	enableTermination4[6]	enable Termination group4 channel 6
EportRX				7	enableTermination4[7]	enable Termination group4 channel 7
EportRX		325	inEportCtr197	0	enableTermination5[0]	enable Termination group5 channel 0
EportRX				1	enableTermination5[1]	enable Termination group5 channel 1
EportRX				2	enableTermination5[2]	enable Termination group5 channel 2
EportRX				3	enableTermination5[3]	enable Termination group5 channel 3
EportRX				4	enableTermination5[4]	enable Termination group5 channel 4
EportRX				5	enableTermination5[5]	enable Termination group5 channel 5
EportRX				6	enableTermination5[6]	enable Termination group5 channel 6
EportRX				7	enableTermination5[7]	enable Termination group5 channel 7
EportRX		326	inEportCtr198	0	enableTermination6[0]	enable Termination group6 channel 0
EportRX				1	enableTermination6[1]	enable Termination group6 channel 1
EportRX				2	enableTermination6[2]	enable Termination group6 channel 2
EportRX				3	enableTermination6[3]	enable Termination group6 channel 3
EportRX				4	enableTermination6[4]	enable Termination group6 channel 4
EportRX				5	enableTermination6[5]	enable Termination group6 channel 5

EportRX				6	enableTermination6[6]	enable Termination group6 channel 6
EportRX				7	enableTermination6[7]	enable Termination group6 channel 7

Ttt

Block	Manual reference	Memory address	Register Name	bits	Signal name	Function
EportTX		254	outEportCtr0	0	modeGroupA0[0]	mode group0 A
EportTX				1	modeGroupA0[1]	mode group0 A
EportTX				2	clockBusFrequencyA0[0]	clock frequency group0 A
EportTX				3	clockBusFrequencyA0[1]	clock frequency group0 A
Eport-EC				4	modeEcA	mode EC channel A
Eport-EC				5	modeEcB	mode EC channel B
Eport-EC				6	modeEcC	mode EC channel C
				7		unused
EportTX		255	outEportCtr1	0	clockPortEnableGroupA0[0]	enable clock group0 channel 0 A
EportTX				1	clockPortEnableGroupA0[1]	enable clock group0 channel 1 A
EportTX				2	clockPortEnableGroupA0[2]	enable clock group0 channel 2 A
EportTX				3	clockPortEnableGroupA0[3]	enable clock group0 channel 3 A
EportTX				4	clockPortEnableGroupA0[4]	enable clock group0 channel 4 A
EportTX				5	clockPortEnableGroupA0[5]	enable clock group0 channel 5 A
EportTX				6	clockPortEnableGroupA0[6]	enable clock group0 channel 6 A
EportTX				7	clockPortEnableGroupA0[7]	enable clock group0 channel 7 A
EportTX		256	outEportCtr2	0	dataPortEnableGroupA0[0]	enable data group0 channel 0 A
EportTX				1	dataPortEnableGroupA0[1]	enable data group0 channel 1 A
EportTX				2	dataPortEnableGroupA0[2]	enable data group0 channel 2 A
EportTX				3	dataPortEnableGroupA0[3]	enable data group0 channel 3 A
EportTX				4	dataPortEnableGroupA0[4]	enable data group0 channel 4 A
EportTX				5	dataPortEnableGroupA0[5]	enable data group0 channel 5 A
EportTX				6	dataPortEnableGroupA0[6]	enable data group0 channel 6 A
EportTX				7	dataPortEnableGroupA0[7]	enable data group0 channel 7 A
EportTX		257	outEportCtr3	0	modeGroupA1[0]	mode group1 A
EportTX				1	modeGroupA1[1]	mode group1 A
EportTX				2	clockBusFrequencyA1[0]	clock frequency group1 A
EportTX				3	clockBusFrequencyA1[1]	clock frequency group1 A
Eport-EC				4	clockBusFrequencyEcA	clock frequency EC channel A
Eport-EC				5	clockBusFrequencyEcB	clock frequency EC channel B
Eport-EC				6	clockBusFrequencyEcC	clock frequency EC channel C
				7		unused
EportTX		258	outEportCtr4	0	clockPortEnableGroupA1[0]	enable clock group1 channel 0 A
EportTX				1	clockPortEnableGroupA1[1]	enable clock group1 channel 1 A
EportTX				2	clockPortEnableGroupA1[2]	enable clock group1 channel 2 A
EportTX				3	clockPortEnableGroupA1[3]	enable clock group1 channel 3 A
EportTX				4	clockPortEnableGroupA1[4]	enable clock group1 channel 4 A
EportTX				5	clockPortEnableGroupA1[5]	enable clock group1 channel 5 A
EportTX				6	clockPortEnableGroupA1[6]	enable clock group1 channel 6 A
EportTX				7	clockPortEnableGroupA1[7]	enable clock group1 channel 7 A
EportTX		259	outEportCtr5	0	dataPortEnableGroupA1[0]	enable data group1 channel 0 A
EportTX				1	dataPortEnableGroupA1[1]	enable data group1 channel 1 A
EportTX				2	dataPortEnableGroupA1[2]	enable data group1 channel 2 A
EportTX				3	dataPortEnableGroupA1[3]	enable data group1 channel 3 A
EportTX				4	dataPortEnableGroupA1[4]	enable data group1 channel 4 A
EportTX				5	dataPortEnableGroupA1[5]	enable data group1 channel 5 A
EportTX				6	dataPortEnableGroupA1[6]	enable data group1 channel 6 A
EportTX				7	dataPortEnableGroupA1[7]	enable data group1 channel 7 A
EportTX		260	outEportCtr6	0	modeGroupA2[0]	mode group2 A
EportTX				1	modeGroupA2[1]	mode group2 A
EportTX				2	clockBusFrequencyA2[0]	clock frequency group2 A
EportTX				3	clockBusFrequencyA2[1]	clock frequency group2 A
				4		unused
				5		unused
				6		unused
				7		unused
EportTX		261	outEportCtr7	0	clockPortEnableGroupA2[0]	enable clock group2 channel 0 A
EportTX				1	clockPortEnableGroupA2[1]	enable clock group2 channel 1 A

EportTX				2	clockPortEnableGroupA2[2]	enable clock group2 channel 2 A
EportTX				3	clockPortEnableGroupA2[3]	enable clock group2 channel 3 A
EportTX				4	clockPortEnableGroupA2[4]	enable clock group2 channel 4 A
EportTX				5	clockPortEnableGroupA2[5]	enable clock group2 channel 5 A
EportTX				6	clockPortEnableGroupA2[6]	enable clock group2 channel 6 A
EportTX				7	clockPortEnableGroupA2[7]	enable clock group2 channel 7 A
EportTX		262	outEportCtr8	0	dataPortEnableGroupA2[0]	enable data group2 channel 0 A
EportTX				1	dataPortEnableGroupA2[1]	enable data group2 channel 1 A
EportTX				2	dataPortEnableGroupA2[2]	enable data group2 channel 2 A
EportTX				3	dataPortEnableGroupA2[3]	enable data group2 channel 3 A
EportTX				4	dataPortEnableGroupA2[4]	enable data group2 channel 4 A
EportTX				5	dataPortEnableGroupA2[5]	enable data group2 channel 5 A
EportTX				6	dataPortEnableGroupA2[6]	enable data group2 channel 6 A
EportTX				7	dataPortEnableGroupA2[7]	enable data group2 channel 7 A
EportTX		263	outEportCtr9	0	modeGroupA3[0]	mode group3 A
EportTX				1	modeGroupA3[1]	mode group3 A
EportTX				2	clockBusFrequencyA3[0]	clock frequency group3 A
EportTX				3	clockBusFrequencyA3[1]	clock frequency group3 A
				4		unused
				5		unused
				6		unused
				7		unused
EportTX		264	outEportCtr10	0	clockPortEnableGroupA3[0]	enable clock group3 channel 0 A
EportTX				1	clockPortEnableGroupA3[1]	enable clock group3 channel 1 A
EportTX				2	clockPortEnableGroupA3[2]	enable clock group3 channel 2 A
EportTX				3	clockPortEnableGroupA3[3]	enable clock group3 channel 3 A
EportTX				4	clockPortEnableGroupA3[4]	enable clock group3 channel 4 A
EportTX				5	clockPortEnableGroupA3[5]	enable clock group3 channel 5 A
EportTX				6	clockPortEnableGroupA3[6]	enable clock group3 channel 6 A
EportTX				7	clockPortEnableGroupA3[7]	enable clock group3 channel 7 A
EportTX		265	outEportCtr11	0	dataPortEnableGroupA3[0]	enable data group3 channel 0 A
EportTX				1	dataPortEnableGroupA3[1]	enable data group3 channel 1 A
EportTX				2	dataPortEnableGroupA3[2]	enable data group3 channel 2 A
EportTX				3	dataPortEnableGroupA3[3]	enable data group3 channel 3 A
EportTX				4	dataPortEnableGroupA3[4]	enable data group3 channel 4 A
EportTX				5	dataPortEnableGroupA3[5]	enable data group3 channel 5 A
EportTX				6	dataPortEnableGroupA3[6]	enable data group3 channel 6 A
EportTX				7	dataPortEnableGroupA3[7]	enable data group3 channel 7 A
EportTX		266	outEportCtr12	0	modeGroupA4[0]	mode group4 A
EportTX				1	modeGroupA4[1]	mode group4 A
EportTX				2	clockBusFrequencyA4[0]	clock frequency group4 A
EportTX				3	clockBusFrequencyA4[1]	clock frequency group4 A
				4		unused
				5		unused
				6		unused
				7		unused
EportTX		267	outEportCtr13	0	clockPortEnableGroupA4[0]	enable clock group4 channel 0 A
EportTX				1	clockPortEnableGroupA4[1]	enable clock group4 channel 1 A
EportTX				2	clockPortEnableGroupA4[2]	enable clock group4 channel 2 A
EportTX				3	clockPortEnableGroupA4[3]	enable clock group4 channel 3 A
EportTX				4	clockPortEnableGroupA4[4]	enable clock group4 channel 4 A
EportTX				5	clockPortEnableGroupA4[5]	enable clock group4 channel 5 A
EportTX				6	clockPortEnableGroupA4[6]	enable clock group4 channel 6 A
EportTX				7	clockPortEnableGroupA4[7]	enable clock group4 channel 7 A
EportTX		268	outEportCtr14	0	dataPortEnableGroupA4[0]	enable data group4 channel 0 A
EportTX				1	dataPortEnableGroupA4[1]	enable data group4 channel 1 A
EportTX				2	dataPortEnableGroupA4[2]	enable data group4 channel 2 A
EportTX				3	dataPortEnableGroupA4[3]	enable data group4 channel 3 A
EportTX				4	dataPortEnableGroupA4[4]	enable data group4 channel 4 A
EportTX				5	dataPortEnableGroupA4[5]	enable data group4 channel 5 A

EportTX				6	dataPortEnableGroupA4[6]	enable data group4 channel 6 A
EportTX				7	dataPortEnableGroupA4[7]	enable data group4 channel 7 A
EportTX		327	outEportCtr15	0	driveStrength0[0]	set drive strength group 0
EportTX				1	driveStrength0[1]	set drive strength group 0
EportTX				2	driveStrength0[2]	set drive strength group 0
EportTX				3	driveStrength0[3]	set drive strength group 0
EportTX				4	driveStrength1[0]	set drive strength group 1
EportTX				5	driveStrength1[1]	set drive strength group 1
EportTX				6	driveStrength1[2]	set drive strength group 1
EportTX				7	driveStrength1[3]	set drive strength group 1
EportTX		328	outEportCtr16	0	driveStrength2[0]	set drive strength group 2
EportTX				1	driveStrength2[1]	set drive strength group 2
EportTX				2	driveStrength2[2]	set drive strength group 2
EportTX				3	driveStrength2[3]	set drive strength group 2
EportTX				4	driveStrength3[0]	set drive strength group 3
EportTX				5	driveStrength3[1]	set drive strength group 3
EportTX				6	driveStrength3[2]	set drive strength group 3
EportTX				7	driveStrength3[3]	set drive strength group 3
EportTX		329	outEportCtr17	0	driveStrength4[0]	set drive strength group 4
EportTX				1	driveStrength4[1]	set drive strength group 4
EportTX				2	driveStrength4[2]	set drive strength group 4
EportTX				3	driveStrength4[3]	set drive strength group 4
EportTX				4	clkDriveStrength0[0]	set clock drive strength group 0
EportTX				5	clkDriveStrength0[1]	set clock drive strength group 0
EportTX				6	clkDriveStrength0[2]	set clock drive strength group 0
EportTX				7	clkDriveStrength0[3]	set clock drive strength group 0
EportTX		330	outEportCtr18	0	clkDriveStrength1[0]	set clock drive strength group 1
EportTX				1	clkDriveStrength1[1]	set clock drive strength group 1
EportTX				2	clkDriveStrength1[2]	set clock drive strength group 1
EportTX				3	clkDriveStrength1[3]	set clock drive strength group 1
EportTX				4	clkDriveStrength2[0]	set clock drive strength group 2
EportTX				5	clkDriveStrength2[1]	set clock drive strength group 2
EportTX				6	clkDriveStrength2[2]	set clock drive strength group 2
EportTX				7	clkDriveStrength2[3]	set clock drive strength group 2
EportTX		331	outEportCtr19	0	clkDriveStrength3[0]	set clock drive strength group 3
EportTX				1	clkDriveStrength3[1]	set clock drive strength group 3
EportTX				2	clkDriveStrength3[2]	set clock drive strength group 3
EportTX				3	clkDriveStrength3[3]	set clock drive strength group 3
EportTX				4	clkDriveStrength4[0]	set clock drive strength group 4
EportTX				5	clkDriveStrength4[1]	set clock drive strength group 4
EportTX				6	clkDriveStrength4[2]	set clock drive strength group 4
EportTX				7	clkDriveStrength4[3]	set clock drive strength group 4
EportTX		332	outEportCtr20	0	modeGroupB0[0]	mode group0 B
EportTX				1	modeGroupB0[1]	mode group0 B
EportTX				2	clockBusFrequencyB0[0]	clock frequency group0 B
EportTX				3	clockBusFrequencyB0[1]	clock frequency group0 B
				4		unused
				5		unused
				6		unused
				7		unused
EportTX		333	outEportCtr21	0	clockPortEnableGroupB0[0]	enable clock group0 channel 0 B
EportTX				1	clockPortEnableGroupB0[1]	enable clock group0 channel 1 B
EportTX				2	clockPortEnableGroupB0[2]	enable clock group0 channel 2 B
EportTX				3	clockPortEnableGroupB0[3]	enable clock group0 channel 3 B
EportTX				4	clockPortEnableGroupB0[4]	enable clock group0 channel 4 B
EportTX				5	clockPortEnableGroupB0[5]	enable clock group0 channel 5 B
EportTX				6	clockPortEnableGroupB0[6]	enable clock group0 channel 6 B
EportTX				7	clockPortEnableGroupB0[7]	enable clock group0 channel 7 B
EportTX		334	outEportCtr22	0	dataPortEnableGroupB0[0]	enable data group0 channel 0 B
EportTX				1	dataPortEnableGroupB0[1]	enable data group0 channel 1 B

EportTX				2	dataPortEnableGroupB0[2]	enable data group0 channel 2 B
EportTX				3	dataPortEnableGroupB0[3]	enable data group0 channel 3 B
EportTX				4	dataPortEnableGroupB0[4]	enable data group0 channel 4 B
EportTX				5	dataPortEnableGroupB0[5]	enable data group0 channel 5 B
EportTX				6	dataPortEnableGroupB0[6]	enable data group0 channel 6 B
EportTX				7	dataPortEnableGroupB0[7]	enable data group0 channel 7 B
EportTX		335	outEportCtr23	0	modeGroupB1[0]	mode group1 B
EportTX				1	modeGroupB1[1]	mode group1 B
EportTX				2	clockBusFrequencyB1[0]	clock frequency group1 B
EportTX				3	clockBusFrequencyB1[1]	clock frequency group1 B
				4		unused
				5		unused
				6		unused
				7		unused
EportTX		336	outEportCtr24	0	clockPortEnableGroupB1[0]	enable clock group1 channel 0 B
EportTX				1	clockPortEnableGroupB1[1]	enable clock group1 channel 1 B
EportTX				2	clockPortEnableGroupB1[2]	enable clock group1 channel 2 B
EportTX				3	clockPortEnableGroupB1[3]	enable clock group1 channel 3 B
EportTX				4	clockPortEnableGroupB1[4]	enable clock group1 channel 4 B
EportTX				5	clockPortEnableGroupB1[5]	enable clock group1 channel 5 B
EportTX				6	clockPortEnableGroupB1[6]	enable clock group1 channel 6 B
EportTX				7	clockPortEnableGroupB1[7]	enable clock group1 channel 7 B
EportTX		337	outEportCtr25	0	dataPortEnableGroupB1[0]	enable data group1 channel 0 B
EportTX				1	dataPortEnableGroupB1[1]	enable data group1 channel 1 B
EportTX				2	dataPortEnableGroupB1[2]	enable data group1 channel 2 B
EportTX				3	dataPortEnableGroupB1[3]	enable data group1 channel 3 B
EportTX				4	dataPortEnableGroupB1[4]	enable data group1 channel 4 B
EportTX				5	dataPortEnableGroupB1[5]	enable data group1 channel 5 B
EportTX				6	dataPortEnableGroupB1[6]	enable data group1 channel 6 B
EportTX				7	dataPortEnableGroupB1[7]	enable data group1 channel 7 B
EportTX		338	outEportCtr26	0	modeGroupB2[0]	mode group2 B
EportTX				1	modeGroupB2[1]	mode group2 B
EportTX				2	clockBusFrequencyB2[0]	clock frequency group2 B
EportTX				3	clockBusFrequencyB2[1]	clock frequency group2 B
				4		unused
				5		unused
				6		unused
				7		unused
EportTX		339	outEportCtr27	0	clockPortEnableGroupB2[0]	enable clock group2 channel 0 B
EportTX				1	clockPortEnableGroupB2[1]	enable clock group2 channel 1 B
EportTX				2	clockPortEnableGroupB2[2]	enable clock group2 channel 2 B
EportTX				3	clockPortEnableGroupB2[3]	enable clock group2 channel 3 B
EportTX				4	clockPortEnableGroupB2[4]	enable clock group2 channel 4 B
EportTX				5	clockPortEnableGroupB2[5]	enable clock group2 channel 5 B
EportTX				6	clockPortEnableGroupB2[6]	enable clock group2 channel 6 B
EportTX				7	clockPortEnableGroupB2[7]	enable clock group2 channel 7 B
EportTX		340	outEportCtr28	0	dataPortEnableGroupB2[0]	enable data group2 channel 0 B
EportTX				1	dataPortEnableGroupB2[1]	enable data group2 channel 1 B
EportTX				2	dataPortEnableGroupB2[2]	enable data group2 channel 2 B
EportTX				3	dataPortEnableGroupB2[3]	enable data group2 channel 3 B
EportTX				4	dataPortEnableGroupB2[4]	enable data group2 channel 4 B
EportTX				5	dataPortEnableGroupB2[5]	enable data group2 channel 5 B
EportTX				6	dataPortEnableGroupB2[6]	enable data group2 channel 6 B
EportTX				7	dataPortEnableGroupB2[7]	enable data group2 channel 7 B
EportTX		341	outEportCtr29	0	modeGroupB3[0]	mode group3 B
EportTX				1	modeGroupB3[1]	mode group3 B
EportTX				2	clockBusFrequencyB3[0]	clock frequency group3 B
EportTX				3	clockBusFrequencyB3[1]	clock frequency group3 B
				4		unused
				5		unused

				6		unused
				7		unused
EportTX		342	outEportCtr30	0	clockPortEnableGroupB3[0]	enable clock group3 channel 0 B
EportTX				1	clockPortEnableGroupB3[1]	enable clock group3 channel 1 B
EportTX				2	clockPortEnableGroupB3[2]	enable clock group3 channel 2 B
EportTX				3	clockPortEnableGroupB3[3]	enable clock group3 channel 3 B
EportTX				4	clockPortEnableGroupB3[4]	enable clock group3 channel 4 B
EportTX				5	clockPortEnableGroupB3[5]	enable clock group3 channel 5 B
EportTX				6	clockPortEnableGroupB3[6]	enable clock group3 channel 6 B
EportTX				7	clockPortEnableGroupB3[7]	enable clock group3 channel 7 B
EportTX		343	outEportCtr31	0	dataPortEnableGroupB3[0]	enable data group3 channel 0 B
EportTX				1	dataPortEnableGroupB3[1]	enable data group3 channel 1 B
EportTX				2	dataPortEnableGroupB3[2]	enable data group3 channel 2 B
EportTX				3	dataPortEnableGroupB3[3]	enable data group3 channel 3 B
EportTX				4	dataPortEnableGroupB3[4]	enable data group3 channel 4 B
EportTX				5	dataPortEnableGroupB3[5]	enable data group3 channel 5 B
EportTX				6	dataPortEnableGroupB3[6]	enable data group3 channel 6 B
EportTX				7	dataPortEnableGroupB3[7]	enable data group3 channel 7 B
EportTX		344	outEportCtr32	0	modeGroupB4[0]	mode group4 B
EportTX				1	modeGroupB4[1]	mode group4 B
EportTX				2	clockBusFrequencyB4[0]	clock frequency group4 B
EportTX				3	clockBusFrequencyB4[1]	clock frequency group4 B
				4		unused
				5		unused
				6		unused
				7		unused
EportTX		345	outEportCtr33	0	clockPortEnableGroupB4[0]	enable clock group4 channel 0 B
EportTX				1	clockPortEnableGroupB4[1]	enable clock group4 channel 1 B
EportTX				2	clockPortEnableGroupB4[2]	enable clock group4 channel 2 B
EportTX				3	clockPortEnableGroupB4[3]	enable clock group4 channel 3 B
EportTX				4	clockPortEnableGroupB4[4]	enable clock group4 channel 4 B
EportTX				5	clockPortEnableGroupB4[5]	enable clock group4 channel 5 B
EportTX				6	clockPortEnableGroupB4[6]	enable clock group4 channel 6 B
EportTX				7	clockPortEnableGroupB4[7]	enable clock group4 channel 7 B
EportTX		346	outEportCtr34	0	dataPortEnableGroupB4[0]	enable data group4 channel 0 B
EportTX				1	dataPortEnableGroupB4[1]	enable data group4 channel 1 B
EportTX				2	dataPortEnableGroupB4[2]	enable data group4 channel 2 B
EportTX				3	dataPortEnableGroupB4[3]	enable data group4 channel 3 B
EportTX				4	dataPortEnableGroupB4[4]	enable data group4 channel 4 B
EportTX				5	dataPortEnableGroupB4[5]	enable data group4 channel 5 B
EportTX				6	dataPortEnableGroupB4[6]	enable data group4 channel 6 B
EportTX				7	dataPortEnableGroupB4[7]	enable data group4 channel 7 B
EportTX		347	outEportCtr35	0	modeGroupC0[0]	mode group0 C
EportTX				1	modeGroupC0[1]	mode group0 C
EportTX				2	clockBusFrequencyC0[0]	clock frequency group0 C
EportTX				3	clockBusFrequencyC0[1]	clock frequency group0 C
				4		unused
				5		unused
				6		unused
				7		unused
EportTX		348	outEportCtr36	0	clockPortEnableGroupC0[0]	enable clock group0 channel 0 C
EportTX				1	clockPortEnableGroupC0[1]	enable clock group0 channel 1 C
EportTX				2	clockPortEnableGroupC0[2]	enable clock group0 channel 2 C
EportTX				3	clockPortEnableGroupC0[3]	enable clock group0 channel 3 C
EportTX				4	clockPortEnableGroupC0[4]	enable clock group0 channel 4 C
EportTX				5	clockPortEnableGroupC0[5]	enable clock group0 channel 5 C
EportTX				6	clockPortEnableGroupC0[6]	enable clock group0 channel 6 C
EportTX				7	clockPortEnableGroupC0[7]	enable clock group0 channel 7 C
EportTX		349	outEportCtr37	0	dataPortEnableGroupC0[0]	enable data group0 channel 0 C
EportTX				1	dataPortEnableGroupC0[1]	enable data group0 channel 1 C

EportTX				2	dataPortEnableGroupC0[2]	enable data group0 channel 2 C
EportTX				3	dataPortEnableGroupC0[3]	enable data group0 channel 3 C
EportTX				4	dataPortEnableGroupC0[4]	enable data group0 channel 4 C
EportTX				5	dataPortEnableGroupC0[5]	enable data group0 channel 5 C
EportTX				6	dataPortEnableGroupC0[6]	enable data group0 channel 6 C
EportTX				7	dataPortEnableGroupC0[7]	enable data group0 channel 7 C
EportTX		350	outEportCtr38	0	modeGroupC1[0]	mode group1 C
EportTX				1	modeGroupC1[1]	mode group1 C
EportTX				2	clockBusFrequencyC1[0]	clock frequency group1 C
EportTX				3	clockBusFrequencyC1[1]	clock frequency group1 C
				4		unused
				5		unused
				6		unused
				7		unused
EportTX		351	outEportCtr39	0	clockPortEnableGroupC1[0]	enable clock group1 channel 0 C
EportTX				1	clockPortEnableGroupC1[1]	enable clock group1 channel 1 C
EportTX				2	clockPortEnableGroupC1[2]	enable clock group1 channel 2 C
EportTX				3	clockPortEnableGroupC1[3]	enable clock group1 channel 3 C
EportTX				4	clockPortEnableGroupC1[4]	enable clock group1 channel 4 C
EportTX				5	clockPortEnableGroupC1[5]	enable clock group1 channel 5 C
EportTX				6	clockPortEnableGroupC1[6]	enable clock group1 channel 6 C
EportTX				7	clockPortEnableGroupC1[7]	enable clock group1 channel 7 C
EportTX		352	outEportCtr40	0	dataPortEnableGroupC1[0]	enable data group1 channel 0 C
EportTX				1	dataPortEnableGroupC1[1]	enable data group1 channel 1 C
EportTX				2	dataPortEnableGroupC1[2]	enable data group1 channel 2 C
EportTX				3	dataPortEnableGroupC1[3]	enable data group1 channel 3 C
EportTX				4	dataPortEnableGroupC1[4]	enable data group1 channel 4 C
EportTX				5	dataPortEnableGroupC1[5]	enable data group1 channel 5 C
EportTX				6	dataPortEnableGroupC1[6]	enable data group1 channel 6 C
EportTX				7	dataPortEnableGroupC1[7]	enable data group1 channel 7 C
EportTX		353	outEportCtr41	0	modeGroupC2[0]	mode group2 C
EportTX				1	modeGroupC2[1]	mode group2 C
EportTX				2	clockBusFrequencyC2[0]	clock frequency group2 C
EportTX				3	clockBusFrequencyC2[1]	clock frequency group2 C
				4		unused
				5		unused
				6		unused
				7		unused
EportTX		354	outEportCtr42	0	clockPortEnableGroupC2[0]	enable clock group2 channel 0 C
EportTX				1	clockPortEnableGroupC2[1]	enable clock group2 channel 1 C
EportTX				2	clockPortEnableGroupC2[2]	enable clock group2 channel 2 C
EportTX				3	clockPortEnableGroupC2[3]	enable clock group2 channel 3 C
EportTX				4	clockPortEnableGroupC2[4]	enable clock group2 channel 4 C
EportTX				5	clockPortEnableGroupC2[5]	enable clock group2 channel 5 C
EportTX				6	clockPortEnableGroupC2[6]	enable clock group2 channel 6 C
EportTX				7	clockPortEnableGroupC2[7]	enable clock group2 channel 7 C
EportTX		355	outEportCtr43	0	dataPortEnableGroupC2[0]	enable data group2 channel 0 C
EportTX				1	dataPortEnableGroupC2[1]	enable data group2 channel 1 C
EportTX				2	dataPortEnableGroupC2[2]	enable data group2 channel 2 C
EportTX				3	dataPortEnableGroupC2[3]	enable data group2 channel 3 C
EportTX				4	dataPortEnableGroupC2[4]	enable data group2 channel 4 C
EportTX				5	dataPortEnableGroupC2[5]	enable data group2 channel 5 C
EportTX				6	dataPortEnableGroupC2[6]	enable data group2 channel 6 C
EportTX				7	dataPortEnableGroupC2[7]	enable data group2 channel 7 C
EportTX		356	outEportCtr44	0	modeGroupC3[0]	mode group3 C
EportTX				1	modeGroupC3[1]	mode group3 C
EportTX				2	clockBusFrequencyC3[0]	clock frequency group3 C
EportTX				3	clockBusFrequencyC3[1]	clock frequency group3 C
				4		unused
				5		unused

				6		unused
				7		unused
EportTX		357	outEportCtr45	0	clockPortEnableGroupC3[0]	enable clock group3 channel 0 C
EportTX				1	clockPortEnableGroupC3[1]	enable clock group3 channel 1 C
EportTX				2	clockPortEnableGroupC3[2]	enable clock group3 channel 2 C
EportTX				3	clockPortEnableGroupC3[3]	enable clock group3 channel 3 C
EportTX				4	clockPortEnableGroupC3[4]	enable clock group3 channel 4 C
EportTX				5	clockPortEnableGroupC3[5]	enable clock group3 channel 5 C
EportTX				6	clockPortEnableGroupC3[6]	enable clock group3 channel 6 C
EportTX				7	clockPortEnableGroupC3[7]	enable clock group3 channel 7 C
EportTX		358	outEportCtr46	0	dataPortEnableGroupC3[0]	enable data group3 channel 0 C
EportTX				1	dataPortEnableGroupC3[1]	enable data group3 channel 1 C
EportTX				2	dataPortEnableGroupC3[2]	enable data group3 channel 2 C
EportTX				3	dataPortEnableGroupC3[3]	enable data group3 channel 3 C
EportTX				4	dataPortEnableGroupC3[4]	enable data group3 channel 4 C
EportTX				5	dataPortEnableGroupC3[5]	enable data group3 channel 5 C
EportTX				6	dataPortEnableGroupC3[6]	enable data group3 channel 6 C
EportTX				7	dataPortEnableGroupC3[7]	enable data group3 channel 7 C
EportTX		359	outEportCtr47	0	modeGroupC4[0]	mode group4 C
EportTX				1	modeGroupC4[1]	mode group4 C
EportTX				2	clockBusFrequencyC4[0]	clock frequency group4 C
EportTX				3	clockBusFrequencyC4[1]	clock frequency group4 C
				4		unused
				5		unused
				6		unused
				7		unused
EportTX		360	outEportCtr48	0	clockPortEnableGroupC4[0]	enable clock group4 channel 0 C
EportTX				1	clockPortEnableGroupC4[1]	enable clock group4 channel 1 C
EportTX				2	clockPortEnableGroupC4[2]	enable clock group4 channel 2 C
EportTX				3	clockPortEnableGroupC4[3]	enable clock group4 channel 3 C
EportTX				4	clockPortEnableGroupC4[4]	enable clock group4 channel 4 C
EportTX				5	clockPortEnableGroupC4[5]	enable clock group4 channel 5 C
EportTX				6	clockPortEnableGroupC4[6]	enable clock group4 channel 6 C
EportTX				7	clockPortEnableGroupC4[7]	enable clock group4 channel 7 C
EportTX		361	outEportCtr49	0	dataPortEnableGroupC4[0]	enable data group4 channel 0 C
EportTX				1	dataPortEnableGroupC4[1]	enable data group4 channel 1 C
EportTX				2	dataPortEnableGroupC4[2]	enable data group4 channel 2 C
EportTX				3	dataPortEnableGroupC4[3]	enable data group4 channel 3 C
EportTX				4	dataPortEnableGroupC4[4]	enable data group4 channel 4 C
EportTX				5	dataPortEnableGroupC4[5]	enable data group4 channel 5 C
EportTX				6	dataPortEnableGroupC4[6]	enable data group4 channel 6 C
EportTX				7	dataPortEnableGroupC4[7]	enable data group4 channel 7 C

17.3 GBTX read-only registers

The GBTX contains a number of registers whose values can be read via the I2C interface or the SC-IC interface. These registers contain the status of internal blocks of the GBTX and can be used for diagnostic purposes. Each 8-bit register has a unique 16-bit address.
The table below lists the registers and their functions.

Table 154 GBTX read-only registers

Block	Manual reference	Memory address	Register Name	bits	Signal name	Function
Efuses		366		0	configFuseData[0]	config fuse value bit 0
Efuses				1	configFuseData[1]	config fuse value bit 1
Efuses				2	configFuseData[2]	config fuse value bit 2
Efuses				3	configFuseData[3]	config fuse value bit 3
Efuses				4	configFuseData[4]	config fuse value bit 4
Efuses				5	configFuseData[5]	config fuse value bit 5
Efuses				6	configFuseData[6]	config fuse value bit 6
Efuses				7	configFuseData[7]	config fuse value bit 7
Efuses		367		0	testFuseData1[0]	test-fuse 1 value bit 0
Efuses				1	testFuseData1[1]	test-fuse 1 value bit 1
Efuses				2	testFuseData1[2]	test-fuse 1 value bit 2
Efuses				3	testFuseData1[3]	test-fuse 1 value bit 3
Efuses				4	testFuseData1[4]	test-fuse 1 value bit 4
Efuses				5	testFuseData1[5]	test-fuse 1 value bit 5
Efuses				6	testFuseData1[6]	test-fuse 1 value bit 6
Efuses				7	testFuseData1[7]	test-fuse 1 value bit 7
Efuses		368		0	testFuseData2[0]	test-fuse 2 value bit 0
Efuses				1	testFuseData2[1]	test-fuse 2 value bit 1
Efuses				2	testFuseData2[2]	test-fuse 2 value bit 2
Efuses				3	testFuseData2[3]	test-fuse 2 value bit 3
Efuses				4	testFuseData2[4]	test-fuse 2 value bit 4
Efuses				5	testFuseData2[5]	test-fuse 2 value bit 5
Efuses				6	testFuseData2[6]	test-fuse 2 value bit 6
Efuses				7	testFuseData2[7]	test-fuse 2 value bit 7
RX logic		369		0	BERTCounter[0]	BERT counter value in RX logic bit 0
RX logic				1	BERTCounter[1]	BERT counter value in RX logic bit 1
RX logic				2	BERTCounter[2]	BERT counter value in RX logic bit 2
RX logic				3	BERTCounter[3]	BERT counter value in RX logic bit 3
RX logic				4	BERTCounter[4]	BERT counter value in RX logic bit 4
RX logic				5	BERTCounter[5]	BERT counter value in RX logic bit 5
RX logic				6	BERTCounter[6]	BERT counter value in RX logic bit 6
RX logic				7	BERTCounter[7]	BERT counter value in RX logic bit 7
RX logic		370		0	BERTCounter[8]	BERT counter value in RX logic bit 8
RX logic				1	BERTCounter[9]	BERT counter value in RX logic bit 9
RX logic				2	BERTCounter[10]	BERT counter value in RX logic bit 10
RX logic				3	BERTCounter[11]	BERT counter value in RX logic bit 11
RX logic				4	BERTCounter[12]	BERT counter value in RX logic bit 12
RX logic				5	BERTCounter[13]	BERT counter value in RX logic bit 13
RX logic				6	BERTCounter[14]	BERT counter value in RX logic bit 14
RX logic				7	BERTCounter[15]	BERT counter value in RX logic bit 15
SC-IC		371	scStatusA	0	scStatusA[0]	result of parity check of previous SC-IC write
SC-IC				1	scStatusA[1]	state of SC-IC state machine A bit 0
SC-IC				2	scStatusA[2]	state of SC-IC state machine A bit 1
SC-IC				3	scStatusA[3]	state of SC-IC state machine A bit 2
SC-IC				4	scStatusA[4]	state of SC-IC state machine A bit 3
SC-IC				5	scStatusA[5]	state of SC-IC state machine A bit 4
SC-IC				6	scStatusA[6]	state of SC-IC state machine A bit 5
SC-IC				7	scStatusA[7]	high if gbld access is not running

SC-IC		372	scStatusB	0	scStatusB[0]	result of parity check of previous SC-IC write
SC-IC				1	scStatusB[1]	state of SC-IC state machine B bit 0
SC-IC				2	scStatusB[2]	state of SC-IC state machine B bit 1
SC-IC				3	scStatusB[3]	state of SC-IC state machine B bit 2
SC-IC				4	scStatusB[4]	state of SC-IC state machine B bit 3
SC-IC				5	scStatusB[5]	state of SC-IC state machine B bit 4
SC-IC				6	scStatusB[6]	state of SC-IC state machine B bit 5
SC-IC				7	scStatusB[7]	high if gbld access is not running
SC-IC		373	scStatusC	0	scStatusC[0]	result of parity check of previous SC-IC write
SC-IC				1	scStatusC[1]	state of SC-IC state machine C bit 0
SC-IC				2	scStatusC[2]	state of SC-IC state machine C bit 1
SC-IC				3	scStatusC[3]	state of SC-IC state machine C bit 2
SC-IC				4	scStatusC[4]	state of SC-IC state machine C bit 3
SC-IC				5	scStatusC[5]	state of SC-IC state machine C bit 4
SC-IC				6	scStatusC[6]	state of SC-IC state machine C bit 5
SC-IC				7	scStatusC[7]	high if gbld access is not running
ROM		374		0	1	fixed value = 8'hA5
ROM				1	0	fixed value = 8'hA6
ROM				2	1	fixed value = 8'hA7
ROM				3	0	fixed value = 8'hA8
ROM				4	0	fixed value = 8'hA9
ROM				5	1	fixed value = 8'hA10
ROM				6	0	fixed value = 8'hA11
ROM				7	1	fixed value = 8'hA12
Registers		375		0	config_reg error_count[0]	count of SEU corrections in registers bit 0
Registers				1	config_reg error_count[1]	count of SEU corrections in registers bit 1
Registers				2	config_reg error_count[2]	count of SEU corrections in registers bit 2
Registers				3	config_reg error_count[3]	count of SEU corrections in registers bit 3
Registers				4	config_reg error_count[4]	count of SEU corrections in registers bit 4
Registers				5	config_reg error_count[5]	count of SEU corrections in registers bit 5
Registers				6	config_reg error_count[6]	count of SEU corrections in registers bit 6
Registers				7	config_reg error_count[7]	count of SEU corrections in registers bit 7
TX control		376		0	status0[0]	tx status: loss of lock count bit 0
TX control				1	status0[1]	tx status: loss of lock count bit 1
TX control				2	status0[2]	tx status: loss of lock count bit 2
TX control				3	status0[3]	tx status: loss of lock count bit 3
TX control				4	status0[4]	tx status: loss of lock count bit 4
TX control				5	status0[5]	tx status: loss of lock count bit 5
TX control				6	status0[6]	tx status: loss of lock count bit 6
TX control				7	status0[7]	tx status: loss of lock count bit 7
TX control		377		0	status1[0]	tx status: txRdy
RX-control				1	status1[1]	rx-status: vcoFast status bit
TX control				2	status1[2]	tx status: txControl state A bit 0
TX control				3	status1[3]	tx status: txControl state A bit 1
TX control				4	status1[4]	tx status: txControl state B bit 0
TX control				5	status1[5]	tx status: txControl state B bit 1
TX control				6	status1[6]	tx status: txControl state C bit 0
TX control				7	status1[7]	tx status: txControl state C bit 1
RX control		378		0	status2[0]	rx status: rxControl state A bit 0
RX control				1	status2[1]	rx status: rxControl state A bit 1
RX control				2	status2[2]	rx status: rxControl state A bit 2
RX control				3	status2[3]	rx status: rxControl state A bit 3
RX control				4	status2[4]	rx status: rxControl rxSkipCycle A
RX control				5	status2[5]	rx status: rxControl rxSelectDataInPhase A
RX control				6	status2[6]	rx status: rxControl rxSwap A
RX control				7	status2[7]	rx status: rxControl rxReady A
RX control		379		0	status3[0]	rx status: rxControl state B bit 0
RX control				1	status3[1]	rx status: rxControl state B bit 1
RX control				2	status3[2]	rx status: rxControl state B bit 2
RX control				3	status3[3]	rx status: rxControl state B bit 3

RX control				4	status3[4]	rx status: rxControl rxSkipCycle B
RX control				5	status3[5]	rx status: rxControl rxSelectDataInPhase B
RX control				6	status3[6]	rx status: rxControl rxSwap B
RX control				7	status3[7]	rx status: rxControl rxReady B
RX control		380		0	status4[0]	rx status: rxControl state C bit 0
RX control				1	status4[1]	rx status: rxControl state C bit 1
RX control				2	status4[2]	rx status: rxControl state C bit 2
RX control				3	status4[3]	rx status: rxControl state C bit 3
RX control				4	status4[4]	rx status: rxControl rxSkipCycle C
RX control				5	status4[5]	rx status: rxControl rxSelectDataInPhase C
RX control				6	status4[6]	rx status: rxControl rxSwap C
RX control				7	status4[7]	rx status: rxControl rxReady C
GBLD access		381		0	gbld_r0[0]	GBLD read data, register0, bit 0
GBLD access				1	gbld_r0[1]	GBLD read data, register0, bit 1
GBLD access				2	gbld_r0[2]	GBLD read data, register0, bit 2
GBLD access				3	gbld_r0[3]	GBLD read data, register0, bit 3
GBLD access				4	gbld_r0[4]	GBLD read data, register0, bit 4
GBLD access				5	gbld_r0[5]	GBLD read data, register0, bit 5
GBLD access				6	gbld_r0[6]	GBLD read data, register0, bit 6
GBLD access				7	gbld_r0[7]	GBLD read data, register0, bit 7
GBLD access		382		0	gbld_r1[0]	GBLD read data, register1, bit 0
GBLD access				1	gbld_r1[1]	GBLD read data, register1, bit 1
GBLD access				2	gbld_r1[2]	GBLD read data, register1, bit 2
GBLD access				3	gbld_r1[3]	GBLD read data, register1, bit 3
GBLD access				4	gbld_r1[4]	GBLD read data, register1, bit 4
GBLD access				5	gbld_r1[5]	GBLD read data, register1, bit 5
GBLD access				6	gbld_r1[6]	GBLD read data, register1, bit 6
GBLD access				7	gbld_r1[7]	GBLD read data, register1, bit 7
GBLD access		383		0	gbld_r2[0]	GBLD read data, register2, bit 0
GBLD access				1	gbld_r2[1]	GBLD read data, register2, bit 1
GBLD access				2	gbld_r2[2]	GBLD read data, register2, bit 2
GBLD access				3	gbld_r2[3]	GBLD read data, register2, bit 3
GBLD access				4	gbld_r2[4]	GBLD read data, register2, bit 4
GBLD access				5	gbld_r2[5]	GBLD read data, register2, bit 5
GBLD access				6	gbld_r2[6]	GBLD read data, register2, bit 6
GBLD access				7	gbld_r2[7]	GBLD read data, register2, bit 7
GBLD access		384		0	gbld_r3[0]	GBLD read data, register3, bit 0
GBLD access				1	gbld_r3[1]	GBLD read data, register3, bit 1
GBLD access				2	gbld_r3[2]	GBLD read data, register3, bit 2
GBLD access				3	gbld_r3[3]	GBLD read data, register3, bit 3
GBLD access				4	gbld_r3[4]	GBLD read data, register3, bit 4
GBLD access				5	gbld_r3[5]	GBLD read data, register3, bit 5
GBLD access				6	gbld_r3[6]	GBLD read data, register3, bit 6
GBLD access				7	gbld_r3[7]	GBLD read data, register3, bit 7
GBLD access		385		0	gbld_r4[0]	GBLD read data, register4, bit 0
GBLD access				1	gbld_r4[1]	GBLD read data, register4, bit 1
GBLD access				2	gbld_r4[2]	GBLD read data, register4, bit 2
GBLD access				3	gbld_r4[3]	GBLD read data, register4, bit 3
GBLD access				4	gbld_r4[4]	GBLD read data, register4, bit 4
GBLD access				5	gbld_r4[5]	GBLD read data, register4, bit 5
GBLD access				6	gbld_r4[6]	GBLD read data, register4, bit 6
GBLD access				7	gbld_r4[7]	GBLD read data, register4, bit 7
GBLD access		386		0	gbld_r5[0]	GBLD read data, register5, bit 0
GBLD access				1	gbld_r5[1]	GBLD read data, register5, bit 1
GBLD access				2	gbld_r5[2]	GBLD read data, register5, bit 2
GBLD access				3	gbld_r5[3]	GBLD read data, register5, bit 3
GBLD access				4	gbld_r5[4]	GBLD read data, register5, bit 4
GBLD access				5	gbld_r5[5]	GBLD read data, register5, bit 5
GBLD access				6	gbld_r5[6]	GBLD read data, register5, bit 6
GBLD access				7	gbld_r5[7]	GBLD read data, register5, bit 7

GBLD access		387		0	gbld_r6[0]	GBLD read data, register6, bit 0
GBLD access				1	gbld_r6[1]	GBLD read data, register6, bit 1
GBLD access				2	gbld_r6[2]	GBLD read data, register6, bit 2
GBLD access				3	gbld_r6[3]	GBLD read data, register6, bit 3
GBLD access				4	gbld_r6[4]	GBLD read data, register6, bit 4
GBLD access				5	gbld_r6[5]	GBLD read data, register6, bit 5
GBLD access				6	gbld_r6[6]	GBLD read data, register6, bit 6
GBLD access				7	gbld_r6[7]	GBLD read data, register6, bit 7
EportRX		390		0	dllLockedGroup0	EportRX dll locked status, group 0
EportRX				1	dllLockedGroup1	EportRX dll locked status, group 1
EportRX				2	dllLockedGroup2	EportRX dll locked status, group 2
EportRX				3	dllLockedGroup3	EportRX dll locked status, group 3
EportRX				4	dllLockedGroup4	EportRX dll locked status, group 4
EportRX				5	dllLockedGroup5	EportRX dll locked status, group 5
EportRX				6	dllLockedGroup	EportRX dll locked status, group 6
EportRX				7	dllLockedEc	EportRX dll locked status, EC channel
EportRX		391		0	channelLockedGroup0[0]	EportRX phase aligner locked status, group0 ch 0
EportRX				1	channelLockedGroup0[1]	EportRX phase aligner locked status, group0 ch 1
EportRX				2	channelLockedGroup0[2]	EportRX phase aligner locked status, group0 ch 2
EportRX				3	channelLockedGroup0[3]	EportRX phase aligner locked status, group0 ch 3
EportRX				4	channelLockedGroup0[4]	EportRX phase aligner locked status, group0 ch 4
EportRX				5	channelLockedGroup0[5]	EportRX phase aligner locked status, group0 ch 5
EportRX				6	channelLockedGroup0[6]	EportRX phase aligner locked status, group0 ch 6
EportRX				7	channelLockedGroup0[7]	EportRX phase aligner locked status, group0 ch 7
EportRX		392		0	channelLockedGroup1[0]	EportRX phase aligner locked status, group1 ch 0
EportRX				1	channelLockedGroup1[1]	EportRX phase aligner locked status, group1 ch 1
EportRX				2	channelLockedGroup1[2]	EportRX phase aligner locked status, group1 ch 2
EportRX				3	channelLockedGroup1[3]	EportRX phase aligner locked status, group1 ch 3
EportRX				4	channelLockedGroup1[4]	EportRX phase aligner locked status, group1 ch 4
EportRX				5	channelLockedGroup1[5]	EportRX phase aligner locked status, group1 ch 5
EportRX				6	channelLockedGroup1[6]	EportRX phase aligner locked status, group1 ch 6
EportRX				7	channelLockedGroup1[7]	EportRX phase aligner locked status, group1 ch 7
EportRX		393		0	channelLockedGroup2[0]	EportRX phase aligner locked status, group2 ch 0
EportRX				1	channelLockedGroup2[1]	EportRX phase aligner locked status, group2 ch 1
EportRX				2	channelLockedGroup2[2]	EportRX phase aligner locked status, group2 ch 2
EportRX				3	channelLockedGroup2[3]	EportRX phase aligner locked status, group2 ch 3
EportRX				4	channelLockedGroup2[4]	EportRX phase aligner locked status, group2 ch 4
EportRX				5	channelLockedGroup2[5]	EportRX phase aligner locked status, group2 ch 5
EportRX				6	channelLockedGroup2[6]	EportRX phase aligner locked status, group2 ch 6
EportRX				7	channelLockedGroup2[7]	EportRX phase aligner locked status, group2 ch 7
EportRX		394		0	channelLockedGroup3[0]	EportRX phase aligner locked status, group3 ch 0
EportRX				1	channelLockedGroup3[1]	EportRX phase aligner locked status, group3 ch 1
EportRX				2	channelLockedGroup3[2]	EportRX phase aligner locked status, group3 ch 2
EportRX				3	channelLockedGroup3[3]	EportRX phase aligner locked status, group3 ch 3
EportRX				4	channelLockedGroup3[4]	EportRX phase aligner locked status, group3 ch 4
EportRX				5	channelLockedGroup3[5]	EportRX phase aligner locked status, group3 ch 5
EportRX				6	channelLockedGroup3[6]	EportRX phase aligner locked status, group3 ch 6
EportRX				7	channelLockedGroup3[7]	EportRX phase aligner locked status, group3 ch 7
EportRX		395		0	channelLockedGroup4[0]	EportRX phase aligner locked status, group4 ch 0
EportRX				1	channelLockedGroup4[1]	EportRX phase aligner locked status, group4 ch 1
EportRX				2	channelLockedGroup4[2]	EportRX phase aligner locked status, group4 ch 2
EportRX				3	channelLockedGroup4[3]	EportRX phase aligner locked status, group4 ch 3
EportRX				4	channelLockedGroup4[4]	EportRX phase aligner locked status, group4 ch 4
EportRX				5	channelLockedGroup4[5]	EportRX phase aligner locked status, group4 ch 5
EportRX				6	channelLockedGroup4[6]	EportRX phase aligner locked status, group4 ch 6
EportRX				7	channelLockedGroup4[7]	EportRX phase aligner locked status, group4 ch 7
EportRX		396		0	channelLockedGroup5[0]	EportRX phase aligner locked status, group5 ch 0
EportRX				1	channelLockedGroup5[1]	EportRX phase aligner locked status, group5 ch 1
EportRX				2	channelLockedGroup5[2]	EportRX phase aligner locked status, group5 ch 2
EportRX				3	channelLockedGroup5[3]	EportRX phase aligner locked status, group5 ch 3

EportRX				4	channelLockedGroup5[4]	EportRX phase aligner locked status, group5 ch 4
EportRX				5	channelLockedGroup5[5]	EportRX phase aligner locked status, group5 ch 5
EportRX				6	channelLockedGroup5[6]	EportRX phase aligner locked status, group5 ch 6
EportRX				7	channelLockedGroup5[7]	EportRX phase aligner locked status, group5 ch 7
EportRX		397		0	channelLockedGroup6[0]	EportRX phase aligner locked status, group6 ch 0
EportRX				1	channelLockedGroup6[1]	EportRX phase aligner locked status, group6 ch 1
EportRX				2	channelLockedGroup6[2]	EportRX phase aligner locked status, group6 ch 2
EportRX				3	channelLockedGroup6[3]	EportRX phase aligner locked status, group6 ch 3
EportRX				4	channelLockedGroup6[4]	EportRX phase aligner locked status, group6 ch 4
EportRX				5	channelLockedGroup6[5]	EportRX phase aligner locked status, group6 ch 5
EportRX				6	channelLockedGroup6[6]	EportRX phase aligner locked status, group6 ch 6
EportRX				7	channelLockedGroup6[7]	EportRX phase aligner locked status, group6 ch 7
EportRX		398		0	phaseSelectOutEc[0]	EportRX selected phase EC channel bit 0
EportRX				1	phaseSelectOutEc[1]	EportRX selected phase EC channel bit 1
EportRX				2	phaseSelectOutEc[2]	EportRX selected phase EC channel bit 2
EportRX				3	phaseSelectOutEc[3]	EportRX selected phase EC channel bit 3
EportRX				4	channelLockedGroupEc	EportRX phase aligner locked status, EC channel
Unused				5		unused
Unused				6		unused
Unused				7		unused
EportRX		399		0	phaseSelectOutGroup0[0]	EportRX selected phase, group0, ch0, bit 0
EportRX				1	phaseSelectOutGroup0[1]	EportRX selected phase, group0, ch0, bit 1
EportRX				2	phaseSelectOutGroup0[2]	EportRX selected phase, group0, ch0, bit 2
EportRX				3	phaseSelectOutGroup0[3]	EportRX selected phase, group0, ch0, bit 3
EportRX				4	phaseSelectOutGroup0[4]	EportRX selected phase, group0, ch1, bit 0
EportRX				5	phaseSelectOutGroup0[5]	EportRX selected phase, group0, ch1, bit 1
EportRX				6	phaseSelectOutGroup0[6]	EportRX selected phase, group0, ch1, bit 2
EportRX				7	phaseSelectOutGroup0[7]	EportRX selected phase, group0, ch1, bit 3
EportRX		400		0	phaseSelectOutGroup0[8]	EportRX selected phase, group0, ch2, bit 0
EportRX				1	phaseSelectOutGroup0[9]	EportRX selected phase, group0, ch2, bit 1
EportRX				2	phaseSelectOutGroup0[10]	EportRX selected phase, group0, ch2, bit 2
EportRX				3	phaseSelectOutGroup0[11]	EportRX selected phase, group0, ch2, bit 3
EportRX				4	phaseSelectOutGroup0[12]	EportRX selected phase, group0, ch3, bit 0
EportRX				5	phaseSelectOutGroup0[13]	EportRX selected phase, group0, ch3, bit 1
EportRX				6	phaseSelectOutGroup0[14]	EportRX selected phase, group0, ch3, bit 2
EportRX				7	phaseSelectOutGroup0[15]	EportRX selected phase, group0, ch3, bit 3
EportRX		401		0	phaseSelectOutGroup0[16]	EportRX selected phase, group0, ch4, bit 0
EportRX				1	phaseSelectOutGroup0[17]	EportRX selected phase, group0, ch4, bit 1
EportRX				2	phaseSelectOutGroup0[18]	EportRX selected phase, group0, ch4, bit 2
EportRX				3	phaseSelectOutGroup0[19]	EportRX selected phase, group0, ch4, bit 3
EportRX				4	phaseSelectOutGroup0[20]	EportRX selected phase, group0, ch5, bit 0
EportRX				5	phaseSelectOutGroup0[21]	EportRX selected phase, group0, ch5, bit 1
EportRX				6	phaseSelectOutGroup0[22]	EportRX selected phase, group0, ch5, bit 2
EportRX				7	phaseSelectOutGroup0[23]	EportRX selected phase, group0, ch5, bit 3
EportRX		402		0	phaseSelectOutGroup0[24]	EportRX selected phase, group0, ch6, bit 0
EportRX				1	phaseSelectOutGroup0[25]	EportRX selected phase, group0, ch6, bit 1
EportRX				2	phaseSelectOutGroup0[26]	EportRX selected phase, group0, ch6, bit 2
EportRX				3	phaseSelectOutGroup0[27]	EportRX selected phase, group0, ch6, bit 3
EportRX				4	phaseSelectOutGroup0[28]	EportRX selected phase, group0, ch7, bit 0
EportRX				5	phaseSelectOutGroup0[29]	EportRX selected phase, group0, ch7, bit 1
EportRX				6	phaseSelectOutGroup0[30]	EportRX selected phase, group0, ch7, bit 2
EportRX				7	phaseSelectOutGroup0[31]	EportRX selected phase, group0, ch7, bit 3
EportRX		403		0	phaseSelectOutGroup1[0]	EportRX selected phase, group1, ch0, bit 0
EportRX				1	phaseSelectOutGroup1[1]	EportRX selected phase, group1, ch0, bit 1
EportRX				2	phaseSelectOutGroup1[2]	EportRX selected phase, group1, ch0, bit 2
EportRX				3	phaseSelectOutGroup1[3]	EportRX selected phase, group1, ch0, bit 3
EportRX				4	phaseSelectOutGroup1[4]	EportRX selected phase, group1, ch1, bit 0
EportRX				5	phaseSelectOutGroup1[5]	EportRX selected phase, group1, ch1, bit 1
EportRX				6	phaseSelectOutGroup1[6]	EportRX selected phase, group1, ch1, bit 2
EportRX				7	phaseSelectOutGroup1[7]	EportRX selected phase, group1, ch1, bit 3

EportRX		404		0	phaseSelectOutGroup1[8]	EportRX selected phase, group1, ch2, bit 0
EportRX				1	phaseSelectOutGroup1[9]	EportRX selected phase, group1, ch2, bit 1
EportRX				2	phaseSelectOutGroup1[10]	EportRX selected phase, group1, ch2, bit 2
EportRX				3	phaseSelectOutGroup1[11]	EportRX selected phase, group1, ch2, bit 3
EportRX				4	phaseSelectOutGroup1[12]	EportRX selected phase, group1, ch3, bit 0
EportRX				5	phaseSelectOutGroup1[13]	EportRX selected phase, group1, ch3, bit 1
EportRX				6	phaseSelectOutGroup1[14]	EportRX selected phase, group1, ch3, bit 2
EportRX				7	phaseSelectOutGroup1[15]	EportRX selected phase, group1, ch3, bit 3
EportRX		405		0	phaseSelectOutGroup1[16]	EportRX selected phase, group1, ch4, bit 0
EportRX				1	phaseSelectOutGroup1[17]	EportRX selected phase, group1, ch4, bit 1
EportRX				2	phaseSelectOutGroup1[18]	EportRX selected phase, group1, ch4, bit 2
EportRX				3	phaseSelectOutGroup1[19]	EportRX selected phase, group1, ch4, bit 3
EportRX				4	phaseSelectOutGroup1[20]	EportRX selected phase, group1, ch5, bit 0
EportRX				5	phaseSelectOutGroup1[21]	EportRX selected phase, group1, ch5, bit 1
EportRX				6	phaseSelectOutGroup1[22]	EportRX selected phase, group1, ch5, bit 2
EportRX				7	phaseSelectOutGroup1[23]	EportRX selected phase, group1, ch5, bit 3
EportRX		406		0	phaseSelectOutGroup1[24]	EportRX selected phase, group1, ch6, bit 0
EportRX				1	phaseSelectOutGroup1[25]	EportRX selected phase, group1, ch6, bit 1
EportRX				2	phaseSelectOutGroup1[26]	EportRX selected phase, group1, ch6, bit 2
EportRX				3	phaseSelectOutGroup1[27]	EportRX selected phase, group1, ch6, bit 3
EportRX				4	phaseSelectOutGroup1[28]	EportRX selected phase, group1, ch7, bit 0
EportRX				5	phaseSelectOutGroup1[29]	EportRX selected phase, group1, ch7, bit 1
EportRX				6	phaseSelectOutGroup1[30]	EportRX selected phase, group1, ch7, bit 2
EportRX				7	phaseSelectOutGroup1[31]	EportRX selected phase, group1, ch7, bit 3
EportRX		407		0	phaseSelectOutGroup2[0]	EportRX selected phase, group2, ch0, bit 0
EportRX				1	phaseSelectOutGroup2[1]	EportRX selected phase, group2, ch0, bit 1
EportRX				2	phaseSelectOutGroup2[2]	EportRX selected phase, group2, ch0, bit 2
EportRX				3	phaseSelectOutGroup2[3]	EportRX selected phase, group2, ch0, bit 3
EportRX				4	phaseSelectOutGroup2[4]	EportRX selected phase, group2, ch1, bit 0
EportRX				5	phaseSelectOutGroup2[5]	EportRX selected phase, group2, ch1, bit 1
EportRX				6	phaseSelectOutGroup2[6]	EportRX selected phase, group2, ch1, bit 2
EportRX				7	phaseSelectOutGroup2[7]	EportRX selected phase, group2, ch1, bit 3
EportRX		408		0	phaseSelectOutGroup2[8]	EportRX selected phase, group2, ch2, bit 0
EportRX				1	phaseSelectOutGroup2[9]	EportRX selected phase, group2, ch2, bit 1
EportRX				2	phaseSelectOutGroup2[10]	EportRX selected phase, group2, ch2, bit 2
EportRX				3	phaseSelectOutGroup2[11]	EportRX selected phase, group2, ch2, bit 3
EportRX				4	phaseSelectOutGroup2[12]	EportRX selected phase, group2, ch3, bit 0
EportRX				5	phaseSelectOutGroup2[13]	EportRX selected phase, group2, ch3, bit 1
EportRX				6	phaseSelectOutGroup2[14]	EportRX selected phase, group2, ch3, bit 2
EportRX				7	phaseSelectOutGroup2[15]	EportRX selected phase, group2, ch3, bit 3
EportRX		409		0	phaseSelectOutGroup2[16]	EportRX selected phase, group2, ch4, bit 0
EportRX				1	phaseSelectOutGroup2[17]	EportRX selected phase, group2, ch4, bit 1
EportRX				2	phaseSelectOutGroup2[18]	EportRX selected phase, group2, ch4, bit 2
EportRX				3	phaseSelectOutGroup2[19]	EportRX selected phase, group2, ch4, bit 3
EportRX				4	phaseSelectOutGroup2[20]	EportRX selected phase, group2, ch5, bit 0
EportRX				5	phaseSelectOutGroup2[21]	EportRX selected phase, group2, ch5, bit 1
EportRX				6	phaseSelectOutGroup2[22]	EportRX selected phase, group2, ch5, bit 2
EportRX				7	phaseSelectOutGroup2[23]	EportRX selected phase, group2, ch5, bit 3
EportRX		410		0	phaseSelectOutGroup2[24]	EportRX selected phase, group2, ch6, bit 0
EportRX				1	phaseSelectOutGroup2[25]	EportRX selected phase, group2, ch6, bit 1
EportRX				2	phaseSelectOutGroup2[26]	EportRX selected phase, group2, ch6, bit 2
EportRX				3	phaseSelectOutGroup2[27]	EportRX selected phase, group2, ch6, bit 3
EportRX				4	phaseSelectOutGroup2[28]	EportRX selected phase, group2, ch7, bit 0
EportRX				5	phaseSelectOutGroup2[29]	EportRX selected phase, group2, ch7, bit 1
EportRX				6	phaseSelectOutGroup2[30]	EportRX selected phase, group2, ch7, bit 2
EportRX				7	phaseSelectOutGroup2[31]	EportRX selected phase, group2, ch7, bit 3
EportRX		411		0	phaseSelectOutGroup3[0]	EportRX selected phase, group3, ch0, bit 0
EportRX				1	phaseSelectOutGroup3[1]	EportRX selected phase, group3, ch0, bit 1
EportRX				2	phaseSelectOutGroup3[2]	EportRX selected phase, group3, ch0, bit 2
EportRX				3	phaseSelectOutGroup3[3]	EportRX selected phase, group3, ch0, bit 3

EportRX				4	phaseSelectOutGroup3[4]	EportRX selected phase, group3, ch1, bit 0
EportRX				5	phaseSelectOutGroup3[5]	EportRX selected phase, group3, ch1, bit 1
EportRX				6	phaseSelectOutGroup3[6]	EportRX selected phase, group3, ch1, bit 2
EportRX				7	phaseSelectOutGroup3[7]	EportRX selected phase, group3, ch1, bit 3
EportRX		412		0	phaseSelectOutGroup3[8]	EportRX selected phase, group3, ch2, bit 0
EportRX				1	phaseSelectOutGroup3[9]	EportRX selected phase, group3, ch2, bit 1
EportRX				2	phaseSelectOutGroup3[10]	EportRX selected phase, group3, ch2, bit 2
EportRX				3	phaseSelectOutGroup3[11]	EportRX selected phase, group3, ch2, bit 3
EportRX				4	phaseSelectOutGroup3[12]	EportRX selected phase, group3, ch3, bit 0
EportRX				5	phaseSelectOutGroup3[13]	EportRX selected phase, group3, ch3, bit 1
EportRX				6	phaseSelectOutGroup3[14]	EportRX selected phase, group3, ch3, bit 2
EportRX				7	phaseSelectOutGroup3[15]	EportRX selected phase, group3, ch3, bit 3
EportRX		413		0	phaseSelectOutGroup3[16]	EportRX selected phase, group3, ch4, bit 0
EportRX				1	phaseSelectOutGroup3[17]	EportRX selected phase, group3, ch4, bit 1
EportRX				2	phaseSelectOutGroup3[18]	EportRX selected phase, group3, ch4, bit 2
EportRX				3	phaseSelectOutGroup3[19]	EportRX selected phase, group3, ch4, bit 3
EportRX				4	phaseSelectOutGroup3[20]	EportRX selected phase, group3, ch5, bit 0
EportRX				5	phaseSelectOutGroup3[21]	EportRX selected phase, group3, ch5, bit 1
EportRX				6	phaseSelectOutGroup3[22]	EportRX selected phase, group3, ch5, bit 2
EportRX				7	phaseSelectOutGroup3[23]	EportRX selected phase, group3, ch5, bit 3
EportRX		414		0	phaseSelectOutGroup3[24]	EportRX selected phase, group3, ch6, bit 0
EportRX				1	phaseSelectOutGroup3[25]	EportRX selected phase, group3, ch6, bit 1
EportRX				2	phaseSelectOutGroup3[26]	EportRX selected phase, group3, ch6, bit 2
EportRX				3	phaseSelectOutGroup3[27]	EportRX selected phase, group3, ch6, bit 3
EportRX				4	phaseSelectOutGroup3[28]	EportRX selected phase, group3, ch7, bit 0
EportRX				5	phaseSelectOutGroup3[29]	EportRX selected phase, group3, ch7, bit 1
EportRX				6	phaseSelectOutGroup3[30]	EportRX selected phase, group3, ch7, bit 2
EportRX				7	phaseSelectOutGroup3[31]	EportRX selected phase, group3, ch7, bit 3
EportRX		415		0	phaseSelectOutGroup4[0]	EportRX selected phase, group4, ch0, bit 0
EportRX				1	phaseSelectOutGroup4[1]	EportRX selected phase, group4, ch0, bit 1
EportRX				2	phaseSelectOutGroup4[2]	EportRX selected phase, group4, ch0, bit 2
EportRX				3	phaseSelectOutGroup4[3]	EportRX selected phase, group4, ch0, bit 3
EportRX				4	phaseSelectOutGroup4[4]	EportRX selected phase, group4, ch1, bit 0
EportRX				5	phaseSelectOutGroup4[5]	EportRX selected phase, group4, ch1, bit 1
EportRX				6	phaseSelectOutGroup4[6]	EportRX selected phase, group4, ch1, bit 2
EportRX				7	phaseSelectOutGroup4[7]	EportRX selected phase, group4, ch1, bit 3
EportRX		416		0	phaseSelectOutGroup4[8]	EportRX selected phase, group4, ch2, bit 0
EportRX				1	phaseSelectOutGroup4[9	EportRX selected phase, group4, ch2, bit 1
EportRX				2	phaseSelectOutGroup4[10]	EportRX selected phase, group4, ch2, bit 2
EportRX				3	phaseSelectOutGroup4[11]	EportRX selected phase, group4, ch2, bit 3
EportRX				4	phaseSelectOutGroup4[12]	EportRX selected phase, group4, ch3, bit 0
EportRX				5	phaseSelectOutGroup4[13]	EportRX selected phase, group4, ch3, bit 1
EportRX				6	phaseSelectOutGroup4[14]	EportRX selected phase, group4, ch3, bit 2
EportRX				7	phaseSelectOutGroup4[15]	EportRX selected phase, group4, ch3, bit 3
EportRX		417		0	phaseSelectOutGroup4[16]	EportRX selected phase, group4, ch4, bit 0
EportRX				1	phaseSelectOutGroup4[17]	EportRX selected phase, group4, ch4, bit 1
EportRX				2	phaseSelectOutGroup4[18]	EportRX selected phase, group4, ch4, bit 2
EportRX				3	phaseSelectOutGroup4[19]	EportRX selected phase, group4, ch4, bit 3
EportRX				4	phaseSelectOutGroup4[20]	EportRX selected phase, group4, ch5, bit 0
EportRX				5	phaseSelectOutGroup4[21]	EportRX selected phase, group4, ch5, bit 1
EportRX				6	phaseSelectOutGroup4[22]	EportRX selected phase, group4, ch5, bit 2
EportRX				7	phaseSelectOutGroup4[23]	EportRX selected phase, group4, ch5, bit 3
EportRX		418		0	phaseSelectOutGroup4[24]	EportRX selected phase, group4, ch6, bit 0
EportRX				1	phaseSelectOutGroup4[25]	EportRX selected phase, group4, ch6, bit 1
EportRX				2	phaseSelectOutGroup4[26]	EportRX selected phase, group4, ch6, bit 2
EportRX				3	phaseSelectOutGroup4[27]	EportRX selected phase, group4, ch6, bit 3
EportRX				4	phaseSelectOutGroup4[28]	EportRX selected phase, group4, ch7, bit 0
EportRX				5	phaseSelectOutGroup4[29]	EportRX selected phase, group4, ch7, bit 1
EportRX				6	phaseSelectOutGroup4[30]	EportRX selected phase, group4, ch7, bit 2
EportRX				7	phaseSelectOutGroup4[31]	EportRX selected phase, group4, ch7, bit 3

EportRX		419		0	phaseSelectOutGroup5[0]	EportRX selected phase, group5, ch0, bit 0
EportRX				1	phaseSelectOutGroup5[1]	EportRX selected phase, group5, ch0, bit 1
EportRX				2	phaseSelectOutGroup5[2]	EportRX selected phase, group5, ch0, bit 2
EportRX				3	phaseSelectOutGroup5[3]	EportRX selected phase, group5, ch0, bit 3
EportRX				4	phaseSelectOutGroup5[4]	EportRX selected phase, group5, ch1, bit 0
EportRX				5	phaseSelectOutGroup5[5]	EportRX selected phase, group5, ch1, bit 1
EportRX				6	phaseSelectOutGroup5[6]	EportRX selected phase, group5, ch1, bit 2
EportRX				7	phaseSelectOutGroup5[7]	EportRX selected phase, group5, ch1, bit 3
EportRX		420		0	phaseSelectOutGroup5[8]	EportRX selected phase, group5, ch2, bit 0
EportRX				1	phaseSelectOutGroup5[9]	EportRX selected phase, group5, ch2, bit 1
EportRX				2	phaseSelectOutGroup5[10]	EportRX selected phase, group5, ch2, bit 2
EportRX				3	phaseSelectOutGroup5[11]	EportRX selected phase, group5, ch2, bit 3
EportRX				4	phaseSelectOutGroup5[12]	EportRX selected phase, group5, ch3, bit 0
EportRX				5	phaseSelectOutGroup5[13]	EportRX selected phase, group5, ch3, bit 1
EportRX				6	phaseSelectOutGroup5[14]	EportRX selected phase, group5, ch3, bit 2
EportRX				7	phaseSelectOutGroup5[15]	EportRX selected phase, group5, ch3, bit 3
EportRX		421		0	phaseSelectOutGroup5[16]	EportRX selected phase, group5, ch4, bit 0
EportRX				1	phaseSelectOutGroup5[17]	EportRX selected phase, group5, ch4, bit 1
EportRX				2	phaseSelectOutGroup5[18]	EportRX selected phase, group5, ch4, bit 2
EportRX				3	phaseSelectOutGroup5[19]	EportRX selected phase, group5, ch4, bit 3
EportRX				4	phaseSelectOutGroup5[20]	EportRX selected phase, group5, ch5, bit 0
EportRX				5	phaseSelectOutGroup5[21]	EportRX selected phase, group5, ch5, bit 1
EportRX				6	phaseSelectOutGroup5[22]	EportRX selected phase, group5, ch5, bit 2
EportRX				7	phaseSelectOutGroup5[23]	EportRX selected phase, group5, ch5, bit 3
EportRX		422		0	phaseSelectOutGroup5[24]	EportRX selected phase, group5, ch6, bit 0
EportRX				1	phaseSelectOutGroup5[25]	EportRX selected phase, group5, ch6, bit 1
EportRX				2	phaseSelectOutGroup5[26]	EportRX selected phase, group5, ch6, bit 2
EportRX				3	phaseSelectOutGroup5[27]	EportRX selected phase, group5, ch6, bit 3
EportRX				4	phaseSelectOutGroup5[28]	EportRX selected phase, group5, ch7, bit 0
EportRX				5	phaseSelectOutGroup5[29]	EportRX selected phase, group5, ch7, bit 1
EportRX				6	phaseSelectOutGroup5[30]	EportRX selected phase, group5, ch7, bit 2
EportRX				7	phaseSelectOutGroup5[31]	EportRX selected phase, group5, ch7, bit 3
EportRX		423		0	phaseSelectOutGroup6[0]	EportRX selected phase, group6, ch0, bit 0
EportRX				1	phaseSelectOutGroup6[1]	EportRX selected phase, group6, ch0, bit 1
EportRX				2	phaseSelectOutGroup6[2]	EportRX selected phase, group6, ch0, bit 2
EportRX				3	phaseSelectOutGroup6[3]	EportRX selected phase, group6, ch0, bit 3
EportRX				4	phaseSelectOutGroup6[4]	EportRX selected phase, group6, ch1, bit 0
EportRX				5	phaseSelectOutGroup6[5]	EportRX selected phase, group6, ch1, bit 1
EportRX				6	phaseSelectOutGroup6[6]	EportRX selected phase, group6, ch1, bit 2
EportRX				7	phaseSelectOutGroup6[7]	EportRX selected phase, group6, ch1, bit 3
EportRX		424		0	phaseSelectOutGroup6[8]	EportRX selected phase, group6, ch2, bit 0
EportRX				1	phaseSelectOutGroup6[9]	EportRX selected phase, group6, ch2, bit 1
EportRX				2	phaseSelectOutGroup6[10]	EportRX selected phase, group6, ch2, bit 2
EportRX				3	phaseSelectOutGroup6[11]	EportRX selected phase, group6, ch2, bit 3
EportRX				4	phaseSelectOutGroup6[12]	EportRX selected phase, group6, ch3, bit 0
EportRX				5	phaseSelectOutGroup6[13]	EportRX selected phase, group6, ch3, bit 1
EportRX				6	phaseSelectOutGroup6[14]	EportRX selected phase, group6, ch3, bit 2
EportRX				7	phaseSelectOutGroup6[15]	EportRX selected phase, group6, ch3, bit 3
EportRX		425		0	phaseSelectOutGroup6[16]	EportRX selected phase, group6, ch4, bit 0
EportRX				1	phaseSelectOutGroup6[17]	EportRX selected phase, group6, ch4, bit 1
EportRX				2	phaseSelectOutGroup6[18]	EportRX selected phase, group6, ch4, bit 2
EportRX				3	phaseSelectOutGroup6[19]	EportRX selected phase, group6, ch4, bit 3
EportRX				4	phaseSelectOutGroup6[20]	EportRX selected phase, group6, ch5, bit 0
EportRX				5	phaseSelectOutGroup6[21]	EportRX selected phase, group6, ch5, bit 1
EportRX				6	phaseSelectOutGroup6[22]	EportRX selected phase, group6, ch5, bit 2
EportRX				7	phaseSelectOutGroup6[23]	EportRX selected phase, group6, ch5, bit 3
EportRX		426		0	phaseSelectOutGroup6[24]	EportRX selected phase, group6, ch6, bit 0
EportRX				1	phaseSelectOutGroup6[25]	EportRX selected phase, group6, ch6, bit 1
EportRX				2	phaseSelectOutGroup6[26]	EportRX selected phase, group6, ch6, bit 2
EportRX				3	phaseSelectOutGroup6[27]	EportRX selected phase, group6, ch6, bit 3

EportRX				4	phaseSelectOutGroup6[28]	EportRX selected phase, group6, ch7, bit 0
EportRX				5	phaseSelectOutGroup6[29]	EportRX selected phase, group6, ch7, bit 1
EportRX				6	phaseSelectOutGroup6[30]	EportRX selected phase, group6, ch7, bit 2
EportRX				7	phaseSelectOutGroup6[31]	EportRX selected phase, group6, ch7, bit 3
EPLL-TX		427		0	TXEPLLLocked	EPLL-TX locked status
TX control				1	txRdy_control	txControl txRdy
EPLL-RX				2	RXEPLLLocked	EPLL-RX locked status
RX control				3	rxRdy_control	rxControl rxRdy
XPLL				4	XPLLLocked	XPLL locked status
				5		unused
				6		unused
				7		unused
Phase shifter		428		0	ttcDivideOut	phase shifter DivideOut
Phase shifter				1	ttcTestDown	phase shifter TestDown
Phase shifter				2	ttcTestUp	phase shifter TestUp
Serialiser				3	txTestUp	Serialiser TestUp
Serialiser				4	txTestDown	Serialiser TestDown
EPLL-RX				5	ePLLRXInstantLock	EPLL-RX instant lock status
EPLL-TX				6	ePLLTXInstantLock	EPLL-TX instant lock status
XPLL				7	xPllInstantLock	XPLL instant lock status
Phase shifter		429		0	ttcEarly[0]	phase shifter Early bit 0
Phase shifter				1	ttcEarly[1]	phase shifter Early bit 1
Phase shifter				2	ttcEarly[2]	phase shifter Early bit 2
Phase shifter				3	ttcEarly[3]	phase shifter Early bit 3
Phase shifter				4	ttcEarly[4]	phase shifter Early bit 4
Phase shifter				5	ttcEarly[5]	phase shifter Early bit 5
Phase shifter				6	ttcEarly[6]	phase shifter Early bit 6
Phase shifter				7	ttcEarly[7]	phase shifter Early bit 7
Phase shifter		430		0	ttcLate[0]	phase shifter Late bit 0
Phase shifter				1	ttcLate[1]	phase shifter Late bit 1
Phase shifter				2	ttcLate[2]	phase shifter Late bit 2
Phase shifter				3	ttcLate[3]	phase shifter Late bit 3
Phase shifter				4	ttcLate[4]	phase shifter Late bit 4
Phase shifter				5	ttcLate[5]	phase shifter Late bit 5
Phase shifter				6	ttcLate[6]	phase shifter Late bit 6
Phase shifter				7	ttcLate[7]	phase shifter Late bit 7
TX control		431		0	txInstantLockgated	Serialiser instant lock status
RX control				1	rxInstantLockRefGated	Deserialiser instant lock status
Power-up				2	powerUpFSMState[0]	state of powerUp FSM bit 0
Power-up				3	powerUpFSMState[1]	state of powerUp FSM bit 1
Power-up				4	powerUpFSMState[2]	state of powerUp FSM bit 2
Power-up				5	powerUpFSMState[3]	state of powerUp FSM bit 3
Power-up				6	powerUpFSMState[4]	state of powerUp FSM bit 4
				7		unused
RX control		432		0	rxRefPllLossOfLockCount[0]	RF PLL loss of lock count bit 0
RX control				1	rxRefPllLossOfLockCount[1]	RF PLL loss of lock count bit 1
RX control				2	rxRefPllLossOfLockCount[2]	RF PLL loss of lock count bit 2
RX control				3	rxRefPllLossOfLockCount[3]	RF PLL loss of lock count bit 3
RX control				4	rxRefPllLossOfLockCount[4]	RF PLL loss of lock count bit 4
RX control				5	rxRefPllLossOfLockCount[5]	RF PLL loss of lock count bit 5
RX control				6	rxRefPllLossOfLockCount[6]	RF PLL loss of lock count bit 6
RX control				7	rxRefPllLossOfLockCount[7]	RF PLL loss of lock count bit 7
EPLL-TX		433		0	EPLLTXlossOfLockCount[0]	EPLL-TX loss of lock count bit 0
EPLL-TX				1	EPLLTXlossOfLockCount[1]	EPLL-TX loss of lock count bit 1
EPLL-TX				2	EPLLTXlossOfLockCount[2]	EPLL-TX loss of lock count bit 2
EPLL-TX				3	EPLLTXlossOfLockCount[3]	EPLL-TX loss of lock count bit 3
EPLL-TX				4	EPLLTXlossOfLockCount[4]	EPLL-TX loss of lock count bit 4
EPLL-TX				5	EPLLTXlossOfLockCount[5]	EPLL-TX loss of lock count bit 5
EPLL-TX				6	EPLLTXlossOfLockCount[6]	EPLL-TX loss of lock count bit 6
EPLL-TX				7	EPLLTXlossOfLockCount[7]	EPLL-TX loss of lock count bit 7

EPLL-RX		434		0	EPLLRXlossOfLockCount[0]	EPLL-RX loss of lock count bit 0
EPLL-RX				1	EPLLRXlossOfLockCount[1]	EPLL-RX loss of lock count bit 1
EPLL-RX				2	EPLLRXlossOfLockCount[2]	EPLL-RX loss of lock count bit 2
EPLL-RX				3	EPLLRXlossOfLockCount[3]	EPLL-RX loss of lock count bit 3
EPLL-RX				4	EPLLRXlossOfLockCount[4]	EPLL-RX loss of lock count bit 4
EPLL-RX				5	EPLLRXlossOfLockCount[5]	EPLL-RX loss of lock count bit 5
EPLL-RX				6	EPLLRXlossOfLockCount[6]	EPLL-RX loss of lock count bit 6
EPLL-RX				7	EPLLRXlossOfLockCount[7]	EPLL-RX loss of lock count bit 7
RX logic		435		0	FECcorrectionCount[0]	Count of FEC corrections in RX, bit 0
RX logic				1	FECcorrectionCount[1]	Count of FEC corrections in RX, bit 1
RX logic				2	FECcorrectionCount[2]	Count of FEC corrections in RX, bit 2
RX logic				3	FECcorrectionCount[3]	Count of FEC corrections in RX, bit 3
RX logic				4	FECcorrectionCount[4]	Count of FEC corrections in RX, bit 4
RX logic				5	FECcorrectionCount[5]	Count of FEC corrections in RX, bit 5
RX logic				6	FECcorrectionCount[6]	Count of FEC corrections in RX, bit 6
RX logic				7	FECcorrectionCount[7]	Count of FEC corrections in RX, bit 7

18. GBTX SIGNALS AND PINS

Important notice: Users should familiarise themselves with the functions of all the pins. Many times pins that relate with features that a particular user is not interested in will still have an overall impact on the ASIC behaviour. As such, the user must set "logic values" on all configuration pins. All input pins (with the exception of XTALN and XTALP that **must** be left floating and pins which already contain internal pull up/down resistors) **must** be driven by a valid logic level: 0 < "logic zero" < 0.2 V or 1.3 < "logic one" < 1.5 V. Unused single-ended input pins should be tie high or low through a resistor (the same resistor can serve many pins). Differential inputs must be set to "0" or "1" by tying the non-inverting input low/high and the inverting input high/low respectively. The method to tie low/high each input of a differential signal is the same as for the single-ended inputs. Output pins can be left floating.

Net Name	Pin Number	X Coord	Y Coord	Pin Use	Function
CLOCKDES0N	H1	-7600	2000	OUT	Output clocks with programmable phase and frequency: Signaling: SLVS Programmable frequencies: 40/80/160/320 MHz Programmable phase: 50 ps resolution
CLOCKDES0P	G1	-7600	2800	OUT	
CLOCKDES1N	J4	-5200	1200	OUT	
CLOCKDES1P	H4	-5200	2000	OUT	
CLOCKDES2N	K5	-4400	400	OUT	
CLOCKDES2P	K4	-5200	400	OUT	
CLOCKDES3N	L5	-4400	-400	OUT	
CLOCKDES3P	L4	-5200	-400	OUT	
CLOCKDES4N	M5	-4400	-1200	OUT	
CLOCKDES4P	M4	-5200	-1200	OUT	
CLOCKDES5N	N2	-6800	-2000	OUT	
CLOCKDES5P	N1	-7600	-2000	OUT	
CLOCKDES6N	M7	-2800	-1200	OUT	
CLOCKDES6P	M6	-3600	-1200	OUT	
CLOCKDES7N	P2	-6800	-2800	OUT	
CLOCKDES7P	P1	-7600	-2800	OUT	
CONFIGSELECT	K7	-2800	400	IN	Selects I2C or SC-IC channel for ASIC configuration
DCLKN0	V6	-3600	-6000	OUT	ePorts output clocks: Signaling: SLVS Programmable frequencies 40/80/160/320 MHz
DCLKP0	W6	-3600	-6800	OUT	
DCLKN1	V5	-4400	-6000	OUT	
DCLKP1	V4	-5200	-6000	OUT	
DCLKN2	T8	-2000	-4400	OUT	
DCLKP2	T7	-2800	-4400	OUT	
DCLKN3	Y6	-3600	-7600	OUT	
DCLKP3	Y5	-4400	-7600	OUT	
DCLKN4	U10	-400	-5200	OUT	
DCLKP4	U9	-1200	-5200	OUT	
DCLKN5	T9	-1200	-4400	OUT	
DCLKP5	T10	-400	-4400	OUT	
DCLKN6	Y11	400	-7600	OUT	
DCLKP6	Y12	1200	-7600	OUT	
DCLKN7	W12	1200	-6800	OUT	
DCLKP7	W11	400	-6800	OUT	
DCLKN8	C15	3600	6000	OUT	
DCLKP8	C16	4400	6000	OUT	
DCLKN9	C12	1200	6000	OUT	
DCLKP9	D12	1200	5200	OUT	

DCLKN10	B12	1200	6800	OUT
DCLKP10	B11	400	6800	OUT
DCLKN11	B8	-2000	6800	OUT
DCLKP11	C8	-2000	6000	OUT
DCLKN12	C9	-1200	6000	OUT
DCLKP12	C10	-400	6000	OUT
DCLKN13	E5	-4400	4400	OUT
DCLKP13	E6	-3600	4400	OUT
DCLKN14	B6	-3600	6800	OUT
DCLKP14	B5	-4400	6800	OUT
DCLKN15	C6	-3600	6000	OUT
DCLKP15	C5	-4400	6000	OUT
DCLKN16	L17	5200	-400	OUT
DCLKP16	K17	5200	400	OUT
DCLKN17	E20	7600	4400	OUT
DCLKP17	F20	7600	3600	OUT
DCLKN18	D20	7600	5200	OUT
DCLKP18	C20	7600	6000	OUT
DCLKN19	D17	5200	5200	OUT
DCLKP19	C17	5200	6000	OUT
DCLKN20	B17	5200	6800	OUT
DCLKP20	B18	6000	6800	OUT
DCLKN21	B14	2800	6800	OUT
DCLKP21	B13	2000	6800	OUT
DCLKN22	Y16	4400	-7600	OUT
DCLKP22	Y15	3600	-7600	OUT
DCLKN23	V15	3600	-6000	OUT
DCLKP23	V16	4400	-6000	OUT
DCLKN24	Y18	6000	-7600	OUT
DCLKP24	Y17	5200	-7600	OUT
DCLKN25	T20	7600	-4400	OUT
DCLKP25	R20	7600	-3600	OUT
DCLKN26	P16	4400	-2800	OUT
DCLKP26	P15	3600	-2800	OUT
DCLKN27	K18	6000	400	OUT
DCLKP27	L18	6000	-400	OUT
DCLKN28	L20	7600	-400	OUT
DCLKP28	M20	7600	-1200	OUT
DCLKN29	J17	5200	1200	OUT
DCLKP29	H17	5200	2000	OUT
DCLKN30	R7	-2800	-3600	OUT
DCLKP30	R8	-2000	-3600	OUT
DCLKN31	V10	-400	-6000	OUT
DCLKP31	V9	-1200	-6000	OUT
DCLKN32	V11	400	-6000	OUT
DCLKP32	V12	1200	-6000	OUT
DCLKN33	W20	7600	-6800	OUT
DCLKP33	V20	7600	-6000	OUT
DCLKN34	L19	6800	-400	OUT
DCLKP34	M19	6800	-1200	OUT
DCLKN35	E19	6800	4400	OUT
DCLKP35	E18	6000	4400	OUT
DCLKN36	E14	2800	4400	OUT

DCLKP36	E13	2000	4400	OUT	
DCLKN37	E7	-2800	4400	OUT	
DCLKP37	E8	-2000	4400	OUT	
DCLKN38	G8	-2000	2800	OUT	
DCLKP38	G9	-1200	2800	OUT	
DCLKN39	G4	-5200	2800	OUT	
DCLKP39	G3	-6000	2800	OUT	
DINN0	U5	-4400	-5200	IN	ePorts data inputs: Signaling: SLVS or LVDS Programmable data rate: 80/160/320 Mb/s Programable input impedance: Hi-Z or 100 Ohm differential
DINP0	U6	-3600	-5200	IN	
DINN1	T6	-3600	-4400	IN	
DINP1	T5	-4400	-4400	IN	
DINN2	U8	-2000	-5200	IN	
DINP2	U7	-2800	-5200	IN	
DINN3	R10	-400	-3600	IN	
DINP3	R9	-1200	-3600	IN	
DINN4	T12	1200	-4400	IN	
DINP4	T11	400	-4400	IN	
DINN5	U11	400	-5200	IN	
DINP5	U12	1200	-5200	IN	
DINN6	U14	2800	-5200	IN	
DINP6	U13	2000	-5200	IN	
DINN7	U17	5200	-5200	IN	
DINP7	V17	5200	-6000	IN	
DINN8	D10	-400	5200	IN	
DINP8	D9	-1200	5200	IN	
DINN9	E10	-400	4400	IN	
DINP9	E9	-1200	4400	IN	
DINN10	D7	-2800	5200	IN	
DINP10	D8	-2000	5200	IN	
DINN11	E4	-5200	4400	IN	
DINP11	F4	-5200	3600	IN	
DINN12	F8	-2000	3600	IN	
DINP12	F7	-2800	3600	IN	
DINN13	B3	-6000	6800	IN	
DINP13	B4	-5200	6800	IN	
DINN14	D5	-4400	5200	IN	
DINP14	D6	-3600	5200	IN	
DINN15	B2	-6800	6800	IN	
DINP15	B1	-7600	6800	IN	
DINN16	G18	6000	2800	IN	
DINP16	F18	6000	3600	IN	
DINN17	G16	4400	2800	IN	
DINP17	F16	4400	3600	IN	
DINN18	D16	4400	5200	IN	
DINP18	D15	3600	5200	IN	
DINN19	E15	3600	4400	IN	
DINP19	F15	3600	3600	IN	
DINN20	E11	400	4400	IN	
DINP20	E12	1200	4400	IN	
DINN21	C11	400	6000	IN	
DINP21	D11	400	5200	IN	
DINN22	T15	3600	-4400	IN	
DINP22	T16	4400	-4400	IN	

DINN23	T17	5200	-4400	IN	
DINP23	T18	6000	-4400	IN	
DINN24	R17	5200	-3600	IN	
DINP24	P17	5200	-2800	IN	
DINN25	N17	5200	-2000	IN	
DINP25	M17	5200	-1200	IN	
DINN26	N18	6000	-2000	IN	
DINP26	M18	6000	-1200	IN	
DINN27	K16	4400	400	IN	
DINP27	L16	4400	-400	IN	
DINN28	K15	3600	400	IN	
DINP28	L15	3600	-400	IN	
DINN29	H16	4400	2000	IN	
DINP29	J16	4400	1200	IN	
DINN30	R4	-5200	-3600	IN	
DINP30	R5	-4400	-3600	IN	
DINN31	R12	1200	-3600	IN	
DINP31	R11	400	-3600	IN	
DINN32	U16	4400	-5200	IN	
DINP32	U15	3600	-5200	IN	
DINN33	R19	6800	-3600	IN	
DINP33	T19	6800	-4400	IN	
DINN34	R16	4400	-3600	IN	
DINP34	R15	3600	-3600	IN	
DINN35	E17	5200	4400	IN	
DINP35	E16	4400	4400	IN	
DINN36	D14	2800	5200	IN	
DINP36	D13	2000	5200	IN	
DINN37	M15	3600	-1200	IN	
DINP37	N15	3600	-2000	IN	
DINN38	G6	-3600	2800	IN	
DINP38	G7	-2800	2800	IN	
DINN39	D4	-5200	5200	IN	
DINP39	C4	-5200	6000	IN	
DION0	Y1	-7600	-7600	BI	ePort bidirectional data ports: Depending on the bus mode behave as outputs or inputs Programmable data rate: 80/160/320 Mb/s Signaling: SLVS when configured as outputs SLVS or LVDS when configured as inputs Impedance when programmed as inputs: Programable: Hi-Z / 100 Ohm differential
DIOP0	Y2	-6800	-7600	BI	
DION1	T4	-5200	-4400	BI	
DIOP1	U4	-5200	-5200	BI	
DION2	Y4	-5200	-7600	BI	
DIOP2	Y3	-6000	-7600	BI	
DION3	V3	-6000	-6000	BI	
DIOP3	W3	-6000	-6800	BI	
DION4	Y7	-2800	-7600	BI	
DIOP4	Y8	-2000	-7600	BI	
DION5	W7	-2800	-6800	BI	
DIOP5	V7	-2800	-6000	BI	
DION6	Y10	-400	-7600	BI	
DIOP6	Y9	-1200	-7600	BI	
DION7	W19	6800	-6800	BI	
DIOP7	V19	6800	-6000	BI	
DION8	A18	6000	7600	BI	
DIOP8	A17	5200	7600	BI	
DION9	A16	4400	7600	BI	

DIOP9	A15	3600	7600	BI	
DION10	A13	2000	7600	BI	
DIOP10	A14	2800	7600	BI	
DION11	A11	400	7600	BI	
DIOP11	A12	1200	7600	BI	
DION12	A10	-400	7600	BI	
DIOP12	A9	-1200	7600	BI	
DION13	A8	-2000	7600	BI	
DIOP13	A7	-2800	7600	BI	
DION14	A5	-4400	7600	BI	
DIOP14	A6	-3600	7600	BI	
DION15	A4	-5200	7600	BI	
DIOP15	A3	-6000	7600	BI	
DOUTN16	H18	6000	2000	OUT	ePorts data outputs: Programmable data rates: 80/160/320 Mb/s Signaling: SLVS
DOUTP16	J18	6000	1200	OUT	
DOUTN17	H20	7600	2000	OUT	
DOUTP17	G20	7600	2800	OUT	
DOUTN18	H19	6800	2000	OUT	
DOUTP18	G19	6800	2800	OUT	
DOUTN19	C19	6800	6000	OUT	
DOUTP19	D19	6800	5200	OUT	
DOUTN20	B20	7600	6800	OUT	
DOUTP20	A20	7600	7600	OUT	
DOUTN21	C18	6000	6000	OUT	
DOUTP21	D18	6000	5200	OUT	
DOUTN22	Y13	2000	-7600	OUT	
DOUTP22	Y14	2800	-7600	OUT	
DOUTN23	V14	2800	-6000	OUT	
DOUTP23	V13	2000	-6000	OUT	
DOUTN24	W16	4400	-6800	OUT	
DOUTP24	W15	3600	-6800	OUT	
DOUTN25	Y20	7600	-7600	OUT	
DOUTP25	Y19	6800	-7600	OUT	
DOUTN26	U20	7600	-5200	OUT	
DOUTP26	U19	6800	-5200	OUT	
DOUTN27	N19	6800	-2000	OUT	
DOUTP27	N20	7600	-2000	OUT	
DOUTN28	N16	4400	-2000	OUT	
DOUTP28	M16	4400	-1200	OUT	
DOUTN29	K19	6800	400	OUT	
DOUTP29	J19	6800	1200	OUT	
DOUTN30	W2	-6800	-6800	OUT	
DOUTP30	W1	-7600	-6800	OUT	
DOUTN31	V8	-2000	-6000	OUT	
DOUTP31	W8	-2000	-6800	OUT	
DOUTN32	T13	2000	-4400	OUT	
DOUTP32	T14	2800	-4400	OUT	
DOUTN33	U18	6000	-5200	OUT	
DOUTP33	V18	6000	-6000	OUT	
DOUTN34	P20	7600	-2800	OUT	
DOUTP34	P19	6800	-2800	OUT	
DOUTN35	F13	2000	3600	OUT	
DOUTP35	F14	2800	3600	OUT	

DOUTN36	B15	3600	6800	OUT	
DOUTP36	B16	4400	6800	OUT	
DOUTN37	C7	-2800	6000	OUT	
DOUTP37	B7	-2800	6800	OUT	
DOUTN38	F5	-4400	3600	OUT	
DOUTP38	F6	-3600	3600	OUT	
DOUTN39	A2	-6800	7600	OUT	
DOUTP39	A1	-7600	7600	OUT	
EFUSEPOWER	P14	2800	-2800	POWER	E-Fuse power: 3.3V during programing, 1.5V otherwise
EFUSEPROGRAMPULSE	P11	400	-2800	IN	E-Fuse write pulse
GND	A19	6800	7600	GROUND	Digital and digital I/O ground
GND	B9	-1200	6800	GROUND	
GND	C13	2000	6000	GROUND	
GND	F10	-400	3600	GROUND	
GND	F11	400	3600	GROUND	
GND	F12	1200	3600	GROUND	
GND	F17	5200	3600	GROUND	
GND	G10	-400	2800	GROUND	
GND	G11	400	2800	GROUND	
GND	G12	1200	2800	GROUND	
GND	G13	2000	2800	GROUND	
GND	H10	-400	2000	GROUND	
GND	H11	400	2000	GROUND	
GND	H12	1200	2000	GROUND	
GND	J10	-400	1200	GROUND	
GND	J11	400	1200	GROUND	
GND	J20	7600	1200	GROUND	
GND	K11	400	400	GROUND	
GND	K12	1200	400	GROUND	
GND	L11	400	-400	GROUND	
GND	L12	1200	-400	GROUND	
GND	M11	400	-1200	GROUND	
GND	M12	1200	-1200	GROUND	
GND	P18	6000	-2800	GROUND	
GND	R14	2800	-3600	GROUND	
GND	W4	-5200	-6800	GROUND	
GND	W9	-1200	-6800	GROUND	
GND	W13	2000	-6800	GROUND	
GND	W17	5200	-6800	GROUND	
GNDCM	J1	-7600	1200	GROUND	Clock manager ground
GNDCM	K2	-6800	400	GROUND	
GNDCM	L2	-6800	-400	GROUND	
GNDCM	M1	-7600	-1200	GROUND	
GNDPS	H2	-6800	2000	GROUND	Phase-shifter ground
GNDPS	J5	-4400	1200	GROUND	
GNDPS	L6	-3600	-400	GROUND	
GNDPS	L9	-1200	-400	GROUND	
GNDPS	N3	-6000	-2000	GROUND	
GNDRX	R1	-7600	-3600	GROUND	Receiver ground
GNDRX	T2	-6800	-4400	GROUND	
GNDRX	U2	-6800	-5200	GROUND	

GNDRX	V1	-7600	-6000	GROUND	
GNDTX	C1	-7600	6000	GROUND	Transmitter ground
GNDTX	C2	-6800	6000	GROUND	
GNDTX	D2	-6800	5200	GROUND	
GNDTX	E2	-6800	4400	GROUND	
GNDTX	F1	-7600	3600	GROUND	
I2CADDRESS<0>	L8	-2000	-400	IN	I2C address: Bit<0> has internal pull-up to VDDIO, Bits<3:1> have internal pull-downs to GND The resistors are 4.7 kohm.
I2CADDRESS<1>	N6	-3600	-2000	IN	
I2CADDRESS<2>	N7	-2800	-2000	IN	
I2CADDRESS<3>	P5	-4400	-2800	IN	
LDRESET	J7	-2800	1200	OUT	Laser driver reset
LDSCL	K8	-2000	400	OUT	Laser driver I2C clock
LDSDA	H8	-2000	2000	BI	Laser Driver I2C data
MODE<0>	N5	-4400	-2000	IN	Transceiver mode select signal
MODE<1>	L7	-2800	-400	IN	
MODE<2>	L10	-400	-400	IN	
MODE<3>	P10	-400	-2800	IN	
REFCLKN	K1	-7600	400	IN	Reference clock differential
REFCLKP	L1	-7600	-400	IN	Input impedance: 100 Ohm differential
REFCLKSELECT	J9	-1200	1200	IN	Selects the internal xPLL or REFCLKN/REFCLKP
RESETB	M10	-400	-1200	IN	Chip reset
RXDATAVALID	N9	-1200	-2000	OUT	Receiver data valid flag
RXINN	U1	-7600	-5200	IN	Receiver differential serial input. Data rate 4.8Gb/s
RXINP	T1	-7600	-4400	IN	Input impedance: 100 Ohm differential
RXLOCKMODE<0>	J8	-2000	1200	IN	Rx lock mode select signal
RXLOCKMODE<1>	M9	-1200	-1200	IN	
RXRDY	N10	-400	-2000	OUT	Receiver ready flag
SCCLKN	R6	-3600	-3600	OUT	Slow control ePort clock: 80 MHz
SCCLKP	P6	-3600	-2800	OUT	
SCINN	P13	2000	-2800	IN	Slow control ePort data input. Data rate 80 Mb/s Programable input impedance: Hi-Z / 100 Ohm differential
SCINP	P12	1200	-2800	IN	
SCL	G5	-4400	2800	IN	I2C slave clock
SCOUTN	P3	-6000	-2800	OUT	Slow control ePort data output. Data rate 80 Mb/s
SCOUTP	P4	-5200	-2800	OUT	
SDA	H5	-4400	2000	BI	I2C slave data line
autoReset (SLVSTESTINJECT)	H15	3600	2000	IN	Enable/disable autoReset feature (GBTXv2 only) (also SLVS Test input)
STATEOVERRIDE	H9	-1200	2000	IN	Disables the automatic power-up state machine
TCK	N8	-2000	-2000	IN	JTAG: TCK and TDI have internal pull-ups to VDDIO. The resistors are 4.7 kohm.
TDI	P7	-2800	-2800	IN	
TDO	P9	-1200	-2800	OUT	
TESTCLOCKIN1	H7	-2800	2000	IN	test clock inputs
TESTCLOCKIN2	J15	3600	1200	IN	
TESTCLOCKOUT	H6	-3600	2000	OUT	test clock output
TESTOUTPUT	K10	-400	400	OUT	test output
TMS	P8	-2000	-2800	IN	JTAG: TMS and TRESETB have internal pull-ups to VDDIO. The resistors are 4.7 kohm. **For normal operation (data transmission) it is mandatory to set the pins TMS and TRESETB to 0V (GND)**
TRESETB	M8	-2000	-1200	IN	
TXDATAVALID	N11	400	-2000	IN	Transmitter data valid qualifier
TXOUTN	D1	-7600	5200	OUT	Transmitter differential serial output. Data rate 4.8Gb/s

TXOUTP	E1	-7600	4400	OUT	External termination impedance : 100 Ohm differential
TXRDY	L13	2000	-400	OUT	Transmitter ready flag
UNUSED	G17	5200	2800		n.a.
VDD	G14	2800	2800	POWER	Digital power: 1.5 V
VDD	G15	3600	2800	POWER	
VDD	H13	2000	2000	POWER	
VDD	H14	2800	2000	POWER	
VDD	J12	1200	1200	POWER	
VDD	J13	2000	1200	POWER	
VDD	J14	2800	1200	POWER	
VDD	K13	2000	400	POWER	
VDD	K14	2800	400	POWER	
VDD	L14	2800	-400	POWER	
VDDCM	J3	-6000	1200	POWER	Clock-Manager power: 1.5 V
VDDCM	K3	-6000	400	POWER	
VDDCM	L3	-6000	-400	POWER	
VDDCM	M3	-6000	-1200	POWER	
VDDIO	B10	-400	6800	POWER	Digital I/O power: 1.5 V
VDDIO	B19	6800	6800	POWER	
VDDIO	C14	2800	6000	POWER	
VDDIO	F9	-1200	3600	POWER	
VDDIO	F19	6800	3600	POWER	
VDDIO	K20	7600	400	POWER	
VDDIO	M13	2000	-1200	POWER	
VDDIO	M14	2800	-1200	POWER	
VDDIO	N12	1200	-2000	POWER	
VDDIO	N13	2000	-2000	POWER	
VDDIO	N14	2800	-2000	POWER	
VDDIO	R13	2000	-3600	POWER	
VDDIO	R18	6000	-3600	POWER	
VDDIO	W5	-4400	-6800	POWER	
VDDIO	W10	-400	-6800	POWER	
VDDIO	W14	2800	-6800	POWER	
VDDIO	W18	6000	-6800	POWER	
VDDPS	G2	-6800	2800	POWER	Phase-Shifter power: 1.5 V
VDDPS	H3	-6000	2000	POWER	
VDDPS	J6	-3600	1200	POWER	
VDDPS	K6	-3600	400	POWER	
VDDPS	K9	-1200	400	POWER	
VDDPS	N4	-5200	-2000	POWER	
VDDRX	R2	-6800	-3600	POWER	Receiver power: 1.5 V
VDDRX	R3	-6000	-3600	POWER	
VDDRX	T3	-6000	-4400	POWER	
VDDRX	U3	-6000	-5200	POWER	
VDDRX	V2	-6800	-6000	POWER	
VDDTX	C3	-6000	6000	POWER	Transmitter power: 1.5 V
VDDTX	D3	-6000	5200	POWER	
VDDTX	E3	-6000	4400	POWER	
VDDTX	F2	-6800	3600	POWER	
VDDTX	F3	-6000	3600	POWER	
XTALN	M2	-6800	-1200	IN	These pins **must** be left floating. Avoid adding any parasitic capacitance to these pins.

XTALP	J2	-6800	1200	IN	

19. PACKAGE

Details of the GBTX ball-grid-array package can be found at the following link:
https://espace.cern.ch/GBT-Project/GBTX/Manuals/Forms/AllItems.aspx

20. SC – CHANNEL PROTOCOL ENCODING/DECODING

This chapter describes the operation of the FPGA firmware used to encode and decode the SC channel data.

The encoding/decoding operations between the byte-wide frames and the 2 bits inserted in the GBT frame are done by physical (PHY) layer blocks borrowed from the GBT-SCA design. These are known as PHY-TX and PHY-RX. Their operation is illustrated in Figure 44. The clock of these blocks is the global 40 MHz clock.

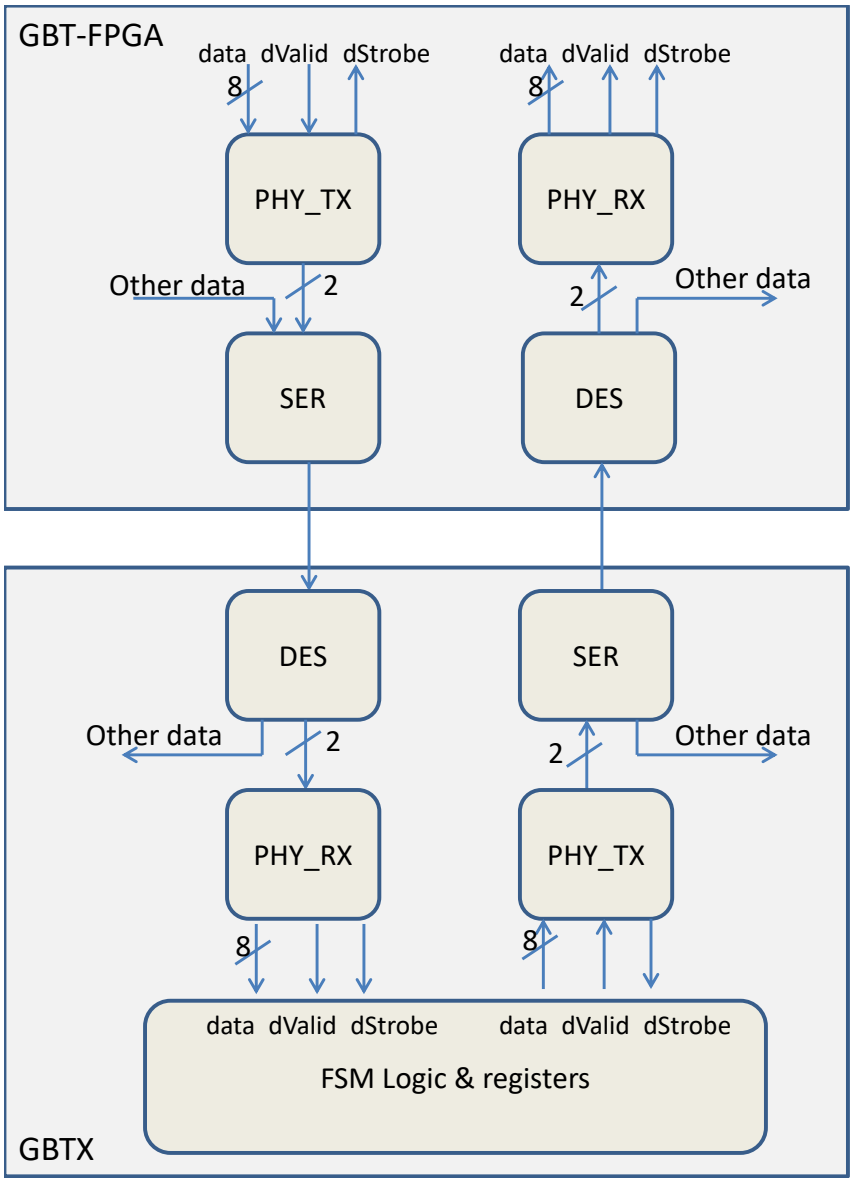


Figure 44: Encoding/decoding blocks for the SC-IC bits.

The same blocks can be used for the opposite coding procedure in the FPGA at the other end of the link.

PHY-TX

For the PHY-TX, the user should assert the **dValid** signal when ready to start transmitting. The **dStrobe** signal is generated by the PHY-TX to tell the user when to send a new 8-bit word of data. On average, **dStrobe** is asserted every

4 clock cycles (8/2), but this depends on the **data** content and the corresponding bit-stuffing by the PHY-TX. All operations are synchronous to the rising edge of the clock.

A timing diagram is shown in Figure 45. The corresponding frame is shown in Table 155 and the operation is the following:

1. The PHY-TX automatically creates the frame-delimiter 01111110 when **dValid** is asserted by the user.
2. The user should then wait until the PHY-TX asserts **dStrobe**. At the clock edge when **dStrobe** is high, the user should enter the first 8-bit word of **data** (00000100).
3. At the next **dStrobe** and clock edge, the user should enter the next 8-bit word of **data**, and repeat this for each subsequent word.
4. After the last data word is entered, the PHY-TX will assert the final **dStrobe** which should be used by the user to de-assert the **dValid**.
5. The PHY-TX automatically creates the frame-delimiter 01111110 to mark the end of the frame.

Note that the number of **dStrobes** (9 in the example) generated by the PHY-TX will be always one more than the number of words in the frame (8 in the example).

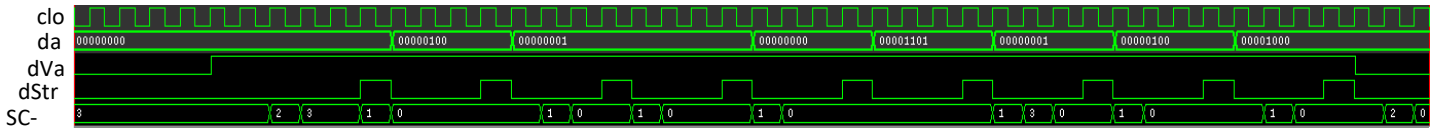


Figure 45: Timing diagram for the PHY-TX signals

Byte	**Example in diagram**
GBTX i2c address (7 bits) + R(1)/W(0) (1 bit)	00000100 (address = 0000010, write operation)
Command (8 bits)	00000001
Number of data words n[7:0]	00000001
Number of data words n[15:8]	00000000 (1 byte of data)
Memory address [7:0]	00001101
Memory address [15:8]	00000001 (address = 269 dec)
data (8 bits)	00000100
Parity word (8 bits)	00001000

Table 155: Example data frame for IC configuration.

PHY-RX

For the PHY-RX, the data is validated by the **dStrobe** signal. A timing diagram is shown in Figure 46 for the same example as above. All operations are synchronous to the rising edge of the clock and the operation is the following:

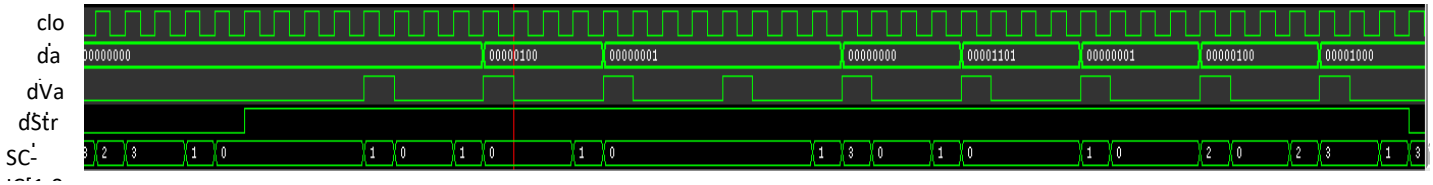


Figure 46 Timing diagram for the PHY-RX signals

1. When the PHY-RX finds the frame-delimiter, it asserts the **dValid**.
2. Some cycles later, it asserts **dStrobe** for the first time. However, the data here should be ignored.
3. The second assertion of **dStrobe** indicates that the first data word (00000100) is available on the bus. The data should be registered by the user on the clock edge when **dStrobe** is high.
4. Each subsequent assertion of **dStrobe** can be used to register the corresponding word.
5. When the PHY-RX finds the frame-delimiter at the end of frame, it de-asserts **dValid**.

Note that the number of **dStrobes** (9 in the example) generated by the PHY-RX will be always one more than the number of words in the frame (8 in the example).

21. POWER CONSUMPTION AND RELATED TID EFFECTS

21.1 Estimating the GBTX power consumption:

The GBTX has six separated power domains. All power supplies voltages are 1.5 V with the exception of the e fuse power (EfusePower) that is 3.3 V. The power taken by each domain depends on the functionality being enabled and the ASIC power dissipation can be estimated as follows:

- I/O (gndIO/vddIO):
  - P = [41] × 8.2 mW (Data SLVS-Tx) + [41] × 8.2 mW (CLK SLVS-Tx) + [41] × 0.65 mW (SLVS-Rx) = 698 mW, (I = 466 mA max)
- TX (gndTx/vddTx):
  - P = [1] x 456 mW, I = 304 mA
- RX (gndRx/vddRx):
  - P = [1] x 330 mW, I = 220 mA
- CORE (GND/VDD):
  - P = 305 mW (Standard cells) + [8] × 3.5 mW (Phase-Aligners) = 333 mW, (I = 222 mA max)
- PS (gndPS/vddPS):
  - P = [1] x 42 mW (PLL) + [8] × 16 mW (channel) + [8] × 8.2 mW (SLVS-Tx) = 236 mW, (I = 157 mA max)
- CM (gndCm/vddCm):
  - P = [2] × 41.5 mW (E-PLL) + [1] x 100 mW (XPLL) = 183 mW, (I = 122 mA max)
- E-Fuses (EfusePower):
  - P = 465 mW , I = 141 mA (Only during e-fuse programming)
- Total:
  - P = 2.2 W (This is worst case power consumption: all functions ON, worst case simulations)

The formulas above can be used to estimate the power consumption of the GBTX chip by replacing the number in square brackets by the appropriate number depending if the function is enabled or not and, if enabled (for the circuits that have multiplicity), how many circuits are active. One should note that these numbers are obtained by simulation and represent thus estimated values. Measured values of the power consumption of a sample GBTX operating as a transceiver with eLinks set to 80 Mb/s are given in Table 156.

Table 156 GBTX sample power consumption measurements

Power domain	Measured power dissipation (mW)
VDD	187
VDDCM	98
VDDIO	187
VDDPS	39
VDDRX	367
VDDTX	301

21.1.1 Example

One example of the use of the formulas above is the case presented in Table 157. In there the power consumption is estimated for a GBTX set as a transceiver with 20 input eLinks and 2 output eLinks operating at 160 Mb/s, one clock driver at 160 MHz and 4 phase adjustable clocks: 2 at 40 MHz and 2 at 160MHz.

A more complete set of cases can be found in the following file: https://espace.cern.ch/GBT-Project/GBTX/Manuals/gbtxPowerConsumption.pdf

Table 157 Example of power consumption calculation

Mode:	TRANSCEIVER
E-Links:	
Data rate [Mb/s]:	160
# Data outputs:	2
# Clock outputs:	1
# Data inputs:	20
EC-Channel:	
State:	Disabled
Phase-Shifter:	
State:	Enabled
# Channels	4
Circuit:	Power [mW]
Transmitter:	
	456
Receiver:	
	330
Clock Manager	
E-PLL:	83
XPLL:	100
Total:	183
Phase-Shifter:	
PLL:	42
Channel:	64
SLVS-TX:	33
Total:	139
I/O:	
Data SLVS-Tx:	16
CLK SLVS-Tx:	8

	Data SLVS-Rx:	13
	Total:	38
CORE:		
	Standard Cells:	305
	Phase-Aligners:	14
	Total:	319
Total Power [mW]:		1464

21.2 TID and power consumption.

It is known, for the 130 nm technology used to build the GBTX ASICs, that narrow width transistors will exhibit leakage currents when subjected to ionizing radiation to a total dose in the vicinity of 1 Mrad (see reference [20] for details). In the GBTX most of the circuits use non minimum size transistors and thus for these circuits the leakage currents are either non-existing or minimal. However, most of the digital logic is implemented with standard cells logic where minimum size transistors are extensively used to minimize the area and power consumption. Table 158 gives the values of currents and power dissipation for a GBTX sample. In this table the pre- and post-rad values are given for the logic core circuit. The power consumption of the digital logic (standard cells) represents 16% of the total power consumption but after radiation it can reach up to 69%. This represents an increase of 333% of the logic core power! However overall it corresponds to an increase of 53% of the GBTX power consumption.

Table 158 GBTX irradiation tests

	Current (mA)	Power (mW)	%
phaseShifter	26	39	3%
clockManager	65	98	8%
I/O	125	187	16%
logic core (pre-rad)	125	187	16%
logic core (post-rad)	541	812	69%
TX	201	301	26%
RX	245	367	31%
Total (pre- rad)	786	1180	
Total (post-rad)	1203	1804	

Irradiation tests on the GBTX show that (see https://espace.cern.ch/GBT-Project/GBTX/Manuals/gbtxPowerConsumption.pdf):

- Basically all power domains except the logic core power domain are unaffected by TID.
- The logic core power domain is highly sensitive to TID, with the current increasing by ~333 %

- The logic core current has a peak at 600 krad;
- As the TID passes the 600 krad the current starts to decrease slowly tending to the pre-rad value;
- If irradiation is stopped the current quickly anneals towards the pre-rad value;
- After annealing if irradiated again the current peaks again but this time to a lower value than that of the preceding irradiation cycle.
- Although the logic core current can increase by 333%, overall this corresponds to an increase of 53% of the GBTX power dissipation.
- During irradiation and annealing the device functionality was continuously being monitored and no degradation was observed.
- Both irradiation and annealing took place at room temperature.

In summary: the system designer should design for the post-rad current values and not the pre-rad ones. That is, when estimating power consumption the logic-core should be assumed to require about 550 mA. This value will be required only when TID is around 600 krad, being lower for lower and higher values of TID. No performance degradation has been observed with TID up to 100 Mrad.

22. GBT PROJECT RELATED DOCUMENTS

The GBT project web site (http://cern.ch/proj-gbt) makes the following documents available:

GBTX Specifications (this document)

ePort Specifications

GBTIA Specifications

GBLD Specifications

GBT-SCA Specifications

GBT-FPGA project

The Versatile Link project site (https://espace.cern.ch/project-versatile-Link/default.aspx) makes the following documents available:

Versatile Link Project description (UPDATE LISTOF DOCUMENTS FROM VERSATILE LINK)

23. REFERENCES

[1] GBTIA specifications http://cern.ch/proj-gbt

[2] GBLD specifications http://cern.ch/proj-gbt

[3] GBT-SCA specification http://cern.ch/proj-gbt

[4] http://www.latticesemi.com/

[5] http://www.xilinx.com/

[6] http://www.altera.com/index.jsp

[7] http://www.actel.com/

[8] R. D. Schrimpf and D. M. Fleetwood, Editors, "Radiation effects and soft errors in integrated circuits and electronic devices," World scientific publishing Co. 2004

[9] H. Nussbaumer, "Computer Communication Systems: Data Circuits Error Detection Data Links," Vol. 1, John Wiley & Sons, 1990

[10] C.J.Marshall, P.W.Marshall, M.A.Carts, R.Reed, S.Baier, K.LaBel "Characterization of transient error cross sections in high speed commercial fiber optic data links", IEEE Radiation Effects Data Workshop, 2001, pp142-145.

[11] P.W.Marshall, P.T.Wiley, R.N.Prusia, G.D. Rash, H.Kim, K.A.LaBel "Proton induced bit error studies in a 10Gb/s Fiber Optic Link" IEEE Transactions on Nuclear Science, vol.51, no.5, October 2004, pp.2736-2739.

[12] S. Lin, D. J. Costello, "Error Control Coding", 2nd ed., Prentice Hall, 2004, ch.2-7.

[13] G. Papotti, "An Error-Correcting Line Coding ASIC for a HEP Rad-Hard Multi-GigaBit Optical Link", Proc. 2nd Conference on Ph.D. Research in Microelectronics and Electronics (PRIME 2006), Otranto (Lecce), Italy, 12-15 June 2006, pp.225-8.

[14] Giulia Papotti, 'Architectural studies of a radiation-hard transceiver ASIC im 0.13 mm CMOS for digital optical links in high energy physics applications," PhD thesis, University of Parma, Italy, January 2007, http://papotti.web.cern.ch/papotti/tesi.pdf

[15] IEEE Std 802.3, 1998 Edition

[16] JEDEC standard, JESD8-13, scalable low-voltage signaling for 400 mV (SLVS-400) http://www.jedec.org/download/search/JESD8-13.pdf

[17]GBT Project: ePort specifications

[18] ATLANTIC reference

[19] P.Leitao et al., "Test Bench Development for the Radiation Hard GBTX ASIC", Proceedings of TWEPP 2014,

http://indico.cern.ch/event/299180/session/11/contribution/22

[20] F.Faccio and G. Cervelli, "Radiation-Induced Edge Effects in Deep

Submicron CMOS Transistors", IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 52, NO. 6, DECEMBER 2005, pp. 2413 – 2420